Energy Aware Scheduling using Reinforcement Learning for 802.15.4e Time-Slotted Channel Hopping

by

Tarana Ara

A thesis submitted to the

School of Graduate and Postdoctoral Studies in partial

fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Faculty of Engineering and Applied Science

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

November 2023

Thesis Examination Information

Submitted by: **Tarana Ara**

Master of Applied Science in **Electrical and Computer Engineering**

Thesis Title: Energy Aware Scheduling using Reinforcement Learning for 802.15.4e Time-Slotted Channel Hopping

An oral defense of this thesis took place on October 25th, 2023, in front of the following examining committee:

Examining Committee:

Chair of Examining Committee: **Dr. Khalid Elgazzar**

Research Supervisor: **Dr. Ramiro Liscano**

Examining Committee Member: **Dr. Shahram S. Heydari**

Thesis Examiner: **Dr. Richard W. Pazzi, Associate Professor, Ontario Tech University**

The above committee determined that the thesis is acceptable in form and content and that the candidate demonstrated satisfactory knowledge of the field covered by the thesis during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Time Slotted Channel Hopping (TSCH) is a medium access control mode defined in IEEE 802.15.4e standard. This protocol is a crucial component of fourth-generation IoT applications, enabling efficient communication through synchronized time slots and channel hopping. By employing TSCH, IoT devices can achieve improved reliability, reduced interference, and increased network capacity. However, the energy consumption of IoT devices remains a significant challenge in large-scale deployments.

This research introduces an energy-aware (EARL) schedule based on Reinforcement Learning (RL) for the 802.15.4e Time-Slotted Channel Hopping (TSCH) mode. The goal is to turn off slots in the 802.15.4e TSCH frame that are not highly utilized so as to conserve energy. By considering a predefined threshold, each node determines the slots that should be deactivated. This adaptive scheduling strategy allows the nodes to conserve energy effectively by minimizing unnecessary radio operations.

Through extensive simulations and evaluations with simple and large-scale network configurations, the proposed energy-aware TSCH scheduling algorithm using Q-learning demonstrates promising results and is compared with the Orchestra protocol. This innovative approach reveals superior performance compared to Orchestra, achieving notably improved outcomes in both packet delivery rate and energy savings.

Keywords— Time-Slotted Channel Hopping; Reinforcement Learning(RL); Q-Learning; TSCH-Sim; Power-Consumption

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_____

Tarana Ara

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Ramiro Liscano, for his unwavering guidance, invaluable insights and support throughout the entire journey of my Master's research work. His expertise, dedication, and mentorship played a big role in shaping my thesis and greatly helped me to finish my degree successfully. The support and helpful feedback I got from him were crucial in refining my research and enhancing the quality of this thesis.

In addition, I would like to thank my husband, Asif M Yousuf, for his support and encouragement throughout the journey of pursuing my degree.

Statement of Contributions

The research topic and the work described in this thesis have been accepted to the following events:

- Ara, T., Singh, T., Vatankhah, A. and Liscano, R., 2022. ”Enhancement of the TSCH-Sim Simulator to Support Manual Scheduling and Routing”,The 17th International Conference on Future Networks and Communications(pp.61-68).(Published in FNC 2022)

- Ara, Tarana, Aida Vatankhaha, and Ramiro Liscano. ”Enhancement of the TSCH-Sim Simulator via Web Service Interface to Support Co-simulation Optimization.”Journal of Ubiquitous Systems and Pervasive Networks.(Published in JUSPN 2023)

I hereby certify that I am the sole author of this thesis. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RL	Reinforcement Learning
TSCH	Time Slotted Channel Hopping
MAC	Medium Access Control
RX	Reception
TX	Transmision
ACK	Acknowledgement
IoT	Internet of Things
WSNs	Wireless Sensor Networks
CH	Channel
ASN	Absolute Slot Number
EB	Enhanced Beacon
UDGM	Unit Disk Graph Model
RPL	Routing Protocol for Low power and lossy Network
PDR	Packet Delivery Ratio
EARL	Energy Aware Reinforcement Learning
TP	Transition Phase

Chapter 1

INTRODUCTION

Industry 4.0 is often referred to as the fourth industrial revolution (IR 4.0) [1] and heavily depends on the utilization of IoT (Internet of Things) and WSNs (Wireless Sensor Networks). IoT, a specialized field of engineering, facilitates the interconnection of numerous small physical objects, enabling them to work together toward common objectives, and gaining significance due to their widespread usage. IoT and WSNs are applied in diverse areas such as environment sensing, traffic tracking, medical systems, and various aspects of daily life [2]. By gathering and processing data from these compact nodes and then transferring it to operators, they play a significant role in driving modernization efforts.

Wireless sensor network (WSN) nodes are typically compact devices, either powered by small batteries or energy-harvesting mechanisms, comprising a microcontroller, a small memory, one or more sensors, and a low-power radio transceiver [3]. Even though those motes are very lightweight and cost-effective, one of the primary limitations is the energy constraint that arises from the challenges of recharging or replacing batteries once the WSN has been deployed.

This finite storage capacity of batteries and the constrained capabilities of energy-harvesting devices significantly influence WSN research, leading to a strong focus on minimizing energy consumption. In order to extend the network's lifespan, it is common to adopt energy-efficient approaches that focus on managing radio activities like sensing and communication while maintaining

a balanced power consumption level. However, scheduling the node's radio operations proves to be the most challenging aspect, as it directly correlates with the Medium Access Control (MAC) layer.

The MAC layer plays a crucial role in coordinating access to the radio channel, exerting a significant impact on the overall network performance. Consequently, it affects energy consumption and important metrics like throughput and latency. A well-optimized MAC protocol could handle the key factors contributing to energy waste, for instance, idle listening, packet collisions, overhearing, and redundant retransmission [4]. Moreover, it must guarantee successful packet reception to their intended destinations in a multi-hop network.

Numerous standardization efforts, such as WirelessHART [5], ISA [6], Bluetooth [7], Zigbee [8], IEEE 802.15.4 [9] have been undertaken to respond to the requirements of diverse industrial IoT applications. Out of these standards, Zigbee is unsuitable for industrial applications due to its deficiencies in determinism, frequency, path diversity, and the unreliability of its medium access control (MAC) [10]. The other two protocols, WirelessHART and ISA, relays on the centralized approach resulting in the inability to reuse resources and consequently leading to scalability challenges.

To overcome the above limitations, the IEEE standard 802.15.4 (in 2012)introduced MAC layer amendments known as IEEE 802.15.4e-2012. It has been evaluated as an extension to the pre-existing IEEE 802.15.4-2011 standard and the most recent version, IEEE 802.15.4-2015 [11]. The IEEE 802.15.4e standard presents a variety of MAC layer functionalities serving diverse application areas within industrial systems. Those MAC behavior modes are Time slotted channel hopping (TSCH), deterministic and synchronous multichannel extension (DSME) as well as low latency deterministic networks (LLDN). Among these, TSCH has attracted considerable worldwide interest due to its remarkable abilities in the field of fourth industrial automation. It has become the most promising technology for future IoT applications by ensuring high reliability, low latency, and energy efficiency.

Even though the standard defines how a TSCH node will communicate with others, it does not define the policies to create and maintain the schedule [12]. Designing an optimal schedule is an

open research in this sector, and to effectively address this issue, several studies have been proposed, including centralized and distributed approach.

With centralized scheduling [13], the schedule is built, maintained, and distributed by a central controller node. This node possesses extensive information about various aspects like network arrangement and routing. This approach is commonly adopted by steady industrial networks where schedule adjustments are infrequent due to the consistent network layout and traffic patterns [14]. Although centralized scheduling can generate highly efficient schedules, adapting to frequent alterations in network characteristics like size scalability and changing application needs can be challenging. Additionally, there might be significant overhead in gathering and disseminating information.

In contrast to the centralized scheduling approach, many studies have opted for the distributed scheduling technique for its potential to address certain limitations. Unlike the centralized approach, in the distributed technique, the schedule is constructed through negotiation between nodes [15]. However, sometimes distributed scheduling faces a challenge known as the visibility issue. Monitoring and troubleshooting a low-power network where decisions are reached through intra-node negotiations can prove to be challenging for the network operator due to the complexity and lack of transparency in the process.

An Autonomous scheduling approach can be an attractive alternative for TSCH MAC scheduling as it overcomes these issues due to its decentralized nature and low complexity features. This research adopts a reinforcement learning-based energy-aware schedule for TSCH networks.

In this methodology, each node independently constructs the schedule, establishing a decentralized system. The decision-making process heavily relies on state-action pairs and reward values, employing a trial-and-error approach. Unlike other autonomous schedules, this novel method operates without necessitating dedicated communication between nodes, thus avoiding additional convergence or signaling overhead. Moreover, the individual nodes require minimal prior knowledge and computational complexity.

The proposed approach operates in real-time, enhancing its adaptability to network architecture and external variations.

1.1 MOTIVATION

In today's fast-changing tech world, the Internet of Things (IoT) has emerged as a crucial factor, attracting industries and consumers with its multifaceted applications. One essential component of IoT is Wireless Sensor Networks (WSNs), which play a vital role in the broad world of IoT applications. These networks, characterized by their low-power sensor nodes, have become instrumental in facilitating seamless communication between devices, systems, and users. However, as the demand for wireless IoT applications continues to grow, the need for advanced solutions becomes increasingly crucial. This entails the convergence of high throughput, unwavering reliability, and energy efficiency to address the challenges posed by the concurrent operation of numerous nodes.

The IEEE 802.15.4e standard has gained widespread acceptance as a foundational framework for establishing connections among energy-efficient, reliable communication devices powered by batteries, commonly known as 'things' in the context of IoT applications. This standard is particularly tailored to scenarios where power availability is limited and demands for data throughput are not overly stringent. Among the collection of MAC behaviors governed by this standard's Time Slotted Channel Hopping (TSCH) is mainly adopted for the fourth industrial application due to its lower latency and high throughput features.

TSCH merges the benefits of channel hopping, enhancing reliability, with time-slotted access that facilitates energy efficiency and congestion-free utilization of wireless connections. In TSCH networks, sensor nodes maintain a predefined schedule that governs their transmission and reception times. Therefore, an optimal schedule is of utmost significance for ensuring optimal network performance. To address this, we propose an energy-aware schedule for the TSCH network utilizing the Reinforcement Learning algorithm. The key aspect of the proposed approach focuses on diminishing energy usage by using online learning to keep unused active slots in a sleep state.

In wireless sensor networks, most energy depletion occurs for unwanted listening of the receiving node. To overcome this issue, the proposed dynamic and learning-based strategy approach provides an optimum scheduling solution while maintaining a balanced packet delivery ratio and lower power consumption.

1.2 NOVELTY OF THIS THESIS

The core novelty of this thesis resides in the pioneering implementation of a learning-based scheduling framework within the Time-Slotted Channel Hopping (TSCH) Medium Access Control (MAC) protocol, with a primary focus on minimizing power consumption for energy-constrained Internet of Things (IoT) devices. The distinctive feature of this work is the integration of reinforcement learning into the TSCH MAC protocol, enabling the establishment of an energy-efficient mechanism for data transmission and reception.

In a decentralized manner, each node autonomously computes its scheduling decisions using a reinforcement learning model, thereby eliminating the need for inter-node signaling overheads. The proposed approach leverages an online learning system that dynamically adapts the wake-up schedule of radio operations based on a predetermined threshold, effectively extending network longevity through periodic transitions between sleep and active states.

Unlike traditional techniques reliant on fixed slot allocations, this research pioneers a flexible and adaptive strategy, utilizing the Q-learning algorithm to enable nodes to independently adjust their radio according to real-time environmental patterns.

1.3 CONTRIBUTIONS

In summary, the main contributions of this thesis are:

- A novel approach to energy-aware scheduling is presented, utilizing Reinforcement Learning (RL) within the IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) medium access

control protocol. The implementation is integrated into the TSCH-Sim Simulator [16]. This approach employs RL algorithms triggered by events like data transmission or reception.

- An adaptive threshold-based method is employed to manage radio on-off activity. Experiments are conducted to identify the optimal threshold value and its effects on the count of active slots.

- A variety of heterogeneous network architectures, ranging from simple to complex, are developed based on factors like density and traffic rate. These architectures are evaluated using the proposed approach, assessing the balance between packet delivery rate and energy consumption.

- The RL algorithm's convergence behavior is explored, considering different transition phases and learning rates required to achieve optimal performance. Additionally, a gradient-based epsilon-decay technique is suggested to enhance exploration and exploitation probabilities for more effective learning convergence.

- The proposed approach is evaluated and compared against the state-of-the-art Orchestra scheduling method. We show how the RL-based scheme retain performance in terms of a trade-off between PDR and energy expenditure.

1.4 ORGANIZATION OF THE THESIS

In this thesis, we organize our work into six sections. In Chapter 2, we conduct a comprehensive literature review of current Medium Access Control (MAC) scheduling methods, categorizing them into centralized, distributed, and reinforcement learning-based approaches.

Moving on to Chapter 3, we delve into background studies concerning the Time-Slotted Channel Hopping (TSCH) protocol and the reinforcement learning algorithm related to our research. We illustrate the key aspects of the TSCH MAC through appropriate diagrams, while also providing

an in-depth explanation of the functioning process of the reinforcement learning-based algorithm. This chapter introduces our energy-conscious reinforcement learning algorithm tailored for TSCH networks. It outlines the algorithm's specifics, including its layered architecture, and offers an overview of the proposed system's workflow.

Chapter 4 offers an outline of the implementation of the proposed approach within a simulation environment. It includes a sequence diagram illustrating the operational steps for a clearer comprehension of integrating the proposed approach with the MAC layer. Additionally, the chapter covers the discussion of simulation tools and the analysis of essential parameters needed for scenario validation.

Chapter 5 is more focused on scenario design and simulation validation. It provides an analysis of various scenarios with specific attributes. Furthermore, this chapter describes selecting parameters for the reinforcement learning approach and provides the formulas for evaluating metrics such as PDR (Packet Delivery Ratio), latency, and power consumption within the proposed system.

Within this chapter, the outcomes of each scenario are discussed in terms of PDR, latency, and number of active slots. Additionally, it offers a brief overview of the outcomes achieved through Orchestra scheduling for each corresponding scenario. A comparative analysis is then presented, accompanied by valid justifications. In the conclusion section of this chapter it provides an overview of the stability analysis of Q values and the approach used to estimate the average packet delivery ratio, illustrated with the assistance of a diagram.

The final chapter of this thesis, Chapter 6, we conclude our work and outline potential future directions for research related to IEEE 802.15.4e TSCH MAC scheduling. Within this chapter, we encapsulate the primary contributions and discoveries of our thesis.

Chapter 2

LITERATURE REVIEW

The 802.15.4e standard has significantly advanced node communication, yet it does not define any policy regarding scheduling mechanisms. As a result, scheduling within TSCH networks has emerged as an active research area. Several studies have explored this domain, intending to design schedulers tailored to specific application requirements, encompassing energy consumption, throughput, and delay considerations. Two primary categories have emerged: Centralized and Decentralized scheduling. Furthermore, a promising and progressively intriguing approach involves autonomous scheduling based on the Reinforcement Learning algorithm. Our work also delves into this area, as we have generated a schedule using a Reinforcement Learning-based algorithm, optimizing energy usage by putting unnecessary listening nodes to sleep mode.

This chapter provides an overview of the existing work on various scheduling approaches for TSCH networks employing conventional and reinforcement learning-based algorithms, highlighting the progress and gaps and setting the stage for the proposed research carried out in this thesis.

2.1 Centralized Scheduling

In a centralized approach, all communication within the network is scheduled by a single node known as the coordinator or manager. This coordinator, often referred to as the Path Computation Element (PCE) [17], is responsible for constructing and managing the network schedule.

One of the known centralized scheduling algorithms for the TSCH network is TASA (Traffic Aware Scheduling Algorithm) was proposed by Palattella et al. [18]. New industrial (IIoT) applications can effectively leverage this protocol, aiming for low latency and employing low-duty cycles to conserve power.

The main objective of TASA is to create an optimized schedule that minimizes the required number of timeslots for efficiently transmitting all the data to the coordinator. Achieving this goal involves utilizing graph theory, explicitly employing matching and coloring processes. The protocol aims to find the most suitable schedule that requires the fewest timeslots to deliver the accumulated data to the coordinator successfully. To achieve efficient scheduling, TASA employs a matching algorithm, which schedules all eligible links simultaneously in the same timeslot. Additionally, the protocol adopts a vertex coloring approach for assigning channel offsets to each slot. The authors also observed that introducing more channels can enhance network metrics such as throughput, reduce latency, and decrease energy consumption.

Jin et al.[19] presented AMUS (Adaptive Multi-hop Scheduling), an early centralized scheduling protocol with the key objectives of ensuring high reliability and low latency. This algorithm efficiently employs an End-of-Q notification technique to conserve nodes' energy consumption. AMUS introduces a novel tentative resource allocation method, which allocates additional slots to vulnerable links that assist in lowering delay caused by interference or collision. AMUS also used the PCE-based protocol to gather information from nodes for computing the schedule. The initial information gathering employs CSMA/CA, followed by calculating and disseminating the first schedule to the nodes. Even though this protocol outperforms TASA regarding reliability and lower latency, it has higher power consumption resulting in an increased duty cycle.

