Transformer-Based Packet Scheduling under Strict Delay and Buffer Constraints

Sylvain Nérondat*†, Xavier Leturc*, Christophe J. Le Martret*, Philippe Ciblat†

*Thales SIX GTS SAS, France
†LTCI, Télécom Paris, IP Paris, 91120 Palaiseau, France
firstname.lastname@{*thalesgroup.com, †telecom-paris.fr}

***Abstract*—This paper presents a packet scheduler for managing multiple links with varying channel capacities, where each link carries multiple data flows with finite buffers and strict delay constraints. Packet loss can result from buffer overflow or delay violations. We propose a deep reinforcement learning scheduler based on an encoder-only transformer (EOT) architecture, capable of handling a variable number of links without dedicated training. Using deep Q-learning, the scheduler minimizes the packet loss rate (PLR). Simulations show that our approach outperforms a state-of-the-art fully connected (FC) scheduler, delivering better performance under diverse configurations of links, packet arrival rates, and channel capacities.**

***Index Terms*—Deep Reinforcement Learning, Packet Scheduling, Transformer.**

I. Introduction

Packet scheduling is a critical mechanism in communication systems, determining how data packets are prioritized and transmitted to ensure efficient resource utilization, low latency, and high-quality service in increasingly demanding network environments. This paper proposes a new deep neural network (NN) architecture for a central packet scheduler that manages several communication links, each with a specific channel capacity. We assume that each link supports several data flows, each with distinct constraints. The packets of each data flow are stored in dedicated first in, first out (FIFO) buffers with finite sizes. Each packet is defined by its waiting time (WT), which is the duration it spends in the buffer since its arrival. We consider two different kinds of traffic: i) the delay constraint (DC) traffic for which the WT cannot exceed a given limit, ii) best-effort (BE) traffic with no constraint. The constraints considered in this work are: i) finite buffer length, and ii) strict delay constraint for DC traffic. A finite buffer size can lead to buffer overflow (BO) when new packets arrive and the buffer is full, while delay violation (DV) may occur if the packet's WT exceeds the delay constraint of DC traffic.

Until recently, scheduling algorithms have relied on heuristics. These heuristics were designed to achieve specific objectives: round-robin (RR) aims to distribute resources equally across all links, proportional fair (PF) seeks to distribute rates fairly across all links, while the log-rule, exp-rule, and modified largest weighted delay first (MLWDF) aim to guarantee bounded delays. These heuristics operate with limited information on flows and buffers, such as: i) average or instantaneous data rate, ii) head of line (HoL) delay, defined as the highest WT, and iii) the number of packets (NP).

More recently, new approaches based on NN architectures trained using deep reinforcement learning (DRL) have been proposed to solve the scheduling problem. One advantage of NN architectures is their ability to accommodate as many features as desired. This makes it straightforward to increase the number of features at the input of the NN compared to heuristics. Moreover, since DRL operates in a model-free context, it allows for the design of bespoke objectives through the reward function. The NN architectures take as input vectors belonging to the state space, which plays a key role in system performance and is defined by the solution designer. Most of the schedulers proposed in the literature (both heuristics and NNs) primarily use the HoL or NP in the buffers, or both.

In this paper we investigate two new state space models for the DC traffic. First, an extended version of the HoL referred to as extended HoL (xHoL) which consists of taking the HoL value plus its multiplicity, i.e. the number of packets sharing the HoL value. The state size is increased by one for each DC buffer compared to the conventional HoL. Second, a state model that considers the WT information of all packets in the buffers, referred to as all packet delays (APD). This state has higher cardinality since it contains more information than the HoL or xHoL, and is thus expected to have better performance.

A good NN architecture for a scheduler after training is expected to verify the following properties:

- It should be permutation equivariant (PE) with regards to the inputs since this property provides superior performance for both learning and inference phases [1].
- It should adapt to changes in the number of links over time while maintaining good performance, a property we denote as number of links independent (NLI).
- It should consider jointly the buffer information of all users to perform the scheduling in order to provide a global and thus potentially optimal solution. We denote this property as global buffer management (GBM).

Note that the terms NLI and GBM are introduced for the first time in this paper, representing new concepts and contributions. It is important to emphasize that while the NLI property enables the architecture to handle a varying number of links, it does not necessarily guarantee good performance. The performance depends on factors such as the training process (algorithm, hyperparameters, dataset...) and the architecture's ability to generalize; therefore it must be validated.

