الجمهـوريـة الجـزائريـة الديـمقراطيـــة الشـعبيـــة
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعـليـم العـالـي و البـحـث العـلـمي
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعـة عمّـار ثليـجـي بالأغـواط
AMAR TELIDJI UNIVERSITY OF LAGHOUAT

كـليـــة العـلـــوم
FACULTY OF SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

Master's Thesis

Field: Mathematics and Computer Science

Specialty: Computer Science

Option: Networks, Distributed Systems and Applications

Submitted By: Amel Cheifa & Rim Miloudi

Q-learning-based medium congestion control in WSN for smart farming applications

---

Defended publicly on June 28th, 2025, before a jury composed of:

Pr. Chaker Abdelaziz KERRACHE	Prof	President
Dr. Tahar BENDOUMA	M.C.A	Examiner
Dr. Tahar ALLAOUI	M.C.A	Examiner
Dr. Lakhdar OULAD DJEDID	M.C.A	Supervisor

Thesis No. ..... – Academic Year 2024/2025

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University Ammar Thelidji LAGHOUAT

Faculty of Science

Computer Science Department

Master Thesis

Speciality: Computer Science System

Q-learning-based medium congestion control in WSN for smart farming applications

Executed by:

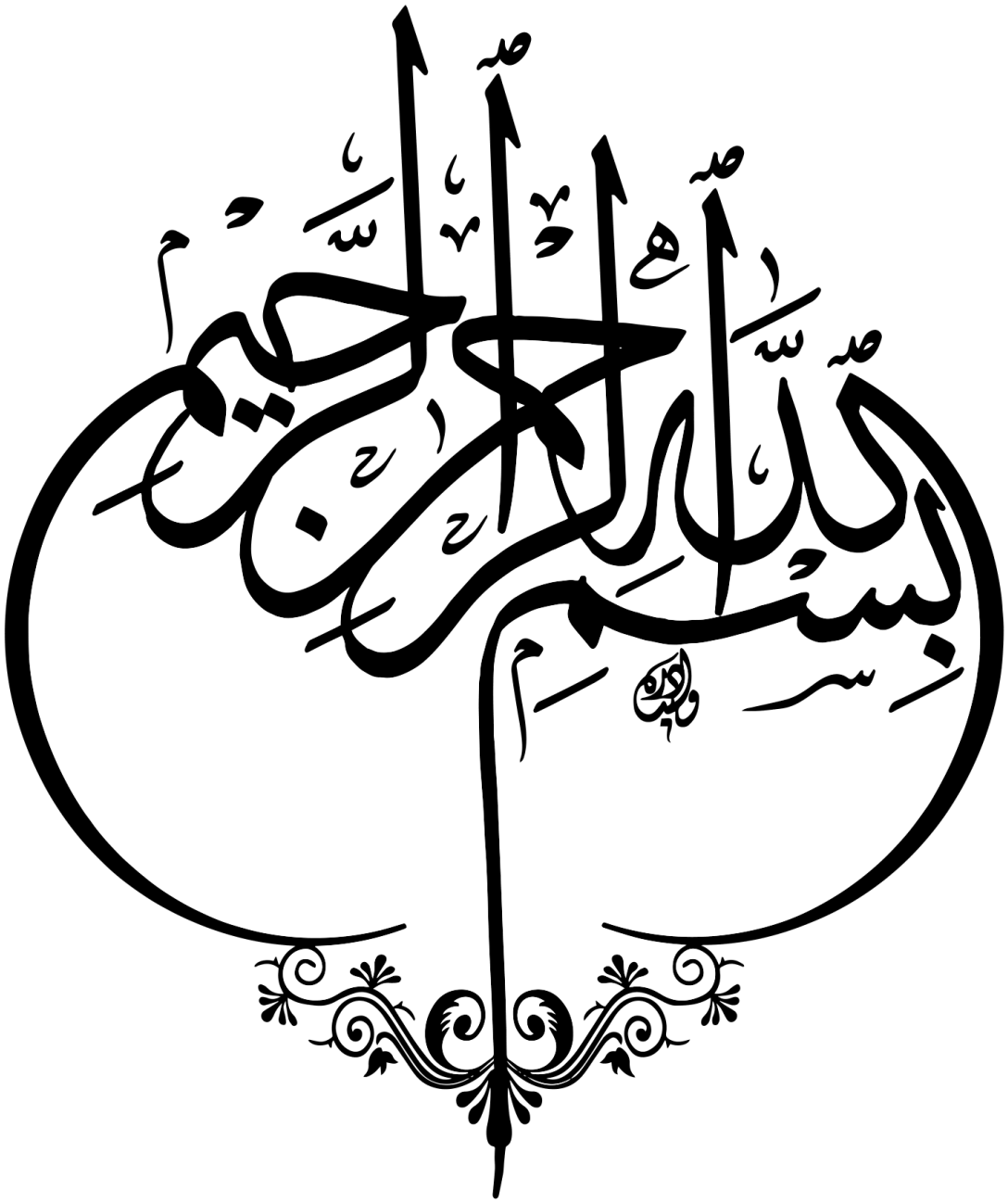
Amel Cheifa
Rim Miloudi

Supervised by:

Lakhdar Oulad Djedid

Academic Year

2024 / 2025


بسم الله الرحمن الرحيم

Dedications

With heartfelt gratitude and profound appreciation,
I would like to dedicate this thesis,

To my loving mother and father,
whose unwavering love, wisdom, and support have been my guiding light.
Your sacrifices, strength, and constant encouragement have shaped every step of my journey.
You have always been my true inspiration and guiding stars, illuminating my path with faith, compassion, and unconditional belief in my dreams.
This achievement is as much yours as it is mine.

To my wonderful siblings,
for their patience, unwavering love, boundless encouragement, and constant belief in me—even when the road was uncertain.
Your presence has always been a comforting reminder that I am never alone on this journey.

To my precious nephew and nieces, whose laughter, curiosity and unconditional affection bring light and joy to my life every single day.

To my cherished friends who stayed with me throughout all my years at university: your unwavering support, late-night study sessions, shared laughter, and countless memories made this journey not only bearable but truly unforgettable.

And lastly, to my best friend for being there in every tear, complaint, shout, laugh, and every step of this journey.

To all those who love me.
To all those I love.

Amel Cheifa

Dedications

With heartfelt gratitude and profound appreciation,
I would like to dedicate this thesis,

To my parents,
Your sacrifices, strength, and constant encouragements This achievement is as much yours as it is mine.

To my wonderful siblings, cousins and any family member.
for their patience, unwavering love, boundless encouragement, and constant belief in me—even when the road was uncertain.
Your presence has always been a comforting reminder that I am never alone on this journey.

To my cherished friends,
who stood by me through all my years at university—
your unwavering support, late-night study sessions, shared laughter, and countless memories made this journey not only bearable but truly unforgettable.
Your friendship has been a pillar of strength and a constant source of joy throughout this academic path.

To my grandmother, who would have been the proudest of me—may she rest in peace.

To my unwavering rock and dear friend, Bruno Ruiz Martinez.
For standing by my side for the last nerving minute.
words can't bring justice for all the appreciation of your support and encouragement.

And lastly, to my best friend—for being there in every tear, complaint, shout, laugh, and every step of this journey.
You've put all of you in this and i am the biggest fan of you for that.

To all those who love me.
To all those I love.

Rim Miloudi

Acknowledgments

First and foremost, no volume of words is enough to express praise to Allah Almighty. All praise is due to Allah, the Lord of the Worlds.

We extend my gratitude towards our supervisor, Dr. Lakhdar OULAD DJEDID, who guided us throughout this research work and helped us with his valuable advice and encouragement.

We would also like to extend our profound gratitude to certain teachers who contributed to our education over the past years.

Finally, We thank all those who, directly or indirectly, contributed to the realization of this work.

Our sincere thanks go to the committee members: Prof. haker Abdelaziz KERRACHE, Dr. Tahar BENDOUMA, and Dr. ahar ALLAOUI, for accepting to review our work.

Amel Cheifa, Rim Miloudi June 2025

Abstract

This thesis proposes a novel Q-learning-based (MAC) protocol to address the challenge of medium congestion control in Wireless Sensor Networks (WSNs) for smart farming applications, where traditional static or centralized MAC protocols often fail to adapt to scalability and resource-constrained agricultural environments. By integrating reinforcement learning into a (TDMA) framework, the protocol enables sensor nodes to autonomously learn optimal transmission strategies, dynamically selecting time slots based on environmental feedback to minimize collisions and improve efficiency. The decentralized learning mechanism allows each node to maintain a Q-table, iteratively refining slot selection to enhance throughput and reduce energy consumption. Simulation results demonstrate significant improvements in collision probability, throughput, and fairness compared to conventional (TDMA) and contention-based protocols, making the approach particularly effective in dense deployments. The protocol's adaptability and scalability highlight its suitability for rural agricultural settings, where reliability and energy efficiency are critical.

Keywords : Wireless Sensor Networks (WSNs), Smart Farming, Medium Access Control (MAC), Q-learning, Reinforcement Learning (RL), Congestion Control, (TDMA).

ملخص

تقترح هذه الأطروحة بروتوكول جديدًا قائمًا على التعلم التعزيزي لحل مشكلة التحكم في الازدحام في شبكات المستشعرات اللاسلكية المطبّقة في الزراعة الذكية، حيث تفشل البروتوكولات التقليدية، التي غالبًا ما تكون ثابتة أو مركزية، في التكيف مع قابلية التوسع والموارد المحدودة التي تميز المجالات الزراعية. من خلال دمج التعلم التعزيزي ضمن بنية ، يمكّن هذا البروتوكول العقد المستشعرة من تعلم استراتيجيات إرسال مثلى بشكل ذاتي، من خلال اختيار الفترات الزمنية الديناميكية بناءً على التغذية الراجعة لتقليل التصادمات وتحسين الكفاءة. وقد تم تصميم نموذج النظام خصيصًا ليتناسب مع الزراعة الذكية، مع دمج خصائص تكيفية مثل فترات الحماية ونوافذ التنافس لمواجهة التغيرات في ظروف الشبكة. تُظهر نتائج المحاكاة تحسناً كبيراً في معدل التصادم، وعرض النطاق، والعدالة مقارنة ببروتوكولات التقليدية وتلك المعتمدة على التنافس، مما يجعل هذا النهج فعّالاً بشكل خاص في البيئات ذات الكثافة العالية. وتؤكد قابلية هذا البروتوكول للتكيف والتوسع على ملاءمته للبيئات الزراعية الريفية، حيث تُعد الموثوقية وكفاءة استهلاك الطاقة من المتطلبات الأساسية.

الكلمات المفتاحية: WSNs, Smart Farming, (MAC) , Q-learning, RL , Congestion Control, (TDMA).

Résumé

Cette thèse propose un nouveau protocole MAC basé sur l'apprentissage Q (Q-learning) afin de relever le défi du contrôle de la congestion du médium dans les réseaux de capteurs sans fil (WSN) destinés aux applications agricoles intelligentes. Dans ces environnements agricoles dynamiques et contraints en ressources, les protocoles MAC traditionnels, qu'ils soient statiques ou centralisés, peinent souvent à s'adapter à l'échelle et à la variabilité des conditions. En intégrant l'apprentissage par renforcement dans une architecture TDMA, le protocole permet aux nœuds capteurs d'apprendre de manière autonome des stratégies de transmission optimales, en sélectionnant dynamiquement des créneaux temporels en fonction des retours de l'environnement, afin de réduire les collisions et d'améliorer l'efficacité. Ce mécanisme d'apprentissage décentralisé permet à chaque nœud de maintenir une table Q, affinant progressivement le choix des créneaux pour maximiser le débit et réduire la consommation énergétique. Les résultats de simulation montrent des améliorations significatives en termes de probabilité de collision, de débit et d'équité par rapport aux protocoles TDMA conventionnels et ceux basés sur la contention, ce qui rend l'approche particulièrement efficace dans les déploiements à forte densité. L'adaptabilité et la scalabilité du protocole soulignent sa pertinence pour les contextes agricoles ruraux, où la fiabilité et l'efficacité énergétique sont essentielles.

Mots Clés : Réseaux de capteurs sans fil (WSN), Agriculture intelligente, Contrôle d'accès au support (MAC), Q-learning, Apprentissage par renforcement, Contrôle de congestion, TDMA.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AI Artificial Intelligence. 16

CSMA Carrier Sense Multiple Access. 16, 28

DQN Deep Q-Network. 72

DRL Reinforcement Learning. 72

IOT Internet of Things. 20, 22

MAC Medium Access Control. 5–7, 16, 38, 71, 72

RL Reinforcement Learning. 5, 6

TDMA Time Division Multiple Access. 5–7, 28, 46, 71

WSN Wireless sensor network. 5–7, 16, 17, 21–24, 38, 71

Chapter 1

Introduction

1.1 Motivation

Smart farming transforms traditional agriculture by integrating technologies like WSN, IoT, cloud computing, and AI to enable real-time monitoring, automation, and data-driven decision-making. This approach enhances crop management, improves resource efficiency, and increases resilience to climate change. It supports precision irrigation, early disease detection, and optimized fertilization—critical for food security amid limited land, water scarcity, and labor shortages. Especially in developing regions, smart farming can boost yields and efficiency, but must be tailored to rural constraints such as poor connectivity, limited energy, and lack of infrastructure.

1.2 Problematic

Despite its potential, large-scale smart farming faces major communication challenges. Frequent packet collisions over shared wireless media cause data loss, energy waste, and delays—issues worsened by rural constraints like limited power and unreliable connectivity. Traditional MAC protocols, often static and centralized, are poorly suited to these environments. Security is also a growing concern, with remote, heterogeneous devices exposed to threats and difficult to maintain. This thesis addresses these issues by proposing intelligent, decentralized MAC solutions using Q-learning and Deep Q-Networks (DQN), enabling nodes to learn optimal, context-aware communication strategies.

1.3 Objective

This thesis aims to design an intelligent and adaptive communication strategy for smart farming using Q-learning. The goal is to reduce packet collisions, improve energy efficiency, and ensure reliable data transmission in rural, resource-constrained environments. By allowing sensor nodes to learn optimal transmission behaviors autonomously, Q-learning offers a decentralized alternative to traditional MAC protocols, enhancing scalability, fairness, and network lifetime. The approach is evaluated through simulations using metrics such as collision probability, throughput, fairness, energy consumption, and delivery success.

1.4 Organization of the Thesis

The organization of this thesis is structured as follows :

- **Chapter two** (2) provides background on smart farming, wireless sensor networks (WSN), medium access control (MAC) protocols, and machine learning techniques, with a focus on reinforcement learning.
- **Chapter three** (3) reviews related work on MAC protocols and AI-based enhancements, then introduces the proposed Q-learning-based Dynamic Slotted-CSMA protocol. It details the system model, algorithm design, and simulation setup used to evaluate its effectiveness.
- **Chapter four** (4) This chapter explains the simulation environment and parameters used to evaluate the proposed protocol. It presents performance metrics such as average medium collision probability, throughput,Beacon Delivery, and access fairness. Comparative results with existing MAC protocols are analyzed to demonstrate the improvements achieved by the Q-learning approach.
- Finally, **Chapter five** (5) • wraps up the thesis by summarizing the key findings and offering recommendations for future research.

Chapter 2

Background

2.1 Introduction

Chapter 2 presents an overview of smart farming systems, emphasizing the need for real-time monitoring and data collection to address challenges in modern agriculture. It highlights the role of Wireless Sensor Networks (WSNs) as a key enabler of efficient and sustainable farming practices.

WSNs support smart agriculture by continuously monitoring critical parameters such as soil moisture, temperature, and humidity. This enables data-driven decision-making that enhances productivity, resource efficiency, and environmental sustainability. The chapter also sets the stage for deeper exploration of WSNs in later sections.

2.2 Smart Farming

2.2.1 Precision Agriculture

Precision agriculture tailors agronomic practices to specific field conditions by leveraging spatial and temporal variations. It aims to optimize the use of resources—water, fertilizers, seeds, and pesticides—thereby improving yield, reducing waste, and conserving the environment.

WSNs play a crucial role by enabling continuous data collection through low-power wireless sensors distributed across fields. These sensors track environmental factors, crop growth, and livestock health, supporting informed and timely decisions [1].

Modern precision agriculture relies on the ”3S” technologies:

- **Remote Sensing (RS)**
- **Geographic Information Systems (GIS)**
- **Global Positioning Systems (GPS)**

These tools, combined with AI techniques, enable accurate crop monitoring and predictive analytics, reducing reliance on traditional, experience-based farming [7].

2.2.1.0.1 Key Benefits:

- Reduced environmental pollution through controlled pesticide use

- Improved irrigation efficiency, minimizing water waste
- Optimized land use and ecological sustainability
- Enhanced crop yield and quality via growth pattern analysis

2.2.1.0.2 Security Challenges:

- Sensor vulnerability to eavesdropping, tampering, and harsh environments
- Risk of location spoofing, potentially disrupting operations

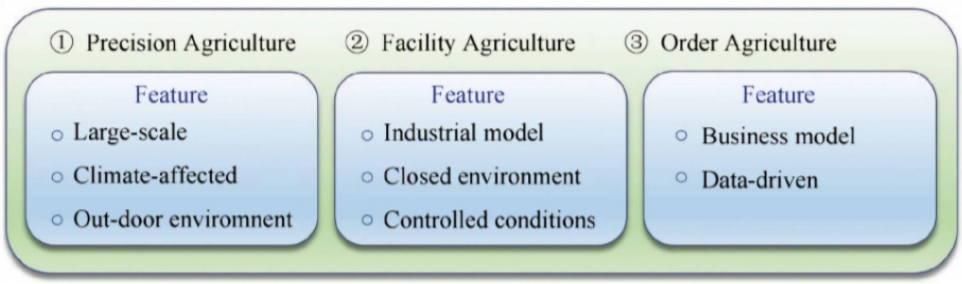


Figure 2.1: Development modes of smart agriculture[1].

2.2.2 Facility agriculture

2.2.3 Facility Agriculture

Facility agriculture focuses on high-quality, high-yield production within controlled environments. Unlike traditional methods, it leverages capital, engineering technologies, and automation to overcome environmental limitations and seasonal variations [8]. Technologies such as biotechnology, IoT, meteorology, and computer control are integrated to manage variables like temperature, humidity, light, and fertilizer in real-time—enabling optimal crop growth in intelligent greenhouses and similar systems.

Facility agriculture, including horticulture and breeding, relies on predictive models and decision-support systems based on historical and genetic data. Though highly efficient, its centralized and automated nature introduces vulnerabilities to cyber threats and unauthorized control system access.

2.2.3.0.1 Key Benefits:

- Year-round production independent of climate.
- Reduced agricultural cycles.
- Enhanced yield and quality.
- Efficient resource usage via intelligent control.

2.2.4 Order Agriculture

Order agriculture addresses rural-urban development gaps—particularly in China—by offering a market-driven production model. It tackles key rural challenges: weak agricultural foundations, safety risks in food quality, and fragmented agricultural information systems [9, 10].

This model involves contracting agricultural production in advance, reducing risks and preventing overproduction [11]. It is supported by traceability systems and supply chain transparency mechanisms using e-commerce and blockchain [12].

2.2.4.0.1 Key Functions:

1. Improves transparency with blockchain-based traceability.
2. Eliminates information silos in agricultural trade.
3. Balances supply and demand.
4. Builds trust between producers and consumers.

Examples include Leng et al.'s dual-chain agricultural supply system [13] and Hua et al.'s blockchain-based provenance system [14], both of which enhance food safety, traceability, and market efficiency.

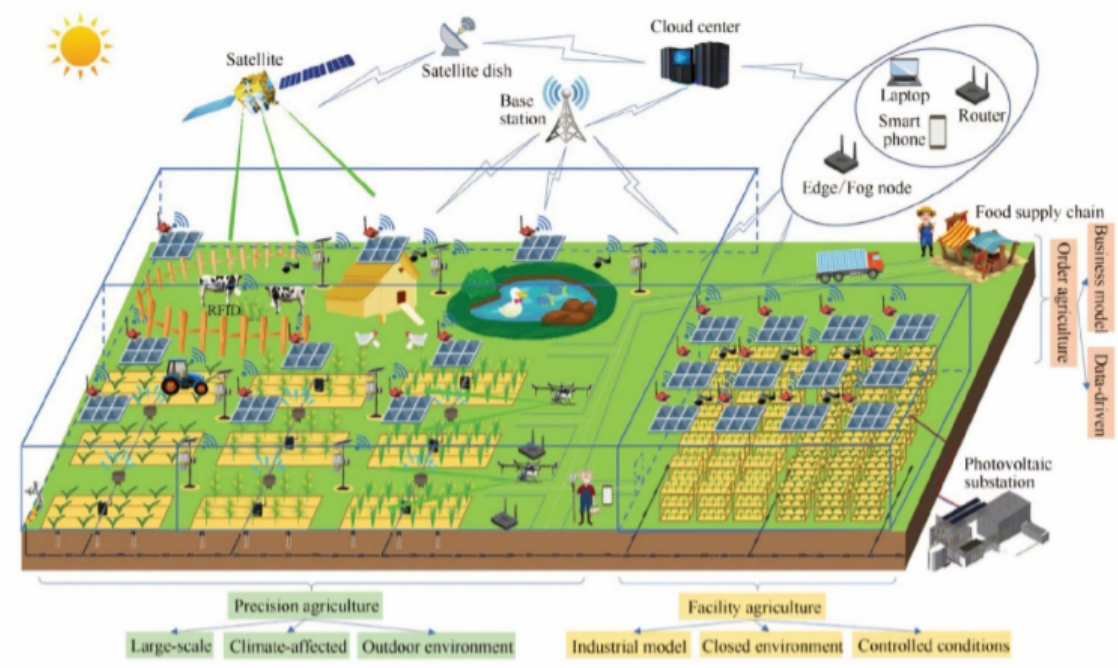


Figure 2.2: Research fields of smart agriculture based on 1) precision agriculture, 2) facility agriculture, and 3) order agriculture[1].

