Open University of Catalonia (UOC) Master's Degree in Data Science

MASTER'S THESIS

Area: 4

Cooperative Contextual Bandits for Wireless Network Optimization

Author: Miguel Casasnovas Bielsa

Tutor: Luis Esteve Elfau

Professor: Ismael Benito Altamirano

Barcelona, May 25, 2025

Credits/Copyright

Attribution-NonCommercial-NoDerivs 3.0 Spain (CC BY-NC-ND 3.0 ES)

3.0 Spain of CreativeCommons.

FINAL PROJECT RECORD

Title of the project:	Cooperative Contextual Bandits for Wireless Network Optimization
Author's name:	Miguel Casasnovas Bielsa
Collaborating teacher's name:	Luis Esteve Elfau
PRA's name:	Ismael Benito Altamirano
Delivery date (mm/yyyy):	05/2025
Degree or program:	Máster Universitario en Ciencia de Datos
Final Project area:	Area 4
Language of the project:	English
Keywords	Multi-Armed Bandits, Reinforcement Learning, Wireless Communications, Medium Access Control

Dedication/Quote

To my family, for their unconditional support.

Acknowledgements

I would like to express my deepest gratitude to my mentors, Luis Esteve (Universitat Oberta de Catalunya, UOC) and Dr. Boris Bellalta (Universitat Pompeu Fabra, UPF), whose invaluable guidance, support, and expertise have been fundamental to the development of this work.

Abstract

Wireless Networks (WNs) are essential for modern communication, providing connectivity for both everyday and emerging applications. With the growing demand for high-reliability, low-latency connectivity, traditional optimization methods fail to adapt to the dynamic nature of these networks. This thesis investigates the application of Contextual Multi-Armed Bandits (CMABs)—a stateless form of Reinforcement Learning (RL)—to enhance the performance of WNs by optimizing decision-making at the Access Points' (APs) Medium Access Control (MAC) layer. In particular, an open-source, event-driven WN simulator has been developed in Python. In the simulator, each AP can function as an independent multi-agent or single-agent learner that learns on the fly. In the multi-agent setting, several CMAB-based agents independently adjust key transmission parameters—specifically, the allocated channel group, primary channel, and contention window size—based on contextual information. Although each agent operates autonomously and is responsible for optimizing a specific functionality, they share a common objective and cooperate to enhance medium access efficiency. In contrast, in the single-agent setting, a single CMAB agent jointly optimizes all the aforementioned parameters of the AP. Our results demonstrate that the cooperative multi-agent architecture consistently outperforms the single-agent approach in terms of network performance, offering also better scalability and reduced complexity—albeit at the cost of increased energy consumption.

Keywords: Multi-Armed Bandits, Reinforcement Learning, Wireless Communications, Medium Access Control

Resumen

Las redes inalámbricas son esenciales para la comunicación moderna, ofreciendo conectividad tanto para aplicaciones cotidianas como emergentes. Con la creciente demanda de conectividad de alta fiabilidad y baja latencia, los métodos tradicionales de optimización no logran adaptarse a la naturaleza dinámica de estas redes. Esta tesis investiga el uso de Contextual Multi-Armed Bandits (CMABs), una variante de aprendizaje por refuerzo sin estado, para mejorar el rendimiento de las redes inalámbricas, optimizando la toma de decisiones en la capa de Control de Acceso al Medio (MAC) de los puntos de acceso. En particular, se ha desarrollado un simulador de redes inalámbricas de código abierto y basado en eventos con Python. En el simulador, cada punto de acceso puede operar como un aprendiz independiente que aprende de manera continua. Específicamente, en la capa MAC de cada AP puede operar un único agente o múltiples agentes. En el caso multiagente, diversos agentes basados en CMABs ajustan de manera independiente parámetros clave de transmisión (como el grupo de canales asignados, el canal primario y el tamaño de la ventana de contención), utilizando información contextual. Aunque cada agente actúa de manera autónoma y se encarga de optimizar una funcionalidad específica, todos comparten un objetivo común y colaboran para mejorar la eficiencia del acceso al medio. En contraste, en el enfoque de un solo agente, un único agente CMAB optimiza todos los parámetros mencionados. Nuestros resultados demuestran que la arquitectura con múltiples agentes cooperando supera significativamente al enfoque con un solo agente en términos de rendimiento de la red, ofreciendo además mejor escalabilidad y menor complejidad, aunque a costa de un mayor consumo de energía.

Palabras clave: 'Multi-Armed Bandits', Aprendizaje por Refuerzo, Comunicaciones Inalámbricas, Control de Acceso al Medio

Contents

List of Figures

List of Tables

Acronyms

A-MPDU	Aggregated Medium Access Control (MAC) Protocol Data Unit
A-MSDU	Aggregated Medium Access Control (MAC) Service Data Unit
A3C	Asynchronous Advantage Actor Critic
ACK	Acknowledgment
AM	Always-max
AP	Access Point
BACK	Block Acknowledgment
BEB	Binary Exponential Backoff
BSS	Basic Service Set
CAP	Credit-Assignment Problem
CB	Channel Bonding
CCA	Clear Channel Assessment
CMAB	Contextual Multi-Armed Bandit
CMAE	Coordinated Multi-Agent Exploration
Coo-MARL	Cooperative Multi-Agent Reinforcement Learning
COSB	Channel Observation-based Scaled Backoff
CPU	Central Processing Unit
CRC	Common Retry Count
CSA	Channel Selection Agent

CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSV	Comma-Separated Values
CTDE	Centralized Training and Decentralized Execution
CTE	Centralized Training and Execution
CTMC	Continuous-Time Markov Chain
CTS	Clear to Send
CW	Contention Window
CWSA	Contention Window Selection Agent
DCB	Dynamic Channel Bonding
DCF	Distributed Coordination Function
DDPG	Deep Deterministic Policy Gradient
DIFS	Distributed Inter-frame Space
DL	Downlink
DQN	Deep Q-Network
DTE	Decentralized Training and Execution
ED	Energy Detect
EDCA	Enhanced Distributed Channel Access
EIFS	Extended Inter-Frame Space
EMA	Exponential Moving Average
FEC	Forward Error Correction
FCS	Frame Check Sequence
GPU	Graphics Processing Unit
GI	Guard Interval
HQL	Hysteretic Q-Learning

i.i.d.	independent and identically distributed
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IQL	Independent Q-Learning
IQR	Interquartile Range
IP	Internet Protocol
JSON	JavaScript Object Notation
LBT	Listen Before Talk
MA-CMAB	Multi-Agent Contextual Multi-Armed Bandit
MA-MAB	Multi-Agent Multi-Armed Bandit
MAB	Multi-Armed Bandit
MAC	Medium Access Control
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MAPC	Multi-AP Coordination
MAPPO	Multi-Agent Proximal Policy Optimization
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent System
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MP-MAB	Multi-Player Multi-Armed Bandit
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit

MSE	Mean Squared Error
NAV	Network Allocation Vector
NLOS	Non-Line-of-Sight
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OP	Only-Primary
OSI	Open Systems Interconnection
PCF	Point Coordination Function
PCSA	Primary Channel Selection Agent
PER	Packet Error Rate
PHY	Physical
PIFS	Point Coordination Function (PCF) Inter-frame Space
POMDP	Partially Observable Markov Decision Process
POSG	Partially Observable Stochastic Game
PPDU	PHY Protocol Data Unit
PPO	Proximal Policy Optimization
PU	Probabilistic Uniform
QoS	Quality of Service
RL	Reinforcement Learning
RSSI	Received Signal Strength Indicator
RTS	Request To Send
SA-CMAB	Single-Agent Contextual Multi-Armed Bandit
SARL	Single-Agent Reinforcement Learning
SCB	Static Channel Bonding

SD	Signal Detect
SDG	Sustainable Development Goal
SGD	Stochastic Gradient Descent
SIFS	Short Inter-frame Space
SINR	Signal-to-Interference-and-Noise Ratio
SNR	Signal-to-Noise Ratio
SRO	Spatial Reuse Operation
SS	Spatial Streams
STA	Station
T-CAP	Temporal Credit Assignment Problem
TCP	Transmission Control Protocol
TD	Temporal Difference
TPE	Tree-structured Parzen Estimator
TS	Thompson Sampling
TWT	Target Wake Time
UCB	Upper Confidence Bound
UDP	User Datagram Protocol
UL	Uplink
UOC	Universitat Oberta de Catalunya
UPF	Universitat Pompeu Fabra
VR	Virtual Reality
W&B	Weights & Biases
WLAN	Wireless Local Area Network
WN	Wireless Network
XR	eXtended Reality

Chapter 1

Introduction

1 Description

The rising demand for high-reliability, low-latency wireless connectivity to support bandwidth-intensive applications, such as eXtended Reality (XR), has highlighted the need for adaptive and efficient network optimization strategies as traditional network management methods, based on predefined rules or heuristics, struggle to adapt to the dynamic and unpredictable nature of these networks [8, 9].

Reinforcement Learning (RL), a subfield of Machine Learning (ML), has emerged as a promising approach for optimizing Wireless Networks (WNs) [10–12], enabling networks to learn and adapt dynamically based on feedback from the environment. In RL an agent learns through interactions with the environment, taking actions based on its observations and receiving rewards based on the outcome of its actions [13]. Compared to supervised and unsupervised learning, RL is more suitable for uncertain environments such as WNs [14].

Recent research has increasingly explored RL-based solutions for WN optimization [10, 15], focusing on spectrum allocation [14] and parameter tuning [16] in the Medium Access Control (MAC) protocol, among others. The MAC protocol—a fundamental component of the Data Link layer (layer 2 in the Open Systems Interconnection (OSI) model)—is responsible for controlling access to the shared communication medium, managing medium contention, and ensuring reliable and efficient data transmission. A particularly promising approach is Multi-Agent Reinforcement Learning (MARL), a subfield of RL in which multiple learning agents operate in a shared environment and either cooperate to maximize a common long-term reward or compete against each other [17]. In the context of WNs, as the MAC protocol involves multiple configurable and interdependent parameters [16], Cooperative Multi-Agent Reinforcement Learning (Coo-MARL) is essential for optimizing network performance [18], as agents collaborate and share information to effectively achieve a common goal.

In Single-Agent Reinforcement Learning (SARL), a single entity learns optimal policies by interacting with the environment. Thus, as the number of functionalities to optimize increases, the agent's action space becomes larger and more complex, significantly increasing computational demand. In contrast, Coo-MARL distributes the optimization task across multiple agents. Each agent focuses on a subset of functionalities, which results in a comparatively smaller action space, leading to faster convergence and enhanced problem-solving efficiency [16]. Nevertheless, Coo-MARL may introduce concerns related to coordination, information transfer, partial observability, and non-stationarity [17, 19, 20].

In this thesis, we develop a simulation-based system that supports both Multi-Agent Contextual Multi-Armed Bandits (MA-CMABs) and Single-Agent Contextual Multi-Armed Bandits (SA-CMABs)—stateless forms of MARL and SARL, respectively—for optimizing MAC-layer decision-making in WNs. In the multi-agent setting, each network Access Point (AP) may host multiple Contextual Multi-Armed Bandit (CMAB)-based agents at the MAC layer. Each agent autonomously adjusts a specific transmission parameter—namely, the allocated channel group, primary channel, and Contention Window (CW) size—based on local contextual information from both the AP and its surrounding wireless environment. Although operating independently, agents share a common objective and learn cooperatively to enhance medium access efficiency. In contrast, in the single-agent setting, the MAC layer hosts a single contextual bandit agent responsible for jointly selecting the aforementioned parameters. Our hypothesis is that a cooperative multi-agent system can achieve comparable or superior performance to a single-agent architecture, particularly in terms of network efficiency and convergence speed, although potentially at the cost of increased energy consumption. Thus, this research contributes to the ongoing exploration of ML-driven solutions in WNs, demonstrating the potential of MARL for optimizing MAC-layer functionalities. Additionally, it provides an open-source WN simulator incorporating CMAB-based architectures, facilitating further research and development in this field.

2 Personal motivation

Holding a Bachelor's degree in Telecommunication Network Engineering and currently pursuing a Master's degree in Data Science, this thesis offers an opportunity to apply my academic background and expertise in both fields by utilizing machine learning techniques to optimize

WN transmissions. Furthermore, it complements my ongoing research in the UPF's Wireless Networking group, contributing to the *MLDR: A Machine Learning-Driven Radio Interface project*[1].

3 Goals

This thesis aims to develop and evaluate a Coo-MARL framework for optimizing WN transmissions using MA-CMABs. The proposed system involves multiple intelligent agents operating at the link layer, each responsible for optimizing a specific transmission parameter—namely, the allocated channel group, primary channel, or CW size. These agents dynamically adjust their decisions in real time (*online learning*) based on contextual information to optimize channel access efficiency. Although each agent operates autonomously, agents collaboratively learn towards a common goal.

Thus, the primary research question is: *Can cooperative MA-CMAB-based approaches enhance WN transmission efficiency?*

To address this question, the following secondary objectives are outlined: (*i*) to design and implement an event-driven WN simulator in Python; (*ii*) to implement and fine-tune MAB-based agents capable of optimizing transmission parameters, ensuring cooperation towards a common goal; and (*iii*) to compare the proposed MA-CMAB architecture against a SA-CMAB baseline in terms of complexity, convergence speed, and performance, assessing whether multi-agent cooperation improves throughput, latency, and overall efficiency compared to traditional single-agent approaches. The scope of this thesis is limited to link-layer optimization in WNs and relies entirely on simulations, without implementation in real-world hardware. The simulator abstracts certain complexities inherent to physical network deployments.

4 Sustainability, diversity, and ethical/social challenges

Sustainability This research presents both positive and negative sustainability implications. On the one hand, optimizing WN transmissions can reduce energy consumption, improve spectrum efficiency, and enhance airtime utilization by minimizing congestion and retransmissions. As a result, it contributes to several United Nations' Sustainable Development Goals (SDGs) [21]: SDG 9 (Industry, Innovation, and Infrastructure) and SDG 12 (Responsible Consumption and Production). On the other hand, machine learning models

[1] https://www.chistera.eu/projects/mldr

require significant computational resources, increasing energy consumption and environmental impact (SDG 13: Climate Action). To mitigate this, the project focuses on optimizing model efficiency and monitors power consumption and carbon emissions using CodeCarbon[2].

Ethical behavior and social responsibility This project has minimal impact on ethical or social aspects given its technical nature. It does not negatively impact employment, as network optimization is typically handled autonomously, and adheres to professional ethical guidelines, ensuring transparency, fairness, and integrity in the system's design and implementation. Furthermore, data is either synthetically generated or gathered in a controlled environment, ensuring privacy and compliance with intellectual property rights.

Diversity, gender, and human rights This project does not directly impact diversity, gender equality, or human rights. Nevertheless, the thesis is publicly available, promoting accessibility irrespective of race, gender, or socioeconomic status, thereby contributing to SDG 10 (Reduced Inequalities). Additionally, the agents learn through trial-and-error interactions with a simulated environment rather than relying on potentially biased datasets, promoting fairness and inclusivity. Finally, by fostering open collaboration and knowledge sharing, the project supports SDG 4 (Quality Education).

5 Methodology

This thesis adopts an **Agile** methodology, organizing the project into iterative sprints and utilizing a Kanban board and a product backlog to ensure continuous progress and efficient task management. The tools used in this project include:

Project Planning and task management TeamGantt[3] is used to create a Gantt chart (see Fig. 1.1), providing a clear visual representation of the project timeline. Notion[4] is leveraged for managing documentation, organizing meeting notes, structuring tasks, and tracking progress through a Kanban board.

Literature review and reference management Relevant literature is sourced from academic databases such as Institute of Electrical and Electronics Engineers (IEEE) Xplore[5],

[2]https://codecarbon.io/
[3]https://www.teamgantt.com/
[4]https://www.notion.com/
[5]https://ieeexplore.ieee.org/

Google Scholar[6], and Arxiv[7]. Zotero[8] is used for reference management, enabling efficient organization of research papers.

Programming language and simulation environment Python[9] is the programming language used. SimPy[10] is used for discrete, asynchronous, event-driven simulation.

Development and debugging Visual Studio Code (VS Code)[11] serves as the Integrated Development Environment (IDE) used for coding, debugging, and testing. GitHub[12] is used for version control. In addition, our implementation is available as open-source on this platform.

Network traffic capture Wireshark[13] is used to capture real-world traffic traces, providing valuable input data for simulating realistic traffic generation in the simulator.

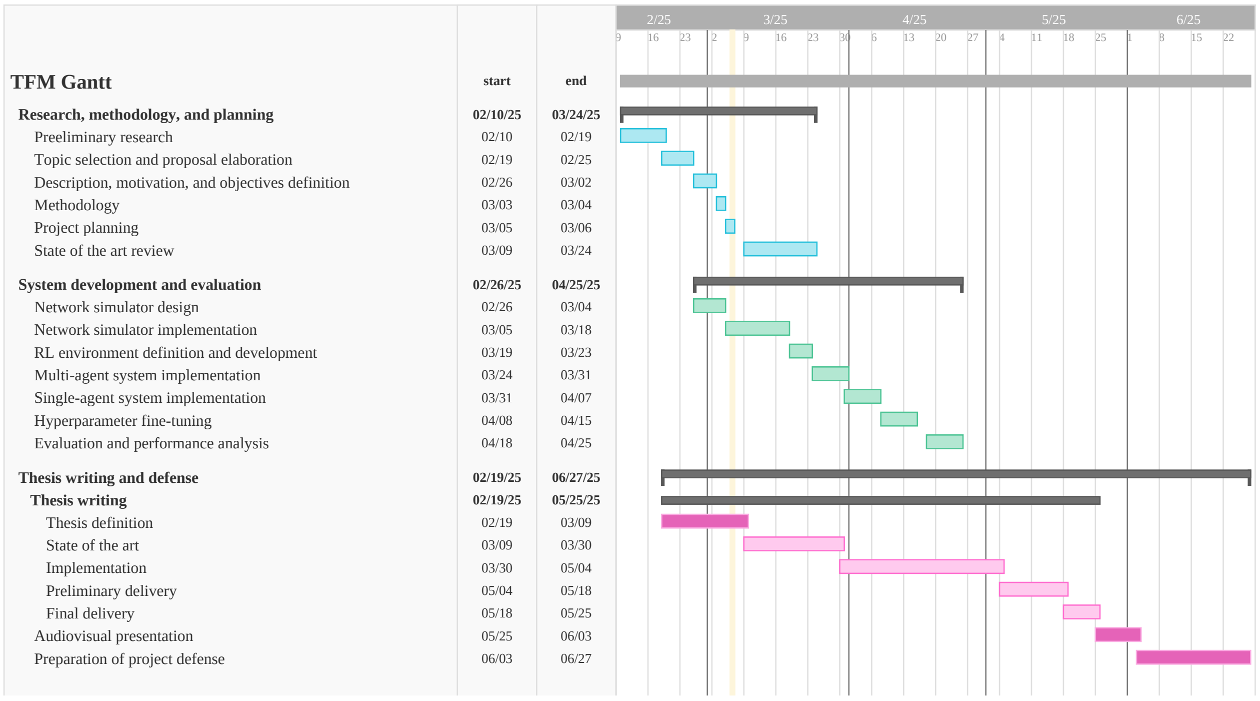
Tracking, monitoring, and tuning Weights & Biases (W&B)[14] is used to track experiments and log metrics, enabling real-time visualizations and comparisons across simulation runs. Optuna[15] is utilized for hyperparameter fine-tuning, automating the optimization process. CodeCarbon[2] is integrated to monitor energy consumption and estimate the carbon footprint.

6 Project planning

This thesis is divided into three main phases spanning around 19 weeks (see Fig. 1.1), each including multiple tasks:

Research, methodology, and planning This phase involves conducting preliminary research to define the thesis topic, establish the motivation, formulate research questions, and set clear objectives. It also includes developing a project plan with specific milestones for each phase, defining the required methodology and tools, and performing a comprehensive literature review to contextualize the work and establish its scientific foundation.

[6] https://scholar.google.com/
[7] https://arxiv.org/
[8] https://www.zotero.org/
[9] https://www.python.org/
[10] https://simpy.readthedocs.io/en/latest/index.html
[11] https://code.visualstudio.com/
[12] https://github.com/
[13] https://www.wireshark.org/
[14] https://wandb.ai/site/
[15] https://optuna.org/



TFM Gantt	start	end
Research, methodology, and planning	**02/10/25**	**03/24/25**
Preeliminary research	02/10	02/19
Topic selection and proposal elaboration	02/19	02/25
Description, motivation, and objectives definition	02/26	03/02
Methodology	03/03	03/04
Project planning	03/05	03/06
State of the art review	03/09	03/24
System development and evaluation	**02/26/25**	**04/25/25**
Network simulator design	02/26	03/04
Network simulator implementation	03/05	03/18
RL environment definition and development	03/19	03/23
Multi-agent system implementation	03/24	03/31
Single-agent system implementation	03/31	04/07
Hyperparameter fine-tuning	04/08	04/15
Evaluation and performance analysis	04/18	04/25
Thesis writing and defense	**02/19/25**	**06/27/25**
Thesis writing	**02/19/25**	**05/25/25**
Thesis definition	02/19	03/09
State of the art	03/09	03/30
Implementation	03/30	05/04
Preliminary delivery	05/04	05/18
Final delivery	05/18	05/25
Audiovisual presentation	05/25	06/03
Preparation of project defense	06/03	06/27

Figure 1.1: Project timeline represented as a Gantt chart. *Source: Self-elaborated using TeamGantt*

System development and evaluation This phase focuses on designing and implementing a modular network simulator. It also involves defining the observation spaces, action spaces, and reward functions for both single-agent and multi-agent architectures. Furthermore, during this phase, contextual bandit algorithms are developed, fine-tuned, and evaluated based on throughput, latency, convergence speed, and energy efficiency.

Thesis writing and defense This phase spans the entire project, adhering to the mandatory submission schedules. It encompasses the continuous writing of the thesis document, the creation of the audiovisual presentation, and the preparation for the defense.

7 Structure

The remainder of the thesis is structured as follows: Chapter 2 provides background on RL (including MABs) and MARL, along with a review of related work. Chapter 3 details the system design and implementation. Chapter 4 evaluates the single-agent and multi-agent architectures, including hyperparameter tuning. Chapter 5 concludes this work, summarizing its contributions and suggesting future research directions.

Chapter 2

State of the art

This chapter provides an overview of the fundamental concepts of Reinforcement Learning (RL) and Multi-Agent Reinforcement Learning (MARL), followed by a review of existing research on the optimization of Wireless Networks (WNs) using RL and MARL techniques.

1 Reinforcement Learning

1.1 Introduction

Reinforcement Learning (RL) is a Machine Learning (ML) paradigm where an '*agent* learns through *trial-and-error* interactions with a dynamic *environment*' [22] to maximize its cumulative reward over time, also known as the *return*. Unlike other learning paradigms, such as *supervised learning* (which learns patterns and relationships from labeled data to make inferences on unseen data [23]) or *unsupervised learning* (which identifies inherent patterns, structures, and relationships from unlabeled data [24]), Reinforcement Learning (RL) relies on interactions with an uncertain environment, following a *sequential decision-making process* where actions influence future states, rewards, and subsequent actions. Consequently, an Reinforcement Learning (RL) agent's observations are non-independent and non-identically distributed [13, 25]. Moreover, in Reinforcement Learning (RL) there is no explicit supervision indicating the correct action [2, 22, 25]; instead, the agent learns solely from reward signals, which may be delayed, meaning that the consequences of an action are not always immediately observable [1, 13, 25].

1.2 Definition of the problem

A *fully observable* Reinforcement Learning (RL) problem is formally modeled as a Markov Decision Process (MDP) [1, 25], as an MDP represents sequential decision-making problems with uncertain outcomes. An MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ [25], where:

$\mathcal{S}$ is the *state space*, the set of all possible states.

$\mathcal{A}$ is the *action space*, the set of actions available to the agent.

$\mathcal{P}$ is the *state transition matrix*, defining the probabilities of transitioning from state s to a state s' given action a:

$$\mathcal{P}_{ss'}^{a} = \Pr\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

$\mathcal{R}$ is the *reward function*, defining the expected reward after taking action a in state s:

$$\mathcal{R}_{s}^{a} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

An MDP satisfies the *Markov property*, meaning that the future state S_{t+1} depends only on the present state S_t and not on past states, i.e., 'the future is independent of the past given the present' [25].

Definition 2.1 (Markov Property) A state S_t satisfies the *Markov property* if and only if

$$\Pr\left[S_{t+1} \mid S_t\right] = \Pr\left[S_{t+1} \mid S_1, \ldots, S_t\right]$$

In contrast, a *partially observable* RL problem is formally modeled as a Partially Observable Markov Decision Process (POMDP) [25], a generalization of an MDP where the agent can only partially observe the state of the environment. A POMDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z} \rangle$ [25], where:

$\mathcal{S}$, $\mathcal{A}, \mathcal{P}, \mathcal{R}$ retain the same definitions as in the MDP model.

$\mathcal{O}$ is the *observation space*, the set of possible observations the agent can perceive from the environment, offering partial information about its true state.

$\mathcal{Z}$ is the *observation function*, defining the probability of observing o given state s' and action a:

$$\mathcal{Z}_{s'o}^{a'} = \Pr\left[O_{t+1} = o \mid S_{t+1} = s', A_t = a\right]$$

1.3 Agent-Environment interaction

RL is a sequential decision-making process in which an *agent* interacts with an *environment* over a discrete sequence of time steps. As illustrated in Fig. 2.1, at each time step t, the agent

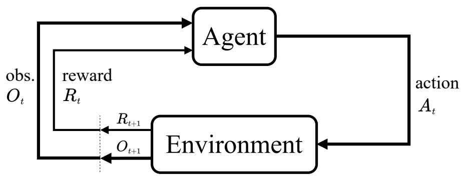


Figure 2.1: The agent-environment interaction loop. *Source: Self-elaborated, adapted from [1].*

perceives an *observation* $O_t \in \mathcal{O}$ of the environment's state S^e_t and selects an *action* $A_t \in \mathcal{A}$ according to its *policy* π. The environment, upon receiving the agent's action, transitions to a new state S^e_{t+1} and provides feedback in the form of a *new observation* O_{t+1} and a *scalar reward* R_{t+1}, which serves as a numerical evaluation of the agent's chosen action. This cycle continues until a terminal condition is satisfied or indefinitely in ongoing tasks. [13, 25]

Thus, every RL problem is fundamentally composed of four elements: the *states*, the *environment*, and the *reward signal* that define the problem; and the *agent* that represents the solution. [13]

1.4 Core elements

1.4.1 States

The *environment state* S^e_t is the environment's internal (and private) representation at time step t and contains all information required to determine the next observation and reward. Since the agent does not have direct access to S^e_t, it relies on an *observation* O_t, which provides a (possibly partial) description of the environment's current state, and, to determine its actions, it maintains an internal representation known as the *agent state* S^a_t, which can be any function of the agent's *history*[1] H_t. [13, 25]

1.4.2 Environment

The *environment* encompasses everything external to the agent. [13, 25] Depending on the observability of S^e_t, environments can be classified as:

- A *fully observable environment* in which the agent can directly observe the environment state, i.e., $O_t = S^e_t = S^a_t$. In this case, the RL problem is modeled as an MDP.
- A *partially observable environment* in which the agent perceives only partial or noisy

[1]The history $H_t = \{O_0, A_0, R_1, O_1, A_1, \ldots, A_{t-1}, R_t, O_t\}$ is the sequence of observations, actions, and rewards up to time t.

information from the environment state, i.e., $S_t^e \neq S_t^a$. This corresponds to a POMDP, where the agent must infer missing information using, for instance:

- A *history-based representation* in which the agent maintains the complete history:

$$S_t^a = H_t$$

- A *Bayesian belief state* in which the agent maintains a probability distribution over possible environment states:

$$S_t^a = (\Pr[S_t^e = s_1], \ldots, \Pr[S_t^e = s_n])$$

1.4.3 Reward Signal

The *reward signal* $R_t \in \mathbb{R}$ is a scalar value that the agent receives at time step t after taking an action, providing feedback on the immediate desirability of its chosen action. [1] Notably, RL is fundamentally based on the *reward hypothesis*:

Definition 2.2 (Reward Hypothesis) All goals can be formulated as the maximization of the expected cumulative reward.

Thus, the RL agent's goal is to maximize the *expected cumulative reward* rather than immediate rewards. Since actions have long-term consequences, learning an optimal behavior may require sacrificing short-term rewards in favor of greater future returns.

For *episodic tasks*, where interactions eventually reach a terminal state, the *return* at time step t, denoted G_t, is defined as the cumulative sum of future rewards until the final time step T:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

For *continuing tasks*, where interactions continue indefinitely, the return may not converge. Thus, future rewards are discounted using a *discount factor* $\gamma \in [0, 1]$ that determines how much future rewards contribute to the return, leading to the *discounted return*:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

that can also be expressed recursively:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

1.4.4 Agent

The *agent* is the decision-making entity, responsible for selecting actions at each time step and learning optimal behavior through trail-and-error interactions with the environment. The agent typically incorporates one or more of the following components: an explicit or implicit *policy*, an explicit or implicit *value function*, and, optionally, a *model* of the environment. [13, 25]

A policy $\pi(\cdot)$ defines the agent's behavior, mapping states to actions. It can be either *deterministic* or *stochastic* depending on the context and problem:

- A *deterministic policy* maps each state $s \in \mathcal{S}$ to a single action $a \in \mathcal{A}$, meaning that the agent always selects the same action in a given state:

$$\pi(s) = a$$

- An *stochastic policy* introduces randomness into the decision-making process, facilitating exploration. In particular, it maps states to probability distributions over possible actions, meaning the agent selects an action probabilistically:

$$\pi(a \mid s) = \Pr[A_t = a \mid S_t = s]$$

Value Functions estimate "how good" (in terms of expected return) it is for an agent to be in a given state (or to take a specific action in a state) under a specific policy [1]:

- An *state-value function for policy* π, denoted as $v_\pi(s)$, estimates the expected return from starting in state s and thereafter following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \quad \forall s \in \mathcal{S},$$

  where $\mathbb{E}_\pi[\cdot]$ denotes the expectation under policy π.