In this paper, we propose a scheduling solution based on DRL using a NN architecture trained to minimize the packet loss rate (PLR) due to BO and DV, employing a deep Q-learning (DQL) algorithm [2]. We have selected an encoder-only transformer (EOT) architecture that possesses the three aforementioned properties: PE, NLI, and GBM. The main contributions of this paper are: i) a synthesis and analysis of the state of the art (SotA) about DRL schedulers, ii) the proposal of two new state spaces, iii) the proposal of a DRL scheduling solution that minimizes the PLR due to BO and DV using an EOT architecture, iv) the performance evaluation and comparison of three different state spaces, and v) the assessment of the generalization capability of the proposed architecture and the SotA in terms of PLR.

The rest of the paper is organized as follows. Section II provides an analysis of the SotA. Section III describes the system model. Section IV introduces the optimization problem. Section V is devoted to the implementation of the evaluated solutions. Section VI presents and analyzes the numerical results. Finally, concluding remarks are offered in Section VII.

II. State of the art of DRL schedulers

We analyze the SotA of scheduling solutions based on NN architectures for which the decision is obtained at the output of the NN, thus discarding hybrid solutions involving a NN but for which the decision is left to a heuristic.

One important feature to classify the solutions proposed in the literature is the way the state vectors are fed at the input of the NN architecture. Let $\mathbf{f}_\ell$ denote the $n_S \times 1$ column state vector of link ℓ where n_S is the number of the state components. We can identify two approaches: i) the state vectors are defined as single column $n_L n_S \times 1$ vectors $[\mathbf{f}_0^T, \mathbf{f}_2^T, \dots, \mathbf{f}_{n_L-1}^T]^T$ where T denotes the transposition operator and n_L is the number of links, ii) the state vectors are fed in series (one after the other) corresponding to gathering the vectors into the $n_S \times n_L$ matrix $[\mathbf{f}_0, \mathbf{f}_2, \dots, \mathbf{f}_{n_L-1}]$. We refer case i) to as vector state input (VSI) and ii) to as matrix state input (MSI).

The state input characterization is instrumental in explaining the SotA with regards to the three architecture properties listed in the introduction. We summarize the analysis of the publications identified in Table I along the following features: i) the state format, VSI and MSI), ii) the architecture property, PE, NLI, and GBM, iii) the information used to elaborate the state space: NP, HoL, xHoL, and APD.

In [3]–[6] the authors use a fully connected (FC) neural network architecture with a VSI. Since the size of the vector and thus the size of the NN input depends on n_L it thus cannot be NLI. It is also not PE since shuffling the state vectors does not provide the same outputs. In contrast, the NN architecture gets as an input the information of all the links and consequently is GBM. To get the NLI property while using the FC, [7] proposes to first select a fixed number of links using a PF heuristic and then to build a VSI with the selected links as input to the network. Since a FC is used, the architecture is not PE and since only a subset of the links are used to perform the scheduling, it is not GBM.

In [8]–[11] authors use a MSI approach and are thus NLI by construction since the input size of the NN architecture depends only on n_S. References [8], [9] use a long-short term memory (LSTM) architecture that involves internal memory (or hidden states). These solutions are not PE since the scheduling decision depends on the input order of the state vectors, and thanks to the memory it can be categorized as GBM. References [10], [11] use a FC architecture and since the inputs are processed independently it is thus PE but not GBM. Notice that in [11] combinations of VSI and MSI have been evaluated but for the sake of clarity Table I reports the best solution found.

From the SotA analysis, we deduce that the VSI approach along with a FC architecture cannot lead to the NLI or PE property. In [7] a trick is proposed to render the solution NLI but to the expense of the GBM property. Solutions based on MSI are NLI by construction but cannot achieve both PE and GBM at the same time. The solution proposed in this paper based on a EOT architecture distinguishes itself from the SotA since it allows to verify the three properties simultaneously.

Regarding the information used in the state vector, all the references use either HoL or NP, or both. Note that in [9] the information used is binary, it is equal to zero if the buffer is empty (i.e. NP = 0) or equal to one otherwise. In this work we propose two new state spaces, namely xHoL and APD.

TABLE I
DRL state of the art synthesis.

		Archi. property			
Ref.	State input	NLI	PE	GBM	State info.
[3]	VSI			✓	HoL
[4]	VSI			✓	NP, HoL
[5]	VSI			✓	NP, HoL
[6]	VSI			✓	NP, HoL
[7]	VSI	✓			NP
[8]	MSI	✓		✓	NP
[9]	MSI	✓		✓	$\mathbf{1}_{[\text{NP}]}$
[10]	MSI	✓	✓		NP, HoL
[11]	MSI	✓	✓		NP, HoL
Ours	MSI	✓	✓	✓	NP, HoL, xHoL, APD

Regarding the BO and DV constraints we have identified that [4], [5] consider both BO and DV constraints, [3], [6], [10], [11] consider the DV constraint, whereas the other references do not consider such constraints. This information is not reported in the table due to lack of space.

