UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Programme of MSc Studies

Science and Technology of Electrical and Computer Engineering

Study and Development of Machine Learning Algorithms for Optimizing Performance in IoT Wireless Networks

Master's Thesis

Kyriakou Aikaterini

Supervisor: Prof. Korakis Athanasios

June 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Programme of MSc Studies

Science and Technology of Electrical and Computer Engineering

Study and Development of Machine Learning Algorithms for Optimizing Performance in IoT Wireless Networks

Master's Thesis

Kyriakou Aikaterini

Supervisor: Prof. Korakis Athanasios

June 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών

Επιστήμη και Τεχνολογία Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Μελέτη και Ανάπτυξη Αλγορίθμων Μηχανικής Μάθησης, για την Βελτιστοποίηση Απόδοσης σε Ασύρματα Δίκτυα IoT

Μεταπτυχιακή Διπλωματική Εργασία

Κυριακού Αικατερίνη

Επιβλέπων: Καθ. Κοράκης Αθανάσιος

Ιούνιος 2023

Approved by the Examination Committee:

Supervisor **Prof. Korakis Athanasios**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Bargiotas Dimitrios**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Argyriou Antonios**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

First and foremost I wish to thank my advisor, Assistant Professor Thanasis Korakis for giving me the opportunity to work in such an inspiring research unit and more importantly for his guidance.

Also I would like to thank Assistant Professor Dimitrios Bargiotas and Assistant Professor Antonios Argyriou, who are in my advisor commitee, for their guidance in this thesis.

Moreover, I would like to thank all the members in NITlab, for supporting me every day and, especially, I would like to thank Kostas Chounos for supporting me and collaborating with me throughout the implementation of this dissertation.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Kyriakou Aikaterini

Master's Thesis

Study and Development of Machine Learning Algorithms for Optimizing Performance in IoT Wireless Networks

Kyriakou Aikaterini

Abstract

In recent years, the continuously increasing demand for outdoor IoT application systems, such as smart metering, has led to IEEE 802.11ah standard gaining interest by the research community. As compared to other LPWAN (Low Power Wide Area Network) protocols, IEEE 802.11ah excels in two key areas for outdoor IoT: low power consumption and wide transmission area coverage. Our investigation focuses on the examination of interference arising between IEEE 802.11ah devices in dense network topologies. The initial experimental evaluation of NRC7292's behaviour denotes that collisions occur between transmitting station nodes even though they are set in different frequencies in the same bandwidth. Our proposal to address this limitation is the application of a Deep Q Learning approach, which aims to utilize NRC7292's parameters of Carrier Sense time for backoff enhancement in order to achieve improved results in terms of PDR and, also, fair spectrum sharing between the station nodes. The results demonstrate that training a DQN agent while experimenting with various TX time configurations to learn the environment, converges to an upgraded performance while revealing the information about the optimal configuration parameters for carrier sensing.

Keywords:

IEEE 802.11ah, Q Learning, Deep Q Network, Carrier Sense, NRC7292

Μεταπτυχιακή Διπλωματική Εργασία

Μελέτη και Ανάπτυξη Αλγορίθμων Μηχανικής Μάθησης, για την Βελτιστοποίηση Απόδοσης σε Ασύρματα Δίκτυα IoT

Κυριακού Αικατερίνη

Περίληψη

Τα τελευταία χρόνια, η διαρκώς αυξανόμενη ανάγκη για ανάπτυξη υπαίθριων συστημάτων του ”Διαδικτύου των Πραγμάτων”, όπως οι έξυπνες μετρήσεις, έχει οδηγήσει στο ενδιαφέρον της ερευνητικής κοινότητας στο πρωτόκολλο ασύρματης επικοινωνίας IEEE 802.11ah. Συγκριτικά με άλλα πρότυπα ασυρμάτων δικτύων που ανήκουν στην κατηγορία των δικτύων ευρείας περιοχής χαμηλής ισχύος, το πρωτόκολλο IEEE 801.11ah υπερτερεί σημαντικά τόσο στην χαμηλή κατανάλωση ενέργειας όσο και στην ευρεία κάλυψη. Η συγκεκριμένη εργασία επικεντρώνει το ενδιαφέρον της στην εξέταση σεναρίων παρεμβολής μεταξύ των συσκευών IEEE 802.11ah σε πυκνές τοπολογίες δικτύου. Η αρχική πειραματική μελέτη του συγκεκριμένου πρωτόκολλου με την χρήση των συσκευών NRC7292 αναδεικνύει ότι είναι εμφανής η παρεμβολή μεταξύ των μεταδόσεων, ακόμη και αν οι σταθμοί έχουν οριστεί σε διαφορετικές συχνότητες στο ίδιο εύρος. Η συνεισφορά της συγκεκριμένης έρευνας αφορά την προσπάθεια εξάλειψης των συγκρούσεων στις μεταδόσεις με την χρήση μεθόδων της ενυσχιτικής μηχανικής μάθησης, με στόχο την βέλτιστη αξιοποίηση των παραμέτρων του χρόνου ανίχνευσης μετάδοσης στις συσκευές NRC7292 για την βελτίωση του ρυθμού μετάδοσης πακέτων αλλά και την δίκαιη αξιοποίηση του καναλιού μετάδοσης. Τα αποτελέσματα των πειραμάτων αναδεικνύουν ότι η εκπαίδευση του μοντέλου DQN με τον πειραματισμό των παραμέτρων χρόνου μετάδοσης και ανίχνευσης συγκλίνει σε βελτίωση της απόδοσης με το πέρασμα των εποχών, παρέχοντας σαν αποτέλεσμα τις βέλτιστες για τις συνθήκες παραμέτρους.

Table of contents

List of figures

List of tables

Abbreviations

ML	Machine Learning
RL	Reinforcement Learning
NN	Neural Network
DQN	Deep Q Network
CS	Carrier Sense
RAW	Restricted Access Window
STA	Station
AP	Access Point
CCA	Clear Channel Assesment
CLI	Command Line Interface
PDR	Packet Delivery Ratio
MDP	Markov Decision Process
MCS	Modulation Coding Scheme

Chapter 1

Introduction

In this chapter we shall outline the key concepts covered in this dissertation. The major goal is to exploit features of emerging IoT technologies such as IEEE 802.11ah by using machine learning techniques to address and restrict potential limitations that may arise in real-time experiments. Additionally, the contribution of our work will be mentioned.

1.1 Introduction

1.1.1 Internet of Things

In recent years, the number of physical devices that are connected to the Internet has increased significantly, which has fueled the growth of the Internet of Things (IoT) area. IoT aspires to interconnect millions of devices while promising fast, efficient, and automatic real-time connectivity, communication, and management without requiring human involvement [10]. The term ”things” includes not only computers but also sensors, vehicles, household electrical appliances, etc. It is believed that the continuous development of IoT will significantly alter many aspects of our lives and open up new innovative areas such as smart cities, smart health, smart metering, and many more. These areas of IoT applications differ in terms of data rate, power consumption, distance coverage, and the number of devices supported. To meet the competing demands of each category, the development of new communication technologies is critical.

In particular, the increasing demand for outdoor IoT applications, such as smart metering and environmental monitoring, which demand low-power consumption devices, leads to new fields of research. Existing wireless communication technologies can be divided into

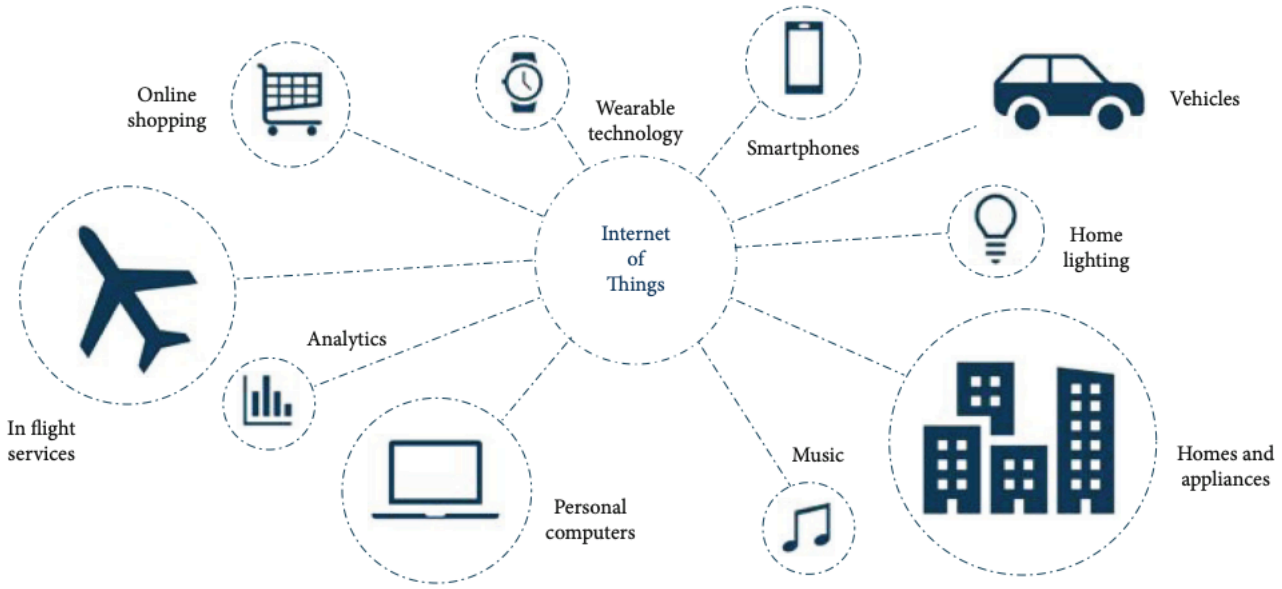

Figure 1.1: Internet of Things [1]

two categories based on their coverage: 1) Wireless Personal Area Network (WPAN),which cover short-range communications up to tens of meters, such as Zig-Bee or Bluetooth and 2) Low-Power Wide Area Network (LPWAN), which cover long-range communications up to tens of kilometers, such as LoRaWAN, NB-IoT, and IEEE 802.11ah. LPWAN technologies achieve efficient IoT connectivity over wider areas while maintaining lower power consumption compared to WPAN.