- An *action-value function for policy* π, denoted as $q_\pi(s, a)$, estimates the expected return from taking an action a in state s and thereafter following policy π:

$$q_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \tag{2.1}$$

A model predicts the environment's behavior, enabling the agent to anticipate the next state and reward and, therefore, act accordingly. [13] Models facilitate *planning*, which involves simulating future states and rewards, rather than relying solely on trial-and-error interactions. Nevertheless, a ground-truth model of the environment is usually not available. [26] The model of the environment includes:

- The *Transition model* $\mathcal{P}$, which predicts the next environment state given a state-action pair:
$$\mathcal{P}^a_{ss'} = \Pr \left[S_{t+1} = s' \mid S_t = s, A_t = a \right]$$
- The *Reward model* $\mathcal{R}$, which predicts the immediate reward resulting from a state-action pair:
$$\mathcal{R}^a_s = \mathbb{E} \left[R_{t+1} \mid S_t = s, A_t = a \right]$$

Agents can be categorized based on the use of policies and value functions as: *value-based agents*, which use an explicit value function but an implicit policy; *policy-based agents*, which use an explicit policy but an implicit value function; or *actor-critic agents*, which use both an explicit policy and value function. Additionally, agents can be classified based on the use of a model of the environment as: *model-free agents*, which learn directly from trial-and-error interactions, relying only on policies and/or value functions; or *model-based agents*, which learn a model of the environment to predict its response to actions, enabling planning and enhancing decision-making.

1.5 Solutions: optimal policy and optimal value functions

Solving an RL problem involves determining the *optimal policy*[2] π_*, which maximizes the expected return for the agent, and thus, the goal is to find the *optimal state-value function* v_* and the *optimal action-value function* q_* [1, 27]:

$$v_*(s) = \max_\pi v_\pi(s) \quad \forall s \in \mathcal{S}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a) \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

[2]Note that there can be multiple optimal policies, sharing the same (optimal) value functions.

The optimal value functions v_* and q_* are interrelated through the *Bellman optimality equations* [28]:

$$\begin{aligned} v_*(s) &= \max_{a\in\mathcal{A}} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] && \text{(by (2.1))} \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \end{aligned}$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right]$$

Notably, once the optimal action-value function $q_*(s,a)$ is determined, the optimal policy π_* can be easily derived from selecting the actions that maximize $q_*(s,a)$ for each state [27]:

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a'\in\mathcal{A}} q_*(s,a') \\ 0 & \text{otherwise} \end{cases}$$

In cases where multiple actions yield the same maximal value, the optimal policy may assign arbitrary probabilities to these actions, ensuring that the sum of their probabilities equals one.

1.6 The exploration-exploitation dilemma

A fundamental challenge in RL is the *exploration-exploitation dilemma*, which represents the trade-off between *exploration* and *exploitation*.

- *Exploitation* involves selecting actions that have previously resulted in high rewards based on the agent's current knowledge. While exploitation maximizes immediate returns, it may prevent the agent from discovering better long-term strategies, leading to suboptimal policies. [2]

- *Exploration*, on the other hand, involves trying new actions. Although this may reduce short-term returns, exploration enables the agent to discover better strategies that could improve its future decision-making. [2, 13]. However, excessive exploration can lead to inefficiency, delaying the agent's learning process and increasing the time required to converge to an optimal policy.

In a planning context, exploration involves trying actions that may improve the agent's internal model of the environment, while exploitation involves following the optimal policy according to the current model. The challenge lies in striking a balance between sufficient exploration to refine the model and enough exploitation to ensure performance. [1] Thus, effectively managing the exploration-exploitation trade-off is essential.

1.7 Multi-Armed Bandits

The Multi-Armed Bandit (MAB) problem (a.k.a. k-armed bandit problem [29, 30]) is considered a simplified, *stateless* form of RL [31], as there is no notion of state or state transitions.

A MAB problem can be interpreted as a one-state MDP. Formally, it is defined by the tuple $\langle \mathcal{A}, \mathcal{R} \rangle$ [32], where:

$\mathcal{A}$ is the *action space*, consisting of k discrete actions (also called *arms*).

$\mathcal{R}$ is a set of unknown reward distributions $\{\mathcal{R}^a\}_{a \in \mathcal{A}}$, where each $\mathcal{R}^a$ defines the stochastic reward distribution associated with pulling arm a, i.e., $\mathcal{R}^a = \Pr[R_t \mid A_t = a]$.

MABs are analogous to playing multiple one-armed bandits (a.k.a. slot machines). At each discrete time step $t \in \{1, 2, \dots, T\}$, the agent selects an action $A_t \in \mathcal{A}$ and receives a reward R_t. Rewards are assumed to be independent and identically distributed (i.i.d.) for each arm [30].

Unlike full RL problems, there is no notion of state, and the agent's actions do not influence future interactions. Each action-reward pair is independent of previous choices, reducing the agent's objective to learning the best arm based solely on observed outcomes.

Let $\mu_a = \mathbb{E}[R_t \mid A_t = a]$ denote the expected reward of arm a, and let $a^* = \arg\max_{a \in \mathcal{A}} \mu_a$ be the optimal arm. The agent's goal is to maximize the expected cumulative reward $\mathbb{E}\left[\sum_{t=1}^{T} R_t\right]$, or equivalently, to minimize the expected *regret* [29], which measures the gap between the agent's accumulated reward and the reward it would have earned by consistently choosing the best arm:

$$\mathrm{Reg}_T = T\mu_{a^*} - \mathbb{E}\left[\sum_{t=1}^{T} R_t\right] = \sum_{a \in \mathcal{A}} \Delta_a \, \mathbb{E}[N_a(T)],$$

where $\Delta_a = \mu_{a^*} - \mu_a$ is the *suboptimality gap* of arm a, and $N_a(T)$ is the number of times arm a has been selected up to time T. [29]

In practice, solving a MAB problem involves designing an algorithm that specifies the agent's action-selection strategy. Popular approaches include ϵ-greedy, Upper Confidence Bound (UCB) [33], and Thompson Sampling (TS) [34], each approaching the exploration-exploitation trade-off differently.

ϵ-greedy uses a probability parameter $\epsilon \in [0, 1]$ to control the trade-off between choosing actions based on the current knowledge (exploitation) and exploring new actions (exploration).

Typically, ϵ decays over time, allowing the agent to explore more in the early stages of learning and gradually shift towards exploitation as the estimates of the expected rewards converge. In particular, with probability ϵ, the agent selects a random action (exploration) and with probability $1 - \epsilon$, the agent selects the action that with the highest estimated reward (exploitation). The ϵ-greedy policy ensures that non-greedy actions (i.e., actions that do not maximize the expected reward estimate) are chosen with a probability $\frac{\epsilon}{|\mathcal{A}(s)|}$, where $|\mathcal{A}(s)|$ is the number of possible actions for state s [1]. Thus, the ϵ-greedy policy is formally defined as:

$$\pi(a) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|\mathcal{A}|} & \text{if } a = \arg\max\limits_{a' \in \mathcal{A}} \hat{\mu}_{a'} \\ \frac{\epsilon}{|\mathcal{A}|} & \text{if } a \neq \arg\max\limits_{a' \in \mathcal{A}} \hat{\mu}_{a'} \end{cases}$$

where $\hat{\mu}_a$ represents the estimated expected reward of arm a.

UCB selects actions optimistically based on confidence bounds. Unlike ϵ-greedy, which explores actions indiscriminately, UCB favors arms that either have high observed rewards (assumed to be bounded in $[0, 1]$) or have been explored less frequently. Formally, the agent selects:

$$a_t = \arg\max_{a \in \mathcal{A}} \left[\hat{\mu}_a(t) + \sqrt{\frac{\alpha \ln t}{2N_a(t)}} \right],$$

where $\hat{\mu}_a(t)$ (often denoted $Q_t(a)$ [1]) is the empirical mean reward of arm a up to time t (i.e., the agent's current estimate of the expected reward for arm a), $N_a(t)$ is the number of times arm a has been selected, and $\alpha > 0$ is a tunable parameter that controls the degree of exploration. [32, 35] For instance, UCB1 [33] uses $\alpha = 4$. The square-root term quantifies the uncertainty in the value estimate of arm a [1], encouraging the algorithm to select less-explored arms more frequently but diminishing exploration over time.

1.8 Contextual Multi-Armed Bandits

The Contextual Multi-Armed Bandit (CMAB) problem (a.k.a. associative search problem [1, 36]) is a generalization of the classical MAB problem, in which the agent observes *contextual information* (or *features*) prior to each action [30, 31]. This additional information allows the agent to tailor its actions to the observed context, enabling more situation-specific decision making.

Formally, a CMAB problem is defined by the tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{R} \rangle$ [37], where:

$\mathcal{A}$ is the *action space*, consisting of k discrete actions (arms).

$\mathcal{X}$ is the *context space*, from which a d-dimensional context vector $\mathbf{X}_t \in \mathcal{X}$ is obtained at each time step t.

$\mathcal{R}$ is a set of conditional reward distributions $\{\mathcal{R}_x^a\}_{a\in\mathcal{A},\mathbf{x}\in\mathcal{X}}$, where $\mathcal{R}_x^a$ defines the reward distribution for arm a given context $\mathbf{x}$, i.e., $\Pr[R_t \mid \mathbf{X}_t = \mathbf{x}, A_t = a]$.

At each time step $t \in \{1, 2, \ldots, T\}$, the agent observes a context $\mathbf{x}_t \in \mathcal{X}$, selects an action $a_t \in \mathcal{A}$, and receives a reward r_t. Rewards are assumed to be conditionally independent given the context and action.

Unlike standard MABs, where the objective is to identify the best overall arm, the goal in CMABs is to learn a *context-dependent policy*, $\pi(\mathbf{x}) = \arg\max_{a\in\mathcal{A}} \mathbb{E}[R_t \mid \mathbf{X}_t = \mathbf{x}, A_t = a]$, mapping each observed context $\mathbf{x}$ to the optimal action a [38]. Thus, CMABs are an intermediate problem between (context-free) MABs and full RL, as agents learn a policy but their actions only affect immediate rewards and not future states [1].

Let $\mu_a(\mathbf{x}) = \mathbb{E}[R_t \mid A_t = a, \mathbf{X}_t = \mathbf{x}]$ denote the expected reward of arm a given context $\mathbf{x}$ and let $a^* = \arg\max_{a\in\mathcal{A}} \mu_a(\mathbf{x})$ be the optimal arm for context $\mathbf{x}$. The agent's goal is to maximize the expected cumulative reward $\mathbb{E}\left[\sum_{t=1}^T R_t\right]$, or equivalently, to minimize the expected *contextual regret*, defined as the gap between the agent's policy and the optimal policy that selects the best arm for each context:

$$\mathrm{Reg}_T = \mathbb{E}\left[\sum_{t=1}^{T} \mu_{a^*}(\mathbf{X}_t) - \mu_{a_t}(\mathbf{X}_t)\right]$$

In practice, solving a CMAB problem involves designing an algorithm that selects actions based on both observed context and past outcomes. Popular approaches include contextual extensions of ϵ-greedy (e.g., Epoch-Greedy [39]), UCB (e.g., LinUCB [40], KernelUCB [41]), and TS (e.g., Contextual TS [42]).

LinUCB, summarized in Algorithm 1, is an online algorithm for linear contextual bandit problems that extends the UCB algorithm to incorporate contextual information in action selection. In particular, LinUCB assumes a disjoint linear model[3] of expected rewards[4]:

$$\mu_a(\mathbf{x}_{t,a}) = \mathbb{E}[R_t \mid \mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^\top \boldsymbol{\theta}_a^*,$$

[3]This model is called *disjoint* because the parameters are not shared among arms; each action a has its own parameter vector $\boldsymbol{\theta}_a^*$.

[4]LinUCB builds from contextual linear bandits, which assume the existence of an unknown parameter vector $\boldsymbol{\theta}^* \in \mathbb{R}^d$ and a known feature map $\psi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^d$, such that the reward can be modeled as $R_t = \langle \psi(\mathbf{X}_t, A_t), \boldsymbol{\theta}^* \rangle + \eta_t$, where η_t is a zero-mean sub-Gaussian noise [29].

where $\mathbf{x}_{t,a} \in \mathbb{R}^d$ is a context-dependent feature vector for arm a at time t and $\boldsymbol{\theta}_a^* \in \mathbb{R}^d$ is the unknown arm-specific parameter vector, satisfying $\|\boldsymbol{\theta}_a^*\|_2 \leq 1$ [43]. Thus, the T-trial *regret* can be defined as:

$$\mathrm{Reg}_T = \sum_{t=1}^{T} \mathbf{x}_{t,a_t^*}^\top \boldsymbol{\theta}_a^* - \sum_{t=1}^{T} \mathbf{x}_{t,a_t}^\top \boldsymbol{\theta}_a^*,$$

where $a_t^* = \arg\max_{a \in \mathcal{A}} x_{t,a}^\top \boldsymbol{\theta}_a^*$ is the optimal action and a_t is the action chosen by the agent [40]. For each arm a, LinUCB maintains an estimate $\hat{\theta}_a$ of the true parameter $\boldsymbol{\theta}_a^*$ using ridge regression:

$$\hat{\theta}_a = \mathbf{A}_a^{-1}\mathbf{b}_a,$$

where $\mathbf{A}_a = \mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d$, with $\mathbf{D}_a \in \mathbb{R}^{t \times d}$ being the design matrix whose i^{th} row represents the feature vector of the arm pulled at time i and $\mathbf{I}_d$ being the identity matrix, and $\mathbf{b}_a = \mathbf{D}_a^\top \mathbf{c}_a$, with $\mathbf{c}_a \in \mathbb{R}^t$ being the response vector whose i^{th} row stores the reward from the arm pulled at time i [29, 44]. At each round t, LinUCB selects action a_t as:

$$a_t = \arg\max_{a \in \mathcal{A}} p_{t,a}$$

where $p_{t,a}$ is the *payoff*, defined as

$$p_{t,a} = \hat{\theta}_a^\top \mathbf{x}_{t,a} + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$$

Here, $\alpha > 0$ is a hyperparameter that controls the exploration-exploitation trade-off. After receiving the reward r_t for action a_t, the agent updates:

$$\begin{aligned}\mathbf{A}_{a_t} &\leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top\\ \mathbf{b}_{a_t} &\leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{t,a_t}.\end{aligned}$$

1.9 Algorithms

RL has led to the development of multiple algorithms to solve complex decision-making problems, including Q-learning [45] ⟨value-based, model-free⟩, Deep Q-Networks (DQNs) [46] ⟨value-based, model-free⟩, Proximal Policy Optimization (PPO) [47] ⟨policy-based, model-free⟩, Asynchronous Advantage Actor Critic (A3C) [48] ⟨actor-critic, model-free⟩, and AlphaZero [49] ⟨actor-critic, model-based⟩, among others.

1.9.1 Q-Learning

Q-Learning is a value-based, model-free RL algorithm that approximates the optimal action-value function q_* using the *Q-function* $Q(s, a)$. Under the conditions of sufficient exploration and appropriate decay of the learning rate, Q-Learning converges to q_* regardless of the policy being followed. [50] Q-Learning is an off-policy *Temporal Difference* (TD) *control* learner[5], meaning that the agent can learn the optimal policy independently of the agent's current actions. It is also a tabular method, meaning it maintains a table of Q-values for each state-action pair and updates these values iteratively. In particular, Q-Learning employs an ϵ-greedy policy to balance exploration and exploitation, and the Q-function is updated iteratively at each time step t using the following update rule based on the *Bellman equation** [28]:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right], \tag{2.2}$$

where $\alpha \in [0, 1)$ is the *learning rate*, which controls the magnitude of each update, and $R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)$ is the so-called *temporal difference error* (TD error, δ_t) that represents the difference between the *Q-target*, $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$, and the current Q-value estimation, $Q(S_t, A_t)$. [53]

1.9.2 Deep Q-Network

Deep Q-Network (DQN) (a.k.a. *Deep Q-Learning*) is a value-based, model-free RL algorithm that extends traditional Q-Learning by integrating deep neural networks to approximate the Q-values. This enables the algorithm to estimate Q-values for high-dimensional state spaces, which are otherwise intractable with traditional tabular methods, addressing the *curse of dimensionality* [28].

Hence, instead of maintaining a Q-table, DQN uses a deep neural network Q_θ with parameters θ (referred to as the *Q-network*) to estimate the optimal Q-values for each state-action pair. The Q-network takes a state s as input and outputs a vector of Q-values, one for each possible action a. The Q-network is updated using *Stochastic Gradient Descent* (SGD) to minimize the TD error. This is achieved by minimizing the loss, defined as the *Mean Squared Error* (MSE) between the Q-target and the current Q-value estimation (see Eq. 2.2). This process ensures that the network parameters converge towards the optimal Q-values. Notably, to balance exploration and exploitation, the agent selects actions according to an ϵ-greedy policy.

[5]TD learning methods [51] update estimates based on the difference (error) between temporally successive predictions. [52]

Since consecutive experiences are highly correlated and SGD requires data samples to be i.i.d. for stable learning, DQN introduces the *experience replay buffer*. The agent stores experiences (S_t, A_t, R_t, S_{t+1}) in a finite buffer $\mathcal{D}$, and when the buffer reaches its capacity, the oldest experiences are replaced by new ones. During training, the agent randomly samples mini-batches from $\mathcal{D}$, rather than using the most recent experience[6], breaking the temporal correlation between experiences. [55]

Moreover, to ensure training stability, DQN introduces a second neural network $\hat{Q}_{\theta^-}$ with parameters θ^- (referred to as the *target network*). The target network has the same architecture as the Q-network, but its weights are updated less frequently by periodically copying the weights from the Q-network. This periodic update helps ensure that the target network provides a stable Q-target for the TD error, mitigating instability.

Thus, instead of using the current Q-network Q_θ to both estimate the Q-value and compute the Q-target, the Q-target is computed using the target network $\hat{Q}_{\theta^-}$:

$$y_i = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(S_{t+1}, a'; \theta^-_{i-1})$$

This ensures that the learning updates remain stable and converge efficiently. Consequently, the loss function used to update the Q-network at step i is [54]:

$$\mathcal{L}_i = \mathbb{E}_{S_t, A_t, R_{t+1}, S_{t+1} \sim \mathcal{D}} \left[\left(y_i - Q(S_t, A_t; \theta_i) \right)^2 \right]$$

2 Multi-Agent Reinforcement Learning

2.1 Introduction

A Multi-Agent System (MAS) consists of multiple autonomous agents that interact within a shared environment to achieve individual or collective goals [56]. In contrast to Single-Agent Reinforcement Learning (SARL), where a single agent learns an (implicit or explicit) optimal *policy* to maximize its return, in Multi-Agent Reinforcement Learning (MARL) multiple agents operate in a shared environment and either cooperate to maximize a common *return* or compete to maximize individual returns [17]. Thus, MARL algorithms learn optimal policies for multiple agents in a MAS[7]. [2]

[6]Using only the most recent experience to update the Q-network leads to standard Q-Learning. [54]

[7]This MARL definition contrasts with learning a policy for a single agent that operates in a MAS in which there is no control over other agents. [2]

MARL introduces additional complexities, primarily stemming from each agent's actions affecting also the outcomes of other agents, creating a dynamic, non-stationary environment from each agent's perspective. Nevertheless, MARL can decompose large, complex decision-making problems into smaller, more manageable sub-problems [2], enhancing scalability and efficiency as each agent's actions space is comparatively smaller than in a SARL. Additionally, MARL can benefit from *coordination* and *experience sharing* [57].

2.2 Definition of the problem

A *fully observable* MARL problem is formally modeled as a Stochastic Game [58] (a.k.a. *Markov Game*). A Stochastic Game is defined by the tuple $\langle N, \mathcal{S}, \{\mathcal{A}^i\}^{i\in N}, \mathcal{P}, \{\mathcal{R}^i\}^{i\in N}\rangle$ [2,59,60], where:

N is the *agent space*, the set of n agents.

$\mathcal{S}$ is the *state space*, the set of all possible states.

$\mathcal{A}^i$ is the *action space* for agent i, the set of actions available to agent i.

$\mathcal{P}$ is the *state transition matrix*, defining the probability of transitioning from state s to state s' given a *joint action* $a = (a^1, \ldots, a^n)$.

$\mathcal{R}^i$ is the *reward function* for agent i, defining the expected reward for agent i after taking action a^i in state s while the other agents take actions a^j.

A *partially observable* MARL problem is formally modeled as a Partially Observable Stochastic Game (POSG). A POSG is defined by the tuple $\langle N, \mathcal{S}, \{\mathcal{A}^i\}^{i\in N}, \{\mathcal{O}^i\}^{i\in N}, \mathcal{P}, \{\mathcal{R}^i\}^{i\in N}, \{\mathcal{Z}^i\}^{i\in N}\rangle$, i.e., the same elements of a Stochastic Game and an *observation space* $\mathcal{O}^i$ and an *observation function* $\mathcal{Z}^i$ for each agent i.

2.3 System-Environment interaction

The MARL system-environment interaction loop, illustrated in Fig. 2.2, generalizes the SARL interaction loop (Fig. 2.1) to multiple agents. At each time step t, each agent $i \in N$ perceives an *individual observation* $O^i_{t+1} \in \mathcal{O}^i$ of the environment's state S^e_t and selects an *individual action* $A^i_t \in \mathcal{A}^i$ according to its *policy* π^i. These actions together form a *joint action* $A_t = \{A^1_t, \ldots, A^n_t\} \in \mathcal{A}$, where $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$ represents the *joint action space*. The

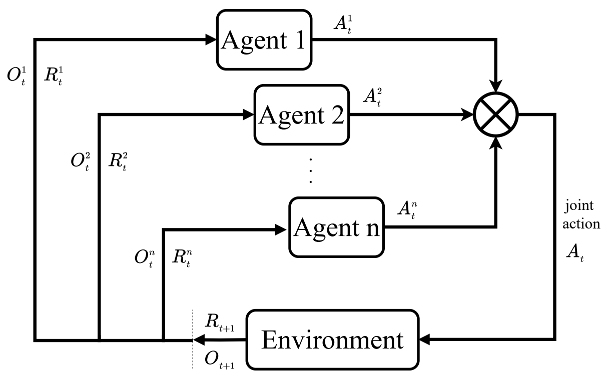


Figure 2.2: The MARL system-environment interaction loop. *Source: Self-elaborated, adapted from [2].*

environment, upon receiving the joint action, transitions to a *new state* S^e_{t+1} and provides feedback to each agent i in the form of a *new individual observation* O^i_{t+1} and an *individual scalar reward*[8] R^i_{t+1}. [2] This cycle continues until a terminal condition is satisfied or indefinitely in ongoing tasks.

In Cooperative MARL (Coo-MARL), agents may also communicate through a *communication network* [60], exchanging information to improve coordination. Nevertheless, in most MARL systems, each agent perceives only the effects of its actions and does not have access to the rewards of other agents. The goal of each agent i is to maximize its expected (discounted) return over time, a property known as *rationality.* [60] The agent achieves this by determining its optimal policy π^i_*, which is the *best response* to the *joint policy* π^{-i} of the other agents[9]. In equilibrium, the agents' strategies form a *Nash equilibrium*, where no agent can improve its outcome by unilaterally changing its policy. Consequently, the value function v^i of each agent depends not only on its own policy but also on the policies of the other agents, represented as $v_{\pi^i,\pi^{-i}}$. [61]

2.4 Categorization

MARL problems can be categorized along several dimensions, including the operation mode, agent homogeneity, the nature of agent objectives, and the learning paradigm. [2]
Based on the operation mode, MARL can be classified as *self-play* or *mixed-play* [2]:

- *Self-play* includes *algorithm* and *policy self-play. Algorithm self-play* involves all agents

[8]The collection of all individual rewards at time step t is referred to as the *joint reward*, $R_t = \{R^1_t, ..., R^n_t\}$.
[9]$-i$ denotes all agents except agent i, and thus π^{-i} represents the collection of policies of all agents other than agent i

using the same learning algorithm, converging to the same type of equilibrium solution. *Policy self-play* involves training an agent's policy directly against itself. An extension of self-play is *population-based training*, in which policies are trained against a distribution of other policies, including past versions of themselves.

- *Mixed-play* involves agents using different learning algorithms.

Another important aspect is that agents can be *heterogeneous* or *homogeneous* [2, 62]:

- *Homogeneous agents* have the same observation and action spaces. In such cases, *parameter sharing* [63] can be leveraged to enhance sample efficiency. If the optimal joint policy consists of identical individual policies, the agents are considered *strongly homogeneous.*

- *Heterogeneous agents* have distinct observation and action spaces.

MARL scenarios can also be categorized based on the nature of the agents' goals (and the associated reward structure) as *(fully) cooperative*, *(fully) competitive*, or *mixed cooperative-competitive* [2, 17, 61, 62, 64, 65]:

In *Cooperative MARL* agents share a common goal and collaborate to either maximize a 'single global reward'[10] or a combination of individual rewards[11] reflecting the agents' contributions. Successful cooperation often requires coordination and consensus [67].

In *Competitive MARL* agents are self-interested in maximizing their own reward, often at the expense of others. This is commonly modeled as a *zero-sum Markov Game* [68] ($\sum_{i=1}^{N} R_t^i = 0$), where one agent's gain is another's loss. In such games, agents often use *minimax strategies* [68], optimizing the worst-case adversarial scenario.

In *Mixed cooperative-competitive MARL*, modeled as a *general-sum game*, agents have both shared and individual goals, aligning in some aspects but differing in others.

Notably, multi-agent problems can be reduced to single-agent problems using *central learning*, or *independent learning* [2, 61]:

- *Central learning* treats the multi-agent system as a single entity and applies SARL on the joint action space, learning a single centralized policy that selects actions for all agents. Thus, agents are typically modeled as *joint-action learners* [69], where the optimal policy is derived by considering the joint actions of all agents in the system.

[10]In the case of a single global reward, the agents' reward functions are identical, meaning all agents receive the same reward at each timestep: $R_t = R_t^1 = R_t^2 = ... = R_t^n$

[11]In the case of individual rewards, the agents' reward functions may differ but still align to a common goal. A common approach models the global reward as the sum of individual rewards: $R_t = \sum_{i=1}^{n} R_t^i$ [66]

- *Independent learning* considers each agent as a separate, independent learner, treating other agents as part of the environment. Thus, agents individually use SARL to learn their policies without explicitly accounting for the presence of others. For instance, each agent may use Q-Learning, DQN, or PPO independently. [70, 71] This approach often implicitly assumes *self-play*. Notably, Multi-Agent Multi-Armed Bandit (MA-MAB) (and its contextual counterpart) is a stateless, independent learning-based MARL problem.

Building upon these learning approaches, MARL systems can be categorized based on the learning paradigm as *Centralized Training and Execution* (CTE), *Decentralized Training and Execution* (DTE), or *Centralized Training and Decentralized Execution* (CTDE) [2]:

- In *Centralized Training and Execution* (CTE) agents share centralized information during both the training and execution, enhancing coordination and mitigate non-stationarity. This approach assumes a centralized information-sharing mechanism, such as a central agent that collects all observations and dictates actions. A *single joint policy*[12] π is learned for all agents, allowing the use of SARL training methods [61]. To reduce complexity, the joint value function is often factored into individual agent-specific value functions, an approach known as *value function decomposition* [65]. Notably, a challenge in CTE is the so-called *lazy agents* phenomenon, where one agent may learn a good policy, while another agent has less incentive to learn, as its actions may hinder the first agent's progress [61].

- In *Decentralized Training and Execution* (DTE) agents operate independently without sharing information nor communicating, ensuring fully *independent learning* as each agents relies solely on its local information to learn and act. Thus, an n-agent MARL problem is decomposed into n decentralized SARL problems, where each agent i holds its own individual policy π^i [61, 72].

- In *Centralized Training, Decentralized Execution* (CTDE) agents share centralized information during training but operate independently during execution, learning policies based only on local observations. In this hybrid approach, each agent i holds an individual policy π^i [61], benefiting from centralized coordination during learning while maintaining flexibility and autonomy during execution.

2.5 Cooperative Multi-Agent Reinforcement Learning challenges

Cooperative MARL (Coo-MARL) inherits several challenges from SARL, such as the *exploration-exploitation dilemma* (see Section 1.6), while introducing additional complexities due to the presence of multiple learning agents, including:

[12]The *joint policy* π denotes the collection of all agents individual policies π^i: $\pi = \{\pi^1, \pi^2, ..., \pi^n\}$

- *Non-stationarity* [2,17,61]: since agents learn concurrently, the environment appears non-stationary from each agent's perspective and the *Markov property* no longer holds. This may lead to the *moving target problem*, where agents struggle to adapt to the changing policies of other agents, often leading to unstable or cyclic learning behaviors. *Autocurricula* [73] may arise in such scenarios, where agents' evolving strategies continuously affect their own and others' learning.
  Non-stationarity is particularly pronounced in independent learning settings [17, 71]. In contrast, centralized training paradigms, such as CTDE, effectively handle non-stationarity during the learning phase by leveraging access to joint observations and actions [74]. A naive approach to handle non-stationarity is to simply *ignore* it—assuming a stationary environment—as done in algorithms such as Q-Learning and UCB. [20] Another strategy involves using *joint action learners*, which learn policies over the entire joint action space; however, this becomes computationally infeasible as the number of agents increases. [17] To explicitly handle non-stationarity, several more sophisticated techniques have been proposed, including *forgetting-based strategies* (e.g., using sliding-window updates or exponential discounting) to enable faster adaptation to environment changes, *opponent modeling* [20] to explicitly infer other agents' behaviors and respond accordingly, and *adaptive experience replay mechanisms* [74] to stabilize learning.

- *Coordination and communication* [60,61]: achieving effective cooperation requires agents to align their actions to maximize the collective return, despite each agents basing their decisions on only local information. Mis-coordination can lead to inefficiencies, such as *action shadowing*, where exploratory actions disrupts the other agents' learning. Communication mechanisms—such as differentiable, neural-based approaches like CommNet [75]—can facilitate coordination, but also introduce additional complexities.

