Deep Reinforcement Learning for Scheduling Uplink IoT Traffic with Strict Deadlines

Benoît-Marie Robaglia*†, Apostolos Destounis*, Marceau Coupechoux†, Dimitrios Tsilimantos*
*Mathematical and Algorithmic Sciences Lab, Paris Research Center, Huawei Technologies Co. Ltd.
†LTCI, Telecom Paris, Institut Polytechnique de Paris

***Abstract*—This paper considers the Multiple Access problem where N Internet of Things (IoT) devices share a common wireless medium towards a central Base Station (BS). We propose a Reinforcement Learning (RL) method where the BS is the agent and the devices are part of the environment. A device is allowed to transmit only when the BS decides to schedule it. Besides the information packets, devices send additional messages like the delay or the number of discarded packets since their last transmission. This information is used to design the RL reward function and constitutes the next observation that the agent can use to schedule the next device. Leveraging RL allows us to learn the sporadic and heterogeneous traffic patterns of the IoT devices and an optimal scheduling policy that maximizes the channel throughput. We adapt the Proximal Policy Optimization (PPO) algorithm with a Recurrent Neural Network (RNN) to handle the partial observability of our problem and exploit the temporal correlations of the users' traffic. We demonstrate the performance of our model through simulations on different number of heterogeneous devices with periodic traffic and individual latency constraints. We show that our RL algorithm outperforms traditional scheduling schemes and distributed medium access algorithms.**

***Index Terms*—Multiple Access, Reinforcement Learning, Proximal Policy Optimization, POMDP, Internet of Things, Wireless sensor networks, scheduling.**

I. Introduction

With the development of the Internet of Things (IoT), billions of devices are expected to be connected, such as smartphones, sensors and tracking devices [1]. This very large amount of wireless connected devices makes the standard multiple access (MA) protocols inefficient due to their tendency to saturate quickly as the number of devices increases. In addition, IoT devices need to transmit time sensitive data as many of their applications are related to tracking and monitoring (home monitoring, factory automation, health monitoring or energy management) [2]. IoT devices are also expected to have very limited wireless communication resources because of their limited battery life [3] and thus need to have minimal interaction with a base station (BS) to operate. Therefore, MA protocols need to operate autonomously with minimal control signalling in this type of systems, commonly known as massive machine type (mMTC) communications systems. Since this type of devices are characterized by having a sporadic traffic, deterministic access schemes such as Time Division Multiple Access (TDMA) and Orthogonal Frequency Division Multiple Access (OFDMA) become inadequate for the mMTC requirements. New protocols, capable of adapting to these new dynamic environments are called for. Machine Learning and more precisely Reinforcement Learning (RL) are therefore promising candidates for the multiple access problem [4].

A. Related Work

Random access protocols: traditional protocols are based on random access. One of the first and the most famous one is the ALOHA protocol [5], where a device attempts to access the channel with a probability p. Another random access protocol extensively studied is the Carrier Sense Multiple Access (CSMA), where the transmitter can "sense" the channel before accessing it [6]. However, most random access protocols suffer from collisions and thus, achieve a sub-optimal throughput. They are also unable to guarantee strict deadlines.

Decentralized RL protocols: distributed approaches are the most appealing solutions as they can be deployed without the need of a central coordinator and thus meet the IoT constraints of scalability and low energy consumption. With the recent development of deep neural networks, deep RL solutions [7] have been proposed as good candidates for solving high dimensional sequential decision making problems. Regarding the distributed multiple access challenges, most approaches use Multi-Agent Reinforcement Learning (MARL) to model IoT sensors. Their objective is to learn how to cooperate in order to access the channel with minimal collisions and thus maximize the throughput. The easiest and most natural approach to extend the Q-learning theory of RL to the multi-agent case is independent deep Q-learning (IQL) [8]. The idea is that each agent learns independently and simultaneously its own deep Q-function while considering the other agents as part of the environment. While IQL does not give any theoretical convergence guarantees as the environment is non-stationary due to the concurrent learning, it has demonstrated good empirical results, in particular in wireless communications. For instance, in multiple access scenarios, several approaches [9]–[11] apply deep IQL to the dynamic spectrum access problem in order to learn a channel access strategy for the secondary users with low collision rate. More generally, [12] applied IQL to multiple access where other nodes are traditional MA protocols (TDMA, ALOHA) and show that their model can learn the optimal transmission strategy (maximum throughput). However, these decentralized models aim at maximizing the throughput by minimizing the number of collisions without considering delay constraints and seem inadequate to solve a system model with strict deadlines.