1.1.2 IEEE 802.11ah

IEEE 802.11ah, also known as Wi-Fi HaLow, is an emerging task group of the IEEE 802.11 standard that has been developed to operate in the sub 1GHz frequency band (863-868.8 MHz in Europe). It is able to support up to 8191 STAs within a 1 kilometer range while guaranteeing at least 100 Kbps of throughput, and it supports various bandwidths such as 1 MHz, 2 MHz, 4 MHz, 8 MHz, and 16 MHz [11].

Wi-Fi HaLoW's features vary from increased data rate to lower power consumption when comparing to other LPWAN protocols. It provides the ability to achieve lower energy consumption compared to other wireless standards while covering a larger geographic area, due to its power saving management architecture of alternating between "awake" and "sleep" mode according on the information received [12]. Moreover, IEEE 802.11ah standard ben-

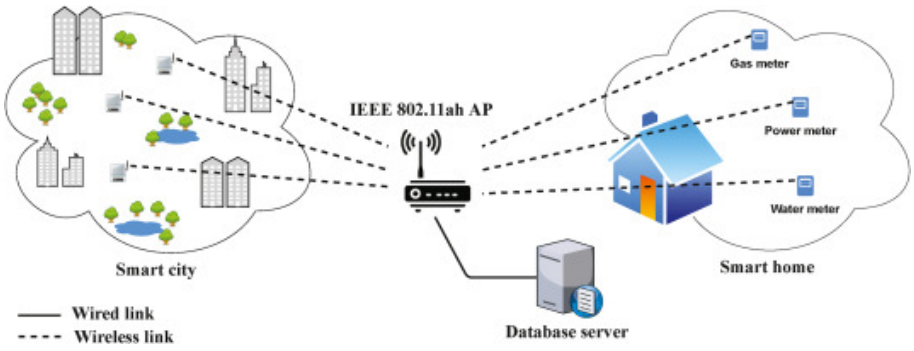

Figure 1.2: IEEE 802.11ah (WiFi HaLoW) [2]

efits in terms of data rate since it offers rates from 150 kbps to 78 Mbps at a range up to 1 kilometer[13]. Another important novelty of Wi-Fi HaLow is the Restricted Access Window MAC layer mechanism which divides the STAs into groups aiming to reduce congestion in channel access and minimize collisions in cases of dense network topologies [14].

IEEE 802.11ah	
Frequency	Sub-1 GHz
Data Rate	150 kbps to 346 Mbps
Coverage range	Up to 1 km
Power Consumption	Low
Devices Supported	8191

Table 1.1: IEEE 802.11ah [9]

1.1.3 Machine Learning

Nowadays, Machine Learning is one of the most emerging technologies of the Artificial Intelligence branch, having applications in numerous categories such as image/voice recognition, self-driving cars, e-commerce product recommendations, and more. Based on the way it is developed, it is categorized into four main types: 1) Supervised Learning, 2) Unsupervised Learning, 3) Semi-supervised Learning and 4) Reinforcement Learning. Unlike the first three categories which demand a usually large set of data to train the ML model, reinforcement learning does not require one as it develops the training data by interacting with the environment. By rewarding desired behavior and punishing undesirable behavior, reinforcement learning trains the model.

One of the most widely used reinforcement learning approaches is Q Learning. Q learning is applied mostly in problems with infinite decisions and its target is the optimal policy to achieve the desired result. The learning process uses and off-policy method which separates the learning from the actions and it controls the action choice based on the Bellman optimality equation and the ε-greedy policy [15]. However, as the most infinite-horizon decision problems produce a large amount of possible actions, Q learning becomes difficult since it leads to huge storage of data necessary for the model training. Therefore, another approach named ”Deep Q Learning” is promoted, which eliminates the above mentioned problems by combining Deep learning with Neural Networks and Q learning for faster computation and less need for data storage.[16].

1.2 Motivation

Our motivation in this dissertation is to focus on the development of a Reinforcement Learning model which will attempt to eliminate the interference between WiFi HaLoW devices. We observed how dense network topologies of IEEE 802.11ah devices negatively impact the overall throughput performance in cases of transmission in different channels, by allowing some devices to capture the transmission bandwidth from others. Subsequently, we observed how the carrier sense parameter variations could significantly affect the data rates and channel utilization in the case described before. By utilizing Deep Q Learning and WiFi HaLoW’s carrier sense parameters we could gain overall improved throughput and almost equal utilization of the channel for all nodes in the network.

1.3 Contribution

The contribution of the current dissertation is summarized as follows:

1. Reinforcement learning approaches were studied for the determination of the optimal carrier sense parameters in IEEE 802.11ah dense IoT topologies.

2. A Deep Q Learning Neural Network was implemented to create an agent for the decision making

3. The algorithm’s efficiency was studied and it presented an optimal selection of parameters in order for all the WiFi HaLoW nodes to obtain equal chance of transmission

and, therefore, equal data rates, while before the distribution was unfair.

4. Moreover, the overall throughput and channel optimization seem to benefit from our approach as observed in the evaluation.

1.4 Dissertation's organization

This dissertation is structured as follows. Section II provides a theoretical background both on Q Learning and on Neural Networks. Section III presents the architecture of our proposed model of the Deep Q Learning approach in IEEE 802.11ah interference mitigation. In Section IV, we present the experimental setup, the experiments executed to determine our algorithm's impact in interference mitigation and the evaluation of them. Finally, in chapter V, we describe the conclusions observed from the evaluation and, also, the future work we intend on researching.

Chapter 2

Background

This chapter focuses on providing essential information about the theoretical background of the Machine Learning areas studied in the current dissertation. All the necessary information about Reinforcement Learning, Q-Learning and Neural Networks is provided in order to exploit as many of their capabilities as possible to improve throughput performance in IEEE 802.11ah interference scenarios.

2.1 Q-Learning

2.1.1 Reinforcement Learning

Reinforcement learning is one of the main areas of Machine Learning alongside with Supervised Learning, Unsupervised Learning and Semi-Supervised Learning. The main objective of the specific ML category is the usage of self-trained agents that aim to maximize their rewards by taking a series of actions in a dynamic environment and, therefore, learn an optimal policy at time discrete steps of iterations. For each action on a given situation of the environment, the agent computes a corresponding reward determining whether the action taken had a positive or a negative influence. The purpose of the agents is to find an optimal policy π which maps states to actions and maximizes the reinforcement learning measurement as it trains. Nonetheless, RL assumes that the environment does not change over time [17]. In comparison with other ML categories, Reinforcement Learning does not require a data set with the optimal value defined. Instead, the agents compute the optimal actions for each state in real-time and train their models. This learning scheme has proven to be efficient for decision making problems with infinite number of actions to perform. The basis of reinforcement

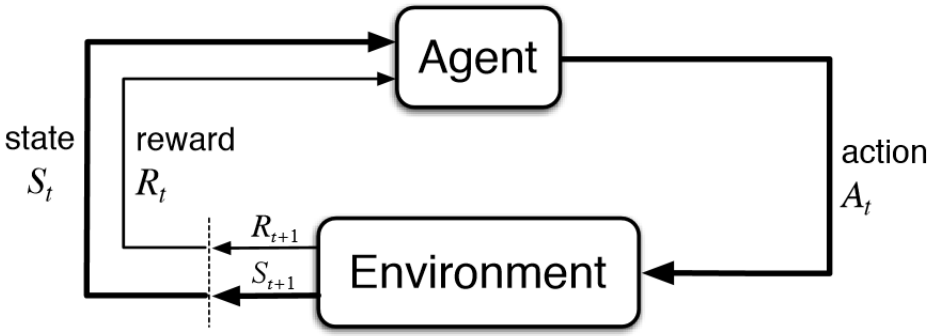

Figure 2.1: Visual representation of Reinforcement Learning overview (Source: [3])

learning relies on the iterative execution of actions given specific environment situations in order to maximize the agent's reward. The sequential action model of the agent relies on the Markov Decision Process (MDP) [18], which is a discrete-time stohastic decision-making process that uses the mathematical framework in order to determine the appropriate solution in a dynamic environment scenario by making a series of decisions.

2.1.2 Overview

One of the most known reinforcement learning techniques is Q-Learning, which searches for optimal policy given an environment situation without requiring a training set of data. Instead, this approach uses agents that perform a series of actions to investigate the environment and maximize the Q-Reward. The agent exploits the state of the environment and the impact of various actions in order to compute a Q-reward. In our case, the Q-reward will be used by the agent to find the optimal CS time configuration policy in order to reduce packet collisions and to maximize PDR. In our proposed model, as a Q-reward we consider the average of the successfully received packets in all APs participating in the network divided by the average percentage difference between each AP's successfully received packets. The reward function is approximated over time using an update equation, in our case the Bellman Equation, that combines actions, states, a discount rate and a Q-learning rate.

