

MARLTC: A Multi-Agent Reinforcement Learning-Based Interference-Aware Transmission Control for LoRaWAN IoT Devices

Juan Aznar-Poveda¹, Laura Acosta-Garcia², Fabian Margreiter³, Marlon Etheredge⁴,
Abolfazl Younesi⁵, *Student Member, IEEE*, Stefan Pedratscher⁶, Joan Garcia-Haro⁷, *Member, IEEE*,
Thomas Fahringer⁸, *Member, IEEE*, and Antonio-Javier Garcia-Sanchez⁹

Abstract—Long-range wide-area network (LoRaWAN) has become a foundational technology in the Internet of Things (IoT) landscape due to its long-range (LoRa) communication and energy efficiency (EE). However, its default adaptive data rate (ADR) mechanism struggles to adapt to dynamic environments and dense network deployments, where co-channel interference and the coupling between transmission parameters limit its ability to ensure reliable and energy-efficient communication. To address these challenges, this article proposes MARLTC, an ADR mechanism based on multi-agent reinforcement learning (MARL) that jointly optimizes spreading factor (SF) and transmission power using realistic observable metrics, including recent transmission history, observed signal-to-noise ratios (SNRs), and distribution of SFs in the network. The transmission configuration problem is modeled as a cooperative Markov game and solved using the centralized training with decentralized execution (CTDE) paradigm, where pretrained policies are deployed at end devices to infer suitable transmission parameters with reduced convergence time. The results using a realistic LoRaWAN simulator show that MARLTC achieves up to 67.9% faster convergence, 62.1% higher EE, and a 5.0% better packet delivery ratio (PDR) compared to state-of-the-art approaches, highlighting its scalability and responsiveness in dense deployments. The practical feasibility of MARLTC is further validated using physical LoRaWAN hardware, proving that the resulting policies

meet the memory and timing constraints of resource-constrained IoT devices.

Index Terms—Internet of Things (IoT), long-range (LoRa) communications, LoRa wide-area network (LoRaWAN), multi-agent reinforcement learning (MARL), wireless networks.

I. INTRODUCTION

LONG-RANGE (LoRa) communications, which operate under the LoRa wide-area network (LoRaWAN) protocol, have become a cornerstone technology in Internet of Things (IoT) deployments due to their low power requirements and extended communication range [1], [2]. LoRaWAN was conceived for sparse network scenarios characterized by infrequent transmissions and small data payloads over long distances [3], [4], [5]. Emerging deployments in domains such as smart cities, industrial automation, and large-scale environmental monitoring increasingly involve dense networks with a growing number of end devices [6], [7]. Such large-scale, dynamic systems require fundamentally new approaches to programming and runtime coordination beyond traditional mechanisms, as also highlighted in the context of next-generation large-scale computing systems [8]. At the MAC layer, LoRaWAN employs a pure ALOHA protocol, which enables low power consumption by eliminating the need for carrier sensing. However, in dense deployments, this uncoordinated access often causes frequent packet collisions and increased co-channel interference, leading to reduced energy efficiency (EE), lower network reliability, and poorer overall system performance.

A common strategy to mitigate interference is to adjust the transmission parameters of the involved end devices, such as the spreading factor (SF) and transmission power. Co-channel interference consists of co-SF interference, which arises when devices using the same SF transmit on the same channel, and inter-SF interference, which results from the imperfect orthogonality among different SFs. This interference creates a strong coupling among the transmission parameters of end devices, making the optimization of EE and network reliability challenging in dense networks. LoRaWAN's adaptive data rate (ADR) mechanism dynamically adjusts the transmission parameters of end devices to optimize EE and network performance [9]. However, it relies on historical averages of link quality metrics, updating parameters only after a given number of uplink messages [10], [11], [12], [13], [14]. This limits

Received 27 December 2025; revised 3 March 2026 and 5 May 2026; accepted 19 May 2026. Date of publication 25 May 2026; date of current version 27 July 2026. This work was supported in part by Forschungsförderungsgesellschaft (FFG) Austria, BRIDGE Project “CSM,” under Grant FO999925697; in part by the Recovery and Resilience Plan of the Slovak Republic under the call “Transformation and Innovation Consortia” under Grant 09I02-03-V01-00012; in part by the Ministerio de Ciencia, Innovación y Universidades (MICIU)/Agencia Estatal de Investigación (AEI)/10.13039/501100011033 and Fondo Europeo de Desarrollo Regional (FEDER)/European Union (EU) under Grant PID2023-148214OB-C21; in part by the Ministerio de Ciencia e Innovación (MCIN)/AEI/10.13039/501100011033 under Grant PDC2025-165710-C22; in part by the MICIU/AEI/10.13039/501100011033 and European Union under Grant PCI2024-153485; and in part by the Partnership for Research and Innovation in the Mediterranean Area (PRIMA) Program (FUSION: Comprehensive and Sustainable Solution to Minimize Food Loss and Waste and Promote Food Security in the Mediterranean Region) under Grant 2431. (Corresponding author: Juan Aznar-Poveda.)

Juan Aznar-Poveda, Fabian Margreiter, Marlon Etheredge, Abolfazl Younesi, Stefan Pedratscher, and Thomas Fahringer are with the Distributed and Parallel Systems Group, University of Innsbruck, 6020 Innsbruck, Austria (e-mail: juan.aznar-poveda@uibk.ac.at; fabian.margreiter@gmail.com; marlon.etheredge@uibk.ac.at; abolfazl.younesi@uibk.ac.at; stefan.pedratscher@uibk.ac.at; thomas.fahringer@uibk.ac.at).

Laura Acosta-Garcia, Joan Garcia-Haro, and Antonio-Javier Garcia-Sanchez are with the Telematics Engineering Group, Technical University of Cartagena, 30202 Cartagena, Spain (e-mail: lauraacostag4@gmail.com; joang.haro@upct.es; antoniojavier.garcia@upct.es).

Digital Object Identifier 10.1109/JIOT.2026.3696353

ADR's responsiveness to channel changes. Moreover, although interference significantly affects communication reliability, the default ADR does not explicitly account for it [15], [16], [17], [18], [19], causing performance degradation as networks scale [20], [21].

To overcome these limitations, proactive or context-aware approaches, such as link quality prediction based on historical data, mobility, or location information, have been investigated [10], [22], [23], though they often rely on contextual information that is complex or unfeasible to obtain in realistic deployment scenarios [14], [24]. Some works have explored reinforcement learning (RL) for optimizing transmission parameters in LoRa networks, but most approaches focus on a single gateway and overlook interdevice interactions, limiting network scalability [25], [26], [27], [28], [29]. Multiagent RL (MARL) addresses these issues by facilitating coordinated learning among devices to enhance overall network performance, scalability, and convergence time [30], [31], [32], [33]. Despite the potential of RL- and MARL-based methods for optimizing LoRa transmission parameters, existing approaches often rely on simplified network models that overlook interference in dense multigateway scenarios, depend on instantaneous feedback metrics that are not directly observable in practical deployments, and assume that computationally intensive agent training can be performed on end devices. Thus, developing adaptive mechanisms that operate effectively under interference and realistic deployment constraints remains an open challenge.

In this article, we present MARLTC, a MARL-based transmission control mechanism for dense LoRaWAN networks that jointly optimizes the SF and transmission power to enhance EE and reliability. MARLTC explicitly accounts for co-channel interference and aims to guarantee a user-defined packet delivery ratio (PDR) target while minimizing energy consumption. We explore multiple policy variants: one optimized for EE, another for packet delivery, and a third enabling dynamic tradeoffs between these objectives. This allows tailoring to specific application needs and supports comparative evaluation across diverse network scenarios. The transmission configuration problem is modeled as a cooperative Markov Game. We adopt a centralized training with decentralized execution (CTDE) paradigm tailored to the LoRaWAN architecture, where policy optimization is performed centrally using information aggregated across gateways, whereas inference is executed independently on each end device using local observations and infrequent, lightweight feedback from the closest gateway. By leveraging the collaborative learning capabilities of MARL, MARLTC adapts to dynamic channel conditions and interference patterns while maintaining scalability, provided that the communication overhead from shared state information remains bounded. It also enables coordinated, distributed decision-making without requiring access to global state or synchronized control, supporting practical deployment in large-scale LoRaWAN networks with minimal protocol modifications and communication overhead. We evaluate MARLTC using FAST-LoRa [34], a novel open-source simulation framework designed for efficient and realistic LoRaWAN network evaluation and analysis. We evaluate MARLTC across diverse urban deployments,

demonstrating its effectiveness in improving EE and reliability compared to state-of-the-art ADR schemes. To ensure real-world viability, we implement a model-conversion pipeline that translates PyTorch-based policies into lightweight C code. We validate the practical feasibility of this implementation on physical LoRaWAN hardware, proving that MARLTC satisfies the strict memory and real-time execution constraints of IoT hardware.

The main contributions of this article are summarized as follows.

- C1: We propose MARLTC, a novel ADR mechanism based on MARL, which optimizes EE and reliability in dynamic and dense LoRaWAN networks.
- C2: We introduce an interference-aware state and reward design that models interdevice coupling via aggregated SF occupancy and transmission history, and enforces reliability through a PDR constraint, enabling energy-optimal, LoRaWAN-compliant decentralized adaptation.
- C3: We develop a scalable cooperative Markov game formulation for LoRaWAN ADR integrating CTDE, a shared policy across end devices, and asynchronous uplink-aligned updates, enabling interference-coupled optimization in dense deployments.
- C4: We demonstrate through extensive simulations using the FAST-LoRa framework [34] that MARLTC outperforms state-of-the-art ADR mechanisms in terms of EE and reliability across diverse urban scenarios.
- C5: We verify the hardware feasibility of MARLTC via a model-conversion pipeline. Experimental results on a physical LoRaWAN testbed demonstrate a 76% reduction in memory footprint and negligible inference latency, confirming that the policies are compatible with resource-constrained IoT hardware.

The remainder of this article is organized as follows. Section II reviews state-of-the-art ADR mechanisms for optimizing LoRaWAN performance, Section III formulates the analytical model underpinning MARLTC, and Section IV describes the proposed framework. Sections V and VI present a comprehensive evaluation of MARLTC against current state-of-the-art algorithms. Section VII assesses the practical feasibility of deploying MARLTC policies on a physical LoRaWAN testbed. Finally, Section VIII summarizes key findings, discusses limitations, and outlines future research directions.

II. RELATED WORK

Various research efforts have proposed enhancements of the transmission parameter selection mechanisms, ranging from minor algorithmic adjustments to more sophisticated solutions. More broadly, the challenge of coordinating distributed components under locality and interaction constraints has long been studied in distributed and parallel systems, for example, in locality-aware programming models such as *Javasympphony* [35], which highlights the importance of explicitly considering communication and coordination behavior. For example, ADR+ [21] improved link quality estimation by averaging multiple previously received packets rather than relying on single-packet metrics. G-ADR and EMA-ADR [13] built on this approach by applying Gaussian and exponential

moving average filters, respectively, to enhance performance in noisy environments. EXPLoRa [36] mitigated congestion by distributing SFs according to the number of end devices, and I-ADR [37] improved convergence reliability by assigning initial SFs based on the signal strength of the first received transmission. Blind ADR [38] improved reliability and EE by eliminating reliance on link quality metrics and cyclically alternating among predefined transmission settings. LR-ADR [10] applied linear regression to predict future link quality based on historical packet reception metrics, achieving better performance in dynamic environments. MADERE [22] estimated the distance to gateways using received signal strength and selected transmission parameters based on the predicted future location of end devices. KDR [23] used trajectory estimation combined with a k-nearest neighbors algorithm applied to a buffer of previous signal-to-noise ratio (SNR) samples to predict future SNR values, thereby enabling the preemptive selection of transmission parameters. HADR [39] and RM-ADR [14] dynamically adapted ADR strategies based on mobility status, determined through positional information or coordinated decision-making between the network server (NS) and end devices. The method proposed in [24] adjusted transmission configurations to meet quality of service requirements, prioritizing communication reliability and EE in dynamic LoRa scenarios.

Some approaches have explored RL for transmission parameter selection [25], [26], [27], [28], [29]. In [25], an algorithm based on DRL is presented, which found suitable transmission settings for end devices, minimizing energy consumption and the bit error rate. ADRL [26] employed geometric methods to predict the trajectory of the end device and a lightweight KNN-based approach to forecast signal strength [23], [40], [41]. Then, deep RL was used to optimize transmission parameters, minimizing energy consumption while maximizing packet delivery and ensuring duty cycle compliance. LoRa-MAB [27] adopted a distributed approach that achieves solid performance but suffers from low EE due to a prolonged exploration phase and additional communication overhead. LP-MAB [28] mitigated these issues by shortening the exploration period and employing a centralized strategy where the NS manages the transmission parameters of end devices in a LoRaWAN network. LoRa-RL [29] leveraged deep RL to enhance EE in a scenario with multiple end devices and a single UAV-based gateway, showing strong results with pretrained agents.

