UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Políticas de Control de Congestión basadas en Reinforcement Learning para mMTC en Redes 5G

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Telecomunicación

AUTOR/A: Agrag , Ecem Nur

Tutor/a: Martínez Bauset, Jorge

Cotutor/a externo: CESANA, MATTEO

CURSO ACADÉMICO: 2022/2023

Abstract

In recent years, the emergence of massive machine type communications (mMTC) has led to a significant increase in the number of connected devices in 5G networks. This rapid growth of mMTC devices has created a new challenge for network operators, as the congestion caused by these devices can result in network failure and service degradation. To address this issue, the Access Class Barring (ACB) method has been implemented in 5G cellular networks. It aims to increase the probability of successful access by randomly delaying access requests of User Equipments (UEs) based on a barring rate and a barring time. Proper selection of those parameters is essential for effective congestion control. However, the 3GPP does not provide any specific algorithm for setting and adapting these parameters. This study focuses on a simplified version of the ACB algorithm using Reinforcement Learning (RL) to dynamically adapt the access probability (barring rate) to maximize network performance. A grant-free access type protocol has been used in this scenario to reduce energy consumption, as they minimize the signalling need for network access. The proposed scheme was evaluated using discrete-event simulation and compared with an ideal and a heuristic congestion control schemes. The results show that RL-based congestion control policies can effectively reduce collisions and improve network efficiency but may require careful tuning of hyperparameters to achieve optimal performance across different metrics.

Keywords: Reinforcement Learning, Access Class Barring, Grant-Free Access, mMTC, IoT.

Resum

En els últims anys, l'aparició de comunicacions massives de tipus màquina (mMTC) ha provocat un augment significatiu del nombre de dispositius connectats a les xarxes 5G. Aquest ràpid creixement dels dispositius mMTC ha creat un nou repte per als operadors de xarxa, ja que la congestió causada per aquests dispositius pot provocar una fallada de la xarxa i la degradació del servei. Per solucionar aquest problema, s'ha implementat el mètode Access Class Barring (ACB) a les xarxes cellulars 5G. El seu objectiu és augmentar la probabilitat d'èxit d'accés retardant aleatòriament les sol·licituds d'accés dels equips d'usuari (UE) en funció d'una taxa de restricció i un temps de restricció. La selecció adequada d'aquests paràmetres és essencial per a un control efectiu de la congestió. Tanmateix, el 3GPP no proporciona cap algorisme específic per configurar i adaptar aquests paràmetres. Aquest estudi se centra en una versió simplificada de l'algorisme ACB que utilitza Reinforcement Learning (RL) per adaptar dinàmicament la probabilitat d'accés (taxa de restricció) per maximitzar el rendiment de la xarxa. En aquest escenari s'ha utilitzat un protocol d'accés del tipus 'grant-free' per reduir el consum d'energia dels dispositius al minimitzar la necessitat de senyalització per a l'accés a la xarxa. L'esquema proposat es va avaluar mitjançant simulació d'esdeveniments discrets i es va comparar amb un esquema de control de congestió ideal i un heurístic. Els resultats mostren que les polítiques de control de la congestió basades en RL poden reduir eficaçment les collisions i millorar l'eficiència de la xarxa, però poden requerir una ajustada acurada dels hiperparàmetres per aconseguir un rendiment òptim en diferents mètriques.

Paraules clau: aprenentatge de reforç, prohibició d'accés a classe, accés grant-free, mMTC, IoT.

Resumen

En los últimos años, la aparición de las comunicaciones masivas de tipo máquina (mMTC) ha provocado un aumento significativo del número de dispositivos conectados en las redes 5G. Este rápido crecimiento de los dispositivos mMTC ha creado un nuevo reto para los operadores de redes, ya que la congestión causada por estos dispositivos puede provocar fallos en la red y degradación del servicio. Para hacer frente a este problema, se ha implantado el método Access Class Barring (ACB) en las redes celulares 5G. Su objetivo es aumentar la probabilidad de éxito del acceso retrasando aleatoriamente las solicitudes de acceso de los equipos de usuario (UE) en función de una tasa y un tiempo de restricción. La selección adecuada de estos parámetros es esencial para un control eficaz de la congestión. Sin embargo, el 3GPP no proporciona ningún algoritmo específico para establecer y adaptar estos parámetros. Este estudio se centra en una versión simplificada del algoritmo ACB que utiliza el aprendizaje por refuerzo (RL) para adaptar dinámicamente la probabilidad de acceso (tasa de restricción) con el fin de maximizar el rendimiento de la red. En este escenario se ha utilizado un protocolo de acceso del tipo ’grant-free’ para reducir el consumo de energía de los dispositivos al minimizarse la necesidad de señalización para el acceso a la red. El esquema propuesto se evaluó mediante simulación de eventos discretos y se comparó con un esquema de control de congestión ideal y otro heurístico. Los resultados muestran que las políticas de control de la congestión basadas en RL pueden reducir eficazmente las colisiones y mejorar la eficiencia de la red, pero pueden requerir un ajuste cuidadoso de los hiperparámetros para lograr un rendimiento óptimo en diferentes métricas.

Palabras clave: Aprendizaje por refuerzo, bloqueo de clases de acceso, acceso grant-free, mMTC, IoT.

Acknowledgements

First and foremost, I owe a great debt of gratitude to Prof. Jorge Martínez-Bauset, my thesis supervisor, who has supported me throughout my thesis with his patience and vast knowledge. His valuable guidance and constant encouragement helped me complete this thesis. I cannot thank him enough for his tremendous support and help.

I would also like to express my sincere thanks to my co-supervisor, Prof. Matteo Cesana, for his support, time, and understanding throughout my thesis.

Furthermore, I would like to extend my gratitude to all my friends, especially Konstantin Chernaev, for his companionship, encouragement, and kindness during the writing period. His support has been a source of motivation and inspiration throughout this journey.

Lastly, and most importantly, I would like to express my heartfelt thanks to my beloved parents and sister for their invaluable presence, support, encouragement, and love.

Contents

Introduction

The evolution of the fifth generation (5G) networks has opened up a new era in wireless communication, promising to deliver ultra-low latency, high-speed data transmission, and high-reliability connectivity. These capabilities have paved the way for the development of massive machine type communications (mMTC), which involves the connection of a massive number of devices to the network, ranging from Internet of Things (IoT) devices to autonomous vehicles. As the world moves towards a new technological era where everything is connected, the demand for MTC and IoT communications via LTE-Advanced networks is growing each year. According to IoT connectivity industry forecasts, the global IoT market is projected to grow from $213 billion in 2021 to $621 billion in 2030, and the number of IoT devices worldwide is expected to exceed 29 billion by 2030 [1].

New generation cellular networks aim to provide extensive coverage through their widespread infrastructure, global connectivity, high quality of service (QoS), robust charging, and security solutions [2,3]. While cellular networks offer the most viable option for UE interconnection, the high density mMTC traffic poses a significant challenge for congestion control in a cellular network. The conventional mechanisms, such as the Transmission Control Protocol (TCP), are designed for low density human-generated traffic, and they are not capable of handling MTC. When a massive number of MTC devices try to access the base stations, severe congestion can occur, causing performance degradation for both MTC and human-to-human (H2H) communications [4,5]. Therefore, there is a need for new congestion control policies that can effectively manage the congestion caused by mMTC traffic.

The Access Class Barring (ACB) method has been implemented in 5G cellular networks to address this issue. ACB is one of the efficient and common approaches which is suggested in 3GPP specifications. It aims to increase the probability of successful access by randomly delaying access requests of User Equipments (UEs) based on a barring rate and a barring time, parameters broadcast by the BS [6]. Proper selection of these ACB parameters is essential for effective congestion control and optimal performance. However, the 3GPP does not provide any specific algorithm for setting and adapting these parameters. Determining how this parameter should be set and adapted in dynamic traffic conditions in scenarios with mMTC is challenging.

Many studies in the literature address the congestion control with ACB for mMTC. The optimization of ACB parameters has been analyzed in [7], [8], [9], and [10], and the performance analysis of ACB in [6] demonstrates its effectiveness for mMTC applications. The evaluation scenario examined in this study is taken from [6]. Unlike the scenario in [6], we only considered what is referred to as low priority traffic there, i.e., mMTC traffic. In addition, instead of using a stationary arrival regime (binomial packet arrivals to nodes), we used the arrival regime proposed by the 3GPP [11], where devices become active (they start contending for access) according to a Beta (3,4) distribution over 2000 subframes (10 seconds). Moreover, it is noteworthy that in contrast to the scenario proposed in the current study, the access probability in the scenario presented in [6] is broadcast in every subframe, rather than every 10 subframes. However, this is not implementable with the current definition of the eNode radio interface.

To enhance the performance of conventional ACB methods in complex and unpredictable 5G networks, reinforcement learning (RL) mechanisms are proposed to dynamically adapt the ACB parameters. This approach has been shown promising in optimizing network performance, as it allows for real-time decision-making and adaptation to uncertain network environments. RL is a type of machine learning algorithm that enables agents to learn optimal policies based on the feedback received from the environment. In the context of congestion control, RL algorithms can learn to allocate network resources in an optimal or quasy-optimal way by continuously observing the network's state and adjusting the congestion control parameters accordingly [12]. Several publications in the literature have analyzed the RL-based ACB mechanism for managing mMTC traffic, including the study presented in [9].

In 5G networks, managing congestion often involves controlling access, which is traditionally accomplished through a grant-based (GB) approach where UEs request network access and awaits a grant before transmitting their packets. However, controlling UE uplink accesses in this way can become challenging for mMTC, as the probability of collisions between multiple devices transmitting at once increases as the number of contending devices increases. This can result in significant delays, decreased network throughput, and higher energy consumption, particularly in situations where a vast number of devices are present [13]. Thus, GB access may not be the most suitable option for mMTC due to its limitations.

To overcome the limitations of grant-based (GB) access, researchers are exploring grant-free (GF) access methods. GF transmission involves user equipment (UE) transmitting data over a predetermined set of resources using a contention-based approach, without requiring explicit grants from the base station (BS) [13,14,15]. GF access enables devices to transmit small amounts of data, without waiting for a grant, resulting in lower latency and increased efficiency. Furthermore, GF access is more scalable for a large number of devices and reduces the energy consumption of devices by eliminating the need for frequent requests for network access [14].

In this study, we proposed a grant-free access protocol that utilizes a simplified version of the ACB scheme to improve the performance and adaptability of conventional methods in dynamic and complex environments, such as wireless 5G networks. The proposed scheme implements reinforcement learning access control algorithms at the BS to determine the access probability that maximizes network performance. Unlike the conventional ACB method, the BS broadcasts the access probability ϱ determined by the RL algorithm rather than broadcasting the barring time and rate. The access probability ϱ is broadcasted once every 10 subframes, referred to as a superframe. This type of delayed feedback conforms to the definition of the 3GPP radio interface at a eNode. Since there is no handshaking procedure between UEs and the BS in the GF access approach, UEs can send their packets without waiting for a grant from the BS.

The primary objectives of this master's thesis are to investigate the potential of RL-based congestion control policies for mMTC in 5G networks and to analyze the impact of various RL algorithm parameters and hyperparameters on the performance of congestion control mechanisms. To achieve these objectives, we evaluated the performance of the proposed congestion control scheme using discrete-event simulation and compared it with an ideal and a heuristic congestion control. Performance evaluation was made using three RL algorithms (Q-learning, Double Q-learning, and Expected Sarsa) and their associated parameters.

This thesis is organized as follows:

Chapter 1 provides a general overview of the random-access procedure in LTE-A networks. In the following, contention-based random access and access class barring are explained.

Chapter 2 gives theoretical background about the Markov decision process, value functions, and the Bellman equations. Then, it provides a general overview of reinforcement learning and its algorithms Q-learning, Double Q-learning, and Expected Sarsa.

Chapter 3 presents the details of the scenario designed to evaluate the performance of the proposed grant-free access protocol, assumptions, and performance parameters.

Chapter 4 demonstrates the achieved simulation results and evaluates the performance of our proposed method in detail, considering defined performance parameters.

Chapter 5 concludes the whole thesis, summarizes the methods followed and the results obtained, and provides insights to extend this work.

Objectives

The main objective of this master's thesis is to examine the potential of reinforcement learning (RL) algorithms for the design of congestion control policies in the analysis scenario defined by 3GPP for massive machine type communication (mMTC) traffic.

This study focusses on designing and implementing a grant-free access protocol using a simplified version of the access class barring mechanism to control the access to the upstream base station resources and minimize collisions. The studied protocol broadcasts an access probability to user equipment (UEs), such as sensors, every certain number of subframes, according to the 3GPP upstream subframe specification. Controlling UE accesses by BS with grant-free (GF) access instead of conventional grant-based (GB) access, where UEs request access to the network and wait for a grant before transmitting their packets, has enabled UEs to immediately transmit small amounts of data without waiting for grants. With GF access, lower latency and higher efficiency are achieved without handshaking.

Specifically, we aim to achieve the following objectives:

- Analyze the impact of various RL algorithm parameters and hyperparameters on the performance of congestion control mechanisms.
- Provide insights into the feasibility and effectiveness of RL algorithms for designing congestion control policies in mMTC networks.
- Evaluate the performance of RL-based congestion control policies in terms of subframe throughput, average delay, loss probability, 95 percentile of the delay, successful access probability, average number of collisions per successfully transmitted packet, and last subframe.
- Contribute to the development of more efficient and adaptive RL-based congestion control mechanisms for future mMTC networks.
- Compare the performance of RL-based policies with the heuristic policy and ideal, non-implementable policy that has complete state information of the system and generates the access probability accordingly and highlight their advantages and limitations.

