A Reinforcement-Learning-Based Access Scheme for Low-Latency and Correlated-Traffic MTC Networks

Duc Tuong Nguyen, Xianyi Zhan, Tho Le-Ngoc
Department of Electrical & Computer Engineering, McGill University, Montreal, QC, Canada
Email: tuong.nguyen2@mail.mcgill.ca; xianyi.zhan@mail.mcgill.ca; tho.le-ngoc@mcgill.ca

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771903

***Abstract*—This paper presents an access scheme for machine-type communication (MTC) networks where the base station (BS) is equipped with a massive antenna array and devices have correlated traffic and delay constraints. We formulate an optimization problem to allocate resources and calculate the access probabilities to maximize the throughput with delay constraints. Since the traffic model parameters and event locations are not available to the BS and the throughput with delay constraints is hard to be derived, we propose a reinforcement-learning-based algorithm to solve the problem. Our simulation reveals that our proposed algorithm is superior to a random scheduling baseline both in terms of throughput and delay. More importantly, our proposed algorithm achieves comparable throughput and lower average delay compared to the algorithm that has full information of traffic model parameters and event locations but optimizes throughput without delay constraints.**

I. Introduction

In 5G-and-beyond wireless networks, the focus will be shifted from human-type communication (HTC) to machine-type communication (MTC) [1]. MTC networks are expected to replace wired connections in future factories via the so-called industrial internet of things (IIoT). However, it is challenging to satisfy IIoT applications requirements with wireless networks due to the massive connectivity and low-latency requirements. In addition, MTC devices usually have correlated traffic due to the possible related events. For example, when the temperature in an area exceeds a threshold, devices in that area will possibly want to send their corresponding reports in relation to this event to the base station (BS). As a result, an instantaneously increased amount of transmission demands may lead to a serious congestion that can impact the packet delivery delay. Therefore, it is necessary to design a multiple access (MA) scheme that can exploit this correlation to grant resources to devices having high chance of having packets to reduce their access delay.

The topic of MA schemes for low-latency communication has gained attention recently. 5G New Radio (NR) proposes grant-free access and short transmission time interval (TTI) to support ultra-reliable-low-latency communication (URLLC) [2], [3]. However, 5G NR approaches are not scalable since grant-free access causes severe collisions, and short TTI requires large bandwidth. In [4], the authors consider grant-based random access (RA) with multiple preamble transmissions to reduce the failure probability due to preamble collisions and the RA response error. However, multiple transmissions of preambles might cause high overhead, affecting the spectral efficiency negatively. Similarly, [5] proposes grant-free RA with multiple retransmissions and calculates the optimal number of retransmissions to achieve the target reliability within the latency deadline. The work in [6] supports the coexistence of massive MTC devices and URLLC devices by dedicating resources to URLLC devices to improve the latency and reliability of URLLC devices. However, the approaches in [5], [6] might be inefficient when the number of URLLC devices is high since resources are limited, and collisions will still be severe even with retransmission. The studies in [7], [8] allow devices to share resources by deriving the access probabilities for regular devices and URLLC devices to satisfy the latency and reliability constraints. However, these probabilities are obtained based on traffic parameters, which might be unknown in several IIoT applications.

Exploiting the traffic statistics of MTC devices for uplink resource allocation to achieve high throughput and low latency has been investigated. The study in [9] proposes a preallocation algorithm that exploits the correlation in the access behaviour of devices to provide low-latency access. However, the proposed method is only suitable to industrial stages where events follow a fixed sequence. The work in [10] proposes an algorithm using reinforcement learning (RL) to schedule sensor updates to minimize the sum of age-of-information (AoI) of all sensors under their AoI constraints. However, in this work, sensors only generate packets when they are selected by the controller, while in event-driven applications, the packet arrival is decided by the events. The work in [11] proposes a reconfigurable access scheme where the time frame has a grant-based segment where devices are reserved resources and a grant-free segment where devices contend for resources. The BS decides the device allocation in the grant-based segment and the access probabilities based on the non-empty queue probabilities, which can be learned by a bandit-based algorithm developed in [12]. However, [11], [12] do not consider the delay requirements, which are essential to serve IIoT applications.

The existing studies rarely consider the low-latency access with massive device deployment and correlated MTC traffic.

To address the requirements of practical IIoT networks, we consider MTC networks having a large number of devices with correlated traffic and different delay requirements. We propose an RL-based algorithm to maximize throughput while minimizing the delay violation in the absence of device traffic parameters and event location information at the BS. Using an offline RL algorithm, the BS can train the RL agent during the system operation and correct its output action to ensure safe operation during the training process. Moreover, our algorithm does not require a large initial dataset as supervised learning methods.

