International Journal of Control, Automation and Systems VV(N) (YYYY) 1-3
http://dx.doi.org/10.1007/s12555-xxx-xxxx-xx

ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Multi-Agent Quality of Experience Control

Francesco Delli Priscoli, Alessandro Di Giorgio, Federico Lisi, Salvatore Monaco, Antonio Pietrabissa, Lorenzo Ricciardi Celsi*, and Vincenzo Suraci

Abstract: In the framework of the Future Internet, the aim of the Quality of Experience (QoE) Control functionalities is to track the personalized desired QoE level of the applications. The paper proposes to perform such a task by dynamically selecting the most appropriate Classes of Service (among the ones supported by the network), this selection being driven by a novel heuristic Multi-Agent Reinforcement Learning (MARL) algorithm. The paper shows that such an approach offers the opportunity to cope with some practical implementation problems: in particular, it allows to face the so-called "curse of dimensionality" of MARL algorithms, thus achieving satisfactory performance results even in the presence of several hundreds of Agents.

Keywords: Future Internet, Multi-Agent Reinforcement Learning, Quality of Experience, Quality of Service.

1. INTRODUCTION

A key Future Internet target is to allow applications to transparently, efficiently and flexibly exploit the available resources, with the aim of achieving a satisfaction level that meets the personalized users' needs and expectations. Such expectations could be expressed in terms of a properly defined Quality of Experience (QoE). In this respect, the International Telecommunication Union (ITU-T) defines QoE as *the overall acceptability of an application or service, as perceived subjectively by the end-user* [1]: this means that QoE could be regarded as a personalized function of plenty of parameters of heterogeneous nature and spanning all layers of the protocol stack (e.g., such parameters can be related to Quality of Service (QoS), security, mobility, contents, services, device characteristics, etc.).

Indeed, a large amount of research is ongoing in the field of *QoE Evaluation*, i.e., of the identification, on the one hand, of the personalized expected QoE level (*Target QoE*) for a given user availing her/himself of a given application in a given context (e.g., see [2] and [3] for voice and video applications, respectively), and, on the other hand, of the personalized functions for computing the *Perceived QoE*, including the monitorable Feedback Parameters which could serve as independent variables for these functions (e.g., see [4]). In particular, several works focus on studying the relation between QoE and network QoS parameters (e.g., see [5]).

Another QoE-related key research issue is that of *QoE Control*. Once a QoE Evaluator has assessed the personalized expected QoE level (Target QoE) and the personalized currently perceived QoE level (Perceived QoE), a QoE Controller should be in charge of making suitable *Control Decisions* aimed at reducing, as far as possible, the difference between the personalized Target and Perceived QoE levels. Section 2.1 discusses the nature of the QoE Controller decisions.

QoE Evaluation and QoE Control are also being widely studied in the context of several Future Internet related initiatives such as the MIUR PLATINO project [6] and the FP7 Future Internet PPP initiative (namely, [7] and [8]).

This paper focuses on QoE Control, whereas QoE Evaluation falls outside the scope of the paper. The interested readers are referred to [4] and [9] for an approach to QoE Evaluation that is fully consistent with this paper. Without claiming to present a ready-to-use solution, this paper provides some innovative hints that could ensure an efficient implementation of the QoE Controller.

In Section 2.1, the paper describes how Control Decisions can practically be implemented via the dynamic selection of predefined Classes of Service. In Section 2.2, the paper explains how such a dynamic selection can be performed in a model-independent way – in the authors' opinion, a control-based approach (as in [10] and [11]) relying on any Future Internet model is not practically

Manuscript received December 13, 2015.
This work was supported by the Italian Ministry of Education, Research and University, namely by the PLATINO PON project (www.progettoplatino.it), under Grant Agreement no. PON01_01007.
Francesco Delli Priscoli, Alessandro Di Giorgio, Federico Lisi, Salvatore Monaco, Antonio Pietrabissa, and Lorenzo Ricciardi Celsi are with the Department of Computer, Control and Management Engineering "Antonio Ruberti," University of Rome "La Sapienza," via Ariosto 25, 00185 Rome, Italy (email: {dellipriscoli, digiorgio, lisi, monaco, pietrabissa, ricciardicelsi}diag.uniroma1.it).
Vincenzo Suraci is with eCampus University, via Isimbardi 10, 22060 Novedrate, Italy (email: vincenzo.suraci@uniecampus.it).
* Corresponding author.

viable due to the sheer unpredictability of the involved variables [12] – thanks to the adoption of a multi-agent algorithm. A suitable algorithm was identified in a Multi-Agent Reinforcement Learning (MARL) technique, namely the *MARL Q-Learning* algorithm presented in [13] and [14] (classical multi-agent algorithms – see, e.g., [15, 16] – are based on state-space models). Then, the paper discusses the limitations of MARL Q-Learning with respect to practical implementation (Section 3.1) and how these limitations can be overcome by adopting the proposed heuristic algorithm, hereafter referred to as *H-MARL-Q algorithm* (Section 3.2). Finally, some numerical simulations showing the encouraging performance results of the proposed heuristic algorithm are presented in Section 4.2 with reference to a proof-of-concept scenario (described in Section 4.1) which does not claim to represent any real network.

2. THE QoE CONTROLLER

2.1. QoE Controller Architecture

The QoE Controller makes its decisions at discrete time instants t_k, hereafter referred to as *time steps*, occurring with a suitable time period T, whose duration depends on the considered environment (including technological processing constraints).

We assume that each in-progress *application instance* is handled by an *Agent i* and we define the personalized *QoE Error* at time t_k (indicated as $e_i(t_k)$), relevant to Agent i, as

$$e_i(t_k) = PQoE_i(t_k) - TQoE_i \tag{1}$$

where $PQoE_i(t_k)$ represents the Perceived QoE, i.e., the QoE currently perceived at time t_k by Agent i, and $TQoE_i$ represents the Target QoE, i.e., the personalized QoE which would satisfy the personalized Agent i requirements. So, if this QoE Error is positive, the in-progress application is said to be *overperforming*, since the QoE currently perceived by the Agent is greater than the desired one, whereas, if the QoE Error is negative, the in-progress application is said to be *underperforming*. Note that the presence of overperforming Agents might affect the system performance, since they may require an unnecessarily large amount of resources, which could cause, in turn, the underperformance of other Agents. The goal of the QoE Controller is to guarantee, at every time t_k, a nonnegative QoE Error for all Agents i (for $i = 1,\dots, N$), i.e., to avoid the occurrence of underperforming applications. Furthermore, if it is not possible to guarantee a nonnegative QoE Error for all Agents (e.g., due to insufficient network resources), the QoE Controller should reduce, as far as possible, the QoE Errors of the various Agents while guaranteeing *fairness* among them. Fairness basically consists in making sure that the QoE Errors experienced by the Agents are kept, as far as possible, close to one another.

As shown in Fig. 1, both the Perceived and the Target QoE should be computed by a suitable *QoE Evaluator* based on suitable *Feedback Parameters* resulting from the real-time monitoring of the network, as well as from direct or indirect feedbacks coming from users and/or applications. For a more detailed description of the way the QoE functionalities are embedded in the Future Internet architecture, see [17], [18] and [19].

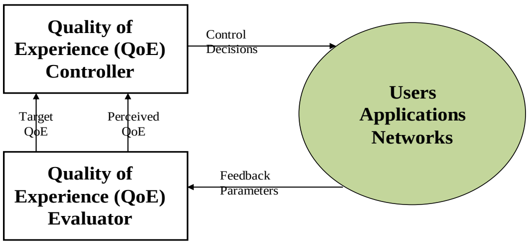
Quality of Experience (QoE) Controller
Control Decisions
Users Applications Networks
Target QoE
Perceived QoE
Quality of Experience (QoE) Evaluator
Feedback Parameters

Fig. 1. Sketch of the QoE architecture for the Future Internet.

In particular, a promising approach [4] is to relate the computation of the Perceived QoE to the *application type* (e.g. real-time HDTV streaming, distributed videoconferencing, File Transfer Protocol, etc.) of each in-progress application instance. Let M denote the total number of application types in the considered environment; let $m \in \{1, ..., M\}$ denote a generic application type; let $i(m)$ denote an Agent (i.e., an application instance) belonging to the m-th application type. Then, the Perceived QoE for Agent $i(m)$, denoted with $PQoE_{i(m)}(t_k)$, is computed as follows:

$$PQoE_{i(m)}(t_k) = g_m(\phi_m(t_k)), \tag{2}$$

where $\phi_m(t_k)$ represents a suitable set of Feedback Parameters for the m-th application type, computed up to time t_k, and g_m is a suitable function relating, for the m-th application type, the Feedback Parameters $\phi_m(t_k)$ with the Perceived QoE. Section 4 shows a simple implementation of (2); more advanced implementations can be found in [9].