While existing RL-based methods have shown great potential in optimizing transmission parameters for individual LoRa devices, they often focus on single-gateway optimization without fully capturing the interactions among multiple devices sharing the same channel and resources in the network, such as interference. Building upon these foundations, MARL enables coordinated learning across multiple agents, allowing devices to adapt their transmission strategies collaboratively [30]. This leads to improvements in scalability and overall network efficiency under dynamic conditions. The work [31] presents a MARL framework for adaptive wireless network management, where each end device autonomously adjusts its transmission parameters using a local agent that optimizes a reward function balancing application-

specific performance with global network fairness. The work demonstrated improved adaptability and responsiveness compared to single-agent and ADR-based approaches. However, this approach employs simplified network models that do not fully account for dense multigateway deployments and interference among end devices. Moreover, the study does not address the practical feasibility of conducting computationally intensive agent training on resource-constrained end devices. Zhao et al. [32] propose a MARL algorithm for energy-efficient transmission in LoRaWAN-based sensor networks, optimizing link quality, energy use, and packet collisions. However, its focus on underground environments limits applicability to standard LoRaWAN deployments. In [33], the proposed MARL-based framework optimizes transmission parameters by leveraging instantaneous estimates of EE and communication reliability. While this framework demonstrates notable improvements over Semtech's ADR mechanism in terms of EE, it relies on instantaneous network performance metrics, derived from an analytical model, that are not directly observable in practical LoRaWAN deployments. Although the model is validated using an established simulation framework, it lacks realistic updates from gateways and the NS needed to capture the dynamics of operational deployments and channel conditions, which significantly limits the framework's practical applicability.

In contrast to prior MARL-based approaches, MARLTC is designed to operate under realistic LoRaWAN network and protocol constraints. Unlike [31] and [32], which adopt simplified interference abstractions, MARLTC explicitly models both co-SF and inter-SF interference, including capture effects, in dense multigateway urban scenarios. Addressing the observability assumptions identified in [33], MARLTC adheres strictly to the LoRaWAN protocol constraints by relying exclusively on locally available transmission histories and sparse, aggregated feedback provided by the NS, thereby avoiding the use of instantaneous global network metrics that are not observable in practical deployments. Finally, distinct from approaches requiring on-device training, MARLTC adopts a centralized training, decentralized execution paradigm aligned with the LoRaWAN architecture, in which computationally intensive policy optimization is performed at the NS while resource-constrained end devices execute only lightweight inference. This design enables scalable and interference-aware adaptive transmission control in dense LoRaWAN networks under realistic deployment conditions.

III. SYSTEM MODEL

In this section, we introduce the key components of the LoRaWAN architecture and the transmission reliability model (including co-channel interference) that forms the basis of our method.

A. LoRaWAN Network

We consider a LoRaWAN network comprising n end devices and m gateways. Let $\mathcal{E} = \{e_1, e_2, \dots, e_n\}$ denote the set of end devices and $\mathcal{G} = \{g_1, g_2, \dots, g_m\}$ the set of gateways. We introduce the indices $i, j \in \{1, \dots, n\}$ to refer to end devices

$e_i, e_j \in \mathcal{E}$ and $k \in \{1, \dots, m\}$ to refer to gateways $g_k \in \mathcal{G}$. The index i is typically used to represent a reference end device, while j may denote another device in the network for modeling interference, with the convention that $i \neq j$. Each end device e_i is configured with a transmission power $p_i \in \mathcal{P}$, an SF $f_i \in \mathcal{F}$, and a coding rate $r_i \in \mathcal{R}$, where $\mathcal{P}$, $\mathcal{F}$, and $\mathcal{R}$ denote the respective finite sets of admissible transmission powers, SFs, and coding rates, respectively [34]. Gateways serve as receivers and are responsible for collecting uplink transmissions from the end devices. The network adopts a star topology, as is standard in LoRaWAN deployments, whereby all end devices communicate directly with any gateway within their communication range. Gateways are interconnected through an NS, which is responsible for coordinating network operations. The NS performs device registration, manages the scheduling of uplink and downlink transmissions, executes ADR algorithms, and enforces policies aimed at ensuring scalability, reliability, and EE.

B. Transmission Reliability

LoRa transmission reliability depends on two conditions: 1) the received signal strength indicator (RSSI) exceeds the receiver's sensitivity threshold, and 2) the signal remains free from collision with concurrent transmissions [42]. Accordingly, the PDR from end device e_i to gateway g_k is modeled as the product of these two probabilities

$$PDR_{ik} = \psi_{ik} \cdot \zeta_{ik} \quad (1)$$

where ψ_{ik} is the probability that the RSSI is sufficient, and ζ_{ik} is the probability of the packet not being corrupted by an interfering transmission [26], [33].

1) *Received Signal Strength Indicator*: The RSSI at gateway g_k from end device e_i is calculated using the log-distance path loss model, which effectively captures signal attenuation in urban and rural environments. This model incorporates transmission power, the distance between end devices and gateways, and shadowing effects resulting from environmental variability. Formally, RSSI is given by

$$RSSI_{ik} = p_i - PL(d_0) - 10\gamma \log_{10} \left(\frac{d_{ik}}{d_0} \right) - N_\sigma \quad (2)$$

where p_i is the transmission power of end device e_i , $PL(d_0)$ is the mean path loss at reference distance d_0 , γ is the path loss exponent, d_{ik} is the distance between end device e_i and gateway g_k , and N_σ is a Gaussian-distributed variable representing shadowing. The SNR of a transmission from end device e_i arriving at gateway g_k can generally be calculated as [43] $SNR_{ik} = RSSI_{ik} - N_0$, where N_0 is the noise floor in dBm.

To determine the probability ψ_{ik} that the signal strength is sufficient for decoding, we rewrite $RSSI_{ik}$ as $z_{ik} - N_{ik}$, with $z_{ik} = p_i - PL(d_0) - 10\gamma \log_{10} (d_{ik}/d_0)$ and $N_{ik} \sim \mathcal{N}(0, \sigma)$. Then,

$$\begin{aligned} \psi_{ik} &= \mathbb{P}(RSSI_{ik} \geq \eta_{fi}) \\ &= \mathbb{P}(N_{ik} \leq z_{ik} - \eta_{fi}) = \frac{1}{2} + \frac{1}{2} \text{erf} \left(\frac{z_{ik} - \eta_{fi}}{\sqrt{2}\sigma} \right) \end{aligned} \quad (3)$$

where η_{fi} is the sensitivity threshold corresponding to SF f_i , and erf is the Gauss error function.

2) *Interference*: To model interference effects (ζ_{ik}), we begin by calculating the time interval during which transmissions from two end devices, e_i and e_j , may overlap. We assume that successful decoding requires at least the final five preamble symbols to be received without collision [44], [45]. The critical time interval is

$$T'_{ij} = T_j + T_i - (n_{pr} - 5) T_i^{sym} \quad (4)$$

where n_{pr} is the number of preamble symbols. Considering Poisson packet generation [33], [45], [46], the probability that the transmission of end device e_j interferes with the one of e_i is

$$h_{ij} = 1 - e^{-\lambda_{ij} T'_{ij}} \quad (5)$$

where λ denotes the average packet generation rate of the involved end devices. The Poisson packet generation model is widely adopted in the literature and state-of-the-art simulation frameworks, as it captures both the discrete and inherently random nature of message transmissions typical of low-power wide-area networks [45], [47], [48]. While practical traffic may deviate from a Poisson process, the average interference probability is primarily determined by the aggregated activity of the concurrently active devices. Furthermore, since MARLTC operates on gateway-level aggregated metrics (e.g., SNR), variations in the traffic model affect these measurements without altering the underlying optimization framework. To assess robustness, we have also implemented a bursty traffic model, in which each end device transmits with a per-interval probability scaled by a burst factor. Additional evaluation results under this model are provided in the Appendix.

The LoRa modulation allows decoding below the noise floor as long as the signal-to-interference ratio (SIR) remains above a spreading-factor-dependent threshold $\omega_{f_i f_j}$ [45], [49]. Hence, the probability that a transmission from e_j interferes with a transmission from e_i received at g_k is

$$\begin{aligned} \mathbb{P}(RSSI_{ik} - RSSI_{jk} < \omega_{f_i f_j}) \\ &= \mathbb{P}(N_{jk} - N_{ik} < \omega_{f_i f_j} - (z_{ik} - z_{jk})) \\ &= \frac{1}{2} + \frac{1}{2} \text{erf} \left(\frac{\omega_{f_i f_j} - (z_{ik} - z_{jk})}{2\sqrt{2}\sigma} \right). \end{aligned} \quad (6)$$

The combined probability that e_i 's transmission remains uncorrupted by all other devices is then

$$\zeta_{ik} = \prod_{j \in \{1..N\}, i \neq j} (1 - h_{ij} \mathbb{P}(RSSI_{ik} - RSSI_{jk} < \omega_{f_i f_j})). \quad (7)$$

This formulation adopts a pairwise independence approximation, commonly used in stochastic wireless network analysis for dense deployments [45], [49], [50]. Instead of modeling the joint interference from all simultaneous transmitters, which is generally intractable, it approximates interference as independent pairwise capture events. This captures the dominant first-order behavior of LoRa reception, including SIR-based capture and spreading-factor-dependent interference, while neglecting cumulative and higher order interference effects.

3) *End Device PDR and EE*: To calculate the overall PDR of a single end device e_i , considering all available gateways, the gateway-specific PDRs can be combined as follows:

$$PDR_i = 1 - \prod_{k \in \{1..K\}} (1 - PDR_{ik}). \quad (8)$$

The resulting end-to-end PDR is thus a nonseparable stochastic objective function of the joint transmission configuration. This mapping explicitly captures the interagent coupling induced by shared spectrum access and spatial reuse across multiple gateways, providing the physical-layer foundation for the proposed multi-agent problem. Subsequently, the EE of an end device e_i can be defined as the number of payload bits successfully transmitted per amount of energy consumed by the end device. The EE in bits/mJ can be calculated as follows:

$$EE_i = \frac{8L \cdot PDR_i}{C_{p_i} T_i} \quad (9)$$

where L is the number of payload bytes, T_i is the time required for transmitting a single packet, and C_{p_i} is the power consumption in mW using a transmission power setting $p_i \in \mathcal{P}$.

IV. MARLTC: A MULTIAGENT ADR

MARLTC is a MARL framework for adaptive transmission control in dense LoRaWAN networks. MARLTC optimizes the SF and transmission power of end devices, considering co-channel interference to meet application-specific reliability targets while minimizing energy consumption. In this section, we describe the problem formulation, training with multiple policies, and the resulting transmission selection algorithm.

A. Multiagent Problem Formulation

We formulate the transmission parameter selection problem as a cooperative Markov game, a multi-agent extension of Markov decision processes [51]. We aim to maximize the EE while maintaining a minimum required PDR by adjusting the SF and transmission power settings of end devices, as follows:

$$\arg \max_{F, P} \sum_{i \in \{1, \dots, n\}} EE_i \quad (10a)$$

with

$$\frac{PDR_i}{PDR_i^{\max}} \geq PDR_i^{\text{thr}} \quad \forall i \in \{1, \dots, n\} \quad (11a)$$

where, $F = \{f_1, \dots, f_n\}$ and $P = \{p_1, \dots, p_n\}$ denote the SF and transmission power settings for all n end devices. The EE EE_i and PDR PDR_i of each device are modeled as functions of the transmission parameters of all devices, i.e., $EE_i = EE_i(F, P)$ and $PDR_i = PDR_i(F, P)$. $PDR_i^{\max}$ denotes the maximum achievable PDR of device e_i , and PDR_i^{thr} is its configurable threshold. Both metrics are stochastic quantities induced by the underlying wireless channel and the collective co-channel interference patterns of concurrent devices. This formulation represents a constrained stochastic optimization problem over a coupled joint action space, where the PDR constraint enforces the feasibility of admissible transmission configurations and prevents the policy from converging to energy-efficient but unreliable operating regimes.

A Markov game [51] for n agents is defined by a set of states $\mathcal{S}$, a collection of action sets $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$, and a collection of observation sets $\{\mathcal{O}_1, \dots, \mathcal{O}_n\}$, one for each agent. The state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_n \rightarrow \mathcal{S}$ specifies the probability distribution over the next state, given the current state and the joint actions of all agents. Each agent e_i receives local observations $o_i : \mathcal{S} \rightarrow \mathcal{O}_i$ and selects actions according to a stochastic policy $\pi_{\theta_i} : \mathcal{O}_i \times \mathcal{A}_i \rightarrow [0, 1]$. In MARL, each agent is associated with a reward function $r_i : \mathcal{S} \times \mathcal{A}_i \rightarrow \mathbb{R}$, aiming to learn a policy that maximizes its expected cumulative return $R_i = \sum_{t=0}^T \gamma^t r_t^i$, where γ is the discount factor and T denotes the time horizon. This multi-agent system is formally characterized as a decentralized partially observable Markov decision process (Dec-POMDP). In this framework, the global network state evolves according to a Markovian transition kernel driven by joint agent actions, while individual end devices operate under partial observability, mitigating the inherent nonstationarity of the dense network through the coordination logic learned during centralized training.