- *Credit-Assignment Problem* (CAP) [2, 56, 61]: determining which agent's actions contributed to the collective return is challenging, especially when agents do not have access to the *joint action space*. Without proper credit assignment, learning becomes inefficient and agents may converge to suboptimal policies.
  A closely related problem is the *Temporal Credit Assignment Problem* (T-CAP) [76], which concerns identifying which past actions are responsible for delayed rewards. Several approaches have been proposed to address the T-CAP, including TD learning and eligibility traces [77]. Another promising direction involves decomposing sparse environment rewards into temporally localized contributions for each agent, enabling more precise assignment of delayed outcomes [78].
  Solutions to the CAP can be categorized as *explicit* and *implicit* credit assignment meth-

ods. Explicit methods attempt to directly estimate each agent's contribution to the global reward. A classic approach involves using *difference rewards*[13] to isolate individual contributions; however, this can be computationally inefficient due to the need to simulate counterfactual scenarios. [80] A notable alternative is the COMA algorithm [81], an actor-critic method that employs a centralized critic to compute a *counterfactual baseline*, marginalizing a single agent's action while keeping the other agents' actions fixed. [17, 71, 80] On the other hand, implicit credit assignment methods avoid direct estimation of agent-specific contributions. A popular class of such methods are value decomposition approaches, which factorize the joint action-value function into individual agent value functions (e.g., VDN [82] and QMIX [83]) [71, 74, 80].

- *Scalability* [2, 17, 61, 84]: the *joint action space* grows exponentially with the number of agents, significantly increasing computational complexity (commonly referred to as the *curse of dimensionality*).
  Independent learning offers scalability but can suffer from instability and non-stationarity [17, 85]. On the other hand, CTDE-based approaches ensure scalability during execution, wile effectively handling non-stationarity during training [74]. Additionally, (selective) *parameter sharing* can further enhance scalability, as it improves sample efficiency and alleviates the problem of parameter growth as the number of agents increases [86].

3 Reinforcement Learning in wireless networks

RL has emerged as a promising approach for optimizing Wireless Networks (WNs) [10–12], as it enables adaptive decision-making based on real-time feedback from the environment. Recent research has increasingly explored RL-based solutions for WN optimization, particularly focusing on spectrum allocation and parameter tuning in the Medium Access Control (MAC) protocol. Indeed MAC protocol optimization has gained significant attention as it is responsible for managing medium contention and ensuring reliable and efficient data transmission.

For instance, research has investigated RL-based approaches to handle channel collisions by adjusting the Contention Window (CW) in Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)[14]. Chen et al. [87] applied Q-learning to periodically optimize the CW size in

[13]In difference rewards, the marginal contribution of agent i is formally computed as: $D_i = G(z) - G(z_{-i})$ where $G(z)$ is the team reward under the joint action z and $G(z_{-i})$ is the reward without agent i's contribution [79].

[14]IEEE 802.11 Distributed Coordination Function (DCF) uses CSMA/CA as its primary channel access mechanism. CSMA/CA uses an exponential backoff scheme to manage access, where the CW determines the waiting time before transmission attempts. See Section 3 for further details.

order to maximize throughput, demonstrating that their approach significantly reduces collision rates and increases system throughput compared to traditional rule-based methods. Similarly, Kim et al. [88] used Q-learning to select the optimal backoff value within the CW, aiming to maximize the transmission success rate. Notably, the authors achieved higher throughput than standard CSMA/CA. Wydmanski and Szott [89] explored Deep Q-Networks (DQNs) and Deep Deterministic Policy Gradient (DDPG) [90] for centralized CW control at the AP, aiming to maximize throughput. Their results demonstrated that both RL-based approaches achieve near-optimal efficiency with low computational cost. Kumar et al. [91] employed a Rainbow DQN [92], which integrates multiple RL enhancements to base DQN (e.g., double Q-learning, prioritized replay, multi-step learning), to optimize minimum CW selection, aiming to maximize network utility. Their approach demonstrated close-to-optimal behavior, outperforming conventional rule-based methods. Beyond CSMA/CA, Ali et al. [93] leveraged Q-learning to predict future channel states (idle or busy) and minimize collision probability. Their approach is based on the Channel Observation-based Scaled Backoff (COSB) protocol [94], which dynamically adjusts CW sizes considering channel congestion instead of resetting CW size as in DCF.

A particularly promising approach for MAC protocol optimization in WNs is MARL. Several studies have explored non-cooperative MARL, where agents independently optimize MAC parameters without explicit collaboration. For instance, Keshtiarast et al. [16] considered each agent as an individual, decentralized learner using the PPO algorithm to adjust multiple MAC parameters, including maximum channel occupancy time, transmission power, Modulation and Coding Scheme (MCS), backoff parameters (e.g., sensing slot duration, minimum CW, energy detection threshold), and deferral period. Their approach aimed to maximize throughput while minimizing airtime utilization in cellular networks, considering a Listen Before Talk (LBT)[15] protocol. Their DTE-based approach converged slightly faster in comparison against a CTE paradigm due to the latter's larger action space, and their method outperformed standard cellular MAC protocols. Xiao et al. [12] used a CTDE-based Multi-Agent Proximal Policy Optimization (MAPPO) [95] to maximize the minimum throughput among stations, using a simple two-action space (transmitting or not transmitting). Their approach outperformed standard CSMA/CA and independent PPO-based approaches in terms of fairness and efficiency. Guo et al. [96] enhanced the QMIX [83] MARL algorithm by incorporating individual Q-values for fairness and stability. Their proposed distributed channel access strategy, based on CTDE, aimed to maximize throughput while ensuring fairness, prioritizing stations with longer transmission

[15]LBT is a channel access mechanism that requires users to sense the medium before initiating transmission. DCF is based on the LBT mechanism.

delays. Their approach demonstrated superior throughput, delay, and jitter performance compared to baseline CSMA/CA. Notably, the authors considered a similar action space to [12]. Similarly, Sohaib et al. [97] developed a distributed multi-channel access protocol using a dueling Q-network-based MARL approach, allowing nodes to deterministically schedule transmissions across time slots. Their method improved both throughput and fairness compared to conventional RL approaches and centralized scheduling policies. On the other hand, Qi et al. [98] used a CTDE-based Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [99] approach to dynamically allocate channels in a dense IEEE 802.11 DCF-based scenario, demonstrating reduced latency compared to state-of-the-art methods.

Furthermore, Coo-MARL approaches have been investigated to enhance collaboration among wireless nodes, particularly in cellular networks. Miuccio et al. [100, 101] designed a CTDE-based Coo-MARL framework where homogeneous agents optimized joint policies using MAPPO. Their framework aimed to maximize successful packet transmission while minimizing transmission time and collisions, using a simple three-action space (transmitting or deleting the oldest data unit in the buffer or doing nothing). Similarly, Valcarce and Hoydis [18] demonstrated that tabular RL agents could jointly learn MAC signaling and channel access policies, achieving efficient channel utilization through self-play. Their CTDE approach, using Independent Q-Learning (IQL), aimed to maximize channel access efficiency while adhering to predefined signaling constraints. Notably, the authors considered a similar action space to [100, 101]. Naeem et al. [102] introduced a Coordinated Multi-Agent Exploration (CMAE) framework, where agents progressively learn policies from low- to high-dimensional state spaces, leading to improved goodput[16] and reduced collision rates compared to traditional MARL approaches as MADDPG and Q-Learning. Notably, the authors considered a similar action space to [18, 100, 101], but rewarded agents positively if data units were delivered and negatively otherwise. Dutta and Biswas [103] proposed an online learning MAC framework using Hysteretic Q-Learning (HQL) [104], enabling network nodes to independently learn transmission probabilities while mitigating self-collisions, without being aware of other nodes' actions. Their CTDE approach achieved near-optimal throughput even under high network loads. On the other hand, Mota et al. [105] explored a CTDE-based Coo-MARL approach with explicit inter-agent communication, where base stations and user equipment cooperatively learned channel access policies and signaling policies using MADDPG. Their approach highlighted the importance of signaling policy learning, significantly improving network performance in terms of goodput compared to contention-free baselines. Notably, the authors consider a similar action space

[16]Goodput is the application-level throughput, i.e., the number of successfully delivered payload bits per unit time.

to [18, 100–102], with a positive or negative reward depending on the reception of data units, similar to [102]. Beyond cellular networks, Moon et al. [106] adopted a CTDE Coo-MARL framework, using MAPPO to dynamically adjust CW sizes and control queue rates. Their approach improved network utility and reduced queuing delay compared to IEEE 802.11 DCF.

Beyond full RL-based models, *online learning* through MABs has also been extensively explored for WN optimization, especially due to their fast convergence and low computational complexity. For instance, Szczech et al. [107] proposed a lightweight TS-based MAB agent that dynamically adjusts CW size, enables/disables frame aggregation, and toggles RTS/CTS at the AP to optimize a multi-objective reward function balancing throughput, latency, and fairness. The authors considered a single-Basic Service Set (BSS)[17] and demonstrated significant improvements in both latency and throughput over standard IEEE 802.11. Carrascosa et al. [108, 109] introduced a decentralized MAB-based approach for dynamic AP-STA association, using an Opportunistic ϵ-greedy with Stickiness strategy, where STAs independently selected their most suitable AP. Similarly, Lopez et al. [110] adopted decentralized MAB agents using TS to optimize channel selection and AP association. Their approach, based on effective channel load and airtime, significantly outperformed static configurations. On the other hand, Wojnar et al. [111] developed a hierarchical two-level MAB framework for spatial reuse that efficiently selects AP-STA pairs for concurrent transmissions. The authors evaluated multiple MAB algorithms (ϵ-greedy, UCB, TS, Softmax [112]), finding UCB particularly effective, offering fast convergence, adaptability, and sustained performance under dynamic conditions. Martínez et al. [113] proposed a CMAB-based approach for WN selection, using the traffic type as contextual information. Maghsudi et al. [114] proposed a decentralized, adversarial Multi-Player Multi-Armed Bandit (MP-MAB) framework that jointly optimizes channel selection and transmission power. Transmitters acted as self-interested agents aiming to maximize channel-quality-based rewards and achieved a stable equilibrium in non-coordinated environments. Similarly, Wilhelmi et al. [115] studied MP-MAB algorithms (e.g., ϵ-greedy, UCB, TS) for spatial reuse through dynamic channel and transmission power selection in decentralized WNs. The authors considered a throughput-based reward and their results highlighted that sequential learning can achieve fairness while reducing temporal throughput variability compared to simultaneous decision-making.
Regarding Multi-Agent Multi-Armed Bandits (MA-MABs), Barrachina-Muñoz et al. [116] explored MA-MAB algorithms (ϵ-greedy, UCB, TS) for primary channel selection and channel width adaptation in dense IEEE 802.11 DCF-based scenarios, aiming to maximize throughput. The authors evaluated their approach in multi-agent deployments, where APs acted as inde-

[17]A BSS comprises a single Access Point (AP) and one or more associated Stations (STAs).

pendent, non-cooperative learners. Their mechanism outperformed static configurations but the authors highlighted the need for collaboration (and inter-agent communication) to achieve fairness. Wilhelmi et al. [117] proposed a coordinated MA-MAB framework for spatial reuse, where APs share feedback via Multi-AP Coordination (MAPC) (a coordination feature introduced in recent IEEE 802.11 amendments) and apply ϵ-greedy or TS with joint rewards (e.g., average, min-max) to optimize transmission power, demonstrating enhanced network performance over current IEEE 802.11 operation.
Other researchers have investigated Multi-Agent Contextual Multi-Armed Bandit (MA-CMAB) solutions. Iturria et al. [118] evaluated cooperative and non-cooperative MA-CMAB strategies for spatial reuse, using distributed agents at APs that optimize transmission power and Clear Channel Assessment (CCA) thresholds to maximize fairness and minimize starvation. Notably, cooperative algorithms (SAU-Sampling [119] and Reward-Cooperative ϵ-greedy MA-MAB) outperformed non-cooperative ones (ϵ-greedy, UCB, and TS), highlighting the suitability of joint rewards.

Thus, the literature demonstrates extensive research on RL and MARL for WN MAC optimization, particularly on channel collision handling and channel allocation. Nevertheless, the application of Coo-MARL (and its stateless form, cooperative MA-(C)MAB) in non-cellular WNs remains scarce. In this thesis, we leverage cooperative MA-CMABs to optimize MAC-layer decisions in WNs, such as the selection of the channel group—addressing the *channel allocation problem* [10], which involves determining the appropriate primary channel and channel width—as well as the configuration of CSMA/CA parameters, specifically the CW size. We compare our multi-agent approach against a Single-Agent Contextual Multi-Armed Bandit (SA-CMAB) baseline in terms of performance and power consumption. Hence, similar to most prior RL/MARL works on WN optimization, which are simulation-based, we first develop an open-source, event-driven WN simulator in Python, based on the SimPy library and implementing IEEE 802.11 DCF as the fundamental MAC mechanism. Notably, several IEEE 802.11 DCF open-source, SimPy-based WN simulators can be found in the literature [120–122].

For more detailed surveys on the application of ML in WN optimization, please refer to [10,15].

Chapter 3

Implementation

1 IEEE 802.11 DCF-based simulator

The event-driven IEEE 802.11 Distributed Coordination Function (DCF) simulator, developed for this thesis using Python, is open access and available on GitHub[1]. It adopts a modular, event-driven architecture to model IEEE 802.11-based wireless networks and leverages SimPy, a discrete-event simulation framework in Python. Simulation time progresses in microsecond-level resolution, ensuring fine-grained modeling of Medium Access Control (MAC) and Physical (PHY)-layer interactions.

This section provides a high-level overview of the simulator, describing its inputs and outputs, as well as its default operation—independent of any Reinforcement Learning (RL) components. Additional implementation details—including an overview of IEEE 802.11 fundamentals, the simulator's features and abstractions, SimPy, and the validation methodology—are provided in Appendix A.

Inputs and outputs

The simulator is highly configurable through the `user_config.py` and `sim_params.py` files. The `user_config.py` file enables high-level customization of the simulation environment, including parameters such as simulation duration, random seed for reproducibility, network topology (e.g., spatial positions of nodes and characteristics of their traffic sources) and optional features such as displaying events in the console during runtime, recording events, saving traffic traces, logging metrics using Weights & Biases (W&B)[14], or collecting statistics. Complementarily, the `sim_params.py` file defines lower-level technical parameters such as transmission queue sizes, slot time duration, number of Spatial Streams (SS), Packet Error Rate (PER), among others.

[1] `https://github.com/miguelcUPF/MARL-WiPySim`

Default simulation parameters are summarized in Tbl. 3.1. All user-defined configuration are validated at the beginning of a simulation run, preventing invalid setups.

The simulator may output event traces in the console to monitor runtime behavior. It may also record events and traffic traces, generate illustrative figures, and record metrics from executions in Weights & Biases (W&B), depending on the configuration settings. Upon completion of each simulation run, statistics may be stored in JavaScript Object Notation (JSON) format. The collected statistics include: (*a*) global-level metrics such as the total number of successful and failed transmissions, aggregate throughput, and overall packet loss ratio; (*b*) per-node metrics, including transmission statistics (e.g., number of successful and failed transmissions, total packets transmitted, and average queue lengths), reception statistics (e.g., number of received packets and reception rate), and general node information (e.g., Basic Service Set (BSS) identifier, associated Access Point (AP) or Stations (STAs), and time spent in each state—idle, contending, transmitting, or waiting for reception); (*c*) per-channel metrics, including airtime and channel utilization.

Standard operation

The simulator adopts a transmitter-receiver abstraction: a source generates traffic at the transmitter, which passes from the transmitter's MAC layer to its PHY layer, then propagates over the wireless medium to the receiver's PHY layer, and finally handed to the receiver's MAC layer. Once successfully received, the data is delivered to an abstract traffic sink, representing the application endpoint. Thus, each node in the simulation includes both MAC and PHY layer abstractions. The MAC layer manages packet queuing, channel access, backoff, (re)transmission, frame aggregation, and acknowledgment procedures, serving as the interface between traffic sources and the underlying PHY layer (see Fig. C.7 for a flowchart of the simulator's transmitter-side MAC layer operation). The PHY layer determines the Modulation and Coding Scheme (MCS) and is responsible for frame transmission and reception, acting as the

[1] Jain's fairness index, $\mathcal{J} \in [0,1]$, is defined as $\mathcal{J} = \left(\sum_i x_i\right)^2 / \left(n \sum_i x_i^2\right)$, where n is the number of coexisting BSSs and x_i denotes the resource allocation (here, average goodput) of the i^{th} BSS.

[2] It is assumed that all nodes transmit with the same power level.

[3] Considering Non-Line-of-Sight (NLOS) indoor environments [123].

[4] $T_{\text{DIFS}} = T_{\text{SIFS}} + 2T_{\text{e}}$

[5] $T_{\text{CTS,to}} = T_{\text{SIFS}} + T_{\text{CTS}} + T_{\text{e}}$, where T_{CTS} is the transmission time of a CTS frame using a basic 20 MHz channel, a single SS, MCS 0, and 0.8 μs GI

[6] $T_{\text{PIFS}} = T_{\text{SIFS}} + T_{\text{e}}$

[7] According to the 802.11 standard, $T_{\text{EIFS}} = T_{\text{DIFS}} + T_{\text{SIFS}} + T_{\text{ACK}}$, where T_{ACK} denotes the transmission time of an ACK frame at the lowest PHY mandatory rate. Nevertheless, to preserve temporal alignment and synchronized backoff counters among contending nodes, in our implementation, EIFS is abstracted as equal to DIFS plus the CTS or BACK timeout (depending on the collision type).

Table 3.1: Simulation parameters default values.

	Description	**Value**		**Description**	**Value**
f_c	Carrier frequency	5 GHz	N_{SD} *	Number of subcarriers	20 MHz: 234
MCS *	MCS index	0 - 11			40 MHz: 468
N_{SS}	Spatial Streams	2			80 MHz: 980
P_{tx}	Transmit power[2]	20 dBm	G_{tx}	Transmit gain	0 dB
γ	Path loss exponent[3]	4	G_{rx}	Reception gain	0 dB
p_{err}	Packet Error Rate	0.1			
N_{Q}	MAC TX queue size	100 pkts	CW_{min}	Minimum CW	16
CRC	Common Retry Count	7	CW_{max}	Maximum CW	1024
RTS/CTS	RTS/CTS enabled?	Yes	L_{RTS}	RTS frame size	20 B
RTS_θ	RTS threshold	2346 B	L_{CTS}	CTS frame size	14 B
L_{MAC}	MAC header size	32 B	L_{FCS}	FCS size	4 B
L_{PHY}	PHY header size	24 B	L_{DELIM}	MPDU delimiter size	4 B
$L_{\text{A-MPDU}}^{\text{max}}$	Max. A-MPDU size	65535 B	L_{PAD}	MPDU padding size	3 B
$L_{\text{DATA}}^{\text{max}}$	Max. data packet size	1280 B	$L_{\text{BACK}}^{\text{mpdu}}$	BACK size per MPDU	2 B
T_{e}	Empty slot duration	9 μs	T_{SIFS}	SIFS duration	16 μs
T_{GI}	GI duration	0.8 μs	T_{DIFS}	DIFS duration[4]	34 μs
$T_{\text{CTS,to}}$	CTS timeout[5]	60 μs	T_{PIFS}	PIFS duration[6]	25 μs
$T_{\text{BACK,to}}$	BACK timeout	281 μs	T_{EIFS} *	EIFS duration[7]	85 or 315 μs
N_{c}	Number of channels	4	CB	Channel Bonding	SCB

* hardcoded in the simulator.

intermediary between the underlying MAC layer and the wireless medium. The medium propagates frames between transmitter and receiver PHY layers, modeling effects such as collisions and frame errors.

Network topology

The network comprises one or more BSSs, each consisting of a single AP and one or more associated STAs, depending on the configuration. Each node (whether AP or STA) is assigned a unique identifier and three-dimensional spatial coordinates (x, y, z), either user-defined or randomly generated at runtime.

Traffic

Each node encompasses one or more configurable traffic sources. Each traffic source generates data traffic, targeting a destination. A traffic source can be instantiated as either a traffic generator or a traffic loader:

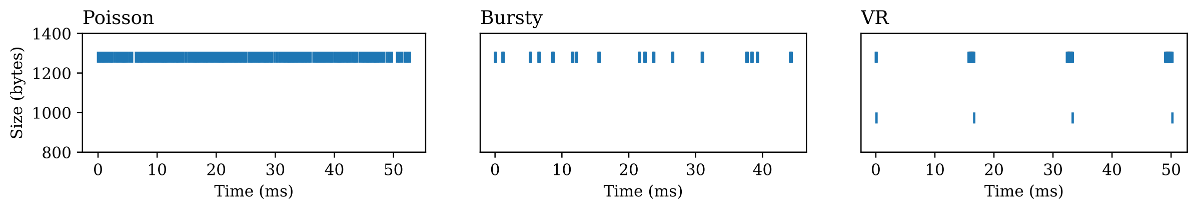


Figure 3.1: Traffic generation patterns for Poisson, Bursty, and VR traffic models, considering a 100 Mbps traffic load (and 60 fps for VR) and using the simulator's default parameter values.

- A *traffic generator* creates synthetic traffic flows using Poisson, Bursty, or Virtual Reality (VR) traffic models (see Fig. 3.1). Furthermore, it supports a full-buffer mode, where packets are continuously generated to ensure that the transmission queue always remains full.

- A *traffic loader* replays pre-recorded traffic traces from Comma-Separated Values (CSV) files, facilitating reproducibility across simulation runs. It supports loading traces from real-world Wireshark captures or previous simulations. Each trace must include packet arrival timestamps and packet sizes in bytes.

To enable reproducibility, the traffic generated at a node can be independently recorded based on its destination.

Channel assignment and MCS selection

Each AP (and associated STAs) is assigned a set of allocated channel(s) at the beginning of the simulation. If not specified, channels are randomly selected from the set of valid channels. Likewise, the primary channel is selected at random from the allocated set unless explicitly defined.

The MCS for each AP-STA pair is determined at the start of the simulation based on the RSSI, which depends on the path loss, and thus their Euclidean distance, as defined in Section A.2.

Enqueuing and backoff

At the transmitter, each created data packet is enqueued into the MAC layer's transmission queue as an MAC Protocol Data Unit (MPDU), considering the sizes of the MAC header and Frame Check Sequence (FCS), but without explicitly modeling them. If the queue overflows, incoming packets are dropped. MPDUs remain in the queue until successfully acknowledged or discarded due to exceeding the retry limit.

If the queue is non-empty, the MAC follows the channel access procedure described in Section A.1.1. In summary, if the primary channel remains continuously idle for a Distributed Inter-frame Space (DIFS) (or Extended Inter-Frame Space (EIFS)) duration, the backoff procedure begins. The backoff counter is randomly initialized based on the Contention Window (CW) size and is decremented every idle slot time. If the channel becomes busy during the backoff process, the countdown is paused and resumes only after another idle DIFS (or EIFS). Once the backoff counter reaches zero, the transmission proceeds, using either Static Channel Bonding (SCB) or Always-max (AM) Dynamic Channel Bonding (DCB)—depending on the configuration—as described in Section A.1.2.

Carrier sensing is modeled via SimPy events. If the channel is busy, the node waits for an 'idle' event to signal the end of the ongoing transmission or Network Allocation Vector (NAV) reservation. Once the channel is idle, a DIFS (or EIFS) 'timeout' event is scheduled. If a 'busy event' is triggered before the timeout completes, the sensing process restarts. Otherwise, if the timeout event succeeds, the node considers the channel available and initiates the backoff procedure.
On the other hand, the availability of secondary channels (i.e., whether they were idle during the Point Coordination Function (PCF) Inter-frame Space (PIFS) period preceding transmission) is determined based on their last idle and busy timestamps, so no explicit event triggers are required.

Frame aggregation

Upon completing the backoff, multiple MPDUs are aggregated into a single Aggregated Medium Access Control (MAC) Protocol Data Unit (A-MPDU). Only packets destined to the same receiver as the head-of-queue packet are aggregated, constrained by a maximum aggregate size. The size of each subframe in the aggregate accounts for the overhead introduced by the delimiter and padding.

RTS/CTS exchange

After aggregation, if enabled and if the A-MPDU size exceeds the RTS threshold, the transmitter initiates a Request To Send (RTS)/Clear to Send (CTS) exchange. An RTS frame is transmitted over the primary channel. The CTS timeout timer starts immediately after the RTS frame has been fully transmitted, that is, after the last bit of the RTS frame is pushed to the medium. If a CTS frame is not received within the timeout period, transmission is deferred and a new backoff stage begins after sensing the channel continuously idle during DIFS, doubling the CW size (up to $\mathrm{CW}_{\mathrm{max}}$). Otherwise, the A-MPDU transmission proceeds and the backoff stage is reset.

The receiver, upon successfully receiving an RTS, replies with a CTS—if not currently transmitting or engaged in another reception—over the primary channel after a Short Inter-frame Space (SIFS) period.

NAV

If an RTS or CTS transmission is successful, the simulator adopts a simplified approach to NAV handling: instead of maintaining explicit per-node NAV timers, all nodes contending on the same primary channel treat the medium as busy for the duration of the scheduled transmission, deferring their access accordingly.

A-MPDU transmission and acknowledgment

After a successful RTS/CTS exchange (if enabled), the transmitter sends the A-MPDU after a SIFS interval, using a set of channels determined by the channel access scheme—either SCB or AM DCB—and the assigned MCS. If no RTS/CTS is used, the SIFS period is omitted. A Block Acknowledgment (BACK) timeout starts after the last bit of the A-MPDU is transmitted. If the corresponding BACK frame is not received within the timeout period, the aggregated MPDUs are considered as failed. If RTS/CTS exchange is not enabled, a new backoff stage begins after sensing the channel idle during DIFS and adjusting the CW accordingly. If the corresponding BACK frame is received, successfully acknowledged MPDUs are removed from the transmission queue. Any failed MPDUs are retained in the transmission queue for subsequent transmission unless their retry count exceeds the limit.

The wireless medium probabilistically corrupts MPDUs based on the configured PER.

The receiver, upon receiving an A-MPDU, generates a BACK frame the includes the success/failure of each subframe. The size of the BACK depends on the number of subframes in the aggregate. Successfully received MPDUs are processed, passing the payload data from the MAC layer to the traffic sink.

Channel occupation and collisions

Any ongoing transmission on a channel occupies that channel for the entire transmission duration. If another node attempts to transmit on a busy channel, a collision occurs, and both frames involved are marked as corrupted. Otherwise, the receiver successfully receives the frame at its PHY layer.

Nodes attempt to access the channel depending on the outcome of the previous transmission on the channel. If a collision occurred in the channel, rather than considering a DIFS interval before initiating its next transmission attempt, the node considers EIFS. Otherwise, the node considers a DIFS.

2 Reinforcement Learning integration

2.1 Design considerations

Scope

The simulator's APs can act as independent learners, each unaware of the others' actions. Nevertheless, this thesis focuses on Wireless Local Area Network (WLAN) scenarios involving a single decentralized RL-driven AP. Scenarios with multiple RL-enabled APs introduce substantial complexity due to adversarial inter-AP interactions and the inherent non-stationarity that arises when operate concurrently and compete for shared wireless resources [117]. Indeed, in such environments, agents may continuously adapt to one another, leading to unstable convergence and policy-chasing behavior. A comprehensive study of such scenarios is left for future work.Considering a single learning AP enables us to focus on exploring the benefits of cooperative multi-agent approaches against single-agent strategies. Hence, this thesis adopts a cooperative multi-agent approach in which several agents operating in the AP MAC layer—each responsible for a subset of decisions—share a common goal and optimize a joint reward function, evaluating it against a single-agent setting. This intra-AP multi-agent approach maintains problem complexity tractable by decomposing the joint action space [117], thus reducing the dimensionality of each agent's decision-making process.

This thesis primarily evaluates online learning performance in static network environments, where non-learning APs use fixed channel assignments. Nevertheless, to assess its operation in more dynamic (and realistic) scenarios, an evaluation involving non-learning APs that randomly change their operating channels based on simple stochastic heuristics is also carried out, enabling us to assess the learning agent's adaptability and responsiveness.

Abstractions

As described in Section A.2, four basic 20 MHz channels are considered. Thus, the set of valid channel groupings is $\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}, \{1, 2, 3, 4\}\}$, enabling bandwidths of 20, 40, and 80 MHz. Instead of relying on Binary Exponential Backoff (BEB), described in

Section A.1.1, the CW size remains unchanged throughout the entire transmission cycle[8]—including RTS retries—for the learning AP. Since all associated STAs must operate over the same channel configuration as their serving AP, it is assumed that the selected channel group and the primary channel are known to all associated STAs before each transmission. This coordination is abstracted in the simulator and not explicitly modeled. Furthermore, SCB is considered—as DCB may diminish the impact of channel group selection—and RTS/CTS exchange is enabled across all scenarios.

Agent decision timing and reward structure

Agent decisions are made at the beginning of each transmission cycle. The reward is provided at the end of the cycle, determined by either the successful reception of the BACK frame or its timeout. In the event of a transmission failure—such as due to secondary channels unavailability during the PIFS period or a CTS timeout—the agent's selected action persists until the end of the ongoing transmission cycle. This structure ensures that feedback is received after an action's outcome, thereby avoiding delayed rewards[9].

In the single-agent setup, the agent selects an action at the beginning of every transmission cycle. In the multi-agent approach, since decision-making is distributed across several specialized agents, each responsible for a distinct aspect of the MAC-layer, decisions may occur at different logical stages within the transmission cycle. In particular, channel group and primary channel selection must happen before carrier sensing, but CW size selection may occur immediately after sensing, just prior to initiating the backoff procedure. Our evaluation adopts a uniform decision point at the start of each transmission cycle for both single-agent and multi-agent setups, enabling a fairer comparability.

Furthermore, in the multi-agent architecture, all agents may operate at the same time (e.g., at each transmission cycle) or at distinct intervals (e.g., channel group selection every four cycles, primary channel selection every two cycles, and CW selection every cycle). Our evaluation focuses solely on the simultaneous operation of all agents.

Regarding the reward structure, initial designs were based on (average) per-packet delay, accounting for packets' queue waiting times. Nevertheless, considering that packet retransmissions may occur, this could introduce feedback noise from prior actions. Thus, rather than using a

[8]A transmission cycle encompasses the initial carrier sensing, backoff, RTS/CTS exchange (if enabled), data transmission, and acknowledgment handling. Thus, a transmission cycle completes once data transmission succeeds or times out

[9]Delayed rewards refer to feedback that is only available after several time steps. A time step is defined in terms of agent decisions, rather than simulation clock time.

per-packet reward, our reward is based on the duration of a transmission cycle—from the initial sensing phase to the reception of a BACK frame or its timeout. This captures the efficiency of the channel access procedure. Notice that, give the RTS/CTS exchange and the assumption of no hidden nodes, a BACK timeout is highly unlikely. Hence, the agent's goal is to minimize the duration of transmission cycles, promoting faster and more efficient medium access.

