Delay Laxity-Based Scheduling with Double-Deep Q-Learning for Time-Critical Applications

Xiangyu Ren, Jiequ Ji, Lin Cai

Dept. Electrical and Computer Engineering, University of Victoria. Victoria, BC, Canada
Email: jamesrxy@uvic.ca, jiequji@uvic.ca, cai@ece.uvic.ca

2022 IEEE 30th International Conference on Network Protocols (ICNP) | 978-1-6654-8234-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICNP55882.2022.9940259

***Abstract*—In this paper, we propose a novel delay-aware selective admission and scheduling algorithm for time-critical applications to guarantee the delay requirement of each packet in a single-hop downlink network. We consider a series of priorities among packets. To avoid always starving low-priority packets, we define a delay-laxity concept and introduce a new output gain model as our network utility function. In this context, we formulate a multi-objective optimization problem that minimizes the average queue backlog and maximizes the average network utility under the constraints of guaranteeing per-packet delay and achieving fairness among users. To solve this problem, we model our problem as a Markov Decision Process and propose a Double Deep Q Network-based algorithm to learn the optimal policy. Simulation results show that the proposed algorithm can achieve significant improvements in average delay, delay-outage drop rate, and goodput compared with the existing stochastic schemes. Moreover, the proposed algorithm outperforms the conventional Q-learning algorithm in terms of reward and learning speed.**

I. Introduction

The increasing popularity of intelligent mobile devices is spurring the emergence of new applications with advanced features such as unmanned driving, digital twin, and extended reality [1], [2]. Moreover, all of these applications are time-critical and typically have a stringent delay requirement to satisfy their quality of experience (QoE) which is not fully addressed in existing wireless systems [2]. How to effectively manage premium wireless resources to effectively support the QoE of time-critical applications remains an open issue.

The scheduling problem of wireless resources has been extensively studied in various contexts, where throughput is considered the main objective [3]–[5]. However, such throughput-optimal scheduling policies often lead to severe queuing delays since they only schedule flows with the maximum backlog-gradient [6]. To tackle the delay issue, several delay-based scheduling algorithms were proposed to tradeoff throughput and delay performances [7], [8]. For example, in [7], the delay of the head-of-queue packet is served as a weight of the max-weight decision.

In spite of the QoS performances that can be achieved, these solutions cannot effectively and efficiently support the emerging applications. Especially for the time-critical applications, where packets arriving late is as severe as being dropped. Existing works achieve maximum network utility and minimum average delay based on the assumption that all packets in the same flow yield the same utility and no delay requirement is specified. Such assumption is unreasonable since packets of the same flow may be prioritized according to the importance and different delay requirements in practice and thus, should have different utility after successful transmission. Moreover, packets suffering excessive delay should be dropped early in the network instead of being transmitted to the users.

To bridge the gap, we propose a delay-aware selective admission and scheduling (DASS) algorithm to guarantee the worst-case delay of any non-dropped packet by selectively scheduling packets in the buffer. Specifically, we consider a single-hop wireless network where packets are prioritized with various delay requirements named delay budgets. When a packet experiences a delay exceeding its delay budget, it is dropped immediately. Furthermore, to differentiate the utility of different packets with different delay budgets, a delay-laxity concept is defined. Based on this, we introduce a gain model as the network utility function of our system. In this context, we formulate a multi-objective optimization problem (MOOP) that minimizes the average queue length while maximizing the average network utility under the constraints of guaranteeing per-packet delay and achieving fairness among users.

Solving the MOOP using dynamic programming can be computationally intensive in a wireless network. Recently, reinforcement learning (RL) has emerged as an effective solution thanks to its capability of learning an optimal policy only by interacting with the environment without any prior knowledge. There have been developed several works on the use of RL-based algorithms to address resource management and traffic control in wireless networks [9]–[11]. Inspired by these works, we propose a deep reinforcement learning algorithm to solve the MOOP. Specifically, we first transform the MOOP into a scalar problem using the weighted sum approach. Next, we model it as a Markov Decision Process (MDP) and propose an algorithm based on Double Deep Q Network (DDQN) [12] to solve it. The proposed algorithm overcomes the curse of dimensionality with large state and action spaces. Finally, we verify the performance of the proposed DASS scheme and DDQN-based algorithm in terms of delay, goodput, and delay-outage drop rate via extensive simulations.