II. System Model

We consider a multipoint-to-point MTC network consisting of one BS and D devices. In particular, we focus on the uplink data transmission from devices to the BS since most applications in MTC networks involve reporting information to the BS. The BS is equipped with M antennas while each device has only one antenna since devices in MTC networks are usually size- and power-limited.

In order to estimate the channel state information (CSI) at the BS, each device wishing to transmit their data has to transmit a preamble beforehand. There are $N_{\rm p}$ orthogonal preambles available in the network, and the BS periodically informs them to devices. When receiving data from multiple devices transmitting in the same time slot, the BS uses beamforming to distinguish the data of each device from the others. Moreover, all devices utilize power control so that the expected received power at the BS is the same for all of them.

A. Reconfigurable Access Scheme

The network operates on a frame-by-frame basis. Each time frame is started with a beacon followed by $N_{\rm ts}$ time slots, divided into two segments, as illustrated by Figure 1.

Beacon: At the start of each time frame, the BS broadcasts a beacon including the set of devices allocated to the grant-based segment, their allocated time slots and preambles, and the access probabilities of other devices.

Grant-based: In this segment, using information broadcast in the beacon, granted devices transmit their packets in the corresponding allocated time slots. Up to $N_{\rm p}$ devices can be granted the same time slot, and each device in this segment will be allocated a preamble. Devices transmit their assigned preambles first, followed by their data payload. The BS uses the estimated device channel information for beamforming to cancel interference for the data of each device. In this segment, transmission is considered successful if the signal-to-interference-and-noise ratio (SINR) after beamforming is larger than or equal to a threshold value, $\gamma_{\rm th}$. The length of this segment is $N_{\rm gb}$ ($\leq N_{\rm gb,max}$) time slots.

Grant-free: In this segment, devices that have packets for transmission but are not granted a time slot contend with each other. If every device in this segment acts greedily by utilizing the access probability $p = 1$ when it has packets to transmit, severe packet collisions will happen, resulting in high packet delay. To address this problem, p-persistent ALOHA is used as follows [13]. In each time slot of this segment, using the access probability p broadcast in the beacon, a device having packets in its queue randomly chooses a preamble from $N_{\rm p}$ available ones and transmits this preamble followed by its packets.

The BS also uses the received preamble signal to estimate the device channel information and then uses the device channel estimation for beamforming to cancel interference for the data of each device. The transmission of this device will be considered successful if its chosen preamble is not selected concurrently by other devices and its SINR is not smaller than the threshold $\gamma_{\rm th}$. At each frame, after one successful transmission, the device is not allowed to transmit any other packets. The length of the grant-free segment is $N_{\rm gf} = N_{\rm ts} - N_{\rm gb}$.

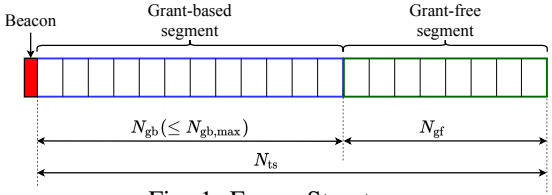

Fig. 1: Frame Structure.

B. Traffic Model

We assume the traffic model of each device to be the two-state Markov chain presented in [14]. This Markov chain has two states: regular and alarm, as illustrated in Figure 2. Each device d operates in the regular state with a packet arrival rate of $R_{{\rm R},d}$ until it is triggered by events and changes into the alarm state with a packet arrival rate of $R_{{\rm A},d}$. Devices are uniformly distributed in the cell area, and events are assumed to be Poisson spatial processes with density $\lambda_{\rm E}$. Call $\mathbf{\Phi}_{\rm E}(t)$ the set of events that happens at time frame t, the probability of going to the alarm state at time frame t, $p_{\mathbf{x}_d}(t)$, depends on the distances from the location of the device, $\mathbf{x}_d$, to the epicenter of each event, $\mathbf{y} \in \mathbf{\Phi}_{\rm E}(t)$ and is calculated as

$$p_{\mathbf{x}_d}(t) = 1 - \prod_{\mathbf{y} \in \Phi_{\rm E}(t)} (1 - p_{\mathbf{x}_d \mathbf{y}}), \tag{1}$$

where $p_{\mathbf{x}_d \mathbf{y}} = \exp(-\|\mathbf{x}_d - \mathbf{y}\|^2)$. At the beginning of each time frame, device d generates a packet with probability $R_{{\rm R},d}$ or $R_{{\rm A},d}$ depending on its current state. The generated packet is added to the queue of the device. In addition, each device has a latency deadline $L_{d,\max}$ for packet delivery.