Implementation details

The decision logic for both single-agent and multi-agent approaches is encapsulated in a dedicated module (`rl_agents.py`). The simulator has been adapted to incorporate all RL-specific adjustments, overriding the default channel group selection, primary channel selection, and CW size logic. Other MAC-layer operations—including frame aggregation, RTS/CTS handling, acknowledgment, and queue management—remain unmodified. W&B is further leveraged for logging agent decisions, reward metrics, and transmission statistics. In addition, it has been incorporated Optuna[15] to tune the agents' hyperparameters, and CodeCarbon[2] to estimate energy consumption, enabling comparison between the single-agent and multi-agent approaches from a sustainability perspective.

2.2 Why Contextual Multi-Armed Bandits?

CTDE approaches, considered as one of the most promising paradigms for Coo-MARL due to their scalability and ability to handle non-stationarity [124], have been widely adopted in the context of wireless MAC optimization [101], especially in scenarios involving multiple RL-enabled APs, as they facilitate global performance optimization while promoting inter-agent fairness [107].

However, since this thesis primarily focuses on scenarios involving a single RL-enabled AP, centralized training is neither necessary nor practical. Furthermore, the applicability of CTDE in dynamic and unpredictable wireless environments is notably limited, as it often generalizes poorly to conditions not encountered during training [101]. Indeed, such approaches often require extensive training over a large number of episodes to mitigate overfitting, leading to substantial computational overhead. Moreover, CTDE assumes a centralized training phase, which is impractical in distributed WLAN deployments [101, 107]. Indeed, state-of-the-art CTDE-based solutions typically rely on simulation-based training under omniscient environment assumptions, neglecting the communication and synchronization overheads inherent in real deployments [101]. Their performance during execution is also tightly coupled to the fidelity of the training environment, requiring fine-tuned simulations tailored to the specific deployment scenario [101]. Moreover, centralized solutions also face inherent scalability limitations [107], particularly as the number of agents increases.

On the other hand, decentralized learning approaches may be better suited for WNs—especially when inter-AP cooperation is feasible—but only if the optimization of each AP's local objectives leads to a globally efficient outcome, i.e., a Pareto-optimal solution. [107] In practice, however, cooperation among competing devices often requires explicit inter-device communication and coordination, which introduces additional overhead and is generally unfeasible in current WLAN deployments. [107] Consequently, this thesis adopts a decentralized approach without cooperation between APs, relying instead on intra-AP cooperation.

In particular, in the multi-agent architecture, this thesis adopts an independent learning approach. Independent learning is the simplest way to address a multi-agent problem, treating it as a collection of single-agent problems [96]. Independent learners often struggle with non-stationarity and the credit assignment problem, especially in cooperative scenarios where agents' actions jointly influence the outcome [96]. Nevertheless, although independent learning lacks formal theoretical guarantees for finding globally optimal policies, in practice it often achieves satisfactory performance [71, 125]. Indeed, wireless nodes operating selfishly in adversarial settings can converge to proportionally fair outcomes, even without explicit coordination. [115]

This thesis leverages CMABs. CMABs offer a compelling balance between adaptability, scalability, and implementation simplicity, enabling lightweight and efficient online learning. Thus, CMABs are particularly well-suited for WLAN scenarios—as demonstrated by multiple prior studies (see Section 3).

Full RL approaches such as tabular Q-learning or DQN-based methods primarily aim to optimize long-term cumulative rewards (i.e., return). However, in WNs the focus is on the immediate outcome of individual transmissions, as actions typically affect only the current transmission attempt. Consequently, learning algorithms that prioritize short-term feedback, such as (C)MABs, are more suitable. Furthermore, tabular methods suffer from the *curse of dimensionality*, requiring the learning of an action-value function over a potentially large and sparse state-action space. This leads to slow convergence, especially in environments with high-dimensional action and state spaces, even after discretizations. In contrast, CMABs offer lightweight learning by leveraging action representations through feature vectors, achieving significantly faster convergence.
MABs are also significantly simpler to implement [118]—often at the expense of yielding suboptimal policies [118]—and do not require *a priori* knowledge of the environment's dynam-

ics [107, 111]. Additionally, MABs naturally support online learning, enabling agents to adapt in real-time to changing conditions and generalize across varying scenarios, as required in real-world IEEE 802.11 networks [111, 117]. Notably, MABs typically enable faster online learning than Q-learning and DQN-based approaches [116], and thus are more appropriate in dynamic scenarios [118].

The wireless environment is inherently stochastic and non-stationary due to varying traffic patterns, channel contention, and interference from neighboring APs. Thus, our solution explicitly accounts for non-stationarity, leveraging learning methods that handle it. Classic MABs, such as ϵ-greedy and UCB, assume stationary reward distributions [126], but there are several variants that deal with non-stationarity (e.g., Sliding-Window UCB [127] and Discounted UCB [127]). In addition to algorithmic approaches, environments that appear non-stationary can often be treated as stationary by incorporating context [29]. Furthermore, it is possible to consider hierarchical MABs structures, where each action at a higher level triggers an independent MAB at the subsequent level such that, from the perspective of each sub-agent, the environment appears stationary. This may lead to faster convergence, but introduces challenges such as the combinatorial explosion in the number of agents and increased data sparsity.

3 Action, context, and reward design

In a CMAB problem, at each time step t, a learner observes a context vector $\mathbf{x}_t \in \mathbb{R}^d$, selects an action $a_t \in \mathcal{A}$, and receives a reward $r_t(a_t, \mathbf{x}_t)$. This section presents the action and context spaces, as well as the reward structure, for both the single-agent and multi-agent CMAB-based architectures used in this thesis.

3.1 Action Space

Multi-agent

The multi-agent design decomposes decision-making across three specialized agents, each controlling a different aspect of the AP's MAC-layer behavior and operating over a discrete, finite action space:

- **Channel Selection Agent (CSA)**: selects one seven possible contiguous channel groupings a_t^{CSA} from the action space $\mathcal{A}^{\text{CSA}}$:

$$\mathcal{A}^{\text{CSA}} = \{\{1\},\{2\},\{3\},\{4\},\{1,2\},\{3,4\},\{1,2,3,4\}\}, \quad |\mathcal{A}^{\text{CSA}}| = 7$$

These groupings correspond to bandwidths of 20, 40, and 80 MHz, and are mapped to integer indices (e.g., $0:\{1\}, 1:\{2\}, ..., 6:\{1,2,3,4\}$). Wider bandwidths enable faster data transmissions but increase the probability of spectral overlapping and contention, potentially degrading performance. [98]

- **Primary Channel Selection Agent (PCSA)**: selects a primary channel a_t^{PCSA} from the set of 20 MHz basic channels included in the currently selected channel group a_t^{CSA}, using *action masking*. Thus, the available actions at each time step t depend on the current CSA selection and are a subset of the action space $\mathcal{A}^{\text{PCSA}}$:

$$\mathcal{A}^{\text{PCSA}} = \{1,2,3,4\}\,, \quad |\mathcal{A}^{\text{PCSA}}| = 4$$

For instance, if $a_t^{\text{CSA}} = \{1,2\}$, then $a_t^{\text{PCSA}} \in \{1,2\}$, whereas if $a_t^{\text{CSA}} = \{1,2,3,4\}$, then $a_t^{\text{PCSA}} \in \{1,2,3,4\}$. Likewise, primary channel selections are also mapped to integer indices (e.g., $0:\{1\}, 1:\{2\}, \dots, 3:\{4\}$).

- **Contention Window Selection Agent (CWSA)**: selects the CW size a_t^{CWSA} from the action space $\mathcal{A}^{\text{CWSA}}$:

$$\mathcal{A}^{\text{CWSA}} = \left\{2^{i+4} \mid i = 0,1,\dots,6\right\} = \{16,32,64,128,256,512,1024\}, \quad |\mathcal{A}^{\text{CWSA}}| = 7$$

This covers the standard range of CW sizes defined in IEEE 802.11 [107].[10] The CW size determines the AP's aggressiveness in accessing the wireless medium. Larger CW values reduce the likelihood of collisions but at the expense of increased backoff delay. Likewise, each CW size is mapped to an integer index (e.g., $0:16$, $1:32$, $\dots$, $6:1024$).

Single-agent

In the single-agent approach, the agent selects a joint action $\left(a_t^{\text{CSA}}, a_t^{\text{PCSA}}, a_t^{\text{CWSA}}\right)$. The action space is thus the combinatorial product of the individual decisions, considering the conditional dependence between $\mathcal{A}^{\text{CSA}}$ and $\mathcal{A}^{\text{PCSA}}$:

$$|\mathcal{A}^{\text{SA}}| = \sum_{c \in \mathcal{A}^{\text{CSA}}} |\mathcal{A}^{\text{PCSA}}(c)| \cdot |\mathcal{A}^{\text{CWA}}| = 84$$

[10] An alternative design was considered, using a reduced action set $\mathcal{A}^{\text{CWSA}} = \{\text{decrease}, \text{maintain}, \text{increase}\}$ to promote smoother and more stable adjustments to the CW size. However, this approach required the current CW size to be included in the context, as the meaning of each relative action depends on the current value, thereby breaking the statelessness assumption of MABs.

3.2 Context Space

Context vectors provide side information to the agent(s) at each decision step, enabling more informed and context-aware decisions. Our selected features are based on domain expertise but could be refined via ablation studies. All values are normalized to $[0, 1]$, ensuring numerical stability and comparable feature scales. It is assumed that, in a real system, APs can retrieve contextual information such as the channel occupancy ratio.

Multi-agent

In the multi-agent approach, each agent receives a context vector tailored to its specific decision domain and operates independently. Note that context features are not influenced by the agent's actions, thus preserving statelessness:

CSA context ($\mathbb{R}^9$)

- ($\mathbb{R}^4$) Channel occupancy ratio[11] for each basic channel.
- ($\mathbb{R}^4$) Binary occupancy flag for each basic channel: 0 (idle), 1 (busy).
- ($\mathbb{R}$) Transmission queue occupancy ratio[12].

PCSA context ($\mathbb{R}^9$)

- ($\mathbb{R}$) Normalized index of the currently selected channel group.
- ($\mathbb{R}^4$) Channel occupancy ratio for each basic channel.
- ($\mathbb{R}^4$) Binary occupancy flag for each basic channel.

CWSA context ($\mathbb{R}^{11}$)

- ($\mathbb{R}$) Normalized index of the currently selected channel group.
- ($\mathbb{R}$) Normalized index of the currently selected primary channel.
- ($\mathbb{R}^4$) Channel occupancy ratio for each basic channel.
- ($\mathbb{R}^4$) Binary occupancy flag for each basic channel.

[11]The channel occupancy ratio is estimated as the fraction of time each channel is sensed as busy—using physical carrier sensing—excluding periods during which the AP itself or its associated STAs are transmitting, in order to avoid circularity and preserve statelessness. A sliding window of duration $w_{\mathrm{oc}} = 100$ ms is applied to smooth short-term fluctuations and capture meaningful temporal patterns. At startup, until the window reaches w_{oc}, the estimate is computed over the elapsed time since system initialization.

[12]Transmission queue occupancy ratio is computed as the ratio ratio between the number of packets in the transmission queue and the maximum queue size.

- ($\mathbb{R}$) Transmission queue occupancy ratio.

Note that all agents observe the full set of basic channels in their context vectors—regardless of the currently selected channel group—to ensure consistent input dimensionality. It is assumed that including the currently selected channel group in the context enables both the PCSA and CWSA to infer which channel-specific features are relevant at each decision step.

Single-agent

In the single-agent architecture, the context ($\mathbb{R}^9$) includes:

- ($\mathbb{R}^4$) Channel occupancy ratio for each basic channel.
- ($\mathbb{R}^4$) Binary occupancy flag for each basic channel.
- ($\mathbb{R}$) Transmission queue occupancy ratio.

Note that the currently selected channel group and primary channel are not included in the context. This design preserves statelessness and avoids implicitly conditioning the agent on its prior actions.

3.3 Reward Structure

Multi-agent and Single-agent

The reward r_t at time t is defined as:

$$r_t = -d_t,$$

where d_t is the total transmission cycle duration (in μs), from initial carrier sensing to BACK frame reception or timeout. This reward function encourages minimizing the duration of transmission cycles, thereby promoting faster and more efficient medium access. The reward is shared among (intra-AP) agents, fostering cooperation and joint optimization toward a common goal. Furthermore, the simplicity of this formulation helps avoid unintended behaviors such as reward hacking [128].

4 Contextual bandit algorithms

There are multiple contextual bandit algorithms designed to balance exploration and exploitation, such as LinUCB [40]. LinUCB models the expected reward as a linear function of the context and selects actions using an UCB strategy (see Section 1.8). Nevertheless, LinUCB assumes stationarity in the reward distributions—an assumption that often does not hold in dynamic environments such as WNs. Notably, several adaptations of LinUCB have been proposed to handle non-stationary environments, such as Sliding Window LinUCB (SW-LinUCB) [129] and Discounted LinUCB (D-LinUCB) [130].

In this thesis, SW-LinUCB is adopted as the learning algorithm for both the Single-Agent Contextual Multi-Armed Bandit (SA-CMAB) and the cooperative Multi-Agent Contextual Multi-Armed Bandit (MA-CMAB) architectures. Additionally, a simple linear contextual bandit algorithm is proposed, combining ϵ-greedy exploration with an online Stochastic Gradient Descent (SGD)-based update rule (specifically, RMSProp [131]). In Section 2, the proposed algorithm is benchmarked against SW-LinUCB, evaluating its convergence speed and overall performance in both single-agent and multi-agent settings.

Below, both SW-LinUCB and the proposed ϵ-greedy RMSProp-based algorithm are described. Both algorithms have been implemented from scratch in the simulator, without relying on existing implementations (e.g., [37]), enabling custom features such as action masking. In the multi-agent architecture, all agents are considered homogeneous: each agent operates independently but uses the same learning algorithm (e.g., SW-LinUCB) and hyperparameters (finetuned in Section 1).

SW-LinUCB, summarized in Algorithm 2, extends LinUCB to handle non-stationary environments by incorporating a time-discounting mechanism through a sliding window of size w.[13] In particular, SW-LinUCB modifies the original LinUCB payoff $p_{t,a}$ as follows:

$$p^w_{t,a} = \gamma_t \hat{\theta}_a^\top \mathbf{x}_t + \alpha\sqrt{\mathbf{x}_t^\top \mathbf{A}_a^{-1}\mathbf{x}_t}, \tag{3.1}$$

where γ_t is a discount factor that penalizes arms that have been selected frequently in recent trials, encouraging diversity in the selection of sub-optimal arms. Note that, unlike LinUCB, SW-LinUCB considers an arm-independent context vector $\mathbf{x}_t$, rather than $\mathbf{x}_{t,a}$. This coefficient is computed as:

$$\gamma_t = 1 - \frac{Occ_w(a,t)}{w},$$

where $Occ_w(a,t)$ denotes the number of times arm a has been pulled in the last w rounds[14].

[13]The authors consider $w = k$, with k being the number of arms, to balance exploration and exploitation. In our implementation, w is a tunable parameter; setting $w = 0$ disables discounting (i.e., $\gamma_t = 1$), thus yielding the standard (stationary) LinUCB algorithm.

[14]SW-LinUCB maintains a binary sequence $\mathbf{E}_{t,a}$ of length w for each arm a, where a 1 indicates that the arm was selected in a given round. For instance, with $w = 6$, the sequence 101001 means that arm a was selected in rounds $t-6$, $t-4$, and $t-1$. This leads to $Occ_w(a,t) = 3$, and consequently a discount factor of $\gamma_t = 0.5$.

ϵ-greedy RMSProp-based algorithm, summarized in Algorithm 3, is a lightweight contextual bandit algorithm that combines a simple ϵ-greedy exploration strategy with online updates via RMSProp to learn arm-specific parameter vectors, under the (disjoint) linear realizability assumption [42,43]. Specifically, it assumes the expected reward for arm a given context $\mathbf{x}_t$ can be modeled as: $\mathbb{E}[R_t \mid \mathbf{x}_t] = \mathbf{x}_t^\top \boldsymbol{\theta}_a^*$, where $\boldsymbol{\theta}_a^* \in \mathbb{R}^d$ is an unknown parameter vector associated with arm a (see Section 1.8).
After receiving reward r_t for the selected arm a_t, the algorithm updates the estimate $\hat{\boldsymbol{\theta}}_{a_t}$ of the true parameter vector $\boldsymbol{\theta}_{a_t}^*$ using RMSProp [131], an adaptive variant of SGD. SGD is a simple and efficient optimization method, especially effective in high-dimensional spaces [132] and the (vanilla) SGD update rule [133] is as follows:

$$\hat{\boldsymbol{\theta}}_a \leftarrow \hat{\boldsymbol{\theta}}_a - \eta \cdot g_t, \tag{3.2}$$

where $\eta > 0$ is the learning rate and $g_t = \nabla_{\hat{\boldsymbol{\theta}}_a} \mathcal{L}(\hat{\boldsymbol{\theta}}_a)$ is the gradient of the loss function $\mathcal{L}$ with respect to $\hat{\boldsymbol{\theta}}_a$.[15] Nevertheless, rather than using a fixed learning rate, RMSProp is considered.[16]
RMSProp divides the learning rate by an exponentially decaying average of squared gradients, enhancing stability and speeding up convergence [138]. It modifies the vanilla SGD update rule (Eq. 3.2) as follows [138]:

$$\hat{\boldsymbol{\theta}}_a \leftarrow \hat{\boldsymbol{\theta}}_a - \frac{\eta}{\sqrt{v_t + \varepsilon}} \cdot g_t, \tag{3.3}$$

where

$$v_t = \gamma v_{t-1} + (1-\gamma) g_t^2$$

Here, $\gamma \in (0, 1]$ is the decay factor (usually, 0.9, as suggested by the authors [131]) and $\varepsilon = 1e{-}8$ is a small constant that avoids division by zero.
The algorithm minimizes the commonly adopted squared error loss [134], defined as:

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_a) = \frac{1}{2}(\hat{r}_{t,a} - r_t)^2$$

where $\hat{r}_{t,a}$ denotes the estimated reward for arm a based on the current parameter vector and $(\hat{r}_{t,a} - r_t)$ represents the prediction error.
Considering the non-stationarity of WNs, the algorithm maintains an additional exponentially smoothed estimate of each arm's parameter vector, denoted as $\hat{\boldsymbol{\theta}}_a^{\text{ema}}$, which is updated using an

[15] A weighting factor w_t may also be incorporated [134]: $w_t \cdot g_t$.
[16] Besides RMSProp, other popular gradient descent optimization algorithms include Adagrad [135], Adadelta [136], and Adam [137]. Notably, RMSProp builds upon Adagrad, resolving Adagrad's radically diminishing learning rates. [138]

Exponential Moving Average (EMA) rule [1, 139]:

$$\hat{\boldsymbol{\theta}}_a^{\text{ema}} \leftarrow \alpha_{\text{ema}} \cdot \hat{\boldsymbol{\theta}}_a^{\text{ema}} + (1 - \alpha_{\text{ema}}) \cdot \hat{\boldsymbol{\theta}}_a,$$

where $\alpha_{\text{ema}} \in [0, 1)$ is the smoothing factor. This EMA-smoothed parameter is used solely for action selection, allowing the agent to better adapt to recent changes in the environment. The raw parameter vector $\hat{\boldsymbol{\theta}}_a$ is updated using the RMSProp rule (Eq. 3.3), as described above.
To address the exploration-exploitation dilemma, the algorithm uses ϵ-greedy, which has shown robust performance in CMABs [140]. Using a constant ϵ ensures adaptability in non-stationary environments such as WNs [140]. As described in Section 1.7, at each time step t, the algorithm selects the arm a with the highest estimated reward $\hat{r}_{t,a}^{\text{ema}}$ with probability $1 - \epsilon$ (exploitation), and a random arm with probability ϵ (exploration). Note that the estimated reward used for action selection, $\hat{r}_{t,a}^{\text{ema}}$, is computed using the exponentially smoothed parameter vector:

$$\hat{r}_{t,a}^{\text{ema}} = \mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_a^{\text{ema}}$$

Upon receiving reward r_t after selecting an action based on context $\mathbf{x}_t$, the algorithm computes the prediction error and performs a gradient descent step using the raw parameter vector. The gradient term in Eq. 3.3 becomes:

$$g_t = (\hat{r}_{t,a} - r_t) \cdot \mathbf{x}_t$$

where $\hat{r}_{t,a} = \mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_a$.[17] Note that, similar to SW-LinUCB, an arm-independent context vector $\mathbf{x}_t$ is considered.

Our proposed SGD-based algorithm is lightweight and computationally efficient, with an update complexity of $\mathcal{O}(d)$ [141]. Note that matrix-inversion-based methods such as LinUCB or SW-LinUCB incur a higher computational cost: $\mathcal{O}(d^3)$. Other linear contextual bandit algorithms, leveraging online SGD, can be found (e.g., [134, 142, 143]). In particular, [142] relies on TS exploration and single-step SGD updates, while [143] uses ϵ-greedy and SGD updates with inverse probability weighting.
Additionally, SW-LinUCB—like other UCB-based methods—requires rewards to be bounded in

[17]Technically, since the selected action is based on $\hat{\boldsymbol{\theta}}_a^{\text{ema}}$, the chain rule introduces an additional factor of $(1 - \alpha)$ in the gradient g_t. Nevertheless, this dependency is ignored to avoid added complexity, treating $\hat{\boldsymbol{\theta}}_a$ as the effective prediction parameter.

the interval $[0, 1]$[18], whereas our proposed algorithm uses a simple ϵ-greedy strategy—which can, in principle, operate on unbounded reward signals. However, in RMSProp, large-magnitude rewards can produce extremely large gradients that shrink the learning rate and lead to minimal weight updates, stalling learning—especially early in training. Thus, since our algorithm relies on RMSProp and our reward signals are negative and unbounded (as defined in Section 3.3), reward normalization is also required. Nevertheless, normalization in online learning settings is non-trivial. Techniques such as batch normalization [144] are not directly applicable [145, 146] and non-linear transformations (e.g., sigmoid normalization [147]) can distort the reward structure, violating the linear realizability assumption. Z-score normalization, particularly using EMAs [148, 149], is better suited for online learning but does not guarantee outputs within $[0, 1]$. Therefore, our implementation adopts a min-max normalization strategy with clipping to predefined bounds, a method also used in adaptive algorithms such as AdaUCB [150]. Specifically, the normalized reward $\widetilde{r}_t$ is computed as:

$$\widetilde{r}_t = \frac{\max(\min(r_t, r_{\min}), r_{\max}) - r_{\min}}{r_{\max} - r_{\min}}$$

Here, $r_{\max}$ and $r_{\min}$ define the expected upper and lower bounds of the reward signal, and the inner clipping operation, $\max(\min(r_t, r_{\max}), r_{\min})$, ensures that the observed reward r_t does not exceed the specified range. In practice, these thresholds can be estimated online [150]; however, for simplicity and stability, fixed values are used in our implementation: $r_{\min} = -10$ ms (a value that approximately captures the 80th percentile latency of a Wi-Fi hop [151]) and $r_{\max} = 0$ ms (since the defined reward is strictly negative). It is important to mention that the choice of clipping thresholds is scenario-dependent. A wider range may capture large delays but compress the differences in rewards, thereby reducing their granularity and distinguishability. Conversely, narrow bounds may lead to excessively frequent clipping.

[18]In the presence of unbounded rewards, the exploration bonus (i.e., the right-hand-side term in Eq. 3.1) may become disproportionately small relative to the scale of the expected reward, $\hat{\theta}_a^\top \mathbf{x}_t$, thus diminishing the algorithm's exploratory behavior. Additionally, in the case of consistently negative rewards, the discounting mechanism may encourage frequently selected arms—contrary to its intended purpose.

Chapter 4

Hyperparameter tuning and Evaluation

1 Hyperparameter tuning

Before evaluating the performance of the proposed single-agent and multi-agent architectures, it is essential to optimize the hyperparameters of the underlying contextual bandit algorithms: SW-LinUCB and the proposed ϵ-greedy RMSProp-based algorithm. These algorithms include several tunable parameters—such as learning rates, smoothing factors, and exploration coefficients—that significantly influence the agents' learning behavior and effectiveness. Proper hyperparameter tuning not only improves convergence speed and reward maximization but also enhances the robustness and generalizability of the learning algorithms across diverse network deployments and runtime conditions.

1.1 Tuning methodology

Hyperparameter tuning is conducted using Optuna[15], an automatic hyperparameter optimization framework. Optuna leverages Bayesian Optimization [152], a method that efficiently explores the hyperparameter space—reducing the number of trials needed to find near-optimal configurations—and outperforms classical strategies such as Random Search [153] and Grid Search [154]. Specifically, Optuna uses the Tree-structured Parzen Estimator (TPE) algorithm [155].

Hyperparameters are tuned separately for the single-agent and multi-agent architectures. In the multi-agent setting, all agents operate at the beginning of each transmission cycle and are considered homogeneous, i.e., each agent uses the same learning algorithm and hyperparameters.

For each algorithm–architecture pair, the optimization process evaluates 100 candidate configurations. Our proposed algorithm involves a larger hyperparameter space, and thus, usually requires more optimization iterations. Nevertheless, additional iterations did not yield significant improvements in rewards.

Similar to the approach adopted in [111], each trial is assessed over a fixed set of scenarios. For each scenario, the mean reward over the entire simulation run is computed. The trial score is then defined as the average of these per-scenario means and the configuration that achieves the highest trial score is selected. Tbl. 4.1 details the hyperparameters tuned for each algorithm and their respective search spaces.

Each scenario is assigned a fixed random seed to ensure reproducibility and consistent comparison across trials. The set of scenarios spans a broad spectrum of wireless deployments and simulation durations, promoting generalizability while supporting both fast convergence and long-term performance. Specifically, scenarios vary in the number of Basic Service Sets (BSSs)—$n_{\text{bss}} \in \{2, 3, 4\}$—and in simulation durations—$t_{\text{sim}} \in \{1, 2, 4, 8\}$ seconds. To avoid introducing sampling bias, all possible $(n_{\text{bss}}, t_{\text{sim}})$ combinations are considered, yielding 12 scenarios.

Each scenario features a single learning Access Point (AP) and uses the simulation parameters defined in Tbl. 3.1. Each BSS includes a single transmitter-receiver pair, and only Downlink (DL) traffic is considered (i.e., from the AP to its associated client). Network nodes are randomly placed within a 10×10×2 m space.
Channel allocations for all APs—excluding the learning AP—are fixed during the simulation and sampled uniformly at random from the set of valid channel groupings:

$$\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}, \{1, 2, 3, 4\}\}$$

For these APs, the primary channel is assigned as the lowest-indexed channel in the selected group, since—in the 5 GHz band—the first 20 MHz segment of any 40, 80, or 160 MHz channel almost always serves as the primary [156]. Each AP generates traffic throughout the simulation using a traffic model selected uniformly at random from: Poisson, Bursty, Virtual Reality (VR), or full-buffer. Traffic characteristics—such as load (ranging from 10 and 500 Mbps), inter-arrival times, and frame rates—are also randomly drawn per scenario, enabling the tuning process to account for diverse scenarios.

Table 4.1: Tuned hyperparameters and search ranges.

Algorithm	Param.	Description	Search range
SW-LinUCB	α	UCB exploration coefficient	$\mathcal{U}(0.1,\ 2.0)$
	w	Sliding window size	$\mathcal{U}(0,\ 84)$
RMSProp-based	ϵ	ϵ-greedy exploration rate	$\mathcal{U}(0.01,\ 0.30)$
	η	RMSProp learning rate	$log\ \mathcal{U}(10^{-4},\ 10^{-1})$
	γ	RMSProp decay factor	$\mathcal{U}(0.70,\ 0.99)$
	α_{ema}	EMA smoothing factor	$\mathcal{U}(0.01,\ 0.30)$

1.2 Optimization results

The best hyperparameter configurations identified for each algorithm–architecture pair are summarized in Tbl. 4.2, along with the average reward achieved by each configuration across all scenarios. Notably, each tuning process required more than 30 hours to complete, even using pruning strategies to discard clearly suboptimal configurations during the search.

Table 4.2: Best hyperparameters found and average reward obtained across all scenarios.

(a) SW-LinUCB.

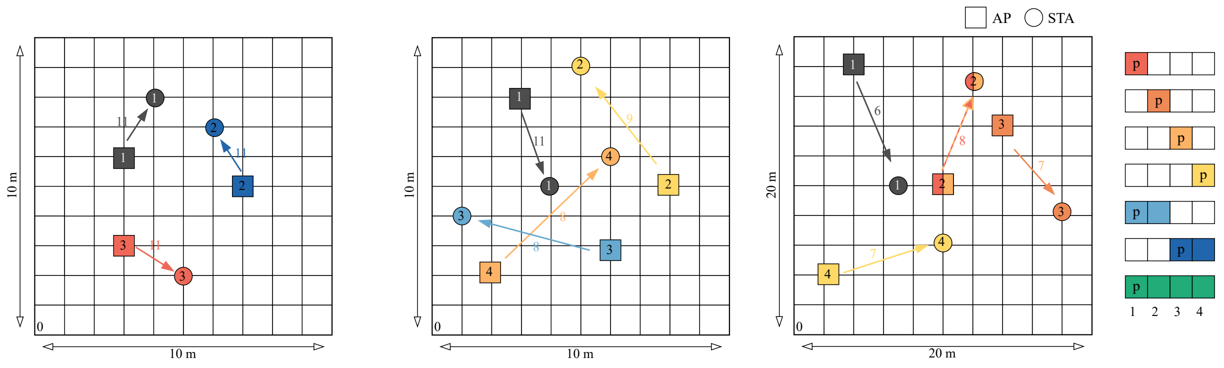
	α	w	reward
Single-agent	0.113	38	-4474.3
Multi-agent	0.220	35	-3159.2

(b) RMSProp-based algorithm.

ϵ	η	γ	α_{ema}	reward
0.0161	0.0194	0.758	0.064	-4737.3
0.0187	0.0514	0.836	0.197	-3057.7

For SW-LinUCB, the best-performing configurations include: relatively low exploration coefficients (α), indicating a clear tendency toward exploitation; and moderate sliding window sizes (w), encouraging diversity in exploration by incorporating a broader and more recent history. For the proposed ϵ-greedy RMSProp-based algorithm, the identified configurations include: extremely low ϵ values, showing a strong preference for exploitation; moderate learning rates (η), promoting gradual and stable parameter updates; high decay factors (γ), preserving the influence of past gradient updates while still allowing responsiveness to recent changes; and small smoothing factors (α_{ema}), giving increased influence to recent updates in the parameters used for arm selection.