In our specific problem, we adopt a CTDE approach. During training, interconnected gateways and the NS collect transmission data and global context to learn a shared policy applicable to all agents. This centralized training phase leverages the broader network view available at the gateways to improve policy coordination and convergence. Once training is complete, policy inference is executed independently at each end device, based on a combination of local observations and a limited amount of aggregated network information.

All end devices are assumed to be homogeneous in capabilities and objectives. In terms of capabilities, each end device is endowed with an identical set of controllable actions, namely the selection of the SF and the transmission power level. This assumption does not preclude differences in manufacturer, hardware generation, or other device-specific characteristics; rather, it reflects the practical deployment scenario considered in this work, in which all devices comply with the LoRaWAN standard and therefore expose a common configuration space to the MARL agents. With respect to objectives, all agents are trained to optimize a common global goal, namely, maximizing the overall EE of the network while satisfying a prescribed PDR requirement. Consequently, all agents pursue identical optimization criteria and act within the same action space, while remaining responsive to different local observations induced by spatially and temporally varying network conditions, such as differences in experienced interference levels. This allows for a shared formulation where observation spaces, action spaces, reward functions, and policy structures: $\mathcal{O}_i = \mathcal{O}_j$, $\mathcal{A}_i = \mathcal{A}_j$, $r_i = r_j$, $\pi_{\theta_i} = \pi_{\theta_j} \quad \forall i, j \in \{1, \dots, n\}$. This shared policy supports scalability, efficient coordination, and generalization while enabling robust adaptation to dynamic and heterogeneous network conditions.

1) *Observation*: The observation space comprises a compact feature set designed to capture the dominant latent variables of the network, specifically link quality and co-channel interference. We define the local observation of every agent e_i at its individual step τ as

$$o_i^\tau = (\mathcal{H}_i^\tau, q_f^\tau, PDR_i^{\text{thr}}) \quad (12)$$

where $\mathcal{H}_i^\tau$ is the transmission history set, which contains information about the past h packet transmissions from end device e_i that were successfully received by at least one gateway in the network; q_f^τ is the share of end devices currently configured to use each SF $f \in \mathcal{F}$; and PDR_i^{thr} is a value controlling the desired relative PDR threshold normalized to an interval of $[0, 1]$. While this observation does not constitute a sufficient statistic for the global network state, it minimizes the information overhead during decentralized execution while preserving the essential features required to approximate the gradient of the joint objective function. We define the transmission history set of past packet transmissions included in observations as follows:

$$\mathcal{H}_i^\tau = \left(\mathcal{H}_i^{\tau(1)}, \dots, \mathcal{H}_i^{\tau(h)} \right) \quad (13)$$

where $\mathcal{H}_i^{\tau(1)}$ represents the last packet transmission before τ , and $\mathcal{H}_i^{\tau(h)}$ represents the h th last packet transmission before τ . Each transmission set is defined as follows:

$$\mathcal{H}_i^{\tau(v)} = \left(SNR_{i,1}^{\tau(v)}, \dots, SNR_{i,m}^{\tau(v)}, f_i^{\tau(v)}, p_i^{\tau(v)} \right) \quad (14)$$

where $SNR_{i,m}^{\tau(v)}$ is the normalized SNR of the v th last packet transmission before τ from end device e_i as received by gateway g_m , and $f_i^{\tau(v)}$ and $p_i^{\tau(v)}$ are the normalized SF and transmission power settings used for that transmission. Normalization is used to map the actual SNR of packet transmissions to the $[-1, +1]$ interval, as suggested by [52], following initial experiments that indicate improved training results. Transmissions with an SNR below $SNR^{min} = -20$ dB should not be received by a gateway, even with the SF resulting in the highest receiver sensitivity. Values above $SNR^{max} = 10$ dB indicate transmission strength well above the requirements. Although RSSI could be employed to assess transmission quality, we opted for SNR based on empirical observations that demonstrated that SNR provides a more reliable and discriminative metric for characterizing link quality in our target scenarios. This choice is also motivated by the fact that SNR inherently accounts for both the received signal power and the noise floor, yielding a more informative indication of link quality. Moreover, in LoRaWAN systems, RSSI typically spans a wide range (e.g., approximately from -140 to -40 dBm), whereas SNR generally ranges from about -20 to $+10$ dB, directly representing how much the signal stands above background noise and interference, offering a narrower and more consistent range that correlates more closely with actual decoding performance. In short, the narrower and more informative range of SNR facilitates RL by reducing the effective observation space, thereby improving learning stability and convergence in interference-rich, dense network scenarios. Consequently, the agents' observations were designed to incorporate SNR measurements as they led to improved performance and more consistent learning outcomes.

We include q_f^τ in observations to give agents the ability to recognize and avoid congested SF settings. The value PDR_i^{thr} controls the desired relative PDR threshold. We apply max-min scaling using $PDR_i^{thr_min} = 0$ to allow no priority to be given to achieved PDR and $PDR_i^{thr_max} = 0.95$ to limit the priority given to achieved PDR. While PDR can

theoretically have a value of 1 (100% of all packets successfully received), we observed during initial experiments that setting such a strict PDR constraint hindered the learning of our agents. The PDR threshold is included in observations to allow dynamic, individual, and continuous adaptation of PDR prioritization. For additional details on the design and implementation of MARLTC, including how these observation metrics are obtained, we refer the reader to Section IV-F.

2) *Action*: The action taken by each agent based on the observation made in its step τ is

$$a_i^\tau = (f_i^\tau, p_i^\tau) \quad (15)$$

where $f_i^\tau \in \mathcal{F}$ is the SF setting, and $p_i^\tau \in \mathcal{P}$ is the transmission power setting chosen by the agent and assigned to the end device for future transmissions. Concluding, each agent's set of possible actions is $\mathcal{A} = \mathcal{F} \times \mathcal{P}$, resulting in an action space dimension $|\mathcal{F}| \times |\mathcal{P}|$.

3) *Reward*: We define the reward function as

$$r_i^\tau = \begin{cases} EE_i, & \text{if } \frac{PDR_i}{PDR_i^{max}} \geq PDR_i^{thr} \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

where EE_i is the normalized EE, PDR_i is the PDR considering the current network configuration, PDR_i^{max} is the theoretically maximum achievable PDR, and PDR_i^{thr} is the specified PDR threshold controlling the prioritization of PDR over EE, all of an end device e_i . The main EE objective is addressed by basing the reward function on the EE of each end device. The PDR constraint introduced in (11a) is addressed by including the desired PDR threshold as a condition within the reward function. A hard constraint (yielding a reward of zero when the PDR threshold is violated) was deliberately chosen over a soft penalty formulation. This design strictly prevents the agent from exploiting the reward function by sacrificing communication reliability for extreme EE, while simultaneously eliminating the need to tune an additional penalty coefficient hyperparameter. We normalize EE using the theoretically maximum achievable EE end devices [52], which depends on the assumed power consumption profile and the SF resulting in the shortest packet transmission time, and it is considered identical for all end devices within the network. Its value is empirically determined by simulating a network with an end device placed at the same location as a gateway and using the lowest possible transmission power setting, as well as the SF resulting in the shortest packet transmission time.

B. Training Environment

For the training of MARL policies, we developed a custom training environment based on FAST-LoRa¹ [34], which enables the generation of observations and rewards in response to actions from agents, as described in Section IV-A. FAST-LoRa incorporates detailed physical-layer effects, including co-spreading and inter-SF interference and capture effects. The training environment simulates a LoRaWAN network in which multiple agents, each representing an end device, interact with a dynamic and interference-prone wireless channel. Each episode starts by initializing the FAST-LoRa simulator with the

¹CodeOcean capsule available at <https://codeocean.com/capsule/1001925>

spatial layout of end devices and gateways, as well as with relevant physical-layer and MAC-layer parameters. End devices are initialized with default transmission settings, including the well-known ADR Backoff mechanism, which allows them to adapt their parameters in the event of connectivity loss. Throughout the simulation, end devices periodically attempt to transmit packets, and the simulator models whether these transmissions are successfully received by any gateway based on sampled SNR values and interference conditions.

A buffer maintains a history of successful transmissions ($\mathcal{H}_i^T$) for each agent, which is later used to construct observations. Agents receive observations once they have accumulated enough transmission history. These observations include recent transmission performance, current network-wide SF usage, and a normalized target PDR. Based on this, each agent selects new transmission parameters (e.g., SF and power level), which are assumed to be successfully applied. The environment continues to simulate transmissions, generate observations and rewards, and accept actions from agents until a fixed simulation time horizon is reached. Rewards are computed to incentivize efficient and reliable communication, balancing EE and successful delivery. Communication loss triggers adjustments in transmission settings using a simplified ADR back-off mechanism aligned with the Semtech specification [53]. In FAST-LoRa, the ADR backoff mechanism is modeled through the function $\theta_{\text{ADRBackoff}}(\cdot)$, which encapsulates the configuration parameters governing how an end device reacts to prolonged communication loss. Specifically, $\theta_{\text{ADRBackoff}} = (n_{\text{ADR_ACK_LIMIT}}, n_{\text{ADR_ACK_LIMIT}})$ defines the timeout thresholds and the set of admissible transmission parameters used during recovery. The parameter $n_{\text{ADR_ACK_LIMIT}}$ specifies the maximum number of consecutive uplink transmissions without receiving an ADR response before the end device explicitly requests new transmission parameters, while $n_{\text{ADR_ACK_DELAY}}$ determines the additional number of uplink attempts after which recovery actions are triggered. To support reproducibility and further research, the full implementation of MARLTC is publicly available at [54].

C. Policy Optimization

To train policies shared among all agents interacting with an environment, we use proximal policy optimization (PPO) [55], a well-known state-of-the-art RL algorithm suitable for multi-agent scenarios. The training process is implemented using Ray RLlib,² version 2.23.0, which provides scalable and flexible RL primitives for multi-agent environments. RLlib supports asynchronous agent interaction by allowing agents to operate on independent internal time steps. Specifically, while global environment time progresses uniformly at step t , only a subset of agents may receive an observation and produce an action at that point. For this purpose, we have introduced the concept of an individual step τ for each agent, which maps to the global time step t at which the agent receives an observation and takes an action. This enables accurate temporal tracking and robust policy updates, even when agents interact with the environment at irregular or asynchronous intervals.

TABLE I
HYPERPARAMETER USED FOR TRAINING

Hyperparameter	Value
n	$\sim \mathcal{U}(\{50, \dots, 500\})$
m	4
L	1000 m
X	$\sim \mathcal{U}([0, L]^2) m^2$
Y	$\{(500 + 250 \cos \frac{2k\pi}{K}, 500 + 250 \sin \frac{2k\pi}{K}) k \in \{1, \dots, K\}\} m^2$
EE^{max}	16 $\text{bits}/m.J$
h	20
$\theta_{\text{ADRBackoff}}(\cdot)$	$n_{\text{ADR_ACK_LIMIT}} = 64, n_{\text{ADR_ACK_DELAY}} = 32$
T	$2 \times 24 \times 3600 \text{ s}$
Φ_π, Φ_V	Fully connected, ReLU activation function 4 layers with [256, 256, 256, 64] neurons shared parameters between Φ_π and Φ_V Adam [57] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$
$\mathcal{O}$	
α	$5 \cdot 10^{-5}$
γ	$\sim \text{LogUniform}_{10}(0.9, 0.999)$
ϵ	0.25
λ	0.9
c_1	$\sim \text{LogUniform}_{10}(0.01, 1)$
c_2	$\sim \text{LogUniform}_{10}(10^{-4}, 10^{-2})$
n_{iter}	100
n_{step}	4096
n_{epoch}	5
$n_{\text{minibatch}}$	128
$n_{\text{population}}$	4
$n_{\text{pbt}}^{\text{iter}}$	5
$n_{\text{pbt}}^{\text{iter_burn_in}}$	0
Presample	0.25
η	$\bar{r}_\pi = \frac{\sum_{i=0}^N \sum_{\tau} r_i^\tau}{N}$
ρ	0.25
σ	0.2

D. Training Hyperparameters

While there are well-known recommendations for the initialization of certain hyperparameters [52], others require proper tuning to achieve satisfactory results. To this end, we employ population-based training (PBT) [56], a hyperparameter tuning strategy commonly used in numerous deep RL tasks, which provides faster results than sequential optimization and uses fewer computational resources than parallel random or grid search. The set of hyperparameters chosen for training has been summarized in Table I.