Based on the Markov model illustrated in Figure 2, the steady-state probability vector of the Markov chain of device d at time frame t is $[\pi_{{\rm R},\mathbf{x}_d}(t)\ \pi_{{\rm A},\mathbf{x}_d}(t)]$, where $\pi_{{\rm R},\mathbf{x}_d}(t) + \pi_{{\rm A},\mathbf{x}_d}(t) = 1$. In this vector, $\pi_{{\rm R},\mathbf{x}_d}(t)$ and $\pi_{{\rm A},\mathbf{x}_d}(t)$ are the probabilities of being in the regular state and alarm state, respectively. The values of these probabilities are

$$\pi_{{\rm R},\mathbf{x}_d}(t) = \frac{1 - b_d(t)}{1 + p_{\mathbf{x}_d}(t) - b_d(t)}, \pi_{{\rm A},\mathbf{x}_d}(t) = \frac{p_{\mathbf{x}_d(t)}}{1 + p_{\mathbf{x}_d}(t) - b_d(t)}. \tag{2}$$

The expected packet arrival rate of device d is

$$R_d(t) = \frac{1 - b_d(t)}{1 + p_{\mathbf{x}_d}(t) - b_d(t)} R_{{\rm R},d} + \frac{p_{\mathbf{x}_d}(t)}{1 + p_{\mathbf{x}_d}(t) - b_d(t)} R_{{\rm A},d}. \tag{3}$$

In this traffic model, the traffic of devices is spatially correlated. Devices that are closer to events will have high values of $p_{\mathbf{x}_d}(t)$ and from (2) and (3), their expected arrival rates will be higher than devices that are far from the events. Therefore, devices that are close to each other have high correlation in their traffic profiles. Once an event happens and we receive data packets from some devices, this spatial correlation means that nearby devices have high probabilities of having packets and radio resources should be allocated to these devices to reduce latency and improve throughput.

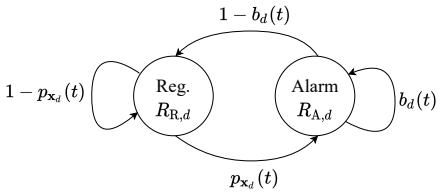

Fig. 2: State transition diagram of the traffic model.

Each time a device d successfully transmits a packet, it piggybacks an extra bit, $q_d(t)$, reporting whether its queue is empty ($q_d(t) = 0$) or non-empty ($q_d(t) = 1$, i.e., it has packets backlogged in the queue to transmit).

C. Channel Model

The channel model under consideration is an uncorrelated Rayleigh fading channel, and the channel response is assumed to be unchanged within each time slot. The channel response vector between device d and the BS is $\mathbf{g}_d = \sqrt{\ell_d}\mathbf{h}_d \in \mathbb{C}^M$. ℓ_d is the large-scale coefficient of the channel between device d and the BS. $\mathbf{h}_d \sim \mathcal{CN}(0, \mathbf{I_M})$ is the small-scale fading vector of the channel between device d and the BS.

D. Overview of Zero-Forcing Beamforming

This section is to review zero-forcing (ZF) beamforming and the corresponding formulas to calculate the SINR of each device as provided in [13]. Call $\mathcal{N}_s$ the set of devices transmitting in time slot s, following [13], we will present the formula of SINR of the first device in $\mathcal{N}_s$ with channel vector $\mathbf{h}_1$ as an example. Define $\mathcal{Q} = \{1, 2, ..., Q\}$ the set of preambles chosen by all the transmitting devices *other than* device 1, $\mathcal{W}_q$ the set of devices selecting preamble $q \in \mathcal{Q}$ and w_q an arbitrary element in $\mathcal{W}_q$. In addition, call $\mathbf{a}_q = \sum_{d\in\mathcal{W}_q} \mathbf{h}_d$, $\mathbf{A} = [\mathbf{h}_1, \mathbf{a}_2, ..., \mathbf{a}_Q]$ and $\mathbf{B} = (\mathbf{A}^{\mathrm{H}}\mathbf{A})^{(-1)}\mathbf{A}^{\mathrm{H}}$, the SINR of device 1 is as follows

$$\gamma^1_{\mathrm{ZF}} = \frac{\rho_{\mathrm{R}}}{\rho_{\mathrm{R}}\left|\sum_{d\in\mathcal{W}_q\setminus w_q : s\in\mathcal{Q}} \sqrt{2}\mathbf{b}_1^{\mathrm{T}}\mathbf{h}_d\right|^2 + ||\mathbf{b}_1^{\mathrm{T}}||^2}, \quad (4)$$

where $\mathbf{b}_1^{\mathrm{T}}$ is the first row of the matrix $\mathbf{B}$, ρ_R is the received signal-to-noise ratio (SNR) of devices, and the notation $\mathcal{W}_q \setminus w_q$ means the set $\mathcal{W}_q$ excluding the element w_q.