Notably, the multi-agent architecture achieved significantly higher average rewards than the single-agent setting for both algorithms. This suggests that Multi-Agent Contextual Multi-Armed Bandits (MA-CMABs) are likely to outperform Single-Agent Contextual Multi-Armed Bandits (SA-CMABs) in the upcoming evaluation phase.



(a) Scenario A.

(b) Scenario B.

(c) Scenario C.

Figure 4.1: BSS deployments and channel group assignments for each toy scenario. The learning AP belongs to BSS 1. Arrows indicate traffic direction and the MCS values are annotated. Only the x-y plane is illustrated. Note that Scenario C spans a larger area.

2 Evaluation

This section evaluates the performance of the implemented Contextual Multi-Armed Bandit (CMAB) algorithms in both the single-agent and multi-agent architectures. In addition to comparing the performance of the SW-LinUCB algorithm and the proposed ϵ-greedy RMSProp-based algorithm, this section also investigates the advantages of cooperative multi-agent learning over the traditional single-agent architecture—addressing the objectives defined in Section 3. Furthermore, legacy non-learning baselines, representing conventional IEEE 802.11 operation, are included to highlight the benefits of CMAB-based approaches in optimizing Wireless Networks (WNs).

The evaluation uses the best hyperparameter configurations identified in Section 1 (see Tbl. 4.2) and each trial runs on a Windows 11 desktop equipped with an AMD Ryzen 9 7900X 12-Core Processor CPU and an NVIDIA GeForce RTX 4070 Ti GPU.

2.1 Toy scenarios

Three controlled toy scenarios (A, B, and C)—distinct from those used for hyperparameter tuning (Section 1)—are designed to evaluate the adaptability, efficiency, and learning capability of the proposed CMAB algorithms. Each (reproducible) scenario runs for 60 seconds of simulated time and is repeated across five independent trials, using the simulation parameters specified in Tbl. 3.1. Scenarios A and B are deployed in a $10 \times 10 \times 2$ m space, while Scenario C spans a larger $20 \times 20 \times 2$ m area. The number of co-located BSSs increases from three in Scenario A to four in Scenarios B and C, as illustrated in Fig. 4.1.

As in Section 1, each scenario includes a single Reinforcement Learning (RL)-enabled AP (specifically, the AP in BSS 1, i.e., AP 1) that dynamically adjusts its channel group, primary channel, and Contention Window (CW) size. The remaining APs operate—in general—with static configurations: each is assigned a channel group, uses the first basic channel in that group as the primary, and relies on the Binary Exponential Backoff (BEB) procedure—which increases the CW size after each unsuccessful transmission, as described in Section A.1.1. All APs generate DL traffic throughout the simulation. In Scenario A, all APs operate under saturated conditions (i.e., full-buffer traffic), modeling high-contention environments—which represent the worst-case scenario for network performance. In contrast, Scenario B and C include non-saturated, heterogeneous traffic models to emulate more realistic usage patterns.

Scenario A (Tbl. D.1) serves as a controlled baseline designed to test the ability of the CMAB-based solutions to identify (and exploit) a clearly optimal channel allocation. It consists of three BSSs, all operating under saturated (full-buffer) traffic conditions and transmitting using the same Modulation and Coding Scheme (MCS). From a network administrator's perspective, the optimal decision is obvious: channel $\{2\}$ is unassigned to any of the non-learning APs and is therefore interference-free. This scenario evaluates whether the implemented CMAB-based solutions can successfully learn to select this optimal channel group, validating the correct functioning of the learning mechanisms.

Scenario B (Tbl. D.2) represents a more realistic environment. It includes four BSSs, each with distinct traffic models and MCSs. Specifically, AP 1 (the learning AP) maintains saturation conditions (full-buffer); AP 2 uses a Poisson model (20 Mbps); AP 3 simulates a latency-sensitive VR streaming application (80 Mbps, 90 fps); and AP 4 generates bursty traffic (40 Mbps). In this scenario, there is no clearly optimal static channel assignment, as the airtime demands and transmission durations differ significantly across APs. This scenario evaluates the learning algorithms' ability to learn a suitable configuration that outperforms legacy static configurations in realistic and non-trivial scenarios.

Scenario C (Tbl. D.3) involves four equally-loaded BSSs using a Poisson traffic model (50 Mbps each), deployed in a larger physical area. This scenario introduces runtime environmental changes to evaluate the adaptability of the learning agents. Initially, channel $\{3\}$ is interference-free. However, at the simulation midpoint, AP from BSS 2 changes its channel assignment, moving from $\{1\}$ to $\{3\}$. These change substantially alters the interference landscape, significantly reducing contention on channel $\{1\}$ and introducing interference in the previously free channel. This scenario assesses the learning AP's ability to not only learn effective initial configuration but also to adapt its strategy in response to environmental changes, demonstrating responsiveness to the non-stationary and evolving nature of real-world WNs.

As a benchmark, non-learning baselines are included for each scenario. In these baselines, the AP in BSS 1 does not perform any learning and instead adopts an static configuration, as the other APs. In each scenario, multiple simulation runs are conducted for each possible static channel group allocation for AP 1, thereby enabling a direct comparison between CMAB-driven optimization and conventional IEEE 802.11 operation.

2.2 Results

This section presents the results obtained for the learning-enabled BSS (i.e., BSS 1) across the three toy scenarios, comparing the performance of each algorithm–architecture combination (ϵ-greedy RMSProp-based and SW-LinUCB, in single-agent and multi-agent architectures) against static non-learning baselines.
Performance is assessed using the negative transmission cycle duration—hereinafter referred to simply as *reward*, regardless of whether the BSS has learning capabilities, as it is equivalent to the raw (non-normalized) joint reward signal of learning agents—alongside goodput[1], and airtime utilization[2] across all toy scenarios. In addition, environmental impact is evaluated using CodeCarbon's energy consumption and carbon emission estimates.

Scenario A

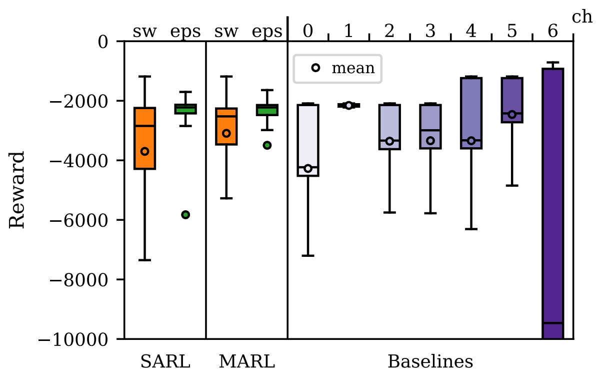
As illustrated in Fig. 4.2b, the highest goodput for BSS 1 is achieved using static channel configurations: channel $\{2\}$ ('ch' index 1) achieves 209.3 Mbps and, surprisingly, channel group $\{3, 4\}$ ('ch' index 5) yields 210.3 Mbps. Notably, Fig. 4.2a indicates that channel 2 exhibits a narrower Interquartile Range (IQR) in reward, suggesting more consistent transmission cycle delays as it is free from external interference. In contrast, $\{3, 4\}$ benefits from increased bandwidth (40 MHz), enabling faster transmission rates and, thus, minimal transmission cycles and airtime (Fig. C.1a). Nevertheless, this configuration experiences higher delay variability due to increased contention, as channel 3 is also allocated to BSS 2. Indeed, as seen in Fig. C.1b, this channel reuse reduces BSS 2's goodput but, as shown in Tbl. D.4, results in enhanced goodput fairness across BSSs. Conversely, other static configurations—such as $\{1, 2, 3, 4\}$ ('ch' index 6)—significantly degrade performance and fairness, underscoring the detrimental impact

[1]Goodput is the amount of application-level data successfully received at the destination per unit of time, measured in bits per second (bps). Here, instantaneous goodput is sampled at the receiver (STA 1) upon reception of each data frame. Thus, the average is computed as a time-weighted mean rather than a simple arithmetical average due to the non-uniform sampling interval.

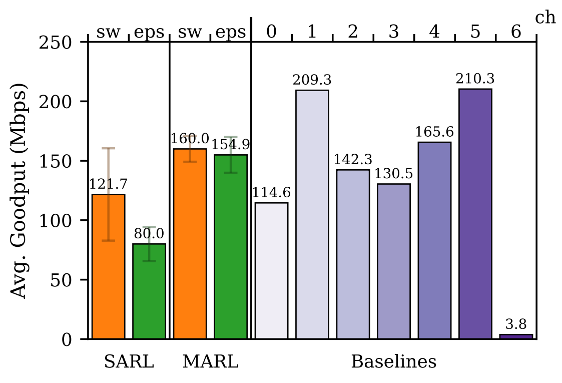
[2]Airtime utilization measures the proportion of time the wireless medium is actively used for transmission by an AP.

of misconfiguration and the importance of avoiding inter-BSS interference.

On the other hand, under SA-CMABs, SW-LinUCB achieves 121.7 Mbps, while the proposed ϵ-greedy RMSProp-based agent achieves 80.0 Mbps on average. The underperformance of our method in this setting can be attributable to its small learning rate, as it slows adaptation, and its uniform random exploration strategy that hinders selecting optimal joint actions early. Indeed, as illustrated in Fig. C.4b, the agent exhibits minimal action diversity and thereby its performance becomes highly-dependent on its initial conditions. Thus, in the single-agent setting, our algorithm may require longer executions to sufficiently explore the action space and find high-reward configurations.
Conversely, under MA-CMABs, both learning algorithms significantly improve, achieving comparable performance: SW-LinUCB reaches 160.0 Mbps and our method 154.9 Mbps. Moreover, both algorithms yield enhanced goodput fairness compared to the single-agent setting, as shown in Tbl. D.4. Fig. C.4 confirms that both algorithms frequently select near-optimal configurations, i.e., $(1, \cdot, \cdot)$ or $(5, \cdot, \cdot)$, especially in the multi-agent architecture. SW-LinUCB maintains greater action diversity—likely due to its dynamic confidence bounds—while our ϵ-greedy algorithm—using a small ϵ—tends to stick with high-reward configurations once found. This results in consistently high rewards and small reward variability, as illustrated in Fig. 4.2a. Nevertheless, occasional low-reward outliers, resulting from sporadic selection of poor actions, reduce its average reward. This is likely due to its uniform random exploration of actions, which leads to frequent selection of suboptimal configurations such as large CWs or wide channel groups prone to interference.



(a)



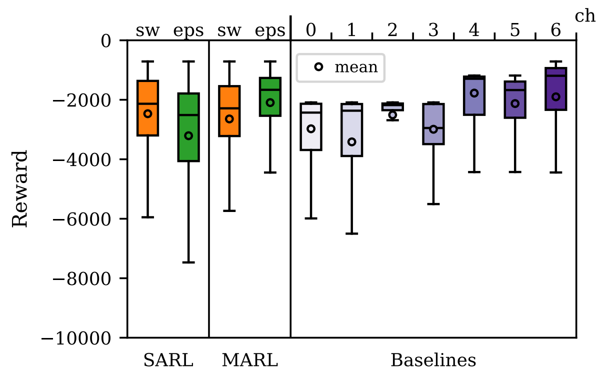
(b)

Figure 4.2: Reward and goodput for BSS 1 in Scenario A for each algorithm–architecture pair and the static non-learning baseline. Results are averaged over five independent trials; goodput bars and error bars represent the mean and standard deviation across trials, respectively. The reward boxplot reflects the distribution of all reward values collected across the five trials. ch: channel group index, i.e., $0 : \{1\}, 1 : \{2\}, 2 : \{3\}, 3 : \{4\}, 4 : \{1, 2\}, 5 : \{3, 4\}, 6 : \{1, 2, 3, 4\}$

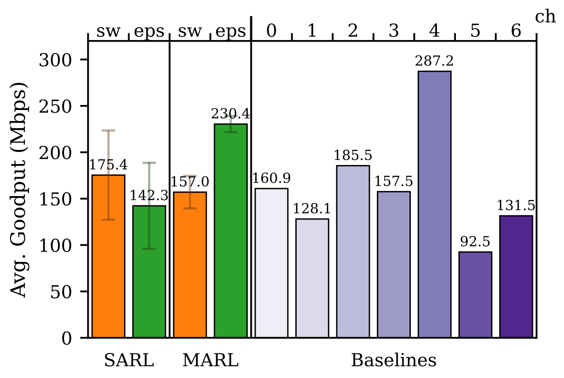
Scenario B

As illustrated in Fig. 4.3b, in this scenario the highest goodput for BSS 1 is again achieved using an static configuration: $\{1, 2\}$ ('ch' index 4) reaches 287.2 Mbps. Nevertheless, this comes at the expense of severe degradation in BSS 3, as shown in Fig. C.2b, since both BSSs operate over channels 1 and 2. Thus, despite BSS 3 demanding higher throughput than other neighboring BSSs, using the broader channel allocation $\{1, 2\}$ offers increased performance for BSS 1, as it provides a higher bandwidth and there is a single contender. Indeed, other static configurations, such as $\{3, 4\}$ and $\{1, 2, 3, 4\}$, perform significantly worse due to inefficient spectral reuse.

Among learning-based approaches, our proposed algorithm, in the MA-CMAB setting, achieves the second-best performance: an average goodput of 230.4 Mbps. As seen in Fig. C.5b, the channel selection agent consistently converges to joint action $(4, \cdot, \cdot)$, demonstrating successful learning. In contrast, under the SA-CMAB setting, our algorithm occasionally selects the optimal action but fails to do so consistently across trials, leading to reduced goodput (142.3 Mbps) and broader error bars in Fig. 4.3b. Again, as seen in Fig. C.5b, the agent exhibits minimal action diversity. Conversely, SW-LinUCB achieves higher goodput in the single-agent setting (175.4 Mbps) but under the multi-agent architecture its performance drops significantly (157.0 Mbps) compared to our algorithm. This decline leads to improved fairness (see Tbl. D.4), although no significant enhancements occur in the goodput of other BSSs (see Fig. C.2b), indicating that the fairness gains primarily result from a performance decrease in the learning BSS rather than actual network-wide improvements. Indeed, since each BSS experiences a distinct traffic load, fairness metrics generally do not accurately reflect equitable resource distribution. Additionally, as shown in Fig. C.5b, SW-LinUCB exhibits high action diversity and does not converge towards a single channel group configuration.



(a)



(b)

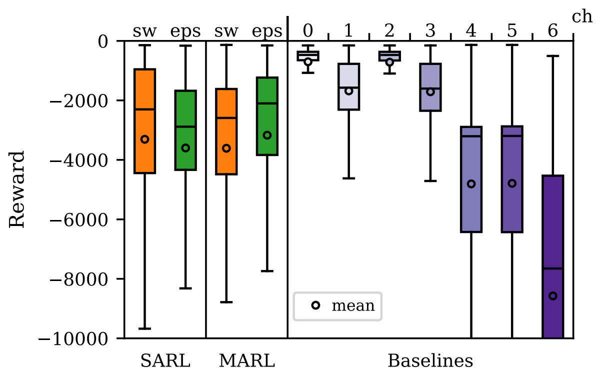
Figure 4.3: Reward and goodput for BSS 1 in Scenario B for each algorithm–architecture pair and the static non-learning baseline.

Scenario C

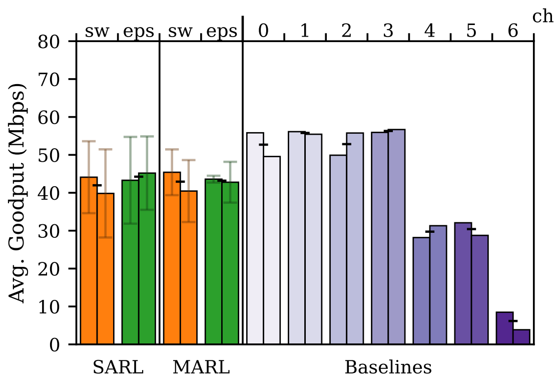
Fig. 4.4b shows that static configurations using 20 MHz channels (i.e., ‘ch’ indices 0 to 3) yield the highest goodput among non-learning baselines, as they offer sufficient bandwidth to support the 50 Mbps traffic load without causing excessive contention or co-channel interference. In contrast, wider channels—such as 40 MHz (‘ch’ indices 4 and 5) and 80 MHz (‘ch’ index 6)—perform significantly worse due to their greater susceptibility to interference. In particular, under the Static Channel Bonding (SCB) policy, activity on any secondary channel in the bond defers the entire transmission. This leads to reduced goodput fairness, as detailed in Tbl. D.4.

On the other hand, both learning-based algorithms achieve moderately high and comparable goodput, regardless of the underlying architecture. Notably, performance remains consistent across both simulation halves, suggesting stable behavior and effective adaptability even with the mid-simulation shift in available frequencies.

Nevertheless, the fact that the 20 MHz static baselines exhibit similar performance in both halves indicates that the environmental change had limited impact on goodput—likely due to the relatively low data rate, which may mask potential learning gains. Still, the learning agents demonstrate robustness by avoiding poor decisions that could degrade performance. Furthermore, as illustrated in Fig. C.6b, our algorithm in the multi-agent setting consistently selected idle channels in each half of the simulation: channel $\{3\}$ (70.9%) and channel group $\{3, 4\}$ (20.1%) in the first half, and channel $\{1\}$ (53.6%) and channel group $\{1, 2\}$ (33.1%) in the second half. This behavior demonstrates the algorithm’s ability to effectively adapt to the introduced changes in spectrum availability.



(a)



(b)

Figure 4.4: Reward and goodput for BSS 1 in Scenario C for each algorithm–architecture pair and the static non-learning baseline. For goodput, each bar pair represent the average across trials for the first and second halves of the simulation, respectively.

Table 4.3: Average estimated energy consumption (kWh) using CodeCarbon across Scenarios A, B, and C.

	SW-LinUCB			RMSProp-based		
Single-agent	6.9e-03	9.6e-03	1.5e-02	4.9e-03	8.2e-03	1.2e-02
Multi-agent	5.8e-02	7.2e-02	7.9e-02	5.4e-02	8.8e-02	8.6e-02
	A	B	C	A	B	C

Table 4.4: Average estimated CO_2 emissions (kg) using CodeCarbon across Scenarios A, B, and C.

	SW-LinUCB			RMSProp-based		
Single-agent	1.2e-03	1.7e-03	2.6e-03	8.6e-04	1.4e-03	2.1e-03
Multi-agent	1.0e-02	1.3e-02	1.4e-02	9.4e-03	1.5e-02	1.5e-02
	A	B	C	A	B	C

Environmental impact

Tables 4.3 and 4.4 summarize the average energy consumption (in kWh) and the corresponding equivalent CO_2 emissions (in kg) estimated using CodeCarbon across Scenarios A, B, and C. The results reveal that multi-agent approaches consistently consume more energy than single-agent settings, and consequently lead to higher CO_2 emissions. On the other hand, comparing both algorithms, the proposed ϵ-greedy RMSProp-based generally results in lower energy consumption and emissions than SW-LinUCB, especially in the single-agent setting. Indeed, even though SW-LinUCB shows lower total energy consumption in the multi-agent setting for Scenarios B and C, our proposed algorithm exhibits lower energy consumption per decision, underscoring its superior computational efficiency (see Section 4): $2.2 \cdot 10^{-6}$ kWh against $6.4 \cdot 10^{-6}$ kWh in Scenario B, and $4.1 \cdot 10^{-6}$ kWh against $4.3 \cdot 10^{-6}$ kWh in Scenario C.

Summary

The results suggest that parameter configuration via online learning—particularly through cooperative MA-CMABs—provides a practical and effective alternative to legacy static operation, offering adaptability to environmental conditions and avoiding the adverse effects of misconfiguration.

Notably, across the evaluation scenarios, MA-CMABs consistently deliver higher performance than SA-CMABs. Nevertheless, from a sustainability perspective, the results indicate that CMAB-based single-agent architectures exhibit significantly lower consumption and reduced environmental impact, despite their increased action space.

On the other hand, the results show that SW-LinUCB significantly outperforms our proposed algorithm in the single-agent architecture. Conversely, our algorithm outperforms SW-LinUCB in the multi-agent setting—achieving superior goodput in Scenario B and comparable performance in Scenario A and C. Furthermore, in terms of sustainability, our algorithm demonstrates lower energy consumption and CO_2 emissions, particularly due to its computational efficiency. Nevertheless, our algorithm exhibits occasional extreme transmission delays due to its uniform random exploration mechanism, leading to high temporal variability in goodput, as also noted for ϵ-greedy-based strategies in [115]. Indeed, even in the non-densely populated scenarios considered, the selection of particularly large CW sizes—such as 1024—consistently degrades performance, prompting reconsideration of the action space and further refinement of the exploration-exploitation strategy—particularly in the single-agent case.

Furthermore, the results demonstrate that certain channel configurations may increase an AP's performance at the expense of neighboring peers, highlighting the consequences of selfish BSS decisions and the need for coordination across APs to enable fairness.

These results highlight that no single configuration is universally optimal and that the most effective strategy depends on deployment priorities and the desired trade-off between network performance and environmental sustainability.

Chapter 5

Conclusions and future work

1 Conclusions

In this thesis, we developed an event-driven WN simulator in Python, in which we implemented both SA-CMAB and MA-CMAB architectures for dynamically optimizing Medium Access Control (MAC)-layer parameters—including the allocated channel group, primary channel, and CW size—at a single learning AP, based on contextual information from both the AP itself and its surrounding wireless environment, and using transmission cycle delay as a learning reward signal. In particular, we leveraged a state-of-the-art contextual bandit algorithm, SW-LinUCB, and proposed a novel, simpler linear contextual bandit method that combines ϵ-greedy exploration with an RMSProp update rule. Both algorithms were implemented and evaluated under the aforementioned architectures and across three illustrative scenarios involving a single independent learning AP operating in a network of non-learning peers, after appropriate hyperparameter tuning.

Our results demonstrate that cooperative MA-CMABs architectures can achieve superior transmission performance compared to SA-CMABs settings, albeit at the cost of increased energy consumption—thereby confirming the hypothesis stated in Section 1 and underscoring the fundamental trade-off between performance and environmental impact. Furthermore, our findings demonstrate that, in the multi-agent setting, the proposed ϵ-greedy RMSProp-based algorithm achieves comparable or even superior performance relative to SW-LinUCB, while also exhibiting lower energy consumption and CO_2 emissions—given its computational efficiency. Therefore, in terms of performance, the best outcomes were obtained using our proposed algorithm in a cooperative MA-CMAB architecture.

Nevertheless, in the single-agent setting, our algorithm underperformed—likely due to the small ϵ values and low learning rates used during training, which may have hindered timely convergence to high-reward configurations. Additionally, our results indicate that ϵ-greedy can occasionally lead to extreme transmission delays, highlighting the need for adaptive or smarter exploration strategies to further refine the algorithm, especially in single-agent deployments.

Thus, this research effectively addressed the central research question stated in Section 3, demonstrating the potential of cooperative MA-CMABs for optimizing MAC-layer functionalities in WNs. The thesis also fulfilled all the objectives defined in Section 3: (*i*) an open-source, event-driven WN simulator has been implemented in Python, providing a flexible platform for testing learning-based optimization algorithms tailored for IEEE 802.11 networks; (*ii*) both single-agent and cooperative multi-agent CMAB-based agents—fine-tuned using Optuna—capable of adjusting key MAC parameters in real-time have been incorporated; and (*iii*) both architectures have been compared in terms of several performance indicators, including goodput (a.k.a. application-level throughput), transmission cycle delay (indicative of latency), and power consumption (reflecting computational complexity), as well as against static legacy configurations.

2 Future work

This work demonstrates the potential of CMAB-based approaches for online wireless channel access optimization in decentralized settings, opening up several promising research directions. As part of my ongoing research within the UPF's Wireless Networking research group, the directions outlined below will be investigated in future work.

Extended action and context spaces
The design of the action space may require reevaluation and refinement, given the observed performance degradation associated with overly large CW sizes. Beyond the current configuration options (channel group, primary channel, and CW size selection), specialized agents can be incorporated to control additional MAC and PHY parameters, such as enabling or disabling A-MPDU frame aggregation and RTS/CTS protection, or selecting the MCS and transmission power.
On the contextual space, our aim is to conduct ablation studies to assess the individual contribution of each contextual feature to learning effectiveness. Also, our goal is to explore alternative or complementary context variables—such as signal quality indicators or interference levels—to enhance observability and enable more accurate policy adaptation.

Reward design and shaping

Given the cooperative nature of our multi-agent architecture, the current reward signal—based on total transmission cycle delay—can be decomposed into weighted components (e.g., sensing, backoff, data transmission, and residual delays). These weights can be adjusted to reflect the relative influence of each agent on the corresponding delay components. This decomposition enables reward shaping [157], potentially enhancing credit attribution and learning effectiveness. In contrast, the single agent architecture does not require such mechanisms, as credit assignment is not a concern.

Furthermore, alternative reward formulations can be explored. For instance, the reward can be scaled by the number of successfully transmitted packets to encourage agents to maximize effective data delivery. Similarly, as proposed in recent studies, the reward function can be designed to maximize throughput or efficiency (e.g., in [16,97,107,158]) and/or minimize airtime (e.g., in [16, 158]), discouraging greediness and promoting fairness.

Dynamic, multi-learner environments

Future evaluations will consider more dynamic and complex wireless network environments, including user mobility, uplink traffic, and multiple clients per BSS—features already supported by our simulator. These scenarios could further highlight the benefits of learning-based adaptation over static configurations.

In addition, we will investigate scenarios involving multiple learning-enabled APs, which, as discussed in Section 2.1, introduce greater complexity in learning dynamics.

Framework refinements

Isolated evaluations under highly dense network scenarios highlighted the need to refine the current agent operation strategy. Currently, agents operate on a per transmission cycle basis: once a channel group is selected, the decision is sustained until the transmission either succeeds or times out. However, in densely populated scenarios, a transmission may be significantly delayed if the channel is never sensed idle during a DIFS period, leading to prolonged access delays. Hence, a more effective strategy would involve setting a transmission cycle timeout. This mechanism would allow the agent to re-evaluate and reselect channel configurations if it fails to gain access within a reasonable timeframe. Moreover, this approach would naturally define the lower bound for reward normalization, eliminating the need to arbitrarily assign a minimum reward value and perform clipping.

On the other hand, our proposed algorithm—namely, the ϵ-greedy RMSProp-based algorithm—can be further refined to enhance learning stability and efficiency, especially in the single-agent setting. Since the standard ϵ-greedy strategy explores all actions uniformly, including those

that consistently yield poor rewards, more selective strategies could be adopted. For instance, variants such as ϵ-greedy with Stickiness [108,109], or alternative approaches such as Thompson Sampling (TS) [34] or Boltzmann exploration [159], may offer superior performance.
Additionally, given that in the multi-agent architecture agents can operate at distinct intervals, a higher-level meta-controller can be incorporated to govern their activation schedule, enabling a hierarchical learning framework. This temporal decoupling may improve learning stability and reduce computational overhead, as it reduces decision frequency.

Beyond contextual bandits

Our aim is to also investigate full RL approaches, as these can capture the long-term consequences of agents' actions. To this end, our plan is to integrate PettingZoo [160]—a Python library for conducting research in MARL, analogous to a multi-agent version of Gymnasium [161]—into our simulator and implement state-of-the-art algorithms using libraries such as MARLlib [162], comparing their performance against our current CMAB-based framework. In particular, our interest lies in Centralized Training and Decentralized Execution (CTDE) approaches, such as QMIX [83]. As discussed in Section 2.4, CTDE enables agents to share information during training in order to learn coordinated strategies, while maintaining decentralized operation at runtime—making it especially suitable for scenarios involving multiple learning-enabled APs. Nevertheless, CTDE-based strategies also present limitations, as detailed in Section 2.2.

Simulator enhancements

Lastly, our plan is to enhance the simulator's fidelity by modeling interference and signal propagation effects (e.g., shadowing, fading) and supporting Enhanced Distributed Channel Access (EDCA) to enable traffic prioritization.

Bibliography

[1] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[2] Stefano V Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

[3] Robson Costa, Paulo Portugal, Francisco Vasques, Carlos Montez, and Ricardo Moraes. Limitations of the IEEE 802.11 DCF, PCF, EDCA and HCCA to handle real-time traffic. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 931–936. IEEE, 2015.

[4] Ali Balador, Mahtab Ghasemivand, Ali Movaghar, and S Jabbehdar. An adaptive contention window control for improving DCF throughput and fairness. *European J. Scientific Research*, 45(2):310–323, 2010.

[5] Sergio Barrachina-Muñoz, Francesc Wilhelmi, and Boris Bellalta. To overlap or not to overlap: Enabling channel bonding in high-density WLANs. *Computer Networks*, 152:40–53, 2019.

[6] Sergio Barrachina-Munoz, Francesc Wilhelmi, and Boris Bellalta. Dynamic channel bonding in spatially distributed high-density WLANs. *IEEE Transactions on Mobile Computing*, 19(4):821–835, 2019.

[7] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 21(1), 1984.

[8] Alessio Zappone, Marco Di Renzo, and Mérouane Debbah. Wireless networks design in the era of deep learning: Model-based, AI-based, or both? *IEEE Transactions on Communications*, 67(10):7331–7376, 2019.

[9] Navid Naderializadeh, Jaroslaw J Sydir, Meryem Simsek, and Hosein Nikopour. Resource management in wireless networks via multi-agent deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 20(6):3507–3523, 2021.

[10] Szymon Szott, Katarzyna Kosek-Szott, Piotr Gawłowicz, Jorge Torres Gómez, Boris Bellalta, Anatolij Zubow, and Falko Dressler. Wi-Fi meets ML: A survey on improving IEEE 802.11 performance with machine learning. *IEEE Communications Surveys & Tutorials*, 24(3):1843–1893, 2022.

[11] Hannaneh Barahouei Pasandi and Tamer Nadeem. Towards a learning-based framework for self-driving design of networking protocols. *IEEE Access*, 9:34829–34844, 2021.

[12] Jianbin Xiao, Zhenyu Chen, Xinghua Sun, Wen Zhan, Xijun Wang, and Xiang Chen. Online multi-agent reinforcement learning for multiple access in wireless networks. *IEEE Communications Letters*, 27(12):3250–3254, 2023.

[13] Luis Esteve Elfau. Introducción al aprendizaje por refuerzo. *Fundació Universitat Oberta de Catalunya (FUOC)*, september 2021.

