Packet Drop Probability-Optimal Cross-layer Scheduling: Dealing with Curse of Sparsity using Prioritized Experience Replay

Mohit K. Sharma, *Member, IEEE*, Tan Peng Hui, *Member, IEEE*,
Ernest Kurniawan, *Member, IEEE*, and Sun Sumei *Fellow, IEEE*

***Abstract*—In this work, we develop a reinforcement learning (RL) based *model-free* approach to obtain a policy for joint packet scheduling and rate adaptation, such that the packet drop probability (PDP) is minimized. The developed learning scheme yields an *online cross-layer scheduling policy* which takes into account the randomness in packet arrivals and wireless channels, as well as the state of packet buffers. Inherent difference in the time-scales of packet arrival process and the wireless channel variations leads to sparsity in the observed reward signal. Since an RL agent learns by using the feedback obtained in terms of rewards for its actions, the sample complexity of RL approach increases exponentially due to resulting sparsity. Therefore, a basic RL based approach, e.g., double deep Q-network (DDQN) based RL, results in a policy with negligible performance gain over the state-of-the-art schemes, such as shortest processing time (SPT) based scheduling. In order to alleviate the sparse reward problem, we leverage prioritized experience replay (PER) and develop a DDQN-based learning scheme with PER. We observe through simulations that the policy learned using DDQN-PER approach results in a 3-5% lower PDP, compared to both the basic DDQN based RL and SPT scheme.**

***Index Terms*—Cross-layer scheduling, packet drop probability, prioritized experience replay, reinforcement learning.**

I. Introduction

Minimizing delay in data transmission has been a long standing objective in wireless communications research. Recently emerged applications, such as ultra reliable low latency communications (URLLC) [1], make it even more challenging. The earlier work on delay minimization focused on characterizing the power-delay trade-off [2] and leveraging it through cross-layer design techniques [3], e.g., rate adaptation or power control [4]. In particular, these work aim to minimize queuing delay through a joint queue-aware and channel-aware physical layer scheduling scheme. In order to maintain the energy consumption for data transmission within an acceptable limit, scheduling policies with a provision for dropping a fraction of incoming packets are also studied [5]–[7].

In contrast to the above work, where only a single queue is considered at network layer, the authors in [8] design a joint policy which routes the incoming data over the multiple queues and simultaneously updates the service rate of the queues. However, the goal in this work is to maximize the throughput. In [9], Bodas et al. design a scheduling policy which minimizes the buffer overflow probability. On the other hand, it is well known that when the service rate of the queues are fixed, a policy, termed as *shortest processing time* (SPT) policy, which schedules the incoming data packets to a queue with least expected processing time yields an optimal delay [10]. In general, it is not clear how the delay performance of the SPT policy is affected with the variable service rates. Also, most of the above work aim to minimize the statistical average delay. In contrast, we focus on a scenario where packets have a deadline, beyond which they become obsolete [11].

In this work, we consider a point-to-point system where a transmitter needs to send randomly arriving input data packets to a receiver, within a *fixed time-interval* from their time of arrival. Otherwise, the packets which violate their deadline are dropped. At the transmitter, an incoming data packet is scheduled to a queue, selected from a set of parallel queues. Each queue is served over an orthogonal resource, at a service rate which is determined by the policy adopted by the transmitter. The goal here is to learn an optimal policy which minimizes the packet drop probability (PDP). The policy attempts to minimize the PDP by scheduling input data packets to an appropriate queue and adapting the service rates of the queues, based on the information available about queue lengths and the states of the wireless channels over which queues are being served. The packet arrival process and wireless channels are assumed to follow a stationary but *unknown* distribution.