The rest of the paper is organized as follows. In Sec. II, we explain our system model and formulate the network optimization problem with detailed analysis. In Sec. III, we discuss the proposed DDQN-based algorithm. Extensive simulations are designed to verify the performance of our model and algorithm in Sec. IV. Finally, we conclude our work in Sec. V.

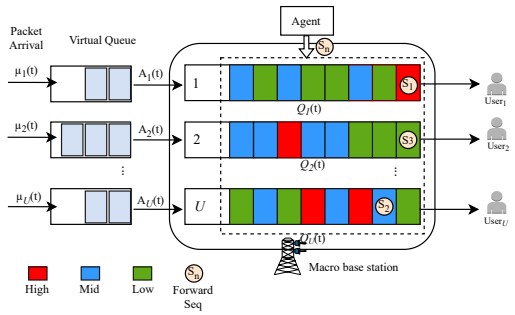


Fig. 1: Delay-aware selective scheduling.

II. System Model and Problem Formulation

A. Network Model

We consider the downlink scheduling problem of a single-hop network as shown in Fig. 1, which consists of one macro base station (MBS) and U ground users. Let $\mathcal{U} = \{1, \cdots, U\}$ denote the set of users that are associated with the MBS.

The single-hop network is assumed to operate in discrete time with normalized time slots $t \in \{0, 1, \cdots\}$. All random arriving packets are queued separately for transmission over each downlink. We define the set of packets as $\mathcal{K} = \{1, \cdots, K\}$ which comprises three classes of packets with different delay requirements denoted by $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{L}$, i.e., $\mathcal{K} \triangleq \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}$. The packet arrival rate $\mu_u(t)$ for each user is assumed to be independent and identically distributed over time slots. As shown in Fig. 1, we consider admission control before injecting each packet into the corresponding buffer. Let $\mathbf{A}(t) \in \{0, 1\}_{U \times 1}$ denote the packet admission vector, where $A_u(t) = 1$ if the packet is injected to queue u at time slot t and $A_u(t) = 0$ otherwise. For convenience, we assume that each packet has the same size and that at most one packet arrives at each downlink at each time slot, so $\sum_{u \in \mathcal{U}} A_u(t) \leq U$.

Note that, unlike most works that consider the first-in-first-out (FIFO) queue management, the packet scheduler of the MBS (i.e., agent) in the proposed DASS model selects the optimal packet from all queues to forward by jointly considering link conditions and network-utility maximization[1].

B. Association and Transmission Model

Let $\mathbf{S}(t) = \{S_1(t), \cdots, S_U(t)\}$ denote the user association vector, and its element $S_u(t) = 1$ if user u associates with the MBS at time slot t and $S_u(t) = 0$, otherwise. For a given wireless channel, we assume that the MBS serves one user at each time slot, which yields the following constraints

$$\begin{aligned} &S_u(t) \in \{0, 1\}, \ \forall u, t, \\ &\sum\nolimits_{u \in \mathcal{U}} S_u(t) = 1, \ \forall t. \end{aligned} \tag{1}$$

For simplicity, we consider the scheduling for a single channel, i.e., at most one packet can be transmitted per time slot, which can be easily extended when there are a number of orthogonal channels. Accordingly, we introduce discrete binary transmission variables $\eta_{u,k}(t) \in \{0, 1\}$ to indicate the transmission status of packet k of queue u at time t. Then the transmission constraints are given by

$$\sum\nolimits_{k \in \mathcal{K}} \eta_{u,k}(t) \leq 1, \ \forall u \in \mathcal{U}, k \in \mathcal{K}. \tag{2}$$

Note that the probability of successful packet transmission over each downlink is affected by the channel condition of the corresponding downlink. We use $\mathbf{C}(t) = \{C_1(t), \cdots, C_U(t)\}$ to denote the channel condition vector, which is assumed to be known by the MBS at the beginning of each time slot. For given vectors $\boldsymbol{\eta}(t)$ and $\mathbf{C}(t)$, the probability of successful packet transmission over downlink u is expressed as

$$Pr(\text{downlink to } u \text{ success} | \boldsymbol{\eta}(t), \mathbf{C}(t)) = \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)), \tag{3}$$

where the probability function $\Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t))$ for $u \in \mathcal{U}$ takes only real values between 0 and 1. Moreover, we introduce an indicator variable $1_u(t)$ to indicate the successful transmission of downlink u:

$$1_u(t) = \begin{cases} 1, & \text{with probability} \quad \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)), \\ 0, & \text{with probability} \quad 1 - \Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t)). \end{cases} \tag{4}$$