This paper [20] introduces the Multichannel Optimized Delay time Slot Assignment (MODESA) scheduling approach for the TSCH network. It is suitable for homogeneous networks, where the data rate is the same for all the motes. In this protocol, at every iteration, it allows only one node and provides an appropriate link to assign one of its required transmissions. Moreover, it helps

reduce buffer congestion by first scheduling the node with the higher number of packets. However, this technique has been enhanced and extended [21] to accommodate heterogeneous traffic.

2.2 Decentralized Scheduling

Unlike the centralized scheduler, in the distributed approach, individual nodes engage in negotiations with their neighboring nodes and independently make decisions regarding the scheduling of links with specific neighbors. Here, we outline the key distributed solutions below.

The first decentralized scheduling algorithm for TSCH network was proposed by Tinka et.al [22] and conducted with mobile networks. This protocol comes with two variations: Aloha-based and Reservation based scheduling. The researchers utilize advertisement packets to transmit information about available slots and the corresponding channels for connectivity. Additionally, Connection Request packets serve the purpose of establishing links between nodes. In the case of Aloha, a single channel is allocated for broadcasting advertisements to attract new neighbors. However, Reservation-based scheduling improves upon Aloha by introducing a dedicated timeslot for advertisements. Simulation results indicate that both methods effectively handle dynamic nodes, with the reservation-based algorithm exhibiting superior performance compared to the Aloha-based algorithm. Nevertheless, it's important to note that neither of these solutions is well-suited for nodes with energy constraints.

A Distributed Scheduling algorithm (DIS-TSCH) for the time-synchronized channel-hopping network is introduced by Wang et al. in [23] . This protocol primarily relies on the node's local information, such as location inherited from RPL (e.g., rank, graph depth etc.). The approach's core concept is allocating single slots to all leaf nodes, while intermediate nodes are assigned multiple slots depending on the number of children. Afterward, for every designated time slot, a channel offset is assigned. While this schedule might result in some idle slots, simulation results demonstrate that this method achieves high throughput, low latency, and minimal control overhead, consequently leading to reduced power consumption.

Another distributed scheduling for IEEE 802.15.4e networks is Wave [24]. This algorithm is a convenient approach for raw data converge-cast, as it is a multichannel and slot assignment algorithm with traffic awareness. In this approach, every single node (within range) has knowledge about its parent and conflicting nodes. In this protocol, information is acquired through signaling between neighboring nodes, and the initial Wave is generated by incorporating all the partial information available to each node. The sink node solely computes the first Wave and disseminates this data within the tree. Subsequently, individual nodes autonomously compute the subsequent waves. This protocol performs better when applied to a heterogeneous network than a homogeneous one.

In [25], Soua et. al introduced DiSCA (Distributed Scheduling for Convergecast in Multichannel WSNs Algorithm), a distributed interference and traffic-aware scheduler tailored for TSCH networks. The main goal of DiSCA is to minimize the timeslots needed for transmitting total packets to the sink. It prioritizes scheduling for nodes with higher packet transmission requirements. Each iteration generates a micro-schedule that can overlap, reducing the number of slots. It also considers the sink, which is equipped with multiple radio interfaces and weighs varying traffic loads generated by other nodes. Simulation shows that DiSCA outperforms Wave [24] regarding the required number of slots for different configuration setups.

Another well-known scheduling protocol, Autonomous Link Based Cell Scheduling (ALICE), was introduced by Kim et al. [26]. This algorithm has been further investigated by Elsts et al. [27] for facilitating multichannel. ALICE is a kind of autonomous scheduling protocol which manages resources as links by allocating cells (timeslots) for one-directional links between neighboring nodes. Utilizing traffic direction operations, routing, and slot frame management ensures that downstream and upstream communications remain isolated and do not interfere with each other. As both referenced papers demonstrated, it exhibits enhanced performance compared to other autonomous schedule techniques.

Jeong et al.[28] presented a new approach, introducing the On-Demand TSCH Scheduling with

Traffic Awareness (OST) algorithm. This novel protocol autonomously allocates resources based on the nodes' traffic load. Notably, OST's unique feature allows it to adjust the timeslot frequency in real-time as required dynamically. This dynamic resource allocation enables nodes to transmit their accumulated data swiftly. The study's results demonstrate a significant enhancement in reliability by 60% and a remarkable 52% improvement in energy efficiency compared to other existing state-of-the-art protocols.

The Minimal Scheduling Function (MSF) [29] is a scheduling mechanism developed by the 6TiSCH working group. This protocol consists of two distinct phases: the joining and operational phases. During the joining phase, a node enters the network and allocates a shared slot for TSCH, RPL, and control messages. Following this, the node assigns two additional slots: one for receiving data from its child nodes and the other for transmitting data to its parent node, ultimately reaching the sink. The latter phase comes into play when two nodes negotiate using the 6TiSCH protocol stack to allocate cells. MSF is closely reliant on RPL, making it vulnerable to network failures in the event of RPL errors. Additionally, authors in [30] compared MSF with the On-The-Fly (OTF) protocol [31] and concluded that MSF's performance might be compromised by issues in the 6TiSCH stack, particularly related to RPL configuration, similar to other resource negotiation-based scheduling methods.

Orchestra [32] is an autonomous scheduling approach that operates without the need for either a centralized or distributed scheduler. It stands out for its simplicity, lightweight nature, and adaptability. Relying on RPL (network topology), Orchestra quickly responds to sudden changes in the network. The protocol eliminates the necessity for negotiations; nodes independently compute and maintain their schedules, automatically updating them if the topology changes, all without incurring signaling overhead. The Orchestra schedule comprises three distinct slot frames, each exclusively allocated to handle specific traffic types: MAC TSCH beacons, RPL signaling traffic, and application data. The length of each slotframe introduces a trade-off, aiming to balance network capacity, latency, and energy consumption. Simulation results demonstrate that Orchestra achieves a higher

delivery ratio while maintaining a favorable trade-off between latency and energy consumption.

2.3 RL-Based Scheduling

An RL-based approach is a goal-oriented process where an agent engages in several actions to reach its objective. Each time the agent interacts with the environment utilizing available actions and based on the performance, it earns a reward value specifying how well it performs.

The RL-based process involves a fundamental choice between a multiagent negotiation approach and a single-agent localized decision-making strategy to optimize metrics (e.g., throughput, latency, channel) or derive an ideal schedule for WSNs.

In the multiagent framework, nodes collaboratively negotiate and coordinate to formulate a collective outcome that best serves the network's objectives. This cooperative approach necessitates exchanging information, where nodes share data about their local conditions, capabilities, and demands. Conversely, in the single-agent procedure, nodes autonomously reach their target based on individual assessments of reward and available resources, with limited interactions with other nodes [33].

However, In both scenarios, the key point lies in how nodes leverage reinforcement learning (RL), using feedback mechanisms and reward signals to guide their decisions. The following sections cover most of the recent RL-based scheduling algorithms for wireless sensor networking.

In [34], Nguyen-Duy et. al have proposed an RL-based algorithm for radio scheduling of TSCH network. RL-TSCH leverages a single channel of the medium and employs RL techniques to minimize collisions among transmitting nodes. The protocol takes into consideration both the number of packets in the transmission buffer and the remaining energy level of each node. At the start of each timeslot, RL-TSCH decides whether to activate or deactivate the node's radio based on the node's current and previous state. During the active timeslots, RL-TSCH behaves similarly to the Minimal Scheduling Algorithm (MSA) of TSCH, while during the non-active timeslots, it deactivates its radio. The algorithm utilizes Q-learning with an action space that involves selecting the number of

active slots. The reward function is designed to minimize energy consumption while ensuring high throughput and reliability. Experimental results demonstrate that RL-TSCH significantly reduces energy consumption compared to MSA.

This paper [35] presents a Reinforcement Learning (RL) based Medium Access Control (MAC) protocol to extend network lifetime for wireless sensor networks. The proposed method uses the Q learning algorithm to save energy, which adjusts the radio sleeping and active periods based on data traffic load and neighboring state information. Each agent's action is to determine when it should be active or asleep for every single time slot. The reward function is computed considering the current node state and the neighboring node's status. Performance is evaluated on a real-time testbed (small network), and Contiki Cooja simulator (large-scale). The outcome shows that QL-MAC considerably reduces energy expenditure while preserving better PDR than CSMA/CA.

In [36], Park et. al introduced a multiagent Reinforcement Learning (RL) based TSCH scheduling that allows contention but minimizes collision in a dense network. According to this algorithm, each node acts as an agent that learns the transmission slot with the lowest failure rate and sends the packet only on that slot. An action peeking (AP) mechanism has been adopted to have better convergence in this multiagent system. During AP, a node can observe the communication between neighboring nodes in the listening mode, which assists it in having reservation knowledge about the current slot. Results show that QL-TSCH outperforms both Orchestra and FTA [37] in terms of PDR, a better end-to-end packet delay than Orchestra.

Pratama et al. [38] proposed a reinforcement learning-based Q-learning technique that calculates the optimal number of cells for the TSCH network. To find the required number of cells during each slotframe iteration, the system uses previous state information such as queue utilization, cell utilization, and reliability. The reward function is calculated based on the queue utilization, the number of unused dedicated cells, and the number of drop packets. The result comparison has been made with the QL-TSCH and MSF (Minimal Scheduling Function) scheduling. The experiment outcome shows that the proposed algorithm has a better network lifetime than QL-TSCH and MSF

algorithms. Again it achieves a better packet delivery ratio in different scenarios. Even though the number of slots (101 slots) is the same as MSF, the experimented system has comparatively lower latency than MSF.

A self-organizing [39] reinforcement learning approach for scheduling nodes' wake-up cycles in a wireless sensor network is proposed by Mihaylov et. al . In this protocol, each node learns to stay awake when it needs to communicate with its parents/children nodes (nodes that belong to the same coalition). At the same time, the node learns to stay asleep when neighboring nodes on the same hop communicate (nodes in another coalition). In other words, the node desynchronizes with the neighboring nodes not in its coalition to avoid radio interference and packet loss. Each agent stores a quality value' for each timeslot, which is updated every time an event (overheard, sent or received packets, idle listening) occurs during that slot. The node will stay awake for those consecutive timeslots (of a length equal to the duty cycle) with the highest sum of Q-values. Evaluating this protocol in different topologies has shown that it provides much lower end-to-end latency than S-MAC [40].

In [41], Liu et al. introduced RL-MAC as a reinforcement learning-based MAC protocol for WSNs. The core objective of RL-MAC is the optimization of the radio on-off mechanism within nodes, aimed at minimizing energy consumption in sensor nodes. A distinctive feature of the proposed algorithm lies in its adaptability to both the traffic generation pattern of the node itself and that of its neighboring nodes. The method encounters the number of packets queued for transmission at the beginning of the slotframe. The action is the reserved active time for the node. In this approach, a node can actively infer the state of other nodes to achieve the cumulative goal for a wide range of traffic conditions. In comparison with established MAC protocols like S-MAC and T-MAC, the performance evaluation of RL-MAC demonstrates that it outperforms in terms of energy efficiency and throughput.

Phung et. al [42]presents a multichannel protocol tailored for WSNs focused on data collection.

The core of this protocol relies on an innovative scheduling algorithm driven by reinforcement learning principles. The primary objective of this algorithm is to mitigate energy consumption attributed to collisions, idle listening, and the deafness problem within WSNs. The algorithm effectively tackles the combined challenge of route selection and transmission scheduling in a completely decentralized manner, eliminating the need for node coordination. In simpler terms, nodes are taught not just which parent to forward data to but also the optimal channel for transmission.

In practical terms, a node takes an action from a predefined set of options during each time slot. The node then records the success probability for each action performed in that slot. This probability is updated based on the chosen action, forming the foundation for selecting the most favorable actions during the scheduling phase. if an action proves successful in a given slot, it is repeated in the subsequent frame. Conversely, an unsuccessful action prompts the node to randomly select another action from the available options in the next frame. The outcomes of the proposed protocol indicate its superior performance compared to a frequency-hopping protocol named McMAC [43] concerning Packet Delivery Ratio (PDR) and end-to-end latency.

Reference name	Objectives	MAC Layer	Comments
Hung et al. [34]	Energy consumption	IEEE 802.15.4e	Small networks with few nodes, homogeneous traffic rates
Park et al. [36]	High throughput	IEEE 802.15.4e	No receiving schedule, nodes listening through entire slotframe
Yolanda et al. [38]	Dynamic cell allocation	IEEE 802.15.4e	Emphasized on keeping low cell count and packet loss
Mihaylov et al. [39]	Low latency	Other MAC	Nodes stay awake for consecutive slots(Lng:duty cycle) for all nodes
Phung et al [42]	Throughput & energy	Other MAC	Emphasized on learning based channel utilization and lower packet collision
Claudio et al. [35]	Energy saving	Other MAC	No insights on learning transitional phase and overall network latency
Liu et al. [41]	Throughput & energy	Other MAC	Did not provide any clue on delay measurement
Stefano et al.[44]	Energy saving	Other MAC	Emphasized on traffic load and neighboring condition

Table 2.1: Summary of existing studies concentrating RL-based algorithm

Table 2.1 provides an overview of the established methods that exclusively employed the Reinforcement Learning-based algorithm. The information presented in the table demonstrates that a majority of investigations utilized the Q-learning algorithm to fulfill their goals.

Hence the 'Approach' column indicates the pieces of information that were utilized leveraging the Q-learning algorithm. Moreover, among these studies, few primarily focused on time-slotted channel-hopping networks.

Our proposed work introduces an innovative approach to tackle the problem of idle radio reception, aiming to reduce power consumption while maintaining a balanced packet delivery. In contrast to prior research, where the RL algorithm was predominantly applied to transmission scheduling, our study explores deeper into achieving an energy-efficient scheduling solution that encompasses both transmission and reception aspects.

In contrast to existing work, the proposed approach operates without necessitating dedicated communication between nodes, thus avoiding additional convergence or signaling overhead.

Chapter 3

METHODOLOGY

This chapter presents the technical background studies and methodology of the proposed approach. A brief description of Time Slotted Channel Hopping (TSCH) MAC protocol and Reinforcement Learning (RL) based algorithm has been carried out to provide a foundation for the work presented in this thesis. The Medium Access Control (MAC) protocol within the data link layer is a vital controller of energy resources in modern communication systems. MAC protocols are crucial in minimizing unnecessary energy expenditure by intelligently managing how devices access the shared communication medium. A key aspect of these MAC's energy-saving proficiency lies in its scheduling mechanisms. By strategically allocating time slots for data transmission and reception, the MAC protocol limits collisions, reduces idle listening periods, and ensures efficient channel utilization.

Reinforcement Learning (RL) has emerged as a remarkable tool in addressing various challenges of Wireless Sensor Networking (WSN) by offering innovative solutions to enhance multiple aspects, such as throughput improvement, reduced delays, and efficient energy conservation. Notably, the integration of RL into WSNs brings about a remarkable change, enabling these networks to efficiently adapt to dynamic environments, upgrade operational efficiency, and promote promising solutions to complex problems.

To save energy, the proposed approach utilized TSCH MAC which functioned according to a

schedule generated by the RL-based algorithm. The schedule facilitates control over the transitions of the node's active and sleep modes in real-time, resulting in power conservation.

3.1 TIME SLOTTED CHANNEL HOPPING (TSCH) MAC PROTOCOL

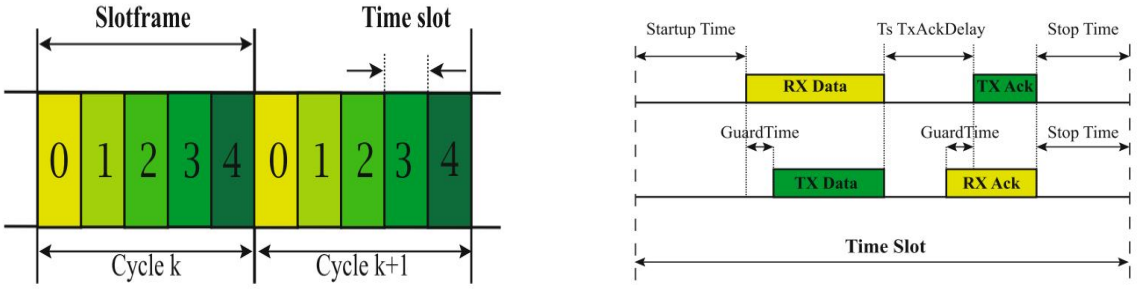
In order to facilitate the successful functionalities of today's innovative system, industrial IoT devices need to operate swiftly regardless of the number of connected devices, should exhibit reliable and predictable behavior, withstand challenging environmental conditions, and maintain strong security and resilience measures. Hence, thanks to IEEE 802.15.4e standard [45], aiming to establish a range of MAC behavior modes suited for distinct application domains within the realms of process and factory automation.

Among these modes, one notable mode is TSCH (Time-Slotted Channel Hopping), which offers significant advantages in terms of high determinism, reliability, and low power consumption. By employing time synchronization and channel hopping mechanisms, TSCH effectively addresses the stringent demands of the fourth industrial network. The following sections briefly cover the fundamental elements of the TSCH protocol.

3.1.1 Slot Frame

A slot frame represents a self-repeating unit that operates periodically to synchronize all the nodes in a network. Each slot frame has an associated handle. TSCH can accommodate both single and multiple slot frames. The multiple slot frame can establish distinct communication schedules for various node groups or operate the network with varying duty cycles.

The size of the slotframe is flexible and can be adjusted based on the application's specific requirements. A larger slotframe can have better throughput but increase the overall network latency. On the other hand, reducing the length of the slotframe increases the frequency of time slot repetition, leading to improved reliability and higher power consumption. Figure 3.1 (a) depicts the slotframe of a TSCH network.