Based on prior studies on LoRaWAN scalability, each gateway can reliably support several hundred to a few thousand devices [21], [58], [59]. During the training, the number of end devices n is sampled uniformly from 50 to 500 when initializing each training environment, enabling coverage of various congestion levels. Devices are placed randomly according to a uniform distribution (X) within a $(1000\text{-}m^2)$ area to ensure spatial diversity. This configuration forces the learning agent to generalize over a wide set of radio conditions and traffic patterns, promoting robustness and increased EE across deployments. We employ $m = 4$ gateways in a circular pattern (Y) with identical positions across all environments to reduce model complexity and focus learning on device-side adaptation. While this introduces structural bias into the learned policies, it enables tractable training and reproducibility while preserving key characteristics of multigateway setups.

Agents adjust transmission parameters after every $h = 20$ packet transmissions, matching Semtech's ADR implementation for comparability and alignment with existing standards. Each training environment simulates a network for $T = 2$ days of virtual time. By observing the learning curves across multiple training runs (see Section IV-E), we found that this

²<https://docs.ray.io/en/releases-2.23.0/rllib/index.html>

simulation horizon consistently allowed agents to converge within the simulation period, exposing them to both early phase and steady-state network behavior.

Policy and value function networks share weights and follow an architecture adapted from [52], tuned via ablation studies. Optimization uses Adam [57] with a reduced learning rate $\alpha = 5 \cdot 10^{-5}$, which provided stable convergence during early experiments. All other optimizer settings follow the defaults. PPO hyperparameters, including the reward discount factor γ , value loss coefficient c_1 , and entropy regularization coefficient c_2 , and their initialization distributions, are adopted from [52]. PBT is employed to dynamically tune these critical parameters during training.

Each trial (distinct hyperparameter configuration) trains for $n_{iter} = 100$ iterations, with $n_{step} = 4096$ steps per iteration. This setup ensures sufficient experience collection even under the extreme case of 500 devices, each transmitting every second and achieving a 100% PDR. In realistic conditions, where the PDR is lower, this configuration is ample to populate the replay buffer. Collected experiences per iteration are used for $n_{epoch} = 5$ training epochs with minibatches of size $n_{minibatch} = 128$, as standard in PPO. This setup balances training stability with sample efficiency.

PBT runs with $n_{population} = 4$ concurrent trials. Exploitation and exploration occur every $n_{iter}^{pbt} = 5$ iterations. Due to computational constraints, noburn-in is applied ($n_{iter}^{pbt_burn_in} = 0$), which may lead to premature pruning of trials. Trials are ranked by average reward $\eta = \bar{r}_\pi$ from the previous iteration. The bottom $\rho = 0.25$ quantile is replaced by clones of better-performing trials, with hyperparameters either resampled with probability $p_{resample} = 0.25$ or perturbed by a factor $\sigma = 0.2$ [56].

E. Trained Policies

This section presents the three distinct policies trained in this work, each optimized for different tradeoffs between EE and PDR. These policies reflect varying operational priorities, ranging from aggressive energy saving to reliable communication, with one policy also enabling dynamic adjustment at runtime.

- P1: A policy maximizing the priority of the achieved EE without considering any PDR constraints.
- P2: A policy maximizing the PDR while aiming to achieve maximum EE.
- P3: A policy that can be dynamically reconfigured to achieve the desired PDR while aiming to achieve maximum EE. Instead of a uniform distribution for the PDR^{thr} , we employ a power-transformed uniform distribution $\mathcal{D}$. This is because this policy behavior is most sensitive to changes in the PDR threshold value in the upper range, requiring increased focus on that range during training.

The PDR thresholds employed in each policy have been summarized in Table II. The training progress of the best-performing trial resulting in the final policies is visualized in Fig. 1. It shows the finally achieved mean EE and PDR of end devices and the share of end devices with a final PDR below the configured PDR^{thr} across all environments simulated within a training iteration, as well as the observed

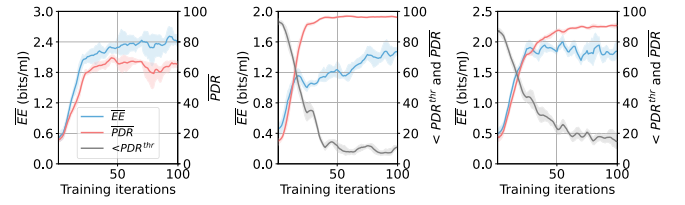


Fig. 1. Learning curves of policies P1–P3 (left to right). P1 maximizes EE, P2 maximizes PDR while maintaining EE, and P3 targets a configurable PDR. The figure includes the average EE, PDR, and the fraction of devices below the PDR target (P2–P3), illustrating convergence during training.

TABLE II
PDR THRESHOLDS USED FOR THE DIFFERENT
TRAINED POLICY VARIANTS

Policy	Description	PDR Threshold (PDR^{thr})
Policy 1 (P1)	Static Low PDR Priority	0
Policy 2 (P2)	Static High PDR Priority	0.95
Policy 3 (P3)	Dynamic PDR Priority	$\mathcal{D}(0, 0.95)$

variations of these metrics. The results indicate that the policy convergence occurred during the training.

F. MARLTC Algorithm

This section describes the proposed MARLTC algorithm for the efficient selection of transmission parameters in LoRaWAN networks. MARLTC employs similar metrics as standard ADR algorithms [60]. Therefore, only minor modifications to existing LoRaWAN NSs are required to integrate MARLTC. MARLTC is also compatible with standard LoRaWAN end devices since it does not require any modifications to their firmware or the communication protocol. By leveraging the well-known centralized training and decentralized execution MARL paradigm, the pretrained policies are deployed at the end devices, which dynamically perform inference on these policies to determine the optimal transmission parameters of end devices.

As shown in Algorithm 1, at the end devices, MARLTC first requires as input the trained policy (π), the number of SNR measurements in observations (h), and the required PDR^{thr} . This is especially relevant when a dynamic policy is used (i.e., P3). Upon initializing a MARLTC instance, a buffer $\mathcal{H}$ is created to store information on received packet transmissions.

Whenever a new packet transmission occurs, information about that transmission, such as the measured SNR and the transmission parameters used, is normalized and stored in the NS. Once the buffer reaches the required size h , the closest gateway sends the transmission history $\mathcal{H}$ and the aggregated network information q_f^T to the end device using a downlink message. The SNR history is encoded using one absolute 8-bit value per gateway, followed by 19 delta-encoded samples quantized on 4 bits each (range from -8 to $+7$ dB), and q_f^T is represented using 1 byte. Considering the best four gateways, this results in a total downlink payload of 43 bytes. It is worth noting that gateways do not participate in the decision-making process and incur no additional computational load beyond standard LoRaWAN operation. Moreover, the downlink updates remain within standard LoRaWAN payload limits

The convergence time t_{conv} provides valuable insights into how fast a given ADR mechanism reaches suitable transmission parameters. We define t_{conv} as the earliest instant at which the network enters a stable state in terms of the average EE EE^τ and packet delivery ratio PDR^τ remaining within a $\pm 5\%$

band around their respective final values for a continuous window of $\Delta T = 0.05 \times T$

$$t_{\text{conv}} = \min \left\{ t \mid \forall \tau \in [t, t + \Delta T] : \left| \frac{EE^\tau - EE_{\text{final}}}{EE_{\text{final}}} \right| \leq \epsilon \wedge \left| \frac{PDR^\tau - PDR_{\text{final}}}{PDR_{\text{final}}} \right| \leq \epsilon \right\} \quad (19)$$

where $\epsilon = 0.05$ (5%), and τ denotes the metric sampling instants during the simulation.

C. Baselines

We compare MARLTC against three well-known state-of-the-art baselines: ADR, G-ADR, and LP-MAB. ADR is the default ADR mechanism proposed by Semtech, commonly used as a baseline in LoRaWAN research. For evaluation, we implemented ADR as specified in [60], adapting it to comply with the EU863–870 LoRaWAN channel plan [61]. ADR maintains a buffer of recent SNR values and uses the maximum value to estimate the link margin, which then determines the number of steps to adjust the SF and transmission power. G-ADR is an enhanced version of the default ADR algorithm introduced in [13], aiming to improve stability by mitigating the impact of SNR outliers. Instead of using the raw maximum SNR value, G-ADR applies Gaussian filtering to the buffer of received SNRs, removing values that fall outside a defined statistical range. The filtered average of the values is then used in place of the maximum SNR when calculating transmission parameter adjustments. G-ADR prevents overly optimistic link quality estimates, reducing the likelihood of performance degradation due to aggressive data rate increases. LP-MAB [28] is an ADR mechanism based on the multiarmed bandit (MAB) framework, where each transmission configuration is modeled as an arm. The algorithm combines nonstationary adversarial and stochastic strategies, employing exponential weighting for exploration and exploitation alongside successive elimination. We implemented LP-MAB in Python according to the original paper, using a transmission power-based reward and EXP3 probability updates, while explicitly maintaining exploration and exploitation phases with a sliding observation window.

VI. EVALUATION RESULTS

In this evaluation section, we comprehensively assess the performance of the proposed MARLTC algorithm compared to existing solutions under a variety of network conditions. The evaluation begins with a comparison between dynamic and static policies employed by MARLTC, establishing the settings used for subsequent experiments. We then analyze multiple variants of MARLTC across diverse scenarios, including networks with a variable number of end devices, gateways, and device placement areas, exposing also the policy to conditions not observed during training and assessing its generalization to unseen configurations. In addition, we evaluate dynamic network conditions by simulating periodic addition and removal of end devices, demonstrating the low convergence time and rapid responsiveness of MARLTC under topology changes. This multifaceted analysis highlights the adaptability and effectiveness of MARLTC under heterogeneous network conditions.

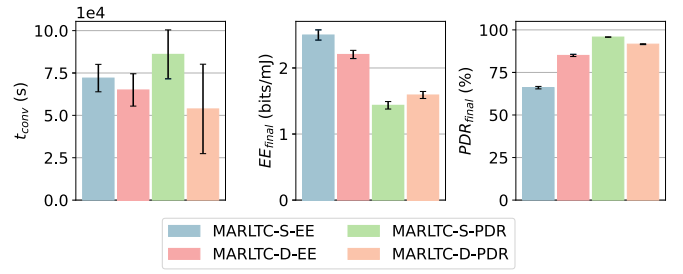


Fig. 2. Performance metrics of MARLTC policies over ten runs. Static policies (MARLTC-S-EE, MARLTC-S-PDR) target maximum EE or PDR, while dynamic policies (MARLTC-D-EE, MARLTC-D-PDR) allow configurable PDR.

A. MARLTC Policies Comparison and Fairness Analysis

The performance of MARLTC using different trained policies is evaluated over ten independent runs, with a time limit of $T = 3$ days per run. The simulated environment comprises $n = 500$ end devices randomly distributed within a square area of 1000 m^2 and connected to $m = 4$ gateways positioned as specified in Table I. No device turnover is considered in this experiment. We compared the following trained policies.

- 1) MARLTC-S-EE using a static policy trained to always aim for maximum EE (P1).
- 2) MARLTC-D-EE dynamic policy (P3) with $PDR^{\text{thr}} = 0$ to maximize EE. This policy enables the dynamic configuration of the desired PDR using the input parameter PDR^{thr} .
- 3) MARLTC-S-PDR involves a static policy trained to always aim for a maximum PDR while maintaining the highest possible EE (P2).
- 4) MARLTC-D-PDR dynamic policy (P3) with $PDR^{\text{thr}} = 1$ to maximize the PDR. This policy enables the dynamic configuration of the desired PDR using the input parameter PDR^{thr} .

Fig. 2 shows the average and standard deviation of the convergence time, EE, and PDR across all simulations per algorithm. While the quality of the network state achieved by MARLTC using statically trained policies (for either a maximum PDR or EE) exceeds the performance with a dynamic policy configured to prioritize either EE or the PDR, the convergence time using the dynamic policy is lower. Despite their larger state and action space, the dynamic policies may converge faster due to the lower quality network states they tend to produce, effectively reducing the difficulty of reaching the performance threshold. The effects of reduced convergence times are evaluated in Section VI-D.