Overall, this study aims to contribute to ongoing efforts towards developing effective and adaptive congestion control policies for mMTC networks that can learn from experience and adapt to changing network conditions. By achieving these objectives, we hope to provide valuable insights into the potential of RL-based mechanisms for improving network efficiency, reducing collisions, and enhancing quality-of-service for mMTC applications in 5G networks.

1 Random Access in LTE-A

This chapter provides a general overview of the random access procedure in LTE-A networks. In the following, contention-based random access and access class barring will be explained in sections 1.1 and 1.2, respectively.

Random access is an essential mechanism in LTE-A for supporting mMTC. mMTC is a key enabler of the Internet of Things (IoT) and refers to the communication between many low-power, low-cost, and low-data-rate devices. Random access provides an efficient way for MTC devices to transfer data packets to a base station.

It is essential for the MTC devices (UEs) to initiate the random access procedure to the base station (known as eNodeB in LTE) in the following five situations according to reference [16]:

1. during the initial access to the network, when establishing a connection;
2. when the device receives or transmits new data;
3. when no scheduling request resources are configured on the uplink control channel for transmitting new data;
4. during handover to prevent a session drop;
5. after a radio link failure in order to re-establish the connection.

To handle all these situations, two different modes of Random Access (RA) procedure are defined in LTE-A: contention-free and contention-based. The contention-free mode assigns orthogonal transmission resource units (resource blocks in OFDMA) that we refer to from now on as timeslots, to each device based on a pre-defined schedule to avoid collisions. It is used in situations such as downlink data arrival, positioning or handover. On the other hand, the contention-based mode involves devices contending for the transmission medium without prior coordination, which can lead to collisions. It is mostly used by UEs for changing the radio resource control state from idle to connected, recovering from a radio link failure, uplink synchronization, or sending scheduling requests [17]. In this study, we focus on the analysis of the contention-based random access procedure.

1.1. Contention-based random access

Contention-based random access is a method of accessing a shared communication medium, such as a wireless network, where multiple devices contend for the transmission medium at the same time without any prior coordination. Random access attempts of UEs are allowed in predefined time/frequency resources called random access opportunities (RAOs). Before initiating the procedure, UEs need to receive basic configuration parameters such as the available time-slots (RAOs) for transmitting preambles. This information is broadcast periodically by the eNodeB [6]. After obtaining this information, the UE can start the four-message handshake process illustrated in Figure 1.1 [19].

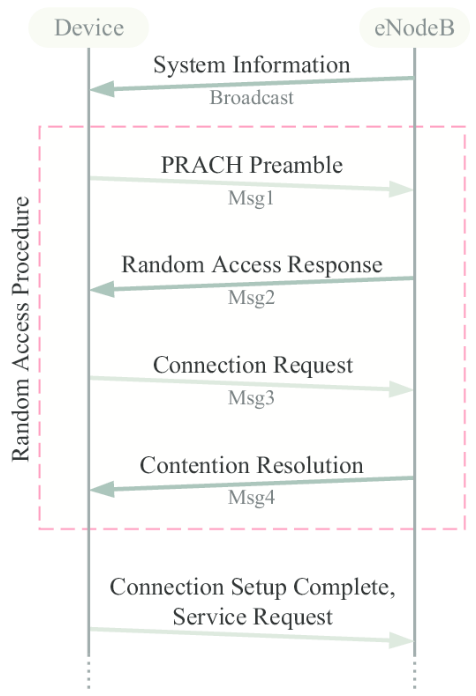


Figure 1.1: Contention-based random access procedure

Message 1, RACH preamble transmission: Each UE transmits a preamble as an access request to the eNB for a dedicated resource blocks in the upcoming RAO. This preamble is selected randomly from a pool of up to 64 orthogonal preambles known to both UEs and eNBs. If multiple devices send the same preamble in the same RA time-slot, a collision occurs. However, if different preambles are used, the eNB can distinguish them from each other due to their orthogonality. After detecting the preamble successfully, the eNB sends a random-access response message [17,10,20].

Message 2, random access response (RAR): It is responsible for allocating time-frequency resources for UEs to transmit Message 3. It contains one uplink grant for each detected preamble. UEs, wait for a predefined time window to receive the uplink grant. If no uplink grant is received by the end of this window and the maximum number of access attempts has not been reached, the UEs wait for a random time and then perform a new access attempt by selecting a new preamble and transmitting it at the next RAO [17,18,20].

Message 3, connection request: Once the UE has received the RAR, it sends a connection request with its ID using the uplink resource specified by the eNB. It indicates that the UE wants to establish a connection and begin transmitting data. In some cases, if the eNB correctly decodes the preambles transmitted by multiple UEs, they may transmit their connection request, over the same physical resources. This can cause a collision where multiple UEs transmit their messages simultaneously, resulting in interference and making it difficult or impossible for the eNB to decode the messages [17].

Message 4, contention resolution: Upon reception of a connection request, the eNB broadcasts a contention resolution message including the ID of related UE. Then eNB allocates the required data resources for UE. If UE does not receive a response to a preamble or a contention request message, it restarts the procedure. Each UE repeats this procedure until establishing a connection or reaching the maximum allowed number of preambles retransmissions [6].

By using this four-message handshake, contention-based access systems can ensure that only one device transmits data at a time, avoiding collisions and ensuring reliable data transfer.

1.1.1) Backoff procedure

According to the LTE-A standard [16], in case of failure during the RACH procedure, regardless of the cause, the UE has to perform a backoff procedure before re-transmit a new preamble in the next RAO. In order to reduce the collision, the UE waits for a random time, TBO [ms], until it can attempt to transmit a new preamble as follows

$$TBO = t\,(0, BI) \tag{1.1}$$

where $t(\cdot)$ stands for uniform distribution, BI is the backoff indicator broadcasted by the eNB in the RA response and its value ranges from 0 to 960 ms. The RA Response is read by all UEs which transmit a RACH preamble in the previous RAO. It is indicated that every UE that failed the access attempt receives the BI [9].

1.2. Access class barring

Access Class Barring (ACB) is a mechanism aimed at controlling congestion by limiting the maximum number of UEs that simultaneously access the eNB. It accomplishes this by categorizing all UEs into 16 different access classes (ACs) from 0 to 15 based on service requirements. MTC devices are assigned an AC between 0 and 9.

Other classes can be used to give priority access to specific MTC devices or groups of devices that require a higher level of service [20, 11].

ACB aims to reduce the number of access requests per RAO by redistributing access requests from UEs over time and it is applied only to UEs that have not yet started their random access procedures. In case ACB is not implemented, all ACs are allowed to access PRACH. With the implementation of ACB, the eNB broadcasts mean barring times, $T_{ACB} \in$ {4, 8, 16,…, 512 s}and barring rates, $P_{ACB} \in$ {0.05, 0.1,...,0.95} through System Information Block Type 2 (SIB2) for the upcoming RAO. Barring factors are generally applied to ACs 0-9, while the special categories are exempted from the barring process [11].

At the start of the random access procedure, each UE generates a random number q, between 0 and 1 ($t[0, 1)$), and if q is less than or equal to P_{ACB}, the UE selects and transmits its random preamble; otherwise, the UE waits for a random time calculated using the Equation (1.2).

$$T_{barring} = [0.7 + 0.6\, t[0,1)] \times T_{ACB} \tag{1.2}$$

This process is repeated until the UE generates a random number lower than P_{ACB} and sends its preamble. In this way, ACB reduces the number of access requests per RAO. Demonstration of ACB scheme is given in Figure 1.2 [9].

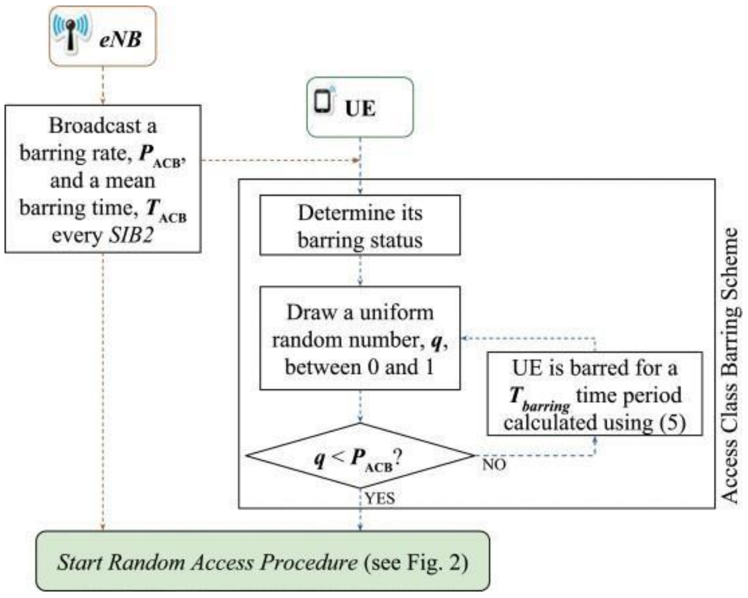

eNB
UE
Broadcast a barring rate, P_{ACB}, and a mean barring time, T_{ACB} every SIB2
Determine its barring status
Draw a uniform random number, q, between 0 and 1
UE is barred for a $T_{barring}$ time period calculated using (5)
$q < P_{ACB}$?
NO
YES
Access Class Barring Scheme
Start Random Access Procedure (see Fig. 2)

Figure 1.2: Access class barring scheme.

2 Reinforcement Learning Algorithms

Nowadays, 5G networks are expected to support various applications with diverse requirements, becoming increasingly heterogeneous and decentralized. Conventional resource management methods that rely on complete and accurate knowledge of the systems are impractical due to the unpredictable nature of the wireless network environments of 5G. To overcome this challenge, reinforcement learning (RL) has emerged as a viable solution for making real-time dynamic decisions in uncertain network situations.

A solid understanding of the Markov decision process is essential for understanding the concept of reinforcement learning. This chapter will begin by introducing the Markov decision process, as well as value functions and the Bellman equations, which are key components of the process. Following this, reinforcement learning will be introduced, and its algorithms Q-learning, Double Q-learning, and Expected Sarsa will be examined.

2.1. Markov decision process

A Markov decision process (MDP) allows modelling the evolution of the state of a system over time when the system follows a certain action policy. An action policy defines the action taken at each state. By assigning rewards to each of the possible actions that can be taken at each state, a policy that maximizes the long-term reward, the optimal policy, can be determined.

More formally, an MDP is characterized by the following elements:

1. States: A finite or infinite set of all possible conditions or situations that the system can be in [12].

2. Actions: A finite or infinite set of possible actions that can be chosen at each state [12].

3. Transition probabilities: The probability that the system transits from one state to another depends on the action taken and the present state of the system. This probability is referred to as the transition probability. Let

$$P^a_{ss'} = Pr\{S_{t+1} = s' | S_t = s, A_t = a\} \tag{2.1}$$

be the probability of moving to state s′ when action a is taken in state s, and t denotes the time [22].

4. Rewards: Rewards help to define the system operation goal. The system receives a reward each time an action is taken at any system state. This reward signal defines the immediate benefits (positive or negative) that result from taking the specific action in a particular state. The system should select the best actions to maximize the total reward it receives over time [12]. Let

$$R^a_{ss'} = E\{R_{t+1} | S_t = s, A_t = a\} \tag{2.2}$$

be the reward function, where t is the time, and E is the expected value for the reward [22].

5. Discount Factor: It is a number between 0 and 1 that represents how much the system values immediate rewards over future rewards. A discount factor of 0 means that only immediate rewards are valuable. While a discount factor of 1 means all rewards are equally valuable, regardless of how far in the future they occur [12].

The concept of **return** is commonly associated to the discount factor. The return at time *t* is the cumulative discounted reward obtained over the sequence of actions taken from time *t* onwards. It is defined as,

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

The discount factor $\gamma \in [0, 1]$ allows to define the present value of future rewards [3].

6. Policy: The policy is a mapping between a state and action taken at that state. There are two types of policies: deterministic policies and stochastic policies. A deterministic policy maps each state to a single action, while a stochastic policy maps each state to a probability distribution over actions. Deterministic and stochastic policies are given in Equation (2.4) and Equation (2.5), respectively,

$$a = \pi(s)\,, \tag{2.4}$$

$$\pi(a|s) = P[A_t = a | S_t = s], \tag{2.5}$$

where (2.4) π is the policy, i.e., the probability that, at time t, action a is chosen at state s [23,24].

2.1.1. Value functions

The value function represents the expected long-term return that the system can expect to receive by starting from a given state and following a specific policy. It assigns a value to each state or state-action pair, indicating how valuable that state or action is in terms of the future rewards that can be received.

The value function is important since it helps to choose the best action to take at each state in order to maximize the expected long-term return. In addition, the action-value function provides a way to estimate the expected future return for each possible action taken at each possible state. By choosing the action with the highest value at each state, the system will achieve the highest long-term return. In general, there are two types of value functions: state-value functions and action-value functions [12,24].