(a) TSCH Slotframe (b) TSCH Timeslot

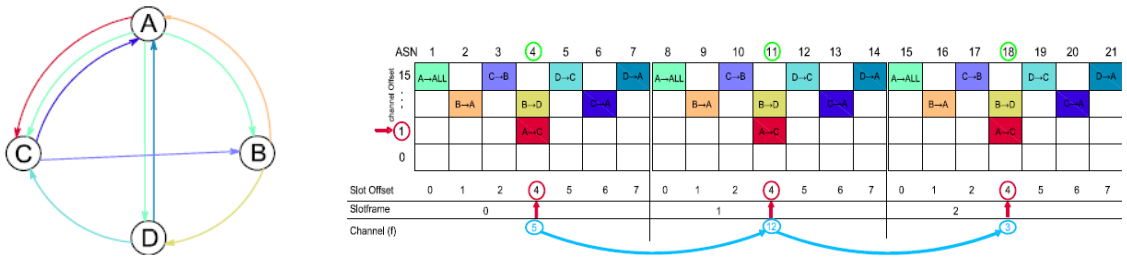
Figure 3.1: The structure of slotframe and timeslot of a TSCH network [46]

3.1.2 Time Slot

A slot frame can be further divided into smaller subunits known as a timeslot to exchange data and acknowledgment between devices as depicted in Figure 3.1 (b). The timeslot of each successive slotframe can be identified using Absolute Slot Number (ASN). The ASN value represents the total number of timeslots that have elapsed since the start of the network. At the start of the network, the value of ASN is initialized to zero and gradually increased by one at every timeslot as follows:

$$ASN = (K * S + t) \tag{3.1}$$

Where K denotes slot frame cycle, S belongs to slotframe size, and t is the timeslot. A timeslot can be either dedicated or shared. In the case of a dedicated timeslot, it is exclusively assigned to a single owner, and any re-transmissions take place during the subsequent time slot. Conversely, shared timeslots are deliberately allocated to multiple devices for transmission. As a result, the likelihood of collision is higher, and this collision is identified when the acknowledgment is not received. Hence, in order to minimize the chances of collisions, a re-transmission back-off algorithm is applied to shared links. This algorithm involves increasing the size of the back-off window after each transmission failure and resetting it to its minimum value following a successful transmission.



(a) Network Topology (b) Channel Hopping mechanism

Figure 3.2: These figures illustrate the channel hopping mechanism for 3 cycles where the slotframe contains 7 timeslots.[48] Hence each time the frequency is generated using equation 3.2

3.1.3 Channel Hopping

Channel hopping is another technique employed in TSCH networks. Transmitting data across various frequencies enables TSCH to facilitate communication via multiple channels, known as multichannel communication [47], as illustrated in Figure 3.2. The utilization of different frequencies generates frequency diversity, which helps mitigate the impact of interference and multipath fading. This, in turn, enhances reliability. TSCH implements channel hopping by utilizing 16 distinct communication channels, each identified by a channel offset within the range of (0, 15). In a TSCH network, all the nodes share the same hopping sequence. The physical channel (real channel for packet transmission) of a shared or dedicated cell is determined as follows:

$$f = F[(ASN + ChannelOffset)\%N_{channels}] \tag{3.2}$$

F is a lookup table containing a sequence of available physical channels, and $N_{channels}$ denotes the total number of available channels.

3.1.4 Network Formation

The initiation of network formation in TSCH begins with a PAN coordinator, which broadcasts the network's existence through an enhanced beacon (EB) [49]. EB is a frame that contains all the essential information, such as time synchronization, channel hopping, slotframe, and timeslot, that requires a node to join the network. When a new node intends to join the network, it initially starts listening to the available channel and waits until it receives an EB transmitted by any neighboring nodes. Upon receiving the EB frame, the node synchronized itself and initialized the slotframe and links in the EB. Then it changes its mode to the TSCH network.

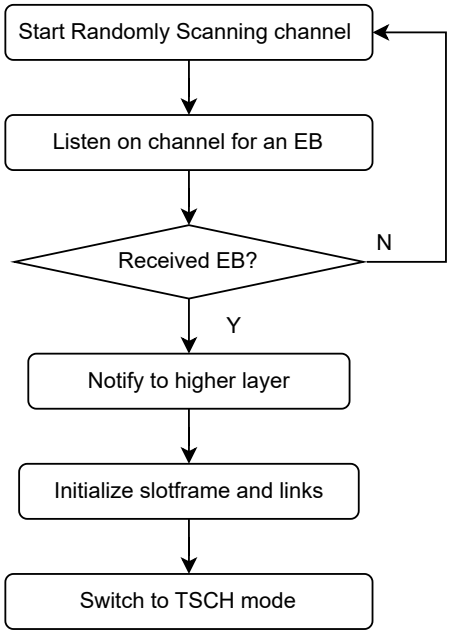


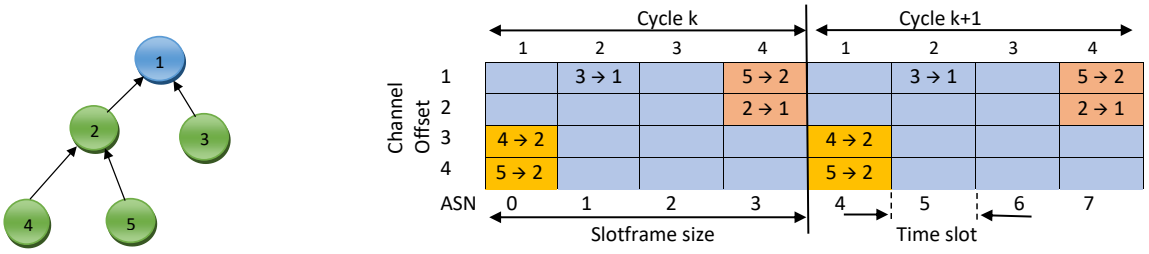
Figure 3.3: TSCH network formation procedure

After completing the joining procedure, the node starts transmitting EB to neighboring nodes. Figure 3.3 demonstrates the general process of joining the TSCH network.

3.1.5 TSCH Scheduling

In the TSCH network, a schedule determines the operation of a node at a particular timeslot. That means the schedule is responsible for defining slots for transmission (either data packet or control packets), reception, and idle slots that allow the node to preserve energy. In addition, the schedule specifies the appropriate channel offset to be utilized for communication purposes.

In TSCH, a schedule can be defined as a matrix of channel offset and time slots representing the cell. A cell located at the intersection between a designated row and column (i.e., channel offset and time slot offset) of that matrix describes a link between neighbor nodes at the data link layer. For a network of five nodes and four edges, an example of a schedule with four timeslots and four-channel offsets is illustrated in Figure 3.4 (b). Thereby, there are 16 cells in this slot frame. Each cell in the TSCH slot frame is considered half-duplex. This clarifies that a node can not transmit and receive at the same time slot or receive from multiple nodes during the same time slot, even if the transmission occurs on a different frequency band. These problems lead collision of packets, as depicted in Figure (b). Considering the first case at time slot 4, node two receives data from node five and sending back to node one, which results in a collision. Again, for the latter case, which happened at time slot 1, nodes four and five send a packet to node two, which also leads collision of packets.



(a) A tree topology

(b) Scheduling of the corresponding topology

Figure 3.4: A simple TSCH schedule with collision scenario

3.1.6 TSCH Timeslot Mode

A TSCH schedule can incorporate the various state of a time slot, which serve as indications for nodes to either transmit, listen, or set their radios to sleep mode. The IEEE 802.15.4e standard has distinguished seven different categories of state for a times slot [50]. A brief description of each of them is given below:

- **TxDataRxAck**: During this particular time slot, a node transmits a frame and gets an acknowledgment (ACK) once the data has been received.
- **TxData**: The node sends a frame without expecting to receive an acknowledgment. More specifically, it species a broadcast transmission
- **RxDataTxAck**: During this time, a node actively listens to the channel and receives a frame, then responds with an ACK to confirm the successful receipt of the transmitted frame.
- **RxData**: In this time state, a node listens and receives a frame, but no ACK is required to send. Consequently, it defines a broadcast reception.
- **RxIdle**: When a node indefinitely listens to a particular channel but does not receive a frame.
- **Sleep**: The mote remains idle and does not engage in transmission or reception activities.
- **TxDataRxNoAck**: The node transmits a frame and expects an ACK, but no ACK is received. A collision of the data frame could cause this situation.

3.2 REINFORCEMENT LEARNING TECHNIQUE

Reinforcement Learning (RL) is defined as a segment of machine learning that diverges from the conventional path of supervised learning by not relying on fixed datasets for its learning process. In reinforcement learning, labeled data typically is not utilized for training purposes. However, this doesn't imply a complete absence of guiding information [51].

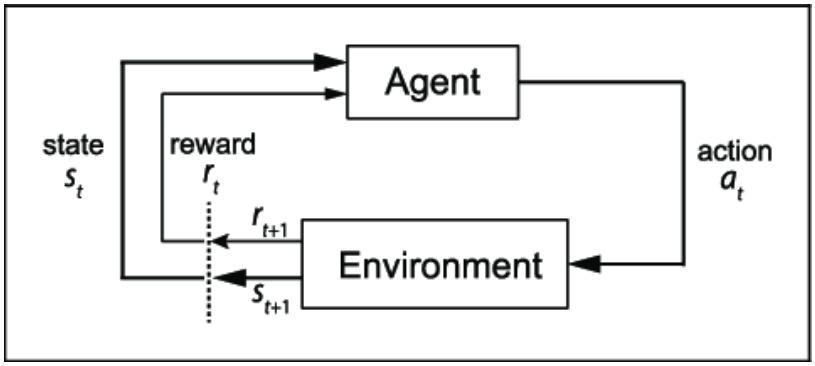


Figure 3.5: Reinforcement Learning procedure [58]

The system operates based on an established reinforcement learning protocol, triggering a feedback signal referred to as a 'reward' upon achieving the intended outcome. For instance, in the context of a robot's walking control, the distance covered could serve as the reward metric. Similarly, within a Go game program, the outcome of victory or defeat translates into the reward. In cases of loss, the reward takes the form of a negative value, commonly termed as a penalty. Reinforcement learning (RL) is widely employed in various fields and domains, encompassing industrial manufacturing [52], robot control [53], simulation [54], optimization, scheduling and gameplay [55],[56].

To get into the working procedure, the RL algorithm involves an agent that interacts with an environment and learns optimal action through a trial-and-error approach, as depicted in Figure 3.5. According to Sutton and Barto (2018) [57], reinforcement learning is a machine learning method that relies on gathering experience from engaging with the environment in order to acquire knowledge. During this learning process, the agent is oriented towards achieving specific goals and must explore various actions to determine which ones lead to the most favorable rewards.

The relationship between an agent and its environment is established through states, actions, and rewards, which are precisely defined within the formal framework of Markov Decision Processes. In the learning process, an agent in state S can take an action a that leads it to another state S' and receives a reward r from the environment. This process is repeated multiple times until the algorithm converges, deriving an optimal policy Π.

RL can be easily implemented within a distributed architecture such as Wireless Sensor Networks (WSNs), where each node aims to select actions that are anticipated to maximize its long-term rewards. In this setup, nodes within the WSNs interact with their respective environments, receiving feedback as rewards or penalties based on their actions. By utilizing reinforcement learning, the nodes learn to make decisions that optimize their cumulative rewards over time, contributing to the overall performance and efficiency of the network. Reinforcement learning has found practical use in various scenarios, such as training game-playing agents [59], educating robots with the ability to learn [60], and developing content placement agents that can intelligently suggest articles or advertisements based on individual user preferences [61].

An RL framework can be structured either as a single agent or a multi-agent system, depending upon the nature of the challenge at hand and the complexity of the environment. The fundamental contrast between single-agent RL and multi-agent RL lies in the agent's engagement with the environment and its effort to optimize individual rewards in the former, whereas in the latter, within a multi-agent system, agents engage not only with the environment but also with each other, striving to enhance their collective rewards.

One of the most popular RL algorithms is Q-learning. The Q-Learning algorithm operates as a model-free, off-policy method within the domain of reinforcement learning. In this learning paradigm, action selection rely on state's value, determined through an updating mechanism. The agent opts for actions that yield the highest rewards from its environment, where these rewards can vary in positivity or negativity based on the environment's characteristics. It is worth highlighting that the utilization of Q-learning has been widespread and effective in addressing various issues problem Wireless Sensor Networks (WSNs) (e.g.[62], [63],[64],[65]).

This thesis will focus on Q-learning related to our objectives of preserving energy in a TSCH network. A detailed discussion on Q-learning is depicted in the following section.

3.2.1 Q -Learning Algorithm Details

The Q-learning algorithm is defined using a collection of elements (S, A, T, R). Here, S represents a discrete group of states within the environment, A signifies a discrete set of available actions, T stands for the state transition function $S \times A \times S$, producing values between 0 and 1, and R stands for the reward function $S \times A \rightarrow R$. Q learning possesses a compelling feature: it starts with no prior understanding of state transitions and reward functions, which it gradually learns from interactions with the environment. At each step, the agent gets information about its state $s \in \mathcal{S}$ from the environment and picks an action $a \in \mathcal{A}$. Once this action is taken, the environment's state changes, leading to the emergence of a reinforcement signal $r \in \mathcal{R}$. This signal is then used to assess the decision quality by updating the relevant Q(s, a) values [66].

The primary objective of an agent is to select actions that maximize the value function,$Q(s, a)$ associated with the specific state-action pair [67]. Hence, $Q(s, a)$ denotes the anticipated cumulative discounted rewards resulting from the action taken in state s.

$$Q(s_{t+1}, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Max_a Q(s_{t+1}, a)] \tag{3.3}$$

where α is the learning rate ($0 < \alpha < 1$) parameter and is used to tune the learning speed. The discounting factor, $0 < \gamma < 1$, influences the agent's preference for prioritizing either immediate rewards or long-term rewards; r represents the reward obtained when the agent executes action a in state s, which consequently leads to state $Q(s_{t+1})$. Moreover, $Max_a Q(s_{t+1}, a)$ signifies the highest Q-value achievable among the actions that can be taken in the subsequent state.

In order to better converge for the Q-learning algorithm, it is essential to explore every state-action pair. This can be done by taking advantage of exploration and exploitation probability. In this regard, an agent will explore a random action with the probability ϵ and exploit the environment with the best-known action with probability $(1-\epsilon)$. There are various techniques to determine those probability values. It could be a fixed value or a decaying ϵ-greedy approach [68]. In the latter case,

the value of epsilon decreases solely over time. The gradient-based ϵ-greedy method that has been used for this experiment is as follows:

$$epsilon = epsilon + learningRate * gradient \tag{3.4}$$

$$gradient = -1 * epsilonDecayRate \tag{3.5}$$

$$epsilon = Max(0, Min(1, epsilon)) \tag{3.6}$$

Where $epsilonDecayRate$ indicates the rate at which the exploration probability gradually decreases over time.

3.3 PROPOSED APPROACH WORKFLOW

A three-layer network architecture has been considered to facilitate the implementation of the proposed approach. This architecture comprises the layers of Application, Network, and MAC (Medium Access Control). The subsequent sections provide concise descriptions of each of these layers.

3.3.1 Application And Network Layer

An application layer has been designed to interact with the network layer while constructing the whole network architecture. Its primary function is to configure the network topology and define essential attributes for the simulation. To build the topology, the application layer requires information on the total number of nodes needed for deployment and their respective categories.

The layer assigns a unique ID for each node and specifies its category. Moreover, the layer is responsible for defining the packet generation rate for each distinct sender node. This rate determines how frequently each node generates data after a certain period. Additionally, the application layer determines the position, layout, and connections among all the nodes in the network. This step is

essential for establishing the overall structure of the network.

The network layer facilitates the movement of data packets towards the central sink node. Due to the random positioning of nodes, not all nodes are located near the root node. To address this, a multi-hop routing approach has been employed, using the nearest next-hop identifier. This strategy guarantees the successful delivery of each packet to its intended destination.

3.3.2 MAC Layer

The MAC (Medium Access Control) layer ensures efficient and reliable communication among nodes. It conducts several core functions vital for the proper functioning of the network. It defines the transmission range, which determines the maximum distance a node can communicate with its neighbors. Moreover, using a radio propagation model, the network layer establishes links between nodes. The model considers the transmission range and ensures that nodes within this range are connected. Another essential role of this layer is to calculate nodes' Received Signal Strength Indicator (RSSI) value to determine the signal strength for efficient data transmission.

The proposed system has been considered the Unit Disk Graph Model (UDGM) for radio propagation. In this model, signal propagation involves circular areas centered on the transmitting node to simulate radio coverage. The likelihood of successful packet reception depends on the distance and is computed using the following formula:

$$P_{success} = 1 - d^2 * (1 - UDGM\_RX\_Success) \tag{3.7}$$

The central aspect of this research revolves around the MAC layer, which incorporates a learning algorithm to efficiently manage the on-off periods of the radio transmission module. Specifically, it is responsible for regulating access to the transmission medium and interacting with the network layer for engaging transmission and acknowledgment.

The proposed approach has considered a single slotframe comprised of multiple shared cells

based on network architecture, as shown in Figure 3.6. The first cell is an EB cell mainly used to broadcast an Enhanced Beacon (EB) to neighboring nodes to join the TSCH network. On the other hand, rest cells are shared and used to transmit or receive unicast packets or to sleep. The proposed algorithm is responsible for scheduling only the unicast cells.