The EE and PDR of the network over simulation time are shown in Fig. 3, with average and standard deviation across all simulations per algorithm. The final EE of MARLTC using the dynamic policy (where prioritization of EE over the PDR can be adjusted at runtime) remained 11.9% below the final EE of MARLTC using the corresponding static policy (non-adjustable at runtime). Similarly, the final PDR of MARLTC with the same dynamic policy configured for maximum PDR priority remained 4.3% below the PDR achieved using the corresponding static policy. The ability to adapt an algorithm's behavior at runtime appears to come at the cost of reduced

TABLE III
JAIN FAIRNESS INDEX ACROSS END DEVICES

Metric	MARLTC-D-EE	MARLTC-D-PDR	MARLTC-S-EE	MARLTC-S-PDR
PDR	0.98	0.99	0.92	1.00
EE	0.51	0.48	0.55	0.44

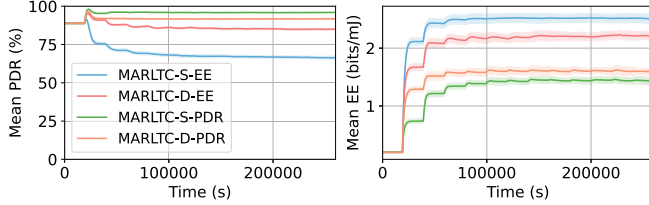


Fig. 3. Network performance over simulation time for different MARLTC policies.

maximum performance with respect to the prioritized metric. Specifically, the observed final EE and PDR are consistently lower compared to those achieved by statically trained policies.

A further relevant dimension in the performance evaluation of MARLTC concerns fairness across end devices. While aggregate metrics such as mean PDR and EE quantify overall network efficiency, they do not capture whether performance is evenly distributed among nodes or the potential starvation of some devices receiving significantly worse transmission parameters than others. To explicitly assess this aspect, we compute the Jain fairness index [62] across end devices based on PDR and EE. This metric provides a normalized measure between 0 and 1, where values close to one indicate uniform performance across devices and lower values reveal increasing disparity. Table III reports the Jain fairness index over ten runs for all evaluated policies. With respect to PDR, all policies attain high fairness indices, indicating that reliability is distributed nearly uniformly across devices under both static and dynamic settings. This demonstrates that the learning process does not enhance aggregate reliability at the expense of a subset of nodes, thereby preventing starvation in packet delivery. The Jain index for EE is inherently lower due to the spatial distribution of end devices. To meet target PDR levels, end devices must employ different transmission parameters, which reduces EE fairness. This also reflects the tradeoff between PDR and EE: stricter PDR constraints reduce EE fairness, while stringent energy constraints degrade PDR fairness. Overall, although MARLTC does not explicitly optimize for fairness, its Markov game formulation and reward design implicitly promote balanced performance across end devices.

B. Scalability Analysis With Increasing Number of EDs

To assess the performance and scalability of MARLTC in dense LoRaWAN networks compared to baseline algorithms, we conduct a series of experiments in which the number of end devices is systematically varied. We employ the same simulation configuration described in Section VI-A, varying the number of end devices n , which takes values of 200, 500, 1000.

Fig. 4 presents the performance metrics obtained across all simulations for each algorithm, evaluated under varying

numbers of end devices in the network. Table IV provides a comparative analysis of the metrics achieved by each algorithm relative to those attained by the ADR algorithm, which serves as the main baseline. As can be observed, MARLTC performed well with a varying number of end devices, even in network sizes not seen during training. First, the results indicate vastly reduced network convergence times with MARLTC in all tested configurations compared to ADR, G-ADR, and LP-MAB, with reductions of 33.9%, 67.9%, and 16.6%, respectively.

Among all MARLTC variants, the final PDR consistently exceeded that of ADR, with gains ranging from 9.1% to 24.1% across the evaluated network scenarios, highlighting the algorithm's effectiveness in maintaining reliable packet delivery even under high congestion. Notably, the MARLTC variant statically prioritized for high PDR outperformed G-ADR in networks comprising 1000 end devices, achieving a 5.0% higher final PDR alongside a substantially greater final EE, with an increase of 62.1%. Only the version statically configured with a high EE priority reached a final EE comparable to ADR, with an average improvement of 1.2%, although this came at the cost of a reduced final PDR, which was, on average, 14.7% lower than ADR. The difference between MARLTC variants and ADR and G-ADR slightly decreased with an increasing number of end devices, especially for versions with high, nonadjustable PDR prioritization. LP-MAB achieved a final PDR comparable to MARLTC-S-EE, though slightly lower than other variants due to its exploration phases that test different transmission parameter combinations. Its final EE remained just below G-ADR, as the reward reduces transmit power but does not penalize high SFs, increasing time-on-air (ToA) and energy consumption.

Comparing network state over simulation time with different algorithms (shown in Fig. 5), we observe different convergence behavior in MARLTC, LP-MAB, ADR, and G-ADR. In networks with a lower number of end devices [Fig. 5(a)] and relatively low congestion (initial PDR around 98%), most MARLTC algorithms assigned transmission parameters to end devices, resulting in a high EE already with an initial update after 20 000 s, thereafter quickly converging to the final network state. LP-MAB exhibited slower but stable convergence, maintaining PDR comparable to MARLTC variants while slightly lagging in EE. In networks with a higher number of end devices [Fig. 5(b)] and relatively high congestion (initial PDR around 66%), the transmission parameter assignment by MARLTC algorithms resulted in more gradual convergence to the final network state. The MARLTC algorithm that used a policy trained for static, nonadjustable high PDR (MARLTC-S-PDR) was especially affected by high initial network congestion, explaining the overproportional increase in convergence time with an increasing number of end devices compared to other MARLTC algorithms. LP-MAB maintained moderate PDR under these conditions, highlighting its robustness but lower EE compared to MARLTC variants.

Takeaway: MARLTC improves PDR while maintaining EE with reduced convergence times that enable the prompt adaptation of transmission parameters to changing LoRaWAN conditions, improving responsiveness and efficiency. MARLTC also scales with the number of end devices.

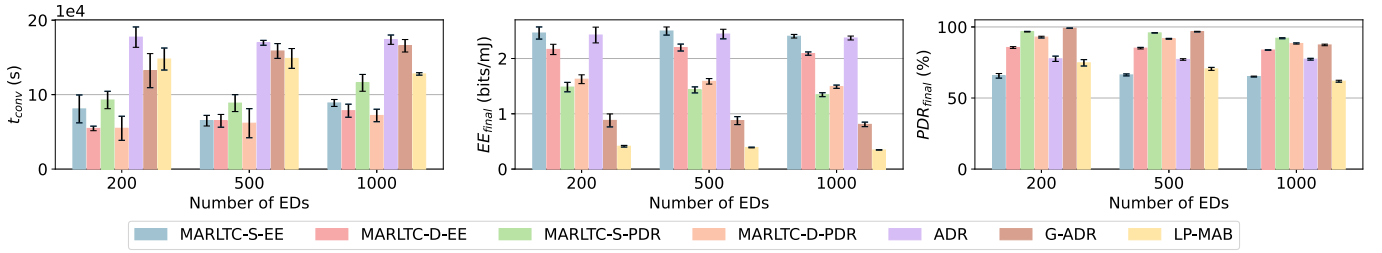


Fig. 4. Network performance for varying numbers of end devices uniformly distributed within a (1000-m²) area.

TABLE IV

RELATIVE PERFORMANCE METRICS COMPARED TO THE DEFAULT ADR MECHANISM FOR DIFFERENT NUMBERS OF END DEVICES. NEGATIVE VALUES INDICATE REDUCTIONS, AND POSITIVE VALUES INDICATE IMPROVEMENTS RELATIVE TO ADR

Metric	t_{conv} (% difference)				EE_{final} (% difference)				PDR_{final} (% difference)			
Number of end devices	200	500	1000	Mean	200	500	1000	Mean	200	500	1000	Mean
Algorithm												
MARLTC-S-EE	-51.0	-59.2	-51.7	-54.0	+0.4	+2.2	+0.9	+1.2	-15.5	-14.0	-14.4	-14.7
MARLTC-D-EE	-57.2	-60.9	-53.3	-57.1	-10.4	-10.1	-11.9	-10.8	+9.4	+10.1	+9.1	+9.6
MARLTC-S-PDR	-55.7	-49.3	-33.9	-46.3	-40.0	-41.1	-43.6	-41.6	+24.1	+24.1	+20.1	+22.8
MARLTC-D-PDR	-63.6	-67.9	-57.9	-63.1	-33.6	-35.0	-37.7	-35.5	+19.0	+18.8	+15.3	+17.7
G-ADR	-20.6	-9.2	-5.6	-11.8	-63.2	-63.5	-65.2	-64.0	+27.3	+25.3	+14.4	+22.3
LP-MAB	-16.6	-12.4	-26.5	-18.5	-83.0	-84.0	-85.4	-84.1	-3.7	-8.6	-20.0	-10.8

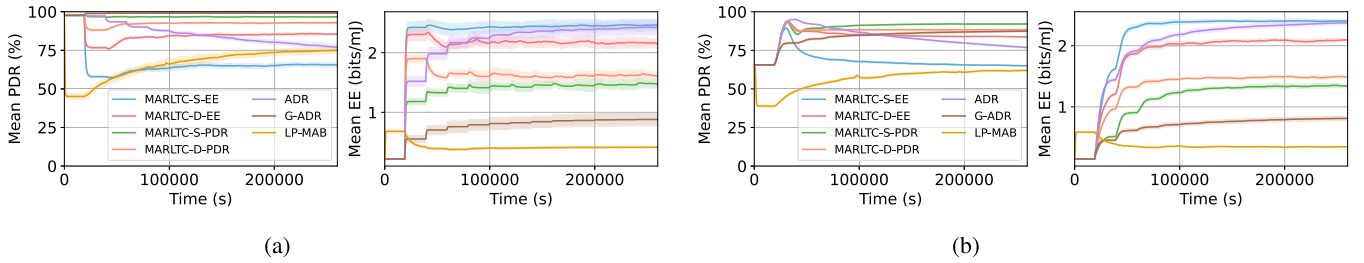


Fig. 5. Network performance over time for (a) 200 and (b) 1000 end devices uniformly distributed within a (1000-m²) area, showing the effect of network size on convergence and stability.

C. Performance Analysis for Different Placement Areas

In this experiment, we evaluate MARLTC performance under varying spatial distributions of end devices, including those not seen during training. This allows us to assess the generalization ability of the learned policies when exposed to unseen environmental conditions. It is important to note that evaluating the policies in a substantially different environment in terms of area represents a worst case scenario, which enables the assessment of the limits of generalization and the behavior of learned strategies under conditions beyond the training environment. In such cases, retraining may be required in practice, as agents must learn the environmental and channel characteristics to determine appropriate transmission configurations. In practice, this process can be accelerated by initializing the policy with parameters from a previously trained model rather than training from scratch. By leveraging prior knowledge embedded in the pretrained weights, the learning process can converge more rapidly and require fewer interactions with the environment, thereby improving adaptation efficiency and enhancing real-world deployment feasibility.

For this experiment, simulation parameters remain unchanged, except for end-device positions (X), which are

uniformly distributed within square areas of size $L \times L$ m² (see Table I), where $L \in \{500, 1000, 2000\}$ m, to simulate different spatial densities. The results are presented in Fig. 6 and Table V. As in Section VI-B, all MARLTC variants achieved reduced convergence times compared to ADR, G-ADR, and LP-MAB, with improvements ranging from 17.3% to 75.3%. All the baselines experienced an increased convergence time as a function of the placement area, especially LP-MAB, due to its exploration time, where suboptimal transmission parameters may be selected. The MARLTC variant trained with a fixed high EE priority slightly outperformed ADR in final EE (by 2.4%) in the network with a (1000-m²) placement area, i.e., the same configuration used during training. In all other scenarios, this variant's final EE fell below that of ADR but consistently exceeded LP-MAB and G-ADR's performance. While all the learning-based algorithms (MARLTC and LP-MAB) achieved a higher final PDR than ADR in networks with placement areas up to 1000 m², their performance declined in scenarios with larger spatial areas of 2000 m². This is because the models were trained in a different spatial environment of 1000 m², where the highest SFs (e.g., SF12) provided little benefit relative

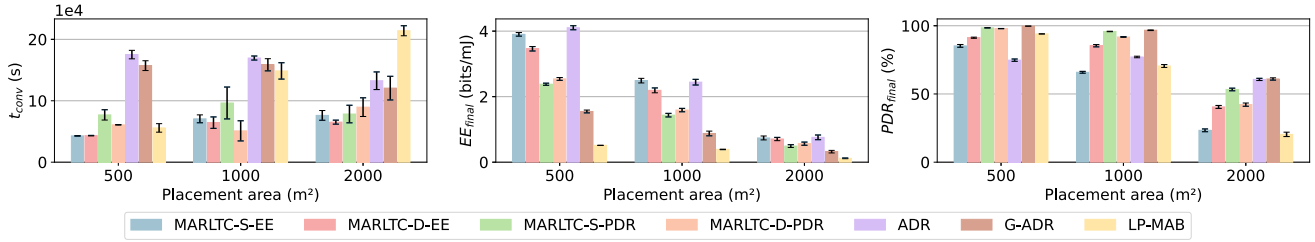


Fig. 6. Network performance for 500 end devices uniformly distributed within square areas of varying size.