The state-value function $\boldsymbol{v_\pi(s)}$ is the expected return starting from state s, and then following policy π. We can define $v_\pi(s)$ formally by [12].

$$v_\pi(s) = E_\pi[\, G_t \mid S_t = s] \tag{2.6}$$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π. We can define $q_\pi(s, a)$ formally by [12].

$$q_\pi(s, a) = E_\pi[\, G_t \mid S_t = s, A_t = a] \tag{2.7}$$

where $E_\pi[\cdot]$ represent the expected value of a random variable given that the agent follows policy π and t is any time step.

The optimal state-value function $\boldsymbol{v_*}$(s) defines the maximum expected return when the system starts from state s over all policies,

$$v_*(s) = \max_\pi v_\pi(s) \tag{2.8}$$

The optimal action-value function q∗(s, a) defines the maximum of the expected return when the system starts from state s and takes action a overall policies,

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{2.9}$$

$\forall s \in S$ and $\forall a \in A(s)$[4]. By computing optimal state-values or action-values the optimal policy can be determined.

A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi^F}(s)$ for $\forall s \in S$. For any MDP [24],

- There is always at least one policy that is better than or equal to all other policies. It is called an optimal policy, $\pi_* \geq \pi$.
- All optimal policies achieve the optimal value function, $v_{\pi_*} \geq v_*(s)$.
- All optimal policies achieve the optimal action-value function, $q_{\pi_*} \geq q_*(s)$.

2.1.2. Bellman equations

The Bellman equations relate the state transition probabilities, state values, and long-term average reward. By iteratively solving the set of Bellman linear equations, the optimal actions at each state can be determined. However, to solve MDPs analytically, the state transition probabilities are required, which are not available in many practical scenarios.

The Bellman equation states that value functions can be decomposed into two parts: immediate reward and discounted future rewards. Using this equation makes the computation of the value function easier, as we can break down a complicated problem into simpler, recursive subproblems. The Bellman equation can be defined as [23,25]

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \tag{2.10}$$

2.1.2.1. Bellman expectation equation

The Bellman expectation equation is a specific form of the Bellman equation. It is used to calculate the expected value of a state under a given policy. The Bellman expectation equation for state-value functions is defined as [23]:

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \tag{2.11}$$

The action-value function can similarly be decomposed,

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a] \tag{2.12}$$

The state values can be expressed as a function of Q-values,

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a) \tag{2.13}$$

When necessary adjustments are made, the equations below are obtained.

$$q_\pi(s,a) = R^a_{ss'} + \gamma \sum_{s^F \in S} P^a_{ss^F} v_\pi(s') \tag{2.14}$$

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s)\left(R^a_{ss'} + \gamma \sum_{s^F \in S} P^a_{ss^F} v_\pi(s')\right) \tag{2.15}$$

$$q_\pi(s,a) = R^a_{ss'} + \sum_{s^F\in S} P^a_{ss^F} + \sum_{a'\in A} \pi(a'|s') q_\pi(s',a') \tag{2.16}$$

2.1.2.2. Bellman optimality equation

The Bellman optimality equations are used to determine the optimal value function and the optimal policy for a given problem. The equations express the optimal value of a state in terms of the maximum expected reward obtained by taking an optimal action in that state.

Bellman optimality equations for $v_*(\mathrm{s})$ and $q_*(\mathrm{s,a})$ are defined as [23]:

$$v_*(s) = \max_a R^a_{ss'} + \gamma \sum_{s^F\in S} P^a_{ss^F} v_*(s') \tag{2.17}$$

$$q_*(s,a) = R^a_{ss'} + \gamma \sum_{s^F\in S} P^a_{ss^F} \max_{a'} q_*(s',a') \tag{2.18}$$

2.2. A brief introduction to reinforcement learning

Before introducing reinforcement learning in more detail, it's essential to understand the general context of machine learning and its different categories.

Machine learning (ML) is a subfield of artificial intelligence that involves developing algorithms and models that can learn from data and make predictions on data. There are several machine learning types, including supervised learning, unsupervised learning, and reinforcement learning [12, 26].

Supervised learning involves training an agent using a labelled dataset provided by a knowledgeable external supervisor. The input data and the corresponding output are given to the agent as input during the training process. The goal of supervised learning is to learn a mapping between the input and the output so that the agent can generalize well to new, unseen input data. Although it is an important type of learning, it is not sufficient for solving interactive problems. Examples of supervised learning include image classification, speech recognition, and regression analysis [26].

Unsupervised learning involves training an agent using an unlabelled dataset to detect hidden patterns or structures in the data, such as clusters or anomalies. If the goal is to discover similarities or relationships between data points, such as clustering similar data points together, unsupervised learning is the appropriate choice. Examples of unsupervised learning include clustering, anomaly detection, and dimensionality reduction [26].

Reinforcement learning (RL) involves training an agent to make decisions based on rewards and penalties received from the environment. Similar to how humans and animals learn from experience, an agent learns to take actions that maximize the cumulative rewards or minimize the cumulative penalties through trial and error [12,27].

RL stands out among other machine learning algorithms as it allows agents to learn and adapt to new situations in real time without explicit programming. It is especially beneficial for applications where optimal action cannot be easily deduced from the given information but instead depends on the output of a series of actions taken over time, such as robotics, gaming, control, and finance. Besides, RL has indeed shown promising results in the field of congestion control. RL algorithms can dynamically adapt to changing network conditions and learn policies that optimize network performance. A comparison of major machine learning types is shown in Figure 2.1 [24].

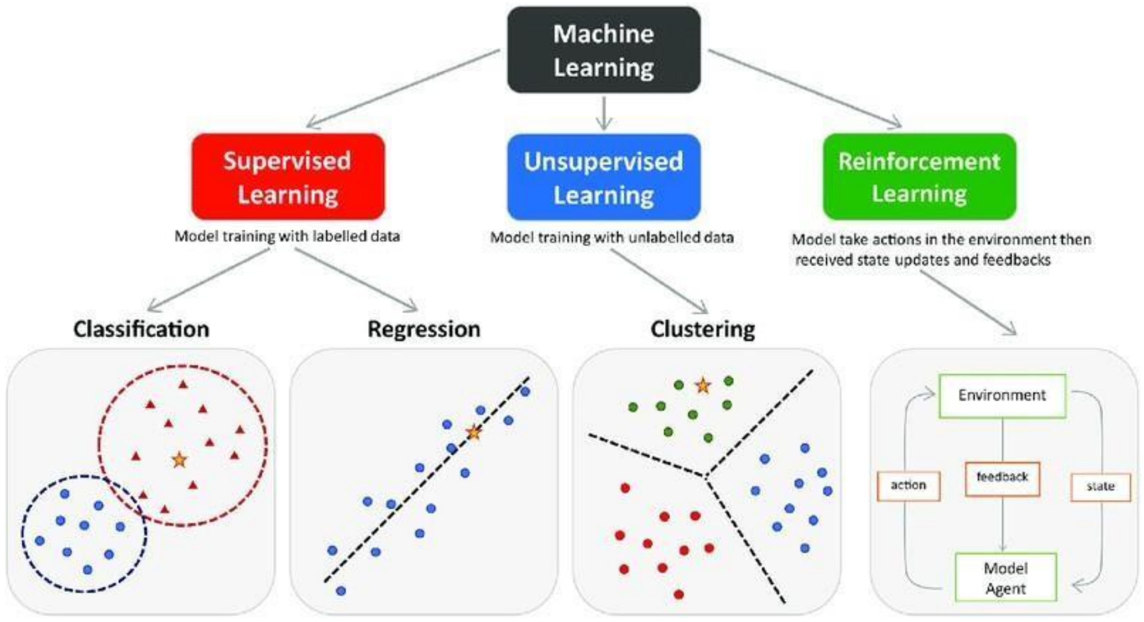


Figure 2.1: Categorization of machine learning as supervised learning, unsupervised learning and reinforcement learning.

2.2.1. The agent–environment interface

The reinforcement learning process involves the interaction between an agent and its environment, as shown in Figure 2.2. The agent, responsible for learning and making decisions, interacts with the environment which is outside of itself [12].

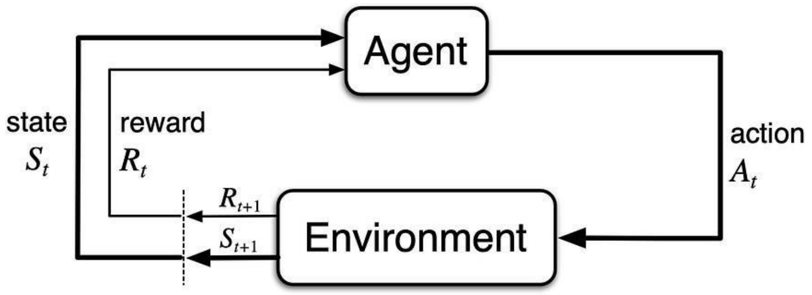


Figure 2.2: Reinforcement learning model.

At each time step t, the agent observes the current state of the environment S_t and chooses an action A_t to execute. Then, the environment transits to a new state S_{t+1} and generates a new reward R_{t+1}. This process is repeated until the agent approaches an optimal behaviour [12].

As described in the previous section, the Bellman optimality equation plays a crucial role in reinforcement learning, as it simplifies the calculation of the value function. There are several methods to solve the Bellman optimality equation in RL. Some of these methods are given below [12].

- Dynamic programming (DP) methods;
- Monte Carlo methods;
- Approximate solution methods;
- Policy gradient methods;
- Temporal difference (TD) methods.

2.2.2. Temporal-difference learning

Temporal-Difference (TD) learning stands out among other methods thanks to its ability to efficiently handle complex and large-scale problems, learn from partial feedback, adapt to changes in the environment in real-time, and handle delayed rewards. These properties make TD learning a powerful tool for congestion control policies in 5G networks. By using TD learning methods, agents can quickly adapt to changing network conditions and optimize their contention control policies for better network performance.

In this thesis, we will focus on Q-learning, Expected Sarsa, and Double Q-learning, which are TD learning methods.

2.2.2.1. Q-learning: Off-policy TD control

The development of an off-policy TD control algorithm known as Q-learning was one of the first breakthroughs in reinforcement learning [12]. As a model-free algorithm, Q-learning does not require prior knowledge of the MDP dynamics. It estimates the optimal action-value function, called the Q-function, representing the expected cumulative reward of taking a specific action in a particular state [28].

As mentioned, Q-learning is an off-policy reinforcement learning algorithm that uses a target policy to estimate the optimal Q-values while following a different behavioural policy to collect data. The target policy is typically greedy and chooses the action with the highest expected Q-value. However, the behavioural policy can be an ε-greedy and chooses the best action with probability (1-ε) and a random action with probability ε to encourage exploration. This approach allows the agent to learn the optimal policy while exploring the environment and collecting data using a different policy [24,28].

The Q-learning algorithm iteratively updates the Q-function using the Bellman equation:

$$Q(S, A) = R + \gamma \max_{a} Q(S', a) \tag{2.19}$$

where $Q(S, A)$ and $Q(S', a)$ are the return from the current state and the next state, respectively, R is the observed reward, and γ is the discount factor that determines the importance of future rewards. The max operator selects the action that maximizes the Q-function for the next state [28].

At each time step t, the agent observes the current state S_t, performs action A_t according to its exploration policy and observes the reward R_{t+1} and state S_{t+1}. Then it performs an action with the maximum possible reward for the next state and uses

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a} Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{2.20}$$

that for updating the current Q-value. The Q-function is updated using the following equation: where α is the learning rate that determines how much weight is given to new information compared to previous information [12,28]. The steps of the Q-learning algorithm are shown in the figure below [28].

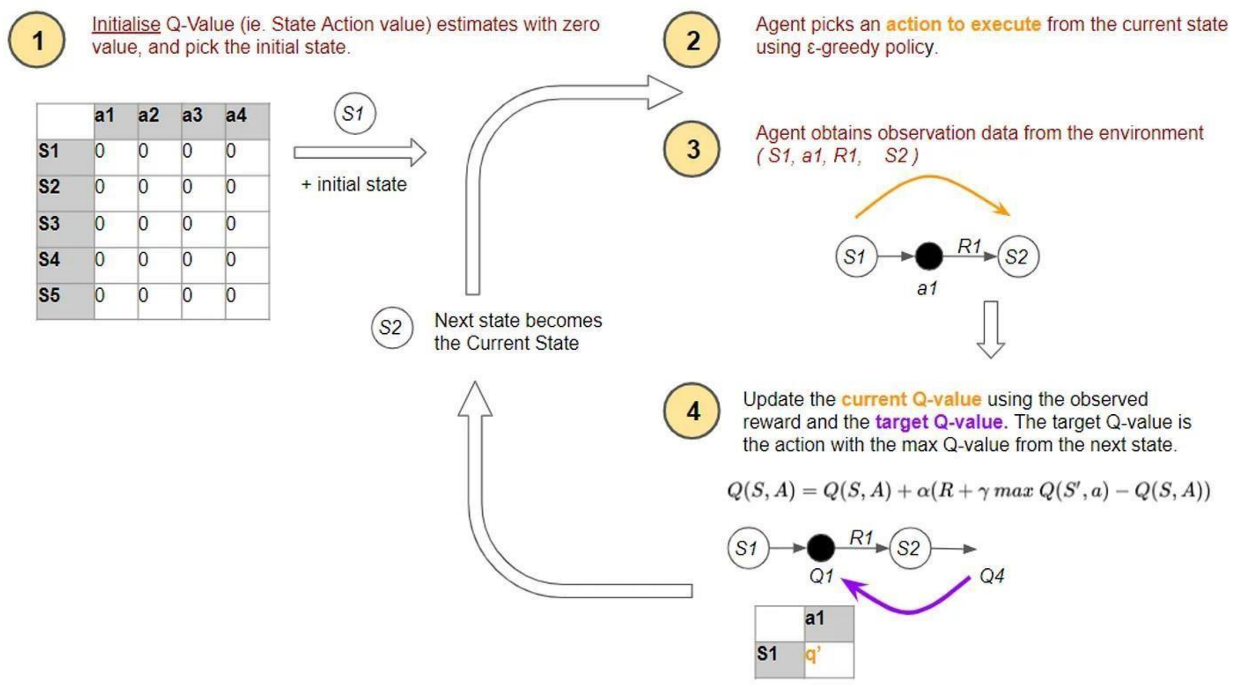

1
Initialise Q-Value (ie. State Action value) estimates with zero value, and pick the initial state.
a1 a2 a3 a4
S1 0 0 0 0
S2 0 0 0 0
S3 0 0 0 0
S4 0 0 0 0
S5 0 0 0 0
S1
+ initial state
2
Agent picks an action to execute from the current state using ε-greedy policy.
3
Agent obtains observation data from the environment (S1, a1, R1, S2)
S1 a1 R1 S2
S2 Next state becomes the Current State
4
Update the current Q-value using the observed reward and the target Q-value. The target Q-value is the action with the max Q-value from the next state.
Q(S, A) = Q(S, A) + α(R + γ max Q(S′, a) − Q(S, A))
S1 R1 S2 Q1 Q4
a1
S1 q'

Figure 2.3: Q-learning algorithm.