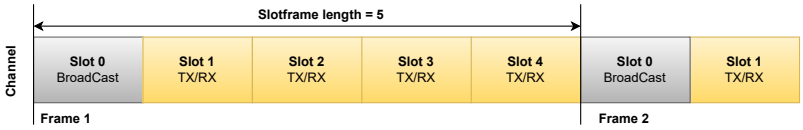


Figure 3.6: Slot frame structure

In order to access the medium, Q-learning has been embedded in the MAC layer. The primary objective of the algorithm is to acquire an optimal wake-up schedule, effectively reducing the number of slots during which the node's radio remains active. By doing so, it aims to mitigate energy wastage caused by factors like idle listening and significant sources of energy dissipation on the receiving side. By leaning on an adaptive schedule, the system can gradually reduce power consumption while maintaining the network's other metrics, such as PDR and latency.

Energy Aware Reinforcement Learning (EARL)

The proposed approach is based on a reinforcement learning algorithm and is suitable for online learning. Since the system is reactive, an agent engages in an environment, chooses a particular action at each step, and obtains a reward in response from the environment. As the optimal action is not known in advance, the agent must learn from its experiences by executing a sequence of diverse actions and inferring the best one based on the received rewards.

In the Q-learning algorithm, the agent's actions rely on a 'Q-function' to assess the value of a specific action in its current state. Thus, each node computes the Q-function for every timeslot within a frame, and these Q-values are stored and updated across the slotframe. In our scenario, each calculated Q-value indicates the effectiveness of turning the radio ON at a specific slot in the

frame, ultimately creating an efficient wake-up schedule for the node in the current slotframe. In this approach, the features of the Q-learning algorithm are declared below:

- **Agent**: Each node of a network is considered an agent.
- **Action**: The available actions to a node decide whether it should stay active for communication or turn off the radio to enter the sleep mode during each single timeslot.
- **Action Space**: Since the action is related to scheduling timeslots, the action space is proportional to the number of slots within a frame.
- **Reward**: Reward defines the positive and negative feedback of a taken action involving successful packet transmission and acknowledgment.
- **Policy**: In the context of Q-learning, the policy serves as the guiding principle for agents to make decisions in each state, aiming to maximize their overall reward. Several methods can be used to determine which action to take. One straightforward approach is greedy selection, where the agent always chooses the action with the highest state-action value, focusing solely on exploitation. However, this method may lead to locally optimal policies in many optimization problems, which may differ from the globally optimal solution.

  In contrast, exploration is a strategy where the agent deliberately chooses non-optimal actions in its current situation to gain more knowledge about the problem. This knowledge helps the agent avoid getting stuck in locally optimal policies and eventually reach the globally optimal solution. But, excessive exploration can significantly decrease the performance of the learning algorithm. Achieving the right balance between exploration and exploitation is a significant challenge in Q-learning.

  A simple approach suggested to address this challenge is the ϵ-greedy strategy, where a parameter ϵ ($0 < \epsilon < 1$) determines the probability of exploration. A higher value of ϵ corresponds to a greater likelihood of exploration. The choice of ϵ significantly impacts the

algorithm's performance. Several studies concluded [57] that nonzero ϵ values generally yield better results compared to the blindly greedy case. However, excessive exploration becomes unnecessary after an initial period of interaction between the agent and the environment, assuming that the state-action-pair values remain constant. Therefore, the proposed system potentially improves the basic ϵ-greedy approach by gradually reducing the value of ϵ as described in previous section 3.2.1 during the learning process. By doing so, we aim to improve the agent's capability to gain new knowledge while preventing any performance decline employed by a constant ϵ value.

According to the proposed method, every node has its own Q table that stores a set of Q-values. The length of the Q table is equal to the slotframe size. The Q values correspond for each timeslot during the learning phase. Hence, the learning phase specifies the duration commencing after the network stabilization period and persists until the transition state begins.

For every individual node, the Q-function is calculated for each slot within the slotframe. The resultant Q-values are saved and progressively revised as frames progress. Each of these Q-values signifies the advantage of activating the node during a particular slot to receive or transmit data. As a result, this collection of values governs the node's activation pattern throughout the ongoing frame. The proposed method employs a single Q table for both transmission and reception, and the Q values are modified in response to specific slot events like data transmission or reception. When considering a particular node, denoted as n, the quality value of a specific timeslot t is estimated using the subsequent update formula:

$$Q^n_{t,f+1} \leftarrow (1-\alpha)Q^n_{t,f} + \alpha(r + \gamma Max_a Q^n_{t,f+1}) \tag{3.8}$$

where $Q^n_{t,f}$ specifies the Q-value associated to the timeslot t of the current frame f, and $Q^n_{t,f+1}$ is the updated value of the same slot but for subsequent frame.

Notably, it should be mentioned that initially, all nodes have their radio active in every timeslot,

indicating that they are actively monitoring the entire frame. At the onset of the transition phase, nodes make a determination to either remain active or enter sleep mode, as depicted in the equation below.

$$Radio_{\{slot\ i\}} = \begin{cases} Active, & Q_i^n \geq Threshold \\ Sleep, & Otherwise \end{cases}$$

If the Q value of a particular slot i falls below this threshold, the node will enter a sleep mode throughout the entire slot duration to conserve energy. Otherwise, the node will stay active to receive or transmit data.

Algorithm Details

Algorithm 1 presents the Q-learning-based TSCH MAC scheduling, a dynamic and iterative procedure that aims to optimize communication efficiency. The proposed system is designed as event-based inside the simulator.

At the inception of each slotframe cycle, the algorithm embarks on the MAC scheduling procedure. If the packet buffer contains data, the algorithm iterates through the buffer, selecting an action based on the $ChooseAction$ function. This function balances between exploration and exploitation strategies: occasionally opting for exploration (with a probability determined by ϵ) and otherwise favoring exploitation. During exploration, a random action is selected within the slotframe, allowing the algorithm to discover new scheduling possibilities. Conversely, during exploitation, the algorithm leverages the best-known actions to optimize the current schedule.

After each action is executed at the MAC layer, the algorithm observes a reward and proceeds to update the Q-table accordingly. The reward is determined by whether an acknowledgment is received. If an acknowledgment is received, the reward is positive; otherwise, it is zero.

Algorithm 1: EARL based TSCH Scheduling

Initialize parameters: Q_table, $\gamma, \alpha\ \epsilon$, $action\_space : \mathcal{A}$, $gradient$;
Func `TSCH_Schedule()`**:**
    Repeat at the beginning of each slotframe cycle;
    if $(Pkt\_Buffer\_has\_packet)$ **then**
        for $i = 0$*;* $i < length(Buffer)$*;* i**++ do**
            $scheduled\_slot \leftarrow$ `ChooseAction(`Pkt_Src`)`;
        end
        `Update_epsilon(`ϵ`)`;
    end
    else
        No Schedule is Needed;
    end
Func `ChooseAction(`$Node_i$`)`**:**
    $P_{exploitation} \leftarrow Uniform\_rand()$;
    if $(P_{exploitation} < \epsilon)$ **then**
        Do exploration with ϵ;
        return $random\ action\ a \in \mathcal{A}$;
    end
    else
        Do exploitation with $(1 - \epsilon)$;
        return $best\_known\ action\ max_{a \in \mathcal{A}} Q(s, a)$;
    end
After performing each $action$ at MAC layer, observe $reward$ and update Q_table;
$reward \leftarrow$ `get_reward()`;
$Q\_table \leftarrow$`Update_Qtable(`$reward$`)`;
Func `get_reward()`**:**
    if $(Acknowledgment == True)$ **then**
        $r \leftarrow 1$;
    end
    else
        $r \leftarrow 0$;
    end
    return r
Func `Update_Qtable(`r`)`**:**
    $Q^n_{t,f+1} \leftarrow (1 - \alpha)Q^n_{t,f} + \alpha(r + \gamma Max_a Q^n_{t,f+1})$
Func `Update_epsilon(`$curr\_\epsilon$`)`**:**
    $\epsilon = curr\_\epsilon + \alpha * gradient$;
    $gradient = -1 * epsilonDecayRate$;
    return $Max(0, Min(1, \epsilon))$;

On the Receiving end, as the algorithm starts (i.e., before entering the transition phase), all nodes are actively engaged in continuous listening throughout each time slot according to algorithm 2. Subsequently, during the transition phase, nodes exclusively focus their listening efforts on time slots where their corresponding Q-values are either equal to or exceed a predetermined threshold.

Upon receiving a frame, an agent (i.e, node) identifies whether the frame's destination id corresponds to its own. If the frame's destination id aligns with the agent's id, the frame is accepted; otherwise, it is discarded. Following a frame's successful reception, the node updates the acknowledgment and the Q table for the respective timeslot. Upon reception at the data link layer, if the current receiving node is not the ultimate destination, the frame is subsequently forwarded to the network layer to facilitate its onward transmission toward the final destination.

After transmitting a frame, an agent (sender node) waits for acknowledgment. An acknowledgment could be true or false based on the success or failure of the frame due to collision. Upon receiving an acknowledgment, an agent can determine its action's effectiveness.

The Q-table is continuously updated using the reward gained from the executed action, with the goal of maximizing cumulative rewards over time. The update involves calculating the new Q-value utilizing the existing Q-value, the reward received, and the potential future rewards associated with the chosen action. On the transmission (TX) side, updating the Q value relies on both the exploration or exploitation policy, whereas the receiving (Rx) side employs only the exploitation policy. This updating step consistently improves the Q-table and refines the scheduling decisions made by the algorithm.

Additionally, the proposed EARL algorithm maintains an adaptive exploration rate, which gradually adjusts to encourage exploration at the start and shifts to exploitation. This process enables the algorithm to balance discovering new scheduling strategies and exploiting well-performing actions.

Algorithm 2: Energy Aware learning based Packet Reception

Initialize: $threshold : \phi, transition\_phase : TP$

```
foreach transmitted_pkt do
    if (TP) then
        if Q(ts) ≥ φ then
            Set Radio Active;
        end
        else
            Set Radio Sleep;
        end
    end
    else
        Active on every slots;
    end
    while (Radio ON) do
        if (this.id == frame_destination_id) then
            received frame;
            update Acknowledgment;
            upgrade Q(s, a);
        end
        else
            Discard frame;
        end
    end
end
```

Chapter 4

IMPLEMENTATION

This chapter introduces the experimental framework created to simulate our novel energy-aware reinforcement learning algorithm, referred to as EARL. It also offers a comprehensive explanation of the integration of our proposed approach into the simulation environment and provides a detailed discussion of the parameter analysis associated with the simulation.

4.1 Implementation Details

This section briefly overviews the simulation tool and network configuration of the designed topology. Furthermore, it presents a detailed description of leveraging the proposed approach with the medium access layer. To illustrate this integration, a sequence diagram is presented, outlining the workflow aligning with the existing simulator.

4.1.1 Simulation Tool

Simulators and emulators play a crucial role in designing wireless IoT applications prior to their real-world implementation. Each of these tools comes with its own strengths and limitations, tailored to specific types of applications. Simulators focus on assessing the software aspect of the

application, observing how it functions within a software-defined environment. For this experiment, we chose the TSCH-Sim simulator. TSCH-Sim was chosen over Cooja [69] because it is only suitable for networks with up to a few hundred nodes [16] and tends to encounter challenges related to synchronization and other issues related to custom code extensions. Furthermore, it is selected over the 6TiSCH simulator [70] as it lacks any non-standard schedules and routing protocols and does not have an architecture that would facilitate the addition of custom extensions.

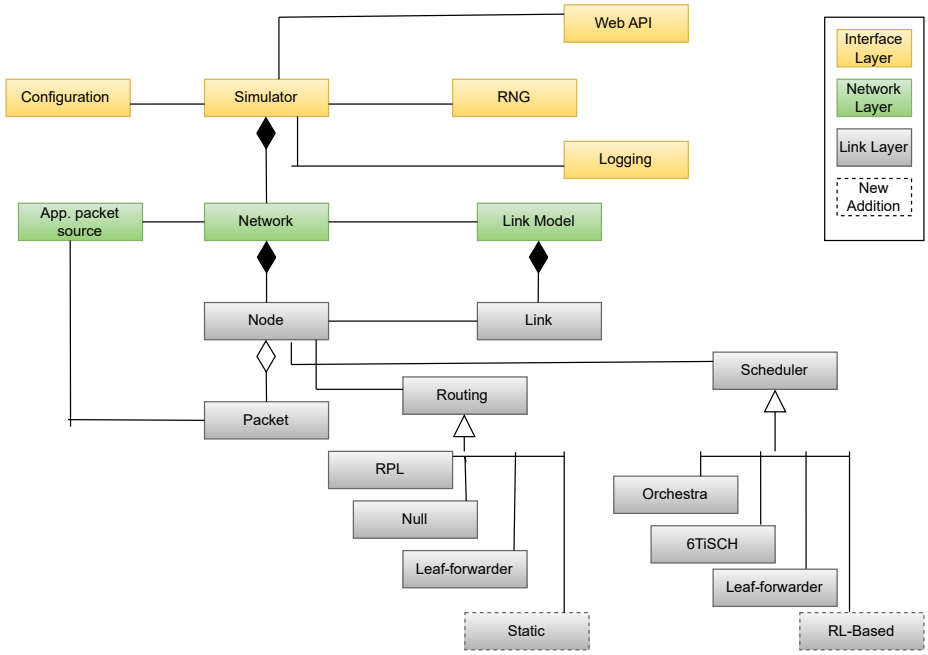


Figure 4.1: TSCH-Sim class diagram showing original and additional new classes

TSCH-Sim is a new discrete event simulator for TSCH and 6TiSCH networks. TSCH-Sim is constructed using modern JavaScript and incorporates modules to make it easily extensible by users, allowing seamless integration with the Web. The simulator adheres to the TSCH protocol specified in the IEEE 802.15.4-2015 standard [71] while supporting features from the evolving 6TiSCH standards and routing protocols like RPL [72]. The fundamental architecture of TSCH-Sim is comprised of several loosely connected components. Figure 4.1 illustrates a class diagram depicting the original internal structure and the supplementary classes we developed for the simulator.

4.1.2 Network configuration

The chosen simulator contains a configuration file that carries all the necessary attributes required to create the network and depicted in the Listing 4.1. In the configuration file, $NODE\_TYPES$, is an object key that contains an array of different types of nodes that are used to construct the network. Each distinct node type must include crucial parameters like the node's $NAME$, $START\_ID$, and $COUNT$, denoting the initial ID and total number of nodes of this type, respectively. Each type's total number of nodes increases sequentially up to the specified $COUNT$ value.

Additionally, the configuration file defines the packet generation frequency for each node and their respective final destinations specified as $APP\_PACKET\_PERIOD\_SEC$ and TO_ID. For instance, nodes 2 and 3 generate packets every 1.5 seconds destined for node 1, as demonstrated in the Listing below. Apart from network construction information, the config file also allows the specification of essential parameters for the TSCH protocol. These parameters encompass the total number of channels required for channel hopping, the number of retransmissions in the event of packet failure, the packet buffer size, and packet size. Within the configuration, we incorporate a random seed generation process with the number of simulation executions. This process ensures that a distinct set of outcomes is produced for each simulation, eliminating any repetition and enhancing result diversity.

The position of each node is defined using the $POSITION$ object key, which holds the unique id and X, Y coordinates for individual node. Furthermore, in order to establish communication between neighboring nodes using a radio propagation model, it is essential to specify connections within the config file. The $CONNECTION$ key, depicted in the Listing, represents a connection between two nodes within range of each other. It is important to clarify that the $CONNECTION$ specification in the network is distinct from routes and is specifically related to the link between nodes. It serves as a requirement for two nodes to communicate, and if ACKs (Acknowledgements) are utilized, it must be bidirectional.

```
{"SIMULATION_DURATION_SEC":800,
"MAC_HOPPING_SEQUENCE":TSCH_HOPPING_SEQUENCE_2_2,
"APP_WARMUP_PERIOD_SEC":100,
"Max_QUEUE_SIZE":16,
"ACTION_SPACE":15,
"SIMULATION_NUM_RUNS":10,
"NODE_TYPES":[
 {
 "NAME":"root",
 "START_ID": 1,
 "COUNT": 1
 },
 {
 "NAME": "intermediate",
 "START_ID": 2,
 "COUNT": 2,
 "APP_PACKETS":{"APP_PACKET_PERIOD_SEC": 1.5,
  "TO_ID": 1}
 },
 ..........
 {
 "NAME": "leaf",
  "START_ID": 5,
  "COUNT": 1,
  "APP_PACKETS":{"APP_PACKET_PERIOD_SEC": 3,
  "TO_ID":1}
  }
  ],
 "POSITIONS":[
 {"ID":1,"X":20.00,"Y":25.00},
 {"ID":2,"X":22.50,"Y":22.00},
 .....
 {"ID":5,"X":28.00,"Y":13.00}
 ],
  "CONNECTIONS":[
 {"FROM_ID":2,"TO_ID":1,
           "LINK_MODEL":"UDGM"},
 {"FROM_ID":1,"TO_ID":2,
           "LINK_MODEL":"UDGM"},
 ......
 {"FROM_ID":5, "TO_ID":2,
           "LINK_MODEL":"UDGM"},
 {"FROM_ID":2,"TO_ID":5,
           "LINK_MODEL":"UDGM"}
 ]
 }
```

Listing 4.1: A sample of the configuration file for 5 nodes network

4.1.3 EARL Algorithm Integration with MAC Layer

In order to access the medium access control (MAC) layer effectively, it is essential to incorporate it into a schedule. A Reinforcement Learning (RL)-based algorithm has been adopted for this research. The implementation of the proposed EARL algorithm is summarized in sequence diagram 4.2.

