Computer Networks 182 (2020) 107493

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Delay-aware dynamic access control for mMTC in wireless networks using deep reinforcement learning

Diego Pacheco-Paramo [a,b,*], Luis Tello-Oquendo [c]

[a] reconoSER ID, Bogotá, Colombia
[b] Escuela de Ciencias Exactas e Ingeniería, Universidad Sergio Arboleda, Bogotá, 111221, Colombia
[c] College of Engineering, Universidad Nacional de Chimborazo, Riobamba 060108, Ecuador

ARTICLE INFO

Keywords:
Delay
Double Deep Q-Learning
Access Class Barring
Massive machine type communications

ABSTRACT

The success of the applications based on the Internet of Things (IoT) relies heavily on the ability to process large amounts of data with different Quality-of-Service (QoS) requirements. Access control remains an important issue in scenarios where massive Machine-Type Communications (mMTC) prevail, and as a consequence, several mechanisms such as Access Class Barring (ACB) have been designed aiming at reducing congestion. Although this mechanism can effectively increase the total number of User Equipments (UEs) that can access the system, it can also harm the access delay, limiting its usability in some scenarios. In this work, we propose a delay-aware double deep reinforcement learning mechanism that can dynamically adapt two parameters of the system in order to enhance the probability of successful access using ACB, while at the same time reducing the expected delay by modifying the Random Access Opportunity (RAO) periodicity. Results show that our system can accept a simultaneously massive number of machine-type and human-type UEs while at the same time reducing the mean delay when compared to previously known solutions. This mechanism can work adequately under varying load conditions and can be trained with real data traces, which facilitates its implementation in real scenarios.

1. Introduction

The Internet of Things (IoT) relies on the ability to gather, process, and analyze massive amounts of data to make the appropriate decisions at the right time. Computing systems have addressed this requirement through the development of robust architectures that support real-time analysis of massive data. From a radio access point of view, several technologies have been proposed which aim at providing extensive coverage, low power, and typically, low data rate transmissions. However, the characteristics of machine-type communications (MTC) [1] impose different demands than those that existed for human-to-human (H2H) communications, causing that current systems do not behave appropriately under these new conditions [2]. Among the possible wireless networks that compete in the IoT market, cellular networks are one of the main options due in part to its extended coverage, existing infrastructure, and standardization efforts. Nevertheless, it is still necessary to adapt some mechanisms to new traffic types such as MTC.

In cellular networks, high traffic loads can be controlled through the Access Class Barring (ACB) mechanism, which defines a barring rate that is broadcasted by the base station, delaying the access of a percentage of the active users (named UE herein). This mechanism can successfully control high loads of simultaneous UEs trying to access the medium, which results in fewer collisions and, therefore, in the long term, a higher amount of UEs accessing the network. However, this mechanism also produces a higher delay, which could be undesirable for specific applications. Therefore, ACB must be combined with other mechanisms that allow reducing its impact on delay. This can be done by varying the periodicity of random access opportunities (RAOs), which define the available slots where the devices can contend for network access. We propose a Delay-Aware Double Deep Q-learning mechanism that can adapt both the barring rate of ACB, in order to increase the successful access probability of UEs even under very high load scenarios, and also the RAO periodicity to reduce the impact on delay. These parameters are adapted dynamically, which allows the system to work successfully under different traffic conditions. Since Reinforcement Learning (RL) is a method that relies on data, we use traces obtained from a Telco to represent H2H traffic. We also rely on the fact that both the barring rate and the RAO periodicity can be modified; that is, our solution complies with the standards.

The contributions of this paper can be summarized as follows:

* Corresponding author at: reconoSER ID, Bogotá, Colombia.
E-mail address: diego.pacheco@reconoserid.com (D. Pacheco-Paramo).

https://doi.org/10.1016/j.comnet.2020.107493
Received 30 January 2020; Received in revised form 13 June 2020; Accepted 15 August 2020
Available online 25 August 2020

- We model the access control problem as a Partially Observable Markov Decision Process (POMDP) and design an adaptive access control mechanism based on double deep RL that modifies the barring rate of ACB and the RAO periodicity simultaneously. The ability to modify the RAO periodicity allows greater control of the access delay, which considerably reduces the mean access delay when compared to our previous work in [3]. Hence, in this paper it is possible to optimize two conflicting objectives: Increase the successful access probability and reduce the mean access delay.
- We evaluate the system under simultaneous H2H and Machine-to-Machine (M2M) traffic, where the former is obtained from traces of a Telco, and the latter is modeled after the standards for heavy load scenarios.
- We compare our Delay-Aware Double Deep Q-Learning mechanism with two previous dynamic solutions and show that our mechanism can reduce the mean access delay while maintaining full successful access probability.

The remainder of this paper is organized as follows. In Section 2, we review the different solutions that have been proposed for handling the access control problem with mMTC. In Section 3, we detail the random access procedure and the ACB scheme following the 3GPP standards closely. Then, in Section 4, we model the access control problem as a POMDP and describe our Delay-Aware Double Deep Q-Learning implementation. In Section 5, we evaluate our proposed scheme under different traffic conditions and compare it against two other solutions that are also dynamic through different key performance indicators (KPIs). Finally, Section 6 concludes the paper. Note that in Table 1 we provide the notations used throughout the paper.

2. Related work

There have been numerous research efforts devoted to optimizing the ACB barring factor for handling massive MTC (mMTC) connection attempts on the RACH through either static or dynamic approaches [4–9]. However, some studies [10–12] offer complex procedures, use questionable assumptions for getting high performance, or do not conform with network specifications (e.g., without considering the number of uplink grants or the updating period of notification information by the base station).

Duan et al. [13] presented an ACB scheme that calculates the optimal barring rate at each RAO based on the estimation of the number of contending UEs using preamble information (i.e., successful, unused). They also provide a scheme to dynamically select the number of available preambles allocated to MTC devices. The performance of the ACB schemes mentioned above is typically compared with that of idealized solutions that exploit the advantages of having full state information [7,13,14]. These full state information solutions are impractical but provide an upper bound to the performance of the ACB scheme. In this paper, we use as a benchmark the idealized and full state information scheme presented by Duan et al. [13].

RL-based ACB schemes are suitable approaches to optimize the access control for wireless networks, and in particular, for cellular networks such as LTE-A and NB-IoT [15–17]. El-Hameed and Elsayed [18] propose a Q-Learning mechanism that aims to assign preambles to H2H or M2M UEs according to the traffic intensity. However, their scheme requires that the system knows how much traffic per UE type is offered to assign access priorities, similarly as Extended Access Barring. Bello et al. [19] propose a Q-Learning approach in which each M2M UE has to learn when to transmit. This mechanism does not use ACB, and it is entirely decentralized. Bear in mind that decentralized access schemes do not conform to current LTE-A recommendations. Likewise, Yu et al. [20] propose a decentralized mechanism to optimize access control. In this case, the UEs can learn the features of different coexisting medium access control mechanisms, and adjust their transmissions using cognitive radio. Although this scenario is very promising, it heavily relies on the processing capacities of the terminals, which is not feasible in scenarios with low-power, low-processing devices such as those frequently found in IoT applications. A deep RL-based ACB scheme was proposed in [21]; it considered differentiated MTC services so that the high priority MTC UEs could transmit their data in a short time. Moon et al. [22] used Q-learning to adjust the barring factor by observing the access success rate. However, in these proposals is assumed that the base station has complete knowledge of the number of UEs contending for resources in the network, which is impractical.

In our previous work [3], we proposed a Double Deep Q-Learning (DDQL) solution that was able to adapt to dynamic traffic conditions through the barring rate but suffered from high delays. In this work, we present a new mechanism that considerably reduces the access mean delay while maintaining the successful access rate concerning our previous proposal. We are able to address this two conflicting objectives by simultaneously adapting the barring rate and the periodicity of the RAOs, unlike our previous work which only addressed the reduction of congestion through the adaptation of the barring rate. Also, this mechanism can work properly only with the available information at the base station which allows its integration into current cellular systems.