2.1.3 States, Actions, Rewards

Q Learning is an off-policy value-based reinforcement learning method which finds an optimal policy for the environment configuration by updating a reward function iteratively based on the environment situations and the possible actions for each situation. This sec-

tion will thoroughly explain the terms that Q Learning algorithm uses in order to determine how they can be applied in the problem this dissertation is focusing on improving. Based on Markov Decision, Processes which are the basis of RL and Q learning, the terms for the interpretation of the environment are the following:

- States S is the states set, which is a finite set of possible situations that can be observed by the agent in the environment.
- Actions A is the actions set, which is the finite set of possible actions the agent can execute given a state S_t. In most cases, the action set is the same for all the possible states.
- Reward R is the information that the agent computes about how an action executed at the presence of a given state of the environment is affecting its overall learning about the optimal policy.

Since Q Learning is a discrete-time problem, the above terms are also defined for each given step of the sequential decision-making problem.

- S_t is the state of the environment during time step t, where $S_t \in S$
- A_t is the action applied during time step t at a given state S_t, where $A_t \in A$
- R_{t+1} is the reward value given to the agent at time step $t+1$ about the execution of an action to the next state, where $R_T \in R$

Given this information, an example of the decision-making sequential training will lead to an episode represented as the following: $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_N, A_N, R_{N+1}$

2.1.4 Q-Table

In order to train the RL agents for sequential acting and learning, it becomes necessary to store the information data about states, actions and rewards. Q Learning algorithm uses a data table to store State-Action Q-values called Q-Table. The Q-table consists of rows that represent states and columns that represent possible actions, and each cell contains the estimated Q-value for applying action A to state S. The Q-value represents the maximum expected future reward that the agent will receive if the current action is applied to the current state.

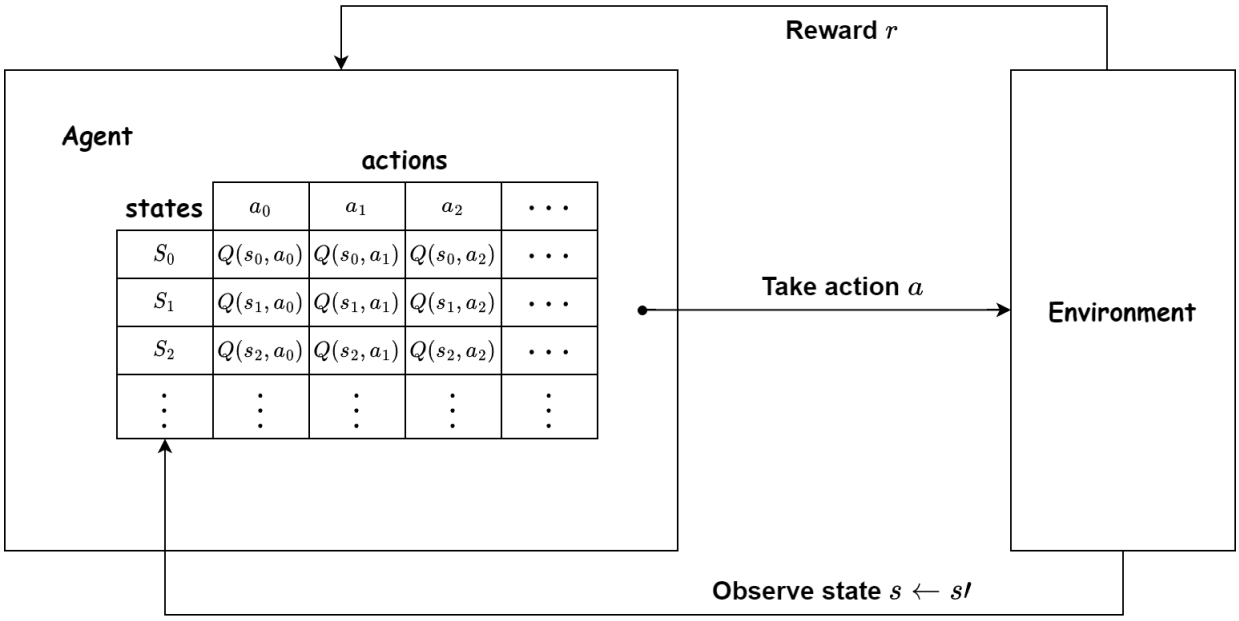

Figure 2.2: Q-Table functionality in RL [4]

Initially, the Q-Table values are set to zero, but as the agent trains and learns from the environment through time, it updates those Q-values until they converge to the optimal ones. Q-table updates its values with a Q-function, which is based on the Bellman Equation that will be described in detail in subsequent chapter section. An example of the Q-table and how it operates in Q-Learning environments can be shown in Figure 2.2

2.1.5 Expected Return

As mentioned before, the purpose of the Q Agent is to maximize the cumulative rewards during a training episode and it can be represented as follows:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... = \sum_{t=0}^{T} R_{t+1}, t \in T \tag{2.1}$$

In this definition R_t is the reward at time step t, where $t \in T$ and T is a finite amount of time steps during one training episode. The summary of all the rewards during an episode comprise the expected return. The return can also be infinite and not have a limit of maximum time step T. For this case we will introduce the discount factor.

2.1.6 Discount Factor

As the amount of possible iterations can become very large for each training episode of Q Learning agent, there arises a need for promoting the most immediate rewards compared to the ones that occur later in time. For this purpose, the update equation should take into consideration the discount factor γ which is important for the discovery of the optimal policy as early as possible during training. The discount factor γ is a way to weight later rewards overtime by being introduced to the update equation. Therefore, the return mentioned before is restructured as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^K R_{t+1}, t \in T \tag{2.2}$$

2.1.7 Bellman Optimality Equation

The Bellman Equation in Reinforcement Learning results in the selection of the policy that gives the highest reward expectation, while taking into consideration the discount factor γ, and is defined as follows:

$$Q_\pi(s_t, a_t) = E[R_{t+1} + \gamma \max_{a'} Q_\pi(s_{t+1}, a_{t+1})] \tag{2.3}$$

By taking advantage of the Bellman Equation, the Q-table cells are iteratively updated for each state-action pair until the Q-function converges to the optimal Q-function Q_π.

$$\begin{aligned} Q_{new}(s_t, a_t) &= (1-\alpha)Q(s_t, a_t) + \alpha(R_t + \gamma \max_a Q(s_{t+1}, a) \\ &= Q(s_t, a_t) + \alpha[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \end{aligned} \tag{2.4}$$

where α is the learning rate, γ is the discount factor, $Q(s_t, a_t)$ is the old value, R_t is the reward, $\max_a Q(s_{t+1}, a)$ is the estimate of the optimal future value and $(R_t + \gamma * \max_a Q(s_{t+1}, a)$ is the new value.

2.1.8 Epsilon-Greedy Exploration Strategy

Another important aspect of Q Learning that requires to be considered is the strategy for choosing between exploration and exploitation. The specific tradeoff lies between allowing

the agent to enrich its existing knowledge, by trying random actions hoping to result a positive impact in future decision, and being able to exploit the history of state-action estimations calculated by previous explorations. To balance this tradeoff for the Q learning training process, the ϵ-Greedy Algorithm is suggested, which consists a method that balances exploration and exploitation options while agent is interacting with the environment. The ϵ value refers to the probability of choosing to explore new random combinations and gather information about the corresponding rewards. If set to zero, the agent will never explore and will always exploit its existing knowledge, while if set to 1, the agent will be forced to continously explore random actions and will never exploit existing knowledge. Consequently, as agent is asked to perform actions for given situations the decision is made based on the selection of a random number $n \in (0, 1)$, and if this number is greater than the ϵ value, then the agent chooses to exploit, otherwise, the agent explores the next action.

$$\text{Next Action} = \begin{cases} \max_a Q(s_t, a), & \text{with probability } 1 - \epsilon, \\ \text{Random}, & \text{with probability } \epsilon \end{cases} \tag{2.5}$$

2.2 Neural Network

2.2.1 Architecture

One of the most breakthrough discoveries in Machine Learning area is the development of Artificial Neural Networks [19], or simply called Neural Networks (NN), which were inspired by the functionality of biological neural networks, and, nowadays, comprise the base of Deep Learning algorithms by creating a layered structure of interconnected neurons which resembles the human brain's neurons [20]. NNs are purposed to process a massive variety of data for classification, modeling of complicated patterns, pattern recognition or even predicting information.

Their basic computation component is the neuron, or node, which takes input from another node or from external components, calculates an output based on an activation function and returns it to subsequent nodes or as the result of the NN. Each one of the nodes have a corresponding value called weight, which determines the importance of this specific node to the overall output of the system, whether it is positive or negative [21]. The total connection of each neuron and its related weight is depicted through matrices. Neural networks leverage

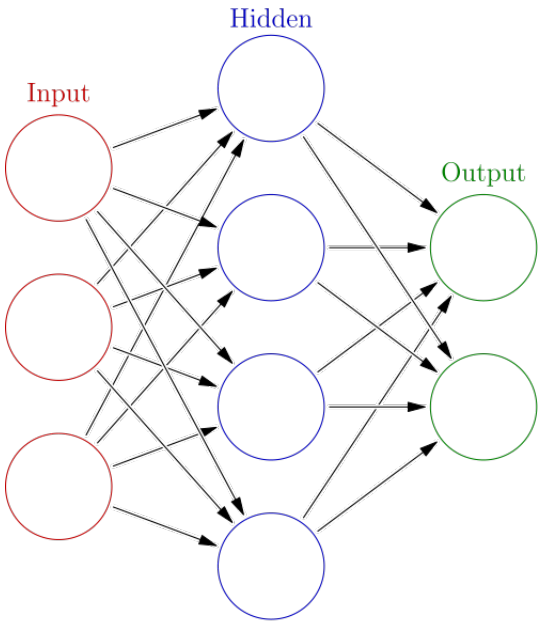

Figure 2.3: Neural Network Architecture [5]

a hierarchical structure of layers categorized in three types. These fundamental layer categories a Neural Network's architecture are the input layer, the hidden layers and the output layer, and each type of layer is independent from the other ones and can have an arbitrary number of computing nodes. A representation of a NN and its structure of nodes and layers can be shown in Figure 2.3.