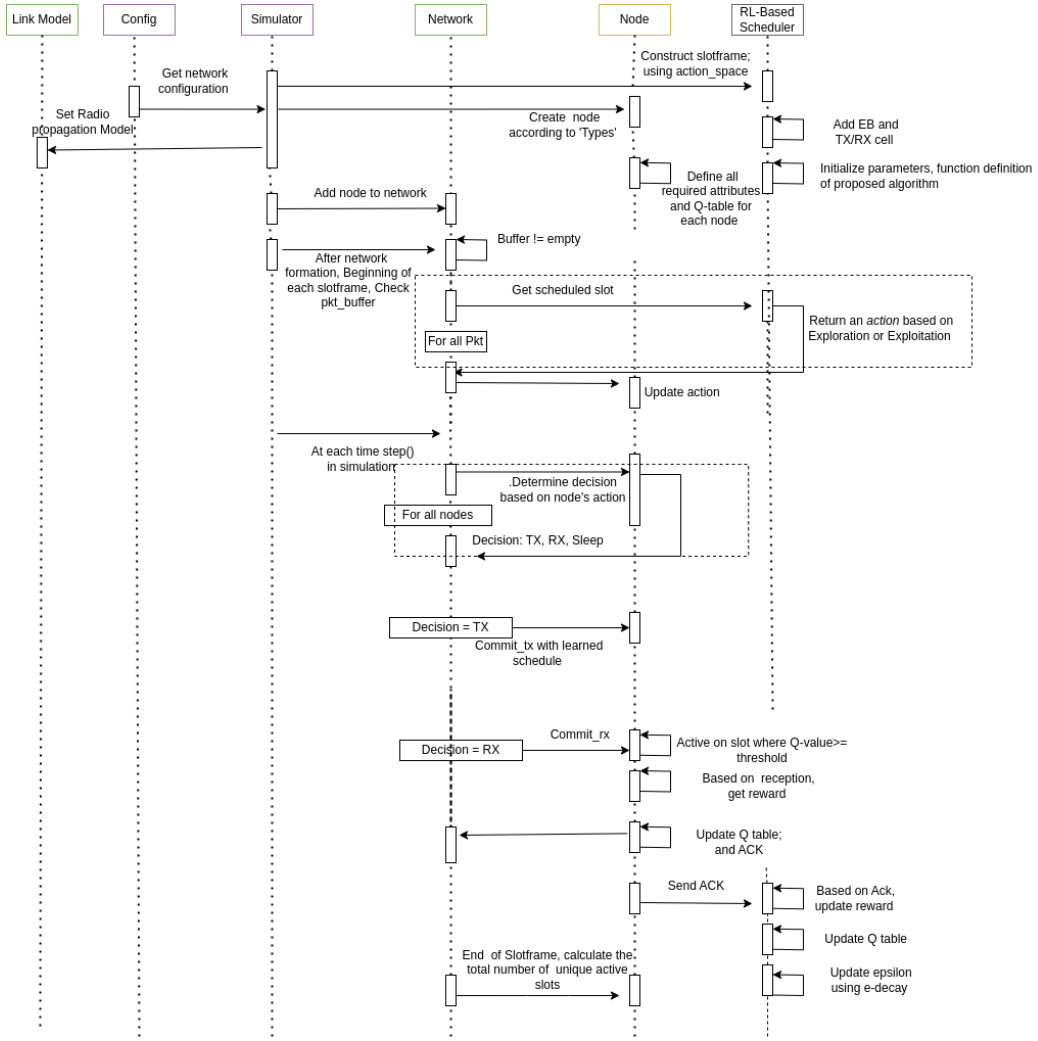


Figure 4.2: Sequence diagram representing the integration of the EARL algorithm within MAC and Network modules of the corresponding simulator

To complete the network formation phase, at the beginning of the simulation, each node randomly scans the channel to join the Time-Slotted Channel Hopping (TSCH) network, as discussed in section 3.1.4. Upon receiving an Enhanced Beacon (EB) from a neighboring node, the intended node joins the network, synchronizes its time with the parent node, and updates its routing table.

Once the network formalization is complete, each node starts generating packets according to the packet generation rate defined in the network configuration earlier. These packets contain

essential information such as Source, Destination, Packet Sequence Number, and Payload Size, and are stored in the packet buffer.

According to the proposed approach, at the starting of each slotframe cycle, the system checks if there is a packet in the buffer. If so, it invokes the Q-learning algorithm to schedule the packet for transmission. This periodic process is seamlessly implemented and integrated to align with the existing network module of the simulation flawlessly.

In order to schedule a packet, a node requires to select an available action from action space, which represents timeslots. This decision-making process is facilitated by the agent invoking the action-choosing function, which is implemented within the $Q\_learning\_scheduler.mjs$ module and aligned with the simulator's MAC and Network modules.

The Q-learning algorithm selects an action based on a defined policy, which is influenced by a probability value called $P_{exploitation}$. This probability is generated using a uniform random number generation approach with the aid of a seed and mask. The use of seed and mask ensures diversity and uniform distribution during generation. Based on the $P_{exploitation}$ value, the agent decides whether to pick a random timeslot within the frame or the action with the highest Q value. To achieve learning convergence, a gradient-decay approach is employed inside the $Q\_learning\_scheduler.mjs$ module.

Initially, the exploration value ϵ is set high at 0.8, meaning the agent is supposed to choose random actions 80% of the time and known actions only 20% of the time. However, this value gradually decreases with learning stability. Once the corresponding timeslot is chosen, the agent updates its action. After scheduling all packets in the buffer, each utilizing its corresponding node, the simulator executes the transmission and reception processes.

The time and ASN (Absolute Slot Number) values increase by one at each simulation step. During each step of the simulation, the proposed approach calculates the current timeslot using the present ASN value of the simulation and the defined action space. As mentioned in section 3.3.2, all timeslots are designed as shared, meaning a slot can be used for data transmission or reception at

any given time. For each timeslot, the proposed system iterates through each node and determines the role of a transmitter, receiver, or in sleep mode. During learning phase, at each timeslot, if a node's action matches the current timeslot, the node selects the corresponding cell for transmission or reception. The node then checks for a packet in that corresponding cell. If there is no packet present, the node assumes the role of RX (receiver) in the schedule, indicating that the node is in listening mode to receive a packet. If there is a packet, the node obtains it and acts as a transmitter (TX), indicating that the node has a packet to transmit during the scheduled timeslot.

After determining the schedule decision, the TX nodes transmit packets and await acknowledgment. According to the proposed algorithm, at the beginning, all the nodes have their radio ON on every timeslot for the entire frame. So, upon receiving a packet, if the packet is not destined for the present node, it simply rejects the packet. Conversely, a packet is received only by the intended node. In case a node receives two packets simultaneously from neighboring nodes, instead of losing both packets due to collision, the node accepts the packet with the higher Received Signal Indicator (RSSI) value.

Upon receiving an acknowledgment for the packet the transmitter (sender node) sent, a node determines the reward for the action taken. A true acknowledgment implies a positive reward, indicating that the targeted node successfully receives the packet and the taken action is suitable for scheduling data packets. Whereas a false acknowledgment denotes, a packet might be collided due to collision. Based on the acknowledgment, the estimated reward value serves as a definitive measure of the performance of the actions undertaken and the rescheduling process. According to the reward, the Q table is updated, and the epsilon value is degraded using the gradient decay approach at each epoch. When the learning convergence is reached, marking the onset of the transition phase, the activation of the node at a specific time slot is determined by the Q value, which is regulated by a predefined threshold.

4.2 Simulation Setup

The proposed approach is implemented and tested considering a discrete event simulator named TSCH-Sim, designed by [16]. It is a discrete event simulator that supports all major protocols. The simulator allows a configuration file of any network topology that needs to be validated. The configuration file supports all the crucial parameters required to construct the network.

In this simulator, the MAC layer is designed following the TSCH protocol. The Q-learning algorithm is tailored to MAC to allow the appropriate medium access among nodes. We considered both simple and large-scale networks to validate our algorithm. All the networks are heterogeneous, implying that the packet generation rate varies from node to node. For the radio propagation model, UDGM (Unit Disk Graph Model) has been considered.

For the experiments, two radio channels are considered for the channel hopping mechanism of the TSCH protocol. The TSCH-Sim simulator comes with a warm-up period, which signifies a duration during which packet generation is prohibited until the period expires. This provision guarantees that when it's time to generate a frame (as nodes generate frames at varying time intervals), the specific node must have already joined the network, as a frame cannot be generated by a node that has not yet joined. A warm-up period can be helpful (but not mandatory) to ensure accurate packet generation. However, the EARL algorithm can operate properly whether there is a warmup period activated or not. The proposed RL (Reinforcement Learning) approach schedule depends on the frame availability in the buffer and relies on only frame utilization to allocate timeslots.

For the experiment, the timeslot duration is considered as ten milliseconds, which means that during this period, a node transmits data and receives an acknowledgment based on the success or failure of the packet. The slotframe is comprised of a bunch of those timeslots. In this experiment, we experienced different slotframe lengths to see how effectively it generates an optimal schedule accounting for the latency and power consumption attributes.

4.2.1 RL Parameter Determination

This section discusses the parameter selection of the RL-based algorithm tailored with the reason behind choosing the suitable one. The subsequent sections provide a comprehensive elaboration on these aspects.

Learning Rate Analysis

The learning rate α plays a crucial role in tuning the speed at which the algorithm's learning converges. A higher $\alpha \in [0, 1]$ value leads to a quicker convergence of the Q value towards the reward r. Typically, α is set to a small value to provide some robustness in achieving a stable state during the learning process. We examined different learning rates, including 0.03, 0.95, and 0.5, and found that 0.03 offered the best convergence and greater stability for the protocol, as illustrated in Figure 4.3.

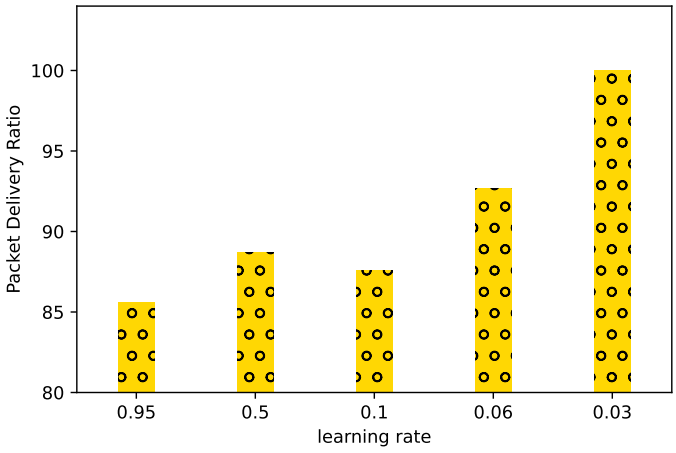


Figure 4.3: Learning rate Analysis

In the process of determining the learning rate value, we took into account scenario one. Another important parameter to consider is the frame length N. It's essential to ensure that N is sufficiently large to guarantee the availability of unique slots for nodes. If there aren't enough slots in a frame, nodes won't be able to find unique slots. However, if N is overestimated, it can introduce additional

latency and reduce the maximum achievable throughput. Our experiment explored three different frame sizes: 9, 15, and 25.

Exploration Probability Analysis

Several experiments were conducted to determine the optimal balance between exploration and exploitation probabilities. Both fixed values and a gradient descent approach that gradually reduces the exploration probability over time were tested. The experimental data and results indicate a clear superiority of the gradient method over the fixed exploration probability as illustrated in Figure 4.4. Consequently, based on these findings, the decision was made to utilize the gradient method as the

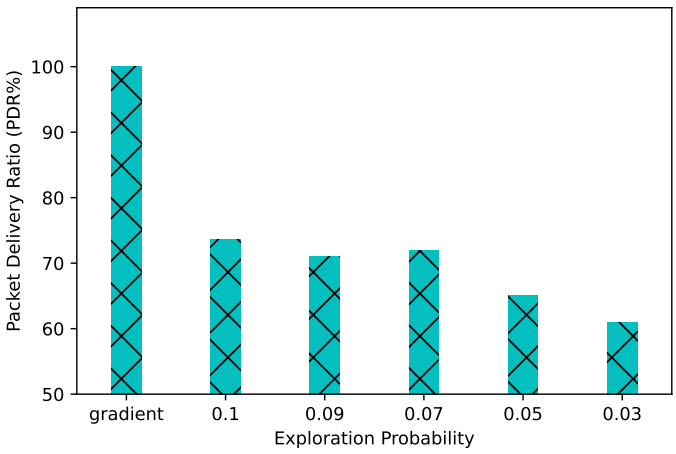


Figure 4.4: Exploration probability analysis

primary strategy in this experiment.

To obtain the optimal gradient, various values were explored. For instance, an exploration probability of 0.02 leads to increased exploration while causing higher energy usage.

An epsilon value of 0.8 with an epsilon_decay_ rate of 0.09, coupled with a learning rate of 0.03, demonstrated a promising equilibrium between exploration and exploitation. With each iteration, the exploration probability gradually decreases by gradient, facilitating a smooth convergence in the learning process. Consequently, it ensures high packet delivery while maintaining optimal energy

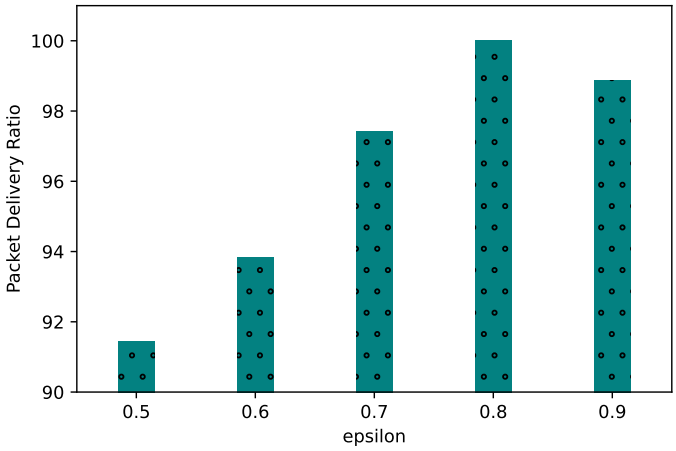


Figure 4.5: Epsilon Analysis

savings. The same topology (small-scale scenario) was used for this test. Figure 4.5 illustrates the analysis of epsilon with varying parameters.

Reward Evaluation

The reward is chosen based on successful packet transmission or failure. The positive reward is achieved only after getting successful reception of a packet or a True acknowledgment. On the contrary, a negative reward implies the failure of a data packet or False acknowledgment. A set of experiments has been conducted to find the one that suits better. Positive reward values of 1 and negative rewards of 0, -1, and -10 were tested. A combination of 1 and -1 achieved a better steady state of the proposed approach and tested through scenario one of seven nodes network.

It is experienced that a negative reward of -1 has a stronger influence on the current Q value when the Q value is positive, and vice versa. As a result, a time slot that consistently yields negative rewards is less likely to be the preferred choice. Consequently, nodes will actively seek timeslots that reliably provide positive rewards. This learning process guides the network towards an optimal steady state condition where all nodes occupy distinct timeslots. This behavior resembles that of a schedule-based network, but it eliminates the necessity for scheduling information exchange or the

determination of node priorities for each time slot. This is particularly crucial in Wireless Sensor Networks (WSNs) where centralized control is absent.

Chapter 5

ANALYSIS AND COMPARISON

This chapter presents the design of various network scenarios, ranging from simple configurations to more intricate, large-scale network architectures. We provide an overview of the distinctive features and specific requirements associated with each of these network designs. Furthermore, it discusses the evaluation metrics to compute the performance of the proposed approach and the parameter selection for the reinforcement learning approach.

Subsequently, we discuss the performance evaluation of the proposed algorithm within the context of each designed network scenario. This evaluation encompasses an analysis of simulation results, focusing on key metrics such as Packet Delivery Ratio (PDR), latency, and the number of active slots, which serve as an indicator of energy consumption.

Toward the conclusion of this chapter, we validate each scenario using the state-of-the-art Orchestra scheduling protocol. This validation process allows for a comprehensive comparative analysis, offering insights into the effectiveness of our proposed approach. Furthermore, within the concluding segments of this chapter, we analyze the Q values stability over time and provide a brief description of the computational method used to determine the average Packet Delivery Ratio (PDR) with its representation through a Boxplot diagram.

5.1 Scenario Design and Results Analysis of EARL algorithm

The application of monitoring finds widespread use in industrial settings, involving the placement of sensor nodes in a random fashion. These sensors transmit data towards a single designated destination, often routed through intermediary sensors along the way. This gives rise to a traffic pattern known as converge-cast, where the intensity of data traffic increases notably as the network scales up and the destination is approached. It is imperative to consider this phenomenon to meet the stringent requirements of reliability and energy efficiency in industrial contexts. This underscores the significance of employing a well-optimized Medium Access Control scheduler, which can effectively manage data flow and prevent issues like excessive queuing and packet loss. This thesis presents three distinct scenarios characterized by light to high node density and embedded within a heterogeneous network architecture.

Table 5.1 provides a comprehensive overview of three distinct network scenarios specified by unique attributes and requirements.