III. Problem Formulation

Let $\mathbf{X} : D \times N_{\mathrm{ts}}$ be the time slot allocation matrix, where $x_{d,s} = 1$ indicates device d is assigned to time slot s and $x_{d,s} = 0$ otherwise. If $x_{d,s} = 0\ \forall s$, device d is allocated to the grant-free segment. Let $\mathbf{P} = [p_d]_{\forall d}$ be the vector containing the access probability of each device. Define a vector $\mathbf{Z} = [z_d]_{\forall d}$, where $z_d = \sum_{s=1}^{N_{\mathrm{ts}}} x_{d,s}$ indicates whether any time slot is allocated to device d or not. At the beginning of each time frame t, the BS has to calculate the allocation matrix $\mathbf{X}$ and the access probability vector $\mathbf{P}$ that solves the following optimization problem

$$\underset{\mathbf{X},\mathbf{P}}{\text{maximize}} \quad S(t) = S'(t) - \sum_{d\in D_{\mathrm{r}}(t)} \chi(L_d(t), L_{d,\max}),$$

subject to:

$$\begin{aligned}
&\text{C5.1: } x_{d,s} \in \{0,1\},\ \forall d, s,\\
&\text{C5.2: } z_d p_d = 0,\ \forall d,\\
&\text{C5.3: } z_d \in \{0,1\},\ \forall d,\\
&\text{C5.4: } 0 \le p_d \le 1,\ \forall d,\\
&\text{C5.5: } N_{\mathrm{gb}} \le N_{\mathrm{gb,max}},
\end{aligned} \quad (5)$$

where $S(t)$ is the throughput with delay constraints of time frame t, defined as the total number of packets received within their latency deadlines at time frame t. $S'(t)$ is the total throughput of the grant-based and grant-free segments of time frame t and is calculated as the total number of packets received during the time frame. $D_{\mathrm{r}}(t)$ is the set of devices successfully transmitting their packets at time frame t and $L_d(t)$ is the latency of the packet transmitted by device d at time frame t. $\chi(x, y)$ is a function indicating whether x is larger than y or not (i.e. $\chi(x, y) = 1$ if $x > y$ and $\chi(x, y) = 0$ otherwise). Condition C5.2 ensures that each device is only assigned to either grant-based or grant-free segment. Condition C5.3 guarantees that each device is allocated at most one time slot and condition C5.4 ensures the access probability is between 0 and 1. Finally, condition C5.5 is to limit the length of the grant-based segment to $N_{\mathrm{gb,max}}$ time slots.

IV. Proposed Reinforcement-Learning-Based Algorithm

In (5), it is hard to derive an expression of the objective function due to the delay term and the unknown traffic model parameters, as well as the unavailability of the event positions at the BS. Thus, the scheduling algorithm needs to learn and exploit the traffic correlation to schedule transmissions. For our problem, supervised learning is inapplicable since it requires a long time to collect samples at the beginning. Therefore, we propose using RL since an RL agent can be trained during the system operation and as a result, the initial data collection time is removed. In addition, we use offline RL since it allows us to adjust the action by piggyback bits to ensure safe operation and better convergence.

RL problems are usually modeled as a Markov decision process (MDP), which consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a set of rewards $\mathcal{R}$. A policy is a function mapping the state $\mathbf{s}$ to the action $\mathbf{a}$ and is generally represented as a conditional probability distribution $\pi(\mathbf{a}|\mathbf{s})$. At time frame t, the RL agent, deployed at the BS, observes an input state vector $\mathbf{s}_t \in \mathcal{S}$ and uses the policy π to output an action vector

$\mathbf{a}_t \in \mathcal{A}$. The action $\mathbf{a}_t$ is taken in the environment, and the environment, E, returns an immediate reward r_t and next state vector $\mathbf{s}_{t+1}$ as a result of the action.

In our work, to learn the traffic correlation and maintain the algorithm scalability, the RL agent considers the queuing and delay status of all devices from the state vector and reward, and outputs $\mathbf{X}$ and $\mathbf{P}$ as its action. However, this action will have $D \times N_{\rm ts} + D$ elements, which might be too large and thus, will negatively affect the convergence. In addition, $\mathbf{X}$ and $\mathbf{P}$ directly yielded by neural networks might be infeasible. Therefore, we reduce the action space to D elements by making the RL agent output a value $\theta_d(t) \in [0, 1]$ for each device d. This value is used as the non-empty queue probability of device d, and Algorithm 1 of [11] is used to calculate the value of $\mathbf{X}$ and $\mathbf{P}$. This is an iterative algorithm that incrementally increases $N_{\rm gb}$, allocates devices having high non-empty queue probabilities to the grant-based segment and calculates the access probabilities in the grant-free segment for the rest of devices until the estimated throughput cannot be further increased or $N_{\rm gb}$ reaches $N_{\rm gb,max}$. In other words, the state, action and reward are selected as follows:

- **Input state vector**: Let $v_d(t)$ denote the last time that the BS receives a packet from device d, and $v_{{\rm c},d}(t)$ denote the creation time of that packet. The input state of each device contains the duration from the last time the BS receives a packet from the device until the current time frame, the duration from the creation time of that packet until now, the latency deadline of that device and the piggyback bit of the device. In other words, the input state vector is $\mathbf{s}_t = [t - v_d(t), t - v_{{\rm c},d}, L_{d,\max}, q_d(t)]_{\forall d}$.
- **Action vector**: For each device d, the RL agent outputs a value $\theta_d(t) \in [0, 1]$. This value is interpreted as the non-empty queue probability of device d and used in Algorithm 1 of [11] to calculate the value of $\mathbf{X}$ and $\mathbf{P}$. In other words, the action vector is $\mathbf{a}_t = [\theta_d(t)]_{\forall d}$.
- **Reward**: The reward is the throughput with delay constraints of time frame t: $r_t = S(t)$.

The goal of RL is to maximize the return $G_t = \sum_{i=t}^{T} \gamma^{(i-t)} r_i$, with a discount factor $\gamma \in [0, 1]$ The action-value function, called Q-function, is the expected return achieved by an action $\mathbf{a}_t$ when the system is in state $\mathbf{s}_t$ and uses policy π:

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{r_{i\geq t}, \mathbf{s}_{i>t} \sim E, \mathbf{a}_{i>t} \sim \pi}[G_t | \mathbf{s}_t, \mathbf{a}_t]. \tag{6}$$

The Q-function satisfies the following recursive relationship, known as the Bellman equation:

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{r_{i\geq t}, \mathbf{s}_{i>t} \sim E}[r_t + \gamma \mathbb{E}_{\mathbf{a}_{i>t} \sim \pi}[Q^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})]]. \tag{7}$$

In this work, we use the deep deterministic policy gradient (DDPG) algorithm [15]. This algorithm uses a deterministic policy so it can be described as a function $\mu : \mathcal{S} \rightarrow A$ and the inner expectation is avoided:

$$Q^{\mu}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{r_{i\geq t}, \mathbf{s}_{i>t} \sim E}[r_t + \gamma Q^{\mu}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})]. \tag{8}$$

The DDPG algorithm uses two neural networks, an actor network and a critic network. The actor network with weights ϕ^{μ} represents the current policy and is denoted as $\mu(\mathbf{s}_t|\phi^{\mu})$. The critic network with weights ϕ^{Q} represents the Q-function and is denoted as $Q(\mathbf{s}, \mathbf{a}|\phi^{Q})$. The algorithm also uses two other neural networks, $\mu'(\mathbf{s}_t|\phi^{\mu'})$ and $Q'(\mathbf{s}, \mathbf{a}|\phi^{Q'})$, which are the copies of $\mu(\mathbf{s}_t|\phi^{\mu})$ and $Q(\mathbf{s}, \mathbf{a}|\phi^{Q})$. $\mu'(\mathbf{s}_t|\phi^{\mu'})$ and $Q'(\mathbf{s}, \mathbf{a}|\phi^{Q'})$ are called target networks and are slowly updated to the actor and critic networks, respectively. Additionally, the algorithm uses a replay buffer $\mathcal{B}$, which stores up to $N_{\rm B}$ previous transitions $(\mathbf{s}_i, \mathrm{a}_i, r_i, \mathbf{s}_{i+1})$. At each training iteration, a batch of $N_{\rm e}$ transitions is randomly sampled from $\mathcal{B}$ to train the actor and critic networks. The critic network is trained to minimize the following loss function:

$$L = \frac{1}{N_{\rm e}} \sum_{i} (y_i - Q(\mathbf{s}_i, \mathbf{a}_i|\phi^{Q}))^2, \tag{9}$$

where $y_i = r_i + Q'(\mathbf{s}_{i+1}, \mu'(\mathbf{s}_{i+1}|\phi^{\mu'})|\phi^{Q'})$. The actor network is trained to maximize the following function:

$$J(\phi^{\mu}) = \mathbb{E}_{s_i \sim \mathcal{B}}[Q(\mathbf{s}_i, \mu(\mathbf{s}_i|\phi^{Q})|\phi^{\mu})]. \tag{10}$$

The gradient of this function is as follows:

$$\nabla_{\phi^{\mu}} J \approx \frac{1}{N_{\rm E}} \sum_{i} \nabla_a Q(\mathbf{s}, \mathbf{a}|\phi^{Q}) \nabla_{\phi^{\mu}} \mu(\mathbf{s}|\phi^{\mu})|_{\mathbf{s}=\mathbf{s}_i, \mathbf{a}=\mu(\mathbf{s}_i)}. \tag{11}$$

In order to improve the convergence of the algorithm and ensure safe operation of the access scheme during training, we propose to adjust the output action of the agent by using the piggyback bit as follows. For each device d, if $q_d(t) = 1$, we set $\theta_d(t) = 1$ regardless of the value outputted by the RL agent. This adjustment is necessary since it ensures that devices that piggyback a non-empty queue should be allocated resources. The algorithm to train the RL agent to solve (5) is summarize in Algorithm 1.