Although Q-learning has several advantages, including its simplicity, generality, and ability to handle large state spaces, it can suffer from slow convergence and instability due to the high variance of the Q-function estimates. Various extensions and improvements have been proposed in the literature, such as Double Q-learning and Deep Q-learning [24].

2.2.2.2. Expected Sarsa

Expected Sarsa is another model-free reinforcement learning algorithm that learns the optimal action-value function in a Markov decision process (MDP) through trial and error. It is a different version of the Sarsa algorithm that estimates the action-value function of a policy by performing a sample-based update at each time step.

It estimates the expected value of the following action-value function for a given state instead of updating it based on the following state and the following action. While Q-learning estimates the optimal policy directly, Expected Sarsa estimates the expected value of the policy being followed. With this approach, Expected Sarsa can achieve more stable learning and better convergence than Q-learning [29].

To implement Expected Sarsa, Q-values for all state-action pairs are initialized. Then, the following steps are repeated.

1. At each time step, the agent observes the current state S and selects an action A according to a policy, which could be an ϵ-greedy policy.

2. The agent then observes the reward R_{t+1} and the next state S_{t+1}, and uses the current Q-values and policy to compute the expected value of the next state-action pair, denoted by $Q(S_{t+1}, A_{t+1})$.

3. Finally, the agent updates the Q-value for the current state-action pair using the observed reward and the expected value of the next state-action pair:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma E_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)] \tag{2.21}$$

$$\leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1})\, Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{2.22}$$

where α is the learning rate, γ is the discount factor, $\pi(a \mid S_{t+1})$ is the probability of taking action a in the next state S_{t+1} under the current policy, and $\sum_a \pi(a \mid S_{t+1})\, Q(S_{t+1}, a)$ is the expected value of the next action-value function [1].

Expected Sarsa has similarities with on-policy Sarsa and off-policy Q-Learning algorithms, but it differs in the action value function it follows. Expected Sarsa can be used either as an on-policy or off-policy and this feature makes Expected Sarsa much more flexible than both algorithms [12,29].

Overall, Expected Sarsa is a beneficial algorithm for RL problems where the optimal policy may involve stochastic actions, rather than deterministic actions. It estimates the expected value of the next state-action pair instead of using the maximum Q-value, as in the Q-learning algorithm. This can make Expected Sarsa more stable and less prone to overestimation, especially in stochastic environments [29].

2.2.2.3. Double Q-Learning

Double Q-learning is an extension of Q-learning, which addresses the overestimation problem of Q-values that may occur when the same set of parameters is used to both choose and evaluate actions. By using two separate Q-functions, Q1 and Q2, Double Q-learning aims to reduce overestimation errors, leading to enhanced performance and stability. These Q-functions share the same policy for selecting actions, but they learn independently. At each time step, the agent selects one of the Q-functions randomly to update and uses the other Q-function to determine the action to take [12,30]. The update rule for Double Q-learning is as follows:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\,[R_{t+1} + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)] \tag{2.23}$$

$$Q_2(S_t, A_t) \leftarrow Q_2(S_t, A_t) + \alpha\,[R_{t+1} + \gamma Q_1(S_{t+1}, \arg\max_a Q_2(S_{t+1}, a)) - Q_2(S_t, A_t)] \tag{2.24}$$

where S_t is the current state, a is the action, R_{t+1} is the reward, S_{t+1} is the next state, α is the learning rate, and γ is the discount factor.

Overall, these three RL algorithms (Q-learning, Expected Sarsa, and Double Q-learning) are efficient approaches in training agents to make decisions in complex environments. By estimating the value of each action in each state, these algorithms can learn to choose the best possible action at each step of the learning process, maximizing the cumulative reward received over time. It makes them powerful methods in challenging environments.

3 Preliminaries, Scenario and Assumptions

This section presents the details of the scenario designed to evaluate the performance of the proposed grant-free access protocol, assumptions, and performance parameters. The evaluation scenario under study is taken from [6]. We only evaluate what in [6] is referred to as low priority traffic, i.e., mMTC traffic. However, instead of using a stationary arrival regime (binomial packet arrivals to nodes), we use the arrival regime proposed by the 3GPP, where devices become active (they start contending for access) according to a Beta (3,4) distribution over 10 seconds (2000 subframes) as described in [11]. Then, the evaluation scenario studied is more stringent and realistic than the one used in [6].

3.1. Network and traffic models

The system model for this study focuses on a wireless network consisting of a single cell, with subframes structured in V = 10 timeslots for the uplink. This would be equivalent to deploying one RAO every subframe with a total of 10 preambles in the hand-shaking scenario. However, in a grant-free scenario, sensors transmit their packet without previous signalling with the base station. The network comprises 20,000 user equipments (UEs) that contend for access to an eNB (base station, BS) using a random access protocol.

We assume that all sensors share the same radio resources and transmit their packets that fit in a timeslot. A collision occurs if multiple packets are transmitted during the same timeslot, resulting in the loss of all packets. In such cases, the collided packets are retransmitted up to a maximum of 10 times before being considered lost. As in [6], we assume that for each subframe, the BS can detect the number of holes (timeslots that where not occupied by any transmission, h), successes (timeslots with a single packet transmission, s) and collisions (timeslots with more than one packet transmissions, c). An example of operation of the system under study is given in Figure 3.1 below.

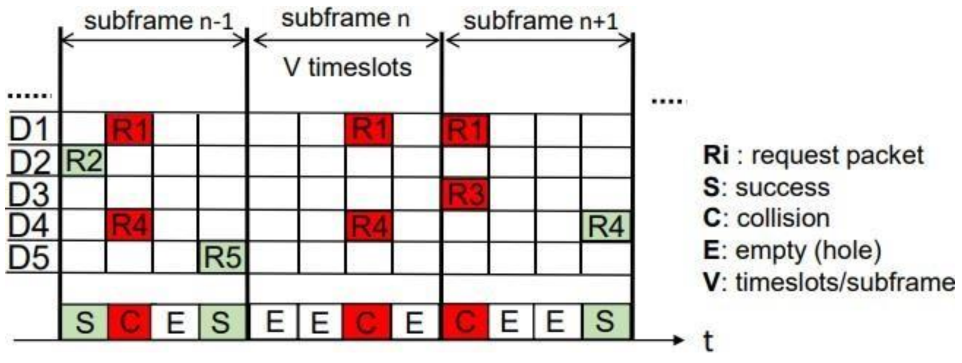


Figure 3.1: System model.

In this study, we implemented the network configuration and the Traffic Model 2 suggested in the 3GPP specification [11]. Suggested traffic models for Machine Type Communication (MTC) are presented in Table 3.1.

Characteristics	**Traffic Model 1**	**Traffic Model 2**
Number of M2M UEs	1000, 3000, 5000, 10000, 30000	1000, 3000, 5000, 10000, 20000
Arrival distribution	Uniform	Beta (3,4)
Distribution period (T)	60 seconds	10 seconds

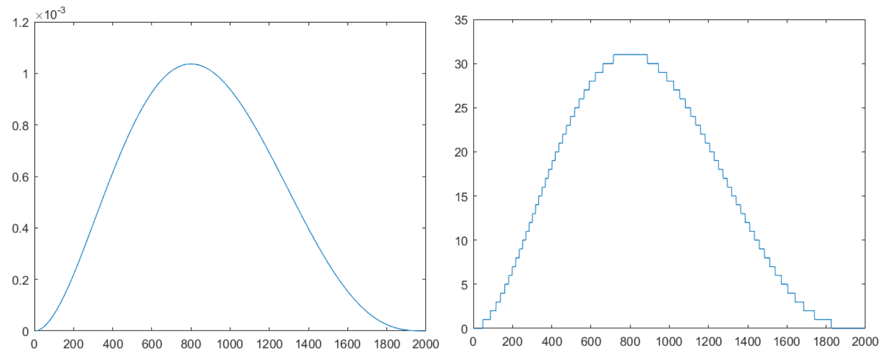
Table 3.1: Suggested M2M traffic models for RACH evaluation.

Traffic model 1 represents a uniform distribution of UEs over a period for simulating a non-synchronized access behaviour of UEs in the network. On the other hand, traffic model 2 represents an extreme scenario where a large number of UEs attempt to access the network simultaneously in a highly synchronized manner.

In the proposed model, as described in traffic model 2, the number of UEs is selected as 20,000 and activations for packet transmission follow a Beta (3,4) distribution over $T_D = 2000$ subframes. The pdf at subframe n, where $n \leq T_D$, is expressed as

$$p_A = 60n^2(T_D - n)^3/T_D{}^6 \tag{3.1}$$

where pA is the probability that arrivals occur at subframe n, where $n \leq T_D$. For 20,000 UEs, the number of sensor activations per subframe is shown in Figure 3.2.



(a) (b)

Figure 3.2: (a) Probability density function (pA) and (b) number of sensor activations per subframe (20,000 UEs).

3.2. Congestion control scheme

In the proposed scheme, the base station broadcasts an access probability ϱ once every 10 subframes, referred to as a superframe. This approach represents a simplified version of the Access Class Barring scheme proposed by the 3GPP.

In our scheme, with probability ρ, each active UE randomly chooses one of the slots within the current subframe to transmit a packet to the BS. At the same time, other UEs postpone their transmission to the next subframe with probability 1- ρ. Then, the BS observes each slot of the current subframe and counts the number of holes, successes, and collisions. It continues to observe these values for ten subframes and at the end of the 10th subframe, it determines a new access probability. This type of delayed feedback defined is in accordance with the definition of the eNode radio interface by the 3GPP. It should be noted that it differs from the approach in [6], where the feedback is performed every subframe and, therefore, is not implementable with the current definition of the eNode radio interface.

To minimize the number of collisions per subframe, the number of contending UEs per subframe must be equal to the number of available timeslots in the subframe [30].

To determine a new ρ, the BS estimates the average number of UEs that transmitted per subframe. For that, the BS must consider the total number of active UEs, i.e., both new arrivals during the superframe and backlogged devices that will attempt to access the BS during the next superframe. Backlogged UEs refer to those that delayed their transmission in the current superframe due to the access probability, and those that suffered collisions and will retry. In the next superframe, all active UEs will attempt to transmit their packets following the access probability broadcasted by the BS, computed as,

$$\rho = V/MAS\,, \tag{3.2}$$

where MAS is the number of active UEs per subframe estimated by the BS for the next subframes. In our case, a reinforcement learning algorithm is used to determine the access probability ρ. The details of the algorithm will be discussed in a Section below.

3.3. Performance parameters proposed by 3GPP

The five key performance indicators (KPIs) proposed by 3GPP are now defined [11]. We refer to the access period (AP) as the number of subframes required to empty the queue of contending sensors that access the system, i.e., the number of subframes required for the last sensor to abandon the system.

1. Collision probability, defined as

$$P_C = \frac{Number\ of\ collided\ UE\ transmisson}{Total\ number\ of\ succesful\ transmissons\ during\ AP} \tag{3.3}$$

2. Access success probability, denoted as P_S, defined as the fraction of UEs that successfully transmit its packet.
3. Statistics of the number of times UEs transmit a packet before successfully completing the random access procedure. This KPI is measured by its average value, represented as E[k].
4. The access delay statistics represent the time interval between the instant in which the sensors become active and the successful completion of the random access procedure. To evaluate this KPI, we generate its cumulative distribution function (CDF) and calculate its average and the 95th percentile, referred to as D_{95}.
5. Statistics of the simultaneous packet transmissions per timeslot. This parameter is not evaluated in this study.