2.2.2 Perceptron

A very common and simple construction of a Neural Network is the Perceptron , that consists of fully-connected nodes that transmit the information from one layer to another one. This type of Neural Network contains feed-forward structure of training, where the input nodes accept information data, they forward it to the hidden nodes where some transformation is carried out and in their turn they forward it to the output node which is responsible for the result propagation of the whole network. This process only happens in a direction from the input layers to the output layer and one of its used examples are the Single Layer Perceptron [22]. However, there is another approach which does involve the iterative update of weights returned from the output layer to the input layer, moving in a backward direction, called the Multi-Layer Perceptron [23], that uses the Back Propagation algorithm for training its model.

Back Propagation Algorithm

The process of learning in a Multi-Layer Perceptron (MLP) is called the back propagation algorithm and is categorized in the supervised ML, which denotes that the training data consist of a feature and an associated label that is considered to be the target value[24]. MLP combines the forward propagation and the backwards propagation, in order to achieve a learning model that minimizes the loss between calculated values and optimal values. Its main process is divided in three fundamental parts and the two latter are repeated until convergence to the desired learning output.

- Initialization phase, where all the weights of the network's nodes are initialized
- The forward propagation, where an input is presented to the network, the hidden nodes apply on it the activation function and the error function computes the divergence from its target value
- The backward propagation, where the errors calculated previously are propagated to the previous layers and adjustment of the weights is performed.

2.2.3 Activation Function

Activation functions are a key feature of NN since they can substantially determine the learning process by mapping complicated input data to corresponding outputs. Without activation functions, Neural Networks would lower their abilities to the ones of Linear Regression Model since they could only process linear equation, thus, limiting the variety of complex tasks they are able to solve. [25]. The important functionality of the activation function is the transformation of an input signal to an output signal, by applying mathematical functions on it. These transformed signals are fed from one NN layer to another in order to approach each corresponding model. Some of the main types of Activation Functions are:

- Linear
- Sigmoid
- Tanh
- ReLU

- SoftMax

In this implementation we chose to study the Rectified Linear Unit (ReLU) activation function [26], which is a linear function that bypasses the negative outputs by returing a zero and maintains the positive ones by returning them exactly as they are.

$$f(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

2.2.4 Error Function

Training process of Neural Networks requires modification of each layer's weights, seeking for minimization of a predefined error metric that could result to improved and more accurate model learning. To achieve this, the error (or loss) functions are introduced to NN training. A loss function interprets the "cost" of the selection of a single training input and evaluates how well the corresponding algorithm models the data set.

In Deep Learning, the most commonly used error function is Mean Squared Error (MSE) [27] which calculates the squared difference between the value and the model prediction and averages it across all the data set. It can be calculated with the following formula:

$$\text{MSE} = \frac{1}{N}\sum_{i}^{N}(Y_i - Y_i')^2 \tag{2.7}$$

where Y_i is the actual value, Y_i' is the optimal model value and N is the number of data in the data set.

2.2.5 Optimizer

Neural Networks also utilize optimizer algorithms, which are optimization methods used to adjust their fundamental attributes such as weights or learning rates with zero human interaction, aiming to improve the training pace and accuracy, and the model's performance.

There is a variety of available optimizers implemented for Neural Networks, but in this section we will focus on Stohastic Gradient Descent (SGD) [28] optimizer. SGD is an algorithm that starts from random values of parameters weights w an d learning rate η and iterativaly updates them until an approximate minimum is obtained. This method maintains a single learning rate for all weight updates which is not updated during training. The weights update equation of SGD is the following:

$$w := w - \eta \nabla Q_i(w) \tag{2.8}$$

2.2.6 Learning Rate

One of the most important hyper-parameters of a NN is the learning rate which determines how often the algorithm is updating its weights, and as a result, at what speed it is training the model. Learning rate selection is a very essential aspect of a NN's formation since it can control the decision making and could lead either to skipping the optimal solution if too large or requiring a huge amount of iterations to converge if too small.

2.3 Deep Q Network

As mentioned in previous section, Q-Learning requires the creation of a Q-table to store the Q-values for each possible pair of state-action. In cases of very large amounts of states and actions combinations, this data structure seems impractical to store and access since it can magnify up to $|S| \times |A|$, where $|S|$ is the amount of possible states and $|A|$ is the amount of possible actions. To avoid such a huge storage need, the approach proposed to leverage Q Learning is the Deep Q Network, an approach that combines the advantages of Q Learning algorithm And Neural Networks and its main purpose is to learn optimal policies from high dimensional input data of states and actions.

In DQN, the same process of Q Learning is adjusted to an approach that utilizes Neural Networks instead of a Q-table. The NN repeatedly samples all actions in all states until convergence to the corresponding optimal Q-values. DQN agent's initialization defines some fundamental configuration parameters both for NN and for Q-Learning such as:

- NN Layers and their activation functions
- Learning Rate
- Optimizer
- Memory replay buffer
- Batch size

- Error function
- Discount factor γ
- Greedy exploration probability ϵ

The most important feature of DQN is the memory replay buffer [29], which stores the path of states, actions and rewards taken in the past, so that the agent is able to maintain a limited history that will help in achieving learning. During each training iteration, some of the memory buffer combinations are sampled and given as input to the Neural Network to re-evaluate its weights and provide improved rewards. The process of DQN training is described in this chapter:

1. A sample of An observation of time step t is provided to the NN as input and it results in the expected reward for each possible action that can be applied in the specific state.
2. Following the ϵ-greedy exploration method, either a random action with a probability ϵ or a maximum reward action with probability $1 - \epsilon$ is selected and the reward for the chosen action is calculated and produced as an output of the NN.
3. All the information about the selected combinations of states, actions and their resulting rewards are stored in the memory replay buffer.
4. For each iteration, a number of epochs determined by the batch size are randomly sampled from the memory replay buffer and the agent calculates the Q-values and trains its model.
5. This process is repeated until convergence of the desired reward to a target value or until the repetitions of epochs are completed.

A representation of how DQN works can be shown in figure 2.4.

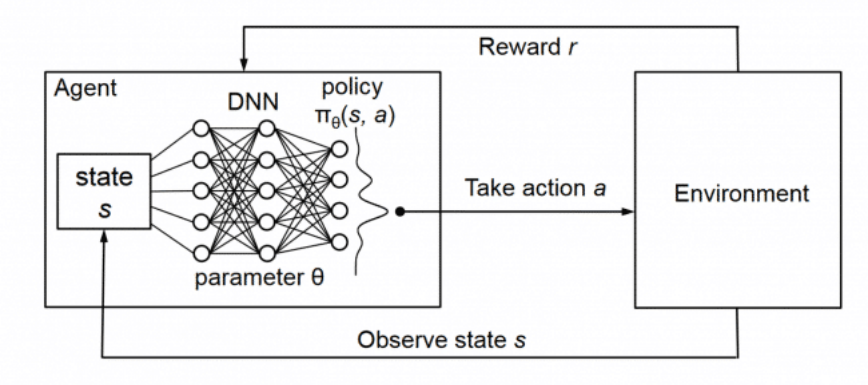

Figure 2.4: Deep Reinforcement Learning [6]

Chapter 3

Proposed Model

In this chapter, a detailed explanation of our proposed model is analyzed. The architecture described summarizes the interference problem IEEE 802.11ah devices face in dense topologies, the reinforcement learning algorithm and its configuration parameters, WiFi HaLoW's configuration parameters for carrier sense, and a summary of our infrastructure's topology.

3.1 Problem Formulation

For the formation of an optimization algorithm that finds the configuration parameters maximizing throughput and providing fair channel utilization in cases of interference in WiFi HaLoW networks, this dissertation proposes an approach that profits from reinforcement learning techniques while varying carrier sense configuration parameters. As mentioned in [30], IEEE 802.11ah comes with an efficient and fair channel access allocation mechanism called RAW, which divides the STAs into groups and performs medium access only for STAs belonging in the same RAW group over a specific time period, leading to the decrease of collisions among the various stations.

Nonetheless, what is observed is that interference persists in WiFi HaLoW networks despite the fact that RAW is implemented in its core. Specifically, the ambiguous scenario occurs when the devices assigned to transmit to different channels, which are located in distant frequencies, present a massive decrease in terms of throughput. What we discovered experimentally is that the performance degradation is significant even when multiple Station - Access point pairs broadcast packets simultaneously in distant frequency configurations. For example, two pairs of nodes transmitting concurrently in a 2MHz bandwidth, allocated

in different channels can lead to a performance degradation of more than 50% by falling to a throughput of $2Mbps$, while transmitting separately can lead up to $5Mbps$ of data rate.

Such a behavior is not considered normal due to the gap between the two edge frequencies tuning and taking into consideration that IEEE 802.11ah implements the RAW channel access mechanism to avoid collisions. Therefore, we researched in depth the configurations available to optimize in WiFi HaLow and we concluded to some of its parameters that could control the resulting throughput in such cases.