Table 2.1: Three Typical Development Modes of Smart Agriculture[1].

Type	Country	Feature	Trends	Characteristics of type
Precision agriculture[4]	U.S.	Less population, more land, and developed industry	Develop field agriculture, replace manpower with mechanization, and decrease manpower operation.	Large-scale, climate affected, outdoor environment
Facility agriculture[15]	Japan	More population and less land, scientific and technological progress	Improve crop genes, improve production conditions, and improve the land utilization rate.	Industrial model, closed environment, controlled conditions
Order agriculture[5]	Western Europe	Moderate population and land, developed economy	Increase agricultural scale, develop agricultural management, increase output rate.	Business model, data-driven

2.2.5 Summary

Precision agriculture, facility agriculture, and order agriculture represent the primary developmental paradigms of smart agriculture. These paradigms integrate traditional agricultural practices conducted outdoors with innovative agricultural methods employed indoors, alongside the agricultural product industry chain, utilizing advanced technologies. The diverse applications of smart agriculture derived from these developmental modes enhance the yield and quality of grain and other agricultural products, decrease production costs, and promote transparency in the trading of agricultural goods.[1]

2.3 Wireless Sensor Network in the Internet of Things (WSN- IoT)

2.3.1 Difference Between WSN and IoT

The designation Internet of Things, abbreviated as IOT, is experiencing an increasing degree of interest within society. It constitutes a comprehensive notion that encompasses not only sensors but a wide array of devices. These characteristics enable the devices to interconnect and exchange data through the Internet. Of paramount significance, each device is allocated an Internet Protocol address (ID)[16]. Such devices represent hardware characterized by considerable diversity. For instance, it may encompass anything from economically accessible consumer electronics (such as automobiles and domestic appliances) to extensive industrial infrastructure. Notably, the trajectory of this phenomenon is projected to result in a data volume that surpasses the cumulative total of all cellular infrastructure[2].

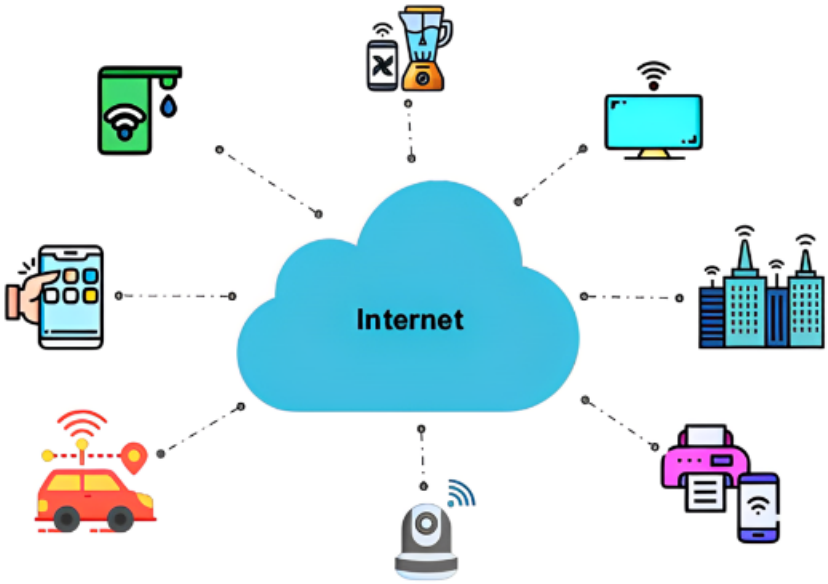


Figure 2.3: Internet of Things[2].

WSN refers to Wireless Sensor Network. It consists of multiple wireless sensor agents for measuring diverse environmental and physical parameters in designated or inadvertently covered areas. Each sensor node gathers local data, processes it, and wirelessly transmits it to a central node, known as the base station (BS) or sink node, thereby serving as a data collection element. Sensor nodes relay information about their vicinity until reaching the gateway, where the aggregated data is collated[17]. The central node is linked to the internet for public dissemination of the phenomenon. The data amassed by sensor nodes is applicable in various fields, including environmental monitoring, healthcare, and industry. The design of the WSN is elaborated upon.

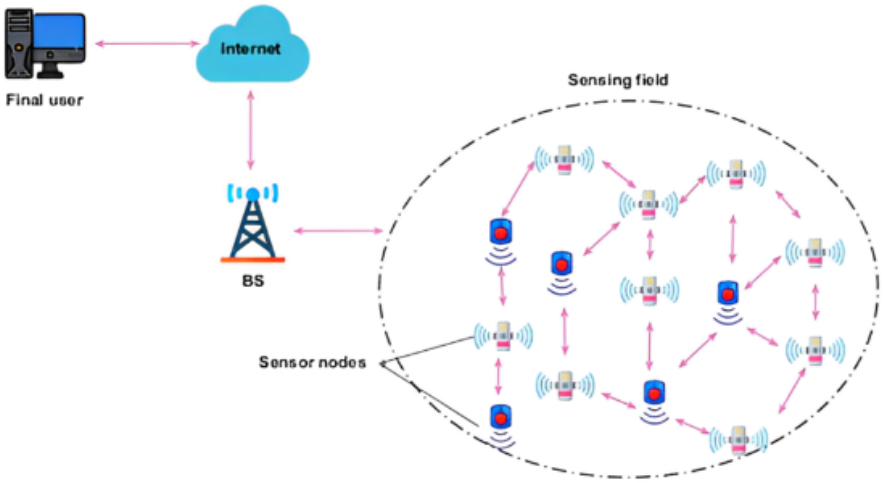


Figure 2.4: Wireless sensor network[2].

A WSN is a rapidly expanding component of the evolving IoT sector. It plays a crucial role

in the IOT by transmitting data through various sensors. This data is subsequently prepared for analysis and decision-making. In IoT-based WSNs, sensor nodes diligently monitor their surroundings and continually report to the base station via wireless connections upon the occurrence of an event[2, 18].

2.3.2 Characteristics

WSN is increasingly essential for autonomous parameter measurement in real environments. Therefore, efficient deployment necessitates consideration of WSN characteristics. The subsequent section delineates the significant features of WSN[19, 20].

- **Cost efficiency:** The primary objective of WSN is to optimize the placement of numerous sensing nodes to minimize individual node costs, thereby reducing overall network expenses.
- **Energy optimization:** The energy-intensive operations of WSN are concentrated in sensor nodes, which predominantly consume power during processing, transmission, and storage functions.
- **Communication Capabilities:** Wireless Sensor Networks (WSNs) facilitate both unidirectional and bidirectional wireless communication, which necessitates the use of radio waves as the medium for transmission.
- **Security and Privacy:** It is imperative to implement robust security measures at the sensor level to safeguard against inadvertent data corruption and unauthorized modifications. Such measures are essential to ensure the integrity and confidentiality of the data stored within the system.
- **Distributed Sensing and Processing:** The spatial arrangement of sensor nodes within a Wireless Sensor Network (WSN), whether uniformly or randomly distributed, facilitates distributed sensing and computational processes. This configuration not only promotes system stability but also enhances the efficiency of data collection, organization, and processing.
- **Dynamic Topology of the Network:** WSNs are characterized by their inherently dynamic nature, necessitating that nodes be equipped with mechanisms capable of self-adaptation and reconfiguration in response to changes in communication channels or sensor malfunctions.
- **Self-Organization:** Given that sensor nodes are often deployed in remote locations that are infrequently visited or previously unmonitored, it is imperative that they operate in a coordinated and self-organized manner. This enables them to collaboratively design and autonomously adapt their distributed algorithms.
- **Multi-Hop Communication:** The implementation of multi-hop communication is essential, as it allows nodes to relay data across effective pathways to distant points that may exceed the limitations of direct radio frequency transmission.
- **Application-specific Design:** Wireless Sensor Networks (WSNs) are tailored to meet the specific requirements of various applications, necessitating a random distribution of nodes to effectively address the unique demands of each use case.

- **Resilient Functionality:** Sensor nodes must exhibit error and fault tolerance, incorporating features such as self-testing, self-calibration, and self-repair capabilities. This resilience is essential for ensuring sustained operation in extensive and challenging environmental conditions.
- **Compact Physical Dimensions:** The operational scope of WSNs is inherently constrained, resulting in the development of miniature and lightweight nodes. This compactness limits their energy and power availability for communication purposes[20].

2.3.3 Network Applications

Sensor nodes are capable of detecting or monitoring a diverse array of physical factors or conditions, including but not limited to light, sound, pressure, temperature, humidity, soil composition, and the quality of air or water. Additionally, they can assess the characteristics of objects, such as size, weight, location, speed, and direction. The versatility of these sensors allows for their deployment in various environments, while the advent of low-cost sensors and wireless communication technologies has contributed to a reduction in both implementation costs and delays. Consequently, Wireless Sensor Networks (WSNs) have found numerous applications across both civilian and military domains[2].

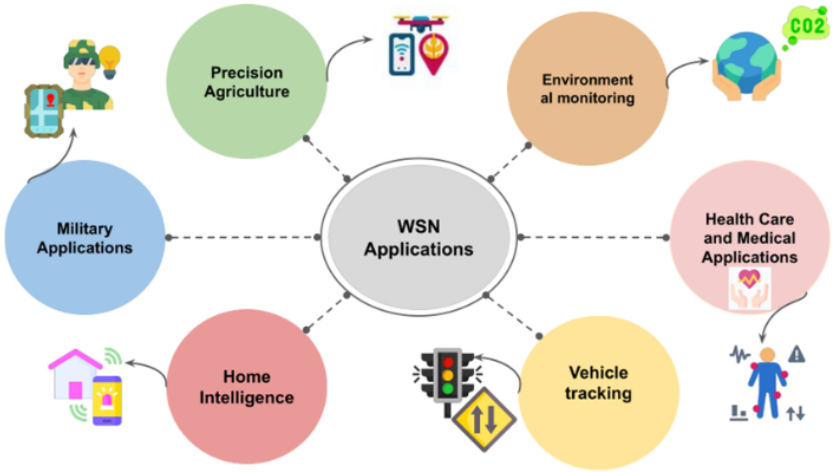


Figure 2.5: Wireless sensor network applications[2]

- **Precision Agriculture:** WSNs monitor soil, humidity, and temperature to optimize seed density, irrigation, fertilizer use, and yield prediction [21].
- **Environmental Monitoring:** WSNs detect wildfires, floods, gas leaks, and pollutants (e.g., CO_2) in hazardous or remote environments [21].
- **Health Care:** Ingestible or implantable sensors gather physiological data (e.g., glucose, pH, heart rate) for continuous monitoring and early disease detection [22].
- **Military:** WSNs aid in surveillance, threat detection, and monitoring enemy movements. They support self-configuration, fault tolerance, and rapid deployment [23].
- **Home and Transport:** WSNs enable smart metering and monitor traffic patterns to detect violations and ease congestion [21, 24].

2.3.4 Sensor Unit Architecture

A sensor node includes four main units: sensing (with ADC), processing, transceiver, and power (usually a battery) [2, 25]. Additional modules (e.g., GPS, motors) may be added depending on application needs. Lightweight OSs like TinyOS, Contiki, and RIOT support communication and energy management.

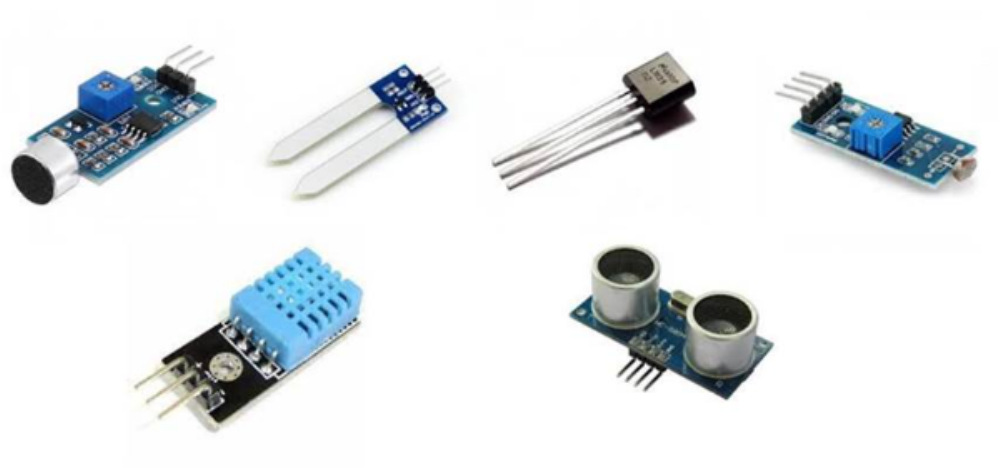


Figure 2.6: Sensors [2]

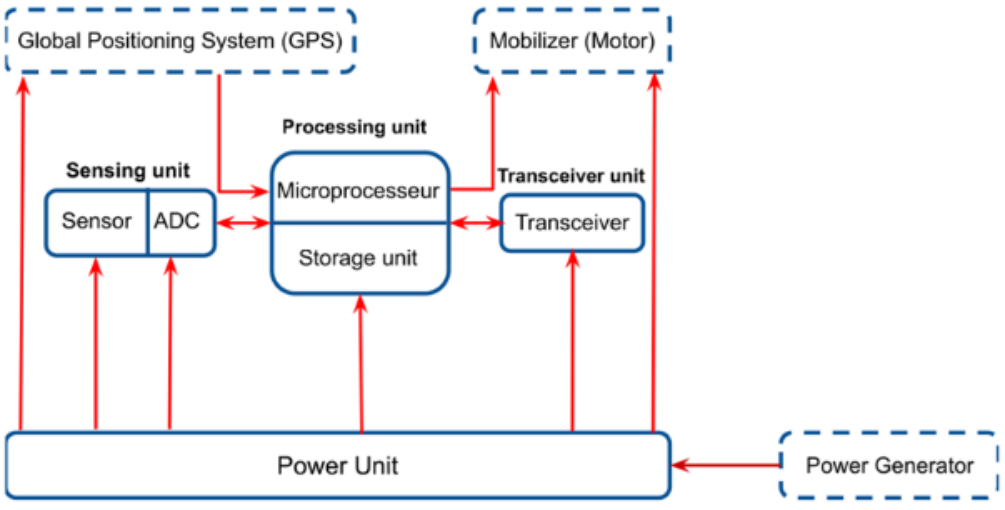


Figure 2.7: Sensor Units [2]

2.3.5 Network Architecture

A WSN consists of sensor nodes and one or more base stations. Data is transmitted using either single-hop or multi-hop paths [26].

- **Flat Architecture:** All nodes are equal in capability. Data reaches the sink via multi-hop routing through neighbors. This is simple and cost-effective, but energy-intensive for large networks.

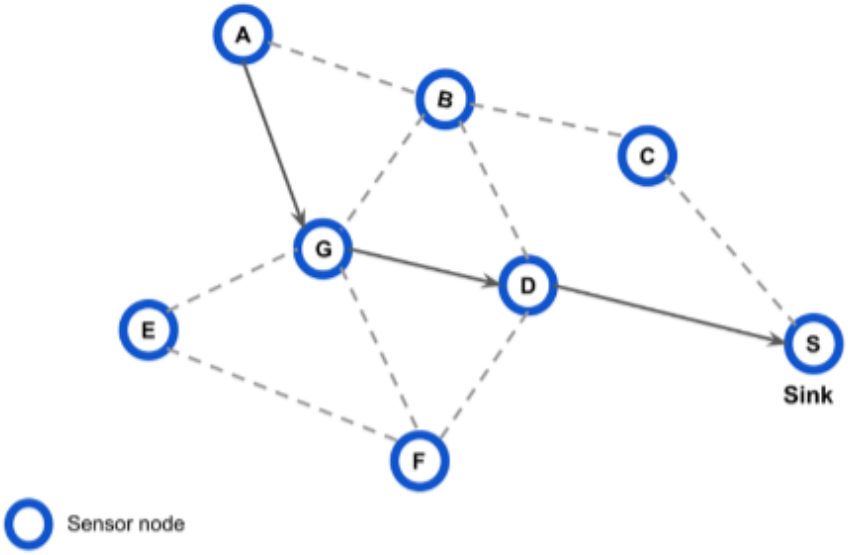


Figure 2.8: Flat Architecture [2]

- **Hierarchical Architecture:** Nodes form clusters. Each cluster has a Cluster Head (CH) that aggregates and forwards data to the sink, reducing transmissions and improving scalability.
  - Nodes: sense, process, and transmit data.
  - Clusters: logical groups of nodes.
  - CHs: manage intra-cluster data and relay it to the sink.
  - Base Station: central hub with high power and processing capacity.
  - End Users: access data via Internet or local interfaces.

2.3.6 Protocol Stack

WSN protocol stacks are simplified versions of the OSI model, reduced to five layers for efficiency: Application, Transport, Network, Data Link, and Physical layers [2]. Management planes (power, mobility, task) support energy savings and node coordination.

2.3.7 IoT Architecture in Agriculture

Smart farming relies on IoT, divided into:

- **Perception Layer:** Sensors gather environmental and crop data.
- **Transmission Layer:** Transfers data to upper layers.
- **Application Layer:** Handles storage (cloud/edge), monitoring (e.g., SCADA), and analysis (e.g., decision models) [1].

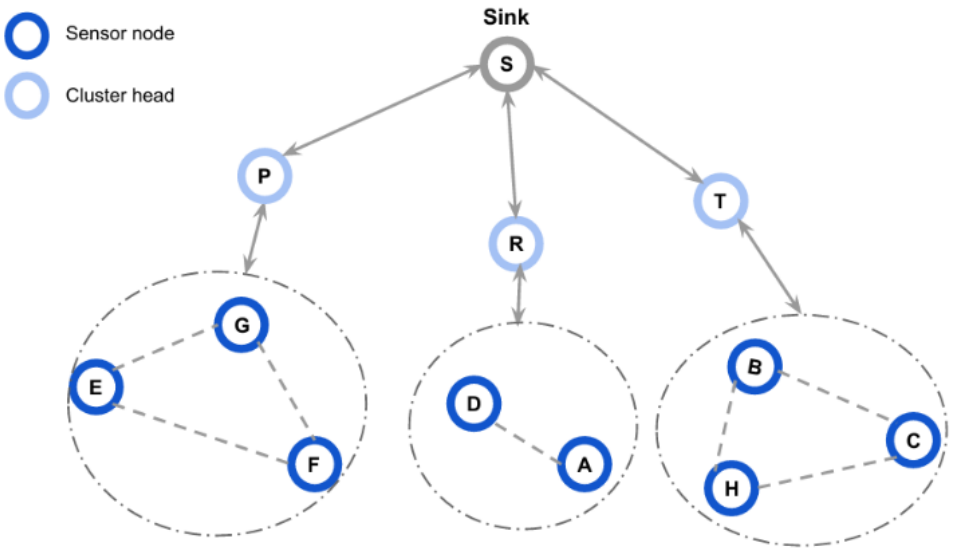


Figure 2.9: Hierarchical Architecture [2]

2.3.8 Medium Access Control (MAC) Protocols

MAC protocols manage how nodes share the wireless medium. They are grouped into:

1. **Contention-Based MAC (CSMA/CA)** [3]:

- **Carrier Sensing:** Nodes listen before transmitting (Clear Channel Assessment).
- **Back-off:** If busy, nodes wait a random delay before retrying.
- **Retransmission:** Failed unicast messages are retried with increasing delay. Broadcasts use a single back-off and no ACKs.