TABLE V

RELATIVE PERFORMANCE METRICS COMPARED TO THE DEFAULT ADR MECHANISM FOR DIFFERENT PLACEMENT AREAS (IN SQUARE METERS). NEGATIVE VALUES INDICATE REDUCTIONS, AND POSITIVE VALUES INDICATE IMPROVEMENTS RELATIVE TO ADR

Metric	t_{conv} (% difference)				EE_{final} (% difference)				PDR_{final} (% difference)			
Placement area (m^2)	500	1000	2000	Mean	500	1000	2000	Mean	500	1000	2000	Mean
Algorithm												
MARLTC-S-EE	-75.3	-58.3	-41.6	-58.4	-5.3	+2.0	-1.4	-1.6	+14.0	-13.9	-61.2	-20.3
MARLTC-D-EE	-75.2	-57.5	-40.2	-57.7	-15.8	-9.7	-6.9	-10.8	+21.4	+9.6	-32.8	-0.6
MARLTC-S-PDR	-59.7	-50.8	-38.6	-49.7	-42.6	-41.2	-35.3	-39.7	+31.1	+24.1	-11.4	+14.6
MARLTC-D-PDR	-62.7	-62.3	-24.9	-49.9	-37.7	-34.5	-25.9	-32.7	+30.4	+18.8	-29.6	+6.5
G-ADR	-10.4	-9.2	-7.6	-9.1	-62.0	-63.5	-58.5	-61.3	+32.8	+25.3	+1.2	+19.8
LP-MAB	-68.2	-12.4	+61.4	-6.4	-87.5	-84.0	-84.5	-85.3	+25.7	-8.6	-66.4	-16.4

to their increased ToA, interference, and energy cost. As a result, the agent pruned these actions to maximize its reward, especially for MARLTC-S-EE and LP-MAB, which optimizes EE without enforcing any PDR requirement. When deployed in larger areas without retraining, this learned preference persists, limiting coverage for distant end devices and causing the observed drop in PDR.

Fig. 7 provides further insights into the assigned transmission parameter settings of MARLTC. Fig. 7 (left) presents the distribution of assigned SF configurations as a function of the distance to the nearest gateway, estimated using Gaussian kernel density estimation. The background shading in gray indicates the corresponding average PDR of end devices across the distance range. Fig. 7 (right) illustrates the distribution of the SF settings in networks with a placement area of 2000 m^2 . We observe that ADR [Fig. 7(c)] used all the available SF settings, while MARLTC [Fig. 7(a) and (b)] avoided using the highest SF setting, which would have resulted in a longer communication range and better PDR for the end devices further away from the closest gateway. This led to the drop of the final PDR observed in network variants with larger placement areas (refer to Fig. 6). This reflects the need for training environments that resemble the target deployment in terms of area. In future work, we will explore the development of more general policies trained across randomized area sizes and channel conditions to enhance spatial generalization.

Takeaway: MARLTC generalizes well to unseen end-device placements, consistently reducing convergence times across all spatial distributions. The performance decrease in larger areas, caused by the natural underutilization of high SFs during training, reflects the need for training environments to resemble the target deployment in terms of area.

D. Dynamic End Device Turnover

To analyze the importance and impact of the convergence time of MARLTC and competitor algorithms, we evaluate

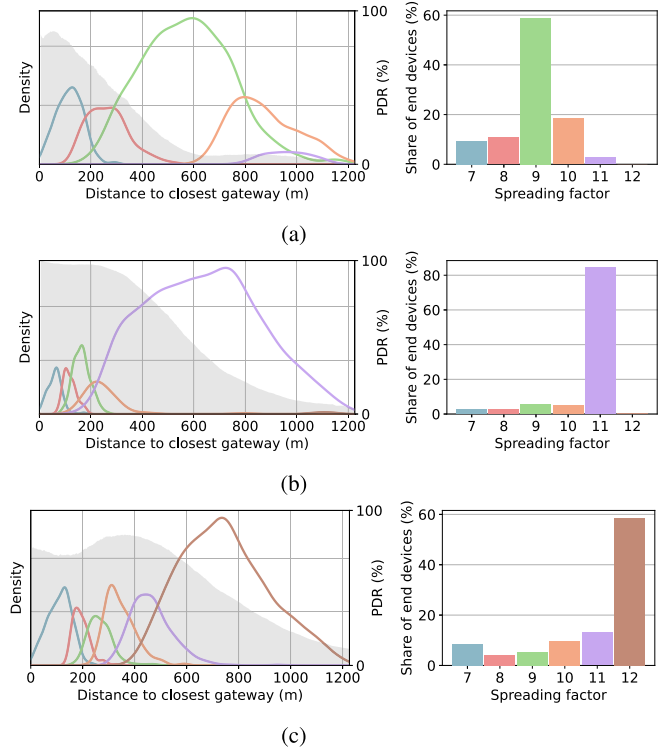


Fig. 7. SF distributions and PDR over distance to closest gateway in a 2000- m^2 network. (a) MARLTC-S-EE. (b) MARLTC-S-PDR. (c) ADR.

their performance in dynamic network conditions introduced through periodic device turnover. In this scenario, 10%, 20%, or 50% of the initial $n = 200$ end devices are randomly replaced every 10 h throughout the simulation, emulating real-world deployments where devices may frequently join or leave the network, challenging the adaptability of the algorithms.

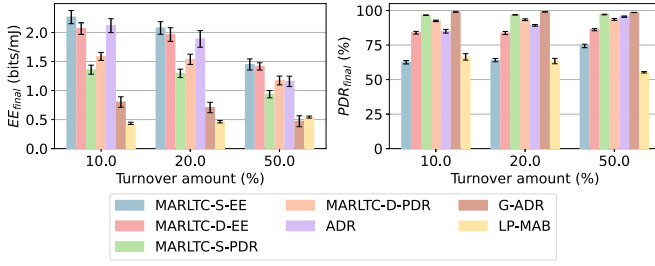


Fig. 8. Network performance under different device turnover rates. For $n = 200$ end devices, 10%, 20%, or 50% are randomly replaced every 10 h to evaluate algorithm adaptability in dynamic network conditions.

TABLE VI

RELATIVE DIFFERENCES IN EE_{final} AND PDR_{final} COMPARED TO THE DEFAULT ADR BASELINE FOR VARYING TURNOVER AMOUNTS. POSITIVE VALUES INDICATE IMPROVEMENTS RELATIVE TO ADR, AND NEGATIVE VALUES INDICATE REDUCTIONS

Metric	EE_{final} (% difference)				PDR_{final} (% difference)			
	10%	20%	50%	Mean	10%	20%	50%	Mean
Algorithm								
MARLTC-S-EE	+6.7	+10.2	+25.5	+14.1	-26.1	-27.6	-21.9	-25.2
MARLTC-D-EE	-2.3	+4.9	+22.3	+8.3	-1.2	-6.2	-9.7	-5.7
MARLTC-S-PDR	-35.4	-30.7	-19.2	-28.4	+13.6	+8.6	+1.8	+8.0
MARLTC-D-PDR	-26.0	-19.7	+1.2	-14.8	+8.6	+5.0	-1.9	+3.9
G-ADR	-62.1	-62.5	-59.3	-61.3	+16.4	+11.0	+3.2	+10.2
LP-MAB	-79.4	-75.4	-53.1	-69.3	-21.9	-28.9	-42.1	-31.0

Fig. 8 shows the EE and PDR across all simulations, while Table VI reports differences relative to ADR as the baseline. Due to the high variability in the final EE and PDR caused by the periodic replacement of end devices, the convergence time defined in 19 cannot be reliably measured and is therefore intentionally omitted from Fig. 8. The results show that the policy trained for static prioritization of EE achieved, on average, a 14.1% higher final EE than ADR, with an increasing lead at higher portions of end devices replaced (6.7% EE with 10.2% of end devices, 25.5% with 50% of end devices). All MARLTC variants' final EE was less affected by an increasing portion of replaced end devices than ADR, G-ADR, and LP-MAB. The final PDR remained independent of replaced end devices or slightly increased with a growing portion of replaced end devices for all MARLTC variants, compared to a clear increase of the final PDR for ADR and a minimal decrease of the final PDR for G-ADR and LP-MAB. In network variants with 50% of end devices regularly replaced, the MARLTC variant using a static, non-adjustable policy trained to prioritize the PDR obtained the highest final PDR among all MARLTC variants. However, its performance remained, on average, 2.2% below that of G-ADR, which achieved the overall highest final PDR.

Fig. 9 provides insights into the average EE and PDR over time of networks with periodical end device replacement. The effects of fast convergence times of most MARLTC variants observed in earlier scenarios become clearly visible in Fig. 9. Algorithms that assign transmission parameters so that significant EE gains are achieved during early optimization steps tend to maintain higher overall EE under periodic end-device turnover. This effect becomes more pronounced as the

proportion of replaced devices increases [see Fig. 9(a) and (b)]. Most MARLTC variants demonstrated strong performance under these conditions.

Takeaway: MARLTC outperformed the ADR baselines in dynamic networks with device turnover, primarily due to its reduced convergence times.

E. Generalization to Varying Gateway Density and Topology

To complement the analysis in Section VI-C, where we evaluated MARLTC under varying spatial distributions of end devices, we now assess its generalization capability beyond the training scenario by varying the number of gateways and the intergateway radius. This evaluation considers deployment conditions not seen during training and represents a meaningful stress test of the learned ADR policies under infrastructure and topology changes. We simulated networks of $n = 100$ devices for $T = 9$ h per run, averaging results over ten independent runs. The number of gateways was varied in $m = \{1, 4, 8, 16, 32\}$ and the intergateway radius in $r = \{50, 150, 250, 350, 500\}$ m. We recall that the model was trained with four gateways and a 250-m radius. For each configuration, we computed the average EE and PDR.

As shown in Fig. 10, EE increases monotonically with gateway density across all radii, reaching up to 2.02 bits/mJ at 32 gateways and 350 m, while PDR remains consistently high (up to 0.85). The higher gateway density enables the use of lower SFs and transmission power levels, which reduces ToA and transmit energy, thereby improving EE while maintaining high reliability. Performance near the training configuration is comparable across neighboring radii, indicating that the learned policy does not overfit to a specific spatial scale. Even under substantial infrastructure changes, PDR remains above 0.78 in most multigateway scenarios, while EE scales proportionally with the available infrastructure. In the extreme single-gateway case, both EE and PDR decrease uniformly across radii, reflecting inherent network constraints rather than instability of MARLTC.

Takeaway: MARLTC generalizes effectively to varying gateway densities and spatial layouts. It maintains reliable packet delivery while adapting EE to the available infrastructure, confirming robustness beyond the training scenario.

VII. DEPLOYMENT FEASIBILITY

In this section, we discuss the practical deployability of MARLTC in physical LoRaWAN environments, focusing on protocol compliance, system architecture, and embedded execution constraints under realistic operational conditions.

As described in Section IV, MARLTC is designed to operate within the constraints of the LoRaWAN specification and does not require modifications to the MAC protocol. It embeds the aggregated network state and SNR history into standard downlink messages. The downlink payload for the observation tuple fits within the maximum payload size defined by the LoRaWAN regional parameters for the considered data rates. The periodicity of the updates is aligned with the default ADR backoff mechanism, thereby ensuring that control traffic remains sparse and predictable while respecting duty-cycle constraints. Moreover, the CTDE architecture is naturally compatible with LoRaWAN's star-of-stars topology, as it preserves

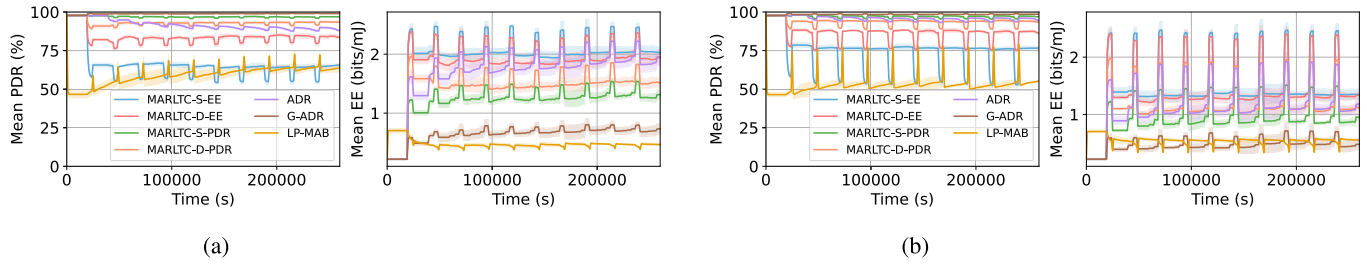


Fig. 9. Network performance over time for different turnover rates, illustrating the effect of replacing different fractions of end devices over time on algorithm adaptability and convergence. (a) 20% turnover amount. (b) 50% turnover amount.