3. Random access procedure and access control

The Random Access (RA) procedure is performed every time that a UE wants to switch from idle to connected mode. The UEs first acquire the network configuration parameters; then, they are subjected to the ACB scheme; finally, after passing the barring check, they perform the RA procedure as illustrated in Fig. 1.

The Random Access Channel (RACH) is used to signal the connection request; it is allowed in predefined time/frequency resources, hereafter RAOs [23,24]. The base station has a number of preambles (r) available for initial access to the network; these preambles [24,25] are transmitted by the UEs for attempting the first access to the network. The *Master Information Block* (MIB) and the *System Information Blocks* (SIBs) are resources used by the base station to broadcast the configuration information periodically. In particular, the SIB2 includes some basic parameters, such as the periodicity of RAOs (T_{RAO}) and the barring parameters. There are six possible values for $T_{\mathrm{RAO}} = \{1, 2, 3, 5, 10, 20\}$ measured in ms. These values are set by the parameter *prach-ConfigIndex* [24,26]. Note that according to the specifications [26], the SIB2 messages are sent every 80 ms, and therefore when T_{RAO}=5, there are 16 RAOs between two SIB2 messages, and when T_{RAO}=20, there are only four RAOs between two SIB2 messages, as can be seen in Fig. 2.

Upon arrival, the UEs are subjected to the ACB scheme. The main goal of ACB is to redistribute the access requests of UEs through time using a barring rate (P_{ACB}) and a barring time (T_{ACB}); by doing so, the number of access requests per RAO is reduced. This fact helps to evade massive-synchronized accesses demands to the RACH, which might endanger the fulfillment of QoS objectives. UEs subjected to the ACB scheme must perform a barring check before initiating the RA procedure (i.e., before the transmission of their first preamble) as described in Algorithm 1 [26,27]. There are 16 possible values for $P_{\mathrm{ACB}} = \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1\}$.

UEs that succeed in the barring check are no longer subject to the ACB scheme and proceed to perform the RA procedure as follows.

A four-message handshake is performed in the contention-based random access. In Message 1 (*Msg1*), a UE transmits a randomly chosen preamble from the preamble pool during one of the available RAOs. A preamble will be detected at the base station if it has not been chosen by more than one UE in the same RAO. Otherwise, a collision occurs. Then, the base station sends a random access response message, Message 2 (*Msg2*), which includes one uplink grant for each detected preamble. *Msg2* is used to assign time–frequency resources to the UEs for the transmission of Message 3 (*Msg3*). UEs wait for a predefined

Table 1
Important notation utilized in this paper.

Notation	Description
IoT	Internet of Things
QoS	Quality of Service
mMTC	Massive Machine-type Communication
ACB	Access Class Barring
UE	User Equipment
RAO	Random Access Opportunity
H2H	Human-to-human Communications
RL	Reinforcement Learning
POMDP	Partially Observable Markov Decision Process
DDQL	Double Deep Q-Learning
3GPP	Third Generation Partnership Project
LTE-A	Long-term Evolution Advanced
NB-IoT	Narrow Band IoT
KPI	Key Performance Indicator
RA	Random Access
RAR	Random Access Response
RACH	Random Access Channel
PRACH	Physical RACH
MIB	Master Information Block
SIB	System Information Block
RTT	Round-trip Time
EPS	Evolved Packet System
r	Available preambles for contention-based RA
prach-ConfigIndex	PRACH Configuration Index
preambleTransMax	Maximum number of preamble transmissions
T_{RAO}	Periodicity of RAOs
P_{ACB}	Barring rate
T_{ACB}	Barring time
W_{RAR}	RAR window size
N_{psu}	Number of preambles successfully received
N_{RAR}	Maximum number of uplink grants per subframe
N_{UL}	Maximum number of uplink grants per RAR window
P_d	Preamble detection probability
BI	Backoff Indicator
ER	Size of the experience replay buffer
γ	Discount Factor
ϵ	Exploration probability
N_L	Num. hidden layers (feedforward neural network)
τ	Update period (events) of the second neural network
$\mathcal{A}$	Delay-Aware DDQL Action set
$\mathcal{S}$	Delay-Aware DDQL State set
a	Action that represents a combination of P_{ACB} and T_{RAO} values
s	State
$\mathcal{R}$	Reward set associated to the action a taken in state s
$Q(s,a,\theta)$	Q value for a given action a, taken in state s and a neural network defined by the set θ
θ^- and θ	Weights that define each neural network
$\overline{N_{psu}}$	Mean number of successfully received preambles among the RAOs available between two SIB2 messages
$CV_{N_{psu}}$	Coefficient of variation of the number of successfully received preambles among the RAOs available between two SIB2 messages
ΔN_{psu}	Difference of $\overline{N_{psu}}$ between the current and the previous observation
L_i	Iteration loss function in DDQL
Y^{DDQL}	Target for DDQL
N_{pt}	Number of UEs transmitting a preamble on a given RAO
D	Access Delay
k	Number of preamble transmissions
P_s	Probability of successful access

time window to receive the uplink grant. If the end of this window receives no uplink grant 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. That is, they select a new preamble and transmit it at the next RAO. The UEs that receive an uplink grant send their connection request message, *Msg3*, using the resources specified by the base station. Finally, the base station responds to each *Msg3* transmission with a contention resolution message (*Msg4*). The interested reader is referred to [23,26,28–30] for further details.

4. Delay-Aware Double Deep Q-learning solution

As explained in Section 3, the access control mechanism in the Base Station defines specific slots called RAOs, when UEs are allowed to request access to the system. The period between RAOs (T_{RAO}) is defined by the parameter *prach-ConfigIndex*. Also, the number of simultaneous UEs that are allowed to contend for access to the network in a specific RAO is controlled through the ACB barring probability P_{ACB}. Normally, there is a trade-off between the successful access probability and the mean access delay. We propose a new solution that aims to reduce the congestion (and increase the successful access probability) on each RAO through P_{ACB} while at the same time reducing the mean access delay through T_{RAO}. That is, in scenarios with high load, our solution should reduce P_{ACB} so fewer UEs contend for access, while at the same time reducing T_{RAO}, so more RAOs are available in the same

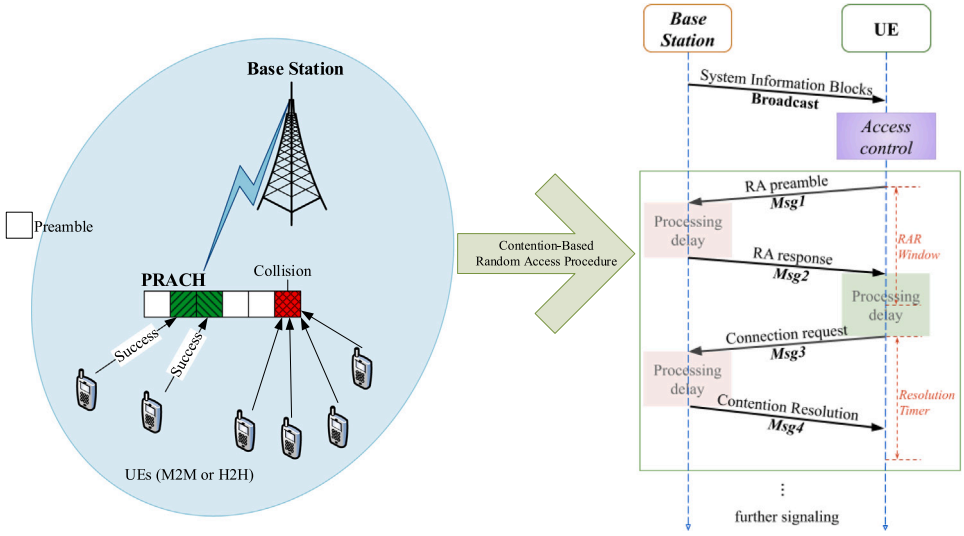
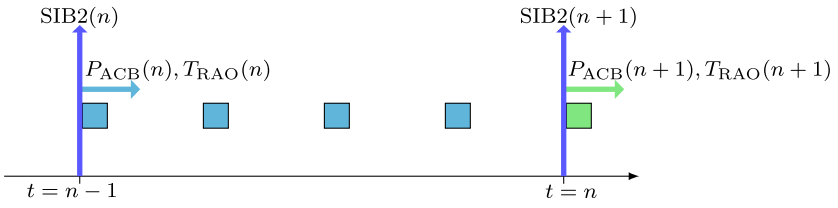
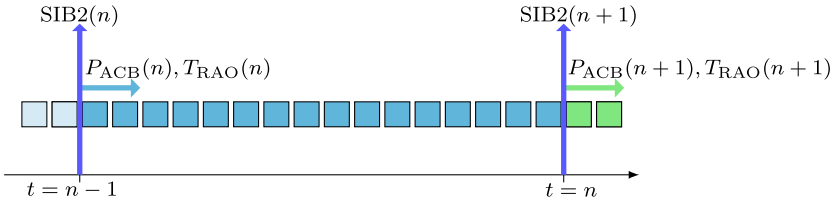


