A Q-learning Approach for Machine-Type Communication Random Access in LTE-Advanced

Amaal S. A. El-Hameed
amaalgex@yahoo.com

Khaled M. F. Elsayed, senior member IEEE
khaled@ieee.org

Department of Electronics and Communications Engineering
Faculty of Engineering, Cairo University, Giza, Egypt 12613

Keywords: Congestion control, Q-learning algorithm, resource separation, RACH, LTE-Advanced.

Abstract: Due to the wide proliferation of the 3GPP long term evolution (LTE) and LTE-Advanced systems as the air interface for 4th generation (4G) wireless systems and beyond, telecommunication operators became more interested in using the LTE infrastructure to meet the projected surge in demand for Machine-to-Machine (M2M) and IoT communications. As a result, the LTE-Advanced system must evolve to provide these services as an overlay over the original network that is originally designed for human-to-human (H2H) communication. In this setup, M2M communication devices share the same random access channel (RACH) with the H2H communication devices. Since M2M communication is expected to be massive, consequently the RACH is a new bottleneck in the LTE-A system. This triggered the need to enhance the operation of the RACH to meet the needs to support the low power and low data rates of M2M devices. The current standardized scheme to request access to the system is known to suffer from congestion and overloading in the presence of a huge number of devices. Accordingly, recent research pointed to the need of designing more efficient ways to manage the random access channel in such setups. Most of previous works focused on solving the congestion problem in RACH in the presence of M2M solely, but not with the existence of both M2M and H2H services. In this work we propose a new random access channel scheme based on Q-learning approach to reduce the congestion problem. The scheme adaptively divides the available preambles between both M2M and H2H devices in a way that provides an acceptable service for the H2H devices and maximizes the number of active M2M devices. The adaptation is done based on current demand levels from both the H2H and M2M devices and the observed service levels. The results indicate that the proposed approach provides high random access channel success probability for both M2M and H2H devices even with the huge number of M2M devices.

1. INTRODUCTION

Machine-to-machine (M2M) communication system is a network which includes a large number of machine type communication (MTC) devices that can communicate with little or no human intervention in order to accomplish specific tasks [1]. They became a key component of the services driving the evolution of the fifth generation (5G) wireless systems to meet the huge demand driven by the proliferation of Internet of Things (IoT) [2].

The benefits of M2M are numerous. It allows organizations to streamline operations, optimize resources and even create opportunities for new business models.

The characteristics of M2M traffic is different as compared to H2H network traffic. As H2H communication obeys a certain session length, data volume and interaction frequency, M2M traffic follows other specific traffic patterns, which can be described as follows:

- The traffic of the M2M devices is mainly in the uplink direction from an M2M device to the eNodeB.
- Packet sizes are small, but transmitted periodically by numerous endpoints. As most of M2M devices are reporting sensor data, such as temperature, humidity, etc. Thus, they have transmission timing restrictions to maintain a certain data rate or certain QoS to send their data before it becomes useless.

The 3GPP provided extensions such as LTE-M and NB-IoT [3] to address this need. Currently, there exists multiple proprietary low-power long-range wireless networks such as LoRAWAN [4] and Sigfox that provide a dedicated service for MTC. However, these services require the deployment of new equipment which means additional CAPEX and OPEX. On the other hand, to implement the M2M communication as an overlay over the LTE/LTE-A system, connection establishment is needed for both H2H devices and M2M devices. This simultaneous demand causes several problems [5].

One problem is low transmission efficiency, caused by the large size of packet overhead compared to small data size for M2M communications and the overhead of the packets used by MTC devices and H2H communications devices to achieve synchronization with eNodeB as well as contention resolution. Thus, the signalling channel of a network becomes overloaded and collisions occur to fail the access if too many

M2M devices attempt to access the network. The problem is even worse for battery-powered M2M devices where most of their power maybe wasted on channel access. Another problem is congestion, including air interface congestion and core network (CN) congestion. As described in [6], the number of MTC devices within a cell can be significantly large, *e.g.,* thousands of devices accessing a single base station. If a large number of these devices attempt to access the base station within a short period of time, congestion will take place [7].

There are several papers discussing the congestion problem in M2M communications. Different solutions are suggested such as; Access Class Barring (ACB) [9], Dynamic Access Class Barring (DACB) [13], Prioritized Random Access (PRA) [14]. These schemes are detailed in section 2. All these schemes discussed the random access channel congestion problem taking into consideration the existence of M2M devices only, while the proposed approach attempts to solve the congestion in random access channel with the existence of both H2H and M2M devices.

1.1 Contributions of the Paper

This paper addresses the congestion problem caused by MTC devices in the random access channel. The scheme is based on using the Q-learning approach where the system divides the random access resources between both M2M and H2H devices according to two parameters: 1) the QOS required for the H2H communication presented as the accepted level of blocking probability upon requesting access to the system; 2) the number of available random access resources presented as the preambles used for the random access process.

The rest of this paper is organized as follows: In section 2, we provide an overview of related work. Section 3 presents an overview of Q-learning fundamentals, the system model and the details of the proposed approach. Performance evaluation is presented in Section 4. The paper is concluded in Section 5.

2. RELATED WORK

There have been several attempts to handle the problem of random access channel congestion in LTE. They can be shown in [8-10]. Several basic solutions are proposed and studied in [9-12], among which Access Class Barring (ACB) is the main solution [9]. In ACB, the user generates a random number q that compares it with the access barring factor p ; $(0 < p < 1)$. If the random number is smaller than the barring factor, then the user is allowed to access the random access channel with probability p, else the user is permitted probability $1 - p$ for a barring time duration T_b then starts another trial. Other schemes are proposed to solve the congestion problem as Dynamic Access Class Barring (DACB), separate random access resources, Prioritized Random Access (PRA) scheme, Dynamic PRACH (DPRACH) slot allocation method, and Slotted Access [13-17]. In DACB [13], an ACB scheme with an adaptive ACB factor is used. The ACB factor is dynamically adjusted according to the network load state. The separation of resources can be made either by allocating separate RA slots for H2H communication devices and M2M communication devices, or by splitting the available preambles into HTC and MTC subsets. Prioritized RA is another optimization approach based on RA resource separation and ACB mechanism. The available RACHs are virtually separated into three groups: H2H communication, random M2M communication, and scheduled M2M communication/emergency service. A prioritized access algorithm is developed to ensure QoS guarantee for the application classes as well as virtual groups [14-15]. In slotted RA scheme, each M2M device is assigned to a dedicated RA opportunity and only allowed to perform RA in its own dedicated access slot [16].

In addition to normal ACB, extended access barring (EAB) is also proposed as the baseline solution to relieve the RA congestion for massive MTC in LTE systems. In EAB, the M2M communication devices are classified based on their QoS requirements. It selectively controls access attempts from devices that are considered more tolerant to delays [17]. In [18], the 3GPP RAN-Group proposed an adaptive access barring method based on EAB algorithm which optimizes the EAB parameters depending on the congestion coefficient, i.e. the ratio between the collided preambles and the successful preambles over a given time duration. In their proposed method, the eNB does not know the number of M2M devices in its coverage area. Thus, they attempt to predict the number of M2M devices based on the estimation of

access intensity which can be measured at an eNB. Another mechanism is proposed for massive access control. It serves the networks that only contains M2M devices. They differentiate M2M services into delay-sensitive and delay-tolerant services. Based on the traffic loads of the two types of services, they allocate the RACH resource to each type of services [19]. J. Kim, et al. proposed a RA procedure power-ramping scheme that replaces the standard LTE RA procedure to increase the RA success probability even when severe RACH congestion problem occurs. This scheme causes a difference in the received power of messages at the eNB, so that the random access occurs frequently. In addition, it is easy to apply because the proposed scheme adopts the power ramping scheme that has already been used in the cellular system. Moreover, by applying this new technique, a device that has performed many retransmissions would have a higher RA success probability; thus, the connection failure probability could be reduced [20]. Another approach was introduced in [21] where the authors propose two solutions to mitigate the impact of M2M communication in the context of LTE-A network. They model how the random access resources should be allocated to the different types of devices as a bankruptcy problem.

Recently, Q-learning was introduced for solving the problem of random access channel congestion. In [22], the authors propose a reinforcement learning algorithm to vary the barring parameter of ACB to different traffic conditions, hence reducing the congestion and the number of collisions in the RACH. Luis et al. [23] propose a universal MAC protocol based on Deep reinforcement learning (DRL) called DLMA for heterogeneous wireless networks to achieve near-optimal performance with respect to the objective without knowing the operating mechanisms of co-existing MACs.

3. THE PROPOSED Q-LEARNING METHOD

In the proposed approach, we use the Q-learning algorithm to solve the problem of congestion and resource separation between H2H and M2M devices in the random access channel.

3.1 System Model

In LTE, devices access RACH when they request initial network access or when they handover between different cells. RACH is an uplink channel that is used to transfer control information from the mobile device to the network [24-26].

To access the RACH, devices go through a procedure called the random access procedure. It has two types; contention free that is used for the applications with timing restrictions that can't handle any delay. It has 10 preambles of the 64 LTE preambles reserved to him. While the other type is called contention based procedure. It has the rest 54 preambles to use. The procedure consists of four steps as shown in Fig. 1:

1. random access preamble transmission;
2. random access response reception;
3. L2/L3 message transmission;
4. contention resolution reception.