Then the discrete transmission variable $\eta_{u,k}$ in (2) can be rewritten as[2]

$$\hat{\eta}_{u,k}(t) = \eta_{u,k}(t) 1_u(t). \tag{5}$$

C. Queuing Model

We define $\mathbf{Q}(t) = \{Q_1(t), \cdots, Q_U(t)\}$ as the set of packets currently backlogged in each queue. Let $x_{uk}(t)$ denote the sojourn time of packet k in queue u at time slot t. In addition, the maximum tolerable queuing delay for the three classes of packets are denoted by Th_H, Th_M, and Th_L, respectively. Packet k will be dropped when its sojourn time in queue u is larger than its maximum tolerable queuing delay. Thus, we introduce a binary delay outage drop variable $D_{u,k}(t)$ to indicate the drop state of each packet, i.e.,

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \quad \forall k \in \mathcal{H} \cup \mathcal{M} \cup \mathcal{L}. \tag{6}$$

To sum up, the queuing dynamics of queue u, including forwarding packet j to the corresponding user while dropping the delay outage packets in the queue, is expressed as

$$\begin{aligned} Q_u(t+1) = \max\{Q_u(t) - S_u(t)\hat{\eta}_{u,j} - \sum_{i \neq j, i \in Q_u(t)} D_{u,i}(t), 0\} \\ + A_u(t), \quad \forall i, j \in Q_u(t). \end{aligned} \tag{7}$$

D. Scheduling and Gain Model

Given the sojourn time $x_{uk}(t)$ of packet k in queue u, we define the delay-laxity as follows:

$$L_{uk}(t) = \text{Th}_k - x_{uk}(t), \tag{8}$$

[1]The implementation of DASS mechanism can be achieved by using a set of priority FIFO queues for each user such that packets of the same priority are backlogged in the same buffer and their delay-laxity (defined in Sec. II-D) follows an ascending order.

[2]Since the maximum utility is independent of interlink success correlations [7], it suffices to use only the marginal distribution function $\Phi_u(\boldsymbol{\eta}(t), \mathbf{C}(t))$ for each $u \in \mathcal{U}$.

which measures the remaining queuing delay budget. The output gain of scheduling packet k is denoted by $r_{u,k}(t) = \frac{\omega_k}{L_{uk}(t)}$, where ω_k is the gain weight which is also classified in terms of delay requirements, i.e., $\omega_k \in \{\omega_H, \omega_M, \omega_L\}$ and $\omega_H > \omega_M > \omega_L$ [3].

In addition, we consider a discounted potential output gain of each queue to reflect each scheduling decision's consequence, which is given by

$$\hat{G}_u(t) = \rho \sum_{k' \in X_u(t)} r_{u,k'}(t) = \rho \sum_{k' \in X_u(t)} \frac{\omega_{k'}}{L_{uk'}(t)}, \ \forall u \in \mathcal{U}, \tag{9}$$

where $X_u(t)$ denotes the set of remaining packets in queue u after scheduling and delay outage drop; $r_{u,k'}(t)$ denotes the gain to be obtained by scheduling packet k' in $X_u(t)$; $\rho \in [0,1]$ denotes the discount factor. The output gain obtained by transmitting packet k in queue u can be written as follows

$$G_{u,k}(t) = r_{u,k}(t) + \sum_{u=1}^{U} \hat{G}_u(t). \tag{10}$$

Hence, by aggregating the above equations, the average achievable gain of all queues at each time slot is given by

$$\begin{aligned} G(t) = & \frac{1}{U} \sum_{u \in \mathcal{U}} \Bigg[\sum_{k \in X_u(t)} \frac{\omega_k}{L_{uk}(t)} (S_u(t)\hat{\eta}_{u,k} \\ & + S_u(t)\rho(1-\hat{\eta}_{u,k}) + (1-S_u(t))\rho) \Bigg], \end{aligned} \tag{11}$$

which also represents the network utility defined in our system.