Fig. 1. Contention-based random access procedure.



(a) $T_{\mathrm{RAO}} = 20\,\mathrm{ms}$; 4 RAOs.



(b) $T_{\mathrm{RAO}} = 5\,\mathrm{ms}$; 16 RAOs

Fig. 2. Random Access Opportunities for different T_{RAO} values.

Algorithm 1: ACB Scheme

repeat
  Set the mean barring time $T_{\mathrm{ACB}}(n)$ and the barring rate $P_{\mathrm{ACB}}(n)$ broadcast by the base station in the nth SIB2;
  Generate $\mathcal{U}[0,1)$ = a random number with uniform distribution between 0 and 1;
  if $\mathcal{U}[0,1) \leq P_{ACB}(n)$ **then**
    initiate the random access procedure;
  else
    Generate a new $\mathcal{U}[0,1)$;
    Set the barring time as

$$T_{\mathrm{barring}} = [0.7 + 0.6\,\mathcal{U}[0,1)]\; T_{\mathrm{ACB}}(n); \tag{1}$$

    wait for T_{barring};
  end
until *the random access procedure is initiated*;

window of time, reducing the overall delay. In this section, we first model the access control problem as a POMDP. Then, we propose a Double Deep-Q Learning solution to the POMDP.

4.1. System model:POMDP

As illustrated in Fig. 1, the Base Station provides access to both M2M and H2H UEs distributed under its coverage area. However, the base station does not differentiate between the two types of UEs. The Base Station can only know the number of contending UEs by the number of preambles successfully received N_{psu} on each RAO. However, the number of received preambles might differ from the number of sent preambles due to collisions or transmission errors. Collisions occur when two or more UEs transmit the same preamble in the same RAO. Therefore, the Base Station cannot know the real number of contending UEs. On the other hand, the Base Station can only communicate with the UEs through SIB2 messages to update the T_{RAO} and P_{ACB} values. Hence, we have a discrete-time system, synchronized with the constant period of SIB2 messages, as it can be seen in Fig. 2. The state–space S of the system is defined on these specific times and should account for the accumulation of observations that occurred during each period.

Therefore, in Fig. 2(a), the state at time n is composed of the observations of 4 RAOs, while in Fig. 2(b), the state at time n is composed of the observations of 16 RAOs.

Each state s that belongs to $\mathcal{S}$ is composed of 5 variables: $s = \left(\overline{N_{psu}}, CV_{N_{psu}}, \Delta N_{psu}, P_{\text{ACB}}, T_{\text{RAO}}\right)$. First, there is the mean number of successfully received preambles among the RAOs available between two SIB2 messages, $\overline{N_{psu}}$. Therefore in Fig. 2(a), the mean will be calculated between 4 observations, while in Fig. 2(b) the mean will be calculated between 16 observations. These values are discretized so $\overline{N_{psu}} \in \mathbb{N}$. However, since there are only $r = 54$ available preambles, $\overline{N_{psu}} \leq 54$. Due to the variations of received preambles that can occur between two RAOs, we have included $CV_{N_{psu}}$, which is the coefficient of variation of the number of successfully received preambles among the RAOs available between two SIB2 messages. These values are discretized so, $CV_{N_{psu}} \in \{0, 0.2, 0.4, 0.6, 0.8\}$. In order to understand how the traffic is changing between different SIB2 periods, we have included ΔN_{psu}, which represents the difference of $\overline{N_{psu}}$ between the current (n) and the previous observation ($n-1$). This value only represents three variations: If the difference is positive, that is if the mean traffic is increasing, ΔN_{psu}=1. If the traffic is decreasing, ΔN_{psu}=2. If the mean traffic remains constant, ΔN_{psu}=3. Finally, we include in the state s, the values of P_{ACB} and T_{RAO}, which directly affect the number of contending UEs on the observed RAOs. As it was seen in Section 3, there are 16 possible values for P_{ACB} and 6 possible values for T_{RAO}. Therefore, the state–space $\mathcal{S}$ is composed of 79 200 states. The action set $\mathcal{A}$ is defined by the two control variables of the system, that is P_{ACB} and T_{RAO}. That is, each action a, represent a combination of P_{ACB} and T_{RAO} values that will affect the next SIB2 period. Therefore, $\mathcal{A} = \{1, 2, 3, \ldots, 96\}$.

Finally, we define the set of rewards $\mathcal{R}$ associated to each state in $\mathcal{S}$. It is quite complex to define specific rewards for each of the 79 200 states. Therefore, in order to define the set of rewards $\mathcal{R}$, we have decided to focus on the impact that each of our control variables have on the performance of the system. First, we will focus on P_{ACB}, which controls the ACB mechanism, and aims at reducing the number of collisions on each RAO. According to [30], the Probability Mass Function (PMF) of the number of successfully received preambles N_{psu} at the base station when there are N_{pt} UEs transmitting a preamble on a given RAO, and assuming that every preamble reaches the base station is:

$$P_n(z) \triangleq \Pr(z \mid n) = \sum_{c=0}^{c_{\max}} P_n(z, c) \tag{2}$$

being

$$P_n(z, c) = \frac{r - (z - 1 + c)}{r} P_{n-1}(z - 1, c) + \frac{z + 1}{r} P_{n-1}(z + 1, c - 1) + \frac{c}{r} P_{n-1}(z, c), \tag{3}$$

where $P_0(0, 0) = 1$, n is the number of contending UEs in a RAO (i.e., the UEs that transmit a preamble selected among the r available preambles with equal probability), z is the number of preambles selected by exactly one UE, c is the number of collided preambles, and $c_{\max} = \min\{r, \lfloor n/2 \rfloor\}$.

Following Eq. (2), we can obtain the number of received preambles with the highest probability for a given number of sent preambles, as shown in Fig. 3. It can be seen that while the number of sent preambles in a RAO is lower or equal to 10, it is more likely that the base station successfully receives as many preambles as were sent. However, when there are more than 10 preambles sent in a RAO, it is more likely to successfully receive less preambles than those that were sent. Collisions cause this; that is, we are not considering other causes such as transmission errors. Therefore, if we want to reduce the uncertainty on the base station associated with collisions, the number of UEs that transmit should be equal or lower than 10. Although it is not possible to guarantee this through ACB, we define our reward function in such a way that it promotes this behavior by penalizing the system every time that $\overline{N_{psu}} > 10$, that is, when it is evident that more than 10 UEs are contending for access. On the other hand, we want that the system is able to reduce the delay, by creating more RAOs when the traffic is high. Therefore, our reward function promotes this behavior through an adequate value of T_{RAO} while considering the current value of P_{ACB}.

Table 2
Reward Function $\mathcal{R}$.