	Scenario 1	Scenario 2	Scenario 3
Network size	7	20	50
No. of neighbors	1.7	2	3
Avg. hop	1.5	2.2	3.1
Depth	2	3	5
Packet interval(s)	1, 2.5	1, 2, 3.5, 4.6, 7, 9, 12.5	1, 3, 4, 5.5, 7.5, 8, 9, 10.5, 12, 13.5,15

Table 5.1: Summary of three different network scenario designs

Scenario 1 is designed around a small-scale network configuration that necessitates short data intervals as depicted in Figure 5.1.

This scenario boasts an average hop count of 1.5 hops to reach the root node, with a maximum depth of 2 hops. Within this network, six nodes are actively transmitting packets, employing a range of data rates, including 1s and 2.5s. Notably, the average number of neighbors in this architecture

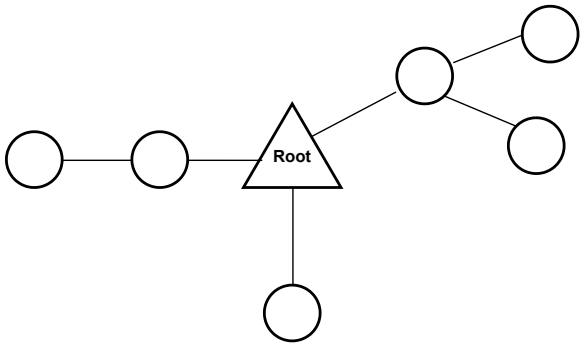


Figure 5.1: 7 nodes network architecture

stands at approximately 1.7, indicating a predominantly low level of node connectivity, typically fewer than two neighbors per node.

Scenario 2, in contrast, represents a network scenario tailored for high-traffic applications, demanding a multi-hop network infrastructure capable of handling substantial packet loads, illustrated in figure 5.2. On average, it requires 2.2 hops to reach the root node, with a maximum depth of

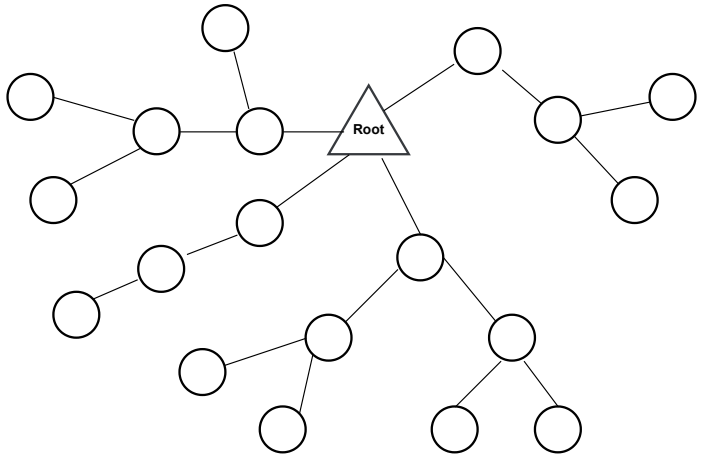


Figure 5.2: 20 nodes network architecture

three hops. In this configuration, nineteen nodes engage in packet generation, employing a diverse set of data rates ranging from 1s to 12.5s. This mix of data rates implies a combination of light and heavy traffic generation patterns. In this scenario, the average number of neighbors is approximately 1.9, indicating that each node maintains an average of two connections with other nodes, thereby

facilitating a medium-scale multi-hop network communication.

Scenario 3 is designed to enhance network scalability by incorporating a greater number of nodes, resulting in a more extensive multi-hop network structure. To transmit a packet to the root node, an average of three hop counts is required, and the depth of this network architecture reaches up to five hops. This scenario involves the participation of forty-nine nodes, each generating packets at varying data rates, as detailed in the accompanying table. Furthermore, this network architecture exhibits an average number of more than two neighboring nodes, thus establishing a framework for a large-scale, multiple multi-hop network configuration.

A set of Experiments has been conducted to evaluate the EARL based TSCH MAC protocol under varying traffic rates. Table 5.2 presents the parameters associated with the general situation, particularly highlighting the Q-learning-based MAC approach.

Parameter	**Scenario 1**	**Scenario 2**	**Scenario 3**
Simulation duration	800	800	1600 sec
Nodes	7	20	50
Slotframe size	9, 15	25	25
Warm-up period	100	200	500 sec
Radio Model	UDGM	UDGM	UDGM
Packet size	22 bytes	22 bytes	22 bytes
Max re-transmission	7	7	7
Number of simulations	10	10	10
α	0.03	0.03	0.03
γ	0.95	0.95	0.95

Table 5.2: Simulation and Q-learning parameters

Regarding the scenario details, a configuration with seven nodes was deployed in a compact setting, while a medium and larger-scale scenario with the multi-hop involving 20 and 50 nodes was randomly placed. For each distinct configuration, the nodes transmit messages containing a consistent 22-byte payload, enabling a transmission range of approximately 40 and 100 meters.We evaluated the maximum number of MAC re-transmissions as seven attempts. A buffer size of both

16 and 100 has been utilized across multiple network scenarios. This buffer size implies the maximum number of outgoing packets toward each neighboring node. This value also indicates that a queue overflow can occur above this number due to exceeding the scheduling of the slotframe. To prevent the repetition of generating identical schedules and ensure the efficacy and efficiency of the proposed algorithm, each scenario is simulated with a random seed generation and distinct values assigned to the $Number\_of\_Run$ parameters.

The proposed approach operates in a decentralized manner, which implies avoiding signaling overhead among agents. Nodes primarily focus on their actions and rewards to create their optimum schedule. This approach remains stable even when the number of nodes increases, maintaining a consistent packet delivery rate while minimizing energy consumption. This is achieved through a dynamic learning-based radio wake-up scheme, which balances the trade-off between packet delivery rate (PDR) and power usage.

The experiments consider slotframe sizes of 9, 15, and 25 to test various network scenarios. The size of each agent's action space is determined by the number of slots within a slotframe. Generally, a larger action space leads to slower convergence of the reinforcement learning (RL) algorithm, as demonstrated by Leng et al. [73]. Conversely, a small action space may result in suboptimal solutions and increase energy consumption in the system. Determining the right number of time slots within a frame is a topic for future research. However, the appropriate values for these slotframes are determined through simulation experiments and by following related RL-based experiments conducted by Park et al. [35].

At the beginning of the simulation, all slots are continuously active until the nodes get enough experience with taken actions and come together towards the best radio schedule for saving energy. After the transition phase, nodes turn the Radio ON only based on the Q table value determined by the threshold. To identify the optimal transition phase, a series of periods are explored and tested across three distinct network scenarios. Upon evaluating the results of these experiments, it has been determined that a transition phase (TP), calculated using the equation provided below, yields

superior learning convergence.

$$TP = (TotalSimulation\,time - Warmup\,period) * 30\% \tag{5.1}$$

Power consumption is quantified in relation to the total number of active slots within a slotframe. A critical aspect of our configuration revolves around determining the appropriate threshold value. This value plays a crucial role in slot activation and significantly influences the overall performance of the protocol, particularly when the number of nodes within the network changes. Consequently, we have thoroughly examined threshold values, which have been segmented into three distinct analyses: one for scenarios involving small as presented in Table 5.3, and another two for medium and large scale configuration, as depicted in Table 5.4, 5.5. These analyses allow us to gain deeper insights into how varying threshold values impact the protocol's performance under different network conditions, offering a more comprehensive understanding of the system's behavior.

5.1.1 Evaluation Metrics

The performance of the proposed RL-based MAC scheduling approach has been assessed in various scenarios, including both small and large-scale settings. Our evaluation primarily centered on three key performance metrics: the packet delivery ratio (PDR), energy consumption, and end-to-end delay. PDR serves as a critical indicator of protocol efficiency in any network, as it offers valuable insights into the protocol's effectiveness by considering its impact on power consumption and, consequently, the longevity of network nodes.

Measuring latency is a vital aspect of assessing network performance, as it directly influences the efficiency and reliability of data transmission. Delays in packet delivery can result in packets arriving out of sequence or exceeding their allowable transmission time, ultimately diminishing the PDR. Conversely, lower latency tends to enhance PDR by ensuring timely and accurate delivery of packets, reducing the likelihood of congestion and packet loss in the network. The calculation of

these metrics followed to the formulas outlined below.

- **Packet Delivery Ratio (PDR)**: The packet delivery ratio essentially quantifies the efficiency with which data packets are successfully delivered from a source node to a destination node within a network. It can be calculated by comparing the number of packets successfully delivered to their destination with the total number of packets sent. A higher PDR indicates better network performance and reliability. The PDR is calculated using the following formula,

$$PDR = \frac{\sum_{i=1}^{N} packet\ successfully\ received}{Total\ number\ of\ Packet\ sent} * 100 \tag{5.2}$$

- **Power Consumption**: The lower power consumption is achieved by employing unnecessary listening slots to sleep mode. The proposed approach determines the number of slots active during the communication. Each timeslot is shared, and hence, multiple nodes can utilize a single slot. The system, therefore, undertakes the task of estimating the number of distinct slots that remain active within a slotframe. Then, the average number of active slots is computed using the equation provided below:

$$Avg\ slots\ on = \frac{\sum_{i=1}^{N} Number\ of\ slots\ on\ in\ sloframes}{Total\ Number\ of\ slotframes} \tag{5.3}$$

- **Latency**: Network delay refers to the total time (propagation, transmission, queuing, and processing period) a packet takes to travel from a source node to a destination node. It is estimated in seconds. Hence, the delay is evaluated by taking the difference between the time a packet is generated and is successfully received by the destination. The average delay has been estimated utilizing the below equation,

$$Delay = \frac{\sum_{i=1}^{N} (time(i)_{received} - time(i)_{generated})}{Total\ packets} \tag{5.4}$$

The execution of a run is as follows:

- Warm-up-time: until this time all the nodes boot.
- Warm-up + random value: nodes start transmission towards root using learned schedule.
- X sec: Start measuring metrics.

here, X defines the time to wait for the convergence of the learning. The value of X depends on the network architecture. These timings and settings are described in more detail in section 5.1.2 below.

5.1.2 Scenario One Results

We considered a tree topology of seven nodes in a small-scale scenario. Here, node one is the root node, two nodes are assigned as intermediate, and the rest are leaf nodes. Leaf nodes are capable of only transmitting data. On the other hand, intermediate nodes are designed to both send and receive data. More specifically, these nodes can generate packets and transmit their packet as well as packets of their child node.

As mentioned in the above section 5.1, we considered heterogeneous network architecture to validate our proposed approach, which implies that nodes generate packets at different time frames. Each node transmits 22-byte packet, maintaining the UDGM radio model with a range of forty meters. A slotframe size of 15 timeslots has been used. Table 5.3 reports the threshold analysis for the small-scale scenario.

Threshold	0.4	0.8	0.9
PDR	100%	93.63%	76%
Slots ON	20.01%	18.56%	14.86%

Table 5.3: Scenario one Threshold analysis with respect to PDR and Slots On

Selecting the optimal threshold value in this context holds significant importance due to its direct influence on conserving energy through radio on-off activity. The key factor in determining

when to turn off the radio relies most on this chosen threshold value. While diverse scenarios were explored using varying thresholds, the focus here is on the most relevant outcomes.

For this small scaenario, we've configured the learning rate (α) to be 0.03, the exploration probability (ϵ) to be 0.8, and an epsilonDecayRate of 0.09. The determination of packet delivery and the total count of active slots takes place following the transition phase.

It can be seen that setting the threshold at 0.4 results in a higher packet delivery ratio (PDR). This optimal scheduling involves an average of 3 active slots. Figure 5.3 depicts the percentage of packet delivery ratio with respect to the total number of active slots for this network architecture. After a transition phase, all the nodes within the system adeptly converge toward an optimal wake-up schedule. Once established, this optimal radio schedule becomes persistent over time, effectively preserving energy caused by idle listening on the receiving side.

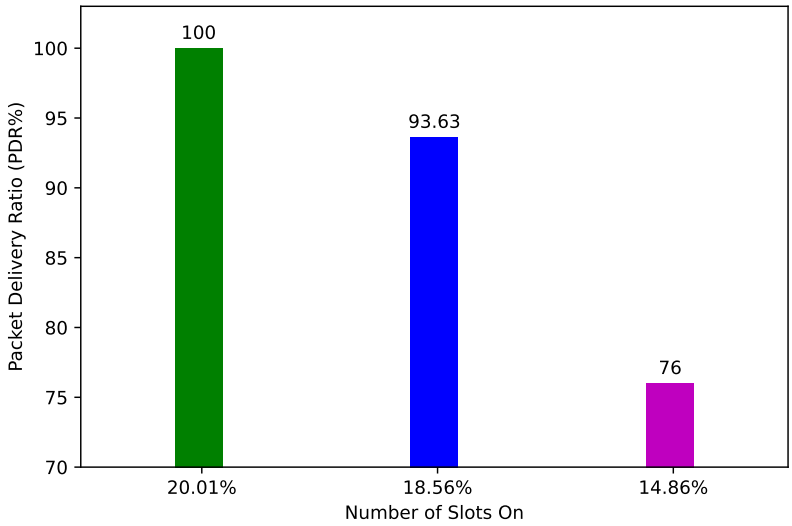


Figure 5.3: Scenario one PDR vs Slot_on

5.1.3 Scenario Two Results

A network architecture comprising 20 nodes has been deployed on a medium scale in a multi-hop interconnected manner. Within this topology, 19 nodes are designated as the source, while one node

is the sink node. The distance between nodes is set at 40 meters. A frame length of 25 slots has been chosen, where each of these is 10 milliseconds to accommodate the packet delivery to the destined node.

The simulation period is chosen as 800 seconds. The warm-up period is considered as 200 seconds since the simulation started. This warm-up phase signifies the completion of network formation, during which all nodes successfully join the TSCH network. The action space is equivalent to the frame length. In the initial phase, all nodes maintain their radios in an active state, listening to every slot until they enter the transition phase. The transition phase is estimated using the equation 5.1. According to the formula, the transition phase is set as 380 seconds for this topology.

In order to find an optimal balance between packet delivery ratio (PDR) and power dissipation, we perform simulations using various threshold values. Each test case employs a learning rate of 0.03.

Threshold	0.4	0.6	0.7	0.8	0.9
PDR	98.1%	90.79%	80.20%	71%	59.96%
Slots ON	14%	12.83%	11.94%	11.61%	9.04%

Table 5.4: 20 nodes architecture Threshold analysis with respect to PDR and Slots On

Table 5.4, depicts the data for the packet delivery ratio and the total number of active slots. It can be seen that the most favorable balance between PDR and energy consumption is achieved when employing a threshold value of 0.4. As indicated in the table, configuring the threshold at 0.7 yields intermediate results, while a value of 0.9 results in higher average packet loss and standard deviations. The results obtained highlight that when prioritizing reduced energy consumption, there is a slight reduction in the PDR value. As a result, in a medium-scale network with 20 nodes with a multi-hop heterogeneous configuration, an optimal scheduling configuration necessitates an average of 3.5 slots where the radio remains active due to multiple nodes participating in communication at various intervals. Figure 5.4 depicts the percentage of PDR with respect to the total number of slots active at that time.

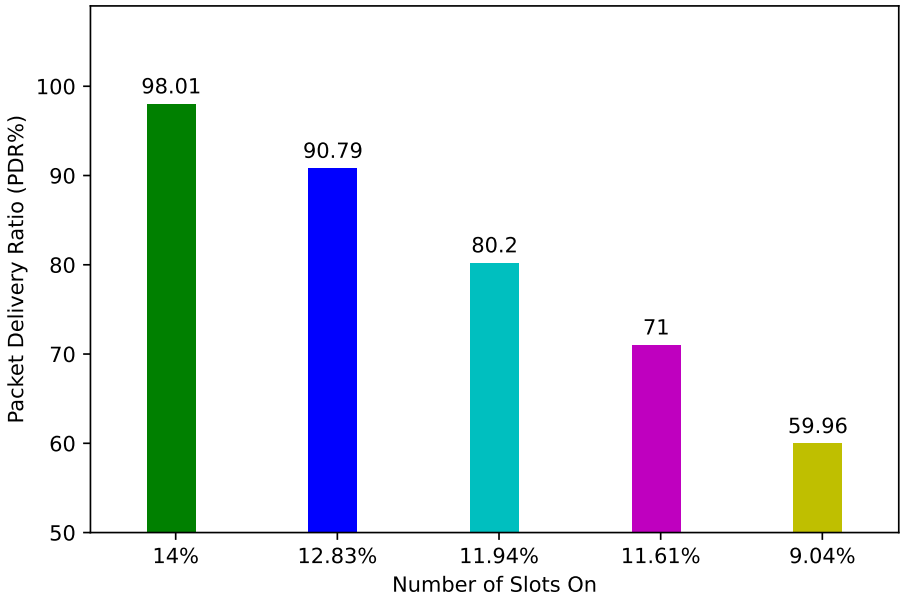


Figure 5.4: 20 nodes architecture PDR vs Slot_on

5.1.4 Scenario Three Results

To assess the scalability of the proposed approach, we conducted an evaluation in a significantly more heterogeneous and densely populated network consisting of 50 nodes. Similar to the other two network topologies, this network features a designated root node, with the remaining nodes serving as source nodes. A slotframe size of 25 slots is considered to facilitate communication and coordination. As the network is denser compared to the previous one, it demands more time to complete the network joining process. To achieve this, a simulation duration of 1600 seconds has been selected, with a warm-up period extending to 500 seconds. The transition phase duration is computed using equation 5.1 and is determined to be 830 seconds.