A relevant drawback that could be immediately associated with such a method of evaluating the Perceived QoE for every Agent at each time step is the fact that an Agent can intentionally underreport its own Perceived QoE in order to increase the amount of network resources allocated to it. Such a problem falls within the area of mechanism design [20]. In this respect, in [21], Delli Priscoli et al. propose an interesting solution (compliant with the control algorithm discussed here) that allows to determine whether the Agent feedbacks are being fair or not. Such a solution is capable of ensuring an acceptable degree of robustness to possible episodes of dishonest Agent conduct.

The Target QoE, denoted with $TQoE_i$, can be derived from a suitable analysis of the available Feedback Parameters (e.g., by using unsupervised machine learning techniques), or it can simply correspond to a reference value which is assigned by the Telco operator, taking into account the commercial profile of the user.

In this paper, we propose a solution in which the distributed Agents associated to the application instances are embedded in properly selected network nodes (e.g., in the mobile user terminals): the Agents are in charge of the monitoring and actuation functionalities whereas the control functionalities are centralized in the QoE Controller.

In particular, whenever a new application instance is born, the associated Agent i is in charge of evaluating the personalized Target QoE $TQoE_i$ (which remains unchanged for the whole lifetime of the application instance), of computing its own personalized Perceived QoE $PQoE_i(t_k)$ and of communicating the monitored values to the QoE Controller. As a result, at each time t_k, the QoE Controller, based on the received values for $TQoE_i$ and $PQoE_i(t_j)$ up to time t_k ($i = 1,\ldots, N$; $j = 0, 1,\ldots, k$), has to choose the most appropriate action $a_i(t_k)$ (for $i = 1,\ldots, N$) which the Agent i should enforce at time t_k, i.e., the most appropriate *joint action* $(a_1(t_k), a_2(t_k),\ldots, a_N(t_k))$ which the N Agents should enforce at time t_k. At each time t_k, the chosen joint action is broadcast to the N Agents: then, the i-th Agent has to enforce the corresponding action $a_i(t_k)$.

Note that the proposed arrangement is based on the presence of a centralized entity (i.e., the QoE Controller), collecting the Agents' observations, which runs the MARL algorithm and broadcasts the resulting Control Decisions to the Agents. Therefore, any direct signal exchange among the Agents is avoided, thus limiting the overall signalling overhead.

The QoE Controller outputs, i.e, the joint action chosen by the QoE Controller, may include for each Agent the choice of QoS Reference Values (e.g., the expected priority level, the tolerated transfer delay range, the minimum throughput to be guaranteed, the tolerated packet loss range, the tolerated dropping frequency range, etc.), of Security Reference Values (e.g., the expected encryption level, the expected security level of the routing path computed by introducing appropriate metrics, etc.), and of Content/Service Reference Values (e.g., the expected content/service mix, etc.).

The QoE Controller has to dynamically select, for each in-progress application instance, the most appropriate Reference Values which should actually drive, thanks to suitable underlying network procedures (which are outside the scope of this paper), the Perceived QoE as close as possible to the Target QoE (for further details, see [12] where the above-mentioned Reference Values are referred to as *Driving Parameters*). However, since the control action has a large number of degrees of freedom, the exploration of the solution space may take a large amount of time, thus making the task of the QoE Controller excessively complex. A simpler (yet less fine-grained) control task arises if the management of the underlying networks is arranged into *Classes of Service* (CoS), as described in [22].

In this paper, we assume that each CoS is associated with a predefined set of QoS Reference Values. Nevertheless, the proposed approach can be applied even in the case when each CoS is associated with a set of Reference Values that are not necessarily related to QoS issues only, but also, for instance, to Security parameters, and/or to Content/Service characteristics, etc. Let S indicate the total number of CoSs and let $a_i(t_k) \in \{c_1, c_2, \ldots, c_S\}$ indicate the action performed by the i-th Agent (i.e., the CoS chosen by the i-th Agent) at the time instant t_k.

In current telecommunication networks, a static CoS assignment policy is adopted: each application instance is given a CoS for its entire lifetime; the CoS associated to a given application instance should be the one whose QoS Reference Values satisfy "on the average" the application requirements. Nevertheless, it is evident that such a static association does not take into account either *personalized* application requirements or contingent situations taking place in the telecommunication networks, such as congestion events. So, a static CoS assignment may generally lead to poor performance in terms of the personalized QoE perceived by each user. Hence, this paper considers dynamic CoS-to-application assignment as the methodological means to accomplish the above-mentioned goals in terms of QoE Error reduction and fairness. This means that, at each time instant t_k, the QoE Controller has to decide, in real time, which is the most appropriate CoS to be associated with each in-progress application instance (e.g., if the Agents are embedded in mobile user terminals, the QoE Controller decisions can be implemented by inserting the selected CoS identifier in the header of the packets transmitted by the terminals). Up to the authors' knowledge, apart from [18], [19], [12] and [23], such a dynamic assignment approach has never been investigated so far.

Indeed, meeting the Target QoE for the in-progress applications, in conjunction with an efficient exploitation of the available bandwidth, could be a rather challenging issue, especially in wireless networks with limited bandwidth resources. In this respect, optimal adaptive control strategies could be key factors to cope with such an issue. Moreover, due to the data-intensive nature of multimedia streaming services as well as due to the increasingly demanding requirements in terms of QoS/QoE, Reinforcement Learning based algorithms are being used more and more in telecommunication networks, as long as they prove to be computationally efficient and sufficiently scalable [24].

2.2. The MARL-Q Algorithm for the QoE Controller

This paper focuses on the problem of designing the QoE Controller algorithm. It should be evident that, in order to solve this problem by means of traditional model-based control techniques, the QoE Controller should know – or at least estimate – the correlation between its decisions (namely, the selected QoE Controller outputs) and the Perceived QoE. However, no model of the very complex plant regulated by the QoE Controller (namely, the plant receiving the QoE Controller outputs in input and producing the Perceived QoE as its output) can be assumed, since it depends on plenty of hardly predictable factors (such as traffic characteristics of the ongoing applications, network topologies, resource management algorithms, QoE Evaluation methods and so on).

In light of the above, the QoE Controller decision strategy must be learned online by *trial and error*. This is why we propose that the QoE Controller makes use of a model-free MARL algorithm in order to evaluate, at each time step t_k, the *joint policy* $\boldsymbol{\pi}(a_1(t_k), a_2(t_k), \ldots, a_N(t_k)) = \boldsymbol{\pi}(a_1, a_2, \ldots, a_N)$ which, once enforced by the Agents, tracks the discussed goals in terms of QoE Error. The proposed MARL algorithm works on the basis of the observation of a *joint reward* $r(t_{k+1}, a_1(t_k), a_2(t_k), \ldots, a_N(t_k)) = r(t_{k+1}, a_1, a_2, \ldots, a_N)$, i.e., of the numerical reward (the same for all the N Agents) which is received by each Agent at time t_{k+1} as a consequence of the enforcement, at time t_k, of the joint policy $\boldsymbol{\pi}(a_1, a_2, \ldots, a_N)$. The MARL algorithm in question is aimed at maximizing the *long-run return* $R(\boldsymbol{\pi})$, namely at maximizing the expected discounted return:

$$R(\boldsymbol{\pi}) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^{k} \cdot r(t_{k+1}, a_1, a_2, \ldots, a_N)\right\} \qquad (3)$$

where $\gamma \in [0,1)$ is the *discount rate*, which weighs immediate versus delayed rewards, and $E_{\pi}\{\cdot\}$ denotes the expected value under policy $\boldsymbol{\pi}$.

In order to set up a MARL problem, we have to select the state space, the action spaces and the reward function.