In order to develop a *model-free, online learning* mechanism we use a deep reinforcement learning (DRL) [12] based approach. In general, due to cross-layer nature of the design problem, the time-scale of evolution for the input data arrival process differs from the coherence time of the wireless channels. While the policy operates at the time-scale matched with the faster varying process, e.g., if the channel varies at a faster time-scale, than the packet arrival process, the service rates of the queues adapted at each time slot whose duration equals the coherence time of the channel. On the other hand, the reinforcement learning (RL) agent observes a reward only when a packet is either dropped or successfully processed. In particular, the reward signal for this joint scheduling problem is sparse. This renders the RL process ineffective, as the sample complexity of RL based learning approach increases exponentially [13]. In order to tackle this issue, we leverage

Mohit Sharma, Tan Peng Hui, Ernest Kurniawan, and Sun Sumei are with the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (E-mails: {mohit_sharma, phtan, ekurniawan, sunsm}@i2r.a-star.edu.sg). This research has been partly supported by the 5G-AMSUS project.

the *prioritized experience replay* (PER) [14] which yields an improved policy, compared to both the conventional RL and SPT policy. Specifically, our contributions in this work are as follows:

1) We develop an online, model-free RL based mechanism to learn a joint scheduling and rate-adaptation policy, and illustrate the impact of the sparse nature of the reward signal on the learning performance.
2) In order to deal with the sparse reward problem, we modify the RL approach by replacing experience replay in the conventional DRL framework with PER.
3) Through simulations, we illustrate that the DRL with PER results in up to 5% lower PDP, compared to both the conventional DRL and SPT.

We note that, in general, our results illustrate the application of PER in presence of sparse rewards which are usually encountered in a wide range of communication systems consisting of multiple random processes varying at different time-scales. For instance, generally, in an energy harvesting communication system, the energy harvesting process and the wireless channel vary at different time-scales [15].

The rest of the paper is organized as follows: in Sec. II we describe the system model and the problem formulation, in Sec. III we present our DRL based solution, in Sec. IV we present simulation results, and present our conclusions in Sec. V.

II. System Model and Problem Formulation

As shown in Fig. 1, we consider a point-to-point communication system where a transmitter and receiver communicate with each other over K parallel data streams, e.g., multiple orthogonal frequency resources. The index of time slots is denoted by n, where $n \in \mathbb{Z}^+$. At the start of the n^{th} slot, a packet of size d_n arrives at the transmitter, after packet inter-arrival duration, i_n. Both the packet size, d_n, and packet inter-arrival duration, i_n, are assumed to be drawn from *stationary and unknown* distributions, denoted by $f_D(d)$ and $f_I(i)$, respectively.

Upon arrival, a packet is immediately routed by the scheduler to a queue, selected from a set of parallel queues, where the packet is served in a first-in first-out fashion. Each data packet is required to be successfully delivered to the receiver within T slots (fixed) from its time of arrival. Otherwise, the packet is *dropped*. For instance, a packet arriving in the n^{th} slot needs to be served completely by the $(n+T)^{\text{th}}$ slot.

Note that, each queue is served over an orthogonal wireless channel, i.e., each given data stream between the transmitter and receiver corresponds to a distinct queue. In the n^{th} slot, the wireless channel between the k^{th} queue and receiver is denoted by $h_k(n)$, where $k \in \{1, \cdots, K\}$. The wireless channels, $\{h_k\}_{k=1}^K$, are assumed to be distributed according a *stationary, unknown distribution* denoted by $f_{H_1,H_2,\cdots,H_K}(h_1, h_2, \cdots, h_k)$. Although, the distribution $f_{H_1,H_2,\cdots,H_K}(\cdot)$ is unknown, the channel states, $h_k(n)$, for all $1 \leq k \leq K$ and n, are known at the transmitter. Without loss of generality, we assume that the wireless channels,

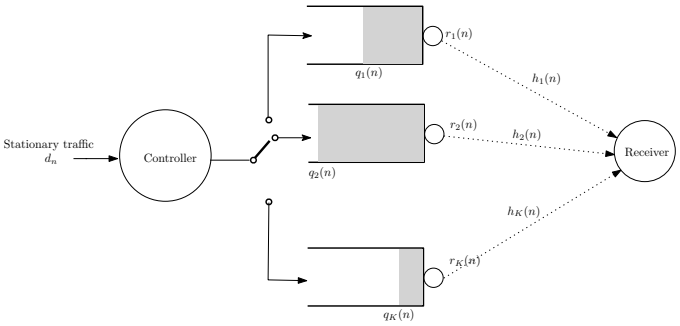

Fig. 1: System model: the data is transmitted over K parallel data streams, e.g., K orthogonal block fading channels. The complex fading channel from the transmitter to receiver, over the k^{th} data stream, in the n^{th} slot is denoted by $h_k(n)$. Further, the length of the k^{th} queue and the size of the data packet arrived in the n^{th} slot are denoted by $q_k(n)$ and d_n, respectively.

between the transmitter and receiver, vary at a *faster time-scale*, compared to the time-scale at which data packets arrive at the transmitter. We consider the slot duration to be equal to the channel coherence time which, in turn, implies that the channels do not change significantly during a slot. Also, for ease of presentation and without loss of generality, we set the channel coherence time to unity.