III. System Model

We consider a communication network with n_L active links characterized by the two kinds of traffic: DC and BE. For each traffic of each link, the ingoing packets are stored into FIFO buffers leading to a total number of $n_Q := 2n_L$ buffers, where $:=$ stands for "by definition". Let $\ell \in \{0, \dots, n_L - 1\}$ be the index of the link, and let $p \in \{0, 1\}$ be the index of their traffic. By convention, we consider that $p = 0$ (resp. $p = 1$) corresponds to DC (resp. to BE). A buffer associated with the ℓth link and pth traffic is numbered $i := 2\ell + p$. Consequently, $\ell_i := \lfloor i/2 \rfloor$ and $p_i := i \mod 2$ where ℓ_i and p_i are the link and the traffic related to buffer i. In the rest of the paper, we use equivalently i or (ℓ_i, p_i) for the buffer i.

We assume that a buffer can contain at most B packets. Packets arriving once a buffer is full are discarded and a BO occurs, thus leading to packet loss. We define the slot as the smallest unit of time. Each packet arriving in a buffer has a WT set to 0 that is incremented by 1 at each time slot. Let $d_{j,i}$ be the WT of the jth packet in the ith buffer with $j \in \{0, \dots, B-1\}$ assuming that the packets are ordered in the buffer according to their WT in the decreasing order, i.e. $d_{0,i} \geq d_{1,i} \geq \cdots \geq d_{B-1}$. By convention $d_{j,i} = -1$ represents an empty entry. For DC buffers, we have $d_{j,2\ell_i} \leq D$, where D is the maximum WT granted to a packet, i.e. the maximum number of slots a packet can be stored in the buffer. For BE buffers, we have $d_{j,2\ell_i+1} \in \{-1, 0, 1, 2, \dots\}$, since there is no delay constraint.

Let n_i denote the number of packets in buffer i during a given slot, i.e. the number of entries $d_{j,i}$ whose value is greater than -1. Note that $d_{0,i}$ is identified as the HoL of buffer i. Let n_i^{HoL} represent the number of packet with a delay equal to $d_{0,i}$ in buffer i. We note $\mathcal{C} := \{n_0^c, \cdots, n_{n_L-1}^c\}$, as the set of channel capacities for the different links, where n_ℓ^c represents the number of packets that the channel associated with link ℓ can support for error-free transmission. Finally, let $n_{\max}^c := \max \mathcal{C}$. At each slot, a buffer i is selected by the scheduler. The number of packets transmitted, denoted as $n_i^t := \min(n_{\ell_i}^c, n_i)$, corresponds to the oldest packets extracted from this buffer and sent through the channel.

For DC traffic, after packet extraction from the buffer, let $n_{2\ell_i}^d$ represent the number of remaining packets with a WT equal to D in buffer $2\ell_i$. Since these packets will have a WT equal to $D + 1$ in the next slot, thereby violating the delay constraint, they need to be removed. Such packets are thus discarded, leading to packet loss. Let also $n_{2\ell_i}^e := n_{2\ell_i}^t + n_{2\ell_i}^d$ denote the total number of packets that leave buffer $2\ell_i$, either due to extraction for transmission or due to DV.

For BE traffic, let $n_{2\ell_i+1}^e := n_{2\ell_i+1}^t$ denote the total number of packets that leave buffer $2\ell_i + 1$ solely due to extraction.

Then, the WT of each remaining packet is incremented by 1, and the number of remaining empty memories in the buffer is equal to $\kappa_i := B - n_i + n_i^e$. After that, n_i^r packets arrive following a Poisson distribution with parameter $\lambda_i \in [0, \lambda_{\max}]$. We note $\Lambda := \sum_{i=0}^{n_Q-1} \lambda_i$. There are $n_i^o := \max(0, n_i^r - \kappa_i)$ additional discarded packets due to BO. This process is repeated at each slot.

Our objective is to design a scheduler minimizing the PLR due to BO and DV.