[14] Sergio Barrachina-Muñoz, Alessandro Chiumento, and Boris Bellalta. Stateless reinforcement learning for multi-agent systems: The case of spectrum allocation in dynamic channel bonding WLANs. In *2021 Wireless Days (WD)*, pages 1–5. IEEE, 2021.

[15] Zhichao Zheng, Shengming Jiang, Ruoyu Feng, Lige Ge, and Chongchong Gu. Survey of reinforcement-learning-based MAC protocols for wireless ad hoc networks with a MAC reference model. *Entropy*, 25(1):101, 2023.

[16] Navid Keshtiarast, Oliver Renaldi, and Marina Petrova. Wireless MAC Protocol Synthesis and Optimization with Multi-Agent Distributed Reinforcement Learning. *IEEE Networking Letters*, 2024.

[17] Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11):4948, 2021.

[18] Alvaro Valcarce and Jakob Hoydis. Toward joint learning of optimal MAC signaling and wireless channel access. *IEEE Transactions on Cognitive Communications and Networking*, 7(4):1233–1243, 2021.

[19] Donghwan Lee, Niao He, Parameswaran Kamalaruban, and Volkan Cevher. Optimization for reinforcement learning: From a single agent to cooperative agents. *IEEE Signal Processing Magazine*, 37(3):123–135, 2020.

[20] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz De Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

[21] United Nations. Sustainable Development Goals. `https://www.un.org/sustainabledevelopment/`. Accessed: May 25, 2025.

[22] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[23] Jordi Casas Roma and Julià Minguillón Alfonso. Métodos supervisados. *Fundació Universitat Oberta de Catalunya (FUOC)*, september 2024.

[24] Jordi Casas Roma. Métodos no supervisados. *Fundació Universitat Oberta de Catalunya (FUOC)*, september 2024.

[25] David Silver. Lectures on Reinforcement Learning. `https://www.davidsilver.uk/teaching/`, 2015. Accessed: May 25, 2025.

[26] OpenAI Spinning Up. User Documentation. Introduction to rl. part 2: Kinds of rl algorithms. `https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html`. Accessed: May 25, 2025.

[27] Luis Esteve Elfau. Procesos de decisión de Markov. *Fundació Universitat Oberta de Catalunya (FUOC)*, september 2021.

[28] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

[29] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[30] Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

[31] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE congress on evolutionary computation (CEC)*, pages 1–8. IEEE, 2020.

[32] Ashwin Rao. Multi-Armed Bandits: Exploration versus Exploitation. `https://web.stanford.edu/class/cme241/lecture_slides/MultiArmedBandits.pdf`, 2021. Accessed: May 25, 2025.

[33] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[34] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[35] A. J. Ganesh. Multi-armed bandits. https://people.maths.bris.ac.uk/~maajg/teaching/stochopt/ucb.pdf, October 2019. Accessed: May 25, 2025.

[36] Andrew G Barto, Richard S Sutton, and Peter S Brouwer. Associative search network: A reinforcement learning associative memory. *Biological cybernetics*, 40:201–211, 1981.

[37] David Cortes. Adapting multi-armed bandits policies to contextual bandits scenarios. *arXiv preprint arXiv:1811.04383*, 2018.

[38] Medium. Towards AI. Mastering Contextual Bandits: Personalization and Decision Making in Real Time. https://pub.towardsai.net/mastering-contextual-bandits-personalization-and-decision-making-in-real-time-41b4d3d3bae6#1a81, 2024. Accessed: May 25, 2025.

[39] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. *Advances in neural information processing systems*, 20, 2007.

[40] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[41] Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

[42] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR, 2013.

[43] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[44] Li Zhou. A survey on contextual multi-armed bandits. *arXiv preprint arXiv:1508.03326*, 2015.

[45] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[47] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[48] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[49] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[50] Paolo Dini. Métodos de diferencia temporal. *Fundació Universitat Oberta de Catalunya (FUOC)*, september 2021.

[51] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[52] Gerald Tesauro et al. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[53] Hugging Face. Introducing Q-Learning. `https://huggingface.co/learn/deep-rl-course/unit2/q-learning`. Accessed: May 25, 2025.

[54] TensorFlow. Introduction to RL and Deep Q Networks. `https://www.tensorflow.org/agents/tutorials/0_intro_rl`, 2023. Accessed: May 25, 2025.

[55] Laura Ruiz Dern. Deep Q-networks. *Fundació Universitat Oberta de Catalunya (FUOC)*, september 2021.

[56] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[57] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[58] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[59] Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *ICML*, pages 89–94, 2000.

[60] Dom Huh and Prasant Mohapatra. Multi-agent reinforcement learning: A comprehensive survey. *arXiv preprint arXiv:2312.10256*, 2023.

[61] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

[62] Lukas M Schmidt, Johanna Brosig, Axel Plinge, Bjoern M Eskofier, and Christopher Mutschler. An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1342–1349. IEEE, 2022.

[63] Justin K Terry, Nathaniel Grammel, Sanghyun Son, Benjamin Black, and Aakriti Agrawal. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

[64] Lucian Busoniu, Robert Babuska, and Bart De Schutter. Multi-agent reinforcement learning: A survey. In *2006 9th international conference on control, automation, robotics and vision*, pages 1–6. IEEE, 2006.

[65] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

[66] Soummya Kar, José MF Moura, and H Vincent Poor. Distributed reinforcement learning in multi-agent networks. In *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 296–299. IEEE, 2013.

[67] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multiagent coordination. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1859–1864. IEEE, 2005.

[68] John Von Neumann and Oskar Morgenstern. Theory of games and economic behavior, 2nd rev. 1947.

[69] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

[70] Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2409.03052*, 2024.

[71] Ken Ming Lee, Sriram Ganapathi Subramanian, and Mark Crowley. Investigation of independent reinforcement learning algorithms in multi-agent environments. *Frontiers in Artificial Intelligence*, 5:805823, 2022.

[72] Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024.

[73] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.

[74] Jiaxin Liang, Haotian Miao, Kai Li, Jianheng Tan, Xi Wang, Rui Luo, and Yueqiu Jiang. A Review of Multi-Agent Reinforcement Learning Algorithms. *Electronics*, 14(4):820, 2025.

[75] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

[76] Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning.* University of Massachusetts Amherst, 1984.

[77] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

[78] Aditya Kapoor, Sushant Swamy, Kale-ab Tessera, Mayank Baranwal, Mingfei Sun, Harshad Khadilkar, and Stefano V Albrecht. Agent-Temporal Credit Assignment for Optimal Policy Preservation in Sparse Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2412.14779*, 2024.

[79] Logan Yliniemi and Kagan Tumer. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In *Asia-Pacific conference on simulated evolution and learning*, pages 407–418. Springer, 2014.

[80] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:11853–11864, 2020.

[81] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[82] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[83] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[84] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11:387–434, 2005.

[85] Guannan Qu, Yiheng Lin, Adam Wierman, and Na Li. Scalable multi-agent reinforcement learning for networked systems with average reward. *Advances in Neural Information Processing Systems*, 33:2074–2086, 2020.

[86] Dapeng Li, Na Lou, Bin Zhang, Zhiwei Xu, and Guoliang Fan. Adaptive parameter sharing for multi-agent reinforcement learning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6035–6039. IEEE, 2024.

[87] Yen-Wen Chen and Kuo-Che Kao. Study of contention window adjustment for csma/ca by using machine learning. In *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 206–209. IEEE, 2021.

[88] Tae-Wook Kim and Gyung-Ho Hwang. Performance enhancement of csma/ca mac protocol based on reinforcement learning. *Journal of information and communication convergence engineering*, 19(1):1–7, 2021.

[89] Witold Wydmański and Szymon Szott. Contention window optimization in IEEE 802.11 ax networks with deep reinforcement learning. In *2021 IEEE wireless communications and networking conference (WCNC)*, pages 1–6. IEEE, 2021.

[90] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[91] Abhishek Kumar, Gunjan Verma, Chirag Rao, Ananthram Swami, and Santiago Segarra. Adaptive contention window design using deep Q-learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4950–4954. IEEE, 2021.

[92] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[93] Rashid Ali, Nurullah Shahin, Yousaf Bin Zikria, Byung-Seo Kim, and Sung Won Kim. Deep reinforcement learning paradigm for performance optimization of channel observation–based MAC protocols in dense WLANs. *IEEE Access*, 7:3500–3511, 2018.

[94] R Ali, N Shahin, Y-T Kim, B-S Kim, and SW Kim. Channel observation-based scaled backoff mechanism for high-efficiency WLANs. *Electronics Letters*, 54(10):663–665, 2018.

[95] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

[96] Ziyang Guo, Zhenyu Chen, Peng Liu, Jianjun Luo, Xun Yang, and Xinghua Sun. Multi-agent reinforcement learning-based distributed channel access for next generation wireless networks. *IEEE Journal on Selected Areas in Communications*, 40(5):1587–1599, 2022.

[97] Muhammad Sohaib, Jongjin Jeong, and Sang-Woon Jeon. Dynamic multichannel access via multi-agent reinforcement learning: Throughput and fairness guarantees. *IEEE Transactions on Wireless Communications*, 21(6):3994–4008, 2021.

[98] Hang Qi, Hao Huang, Zhiqun Hu, Xiangming Wen, and Zhaoming Lu. On-demand channel bonding in heterogeneous WLANs: A multi-agent deep reinforcement learning approach. *Sensors*, 20(10):2789, 2020.

[99] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[100] Luciano Miuccio, Salvatore Riolo, Mehdi Bennis, and Daniela Panno. Design of a Feasible Wireless MAC Communication Protocol via Multi-Agent Reinforcement Learning. In *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, pages 94–100. IEEE, 2024.

[101] Luciano Miuccio, Salvatore Riolo, Sumudu Samarakoon, Mehdi Bennis, and Daniela Panno. On learning generalized wireless MAC communication protocols via a feasible multi-agent reinforcement learning framework. *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[102] Faisal Naeem, Nadir Adam, Georges Kaddoum, and Omer Waqar. Learning MAC protocols in HetNets: A cooperative multi-agent deep reinforcement learning approach. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2024.

[103] Hrishikesh Dutta and Subir Biswas. Towards multi-agent reinforcement learning for wireless network protocol synthesis. In *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 614–622. IEEE, 2021.

[104] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69. IEEE, 2007.

[105] Mateus P Mota, Alvaro Valcarce, Jean-Marie Gorce, and Jakob Hoydis. The emergence of wireless MAC protocols with multi-agent reinforcement learning. In *2021 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2021.

[106] Sangwoo Moon, Sumyeong Ahn, Kyunghwan Son, Jinwoo Park, and Yung Yi. Neuro-DCF: Design of wireless MAC via multi-agent reinforcement learning approach. In *Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pages 141–150, 2021.

[107] Kamil Szczech, Maksymilian Wojnar, Katarzyna Kosek-Szott, Krzysztof Rusek, Szymon Szott, Dileepa Marasinghe, Nandana Rajatheva, Richard Combes, Francesc Wilhelmi, Anders Jonsson, et al. Towards Specialized Wireless Networks Using an ML-Driven Radio Interface. *arXiv preprint arXiv:2502.20996*, 2025.

[108] Marc Carrascosa and Boris Bellalta. Decentralized AP selection using multi-armed bandits: Opportunistic ε-greedy with stickiness. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2019.

[109] Marc Carrascosa and Boris Bellalta. Multi-armed bandits for decentralized AP selection in enterprise WLANs. *Computer Communications*, 159:108–123, 2020.

[110] Álvaro López-Raventós and Boris Bellalta. Concurrent decentralized channel allocation and access point selection using multi-armed bandits in multi BSS WLANs. *Computer Networks*, 180:107381, 2020.

[111] Maksymilian Wojnar, Wojciech Ciezobka, Katarzyna Kosek-Szott, Krzysztof Rusek, Szymon Szott, David Nunez, and Boris Bellalta. IEEE 802.11 bn Multi-AP Coordinated Spatial Reuse with Hierarchical Multi-Armed Bandits. *IEEE Communications Letters*, 2024.

[112] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[113] Lluís Martínez, Josep Vidal, and Margarita Cabrera-Bean. Contextual Multi-Armed Bandits for Non-Stationary Wireless Network Selection. In *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pages 285–290. IEEE, 2023.

[114] Setareh Maghsudi and Sławomir Stańczak. Joint channel selection and power control in infrastructureless wireless networks: A multiplayer multiarmed bandit framework. *IEEE Transactions on Vehicular Technology*, 64(10):4565–4578, 2014.

[115] Francesc Wilhelmi, Cristina Cano, Gergely Neu, Boris Bellalta, Anders Jonsson, and Sergio Barrachina-Muñoz. Collaborative spatial reuse in wireless networks via selfish multi-armed bandits. *Ad Hoc Networks*, 88:129–141, 2019.

[116] Sergio Barrachina-Muñoz, Alessandro Chiumento, and Boris Bellalta. Multi-armed bandits for spectrum allocation in multi-agent channel bonding WLANs. *IEEE Access*, 9:133472–133490, 2021.

[117] Francesc Wilhelmi, Boris Bellalta, Szymon Szott, Katarzyna Kosek-Szott, and Sergio Barrachina-Muñoz. Coordinated Multi-Armed Bandits for Improved Spatial Reuse in Wi-Fi. *arXiv preprint arXiv:2412.03076*, 2024.

[118] Pedro Enrique Iturria-Rivera, Marcel Chenier, Bernard Herscovici, Burak Kantarci, and Melike Erol-Kantarci. Cooperate or not cooperate: Transfer learning with multi-armed

bandit for spatial reuse in Wi-Fi. *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[119] Rong Zhu and Mattia Rigotti. Deep bandits show-off: Simple and efficient exploration with deep networks. *Advances in Neural Information Processing Systems*, 34:17592–17603, 2021.

[120] Cecylia Borek. Master's Thesis: An IEEE 802.11 Channel Access Simulator based on the Python NumPy Library. https://github.com/cecyliaborek/DCF-NumPy-simulation, 2021. Accessed: May 25, 2025.

[121] Paweł Topór. Master's Thesis: A Python-based Simulator of Channel Access in IEEE 802.11 Networks using Simpy library. https://github.com/ToporPawel/DCF-Simpy, 2021. Accessed: May 25, 2025.

[122] Maksymilian Wojnar, Wojciech Ciezobka, Artur Tomaszewski, Piotr Cholda, Krzysztof Rusek, Katarzyna Kosek-Szott, Jetmir Haxhibeqiri, Jeroen Hoebeke, Boris Bellalta, Anatolij Zubow, Falko Dressler, and Szymon Szott. Coordinated Spatial Reuse Scheduling With Machine Learning in IEEE 802.11 MAPC Networks. 2025.

[123] Zayan El Khaled, Wessam Ajib, and Hamid Mcheick. Log distance path loss model: Application and improvement for sub 5 GHz rural fixed wireless networks. *IEEE Access*, 10:52020–52029, 2022.

[124] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. *arXiv preprint arXiv:1910.05366*, 2019.

[125] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

[126] Lluís Martínez, Margarita Cabrera-Bean, and Josep Vidal. A Multi-Armed Bandit Model for Non-Stationary Wireless Network Selection. In *2021 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2021.

[127] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[128] Yinlong Yuan, Zhu Liang Yu, Zhenghui Gu, Xiaoyan Deng, and Yuanqing Li. A novel multi-step reinforcement learning method for solving reward hacking. *Applied Intelligence*, 49:2874–2888, 2019.

[129] Nicolas Gutowski, Tassadit Amghar, Olivier Camp, and Fabien Chhel. Global versus individual accuracy in contextual multi-armed bandit. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing*, pages 1647–1654, 2019.

[130] Yoan Russac, Claire Vernade, and Olivier Cappé. Weighted linear bandits for non-stationary environments. *Advances in Neural Information Processing Systems*, 32, 2019.

[131] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6a: Overview of mini-batch gradient descent. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012. Accessed: May 25, 2025.

[132] Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. Online stochastic gradient descent on non-convex losses from high-dimensional inference. *Journal of Machine Learning Research*, 22(106):1–51, 2021.

[133] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[134] Xi Chen, Zehua Lai, He Li, and Yichen Zhang. Online statistical inference for contextual bandits via stochastic gradient descent. *arXiv preprint arXiv:2212.14883*, 2022.

[135] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[136] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[137] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[138] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[139] Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *arXiv preprint arXiv:2411.18704*, 2024.

[140] Nicos G Pavlidis, Dimitris K Tasoulis, and David J Hand. Simulation studies of multi-armed bandits with covariates. In *Tenth international conference on computer modeling and simulation (uksim 2008)*, pages 493–498. IEEE, 2008.

[141] R Zadeh, H Li, B He, M Lublin, and Y Perez. Cme 323: Distributed algorithms and optimization, spring 2015. *University Lecture*, 2015.

[142] Qin Ding, Cho-Jui Hsieh, and James Sharpnack. An efficient algorithm for generalized linear bandit: Online stochastic gradient descent and thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 1585–1593. PMLR, 2021.

[143] Haoyu Chen, Wenbin Lu, and Rui Song. Statistical inference for online decision making via stochastic gradient descent. *Journal of the American Statistical Association*, 116(534):708–719, 2021.

[144] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[145] Qianli Liao, Kenji Kawaguchi, and Tomaso Poggio. Streaming normalization: Towards simpler and more biologically-plausible normalizations for online and recurrent learning. *arXiv preprint arXiv:1610.06160*, 2016.

[146] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[147] T Jayalakshmi and A Santhakumaran. Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1):1793–8201, 2011.

[148] Vitaliy Chiley, Ilya Sharapov, Atli Kosson, Urs Koster, Ryan Reece, Sofia Samaniego de la Fuente, Vishal Subbiah, and Michael James. Online normalization for training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[149] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 194–203, 2021.

[150] Huasen Wu, Xueying Guo, and Xin Liu. Adaptive exploration-exploitation tradeoff for opportunistic bandits. In *International Conference on Machine Learning*, pages 5306–5314. PMLR, 2018.

[151] Kaixin Sui, Mengyu Zhou, Dapeng Liu, Minghua Ma, Dan Pei, Youjian Zhao, Zimu Li, and Thomas Moscibroda. Characterizing and improving wifi latency in large-scale

operational networks. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 347–360, 2016.

[152] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[153] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305, 2012.

[154] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: a big comparison for NAS. *arXiv preprint arXiv:1912.06059*, 2019.

[155] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

[156] David Coleman. Wi-Fi 6 & 6E for Dummies, 2022.

[157] Bingling Huang and Yan Jin. Reward shaping in multiagent reinforcement learning for self-organizing systems in assembly tasks. *Advanced Engineering Informatics*, 54:101800, 2022.

[158] Mingqi Han, Xinghua Sun, Wen Zhan, Yayu Gao, and Yuan Jiang. Multi-agent reinforcement learning based uplink OFDMA for IEEE 802.11 ax networks. *IEEE Transactions on Wireless Communications*, 23(8):8868–8882, 2024.

[159] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. *Advances in neural information processing systems*, 30, 2017.

[160] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

[161] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032*, 2024.

[162] Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Xiaodan Liang, Zhihui Li, Xiaojun Chang, and Yaodong Yang. Marllib: A scalable and efficient multi-agent reinforcement learning library. *Journal of Machine Learning Research*, 24(315):1–23, 2023.

[163] IEEE 802 LAN/MAN Standards Committee et al. IEEE Standard for Information technology-Telecommunication and information exchange between systems-Local and metropolitan area networks-Specific requirements Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment1: Radio Resource Measurement of Wireless LANs. *http://standards. ieee. org/getieee802/download/802.11 n-2009. pdf*, 2009.

[164] 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 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021.

[165] Leonardo Lanante and Sumit Roy. Analysis and optimization of channel bonding in dense IEEE 802.11 WLANs. *IEEE Transactions on Wireless Communications*, 20(3):2150–2160, 2020.

[166] Mun-Suk Kim, Tanguy Ropitault, Sukyoung Lee, and Nada Golmie. A throughput study for channel bonding in IEEE 802.11 ac networks. *IEEE Communications Letters*, 21(12):2682–2685, 2017.

[167] Boris Bellalta, Alessandro Checco, Alessandro Zocca, and Jaume Barcelo. On the interactions between multiple overlapping WLANs using channel bonding. *IEEE Transactions on Vehicular Technology*, 65(2):796–812, 2015.

[168] Azadeh Faridi, Boris Bellalta, and Alessandro Checco. Analysis of dynamic channel bonding in dense networks of WLANs. *IEEE Transactions on Mobile Computing*, 16(8):2118–2131, 2016.

[169] Yi Guo, I-Tai Lu, Juan Fang, Gang Liu, and Jianhua Ge. MAC Layer Approaches for Mitigating the Spectrum Underutilization Due to Overlapping BSS Problem in WLAN. *Wireless Personal Communications*, 105:293–311, 2019.

[170] Boris Bellalta. IEEE 802.11 ax: High-efficiency WLANs. *IEEE wireless communications*, 23(1):38–46, 2016.

[171] INET Framework for OMNEST/OMNeT++. IEEE 802.11 Frame Aggregation. https://inet.omnetpp.org/docs/showcases/wireless/aggregation/doc/index.html. Accessed: May 25, 2025.

[172] Boris Bellalta et al. *Flow-level QoS guarantees in IEEE 802.11 e-EDCA based WLANs.* Citeseer, 2006.

[173] Jeremy Sharp. 802.11 Frame Types and Formats. https://howiwifi.com/2020/07/13/802-11-frame-types-and-formats/, 2020. Accessed: May 25, 2025.

[174] Wireless LAN Professionals. MCS Table and How To Use it. https://wlanprofessionals.com/mcs-table-and-how-to-use-it/. Accessed: May 25, 2025.

[175] Theodore S Rappaport. *Wireless communications: principles and practice.* Cambridge University Press, 2024.

[176] SemFio Networks. MCS Table (Update with 802.11ax data rates). https://semfionetworks.com/blog/mcs-table-updated-with-80211ax-data-rates/, 2019. Accessed: May 25, 2025.

[177] Francesc Wilhelmi, Sergio Barrachina-Muñoz, Cristina Cano, Ioannis Selinis, and Boris Bellalta. Spatial reuse in IEEE 802.11 ax WLANs. *Computer Communications*, 170:65–83, 2021.

[178] Edward J Oughton, William Lehr, Konstantinos Katsaros, Ioannis Selinis, Dean Bubley, and Julius Kusuma. Revisiting wireless internet connectivity: 5G vs Wi-Fi 6. *Telecommunications Policy*, 45(5):102127, 2021.

[179] Yang Xiao. Performance analysis of priority schemes for IEEE 802.11 and IEEE 802.11 e wireless LANs. *IEEE transactions on wireless communications*, 4(4):1506–1515, 2005.

[180] Giuseppe Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on selected areas in communications*, 18(3):535–547, 2000.

[181] TETCOS LLP. Wi-Fi: Understand the Bianchi model and compare simulation results against analysis. https://tetcos.com/pdf/WiFi-NetSim-results-vs-Bianchi-predictions.pdf. Accessed: May 25, 2025.

[182] Boris Bellalta, Alessandro Zocca, Cristina Cano, Alessandro Checco, Jaume Barcelo, and Alexey Vinel. Throughput analysis in CSMA/CA networks using continuous time Markov networks: A tutorial. *Wireless Networking for Moving Objects: Protocols, Architectures, Tools, Services and Applications*, pages 115–133, 2014.

[183] Bruno Nardelli and Edward W Knightly. Closed-form throughput expressions for CSMA networks with collisions and hidden terminals. In *2012 Proceedings IEEE INFOCOM*, pages 2309–2317. IEEE, 2012.

[184] Soung Chang Liew, Cai Hong Kai, Hang Ching Leung, and Piu Wong. Back-of-the-envelope computation of throughput distributions in CSMA wireless networks. *IEEE Transactions on Mobile Computing*, 9(9):1319–1331, 2010.

[185] Boris Bellalta, Francesc Wilhelmi, Lorenzo Galati-Giordano, and Giovanni Geraci. Performance Analysis of IEEE 802.11 bn Non-Primary Channel Access. *arXiv preprint arXiv:2504.15774*, 2025.

Appendix A

IEEE 802.11 overview, simulator design, and validation

A.1 IEEE 802.11 basics

A.1.1 IEEE 802.11 Distributed coordination function (DCF)

IEEE 802.11-based WLANs operate in shared, unlicensed frequency bands, including the 2.4 GHz, 5 GHz, and 6 GHz spectrums. Due to the limited number of non-overlapping channels[1] and the open nature of these bands, WLANs are inherently susceptible to interference, collisions, and medium contention. To coordinate access to the shared wireless medium, IEEE 802.11 defines the Distributed Coordination Function (DCF) as its fundamental Medium Access Control (MAC) protocol [163].

DCF is a contention-based access scheme that relies on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to manage access to the wireless communication medium. Before transmitting a frame, a node[2] performs a Clear Channel Assessment (CCA) to determine whether the medium is idle or busy. If the (primary) channel is sensed to be continuously idle for a Distributed Inter-frame Space (DIFS) duration, the node initiates a randomized backoff procedure governed by the Binary Exponential Backoff (BEB) algorithm to reduce the probability of simultaneous transmissions.

BEB functions as a collision avoidance mechanism by selecting uniformly at random a backoff counter from the range $\{1, \ldots, \mathrm{CW}\}$, where CW denotes the current Contention Window (CW)

[1]A *channel* in IEEE 802.11 corresponds to a fixed frequency band used for wireless communication. The number and width of available channels depend on the frequency band and regulatory constraints.

[2]Hereinafter, a node refers to any IEEE 802.11 device, including both APs and STAs.

size. This counter decrements by one for every idle time slot[3]. If the (primary) channel becomes busy before the backoff counter reaches zero, the countdown is paused and resumes only after the channel is sensed to be idle again for a complete DIFS period. Once the backoff counter reaches zero, the node initiates a data frame (or RTS frame, if the RTS/CTS mechanism is enabled) transmission.

Upon successful reception of the data frame, the receiver responds with an Acknowledgment (ACK) (or a Block Acknowledgment (BACK) in the case of aggregated frames) after a Short Inter-frame Space (SIFS) interval, confirming the successful transmission. If the ACK/BACK is not received within a predefined timeout, the sender infers that a collision has occurred and schedules a retransmission[4], updating the CW accordingly.

Initially, the CW is set to a minimum value, $\mathrm{CW}_{\min}$ $(i = 0)$. Upon a transmission failure, the node enters the next *backoff stage*, doubling the CW up to a maximum value, $\mathrm{CW}_{\max}$. The CW size at backoff stage i is thus given by:

$$\mathrm{CW}_i = \min(2^i \cdot \mathrm{CW}_{\min}, \mathrm{CW}_{\max}),$$

where $\mathrm{CW}_{\max} = 2^m \cdot \mathrm{CW}_{\min}$, and m denotes the maximum backoff stage. After a successful transmission, i is reset to 0, and the CW returns to $\mathrm{CW}_{\min}$.

DCF optionally supports a Request To Send (RTS)/Clear to Send (CTS) handshake to mitigate the *hidden node problem*, which arises when a transmitting node is unable to detect another node's ongoing transmission due to signal attenuation or physical distance. In this case, the sender first transmits an RTS frame[5]. Upon reception of the RTS, the receiver—if available to receive—replies with a CTS frame after a SIFS interval. Upon reception of the CTS, the sender transmits the data frame after another SIFS period.

The RTS/CTS mechanism also facilitates *virtual carrier sensing* through the Network Allocation Vector (NAV). The NAV specifies the duration for which the channel is expected to remain occupied by an ongoing transmission. This value, included in each frame's MAC header, instructs any node that receives such a frame to defer channel access for the indicated duration, thereby mitigating packet collisions.

[3]A *time slot* in IEEE 802.11 serves as the fundamental timing unit for carrier sensing and backoff countdown.

[4]Retransmissions are attempted up to a predefined retry limit. Once the limit is reached without success, the frame is discarded.

[5]RTS/CTS is typically used only if the data frame size exceeds a predefined threshold to avoid unnecessary overhead for short frames.

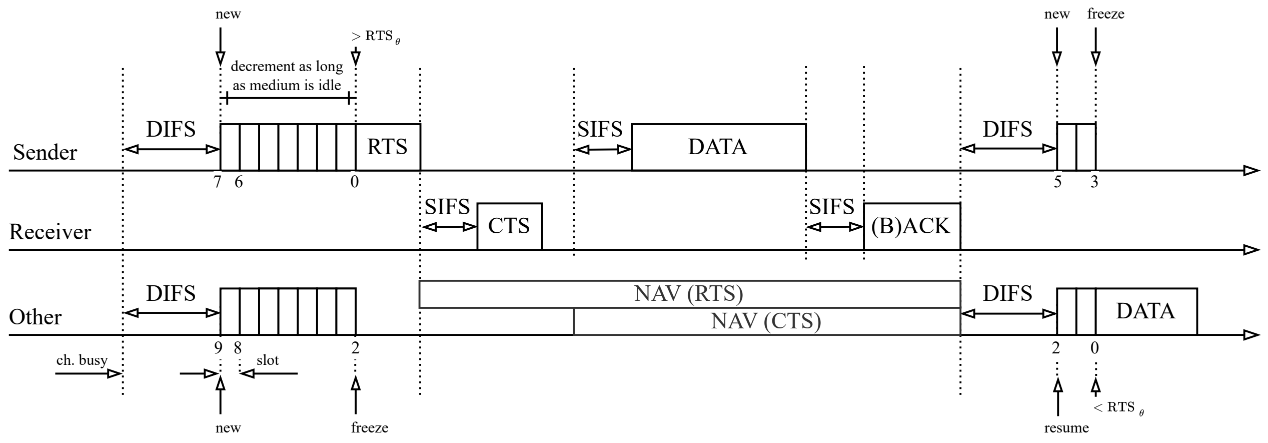


Figure A.1: IEEE 802.11 DCF channel access procedure, including the optional RTS/CTS exchange. *Source: Self-elaborated, adapted from [3, 4].*

Notably, if a node receives a corrupted or undecodable frame—typically due to interference or a collision—it defers its transmission for an Extended Inter-Frame Space (EIFS) duration instead of the usual DIFS, thereby extending the waiting period before attempting to access the channel again to allow any potential acknowledgment from the intended receiver to be transmitted without interference.