V. Illustrative Results

In order to evaluate the proposed learning-based algorithm, simulations are carried out in Python and neural networks are created and trained by Tensorflow 2.4.1. We randomly generate the length and the timing of events as follows. Each time events happen, their duration is randomly generated by an exponential distribution with parameter $\lambda_{\rm L}$. When events end, the length of time until events happen again is generated by an exponential distribution with parameter $\lambda_{\rm T}$. The latency deadlines are set as follows. The first 50 devices have a latency deadline of 4 time frames. The next 150 devices have a latency deadline of 10 time frames. The rest of devices have a latency deadline of 12 time frames. Other simulation parameters are summarized in Table I. The simulation results are shown in Figures 3-5. For simulations, throughput is expressed in units of number of packets per time frame (pckt/tf).

The actor network is fully-connected (FC) and has one input layer, three hidden layers and one output layer. The input and hidden layers are of size $D \times 2$ and use rectified linear unit (ReLU) activation function, $ReLU(x) = max(0, x)$, to avoid the vanishing gradient problem [16]. The output layer is of size

Algorithm 1 RL Algorithm to solve (5).

Randomly initialize the critic network $Q(\mathbf{s}, \mathbf{a}|\phi^Q)$ and actor network $\mu(\mathbf{s}|\phi^\mu)$ with weights ϕ^Q and ϕ^μ.
Initialize target network Q' and μ' with weights.
Initialize the replay buffer $\mathcal{B}$ and set the maximum number of time frames to train the RL agent, $T_{\max}$.
for $t = 1, 2, .., T_{\max}$ **do**
  Obtain the output action of the RL agent: $\mathbf{a}_t = \mu(\mathbf{s}_t|\phi^\mu)$.
  for $d = 1, 2, .., D$ **do**
    $\theta_d(t) \leftarrow \theta_d(t) + \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(0, \sigma^2)$ is Gaussian noise with zero mean and σ^2 variance.
    if $q_d(t) = 1$ **then**
      $\theta_d(t) \leftarrow 1$.
    end if
  end for
  Using $\mathbf{a}_t = [\theta_d(t)]_{\forall d}$ as non-empty queue probabilities, use Algorithm 1 of [11] to obtain $\mathbf{X}$ and $\mathbf{P}$.
  Using $\mathbf{X}$ and $\mathbf{P}$ to configure transmission at time frame t and obtain the throughput with delay constraints $S(t)$ as reward r_t and the next state $\mathbf{s}_{t+1}$. Store the transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ into the replay buffer $\mathcal{B}$.
  Sample a random batch of N_e transitions $(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_{i+1})$.
  Update the critic network by minimizing the loss in (9).
  Update the actor network by maximizing the function in (10) by using the gradient of the function in (11).
  Update the target network weights as
$$\phi^{Q'} \leftarrow \tau\phi^{Q'} + (1-\tau)\phi^Q$$
$$\phi^{\mu'} \leftarrow \tau\phi^{\mu'} + (1-\tau)\phi^\mu$$
end for

Cell radius	10m
D, M, N_p	316, 70, 20
λ_E	0.14 events/m^2
$R_{R,d}$, $R_{A,d}$	0.1, 1 for all devices
$b_d(t)$	$p_{\mathbf{x}_d}(t)$
λ_L, λ_T, $T_{\max}$	100, 50, 7000 time frames
ρ_R, γ_{th}	8, 8 dB
N_{ts}, $N_{gb,\max}$	10, 6 time slots
γ, σ^2	0.5, 0.0625
N_B, N_e	5000, 1024 transitions

TABLE I: Simulation parameters.

D and uses the sigmoid activation function $\sigma(x) = 1/(1 + e^{-x})$ so that its outputs can be interpreted as probabilities. For the critic network, the input state vector is passed through one FC layer of size 32, two hidden layers of size 64. The action vector is passed through one FC input layer and one FC hidden layer of size 64. The output of the two networks are concatenated and passed through three additional hidden layers of size $D \times 2$ before the output layer of size 1. All layers of the critic network use the ReLU activation function.