Digging deeper into IEEE 802.11ah and, in specific, the Newracom device NRC7292 [8], we discovered a variety of parameters one can tune in order to control various aspects of performance. Some of these parameters are the station's TX time, which consists of carrier sensing time and blank time, the duty cycle, used for backoff mechanism duration configuration, and the Clear Channel Assessment (CCA) threshold, which defines the sensitivity threshold in channel interference sensing. Varying the values of these parameters can lead to an increase or decrease of the transmission's performance and their configuration is easily accessible through a Command Line Interface (CLI) which does not require rebuilding the software driver everytime a change is required but it can propagate in real-time, even when the transmission is ongoing.

In the specific dissertation, we focus on exploring the effects of the parameter's CS time tuning, by enabling each station to wait for a specific time period in order to sense if the channel is available or occupied by another station and whether it should transmit data or not. Different values of this parameter could cause either increase or decrease of the transmission's performance and they vary from $min(csTime)$ equal to 0 to $max(csTime)$ equal to 12 milliseconds in carrier sense time and from $min(blankTime)$ equal to 0 to $max(blankTime)$ equal to 4294967 in blank time. Therefore, the amount of possible combinations of carrier sense time and blank time for N pairs of nodes is $(max(csTime) * max(blankTime))^N$ which becomes a huge number for manual selection and evaluation. To this end, we reached to Machine Learning approaches to help us determine which configuration values are optimal for each possible setup of nodes, in order to establish fairness in terms of channel utilization and increase in terms of Packet Delivery Ratio (PDR).

The development of a Machine Learning model in most cases require a data set that contains enough and valid data for training and, therefore, obtaining valid results in each problem. However, in our case the available data does not pre exist which forces us to resort to

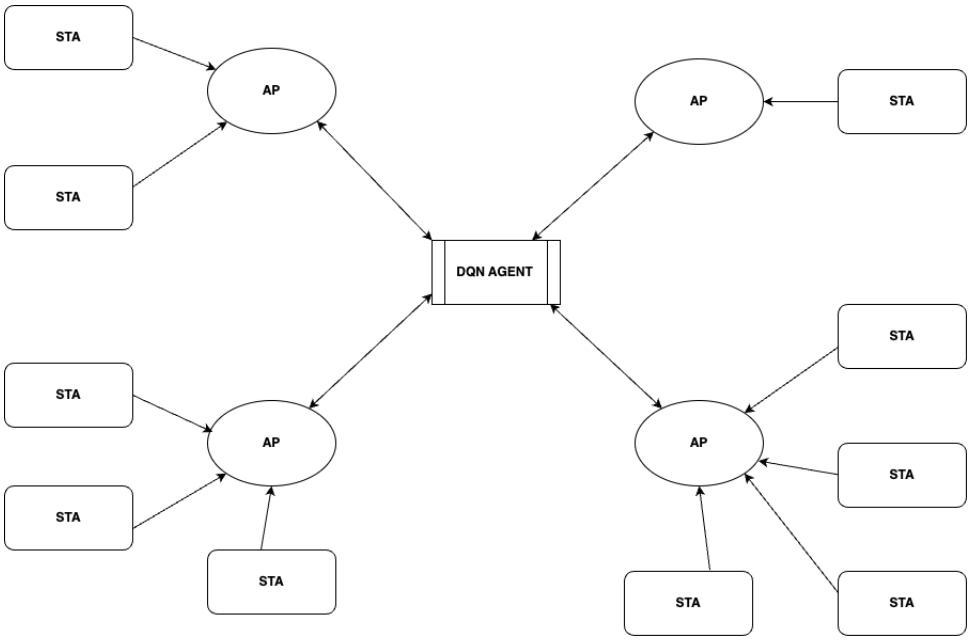

Figure 3.1: Example network with DQN proposed model

the Reinforcement Learning solution that does not require a considerable size of data to train its models. Q Learning is chosen among all the reinforcement learning techniques, since it is very simple and it is considered to be one of the best for finding optimal parameters in a problematic scenario.

Our proposed model combines IEEE 802.11ah carrier sense time configuration and Q Learning aiming to obtain an optimal set of parameters for all pairs of nodes which eliminate the interference opposed by density of topology and acquire improved PDR, enhanced channel utilization and fair distribution of bandwidth between the stations.

3.2 System Model Information

Our suggested model follows a centralized concept where we train one agent with one neural network that is responsible for the CS time management of all nodes participating in the network. The network topology that we are focusing on can be depicted in an example in figure 3.1 and is comprised of:

- N Stations (STAs) that transmit information to one Access Point.
- M Access points (APs). Multiple STAs can be connected to the same AP.

- 1 Agent: A remote node that gathers and controls information about all the nodes and also stores and trains the DQN model.
- K Arrays of combinations of CS Time and Blank Time in all pairs of nodes.

3.3 IEEE 802.11ah parameters

Wi-Fi HaLow's hardware, which is going to be analyzed in subsequent chapter, provides a variety of configurable parameters through a command line interface. That CLI can be easily accessed and updated without requiring to interfere in driver's code and can also propagate results in real-time. Some examples of its configurable parameters are:

- Transmission time, which defines Carrier Sense time and blank time
- Duty cycle time
- Clear Channel Assessment Threshold

In this dissertation, we focus on leveraging the first parameter, transmission (TX) time. The configuration of TX time requires two separate configurations:

1. Carrier Sense Time $\in [0, 12480]$ in $\mu seconds$
2. Blank Time $\in [0, 4, 294, 967, 295]$ in $\mu seconds$

CS time defines the amount of time that the device "listens before talk". This time specification denotes for how much time, in $\mu seconds$, the station should wait and monitor the channel traffic to confirm that the channel is idle before attempting to transmit. Blank time is the amount of time that the device should wait before transmitting again, after an execution of transmission. Figure 3.2 shows the application of CS time and blank time during a transmission.

3.4 Proposed Model

This dissertation's model consists a centralized approach where one remote machine learning device acts as a coordinator that gathers information about all the pairs of nodes in

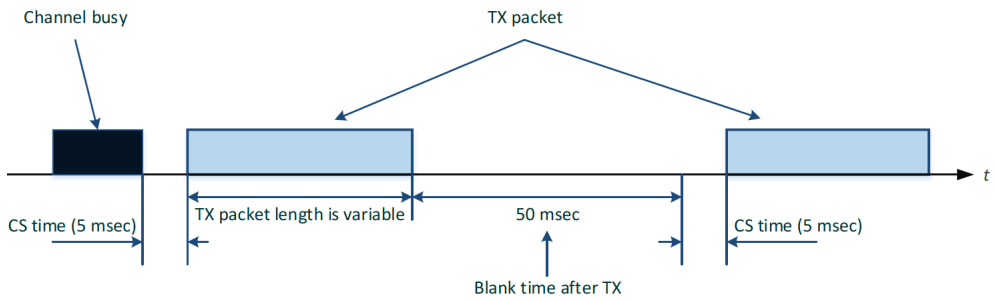

Figure 3.2: Carrier sense time and blank time visual representation (Source: [7])

the network and utilizes this information in developing a DQN agent that aims to find the optimal CS time configuration for all pairs of nodes of the network. The learning is composed of wireless environmental learning and optimal configurations selection. For the environmental learning, the proposed model is trained during multiple epochs. One epoch consists of channel allocation, Q value observation, and learning process. The NN acts as the agent of the learning scheme and it is given a state set S, an action set A, and a Q-value for the computation of the related reward. The CS time and blank time configuration selection for the next epoch is determined based on the state of the current epoch.

- State set S_t: The combinations of CS time and blank time configurations for all pairs of nodes connected to the network during epoch t.
- Action set A_t: The combinations of increasing or decreasing CS time and blank time for all pairs of nodes of the network during epoch t
- Q-reward Q_{t+1}: The average of the received packets in all nodes divided by the percentage difference between the number of received packets between the pairs of nodes.

The reward tries to maximize two factors at once by rewarding actions that try to maximize the number of successfully packets and, at the same, minimize the difference between the amount of received packets. This implementation aims to increase the average PDR and establish fair opportunities of transmission between the stations of the network.

For the parameters configuration, the ML unit defines a combination of CS time and blank time for all stations based on the output of the neural network. The neural network receives as input the state s_t and produces an output which defines the Q-reward. Then, the selection of the parameters is random with probability ϵ and based on rewards with probability

$(1 - \epsilon(t))$, where ϵ is the greedy exploration probability mentioned in previous chapters. In our approach, this probability is reduced either when the training epoch has converged to the upper thresholds of PDR and percentage difference, or at the end of each epoch.

Our learning scheme contains forward propagation and back propagation in the NN. In the first step, of forward propagation, the NN receives input and computes its output with the activation function being ReLU. In the second step, of back propagation, the NN weights are trained by using an error function. We also chose Root Mean Squared Error (RMSE) as the error function and Stohastic Gradient Descent (SGD) as the weights optimizer.

3.5 Implementation Configurations

In this section we are goint to analyze the selection of configurations and features of Q Learning, Neural Network and WiFi HaLow to achive our goals which are equal transmission opportunity, high bandwidth utilization and improved throughput.

3.5.1 Q Learning

- **States**

  The state set S, which contains combinations CS time and blank time parameters for all stations in the participating network.

- **Actions**

  The action set A denotes the combinations of increase or decrease of CS time for each station. For example, two pairs of nodes would have the following possible actions:

  - Increase CS time on both stations
  - Increase CS time on first station and decrease CS time on second station
  - Decrease CS time on first station and increase CS time on second station
  - Decrease CS time on both stations

  The total length of actions is, therefore, $|A|^N$, where N is the number of stations in the network.