P_{ACB}	T_{RAO}	$\overline{N_{psu}}$	$CV_{N_{psu}}$	ΔN_{psu}	r
–	Low	–	–	–	−100
–	High	–	–	–	−100
Very High	Medium	Low	< 0.4	1	20
Very High	Medium	Low	< 0.4	2,3	80
Very High	Medium	Low	>= 0.4	1	20
Very High	Medium	Low	>= 0.4	2,3	40
Very High	Medium	Medium	< 0.4	1	20
Very High	Medium	Medium	< 0.4	2,3	60
Very High	Medium	Medium	>= 0.4	2,3	20
Very High	Medium	High	< 0.2	2	20
Very High	Medium	Very High	< 0.2	1	−100
Very High	Medium	Very High	< 0.2	2,3	−80
Very High	Medium	Very High	>= 0.2	1	−100
Very High	Medium	Very High	>= 0.2	2,3	−80
Medium	Medium	Medium	< 0.4	1	40
Medium	Medium	Medium	< 0.4	2	80
Medium	Medium	Medium	< 0.4	3	60
Medium	Medium	Medium	>= 0.4	1	40
Medium	Medium	Medium	>= 0.4	2,3	60
Medium	Medium	High	< 0.2	2	40
Medium	Medium	High	< 0.2	3	20
Medium	Medium	High	>= 0.2	2	20
Medium	Medium	Very High	< 0.2	1	−60
Medium	Medium	Very High	< 0.2	2,3	−40
Medium	Medium	Very High	>= 0.2	1	−60
Medium	Medium	Very High	>= 0.2	2,3	−40

The detailed reward function $\mathcal{R}$ is described in Table 2. We have used values between [-100,100] in order to show more clearly the differences in rewards. For a better understanding of the criteria used to define the reward function, it is necessary to define ranges for P_{ACB}, T_{RAO} and $\overline{N_{psu}}$. In the case of P_{ACB}, we have defined five ranges: *very low* when $P_{\text{ACB}} < 0.3$, *low* when $0.3 \leq P_{\text{ACB}} < 0.5$, *medium* when $0.5 \leq P_{\text{ACB}} < 0.7$, *high* when $0.7 \leq P_{\text{ACB}} < 1$ and *very high* when $P_{\text{ACB}} = 1$. In the case of T_{RAO} we have defined three ranges: *low* when $T_{\text{RAO}} < 3$, *medium* when $3 \leq T_{\text{RAO}} < 10$ and *high* when $T_{\text{RAO}} >= 10$. Finally, for $\overline{N_{psu}}$ we have defined four ranges: *low* when $\overline{N_{psu}} \leq 3$, *medium* when $3 < \overline{N_{psu}} < 7$, *high* when $7 \leq \overline{N_{psu}} \leq 10$ and *very high* when $N_{psu} > 10$. Please notice that the ranges for N_{psu} were chosen according to the explanation above. The first observation that must be done about $\mathcal{R}$ is that those action-state combinations that do not appear in Table 2 have reward $r = 0$. On the other hand, all the action-states combinations where T_{RAO} is *low* or *high* are avoided, and have a reward $r = -100$. In the former case, this is done to avoid having too many RAOs, which might result in a waste of resources in the base station. In the latter case, this is done to avoid increasing the mean delay. A *very high* value of P_{ACB} is desirable when there is *low* traffic in order to accept UEs with a very low delay. Therefore, we reward this action when $\overline{N_{psu}}$ is *low*. The amount of the reward depends on $CV_{N_{psu}}$ and ΔN_{psu}. The reward grows when there is a small variation and when the traffic does not grow. When $\overline{N_{psu}}$ is *medium* we use the same criteria, but in this case the rewards are smaller. If $\overline{N_{psu}}$ is *high*, then we only give a reward when there is very little variation and the traffic is decreasing. On the other hand, it is not convenient to accept all UEs when $\overline{N_{psu}}$ is *very high*, and therefore we assign negative values for r, based on $CV_{N_{psu}}$ and ΔN_{psu}. Since we want that the system adapts to dynamic traffic conditions, we have to set rewards when $P_{\text{ACB}} < 1$. The objective of reducing P_{ACB} is to maintain $\overline{N_{psu}}$ below 10, according to what was explained earlier. Therefore, we consider that the *medium* range of P_{ACB}

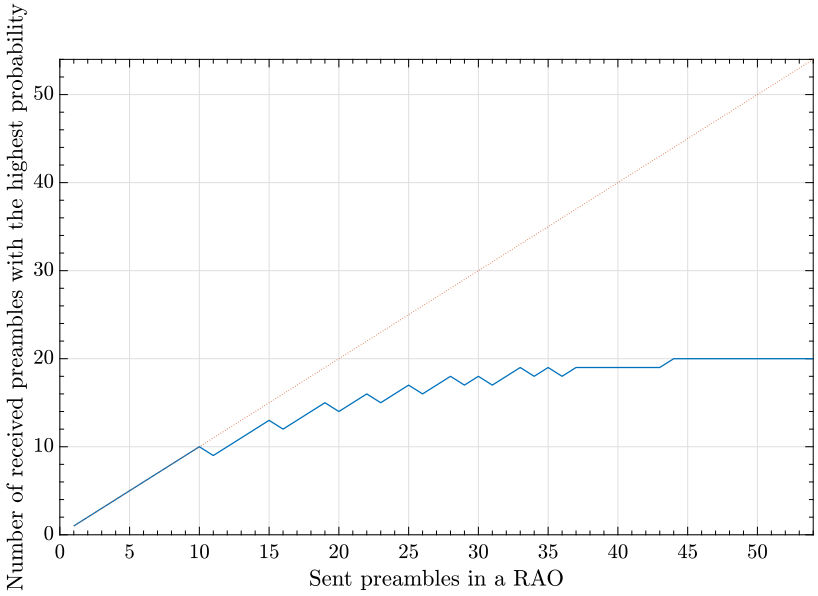


Fig. 3. Number of received preambles with highest probability.

is adequate for that purpose. Since P_{ACB} should not be lower than zero when $\overline{N_{psu}}$ is *low*, the reward remains zero in those cases. As $\overline{N_{psu}}$ grows to *medium*, we assign rewards depending on $CV_{N_{psu}}$ and ΔN_{psu}. The reward is higher if $CV_{N_{psu}}$ is low, and grows as ΔN_{psu} shrinks. If $\overline{N_{psu}}$ grows to *high*, the same criteria is used, although the rewards are lower. On the other hand, if $\overline{N_{psu}}$ is *very high*, then the assignment of P_{ACB} has not been successful, and therefore we try to avoid those states.

4.2. Delay-Aware Double Deep Q-learning implementation

Double Deep Q-Learning [31] (DDQL), aims at optimizing an objective function while representing the action values $Q(s,a)$ through a neural network [32]. Therefore, this solution replaces the Q tables used in traditional Q-Learning [33] with a neural network, which allows representing continuous state spaces or action sets, but more importantly, it can assign action values to previously unvisited states. On each iteration i of DDQL, the system aims to minimize a loss function defined by:

$$L_i = \frac{1}{2}\left(r + \gamma\, Q(s', \max_{a' \in \mathcal{A}} \left[Q(s', a', \theta_i)\right], \theta_i^-) - Q(s, a, \theta_i)\right)^2, \tag{4}$$

where r is the reward associated to the action a taken in state s, γ is the discount factor, which controls the impact of future rewards, $Q(s', a', \theta_i^-)$ represents the Q value for a given action a', state s' and neural network defined by the set θ_i^-, and $Q(s, a, \theta_i)$ represents the Q value for a given action a, taken in state s and a neural network defined by the set θ_i. The sets θ_i^- and θ_i represent the weights that define each neural network for iteration i. This error function is minimized on each iteration by modifying the weights θ_i (associated with the prediction). Please notice that this error minimization function depends on the definition of $\mathcal{R}$, and therefore it is expected that in the long term the impact of $\mathcal{R}$ grows. As it was shown, DDQL uses two different neural networks aiming at reducing the inherent overestimations that result from calculating the future rewards of taking action a on a state s derived from the *max* operator seen in Eq. (4). This overestimation may result in finding a local optimum, or in divergence [31]. In DDQL, one neural network (θ^-) is used to evaluate the current policy, while the other is used to obtain the action that maximizes future rewards (θ). Therefore, the target for DDQL is represented by:

$$Y^{DDQL} = r + \gamma\, Q(s', \max_{a' \in \mathcal{A}} \left[Q(s', a', \theta)\right], \theta^-), \tag{5}$$

where the roles of each neural network are interchanged every τ iterations; also, experience replay is implemented [34]. The purpose of this technique is to break the dependency of consecutive experiences of the system with the environment by saving them on a buffer (of size ER) and then sampling them randomly for training. In our implementation, each neural network consists of a Multi-layer Perceptron with an input layer with five neurons (one for each variable of the state–space), N_L hidden layers of 5 neurons, and an output layer of 96 neurons (one for each action).