Fig. A.1 illustrates the IEEE 802.11 DCF procedure, incorporating the RTS/CTS handshake. The duration of a successful transmission in this case is given by:

$$T_{\text{succ}} = \begin{cases} T_{\text{RTS}} + T_{\text{SIFS}} + T_{\text{CTS}} + T_{\text{SIFS}} + T_{\text{DATA}} + T_{\text{SIFS}} + T_{\text{(B)ACK}} + T_{\text{DIFS}} + T_e & \text{no coll.} \\ T_{\text{RTS}} + T_{\text{SIFS}} + T_{\text{CTS}} + T_{\text{SIFS}} + T_{\text{DATA}} + T_{\text{EIFS}} + T_e & \text{coll.} \end{cases} \tag{A.1}$$

where T_e is the average duration of an empty backoff slot, T_{DIFS}, $T_{\text{DIFS/EIFS}}$ and T_{SIFS} are the DIFS, EIFS (if a collision occurs), and SIFS durations, respectively, and T_{RTS}, T_{CTS}, T_{DATA}, and $T_{\text{(B)ACK}}$ denote the transmission times for the RTS, CTS, data, and acknowledgment frames, respectively [5].

In contrast, if the RTS/CTS exchange is not employed the duration of a successful transmission is:

$$T_{\text{succ}} = \begin{cases} T_{\text{DATA}} + T_{\text{SIFS}} + T_{\text{(B)ACK}} + T_{\text{DIFS}} + T_e & \text{no coll.} \\ T_{\text{DATA}} + T_{\text{EIFS}} + T_e & \text{coll.} \end{cases}$$

A.1.2 Channel bonding

Latest IEEE 802.11 amendments support the bonding of contiguous, non-overlapping (basic) 20 MHz channels into wider 40 MHz, 80 MHz, 160 MHz, and 320 MHz bands [164], enabling higher data rates but increasing the likelihood of contention and interference particularly, in dense deployments.

This technique, known as *Channel Bonding* (CB), is implemented using Static Channel Bonding (SCB) or Dynamic Channel Bonding (DCB). A 20 MHz channel is designated as the *primary* (control) *channel* and the remaining bonded channels are considered *secondary channels*. The primary channel is used for transmitting control and management frames such as RTS, CTS, or (B)ACK. In contrast, data frames are transmitted over the entire bonded channel (or over a subset composed of the primary and contiguous secondary channels). The primary channel is also used for carrier sensing and contention, as described in Section A.1.1. Under CB, before initiating transmission (i.e., before the backoff counter reaches zero), the node also senses the secondary channels for a Point Coordination Function (PCF) Inter-frame Space (PIFS) period.

SCB assumes that waiting for the entire bonded band to be idle is more efficient than using only an idle portion [165]. Nevertheless, numerous analytical and empirical studies have demonstrated that this assumption does not hold true [166–168]. Thus, in SCB, if any channel in the bonded band is busy, the device defers transmission, treating the entire bonded band as occupied—potentially leading to spectrum underutilization [169]. This means that all secondary channels must remain idle during the PIFS period. If not, the node starts the contention process again [98].

In contrast, DCB adapts to the instantaneous spectrum occupancy, utilizing only the available portion of the spectrum for each transmission [170]—enabling more frequent transmissions and improved spectrum utilization. This means that idle secondary channels, contiguous to the primary channel, can be used for transmission alongside the primary channel—according to the channel access specification defined in the 802.11 standards. If none of the secondary channels are idle during the PIFS period, the node proceeds to transmit only on the primary channel. Several distinct DCB policies have been proposed, including Always-max (AM) and Probabilistic Uniform (PU). The AM policy selects the largest group of adjacent 20 MHz channels (including the primary) that are sensed as idle during the PIFS period, thereby maximizing transmission bandwidth. If multiple channel sets with the same maximum width are available, one of them is chosen uniformly at random. In contrast, the PU policy selects uniformly at random one of these idle channel sets, regardless of their bandwidth. [6]

Fig. A.2 illustrates the temporal evolution of the CSMA/CA access mechanism under distinct CB policies, including SCB, AM, and PU, as well as baseline, Only-Primary (OP) (or single-channel) operation, which considers only the primary channel for all transmissions.

A.1.3 Frame aggregation

Recent IEEE 802.11 amendments introduce *frame aggregation* at the MAC layer. This mechanism enables multiple data units—up to a standard-defined maximum size—to be transmitted in a single transmission, enhancing data transmission efficiency by reducing the number of contention periods, interframe spacing (e.g., DIFS and SIFS), and acknowledgment frames. Furthermore, it minimizes protocol overhead, as multiple data units are transmitted using a single Physical (PHY) header and, depending on the aggregation mechanism, a single MAC header.

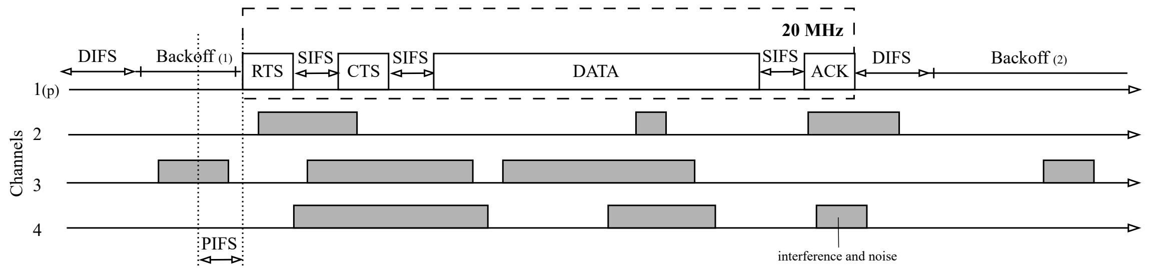
There are two aggregation mechanisms defined in the standard: MAC Service Data Unit (MSDU)[6] aggregation and MAC Protocol Data Unit (MPDU)[7] aggregation. [171]

In MSDU aggregation, multiple MSDUs are grouped together and encapsulated using a single MAC header to form a single MAC frame, known as an Aggregated Medium Access Control (MAC) Service Data Unit (A-MSDU). An A-MSDU includes only MSDUs destined to the same receiver and transmitted by the same transmitter, and is sent using a single PHY-layer header. Since all payloads are encapsulated under one MAC and PHY header, the aggregation significantly reduces per-packet overhead. However, since the A-MSDU is treated as a single MAC frame, any error detected during transmission requires the retransmission of the entire aggregate, even if only one MSDU was corrupted.

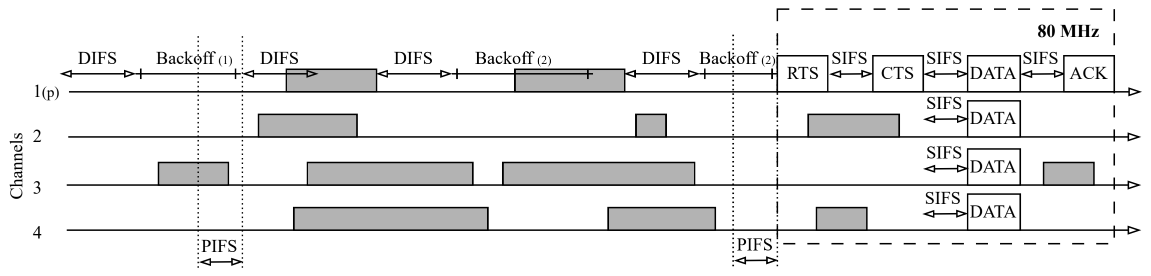
In MPDU aggregation, multiple MPDUs are concatenated to form an Aggregated Medium Access Control (MAC) Protocol Data Unit (A-MPDU). Like the A-MSDU, an A-MPDU includes frames from the same transmitter to the same receiver and is transmitted using a single PHY-layer header. However, unlike A-MSDU, each MPDU in an A-MPDU retains its own MAC header and FCS, allowing for independent error checking and retransmission. In particular, at the receiver side, a single BACK frame is sent in response to an A-MPDU, rather

[6]An MSDU is a data unit received by the MAC layer from the upper layer. It contains the payload data to be encapsulated in a MAC frame for transmission.

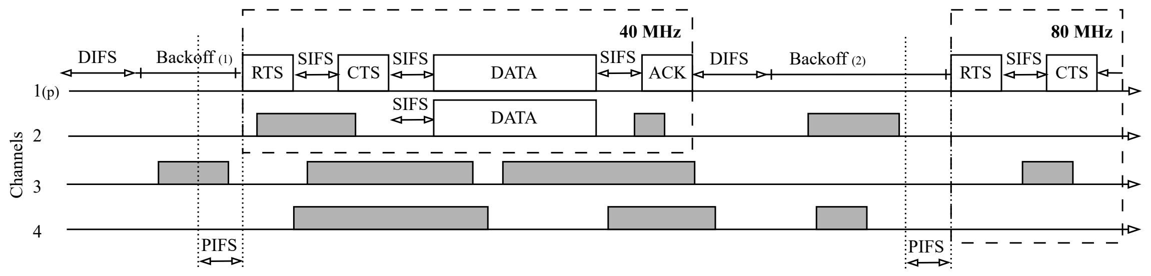
[7]An MPDU is a complete MAC frame, including the MAC header, the encapsulated MSDU, and a Frame Check Sequence (FCS) trailer that enables error detection. This is the data unit passed to the PHY layer for transmission over the wireless medium.



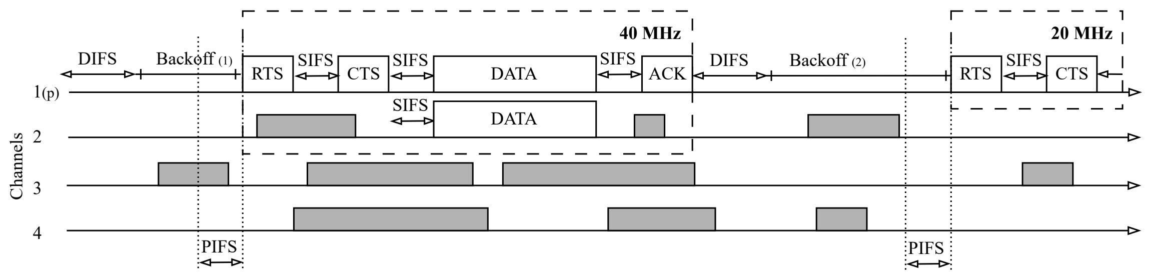
(a) Only-Primary (OP)



(b) Static Channel Bonding (SCB)



(c) Always-max (AM)



(d) Probabilistic Uniform (PU)

Figure A.2: Temporal evolution of the CSMA/CA channel access mechanism under four distinct CB policies: (a) OP, (b) SCB, (c) AM, and (d) PU. Each scenario assumes four available 20 MHz channels, $\{1(\mathrm{p}), 2, 3, 4\}$, where channel 1 is designated as the *primary* channel. Control and management frames—such as RTS, CTS, and (B)ACK—are transmitted exclusively on the primary channel, and thus their duration remains constant across policies. In contrast, data frame durations decrease with increasing channel bandwidth. *Source: Self-elaborated, adapted from [5, 6].*

than individual ACKs for each MPDU. The BACK frame contains a bitmap indicating which MPDUs were received correctly. If any subframes were not successfully received, those are retransmitted in a subsequent transmission attempt. In line with the standard DCF procedure, if the BACK frame is not received within a specified timeout interval, the transmitter assumes that the entire set of MPDUs has failed and schedules them for retransmission.

Thus, both A-MSDU and A-MPDU mechanisms aim to reduce the per-packet overhead by minimizing the number of PHY (and, in the case of A-MSDU, MAC) headers, interframe spaces, and channel access attempts. MSDU aggregation introduces less overhead than MPDU aggregation, as the entire aggregate uses a single MAC header. Nevertheless, MPDU aggregation offers greater robustness and flexibility, as it allows selective retransmission of only failed subframes rather than requiring the retransmission of the entire aggregate.

A.2 Simulator's IEEE 802.11 features and abstractions

To balance fidelity and tractability, the simulator abstracts certain IEEE 802.11 features. This section outlines the key abstractions and assumptions made in the simulator's design.

Network and Traffic modeling

- **Network topology**: The simulator models a network composed of multiple Basic Service Sets (BSSs), each comprising a single AP and one or more associated STAs. However, unless otherwise stated, throughout this thesis each BSS includes a single STA. The network topology remains static within each simulation run: nodes are placed at fixed positions, and there is no user mobility or handovers (i.e., STA transitioning between APs) are modeled.

- **Association**: The beaconing, scanning, authentication, and association procedures defined in the IEEE 802.11 standard are not implemented. Instead, all STAs are assumed to be pre-associated with their APs at the beginning of the simulation.

- **Traffic**: Traffic flows can be synthetically generated or loaded from trace files to enable reproducibility and realism. Supported configurable synthetic models, which do not support congestion-aware adaptation, include *Poisson* traffic (exponentially distributed inter-arrival times), *Bursty* traffic (exponentially-distributed inter-burst intervals), and *Virtual Reality* (VR) traffic (burst sizes and inter-arrival times based on a fixed frame rate). Additionally, a *full-buffer* mode is supported, simulating a continuously saturated source. All generated or loaded traffic is assumed to originate from layers above the

MAC layer and to include the corresponding upper-layer protocol headers (e.g., User Datagram Protocol (UDP)/Transmission Control Protocol (TCP) at the Transport layer and Internet Protocol (IP) at the Network layer). Nevertheless, upper-layer functionalities are not modeled.

- **Communication**: The simulator models only *unicast* (i.e., one-to-one) communications between APs and their associated STAs. Thus, no *broadcast* (one-to-all) or *multicast* (one-to-group) traffic is supported. Only Downlink (DL) traffic (i.e., data transmitted from an AP to an associated STA) is considered throughout the thesis unless otherwise stated. Uplink (UL) traffic is also supported by the simulator but remains outside the scope of our evaluations. Additionally, there is no inter-BSS communication modeled, nor is STA-to-STA communication considered. Indeed, intra-BSS forwarding[8] is not yet modeled. Thus, all communication is restricted to AP-STA communication.

MAC layer modeling

- **Channel access**: The simulator implements the legacy IEEE 802.11 DCF for channel access, as described in Section A.1.1, modeling a best-effort network without packet prioritization; therefore, Enhanced Distributed Channel Access (EDCA)[9] is not supported. Notably, the simulator assumes that nodes contending on the same primary channel decrement their backoff counters synchronously. Furthermore, a fixed BACK timeout is assumed, meaning that the time the sender waits for BACK frame after transmitting an A-MPDU is constant and does not depend on the number of aggregated MPDUs or the data rate. Any BACK frame received after this timeout is considered outdated and ignored.

- **Carrier sensing**: The simulator does not model Signal Detect (SD) and Energy Detect (ED) thresholds—to detect 802.11 and non-802.11 radio transmissions, respectively—as specified in IEEE 802.11 [163]. Instead, a simplified CCA abstraction is used: the medium is considered busy if any transmission is ongoing in the channel, regardless of the signal strength or distance between nodes. Thus, medium access is based on perfect carrier sensing without noise-induced deferrals or missed detections as any node will sense and avoid another ongoing transmission.

- **Queuing**: Each node maintains a fixed-size transmission queue at the MAC layer. Incoming packets are dropped if the queue is full.

[8] Intra-BSS forwarding refers to the transmission of packets between STAs within the same BSS via the AP.

[9] EDCA is an enhancement of DCF that relies on CSMA/CA while incorporating Quality of Service (QoS) mechanisms. In particular, EDCA defines four access categories—Background (0), Best Effort (1), Video (2), and Voice (3)—each with different prioritization and contention parameters. [172]

- **Management and control frames**: Only RTS, CTS, and BACK control frames are modeled. Other management frames such as beacons, probe request/responses, and association frames are not implemented.[10]. These control frames are transmitted using Modulation and Coding Scheme (MCS) 0, a single Spatial Streams (SS), and a Guard Interval (GI) of 0.8 μs, i.e., at a low PHY data rate.

- **Aggregation and fragmentation**: The simulator supports MPDU aggregation but does not implement MSDU aggregation nor fragmentation[11]. Thus, the basic data unit for transmission is an A-MPDU, which is acknowledged using a BACK frame.

- **Retransmissions**: A single Common Retry Count (CRC) is used for MAC-layer retransmissions, rather than separate short and long retry counters as defined in the standard [163]. Retransmissions are triggered according to collision events and the associated BACK feedback.

- **MAC header**: MAC-layer headers are included as a fixed byte overhead per frame. Header fields are not individually modeled.

PHY layer modeling

- **Transmit power and beamforming**: Each node transmits at a fixed power level, with no support for dynamic transmission power control or directional beamforming. Thus, signal propagation is assumed isotropic, and, thereby, the Received Signal Strength Indicator (RSSI) at a given distance is the same regardless of the propagation direction.

- **Modulation**: The modulation process is abstracted via Modulation and Coding Scheme (MCS) indices and their associated PHY data rates. An MCS index is selected based on the receiver's Received Signal Strength Indicator (RSSI), leveraging an MCS lookup table (see [174]) and remains fixed throughout the simulation. Thus, no rate adaptation algorithm is modeled. In our simulator, the RSSI is approximated by the computed received power, P_{rx}, which is given by [175]:

$$P_{\text{rx}} = P_{\text{tx}} + G_{\text{tx}} + G_{\text{rx}} - \text{PL}(d),$$

  where P_{tx} is the transmit power, G_{tx} and G_{rx} are the transmit and receive antenna gains, respectively, and $\text{PL}(d)$ is the path loss at distance d.

[10]For a comprehensive list of 802.11 frame types and formats, see [173]

[11]Fragmentation refers to dividing larger frames into smaller frames to increase the probability of successful transmission, but at the cost of increased overhead.

- **Transmission delay**: The transmission delay, defined as the time it takes to push all frame bits to the medium, depends on the frame size and the PHY data rate. PHY data rates are computed, assuming an Orthogonal Frequency Division Multiplexing (OFDM)-based [12] and 802.11ax (a.k.a. Wi-Fi 6) compliant PHY layer, as follows [176]:

$$\text{Data Rate} = \frac{N_{\text{SD}} \cdot N_{\text{BPSCS}} \cdot R \cdot N_{\text{SS}}}{T_{\text{DFT}} + T_{\text{GI}}},$$

  where N_{SD} is the number of data subcarriers (e.g., 234 for a 20 MHz channel), N_{BPSCS} is the number of coded bits per subcarrier per spatial stream (e.g., 6 for 64-QAM), R is the coding rate (e.g., 3/4), N_{SS} is the number of SS, T_{DFT} is the OFDM symbol duration (12.8 μs), and T_{GI} is the guard interval duration (e.g., 0.4 μs), as detailed in [176].

- **Error correction**: No PHY-layer error correction mechanisms (such as Forward Error Correction (FEC)) are implemented.

- **Advanced mechanisms**: No advanced PHY mechanisms such as Orthogonal Frequency Division Multiple Access (OFDMA)[13], Multiple-Input Multiple-Output (MIMO)[14], Target Wake Time (TWT)[15], Spatial Reuse Operation (SRO)[16], or BSS Coloring[17] are implemented.

- **PHY header:** PHY-layer preambles and headers are represented as a fixed byte overhead per transmission. Their internal structure is not explicitly modeled. PHY headers are assumed to always be received correctly.

Spectrum and signal modeling

- **Channel**: The simulator operates in the 5 GHz band with a simplified channelization scheme comprising of up to 8 basic 20 MHz channels. However, throughout this thesis, only four basic 20 MHz channels are utilized, corresponding to the following channel allocations: $\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}, \{1, 2, 3, 4\}\}$. It is assumed that the selected channels are known by all associated STAs at transmission time.

[12] OFDM divides the channel into multiple orthogonal subcarriers, allowing simultaneous transmission of multiple symbols.
[13] OFDMA divides the available frequency band into multiple orthogonal sub-channels, called resource units, enabling multiple users to access the network simultaneously (frequency multiplexing).
[14] MIMO enables the simultaneous transmission and reception of multiple data streams by using multiple antennas (spatial multiplexing).
[15] TWT enhances power efficiency by scheduling device wake times.
[16] SRO allows nodes to dynamically adjust CCA thresholds to reuse the spectrum more efficiently and avoid unnecessary deferrals. [177]
[17] BSS Coloring assigns identifiers to distinguish between intra- and inter-BSS transmissions on the same channel, aiding spatial reuse. [178]

- **Channel Bonding (CB)**: Both SCB and AM DCB are supported, but SCB is assumed throughout the thesis unless otherwise stated.

- **Interference**: Adjacent channel interference is assumed negligible. Similarly, inter-BSS interference (i.e., from neighboring BSSs operating on overlapping channels) is not modeled. Thus, no *hidden terminal problem*—where a node cannot detect a transmission causing a collision—or *exposed terminal problem*—where a node unnecessarily defers transmission despite not causing interference—are modeled. Similarly, signal quality is modeled using Signal-to-Noise Ratio (SNR)[18], not Signal-to-Interference-and-Noise Ratio (SINR), as no interference effects are considered.

- **Signal propagation and path loss**: Propagation delay, defined as the time it takes a signal to travel from the sender to the receiver, is considered negligible. A deterministic free-space log-distance path loss model is used, assuming a reference distance of 1 m. *Shadowing*[19] and *fading*[20] are not considered. Thereby, the path loss between transmitter and receiver is computed as follows [175]:

$$\mathrm{PL}(d) = \mathrm{PL}(d_0) + 10 \cdot \gamma \cdot \log_{10} \frac{d}{d_0},$$

  where γ is the path loss exponent and d_0 is the reference distance. Assuming $d_0 = 1$ m and free-space conditions, the reference path loss is given by:

$$\mathrm{PL}(d_0) = 20 \log_{10}(f_c) - 147.55,$$

  where f_c is the carrier frequency in Hz.

- **Collisions:** Concurrent transmissions on overlapping channels are assumed to result in collisions, regardless of the degree of overlap between the transmissions. As any node is assumed to perfectly sense any ongoing transmission, collisions only happen if two nodes start transmitting at the same time (i.e., their backoff counters reach zero in the same slot), resulting in a simultaneous transmission attempt.

- **Errors**: A fixed Packet Error Rate (PER)-based abstraction models MPDU losses, approximating the likelihood of packet loss due to transmission errors at the PHY layer.

[18]SNR is defined as the ratio of the received signal power, P_{rx}, to the noise power, $P_{\mathrm{noise}} = kTB$. Here, k is Boltzmann's constant, T is the temperature (in Kelvin), and B is the bandwidth.

[19]Shadowing refers to signal attenuation caused by obstacles between transmitter and receiver, typically modeled as a log-normal (Gaussian) random variable $\mathcal{X}$ with standard deviation σ (in dB). [179]

[20]Fading denotes rapid signal fluctuations due to multipath propagation, often modeled using Rayleigh distributions.

A.3 Simulation framework: SimPy

The simulator leverages SimPy, a discrete-event simulation framework in Python. Each simulation instance is encapsulated in a SimPy environment (`env = simpy.Environment()`). An environment handles the progression of simulation time, schedules events, and manages the lifecycle of concurrent processes.

Time progression in SimPy is event-driven, meaning that the simulation advances from an scheduled event to the next. A simulation is started by invoking `env.run(until=<time>)`, where `<time>)` specifies the simulation duration. The simulation time can be accessed at any point using `env.now`, which is used extensively throughout the simulator to timestamp events, schedule delays, measure durations, and track time progress.

The fundamental components of SimPy are processes and events.

In SimPy, a process represents a sequence of timed events and is defined as a generator function that yields events (`yield <event>`). Processes run asynchronously and are instantiated using `env.process()`. When a process yields an event, its execution is suspended until that event occurs. Notably, multiple processes can yield the same event and will resume independently once the event is triggered. A process itself can also be yielded using `yield <process>`.
In the simulator, any function that yields an event is generally treated as a process to enable asynchronous execution. For example, while one node may be waiting for a CTS response, another can concurrently proceed with its backoff countdown, each operating within its own isolated process. Yield statements are used to model sequential protocol logic. For instance, a CTS response cannot be transmitted until a SIFS period has elapsed.

In SimPy, an event represents an occurrence at specific simulation time. Events are instantiated using `env.event()` and can be triggered as successful using `<event>.succeed()`.The Boolean attribute `<event>.triggered` indicates whether an event has occurred or not.
In the simulator, for instance, events are used to signal a channel becoming idle or busy, the reception of a CTS or BACK frame, or the occurrence of a collision.

A timeout event is a specific type of event that completes after a certain amount of specified time. It is created using `env.timeout(<delay>)`. Yielding a timeout event suspends the process for the simulated time interval.
In the simulator, timeouts are used to model durations such as inter-frame spaces (SIFS, DIFS), frame transmission times, backoff slot times, response timeouts (e.g., for CTS and BACK), and

packet inter-arrival times in traffic generators.

SimPy also provides combinator constructs for event handling, such as `AllOf(env,[<events>])` and `AnyOf(env,[<events>])`. Yielding an `AllOf` waits until all listed events are completed, whereas `AnyOf` proceeds when any of them occurs.
For instance, `AllOf` can be used to ensure that a group of channels is simultaneously idle before a node begins transmitting. On the other hand, `AnyOf` can be utilized to determine if a channel becomes busy before the expiration of a DIFS timeout, enabling the node to assess whether to interrupt its backoff process or defer transmission based on the channel's status.

A.4 Simulator validation

The simulator has been thoroughly tested and validated to ensure its correctness and reliability. In particular, its operation has been verified through detailed inspection of console event logs across a wide range of controlled scenarios, including single- and multi-BSS environments, with nodes sharing a single channel or operating on non-overlapping, partially overlapping and fully overlapping frequency bands.

DCF validation

Beyond empirical verification, the simulator's accuracy has also been validated analytically. Specifically, the medium access procedure, DCF, has been evaluated against Bianchi's well-known analytical model [180], which derives closed-form expressions for the transmission and collision probabilities under saturation conditions.

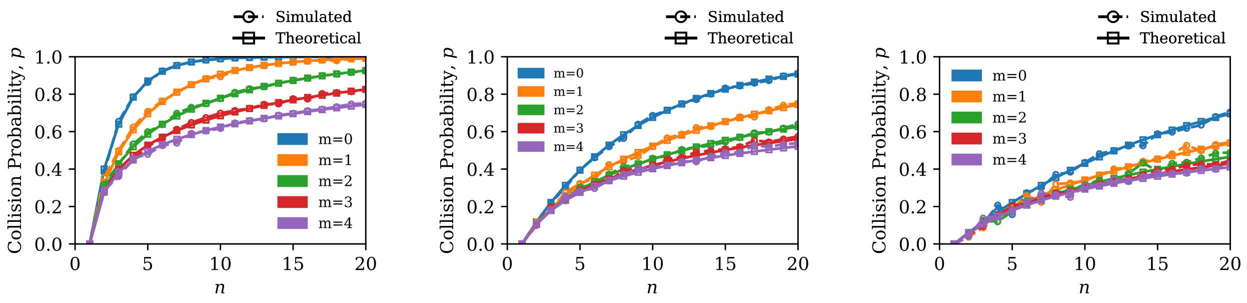
According to Bianchi's model, the probability τ that a node transmits in a randomly chosen slot and the conditional collision probability p are related through the following coupled non-linear equations:

$$\tau = \frac{2(1-2p)}{(1-2p)(\mathrm{CW}_{\min}+1) + p\cdot \mathrm{CW}_{\min}(1-(2p)^m)}$$

$$p = 1-(1-\tau)^{n-1}$$

where $\mathrm{CW}_{\min}$ is the minimum contention window size, m is the maximum backoff stage, and n is the number of contending nodes. These equations can be numerically solved using a relaxed fixed-point iteration method, as described in [181].

Fig. A.3 compares the collision probability observed in the simulator against Bianchi's ana-



(a) $\mathrm{CW}_{\min} = 4$ (b) $\mathrm{CW}_{\min} = 16$ (c) $\mathrm{CW}_{\min} = 32$

Figure A.3: Simulator and Bianchi's analytical model collision probabilities under varying minimum contention window sizes ($\mathrm{CW}_{\min}$), maximum backoff stages (m), and number of contending nodes (n). Simulator runs consider a single shared channel with all nodes operating in full-buffer mode.

lytical predictions, demonstrating that the simulator accurately reproduces the DCF medium access behavior, as the collision probabilities closely match.

Static and Dynamic Channel Bonding validation

Furthermore, to validate the correct operation of SCB and DCB, we used Continuous-Time Markov Chains (CTMCs) [182], which have been shown to provide accurate throughput[21] estimates for CSMA/CA-based networks [168, 183, 184].

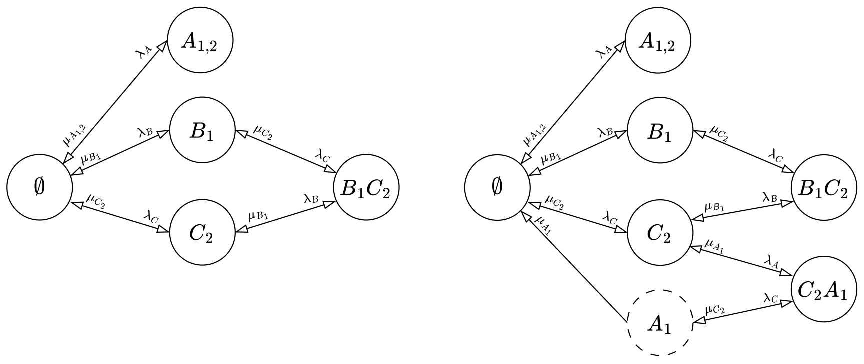
Our considered scenario includes three BSSs: BSS A on channels 1 and 2 (40 MHz), BSS B on channel 1 (20 MHz), and BSS C on channel 2 (20 MHz).

Fig. A.4a and Fig. A.4b depict the CTMC models under SCB and DCB, respectively. Each state in the CTMC represents a particular set of concurrently transmitting BSSs. Transitions between states occur at rate $\lambda = 1/\mathbb{E}[\mathrm{bo}]$, the channel access rate, when a BSS initiates a transmission, and at rate $\mu = 1/\mathbb{E}[T_{\mathrm{succ}}]$, the transmission rate, when a BSS completes its transmission [185]. Here, $\mathbb{E}[\mathrm{bo}]$ is the mean backoff duration, and $\mathbb{E}[T_{\mathrm{succ}}]$ is the mean transmission time (as defined in Eq. A.1). The throughput Γ (in bps) of BSS n is calculated as:

$$\Gamma^{(n)} = \sum_{\forall s : n \in s} \pi_s \cdot \mu_n^s \cdot L \cdot (1 - p_{\mathrm{err}}),$$

where π_s is the steady-state probability of state s, μ_n^s is the transmission rate of BSS n in state s, and L is the transmitted PHY Protocol Data Unit (PPDU) size in bits [168]. The

[21]Throughput is defined as the amount of data successfully received at the destination per unit of time, typically measured in bits per second (bps).