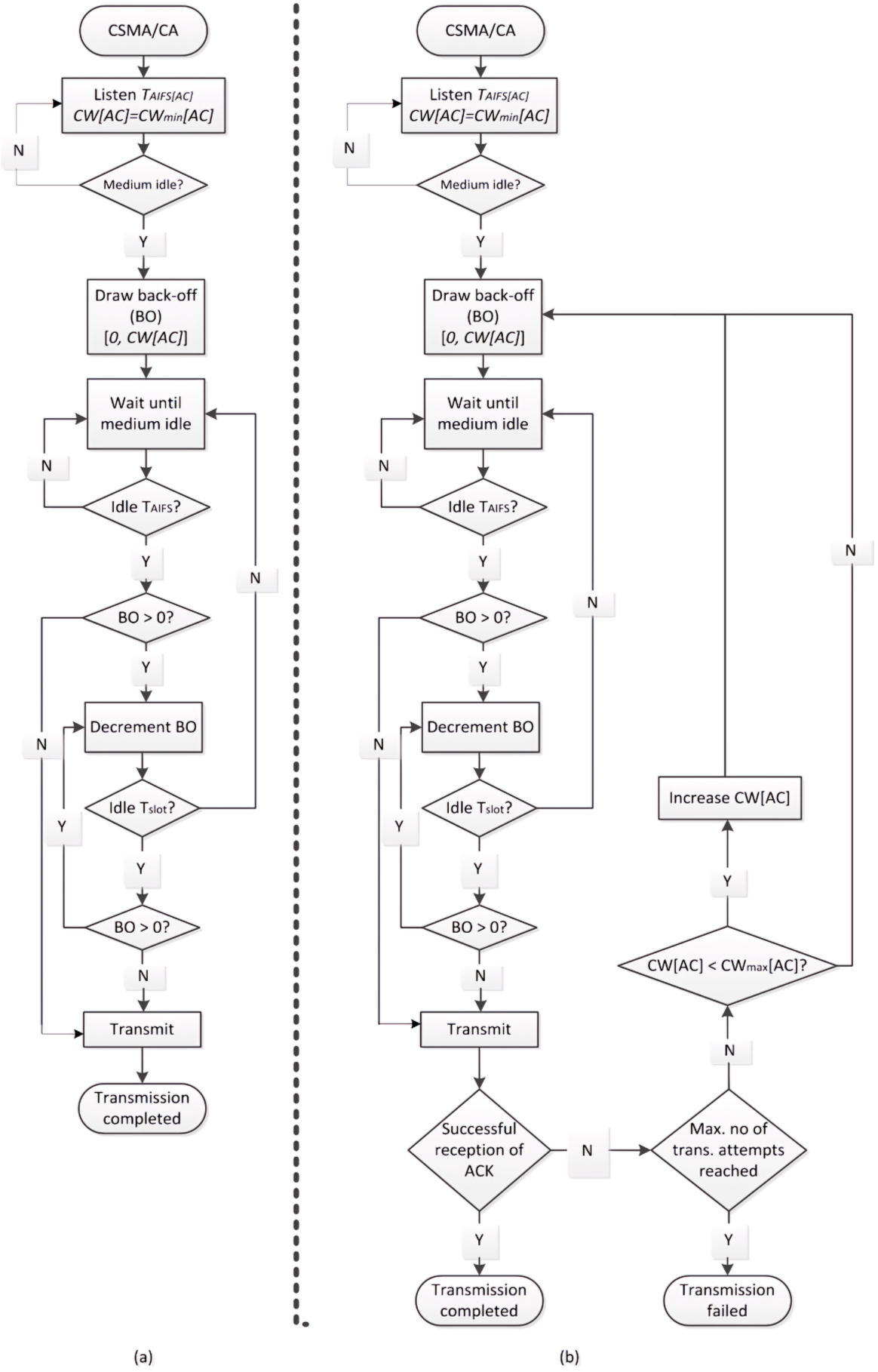


Figure 2.10: CSMA channel access procedure ; (a) broadcast mode, and (b) uni-cast modecite[3]

- **Scheduled-Based MAC (TDMA-based):** Assigns fixed time slots per node to reduce energy use and avoid collisions. Radios sleep when inactive. While energy-efficient, it requires tight synchronization and may cause latency as node count increases [15].
- **Hybrid MAC:** Merges contention-based (e.g., CSMA/CA) and schedule-based (e.g., TDMA) methods. Adapts to traffic: uses CSMA under low load and TDMA during high load to balance efficiency and flexibility [27].
- **IEEE 802.15 – Wireless Specialty Networks:** Develops global standards for WPANs, Bluetooth, IoT, mesh, wearables, and OWC. It ensures interoperability, including newer domains like Terahertz (THz) and optical wireless communications [28].

1. **Standards :** The IEEE 802.15 Working Group has established a series of standards for WPANs and other short-range wireless communication technologies. Below is an overview of some of the key standards within this group:

Table 2.2: IEEE 802.15 Standards Overview

Standard	**Description**
IEEE 802.15.1	Specifies MAC and PHY layers for Wireless Personal Area Networks (WPANs), defining access rules and transmission characteristics.
IEEE 802.15.2	Focuses on coexistence mechanisms for WPANs operating in unlicensed frequency bands.
IEEE 802.15.3	Designed for high-data-rate WPANs, suitable for multimedia transmission.
IEEE 802.15.4	A widely adopted low-rate WPAN standard for energy-constrained sensor networks.
IEEE 802.15.5	Enables mesh networking for WPANs, supporting multi-hop communication to extend coverage.
IEEE 802.15.6	Defines Wireless Body Area Networks (WBANs) for communication among wearable medical devices.
IEEE 802.15.7	Utilizes visible light for short-range optical wireless communication, as an alternative to RF.
IEEE 802.15.8	Establishes peer-to-peer communication protocols without central coordination.
IEEE 802.15.9	Introduces a key management protocol for secure data authentication and encryption in WPANs.
IEEE 802.15.10	Addresses Layer 2 routing techniques for packet forwarding in dynamic IEEE 802.15.4-based networks.
IEEE 802.15.4 PHY	Provides 11 channels at 868/915 MHz and 16 channels at 2.4 GHz using various modulation techniques to reduce interference.
IEEE 802.15.4 MAC	Manages channel access using CSMA/CA, frame handling, and supports different topologies (star, mesh, cluster).
IEEE 802.15.4e MAC	An enhanced MAC layer using Time Synchronized Mesh Protocol (TSMP) for scheduled multi-hop communication.

1. **Technologies:** This section introduces key wireless communication technologies used in smart farming and IoT systems:

   (a) **ZigBee:** A low-power wireless protocol supporting mesh and cluster tree topologies. ZigBee operates up to 100 m with modest data rates (50–200 kbps), enabling two-

way communication. Developed by the ZigBee Alliance and standardized in 2004, ZigBee 3.0 followed in 2016 [5].

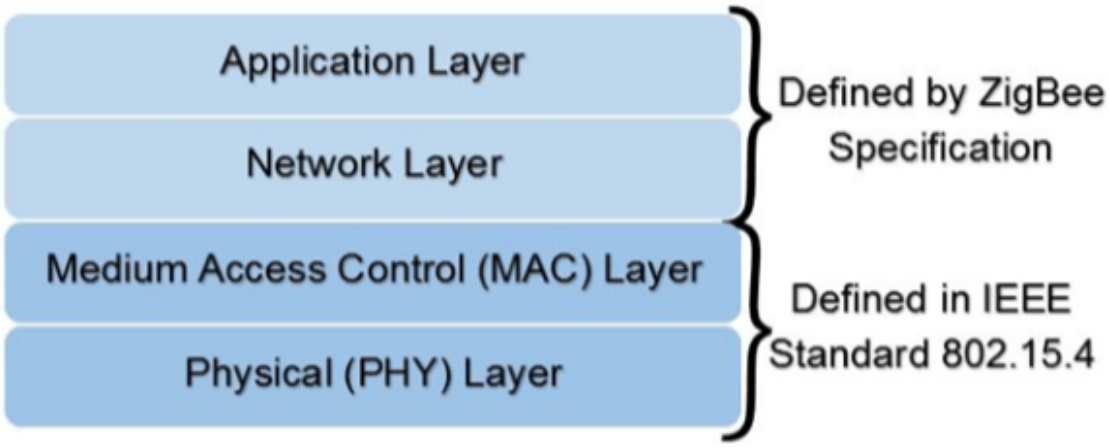


Figure 2.11: IEEE 802.15.4 and ZigBee protocol stack architecture [4]

(b) **Bluetooth:** Bluetooth Classic supports high-quality data/audio transmission, while Bluetooth Low Energy (LE) enables ultra-low-power communication over 40 channels in the 2.4 GHz ISM band. BLE supports point-to-point, broadcast, and mesh topologies and is used in indoor positioning systems [5].

(c) **WirelessHART:** An extension of the HART protocol for industrial wireless communication. It operates in the 2.4 GHz ISM band using a time-synchronized, self-healing mesh network. Devices act as both nodes and repeaters, coordinated via a network manager [29].

(d) **LoRaWAN:** A low-power wide-area network (LPWAN) protocol developed by Semtech. LoRaWAN uses Chirp Spread Spectrum (CSS) modulation and adjustable spreading factors (SF 7–12) for long-range, low-rate data transmission [5].

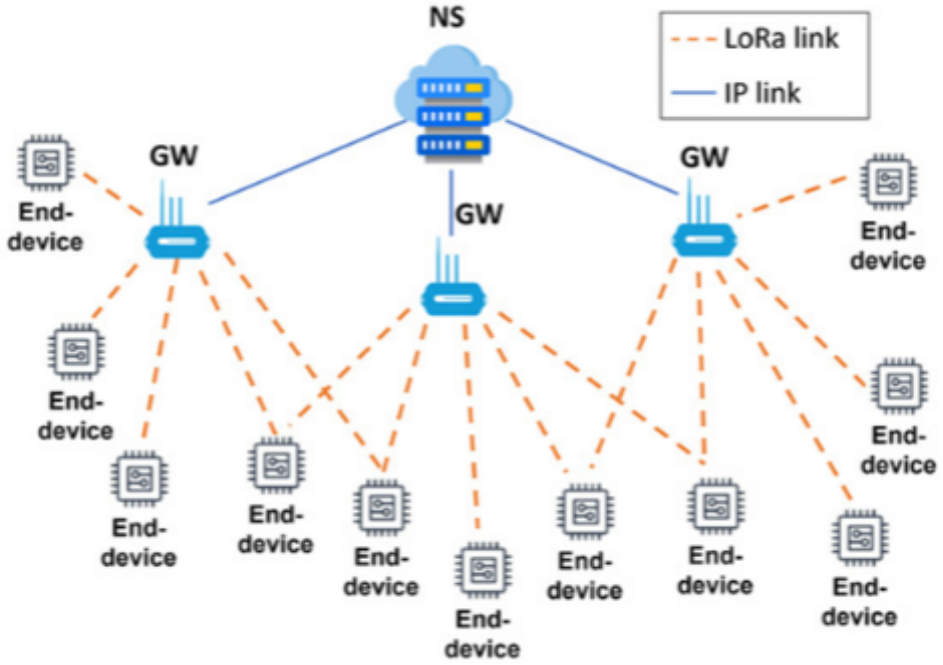


Figure 2.12: The LoRaWAN topology [5]

(e) **nRF24L01:** A 2.4 GHz single-chip transceiver featuring Enhanced ShockBurst™ for ultra-low-power applications. It supports 2 Mbps data rates across 125 channels using GFSK modulation, interfacing via SPI with minimal external components.

2.3.9 Communication Requirements for Smart Farming

Efficient communication in smart farming is shaped by environmental factors and the communication process itself. In wireless sensor networks (WSNs), the Medium Access Control (MAC) layer plays a key role in ensuring reliable data exchange.

2.3.9.1 Performance Requirements

- **Energy Efficiency:** Critical in battery-powered WSNs; achieved via techniques like duty cycling.
- **Latency:** Minimized through fast medium access and effective contention resolution.
- **Fairness:** Ensures equal access to the medium for all nodes using mechanisms like randomized backoff.
- **Scalability:** Maintains performance with increasing nodes, often through hierarchical clustering [3, 30].

2.3.9.2 Challenges

- **Mobility:** Rapid topology changes require decentralized, lightweight protocols [31].
- **Hidden Terminal Problem:** Causes collisions due to undetectable transmitters [32].

- **Scalability and Congestion:** Node density can cause channel congestion and collision.
- **Energy Constraints:** Nodes require energy-efficient hardware and software [33, 34].
- **Security Risks:** WSNs are vulnerable to attacks; secure transmission protocols are essential.

2.3.10 Conclusion

Chapter 2 examined the role of WSNs in smart farming, outlining their architecture, topologies, and core communication protocols—especially at the MAC layer. Technologies like ZigBee, Bluetooth, LoRaWAN, and NRF24L01 were briefly introduced. The chapter provides a strong foundation for the following sections, which will explore WSN-based implementations tailored to smart agriculture.

2.4 Machine Learning: Fundamental and Techniques

2.4.1 Definition

Machine Learning (ML) enables systems to learn from data and improve performance without explicit programming [35]. As defined by Mitchell: a machine learns from experience (E) on task (T) with performance (P) if P improves with E [36].

2.4.2 Types of Machine Learning

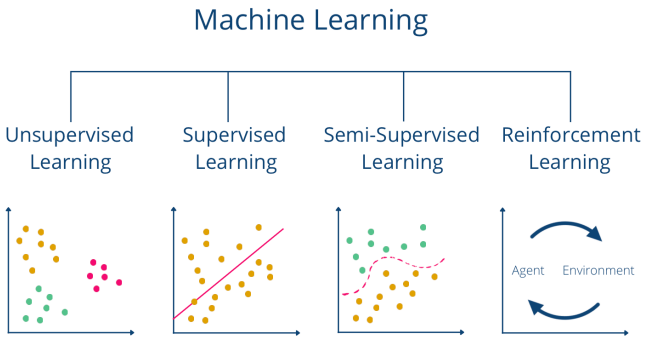


Figure 2.13: Overview of ML categories

1. **Supervised Learning:** Uses labeled data (X, Y) to learn a mapping function $Y = f(X)$ [37].
    - **Classification:** Predicts discrete labels (e.g., spam detection).
    - **Regression:** Predicts continuous outputs.

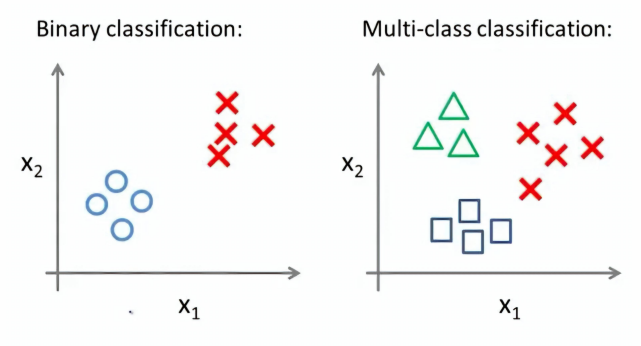


Figure 2.14: Types of classification

2. **Unsupervised Learning:** Analyzes unlabeled data X to identify patterns [37].
   - **Clustering:** Groups similar data points [38].
   - **Association:** Detects variable relationships.
   - **Dimensionality Reduction:** Reduces input features while preserving structure.

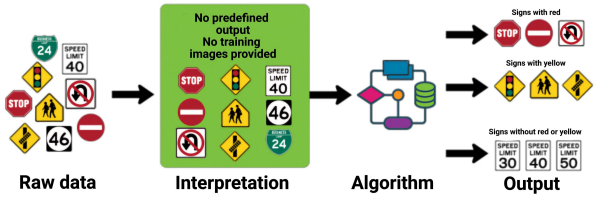


Figure 2.15: Example of unsupervised learning

3. **Reinforcement Learning (RL):** An agent interacts with an environment to maximize cumulative rewards based on trial-and-error [38].
   - **Markov Decision Process (MDP):** Defined by tuple $\langle S, A, R, P \rangle$, where S is states, A actions, R reward, and P transition probability. The agent follows a policy $\pi : S \to A$ to optimize long-term rewards [39].

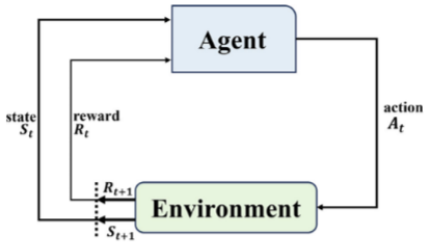


Figure 2.16: Agent–environment interaction in MDP

RL Components and Phases

RL enables agents to learn optimal behavior via interaction with an environment. The process relies on key components and unfolds across several stages [37].

Components:

- **Agent:** Learns and makes decisions autonomously.
- **Environment:** Simulated real-world space reacting to agent actions.
- **State (S):** Current situation perceived by the agent.
- **Action (A):** Possible decisions made by the agent.
- **Reward (R):** Feedback signal indicating action success.
- **Policy (π):** Strategy mapping states to actions.
- **Value Function (V):** Expected cumulative reward from a state.
- **Exploration vs. Exploitation:** Trade-off between trying new actions and maximizing known rewards.

Learning Phases:

1. **Perception:** The agent observes the state and takes an action.
2. **Feedback:** It receives a reward or punishment.
3. **Learning:** It updates its strategy based on the feedback.
4. **Evaluation:** The agent assesses actions for future decisions.

2.4.3 Reinforcement Learning

Taxonomy

Reinforcement Learning (RL) is inspired by how agents learn through interaction and adaptation to achieve goals. It is broadly categorized into *model-based* and *model-free* approaches. In model-based RL, the agent builds an internal model of the environment to plan and make decisions. While effective when the model is accurate, it can struggle with complex or unpredictable tasks. Model-free RL avoids explicit modeling by learning directly from interactions. Agents estimate future rewards to guide decisions, typically using value-based methods like Q-learning or policy optimization. These can be further classified into *round-update* and *step-update* techniques, with Q-learning belonging to the latter.

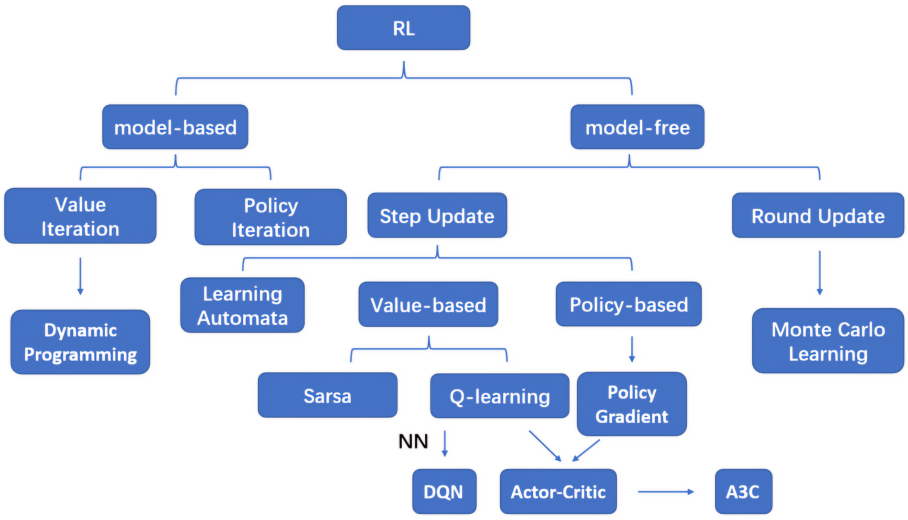


Figure 2.17: A taxonomy of reinforcement learning.[6]

1. Q-Learning Algorithms

Q-learning is a model-free RL algorithm that uses a Q-table $Q(s, a)$ to estimate the expected reward of taking action a in state s. The agent updates this table through trial and error, aiming to learn an optimal policy that maximizes long-term rewards without prior knowledge of the environment [6].

The *actor-critic* method blends value-based techniques (like Q-learning) with policy-based approaches for better learning efficiency. Another notable approach is the *Learning Automaton (LA)*, which learns optimal actions probabilistically by interacting with the environment. LAs are advantageous for scenarios where environmental knowledge is limited.

Recent work applies LAs to congestion-aware MAC protocols and routing decisions, demonstrating their adaptability in dynamic wireless networks.

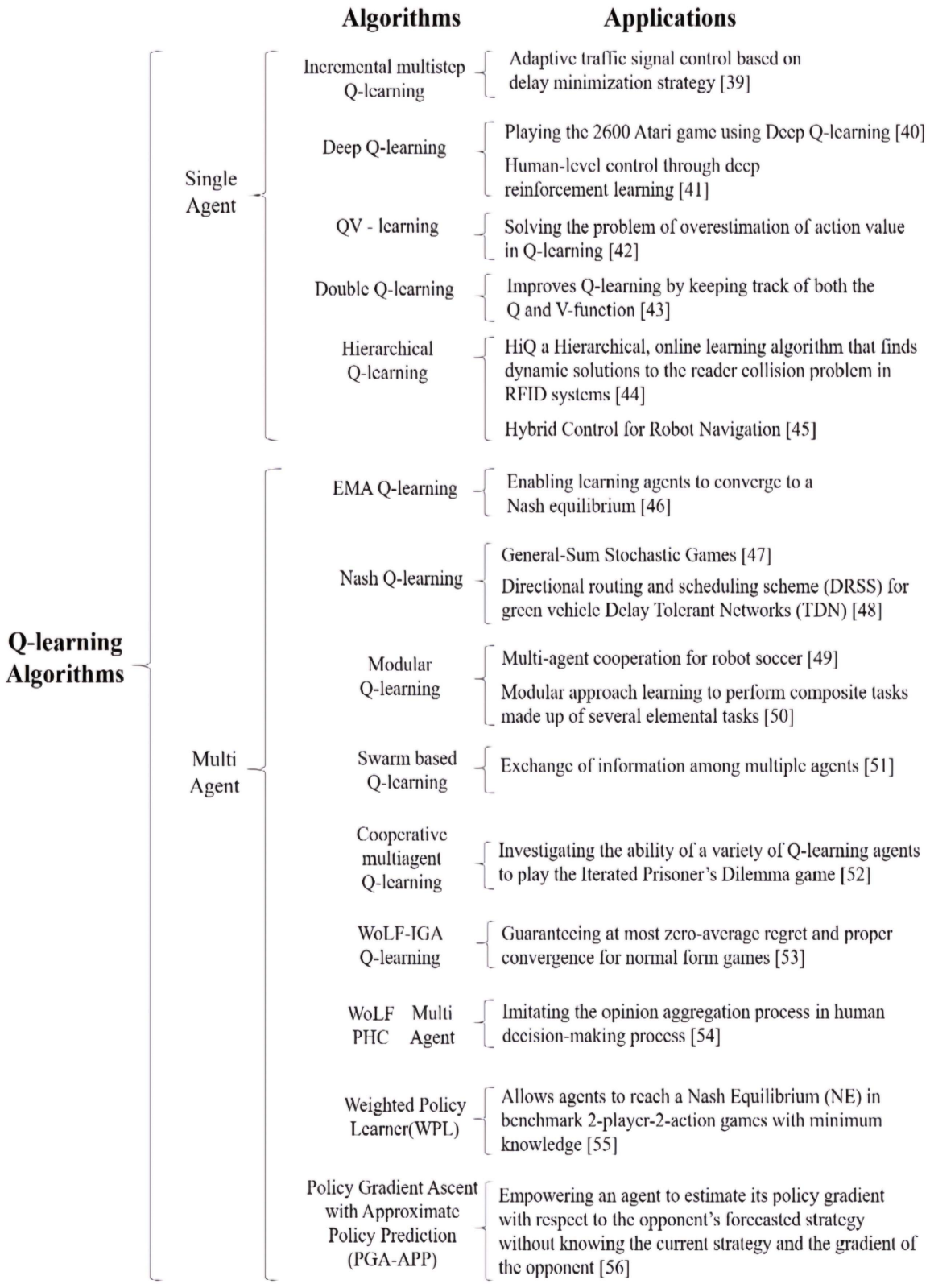


Figure 2.18: Classification of Q-learning algorithms.[6]