Algorithm 2 describes the Delay-Aware DDQL mechanism. In every RAO, the BS saves the N_{psu} value, as shown in line 17. When a SIB2 message is expected to be sent immediately after the RAO, it is possible to calculate the state variables and obtain s, as it is shown in line 3. Then, it is possible to define what will be the next action that the system will take, that is, the P_{ACB} and T_{RAO} values that will be sent to the UEs in the following SIB2. This action depends on the parameter ϵ, which is the exploration probability. Following a greedy policy, the system will always choose the action that maximizes the expected immediate reward, but it is possible to choose a random one to explore the state–space. In line 5, the system calculates the reward r that results from the values obtained in line 3, and all the information is then saved in the buffer of size NL. If the buffer has reached its capacity, then it is time for training the system. The training will be done in batches of size τ, and will use two neural networks with weights θ and θ^-, respectively. For each observation, the weights θ will be trained to minimize the loss function per iteration, as shown in Eq. (4). Every time the batch of size τ is finished, the weights θ^- are updated to θ. Please recall that the weights θ^- are used to calculate the expected Q value, as shown in Eq. (5).

5. Experimentation

We consider a single base station that provides coverage for UEs of M2M and H2H communications simultaneously. We consider that M2M traffic follows the specifications [35], where it is stated that in high load scenarios, M2M traffic can show a bursty behavior associated with the simultaneous activation of, e.g., alarms, or event-driven wireless sensor networks. This behavior is represented by a Beta distribution (3,4) for 10 s. The intensity for a high load scenario is set in 30 000 M2M UEs. In the case of H2H traffic, we use traces from the Telco Telecom Italia, which provided the data as part of a "Big Data Challenge" in 2014. Since this data is aggregated for periods of 10 min, we consider that H2H traffic has a constant intensity during this period. Also, because this data does not provide units, we use the observation

Algorithm 2: Delay-Aware Double Deep Q-Learning Mechanism.

while there are new RAOs available **do**
  if a SIB2 will be sent immediately after RAO **then**
    calculate $\overline{N_{psu}}$, $CV_{N_{psu}}$, ΔN_{psu}, and state s
    select action a (P_{acb} and T_{RAO}) according to ϵ policy
    calculate reward r
    save a,s,r,s' in buffer
    if buffer of size NL is full **then**
      randomize buffer order
      while there is data left in buffer **do**
        take subset of size τ data from buffer
        while there is data in τ subset **do**
          for data in τ train θ to minimize the loss function in Eq. Eq. (4)
        end while
        clean subset τ data from buffer
        update the neural networks: $\theta^{-}=\theta$
      end while
    end if
  else
    save N_{psu}
  end if
end while

provided in [36], where it is stated that the maximum load that can be provided by a base station is 55 EPS Radio Access Bearer setups per second. Based on this value, we normalize the H2H traffic from the traces. Unless otherwise stated, the values of the system are as appear in Table 3. In this work, we evaluate the performance of the system through different KPIs: Mean Delay $E[D]$, Mean Number of preamble transmissions $E[k]$, and Probability of successful access P_s. These KPIs can be obtained individually for each type of UE, and have been evaluated in previous studies.

It is essential to make a distinction between the training and evaluation of the system. For training our Delay-Aware DDQL solution, we use 144 episodes, each one associated with the data obtained from the traces for H2H UEs for one day, which are aggregated in periods of 10 min. Therefore, on each of these episodes, the H2H traffic is constant, and the traces give its intensity. We have picked the trace obtained on November 1 from the most active cell on that day. In the case of M2M traffic, we use on each episode a beta distribution (3,4) with 30 000 UEs. The parameter ϵ is set to 1 at the beginning of each episode, and then it is reduced linearly to 0. On the other hand, in the evaluation phase, we use a single episode where H2H traffic is constant and has a maximum intensity of 55 user arrivals per second. For M2M traffic, we also use a beta distribution (3,4), although the intensity changes according to the experiment. Please notice that although training is done with 30 000 M2M UEs, the experiments can be done with different values: 1000, 10 000, 20 000, 30 000 and 40 000 M2M UEs, that is, the system does not need to be retrained for each intensity value. For the evaluation of the Delay-Aware DDQL solution, the parameter ϵ is set to 0, that is, we use the best policy. In every result shown, we perform 100 independent experiments.

In Figs. 4 and 5, the variation of P_{ACB} and T_{RAO} can be seen for our Delay-Aware DDQL mechanism when there are 30 000 M2M UEs and the Backoff Indicator is set to 120 ms. It can be seen that at the beginning of the simulation, as M2M UEs begin to transmit, P_{ACB} suffers a significant reduction that sets its value close to 0, and then again to 1. This behavior occurs because when there are no UEs, T_{RAO} is 20, and when suddenly UEs begin to transmit, the system rapidly reacts to adapt to this new traffic conditions. After this, the system maintains T_{RAO} between 3 and 5. It should be noted that part of the dynamic adaptation of P_{ACB} and T_{RAO} consists of simultaneously reducing T_{RAO} with P_{ACB} in order to reduce collisions. In Fig. 5, it can be seen that after the RAO number 5000, there is a substantial reduction of T_{RAO}. This occurs not because the value is set to 0, but because whenever a simulation ends as all the UEs are served, the value for the rest of RAOs is set to zero by default. Therefore, the normal behavior when the load goes back to low values is to set T_{RAO} to 5.

Table 3
Default System Configuration for Evaluation Purposes.

Parameter	Setting
PRACH Configuration Index	*prach-ConfigIndex* = 6 [26]
Periodicity of RAOs	5 ms [26]
Subframe length	1 ms [26]
Available preambles for contention-based random access	$r = 54$ [26]
Maximum number of preamble transmissions	*preambleTransMax* = 10 [26]
RAR window size	$W_{\mathrm{RAR}} = 5$ subframes [26]
Maximum number of uplink grants per subframe	$N_{\mathrm{RAR}} = 3$ [26]
Maximum number of uplink grants per RAR window	$N_{UL} = W_{\mathrm{RAR}} \times N_{\mathrm{RAR}} = 15$ [26]
Preamble detection probability for the kth preamble transmission	$P_d = 1 - \frac{1}{e^k}$ [35]
Backoff Indicator	BI = 20 ms [26]
Re-transmission probability for *Msg3* and *Msg4*	0.1 [26]
Maximum number of *Msg3* and *Msg4* transmissions	5 [26]
Preamble processing delay	2 subframes [28]
Uplink grant processing delay	5 subframes [28]
Connection request processing delay	4 subframes [28]
Round-trip time (RTT) of *Msg3*	8 subframes [28]
RTT of *Msg4*	5 subframes [28]
Discount factor	$\gamma = 0.7$ [3]
Num. hidden layers (feedforward neural network)	$N_L = 10$ [3]
Update period (events) of the second neural network	$\tau = 100$ [3]
Buffer size for experience replay	$ER = 500$ [3]

In Fig. 6, the performance of the Delay-Aware DDQL mechanism can be observed when there are 30 000 M2M contending UEs. It can be seen that the system maintains the number of successful accesses below 10, according to the definition of our reward function. In fact, it is kept most of the time between 7 and 8, which is sufficient to keep the collision number below 6. Let us recall that the system can control only the number of UEs contending for the first time, and therefore we can see that the total number of transmitted preambles reaches almost 18.