E. Problem Formulation

Maximizing the average output gain defined in Eq. 11 is equivalent to maximizing the network utility and minimizing the average delay-outage drop rate. In addition, we consider the utility fairness among users. Let $F(\bar{\boldsymbol{A}})$ denote a concave and non-decreasing function of the U-dimensional vector $\bar{\boldsymbol{A}} = \{\bar{A}_1, \cdots, \bar{A}_U\}$, where $\bar{A}_u$ is the time-averaged admission rate of user $u \in \mathcal{U}$. Then we have

$$F(\bar{\boldsymbol{A}}) = \sum_{u=1}^{U} F_u(\bar{A}_u) = \sum_{u=1}^{U} \log(1 + \nu_u \bar{A}_u), \tag{12}$$

where ν_u is a positive constant. To this end, we formulate an MOOP that minimizes the average queue length while maximizing the average output gain under the constraints of guaranteeing per-packet delay and achieving network stability and fairness among users over T time slots. Mathematically, this problem can be written as follows

$$\text{P0:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} \left\{ -\frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)], \ \frac{1}{T} \sum_{t=0}^{T} G(t) \right\} \tag{13a}$$

$$\text{s. t} \quad \sum\nolimits_{u=1}^{U} F_u(\bar{A}_u) \geq \sum_{u=1}^{U} F(\bar{\gamma}_u), \tag{13b}$$

$$\bar{A}_u \leq \frac{1}{T} \sum\nolimits_{t=0}^{T} \mathbb{E}[S_u(t)\hat{\eta}_{u,k}(t)], \forall u \in \mathcal{U}, \tag{13c}$$

$$S_u(t) \in \{0,1\}, \ \sum\nolimits_{u \in \mathcal{U}} S_u(t) = 1, \tag{13d}$$

$$\hat{\eta}_{u,k}(t) \in \{0,1\}, \sum_{k \in X_u(t)} \hat{\eta}_{u,k}(t) \leq 1, \tag{13e}$$

$$D_{u,k}(t) = \begin{cases} 1, & \text{if } x_{uk}(t) \geq \text{Th}_k, \\ 0, & \text{if } x_{uk}(t) < \text{Th}_k, \end{cases} \tag{13f}$$

where $\bar{A}_u = \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}[A_u(t)]$ denotes the average number of packets admitted to queue u over T time slots and $\bar{\gamma}_u \in \{0,1\}$ denotes the auxiliary variable that is helpful for achieving fairness among users with random arrival rate [13]. Constraint (13c) guarantees the mean rate stability while constraint (13f) guarantees per-packet delay.

To simplify the problem, the weighted sum method [14] is employed for MOOP scalarization, where two parameters λ_1 and λ_2 are defined to characterize the importance of each objective, respectively. Then the original problem **P0** can be transformed based on the Lagrangian method as follows

$$\begin{aligned} \text{P1:} \max_{\mathbf{A},\mathbf{S},\boldsymbol{\eta},\mathbf{D}} & \Bigg\{ \lambda_2^n \frac{1}{T} \sum_{t=0}^{T} G(t) - \lambda_1^n \frac{1}{T} \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(t+1)] \\ & + \nu_1 \sum_{u=1}^{U} F_u(\bar{A}_u) \Bigg\}, \end{aligned} \tag{14a}$$

$$\text{s. t} \quad \nu_1 \geq 0, \tag{14b}$$

$$(13c) \sim (13f), \tag{14c}$$

where constraint (13b) is replaced by $\nu_1 \geq 0$, while the rest of the constraints remain the same.

III. Algorithm Design

We consider a smart agent in the MBS to learn user scheduling and association, admission control, and delivery of all packets. Since the admission and delivery of each packet are affected by the current network environment and the actions of the switch, the learning task of the agent satisfies the Markov property. Thus, we model **P1** as an MDP, which can be described by the tuple $\Omega = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S} = \{s_t | t = 1, \cdots, T\}$ denotes the state space describing the environment, and $s_t = \{1_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}, u \in \mathcal{U}, k \in Q_u(t)$; $\mathcal{O} = \{o_t | t = 1, \cdots, T\}$ denotes the observation space, where $o_t = \{C_u(t), Q_u(t), x_{u,k}(t), L_{uk}(t)\}$; The action space is denoted by $\mathcal{A} = \{a_t | t = 1, \cdots, T\}$, where $a_t = \{A_u(t), S_u(t), \eta_{u,k}(t), D_{u,k}(t)\}$; $\mathcal{P}$ denotes the state transition function with $P_{s_t, s_{t+1}}(a_t)$ being the probability that the current state s_t transfers to the next state s_{t+1} when action a_t is performed; $\mathcal{R}$ denotes the reward function that maps the network state and the joint actions of the agent to rewards; and $\gamma \in [0,1]$ denotes the discount factor.