- We consider a static game, i.e., a game with only a single state: such an assumption, on the one hand, is not limiting in our context, and, on the other hand, greatly reduces the computational complexity which in MARL is exponential in the number of state and action variables.
- Following the discussion on dynamic CoS assignment, the action set A_i of Agent i coincides with the set of CoSs, i.e., $A_i = \{c_1, c_2, \ldots, c_S\}$, $i = 1, \ldots, N$. In other words, action $a_i(t_k)$, performed by Agent i at time t_k, can be equal to either c_1, or c_2, …, or c_S. The cardinality of the *joint action space* $\mathbf{A} = A_1 \times \ldots \times A_N$ is equal to $|A_1| \cdot |A_2| \cdot \ldots \cdot |A_N| = S^N$.
- The function expressing the joint reward $r(t_{k+1}, a_1, a_2, \ldots, a_N)$ should be consistent with the discussed goals in terms of QoE Error; in this respect, each candidate joint reward should be a non-increasing function of the N error values $|e_i(t_k)|$ (for $i = 1, \ldots, N$). In Section 4.1, the choice of suitable joint reward functions will be discussed.

In particular, we propose to apply the Multi-Agent Q-Learning algorithm [13] (hereinafter referred to as *MARL-Q algorithm*) which is proved to converge to an optimal policy $\boldsymbol{\pi}^*(a_1, a_2, \ldots, a_N)$, i.e., to a policy which maximizes the expected discounted long-run return $R(\boldsymbol{\pi})$. The algorithm is the multi-agent extension of the well-known (single-agent) Q-Learning algorithm ([25]), already succesfully applied to QoE/QoS control in communication networks ([26], [27]).

The MARL-Q algorithm relies on the estimation of the optimal action-value function $Q(s, a_1, a_2, \ldots, a_N)$, defined as the expected return of the system when it starts from state s, takes the joint action $(a_1, a_2, \ldots, a_N)$, and follows policy $\boldsymbol{\pi}$ thereafter. In the previously defined centralized context, at each time step t_k, this algorithm (i) evaluates a joint policy $\boldsymbol{\pi}(a_1, a_2, \ldots, a_N)$ – which sums up the behaviour of all the N Agents and is initialized arbitrarily – and (ii) improves such a policy by making it ε-greedy with respect to the current action-value function [28], thus yielding a better joint policy $\boldsymbol{\pi}'$ to be evaluated and improved at the next iteration.

In detail, the policy evaluation step (i) is performed by the MARL-Q algorithm by updating the action-value function $Q(t_k, a_1, a_2, \ldots, a_N)$ according to the following update rule:

$$\begin{aligned} Q(t_k, a_1, a_2, \ldots, a_N) = {} & (1 - \alpha(t_k))\, Q(t_{k-1}, a_1, a_2, \ldots, a_N) + \\ & + \alpha(t_k)\, [r(t_k, a_1, a_2, \ldots, a_N) + \\ & + \gamma \max_{a_1 \in A_1, \ldots, a_N \in A_N} Q(t_{k-1}, a_1, a_2, \ldots, a_N)], \end{aligned} \qquad (4)$$

where $\gamma \in [0,1)$ is the discount rate and $\alpha(t_k)$ is a sequence of *learning rates*, which are key parameters that should satisfy the standard stochastic approximation conditions for convergence [29]. The argument t_k denotes the value of the action-value function computed at time t_k, whereas the argument s is omitted since we are considering a single state problem.

The policy improvement step (ii) consists in performing, with probability equal to ε, a random joint action $(a_1', a_2', \ldots, a_N')$ and, with probability equal to 1 – ε, the following greedy joint action $(a_1', a_2', \ldots, a_N')$:

$$(a_1', a_2', \ldots, a_N') = \arg \max_{a_1 \in A_1, \ldots, a_N \in A_N} Q(t_k, a_1, a_2, \ldots, a_N) \qquad (5)$$

The parameter $\varepsilon \in (0,1)$ is the *exploration rate*. A large value of ε guarantees that different policies with respect to the current best one are explored, and thus avoids that the QoE Controller remains stuck in a local minimum (*exploration*); on the other hand, a small value of ε lets the QoE Controller choose the best action based on the current estimate of the action-value function (*exploitation*).

So, at each time step t_k, the centralized QoE Controller – based on the Perceived QoE values $PQoE_i(t_k)$ ($i = 1, \ldots, N$) transmitted by the Agents at time t_k, and on the knowledge of the Target QoE values $TQoE_i$ ($i = 1, \ldots, N$) transmitted by the Agents at the time of their birth – performs the following tasks until the optimal action-value function Q^* (and the optimal policy $\boldsymbol{\pi}^*$) is found:

T1) it updates the action-value function Q according to (4);

T2) it determines the joint action $(a_1', a_2', \ldots, a_N')$ in a random way with probability equal to ε, and according to (5) with probability equal to 1 – ε;

T3) it broadcasts the chosen joint action $(a_1', a_2', \ldots, a_N')$ to all Agents so that Agent i consequently enforces action a_i';

T4) it computes the corresponding joint reward $r(t_{k+1}, a_1', a_2', \ldots, a_N')$ according to the selected reward function which should include, as independent variables, the Perceived QoE values $PQoE_i(t_k)$ ($i = 1$,

…, N) and the Target QoE values $TQoE_i$ ($i = 1, …, N$).

The algorithm converges under a generic initial policy.

By varying the learning rates, the exploration rate and the discount rate, the convergence speed of the algorithm and the quality of the solution significantly change; the parameters used in the simulations reported in Section 4 have been tuned by running the simulations several times.

3. PROPOSED HEURISTIC MARL-Q BASED (H-MARL-Q) ALGORITHM

3.1 Limitations of MARL-Q

The analysis of the contents of the previous section offers us the opportunity to discuss the following issues.

- The main challenge arisen in MARL is the so-called *curse of dimensionality* [14]: in fact, as Reinforcement Learning algorithms (such as Q-Learning) estimate values for each possible state or state-action pair, the computational complexity of MARL is exponential in the number of state and action variables and, therefore, in the number of Agents; in addition, the Agents' rewards are correlated and then they cannot be maximized independently of one another. The *runtime* of the MARL-Q algorithm (i.e., the time the algorithm needs to perform the specific task it has been designed for) directly depends on the cardinality S^N of the joint action space. As a matter of fact, at each time step, the *max* operator in (5) has to consider S^N values; in this respect, it is particularly important to note that, in a Future Internet framework where the QoE Controller should be able to handle even thousands of Agents and dozens of CoSs, S^N would become a really huge value. For this reason, the task of implementing the dynamic CoS assignment according to the MARL-Q algorithm discussed in the previous section is inherently complex from a computational point of view and, as a result, it is extremely runtime-consuming. Such a relevant issue claims for a reasonable reduction of the size of the joint action space (and, hence, of the computational effort of the learning algorithm).
- The issue of the nonstationarity of multi-agent learning arises too, since all Agents in the system are simultaneously learning: each Agent is faced with a moving-target learning problem and consequently the best policy changes as the other Agents' policies change. In this respect, the exploration strategy is crucial for the efficiency of MARL algorithms. Agents explore to obtain information not only about the environment, but also about the other Agents, for the purpose of implicitly building models of these Agents. In other words, the need for coordination stems from the fact that the effect of any Agent's action on the environment depends also on the actions taken by the other Agents. Nonetheless, too much exploration should be avoided, as it may destabilize the learning dynamics of the other Agents.

In order to address the above-mentioned limitations, this paper presents an innovative heuristic algorithm, hereafter referred to as *H-MARL-Q algorithm* and derived from the MARL-Q algorithm described in Section 2.2. Such a heuristic algorithm, in comparison with the latter, considerably reduces the joint action space, thus significantly accelerating the task of dynamic CoS mapping, without teasing out an excessive amount of exploratory and information-gathering actions (hence, preserving an acceptable level of environment exploration). As shown in Section 4, the proposed H-MARL-Q algorithm has also turned out to be successful in addressing the issue of the algorithm *scalability*, yielding satisfactory results even when the number of Agents is counted in the order of thousands (as it will happen in the upcoming Internet of Things era).

3.2 H-MARL-Q Algorithm Description

The H-MARL-Q algorithm only considers a suitably selected subset of the joint action space, reasonably yielding an approximate solution to the dynamic CoS assignment problem presented in Section 2.