2. **Single-Agent :**

- Basic Q-Learning: Q-learning uses an off-policy strategy to separate the acting policy from the learning policy. This means that even if the action taken in the next state is suboptimal, it does not affect the update of the Q-function for the current state. Although the action may be incorrect, Q-learning can still learn effectively due to its off-policy nature. The condition for updating the Q-value is given as follows [40]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

- **Basic Q-Learning:** An off-policy algorithm where agents update Q-values based on observed rewards. The update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

  Here, α is the learning rate, R is the reward, and γ is the discount factor.
- **Deep Q-Learning (DQN):** Extends Q-learning with neural networks to approximate Q-values in large state spaces. It uses experience replay and a target network to improve stability.
- **Hierarchical Q-Learning:** Divides the learning task into high-level and low-level actions, improving learning efficiency in complex environments.
- **Double Q-Learning:** Uses two Q-value estimators to reduce overestimation bias in stochastic settings by separating action selection and evaluation.

$$Q_a(s,a) \leftarrow Q_a(s,a) + \alpha(s,a) \left(R + \gamma Q_b(s',a') - Q_a(s,a) \right)$$

$$Q_b(s,a) \leftarrow Q_b(s,a) + \alpha(s,a) \left(R + \gamma Q_a(s',a') - Q_b(s,a) \right)$$

  Double Q-learning separates the value function utilized in Q-learning that dictates the action to mitigate the value deviation within the Q-learning methodology. The current algorithm mirrors that of Q-learning. The equation is partitioned into a pair of equations, and the values are obtained in a selective and random manner. The algorithm is represented as follows:

3. **Multi-Agent :**

- Modular Q-Learning: Modular Q-learning was developed to overcome the inefficiency of traditional Q-learning in multi-agent environments, where the state space grows exponentially with the number of agents. It addresses this by breaking down a large learning task into smaller sub-tasks, applying Q-learning to each. During action selection, each module generates Q-values for available actions, while a mediator selects the optimal action. The resulting reward is then used to update the Q-values in each module [40]:

$$a \leftarrow \arg\max \sum_{i \in \{1,\ldots,n\}} Q_i(s,a)$$

- Ant Q-Learning: Ant Q-learning integrates the ant system (AS) with Q-learning. In AS, ants leave pheromones while returning to their nest, eventually favoring the path with the highest concentration [41]. Ant Q-learning improves upon traditional AS by using multiple collaborating agents that share AQ-values during learning. The goal is to acquire AQ-values that can achieve a stochastically better target value.
- Nash Q-Learning: Nash Q-learning represents a modification of the Q-learning approach that is applicable in multi-agent settings. Within a multi-agent setting, it is essential to account for the actions of all agents involved. For a given number of n agents, the Q-value is expressed as Q (S, A1, A2,…An) rather than Q (S, A). In light of this, the Nash Q-Learning function is derived by adapting equation in the following manner:

$$Q_t(s, a_1, a_2, \ldots, a_n) = (1 - \alpha_{t-1})Q_t(s, a_1, a_2, \ldots, a_n) + \alpha_{t-1}\left[\gamma_{t-1} + \beta \,\mathrm{Nash}\, Q_{t-1}(s')\right]$$

$$\mathrm{Nash}\, Q_{t-1}(s') = \pi_1(s') \cdot \pi_2(s') \cdot \cdots \cdot \pi_{t-1}(s')$$

- Swarm-Based Q-Learning: Conventional Q-learning can be slow in finding optimal solutions, especially in complex or multi-agent environments. Swarm-based Q-learning integrates Particle Swarm Optimization (PSO) to improve efficiency, as PSO can quickly find global optima in large solution spaces. Studies have shown that combining PSO with Q-learning, Sarsa, and ant colony optimization enhances reinforcement learning performance [40].

Chapter 3

Related work

3.1 Introduction

Advances in manufacturing, electronics, and wireless communication have enabled the development of small, energy-efficient, and multifunctional Wireless Sensor Networks (WSNs), supporting applications like environmental monitoring, object tracking, and surveillance. As energy efficiency remains a core challenge—especially in the context of the Internet of Things (IoT)—research has increasingly targeted low-power WSN architectures and optimized communication protocols.

This chapter classifies MAC protocols into contention-based, schedule-based, and hybrid types, reviewing their strengths and limitations. The MAC layer is critical in WSNs for managing access to the shared wireless medium while addressing energy constraints, scalability, and data quality.

Recently, Artificial Intelligence (AI) techniques—such as reinforcement learning, deep Q-networks (DQNs), and federated learning—have been integrated into MAC design. These AI-driven protocols enable adaptive tuning based on real-time network conditions, improving energy efficiency and performance, and marking a shift toward intelligent, autonomous WSNs.

3.2 Taxonomy

Medium Access Control (MAC) protocols in Wireless Sensor Networks (WSNs) can be generally categorized into two primary types: contention-based protocols and schedule-based protocols.

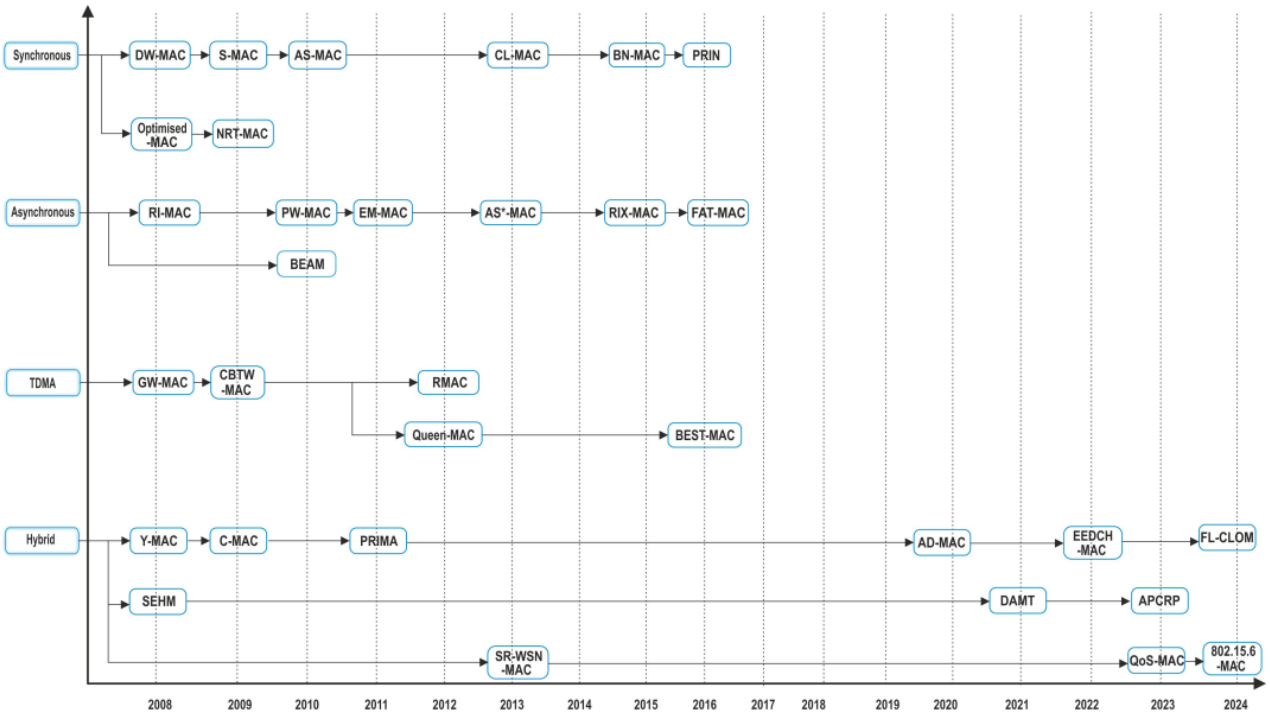


Figure 3.1: Taxonomy of WSN MAC protocols

3.3 Literature Review

This section delivers a clear and structured summary of key Medium Access Control (MAC) protocols designed for Wireless Sensor Networks (WSNs), highlighting their core principles and, where applicable, outlining their respective advantages and limitations. The protocols are systematically categorized into synchronous and asynchronous contention-based approaches, Time Division Multiple Access (TDMA)-based protocols, and hybrid models, as illustrated in the taxonomy in Figure 3.1. Within each category, several representative protocols are discussed in detail. In particular, this section provides an in-depth evaluation of synchronous MAC protocols that follow a contention-based access strategy.

Comparison of MAC Protocols in WSNs

Protocol	Type	Advantages	Disadvantages
DW-MAC	Synchronous Duty-Cycled	Reduces latency; wakes only intended nodes.	Requires synchronization; limited scalability.
OPTIMIZED MAC	Synchronous Duty-Cycled	Adapts duty cycle to traffic.	High energy from forced changes; weak simulation setup.
S-MAC	Synchronous Duty-Cycled	Efficient long-data handling; less idle listening.	Multi-hop latency; collisions in broadcasts.
WAVE-MAC	Synchronous Duty-Cycled	Reduces delay; good for large WSNs.	Needs accurate location info; range issues.
PRIN	Priority-Based	High throughput; QoS queues.	Fails under interference; retransmission loss.
RI-MAC	Asynchronous	High delivery rate; energy-efficient.	Needs timing estimation; collision risk.
BEAM	Asynchronous	Reduces overhearing; adjusts payload.	Complex preambles; basic mode less reliable.
PW-MAC	Asynchronous	Prevents collisions with wake-up.	High parameter overhead; weak in dense networks.
EM-MAC	Asynchronous	Tolerates interference; supports multiple channels.	High computation; more beacon energy use.
GW-MAC	TDMA-Based	Energy-efficient; better data handling.	Needs wired anchors; lacks MAC standard.
CBTW	TDMA-Based	Cuts duplicates; supports aggregation.	Latency; overhead from clustering.
RMAC	TDMA-Based	Less contention; time slot stealing.	Reconfig needed on topology change; energy cost.
QUEEN-MAC	TDMA-Based	Multi-channel; schedules efficiently.	No collision handling; sink overloads.
C-MAC	Hybrid	Concurrent transmissions; high throughput.	No sleep mode; not good for light traffic.
SEHM	Hybrid	Almost zero energy use; no collisions.	Delays in high traffic; complex clustering.
PRIMA	Hybrid	Combines TDMA/CSMA; QoS support.	High overhead; node power drain.
AD-MAC	Hybrid	More energy-efficient than S-MAC.	Complex adaptation; fairness concerns.
IEEE 802.15.6	Hybrid	Supports CSMA/CA, TDMA, polling.	Complex to coordinate; polling cost.
QoS-MAC	Hybrid	QoS support; less contention.	Complex; uses more resources.

3.3.1 Artificial Intelligence drivin-MAC Protocols for WSN

3.3.1.1 Introduction

Wireless Sensor Networks (WSNs) are widely used in environmental monitoring, automation, and smart cities. Due to the limited energy of sensor nodes, efficient MAC protocols are essential. Traditional methods like CSMA/CA and TDMA often struggle with dynamic topologies and variable traffic.

To address these issues, **Artificial Intelligence (AI)**, particularly **Reinforcement Learning (RL)**, is applied to optimize MAC layer performance. RL enables sensor nodes to learn transmission strategies through environmental feedback, improving adaptability and reducing collisions.

3.3.1.2 Q-Learning-Based MAC Protocols

Q-learning is a model-free RL algorithm that helps nodes choose actions (e.g., slot selection, backoff) based on rewards, without centralized control.

- **eOQ-MAC**: Enhances emergency data delivery by adding emergency slots in TDMA frames.
- **Rate-Adaptive CSMA/CA**: Uses Q-learning to adjust contention windows based on channel conditions and energy usage.
- **Backoff Improvement**: In VANETs, nodes learn optimal backoff times to reduce collisions and delays.
- **CW Adjustment Scheme**: Combines Q-learning and supervised learning to dynamically optimize contention window sizes.
- **Performance Enhancement CSMA**: Adapts backoff and scheduling to energy, channel state, and traffic.
- **QUC Protocol**: Q-learning-based unslotted CSMA/CA for asynchronous smart utility networks.
- **RL-MAC**: Learns optimal duty cycles for energy-efficient operation under varying traffic loads.
- **UWSN Q-Learning MAC**: Adapts to high delay and energy constraints in underwater sensor networks using a Q-greedy policy.

These approaches demonstrate the potential of AI to create adaptive, energy-efficient, and scalable MAC protocols tailored to complex WSN environments.

Table 3.1: Summary of MAC Protocols Using Machine Learning

Protocol (Reference)	**MAC Components**	**Network Type**	**ML Algorithm**	**Learning Features**	**Performance Outcome**
eOQ-MAC	Superframe (OC)	Wireless Networks	Q-learning	Packet loss rate, emergency slot timing, throughput	Reduced packet loss rate for emergency data transmissions.
Distributed Rate Adaptive CSMA/CA	Backoff (MM)	Wireless Networks	Q-learning	Channel status, contention window (CW), energy usage	Enabled energy-efficient transmission with faster convergence.
Backoff Improvement	Backoff (MM)	Vehicular Networks	Q-learning	Channel status, success rate	Improved data exchange with fairness assurance.
CW Adjustment Scheme	Backoff (MM)	Wireless Networks	Q-learning, Supervised Learning	Contention window (CW), throughput	Reduced collisions and enhanced throughput.
Performance-Enhanced CSMA	Backoff, Scheduling (MM)	Wireless Networks	RL	Channel conditions, traffic load, energy level	Achieved stable throughput under dynamic conditions.
QUC Protocol	Backoff (MM)	Smart Utility Networks	Q-learning	Throughput, delay	MAC layer performance improved by 20%.
RL-MAC	Duty Cycle	Wireless Sensor Networks	RL	Neighbor traffic load	Maintained high throughput with low power usage.
Energy-Efficient MAC for UWSNs	Collision Avoidance, Energy Management	UWSNs	Q-learning	Noise level, interference, packet collisions	Extended network lifespan, reduced energy usage, and improved throughput.

3.3.1.3 Deep Q-Networks (DQNs) in MAC Design

Deep Q-Networks (DQNs) apply deep neural networks to approximate Q-values, allowing agents to learn optimal MAC strategies in complex, high-dimensional environments. In WSNs, several protocols have adopted DQNs to enhance performance.

DLMA, for instance, intelligently selects transmission slots to maximize throughput and fairness in heterogeneous networks. Other DQN-based MAC protocols use channel observations to predict future states and adapt contention window sizes, reducing collisions in dense networks. AEH-CSMA/CA leverages deep Q-learning to adjust backoff windows based on ambient energy availability, improving throughput under low-power conditions.

In underwater WSNs, DAWPC-MAC uses DQNs to manage power control, collision avoidance, and path loss, highlighting DQN's adaptability in challenging and dynamic environments. These examples underline the growing role of deep reinforcement learning in developing intelligent, energy-efficient MAC protocols for next-generation wireless networks.

Table 3.2: Summary of DQN-based MAC Protocols

Protocol (References)	**MAC Aspects Improved**	**Network**	**Algorithm**	**Reported Performance**
DLMA	Throughput, Fairness	Heterogeneous Networks	DRL	Maximizes total throughput and ensures fairness by selecting specific time slots.
Channel-Observation-Based MAC	Collision Avoidance	Dense WSNs	DRL	Predicts future channel states and controls contention window size to reduce collisions.
AEH-CSMA/CA	Throughput, Energy Efficiency	IoT Devices	Deep Q-learning	Intelligently adjusts backoff window size based on ambient energy, achieving high throughput with low energy supply.
DAWPC-MAC	Collision Avoidance, Energy Efficiency, Reliability	Underwater Environments	DQN	Optimizes resource allocation, minimizes transmission conflicts, dynamically adjusts to fluctuating network conditions, improves packet delivery ratio, throughput, and utility.

3.3.1.4 DRL with Graph Neural Networks (GNNs) for Enhanced MAC Performance

The integration of Graph Neural Networks (GNNs) with DRL provides a powerful approach for processing graph-structured data, where nodes represent devices and edges represent connections in a network. This combination allows the DRL agent to consider the network topology and the relationships between nodes when making MAC layer decisions. The Neuro-DCF protocol is an example of this approach, which trains a CSMA-based MAC protocol using a DRL algorithm that incorporates a GNN-based training construction. This GNN helps in creating a unified training approach that can incorporate various perturbation patterns and network configurations. By operating on graph-type data, Neuro-DCF can consider multiple network

factors such as throughput, delay, and channel utilization, aiming to accelerate the speed of decision convergence. Simulations have shown that this scheme can improve delay performance while maintaining optimal utility. The use of GNNs in Neuro-DCF to model network topology and dependencies represents a significant advancement in DRL-based MAC protocols, enabling more informed decision-making by considering the relationships between nodes. Beyond Neuro-DCF, DRL with GNNs holds potential for other applications in WSN MAC protocols, such as resource allocation and interference management, drawing from broader trends in wireless networks.

3.3.2 Comparative Analysis of AI Techniques

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) are key AI techniques used to enhance MAC protocols in WSNs. RL helps nodes adapt parameters like duty cycles and contention windows based on network feedback, offering low computational cost but limited scalability due to its tabular nature. DRL addresses this by using neural networks to manage complex state spaces, making it more suitable for dense or dynamic environments, though at a higher computational cost.

While RL protocols like RL-MAC are effective in simpler settings, DRL approaches such as DLMA show better performance in complex scenarios. The following table summarizes various RL- and DRL-based MAC protocols, highlighting their improvements and trade-offs.

Table 3.3: Comparison of AI Techniques for MAC Protocol Enhancement

AI Technique	**Primary Goals in MAC Enhancement**	**Strengths**	**Limitations**
Reinforcement Learning (RL)	Adaptability, Optimization of parameters (e.g., duty cycle, contention window), Prioritization of traffic	Learns optimal policies through interaction, Suitable for dynamic environments, Can prioritize critical data	May require significant training time, Performance depends on reward function design
Deep Reinforcement Learning (DRL)	Handling complex state and action spaces, Learning intricate patterns	Effective in high-dimensional environments, Can learn complex control policies	Higher computational resource requirements, More complex to design and train

3.4 Discussion : gaps and limitations

Medium Access Control (MAC) protocols in wireless sensor networks (WSNs) are typically either contention-based (e.g., CSMA) or schedule-based (e.g., TDMA). CSMA offers simplicity and adaptability but struggles with collisions in dense networks. TDMA provides collision-free communication but often relies on centralized control, reducing flexibility in dynamic environments.

To address these issues, hybrid MAC protocols combining CSMA/CA and TDMA principles have emerged. These allow nodes to perform carrier sensing before transmitting in assigned slots, reducing collisions while prioritizing successful transmitters.

Despite these improvements, challenges such as hidden node interference, idle listening, and limited scalability persist. AI techniques, particularly Reinforcement Learning (RL), offer adaptive, data-driven solutions. Q-learning enables nodes to learn optimal transmission strategies from local feedback, reducing collisions without centralized control. Deep Q-Learning can handle larger state spaces but requires more computational resources.

Supervised learning methods like Random Forests and SVMs can optimize MAC parameters in stable environments but need labeled data and retraining in dynamic settings. Unsupervised methods aid in pattern detection but lack precise control for MAC decisions.

In smart farming, where networks are low-mobility and energy-constrained, lightweight AI methods like Q-learning and simple supervised learning are most practical at the edge. Deep learning, while resource-intensive for sensor nodes, can be used at gateways for global optimization.

A hybrid AI strategy is ideal: lightweight learning at the edge for real-time control, supported by centralized models for broader network management. This balanced approach enables efficient, scalable MAC performance suited to smart agriculture and IoT-based WSNs.

3.5 Conclusion

In summary, this chapter presents a comprehensive analysis of Medium Access Control (MAC) protocols in Wireless Sensor Networks (WSNs). It establishes a classification framework predicated on prioritization, thereby offering a systematic method for the comprehension and assessment of these protocols. The literature review elucidates the present research environment, identifying existing deficiencies and constraints. It emphasizes the importance of integrating traffic characteristics and application-specific needs into the design of MAC protocols. Tackling these issues creates significant avenues for future research and innovation within the discipline.

Chapter 4

Contribution

4.1 Introduction

This chapter presents the simulation setup and evaluation of a Q-learning-enhanced (TDMA) MAC protocol for WSNs. Unlike traditional (TDMA) with fixed slots, the proposed protocol uses decentralized Q-learning to let nodes learn optimal slot selection based on local feedback. A custom Python-based simulator tracks key metrics such as delivery ratio, energy use, throughput, and fairness. Results show that the learning-based approach reduces collisions, improves throughput, and lowers energy use—especially in dense networks. Sensitivity analysis highlights that moderate values of and balance learning speed and stability, while strongly affects convergence and exploration.

4.2 Protocol Description

This section introduces Q-TDMA, a scalable Q-learning-based TDMA MAC protocol designed to ensure fair, collision-free communication in dense sensor networks like smart farming. Unlike static TDMA, Q-TDMA enables each node to learn and adaptively select transmission slots using Q-learning, minimizing collisions over time. By maintaining and updating a Q-table based on transmission outcomes, nodes converge to an efficient, decentralized slot schedule. Q-TDMA improves slot utilization, reduces interference, and operates without centralized control, making it suitable for energy-constrained, dynamic WSN environments.

4.3 System model

The examined system is made up of several sensor nodes that gather physical data, analyze it, and wirelessly send it to a central node known as the BS, or sink node, positioned within one hop of each other. This node then forwards the information to the end user, as illustrated in figure 4.1.