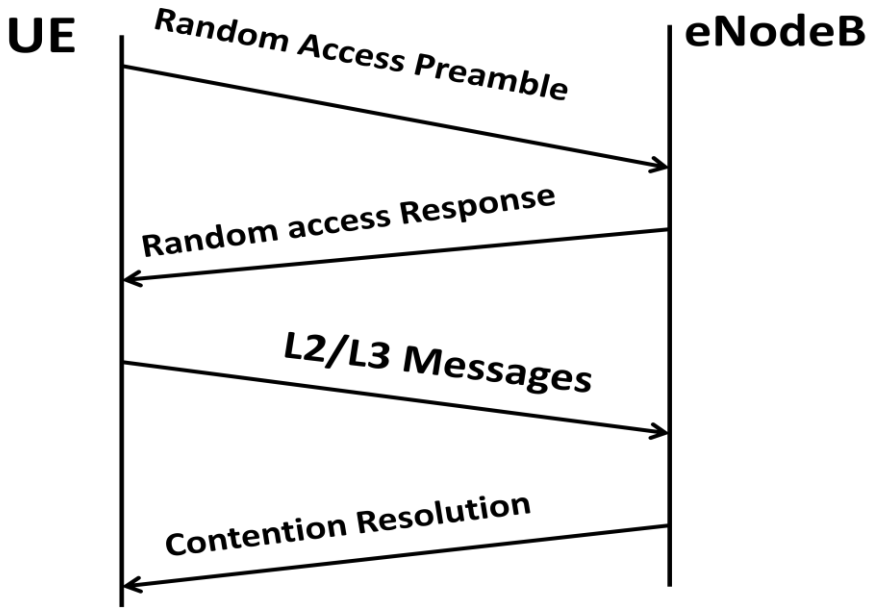

Fig. 1: Contention based random access in LTE

These 54 preambles are available for both M2M communication devices and H2H communication devices to communicate with the eNB as in Fig. 2. We use a bandwidth of 1.4 MHz, where the random access (RA) slot consists of six resource blocks (RBs) each with 180 kHz. The RA slot is used differently by H2H and M2M consequentially. The H2H uses RA slot for performing a random access operation between it and the eNodeB, while M2M uses RA slot for transmitting the data to eNodeB as their data packets are small enough to be transmitted through the RA slot. The random access and the data transmission is done in the six RBs of the RA slot. The preamble is held for different frames by both traffic devices according to how long the connection setup takes by H2H communication devices and how much data is transmitted by the M2M communication devices.

In our system model, we separate resources between both M2M traffic and H2H traffic using the proposed Q-learning algorithm. The proposed Q-learning algorithm is used by the eNodeB to divide the available random access resources between both types of traffics with the consideration of the required QoS for H2H communications. The H2H QoS is the blocking probability that H2H users accept while connection establishment, this value must be less than a threshold value that is defined by the H2H communications, it is either 2% or 5% [27]. At the end of the learning algorithm both traffic types will have the allocated number of preambles that is used by the devices for the random access process while attempting to satisfy the required H2H QoS at the same time.

In the next section, we provide a description for the Q-learning algorithm and how it works.

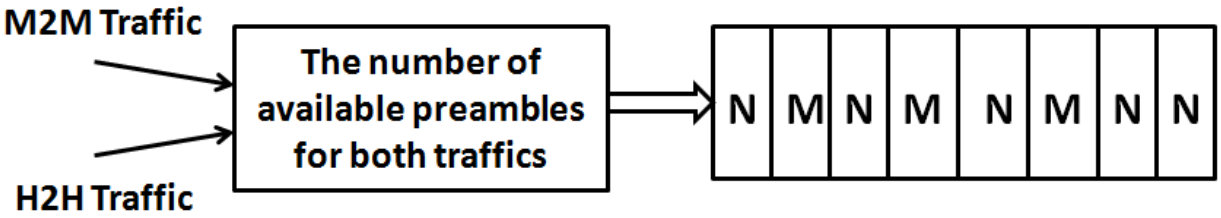

Fig. 2: Random Access in LTE

3.2 Basics of Q-learning

Reinforcement learning is learning how to map situations to actions in order to maximize the reward. There are different types of reinforcement learning algorithms. Q-learning is an example of reinforcement learning algorithms, where an agent interacts with the environment to achieve a certain goal. The agent interacts within the environment through a series of actions that can be performed. At each step, an agent being in a particular state, chooses an action based on what it has learned previously. The actions taken by the agent can have positive or negative outcomes that are called rewards. A reward for each action is calculated and used to determine how good the action works in the current environment. The overall goal is to find an optimal policy that chooses the best action for future iterations that achieves long-term high reward [28-32].

Q-learning was introduced by Watkins in 1989 [29]. An action value function $Q(s_t, a_t)$ at time t is given for every state $s_t \in S$ in a finite Markov decision process, where S is the set of all possible states, and action $a_t \in A(s_t)$ where $A(s_t)$ is the set of all possible actions chosen based on a given policy for state s_p [30]. If action a_t was chosen for state s_t , then the system goes to the next state $s_{t+1} = s' \in S$ with probability $P_{ss'}^{a} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$, and receives a reward r_{t+1} with a certain probability depending on the state s' , $R_{ss'}^{a} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$, where E is the expected value.

A policy is the rule that the agent follows in selecting the actions, given the state it is in. A policy is denoted by $\pi_t(s_t, a_t)$. We define the action value function under a policy π as $Q^{\pi}(s_t, a_t)$:

$$Q^{\pi}(s_t, a_t) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k \; r_{t+k+1} \mid s_t = s, a_t = a\}. \qquad (1)$$

where ɣ is the discount factor, $(0 \leq \gamma \leq 1)$ that decides how important future rewards are for the agent. The objective of Q-learning is to find the optimum policy $\pi^*(s) \in A$ for each state s, such that the total discounted expected reward is maximized. Discounted reward means that after t steps the reward received have less value than the reward received now by a factor γ^{p}.

The Q-learning algorithm iteratively updates a table of all possible states and all possible actions for these states called the Q-table. For each state s and action a pair, the Q-table contains a numerical

value $Q(s,a)$, also called the Q-matrix. It represents the possible reward for taking this action a at this state s. At each time t, the agent selects an action a_{t+1}, the agent calculates a reward r_{t+1} where $r_{t+1} \in R$ and moves to the new state s_{t+1}. On selecting the new state s_{t+1}, the Q-function and Q-table are updated. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest $Q(s,a)$ value in each state. The formula used in updating the Q-function is given by:

$$Q(s_{t+1},a_{t+1}) \leftarrow Q(s_t,a_t) + \alpha(s_t,a_t) \times [r_{t+1} + \gamma \times E\{Q(s_{t+1},a_{t+1}) \mid s_t\} - Q(s_t,a_t)]. \quad (2)$$

where $\alpha(s_t, a_t)$ is the learning rate $(0 < \alpha < 1)$ and represents to what extent the newly acquired information is taken into account. A learning rate α of 1 means that only the most recent rewards are taken into account and a learning rate of 0 means the agent learns nothing, and any current reward is discarded. The value of $E\{Q(s_{t+1},a_{t+1}) \mid s_t\}$ is the estimate of the optimal future value for the action-value function [31-32].

There are different policies that can be used for choosing an action such as the ε-Greedy strategy and the ε-decreasing strategy. The value of ε is in the range $(0 < \varepsilon < 1)$. In ε-Greedy strategy, which is used in our system model, it only selects the best action $(1 - \varepsilon)$ of the time based on the reward and another action is chosen randomly selected for the rest of the time (ε). The higher the value, the more random exploration will occur. The difference between the ε-decreasing strategy and the ε-Greedy strategy is that in the ε-decreasing strategy, the ε decreases with time such that the agent goes through a period of learning by exploring different states then it exploits that for the rest of the time.

3.3 The Proposed Q-learning based RACH Preambles Allocation Scheme

In our system model shown in Fig. 3, there are M M2M communication devices and N H2H communication devices that request access at the beginning of each frame f_t; $(1 < t < T)$. All devices from both types attempt to access the RACH for random access, they send a preamble request for the eNodeB.

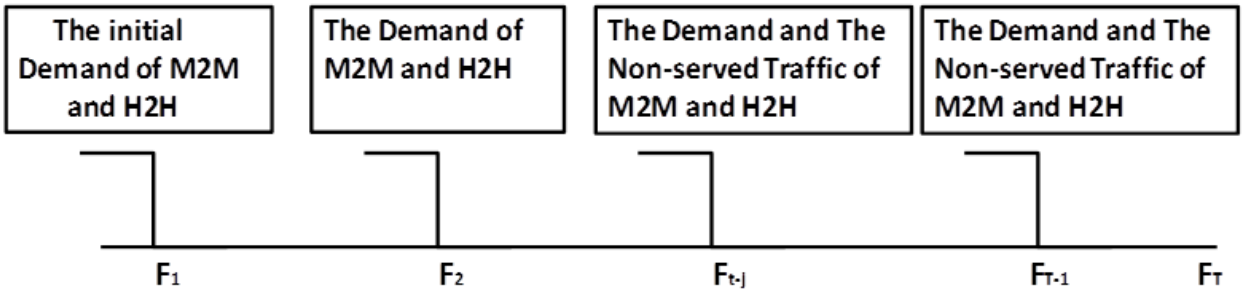

Fig. 3: The Demand Arrival Traffic Model

At the beginning of each frame, we check if the number of available preambles can provide the random access for the demand of both M2M and H2H traffics or not. The non-served users from the present frames are added to the next frame as an added demand for both types of traffic, so that the non-served users can have other opportunities for random access. If any of them succeed to get a preamble, it will be served and out of the system. If they fail to get preambles for consequent 10 frames, they will be considered blocked users. This results in different preambles availability in each frame. If the preambles are equal or greater than the demand, then the preambles are assigned to both types of traffic according to their demands. If not, then the Q-learning algorithm is used for the resources assignment.

The proposed Q-learning algorithm is defined as a Markov decision process (MDP) on a state space consisting of 54 states (the total number of pre-ambles) as in Fig. 4. The state $s_p(t) \in S$ is defined as the number of assigned preambles to M2M devices P_M . The states defined as $s_0(t)$ has $P_{M0} = 0, P_{H0} = 54$, the state $s_p(t)$ has $P_{Mp} = p, P_{Hp} = 54 - p$, and $s_{54}(t)$ will have $P_{M54} = 54, P_{H54} = 0$.