Furthermore, in each slot, to account for the time-varying channel and backlogged queue data, the service rates of the queues, denoted by $\{r_1(n), \cdots, r_K(n)\}$, are updated by the scheduler. Service rate of each queue is chosen from a fixed, finite set of rates, denoted as $\mathcal{R} \triangleq \{R_1, \cdots, R_M\}$, where $R_1 \leq R_2 \leq \cdots \leq R_M$ and $M \geq K$. The rates R_m, where $1 \leq m \leq M$, are assigned to the queues in a *non-overlapping* manner. In particular, let $\mathbb{1}_m^k(n)$ denote the indicator variable which takes value one if, in the n^{th} slot, the rate R_m is assigned to the k^{th} queue, otherwise it is equal to zero. The non-overlapping rate assignment implies $\sum_{k=1}^K \mathbb{1}_m^k(n) \leq 1$, for all $1 \leq m \leq M$ and n, i.e., in a given slot n a rate, R_m, can only be assigned to a unique queue. In practice, such non-overlapping rate assignment could arise due to limited transmit power which needs to be distributed among the modulation and coding schemes (MCSs) at disposal.

In the n^{th} slot, the service rate of k^{th} queue is given as $r_k(n) = \sum_{m=1}^M \mathbb{1}_m^k(n) R_m$, for all $1 \leq k \leq K$. In the n^{th} slot, the transmission over the k^{th} data stream remains in outage if the rate assigned to the k^{th} queue is greater than the rate supported over the channel $h_k(n)$, i.e., $\gamma_k(n) \log\left(1 + \frac{P_t |h_k(n)|^2}{N_0}\right) < r_k(n)$, where $\gamma_k(n)$ depends on the MCS assigned to the k^{th} queue. Here, P_t denotes the nominal transmit power required to achieve the signal-to-noise ratio (SNR) threshold needed for successful decoding at the receiver, when the wireless channel gain is unity, i.e., $|h|^2 = 1$, and data is transmitted at rate R_1. In the above, N_0 denotes the additive white Gaussian noise (AWGN) power at the receiver. Note that, since a packet stays in the queue, until either it is completely served or it dropped, i.e., the deadline by which the packet needed to be served has expired. Therefore, in the scenario when outage occurs in a slot, the remaining part of

the packet in the queue is served again in the next slot, at the rate assigned to the queue in that slot[1]. Mathematically, the evolution of the total number of bits in the queue which remain to be served can be expressed as follows:

$$q_k(n+1) = \max\{q_k(n) + \mathbb{1}^k_{\{d_n \neq 0\}} d_n - \mathbb{1}^k_{\{\text{no-outage}\}} r_k(n), 0\}, \tag{1}$$

where $q_k(n)$ denotes total number of bits in the queue which remain to be served in the k^{th} queue at the start of the n^{th} slot, and $\mathbb{1}^k_{\{d_n \neq 0\}}$ and $\mathbb{1}^k_{\{\text{no-outage}\}}$ are indicator variables. The variable $\mathbb{1}^k_{\{d_n \neq 0\}}$ is equal to one if and only if there is packet arrival in the n^{th} slot, and it is assigned to the k^{th} queue, otherwise it takes value equal to zero, while the variable $\mathbb{1}^k_{\{\text{no-outage}\}}$ equals one only when the transmission from the k^{th} queue is successful. Note that, (1) is written for the scenario when the queue lengths are infinite. However, (1) and subsequent development in this paper can be easily modified to the case when queue lengths are finite.