Centralized protocols: centralized solutions require a central controller to gather relevant information from every device every time it wants to allocate the medium. This constant communication is very heavy in energy and communication resources and thus is very impractical for IoT scenarios. However, centralized protocols have the advantage of allowing transmissions without collisions and potentially achieve a maximum throughput. One way to use a centralized approach while being energy efficient is by exploiting historical data with Machine Learning (ML) [13] to learn an optimal scheduling strategy. Yet, the biggest limitation of these data-based methods is to constitute a proper dataset, let alone getting a labeled training set if we intend to apply supervised learning. Another way to tackle this substantial limitation is to consider centralized solutions with very limited communication where the BS tries to predict the traffic of the devices. A popular method to learn an optimal scheduling policy under uncertainty is Multi-Armed Bandits [14], [15], but unlike RL, actions are not conditioned to states and do not have impact on the environment. In RL, the authors in [16] model a network controller with a deep Q-network to determine the next update time at which each sensor should transmit its observations. Similarly, and closest to our work, [17] tackles the dynamic multi-channel access problem with an actor-critic model where the agent (the access point) can only sense the chosen channel at each iteration.

Even if extensive work has been done to propose centralized solutions for multiple access, to the best of our knowledge this paper is the first that offers a centralized approach tackling partial observability and latency constraints with deep RL.

B. Our contribution

In this paper, we consider the multiple access problem where N heterogeneous devices have to transmit to a BS within a given deadline. To do so, they can access a common communication medium, but only when the BS chooses to schedule them. Therefore, only one device can transmit at each time slot. This has the benefit of avoiding collisions, which typically occur when using standard random access protocols. However, the BS can only observe the state of the last polled device. Thus, it needs to learn efficiently the traffic patterns of the devices in order to maximize the throughput as it does not know when a device has a packet to transmit. Therefore, our model falls in the category of centralized protocols with limited communication and scales linearly with the network size. To incorporate the strict latency constraint in our model, we impose the devices to transmit their packets within a given deadline δ so that if a packet is not sent before, it is discarded. This framework captures the practical challenges of real-time wireless networks [18].

Our approach models the access point as an RL agent: it combines the proximal policy optimization (PPO) algorithm [19] with deep Recurrent Neural Networks (RNNs) to handle partial observability. In addition, we incorporate invalid action masking [20] to speed up the training process and make our algorithm more efficient with large number of devices.

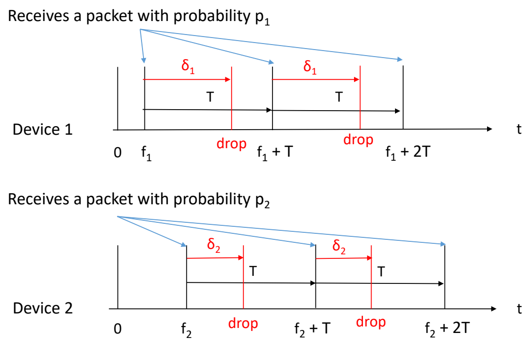

Fig. 1. Traffic model of 2 heterogeneous devices. Every period T, they receive a packet with probability p_i and need to deliver it within δ_i slots. They are also not synchronous: they have an individual offset f_i, $i \in \{1, 2\}$.