Basically, at each time step, the entire joint action space contains plenty of joint actions which have very few possibilities of being the best ones (i.e., the ones which meet the *max* operator in (5)). Unfortunately, such joint actions cannot be identified and discarded *a-priori*, because we do not have any *a-priori* knowledge of the environment; nevertheless, such actions can be identified and removed by carrying out a preliminary analysis of the Agents' dynamic behaviour in a simpler emulated environment. So, the basic underlying idea of the H-MARL-Q algorithm is to perform the following two steps.

Step (a): This step, referred to as *Identification of the Reduced Joint Action Space*, is performed by the QoE Controller *una tantum*, every time a new Agent is born, in order to identify, through the emulation of suitable test environments, an appropriate *Reduced Joint Action Space*.

Step (b): This step, referred to as *Identification of the Suboptimal Joint Action*, is performed, in real time, by the QoE Controller at each time step t_k, in order to identify the joint action ($a_1, a_2, …, a_N$) to be performed at time t_k on the basis of real-time observations of the environment and considering the Reduced Joint Action Space identified in step (a) (and not the entire joint action space $\boldsymbol{A}$). This yields a suboptimal joint policy which constitutes a satisfactory approximate solution to the considered problem.

3.2.1 H-MARL-Q Algorithm Description: Step (a)

Whenever a new Agent, say agent N, is born (i.e., a new application instance is launched), say at time t_k, in a *real* environment in which $N – 1$ Agents i (for $i = 1, 2, …, N – 1$) are already active, the new Agent notifies its existence to the QoE Controller together with its own personalized QoE requirements expressed in terms of Target QoE ($TQoE_N$). Then, the QoE Controller *emulates* the dynamic behaviour of the system in $N – 1$ *two-player test games*, each one involving two Agents: (i) the new Agent N and

(ii) each of the already active Agents i ($i = 1,\ldots, N - 1$). These two-player test games are played in emulated test environments which should reproduce only some key features of the real environment.

Let $[i, j]$ denote the two-player test game involving Agents i and j. In each two-player test game $[i, j]$ the optimal policy $\boldsymbol{\pi}^*(a_i, a_j)$ is obtained by applying the MARL-Q algorithm described in the previous section (clearly, in this case, the number of Agents N appearing in (4) and (5) is equal to two). The optimal policy identifies a pair of deterministic actions (a_i^*, a_j^*) where a_i^* and a_j^* represent the optimal CoS choices that the Agents i and j, respectively, should enforce.

It should be clear that, since the cardinality of the joint action space of each test environment is equal to S^2, the computational complexity of the MARL-Q algorithm is limited, i.e., the algorithm converges to the optimal policy in a limited runtime as shown through real tests in Section 4.

After step (a), at any time t_k at which N Agents are active, the QoE Controller stores $N(N - 1)/2$ optimal action couples:

$$(a_i^*, a_j^*) \quad \text{with } i = 1,\ldots, N, \quad j = 1,\ldots, N, \quad i \neq j. \qquad (6)$$

These couples are used in order to identify a Reduced Joint Action Space containing a reasonable subset of the entire joint action space $\boldsymbol{A}$.

Let $a_i^*[i, j]$ and $a_j^*[i, j]$ denote the optimal action for the i-th Agent and the j-th Agent, respectively, resulting from the two-player test game $[i, j]$. We assume that such a Reduced Joint Action Space consists of the union of N Action Subspaces, where the i-th Action Subspace is associated to the i-th Agent (the sub-tables within the borders in bold in the table below represent such Action Subspaces). Each Action Subspace includes S candidate joint actions (i.e., the rows of each sub-table). The i-th Action Subspace is built by only considering the two-player test games involving the i-th Agent. In particular, each of the S candidate joint actions of the i-th Action Subspace is obtained as follows: for each Agent j, with $j \neq i$, the optimal action $a_j^*[i, j]$ that such an Agent would perform in the two-player test game $[i, j]$ is taken into account, whilst for the i-th Agent all the S possible actions of the A_i set are spanned (each one being considered in a different candidate joint action of the Action Subspace).

By so doing, the Reduced Joint Action Space includes SN candidate joint actions: this certainly entails a drastic reduction with respect to the S^N joint actions that would appear in the entire joint action space $\boldsymbol{A}$.

For instance, if, at the considered time step, $N = 4$ (i.e., the Agents 1, 2, 3 and 4 are active) and $S = 3$ (i.e., the action a_i that Agent i, for $i = 1, 2, 3, 4$, can perform corresponds to the selection of one of the three CoSs c_1, c_2, c_3), each of the $SN = 12$ rows of the table below provides one of the 12 candidate joint actions (in particular, the sub-tables included within the borders in bold identify the $N = 4$ Action Subspaces), while each of the four columns of the table identifies the single actions that can be taken by Agents 1, 2, 3 and 4, respectively, in the overall Reduced Joint Action Space.

Moreover, every time a new Agent, say agent N, dies (i.e., an in-progress application terminates), the Reduced Joint Action Space is updated by eliminating the actions involving Agent N. For instance, referring to the example reported in the table below, if Agent 4 dies, the three joint actions corresponding to the three last rows are removed (i.e., the Action Subspace corresponding to Agent 4 is removed), and all the actions corresponding to the last column are removed, too.

Table 1. Representation of the Reduced Joint Action Space for $N = 4$ and $S = 3$. The columns of the table identify the different Agents, the rows represent the different candidate joint actions, and the sub-tables within the borders in bold represent the so-called Action Subspaces.

c_1	a_2^* [1,2]	a_3^* [1,3]	a_4^* [1,4]
c_2	a_2^* [1,2]	a_3^* [1,3]	a_4^* [1,4]
c_3	a_2^* [1,2]	a_3^* [1,3]	a_4^* [1,4]
a_1^*[1,2]	c_1	a_3^* [2,3]	a_4^* [2,4]
a_1^*[1,2]	c_2	a_3^* [2,3]	a_4^* [2,4]
a_1^*[1,2]	c_3	a_3^* [2,3]	a_4^* [2,4]
a_1^* [1,3]	a_2^* [2,3]	c_1	a_4^* [3,4]
a_1^* [1,3]	a_2^* [2,3]	c_2	a_4^* [3,4]
a_1^* [1,3]	a_2^* [2,3]	c_3	a_4^* [3,4]
a_1^* [1,4]	a_2^* [2,4]	a_3^* [3,4]	c_1
a_1^* [1,4]	a_2^* [2,4]	a_3^* [3,4]	c_2
a_1^* [1,4]	a_2^* [2,4]	a_3^* [3,4]	c_3

3.2.2 H-MARL-Q Algorithm Description: Step (b)

Step (b) of the H-MARL-Q algorithm is performed on the basis of the MARL-Q algorithm presented in Section 2.2 and is applied to the Reduced Joint Action Space identified in step (a). So, in step (b), the QoE Controller has to perform the tasks T1, T2, T3, and T4 described in Section 2.2, with the fundamental difference that, when performing tasks T1 and T2, the Reduced Joint Action Space (having cardinality SN), instead of the entire Joint Action Space (having cardinality S^N), is considered. Since N can be in the order of thousands, it is evident that the proposed approach drastically reduces the required computing power.

4. H-MARL-Q ALGORITHM SIMULATIONS

4.1 Simulation Scenario

This section presents numerical simulations, carried out using MATLAB®, with reference to a simple simulation scenario which does not claim to represent any real network. The presented simulations are just aimed at providing a proof-of-concept of the proposed algorithm in order to highlight its potentialities and criticalities.

We assume the presence of S = 3 different CoSs (e.g., "guaranteed," "premium" and "best effort" services) and M = 3 different application types (i.e., real-time HDTV streaming, distributed videoconferencing and simple File Transfer Protocol). The static CoS assignment policy determines a static association among application types and CoSs (i.e., an application instance belonging to a given application type is assigned the corresponding CoS for its entire lifetime), whereas in the dynamic CoS assignment case, at each time step t_k, an application instance can be assigned any CoS (regardless of the application type) according to the proposed H-MARL-Q algorithm.

We assume that, during our simulations, N Agents are active, each one being involved in an application instance. Such an application instance may belong to one of the three considered application types and is characterized by an average offered transmission bitrate b_i randomly selected in the set {0.6, 1.2, 2} and by a personalized Target QoE $TQoE_i$ (for i = 1, ..., N) randomly selected in the set {0.7, 0.8, 0.9}.