According to the formulated problem **P0**, our objective is twofold: minimizing the average queue length and maximizing the average output gain. As a result, the network reward of T

[3]Such design is helpful for reducing the packet drop rate and avoiding the starvation of low-priority packets. This is due to the fact that there is more reward for delivering low-priority packets with a smaller delay laxity as much as possible rather than dropping them on timeout.

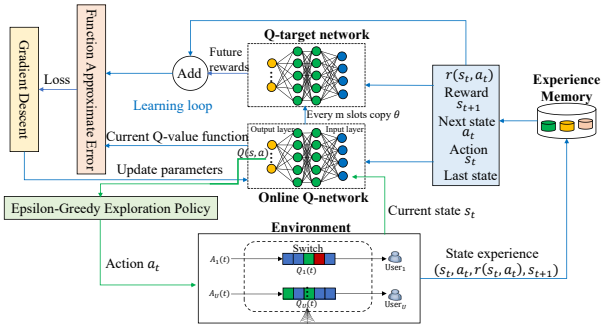


Fig. 2: DDQN framework for delay-laxity-based scheduling.

time slot is defined by

$$r(s_t, a_t) = \lambda_2 \frac{1}{T} \sum_{\tau=0}^{T} G(\tau) - \lambda_1 \frac{1}{T} \sum_{\tau=0}^{T} \sum_{u \in \mathcal{U}} \mathbb{E}[Q_u(\tau+1)]. \quad (15)$$

At time t, the agent observes a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}$ according to a certain policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which receives a reward $r_t = r(s_t, a_t)$ and produces a new state s_{t+1} with the transition probability $P(s_{t+1}|s_t, a_t)$.

The learning system establishes the relationship between the optimal criterion and the optimal policy by introducing a Q-value function, which is defined by the expected sum of discounted rewards obtained by performing action a at state s and following policy π in the next state

$$Q_\pi(s, a) = r(s, a) + \gamma \sum_{s'} P_{s,s'}(a) Q_\pi(s', \pi(s')). \quad (16)$$

The agent continuously improves its policy with the accumulation of experience to yield the maximum value of $Q(s, a)$ for all available states and actions. Overall, the optimal Q-value function is easily obtained when the optimal policy $\pi^*(s) = \max_{\pi} Q_\pi(s, a)$ that maps the set of states and actions is satisfied. Using the Bellman optimality equation, the optimal Q-value function can be expressed by

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s'} P_{s,s'}(a) \max_{a'} Q^*(s', a'), \quad (17)$$

where s' and a' are the state and action at the next time slot. The process of finding the optimal policy by optimizing the Q-value function is called Q-learning.

Although Q-learning has emerged as a prospective learning algorithm based on value, it relies heavily on the set of records in the sample when searching for the optimal policy. In order to address this issue, many scholars have focused on the Deep Q-Network (DQN) algorithm based on the combination of neural networks and Q-learning, which mainly takes the state and action as the input of the neural network to approximate the Q-value function instead of maintaining the Q-table. However, this manner tends to confuse the agent about the selection and evaluation of actions, which leads to Q-value overestimation due to the positive bias of the max operator used in DQN to update its Q-value function [15].

Algorithm 1 DDQN-based Training Algorithm

Input: Action set and network parameters $\{\mathcal{A}, \alpha, F, E, \gamma, C, B\}$
Output: Optimal policy π^* and maximum reward R
Initialize: the online Q-network $Q(s, a; \boldsymbol{\theta})$ with weight $\boldsymbol{\theta}$
Initialize: the target Q-network $(Q(s, a; \boldsymbol{\theta}'))$ with $\boldsymbol{\theta}' = \boldsymbol{\theta}$
for each episode **do**
  Initialize the environment and obtain an initial state s_1
  for each iteration of an episode **do**
    if $\hbar < \varepsilon$ **then**
      Randomly choose an action a_t;
    else
      Choose an action $a_t = \arg\max_{a' \in \mathcal{A}} Q(s_t, a'; \boldsymbol{\theta})$;
    end if
    Execute action a_t and receive a reward $r(s, a)$
    Get the next state s_{t+1} and update the input Q_{t+1}
    Store the current tuple Ω_t in the experience relay memory
    Randomly choose a minibatch from $\{s_t, a_t, r_t, s_{t+1}\}$
    Calculate the target Q-value with optimizer
    if an episode terminates at iteration $t+1$ **then**
      $y(t) = r(s, a)$;
    else
      $y(t) = r(s, a) + \gamma Q(s', \arg\max_{a' \in \mathcal{A}} Q(s_t, a'; \boldsymbol{\theta}); \boldsymbol{\theta}')$;
    end if
    Update the weight of the online network by minimizing the loss function: $\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[(y(t) - Q(s, a; \boldsymbol{\theta}))^2]$
    Update the target network $Q' \leftarrow Q$ after F iterations
  end for
end for