3.4. Implementation of reinforcement learning algorithms

Many centralized access control problems have a key characteristic where the access controller is unaware of how many users are competing for resources and their access times. In the LTE-A system, the base station has the ability to determine the number of successfully received packets per subframe (N_{su}). But, due to various factors such as collisions, interference, or decoding issues, N_{su} often have a different value than the total number of transmitted packets, particularly in heavily congested scenarios [5]. In this study, we propose a control scheme based on the BS calculating the access probability by estimating the number of active UEs in the subframe using reinforcement learning techniques. The access probability is periodically broadcast to the UEs, achieving in this way an efficient congestion control mechanism that enhances network performance.

In our system, an agent is located at the BS and observes UE's access outcomes, represented by the number of (S, C, H) per subframe. The environment is made of UEs accessing the upstream subframe and the interaction between the agent and the environment by the agent broadcasting the access probability ϱ at each superframe. Although the BS has complete knowledge of the number of active UEs contending for resources at each subframe in the simulation model, this information is not available to the RL mechanism, as in a real scenario.

The previous chapter provided an overview of Q-learning, Double Q-learning and Expected Sarsa, which are RL algorithms. In this section we will focus on the implementation of these RL algorithms to the system.

Before starting to implement the RL method, some parameters need to be designed:

- The environment state;
- The set of actions (access probabilities);
- The reward functions;
- The RL scheme/algorithm;
- The exploration distribution (ε), discount factor (γ), and learning rate (α);

3.4.1. The environment state

We defined different agent states to observe their effects on performance. The first state is based on averaging and rounding the number of successes and collisions over the last superframe since the access probability was broadcasted.

$$S_{t1} = (S, C) \qquad S, C \in \mathrm{N} \qquad S, C \in \{0, V\} \tag{3.4}$$

where S/C is the rounded average success/collisions per subframe, and V is the number of available timeslots. To provide more information, a summary of the preceding states is added. S_{t-1} denoted by θ_{t-1} and S_{t-2} denoted by θ_{t-2}.

$$S_{t2} = (\theta_{t-1}, S, C), \quad S, C \in \{0, V\}, \quad \theta_{t-1} \in \{0, 8\} \tag{3.5}$$

$$S_{t3} = (\theta_{t-1}, \theta_{t-2}, S, C), \quad S, C \in \{0, V\}, \quad \theta_{t-2}, \theta_{t-1} \in \{0, 8\} \tag{3.6}$$

θ_{t-2}, θ_{t-1} are designed to keep manageable the size of Q (s, a) table. The summary of the preceding states is included in the state definition, as it might help to improve the estimation performed by the R algorithm (MAS).

3.4.2. The set of actions

In our model, two action sets with 28 and 64 elements are deployed as shown below.

A_{28}= {0.0, 0.01, 0.05, 0.08, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0}

A_{64}= {0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0, 4.2, 4.4, 4.6, 4.8, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0}

The action set A is composed of the actions that change access probability ϱ to one of its possible values. Note that $a \times 10^3$, a ∈ A, is an estimate of the number of active UEs. When action a is taken, the agent broadcasts the access probability shown below.

$$\rho = V/\,(a \times 10^3) \tag{3.7}$$

3.4.3. The reward functions

The reward function is one of the most important parameters of reinforcement learning. We defined different reward functions to explore their impact on performance.
The first reward function is,

$$R_{t1} = S, \quad S \in N, \quad S \in \{0, V\} \tag{3.8}$$

where S is the rounded value of the average number of successes per subframe over the last superframe. To provide more information, we also explored with,

```
if $C_{tot}$ < RWCOMXv then
  err ← RWPv (positive reward)
else
  err ← − RWNv (negative reward)
end if
$R_{t2}$ ← S + err
```

where C_{tot} is the total number of collisions over the last superframe, and RWCOMXv, RWNv, and RWPv are conguration parameters. For example, parameters can be selected as [RWCOMXv, RWPv, -RWNv] = [20, 0.5, 0.1].

3.4.4. Q-learning based congestion control policy

Q-learning is a model free RL algorithm that is used to estimate the optimal action-value function (Q-function) for a given environment. In this study it was used to determine the access probability ϱ that maximized the performance of the proposed congestion control scheme. The main advantage of Q-learning is that it does not require prior knowledge of the environment or a model of the system dynamics, making it well-suited for real-time, online decision making in dynamic environments.

According to the proposed scenario of study, the RL algorithm evolves at discrete time instants (end of superframes), the current state S_t and action A_t are not available until the end of the current superframe, and the next reward R_{t+1} and the next state S_{t+1} are not available until the end of the next superframe. The RL evaluation instants are shown in Figure 3.3.

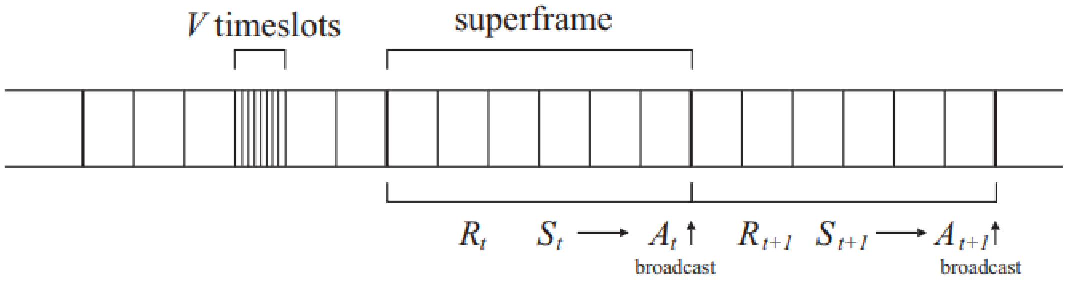

V timeslots
superframe
R_t
$S_t \longrightarrow A_t \uparrow$
broadcast
R_{t+1}
$S_{t+1} \longrightarrow A_{t+1} \uparrow$
broadcast

Figure 3.3: RL evaluation time instants.

Access probability is calculated at the end of a superframe and UEs send their packets according to this probability during each subframe of the next superframe. The implementation steps of Q-learning can be described as follows:

1. At each time step, the agent observes the current state S_t and selects an action A_t according to a ϵ-greedy policy. With probability ε, the agent selects a random action, and with probability 1-ε, it selects the action with the highest Q-value for the current state.

2. According to the selected action, the agent updates the access probability ϱ using the formula $\rho = V/(a \times 10^3)$, where $MAS = a \times 10^3$, is the estimated number of active UEs in the next superframe. Then, ρ is broadcasts to the UEs.

3. After 10 subframes, the agent observes the next state S_{t+1} and calculates the reward R_{t+1}.

4. Finally, the agent updates the Q-value for the current state-action pair is updated Q-learning update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{3.9}$$

5. The agent sets the current state to the next state and repeats the above steps are for each superframe.

3.4.5. Expected SARSA based congestion control policy

Expected Sarsa is another RL algorithm similar to Q-learning. But, instead of selecting the maximum Q-value for the next state-action pair, it uses the expected value of all possible actions. This approach can lead to more stable learning, especially in environments with stochastic rewards or actions. In Expected Sarsa, the agent updates its Q-function estimate using the expected value of the Q-function for the next state-action pair. This algorithm ensures that the agent is not too optimistic or too pessimistic about the value of the next action and can lead to more efficient learning.

Like Q-learning, access probability is calculated at the end of a superframe and UEs send their packets according to this probability during each subframe of the next superframe. The implementation steps of Expected Sarsa can be described as follows:

1. At each time step, the agent observes the current state S_t and selects an action A_t according to a policy, which could be an ϵ-greedy policy.

2. According to the selected action, the agent updates the access probability ϱ using the formula $\rho = V/(a \times 10^3)$, where $MAS = a \times 10^3$, is the estimated number of active UEs in the next superframe. Then, ρ is broadcasts to the UEs.

3. After 10 subframes, the agent observes the next state S_{t+1} and calculates the reward R_{t+1} and uses the current Q-values and policy to compute the expected value of the next state-action pair, denoted by $Q(S_{t+1}, A_{t+1})$.

4. Finally, the agent updates the Q-value for the current state-action pair using the observed reward and the expected value of the next state-action pair with update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma E_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)] \tag{3.10}$$

$$\leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1})\, Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{3.11}$$

5. The agent sets the current state to the next state and repeats the above steps are for each superframe.

3.4.6. Double Q-learning based congestion control policy

Double Q-learning is an extension of Q-learning. It addresses the overestimation problem of Q-values that may occur when the same set of parameters is used to both choose and evaluate actions. The implementation steps of Double Q-learning are very similar to Q-learning. The key difference between them is that Double Q-learning uses two Q-functions, Q1 and Q2, to estimate the state-action values and alternates between them during the learning process. This helps to prevent overestimation of the Q-values and can lead to more accurate estimates of the optimal policy. These Q-functions share the same policy for selecting actions, but they learn independently.

Similar to other RL algorithms mentioned in this study, access probability is calculated at the end of a superframe.

The implementation steps of Double Q-Learning can be described as follows:

1. At each time step, the agent observes the current state S_t and selects an action A_t according to a ϵ-greedy policy. With probability ε, the agent selects a random action, and with probability 1-ε, it selects the action that maximizes the sum of Q1(s,a) and Q2(s,a).

2. According to the selected action, the agent updates the access probability ϱ using the formula $\rho = V/(a \times 10^3)$, where $MAS = a \times 10^3$, is the estimated number of active UEs in the next superframe. Then, ρ is broadcasts to the UEs.

3. After 10 subframes, the agent observes the next state S_{t+1} and calculates the reward R_{t+1}.

4. Finally, the agent selects one of the Q-functions randomly to update and uses the other Q-function to determine the action to take. The update rule for Double Q-learning is as follows:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\,[R_{t+1} + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)] \tag{3.12}$$

$$Q_2(S_t, A_t) \leftarrow Q_2(S_t, A_t) + \alpha\,[R_{t+1} + \gamma Q_1(S_{t+1}, \arg\max_a Q_2(S_{t+1}, a)) - Q_2(S_t, A_t)] \tag{3.13}$$

5. The agent sets the current state to the next state and repeats the above steps are for each superframe.

3.4.7. The exploration, discount factor, and learning rate

The exploration strategy is a crucial element in RL that determines how the agent explores the environment to learn more about it. As discussed in Chapter 2, exploration denotes the process of actively seeking out new and unexplored states and actions in the environment. The agent needs to balance exploration with exploitation, which refers to selecting actions that are known to yield high rewards. Too much exploitation can lead to the agent getting stuck in a suboptimal solution, while too much exploration can lead to inefficient learning. [12].

To achieve this balance, we used an epsilon-greedy approach in our analysis scenario. Epsilon-greedy selects the action with the highest estimated value with probability 1-epsilon and selects a random action with probability epsilon.

Apart from exploration strategies, we also investigate the impact of discount factor and learning rate on the performance of RL-based congestion control policies. The discount factor ranges between 0 and 1 that is used to weigh future rewards in the agent's decision-making process. A high discount factor means that the agent values long-term rewards more, while a low discount factor means that the agent values immediate rewards more. The discount factor is used to calculate the expected return, which is the sum of the discounted future rewards that the agent expects to receive [12].

The learning rate determines how much weight is given to new information compared to old information in updating state-action values. A higher learning rate allows for faster adaptation but may also lead to instability or oscillations in some cases. To analyze the impact of these parameters on congestion control performance, we conducted a series of experiments using different combinations of parameter settings and evaluated their performance based on performance metrics.

4 Experimental Results

In this chapter, the performances of reinforcement learning based congestion control policies for mMTC in 5G networks were evaluated through simulation. Policies obtained by RL algorithms were compared with each other, heuristic and ideal approaches based on performance parameters. Then, the impact of RL parameters and hyperparameters on performance were examined.

In the previous chapter, we presented the details of the scenario designed to evaluate the performance of the proposed congestion control policy, assumptions, and reinforcement learning parameters. To summarize, we implemented a network configuration and the Traffic Model 2 recommending in the 3GPP specification [11]. The number of user equipment (UE) was set at 20,000, and packet transmission activations followed a Beta (3,4) distribution, where the activation of UEs occur along 2000 subframes of 5 ms long, lasting 10 seconds. The access probability ρ is broadcasted by the base station (BS) every 10 subframes, referred to as a superframe, which is in line with 3GPP's eNode radio interface definition. In case of collision, the collided packets are retransmitted up to a maximum of 10 times before being considered lost. Please refer to Chapter 3 for more detailed information. We refer to this type of feedback from the BS to the UEs as delayed feedback. It is different to the one used by the ideal and the studied heuristic approach, which provide feedback every subframe, and we refer to it as immediate feedback.

Note that the operation of the system is episodic. Activation of UEs occur along 2000 subframes. However, due to the operation of the congestion control scheme and sometimes due to collisions, the UEs access to the upstream subframe is spread along time. We refer to an access period as the number of consecutive subframes during which UEs access the upstream subframe. We also refer to the last subframe as the subframe at which the last UE completes its access, i.e., the last subframe of the access period.

4.1. Performance metrics

The following metrics were computed for the purpose of performance analysis of the proposed algorithm:

- Thsf: Subframe throughput. Average number of successful transmissions per subframe.
- E[D]: Average delay. Average number of subframes elapsed from the instant a sensor becomes active, until it successfully completes its packet transmission.
- D_{95}: 95 percentiles of the delay. It is an access delay D_{95} such that 95% of the sensors complete its packet transmission successfully with a delay lower than D_{95}.
- E[Co]: Average number of collisions per successfully transmitted packet.
- AcSP: Successful access probability. Fraction of sensors that complete its packet transmission successfully. AcSP = 1 – LP.
- Last_Sub: Last Subframe. The subframe at which the queue of sensors that access the system becomes empty.