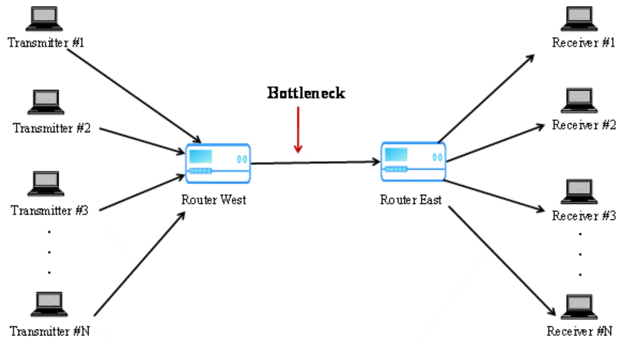

Fig. 2. Dumbbell network topology.

The simulated network has a dumbbell network topology, as shown in Fig. 2, where each of the N transmitters corresponds to one of the N considered Agents. Router West implements a Weighted Fair Queueing (WFQ) scheduler for handling the traffic to be transmitted over the bottleneck link. The related WFQ vector [30] is assumed to be (0.5, 0.3, 0.2), where the i-th element is the weight assigned to the i-th CoS (higher weight means higher priority). The bottleneck link is characterized by an available link capacity B_{link} computed as:

$$B_{link} = \omega \sum_{i=1}^{N} b_i$$

where ω is a parameter in the range (0,1) accounting for traffic congestion; in particular, in our simulations we consider two different situations characterized by ω = 0.7 and ω = 0.8, which represent *High Traffic* and *Medium Traffic* conditions, respectively.

As for the number of active Agents N, in our simulations we consider two cases: N = 100 and N = 1000. For each of these two cases and for each of the two considered traffic congestion conditions, ten simulation runs or *episodes* have been carried out, with a duration of $(15 \cdot 10^3)$ time steps for N = 100 and of $(15 \cdot 10^4)$ time steps for N = 1000: in each simulation run a different association among application instances, application types, average offered bitrates and Target QoE values is performed. Such associations are assumed to be fixed for the entire simulation run.

In the simple proposed simulation scenario, we assume that the set of Feedback Parameters ϕ_m (introduced in Section 2.1) includes, for any m = 1, 2, 3, just a single element denoted as ϕ_{QoS} and that the function g_m, introduced in (2), is computed on the basis of the well-known IQX hypothesis [5]. This means that (2) becomes:

$$PQoE_{i(m)}(t_k)=p_m\, e^{-\sigma_m \phi_{QoS}} + \tau_m \qquad (7)$$

where the parameter ϕ_{QoS} has been assumed to be equal to the difference between the traffic offered by the application instance and the corresponding bitrate currently allocated by the WFQ Scheduler. Note that the latter parameter depends on the CoS appointed at time t_k for the considered application instance, which actually impacts on the priority assigned by the WFQ Scheduler to the packets of the relevant traffic flow. We assume σ_1 = 0.5, σ_2 = 0.7, σ_3 = 1, as well as p_m = 1 and τ_m = 0 for all values of m; with these choices, $PQoE_{i(m)}(t_k)$ is always included in the range [0,1]. The learning rates $\alpha(t_k)$ appearing in (4), according to [31], are set to:

$$\alpha(t_k, a_1, a_2,\dots, a_N)= 1/(1+visit(t_k, a_1, a_2,\dots, a_N)) \qquad (8)$$

where $visit(t_k, a_1, \dots, a_N)$ is the number of times that a specific joint action $(a_1, a_2, \dots, a_N)$ has been enforced up to the iteration at time t_k. The discount rate is set to γ = 0.9. The selected joint reward function, consistent with the general criteria identified in Section 2, is:

$$r(t_k, a_1, a_2, \dots, a_N) = \sum_{i=1}^{N} w_i(t_k)$$

where the absolute value of w_i serves as an appropriately chosen penalty, which the i-th Agent is inflicted with, any time it exhibits either underperforming or overperforming behaviour. A proper choice of w_i may be the following:

- $w_i(t_k)$ = – 100 if $e_i(t_k) < -0.15$ (i.e., if severe underperformance is experienced by Agent i);
- $w_i(t_k)$ = – 10 if $-0.15 \le e_i(t_k) < 0$ (i.e., if minor underperformance is experienced by Agent i);
- $w_i(t_k)$ = – 1 if $0 \le e_i(t_k) < 0.1$ (i.e., if acceptable overperformance is experienced by Agent i);
- $w_i(t_k)$ = – 50 if $e_i(t_k) \ge 0.1$ (i.e., if undesirable overperformance is experienced by Agent i). (9)

In particular, the thresholds on the QoE Error values in (9) have been arbitrarily chosen in order to suitably classify the behaviour of Agent i at time t_k as a result of the joint action taken. Moreover, the initial policy, that is, the initial CoS-to-application association, is randomly generated.

Note that, even though the proposed proof-of-concept does not claim to represent any real network, a bottleneck link characterized by limited available bandwidth capacity can represent the uplink of a given cell of a cellular network. In such a scenario, a number of Agents roaming in the considered cell (and hence sharing the cell available uplink capacity) in the order of some hundreds (as assumed in this section) seems reasonable.

4.2 Numerical Results

This subsection shows the results obtained in the described simulation scenario; in particular, the H-MARL-Q algorithm is applied with a number of Agents N = 100 and N = 1000, both in the High and Medium Traffic conditions.

It should be emphasized that we can deal with such a high number of Agents due to the fact that the proposed H-MARL-Q algorithm relies on a Reduced Joint Action Space, which has cardinality $SN = 300$ in the scenario with 100 Agents ($S = 3$ and $N = 100$), and $SN = 3000$ in the scenario with 1000 Agents ($S = 3$ and $N = 1000$). If the original Joint Action Space were used, a solution relying on the MARL-Q algorithm would be unfeasible, since the cardinality would be $S^N = 3^{100} = 5.2 \cdot 10^{47}$, and $S^N = 3^{1000} = 1.42 \cdot 10^{477}$ in the two scenarios, respectively.

The results obtained with the H-MARL-Q algorithm are compared with the performance of a *Static algorithm* which adopts a static CoS assignment policy. The comparison with the MARL-Q algorithm is impossible due to the curse of dimensionality (as explained in Section 3.1).

The obtained results are expressed in terms of two quantities:

(i) the *Average Absolute QoE Error*, computed as the absolute value of the QoE Error expressed by (1), averaged over all the considered Agents and all the simulation episodes (see Figs. 3 and 4);

(ii) the *QoE Error Standard Deviation*, computed as the standard deviation of the QoE Error vector (e_1, e_2, …, e_N) (where e_i, for i = 1, 2, …, N, is expressed as in (1)) averaged over all the simulation episodes (see Figs. 5 and 6).

Note that the standard deviation accounts for the fairness among Agents: the smaller the standard deviation, the higher the fairness among Agents, as discussed in Section 2.1.

Figs. 3-6 clearly show that the H-MARL-Q algorithm remarkably outperforms the Static algorithm in all of the considered simulation cases. In particular, while under the Static algorithm the Average Absolute QoE Error is appreciably smaller in Medium rather than in High Traffic conditions, under the H-MARL-Q algorithm, for both N = 100 and N = 1000, the Average QoE Error bars corresponding to High and Medium Traffic conditions (see Figs. 3 and 4) exhibit values that are really close to each other: this means that the presented algorithm also allows to overcome the disadvantages related to the impact that the traffic congestion conditions produce on the bottleneck link.

Furthermore, the QoE Error Standard Deviation shown in Figs. 5 and 6 confirms the virtues of the H-MARL-Q algorithm, since the dispersion of the QoE Error values of the different Agents at the end of the learning procedure is significantly closer to zero than in the case when the Static algorithm is applied.

All these results evidently show that the dynamic and personalized selection of the most appropriate CoS for the ongoing application instances yields improved performance results, if compared with a static CoS assignment policy.

In addition, Fig. 7 shows the Average Absolute QoE Error trend, i.e., the evolution of the Average Absolute QoE Error over time.