In Table 4, we compare three solutions for the primary three KPIs. The first solution is our previously presented DDQL mechanism [3] that only adapts P_{ACB}. It can be seen that it shows the worst performance in terms of access delay, although it uses the lowest value of Backoff Indicator (BI = 20 ms) that guarantees the lowest mean access delay. The second solution is our proposed Delay-Aware DDQL mechanism, evaluated with a BI = 120 ms. It can be seen that the total delay is considerably reduced (more than 2s) due to the variation of T_{RAO}. The last solution was the dynamic resource allocation (DRA) proposed by Duan et al. [13], and it also dynamically modifies P_{ACB} and the number of available preambles for MTC UEs. In order to compare this algorithm accurately, we increased the number of maximum retries in the system from 10 to 150. By doing this, and increasing the BI value to 960 ms, it is possible to reach a 100% probability of successful access. It can be seen that the mean delay is close to the one obtained by our presented solution, although the mean number of transmissions is a lot higher, which is detrimental for the energy consumption of MTC devices. The proposed Delay-Aware mechanism can maintain the mean number of transmissions below 2, although it is moderately higher than our previous DDQL mechanism. This is expected since, for the DDQL and Delay-Aware schemes, the maximum number of retries follows the standard (i.e., 10 preamble retransmissions), and therefore a mean value as the one shown by [13] would result in a meager value of P_s.

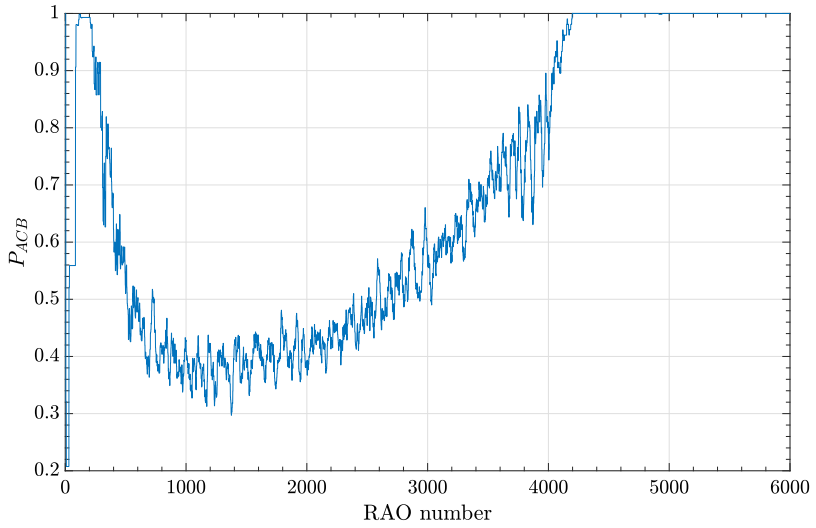


Fig. 4. P_{ACB} variation for Delay Aware DDQL mechanism with 30 000 M2M users.

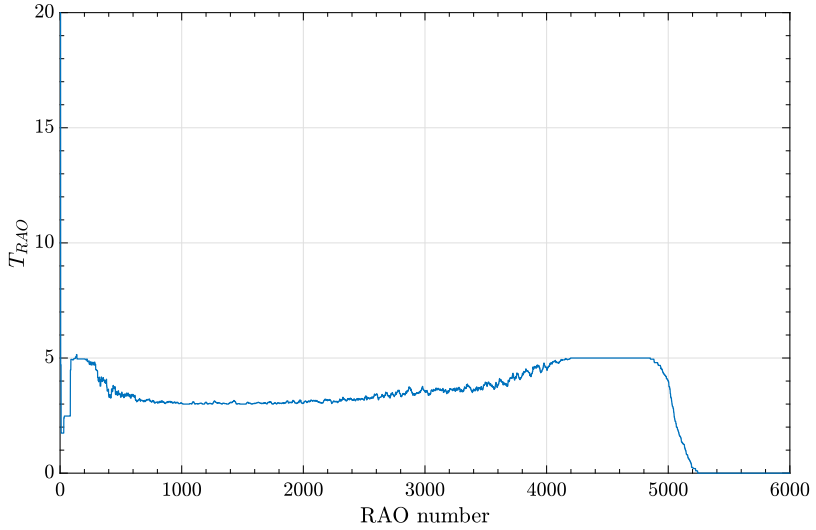


Fig. 5. T_{RAO} variation for Delay Aware DDQL mechanism with 30 000 M2M users.

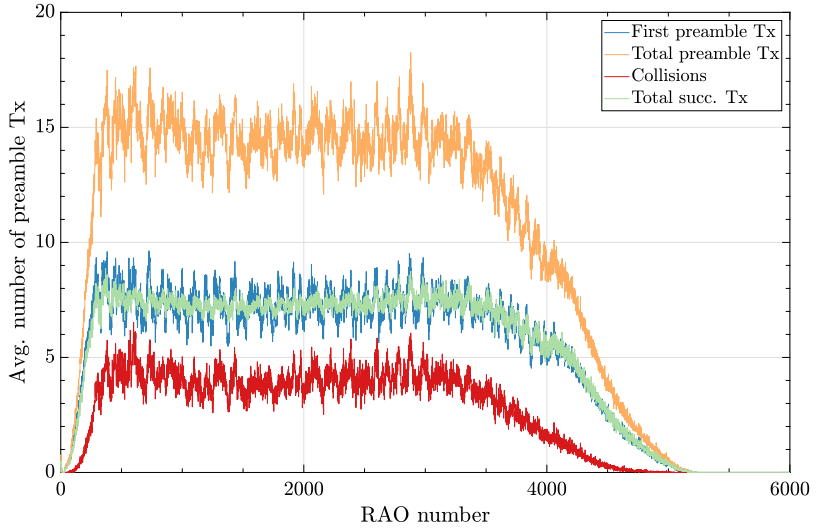


Fig. 6. Preamble transmissions for Delay-Aware DDQL mechanism with 30 000 M2M users.

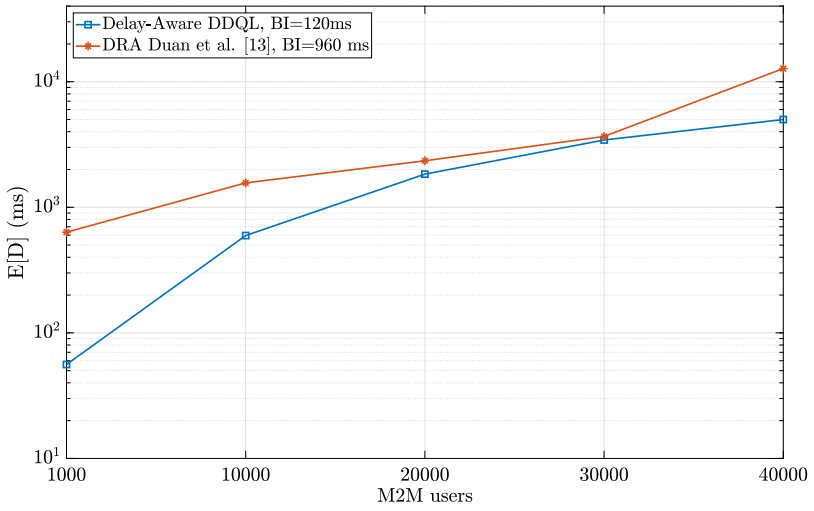


Fig. 7. Mean Access Delay for Delay Aware DDQL mechanism vs. Duan et al. [13] for various M2M loads.