- **Step: Next State, Reward, Done**

  The values of next state, reward and done are computed during each iteration, after the

agent performs an action.

- **Next state**

  Next state is calculated either randomly with probability ϵ or by the optimal reward with probability $(1 - \epsilon)$.

- **Reward**

  The reward Q_{t+1} exploits the information about the number of received packets in average in all APs and, also, the percentage difference of those numbers between the nodes.

  * Average of received packets

    $$\overline{RX} = \frac{1}{N} \sum_{i=1}^{N} rx_i \tag{3.1}$$

    where N is the number of APs.

  * Difference percentage for $N = 2$

    $$P = \frac{|rx_1 - rx_2|}{(\frac{rx_1 + rx_2}{2})} \times 100 \tag{3.2}$$

  The combination of these to leads to the reward which is calculated as follows:

  $$Q_{t+1} = \frac{\overline{RX}}{P} \tag{3.3}$$

  If the difference between the received packets is equal to zero, then the reward relies only on the average $\overline{RX}$.

- **Done**

  The done parameter denotes whether the training during epoch t has converged to the optimal values. It is set to true if the received packets are above an acceptable received packets threshold and if the percentage difference is smaller than a difference threshold.

- **Discount factor** The discount factor is selected at value 0.99 in our implementation which is less than 1. This aims to the promotion of immediate rewards and the denotion of the latter ones during epoch t because raising a value less than 1 to a power will lead to a smaller value. This way, our implementation considers the immediate rewards as more important for the training.

- ϵ**-greedy exploration strategy** In our implementation we promote a dynamic exploration strategy that does not rely only on randomness and one default parameter of ϵ probability. Our approach initializes the ϵ greedy probability to 1 at the beginning of the training and it reduces its value by a decay value of 0.005 each time an epoch is considered done. Therefore, the agent starts with high probability of exploration and as it converges more and more during the training the probability is decreased, thus, giving it the opportunity to exploit its gained knowledge.

3.5.2 Neural Network

In this part, we present some of the configuration parameters of the Neural Network that we have designed.

- **Number of layers**

  During research and experimentation we concluded that 4 layers are the best option for the number of layers of the Deep-Q-Network. Based on that the number of neurons in the first hidden layer ($Layer1$) are 10 and for the second hidden layer ($Layer2$) are 5.

- **Activation Functions**

  For the activation function of each layer we choose the rectified linear unit (ReLU = $max(0, x)$) and for the output function, which indicates the action of the agent, we choose the linear function.

- **Optimizer**

  We choose Stohastic Gradient Descent (SGD) as the optimizer for the update of the weights of the Neural Network, since compared to the Adam optimizer performs almost the same but it is less complex in its computation.

- **Learning Rate**

  We choose the learning rate based on the optimizer function (SGD) and we assign the value of 10^{-2}.

- **Error function**

  In our implementation we use the Mean Squared Error (MSE) as the error function.

The neural network's configurations are summarized in table 3.1:

Neural Network Training Parameters	
Activation Function	ReLu
Optimizer	SGD
Epochs	100
Epoch Length	60 iterations
Learning Rate	0.01
Number of Layers	4 (1 Input, 2 Hidden, 1 Output)
Number of Neurons per Hidden Layer	(10, 5)
Error Function	RMSE

Table 3.1: Neural Network Parameters

In our implementation we also define parameters related to the process of Q-Learning, such as:

- γ is the discount factor used for the update equation of the Q-Reward.
- Greedy parameter $\epsilon(t)$ is the probability of choosing randomly and $(1-\epsilon(t))$ is trying to maximize the Q-reward. At first it is set to 1 and as the training proceeds it is reduced.
- The batch size which defines the amount of Q values gathered in order to train the Agent for one epoch is set to 32.
- The memory buffer that is the storage for the actions, rewards and states throughout the process of the Agent's training has a size of 2000 entries.

The Deep Q Learning parameters are summarized in table 3.2:

3.5.3 IEEE 802.11ah CS Time

For the variation of CS time configuration we limited the possible assigned values, since it was observed that really big values tend to drop down the performance of the executing station. Therefore, we chose to define a space of CS time varying from 0 to $5ms$ with an interval of $1ms$ and blank time from 0 to $50ms$ with an interval of $10ms$. Therefore, for a

Q Learning Parameters	
Discount factor γ	0.99
Exploration Probability Start Value	1.0
Exploration Probability Decay	0.005
Batch Size	32
Replay Memory Buffer Size	2000 entries

Table 3.2: Q Learning Parameters

network of N nodes the possible states of CS time configuration are up to the amount of $|25|^N$.

3.6 DQN Agent Training

In our implementation we define a Deep Q Network (DQN) Agent with the model of the Neural Network and its related configuration parameters. The NN is a Sequential model that consists of 4 layers, the input layer, 2 hidden layers and an output layer. The input layer is formed using the $keras.Input$ type of tensor and the rest of the layers are using the $keras.Dense$ type of layer. After creating the Sequential model as described, we compile it using as loss function the MSE and as an optimizer the SGD with learning rate 10^{-2}. Then, we define the following functions for DQN Agent:

- **Compute action**

  In this function, we sample a variable uniformly over $[0, 1]$ in order to identify whether we will choose the action randomly or based on the existing Q values. If this variable is less than the $\epsilon(t)$, then we choose a random action, otherwise, we choose the action with the highest reward Q-value for the given state S_t.

- **Update greedy epsilon parameter**

  This function reduces the $\epsilon(t)$ parameter by a small amount. It is called after each epoch is finished.

- **Store Epoch**

  This function stores the current state, the action, the reward, the next state and whether

the epoch is done in the memory buffer of the DQN Agent. This information will be used for the choice of actions.

- **Train Model**

  This process is executed at the end of each epoch. We shuffle the data in memory buffer and we select an amount of items according to the batch size defined. Then, the function iterates through these items and for each one computes the Q-values of S_t, updates the Q-target value ($Q(S_t, A_t)$) based on the equation:
  $Q(S_t, A_t) = Q(S_t, A_t) + \gamma * \max_{\alpha \in A} Q(S_{t+1}, \alpha)$.
  Then, we train the model using $keras.model.fit$ for the current state.

3.7 Algorithm

The training phase repeats a training process for multiple epochs in order to train the DQN Agent model. During each epoch, the agent computes actions and stores information about the current state, the next state, the reward and whether the epoch is done for a number of iterations. For every batch size steps of this process, the agent trains the DQN model using its stored data. When the epoch is done, the agent reduces the $\epsilon(t)$ parameter. More details of the training process flow are shown in the following figure.

Algorithm 1 Training Phase

Create DQN Agent

for Every epoch t **do**

Initialize $current\_state$ with an initial observation value of CS time and Blank time values.

for Every step in iterations **do**

Agent computes action for the given $current\_state$.

Agent runs the action and returns $next\_state$, $reward$ and $done$ values.

Agent stores the values from the previous step in a memory buffer.

if Epoch has ended or done == true **then**

Agent updates $\epsilon(t)$

Break

end if

Set $current\_state$ = $next\_state$

end for

if $total\_steps \geq batch\_size$ **then**

Agent trains the model

end if

end for

Chapter 4

Evaluation

4.1 Hardware

4.1.1 NRC7292

Newracom [31] has introduced the world's first industry device called NRC7292, which is the first system on a chip (SoC) 4.1 architecture that is compliant with IEEE 802.11ah protocol. The specific standard of WiFi HaLow can be very useful for outdoor IoT scenarios with requirements for low power consumption and wide area coverage, and can vary in throughput from 150 Kbps to 15 Mbps, based on channel configuration and conditions. Its implementation for power consumption reduction by switching to ”doze” mode is a very important factor of WiFi HaLow. Some important NRC7292's key features are the following:

- Support for Hosted, Hostless and Standalone operation
- Supports 1/2/4 MHz bandwidth
- Throughput from 150 Kbps up to 15 Mbps
- Low power operation options
- Tools for configuration and test
- Software development kit and sample applications

Moreover, we present NRC7292's specifications [8] in the following table 4.1:

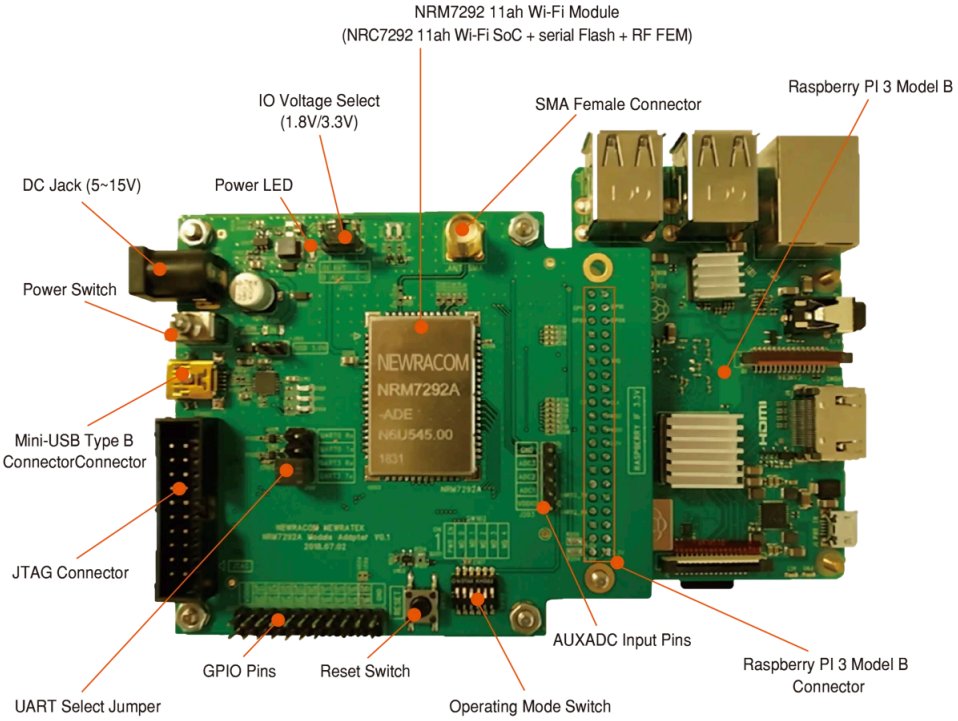

Figure 4.1: Newracom's NRC7292 devie [8]