II. Problem Formulation

A. Traffic model

We consider a network of N devices communicating with a BS over a wireless shared channel on the uplink. The wireless channel is supposed to be time-slotted and at every slot, the BS polls one of the devices for a potential uplink transmission. We assume the traffic pattern of every device to be periodic, i.e. every period of T time slots, a device i has a probability p_i of receiving a new packet. All packets are supposed to require the same transmission time of one time slot. Propagation delay is assumed to be negligible. Moreover, each device has an individual constraint δ_i, such that when a packet has not been transmitted before the constraint δ_i, it is dropped. We allow the devices to be heterogeneous in the following sense:

- They can have different packet arrival probabilities p_i.
- They can have different packet delivery constraints δ_i.
- They are not synchronous: each device is assigned an offset parameter $f_i \in [0, T]$.

However, the traffic period T is assumed the same for every device and is known by the BS. At every slot $t \geq 0$, the probability for a device $i \in [1, N]$ of having a new packet is:

$$q_i(t|f_i, p_i, T) = \mathbb{1}_{\{t[T]=f_i\}} p_i \tag{1}$$

with $\mathbb{1}_{\{.\}}$ the indicator function and $[\cdot]$ the modulo operator such that $x[y] = z$ if there exists $k \in \mathbb{Z}$ such that $x = ky + z$ for $x, y, z \in \mathbb{N}$. An illustration of the traffic is shown in Fig.1.

B. Problem formulation

Compared to traditional MA protocols like ALOHA [5], devices do not decide when to transmit their packets. In our case, the BS is a central controller that decides whether or not a device can transmit. At every time step t, the BS schedules a sensor to send its packet. If the latter has indeed a packet to send, the transmission is successful. On the other hand, if it does not have a packet, it remains idle. We illustrate these interactions in Fig.2. The main advantage of this model is that it guarantees no collision as all decisions are made by the BS. However, in order to minimize the control overhead, we only allow the controller to have access to partial information,

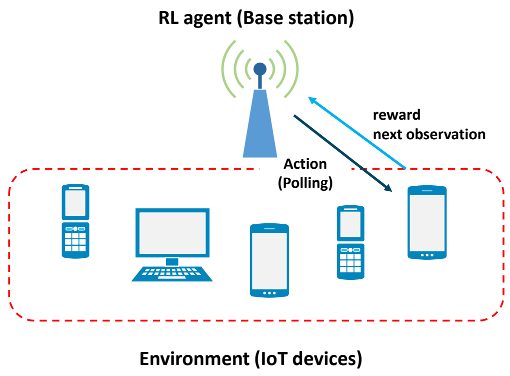

Fig. 2. Multiple access problem modeled as a polling problem where the BS is the RL agent. The IoT devices share a common communication channel with the BS.

that is the information received by the device after polling it, which explains why the BS can poll a device that has no packet to send. Indeed, assuming a centralized agent that can gather relevant information before allocating the communication resources is not realistic to meet the strict performance requirements of the IoT. As a consequence, many packets can be lost and the throughput will be sub-optimal if the RL agent is not able to learn the traffic patterns successfully.

C. Partially observable Markov Decision Process

We model the problem as a Partially Observable Markov Decision Process (POMDP) where an agent observes an environment and interacts with it. We define $s = (s^1, s^2, \dots, s^N) \in \mathbb{N}^N$ the environmental state where $s^i \in \mathbb{N}$ is the time a packet has spent in device i's buffer. For example, $s^i = 0$ means i's buffer is empty. The action set of the agent is $\mathcal{A} = \{1, 2, \dots, N\}$: at each time, the BS schedules a device $a \in \mathcal{A}$ that can transmit its packet if its buffer is not empty. When a sensor i is scheduled, in addition to the main message, it also transmits the number of discarded packets η_t^i since the last time it was scheduled as an additional message. We thus define the reward of the agent r_t as a trade-off between maximizing the throughput (number of successful transmissions) and minimizing the number of discarded packets:

$$r_t(s_t, a) = \beta \mathbb{1}_{\{s_t^a > 0\}} + (1-\beta)\frac{1}{1+\eta_t^a} \tag{2}$$

where $\beta \in [0,1]$ is a hyperparameter balancing the preferences of the agent between both quantities. Note that even if maximizing the throughput and minimizing the number of dropped packets seem equivalent, the agent has different incentives depending on the load. Indeed, if a few devices have very high arrival probabilities, the agent is almost certain to get a strictly positive reward when scheduling these devices. If the number of available slots is limited (inferior to the number of devices), only these devices will be scheduled and the learned policy would not be fair. Thus, balancing the number of successful transmissions with the number of discarded packets is a way to ensure fairness while also maximizing the throughput.

Every step t, the agent can only observe $o_t = (a_{t-1}, s_{t-1}^{a_{t-1}}) \in \mathcal{A} \times \mathbb{N}$, which is the device chosen at time $t-1$ and its corresponding state. The agent keeps a history of the H most recent observations, $h_t = (o_t, o_{t-1}, \dots, o_{t-H+1})$. An action is selected following the policy $\pi(\cdot|h_t)$, which is a probability distribution over the action space $\mathcal{A}$ given the H most recent observations h_t.

Finally, we can write the transition from a state s_t to a state s_{t+1}. A component $i \in \{1, 2, \dots, N\}$ of a new state, knowing the action and the previous state is:

$$s_{t+1}^i | a, s_t^i = \begin{cases} 0 & \text{if } a = i \\ 0 & \text{if } a \neq i \text{ and } s_t^i > \delta^i \\ s_t^i + 1 & \text{if } a \neq i \text{ and } 0 < s_t^i \leq \delta^i \\ X_i \sim \mathcal{B}(q_i) & \text{if } s_t^i = 0 \text{ and } t[T] = f_i \end{cases} \tag{3}$$

where $\mathcal{B}(\cdot)$ is the Bernoulli distribution.

In other words, the next state of a device i becomes 0 if it has been polled in the last time slot or if the last state reached the constraint δ_i. If it has not been polled and has not reached the constraint yet, we increment the last state. Finally, if the last state was 0, we draw a new packet according to a Bernoulli distribution with parameter q_i if $t[T] = f_i$.

III. Single-Agent Reinforcement Learning Approach

In order to solve the POMDP formulated above, we use Deep Reinforcement Learning, which has demonstrated state-of-the-art performances in solving complex sequential decision making problems ([7]) even in a partially observable setting [21]. We propose a policy gradient algorithm with masked invalid actions to solve this POMDP problem.

Policy gradient methods aim at finding the parameterized policy π_θ that maximizes the expected sum of discounted rewards: $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\tau} \gamma^t r_t]$ with θ the vector of parameters, γ the discount factor and τ the length of the trajectory. This optimization problem is usually solved with a stochastic gradient ascent algorithm with a gradient estimate. The most common estimator is given by the policy gradient theorem [22]: $\hat{g} = \hat{\mathbb{E}}_B[\sum_{t=0}^{\tau} \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$ with $\hat{A}_t$ an estimator of the advantage function $A(s_t, a_t)$ and $\hat{\mathbb{E}}_B$ the empirical average over a finite batch of trajectories.