The second parameter of the Q-learning is the action $a_p(t)$; $(1 < p < 54)$, which defines the transition between the states $s_p(t)$; $(1 < p < 54)$. The action $a_p(t)$ is taken at state $s_p(t)$ based on two important input parameters at each state. The first parameter is the total traffic demand of both M2M and H2H devices that includes both the new requests and the blocked requests from the previous frames. The second parameter is the reward value $r_j(t-1)$; $(1 < j < 54)$, which is the third parameter of the Q-learning algorithm. The action $a_p(t)$ is specified below as function of the reward $r_p(t)$. How the action is taken is illustrated below in more details.

The target of the proposed scheme is to find the preambles' assignment to both types of traffic. The target of the policy used is to ensure that M2M devices do not affect the H2H QoS defined as the H2H RACH blocking probability. This condition is represented by the reward value $r_p\,(t)$ calculated at each state. The reward $r_p\,(t)$ represents the difference between the actual blocking probability of H2H users and the threshold blocking probability which is the maximum blocking probability value that H2H users accept through the connection establishment. It can be defined as:

$$r_p\,(t) = \left(\mathrm{PB_p(t)} - \mathrm{PB_{Th}}\ \right) \times 100\,. \tag{3}$$

where $PB_p(t)$ is the actual calculated blocking probability of H2H users at state $s_p(t)$ and $\mathrm{PB_{Th}}$ is the blocking probability threshold value.

The reward can be either negative $r_p\,(t) < 0$, which means that M2M traffic has no effect on the H2H traffic, or positive $r_p\,(t) > 0$, which means that M2M traffic affects the H2H blocking probability and more preambles must be assigned to H2H traffic to ensure its QoS. For each of both positive and negative values, there is a range of values that defines the action $a_p\,(t)$ taken and the next state $s_{p'}\,(t)$. The transition $a_p\,(t)$ and the value of state $s_{p'}\,(t)$ can be different in each time frame according to the reward $r_p\,(t-1)$ value. The reward value range is divided into three ranges defined as "Small Difference" R_1 , "Medium Difference" R_2 , and "Large Difference" R_3 as shown in Table 1. All values used in these ranges are empirically proposed for the proposed approach.

Small Difference: For the positive reward $r_p\,(t)$, it means that the H2H blocking probability is slightly higher than the threshold blocking probability due to the small effect of the M2M devices on the H2H users. Thus, the Q-learning scheme adds one preamble to the H2H devices and thus moves from $s_p(t) \rightarrow s_{p-1}(t+1)$. For negative reward $r_p\,(t)$, it means that the H2H blocking probability is slightly smaller than the threshold blocking probability, thus the Q-learning scheme allocates one more preamble to the M2M devices and thus moves from $s_p(t) \rightarrow s_{p+1}(t+1)$.

Medium Difference: For the positive reward $r_p\,(t)$, it means that M2M devices have high effect on the H2H users, thus the Q-learning scheme allocates two more preambles to the H2H to reflect the M2M effect and thus move from $s_p(t) \rightarrow s_{p-2}(t+1)$.

For negative reward $r_p(t)$, it means that the H2H blocking probability is smaller than the threshold blocking probability, thus the Q-learning scheme allocates two more preamble to the M2M devices and thus moves from $s_p(t) \rightarrow s_{p+2}(t+1)$.

Large Difference: For positive reward $r_p(t)$, it means that M2M devices have huge effect on the H2H users service, thus increasing the number of blocked H2H users, which is not acceptable by H2H traffic. This can be alleviated by increasing the number of H2H assigned preambles by a large number in one step. That is done by adding five preambles to H2H users thus moving from $s_p(t) \rightarrow s_{p-5}(t+1)$.

For negative reward $r_p(t)$, it means that the H2H blocking probability is much smaller than the threshold blocking probability, thus we can add more preambles to M2M devices by moving at a time from $s_p(t) \rightarrow s_{p+5}(t+1)$.

Table 1: The Reward ranges and corresponding actions.

Reward Ranges	**The Reward Values** $r_p(t-1)$	**The Action** $a_p(t)$
Small Difference $R_1 = [0, 1.5]$	$r_p(t-1) \leq R_1$	For positive $r_p(t-1)$: Move from $s_p(t) \rightarrow s_{p-1}(t+1)$ For negative $r_p(t-1)$: Move from $s_p(t) \rightarrow s_{p+1}(t+1)$
Medium Difference $R_2 = [1.5, 3]$	$r_p(t-1) \leq R_2$	For positive $r_p(t-1)$: Move from $s_p(t) \rightarrow s_{p-2}(t+1)$ For negative $r_p(t-1)$: Move from $s_p(t) \rightarrow s_{p+2}(t+1)$
Large Difference $R_3 => 3$	$r_p(t-1) \leq R_3$	For positive $r_p(t-1)$: Move from $s_p(t) \rightarrow s_{p-5}(t+1)$. For negative $r_p(t-1)$: Move from $s_p(t) \rightarrow s_{p+5}(t+1)$.

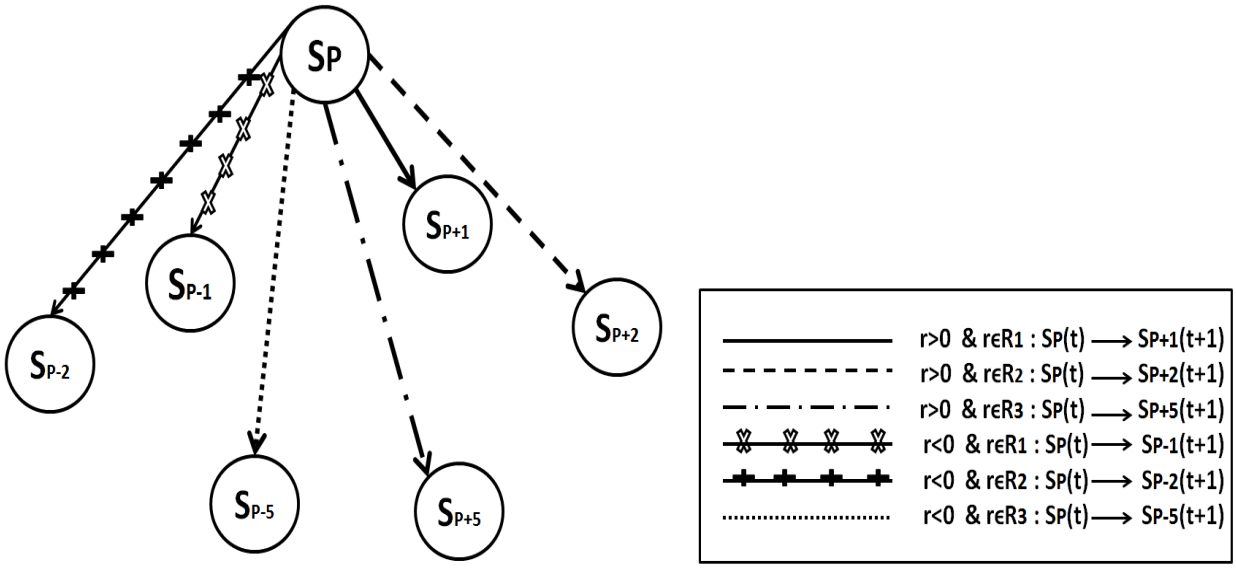

Fig. 4: The Q-learning Markov Decision Process

All the transitions are bounded within the range of $(P_{min},\ P_{max})$, where P_{max} is the maximum available preambles at this time frame and P_{min} is the minimum number of available preambles (that is equal to 0). For example, if the present state is s_p and the next state is s_{p+5}, then $p+5$ is bounded by P_{max} such that $(p+5) \leq P_{max}$, . If the next state is s_{p-5}, then it is bounded by 0 such that $(p-5) \geq P_{min}$.

3.4 The Execution of the Q-learning Scheme

At the beginning of each frame, a check is done to see if the number of available preambles P can or cannot satisfy the random access demand of both M2M and H2H traffic. If they cannot satisfy the demand, we start the Q-learning algorithm. We start at an initial state, s^{init}, where the preambles are initially equally divided between both types of traffics such that $P_M^{init} = \lceil P/2 \rceil, P_H^{init} = P - \lceil P/2 \rceil$. The reward $r(t)$ of the current state; $s^{init}(t)$; is calculated as is Eqn. (3) and according to it an action $a'(t+1)$ for the next state $s'(t+1)$ is chosen to either increase the number of preambles assigned to H2H devices if the reward is positive, or increase the number of preambles assigned to M2M devices if the reward is negative, and $\mathrm{Q}(\mathrm{s}'(\mathrm{t}), \mathrm{a}'(\mathrm{t}))$ is updated as in Eqn. (2). At the end of the algorithm, it returns the number of preambles assigned for both types of traffic $P_M(t)$, $P_H(t)$ and the action-value function $\mathrm{Q}(\mathrm{s}'(\mathrm{t}), \mathrm{a}'(\mathrm{t}))$. At the next frame, when the Q-learning algorithm is called, the algorithm uses the last

state $s'(t-1)$ as its initial state, where the same number of preambles are assigned to both traffics. Then, it goes through the whole previous steps of the algorithm and updates the Q-table using the last value of Q-function $\mathrm{Q}(\mathrm{s}'(\mathrm{t}-1), \mathrm{a}'(\mathrm{t}-1))$ as in Eqn. (2) and returns from it with the new number of preambles assigned for both types of traffic.

The system works for a period of time called the learning phase, where it explores all different states and sets the Q-function matrix that is also called the Q-matrix. The Q-matrix is a matrix of the states as its rows and the actions as its columns. It contains all Q-function values of all states s_i, actions pairs a_j, where $1 \leq i \leq 54 \; and \; 1 \leq j \leq 6$ and it is calculated through the algorithm using Eqn. (2).

Also, the Q-table is set and updated through the whole learning period, until it is finally set at the end of the learning phase. The Q-table is a look-up table that contains all states $s_i; 1 \leq i \leq 54$, all actions taken by these states $a_j; 1 \leq j \leq 6$, all rewards $r_{ij}(s_i, a_j)$ resulting from taking action a_j at state s_i and the new reward $r_{new}(s_i, a_j)$ that represents the actual reward that is calculated when the new preambles are assigned for both traffic users. An example for the Q-table is shown in Table. 2.

Table.2: An Example of the Q-table.