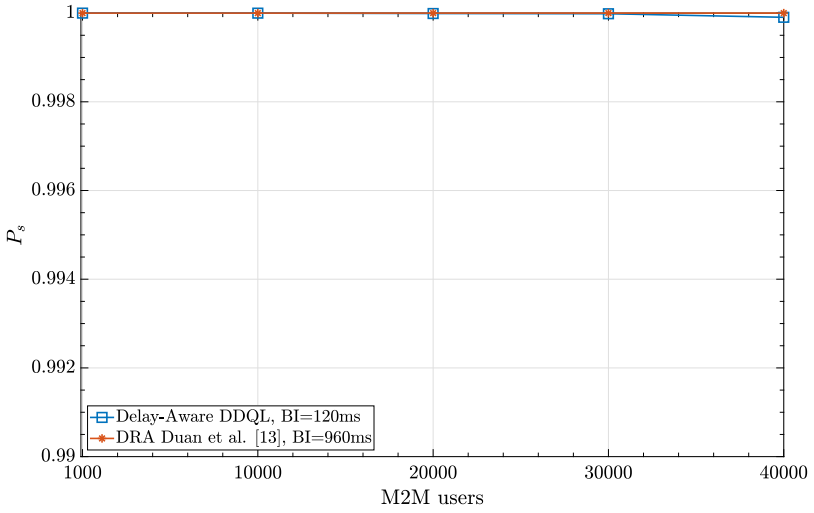


Fig. 8. Probability of Successful Access for all users: DDQL mechanism vs. Duan et al. [13] for various M2M loads.

Table 4
Performance of Access Control Mechanisms for 30 000 M2M UEs.

KPI	DDQL	Delay Aware DDQL	Duan et al. [13]
$E(D)_{Total}$ (s)	5.87	3.43	3.66
$E(D)_{M2M}$ (s)	5.96	3.47	3.69
$E(D)_{H2H}$ (s)	2.98	1.65	2.18
$E(K)_{Total}$	1.77	1.89	8.38
$E(K)_{M2M}$	1.77	1.90	8.45
$E(K)_{H2H}$	1.63	1.72	5.36
$P_{sa-Total}$ (%)	100	100	100
P_{sa-M2M} (%)	100	100	100
P_{sa-H2H} (%)	100	100	100

In Fig. 7, we compare our proposed solution against the DRA solution proposed in [13] as the M2M load varies from 1000 to 40 000 UEs. For the Delay-Aware DDQL mechanism, we use a BI = 120 ms and maintain the maximum number of retries is 10, as it is proposed in the 3GPP standards. For the DRA solution, we use a BI = 960 ms and increase the maximum number of retries to 150. These changes on the DRA solution are done in order to do a fair comparison that guarantees that the acceptance rate is 100% for 30 000 UEs. It can be seen that our proposed solution can maintain a lower delay than the DRA mechanism for every value of M2M UEs. Also, it can be seen that when there are only 1000 M2M UEs, the system can adapt in order to reduce the mean access delay to 56 ms. This value is increased to 5 s when there are 40 000 M2M UEs, less than the mean access delay for our previous DDQL solution with 30 000 M2M UEs. The ability of the system to adapt to different loads shows that it is not necessary to retrain the system as the traffic changes. One interesting aspect is that the DRA solution seems to be reaching the performance of our proposed Delay-Aware DDQL mechanism as the number of UEs grows. However, this behavior is a result of our effort to pick the parameters for DRA that would guarantee full access with a load of 30 000 M2M UEs. Therefore, as the load of M2M is farther from 30 000 UEs, the performance of that algorithm degrades, as can be seen when we have 1000 UEs, or 40 000 UEs. This tendency would continue if we tried with 50 000 UEs.

In Fig. 8, we compare the P_s for all UEs for the two previously mentioned mechanisms as the M2M load varies from 1000 to 40 000

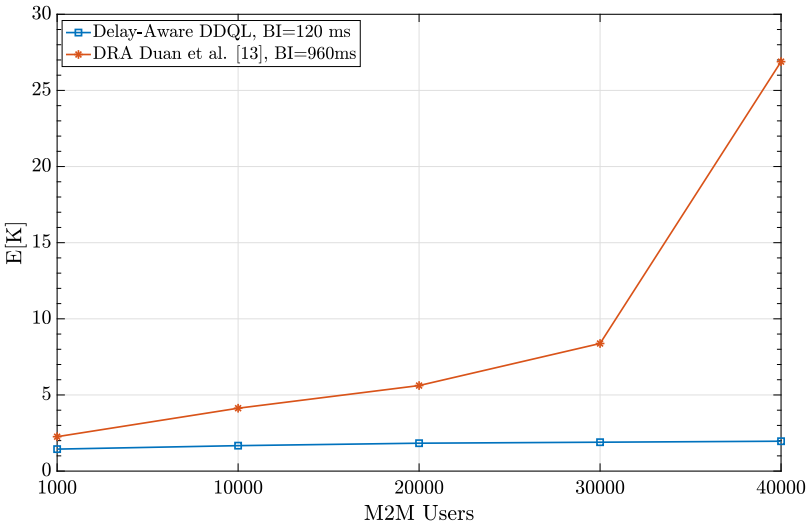


Fig. 9. Mean number of transmissions for all users: DDQL mechanism vs. Duan et al. [13] for various M2M loads.

UEs. It can be seen that the DRA mechanism can maintain $P_s = 100\%$. This occurs because we have increased the maximum number of retries to 150, which is 15 times more than what is proposed in the standards. If we reduce this value, the performance of this mechanism will suffer considerably. On the other hand, our proposed mechanism is not able to maintain $P_s = 100\%$ for the full range of M2M values tested. In fact, when there are 40 000 M2M UEs, $P_s = 99.99\%$, that is, we have a minimal error of around 0.01%. This is an acceptable value considering that we have increased the load 30% over the maximum load proposed in the standards.

In Fig. 9, we compare the mean number of transmissions for all UEs between our proposed Delay-Aware scheme and the DRA solution as the M2M load varies from 1000 to 40 000 UEs. It can be seen that our solution maintains E[K] below 2, even when there are 40 000 UEs. When there are 1000 UEs, E[K]=1.44, and when there are 40 000 UEs, this value increases to 1.96. On the other hand, for the DRA mechanism, this value goes from 2.26 when there are 1000 M2M UEs to 26.89 when there are 40 000 UEs. Having such a high number of transmissions can reduce the lifetime of IoT devices considerably, and therefore can directly affect the long-term performance of the applications.

6. Conclusion

In this work, we proposed a Delay-Aware Double Deep Q-Learning mechanism for access control that dynamically adapts in order to allow successful mMTC access while reducing the mean access delay. This mechanism can coordinate the modification of two parameters (P_{ACB} and T_{RAO}) under different traffic conditions. We have evaluated our system when two types of traffic coexist: M2M and H2H communications. The former was modeled as bursty traffic according to the specifications. The latter was defined following traces from a Telco. Our proposed scheme shows a significant improvement over our previous Double Deep Q-Learning scheme in terms of delay, while it slightly increases the mean number of transmissions. Also, it performs better than a previously well-known dynamic solution. We have validated its performance under different traffic scenarios, showing its ability to perform in real conditions without retraining. The approach proposed in this work shows that it is possible to control several parameters through online tools based in reinforcement learning in order to address multi-objective problems in wireless networks, which could be used for other QoS problems. As future work, we intend to apply similar techniques to fulfill more stringent requirements in wireless communications through the orchestration of several parameters.

CRediT authorship contribution statement

Diego Pacheco-Paramo: Conceptualization, Methodology, Software, Validation, Writing - original draft. **Luis Tello-Oquendo:** Software, Validation, Writing - original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work of Diego Pacheco-Paramo was in part supported by Universidad Sergio Arboleda under project Plataforma de datos para territorios inteligentes, IN.BG.086.19.014. The research of L. Tello-Oquendo was conducted under project CONV.2018-ING010, Universidad Nacional de Chimborazo.