4.2. Ideal policy

In the ideal policy, the BS has complete knowledge of the number of active UEs contending for resources at each subframe. This information is available in the simulation model. Let this number be MAS. Then, the optimal access probability at each subframe is,

$$\rho = V/MAS \tag{4.1}$$

The 3GPP standard proposed that the optimum access probability ρ be broadcast only every superframe. However, the ideal policy follows the immediate feedback approach where access probability is broadcasted in each subframe. The ideal policy results obtained from the simulation are given Table 4.1.

Thsf	**E[D]**	**D_{95}**	**E[Co]**	**AcSP**	**Last_Sub**
3.78E+00	1.83E+03	4110	1.56E+00	9.93E-01	5257

Table 4.1: The results of ideal policy.

4.3. Heuristic policy

In addition to the ideal policy, a heuristic estimation proposed by Rives [32], and deployed in [6], was used to compare our results. This policy is based on a pseudo-

Bayesian estimation algorithm, and it estimates the number of active UEs per subframe. Like the ideal policy, also this policy follows an immediate feedback approach. Although it might not be applicable in the studied scenario, where the arrival process is not stationary, we deployed it as a reference. Note that the traffic process deployed in [6] was stationary and, therefore, different from the one recommended by the 3GPP for evaluating mMTC access schemes.

Error! Bookmark not defined. The results obtained when deploying the Rives estimation algorithm and their relative error when compared to the ideal algorithm are given in Table 4.2.

Thsf	E[D]	D_{95}	E[Co]	AcSP	Last_Sub
3.28E+00	2.21E+03	4751	1.97E+00	9.97E-01	6075
-13.06%	20.60%	15.60%	26.04%	0.47%	15.56%

Table 4.2: The results of heuristic policy.

Figure 4.1 shows the number of active sensors in the queue obtained when deploying the ideal policy and Rives algorithm and the number of active sensors estimated by the Rives algorithm per subframe along an access period.

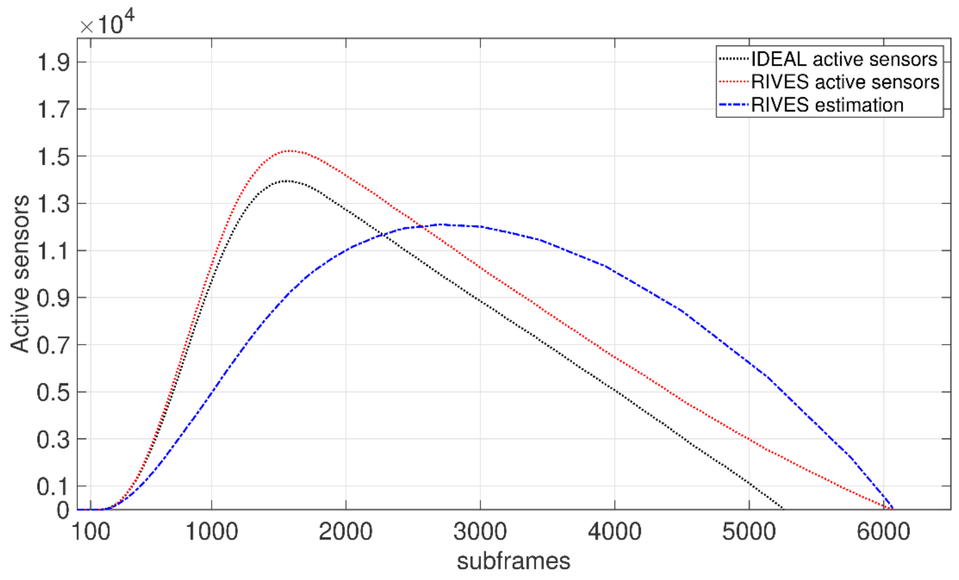
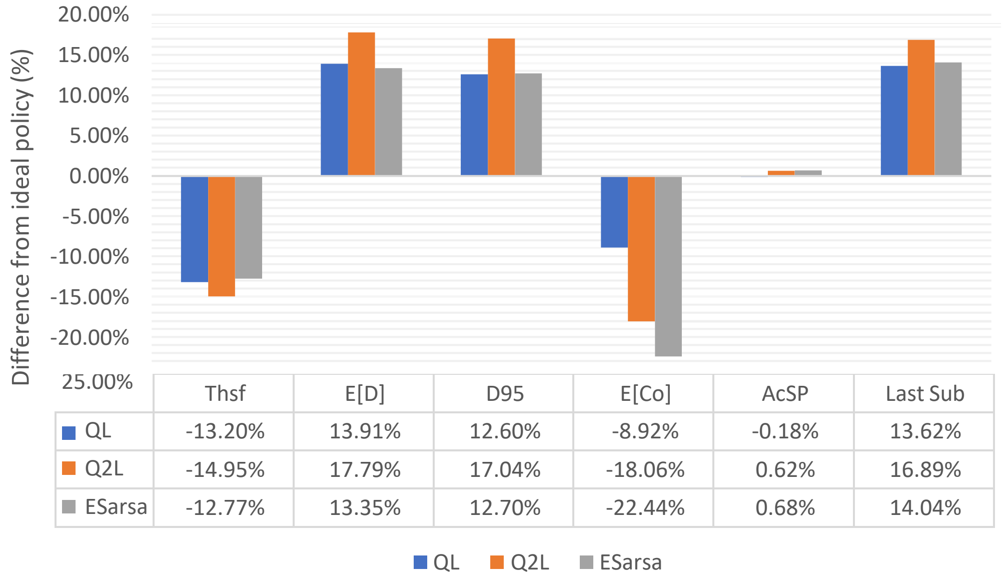


Figure 4.1: Rives estimation with respect to ideal policy.

As can be seen from the Figure 4.1, the Rives algorithm underestimated the number of active sensors until the first half of the access period and overestimated the number of active sensors in the second half. Although the Rives algorithm has an important position in the literature in the field of congestion control, its performance in the studied scenario is not adequate. This might be due to the fact that it was designed for a different traffic arrival process and for stationary traffic scenarios.

4.4. Performance comparison of RL algorithms

One of the main objectives of this thesis is to investigate and evaluate the congestion control performances of Reinforcement Learning (RL) algorithms that have been proposed in the literature. To achieve this goal, the performances of the 3 RL algorithms (Q-Learning, Double Q-Learning, and Expected Sarsa) were compared based on the $S_{t2} = (\theta_{t-1}, S, C)$. Our approach involved quantifying the relative error of the results obtained by the RL algorithms with respect to the ideal policy, thus enabling a clear comparison of the outcomes. The simulation parameters were determined as (RWPv, RWNv, α, γ) = (0.5, 1, 0.3, 0.5), where RWPv is positive reward, RWNv is negative reward, α is learning rate and is discount factor γ. The outcomes of the simulation are shown in Figure 4.2.



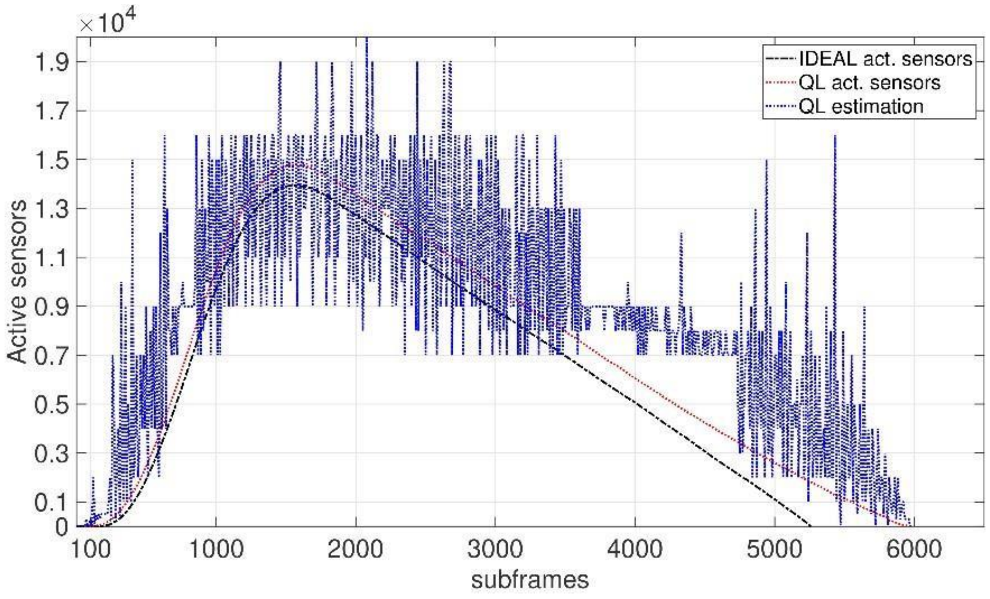
	Thsf	E[D]	D95	E[Co]	AcSP	Last Sub
■ QL	-13.20%	13.91%	12.60%	-8.92%	-0.18%	13.62%
■ Q2L	-14.95%	17.79%	17.04%	-18.06%	0.62%	16.89%
■ ESarsa	-12.77%	13.35%	12.70%	-22.44%	0.68%	14.04%

Figure 4.2: Performance comparison of RL algorithms.

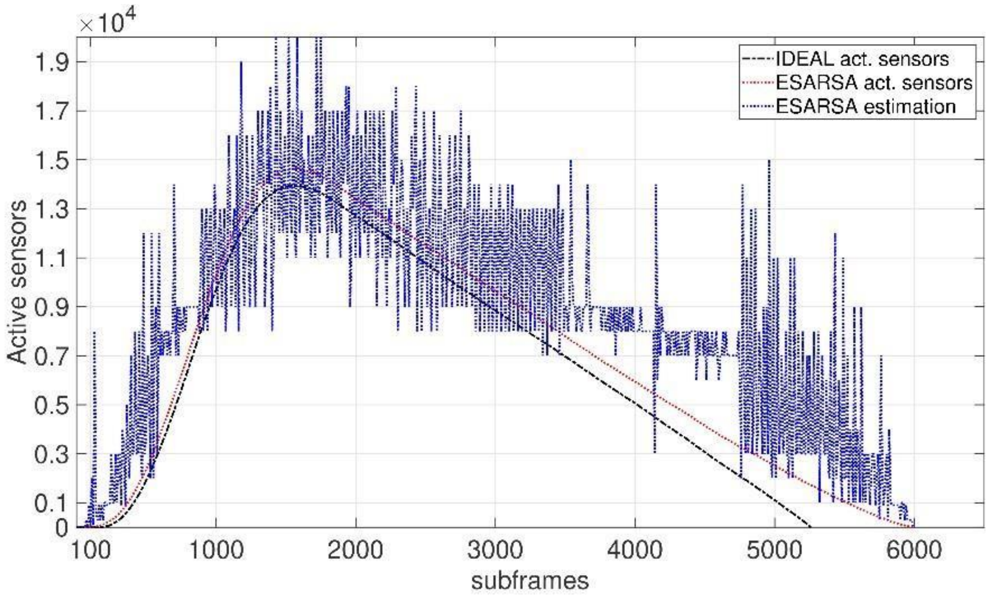
Based on the simulation results, it can be observed that all RL algorithms outperform the ideal scheme in terms of the E[Co] parameter. However, it is important to note that this improvement comes at the expense of enlarging the access period, which may lead to a slight increase in access delay. This trade-off between access delay and energy consumption can be beneficial in scenarios where reducing E[Co] is more critical than minimizing access delay.

On the other hand, all RL-based policies perform worse than ideal policy for other performance parameters such as E[D] and Last Sub. This is because the ideal policy aims only at minimizing the last subframe performance parameter. However, RL-based policies are more flexible as they can be designed to optimize other performance parameters by adequately defining the reward function.

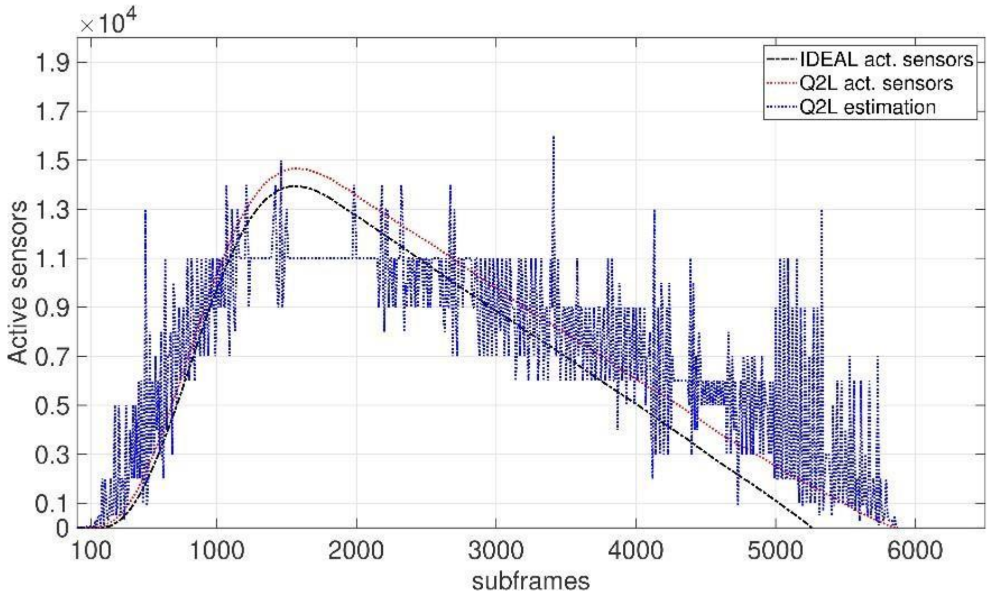
When the proposed policies are compared within themselves, it can be seen that the policy based on Double Q-Learning shows the lowest performance, except for the parameters average number of collisions and access success probability, while the policy based on Expected Sarsa stands out in almost all the parameters.