Table 5.5 presents the results for packet delivery and energy consumption across various threshold values.

According to the table, a threshold value of 0.4 ensures an optimal balance between packet delivery ratio (PDR) and power consumption. Furthermore, the table illustrates that selecting a

Threshold	0.4	0.6	0.7	0.8	0.9
PDR	92.73%	81.30%	76.60%	70.01%	58%
Slots ON	16.53%	13.18%	11.84%	11.56%	9.87%

Table 5.5: 50 nodes architecture Threshold analysis with respect to PDR and Slots On

threshold of 0.7 deactivates more slots to conserve energy while still maintaining a moderate packet delivery performance. Conversely, a threshold value of 0.9 results in a higher average packet loss rate.

In contrast to the previous medium scale populated network, there is a slight increase in the number of active slots required to accommodate sufficient transmissions in this network. As indicated in the table, an optimal schedule necessitates an average of four active slots to achieve improved packet delivery performance. Figure 5.5 depicts the packet delivery ratio vs number of slots on at

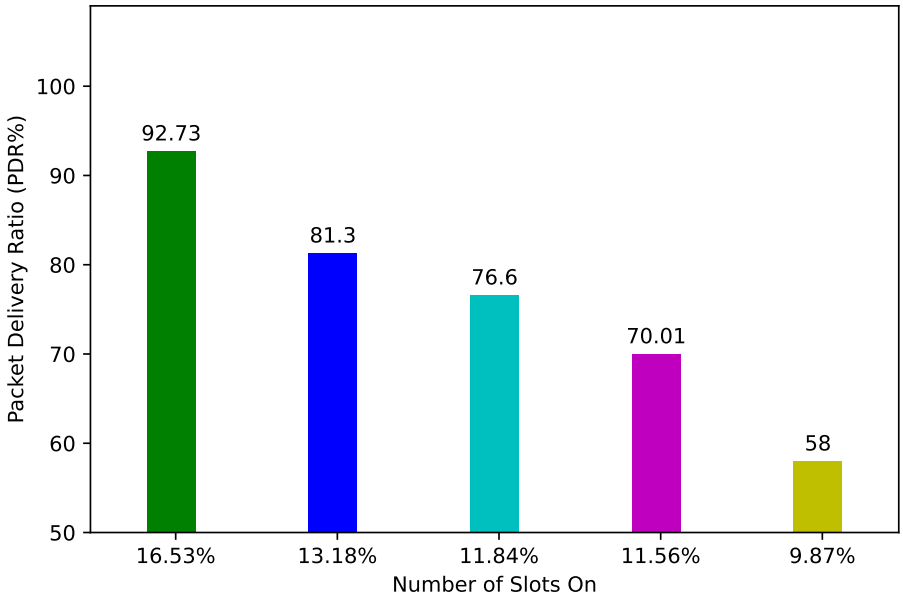


Figure 5.5: 50 nodes architecture PDR vs Slot_on

that time.

In summary, it is noteworthy that the proposed EARL based approach exhibits remarkable performance in three distinct scenarios. Specifically, it excels in achieving an optimal Packet Delivery Ratio (PDR) while efficiently reducing the radio's active state duration. This dual achievement leads to the effective optimization of energy consumption within individual nodes, which, in turn, contributes significantly to prolonging the overall lifespan of the network.

In essence, this approach's versatility and effectiveness make it a valuable solution across a range of network scales, ensuring superior PDR and sustainable and efficient energy utilization. This combination of benefits holds the potential to enhance the longevity and reliability of wireless networks under various operational conditions.

5.1.5 Orchestra Scheduling Result Analysis

To ensure a fair comparison, all three designed scenarios have been validated using the Orchestra scheduling approach. The Orchestra scheduling technique operates as an autonomous scheduler, where individual nodes independently generate schedules without engaging in negotiation with neighboring nodes [74]. This method proves particularly effective for systems demanding robust packet reliability. Each node autonomously determines its schedule in this methodology based on routing information, leveraging the RPL [75] (IPv6 Routing Protocol for Low Power and Lossy Networks) routing protocol. The Orchestra schedule encompasses distinct slotframe of varying lengths, each designated for specific traffic types: Application data, TSCH enhanced beacon (EB), and RPL traffic. A prioritization scheme is employed, wherein the lowest-priority slotframe serves application traffic, and the highest-priority slotframe serves TSCH traffic.

For validation within our small-scale scenario, slotframe sizes of 15, 31, and 397 were assigned to the application, RPL signaling, and TSCH beacon, respectively. All other configuration parameters, including node type, data rate, and radio propagation model, were maintained consistently. Each experiment was conducted five times, setting the num_of_run parameter to ten with random seed generation employed. The reported results are based on the average outcomes.

Upon conducting simulations for a duration of 800 seconds, it was observed that it achieved 100% packet delivery utilizing 22% of active slots. This result is very close to our proposed approach, providing a high reliability of 100% while maintaining different traffic rates.

The second scenario with 20 nodes network configuration was also verified using the Orchestra protocol, with a unicast slotframe size of 25. All other parameters remained constant to ensure a fair comparison between the proposed and this algorithm. Orchestra scheduling exhibited a high Packet Delivery Ratio (PDR) of 100% across various data rates. However, there was a marginal increase in power consumption. This is attributed to Orchestra's design, which aims to provide superior reliability even in contention-based scenarios, necessitating additional resources.

This can be explained by the fact that for each node, one slot for sending and one for receiving, independently from the application scenario, along with dedicated slots for EBs transmission, which minimizes the chances of collisions. Consequently, the number of active slots increased by 21% in the 20-node scenario.

For the third scenario with more extensive network architecture, we further utilized the orchestra scheduling algorithm to the test, keeping all other network parameters unchanged. Once more, we opted for a unicast slotframe size of 25. The findings indicate that as network density increases, the algorithm's reliability diminishes. While employing Orchestra, we observed a Packet Delivery Ratio (PDR) of 74.056%, utilizing 28% of its active slots. This might be because sparse networks encounter fewer collisions than dense networks, where a substantial number of neighbors can lead to increased traffic congestion. This additional energy consumption might be caused by the additional number of nodes requiring more resources in a densely populated area. Table 5.6 overviews the results of the Orchestra schedule, presenting data on the Packet Delivery Ratio (PDR) and the count of active slots across three distinct scenario designs.

	Scenario One	Scenario Two	Scenario Three
PDR	100%	100%	74.06%
Slots ON	22%	21%	28%

Table 5.6: PDR and Slots On Analysis of Orchestra for different scenarios

5.2 TSCH Scheduling Approach Comparison

Figure 5.6 illustrates the packet delivery ratio (PDR) for different network architectures utilizing different approaches. In a simple scale scenario, the proposed EARL-based scheduler, implemented

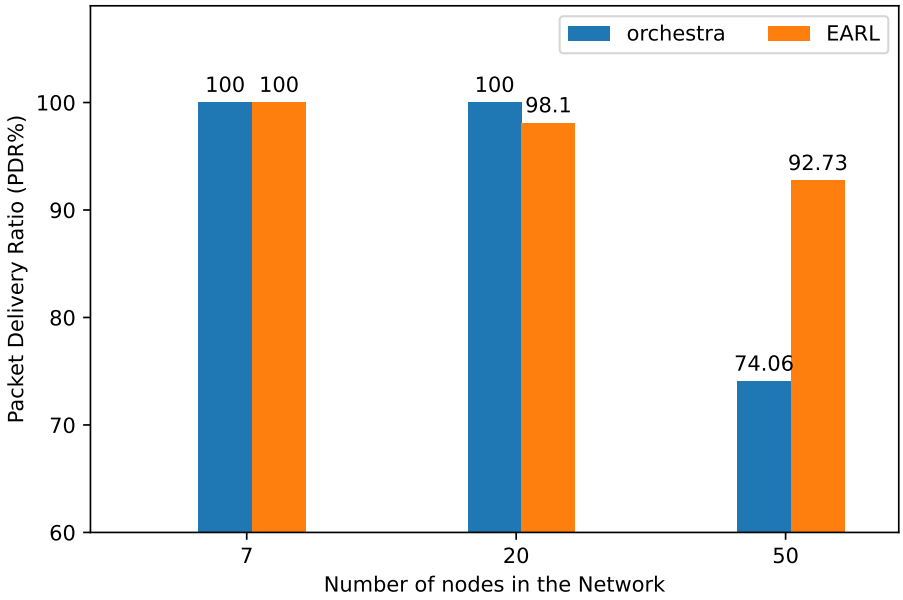


Figure 5.6: Comparison of packet delivery ratio between TSCH scheduler

with a 15-slot slotframe, achieved an impressive packet delivery ratio (PDR) of 100%. In comparison, the state-of-the-art scheduler Orchestra also attained a 100% PDR, with the only notable difference being a slight increase in the total number of active slots, approximately 22%.

In a medium scale network scenario, where were 20 nodes and the slotframe size was set to 25 slots, the proposed EARL approach yielded an impressive PDR of up to 98.122%. Orchestra, due to its collision-free operation, maintained a 100% PDR. However, in this case, the scheduler utilized

around 21% of its active slots, while our method achieved a PDR close to similarity but with a lower percentage of active slots (14%), demonstrating our emphasis on energy conservation alongside a satisfactory packet delivery ratio.

In a more challenging scenario with 50 nodes and a slotframe size of 25 slots, the EARL-based scheduler showcased superior performance, achieving a PDR of 92.738%, in contrast to Orchestra's PDR of up to 74.06%. Notably, Orchestra utilized 28% of its active slots, nearly double the percentage of active slots used by the EARL-based scheduler (16.53%). These results underscore the limitations of Orchestra, particularly in scenarios characterized by higher heterogeneous traffic rates and multiple multi-hop connections. It further highlights the robustness of our proposed learning-based approach, which excels in high-traffic environments, maintains balanced energy consumption and ensures a superior packet delivery ratio. As illustrated in Figure 5.7, the average packet latency

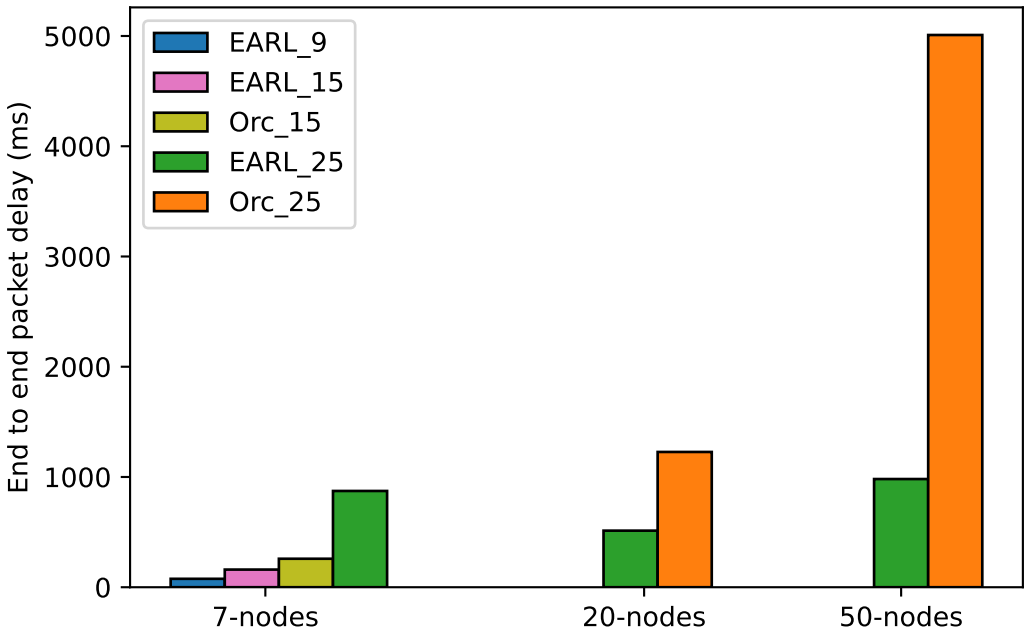


Figure 5.7: Comparison of average packet delay

achieved by the EARL-based method is consistently shorter than that of the Orchestra scheduler across all scenarios.

The extended packet delay observed with the Orchestra scheduler can be attributed to the increased time it takes to route packets to their final destinations, particularly in scenarios involving multiple multi-hop connections. This difference in packet delay is most pronounced in the 50-node network architecture, which represents a five-hop multi-hop environment. In this case, the RL-based scheduler demonstrated an average packet delay of 981.33 milliseconds, in sharp contrast to the 5009 milliseconds delay recorded with the Orchestra scheduler.

5.3 Stability Analysis of Q value

We conducted an analysis of Q value stability over time. Figure 5.8 illustrates the stable condition of a node within a 15-slot timeframe. The learning phase commences at approximately 100 seconds

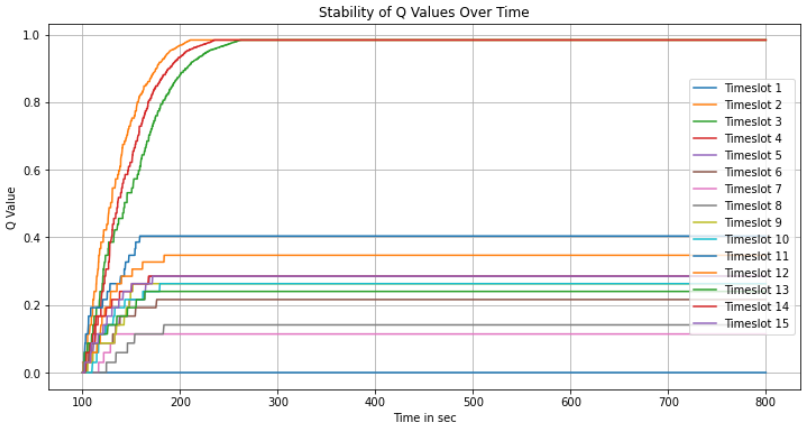


Figure 5.8: Stability analysis of Q value within 15 slots

and approaches stability at around 250 seconds. In this instance, the learning-up period is configured for 300 seconds. From the graph, it is evident that initially, the system explores various timeslots extensively. As it gains experience through a trial-and-error approach, it gradually stabilizes by selecting the top three slots.

Similarly, Figure 5.9 illustrates the stability of the Q value for a node across 25 slots. Here, the learning phase initiates around 500 seconds, and by 600 seconds, the system is on the verge of

stability.

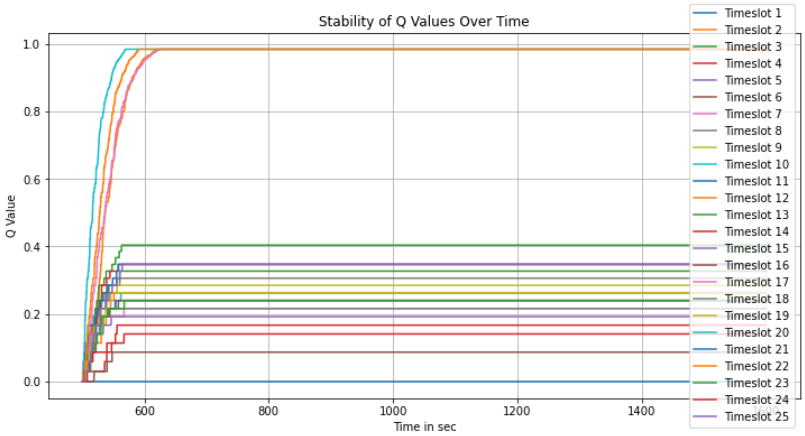


Figure 5.9: Stability analysis of Q value within 25 slots

5.4 Packet Delivery Ratio (PDR) Analysis

Figure 5.10 serves as a visual representation depicting the distribution of Packet Delivery Ratio (PDR%) values across three distinct scenarios. The x-axis represents the scenarios, while the y-axis quantifies the PDR values. We computed the mean of PDR based on the ten simulations of each scenario design. Scenario One, within the framework of this boxplot analysis, exhibits a tightly clustered distribution of Packet Delivery Ratio values, with a mean of 100%. The interquartile range suggests that most data points fall within a narrow range, as the minimum value is 98%, indicating excellent PDR performance.

Moving to Scenario Two, the mean is slightly lower at 98%, but the distribution is slightly wider, ranging from a minimum of 93% to a maximum of 99.99%.

Scenario Three, with a mean of 92%, presents a broader distribution from a minimum of 84% and highlights a distinct range of PDR values.