A. Trust Region Policy Optimization

Vanilla policy gradient algorithms suffer from sample inefficiency leading to a convergence to local optima and large policy steps (resulting in high variance estimators and sometimes performance collapse). The authors in [23] introduced Trust Region Policy Optimization (TRPO) that restricts the amplitude of the policy update with a KL divergence constraint. Formally, TRPO's optimization problem is:

$$\max_\theta \mathbb{E}_{s,a\sim\pi_{\theta\text{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}A(s,a)\right] \tag{4}$$

$$\mathbb{E}_s[KL[\pi_{\theta_{\text{old}}}(\cdot|s), \pi_\theta(\cdot|s)]] \leq \delta \tag{5}$$

where θ_{old} are the policy parameters before the update. However, TRPO is a second order method as it requires the computation of a second order matrix when approximating the KL term which makes it computationally expensive. Proximal Policy Optimization (PPO) [19] uses policy ratio clipping to solve the trust region constraints problem with a first order method and the objective becomes:

$$\mathbb{E}_{s,a}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}A(s,a), \phi(\epsilon)A(s,a)\right)\right] \tag{6}$$

with $\phi(\epsilon) = \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}, 1-\epsilon, 1+\epsilon\right)$ the probability ratio clipped to $(1-\epsilon, 1+\epsilon)$ and $\epsilon > 0$ a hyperparameter. The intuition behind the clipping is to prevent the new policy from getting far from the old one.

B. Speeding up the learning process with action masking

When the number of devices increases, the action space becomes larger, therefore training a policy becomes slower and sometimes requires a deeper architecture (more layers, more neurons) to learn the optimal policy. In order to address this issue, we use a trick called *invalid action masking* [20] to provide an algorithm, namely Filtered PPO, able to solve the polling problem for any number of agents with the same architecture and the same parameters.

The idea is to speed up the training process by masking actions which we know that are suboptimal for the agent because of the structure of the problem. Indeed, the traffic is periodic so one device cannot have more than one packet to transmit during the arrival period T. Thus, we mask the κ last actions made by the agent in the policy by setting their probability to 0. [20] theoretically and experimentally studied the action masking methodology for policy gradient algorithms and prove that it leads to valid policy gradient updates. In our experiments, we took $\kappa = T$.

C. Algorithm overview

In this section, we explain in detail our implementation of PPO for the device polling problem.

First, our algorithm needs to consider the partial observability of our problem as the agent can only see the states of the devices it has previously selected. The authors in [21] tackle partial observability using a recurrent architecture that allows the agent to better approximate the underlying system state based on past action-observation pairs. We denote h_t the action-observation history of the agent at time step t. We adapt the PPO theory to the partially observable setting by using the approach of [21]. In practice, we replace the state s by the action-observation history h and we replace the linear layer by a recurrent one. For the recurrent architecture, we chose a long short-term memory (LSTM) architecture to alleviate the vanishing gradient problem that usually exists by using an RNN [24].

We can define the policy loss based on (6):

$$L_\pi(\theta) = -\hat{\mathbb{E}}_B\left[\min\left(\frac{\pi_\theta(a_t|h_t)}{\pi_{\text{old}}(a_t|h_t)}\hat{A}_t, \phi(\epsilon)\hat{A}_t\right)\right], \tag{7}$$

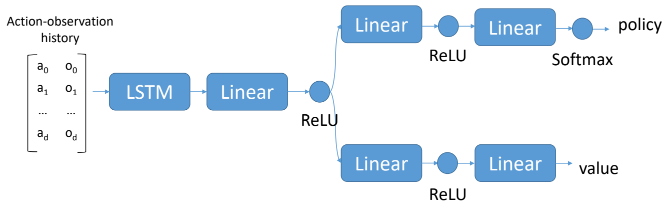

Fig. 3. Neural network architecture

where we recall that $\hat{\mathbb{E}}_B$ is the empirical average over a finite batch of trajectories.

The advantage function A in a POMDP setting is defined by $A(h_t, a_t) = Q(h_t, a_t) - V(h_t)$ which describes the relative value of an action with respect to the value of the state (how much better or worse it is to take this action). $Q(h_t, a_t)$ is the Q-function evaluating the value of choosing the action a_t based on the history h_t, and $V(h_t)$ is the value function indicating the value of the action-observation history h_t. Among all the ways to derive an estimator of the advantage function [25], we chose to use the simplest one using the empirical rewards:

$$\hat{A}_t = \hat{R}_t - \hat{V}_\theta(h_t) \tag{8}$$

with $\hat{R}_t = \sum_{t'=t}^{\tau} r_{t'}$ the sum of future rewards obtained on the trajectory and $\hat{V}_\theta(h_t)$ an estimate of the value function with a neural network. Note that the value and policy network share the same parameters θ.