To address the issues mentioned above, we propose an improved DDQN-based algorithm, which decouples the action selection and Q-value calculation into two separated max function estimators to eliminate overestimation. As shown in Fig. 2, DDQN also has two neural networks, i.e., online network $Q(s, a; \boldsymbol{\theta})$ and target network $Q(s, a; \boldsymbol{\theta}')$, similar to DQN. But the target Q-value for DDQN is replaced by

$$y(t) = r(s, a) + \gamma Q(s', \arg\max_{a'} Q(s', a'; \boldsymbol{\theta}); \boldsymbol{\theta}'). \quad (18)$$

Note that the selection and evaluation of an action follows an **arg max** operator defined by the set of weights $\boldsymbol{\theta}$. Then the other set of weights $\boldsymbol{\theta}'$ is used to evaluate the Q-value of this policy. The ε-greedy method is adopted to balance exploitation and exploration. In this algorithm, the agent aims to continuously learn an optimal policy for optimizing the Q-value function by minimizing the loss function

$$\text{Loss}(\boldsymbol{\theta}) = \mathbb{E}[y(t) - Q(s, a; \boldsymbol{\theta}))^2]. \quad (19)$$

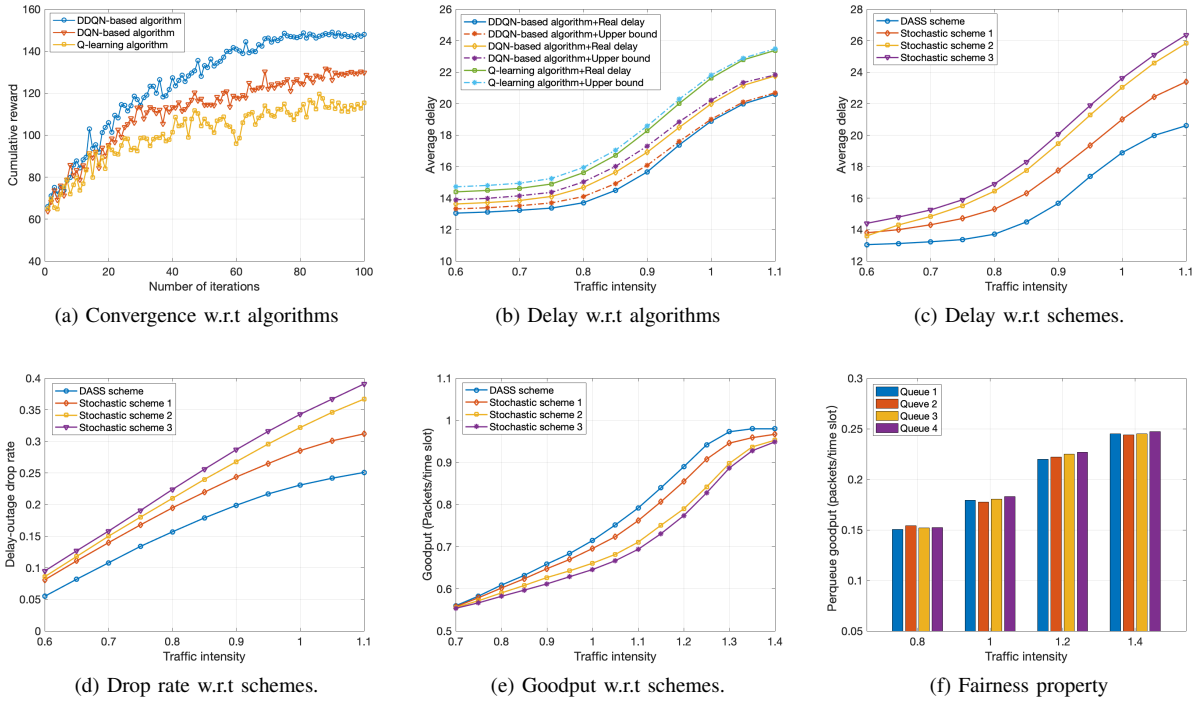
In order to reduce the loss value, the gradient descent method is used to update the weights of the online network. The update of $\boldsymbol{\theta}$ can be expressed by

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \mathbb{E}[(y(t) - Q(s, a; \boldsymbol{\theta})) \nabla Q(s, a; \boldsymbol{\theta})]. \quad (20)$$

The details of the training process are summarized in Alg. 1.