The goal in this work is to design a mechanism to facilitate in learning a policy which minimizes the probability that a packet is dropped, i.e., *to improve the reliability of the link subject to a hard delay deadline*. For the system described above, in a given slot, a policy, $\mathbf{\Pi}$, needs to determine the queue index to which the packet arriving in that slot is assigned to, and update the rates of the queues, $\{r_k(\cdot)\}_{k=1}^K$, subject to the non-overlapping rate assignment constraint, $\sum_{k=1}^K \mathbb{1}^k_m(\cdot) \leq 1$, for all $1 \leq m \leq M$. The above objective can be written as

$$\min_{\mathbf{\Pi}} \limsup_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}^k_D(n), \tag{2}$$

subject to non-overlapping rate assignment constraint. Here, $\mathbb{1}^k_D(\cdot)$ denotes an indicator function which is equal to one, if the deadline for the head-of-the-line packet of the k^{th} queue has expired in the current slot; otherwise it takes value zero. Note that, in order to achieve the above objective, the policy has information about the system state in current slot as well the system state in previous slots. In any given slot, the *system state* comprises of the queue lengths in the current slot, $\{q_k(\cdot)\}_{k=1}^K$, states of the wireless channels, $\{h_k(\cdot)\}_{k=1}^K$, and the size of the packet arriving in that slot d_n. However, we emphasize that, the distributions of the packet arrival process and the wireless channels are not know. Therefore, to learn an optimal online policy for the above system, we adapt a model-free DRL approach. In the next section, we describe our DRL based solution, the shortcomings associated with it, arising due to sparsity of the reward signal, and a PER based DRL approach to tackle the problem of sparse rewards.

III. DRL BASED POLICY AND PER

In this section, before describing our PER based solution for mitigating the sparse reward problem, we first provide a brief introduction to double deep Q-network (DDQN) RL which is necessary for highlighting the challenges encountered due to sparse reward problem, and succinctly presenting our PER based solution.

A. Brief Introduction: DDQN-RL

The Q-learning uses the Q-function, denoted by $Q(S, a)$, which quantifies the cost-to-go from state S when action a is taken. Therefore, provided that the accurate estimate of Q-function is available, the optimal policy, with high probability, chooses an action which yields in the least future cost starting from state S. In the conventional form, the Q-function is expressed in a tabular form, by discretizing the set of states and actions, and estimated recursively. However, the need for discretization of the state and action sets renders it difficult to deal with large state spaces. In order to deal with large state space, usually, a parameterized function approximation for Q-function is used and, in turn, the Q-function is learned through estimating the optimal parameter value.

The deep reinforcement learning leverages the fact that the deep neural networks (DNN) are the universal function approximators, to use them for learning an accurate approximation for the Q-function. The DNN is trained progressively using the data generated through agent's interactions with the environment. In particular, a typical data point for DRL is given by a tuple $(S_n, a_n, S_{n+1}, \mathcal{R}_n)$, representing a transition, i.e., if in state S_n the agent takes an action a_n then a reward $\mathcal{R}_n$ is generated, and the system transits to state S_{n+1}. These transitions are stored in a data buffer, termed as *experience replay buffer*, according to their time of arrival; when the buffer is full, more recent data points are accommodated in the data buffer by removing earlier data points. To train the DNN, a bunch of transitions, equal to the batch size, are chosen through *uniform random sampling* from the experience replay buffer. This process of using past observations of the system for the training purpose is know as *experience replay*. For training, the reference Q-values are provided by another DNN, termed as target Q-network, whose weights are periodically updated with the weights of the main Q-network.

The Q-learning algorithm tend to overestimate the Q-values, due to noise, which adversely affects the performance. The double deep Q-learning improves upon conventional deep Q-learning by resolving this over estimation problem, and provides the state-of-the-art performance. For further details on the deep Q-learning and double deep Q-learning we refer the readers to [12] and [16], respectively.

First, we solve the optimization problem described in the previous section using the DDQN-RL. In the n^{th} slot, the state of our system is given by $S_n \triangleq \{\{q_k(n)\}_{k=1}^K, \{h_k(n)\}_{k=1}^K, d_n\}$ which forms the input to Q-network, and the outputs of the Q-network provide an estimate for the Q-values for all the actions, one corresponding to each action. The details of the DDQN-RL algorithm are provided in the next section. However, as observed through simulation in the next section, the DDQN-RL algorithm does not yield a satisfactory performance improvement over the

[1]Strictly speaking, this would require to fragment an original packet into multiple short packets whose sizes are determined based on the modulation and coding scheme assigned to the queue, across the slots over which a given packet is served.