Let the *settling time* denote the time needed by the Average Absolute QoE Error to reach a steady state. Once an acceptable preliminary agreement among Agents – yielding the selection of the most “promising” joint actions for solving the dynamic CoS assignment problem – has been reached in step (a), the error dynamics, as highlighted in Fig. 7, experiences a rapid decrease over the first 100 iterations of step (b) and then it takes some time to settle down to the steady-state value: in the figure, the settling time is approximately equal to 9000 iterations. So, the overall runtime required by the H-MARL-Q algorithm is the sum of the time t_a necessary to reach the preliminary agreement in step (a) plus the time t_b necessary to perform step (b), where t_b amounts to approximately 9000 iterations for N = 100 and t_a is negligible with respect to t_b. This is indeed an encouraging result which shows that the H-MARL-Q algorithm has to be preferred to the MARL-Q algorithm as the former achieves a satisfactory approximate solution in a reasonably smaller amount of runtime than the latter – whose runtime, instead, actually turns out to be unfeasibly long in scenarios where the number of Agents is counted in the order of hundreds or thousands.

The proposed approach to QoE Control enables a dynamic Class of Service selection aimed at reducing the error between the personalized Perceived QoE and the personalized Target QoE levels by properly driving the control procedures that handle the underlying networks. This result could be obtained by embedding an innovative Multi-Agent Reinforcement Learning algorithm, namely the proposed H-MARL-Q algorithm, in a centralized QoE Controller. Such an algorithm has been tested in a simple simulation scenario, with just the aim of providing a proof-of-concept and without claiming to represent any real network.

5. CONCLUSION

The proposed method presents several practical advantages:

(i) it does not require any *a-priori* knowledge of the environment (i.e., it is *model-free*) thanks to the adoption of a Reinforcement Learning based approach;

(ii) it is decoupled from QoE Evaluation, i.e., it can work in conjunction with any algorithm computing the Target QoE and the Perceived QoE values, and it allows a personalization level up to the single application instance, since the only signal exchanged at the interface between the QoE Controller and the QoE Evaluator is the QoE Error provided by (1);

(iii) it requires minimal signalling overhead since no communication exchange among Agents is needed and very little information has to be exchanged among the centralized QoE Controller and the distributed Agents;

(iv) it is characterized by a very good degree of scalability (thus being able to handle several hundreds of Agents) due to the fact that, as the joint action to be carried out at each time step is sought within a suitable Reduced Joint Action Space, the complexity of the proposed H-MARL-Q algorithm is linear in the number of Agents (as opposed to the well-known MARL-Q algorithm whose complexity is exponential in the number of Agents).

Note that the algorithm presented in this paper assumes the time-invariance of the Target QoE. However, the authors are carrying out further studies, based on concept drift in web/telecommunication systems [32], so as to address also the case of a time-varying Target QoE. In this last case, the Target QoE depends not only on the commercial profile of the users but also on the relevant feedbacks provided by the users themselves.

Moreover, the authors are presently carrying out further research based on a combinatorial multi-armed bandit approach to cooperative online learning [33, 34], with the aim of overcoming the centralized paradigm and, consequently, of developing a solution in which the QoE Control functionalities

can be fully distributed into the Agents.

Finally, note that the overall modular architecture sketched in Fig. 1 – within which Reinforcement Learning algorithms embedded in a QoE Controller play the role of dynamically selecting (on the basis of real-time feedbacks provided by a proper QoE Evaluator) appropriate Reference Values which should drive environment-specific procedures – has proved to be so flexible that the authors are reproducing it also in the domains of intelligent transport systems and telemedicine within the framework of EU-funded research projects.

ACKNOWLEDGEMENT

The authors wish to thank Proff. A. Isidori, C. Gori Giorgi, S. Battilotti, F. Facchinei, and L. Palagi for their continuous support and valuable contributions to the work within the PLATINO project. The authors also wish to thank Ing. J. Capolicchio for the fruitful discussions.

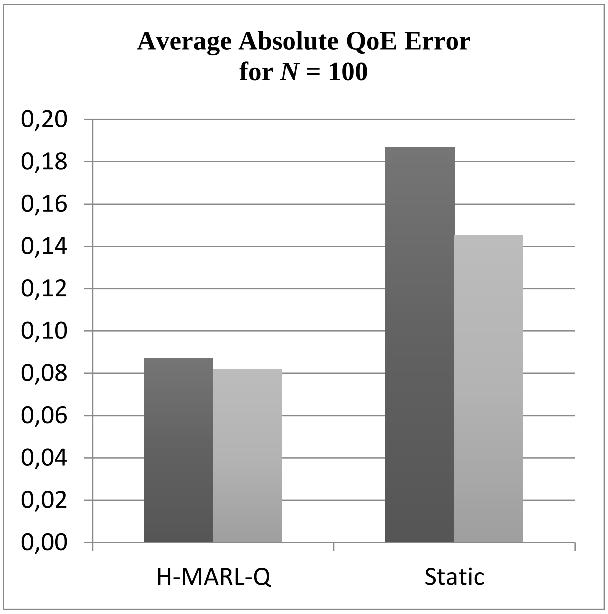

Fig. 3. Average Absolute QoE Error for N = 100. The dark-grey bar and the light-grey bar represent the Average Absolute QoE Error in High and Medium Traffic conditions, respectively.

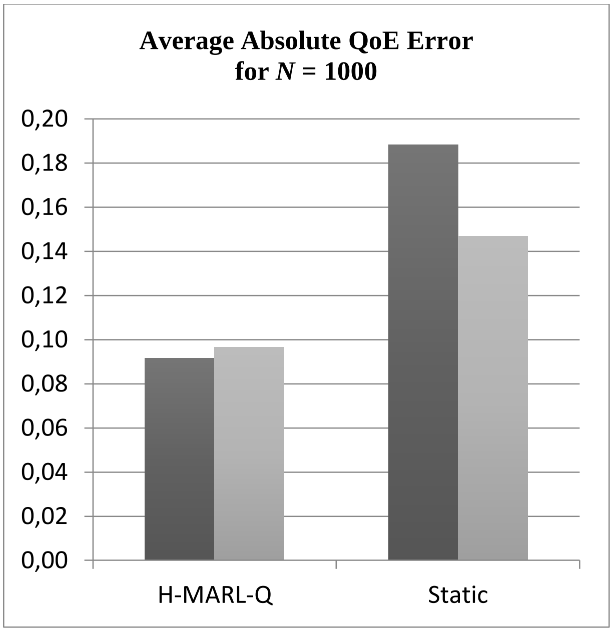

Fig. 4. Average Absolute QoE Error for N = 1000. The dark-grey bar and the light-grey bar represent the Average Absolute QoE Error in High and Medium Traffic conditions, respectively.

ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

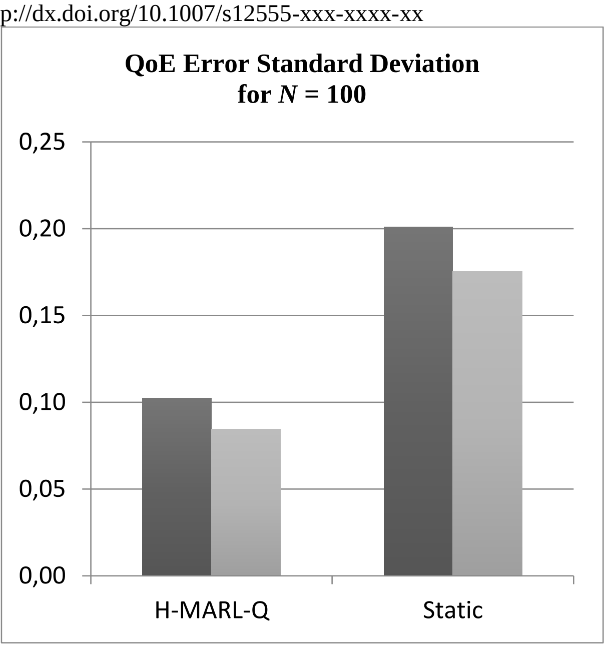

Fig. 5. QoE Error Standard Deviation for N = 100. The dark-grey bar and the light-grey bar represent the QoE Error Standard Deviation in High and Medium Traffic conditions, respectively.