4.2 Evaluation

The current dissertation focuses on the mitigation of collisions between IEEE 802.11ah devices, configured to transmit in different channels, in cases of dense network topologies. Our initial interest in studying this scenario of interference occured during experimental evaluation of WiFi HaLow's achieved data rates when transmitting in different frequencies in the same bandwidth. Specifically, in the case of $2MHz$ bandwidth, where the two possible channel allocations are in frequencies $864MHz$ and $866MHz$, we consider an example setup of two STA-AP pairs where AP 1 is configured in frequency $864MHz$ and AP 2 in frequency $866MHz$ and both APs are represented by Modulation and Coding Scheme (MCS) 7. As mentioned in [32], the expected data rate of one STA to the AP in bandwidth of $2MHz$ at MCS 7 with one spatial stream would vary from $6.5Mbps$ to $7.22Mbps$. Our experiments with the NRC7292 device showed that this throughput reaches a maximum value of $5.5Mbps$, as can be observed with the dashed lines in figure 4.2. sIn the case of two STA-AP pairs, since the channel frequencies are distant to each other, the same data rate performance would be

NRC7292 Specifications			
CPU	Memory	Communication Peripherals	RF Transceiver
ARM Cortex-M3 for Wi-Fi and application	32KB Boot ROM for M0	GPIO X 54	Single-ended RF ports
ARM Cortex-M0 for WLAN	32KB Boot ROM for M3	SPI X 45	Frequency band: 750 to 950MHz
ARM Cortex-M0 for WLAN	752KB System SRAM	UART X 4	Linear TX output power: 0dBm
	XIP with cache (2 ways, 32KB)	I2C X 4	TX gain range: 30dB
		9-bit ADC X 4	RX noise figure: 4dB
			Max. input level: -10dBm
			10bits ADC and DAC

Table 4.1: NRC7292 Specifications table [8]

expected when transmitting concurrently. However, as depicted in figure 4.2, the expected throughput falls to an average value of $2.5Mbps$, which denotes a degradation of almost 50%. Therefore, it is obvious that the two STAs transmitting are creating collisions with each other, even though they are configured at distant frequencies and this behavior should be mitigated by existing RAW mechanism.

The approach followed to minimize the occurring interference between the transmissions is the utilization of NRC7292's parameters, which can be updated by the CLI in real-time without driver rebuild requirements that can lead to latency between the application of parameters and their effect. Specifically, the proposed scheme utilizes the carrier sense time and blank time parameters of NRC7292 CLI and exploits reinforcement learning techniques in order to find an optimal policy for the parameters' configuration in defined channel states. The proposed algorithm uses Q-Learning combined with deep learning and neural networks, creating a Deep Q Network that aims to acquire the optimal CS time and blank time values for PDR and throughput maximization, and channel utilization. The number of epochs is set to 100 and each epoch consists of 60 iterations until the epoch is considered done, which means

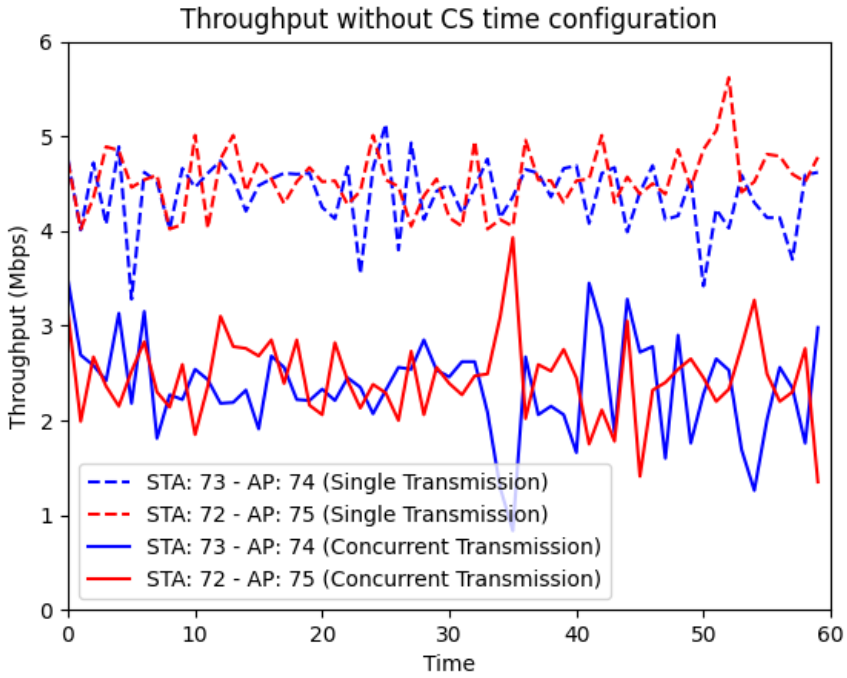

Figure 4.2: Throughput of two STA-AP pairs transmitting in distant channels in 2MHz bandwidth at MCS 7.

that the reward calculated for the current action is above the threhsold of PDR and below the threshold of percentage difference between the pairs of STAs-APs. At most, the training era would require an amount of $|Epochs| \times |Iterations| \times |States|$ calculations of rewards for each possible action.

PDR

Initially, the exploration probability is high, and the DQN agent mostly explores the environment by choosing random actions for each given state. As the training continues for the specified epochs, and more and more states obtain their optimal path of actions, the exploration probability is reduced by a decay number of 0.005 each time. As a result, we would expect that the DQN agent is choosing actions based on the maximum Q values, stored in the memory replay buffer, which leads to both fair and improved PDR for all the nodes participating in the network. The training procedure and its improvement in terms of PDR and fairness can be depicted in figure 4.3, where we have two stations transmitting concurrently to their access points, in different frequencies.

We can observe the same mentioned behavior in our experiment. Firstly, the exploration

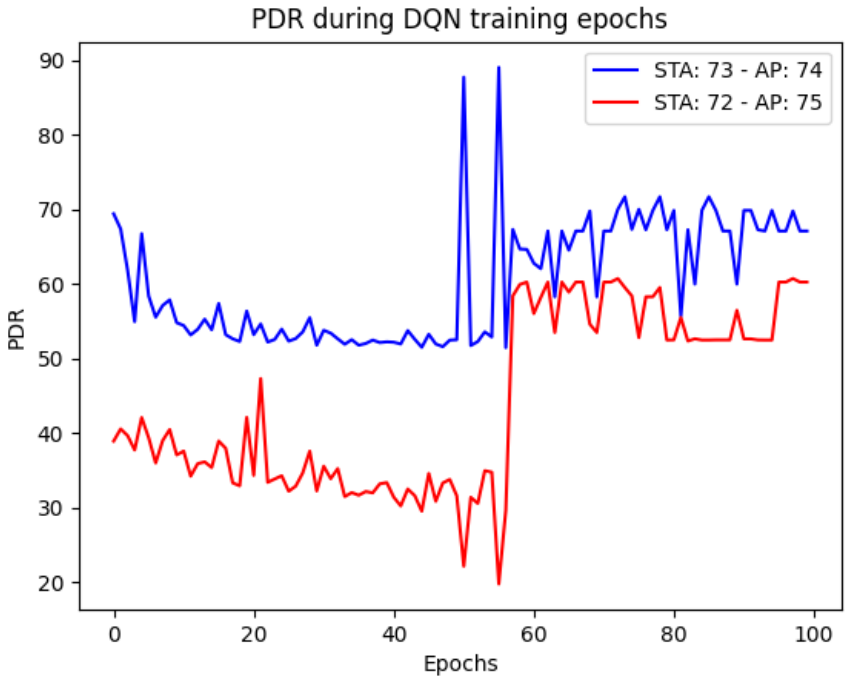

Figure 4.3: PDR optimization during DQN training

probability of the agent is high and the actions are mostly random, which leads to PDR values of 40% and 60% for pairs $72-75$ and $73-74$ respectively. After almost 58 epochs, it is obvious that the training begins to converge at a higher PDR for both nodes from 60% to 70% which is an upgrade of 20% for less strong node and of 10% for the stronger node. Moreover, the percentage difference between the two pairs of nodes decreases from 25% to almost 10%. The training of the DQN agent denotes that after 60 training epochs, the agent will find an optimal configuration for TX time parameters for all possible given states, i.e., combinations of TX time for all nodes.