All sensor nodes wait for a beacon from the sink node before starting data transmission simultaneously, as shown in the figure. The system model employs a time-slotted structure based on Slotted ALOHA (S-ALOHA), where communication is organized around a recurring unit called a virtual slot (Vslot). Vslots synchronize nodes and manage access to the medium. Each Vslot includes a guard time (T_{Guard}) to correct timing errors, an inter-frame space (T_{AIFS}) inspired by IEEE 802.11 DCF, and a transmission period (T_{packet}). These slots repeat in

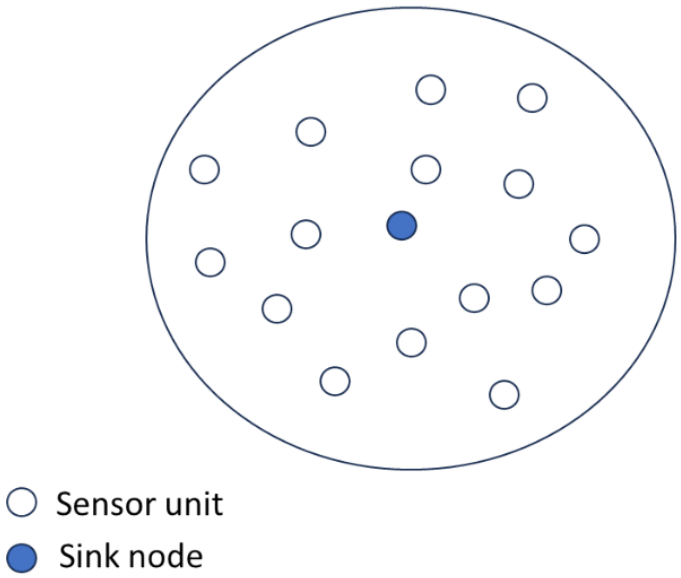


Figure 4.1: group of sensors nodes which collect physical information.

sequence (0 to N). The model assumes symmetric links, equal node ranges, and operation within a single cluster. A Cluster Head (CH) coordinates access, sending beacons and data packets. All nodes transmit fixed-size packets at the start of each second period, aligned using the sink's beacon for MAC-layer synchronization.

$$T_{\text{Vslot}} = T_{\text{Guard}} + T_{\text{AIFS}} + T_{\text{packet}}$$

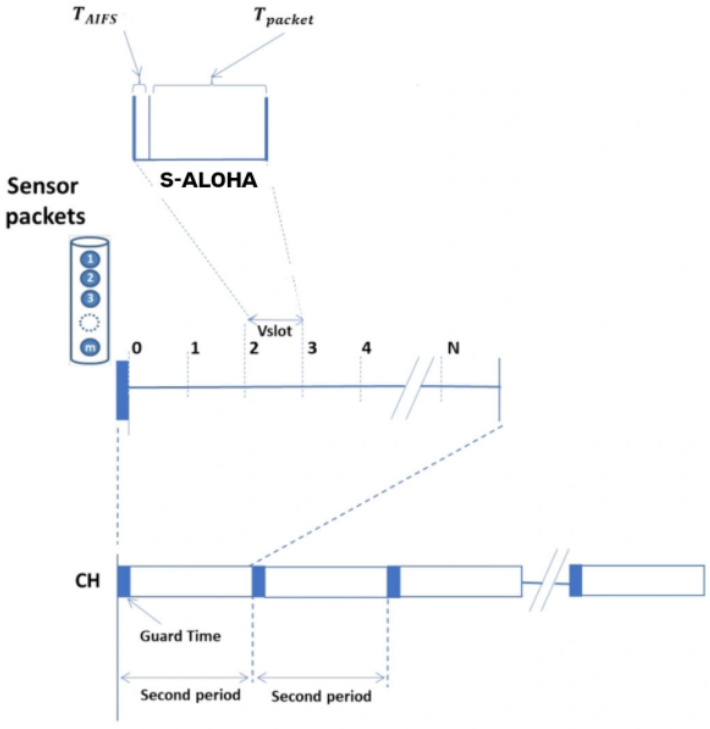


Figure 4.2: Second period, guard interval, and Vslot interval.

4.4 Q-learning Integration

4.4.1 A Model for Transmitting Packets

State Definition: In the proposed Q-learning-enhanced TDMA MAC protocol, each agent's state is defined using multiple internal and network-level parameters to support smarter transmission decisions. Unlike basic single-state models, this approach uses a richer state representation to enhance learning.

Key state features include:

- **Queue length:** Number of pending data packets.
- **Residual energy:** Helps manage energy consumption and extend node lifetime.
- **Channel status:** Reflects recent idle or busy states of the medium.
- **TDMA context:** Includes the current slot index and recent slot usage history.

This multi-dimensional state allows each node to adjust its actions based on traffic demand, energy constraints, and channel dynamics. While it increases state complexity, it significantly improves slot allocation efficiency and overall network performance.

Action Space: Each node selects an action $a_t = \langle j, k \rangle$, representing slot $j \in S$ and delay $k\delta$ within that slot, where $k \in \{0, 1, \ldots, K\}$. This allows fine-grained, decentralized transmission timing, improving slot utilization and reducing collisions without needing neighbor information.

Reward Function: After transmission, a node receives:

$$r_t = \begin{cases} +1 & \text{if ACK received (success)} \\ -1 & \text{if no ACK (failure/collision)} \end{cases}$$

This binary reward guides slot/delay selection. Extensions can include energy and delay penalties for application-specific tuning.

Q-table Management: Each node maintains its own Q-table, mapping state-action pairs (e.g., buffer level, slot index) to expected rewards. This fully distributed design reduces overhead and enables scalability.

Q-learning and Exploration: Nodes update Q-values using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

An ϵ-greedy strategy balances exploration and exploitation, with decaying ϵ:

$$\epsilon = \max(\epsilon_{\min}, \epsilon_0 \cdot e^{-\lambda t})$$

This enables early exploration and later convergence to stable, low-collision schedules. The design suits dynamic, energy-limited WSNs without centralized control.

4.4.2 Learning to Avoid Collisions

To enable autonomous collision avoidance in dense wireless sensor network (WSN) environments, the proposed Q-learning-enhanced TDMA MAC protocol incorporates a distributed reinforcement learning mechanism at each node. The protocol operates over multiple time frames, during which nodes iteratively refine their transmission policies by interacting with the environment and receiving feedback based on packet delivery success. The protocol's operation is structured into several distinct phases: initialization, learning and interaction, TDMA slot selection, data transmission, and adaptation.

Protocol Overview: Each sensor node operates as an independent Q-learning agent with a local Q-table of size $S \cdot K$ (slots $\times$ delays). The table is initialized with random values in $[0, 1]$, and learning is performed online.

Action Selection and Update: Nodes use an ϵ-greedy policy to select actions. After transmission, Q-values are updated based on ACK feedback:

$$Q(a_t) \leftarrow Q(a_t) + \alpha \left[r_{t+1} - Q(a_t) \right]$$

Slot Convergence: Over time, nodes converge to high-reward actions, minimizing collisions. A decaying ϵ ensures gradual transition from exploration to exploitation.

Adaptation: The protocol adapts to dynamic conditions without centralized control. Nodes adjust based on local feedback, enabling scalability.

Hysteretic Q-learning: To improve convergence in multi-agent settings, the update uses two learning rates—α (positive reward) and β (negative reward).

$$Q(a_t) \leftarrow \begin{cases} Q(a_t) + \alpha \left[r_{t+1} - Q(a_t) \right], & \text{if } r_{t+1} - Q(a_t) \geq 0 \\ Q(a_t) + \beta \left[r_{t+1} - Q(a_t) \right], & \text{if } r_{t+1} - Q(a_t) < 0 \end{cases}$$

The temporal-difference error $\mu = r_{t+1} - Q(a_t)$ guides Q-value updates. A higher learning rate α is used for positive updates (successful actions), while a lower rate β handles negative updates (penalized actions), preventing overreaction to transient collisions. This hysteretic update strategy enhances stability, preserves effective past behaviors, and is especially beneficial in dense or dynamic WSNs. It supports robust convergence toward optimal TDMA slot allocation by balancing adaptability with long-term learning.

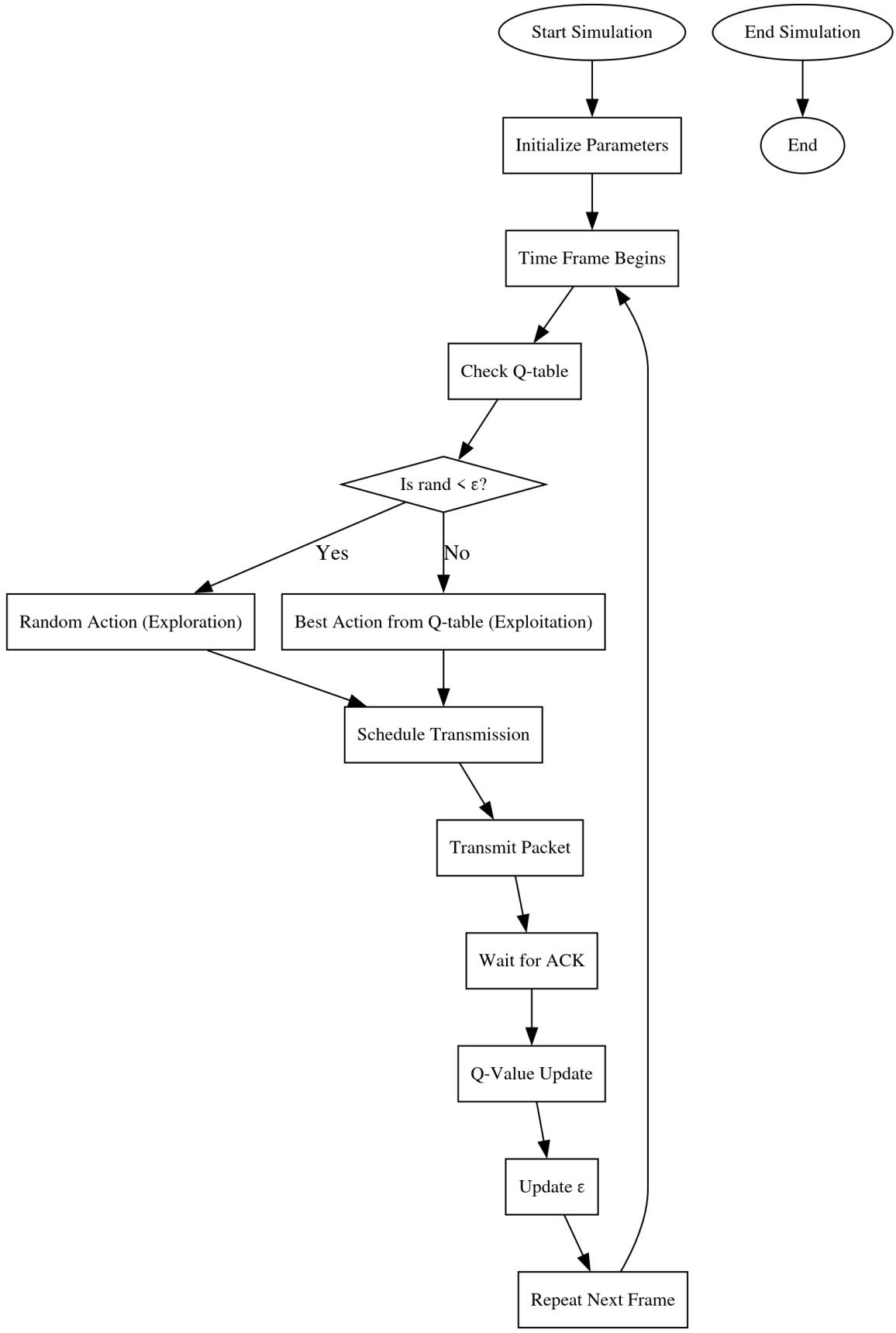


Figure 4.3: Flow chart of the Q-TDMA protocol.

4.4.3 Algorithm of the Q-Learning

Algorithm 1: Q-Learning-based TDMA Slot Selection with ε-Decay

Input : Number of Nodes N, Number of Slots S, Episodes E, Learning Rate α, Initial Exploration Rate ε, Minimum ε_{min}, Decay Rate d

Output: Updated Q-Tables, Collision Statistics, Success Rate per Episode

Initialization:
for *each node* $i \in \{1, ..., N\}$ **do**
    Initialize Q_i as $[S \times 2]$ zero matrix;
    Assign random initial slot s_i;
for *episode* $= 1$ **to** E **do**
    for *each node* i **do**
        Generate random number $r \in [0, 1]$;
        if $r < \varepsilon$ **then**
            action $\leftarrow$ random (exploration);
        else
            action $\leftarrow \arg\max_a Q_i[s_i, a]$ (exploitation);
    for *each node* i **do**
        if *action == switch* **then**
            $s_i \leftarrow$ randomly select a new slot;
        else
            remain in current slot s_i;
    Count number of nodes in each slot;
    for *each node* i **do**
        if *slot* s_i *has only 1 node* **then**
            reward $r = +1$;
        else
            reward $r = -1$;
        Update: $Q_i[s_i, a] \leftarrow Q_i[s_i, a] + \alpha \cdot (r - Q_i[s_i, a])$;
    $\varepsilon \leftarrow \max(\varepsilon_{min}, \varepsilon \cdot d)$;
    Record success/collision statistics for this episode;
Return final Q-tables;

4.5 Simulation and results

4.5.1 Simulation Methodology

The simulation methodology is divided into two phases to evaluate the proposed Q-learning-enhanced TDMA MAC protocol. The first phase focuses on tuning key Q-learning parameters—learning rate (α), discount factor (γ), and exploration rate (ϵ)—to achieve optimal slot utilization, low collision rates, and stable convergence.

In the second phase, the optimized protocol is benchmarked against traditional TDMA and other baseline MAC protocols using metrics such as packet delivery ratio, throughput, collision rate, energy consumption, and Jain's fairness index. This two-step process ensures both protocol optimization and fair comparative validation across diverse network scenarios.

4.5.2 Evaluation metrics

Four different performance metrics have been selected to evaluate the performance of the simulated protocols in both phases of the study. These metrics reflect key Medium Access Control (MAC) layer characteristics and are defined as follows:

1. **Average Medium Collisions:** This metric measures the average number of collisions detected per message sent within the communication range. It is a critical factor for assessing the MAC protocol's efficiency in minimizing collisions, which directly impacts overall network performance and reliability.

2. **Access Fairness:** This refers to the equitable distribution of channel access opportunities among all network nodes. It evaluates how effectively the MAC protocol ensures that each node has a fair chance to access the Cluster Head (CH) for data transmission. Higher fairness indicates more balanced and efficient use of the communication medium.

3. **Rx Throughput:** This metric quantifies the rate at which a receiving node successfully receives data packets over a specific time interval. It reflects the MAC protocol's and the network's efficiency in delivering data to the intended recipients. A higher Rx throughput signifies better performance in terms of successful data delivery.

4. **Beacon Delivery:** This metric represents the fraction of correctly received beacon messages by a given sensor node, divided by the total number of beacons transmitted by its neighbors. It reflects the reception rate of broadcast traffic, which is influenced by the distance between transmitter and receiver. This metric assumes that broadcast traffic sources are uniformly distributed and that retransmissions are not employed.

4.5.3 Phase 1: Performance study

This phase aims to optimize the Q-learning parameters—learning rate (α), discount factor (γ), and exploration rate (ϵ)—to enhance the performance of the Q-TDMA MAC protocol. Controlled simulations evaluate each parameter's effect on collision probability, Rx throughput, beacon delivery, and fairness. By varying one parameter at a time, the study identifies settings that ensure efficient, fair, and adaptive slot allocation under dynamic wireless conditions.

4.5.3.1 Simulation Tool/Platform

The simulation of the Q-learning-enhanced TDMA MAC protocol was conducted using a self-developed event-driven simulator built in **Python** and executed within the **Anaconda** environment. Anaconda was selected due to its robust support for scientific computing, seamless package management, and reproducibility of experiments—an essential feature for reinforcement learning (RL) simulations. It offers a consolidated ecosystem for managing dependencies and running isolated simulation environments efficiently.

Figure 4.4: Anaconda Environment

Programming Language: Python 3.6.13
The implementation of the algorithms and the control of the simulations were carried out using Python 3.6.13. Python is widely used in the field of artificial intelligence research.

Machine Learning Framework: TensorFlow 1.15
TensorFlow was used to implement the reinforcement learning components of the protocol. While the current approach uses table-based Q-learning, TensorFlow enables easy expansion to Deep Q-Networks (DQN) thanks to its efficient numerical support and GPU acceleration.

The simulator was tailored for dynamic wireless sensor networks (WSNs), accurately modeling TDMA structures, packet scheduling, energy constraints, and underwater acoustic delays. Key libraries include **NumPy** (numerical computation), **Pandas** (data handling), and **Matplotlib** (visualization). Custom modules handle node behavior, slot allocation, ACK processing, and distributed Q-updates.

Built within the Anaconda environment, this setup provides a scalable platform for testing learning-based MAC protocols, supporting future extensions to complex topologies and advanced learning strategies.

4.5.3.2 Q-learning Parameters

In the proposed simulation, Q-learning is integrated into the MAC layer to enable adaptive and distributed time-slot selection among sensor nodes. The following parameters and configurations define how the learning mechanism is implemented and executed:

- **Agent Location:**
  Each sensor node acts as an independent Q-learning agent with a local Q-table, enabling fully decentralized learning without centralized coordination. Nodes adapt based on local transmission outcomes (success or failure).

- **Learning Process:**
  Learning is performed online, allowing nodes to update Q-values continuously based on real-time feedback after each TDMA frame. This supports adaptation to dynamic network conditions.
- **Learning Parameters:**
  The algorithm uses a learning rate α, discount factor γ, and an ϵ-greedy policy. Initially, $\epsilon = 1.0$ to promote exploration, and it decays toward $\epsilon_{\min}$ over time.
- **Episodes:**
  The simulation runs over 3,000 TDMA frames (episodes). In each, nodes choose slot-delay actions, transmit, and receive binary rewards to refine their Q-values.
- **Q-table Initialization:**
  Q-tables are randomly initialized in $[0, 1]$ to encourage early exploration. Values are updated over time to reflect optimal slot selections with fewer collisions.

Table 4.1: Description of Key Q-learning Parameters

Parameter Name	**Symbol**	Description
Learning Rate	α	Controls how much new info overrides past knowledge
Discount Factor	γ	Balances importance of future vs. immediate rewards
Exploration Rate	ϵ	Guides the agent from exploration to exploitation
Epsilon Decay	–	The process of gradually decreasing the exploration rate of ϵ in Q-learning to shift from exploration to exploitation as learning progresses

4.5.3.3 Q-Table Example

We have 5 nodes, each containing 5 slots, tested over 1000 episodes. Below is the resulting Q-table obtained from this scenario:

Table 4.2: Node 1 Q-Table and Final Action per Slot

Slot	**Stay Q-Value**	**Switch Q-Value**	**Final Action**
0	-7.79	6.54	Switch
1	10.00	-9.18	Stay
2	-6.63	-7.57	Stay
3	-9.55	-7.24	Switch
4	-5.75	4.53	Switch

This table presents the expected rewards for choosing either the "Stay" or "Switch" action across five different slots (0 through 4). The values reveal that Slot 0 favors switching (6.54 vs -7.79), while Slot 1 strongly recommends staying (10.00 vs -9.18). Some slots like Slot 2 show negative outcomes for both actions, though staying is slightly less unfavorable (-6.63 vs -7.57). This pattern continues through Slots 3 and 4, where switching proves better in Slot 3 (-7.24 vs -9.55) and Slot 4 (4.53 vs -5.75). The table demonstrates how reinforcement learning models quantify decision quality, where higher values indicate more optimal choices at each state.

4.5.4 Simulation Result Analysis

This section presents a sensitivity analysis of Q-learning parameters (α, γ, ϵ). Metrics evaluated include *Average Collision Probability*, *Access Fairness*, *RX Throughput*, and *Beacon Delivery Probability*, under three learning modes: $\epsilon = 1$, $\epsilon = 0$, and decaying ϵ.

1. Impact of Learning Rate (α)

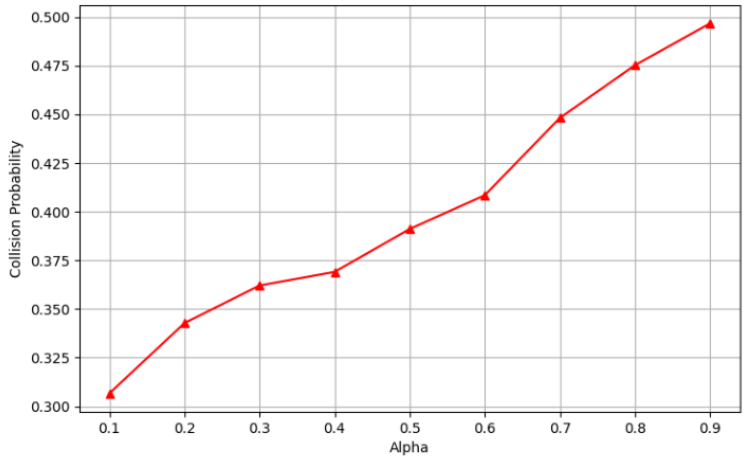


Figure 4.5: The Average Medium Collisions Ratio.

The Average Medium Collisions Probability: As α increases (0.1–0.9), collision probability rises from 0.300 to 0.500, reflecting increased contention. Optimizing α is essential to balance channel access and collision avoidance.