IV. Problem formulation

The PLR minimization is formulated as minimizing the number of lost packets due to BO and DV over an infinite horizon by modeling the scheduling problem as a Markovian decision process (MDP). A MDP is characterized by: i) a set of states $\mathbf{s}_k$, ii) a set of actions a_k, iii) a reward $r(\mathbf{s}_k, a_k, \mathbf{s}_{k+1})$ which represents the gain obtained by transitioning from $\mathbf{s}_k$ to $\mathbf{s}_{k+1}$ after taking action a_k, and iv) the transition probabilities satisfying the Markov property: $\Pr\{\mathbf{s}_{k+1} | \mathbf{s}_k, a_k, \mathbf{s}_{k-1}, a_{k-1}, \dots\} = \Pr\{\mathbf{s}_{k+1} | \mathbf{s}_k, a_k\}$. The agents' actions are guided by a policy μ, such that $a_k := \mu(\mathbf{s}_k)$. The goal of the DRL is to determine the optimal policy μ^* that maximizes expected discounted return [12]:

$$\mu^* = \arg\max_\mu \mathbb{E} \left[\sum_{k=0}^{+\infty} \gamma^k r(\mathbf{s}_k, a_k, \mathbf{s}_{k+1}) \right] \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor. Solving (1) enables the scheduler to select the optimal action $a_k^* = \mu^*(\mathbf{s}_k)$.

In the following, we define tailored state and action spaces, as well as a reward function, to model a MDP that addresses the scheduling problem.

A. State definitions

Since the proposed solution is MSI we define the state of the system at time k by the $n_f \times n_L$ matrix $\mathbf{s}_k := [\mathbf{f}_0, \dots, \mathbf{f}_{n_L-1}]$ where $\mathbf{f}_\ell$ is the $n_f \times 1$ column state vector of the ℓth link with n_f its number of elements (for the sake of notation clarity we drop the k index for $\mathbf{f}_\ell$). The value of n_f depends on the considered features to represent the state vector for both the DC and BE traffics. For the BE traffic, we always consider the NP in the buffer, i.e. $n_{2\ell+1}$. We also always consider the link capacity as a feature of the state, i.e. n_ℓ^c. For the DC traffic we consider three different models as introduced in Section I, hence the state vector can be written as $\mathbf{f}_\ell := [\mathbf{f}_\ell^{\text{DC-}x}, n_{2\ell+1}, n_\ell^c]^T$ where $\mathbf{f}_\ell^{\text{DC-}x}$ is the vector composed with the features specific to the DC traffic for the state space of type x. As a consequence, n_f is equal to the number of entries of $\mathbf{f}_\ell^{\text{DC-}x}$ plus 2. We now define the $\mathbf{f}_\ell^{\text{DC}-x}$ vectors for the three state space models.

1) State-HoL (S-HoL): This model considers the features used in the SotA, i.e. the NP and the HoL, hence:

$$\mathbf{f}_\ell^{\text{DC-S-HoL}} := [d_{0,2\ell}, n_{2\ell}]. \tag{2}$$

In that case we have $n_f = 4$.

2) State-xHoL (S-xHoL): In this model, we slightly improve the S-HoL one by adding the HoL multiplicity, i.e. $n_{2\ell}^{\text{HoL}}$ the number of packets having the HoL value:

$$\mathbf{f}_\ell^{\text{DC-S-xHoL}} := [n_{2\ell}^{\text{HoL}}, d_{0,2\ell}, n_{2\ell}]. \tag{3}$$

In that case we have $n_f = 5$.

3) State-APD (S-APD): In this model we consider the full DC buffer information. One possibility is to set the state vector with all the packet WT values, i.e. $[d_{0,2\ell}, \dots, d_{B-1,2\ell}]$ of length B. Another possibility is to set the state vector with the instantaneous distribution of the packet WT values $[p_{0,2\ell}, p_{1,2\ell}, \dots, p_{D+1,2\ell}]$ of length $D + 2$ defined as the WT histogram normalized by B. Both solutions contain the same information but we propose to use the second one. Indeed, it is very likely that $B \gg D + 2$ and thus the second representation offers a smaller the size for the input vectors, easing the implementation and training. Moreover, the size of the link state is independent of the buffer size. Therefore, we have:

$$\mathbf{f}_\ell^{\text{DC-S-APD}} := [p_{0,2\ell}, p_{1,2\ell}, \dots, p_{D+1,2\ell}]. \tag{4}$$

In that case we have $n_f = D + 4$. The impact of the state vector size on the complexity with regard to the state space models is discussed at the end of Section V.

B. Action space

Let $\mathcal{A} := \{0, \cdots, n_Q - 1\}$ be the action space, i.e. the set of available actions for the scheduler. The action $a_k = i$ corresponds to selecting the buffer i at slot k triggering the transmission of n_i^t packets over the channel.