SPT scheduling scheme. Although the RL based schemes are known to achieve a near-optimal performance, the DDQN-RL algorithm fails to achieve this. The reason for this is explained in the following.

B. Curse of Sparsity

From (2), we observe that the reward signal in each slot is given by the indicator variable $\mathbb{1}_D^k(n)$. Further, the slot duration is determined according to faster time-scale variation in the system; in our case, this is equal to the channel coherence time. Therefore, a policy $\mathbf{\Pi}$ takes actions at the start of every slot. Assuming that the policy $\mathbf{\Pi}$ achieves a reasonable performance, the packet drops are observed very rarely, i.e., *the reward signal is zero for most of the transitions*. For example, consider a scenario where a policy achieves approximately 10% packet drop rate, channel coherence time is 5 msec., and the average processing time for a packet is approximately 300 msec. In this case, on an average, a non-zero reward is observed once after every 600 slots. Therefore, majority of the transitions in the experience replay buffer have the value of the reward equal to zero, i.e., the transitions with non-zero rewards are scarce. Intuitively, since in the RL process the reward signal acts as a feedback for the chosen action, the transitions with reward equal to zero provide very little information for the learning purpose. Moreover, for training purpose, the transitions are sampled uniformly randomly. That is, no priority is given to the transitions which contribute more to the learning of the RL agent. As a consequence, for training, the most recent experience might be chosen much later from its time of arrival, or a very important transition is picked only once. All these factors result in an exponentially high sample complexity, or very slow learning rate. This explains the reason for sub-optimal PDP performance of DDQN-RL, even after a very long training (see Fig. 3).

In order to resolve this issue, we employ the prioritized experience replay, where the data points chosen for training are prioritized based on their informational significance for the RL agent. In the next section, we describe the PER which results in an improved performance, over both the DDQN-RL and SPT, due to efficient learning.

C. Prioritized Experience Replay

The basic idea underlying the PER is that some transitions contribute more to the learning of an RL agent, and the importance of a given transition, from the perspective of contribution towards the learning of the RL agents, varies as the learning progresses. In particular, the PER proposes to more frequently pick the transitions, during the sampling from the experience replay buffer, which are expected to result in high learning progress. One possible measure which captures the usefulness of a transition, from the RL agent's learning point, is the temporal-difference (TD) error which is expressed as [14]

$$\delta_n = \mathcal{R}_n + \gamma Q_{\text{target}}\left(S_n, \arg\max_A Q(S_n, A)\right) - Q(S_{n-1}, a_{n-1}), \tag{3}$$

where $Q_{\text{target}}(\cdot,\cdot)$ and $Q(\cdot,\cdot)$ denote the Q functions represented by the target network and the main network, respectively, and $\gamma < 1$ denotes the discount factor.

Based on the TD-error for the i^{th} transition, δ_i, the prioritization for the i^{th} transition, p_i, is defined to be monotonic in the TD error. The two possible variants for the prioritization function could be *proportional* prioritization and a *rank based* prioritization. For the proportional prioritization, $p_i \triangleq |\delta_i| + \epsilon_p$. where $\epsilon_p > 0$ is a small positive number to ensure that each transition has a non-zero possibility of being sampled, once their TD-error reduces to zero. In the rank based prioritization, $p_i \triangleq \frac{1}{\text{rank}(i)}$. Here, rank$(i)$ is the rank of the i^{th} transition when the experience replay buffer is sorted according to the TD-error $|\delta_i|$. Note that, a new incoming transition is assigned a highest priority before storing it into the experience replay buffer. This ensures that the most recent experiences are played sooner, at least once.

Given the prioritization, as defined above, the PER uses a *stochastic sampling* method where the probability of selecting the i^{th} transition is monotonic in its prioritization p_i. Mathematically, it is written as:

$$P(i) = \frac{p_i^{\alpha_p}}{\sum_j p_j^{\alpha_p}}, \tag{4}$$

where the exponent $\alpha_p > 0$ determines the amount of prioritization used, and $\alpha_p = 0$ corresponds to uniformly random sampling of the experiences from the experience replay buffer. Note that, for both proportional and rank based prioritization the distribution in (4) is monotonic in TD-error, δ_i.