In order to show the effectiveness of the proposed RL algorithm, we compare it with three other agents:

- *Known-Statistics agent*: The BS of this agent knows all the parameters of the Markov chain and the positions of events. From this information, the BS calculates the non-empty queue probability of each device as follows:

$$\theta_d(t) = \begin{cases} 1 - \prod_{t'=v_d(t)+1}^{t} (1 - R_d(t')), & \text{if } q_d(v_d(t)) = 0 \\ 1 & \text{if } q_d(v_d(t)) = 1. \end{cases} \quad (12)$$

With this non-empty queue probability, Algorithm 1 in [11] is used to calculate the value of $\mathbf{X}$ and $\mathbf{P}$ to maximize throughput without delay constraints, $S'(t)$.
- *Random agent*: The BS of this agent does not know the Markov chain parameters and the event locations. The non-empty queue probability of device d is randomly generated from the continuous uniform distribution $U(0, 1)$ if $q_d(t) = 0$, and is set to 1 if $q_d(t) = 1$.
- *RL without action adjustment (RLWA)*: This agent uses the same proposed RL algorithm as the RL agent except it does not adapt the action adjustment via piggyback bits.

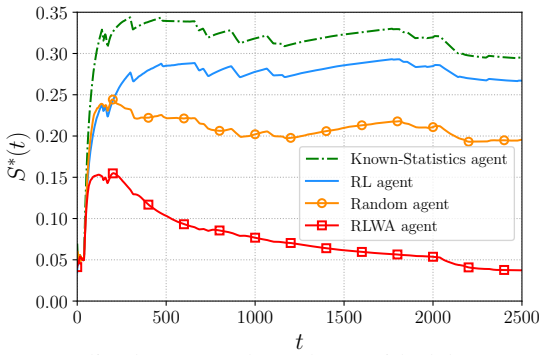

Fig. 3: Normalized average throughput with delay constraints at each time frame t.

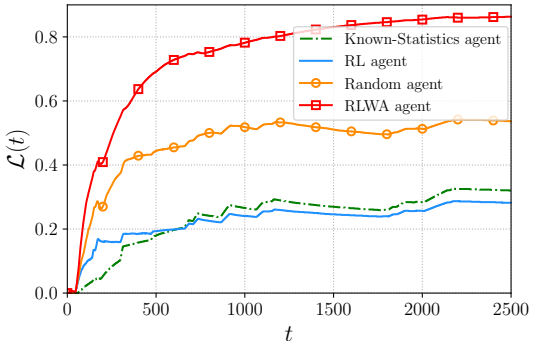

Fig. 4: Accumulated delay ratio at each time frame t.

In Figure 3, normalized averaged throughput with delay constraints, $S^*(t)$, over time is plotted. $S^*(t)$ is calculated as follows:

$$S^*(t) = \frac{1}{DT_w} \sum_{\tau=t-T_w}^{t} S(\tau), \quad (13)$$

where T_w is a time window to average the throughput. Figure 3 shows that the throughput with delay constraints of the RL agent is lower than the Random agent for the first 200 time frames. However, after that, the RL agent achieves higher throughput with delay constraints than the Random agent since it has learned a good policy. The RLWA agent has the lowest throughput with delay constraints since it does not have the

action adjustment so it converges to a bad local minimum, which is confirmed by Figure 5. Overall, the RL agent can achieve 86% to 95% of throughput with delay constraints of the Known-Statistic agent. In comparison, the Random agent and the RLWA agent can only provide 60% to 75% and 10% to 50% of the Known-Statistic agent, respectively.

In Figure 4, the accumulated delay ratio $\mathcal{L}(t)$ is plotted. $\mathcal{L}(t)$ is the ratio of the number of delay-violating packets to the number of all received packets and is calculated as follows

$$\mathcal{L}(t) = \frac{\sum_{\tau=0}^{t} \sum_{d \in \mathcal{D}_r(\tau)} \chi(L_d(\tau), L_{d,\max})}{\sum_{\tau=0}^{t} S'(\tau)}. \quad (14)$$

It can be seen that the Random agent has a substantially higher delay ratio than the RL agent and Known-Statistics agent because it has significantly lower throughput than the two agents as shown in Figure 3. Similarly, the RLWA agent has the highest delay ratio among all agents. The RL agent has a higher delay ratio than the Known-Statistics agent at the beginning, however, the RL agent has the lowest delay ratio after 750 time frames. Although the Known-Statistics agent has the highest throughput with delay constraints, it has a higher delay than the RL agent. This is because the Known-Statistic agent only maximizes the total number of packets, $S'(t)$, without the awareness of the delay. Hence, it receives more packets than the RL agents while violating the delay constraints more often than the RL agent.

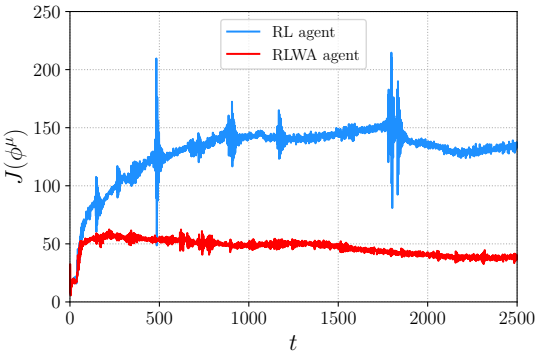

Fig. 5: Value for the function in (10) at each time frame t.