(a) SCB (b) DCB

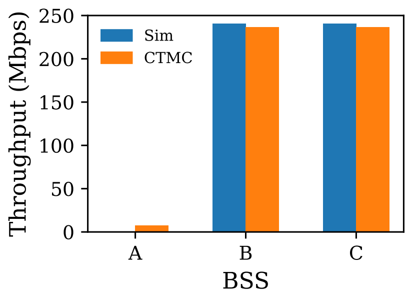
Figure A.4: CTMC models. Circles represent CTMC states. For instance, state $\emptyset$ denotes no active transmissions, $A_{1,2}$ indicates BSS A transmitting over channels 1 and 2 (at 40 MHz), while B_1C_2 indicates BSS B and BSS C transmitting simultaneously on channels 1 and 2, respectively (each at 20 MHz). States outlined with dashed lines correspond to DCB behavior, where a BSS transmits over only part of its allocated channel. Forward and backward transitions occur at rate λ and μ, respectively. To avoid overloading the figure, forward and backward transitions are represented using a single double-sided arrow.

steady-state distribution $\vec{\pi}$ is obtained by solving the global balance equations:

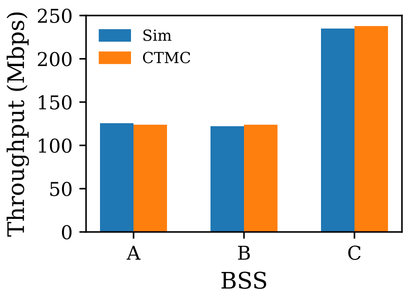
$$\vec{\pi}Q = 0, \quad \text{with} \quad \sum_s \pi_s = 1,$$

where Q is the CTMC transition rate matrix [168].

Fig. A.5 compares the throughput predicted by the CTMCs against simulation results. As shown, the values are closely aligned, confirming the correct implementation of SCB and DCB in the simulator. Small deviations may arise due to CTMCs not explicitly modeling collisions [185].



(a) SCB



(b) DCB

Figure A.5: Throughput comparison between CTMC predictions and simulator results under SCB and DCB. MCS 11, $\text{CW}_{\text{min}} = 16$, $m = 0$, and full-buffer mode were considered for all BSSs. Thus, $\lambda_A = \lambda_B = \lambda_C = 2/((\text{CW}_{\text{min}} - 1) \cdot T_e)$. Since all BSSs transmit PPDUs of equal size, μ depends only on the transmitting bandwidth. Hence, $\mu_{A_1} = \mu_{B_1} = \mu_{C_2}$ (20 MHz), while $\mu_{A_{1,2}}$ (40 MHz) is higher, as its T_{DATA} is shorter.

Appendix B

Algorithms

Algorithm 1 LinUCB with Disjoint Linear Models [40]

```
1: Input: $\alpha \in \mathbb{R}_+$
2: for $t = 1, 2, \ldots, T$ do
3:     Observe context features $\mathbf{x}_{t,a} \in \mathbb{R}^d$ for all $a \in \mathcal{A}_t$
4:     for all $a \in \mathcal{A}_t$ do
5:         if $a$ is new then
6:             $\mathbf{A}_a \leftarrow \mathbf{I}_d$                          ▷ $d \times d$ identity matrix
7:             $\mathbf{b}_a \leftarrow \mathbf{0}_d$                          ▷ $d$-dimensional zero vector
8:         end if
9:         $\hat{\theta}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
10:        $p_{t,a} \leftarrow \hat{\theta}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$
11:    end for
12:    Select arm $a_t = \arg\max_{a \in \mathcal{A}_t} p_{t,a}$ (break ties arbitrarily)
13:    Observe reward $r_t$
14:    $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
15:    $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{t,a_t}$
16: end for
```

Algorithm 2 Sliding Window LinUCB (SW-LinUCB) [129]

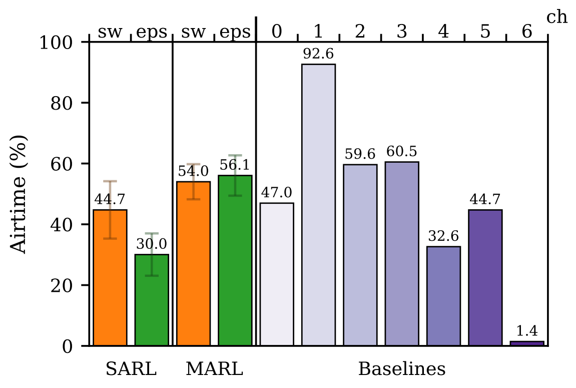
```
1: **Input:** $\alpha \in \mathbb{R}_+$, number of arms $k$
2: $w \leftarrow k$
3: **for** $t = 1, 2, \ldots, T$ **do**
4:     Observe context vector $\mathbf{x}_t \in \mathbb{R}^d$
5:     **for all** $a \in \mathcal{A}_t$ **do**
6:         **if** $a$ is new **then**
7:             $Occ_w(a,t) \leftarrow 0$; $\mathbf{A}_a \leftarrow \mathbf{I}_d$; $\mathbf{b}_a \leftarrow \mathbf{0}_d$; $\mathbf{E}_a \leftarrow [\,]$
8:         **end if**
9:         $\hat{\theta}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
10:        **if** $t > w$ **then**
11:            $Occ_w(a,t) \leftarrow$ number of 1s in $\mathbf{E}_a$
12:        **end if**
13:        $\gamma_t \leftarrow 1 - \frac{Occ_w(a,t)}{w}$
14:        $p^w_{t,a} \leftarrow \gamma_t \hat{\theta}_a^\top \mathbf{x}_t + \alpha\sqrt{\mathbf{x}_t^\top \mathbf{A}_a^{-1}\mathbf{x}_t}$
15:    **end for**
16:    Select arm $a_t = \arg\max_{a \in \mathcal{A}_t} p^w_{t,a}$ (break ties arbitrarily)
17:    **for all** $a \in \mathcal{A}_t$: append 1 to $\mathbf{E}_a$ if $a = a_t$, else append 0
18:    **if** $t > w$ **then**
19:        Truncate $\mathbf{E}_a$ to retain only the last $w$ bits for all $a$
20:    **end if**
21:    Observe reward $r_t$
22:    $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_t\mathbf{x}_t^\top$
23:    $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_t$
24: **end for**
```

Algorithm 3 ϵ-greedy RMSProp-based algorithm with EMA

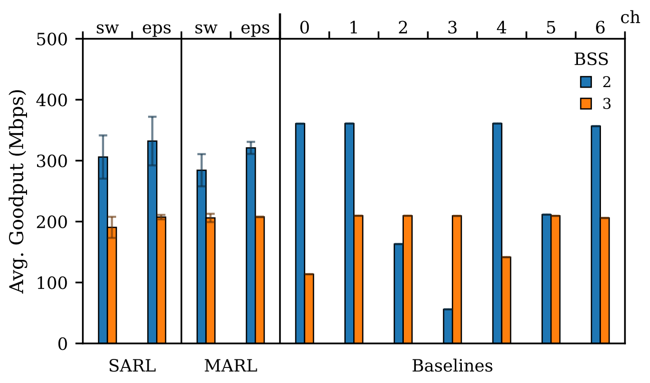
```
 1: Input: $\epsilon \in [0,1]$, $\eta > 0$, $\gamma \in (0,1]$, $\alpha \in [0,1)$
 2: Initialize $v_a \leftarrow \mathbf{0}_d$, $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{0}_d$, and $\hat{\boldsymbol{\theta}}_a^{\text{ema}} \leftarrow \mathbf{0}_d$ for all $a \in \mathcal{A}$
 3: for $t = 1, 2, \ldots, T$ do
 4:     Observe context vector $\mathbf{x}_t \in \mathbb{R}^d$
 5:     for all $a \in \mathcal{A}_t$ do
 6:         $\hat{r}_{t,a}^{\text{ema}} \leftarrow \mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_a^{\text{ema}}$          ▷ Prediction using EMA weights
 7:     end for
 8:     Select arm $a_t \leftarrow \begin{cases} \arg\max_{a \in \mathcal{A}_t} \hat{r}_{t,a}^{\text{ema}} & \text{with probability } 1-\epsilon \\ \text{random arm from } \mathcal{A}_t & \text{with probability } \epsilon \end{cases}$
 9:     Observe reward $r_t$
10:     $\hat{r}_{t,a_t} \leftarrow \mathbf{x}_t^\top \hat{\boldsymbol{\theta}}_{a_t}$          ▷ Prediction using raw weights
11:     $g_t \leftarrow (\hat{r}_{t,a_t} - r_t) \cdot \mathbf{x}_t$
12:     $v_{a_t} \leftarrow \gamma v_{a_t} + (1-\gamma) \cdot g_t^2$
13:     $\hat{\boldsymbol{\theta}}_{a_t} \leftarrow \hat{\boldsymbol{\theta}}_{a_t} - \frac{\eta}{\sqrt{v_{a_t} + \varepsilon}} \cdot g_t$          ▷ RMSProp update
14:     $\hat{\boldsymbol{\theta}}_{a_t}^{\text{ema}} \leftarrow \alpha \cdot \hat{\boldsymbol{\theta}}_{a_t}^{\text{ema}} + (1-\alpha) \cdot \hat{\boldsymbol{\theta}}_{a_t}$          ▷ EMA update
15: end for
```

Appendix C

Figures



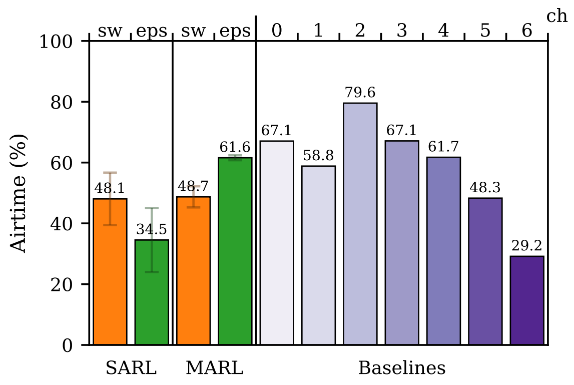
(a)



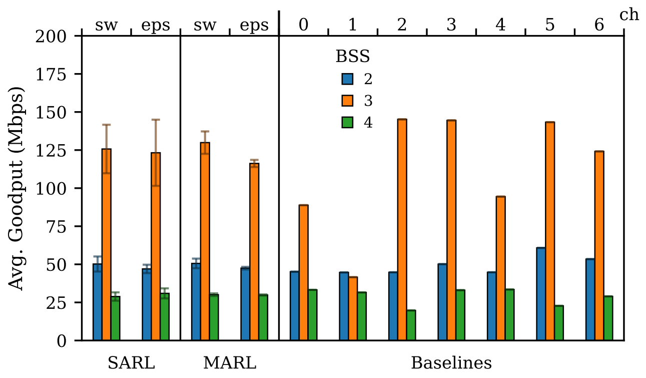
(b)

Figure C.1: (a) Airtime utilization of BSS 1, and (b) goodput of non-learning BSSs in Scenario A, for each algorithm–architecture pair and static non-learning baseline. Results are averaged over five independent trials; bars and error bars indicate the mean and standard deviation across trials, respectively.

[1]In the single-agent architecture, a single agent selects the entire triplet, whereas in the multi-agent architecture, the joint action results from the independent decisions of the three specialized agents. Each component of the triplet (channel group, primary channel, contention window size) corresponds to an index, as described in Section 3.1. For instance, the triplet $(6, 3, 0)$ corresponds to channel group $\{1, 2, 3, 4\}$, primary channel $\{4\}$, and CW size 16.

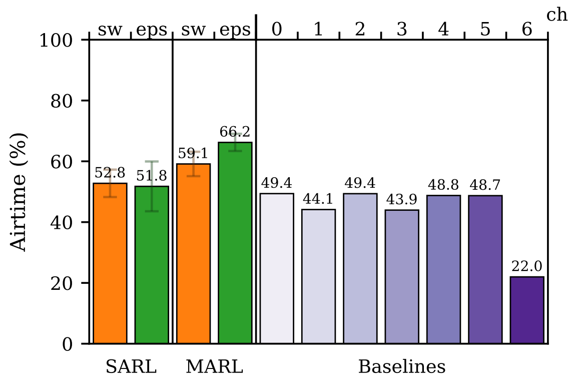


(a)

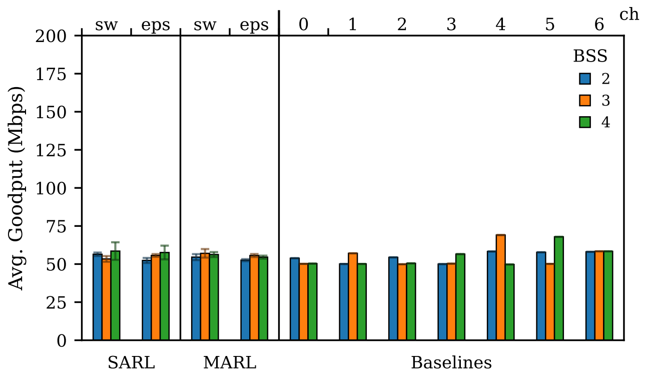


(b)

Figure C.2: (a) Airtime utilization of BSS 1, and (b) goodput of non-learning BSSs in Scenario B, for each algorithm–architecture pair and static non-learning baseline.

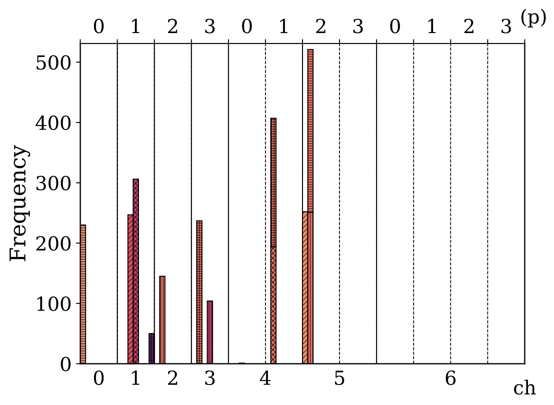


(a)

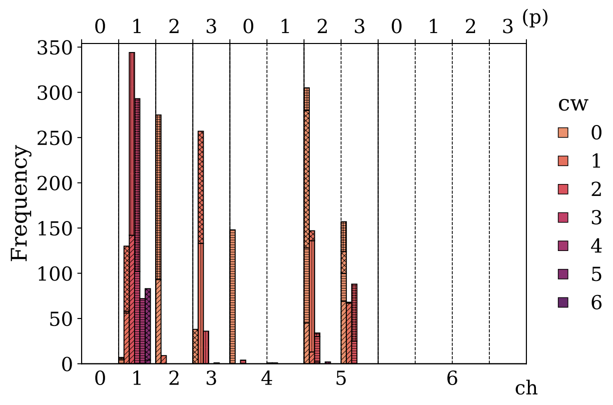


(b)

Figure C.3: (a) Airtime utilization of BSS 1, and (b) goodput of non-learning BSSs in Scenario C, for each algorithm–architecture pair and static non-learning baseline.

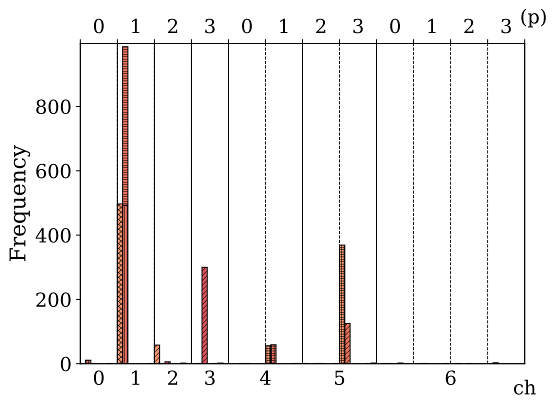


(*i*) SA-CMAB.

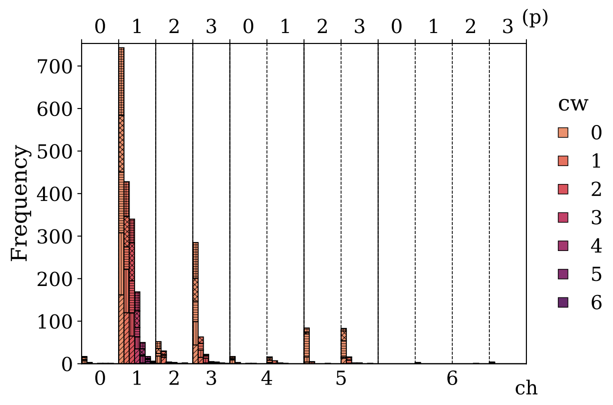


(*ii*) MA-CMAB.

(a) SW-LinUCB.



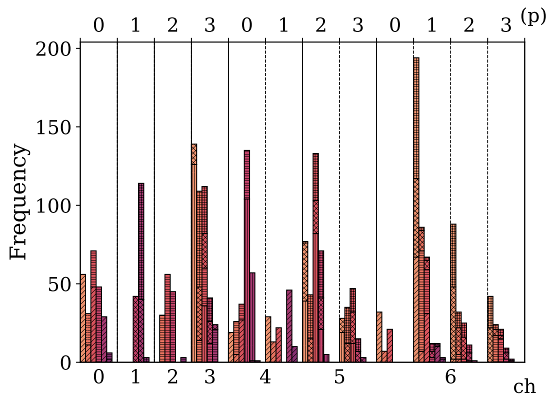
(*i*) SA-CMAB.



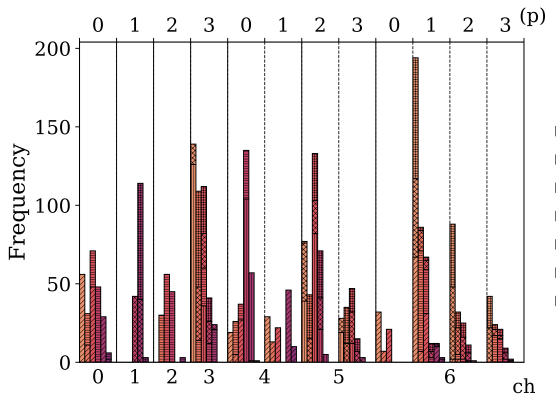
(*ii*) MA-CMAB.

(b) RMSProp-based.

Figure C.4: Histogram of selected joint action indices for each algorithm–architecture pair in Scenario A. Bars are stacked across trials, with distinct hatch patterns indicating different runs. ch: channel group index. (p): primary channel index. cw: contention window size index.[1]

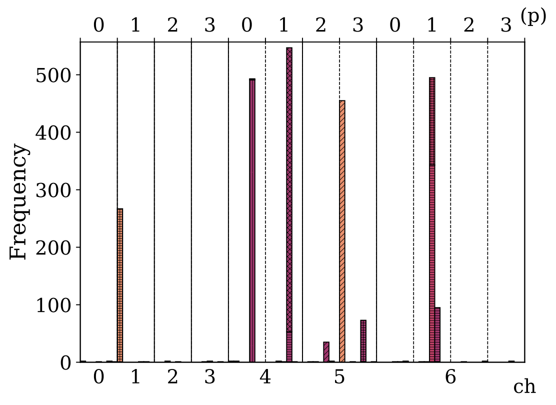


(*i*) SA-CMAB.

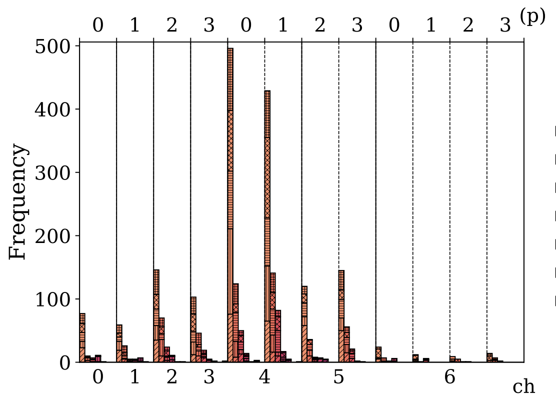


(*ii*) MA-CMAB.

(a) SW-LinUCB.



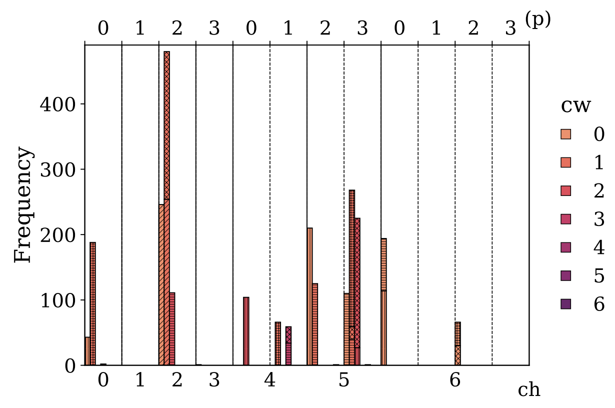
(*i*) SA-CMAB.



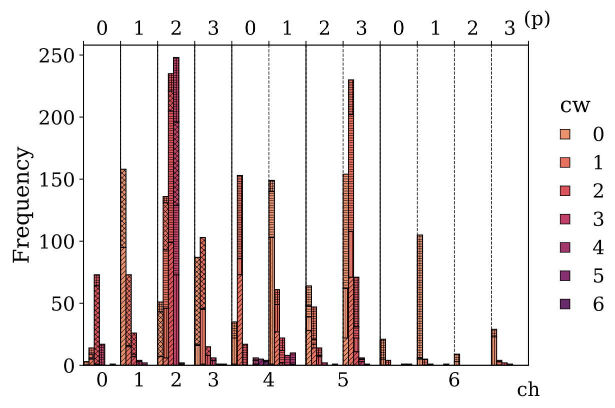
(*ii*) MA-CMAB.

(b) RMSProp-based.

Figure C.5: Histogram of selected joint action indices for each algorithm–architecture pair in Scenario B.

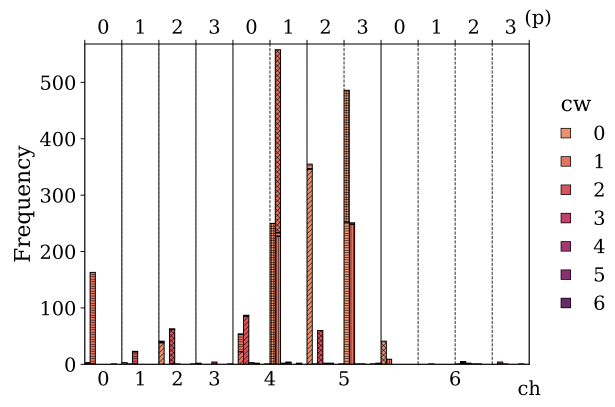


(*i*) SA-CMAB.

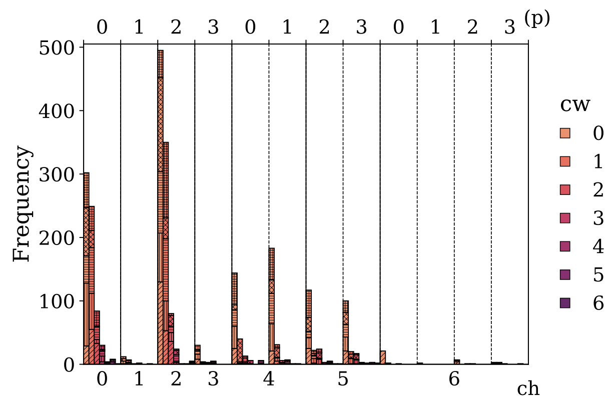


(*ii*) MA-CMAB.

(a) SW-LinUCB.



(*i*) SA-CMAB.



(*ii*) MA-CMAB.

(b) RMSProp-based.

Figure C.6: Histogram of selected joint action indices for each algorithm–architecture pair in Scenario C.

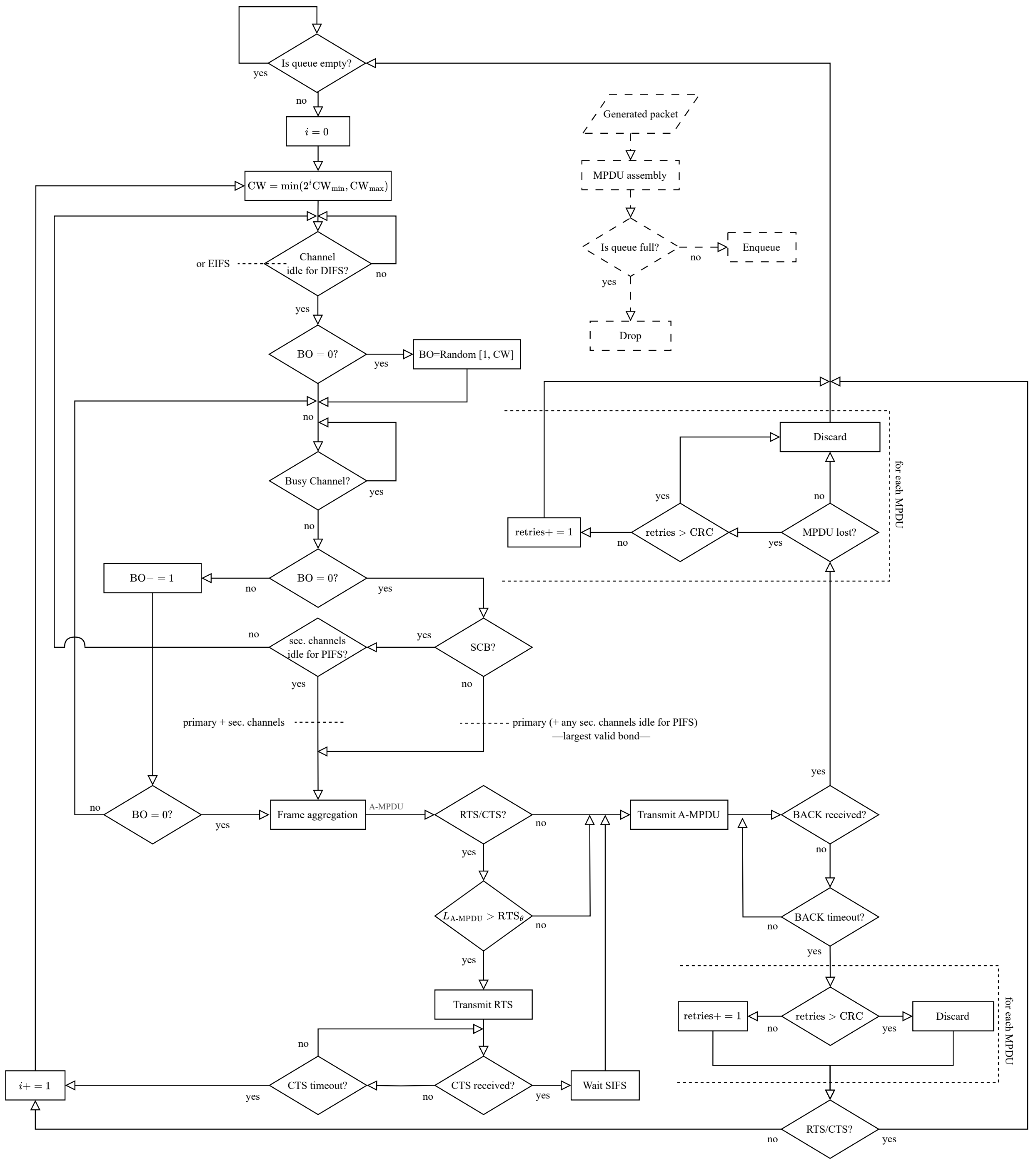


Figure C.7: Flowchart representing the transmitter-side MAC layer operation as implemented in the simulator.

Appendix D

Tables

Table D.1: Configuration details for Scenario A.

BSS	**Positions**		**MCS**	**DL Traffic**	**Channel groups**
1	$(3, 6, 0.5)$	$(4, 8, 0.5)$	11	Full buffer	?
2	$(7, 5, 1)$	$(6, 7, 0.5)$	11	Full buffer	$\{3, 4\}$
3	$(3, 3, 1)$	$(5, 2, 1)$	11	Full buffer	$\{1\}$
	AP	STA			

Table D.2: Configuration details for Scenario B.

BSS	**Positions**		**MCS**	**DL Traffic**	**Channel groups**
1	$(4, 8, 1.5)$	$(4, 5, 0.5)$	11	Full buffer	?
2	$(8, 5, 1.5)$	$(5, 9, 1.5)$	9	Poisson (40 Mbps)	$\{4\}$
3	$(6, 3, 1)$	$(1, 4, 1.5)$	8	VR 90 fps (80 Mbps)	$\{1, 2\}$
4	$(2, 2, 0.5)$	$(6, 6, 1.5)$	8	Bursty (20 Mbps)	$\{3\}$
	AP	STA			

Table D.3: Configuration details for Scenario C.

BSS	**Positions**		**MCS**	**DL Traffic**	**Channel groups**
1	$(4, 18, 1)$	$(7, 10, 1)$	6	Poisson (50 Mbps)	?
2	$(10, 10, 0.5)$	$(12, 17, 1)$	8	Poisson (50 Mbps)	$\{1\} \rightarrow \{3\}$
3	$(14, 14, 1)$	$(18, 8, 0.8)$	7	Poisson (50 Mbps)	$\{2\}$
4	$(2, 4, 1)$	$(10, 6, 1.2)$	7	Poisson (50 Mbps)	$\{4\}$
	AP	STA			

Table D.4: Goodput fairness across Scenario A, B, and C for each algorithm-architecture pair and static baseline, measured using Jain's fairness index[1] [7].

	SA-CMAB		**MA-CMAB**		**Baselines**						
	sw	eps	sw	eps	ch 0	ch 1	ch 2	ch 3	ch 4	ch 5	ch 6
A	0.880	0.801	0.947	0.915	0.740	0.930	0.974	0.816	0.837	1.000	0.630
B	0.724	0.764	0.751	0.644	0.729	0.716	0.674	0.752	0.559	0.765	0.785
C	0.985	0.991	0.988	0.991	0.999	0.996	0.999	0.997	0.928	0.934	0.801

[1] Jain's fairness index, $\mathcal{J} \in [0,1]$, is defined as $\mathcal{J} = \left(\sum_i x_i\right)^2 / \left(n \sum_i x_i^2\right)$, where n is the number of coexisting BSSs and x_i denotes the resource allocation (here, average goodput) of the i^{th} BSS.