States s_i	**Actions a_j**	**Reward $r_{ij}(s_i, a_j)$**	**New Reward $r_{new}(s', a')$**
s_2	a_4	$r_{24}(s_2, a_4)$	$r_{new}(s_3, a_{new})$
s_{35}	a_6	$r_{35\,6}(s_{35}, a_6)$	$r_{new}(s_{40}, a_{new})$

The learning period is the time needed for the Q-matrix to converge. In the simulation, we noted that the algorithm takes around 400 to 420 time frames till the convergence condition is met and the learning phase is finished. As the RMS of the difference is a meaningful measure of the error, we use the RMS value of the error in two subsequent Q-matrices to represent the convergence of the Q-learning approach. The *RMS* value of the action value function $Q(s_t, a_t)$, is calculated for the difference matrix $Q_{Diff}(t)$ given by

$$Q_{Diff}(t) = Q(t+1) - Q(t) \; , \qquad 0 < t < T \tag{4}$$

Then, we evaluate the RMS value for the difference matrix $Q_{Diff}(t)$ as

$$RMS_{Q_{Diff}}(t) = \sqrt{\frac{\sum_{i=1}^{54} \sum_{j=1}^{6} |Q_{Diff}(i,j)|^2}{54 \times 6}} \; . \tag{5}$$

We use this value to check the system convergence. The convergence condition is represented as

$$RMS_{Q_{Diff}} \leq 0.01 . \quad (6)$$

An example of the Q-function evolution can be shown in Table. 3, where we can see the different states $s(t)$, the calculated Q-function of this state $\mathrm{Q}(s(\mathrm{t}), a(\mathrm{t}))$, the next state $s'(t+1)$ and its Q-function $\mathrm{Q}(s'(\mathrm{t}+1), a'(\mathrm{t}+1))$.

3.4.1 Coping with Dynamic Traffic: Re-learning

After the learning phase, the system uses both the Q-matrix and the Q-table for the preambles assignments. It chooses the action $a'(t+1)$ with Max $\mathrm{Q}(\mathrm{s}(\mathrm{t}), \mathrm{a}'(\mathrm{t}))$ from the Q-matrix that moves the system to the state $s'(t+1)$ that provide the preambles assignments that serve both demands and attempt to satisfy the blocking probability for H2H traffic. Then the new blocking probability for H2H is calculated and the new reward $r_{new}(s', a')$ is evaluated as in Eqn. (3). After that we check the Q-table for the new reward value $r_{new}(s', a')$ and compare it with the value stated in the table from the learning phase $r_{ij}(s_i, a_j)$. This will result in two cases:

1) If the evaluated reward $r_{new}(s', a')$ is equal to the reward stated in the Q-table with error of $(1\% \; to \; 5\%)$, then the system decides there is no change in the load and the system continues using the current Q-matrix for the preambles assignment.
2) If the evaluated reward $r_{new}(s', a')$ is different than the reward stated in the Q-table, then the system decides there is a change in the traffic load. Thus, the Q-learning algorithm starts the learning phase again starting with initial state as current state to handle this traffic load change. It continues updating the Q-table till the convergence, then it uses the new Q-table and Q-matrix to handle the system traffic as will be demonstrated in section 4.3. We illustrate the proposed Q-learning algorithm in Fig. 5.

Table. 3: An Example of the Q-matrix Evolution

Present State $S(t)$ and $\mathbf{P_M}, \mathbf{P_H}$	**The reward** $\mathbf{r(t)}$	**Next Q-Function** $\boldsymbol{Q(s'(t+1), a'(t+1))} = [\,]_{54\times6}$	**Next state $S'(t+1)$ and** $\mathbf{P'_M}, \mathbf{P'_H}$
$S(t) = 14$ $P_M(t)$for $M2M = 14$ $P_H(t)$ for $H2H = 5$	$r(4) > 0$ & $r(4) \in R_2$	$Q(S'(t+1), a'(t+1)) = [$ $\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 0 & 0 & 0 & 0 \\ 1.2 & 0 & 0 & 0 & 0 & 0 \\ -1.88 & -1.8 & 0 & 0 & 0 & 0 \\ -1.87 & -1.77 & 0 & 0 & 0.2 & 0 \\ -1.85 & -1.6 & 1.4 & 0.4 & 0 & 0 \\ 1.7 & -1.3 & 0 & 1.5 & 0 & 0.2 \\ 1.5 & 1.42 & 1.04 & 0 & 0.4 & 0 \\ 0.68 & -1.31 & -1.6 & 1.25 & 0 & -0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$]	$S'(t+1) = 12$ $P'_M(t+1)$ for $M2M = 12$ $P'_H(t+1)$ for $H2H = 3$
$S(t+1) = 12$ $P_M(t+1)$for $M2M = 12$ $P_H(t+1)$ for $H2H = 3$	$r(5) < 0$ & $r(5) \in R_2$	$Q(S'(t+2), a'(t+2)) = [$ $\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & -1.76 & -1.8 & 0.3 & 0 & 1.87 \\ 0.2 & 0 & -1.85 & -1.77 & 1.68 & 0.99 \\ 0 & -1.84 & -1.81 & 0 & 1.1 & 0 \\ 1.27 & 1.79 & 1.09 & 1.69 & 1.2 & 1.65 \\ 0 & 0.6 & 1.34 & 0.84 & 0 & 0.22 \\ -1.56 & -0.3 & 0.9 & 1.55 & 0 & 0.62 \\ 1.77 & 1.12 & 1.23 & 0 & 0.14 & 1.05 \\ -1.41 & -1.31 & 1.96 & -1.03 & 0 & 1.95 \\ 0 & 0 & -0.98 & 1.14 & 1.109 & 0.62 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$]	$S'(t+2) = 14$ $P'_M(t+2)$ for $M2M = 14$ $P'_H(t+2)$ for $H2H = 6$

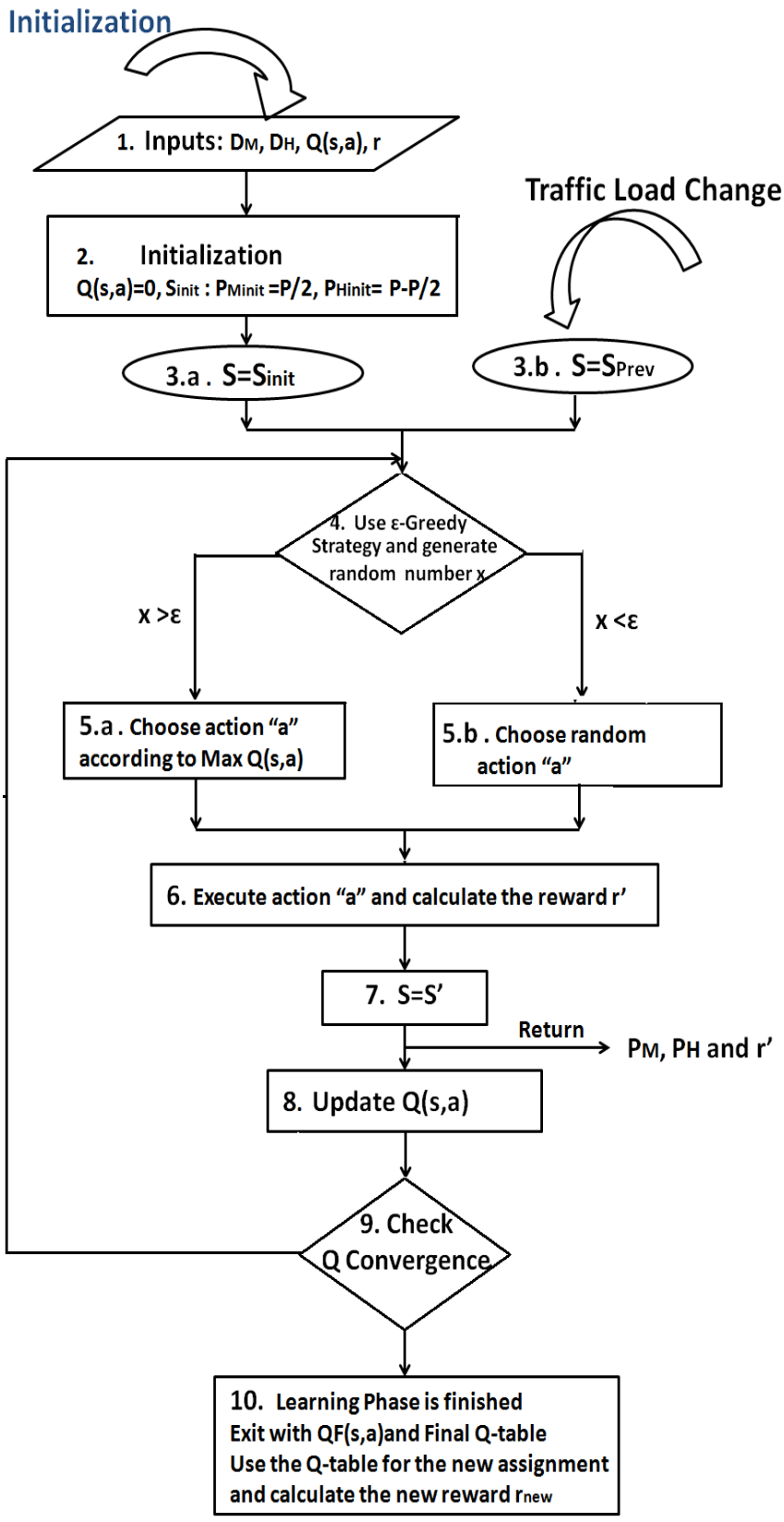

Fig. 5: The Proposed Q-learning Algorithm

4. PERFORMANCE EVALUATION RESULTS

The network has M M2M communication devices and N H2H communication devices which attempt to access the eNodeB in the uplink direction. The number of H2H communication devices is smaller than the number of M2M communication devices as shown in Fig. 6.