(a)



(b)



(c)

Figure 4.3: Comparison of the IDEAL approach with (a) Q-Learning, (b) Expected Sarsa, and (c) Double Q-Learning over the number of active sensors

The graphs show that Double Q-Learning has performed well in estimating the number of active sensors, except where the number of active sensors peaks. However, it started to underestimate the number of sensors, especially after the number of active sensors rose above 11,000.

On the other hand, Q-Learning and Expected Sarsa showed much better results than Double Q-Learning at points where the number of active sensors is maximum. Besides, it can be observed that the predictions made by Expected Sarsa and Q-Learning are similar in terms of estimating the number of active sensors. However, it is worth noting that Expected Sarsa tends to make predictions that follow the shape of the actual active sensor distribution more closely than Q-Learning.

Finally, when the results obtained from Q-Learning and Expected Sarsa are examined, it is observed that both algorithms make estimates very close to the actual number of sensors while the number of active sensors increases, and they begin to overestimate the values while the number of active sensors decreases. The solution of this problem is one of the future research topics. Also, the reduction of the estimation oscillations, both in magnitude and frequency, is a topic for future research. As observed, the tracking of the actual number of active sensors worsens once the slope of the number of active sensors changes its sign. This might suggest that the state definition might require to be supplemented with a filtered value of the queue length estimation. That is, the current state definition based only of success and collisions might be insufficient to detect a slope sign change.

Lastly, when we collect all the algorithms in a single graph, Figure 4.4, we can observe that they all have a longer access period compared to the ideal policy. Among all the algorithms, Double Q-Learning exhibits the shortest access period, while Expected Sarsa has the slightly largest access period and produces results that are closest to the IDEAL.

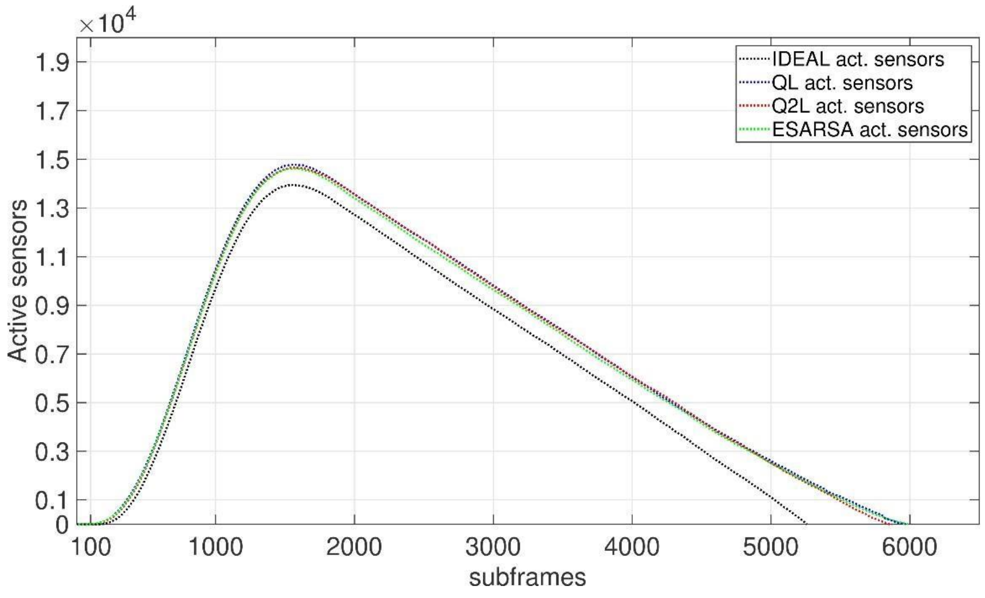
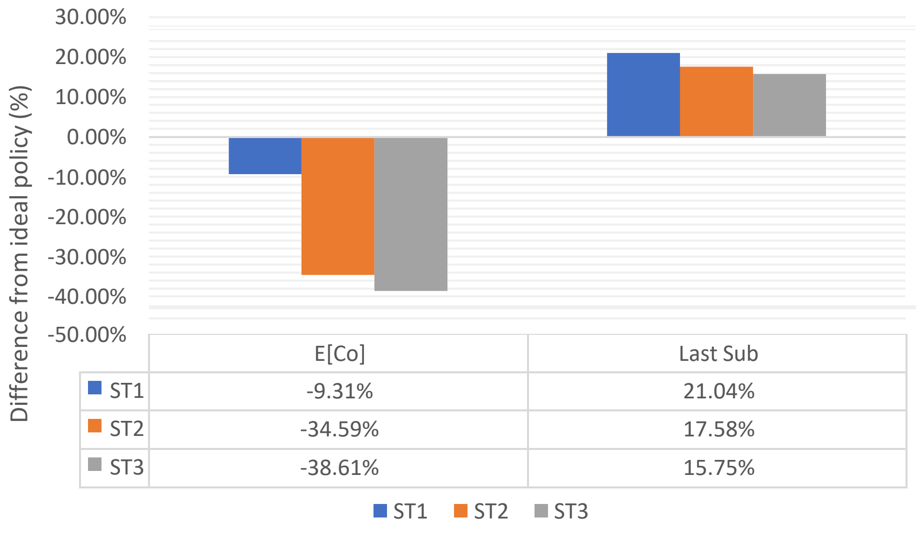


Figure 4.4: Comparison of RL algorithms in terms of active sensors.

4.5. Impact of state representation selection

In order to analyze how state representation impacts performance, we conducted a comparison of three state definitions $(S_{t1} = (S, C), S_{t2} = (\theta_{t-1}, S, C), and\ S_{t3} = (\theta_{t-1}, \theta_{t-2}, S, C))$ using the E[Co] and Last Subframe parameters. The simulation parameters were determined as (RWPv, RWNv, α, γ) = (0.5, 1, 0.3, 0.5). The comparison of state representations using Q-Learning is shown in Figure 4.5.



	E[Co]	Last Sub
🟦 ST1	-9.31%	21.04%
🟧 ST2	-34.59%	17.58%
⬜ ST3	-38.61%	15.75%

Figure 4.5: A performance comparison of state representations using Q-learning.

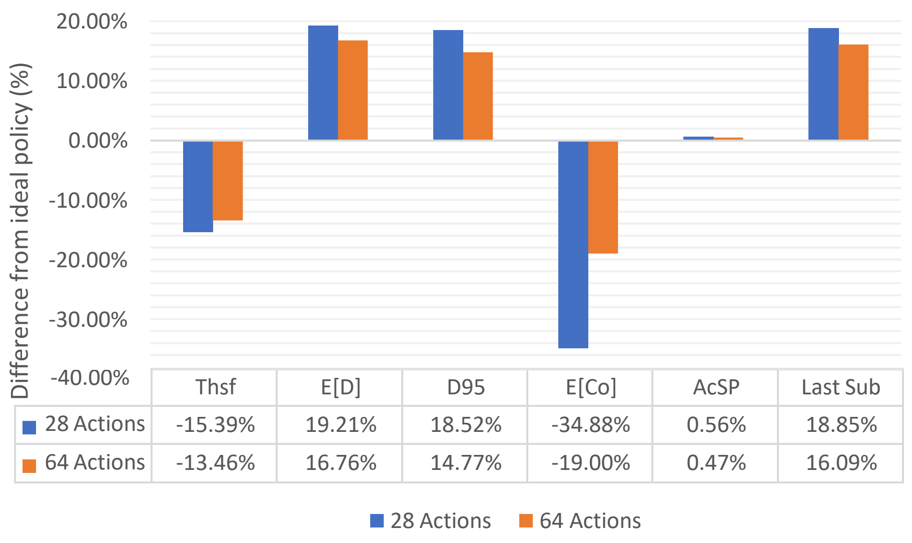
Based on the results obtained, it is evident that ST3 state definition outperforms other state definitions, including the ideal approach, in terms of E[Co] and the Last Sub parameter. On the other hand, ST1 state definition exhibited the poorest results for both parameters.

This means that increasing the contribution of previous states to the current state definition results in a reduction of the average number of collisions, shorter access period, and consequently, a decrease in energy consumption.

4.6. Impact of action set selection

To evaluate the impact of the action set selection on the performance, we conducted a comparison of two sets of actions comprising 28 and 64 elements, taking into account all relevant performance parameters. The simulation parameters were determined as (RWPv, RWNv, α, γ) = (0.5, 1, 0.3, 0.5), and state definition was selected as $S_{t3} = (\theta_{t-1}, \theta_{t-2}, S, C)$.

The comparison of two sets of actions with 28 and 64 elements using Expected Sarsa is shown in Figure 4.6.

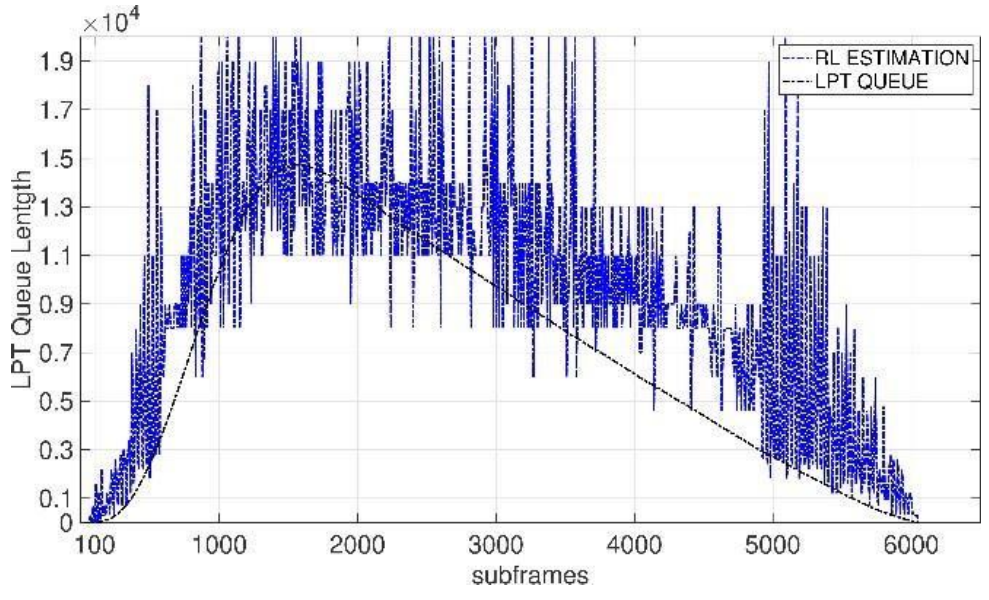


	Thsf	E[D]	D95	E[Co]	AcSP	Last Sub
■ 28 Actions	-15.39%	19.21%	18.52%	-34.88%	0.56%	18.85%
■ 64 Actions	-13.46%	16.76%	14.77%	-19.00%	0.47%	16.09%

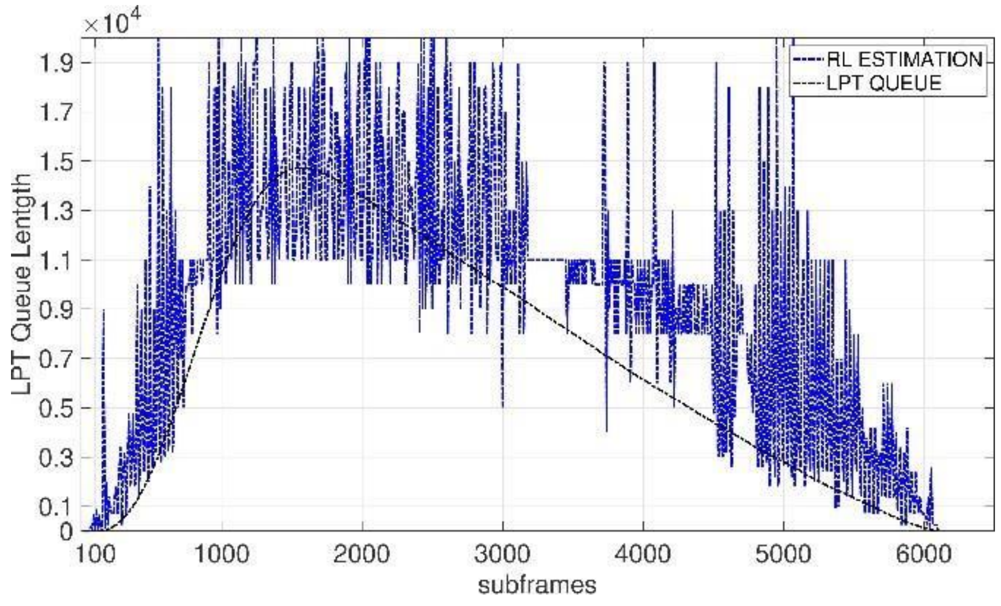
Figure 4.6: A performance comparison of two sets of actions with 28 and 64 elements using Expected Sarsa.