The neural network we use is the one described in Fig.3. All hidden layers have 100 neurons. It takes the action-observation history as input and returns the policy vector and the value of the corresponding input. To update the head corresponding to the value estimator, we minimize the Mean Square Error (MSE) between our value estimator and the sum of future rewards:

$$L_V(\theta) = \hat{\mathbb{E}}_B(\hat{V}_\theta(h_t) - \hat{R}_t)^2. \tag{9}$$

Therefore, the total loss we minimize is:

$$L(\theta) = L_\pi(\theta) + L_V(\theta). \tag{10}$$

Algorithm 1: PPO for centralized multiple-access.

```
Input Initial policy parameters $\theta_0$
for $k$ = 0, 1, 2, ... NUM UPDATES do
    1) Run the policy $\pi_{\theta_k}$ and collect a set of trajectories
    2) Compute the advantage estimates $\hat{A}_t$ with (8)
    3) Compute the policy loss $L_\pi(\theta)$ and the value loss
       $L_V(\theta)$ from (7) and (9)
    4) Minimize the total loss $L(\theta) = L_\pi(\theta) + L_V(\theta)$
       with multiple steps of stochastic gradient descent.
end
```

IV. Experiments

We tested our algorithm with on different number of devices ranging between 6 and 60, with and without offsets. The environment parameters are chosen as follows:

- The arrival probabilities are chosen from $\{0.2, 0.5\}$ with probabilities $(0.5, 0.5)$ respectively.
- The deadlines are chosen from $\{5, 10, 15, 20\}$ with probabilities $(0.1, 0.1, 0.4, 0.4)$ respectively.
- The period is equal to $T = 20$ for all devices.

In the scenario with offsets, the offsets are chosen uniformly in $[0, T]$. Otherwise, they are all set to 0. In the following, the figures show the mean and standard deviation over multiple seeds.

A. Benchmarks

To evaluate the performance of our algorithm, we benchmark it against the following algorithms:

- **Random agent**: schedules a device uniformly at random.
- **Round Robin agent**: schedules devices in a cycle so that the resource is shared equally among the devices.
- **Slotted ALOHA**: all devices can access the medium with the same probability p. As it is not trivial to derive the optimal probabilities analytically, we set the transmission probabilities experimentally, such that the throughput is maximal.
- **Matching Agent**: in the scenario where all offsets are equal to 0, it is possible to derive the optimal scheduler if the statistics of the model are known (arrival probabilities, constraints, period). We call this algorithm the **Matching agent**. The idea is to model the scheduling problem as a *bipartite graph matching* problem where we need to match N devices with T slots. There is an edge between a device i and a slot t if the latter is inferior to the device's constraint, i.e. $t < \delta_i$. We set the weight of device i to p_i in order to maximize the throughput. Note that this scheduling method is completely unfair when the number of devices is greater than the number of slots T. To alleviate this issue, we could set the weights in order to maximize an α-fairness objective but this is beyond the scope of this paper and we leave this for future work. In our experiments, we solve the maximum weighted matching problem using the Hungarian algorithm [26].

B. Simulation results

The parameters of our algorithm are given in Table I.