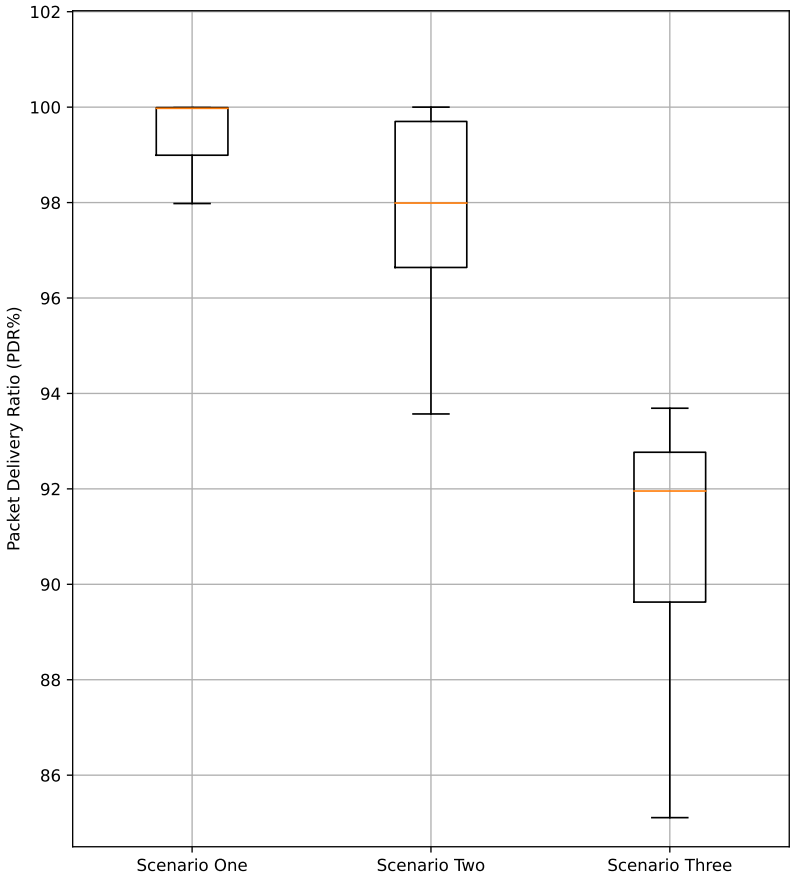


Figure 5.10: Average PDR estimation within scenarios

Chapter 6

CONCLUSION

In conclusion, this paper addresses the medium access scheduling issue of IEEE 802.15.4e TSCH protocol, widely utilized by fourth industrial IoT applications. While state-of-art techniques have been provided various solutions, their performance can be limited when dealing with network metrics such as scalability and adaptability to certain changes in architecture. A reinforcement learning-based approach has been proposed to address these challenges and enhance the network performance.

This thesis presents an energy-aware adaptive scheduling scheme for TSCH networks, that leverages with MAC layer and provides solutions in real time based on the environment. Due to its reactive nature, the proposed algorithm requires very few knowledge and hence reduces the signal heading and computation complexity. The approach is robust to any topology change and dynamically handles the radio on-off activities to conserve power while ensuring high reliability.

The experimental results show that the proposed RL-based algorithm can improve power consumption even when dealing with high-density communication with heterogeneous traffic features. The superiority of learning-capable nodes is illustrated by a comparative study with state-of-art orchestra protocol. Based on the results of the experiments, it can be deduced that schedulers utilizing Reinforcement Learning (RL) have demonstrated their ability to efficiently adjust resource usage according to the needs of the application, ultimately leading to energy conservation.

Furthermore, the suggested approach ensures a reduction in the signaling overhead typically associated with other autonomous scheduling methods. Each node independently creates schedules in a decentralized fashion, eliminating the need for dedicated node communication. Additionally, it helps conserve energy by minimizing communication requirements.

The frame size represents a critical parameter in our protocols, employing a significant impact on network performance, including factors like latency and energy efficiency. It's worth noting that nodes have a specific allocation of slots within each frame. The total number of slots in each frame is predetermined, aiming to avoid collisions in cases of excessively short frame sizes while also minimizing additional delays and throughput reduction associated with overly large frame sizes. The ideal frame size is influenced by various factors.

In our simulation, we determined the optimal frame size through a series of extensive experiments and by considering related work in our study. One potential avenue for future research lies in dynamically determining more precise frame size, eliminating the need for prior frame size estimates in the protocol.

The proposed research work is more focused on putting the idle listening slot in sleep mode. The number of active slots directly correlates with energy consumption because the more active slots you have, the more energy the network consumes. Focusing on the number of active slots provides a clear and straightforward metric for energy conservation, making it easier to demonstrate the effectiveness of the proposed technique in conserving energy. So far, the proposed approach has not utilized any energy model to compute network lifetime. As future research, we would like to integrate an energy model.

The experiment was carried out within a discrete event simulation framework, specifically using TSCH-Sim (Elsts 2020). This approach proves advantageous when access to a physical testbed is challenging. Future steps in this research involve conducting the experiment within the IoT-LAB environment, allowing for the observation of real-world Wireless Sensor Network (WSN) challenges, including issues such as interference, multi-path fading, and precise energy consumption

assessment.

Bibliography

[1] M. Majid, S. Habib, A. R. Javed, *et al.*, "Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review," *Sensors*, vol. 22, no. 6, p. 2087, 2022.

[2] A. Ali, Y. Ming, S. Chakraborty, and S. Iram, "A comprehensive survey on real-time applications of wsn," *Future internet*, vol. 9, no. 4, p. 77, 2017.

[3] M. Kocakulak and I. Butun, "An overview of wireless sensor networks towards internet of things," in *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)*, Ieee, 2017, pp. 1–6.

[4] A. Irandoost, S. Taheri, and A. Movaghar, "Pl-mac: Prolonging network lifetime with a mac layer approach in wireless sensor networks," in *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, 2008, pp. 109–114. DOI: 10.1109/SENSORCOMM.2008.120.

[5] W. Specification, "Wirelesshart specification 75: Tdma data-link layer," *HART Communication Foundation Std*, 2008.

[6] I. ISA, "100.11 a-2009: Wireless systems for industrial automation: Process control and related applications," *International Society of Automation: Research Triangle Park, NC, USA*, 2009.

[7] P. Bhagwat, "Bluetooth: Technology for short-range wireless apps," *IEEE Internet Computing*, vol. 5, no. 3, pp. 96–103, 2001.

[8] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, "Study on zigbee technology," in *2011 3rd international conference on electronics computer technology*, IEEE, vol. 6, 2011, pp. 297–301.

[9] I. C. S. L. S. Committee *et al.*, "Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11ˆ*, 2007.

[10] G. Anastasi, M. Conti, and M. Di Francesco, "A comprehensive analysis of the mac unreliability problem in ieee 802.15.4 wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, pp. 52–65, 2011. DOI: 10.1109/TII.2010.2085440.

[11] "Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 1: Mac sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, 2012. DOI: 10.1109/IEEESTD.2012.6185525.

[12] D. De Guglielmo, S. Brienza, and G. Anastasi, "Ieee 802.15. 4e: A survey," *Computer Communications*, vol. 88, pp. 1–24, 2016.

[13] D. Chen, M. Nixon, and A. W. Mok, "Real-time mesh network for industrial automation," *Cham, Switzerland: Springer*, pp. 1–6, 2010.

[14] M. Nobre, I. Silva, and L. A. Guedes, "Routing and scheduling algorithms for wirelesshart networks: A survey," *Sensors*, vol. 15, no. 5, pp. 9703–9740, 2015.

[15] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne, "6tisch minimal scheduling function (msf), document, draft-chang-6tisch-msf," IETF, Internet Draft, Tech. Rep., 2019.

[16] A. Elsts, "Tsch-sim: Scaling up simulations of tsch and 6tisch networks," *Sensors*, vol. 20, no. 19, p. 5663, 2020.

[17] A. Farrel, J.-P. Vasseur, and J. Ash, "A path computation element (pce)-based architecture," Tech. Rep., 2006.

[18] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15. 4e networks," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC)*, IEEE, 2012, pp. 327–332.

[19] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for ieee 802.15. 4e tsch based industrial low power wireless networks," in *2016 IEEE Wireless Communications and Networking Conference*, IEEE, 2016, pp. 1–6.

[20] R. Soua, P. Minet, and E. Livolant, "Modesa: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks," in *2012 IEEE 31st international performance computing and communications conference (IPCCC)*, IEEE, 2012, pp. 91–100.

[21] R. Soua, E. Livolant, and P. Minet, "Musika: A multichannel multi-sink data gathering algorithm in wireless sensor networks," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2013, pp. 1370–1375.

[22] A. Tinka, T. Watteyne, and K. Pister, "A decentralized scheduling algorithm for time synchronized channel hopping," in *Ad Hoc Networks: Second International Conference, ADHOCNETS 2010, Victoria, BC, Canada, August 18-20, 2010, Revised Selected Papers 2*, Springer, 2010, pp. 201–216.

[23] W.-P. Wang and R.-H. Hwang, "A distributed scheduling algorithm for ieee 802.15. 4e networks," in *2015 IEEE International Conference on Smart City/SocialCom/-SustainCom (SmartCity)*, IEEE, 2015, pp. 95–100.

[24] R. Soua, P. Minet, and E. Livolant, "Wave: A distributed scheduling algorithm for convergecast in ieee 802.15. 4e tsch networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 4, pp. 557–575, 2016.

[25] R. Soua, P. Minet, and E. Livolant, "Disca: A distributed scheduling for convergecast in multichannel wireless sensor networks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 156–164. DOI: 10.1109/INM.2015.7140288.

[26] S. Kim, H.-S. Kim, and C. Kim, "Alice: Autonomous link-based cell scheduling for tsch," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, 2019, pp. 121–132.

[27] A. Elsts, S. Kim, H.-S. Kim, and C. Kim, "An empirical survey of autonomous scheduling methods for tsch," *IEEE Access*, vol. 8, pp. 67 147–67 165, 2020. DOI: 10.1109/ACCESS.2020.2980119.

[28] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "Ost: On-demand tsch scheduling with traffic-awareness," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, IEEE, 2020, pp. 69–78.

[29] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne, "6tisch minimal scheduling function (msf)," *Internet Engineering Task Force, Internet-Draft draft-ietf-6tischmsf-02*, 2019.

[30] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "An evaluation of the 6tisch distributed resource management mode," *ACM Transactions on Internet of Things*, vol. 1, no. 4, pp. 1–31, 2020.

[31] M. R. Palattella, T. Watteyne, Q. Wang, *et al.*, "On-the-fly bandwidth reservation for 6tisch wireless industrial networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, 2015.

[32] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled tsch," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 337–350.

[33] G. Neto, "From single-agent to multi-agent reinforcement learning: Foundational concepts and methods," *Learning theory course*, vol. 2, 2005.

[34] H. Nguyen-Duy, T. Ngo-Quynh, F. Kojima, T. Pham-Van, T. Nguyen-Duc, and S. Luongoudon, "Rl-tsch: A reinforcement learning algorithm for radio scheduling in tsch 802.15. 4e," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2019, pp. 227–231.

[35] C. Savaglio, P. Pace, G. Aloi, A. Liotta, and G. Fortino, "Lightweight reinforcement learning for energy efficient communications in wireless sensor networks," *IEEE Access*, vol. 7, pp. 29 355–29 364, 2019.

[36] H. Park, H. Kim, S.-T. Kim, and P. Mah, "Multi-agent reinforcement-learning-based time-slotted channel hopping medium access control scheduling scheme," *IEEE Access*, vol. 8, pp. 139 727–139 736, 2020.

[37] H. Park, H. Kim, K. T. Kim, S.-T. Kim, and P. Mah, "Frame-type-aware static time slotted channel hopping scheduling scheme for large-scale smart metering networks," *IEEE Access*, vol. 7, pp. 2200–2209, 2018.

[38] Y. H. Pratama and S. Chung, "Rl-sf: Reinforcement learning based scheduling function for distributed tsch networks," in *2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, IEEE, 2022, pp. 5–8.

[39] M. Mihaylov, Y.-A. Le Borgne, K. Tuyls, and A. Nowé, "Decentralised reinforcement learning for energy-efficient scheduling in wireless sensor networks," *International Journal of Communication Networks and Distributed Systems*, vol. 9, no. 3-4, pp. 207–224, 2012.

[40] O. Yang and W. Heinzelman, "Modeling and performance analysis for duty-cycled mac protocols with applications to s-mac and x-mac," *IEEE Transactions on Mobile Computing*, vol. 11, no. 6, pp. 905–921, 2012.

[41] Z. Liu and I. Elhanany, "Rl-mac: A reinforcement learning based mac protocol for wireless sensor networks," *International Journal of Sensor Networks*, vol. 1, no. 3-4, pp. 117–124, 2006.

[42] K.-H. Phung, B. Lemmens, M. Mihaylov, L. Tran, and K. Steenhaut, "Adaptive learning based scheduling in multichannel protocol for energy-efficient data-gathering

wireless sensor networks," *International journal of distributed sensor networks*, vol. 9, no. 2, p. 345 821, 2013.

[43] W. So, J. Walrand, J. Mo, *et al.*, "Mcmac: A parallel rendezvous multi-channel mac protocol," in *2007 IEEE Wireless Communications and Networking Conference*, IEEE, 2007, pp. 334–339.

[44] S. Galzarano, G. Fortino, and A. Liotta, "A learning-based mac for energy efficient wireless sensor networks," in *Internet and Distributed Computing Systems: 7th International Conference, IDCS 2014, Calabria, Italy, September 22-24, 2014. Proceedings 7*, Springer, 2014, pp. 396–406.

[45] D. De Guglielmo, S. Brienza, and G. Anastasi, "Ieee 802.15. 4e: A survey," *Computer Communications*, vol. 88, pp. 1–24, 2016.

[46] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15. 4e networks," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC)*, IEEE, 2012, pp. 327–332.

[47] R. Amini, M. Imani, P. I. Todorov, and M. Ali, "Performance evaluation of orchestra scheduling in time-slotted channel hopping networks," in *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, 2020, pp. 142–147.

[48] M. A. Sordi, O. K. Rayel, G. L. Moritz, and J. L. Rebelatto, "Towards improving tsch energy efficiency: An analytical approach to a practical implementation," *Sensors*, vol. 20, no. 21, p. 6047, 2020.

[49] S. Kharb and A. Singhrova, "A survey on network formation and scheduling algorithms for time slotted channel hopping in industrial networks," *Journal of Network and Computer Applications*, vol. 126, pp. 59–87, 2019.

[50] G. Daneels, E. Municio, B. Van de Velde, *et al.*, "Accurate energy consumption modeling of ieee 802.15. 4e tsch using dual-bandopenmote hardware," *Sensors*, vol. 18, no. 2, p. 437, 2018.

[51] C. Szepesvári, *Algorithms for reinforcement learning*. Springer Nature, 2022.

[52] Y. Gao, R.-Y. Zhou, H. Wang, and Z.-X. Cao, "Study on an average reward reinforcement learning algorithm," *CHINESE JOURNAL OF COMPUTERS-CHINESE EDITION-*, vol. 30, no. 8, p. 1372, 2007.

[53] F. Qi-Ming, L. Quan, W. Hui, X. Fei, Y. Jun, and L. Jiao, "A novel off policy q (λ) algorithm based on linear function approximation," *Chinese Journal of Computers*, vol. 37, no. 3, pp. 677–686, 2014.

[54] Y. Wei and M. Zhao, "A reinforcement learning-based approach to dynamic job-shop scheduling," *Acta Automatica Sinica*, vol. 31, no. 5, p. 765, 2005.

[55] G. Tesauro, "Td-gammon, a self-teaching backgammon program, achieves master-level play," *Neural computation*, vol. 6, no. 2, pp. 215–219, 1994.

[56] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, Springer, 2006, pp. 282–293.

[57] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[58] I. R. Galatzer-Levy, K. V. Ruggles, and Z. Chen, "Data science in the research domain criteria era: Relevance of machine learning to the study of stress pathology, recovery, and resilience," *Chronic Stress*, vol. 2, p. 2 470 547 017 747 553, 2018.

[59] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[60] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[61] X. Zhao, L. Xia, J. Tang, and D. Yin, "" deep reinforcement learning for search, recommendation, and online advertising: A survey" by xiangyu zhao, long xia, jiliang tang, and dawei yin with martin vesely as coordinator," *ACM sigweb newsletter*, vol. 2019, no. Spring, pp. 1–15, 2019.

[62] S. Kosunalp, Y. Chu, P. D. Mitchell, D. Grace, and T. Clarke, "Use of q-learning approaches for practical medium access control in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 146–154, 2016.

[63] W.-K. Yun and S.-J. Yoo, "Q-learning-based data-aggregation-aware energy-efficient routing protocol for wireless sensor networks," *IEEE Access*, vol. 9, pp. 10 737–10 750, 2021.

[64] S. Shamshirband, A. Patel, N. B. Anuar, M. L. M. Kiah, and A. Abraham, "Cooperative game theoretic approach using fuzzy q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 228–241, 2014.

[65] G. Künzel, L. S. Indrusiak, and C. E. Pereira, "Latency and lifetime enhancements in industrial wireless sensor networks: A q-learning approach for graph routing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5617–5625, 2019.

[66] D. Pandey and P. Pandey, "Approximate q-learning: An introduction," in *2010 second international conference on machine learning and computing*, IEEE, 2010, pp. 317–320.

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

[68] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[69] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. A. Grieco, "Autonomous and traffic-aware scheduling for tsch networks," *Computer Networks*, vol. 135, pp. 201–212, 2018.

[70] Y. Tanaka, K. Brun-Laguna, and T. Watteyne, "Trace-based simulation for 6tisch," *Internet Technology Letters*, vol. 3, no. 4, e162, 2020.

[71] I. 8. W. Group *et al.*, "Ieee standard for local and metropolitan area networks-part 16: Air interface for fixed broad-band wireless access systems," *IEEE Std. 802.16-2004*, 2004.

[72] N. Kushalnagar, G. Montenegro, and C. Schumacher, "Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals," 2007.

[73] J. Leng, "Reinforcement learning and convergence analysis with applications to agent-based systems," Ph.D. dissertation, 2008.

[74] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled tsch," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 337–350.

[75] H. Kharrufa, H. A. Al-Kashoash, and A. H. Kemp, "Rpl-based routing protocols in iot applications: A review," *IEEE Sensors Journal*, vol. 19, no. 15, pp. 5952–5967, 2019.

Appendices

Appendix A