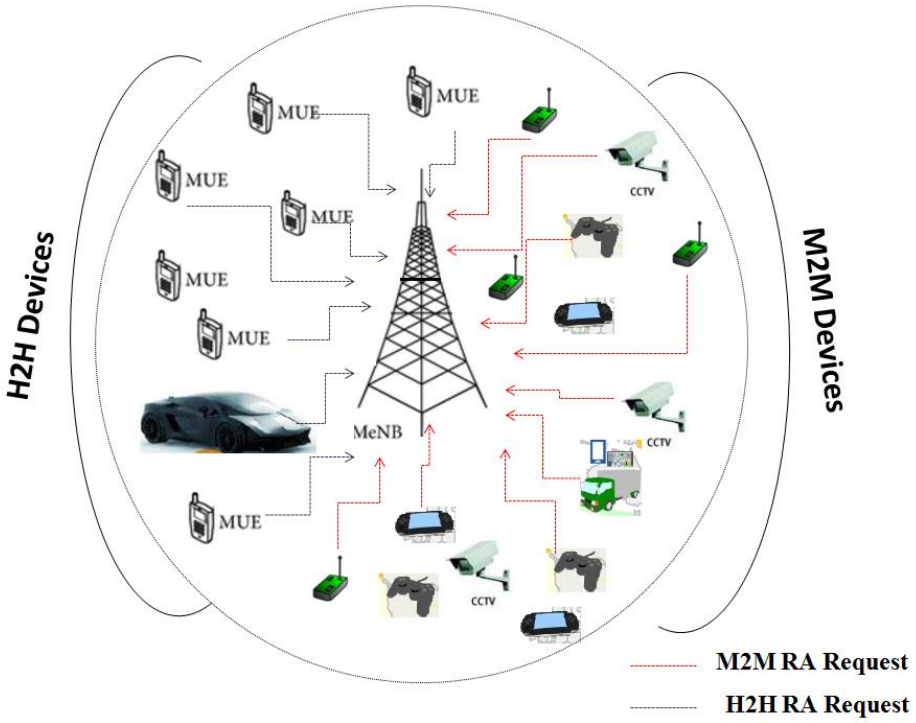

Fig. 6: The Heterogeneous Network Architecture with the coexistence of both H2H and M2M devices

In the simulations, M2M device uses the six RBs of the RA slot of LTE for the message transmission. The M2M device holds the preamble for a number of slots that is function of the message size. The message size is random from 100 to 1000 bytes and transmitted with a data rate of 2.4 Mbps. Using 16-QAM modulation, the number of occupied frames will be in the range of 1 to 3 frames. As for H2H devices, they use the preambles for connection establishment. Thus, it holds the preamble for a number of frames until setting up the communication link. This number randomly ranges from 5 to 10 frames according to the time needed for the setup (5 frames as a best-case network scenario and 10 frames as a worst-case network scenario).

To evaluate the performance of the proposed Q-learning scheme, we consider the following performance metrics:

1) *Access success probability*: the probability of successful RA procedure and the release of the preambles for both H2H and M2M communications.

2) *Total service time*: the total number of time slots taken from the arrival of the demand of both H2H and M2M devices till it is all served.

3) *Average number of re-transmissions*: the average of the re-transmission attempts done by the blocked devices until they are finally served.

We use Matlab to simulate the proposed model. The overall system is simulated as follows:

- Three arrays with size equal to the number of simulated framed are used. One array holds the served users who succeed to get preambles, the second holds the non-served users at a particular frame, and the third indicates the blocked users who failed to get failed to get preambles for five consequent frames. Initially, the arrays are initialized to zero and the number of available preambles is set to P=54.
- The simulation loop starts by generating at each frame the number of H2H and M2M arrivals and their requested demand. Two random number generators are used to determine the M2M communications message size and the period the H2H user holds the preamble. The first random generator is used to generate a random message size for M2M communications and is generated randomly from 100 to 1000 bytes, thus the period the M2M communications device holds the preamble is determined to be from 1 to 3 frames according to the message size. The second random generator is used for H2H communications and generated randomly from 2 to 8 frames according to the time needed for the connection setup (2 frames as a best-case network scenario and 8 frames as a worst-case network scenario).
- At the beginning of each frame, we check if the number of preambles are enough for the demand of both M2M and H2H. If they are enough, both types of traffic are assigned preambles and update the available preambles number. If they are not, we call the Q-learning function that has as inputs the available number of preambles, the demand of both traffics H2H and M2M in accordance with step 1 in the flow chart in Fig. 5 and the initialized action-value function $Q(s,a)$ in accordance with step 2. Then the Q-learning algorithm is started by setting

the state S in accordance with step 3 and it is either 3.a or 3.b according to the load type. The algorithm uses the generated random number to choose an action in step 4, that either goes to step 5-a or step 5-b. After step 5 the action is executed and both the reward and the new $Q(s,a)$ is calculated and the state is updated in accordance with steps 6 to 8. Then it returns the number of preambles assigned to H2H and M2M communications, the new $Q(s,a)$ and the reward r. After that the service arrays are updated. We nominally simulate a period of 2000 frames. The simulation parameters are presented in Table. 4.

Table. 4: Simulation Parameters

Parameter	Value
Network Type	Heterogeneous Network with H2H and M2M traffic
Sub-frame duration	1 ms
Simulation Duration	2000 frames
Number of Preambles	54
Number of RBs	6 RBs per frame
H2H traffic arrival rate	N
M2M traffic arrival rate	M
Number of RBs used by H2H	Random from 2 to 8 RBs
Number of RBs used by M2M	Random from 1 to 3 RBs based on the M2M message size
M2M message size	From 100 to 1000 bytes
M2M Modulation	16-QAM

We use two different scenarios in the calculations of the different performance matrices. At the first scenario, we have two rates of arrivals for both H2H and M2M at each frame as N and M , and they both attempt to access the network by requesting preambles for random access. This scenario is used for the calculation of both the access success probability, and the average number of re-transmission. At the second scenario, M2M and H2H traffic arrive at the beginning of simulations, which is used to calculate the total service time which is the time needed for all traffic to be served and leave the system.

4.1 Basic Performance

In this section, we evaluate the performance of the proposed Q-learning scheme by showing the success probability of both H2H and M2M communications. The success probability is presented in Fig. 7, and Fig. 8.

In Fig. 7, we vary H2H traffic rate N from 1 to 20 and keep M2M traffic rate M constant using four different values M ∈ {0, 1, 5, and 10} to study the effect of changing the M2M traffic on the H2H traffic.

While in Fig. 8, we vary M2M traffic rate M from 1 to 30 and keep H2H traffic rate N constant using three different values of N ∈ {1, 5, and 10} to study the effect of the H2H traffic on the M2M traffic.

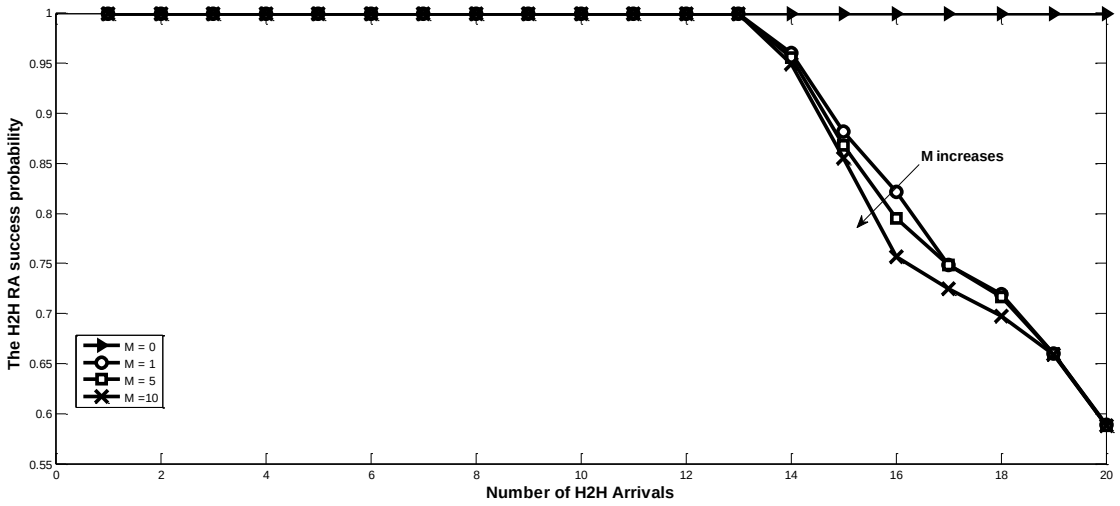

Fig. 7: The H2H RA success probability with the number of H2H arrivals for different number of M2M arrivals M at each frame

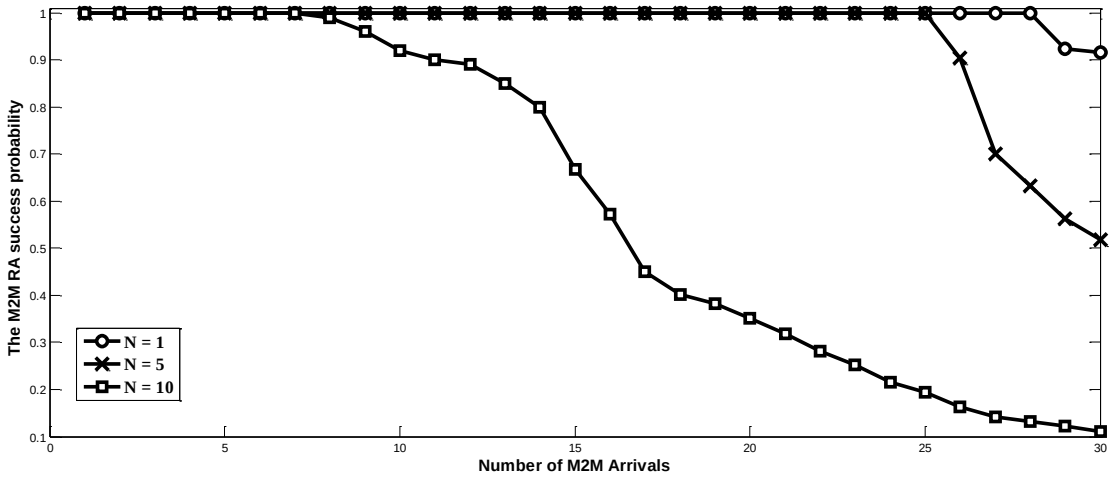

Fig. 8: The M2M RA success probability with the number of M2M arrivals for different number of H2H arrivals N at each frame

Results in Fig. 7 show the RA success probability for the H2H devices. We deduce that as M increases, the H2H success probability reduces, but the effect of changing M is small. For $M = 0$, the H2H success

probability is equal to 1. As *M* increases, the success probability reduces due to the effect of blocked traffic of the M2M demand that increases the whole demand and reduces the available preambles in following frames. This results in reduction of H2H success probability. Thus, the Q-learning starts to assign resources for H2H devices to ensure that they achieve their required blocking probability, and assigns the rest of resources for M2M devices. That improves the H2H RA success probability.