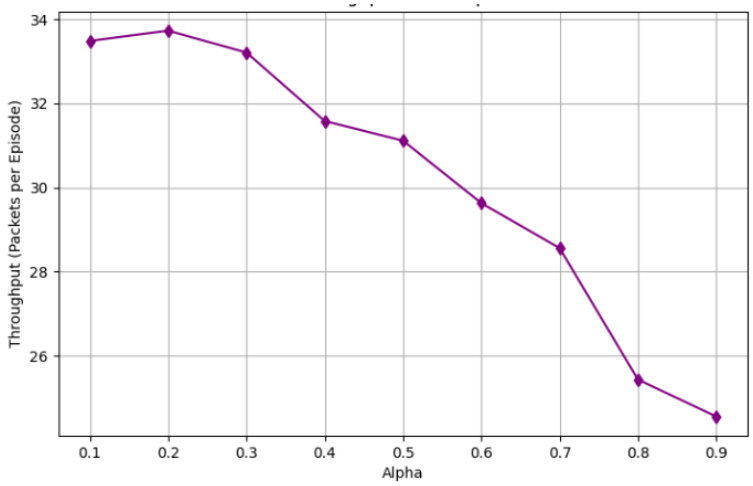


Figure 4.6: The Rx Throughput Ratio.

The Rx Throughput probability: Throughput is low at $\alpha = 0.1$–0.3, peaks at $\alpha = 0.4$–0.6, and decreases when $\alpha > 0.7$ due to collisions. Mid-range α yields optimal performance.

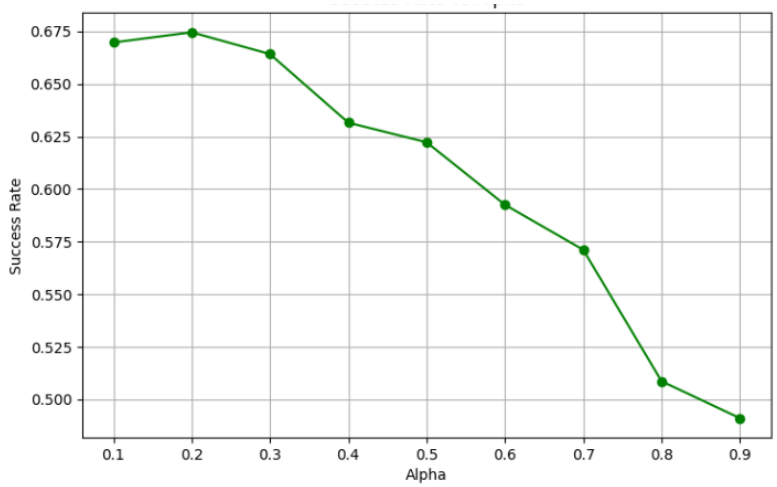


Figure 4.7: The Beacon Delivery Ratio.

The Beacon Delivery probability: Beacon success remains high at low α (0.675), then declines gradually (0.575) and drops sharply past $\alpha = 0.7$, highlighting the importance of tuning for reliability.

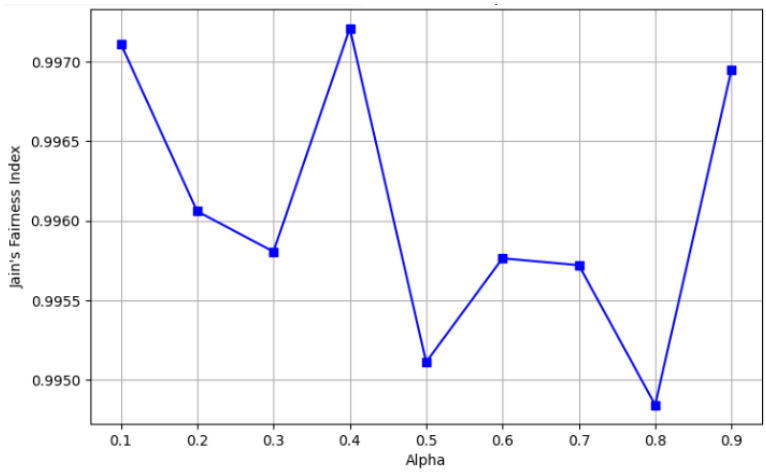


Figure 4.8: The Access Fairness Ratio.

The Access Fairness probability: Fairness remains stable (0.995–0.997) across all α values, indicating consistent bandwidth distribution regardless of traffic intensity.

2. Impact of Discount Factor (γ)

This section analyzes Q-learning parameter sensitivity by varying one parameter at a time while keeping others fixed. Performance metrics—*Average Collision Probability*, *Access Fairness (Jain's Index)*, *RX Throughput*, and *Beacon Delivery Probability*—were recorded for each run. Three learning modes were explored: pure exploration ($\epsilon = 1$), pure exploitation ($\epsilon = 0$), and a decaying ϵ (from 1.0 to 0.01). The objective is to determine the parameter set (α, γ, ϵ) that consistently delivers optimal protocol performance.

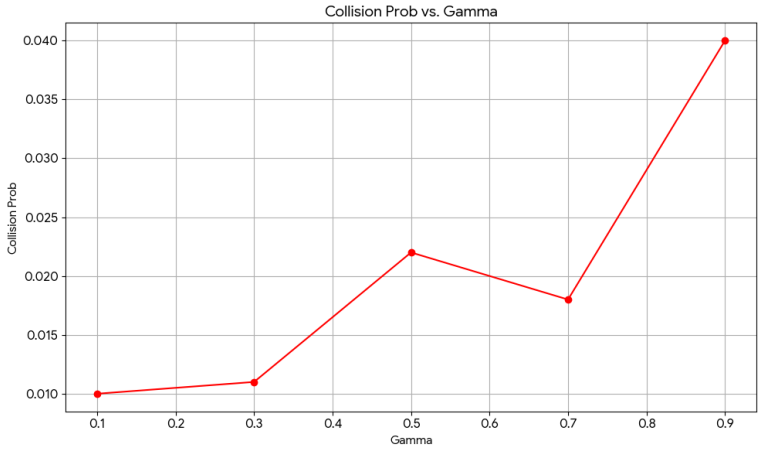


Figure 4.9: The Average Medium Collisions Ratio.

The Average Medium Collisions Probability: Collisions decline from 0.040 to 0.010 as γ increases, showing better learning stability with higher long-term reward focus.

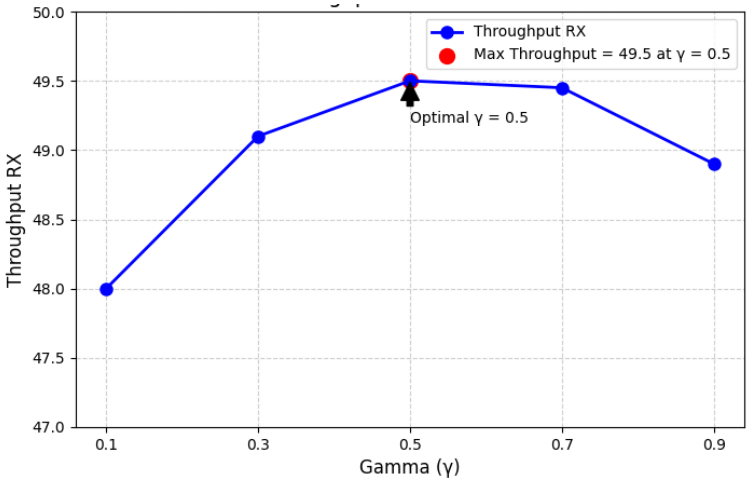


Figure 4.10: The Rx Throughput Ratio.

The Rx Throughput probability: Throughput is lowest at $\gamma = 0.1$–0.3, peaks at $\gamma = 0.5$, and drops beyond due to increased contention, indicating $\gamma^* = 0.5$ as optimal.

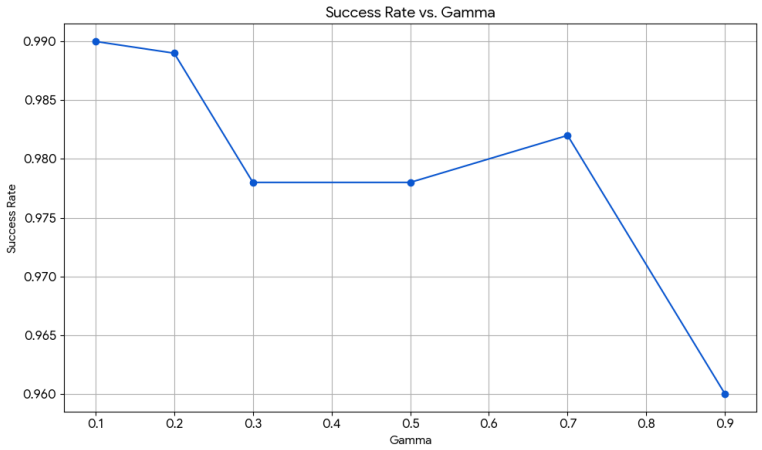


Figure 4.11: The Beacon Delivery Ratio.

The Beacon Delivery probability: Success rate gradually decreases from 0.990 to 0.960 as γ increases. Although minimal, it suggests slight contention from higher long-term planning.

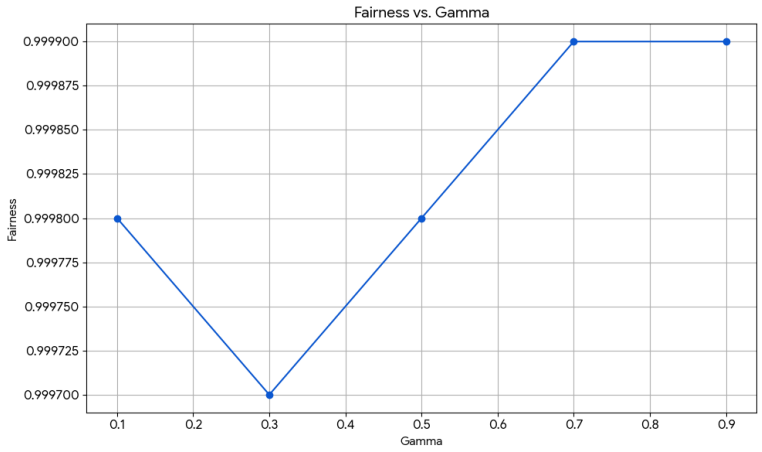


Figure 4.12: The Access Fairness Ratio.

The Access Fairness probability: Fairness remains extremely high (0.99970–0.99990) and stable across all γ, confirming the protocol ensures equitable access regardless of reward bias.

3. Impact of Exploration Rate (ϵ and Decay)

- Exploitation ($\epsilon = 0$)

The Average Medium Collisions Probability:

Exploitation relies on past strategies, keeping collisions stable (0.2–0.5) during episodes 50–1000. While efficient, this limits adaptability in dynamic scenarios.

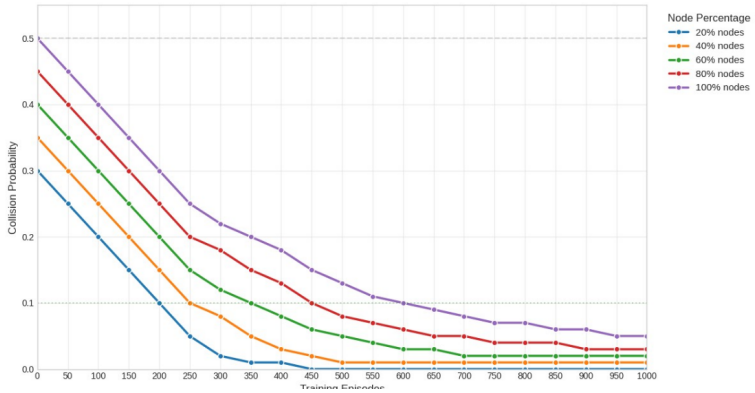


Figure 4.13: The average medium collision Ratio.

The Rx Throughput probability:

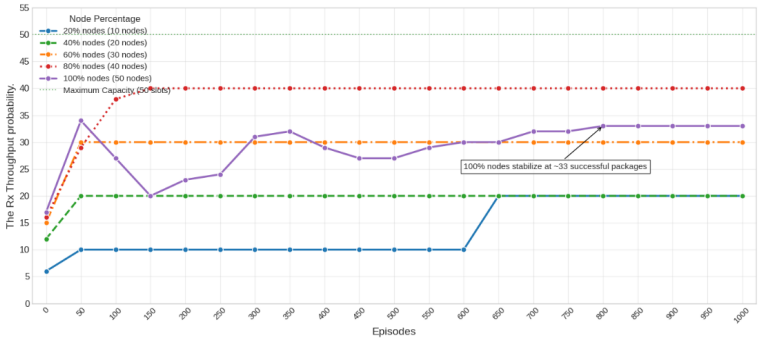


Figure 4.14: The Rx Throughput Ratio.

Throughput stabilizes (33 packets) at high node density, reflecting learned, consistent behavior. However, lack of exploration may prevent discovering better strategies.

The Beacon Delivery probability:

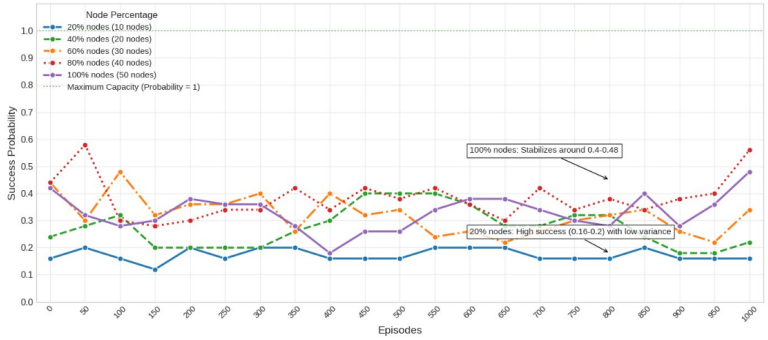


Figure 4.15: The Beacon Delivery Ratio.

Delivery stabilizes as node density increases, showing solid exploitation. Yet, limited exploration could hinder adaptability under future changes.

The Access Fairness probability:

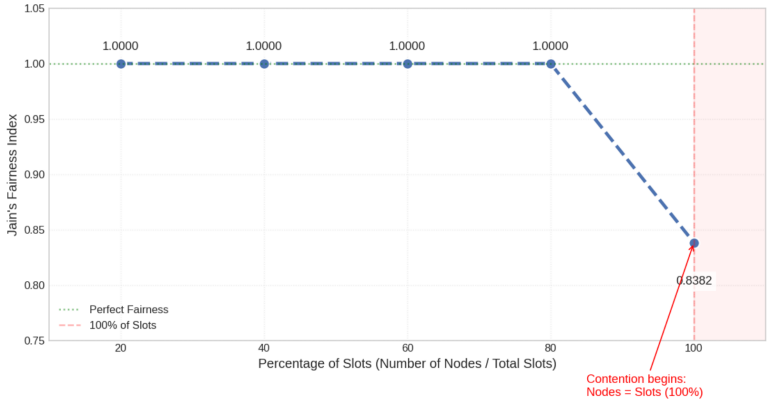


Figure 4.16: The Access Fairness Ratio.

Fairness is high up to 80% load but drops at 100% as some nodes dominate. Exploitation improves performance but can reduce equity without balancing mechanisms.

- Exploration ($\epsilon = 1$)

The Average Medium Collisions Probability:

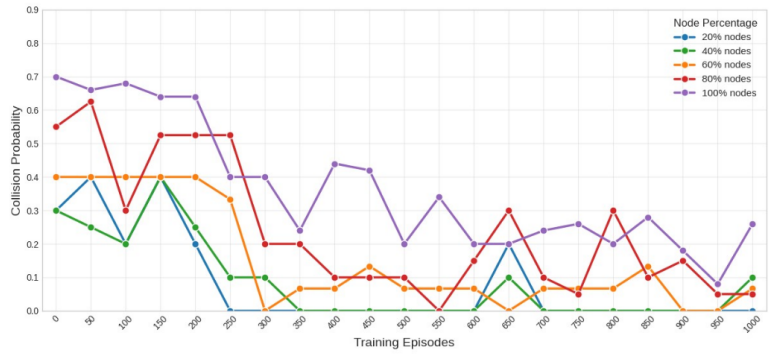


Figure 4.17: The average medium collision Ratio.

Exploration ensures fairness under low loads but leads to higher contention and monopolization under full slot usage. Shows performance–fairness tradeoff.

The Rx Throughput probability:

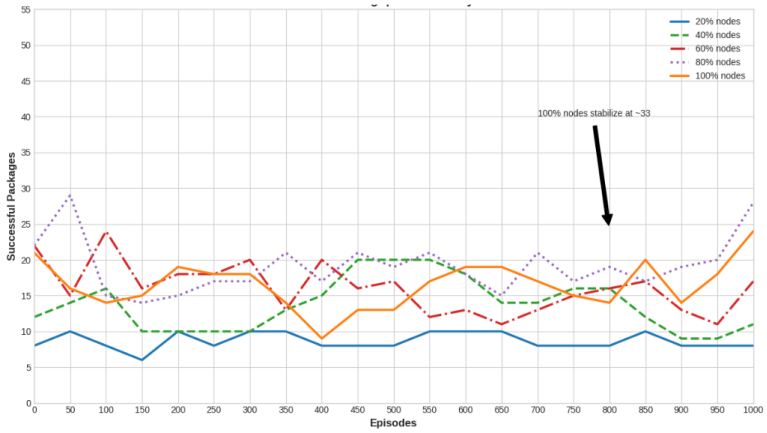


Figure 4.18: The Rx Throughput Ratio.

Early episodes show high throughput variation due to experimentation. Over time, stable performance (33 packets) is achieved through effective learning.

The Beacon Delivery probability:

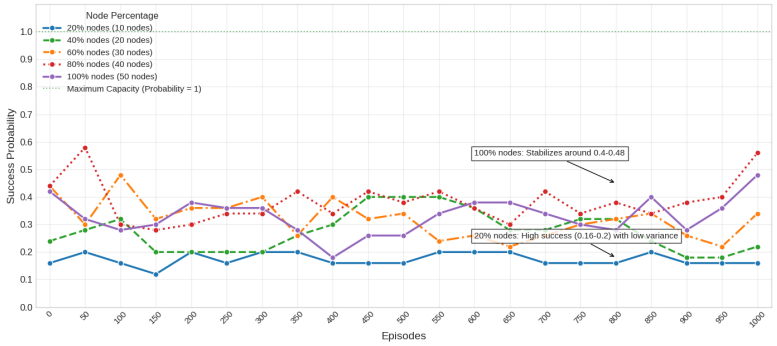


Figure 4.19: The Beacon Delivery Ratio.

Despite congestion (up to 200%), exploration finds viable strategies, with success rates improving and converging (0.4–0.48), showing robustness and adaptability.

The Access Fairness probability:

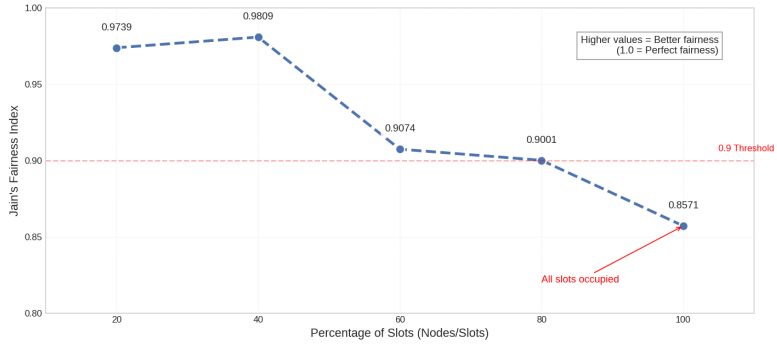


Figure 4.20: The Access Fairness Ratio.

Fairness peaks under moderate loads (40%), drops slightly at full load (0.8571). Fluctuations show continued exploration—essential for adaptability under varying traffic.

- Exploitation/Exploration

The Average Medium Collisions Probability:

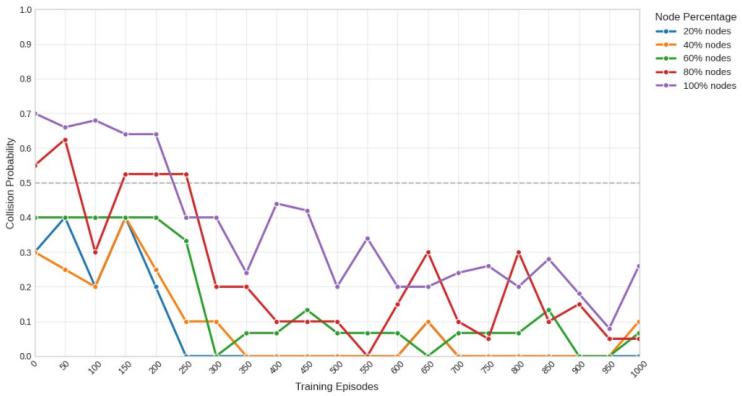


Figure 4.21: The Average Medium Collisions Ratio.

Early exploration leads to volatile collisions (0.0–1.0). Over time, the system stabilizes, balancing reliability and efficiency, while retaining flexibility through occasional exploration.

The Rx Throughput probability:

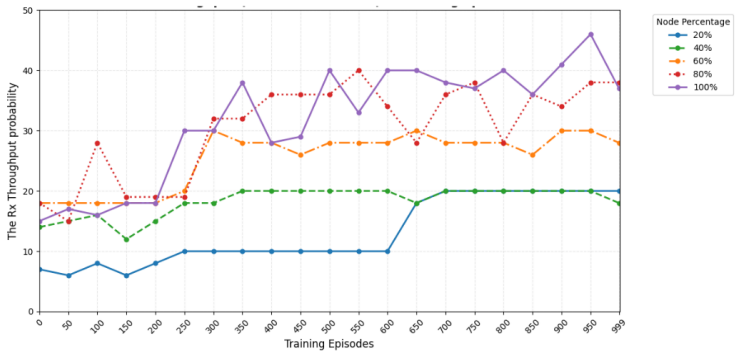


Figure 4.22: The Rx Throughput Ratio.