The use of stochastic sampling results in a biased estimate of Q-values. This is because, to obtain an unbiased estimate of expected value using stochastic update rules requires the updates to correspond to the same distribution as its expectation. To correct for this bias, the PER uses importance sampling (IS) weights computed as

$$w_i = \left(\frac{1}{NP(i)}\right)^{\beta}, \tag{5}$$

where N and β denote the size of the experience replay buffer and IS exponent, respectively. These weights are incorporated in the Q-learning updates by using $w_i\delta_i$, instead of δ_i. For instance, for a batch size equal to one, the DNN parameter update equation becomes

$$\theta = \theta + \eta w_i \nabla_\theta Q(S, a), \tag{6}$$

where η denotes the step size. Intuitively, an update corresponding to an experience which has a lower probability of being selected is multiplied by a large weight, to account for fewer updates. Note that, with $\beta = 1$ this completely compensates for the bias occurred due to non-uniform sampling distribution.

To summarize, the PER replaces the uniform random sampling procedure in deep Q-learning with the stochastic prioritization based on the the TD-error, and corrects the bias, generated due to use of stochastic prioritization, in the estimate

using the IS. For detailed account of PER we refer the readers to [14]. In the following section, we evaluate the performance of PER based DDQN for PDP minimization problem.

IV. Simulation Results

We consider an unmanned aerial vehicle (UAV) consisting of two queues, $K = 2$. The UAV moves at a speed equal to 5 Kmph, and transmits its observations to a ground receiver, which is located at a distance around 100 m., over the carrier frequency of 3 GHz. This corresponds to the channel coherence time and the slot duration being approximately equal to 9 msec. [17]. The channels between queues and the receiver are assumed to be independent of each other, and are identically distributed according to $\mathcal{CN}(0, 1)$. Further, each channel varies from slot to slot in independent and identically distributed fashion. During the simulation, the channel instances are generated using the modified Clark's simulator [18] with 40 oscillators. The set of rates, from which the service rates of queues are updated in each slot, is $R = \{1.6, 1.9, 2.4\}$. Therefore, the set of possible non-overlapping rate tuples, obtained using R, is given by $R_t \triangleq \{(0, 5.9), (1.6, 4.3), (1.9, 4), (2.4, 3.5), (3.5, 2.4), (4, 1.9), (4.3, 1.6), (5.9, 0)\}$. Note that, to keep the the cardinality of the action set lower, and therefore the computational complexity, we restrict the set R_t to only include the rate tuples whose sum equals 5.9. We emphasize that removing this restriction will further improve the performance of both the DDQN-RL based schemes, as expanding the set R_t would provide a finer control over the rates of the queues.

Further, for rate 1.6, we assume that the minimum target SNR required for successful decoding at the receiver is equal to 30 dB. For a unit bandwidth and ambient temp of 300 K., this amounts to approximately 5% channel outage for transmissions at rate 1.6. Also, the data packets are assumed to arrive according to a Pareto distribution with shape and scale parameter equal to 2 and 100, respectively. Life duration of each packet, T, before it is dropped, is assumed to be 500 msec.

To implement the DDQN-RL, we select a DNN with 10 fully connected hidden layers, each having a non-linearity implemented by LeakyRelu with $\alpha = 0.01$, and an input and output layer. The dimension of the input layer equals five, which is equal to the dimension of the system state given by $S = \{q_1(\cdot), q_2(\cdot), h_1(\cdot), h_2(\cdot), d(\cdot)\}$. The dimension of the output layer equals the number of actions available to scheduler. In our case, the total number of actions equals number of queues times the cardinality of the set R_t, i.e., 16. The first hidden layer consists of 26 neurons, and the number of neurons in each successive even indexed hidden layer equals the previous odd-indexed layer, and each successive odd-index hidden layer contains neurons equal to the number of neurons in previous odd-indexed hidden layer minus two. Thus, the number of neurons in first, third, fifth, seventh, and ninth hidden layer are given by 26, 24, 22, 20, and 18, respectively. For training, the mean-squared-error is used as the loss function, and we use the Adam optimization algorithm with batch size equal to 64. The learning rates used for DDQN-RL and DDQN-PER based RL are 9×10^{-6} and 8×10^{-6}, respectively, and Polyak's update method is used for updating the weights of target Q-network. The rest of hyper-parameter values used for implementing DDQN-RL and DDQN-PER based RL are summarized in Table I

TABLE I: Hyper-parameter values: the last three rows in the table below defines the parameter used for exploration. In particular, the exploration parameter ϵ gradually decreases from $\epsilon_{\max}$ to $\epsilon_{\min}$ with decay schedule determined by ϵ_{decay}.