TABLE I
Parameters of the experiments

Parameter	Value
Episode length	50
Discount factor: γ	0.8
Learning rate	0.002
Clipping hyperparameter: ϵ	0.1
Number of epochs	4
Preference parameter: β	0.3
History length H	N

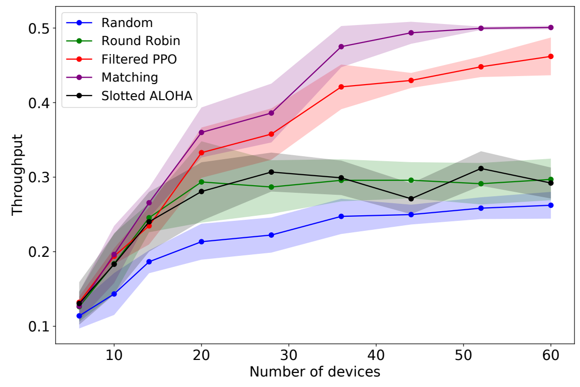

Fig. 4. Evolution of the throughput with the number of devices. The devices are synchronous. The results are computed on 5 seeds.

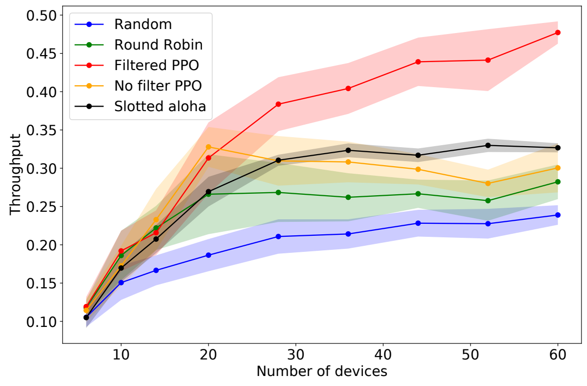

Fig. 5. Evolution of the throughput with the number of devices. The devices are not synchronous. The results are computed on 10 seeds.

We train our algorithm on 200000 timesteps (4000 episodes) and test it on 20000 timesteps (400 episodes). We update the neural network every 200 steps (4 episodes). The neural network weights are initialized with a random orthogonal matrix as described in [27].

Synchronous setting: First, in Fig.4, we test our algorithm on a scenario where all devices are synchronous. This allows us to benchmark it against the optimal scheduler knowing all the environment parameters (matching agent).

Asynchronous setting: Second, we test our algorithm in the more general asynchronous setting. The results are shown in Fig.5, including the version of our proposed algorithm without invalid action masking, called No-filter PPO.

We can notice that in both scenarios, Filtered PPO successfully exploits the heterogeneity in the devices to outperform Round Robin, ALOHA and the Random agent for every number of transmitters in terms of throughput. In the synchronous setting, the performance of our algorithm is close to the performance of the matching algorithm. This is achieved despite the fact that our algorithm is not aware of the traffic probabilities and deadlines. Our algorithm still outperforms Round Robin, ALOHA and Random scheduling in the asynchronous setting where the matching agent cannot be applied. We also note

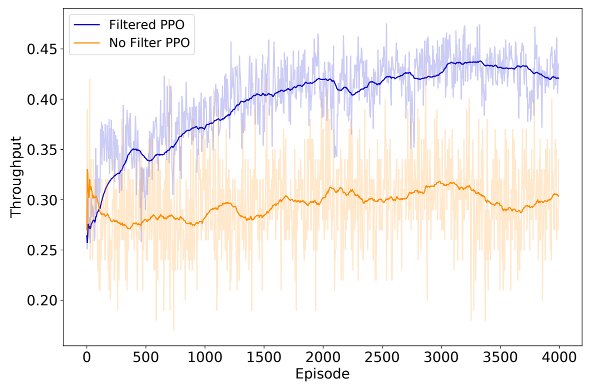

Fig. 6. Evolution of the throughput during the training phase of PPO with and without invalid action masking. The experiment was made with 36 devices and with offsets. The dark curves show the moving average over 70 episodes.

that in the asynchronous setting, ALOHA performs better than Round Robin, unlike the synchronous one, because the probability of having a collision is smaller when devices are not synchronous.