IV. SIMULATION RESULTS

In the experiment, we consider a single-hop wireless network in which $U = 4$ ground users are associated with a single



(a) Convergence w.r.t algorithms (b) Delay w.r.t algorithms (c) Delay w.r.t schemes.

(d) Drop rate w.r.t schemes. (e) Goodput w.r.t schemes. (f) Fairness property

Fig. 3: Comparisons of convergence and QoS performances with different schemes and learning algorithms.

TABLE I: DDQN architecture settings

Neurons e	64	Learning rate α	0.0001
Activation function	ReLU	Discount factor γ	0.999
Update period F	100	Replay memory size C	2000
ε-greedy	$0.9 \rightarrow 0.1$	Minibatch size B	64
Number of episodes	2000	Arrival rate μ	$0.6 \rightarrow 1.4$

MBS. Note that the model can be easily extended with more users by modifying the input parameter U. A total number of $K = 2000$ packets ($H = 400$, $M = 600$ and $L = 1000$) arrive at the MBS over a period of time following a Poisson distribution with an expected arrival rate of μ packets per time slot. The delay budgets for different classes are set as $\text{Th}_H = 10$, $\text{Th}_M = 20$ and $\text{Th}_L = 30$, respectively, with the unit of time slots. The output gain weights of the three classes are set as $\omega_H = 0.5$, $\omega_M = 0.3$, and $\omega_L = 0.2$, respectively, and the discount factor of the potential output gain is set as $\rho = 0.3$. In addition, we assume an equal importance of the two objectives, so $\lambda_1 = \lambda_2 = 0.5$. As for the architecture of the proposed DDQN algorithm, both the online and target network have the same structure with one input layer, two hidden layers, and one output layer. The detailed parameter settings are summarized in Table I.

A. Result Analysis

In the experiment, we mainly compare the proposed DDQN-based algorithm with DQN and Q-learning. In addition, to demonstrate the superiority of the proposed DASS mechanism, we also consider the following three stochastic schemes: (i) scheme 1 jointly optimizes packet transfer scheduling and user association based on DASS but without admission control; (ii) scheme 2 jointly optimizes packet admission and user association but adopts FIFO queue management; and (iii) scheme 3 adopts random packet admission and user association based on FIFO queue management. Note that all schemes are solved by the proposed DDQN-based algorithm.

1) Convergence: We first demonstrate the convergence of the proposed algorithm compared with the other two algorithms as shown in Fig. 3(a). With the increase of iterations, we can observe that the proposed DDQN-based algorithm always outperforms the other two algorithms in terms of cumulative reward and convergence speed. The reason is that the proposed algorithm can prevent the overestimation of the action-value function by decoupling the Q-target. In addition, the convergence speed is significantly higher than that of the other algorithms thanks to the use of neural network $Q(s, a; \boldsymbol{\theta})$ to approximate the target Q-value $Q^*(s, a)$, which can greatly reduce the time of searching for the maximum value.

2) Delay performance: Fig. 3(b) shows the delay performance of different algorithms versus traffic intensity[4] ϱ. The solid and dash lines indicate the average delay calculated based on the observed delay (real delay) and the maximum tolerable delay per packet (upper bound), respectively. It can be observed that all curves increase with ϱ and eventually become saturated when ϱ is sufficiently large. The proposed DDQN-based algorithm shows up to 6% and 13% delay reduction compared to the DQN-based algorithm and the Q-

[4]The traffic intensity ϱ is defined as the ratio of the arrival rate to the capacity region boundary to measure the average resource occupancy.

learning algorithm, respectively. It is also obvious that the real average delay gradually approaches the delay upper bound as ϱ becomes large. The reason is that as ϱ grows, the queue length of each user increases leading to longer queuing delay, and thus packets reach their maximum tolerable queuing time. On the other hand, it shows that the proposed model guarantees the per-packet delay within its upper bound.

In Fig. 3(c), the average delay achieved by various schemes is plotted. Despite the average delay of all the four schemes increase as the traffic intensity ϱ grows, the proposed DASS scheme always achieves the lowest delay. In addition, we note that only our proposed scheme and scheme 1 show the trend of convergence when ϱ exceeds the capacity region, while the others diverge. The reason is that the packet-selection process becomes more crucial for better control of packets with different delay requirements when ϱ is large. The comparisons also reveal the importance of admission control. Overall, our scheme achieves a delay reduction up to 30%.