Paramaeter	**Value**
Replay buffer size	2×10^5
α	0.6
β	0.4
ϵ_p	0.001
$\epsilon_{\max}$	1
$\epsilon_{\min}$	0.0001
ϵ_{decay}	0.9999

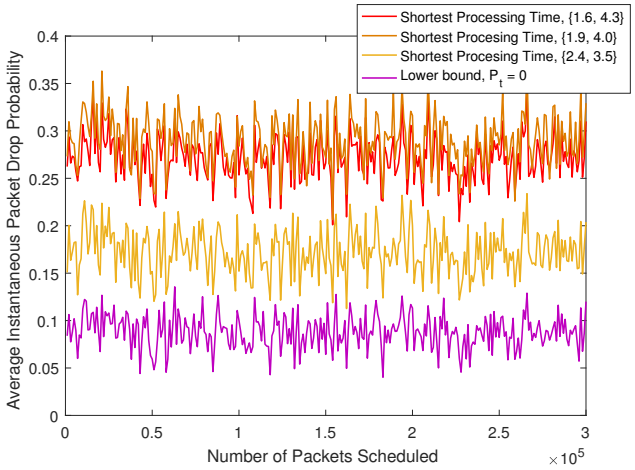

Fig. 2: Performance comparison for various rate allocations for SPT, and lower bound on PDP performance obtained by neglecting the effect of channel outages. The more symmetrical rate allocation for SPT, i.e., $\{2.4, 3.5\}$ results in the least PDP among all three rate allocations.

In order to compare the performance of all three schemes we plot the *average instantaneous PDP*, which denotes the average PDP computed over a window of 1000 packet arrivals. The reason for choosing such a metric is to facilitate the performance comparison of RL based schemes at the steady state, i.e., after the policy learned using an RL based scheme has converged and the transient part is over. This is important because due to sparse rewards the RL based schemes learn at a relatively slower speed, and takes significant time to learn the optimal policy.

In Fig. 2, we compare the PDP performance obtained for various possible rate allocations for the SPT scheme. We observe that the most symmetric rate allocation, i.e., $\{2.4, 3.5\}$, results in the lowest PDP among all possible rate

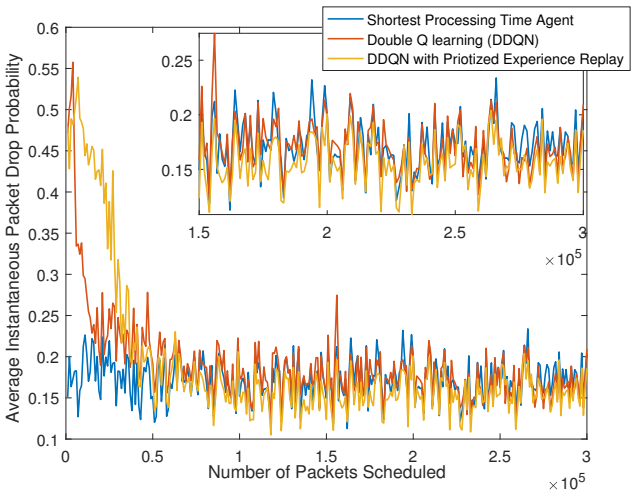

Fig. 3: Packet drop probability performance of policies learned using DDQN and DDQN with prioritized experience replay, and a scheduling policy based on shortest processing time. After initial learning phase, the DDQN-PER scheme results in $1-5\%$ lower PDP, compared to the other two schemes, at approximately 98% percent time-instants.

allocations. Also, we compared the performance of all possible rate allocations against the lower bound for PDP of the system considered in this paper. This lower bound corresponds to the performance of SPT scheme in the scenario when the channel outages are negligible, using the rate allocation $\{2.4, 3.5\}$. Note that, this serves as a lower bound because of the fact that the SPT scheme yields a near-optimal delay, when the services rate of the queues are time-invariant. We note that the lower bound suggests that an optimal scheduling policy can potentially improve the performance by $6-7\%$, compared to the SPT.