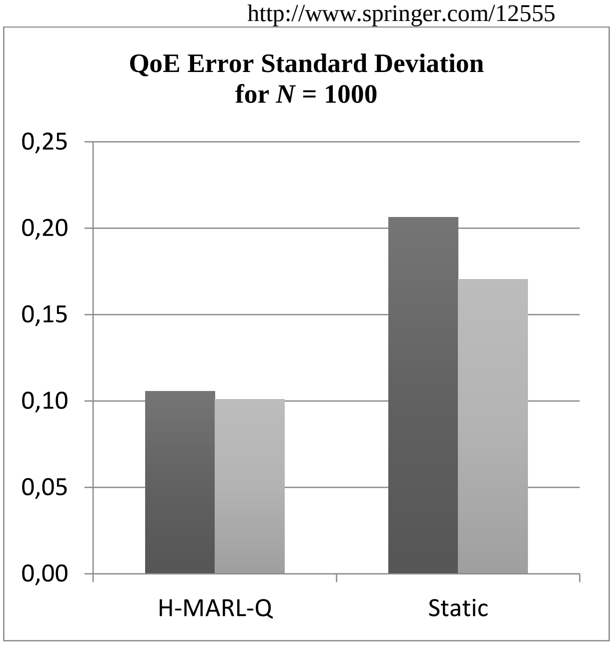

Fig. 6. QoE Error Standard Deviation for N = 1000. The dark-grey bar and the light-grey bar represent the QoE Error Standard Deviation in High and Medium Traffic conditions, respectively.

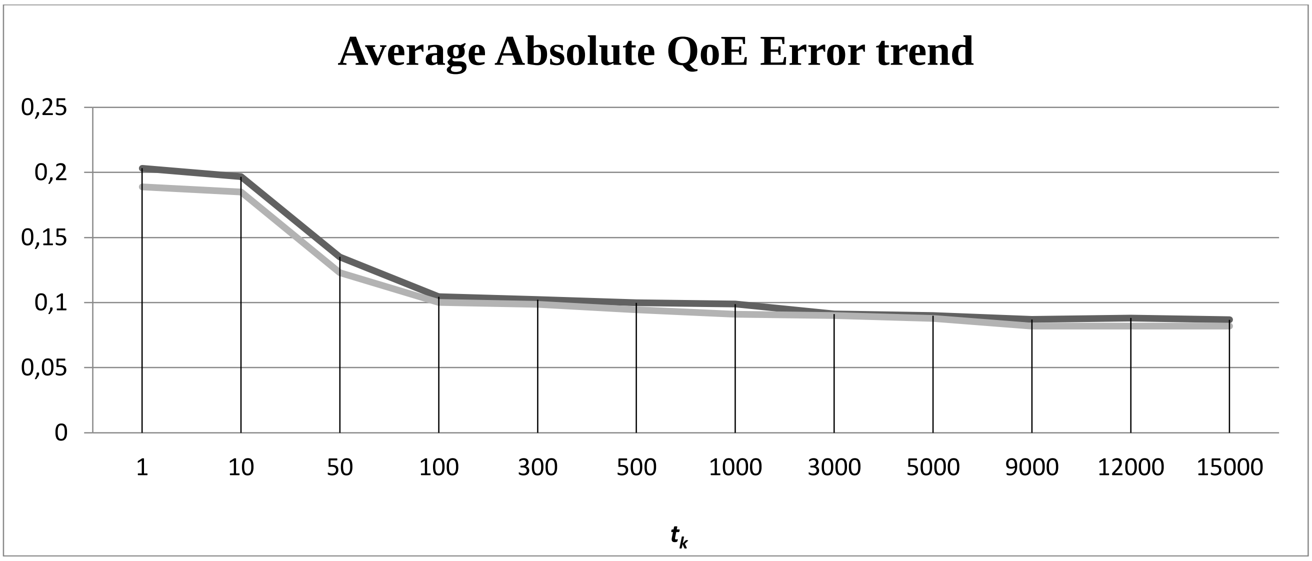

Fig. 7. Average Absolute QoE Error trend, corresponding to step (b) of the H-MARL-Q algorithm, in High (black line) and Medium (grey line) Traffic conditions with N = 100.

REFERENCES

[1] ITU-T, “Amendment 1: Recommendation P.10/G.100. New Appendix I – Definition of Quality of Experience (QoE),” *Telecommun. Stand. Sect. Itu-T*, vol. 100, no. 2006, 2007.

[2] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, and G. Pujolle, “Quality of experience of VoIP service: A survey of assessment approaches and open issues,” *IEEE Commun. Surv. Tutorials*, vol. 14, no. 2, pp. 491–513, 2012.

[3] S. Singh, J. G. Andrews, and G. de Veciana, “Interference Shaping for Improved Quality of Experience for Real-Time Video Streaming,” *IEEE J. Sel. Areas Commun.*, vol. 30, no. 7, pp. 1259–1269, 2012.

[4] S. Canale, F. Facchinei, R. Gambuti, L. Palagi, and V. Suraci, “User profile based Quality of Experience,” in *Proceedings of the 18th International Conference on Circuits, Systems, Communications and Computers (CSCC 2014)*, Santorini Island, Greece, *Advances in Information Science and Applications – Volume II*, 2014.

[5] M. Fiedler, T. Hossfeld, and P. Tran-Gia, “A generic quantitative relationship between quality of experience

and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36-41, 2010.

[6] "Platform for Innovative Services in Future Internet," Italian Ministry of University and Research (MIUR) PLATINO project, grant greement no. PON01_01007, http://www.progettoplatino.it/.

[7] FI-WARE (Future Internet Ware), EU FP7-ICT Large-scale Integrating Project (IP), 2011-2014, grant agreement no. 312826, http://www.fi-ware.eu/.

[8] FI-Core (Future Internet - Core Platform), EU FP7-ICT Large-scale Integrating Project (IP), 2014-2016, grant agreement no. 632893, http://cordis.europa.eu/project/rcn/192274_en.html.

[9] R. Gambuti, A. Di Giorgio, F. Liberati, A. Pietrabissa, V. Suraci, and F. Delli Priscoli, "Profiled Quality of Experience Control," submitted to *Information Technology and Control*, 2016.

[10] F. Delli Priscoli, A. Isidori, L. Marconi, "A Dissipativity-based Approach to Output Regulation of Non-Minimum-Phase Systems," *Systems and Control Letters*, Elsevier Science Pub., vol. 58, pp. 584-591, 2009.

[11] L. Ricciardi Celsi, R. Bonghi, S. Monaco, and D. Normand-Cyrot, "On the Exact Steering of Finite Sampled Nonlinear Dynamics with Input Delays," in *Proceedings of the 1st Conference on Modelling, Identification and Control of Nonlinear Systems* (*MICNON 2015), IFAC-PapersOnLine,* vol. 48, no. 11, pp. 674-679, Saint-Petersburg, June 2015, DOI: 10.1016/j.ifacol.2015.09.265.

[12] L. Ricciardi Celsi, S. Battilotti, F. Cimorelli, C. Gori Giorgi, S. Monaco, M. Panfili, V. Suraci, and F. Delli Priscoli, "A Q-Learning Based Approach to Quality of Experience Control in Cognitive Future Internet Networks," in *Proc. of the 23rd Mediterranean Conference on Control and Automation (MED15)*, pp. 1045-1052, June 16-19, 2015, Torremolinos, Spain, DOI: 10.1109/MED.2015.7158895.

[13] M.M.L. Littman, "Friend-or-foe Q-learning in general-sum Games," in *ICML*, 2001, vol. 1, pp. 322–328.

[14] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *Syst. Man, Cybern. Part C Appl. Rev. IEEE Trans.*, vol. 38, pp. 156–172, 2008.

[15] S. Manfredi, "An algorithm for fast rendezvous seeking of wireless networked robotic systems," *Ad Hoc Networks*, Vol. 11, No.7, pp. 1942-1950, DOI:10.1016/j.adhoc.2012.06.010, 2013.

[16] F. Delli Priscoli, A. Isidori, L. Marconi, A. Pietrabissa, "Leader-Following Coordination of Nonlinear Agents under Time-varying Communication Topologies," *IEEE Transactions on Control of Network Systems*, vol. 2, no: 4, 2015, pp. 393-405, DOI: 10.1109/TCNS.2015.2426752.

[17] M. Castrucci, F. Delli Priscoli, A. Pietrabissa, and V. Suraci, "A Cognitive Future Internet Architecture," *Futur. Internet, Lect. Notes Comput. Sci. Vol. 7858 2013*, vol. 6656, pp. 91–102, 2011.

[18] M. Castrucci, M. Cecchi, F. Delli Priscoli, L. Fogliati, P. Garino, and V. Suraci, "Key Concepts for the Future Internet Architecture," *Future Network and Mobile Summit 2011*, Warsaw, June 2011.