3) Delay-outage drop performance: Next, we verify the delay-outage drop rate performance of DASS as shown in Fig. 3(d). Since the proposed model guarantees per-packet delay by dropping the delay-outage packets, minimizing the delay-outage drop rate is one of the key metrics in evaluating the performance. The proposed DASS scheme achieves a significantly lower drop rate up to 60%. This is because the packet-selection process allows as many packets as possible to be transmitted before reaching their maximum tolerable queuing time. The increasing gap between the proposed scheme and schemes 2 and 3 also indicates the significance of the packet-selection process in our design.

4) Throughput performance: Here, we consider goodput, i.e., the number of successfully received packets over the total measured time period. In Fig. 3(e), the goodput curves achieved by the four schemes are plotted. As expected, the goodput increases as ϱ becomes large and eventually approaches 1 (because we assume at most one packet can be transmitted per time slot). For different values of ϱ, the proposed scheme always achieves the highest goodput with an improvement up to 16.7%. It is clear that scheme 1 gives the second-best results which further shows the significance of the packet-selection process in improving goodput.

An extra comparison experiment is presented to demonstrate the goodput fairness among all users as shown in Fig. 3(f). Each bar represents the average goodput value of the corresponding queue. In addition to the assumption that the packet arrival rate of each user follows the same Poisson distribution, the goodput variation of each queue shows a similar pattern thanks to the introduction of Eq. (12) in our model. Thus, the fairness issue among users can be addressed in our design.

V. Conclusions

In this paper, we investigated the scheduling problem for time-critical applications with different delay requirements in a single-hop downlink network. To address the problem, a delay-aware selective scheduling scheme based on delay laxity and output gain was proposed, in which we particularly considered the drop of packet due to excessive delay. In this context, we formulate a multi-objective optimization problem that minimizes the average queue length while maximizing the average output gain under the constraints of guaranteeing per-packet delay and achieving fairness among users. To cope with the uncertainties in wireless networks, e.g., packets arrival rate and channel condition, we transformed the problem into an MDP and proposed a DDQN-based algorithm to solve it. Simulation results clearly show the superiority of the proposed scheduling scheme and algorithm in reducing delay while improving throughput over other solutions. However, many other issues in practice, e.g., wireless link dynamics, are not considered in the current model. In future work, more effort will be addressed on incorporating environmental dynamics to improve the scalability of our model.

Acknowledgement

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), Mitacs, and Compute Canada.

References

[1] B. Ji, Y. Wang, K. Song, C. Li, H. Wen, V. G. Menon, and S. Mumtaz, "A survey of computational intelligence for 6G: Key technologies, applications and trends," *IEEE Trans. Ind. Inform.*, vol. 17, no. 10, pp. 7145–7154, Oct. 2021.

[2] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, Jun. 2020.

[3] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[4] X. Lan, Y. Chen, and L. Cai, "Throughput-optimal h-qmw scheduling for hybrid wireless networks with persistent and dynamic flows," *IEEE Trans. on Wireless Communications*, vol. 19, no. 2, pp. 1182–1195, 2019.

[5] Y. Chen, X. Wang, and L. Cai, "On achieving fair and throughput-optimal scheduling for tcp flows in wireless networks," *IEEE Trans. on wireless communications*, vol. 15, no. 12, pp. 7996–8008, 2016.

[6] L. Hai, Q. Gao, J. Wang, H. Zhuang, and P. Wang, "Delay-optimal back-pressure routing algorithm for multihop wireless networks," *IEEE Trans. Vehicular Tech.*, vol. 67, no. 3, pp. 2617–2630, Nov. 2017.

[7] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 41–54, Apr. 2012.

[8] Y. Chen, X. Wang, and L. Cai, "Head-of-line access delay-based scheduling algorithm for flow-level dynamics," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5387–5397, 2016.

[9] J. Bae, J. Lee, and S. Chong, "Learning to schedule network resources throughput and delay optimally using q+-learning," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 750–763, Jan. 2021.

[10] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *Proc IEEE Global Commun. Conf. (GLOCOM)*, Jan. 2020, pp. 1–6.

[11] J. Zhu, Y. Song, D. Jiang, and H. Song, "A new deep-q-learning-based transmission scheduling mechanism for the cognitive internet of things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018.

[12] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[13] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[14] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 3–18, Feb. 2018.

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.