In Figure 5, the values of the actor network function $J(\phi^{\mu})$ over time of the RL and RLWA agents are plotted. The RLWA agent converges to a bad local minimum since it does not have the action adjustment mechanism. This adjustment helps the RL agent to have actions with better rewards to learn from, so it converges to a better local minimum than the RLWA agent. In addition, without this adjustment, there are many packets backlogged in device queues, so the RLWA agent always receives low rewards in later stages of operation regardless of its action, which negatively affects its convergence.

VI. Conclusions

This paper presents an RL-based algorithm to maximize throughput and minimize the delay violations for MTC networks with massive MIMO BS and devices having spatially correlated traffic and delay constraints. We formulate an optimization problem to decide the grant-based segment allocation and calculate access probabilities to maximize throughput with delay constraints. Since the BS does not know the traffic parameters and the expression of the throughput with delay constraints is hard to be obtained, we proposed to solve the problem by an RL algorithm that exploits the correlation in device traffic to intelligently determine the resource allocation and access probabilities. The proposed algorithm allows the BS to train the RL agent during operation while adjusting its output action to allow safe operation and better convergence. Simulation results show the great benefits of this adjustment, and reveal that our proposed RL algorithm has good throughput with delay constraints and lower average delay ratio than the algorithm that has full information of the traffic statistics but is not aware of the delay constraints.

Acknowledgment

This work was supported in part by Huawei Technologies Canada and in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] N. H. Mahmood, H. Alves, O. A. López, M. Shehab, D. P. M. Osorio, and M. Latva-Aho, "Six key features of machine type communication in 6G," in *2020 2nd 6G SUMMIT*, 2020, pp. 1–5.

[2] J. Kim, G. Lee, S. Kim, T. Taleb, S. Choi, and S. Bahk, "Two-step random access for 5G system: Latest trends and challenges," *IEEE Network*, vol. 35, no. 1, pp. 273–279, 2021.

[3] A. Omri, M. Shaqfeh, A. Ali, and H. Alnuweiri, "Synchronization procedure in 5G NR systems," *IEEE Access*, vol. 7, pp. 41 286–41 295, 2019.

[4] E. Kim and H. Lee, "A random access for low latency communications," in *2020 ICTC*, 2020, pp. 1042–1044.

[5] B. Singh, O. Tirkkonen, Z. Li, and M. A. Uusitalo, "Contention-based access for ultra-reliable low latency uplink transmissions," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 182–185, 2018.

[6] S. Moon and J.-W. Lee, "Integrated grant-free scheme for URLLC and mMTC," in *2020 IEEE 3rd 5GWF*, 2020, pp. 98–102.

[7] S. Pandey, K. Shandilya, and S. Agarwal, "Prioritized S-ALOHA for URLLC," in *2020 IWCMC*, 2020, pp. 1842–1847.

[8] R. Qi, X. Chi, L. Zhao, and W. Yang, "Martingales-based ALOHA-type grant-free access algorithms for multi-channel networks with mMTC/URLLC terminals co-existence," *IEEE Access*, vol. 8, pp. 37 608–37 620, 2020.

[9] M. Li, C. Chen, C. Hua, and X. Guan, "A learning-based pre-allocation scheme for low-latency access in industrial wireless networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 650–664, 2020.

[10] A. Elgabli, H. Khan, M. Krouka, and M. Bennis, "Reinforcement learning based scheduling algorithm for optimizing age of information in ultra reliable low latency networks," in *2019 ISCC*, 2019, pp. 1–6.

[11] A. D. Shoaei, D. T. Nguyen, and T. Le-Ngoc, "A reconfigurable access scheme for massive-MIMO MTC networks," *IEEE Access*, vol. 9, pp. 65 547–65 559, 2021.

[12] A. Dalili Shoaei, M. Derakhshani, and T. Le-Ngoc, "Reconfigurable and traffic-aware MAC design for VWNs via reinforcement learning," *IEEE Trans. on Comm.*, vol. 67, no. 8, pp. 5490–5505, 2019.

[13] J. Ding, D. Qu, H. Jiang, and T. Jiang, "Success probability of grant-free random access with massive MIMO," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 506–516, Feb 2019.

[14] H. Thomsen, C. N. Manchon, and B. H. Fleury, "A traffic model for MTC using spatial point processes," in *PIMRC*, 2017, pp. 1–6.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th ICLR*, 2016.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS 2011*, ser. Proceedings of Machine Learning Research, vol. 15. Fort Lauderdale, FL, USA: JMLR Workshop and Conference Proceedings, 11–13 Apr 2011, pp. 315–323.