Fig. 8 shows the RA success probability for the M2M devices. It is clear that as N increases, the M2M random access attempt success probability reduces, as the Q-learning does the resources separation based on the condition of H2H blocking probability. Thus, when N increases, more resources are given to H2H devices to ensure the required QoS. This reduces the number of resources assigned to M2M devices which results in a reduction in the M2M success probability.

4.2 Comparisons with other schemes

In order to show the superior performance of the proposed approach, we compare it with two schemes; the Dynamic Access Class Barring (DACB) scheme and the Prioritized Random Access (PRA) scheme.

4.2.1 Access success probability

We compare the H2H success probability and M2M success probability of the three schemes with the increase of the H2H and M2M arrivals. This is shown in Fig. 9, Fig. 10, Fig. 11, and Fig. 12.

4.2.1.1 The effect of increasing the H2H arrivals

In this section, we study the effect of increasing the H2H traffic on the success probability of both H2H and M2M. We keep the number of M2M arrivals *M* constant and equal to 15 devices arrive in each frame, while the number of H2H arrivals *N* varies from 1 to 15 in each frame. This is shown in Fig. 9 and Fig. 10.

Fig. 9 shows the H2H success probability for the schemes; the Q-learning, PRA 25/15/10 which respectively reserve 25/15/10 preambles for H2H traffic and DACB. It shows that the proposed Q-learning approach gives better H2H success probability than DACB and the PRA approach when we preserve 10, 15 preambles for H2H devices. However, when we preserve 25 preambles for H2H devices,

PRA gives the best performance. Finally, DACB gives the lowest success probability. This is as DACB treats M2M and H2H devices on the same base in the random access. Thus, it does not care about any QOS requirements. It basically controls their random access process which results in the performance degradation for both types of communications.

Fig. 10 shows the M2M success probability for the three schemes; the Q-learning, PRA 15, and DACB. It can be seen that increasing the number of H2H devices makes the PRA scheme exhibit equal performance to the proposed Q-learning approach. However, with the increase of H2H arrivals, the proposed Q-learning gives better performance as in the PRA scheme when the number of H2H arrivals increases, their dedicated preambles increase as well, a reduction in the dedicated M2M preambles occurs thus resulting in a lower M2M success probability.

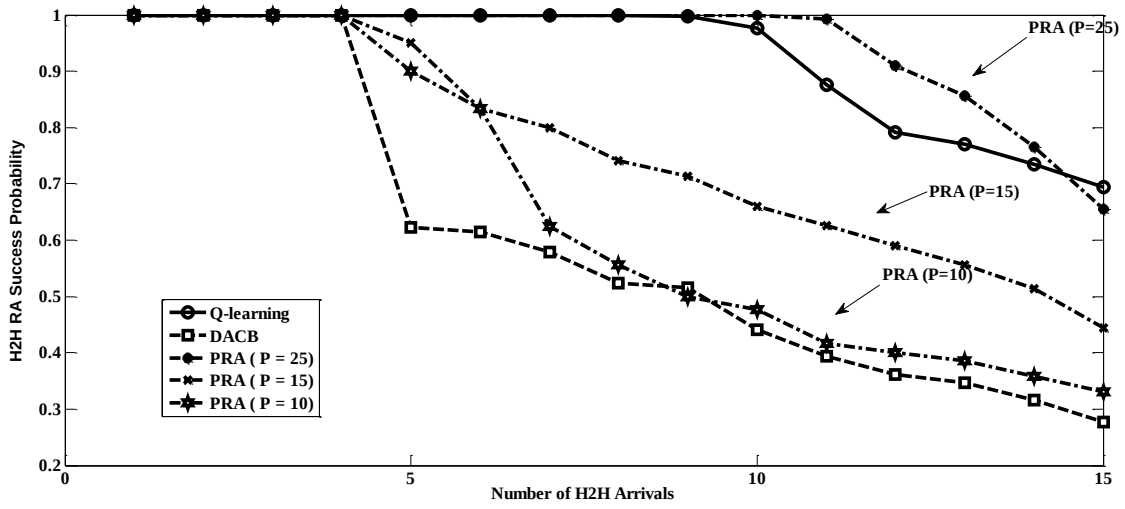

Fig. 9: H2H RA success probability using Q-learning, DACB and PRA with the number of H2H arrivals N at each frame

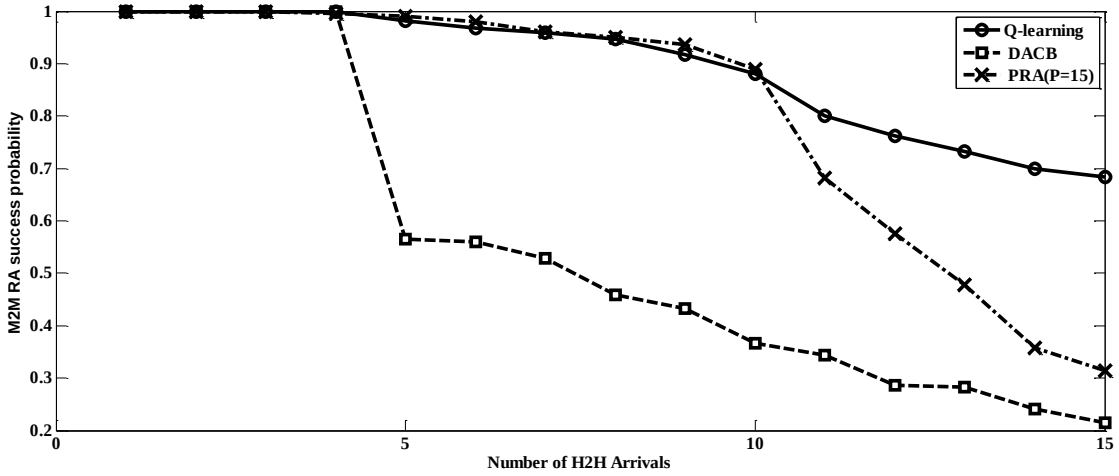

Fig. 10: M2M RA success probability using Q-learning, DACB and PRA 15 with the number of H2H arrivals N at each frame

4.2.1.2 The effect of increasing the M2M arrivals

In this section, we study the effect of increasing the M2M traffic on the success probability of both H2H and M2M. We keep the number of H2H arrivals N constant and equal to 5 devices arrive in each frame, while the number of M2M arrivals M varies from 1 to 20 in each frame. This is shown in Fig. 11 and Fig. 12.

Fig. 11 shows the H2H success probability for the schemes; the Q-learning, PRA 25 and DACB. It shows that PRA 25 gives a constant success probability equal to 1, while the proposed Q-learning approach gives a constant success probability equals to 1 for most of the simulation, however it decreases to 95% when number of M2M devices increases over 16 arrivals per frame. While DACB gives the lowest success probability.

Fig. 12 shows the M2M success probability for the three schemes; the Q-learning, PRA 15, and DACB. From Fig. 12, we deduce that as M2M arrivals increase, the success probability using the Q-learning approach is better than the other approaches. Also, as the M2M arrivals increase, DACB gives better performance than PRA. This is because in PRA, there are constant number of resources reserved for M2M traffic which makes the performance worse with the increase of M2M arrivals.

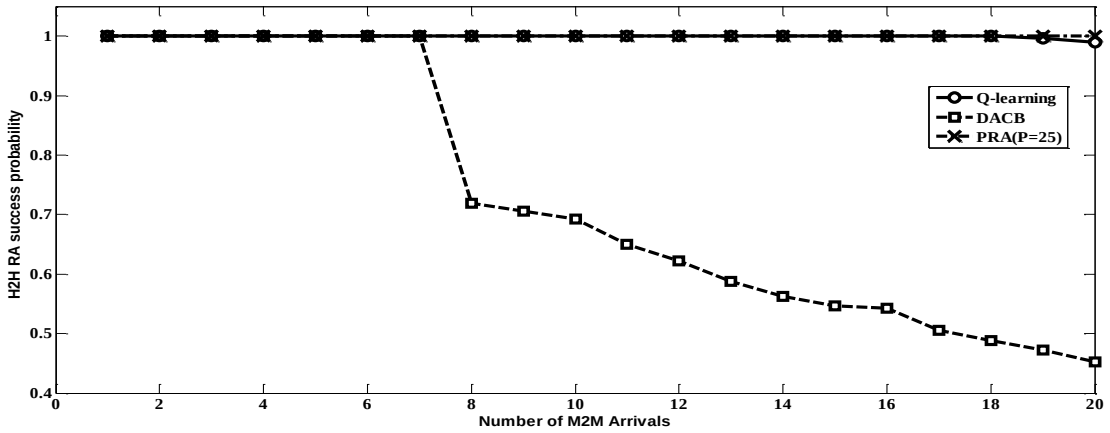

Fig. 11: H2H RA success probability using Q-learning, DACB and PRA 25 with the number of M2M arrivals M at each frame