Throughput

Additionally, the evaluation of this dissertation includes the performance of throughput during DQN training. Throughput is proportional to PDR and its improvement can be depicted in figure 4.4, where it is obvious that it increases after almost 60 training epochs and converges to an optimal result. Initially, throughput of nodes $72-75$ are at $2Mbps$ and of nodes $73-74$ at $3Mbps$, but as the training proceeds, they both converge to an average of $3.5Mbps$ with percentage difference less than or at most equal to 10%. Therefore, the proposed algorithm suggests configurations that lead to an increase of 25% for less sufficient

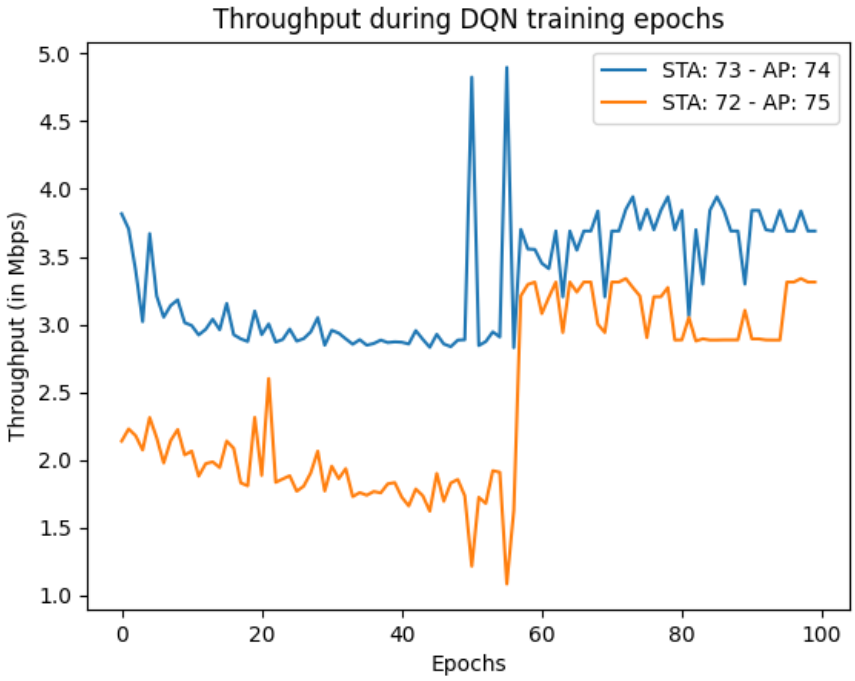

Figure 4.4: Throughput improvement during DQN training

station (72) and of 10% for stronger node in terms of throughput. Moreover, the channel is utilized more, by achieving data rate performance of more than 50% of the maximum value on both pairs of nodes.

Optimal Policy Selection

The major purpose of the proposed algorithm is the selection of an optimal configuration of CS time and blank time values in NRC7292 devices, in order to increase PDR and throughput and impose fairness between the transmitting stations. After converging over a time period of several epochs, the DQN agent is supposed to provide us with an optimal suggested value of CS time and blank time for all pairs of nodes participating in the training network given each possible state of TX time configurations for the nodes. This denotes that starting from each possible combination of TX time states, the agent will result with the optimal value in order to achieve the required thresholds of PDR and fairness.

In the problem scenario this dissertation is studying, it appears to be that station node 73 achieves a little better avergae data rate than station node 72, which can be explained by either enhanced signal strength or improved channel conditions. As the algorithm starts to converge to an improved PDR result, the optimal solution chosen for each possible state across epochs

is selected from the replay memory buffer based on the maximum Q reward value. In our experiment, as mentioned before, the optimal selection of TX time values appears reasonable, since it uses more TX time for back off for more significant node and minimum TX time for captured node, and it can be summarized here:

- For station node 73: CS time $600seconds$, blank time $6000\mu seconds$
- For station node 72: CS time $0seconds$, blank time $0\mu seconds$

The above experiments of evaluating PDR, throughput and optimal policy selection show that our algorithm improves the suffering scenario of interference between IEEE 802.11ah devices set to transmit in close distance but distant frequencies, which was not supposed to exist. Our proposed model tends to increase both PDR and throughput performance by suggesting an optimal policy configuration for TX time parameters, which is achieved by a real-time environment learning through the DQN agent.

Chapter 5

Conclusion

5.1 Summary and Conclusions

Throughout this thesis we presented the low-power wide-range IoT IEEE 802.11ah standard, its capabilities compared to other IoT protocols, but, also, the possible limitations of interference addressed when network topology is dense and nodes are very close to each other in terms of distance. Our interest was focused mainly in leveraging Machine Learning techniques, and especially reinforcement learning, while improving the overall throughput and imposing a fair model for the station nodes of a network.

Our initial experimental evaluation exposed that the behavior of concurrent transmission of different nodes set in distant channels is not the expected, which should maintain the maximum achieved throughput. However, we observed that the data rate in concurrent transmission decreases significantly. To address this limitation, we proposed a combination of Q Learning and Neural Networks that create a Deep Q Network agent which is centralized, collects information about all the nodes' PDR and data rate values and trains a model that leads to the selection of an optimal policy of TX time configurations for NRC7292 devices. Through the proposed DQN agent model evaluation it is observed that the training converges to an enhanced throughput and PDR performance which also leads to fair channel utilization between the stations.

5.2 Future Work

As future work, it is important to study the application of our DQN algorithm approach in more topologies to evaluate its performance and whether the training model will also converge to an optimal configuration policy. Moreover, our plans include the examination of more of NRC7292's CLI parameters in the collisions mitigation problem, such as duty cycle or CCA threshold. Another approach which is worth studying is the parameterization of Enhanced Distributed Channel Access mechanism which is included in RAW algorithm and how the corresponding parameters could affect the interference limitation in dense network topologies.

Bibliography

[1] Wasswa Shafik Lule Sharif Yao Jun, Alisa Craig. Artificial intelligence application in cybersecurity and cyberdefense, 2021.

[2] Abdelkrim Haqiq Hamid Taramit, Luis Orozco-Barbosa. A renewal theory based performance and configuration framework of the ieee 802.11ah raw mechanism, 2023.

[3] Barto-A. G Sutton, R. S. Introduction to reinforcement learning.

[4] Q-table. https://wikidocs.net/174536.

[5] Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network.

[6] Guillaume Köstner Pierre Beckmann and Inês Hipólito. Rejecting cognitivism: Computational phenomenology for deep learning.

[7] Nrc7292 command line application guide. https://github.com/newracom/nrc7292_sw_pkg/blob/master/package/host/doc/UG-7292-007-Commnad%20line%20application.pdf.

[8] Nrc7292 product specifications. https://newracom.com/products/nrc7292.

[9] Elena Lopez-Aguilera Eduard Garcia-Villegas Victor Baños-Gonzalez, M. Shahwaiz Afaqui. Ieee 802.11ah: A technology to face the iot challenge, 2016.

[10] Joanna Paliszkiewicz Jeretta Horn Nord, Alex Koohang. The internet of things: Review and theoretical framework, 2019.

[11] Babu A. V. Sangeetha U. Performance analysis of ieee 802.11ah wireless local area network under the restricted access window-based mechanism, 2018.

[12] Shoukang Zheng Zander Zhongding Lei Yuan Zhou, Haiguang Wang. Advances in ieee 802.11ah standardization for machine-type communications in sub-1ghz wlan, 2013.

[13] Le Tian Jeroen Famaey Adrian Munteanu Ingrid Moerman Jeroen Hoebeke Eli De Poorter Amina Šljivo, Dwight Kerkhove. Performance evaluation of ieee 802.11ah networks with high-throughput bidirectional traffic, 2018.

[14] Steven Latré Le Tian, Jeroen Famaey. Evaluation of the ieee 802.11ah restricted access window mechanism for dense iot networks, 2016.

[15] Peter Dayan Christopher J. C. H. Watkins. Q-learning, 1992.

[16] Google DeepMind. Deep q-learning from demonstrations, 2017.

[17] Andrew W. Moore Leslie Pack Kaelbling, Michael L. Littman. Reinforcement learning: A survey, 1996.

[18] Douglas J. White Chelsea C. White III. Markov decision processes.

[19] K.M. Mohiuddin A.K. Jain, Jianchang Mao. Artificial neural networks: a tutorial, 1996.

[20] Abiodun Esther Omolara Kemi Victoria Dada Nachaat AbdElatif Mohamed Humaira Arshad Oludare Isaac Abiodun, Aman Jantan. State-of-the-art in artificial neural network applications: A survey, 2018.

[21] Leonardo Noriega. Multilayer perceptron tutorial, 2005.

[22] Rajdeep Banerjee Jaswinder Singh. A study on single and multi-layer perceptron neural network, 2019.

[23] LILIANA PERESCU-POPESCU NIKOS MASTORAKIS MARIUS-CONSTANTIN POPESCU, VALENTINA E. BALAS. Multilayer perceptron and neural networks.

[24] Sarah Jane Delany Pádraig Cunningham, Matthieu Cord. Supervised learning.

[25] Anidhya Athaiya Siddharth Sharma, Simone Sharma. Activation functions in neural networks, 2020.

[26] Abien Fred Agarap. A deep learning using rectified linear units (relu), 2018.

[27] B. Goffinet D. Wallach. Mean squared error of prediction as a criterion for evaluating and comparing system models, 1989.

[28] Guanghui Lan Digvijay Boob. Theoretical properties of the global optimizer of two layer neural network, 2017.

[29] Yuchen Xie Zhuoran Yang Jianqing Fan, Zhaoran Wang. A theoretical analysis of deep q-learning, 2020.

[30] V. P. HARIGOVINDAN SRI PAVAN BADARLA. Restricted access window-based resource allocation scheme for performance enhancement of ieee 802.11ah multi-rate iot networks, 2021.

[31] Newracom. `https://newracom.com/`.

[32] Ieee 802.11ah data rates. `https://en.wikipedia.org/wiki/IEEE_802.11ah`.