In Fig. 3, we compare the performance of the policy learned using the proposed DDQN-PER based RL, against the DDQN-RL and SPT schemes. As mentioned above, for a fair comparison, we compare the performance yield once the transient learning period is over, i.e., after observing approximately 50000 packets. We can observe that the DDQN-based RL algorithm yields a performance almost similar to SPT scheme, and for some time-instants it performs even worse than SPT. This is due to the sparse nature of the reward signal. On the other hand, we can observe that the DDQN-PER based RL scheme result in a performance improvement of $1-5\%$ for around 98% time-instants in the steady state phase.

V. Conclusions

In this paper, we developed an RL-based online scheme for learning an optimal cross-layer scheduling policy to minimize the packet drop probability. The difference in the time scales of the packet arrival process and the wireless channel results in a sparse reward problem which adversely affects the performance of DRL schemes. In order to mitigate the impact of sparse rewards, we use the prioritized experience replay with DDQN RL. The policy learned using the DDQN-PER based RL scheme yields a lower packet drop probability, compared to both the shortest processing time scheme as well as the DDQN-RL.

References

[1] C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and B. Vucetic, "A tutorial of ultra-reliable and low-latency communications in 6G: Integrating theoretical knowledge into deep learning," *https://arxiv.org/abs/2009.06010v1*, 2020.

[2] R. A. Berry and R. G. Gallager, "Communication over fading channels with delay constraints," *IEEE Trans. Inf. Theory*, vol. 48, no. 5, pp. 1135–1149, May 2002.

[3] M. Wang, J. Liu, W. Chen, and A. Ephremides, "Joint queue-aware and channel-aware delay optimal scheduling of arbitrarily bursty traffic over multi-state time-varying channels," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 503–517, Jan. 2019.

[4] X. Chen, W. Chen, J. Lee, and N. B. Shroff, "Delay-optimal buffer-aware scheduling with adaptive transmission," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2917–2930, Jul. 2017.

[5] W. Chen, U. Mitra, and M. J. Neely, "Packet dropping algorithms for energy savings," in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 227–231.

[6] Heng Wang and N. B. Mandayam, "A simple packet-transmission scheme for wireless data over fading channels," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1055–1059, Jul. 2004.

[7] C. She, C. Yang, and T. Q. S. Quek, "Cross-layer optimization for ultra-reliable and low-latency radio access networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 127–141, Jan. 2018.

[8] M. J. Neely, E. Modiano, and C. E. Rohrs, "Routing over parallel queues with time varying channels with application to satellite and wireless networks," in *Conf. on Inf. Sciences and Systms.*, Mar. 2002.

[9] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant, "Scheduling in multi-channel wireless networks: Rate function optimality in the small-buffer regime," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 1101–1125, Feb. 2014.

[10] M. van der Boor, S. C. Borst, J. S. H. van Leeuwaarden, and D. Mukherjee, "Scalable load balancing in networked systems: A survey of recent advances," *https://arxiv.org/pdf/1806.05444.pdf*, 2018.

[11] I. Hadar and A. Leshem, "Joint scheduling and beamforming for delay sensitive traffic with priorities and deadlines," in *Proc. ICASSP*, May 2020, pp. 5285–5289.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[13] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2018, pp. 6292–6299.

[14] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv:1511.05952v4 [cs.LG]*, 2015.

[15] S. Reddy and C. R. Murthy, "Dual-stage power management algorithms for energy harvesting sensors," *IEEE Trans. Wireless Commun.*, vol. 11, no. 4, pp. 1434–1445, Apr. 2012.

[16] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *arXiv:1509.06461v3 [cs.LG]*, 2015.

[17] D. W. Matolak and R. Sun, "Air–ground channel characterization for unmanned aircraft systems—part iii: The suburban and near-urban environments," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6607–6618, Aug. 2017.

[18] C. Xiao, Y. R. Zheng, and N. C. Beaulieu, "Novel sum-of-sinusoids simulation models for rayleigh and rician fading channels," *IEEE Trans. Wireless Commun.*, vol. 5, no. 12, pp. 3667–3679, May 2006.