It is observed that the action set comprising 64 elements outperformed the action set comprising 28 elements except AcSP parameter. This result can be explained by the fact that the action set comprising 64 elements offers a wider range of feasible values for modifying the access probability ρ. This results in the system being able to optimize its performance better. In contrast, the action set comprising 28 elements has a more limited range of feasible values, which limits the system's capability to optimize its performance.

Furthermore, an evaluation of the performance of the two action sets was conducted based on the queue length, defined as the number of active sensors in the queue. The results of this analysis are illustrated in Figure 4.7.



(a)



(b)

Figure 4.7: Queue length comparison of two sets of actions comprising (a) 28 and (b) 64 elements using Expected Sarsa.

Even though the queue length is mostly overestimated in both scenarios, it is estimated more accurate when the action set consists of 64 elements.

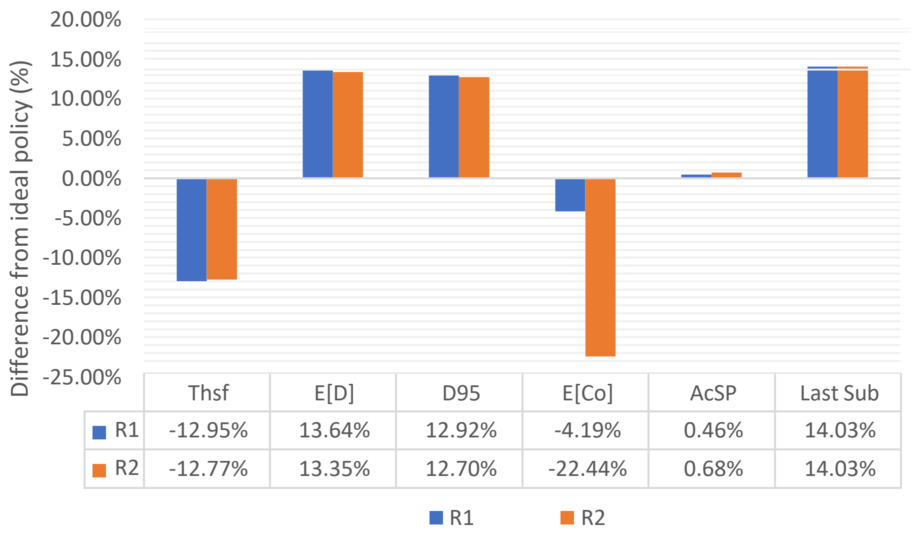
4.7. Impact of reward function selection

The effects of two reward functions, R1 and R2, on performance were investigated by considering all performance parameters. The simulation parameters were determined as (α, γ) = (0.3, 0.5), and state definition was selected as $S_{t3} = (\theta_{t-1}, \theta_{t-2}, S, C)$. The selected reward functions are given below.

$$\text{R1} = [\text{RWCOMXv, RWPv, RWNv}] = [30, 0.5, 0.5] \tag{4.2}$$

$$\text{R2} = [\text{RWCOMXv, RWPv, RWNv}] = [30, 0.5, 1] \tag{4.3}$$

The comparison of reward functions using Expected Sarsa is illustrated in Figure 4.8.



	Thsf	E[D]	D95	E[Co]	AcSP	Last Sub
🟦 R1	-12.95%	13.64%	12.92%	-4.19%	0.46%	14.03%
🟧 R2	-12.77%	13.35%	12.70%	-22.44%	0.68%	14.03%

Figure 4.8: A performance comparison of reward functions using Expected Sarsa.

Figure 4.8 shows that the reward functions R1 and R2 achieve almost the same results except for the E[Co] parameter. However, when the E[Co] parameter is examined, the R2 function shows significantly superior results compared to the R1 function.

Based on the outcomes, we can say that the increase in the negative reward given when the collisions exceed the determined limit caused a decrease in the average number of collisions per successful transmission.

5 Conclusion and future developments

The demand for machine type communication (MTC) and IoT communications through LTE-Advanced networks is increasing every passing year. However, LTE-A was designed for low-density human-to-human traffic and is not suitable for handling high-density massive machine type communication (mMTC). As a result, a large number of MTC devices attempting to access the base station (BS) can cause severe congestion that leads to performance degradation. Reinforcement learning (RL)-based congestion control policies are proposed to meet the need for a new congestion control policy that can effectively manage congestion caused by mMTC traffic.

This study presents a simplified version of the Access Class Barring (ACB) algorithm that utilizes RL to adjust the access probability (barring rate) dynamically to maximize network performance. A grant-free access protocol has been used in this scenario to minimize the energy consumption of devices by eliminating the need for frequent network access requests.

The main objectives of this study are to explore the potential of RL algorithms (Q-learning, Double Q-learning, and Expected Sarsa) for the design of congestion control policies for mMTC traffic in 5G networks and analyze the impact of RL parameters on the performance of congestion control. The performance of the RL-based congestion control policies was evaluated using discrete-event simulation, and they were compared with an ideal and a heuristic congestion control policy. The comparison was made by considering the performance parameters subframe throughput, average delay, 95 percentile of the delay, last subframe, and average number of collisions per successfully transmitted packet.

The simulation results show that RL-based policies outperformed the ideal policy in reducing the average number of collusions and increasing access success probability. However, this performance has been achieved at the expense of extending the access period, which may lead to a slight increase in access delay. This trade-off between energy consumption and access delay can be beneficial in scenarios that focus on reducing average collisions rather than minimizing access delay. Since the ideal policy only aims to reduce the last subframe performance parameter, it shows better results than RL-based policies in other performance parameters such as last subframe and access delay. However, RL algorithms are more flexible, as they can be designed to optimize other performance parameters by adequately defining the reward function.

Comparing the proposed policies within themselves, it can be seen that the policy based on Expected Sarsa stands out in almost all parameters, while the policy based on double Q-learning showed the lowest performance, except for the parameters average number of collisions and access success probability. Then, the policies are compared in terms of access period, and it is observed that all RL-based policies have a longer access period than the ideal policy. Besides, Double Q-Learning exhibits the shortest access period, while Expected Sarsa has a slightly longer access period and produces results closest to the ideal policy.

The study also investigated the impact of state representation, action set, and reward function selection on performance. It was found that increasing the contribution of previous states to the current state definition led to a reduction in the average number of collisions, resulting in a shorter access time and lower energy consumption. Furthermore, using an action set with more elements yielded better overall performance by providing a wider range of feasible values to change the access probability, leading to better system performance optimization. Lastly, increasing the negative reward given when the collisions exceed the predetermined limit resulted in a decrease in the average number of collisions per successful transmission.

In conclusion, RL-based congestion control policies can effectively reduce collisions and improve network efficiency but may require careful tuning of parameters to achieve maximum performance across different metrics. The trade-off between access delay and energy consumption should be considered when designing congestion control policies for mMTC scenarios in 5G networks.

Bibliography

[1] "https://101blockchains.com/iot-connectivity-industry-forecast/" [Online]

[2] 3GPP, TS 23.682, Architecture enhancements to facilitate communications with packet data networks and applications, Mar 2016.

[3] A. Lo, Y. Law, and M. Jacobsson, "A cellular-centric service architecture for machine-to-machine (M2M) communications," IEEE Wireless Commun. Mag., vol. 20, no. 5, pp. 143–151, 2013.

[4] S. Y. Lien, K. C. Chen, and Y. Lin, "Toward ubiquitous massive accesses in 3GPP machine-to-machine communications," IEEE Commun. Mag., vol. 49, no. 4, pp. 66–74, 2011.

[5] M. Y. Cheng, G. Y. Lin, H. Y. Wei, and A. C. C. Hsu, "Overload control for machine-type-communications in LTE-advanced system," IEEE Commun. Mag., vol. 50, no. 6, pp. 38–45, 2012.

[6] T. N. Weerasinghe, V. Casares-Giner, I. A. M. Balapuwaduge and F. Y. Li, "Priority Enabled Grant-Free Access With Dynamic Slot Allocation for Heterogeneous mMTC Traffic in 5G NR Networks," in *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 3192-3206, May 2021, doi: 10.1109/TCOMM.2021.3053990.

[7] M. Tavana, A. Rahmati, and V. Mansouri. Congestion control with adaptive access class barring for lte m2m overload using kalman filters. Computer Networks, 141:222
– 233, 2018. ISSN 1389-1286. doi:https://doi.org/10.1016/j. comnet.2018.01.044.

[8] L. Tello-Oquendo, J. Vidal, V. Pla, and L. Guijarro. Dynamic access class barring parameter tuning in lte-a networks with massive m2m traffic. In 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), pages 1–8. 2018. 54

[9] L. Tello-Oquendo, D. Pacheco-Paramo, V. Pla and J. Martinez-Bauset, "Reinforcement Learning-Based ACB in LTE-A Networks for Handling Massive M2M and H2H Communications," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, 2018, pp. 1-7, doi: 10.1109/ICC.2018.8422167.

[10] D. Pacheco-Paramo, L. Tello-Oquendo, V. Pla, and J. Martinez-Bauset. Deep reinforcement learning mechanism for dynamic access control in wireless networks handling mmtc. Ad Hoc Networks, 94:101939, 2019. ISSN 1570-8705. doi:https://doi.org/10.1016/j.adhoc.2019.101939.

[11] 3GPP, TR 37.868, Study on RAN Improvements for Machine Type Communications, Sep 2011.

[12] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: The MIT Press, 2018.

[13] C. Bockelmann et al., “Towards massive connectivity support for scalable mMTC communications in 5G networks,” IEEE Access, vol. 6, pp. 28969–28992, 2018.

[14] A. C. Cirik, N. M. Balasubramanya, L. Lampe, G. Vos, and S. Bennett, “Toward the standardization of grant-free operation and the associated NOMA strategies in 3GPP,” IEEE Commun. Stand. Mag., vol. 3, no. 4, pp. 60–66, Dec. 2019

[15] A. T. Abebe and C. G. Kang, “Comprehensive grant-free random access for massive & low latency communication,” in Proc. IEEE Int. Conf. Commun., Paris, France, 2017, pp. 1–6.

[16] S. Sesia, M. Baker, and I. Toufik, LTE - The UMTS Long Term Evolution: From Theory to Practice. Wiley, 2011, pp. 421–456.

[17] 3GPP, TS 36.321, Medium Access Control (MAC) Protocol Specification, Sep 2017.

[6] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J. R. Vidal, V. Casares-Giner, and L. Guijarro, “Performance Analysis and Optimal Access Class Barring Parameter Configuration in LTE-A Networks with Massive M2M Traffic,” IEEE Trans. Veh. Technol., vol. PP, no. 99, pp. 1–1, 2017. DOI 10.1109/tvt.2017.2776868.

[19] A. Laya, L. Alonso, and J. Alonso-Zarate, “Is the Random Access Channel of LTE and LTE-A Suitable for M2M Communications? A Survey of Alternatives,” IEEE Commun. Surv. Tuts., vol. 16, no. 1, pp. 4–16, 2014. DOI 10.1109/SURV.2013.111313.00244.

[20] 3GPP, TS 36.321, Medium Access Control (MAC) Protocol Specification, 2012.

[21] 3GPP, TS 22.011, V15.1.0, Service Accessibility, Jun. 2017.

[22] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," IEEE Access, vol. 7, pp. 133653-133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[23] D. Silver, "Lectures on Reinforcement Learning," 2015. [Online]. Available: https://www.davidsilver.uk/teaching/.

[24] T. G. Dietterich, “Hierarchical reinforcement learning with the MAXQ value function decomposition,” J. Artif. Intell. Res., vol. 13, pp. 227–303, 2000.

[25] A. Kumar, “The Bellman Equation,” Towards Data Science, Apr. 25, 2019. [Online]. Available: https://towardsdatascience.com/the-bellman-equation-59258a0d3fa7.

[26] J. Peng, E. Jury, P. Dönnes, and C. Ciurtin, “Machine Learning Techniques for Personalised Medicine Approaches in Immune-Mediated Chronic Inflammatory Diseases: Applications and Challenges,” Front. Pharmacol., vol. 12, 2021. DOI 10.3389/fphar.2021.720694.

[27] TechTarget, “Reinforcement learning,” [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning#:~:text=Reinforcement%20learning%20is%20a%20machine,learn%20through%20trial%20and%20error.

[28] "Reinforcement Learning Explained Visually: Part 4 - Q-learning Step by Step," Toward Data Science, Nov. 28, 2020, [Online]. Available: https://towardsdatascience.com/reinforcement-learning-explained-visually-part-4-q-learning-step-by-step-b65efb731d3e.

[29] H. van Seijen, H. van Hasselt, S. Whiteson and M. Wiering, "A theoretical and empirical analysis of Expected Sarsa," *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Nashville, TN, USA, 2009, pp. 177-184, doi: 10.1109/ADPRL.2009.4927542.

[30] H. van Hasselt, "Double Q-learning," in Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2 (NIPS'10), Red Hook, NY, USA, 2010, pp. 2613-2621.

[31] T. M. Lin, C. H. Lee, J. P. Cheng, and W. T. Chen, “PRADA: Prioritized random access with dynamic access barring for MTC in 3GPP LTE-A networks,” IEEE Trans. Veh. Technol., vol. 63, no. 5, pp. 2467–2472, 2014.

[32] R. Rivest, “Network control by Bayesian broadcast,” IEEE Trans. Inf. Theory, vol. 33, no. 3, pp. 323–328, May 1987.

List of Figures

List of Tables