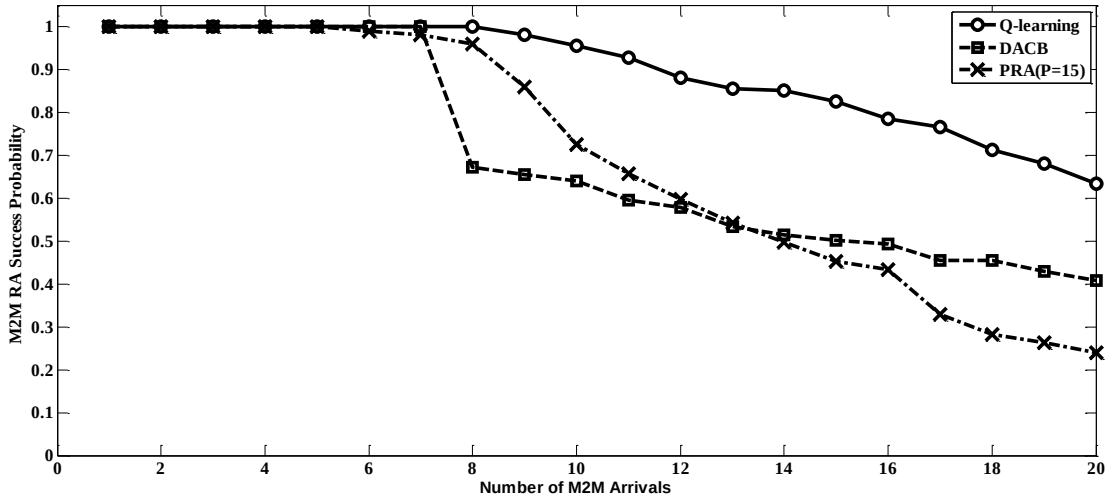

Fig. 12: M2M RA success probability using Q-learning, DACB and PRA 15 with the number of M2M arrivals M in each frame

4.2.2 Average Number of Re-transmissions

In this section, we compare between the three schemes based on the average number of re-transmissions for M2M traffic. This is defined as the average number of access attempts by the blocked devices till they are all served and leave the system.

In the simulations, we set the number of H2H arrivals to be equal to 10 devices arrive in each frame. While the M2M arrivals increase from 1 to 30 devices that request an access at each frame. To

demonstrate the proposed Q-learning scheme performance, we compare it with PRA 25, PRA 15 and DACB schemes.

Fig. 13 shows that the proposed Q-learning approach results in the number of re-transmissions being lower than both the PRA scheme and the DACB. The PRA with 15 preambles for the H2H traffic results in better performance compared with the DACB, but the PRA with 25 preambles for H2H traffic results in the highest number of re-transmissions.

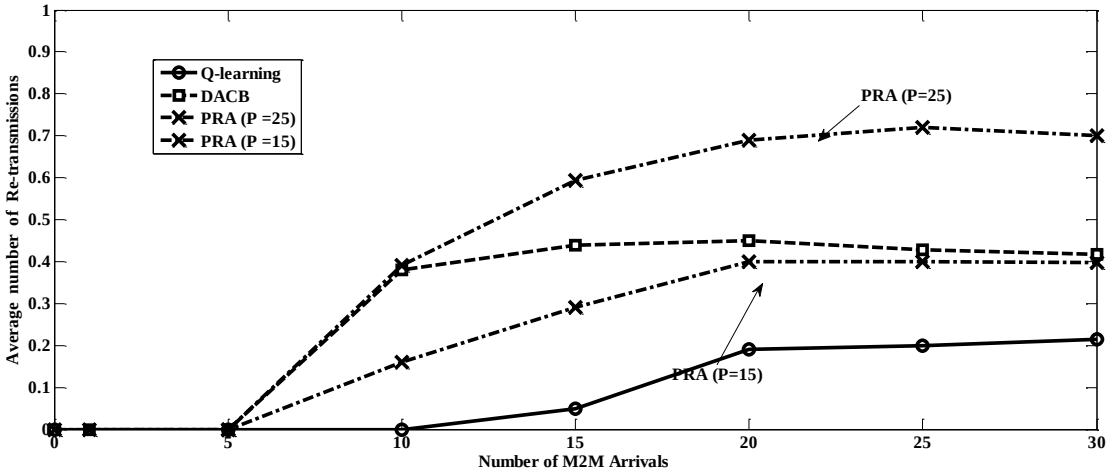

Fig. 13: Comparison between Q-learning, DACB and PRA for the average number of access re-transmissions with the number of M2M arrivals M in each frame

4.2.3 Total Service Time

Here, we compare based on total service time (TST). This is defined as the total number of time frames taken from the arrival of the whole traffic till it is all served. Thus, we calculate TST to serve the whole H2H and M2M traffic arrivals. In H2H communications, it will be from the arrival of the call request till the call establishment and for M2M communications, it is calculated from the arrival of the M2M RA request until the packet is transmitted in the uplink to eNB. The simulation is done for two parameters; the number of preambles P and the number of M2M arrivals M, as shown in Fig. 14 and Fig. 15.

In the simulations, both H2H and M2M traffics arrive at the beginning of simulation. We set the number of H2H arrivals N to 50 devices and the number of M2M arrivals M to 1000 devices. The comparison is done between the proposed Q-learning scheme, the PRA 15, and DACB. The first comparison is demonstrated in Fig. 14, we change the number of preambles P from 1 to 50 preambles and

study its effect on TST of the three schemes. In the second comparison demonstrated in Fig. 15, we set the number of preambles P to 15 preambles, while varying the number of M2M arrivals M from 1 to 3000 arrivals.

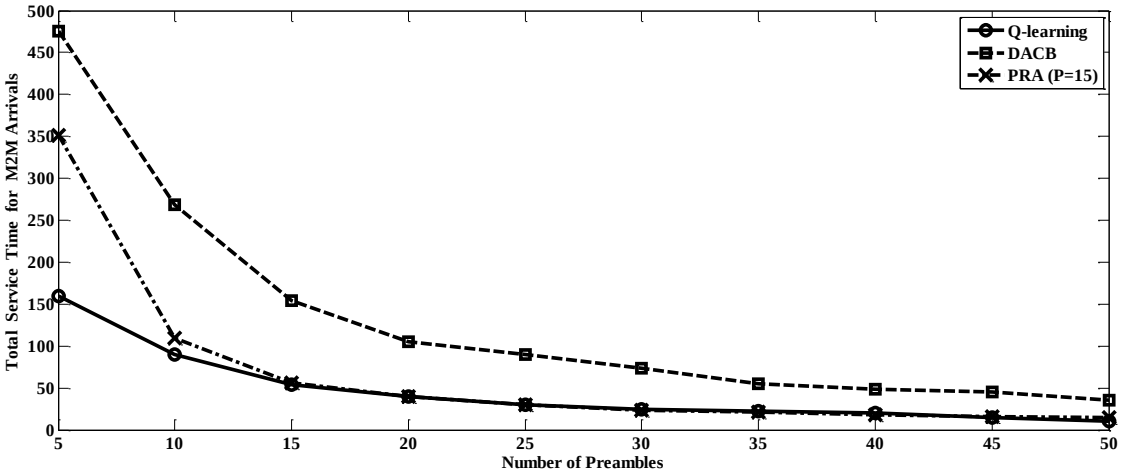

Fig. 14: Comparison between Q-learning, DACB and PRA 15 for M2M total service time against the number of preambles

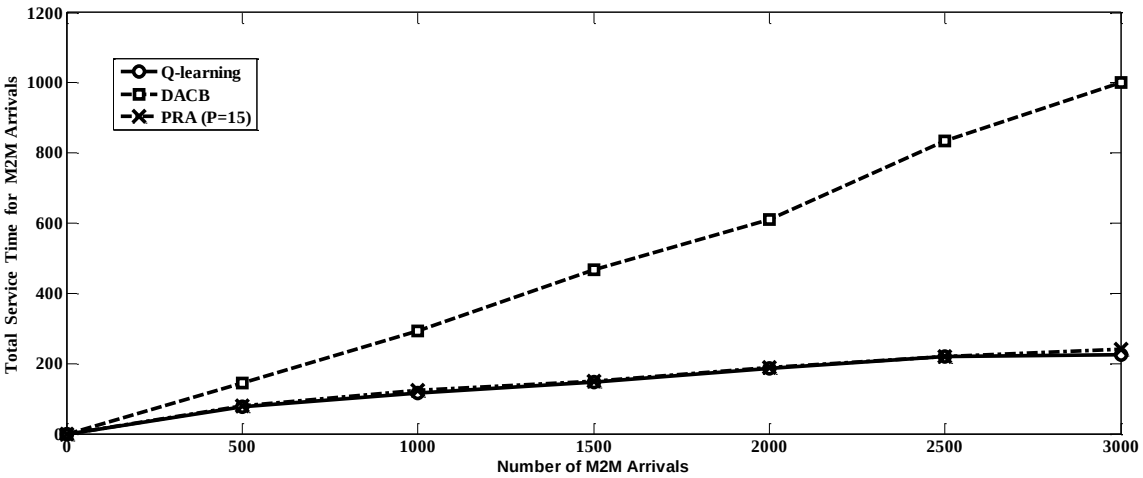

Fig. 15: Comparison between Q-learning, DACB and PRA for M2M total service time for number of preambles P =15, N =50, and M varies from 1 to 3000 devices arrived in each frame

In Fig. 14, it can be seen that TST decreases with increasing number of preambles as the number of RA opportunities increases. Thus, the TST needed to serve the whole traffic decreases. Also, it is clear our approach results in better reduction for TST than both the PRA and DACB schemes.

Fig. 15 shows that TST increases with the increasing of M2M devices, which is normal as long as the load increases, the time needed to serve it increases. It is shown that PRA scheme has TST near the proposed Q-learning approach TST. While DACB shows much larger TST than both the proposed approach and PRA scheme.

4.3 Convergence of the Proposed Q-Learning Scheme

In this section, we show that the Q-Learning scheme converges after spending some learning phase and then it can re-adapt if system conditions change.

Thus, if the convergence condition is achieved, the system stops learning and starts to use its Q-table and the final Q-matrix for preambles assignment.