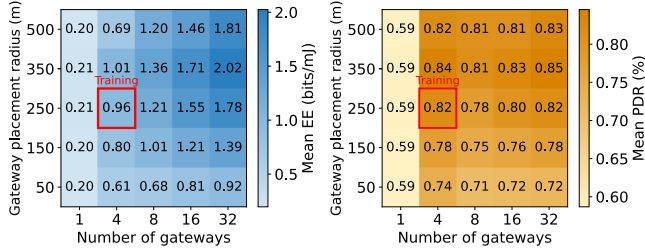


Fig. 10. Heatmaps of EE (left) and PDR (right) for MARLTC variants across different numbers of gateways (1–32) and intergateway radii (50–500 m). The training configuration (4 gateways, 250 m) is highlighted in red.

a strict separation between infrastructure-level communication and the learning process. The training workload, including experience buffering, policy optimization via Ray RLlib, and hyperparameter tuning, is assumed to be executed at the NS. Typically hosted on cloud or edge infrastructure, this component supports continuous MARL training and aggregation without affecting packet routing, while remaining independent of gateway computation and firmware constraints. In this work, we assume models are flashed on end devices; while our evaluation indicates that the trained models are sufficiently general to handle varied conditions, implementing bandwidth-efficient remote retraining and redeployment remains an interesting direction for future work.

The decentralized execution of the trained policy on end devices makes the transition of MARLTC from simulation to physical LoRaWAN deployments constrained by the limited memory, compute capacity, and energy budget of LoRaWAN end devices. While Ray RLlib provides scalability and convenience during training, the resulting PyTorch models may not be directly suitable for resource-constrained host microcontrollers in LoRaWAN end devices. To address this, MARLTC exports trained models to the open neural network exchange (ONNX) format, a standard intermediate representation for neural networks that enables interoperability across frameworks. It then uses the ONNX Runtime [63] to perform model quantization and subsequently compiles the resulting model into dependency-free C code using the `onnx2c` library [64]. This translates ONNX computation graphs into standardized and platform-agnostic C source code that serves as a flexible template for embedded deployment, facilitating final compilation for specific target platforms.

To verify the practical feasibility of MARLTC under embedded constraints, we evaluate the model size, inference latency, and decision parity across the model conversion pipeline. By measuring the agreement between the original PyTorch

TABLE VII
PERFORMANCE COMPARISON OF MARL POLICIES ACROSS PYTORCH, ONNX, AND C-COMPILED VARIANTS. LATENCY IS REPORTED IN μs AND DECISION PARITY IN %

Model	Variant	Size (KB)	Latency (μs)	Parity (%)
P1	PyTorch RayRLlib	801.6	352.8	100.00
	ONNX FP32	712.6	31.6	100.00
	ONNX INT8	192.0	16.1	100.00
	C-Compiled INT8	188.0	352.0	100.00
P2	PyTorch RayRLlib	775.7	160.9	100.00
	ONNX FP32	712.6	18.7	100.00
	ONNX INT8	192.0	13.6	93.07
	C-Compiled INT8	188.0	351.1	93.07
P3	PyTorch RayRLlib	767.9	230.7	100.00
	ONNX FP32	712.6	23.1	100.00
	ONNX INT8	192.0	15.2	99.84
	C-Compiled INT8	188.0	351.6	99.84

model and its C-compiled counterparts, we ensure that the optimization process preserves the integrity of the control logic. For this purpose, we deploy the compiled models on a small-scale LoRaWAN testbed comprising Waveshare SX1262 LoRa HATs attached to Raspberry Pi 5 nodes and an SX1302-based gateway. Although more powerful than typical microcontrollers, this setup validates that the inference code can execute concurrently with a standard LoRaWAN protocol stack without interference. This experiment, therefore, serves as a functional validation of deployability rather than a strict lower bound evaluation of resource usage. While the end-to-end orchestration of MARL policies across massive fleets remains a distinct systems-engineering challenge, our results provide empirical evidence that the underlying policy execution is fully compatible with LoRaWAN hardware constraints. By isolating and validating this primary computational bottleneck, these findings establish the necessary foundation for future large-scale, field-integrated adaptive control. To support reproducibility, all trained and compiled models are released as open-source artifacts.

The results reported in Table VII confirm the effectiveness of the model compression pipeline. High decision parity (up to 100%) across variants proves that quantization maintains control logic integrity while enabling deployment on resource-constrained hardware. The C-compiled INT8 models achieve a binary footprint of 188.0 kB, fitting comfortably within the Flash memory limits of mid-range industrial microcontrollers. They also present a 76% reduction from the PyTorch original

and a $4\times$ reduction compared to FP32 variants. While the bare-metal C latency increases to approximately $350\ \mu\text{s}$, this is primarily due to the transition from the highly optimized, hardware-accelerated kernels of the ONNX Runtime to the generic execution of standard C code. Regardless, this execution time remains negligible compared to typical LoRaWAN ToA or interpacket intervals.

VIII. CONCLUSION

This article presented MARLTC, a MARL-based ADR framework for dense LoRaWAN networks that jointly optimizes transmission power and the SF. MARLTC explicitly accounts for co-channel interference and supports reconfigurable tradeoffs between EE and reliability. MARLTC formulates transmission parameter selection as a cooperative Markov game, enabling coordinated learning of transmission strategies that adapt to interference and dynamic channel conditions while remaining scalable in large-scale networks. Multiple policy variants are trained to meet diverse application requirements, maximizing EE, maximizing the PDR, or balancing both, allowing for tailored performance across varied deployment scenarios. MARLTC is validated using FAST-LoRa, a realistic open-source simulator that incorporates detailed physical-layer effects, including co-SF and inter-SF interference, demonstrating the practicality and effectiveness of the approach. Evaluations across static and dynamic conditions demonstrated that MARLTC consistently outperforms state-of-the-art ADR mechanisms. Convergence time was reduced by up to 67.9%, with comparable or improved final EE and PDR. In dense deployments with 1000 end devices, the static PDR-prioritized policy achieved 62.1% higher EE and a 5.0% higher PDR than G-ADR. Under periodic end-device turnover (up to 50%), MARLTC retained a 27.4% EE advantage over ADR, highlighting its responsiveness to dynamic topologies. To bridge the gap between simulation and deployment, we developed a model-conversion pipeline that translates trained policies into dependency-free, standalone C implementations. Experimental validation on a physical LoRaWAN testbed confirmed that MARLTC policies achieve a 76% reduction in memory footprint and negligible inference latency, satisfying the strict resource and timing constraints of IoT hardware. These results demonstrate that MARLTC constitutes a practical, scalable, and readily integrable enhancement to existing ADR mechanisms in LoRaWAN networks. In future work, we will improve generalization to unseen deployment conditions by training general policies across randomized area sizes to better handle varying spatial domains and extend mobility support. We also plan to explore interference and reliability in large-scale real-world testbeds, providing complementary insights to the simulation results.

APPENDIX

We conducted supplementary simulations using a bursty traffic model with 200 end devices over 9 h, where per-interval transmission probabilities were scaled by dynamic burst factors sampled uniformly between 1 and 10. For each packet interval, the effective transmission rate was $\lambda_{ij} = \lambda \times \text{burst_factor}$, with $\lambda = 1/\text{packet interval}$. Overlapping

TABLE VIII
PDR UNDER BURSTY (B) AND POISSON (P) TRAFFIC FOR DIFFERENT AVERAGE TRANSMISSION INTERVALS

Avg. Transmission Interval (s)	PDR (B) / (P)
250	0.52 / 0.78
500	0.72 / 0.81
1000	0.84 / 0.85
2000	0.95 / 0.99

transmissions were then modeled as $h_{ij} = 1 - e^{-\lambda_{ij}T'_{ij}}$, where T'_{ij} denotes the interference interval between end devices i and j . Table VIII reports the PDR under Poisson and bursty traffic, averaged across the four MARLTC variants. For both traffic models, higher traffic intensity resulting from shorter packet intervals increases the number of transmitted packets and slightly reduces PDR, while MARLTC consistently maintains stable adaptation dynamics across all variants.

REFERENCES

- [1] J. P. S. Sundaram, W. Du, and Z. Zhao, "A survey on LoRa networking: Research problems, current solutions, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 371–388, 1st Quart., 2020.
- [2] P. Gkotsiopoulos, D. Zorbas, and C. Douligeris, "Performance determinants in LoRa networks: A literature review," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1721–1758, 3rd Quart., 2021.
- [3] S. Devalal and A. Karthikeyan, "LoRa technology—An overview," in *Proc. 2nd Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA)*, Mar. 2018, pp. 284–290.
- [4] H. Bougaddou, N. Cherkaoui, and O. Mouhib, "Long-range, low-power for IoT devices: The LoRa network a review," in *Proc. Medit. Smart Cities Conf. (MSSC)*, May 2024, pp. 1–6.
- [5] L. Aldhaheri et al., "LoRa communication for agriculture 4.0: Opportunities, challenges, and future directions," *IEEE Internet Things J.*, vol. 12, no. 2, pp. 1380–1407, Jan. 2025.
- [6] S. Wielandt, S. Uhlemann, S. Fiolleau, and B. Dafflon, "TDD LoRa and delta encoding in low-power networks of environmental sensor arrays for temperature and deformation monitoring," *J. Signal Process. Syst.*, vol. 95, no. 7, pp. 831–843, Jul. 2023.
- [7] B. Cengiz, I. Y. Adam, M. Ozdem, and R. Das, "A survey on data fusion approaches in IoT-based smart cities: Smart applications, taxonomies, challenges, and future research directions," *Inf. Fusion*, vol. 121, Sep. 2025, Art. no. 103102.
- [8] G. D. Costa et al., "Exascale machines require new programming paradigms and runtimes," *Supercomput. Frontiers Innov.*, vol. 2, no. 2, pp. 6–27, 2015.
- [9] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, Nov. 2018.
- [10] V. Moysiadi, T. Lagkas, V. Argyriou, A. Sarigiannidis, I. D. Moscholios, and P. Sarigiannidis, "Extending ADR mechanism for LoRa enabled mobile end-devices," *Simul. Model. Pract. Theory*, vol. 113, Dec. 2021, Art. no. 102388.
- [11] S. Li, U. Raza, and A. Khan, "How agile is the adaptive data rate mechanism of LoRaWAN?," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 206–212.
- [12] N. Benkahla, H. Tounsi, S. Ye-Qiong, and M. Frikha, "Enhanced ADR for LoRaWAN networks with mobility," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2019, pp. 1–6.
- [13] A. Farhad, D.-H. Kim, S. Subedi, and J.-Y. Pyun, "Enhanced LoRaWAN adaptive data rate for mobile Internet of Things devices," *Sensors*, vol. 20, no. 22, p. 6466, Nov. 2020.
- [14] K. Anwar, T. Rahman, A. Zeb, I. Khan, M. Zareei, and C. Vargass-Rosales, "RM-ADR: Resource management adaptive data rate for mobile application in LoRaWAN," *Sensors*, vol. 21, no. 23, p. 7980, Nov. 2021.
- [15] C. Lehong, B. Isong, F. Lugayizi, and A. Abu-Mahfouz, "A spreading factor congestion status-aware adaptive data rate algorithm," *J. Sensor Actuator Netw.*, vol. 10, no. 4, p. 70, Dec. 2021.
- [16] S. Kim and Y. Yoo, "Contention-aware adaptive data rate for throughput optimization in LoRaWAN," *Sensors*, vol. 18, no. 6, p. 1716, May 2018.