References

[1] T. Taleb, A. Kunz, Machine type communications in 3gpp networks: potential, challenges, and solutions, IEEE Commun. Mag. 50 (3) (2012) 178–184.

[2] P. Castagno, V. Mancuso, M. Sereno, M.A. Marsan, Limitations and sidelink-based extensions of 3GPP cellular access protocols for very crowded environments, Comput. Netw. 168 (2020) 107046.

[3] D. Pacheco-Paramo, L. Tello-Oquendo, V. Pla, J. Martinez-Bauset, Deep reinforcement learning mechanism for dynamic access control in wireless networks handling mMTC, Ad Hoc Netw. 94 (2019) 101939.

[4] L. Ferdouse, A. Anpalagan, A dynamic access class barring scheme to balance massive access requests among base stations over the cellular M2M networks, in: 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE, 2015, pp. 1283–1288.

[5] K.T. Ali, S.B. Rejeb, Z. Choukair, A dynamic access control scheme to balance massive access requests of differentiated M2M services in 5G/HetNets, in: 2017 Sixth International Conference on Communications and Networking (ComNet), IEEE, 2017, pp. 1–6.

[6] N. Li, C. Cao, C. Wang, Dynamic resource allocation and access class barring scheme for delay-sensitive devices in machine to machine (M2M) communications, Sensors 17 (6) (2017) 1407.

[7] M. Tavana, A. Rahmati, V. Shah-Mansouri, Congestion control with adaptive access class barring for LTE m2m overload using Kalman filters, Comput. Netw. 141 (2018) 222–233.

[8] L. Tello-Oquendo, J.-R. Vidal, V. Pla, L. Guijarro, Dynamic access class barring parameter tuning in LTE-networks with massive M2M traffic, in: 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), IEEE, 2018, pp. 1–8.
[9] J.-R. Vidal, L. Tello-Oquendo, V. Pla, L. Guijarro, Performance study and enhancement of access barring for massive machine-type communications, IEEE Access 7 (2019) 63745–63759.
[10] H. Kim, S. Lee, S. Lee, Dynamic extended access barring for improved M2M communication in LTE-A networks, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, pp. 2742–2747, http://dx.doi.org/10.1109/SMC.2017.8123041.
[11] R.-H. Hwang, C.-F. Huang, H.-W. Lin, J.-J. Wu, Uplink access control for machine-type communications in LTE-A networks, Pers. Ubiquitous Comput. 20 (6) (2016) 851–862, http://dx.doi.org/10.1007/s00779-016-0961-5.
[12] C.M. Chou, C.Y. Huang, C.-Y. Chiu, Loading prediction and barring controls for machine type communication, in: 2013 IEEE International Conference on Communications (ICC), IEEE, 2013, pp. 5168–5172, http://dx.doi.org/10.1109/ICC.2013.6655404.
[13] S. Duan, V. Shah-Mansouri, Z. Wang, V.W. Wong, D-ACB: Adaptive congestion control algorithm for bursty M2M traffic in LTE networks, IEEE Trans. Veh. Technol. 65 (12) (2016) 9847–9861.
[14] Z. Wang, V.W. Wong, Optimal access class barring for stationary machine type communication devices with timing advance information, IEEE Trans. Wirel. Commun. 14 (10) (2015) 5374–5387.
[15] N. Jiang, Y. Deng, A. Nallanathan, Traffic prediction and random access control optimization: Learning and non-learning based approaches, 2020, arXiv preprint arXiv:2002.07759.
[16] S.K. Sharma, X. Wang, Towards massive machine type communications in ultra-dense cellular IoT networks: Current issues and machine learning-assisted solutions, IEEE Commun. Surv. Tutor. (2019).
[17] N. Jiang, Y. Deng, A. Nallanathan, J.A. Chambers, Reinforcement learning for real-time optimization in NB-IoT networks, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1424–1440.
[18] A.S. El-Hameed, K.M. Elsayed, A Q-learning approach for machine-type communication random access in LTE-advanced, Telecommun. Syst. (2018) 1–17.
[19] L.M. Bello, P. Mitchell, D. Grace, T. Mickus, Q-learning based random access with collision free RACH interactions for cellular M2M, in: Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on, IEEE, 2015, pp. 78–83.
[20] Y. Yu, T. Wang, S.C. Liew, Deep-reinforcement learning multiple access for heterogeneous wireless networks, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–7.
[21] Z. Chen, D.B. Smith, Heterogeneous machine-type communications in cellular networks: Random access optimization by deep reinforcement learning, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.
[22] J. Moon, Y. Lim, A reinforcement learning approach to access management in wireless cellular networks, Wirel. Commun. Mob. Comput. 2017 (2017) 1–17.
[23] 3GPP, TS 36.321, medium access control (MAC) protocol specification, 2020.
[24] 3GPP, TS 36.211 NR, physical channels and modulation, 2020.
[25] D. Chu, Polyphase codes with good periodic correlation properties (corresp.), IEEE Trans. Inform. Theory 18 (4) (1972) 531–532.
[26] 3GPP, TS 36.331, radio resource control (RRC), protocol specification, 2020.
[27] 3GPP, TS 22.011, v15.1.0, service accessibility, 2020.
[28] 3GPP, TR 36.912, feasibility study for further advancements for E-UTRA, 2018.
[29] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, L. Guijarro, Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic, IEEE Trans. Veh. Technol. 67 (4) (2018) 3505–3520.
[30] L. Tello-Oquendo, V. Pla, I. Leyva-Mayorga, J. Martinez-Bauset, V. Casares-Giner, L. Guijarro, Efficient random access channel evaluation and load estimation in LTE-A with massive MTC, IEEE Trans. Veh. Technol. 68 (2) (2019) 1998–2002, http://dx.doi.org/10.1109/TVT.2018.2885333.
[31] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-Learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2015, pp. 2094–2100.
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[33] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[34] L.-J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Mach. Learn. 8 (3–4) (1992) 293–321.
[35] 3GPP, TR 37.868, study on RAN improvements for machine type communications, 2011.
[36] Nokia, Mobile Broadband solutions for Mass Events, Tech. Rep., Nokia, 2014.

Diego Pacheco-Paramo received the B.S. degree in electronics engineering from Universidad de los Andes, Bogotá, Colombia, in 2004. He received the M.S. and Ph.D. degrees from Universitat Politécnica de Valéncia, Spain, in 2009 and 2013, respectively. From 2012 to 2013, he was a visitor researcher in the Broadband Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, USA. From 2014 to 2015, he was a postdoctoral researcher in the Laboratory of Information, Networking and Communication Sciences (LINCS) as a member of Télécom ParisTech in Paris, France. From 2016 to 2019 he was Assistant Professor at Universidad Sergio Arboleda in Bogotá, Colombia. Currently he is CRO at reconoSER ID.

Luis Tello-Oquendo received the electronic and computer engineering degree (Hons.) from Escuela Superior Politécnica de Chimborazo (ESPOCH), Ecuador, in 2010, the M.Sc. degree in telecommunication technologies, systems, and networks, and the Ph.D. degree (Cum Laude) in telecommunications from Universitat Politécnica de Valéncia (UPV), Spain, in 2013 and 2018, respectively. From 2013 to 2018 he was Graduate Research Assistant with the Broadband Internetworking Research Group, UPV. From 2016 to 2017 he was a Research Scholar with the Broadband Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, GA, USA. He is currently an Associate Professor with the Universidad Nacional de Chimborazo. His research interest include MTC, wireless SDN, 5G and beyond cellular systems, IoT, machine learning. He is a member of the IEEE and ACM. He received the Best Academic Record Award from the Escuela Técnica Superior de Ingenieros de Telecomunicación, UPV, in 2013, and the Extraordinary Doctoral Thesis Award from the Escuela de Doctorado, UPV, in 2019.