Finally, we show the impact of invalid action masking on the throughput in the asynchronous setting. One the one hand, we can see in Fig.5 that when the number of devices increases, learning a good policy becomes more difficult for No-filter PPO, leading to a lower throughput than Filtered PPO. On the other hand, Fig.6 shows the evolution of the throughput during training for 36 asynchronous devices. We can see that Filtered PPO outperforms No-filter PPO quite fast and keeps increasing this difference until it converges to a high throughput.

V. Conclusion

In this paper, we modelled the centralized uplink multiple access problem with strict deadlines in IoT as a polling problem with imperfect information and proposed a deep RL algorithm for the BS to learn the polling strategy. We extended the Proximal Policy Optimization algorithm to the partially observable setting using an Recurrent Neural Network and improved its performance on a large number of agents with invalid action masking. We showed that our method successfully manages to learn the traffic patterns of the transmitters despite the partial observability of our problem. Numerical results show that our solution outperforms traditional MA protocols and reaches a performance comparable to the performance of an optimal algorithm aware of the traffic characteristics and of the strict deadlines.

A possible extension to the centralized polling approach is to allow the central agent to poll multiple devices at the same time. Even if this would allow potential collisions, we expect the RL agent to learn how to poll clusters of devices with low probability of collision and thus outperform both centralized and decentralized solutions.

References

[1] C. Bockelmann *et al.*, "Towards massive connectivity support for scalable mmtc communications in 5g networks," *IEEE access*, vol. 6, pp. 28 969–28 992, 2018.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] C. Hägerling, C. Ide, and C. Wietfeld, "Coverage and capacity analysis of wireless m2m technologies for smart distribution grid services," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2014, pp. 368–373.

[4] M. A. Al-Garadi *et al.*, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[5] L. G. Roberts, "Aloha packet system with and without slots and capture," *ACM SIGCOMM Computer Communication Review*, vol. 5, no. 2, pp. 28–42, 1975.

[6] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, 2000.

[7] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] A. Tampuu *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[9] H.-H. Chang *et al.*, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1938–1948, 2018.

[10] Y. Xu, J. Yu, and R. M. Buehrer, "The application of deep reinforcement learning to distributed spectrum access in dynamic heterogeneous environments with partial observations," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4494–4506, 2020.

[11] Y. Xu, J. Yu, W. C. Headley, and R. M. Buehrer, "Deep reinforcement learning for dynamic spectrum access in wireless networks," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 207–212.

[12] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, 2019.

[13] S. Bi, R. Zhang, Z. Ding, and S. Cui, "Wireless communications in the era of big data," *IEEE communications magazine*, vol. 53, no. 10, pp. 190–199, 2015.

[14] A. Slivkins, "Introduction to multi-armed bandits," *arXiv preprint arXiv:1904.07272*, 2019.

[15] Z. Yu, Y. Xu, and L. Tong, "Deadline scheduling as restless bandits," *IEEE Transactions on Automatic Control*, vol. 63, no. 8, pp. 2343–2358, 2018.

[16] J. Hribar, A. Marinescu, G. A. Ropokis, and L. A. DaSilva, "Using deep q-learning to prolong the lifetime of correlated internet of things devices," in *2019 IEEE Int. Conf. Commun. Workshops ICC Workshops 2019*. IEEE, 2019, pp. 1–6.

[17] C. Zhong, Z. Lu, M. C. Gursoy, and S. Velipasalar, "Actor-critic deep reinforcement learning for dynamic multichannel access," in *2018 IEEE Glob. Conf. Signal Inf. Process. (GlobalSIP)*. IEEE, 2018, pp. 599–603.

[18] I.-H. Hou and P. R. Kumar, "Packets with deadlines: A framework for real-time wireless networks," *Synthesis Lectures on Communication Networks*, vol. 6, no. 1, pp. 1–116, 2013.

[19] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[21] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] J. Schulman *et al.*, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] J. Schulman *et al.*, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[26] G. A. Mills-Tettey, A. Stentz, and M. B. Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27*, 2007.

[27] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.