Initially, diverse strategies are tested across densities (20%–100%). Later, stable policies dominate, ensuring consistent throughput while preserving adaptability through selective exploration.

The Beacon Delivery probability:

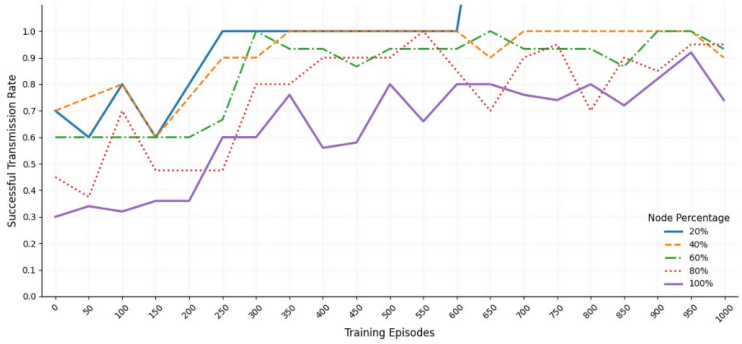


Figure 4.23: The Beacon Delivery Ratio.

Training begins with broad strategy exploration and gradually shifts to exploiting reliable behaviors, especially under dense conditions. The mix supports resilient performance in dynamic networks.

The Access Fairness probability:

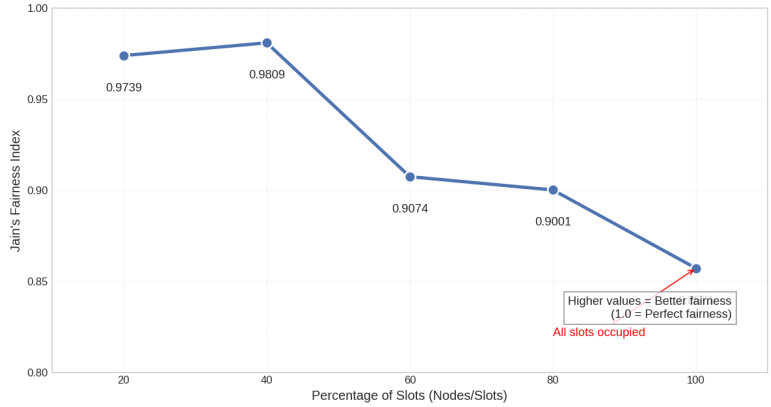


Figure 4.24: The Access Fairness Ratio.

Fairness peaks under light loads (0.98) but drops to 0.85 at full capacity. This shows an adaptive trade-off between efficiency and equity, with fairness consistently above 0.8.

6. Optimized Q-learning Parameters

Based on the comprehensive parameter optimization study, the following values have been selected as optimal for the proposed Q-learning-based MAC protocol:

Table 4.3: Description of Key Q-learning Parameters

Parameter Name	**Symbol**	Value
Learning Rate	α	0.1
Discount Factor	γ	0.5
Exploration Rate	ϵ	$1.0 \rightarrow 0.1$ (exponential decay)

These parameters were chosen to achieve the best balance between learning efficiency, system stability, and adaptability to dynamic network conditions.

- **Learning Rate** ($\alpha = 0.1$): This value provided the ideal trade-off between fast Q-value updates and stable convergence. It demonstrated superior performance by maximizing throughput and minimizing collision probability, without sacrificing fairness.
- **Discount Factor** ($\gamma = 0.5$): This setting was found to be optimal for long-term reward optimization. Significant improvements in success rate and collision reduction were observed up to this value, while higher γ values yielded only marginal gains, indicating diminishing returns.
- **Exploration Rate** ($\epsilon = 1.0 \rightarrow 0.1$): An initial $\epsilon = 1.0$, followed by exponential decay down to $\epsilon = 0.1$, allowed the agent to explore extensively during the early learning phase. Over time, the policy gradually shifted toward exploiting the learned optimal actions, ensuring both rapid convergence and robust performance.

In summary, the parameter set ($\alpha = 0.6, \gamma = 0.5, \epsilon = 1.0 \rightarrow 0.1$) ensures efficient learning, balanced decision-making, and consistent performance across various wireless network scenarios. These values will be used consistently in all subsequent simulations and comparative evaluations of the proposed protocol.

4.5.5 Phase 2: Comparison study

The objective of this study is to demonstrate the performance (or specific advantages) of the proposed Q-learning enhanced TDMA MAC protocol compared to selected baseline protocols under various network conditions. The Q-learning parameters determined in Phase 1 will be fixed for our proposed protocol.

4.5.5.1 The simulated protocols

This section evaluates the performance of Q-learning-based TDMA (Q-TDMA) against S-ALOHA, and CSMA MAC protocols. Q-TDMA serves as the primary reference due to its intelligent time-slot allocation, where nodes dynamically optimize transmission schedules using reinforcement learning. Unlike traditional TDMA, Q-TDMA adapts to traffic demands without rigid synchronization, minimizing collisions and maximizing channel efficiency. To demonstrate Q-TDMA's advantages, we compare it with S-Aloha and CSMA.

By benchmarking against Q-TDMA; we took two variants where the number of nodes is equal to the number of slots thus a 1/1 ratio, as for the other variant the number of nodes is half the number of slots thus a 1/2 ratio. We highlight the trade-offs between learning-driven scheduling (Q-TDMA), and pure random access (S-ALOHA/CSMA).

Table 4.4: MAC Protocol Abbreviations

Abbreviation	Protocol
S-ALOHA	Slotted ALOHA
CSMA	Carrier Sense Multiple Access
Q-TDMA 1	Q-Learning Time Division Multiple Access with 100% (number of nodes = number of slots)
Q-TDMA ½	Q-Learning Time Division Multiple Access with 50% (number of nodes = ½ number of slots)

This section evaluates the performance of Q-learning-based TDMA (Q-TDMA) against S-ALOHA, and CSMA MAC protocols. Q-TDMA leverages reinforcement learning to dynamically optimize slot selection, minimizing collisions in wireless sensor networks. Below is the pseudocode for the Q-TDMA slot selection (exploitation phase) as implemented in OMNeT++ simulations:

Algorithm 2: Slot Selection (Exploitation) for OMNeT++ Simulation

```
Data: Nodes, slots, max_episodes, α
Result: Collision log, Q-tables
Initialize:
for each node do
    Set current_slot = random slot;
    Initialize Q-table as zero matrix [num_slots × 2];
for episode = 1 to max_episodes do
    for each node do
        action = arg max(Q[current_slot]) ;          // Exploit best known action
    for each node do
        if action == switch then
            current_slot = random slot;
        else
            stay on current_slot;
    Count nodes per slot;
    for each node do
        if node's slot has exactly 1 node then
            reward = +10;
        else
            reward = -10;
    for each node do
        Update Q[old_slot, action] += α· (reward - Q[old_slot, action]);
    Record collisions this episode;
Plot collisions per episode;
Print Q-tables;
```

To validate Q-TDMA's effectiveness, we conducted large-scale simulations with 50 nodes competing for 50 slots over 1,000 episodes. The results demonstrate how Q-learning progressively optimizes slot selection to minimize collisions:

Table 4.5: Simulation Results for 50 Nodes, 50 Slots

Episode	**Collisions**	**Success Rate (%)**	**Avg. Reward**	**Notes**
1	34	32	-4.2	Random slot selection, high collisions
2	26	48	2.5	Early Q-learning adjustments
10	16	68	6.8	Learning stabilization begins
50	6	88	8.5	Optimal slot distribution emerging
100	4	92	9.2	Stable performance achieved
500	2	96	9.7	Highly optimized slot selection
1000	1	97.9	9.8	Final results, minimal collisions

We took a snapshot of slot occupancy at Round 150 to illustrates Q-TDMA's adaptive distribution: (0=unused, 1=single node, 2=colliding nodes)

Slot [00-09]: [1, 0, 0, 1, 2, 1, 0, 3, 0, 1]
Slot [10-19]: [1, 2, 1, 1, 0, 1, 0, 1, 0, 1]
Slot [20-29]: [1, 0, 1, 1, 1, 2, 1, 0, 1, 1]
Slot [30-39]: [0, 1, 1, 3, 1, 0, 1, 2, 1, 0]
Slot [40-49]: [1, 0, 1, 1, 0, 1, 1, 1, 1, 1]

4.5.5.2 Simulation Parameters

To evaluate the performance of Q-TDMA against other MAC protocols, including DS-CSMA and IEEE 802.15.4 contention-based schemes, simulations were conducted in OMNeT++ for Q-TDMA and NS-2 for comparative protocols, all operating under the IEEE 802.15.4 physical layer for low-rate wireless personal area networks (LR-WPANs).

Nodes generated 300-byte packets at 3 Mbit/s, with a 1 ms time slot duration (accounting for T-Guard, AIFS, and CWmin intervals) and a 300-meter communication/carrier sensing range under free-space propagation. Unlike contention-based protocols prone to collisions, Q-TDMA—simulated in OMNeT++—leveraged quasi-synchronous time-slotted access to guarantee collision-free transmissions, deterministic latency, and efficient channel utilization.

Table 4.6: Simulation Parameters

Parameter	**Value**
Physical Layer	IEEE 802.15.4 (LR-WPAN)
Packet Size	300 bytes
Data Rate	3 Mbps
Time Slot Duration	1 ms
Communication Range	300 meters
AIFS	16 μs
Physical Slot Duration	9 μs
CW_{min}	6
CW_{max}	Never used (unicast only)
Q-TDMA Hyperparameters	$\epsilon = 0.1$, α, γ

The network configuration consists of a central sink node surrounded by uniformly distributed sensor units within a defined coverage area. All sensor nodes maintain symmetric communication links with the sink node through fixed transmission ranges. The system employs periodic broadcast messaging, where each sensor node transmits data packets to the central sink at regular intervals.

Under ideal channel conditions with no external interference, packet errors occur exclusively due to transmission collisions. The random node distribution follows a uniform and independent placement pattern across the entire network area, ensuring unbiased spatial coverage while maintaining direct connectivity to the central sink.

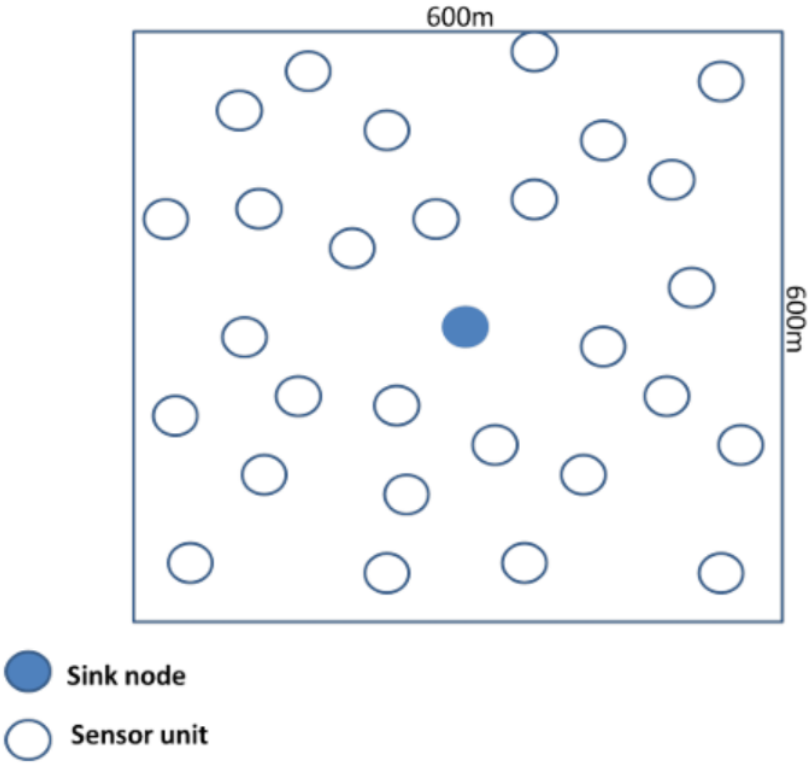


Figure 4.25: Representation of the simulated area.

The experimental setup evaluates protocol performance in handling merging collisions across both sparse and moderately dense network conditions. The analysis concentrates on collision-related impacts, particularly examining direct collisions and hidden terminal problems, with all reported results representing averaged measurements across the sensor network.

The simulation environment is specifically designed to test protocol robustness by incorporating realistic dynamic factors such as node mobility, collision mitigation effectiveness, and transmission reliability under varying network densities.

Table 4.7: The scenario parameters

Parameter	**value**
Simulation area	600 x 600
Number of nodes	[20...100]
Frequency (Hz)	10
Slot interval	1ms
Transmission range	300m
Time of simulation	100s

4.5.5.3 Simulation Resaults

The Average Medium Collisions probability

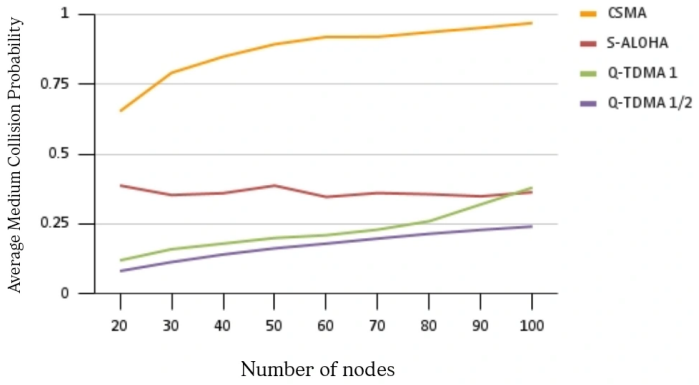


Figure 4.26: The average medium collision Probability.

Figure 4.26 shows the average medium collision probability across node densities relative to THC. CSMA MAC (IEEE 802.15.4) exhibits the highest collision risk, increasing with network density, confirming its limitations in high-load scenarios.

S-ALOHA performs better by leveraging STT spreading to reduce control channel contention. Both Q-TDMA 1 and Q-TDMA ½ significantly lower access delay and collision probability, with Q-TDMA ½ achieving the best overall performance.

The protocol Q-TDMA ½ has the lowest rate of medium access and the fewest collisions compared to all the protocols above.

The Rx Throughput probability

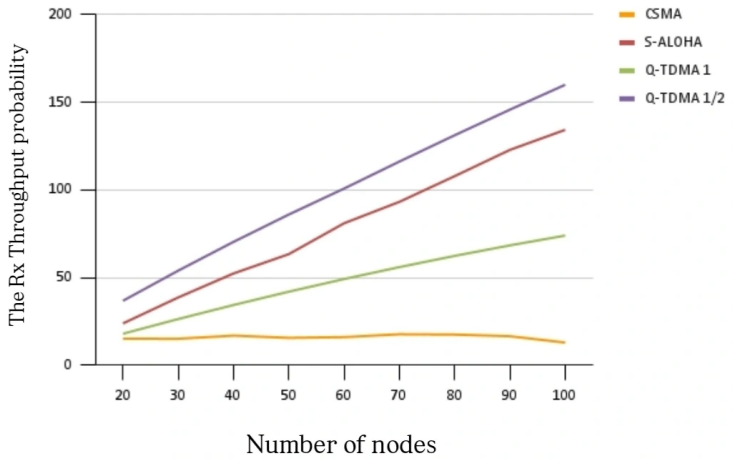


Figure 4.27: The Rx Throughput Probability.

Figure 4.27 shows RxThroughput performance across increasing network density. CSMA MAC maintains low throughput, reflecting its poor scalability under congestion. Q-TDMA 1 improves with node count, indicating better scalability.

S-ALOHA and Q-TDMA ½ achieve the highest throughput, with Q-TDMA ½ slightly outperforming S-ALOHA due to better coordination, reduced collisions, and lower control overhead—ensuring efficient slot usage in dense networks.

The Beacon Delivery probability

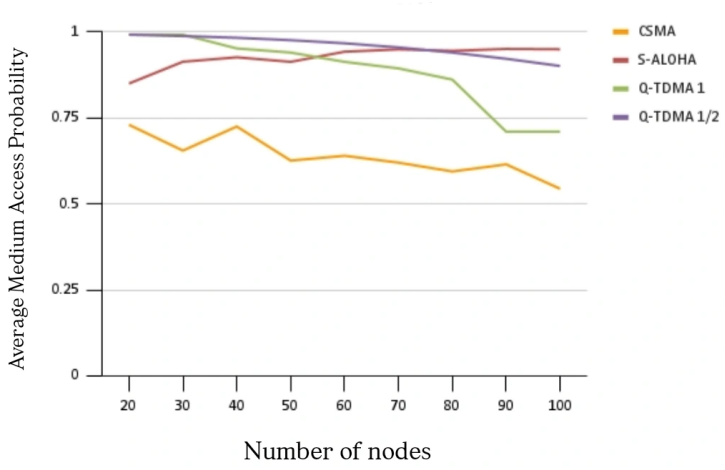


Figure 4.28: The Beacon Delivery Probability.

Figure 4.28 compares beacon delivery ratios as network density increases. CSMA MAC shows sharp performance degradation due to rising collisions. S-ALOHA maintains stable delivery, while Q-TDMA 1 and Q-TDMA ½ consistently achieve the highest ratios (0.9), demonstrating strong robustness and scalability.

These results highlight Q-TDMA's architectural advantage in ensuring reliable communication under dense network conditions.

The Access Fairness probability

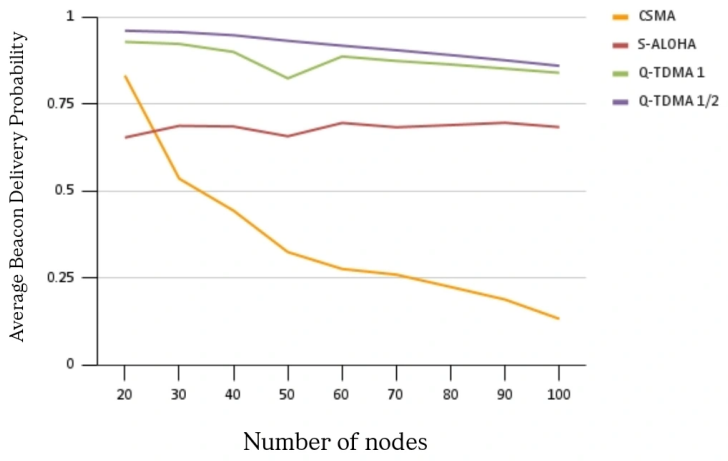


Figure 4.29: The Access Fairness Probability.

Figure 4.29 shows that CSMA MAC's fairness declines with network density, reflecting poor scalability. S-ALOHA maintains stable and high fairness due to its channel partitioning strategy.

Q-TDMA 1 and Q-TDMA ½ offer moderate fairness, with the latter approaching S-ALOHA's performance. Overall, S-ALOHA provides the most equitable access as node count increases.

4.5.6 Discussion of Results and Implications

The study demonstrates the effectiveness of the Q-learning-enhanced TDMA MAC protocol (Q-TDMA) for wireless sensor networks. In Phase 1, tuning identified optimal parameters: $\alpha = 0.1$, $\gamma = 0.5$, and exponentially decaying ϵ ($1.0 \rightarrow 0.1$), balancing adaptation and stability.

In Phase 2, Q-TDMA outperformed CSMA and S-ALOHA in dense scenarios—reducing collisions, improving throughput, and achieving high fairness (Jain's Index > 0.99), especially with Q-TDMA ½ (slots $= 2\times$ nodes). Its decentralized learning supports scalability in dynamic, high-density IoT applications.

Despite its promise, Q-TDMA assumes ideal conditions and static topologies. Future work should address mobility, interference, and scalability via Deep Q-Networks (DQNs), as well as validate performance through hardware testing.

In conclusion, Q-TDMA offers a dynamic, fair, and energy-efficient MAC solution for WSNs. Further research will help confirm its real-world viability and expand its potential in intelligent networking.

Chapter 5

General Conclusion

5.1 Summary of Work

This thesis addressed the communication challenges in smart farming environments, focusing on the critical issue of packet collisions in wireless sensor networks (WSNs). These collisions severely impact the reliability, energy efficiency, and real-time performance of smart farming systems. After reviewing the existing MAC protocols and their limitations, particularly in rural agricultural settings, we identified the need for adaptive, learning-based communication solutions. To this end, we proposed a Q-learning-based (TDMA) (MAC) protocol that enables sensor nodes to autonomously learn optimal time slot allocations, thereby minimizing collisions and improving network efficiency without the need for centralized scheduling. The proposed solution integrates reinforcement learning into the (MAC) layer, allowing nodes to adaptively select transmission slots based on their experiences in the network. We presented the protocol's system model, detailed its learning mechanism, and implemented it in a simulation environment to evaluate its performance. The results showed that the Q-learning-enhanced (TDMA) approach significantly reduces collision probability and improves key performance metrics such as throughput, fairness, and reliability—making it well-suited for smart farming scenarios where timely and energy-efficient data transmission is essential. In conclusion, this work demonstrates that Q-learning is an effective method for dynamic, decentralized medium access control in agricultural WSNs. It contributes to the advancement of intelligent (MAC) design by addressing both performance and adaptability, and lays the groundwork for future research in integrating more advanced learning techniques and security mechanisms tailored to the needs of precision agriculture.