[19] C. Bruni, F. Delli Priscoli, G. Koch, A. Palo, and A. Pietrabissa, "Quality of Experience Provision in the Future Internet," *IEEE Syst. J.*, pp. 1–11, 2014.

[20] N. Nisan and A. Ronen, "Algorithmic mechanism design," *Games and Economic Behavior*, vol. 35, no. 1-2, pp. 166-196, 2001.

[21] F. Delli Priscoli, V. Suraci, A. Pietrabissa, and M. Iannone, "Modelling Quality of Experience in Future Internet Networks," in *Proc. of the Future Network & Mobile Summit (FutureNetw)*, 2012, ISBN: 978-1-905824-16-8.

[22] C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *Proc. 2003 Conf. Appl. Technol. Archit. Protoc. Comput. Commun. - SIGCOMM '03*, pp. 137–148, 2003.

[23] S. Battilotti, C. Gori Giorgi, S. Monaco, M. Panfili, A. Pietrabissa, L. Ricciardi Celsi, and V. Suraci, "A Multi-Agent Reinforcement Learning Based Approach to Quality of Experience Control in Future Internet Networks," in *Proc. of the 34th Chinese Control Conference (CCC2015)*, pp. 6495-6500, July 28-30, 2015, Hangzhou, China, DOI: 10:1109/ChiCC.2015.7260662.

[24] Q. Jiang, H. Xi, and B. Yin, "Dynamic file grouping for load balancing in streaming media clustered server systems," *International Journal of Control, Automation and Systems*, vol. 7, no. 4, pp. 630-637, 2009.

[25] C.J.C.H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.

[26] G. Santhi, Alamelu Nachiappan, Mougamadou Zaid Ibrahime, R. Raghunadhane, and M. K. Favas. "Q-learning based adaptive QoS routing protocol for MANETs," *IEEE Int. Conf. on Recent Trends in Information Technology (ICRTIT)*, pp. 1233-1238., 2011.

[27] A. Pietrabissa, "A Reinforcement Learning Approach to Call Admission and Call Dropping Control in Links with Variable Capacity", *European Journal of Control*, Vol. 17, Issue 1, 2011, pp. 89-103, ISSN 0974-3580, DOI: 10.3166/EJC.17.89-103.

[28] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.

[29] T. Jaakkola, M.I. Jordan, and S.P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Computation*, vol. 6, pp. 1185-1201, 1994.

[30] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 1–12, 1989.

[31] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[32] I. Zliobaité, "Learning under concept drift: an overview," arXiv preprint arXiv:1010.4784, 2010.

[33] Y. Gai, B. Krishnamachari, and Q. Zhao, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 5, pp. 1466-1478, 2012.

[34] J. Xu, C. Tekin, S. Zhang, and M. van der Schaar, "Distributed Multi-Agent Online Learning Based on Global Feedback," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2225-2238, 2015.

Francesco Delli Priscoli was born in Rome, Italy, in 1962. He received the degree in Electronics Engineering (*summa cum laude*) and the Ph.D. degree in Systems Engineering from the University of Rome "La Sapienza" in 1986 and 1991, respectively. Since 1991, he has been working at the University of Rome "La Sapienza," where, at present, he is Full Professor of Automatic Control, Control of Autonomous Multi-Agent Systems, and Control of Communication and Energy Networks. In the framework of his academic activity, he has mainly researched on resource/service/content management procedures and on cognitive techniques for telecommunication and energy networks, by largely adopting control-based methodologies. He is the author of about 180 papers appeared in major international journals (about 60), on books (about 10) and in conference proceedings (about 110). He also holds four patents. He is an associate editor of Control Engineering Practice and a member of the IFAC Technical Committee on Networked Systems. He was/is the scientific responsible, at the University of Rome "La Sapienza," for 31 projects financed by the European Union (Fourth, Fifth, Sixth, Seventh and Eighth Framework Programmes) and by the European Space Agency (ESA), as well as for many national projects and cooperations with major industries. His present research interests concern closed-loop multi-agent learning techniques for Quality of Experience evaluation and control in advanced communication and energy networks, as well as all the related networking algorithms.

Alessandro Di Giorgio was born in Rome, Italy, in 1980. He received the degree (*cum laude*) in Physics in 2005, and the Ph.D. degree in Systems Engineering from the University of Rome "La Sapienza," in 2010. He is currently a Research Fellow in Automatic Control, working on original applications of control systems theory to the resource manegement problem in the field of power systems and telecommunications networks; he is author of about 40 papers and book chapters on these topics, mainly produced in the context of national and European research projects.

Federico Lisi was born in Rome, Italy, in 1986. He received the M.Sc. degree in Artificial Intelligence and Robotics with 110/110 in 2015 from the University of Rome "La Sapienza." He has been working in the MIUR project PLATINO and in the FP7 project SWIPE. His main research interests concern reinforcement learning for multi-agent systems, path planning for autonomous robots, neural networks and data mining.

Salvatore Monaco was born in Udine, Italy, in 1951 and he has been a Full Professor of Systems Theory at the University of Rome "La Sapienza" since 1986. He was a member of the ASI (Italian Space Agency) Scientific Committee from 1989 to 1995, of the Executive Council of the EUCA (European Union Control Association) from 1990 (foundation year) to 1997, and of the ASI Working Group on Evaluation from 1999 to 2001. He has also been a member of the ASI Technological Committee since 1997. He has promoted technological transfer in the area of Automation. In 1995, he served as scientific advisor for the Director of the Joint Research Center of the European Union. Since 2001, he has been president of the council for the degree of Systems and Control Engineering at the University of Rome "La Sapienza" and also president of the Scientific Committee of the "Université Franco-Italienne," an inter-governmental institution for coordinating research and didactics. His research activity is in the field of Systems and Control Theory and applied research in spacecraft control, mobile robot control and control of communication networks.

Antonio Pietrabissa is Assistant Professor at the Department of Computer, Control, and Management Engineering "Antonio Ruberti" (DIAG) of the University of Rome "La Sapienza," where he received his degree in Electronics Engineering and his Ph.D. degree in Systems Engineering in 2000 and 2004." Since 2000, he has worked with the Network Control Laboratory at DIAG, in the framework of National and European projects related to ICT. Since 2007, he has been member of the Scientific and Technical Committee of the Consortium for the Research in Automation and Telecommunication (CRAT). Since 2000, he has participated in 15 research projects funded by the European Union (EU), 2 projects funded by the European Space Agency (ESA), 2 projects funded by the Italian Space Agency (ASI), and 3 projects funded by the Italian Ministry of Education, Universities and Research (MIUR). His main research focus is on the application of systems and control theory methodologies to the analysis and control of networks. He is author of more than 30 journal papers and over 60 conference papers and book chapters on these topics.

Lorenzo Ricciardi Celsi was born in Rome, Italy, in 1990. He received the B.Sc. degree in Electronics Engineering in 2011 and the M.Sc. degree in Control Engineering in 2014, both *summa cum laude* from the University of Rome "La Sapienza." He is currently a PhD Candidate in "Automatica, Bioengineering and Operations Research" at the same university. He has been working on reinforcement learning algorithms within the framework of the FP7 project T-NOVA and the MIUR project PLATINO. He is also working on the development of the intelligent multi-modal transport system foreseen by the H2020 project BONVOYAGE. His main research interests are: nonlinear systems and control theory with application to communication networks as well as to aircraft and spacecraft control, advanced control methodologies for multi-agent systems and machine learning algorithms and methods.

Vincenzo Suraci was born in Rome, Italy, in 1978. He graduated in Computer Engineering *summa cum laude* in 2004 at the University of Rome "La Sapienza." In 2008 he received his Ph.D. degree in Systems Engineering at the Department Computer, Control, and Management Engineering "Antonio Ruberti" (DIAG) of the same university. Currently, he is a Researcher at eCampus University and Project Manager at CRAT. His main research interest is to develop and adapt advanced control and operations research methodologies (such as reinforcement learning, column generation, hybrid automata, and discrete event systems) for the solution of challenging and emerging engineering problems: e.g., connection admission control, access technologies selection, QoE/QoS cognitive control, resource management over heterogeneous technologies, convergence of heterogeneous networks. He has achieved a wide experience in the field of applied research and project management. Since 2011, he has been managing the EU-funded Future Internet Core Platform research project FI-WARE. In 2012, he also applied for a EU Patent request on DVB as a result of his profitable research in the framework of EU research projects.