- [17] N. El Rachkidy, A. Guitton, and M. Kaneko, "Collision resolution protocol for delay and energy efficient LoRa networks," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 535–551, Jun. 2019.
- [18] A. Farhad, D.-H. Kim, P. Sthapit, and J.-Y. Pyun, "Interference-aware spreading factor assignment scheme for the massive LoRaWAN network," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2019, pp. 1–2.
- [19] L. Acosta-Garcia et al., "CADD: Collaborative adaptive duty cycle and data reduction for congestion control in LoRaWAN," *Internet Things*, vol. 37, May 2026, Art. no. 101904. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S254266052600034X>
- [20] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, Sep. 2016.
- [21] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–9.
- [22] A. Durand, N. El Rachkidy, and A. Guitton, "MADERE: Mobile adaptive datarate for LoRaWAN," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2023, pp. 1–6.
- [23] L. Acosta-Garcia, J. Aznar-Poveda, A.-J. Garcia-Sanchez, J. Garcia-Haro, and T. Fahringer, "Proactive adaptation of data rate in mobile LoRa-based IoT devices using machine learning," in *Proc. IEEE 99th Veh. Technol. Conf. (VTC-Spring)*, Jun. 2024, pp. 1–5.
- [24] B. Robbe and W. Danny, "A QoS-aware adaptive mobility handling approach for LoRa-based IoT systems," in *Proc. IEEE 12th Int. Conf. Self-Adapt. Self-Organizing Syst. (SASO)*, Sep. 2018, pp. 130–139.
- [25] L. Acosta-Garcia, J. Aznar-Poveda, A. J. Garcia-Sanchez, J. Garcia-Haro, and T. Fahringer, "Dynamic transmission policy for enhancing LoRa network performance: A deep reinforcement learning approach," *Internet Things*, vol. 24, Dec. 2023, Art. no. 100974.
- [26] L. Acosta-Garcia, J. Aznar-Poveda, A. J. Garcia-Sanchez, J. Hollenstein, J. Garcia-Haro, and T. Fahringer, "ADRL: A reconfigurable energy-efficient transmission policy for mobile LoRa devices based on reinforcement learning," *Internet Things*, vol. 31, May 2025, Art. no. 101577.
- [27] D.-T. Ta, K. Khawam, S. Lahoud, C. Adjih, and S. Martin, "LoRa-MAB: Toward an intelligent resource allocation approach for LoRaWAN," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
- [28] B. Teymuri, R. Serati, N. A. Anagnostopoulos, and M. Rasti, "LP-MAB: Improving the energy efficiency of LoRaWAN using a reinforcement-learning-based adaptive configuration algorithm," *Sensors*, vol. 23, no. 4, p. 2363, Feb. 2023.
- [29] M. Jouhari, K. Ibrahim, J. Ben Othman, and E. M. Amhoud, "Deep reinforcement learning-based energy efficiency optimization for flying LoRa gateways," 2023, *arXiv:2302.05214*.
- [30] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2nd Quart., 2021.
- [31] A. Ivoghlian, Z. Salic, and K. I.-K. Wang, "Adaptive wireless network management with multi-agent reinforcement learning," *Sensors*, vol. 22, no. 3, p. 1019, Jan. 2022.
- [32] G. Zhao, K. Lin, D. Chapman, N. Metje, and T. Hao, "Optimizing energy efficiency of LoRaWAN-based wireless underground sensor networks: A multi-agent reinforcement learning approach," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100776.
- [33] X. Zhang, Z. Lin, S. Gong, B. Gu, and D. Niyato, "Multiagent reinforcement learning with an attention mechanism for improving energy efficiency in LoRa networks," in *Proc. GLOBECOM - IEEE Global Commun. Conf.*, Dec. 2023, pp. 4152–4157.
- [34] L. Acosta-Garcia et al., "FAST-LoRa: An efficient simulation framework for evaluating LoRaWAN networks and transmission parameter strategies," *IEEE Access*, vol. 14, pp. 33518–33533, 2026.
- [35] T. Fahringer, "JavaSymphony: A system for development of locality-oriented distributed and parallel Java applications," in *Proc. IEEE Int. Conf. Cluster Computing. CLUSTER*, Saxony, Germany, Nov. 2000, pp. 145–152.
- [36] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations," in *Proc. IEEE 13th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2017, pp. 1–8.
- [37] A. Farhad, D.-H. Kim, D. Kwon, and J.-Y. Pyun, "An improved adaptive data rate for LoRaWAN networks," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCE-Asia)*, Nov. 2020, pp. 1–4.
- [38] A. Farhad, D.-H. Kim, J.-S. Yoon, and J.-Y. Pyun, "Feasibility study of the LoRaWAN blind adaptive data rate," in *Proc. 12th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Aug. 2021, pp. 67–69.
- [39] A. Farhad and J.-Y. Pyun, "HADR: A hybrid adaptive data rate in LoRaWAN for Internet of Things," *ICT Exp.*, vol. 8, no. 2, pp. 283–289, Jun. 2022.
- [40] J. Aznar-Poveda, E. Egea-López, and A.-J. García-Sánchez, "Cooperative awareness message dissemination in EN 302 637-2: An adaptation for winding roads," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, vol. 53, Jul. 2020, pp. 1–5.
- [41] J. Aznar-Poveda, E. Egea-Lopez, A.-J. Garcia-Sanchez, and J. Garcia-Haro, "Advisory speed estimation for an improved V2X communications awareness in winding roads," in *Proc. 22nd Int. Conf. Transparent Opt. Netw. (ICTON)*, Jul. 2020, pp. 1–4.
- [42] W. Gao, W. Du, Z. Zhao, G. Min, and M. Singhal, "Towards energy-fairness in LoRa networks," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 788–798.
- [43] T. T. Network. (2024). *RSSI and SNR*. [Online]. Available: <https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/>
- [44] M. Bor, U. Roedig, T. Voigt, and J. Alonso, "Do LoRa low-power wide-area networks scale?," in *Proc. 19th ACM Int. Conf. Model.*, 2016, pp. 59–67.
- [45] V. Toro-Betancur, G. Premsankar, M. Slabicki, and M. Di Francesco, "Modeling communication reliability in LoRa networks with device-level accuracy," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, France, May 2021, pp. 1–10.
- [46] M. Slabicki. (2022). *Flora—A Framework for LoRa Simulations With OMNet++*. [Online]. Available: <https://flora.aalto.fi/>
- [47] M. Haenggi and R. K. Ganti, "Interference in large wireless networks," *Found. Trends Netw.*, vol. 3, no. 2, pp. 127–248, Nov. 2009.
- [48] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondón, S. A. Hassan, and M. Gidlund, "Scalability analysis of a LoRa network under imperfect orthogonality," *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1425–1436, Mar. 2019.
- [49] L. Amichi, M. Kaneko, E. H. Fukuda, N. El Rachkidy, and A. Guitton, "Joint allocation strategies of power and spreading factors with imperfect orthogonality in LoRa networks," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3750–3765, Jun. 2020.
- [50] Z. Xu et al., "X-MAC: Achieving high scalability via imperfect-orthogonality aware scheduling in LPWAN," in *Proc. IEEE 30th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2022, pp. 1–11.
- [51] R. Lowe, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6379–6390.
- [52] M. Andrychowicz et al., "What matters for on-policy deep actor-critic methods? A large-scale study," in *Proc. Int. Conf. Learn. Represent.*, 2021.
- [53] Semtech.(2023). *Implementing Adaptive Data Rate*. [Online]. Available: <https://learn.semtech.com/mod/book/view.php?id=174&chapterid=162>
- [54] J. Aznar-Poveda. (2025). *MARLTC: Multi-agent Reinforcement Learning for LoRa Transmission Control*. [Online]. Available: <https://github.com/juanaznarp94/MARLTC>
- [55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [56] M. Jaderberg et al., "Population based training of neural networks," 2017, *arXiv:1711.09846*.
- [57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [58] M. Fragkopoulou, S. Panagiotakis, M. Kostakis, E. K. Markakis, N. Astyrakakis, and A. Malamos, "Experimental assessment of common crucial factors that affect LoRaWAN performance on sub-urban and rural area deployments," *Sensors*, vol. 23, no. 3, p. 1316, Jan. 2023.
- [59] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in ns-3," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2186–2198, Dec. 2017.
- [60] Semtech.(2024). *LoRaWAN—Simple Rate Adaptation Recommended Algorithm*. Accessed: 02. [Online]. Available: <https://www.thethingsnetwork.org/forum/uploads/default/original/2X/7/7480e044aa93a54910dab8ef0adfb5f515d14a1.pdf>
- [61] L. Alliance. (2022). *Rp002-1.0.4 Regional Parameters*. [Online]. Available: <https://resources.lora-alliance.org/technical-specifications/rp002-1-0-4-regional-parameters>
- [62] R. K. Jain et al., "A quantitative measure of fairness and discrimination," Eastern Res. Lab., Digit. Equip. Corp., Hudson, MA, USA, Tech. Rep. DEC-TR-301, 1984, vol. 21, no. 1, pp. 2022–2023.
- [63] O. R. Developers. (2021). *ONNX Runtime*. [Online]. Available: <https://onnxruntime.ai/>
- [64] Kraiskil.(2024). *ONNX2C: An ONNX to C Compiler*. Accessed: May 4, 2026. [Online]. Available: <https://github.com/kraiskil/onnx2c>



Juan Aznar-Poveda received the Ph.D. degree in telecommunications engineering from the Technical University of Cartagena, Cartagena, Spain, in 2022.

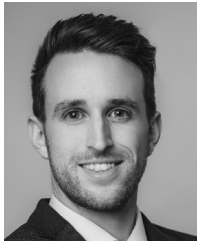
He is currently a Post-Doctoral Researcher with the Distributed and Parallel Systems Group, University of Innsbruck, Innsbruck, Austria. His research interests include distributed systems, cyber-physical systems (CPSs), the Internet of Things, artificial intelligence, and wireless networks and communications.

Dr. Aznar-Poveda was awarded the Prize “Liberalization of Telecommunications” (National Level) for the Best B.Sc. Thesis in 2016.



Laura Acosta-Garcia received the B.Sc. and M.Sc. degrees in telecommunications engineering from the Technical University of Cartagena (UPCT), Cartagena, Spain, in 2021 and 2023, respectively, where she is currently pursuing the Ph.D. degree.

In 2022, she joined the Department of Information and Communication Technologies, UPCT. She was awarded a Research Activity Initiation Scholarship by Santander Bank. Her research interests include wireless networks, the Internet of Things (IoT), machine learning, and reinforcement learning.



Fabian Margreiter received the B.Sc. degree in computer science from the University of Innsbruck, Innsbruck, Austria, in 2025.

He conducts research with the University of Innsbruck (UIBK). Since 2013, he has led the Software and Electronics Engineering Department, Rothlehner Arbeitsbühnen GmbH, Massing, Germany, a German construction machinery manufacturer. In 2022, he joined BOC Group, Vienna, Austria, where he is currently a Cloud and DevOps Engineer. His research interests include distributed systems, cloud

computing, and artificial intelligence.



Marlon Etheredge received the M.Sc. degree in artificial intelligence from Utrecht University, Utrecht, The Netherlands, in 2016. He is currently pursuing the Ph.D. degree with the Distributed and Parallel Systems Group, University of Innsbruck, Innsbruck, Austria.

His main research interests include programming paradigms and runtime systems in the cloud-edge-IoT continuum.



Abolfazl Younesi (Student Member, IEEE) received the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2024.

He is currently a Researcher with the University of Innsbruck, Innsbruck, Austria, affiliated with the Distributed and Parallel Systems Group (DPS), Institute of Computer Science. His work is supported by the European Union’s NextGenerationEU initiative. His research spans distributed intelligence, federated learning, stream processing, cyber-physical systems

(CPSs), the IoT, and the computing continuum, with a particular focus on intelligent resource management and compute optimization.

Mr. Younesi serves as a reviewer for IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and *Future Generation Computer Systems* (FGCS), as well as for international conferences including IEEE QCE, CCGrid, IC FEC, DSD, and HPDC.



Stefan Pedratscher received the M.Sc. degree in computer science from the University of Innsbruck, Innsbruck, Austria, in 2021, where he is currently pursuing the Ph.D. degree with the Distributed and Parallel Systems Group.

His main research interests include serverless computing and stream processing in the edge-cloud continuum.



Joan Garcia-Haro (Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1989 and 1995, respectively.

He was a Visiting Scholar with Queen’s University, Kingston, ON, Canada, from 1991 to 1992, and Cornell University, Ithaca, NY, USA, from 2010 to 2011. He is currently a Full Professor with the Technical University of Cartagena (UPCT), Cartagena, Spain. He is the author or co-author of more than

70 journal articles, mainly in the fields of switching, wireless networking, and performance evaluation.

Dr. Garcia-Haro received the Honorable Mention for the IEEE Communications Society Best Tutorial Paper Award in 1995. He served as the Editor-in-Chief for the IEEE Global Communications Newsletter, including in *IEEE Communications Magazine*, from April 2002 to December 2004. From March 2001 to December 2011, he was the Technical Editor of *IEEE Communications Magazine*.



Thomas Fahringer (Member, IEEE) received the Ph.D. degree from Vienna University of Technology, Vienna, Austria, in 1993.

Since 2003, he has been a Full Professor of computer science with the Institute of Computer Science, University of Innsbruck, Innsbruck, Austria. His main research interests include software architectures, programming paradigms, compiler technology, performance analysis, and prediction for parallel and distributed systems.



Antonio-Javier Garcia-Sanchez received the Ph.D. degree in industrial engineering from the Technical University of Cartagena, Cartagena, Spain, in 2005.

He is currently a Full Professor of telecommunications engineering with the Technical University of Cartagena. He has been the co-author of more than 100 conference papers and journal articles, the Head of several EU research projects in the field of communication networks and optimization, and an inventor/co-inventor of 12 patents or utility models.

His main research interests include wireless sensor networks (WSNs), streaming services, artificial intelligence, the Internet of Things (IoT), and nanonetworks.

Dr. Garcia-Sanchez has been a TPC member or the chair at about 40 international congresses or workshops. He is also a reviewer of several journals listed in the ISI-JCR.