We show the convergence in two different cases; first for the normal usage pattern of the network as and second when a sudden traffic load enters the network, and request random access to transmit their data to the eNB.

It can be shown in Fig. 16 that the proposed Q-learning approach experiences some changes at first that represent the learning phase, then it converges after a number of time frames when it finishes learning and it uses this learning for handling the preambles assignment. When an extra traffic arrives resulting in network load change, the algorithm is capable of capturing the change due to change in the reward value as discussed in section 3.4.1. We can observe that the proposed approach handles this extra load in smaller time than the time spent in the learning phase earlier. This is a result of the using its previously acquired knowledge, thus resulting in quick convergence to the proper values of the Q-matrix that is used later for the preambles assignment for both of types of traffic.

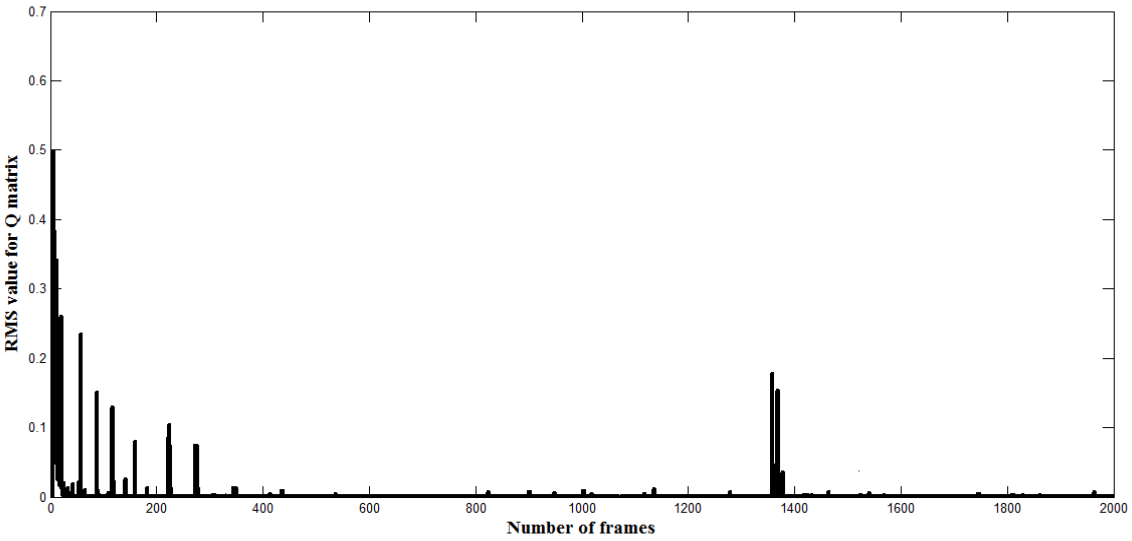

Fig. 16 : RMS for the Q-matrices for a sudden burst load arrives at time t=1400

5. CONCLUSIONS

The support of the M2M communication devices requires enhancements to the operation of the random access channel in LTE and LTE-A. In this paper, we proposed a new scheme to handle the potential congestion in the RACH using Q-learning algorithm. Simulation results show that taking into consideration the H2H devices' QoS requirements, the RA success probability of the M2M devices decreases marginally with the increasing number of H2H devices. While the effect of increasing the M2M devices on the H2H RA success probability is negligible. Also, simulations show that in case of a sudden traffic arrival, the scheme can adapt and handle dynamic changes in small time providing high RA success rate for both H2H and M2M devices.

To show the proposed approach performance, a comparison is made between our approach, DACB [14] and the PRA scheme [15]. We choose those schemes as both give higher success rate compared to fixed ACB and EAB. The comparisons show that the proposed Q-learning approach gives better performance, where it gives higher H2H RA success rate nearly equal to the one provided by PRA. It also results in higher success rates for M2M RA compared with both schemes. Results also show that the proposed approach gives lower TST and lower average number of re-transmissions than these provided by the other two schemes. This shows that the proposed Q-learning approach has good potential for being used to

provision services for both H2H and M2M communications in LTE-Advanced systems due to its excellent performance, adaptability, and simplicity.

REFERENCES

[1] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2M: From mobile to embedded Internet," IEEE Comm. Magazine, vol. 49, no. 4, pp. 36–43, Apr. 2011.

[2] A. Biral, M. Centenaro, A. Zanella, L. Vangelista, and M. Zorzi, "The challenges of M2M massive access in wireless cellular networks," Digital Communications and Networks, Volume 1, Issue 1, pp. 1-19, 2015.

[3] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh, "NB-IoT system for M2M communication," Wireless Communications and Networking Conference Workshops (WCNCW) IEEE, pp. 428-432, 2016.

[4] R. Ratasuk, N. Mangalvedhe, and A. Ghosh, "Overview of LTE enhancements for cellular IoT," Personal Indoor and Mobile Radio Communications (PIMRC) IEEE 26th Annual International Symposium on, pp. 2293-2297, 2015.

[5] Cisco VNI blog post, Cisco VNI Complete Forecast Update: "Key Trends Include Mobility, M2M, and Multimedia Content," 2015.

[6] A. Gotsis, A. Lioumpas, and A. Alexiou, "M2M scheduling LTE: Challenges and new perspectives," IEEE Vehicular Technology Magazine, vol. 7, no. 3, pp. 34 –39, Sep. 2012.

[7] 3GPP, "System improvements for machine-type communications," 3rd Generation Partnership Project (3GPP), TR 23.888 V11.0.0, Sep. 2012.

[8] A. Ksentini, Y. H .Aoul, and T. Taleb, "Cellular-based machine-to-machine: Overload control," IEEE Network, vol. 26, no. 6, pp. 54–60, Nov. 2012.

[9] G. Wang, X. Zhong, S. Mei, and J. Wang, "An adaptive medium access control mechanism for cellular based machine to machine (M2M) communication," in Proc. of IEEE Int'l Conf. on Wireless Information Technology and Systems (ICWITS), Hawaii, HI, Aug. 2010.

[10] 3GPP, "Study on RAN Improvements for machine-type communications," 3rd Generation Partnership Project (3GPP), TR 37.868 V11.0.0, Oct. 2011.

[11] 3GPP, "MTC simulation results with specific solutions," 3rd Generation Partnership Project (3GPP), TSG RAN WG2 #71, R2-104662, Aug. 2010.

[12] 3GPP, "MTC LTE Simulations," 3rd Generation Partnership Project (3GPP), TSG RAN WG2 #71, R2-104663, Aug. 2010.

[13] S. Duan, V. Shah-Mansouri, and V. W. S. Wong, "Dynamic access class barring for M2M communications in LTE networks," 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, pp. 4747-4752, 2013.

[14] 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," in IEEE Transactions on Vehicular Technology, vol. 63, no. 5, pp. 2467-2472, Jun 2014.

[15] S. S. Tsong, C. H. Chiu, Y. C. Cheng, and K. H. Kuo. "Self-adaptive persistent contention scheme for scheduling based Machine Type Communications in LTE system," 2012 International Conference on Selected Topics in Mobile and Wireless Networking, pp: 77-82, July 2012.

[16] H. Wu, C. Zhu, R. La, X. Liu, and Y. Zhang, "Fast adaptive S-ALOHA scheme," Vehicular Technology Conference (VTC-Fall), Quebec City, Canada, Sep. 2012.

[17] 3GPP, "Further Performance Evaluation of EAB Information Update Mechanisms," 3rd Generation Partnership Project (3GPP), RAN WG2#77, R2-120270, Intel, Germany, Feb. 2012.

[18] J. Park, and Y. Lim, "Adaptive Access Class Barring Method for Machine Generated Communications," Mobile Information Systems, vol. 2016, Article ID 6923542, 6 pages, 2016.

[19] Q. Du, W. Li, L. Liu, P. Ren, Y. Wang, and L. Sun, "Dynamic RACH Partition for Massive Access of Differentiated M2M Services," Sensors 16, no. 4: 455, 2016.

[20] J. Kim, and J. Lee,"Exploiting the Capture Effect to Enhance RACH Performance in Cellular-Based M2M Communications," Sensors 17, no. 10: 2169, 2017.

[21] A. David , V. Dario, F. Castro, and Miguel, "Dynamic RACH distribution for M2M massive access in LTE-A", 7th International Conference on the Network of the Future (NOF), pp 1-3, 2016.

[22] L. Oquendo, D. Paramo, V. Pla, and J. Bauset ,"Reinforcement Learning-Based ACB in LTE-A Networks for Handling Massive M2M and H2H Communications," IEEE ICC 2018.

[23] Y. Yu, T. Wang, and S. C. Luis, "Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks," IEEE ICC 2018.

[24] Y. Li, "E-band radios for LTE/LTE-Advanced mobile backhaul," 2010 Workshop on Integrated Nonlinear Microwave and Milli-meter-Wave Circuits (INMMIC), 2010.

[25] S. Choi, W. Lee, D. Kim, K.-J. Park, S. Choi, and K.-Y. Han "Automatic configuration of random access channel parameters in LTE systems," Wireless Days (WD) IFIP, pp. 1-6, Oct 2011.

[26] D. Kim, W. Kim, and S. An, "Adaptive random access preamble split in LTE," 9th International Wireless communications and Mobile Computing Conference (IWCMC), pp. 814-819, 2013.

[27] S. Batabyal, and S. S. Das, "Distance Dependent Call Blocking Probability, and Area Erlang Efficiency of Cellular Networks," IEEE 75th Vehicular Technology Conference (VTC Spring), pp. 1-5. 2012.

[28] L. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, volume 4, pp. 237-285, 1996.

[29] C. J. Watkins," Learning from Delayed Reward," Ph.D. Thesis, Cambridge University, 1989.

[30] C. J. Watkins," Q Learning, Machine Learning," Volume 8, pp. 279-292, 1992.

[31] R. Sutton and A. Barto," Reinforcement Learning: An Introduction," MIT Press, 1998.

[32] V. Mnih, K. Kavukcuoglu, and D. Silver, "Human-level control through deep reinforcement learning", *Nature*, Volume 518, pp. 529-533, 2015.