5.2 Future Perspectives

While this thesis demonstrated the effectiveness of Q-learning for optimizing TDMA-based (MAC) protocols in smart farming scenarios, several directions can be explored to further enhance system performance and applicability:

- **Integration of Deep Reinforcement Learning (DRL):** Future work can incorporate Deep Q-Networks (DQN) or other DRL techniques to better handle complex state spaces and improve scalability when dealing with a large number of nodes and dynamic environmental conditions.
- **Energy-Aware Learning Models:** Incorporating energy metrics into the reward function could enable nodes to balance collision reduction with energy efficiency, prolonging network lifetime—crucial in battery-powered rural deployments.
- **Real-World Deployment and Validation:** Testing the proposed protocol on actual sensor nodes in real smart farming environments would provide deeper insights into its practicality and performance under realistic conditions.
- **Security Enhancements:** Future protocols should consider integrating lightweight security mechanisms to protect against common threats in IoT-enabled agriculture, such as spoofing, jamming, and data manipulation.
- **Hybrid Protocols:**A promising direction is the development of hybrid MAC protocols that combine Q-learning with other strategies (e.g., contention-based or centralized scheduling) to adapt to heterogeneous network conditions.
- **Context-Aware Adaptive Learning:** Integrating environmental context (e.g., crop type, weather, soil conditions) into the learning process could lead to more intelligent decision-making and application-specific optimizations.

Bibliography

[1] X. Yang, Y. Zhou, S. Yang, X. Wang, and L. Deng. A survey on smart agriculture: Development modes, technologies, and security and privacy challenges. *IEEE/CAA Journal of Automatica Sinica*, 8(2):273–302, February 2021.

[2] Sarra Djaafari. *Proposition and Simulation of LEACH-MQTT Routing Protocol for Publish/Subscribe Systems in WSN-based IoT Networks*. Phd thesis, University of Ammar Telidji Laghouat, 2022/2023.

[3] M. Bahallah and T. Djir. Medium access control in wsn network for water management system. Master's thesis, University of Ammar Telidji Laghouat, Faculty of Sciences, Department of Computer Science, 2023/2024.

[4] Daniyal Alghazzawi, Omaima Bamasaq, Surbhi Bhatia, Ankit Kumar, Pankaj Dadheech, and Aiiad Albeshri. Congestion control in cognitive iot-based wsn network for smart agriculture. *IEEE Access*, 9:151401–151420, 2021.

[5] Encyclopedia. Standards for wireless sensor networks in critical infrastructures, 2023.

[6] Z. Zheng, S. Jiang, R. Feng, L. Ge, and C. Gu. Survey of reinforcement-learning-based mac protocols for wireless ad hoc networks with a mac reference model. *Entropy*, 25(1):101, 2023.

[7] M. A. Ferrag, L. Shu, X. Yang, A. Derhab, and L. Maglaras. Security and privacy for green iot-based agriculture: Review, blockchain solutions, and challenges. *IEEE Access*, 8:32031–32053, 2020.

[8] V. V. Hari Ram, H. Vishal, S. Dhanalakshmi, and P. M. Vidya. Regulation of water in agriculture field using internet of things. In *Proceedings of the IEEE Technological Innovation in ICT for Agriculture and Rural Development*, pages 112–115, Chennai, India, 2015. IEEE.

[9] Y. Lin, J. R. Petway, J. Anthony, H. Mukhtar, S. Liao, C. Chou, and Y. Ho. Blockchain: The evolutionary next step for ict e-agriculture. *Environments*, 4(3):50, 2017.

[10] D. Dalohoun, A. Hall, and P. Van Mele. Entrepreneurship as driver of a "self-organizing system of innovation": the case of nerica in benin. *International Journal of Technology Management and Sustainable Development*, 8(2):87–101, 2009.

[11] M. F. Bellemare and J. R. Bloem. Does contract farming improve welfare? a review. *World Development*, 112:259–271, 2018.

[12] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda. Blockchain-based traceability in agri-food supply chain management: a practical implementation. In *Proc. IoT Vertical and Topical Summit on Agriculture*, pages 1–4. IEEE, 2018.

[13] K. Leng, Y. Bi, L. Jing, H. Fu, and I. Van Nieuwenhuyse. Research on agricultural supply chain system with double chain architecture based on blockchain technology. *Future Generation Computer Systems*, 86:641–649, 2018.

[14] J. Hua, X. Wang, M. Kang, H. Wang, and F.-Y. Wang. Blockchain based provenance for agricultural products: a distributed platform with duplicated and shared bookkeeping. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 97–101. IEEE, 2018.

[15] A. Kumar, M. Zhao, K. J. Wong, Y. L. Guan, and P. H. J. Chong. A comprehensive study of iot and wsn mac protocols: Research issues, challenges and opportunities. *IEEE Access*, 6:76228–76262, 2018.

[16] Oracle. Internet of things (iot). `https://www.oracle.com/internet-of-things/`, n.d. Available online.

[17] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, pages 102–114, August 2002.

[18] B. Kiruthika and P. Shyamala Bharathi. Intelligent dynamic trust secure attacker detection routing for wsn-iot networks. *Mathematical Biosciences and Engineering*, 20(2):4243–4257, 2022.

[19] Chiara Buratti, Andrea Conti, Davide Dardari, and Roberto Verdone. An overview on wireless sensor networks technology and evolution. *Sensors*, 9(8):6869–6896, 2009.

[20] Muhaammad R. Ahmed, Xu Huang, Dharmendra Sharma, and Hongyan Cui. Wireless sensor network: Characteristics and architectures. *World Academy of Science, Engineering and Technology International Journal of Information and Communication Engineering*, 6(12), 2012.

[21] S.R. Jino Ramson and D. Jackuline Moni. Applications of wireless sensor networks–a survey. In *Proceedings of IEEE International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICIEEIMT)*, pages 325–329, November 2017.

[22] Dirk Trossen, Dana Pavel, Glenn Platt, Joshua Wall, Philip Valencia, et al. Sensor networks, wearable computing, and healthcare applications. *IEEE Pervasive Computing*, 6(2):58–61, April 2007.

[23] Ishfaq Ahmad, Khalil Shah, Saif Ullah, et al. Military applications using wireless sensor networks: A survey. *International Journal of Engineering Science and Computing*, 6(6):7039–7043, June 2016.

[24] Andrea J. Goldsmith and Stephen B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications*, 9(4):08–27, August 2002.

[25] Ala Eddine Boulifa and Mohammed Habes. Étude comparative entre deux protocoles de routage géographiques sans balises blr et boss dans les réseaux wsns. Master's thesis, University Kasdi Merbah Ouargla, 2020.

[26] Riad Aouabed. *Cross-layer routing protocols in wireless sensor networks*. Doctoral thesis, Ferhat Abbas University Setif, 2023.

[27] GeeksforGeeks. Mac protocol used in wireless sensor networks.

[28] SemiEngineering. Ieee 802.15 wireless specialty networks (wsn).

[29] Syscor Controls & Automation. Wirelesshart, 2024.

[30] DoctorFlamingo3543. Wireless sensor network notes, 2023.

[31] Dheyab Salman and Qahtan Yas. Challenges and issues for wireless sensor networks: A survey. *Journal Name*, 2021.

[32] Kazem Sohraby, Daniel Minoli, and Taieb Znati. *Wireless Sensor Networks: Technology, Protocols, and Applications*. Wiley, 2007.

[33] Sara Amendola, Rossella Lodato, Stefano Manzari, Cecilia Occhiuzzi, and Gaetano Marrocco. Rfid technology for iot-based personal healthcare in smart spaces. *IEEE Internet of Things Journal*, 1(2):144–152, 2014.

[34] Kenneth Montgomery et al. Lifeguard—a personal physiological monitor for extreme environments. In *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pages 2192–2195, 2004.

[35] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020.

[36] Sumit Das, Aritra Dey, Akash Pal, and Nabamita Roy. Applications of artificial intelligence in machine learning: Review and prospect. *International Journal of Computer Applications*, 115(9), 2015.

[37] Y. Khene. Offloading pour les mobile edge computing à l'aide de l'apprentissage par renforcement. Master's thesis, University of Amar Telidji Laghouat, Faculty of Sciences, Department of Computer Science, 2023/2024.

[38] C.A. Azencott. *Introduction au Machine Learning*. Info sup. Dunod, 2019.

[39] Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Apprentissage par renforcement exploitant la structure additive des mdp factorisés. In *JFPDA 2007 - 2e Journées Francophones Planification, Décision, Apprentissage pour la conduite de système*, pages 49–60. Cépaduès, 2007.

[40] B. Jang, M. Kim, G. Harerimana, and J. W. Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7:133653–133667, 2019.

[41] LinkedIn. What are the most suitable protocols and standards? `https://www.linkedin.com/`.

Annexes

Appendix A

Algorithms

This annex contains code listings, extended explanations, or other supplementary material.

A.0.1 Q-Learning-Based TDMA Omnetpp Simulation

A.0.1.1 CentralControllerApp.cc

```
#include "CentralControllerApp.h"
#include <inet/common/INETUtils.h>
#include <inet/common/packet/Packet.h>
#include "inet/common/socket/SocketTag_m.h"

Define_Module(CentralControllerApp);

void CentralControllerApp::initialize(int stage)
{
    ApplicationBase::initialize(stage);
    if (stage == INITSTAGE_LOCAL) {
        port = par("port");
        ackPort = par("ackPort");
        numNodes = par("numNodes");
        messageLength = par("messageLength");
        nextBroadcastDelay = par("nextBroadcastDelay");
        cMessage *timer = new cMessage("broadcastTimer");
        destAddressesStr = par("destAddresses").stdstringValue();
        socket.setOutputGate(gate("socketOut"));
        ackListener = new cMessage("ackWait");
        scheduleAt(simTime() + 1, timer); // Initial delay
    }
}

void CentralControllerApp::handleMessageWhenUp(cMessage *msg)
{
    if (msg->isSelfMessage()) {
        if (strcmp(msg->getName(), "broadcastTimer") == 0) {
            sendBroadcast();
        }
    } else {
        handleAck(check_and_cast<inet::Packet *>(msg));
```

```
33      }
34  }
35
36  void CentralControllerApp::sendBroadcast()
37  {
38      acksReceived.clear();
39      slotCounts.clear();
40      nodeSlots.clear();
41
42      inet::Packet *pkt = new inet::Packet("Broadcast");
43      const auto& payload =
           inet::makeShared<inet::ByteCountChunk>(B(messageLength));
44      pkt->insertAtBack(payload);
45
46      cStringTokenizer tokenizer(destAddressesStr.c_str());
47      while (tokenizer.hasMoreTokens()) {
48          std::string dest = tokenizer.nextToken();
49          socket.sendTo(pkt->dup(),
               inet::L3AddressResolver().resolve(dest.c_str()), port);
50      }
51      delete pkt;
52  }
53
54  void CentralControllerApp::handleAck(inet::Packet *packet)
55  {
56      std::string src =
           packet->getTag<inet::SocketInd>()->getSrcAddress().str();
57      int slot = packet->getTag<inet::IntTag>("slot")->getInt();
58
59      acksReceived.insert(src);
60      nodeSlots[src] = slot;
61      slotCounts[slot]++;
62
63      delete packet;
64
65      if (acksReceived.size() >= (size_t)numNodes) {
66          sendFeedback();
67          scheduleAt(simTime() + nextBroadcastDelay, new
               cMessage("broadcastTimer"));
68      }
69  }
70
71  void CentralControllerApp::sendFeedback()
72  {
73      // Prepare feedback about collisions
74      for (const auto& [nodeAddr, slot] : nodeSlots) {
75          inet::Packet *feedback = new inet::Packet("Feedback");
76          auto payload = inet::makeShared<inet::ByteCountChunk>(B(50));
77
78          double reward = (slotCounts[slot] == 1) ? 10.0 : -10.0;
79          payload->setTag("reward", reward);
```

```
80          payload->setTag("slot", slot);
81
82          feedback->insertAtBack(payload);
83          socket.sendTo(feedback,
               inet::L3AddressResolver().resolve(nodeAddr.c_str()), ackPort);
84      }
85  }
86
87  void CentralControllerApp::finish() {
88      cancelAndDelete(ackListener);
89  }
```

Listing A.1: CentralControllerApp code example

A.0.1.2 CentralControllerApp.h

```
1   #ifndef __CENTRALCONTROLLERAPP_H
2   #define __CENTRALCONTROLLERAPP_H
3
4   #include <inet/applications/base/ApplicationBase.h>
5   #include <inet/transportlayer/contract/udp/UdpSocket.h>
6   #include <set>
7   #include <map>
8
9   class CentralControllerApp : public inet::ApplicationBase, public
       inet::UdpSocket::ICallback
10  {
11    protected:
12      inet::UdpSocket socket;
13      std::set<std::string> acksReceived;
14      std::string destAddressesStr;
15      int port, ackPort, numNodes;
16      double messageLength, nextBroadcastDelay;
17      omnetpp::cMessage *ackListener = nullptr;
18
19      // Collision tracking
20      std::map<int, int> slotCounts; // slot -> count
21      std::map<std::string, int> nodeSlots; // node address -> slot
22
23    protected:
24      virtual void initialize(int stage) override;
25      virtual void handleMessageWhenUp(omnetpp::cMessage *msg) override;
26      virtual void sendBroadcast();
27      virtual void handleAck(inet::Packet *packet);
28      virtual void finish() override;
29      virtual void sendFeedback();
30  };
31  #endif
```

Listing A.2: CentralControllerApp.h header file

A.0.1.3 ResponderApp.cc

```
#include "ResponderApp.h"
#include <inet/common/INETUtils.h>
#include <inet/common/packet/Packet.h>
#include <inet/applications/common/SocketTag_m.h>
#include <random>

Define_Module(ResponderApp);

void ResponderApp::initialize(int stage)
{
    ApplicationBase::initialize(stage);
    if (stage == INITSTAGE_LOCAL) {
        localPort = par("localPort");
        ackPort = par("ackPort");
        minBackoff = par("minBackoff");
        maxBackoff = par("maxBackoff");
        centralAddress =
           inet::L3AddressResolver().resolve(par("centralAddress"));

        // Initialize Q-learning
        numSlots = par("numSlots");
        qTable.resize(numSlots, std::vector<double>(2, 0.0)); // 2
           actions: 0=stay, 1=switch
        currentSlot = intuniform(0, numSlots - 1);

        socket.setOutputGate(gate("socketOut"));
        socket.bind(localPort);
    }
}

void ResponderApp::handleMessageWhenUp(cMessage *msg)
{
    if (msg->isSelfMessage()) {
        sendAck();
        delete msg;
    } else {
        lastSlot = currentSlot;
        lastAction = chooseAction();

        // Execute action
        if (lastAction == 1) { // switch
            currentSlot = intuniform(0, numSlots - 1);
        }
        // else stay in current slot

        delete msg;
        scheduleAt(simTime() + currentSlot * 0.001, new
           cMessage("ackTimer")); // Use slot-based timing
    }
```

```
}

void ResponderApp::sendAck()
{
    auto pkt = new inet::Packet("ACK");
    auto payload = inet::makeShared<inet::ByteCountChunk>(B(100));

    // Include slot information in the ACK
    payload->setTag("slot", currentSlot);
    pkt->insertAtBack(payload);

    socket.sendTo(pkt, centralAddress, ackPort);
}

int ResponderApp::chooseAction()
{
    // Exploit: choose best action for current slot
    if (qTable[currentSlot][0] > qTable[currentSlot][1]) {
        return 0; // stay
    }
    return 1; // switch
}

void ResponderApp::updateQTable(int slot, int action, double reward)
{
    double oldValue = qTable[slot][action];
    qTable[slot][action] = oldValue + alpha * (reward - oldValue);
}
```

Listing A.3: ResponderApp.cc source file

A.0.1.4 ResponderApp.h

```
#ifndef __RESPONDERAPP_H
#define __RESPONDERAPP_H

#include <inet/applications/base/ApplicationBase.h>
#include <inet/applications/common/UdpSocket.h>
#include <vector>

class ResponderApp : public inet::ApplicationBase
{
  protected:
    inet::UdpSocket socket;
    inet::L3Address centralAddress;
    int localPort, ackPort;
    double minBackoff, maxBackoff;

    // Q-learning parameters
    int numSlots = 50;
    int currentSlot;
```

```
    std::vector<std::vector<double>> qTable; // [slot][action]
    double alpha = 0.2;
    int lastAction = 0;
    int lastSlot = 0;

  protected:
    virtual void initialize(int stage) override;
    virtual void handleMessageWhenUp(cMessage *msg) override;
    virtual void sendAck();
    virtual void updateQTable(int slot, int action, double reward);
    virtual int chooseAction();
};
#endif
```

Listing A.4: ResponderApp.h header file

A.0.1.5 omnetpp.ini

```
[General]
network = Reseau
sim-time-limit = 100s
record-eventlog = true

# IPv4 address configuration
*.configurator.typename = "Ipv4NetworkConfigurator"
*.configurator.numHosts = 7  # 1 central + 6 neod
*.configurator.host[*].ipAddress = "10.0.0.{i+1}"
*.configurator.host[*].netmask = "255.255.255.0"

# Wireless channel
*.radioMedium.typename = "Ieee80211ScalarRadioMedium"
*.radioMedium.transmissionRange = 250m
*.radioMedium.interferenceRange = 300m"

# Central node configuration
*.centrale.typename = "AdhocHost"
*.centrale.numApps = 1
*.centrale.app[0].typename = "CentralControllerApp"
*.centrale.app[0].destAddresses = "neod[0] neod[1] neod[2] neod[3]
   neod[4] neod[5]"
*.centrale.app[0].port = 5000
*.centrale.app[0].ackPort = 6000
*.centrale.app[0].numNodes = 6
*.centrale.app[0].messageLength = 1000
*.centrale.app[0].nextBroadcastDelay = 1

# Responder nodes configuration
*.neod[*].typename = "AdhocHost"
*.neod[*].numApps = 1
*.neod[*].app[0].typename = "ResponderApp"
*.neod[*].app[0].centralAddress = "10.0.0.1"
```

```
*.need[*].app[0].localPort = 5000
*.need[*].app[0].ackPort = 6000
*.need[*].app[0].minBackoff = 0.01
*.need[*].app[0].maxBackoff = 0.1
*.need[*].app[0].numSlots = 50  # Number of time slots for Q-learning

# Visualization settings (optional)
*.visualizer.*.enable = true
*.visualizer.*.updateInterval = 1s
*.visualizer.*.animationSpeed = 50

# Logging verbosity (optional)
*.**.scalar-recording = true
*.**.vector-recording = true
```

Listing A.5: Simulation Configuration (omnetpp.ini)

A.1 The Python Simulation code

A.1.1 Q-Learning-Based TDMA Python Simulation

```
import numpy as np
import random
import matplotlib.pyplot as plt

class QLearningAgent:
    def __init__(self, num_slots, alpha=0.3, gamma=0.0, epsilon=1.0,
       epsilon_decay=0.99, epsilon_min=0.01):
        self.num_slots = num_slots
        self.q_table = np.zeros((num_slots, 2))  # Actions: 0 = stay, 1
           = switch
        self.alpha = alpha
        self.gamma = gamma
        self.epsilon = epsilon
        self.epsilon_decay = epsilon_decay
        self.epsilon_min = epsilon_min
        self.current_slot = random.randint(0, num_slots - 1)
        self.energy_used = 0.0
        self.tx_energy = 0.1
        self.collision_energy = 0.1
        self.idle_energy = 0.01

    def choose_action(self):
        if random.random() < self.epsilon:
            return random.choice([0, 1])
        else:
            return np.argmax(self.q_table[self.current_slot])

    def update_q(self, slot, action, reward):
        old_value = self.q_table[slot, action]
        self.q_table[slot, action] = old_value + self.alpha * (reward -
           old_value)

    def act(self, action):
        if action == 1:
            self.current_slot = random.randint(0, self.num_slots - 1)

class TDMAEnvironment:
    def __init__(self, num_nodes, num_slots):
        self.num_nodes = num_nodes
        self.num_slots = num_slots

    def step(self, agents):
        slot_counts = [0] * self.num_slots
        slots_selected = []

        for agent in agents:
            slots_selected.append(agent.current_slot)
```

```
            slot_counts[agent.current_slot] += 1

        rewards = []
        for i, agent in enumerate(agents):
            slot = slots_selected[i]
            rewards.append(10 if slot_counts[slot] == 1 else -10)

        return rewards, slot_counts

def moving_average(data, window_size=50):
    return np.convolve(data, np.ones(window_size)/window_size,
       mode='valid')

def main():
    num_nodes = 50
    num_slots = 50
    episodes = 1000
    agents = [QLearningAgent(num_slots) for _ in range(num_nodes)]
    env = TDMAEnvironment(num_nodes, num_slots)
    collisions_per_episode = []
    throughputs = []
    node_successes = [0 for _ in range(num_nodes)]

    for episode in range(episodes):
        actions = [agent.choose_action() for agent in agents]
        old_slots = [agent.current_slot for agent in agents]

        for i, agent in enumerate(agents):
            agent.act(actions[i])

        rewards, _ = env.step(agents)

        for i, agent in enumerate(agents):
            agent.update_q(old_slots[i], actions[i], rewards[i])
            agent.epsilon = max(agent.epsilon * agent.epsilon_decay,
               agent.epsilon_min)

            if rewards[i] == 10:
                agent.energy_used += agent.tx_energy
                node_successes[i] += 1
            elif rewards[i] == -10:
                agent.energy_used += agent.collision_energy
            else:
                agent.energy_used += agent.idle_energy

        collisions = rewards.count(-10)
        successes = rewards.count(10)
        collisions_per_episode.append(collisions)
        throughputs.append(successes / num_nodes)

        if episode % 50 == 0 or episode == episodes - 1:
```

```
            print(f"Episode {episode}: Epsilon={agents[0].epsilon:.3f},
                Collisions={collisions}, Successes={successes}")

    # Plotting code follows...

if __name__ == "__main__":
    main()
```

Listing A.6: Q-Learning-Based TDMA Python Simulation