C. MDP model

The transition from a state $\mathbf{s}_k$ to the next state $\mathbf{s}_{k+1}$ depends on the number of extracted packets and the number of arriving packets in each buffer, represented by the vector $\mathbf{n}^r := [n_0^r, \dots, n_{n_Q-1}^r]$. From the state and action space definitions in Section IV-A and Section IV-B, and following the same approach as in [3], one can prove that the considered models are MDP since we can identify a transition function t such that $\mathbf{s}_{k+1} = t(\mathbf{s}_k, a_k, \mathbf{n}^r)$ showing that $\mathbf{s}_{k+1}$ depends only on $\mathbf{s}_k$ and a_k (details are omitted).

D. Reward

Let r_o be the reward associated with BO that we define as the opposite of the number of packets discarded due to BO in the DC and BE buffers for all links:

$$r_o(\mathbf{s}_k, a_k, \mathbf{s}_{k+1}) := -\sum_{\ell=0}^{n_L-1} (n_{2\ell}^o + n_{2\ell+1}^o), \tag{5}$$

and r_d be the reward associated with DV that we define as the opposite of the number of packets discarded due to DV for the DC buffers for all links:

$$r_d(\mathbf{s}_k, a_k, \mathbf{s}_{k+1}) := -\sum_{\ell=0}^{n_L-1} n_{2\ell}^d. \tag{6}$$

It is worth noticing that n_i^o and n_i^d in (5) and (6) are taken at step k.

In order to strongly penalize the loss of packets, we define the global reward r as the sum of the exponential of both (5) and (6), which can be written as:

$$r(\mathbf{s}_k, a_k, \mathbf{s}_{k+1}) = \frac{e^{\omega r_d(\mathbf{s}_k, a_k, \mathbf{s}_{k+1})} + e^{\omega r_o(\mathbf{s}_k, a_k, \mathbf{s}_{k+1})}}{2}, \tag{7}$$

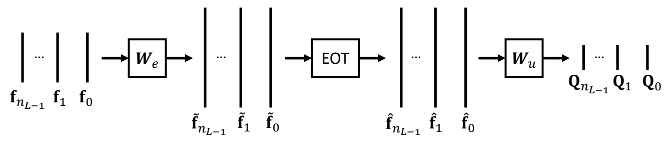
where the parameter $\omega > 0$ is a hyperparameter controlling the behavior of the exponential function.

V. Problem solution

As explained in Section II, the MSI approach is NLI by construction but neither a FC nor a NN architecture with hidden states (such as LSTM) allows to achieve both PE and GBM at the same time. This is the reason why we propose to use an EOT architecture [13] that is able to achieve the three properties. First, it is NLI since we are in the MSI case. Second the EOT is GBM by construction thanks to the attention mechanism that combines all the input vectors through the scale dot product. Third, we render the EOT PE by removing the positional encoding [14].

Since we are dealing with potentially high dimension state space (see Section IV), we train the scheduler architecture using the DQL algorithm proposed in [2]. Thus, the output of our architecture for each input $\mathbf{f}_\ell$ is a 2×1 vector $\mathbf{Q}_\ell$ representing the Q-values for the DC and BE traffics. Then, the buffer to be scheduled is determined by an $\arg\max$ over the n_L Q-value vectors. In addition, to avoid choosing an empty buffer, we apply action masking [15] before selecting an action.

Our architecture is depicted in Fig. 1 and synthesized in Algorithm 1. The input state vectors are first passed through



$\mathbf{f}_{n_L-1}$ ... $\mathbf{f}_1$ $\mathbf{f}_0$
W_e
$\tilde{\mathbf{f}}_{n_L-1}$ ... $\tilde{\mathbf{f}}_1$ $\tilde{\mathbf{f}}_0$
EOT
$\hat{\mathbf{f}}_{n_L-1}$ ... $\hat{\mathbf{f}}_1$ $\hat{\mathbf{f}}_0$
W_u
$\mathbf{Q}_{n_L-1}$ ... $\mathbf{Q}_1$ $\mathbf{Q}_0$

Fig. 1. The proposed architecture with the EOT.

an affine embedding, $\tilde{\mathbf{f}}_\ell := \boldsymbol{W}_e \mathbf{f}_\ell + \mathbf{b}_e$, where $\boldsymbol{W}_e$ is a matrix of dimensions $d_e \times n_f$, and $\mathbf{b}_e$ is a vector of dimensions $d_e \times 1$. Note that we assume $d_e > n_f$. The $d_e \times 1$ vectors $\tilde{\mathbf{f}}_\ell$ are then entered in the EOT block, see [16] for implementation details. The EOT performs multi-head attention on H heads and the size for each head is d_{attn}. The fully connected feed-forward network in the EOT has d_e inputs, the hidden layer has d_{mlp} neurons with ReLU activation function, and the number of outputs is equal to d_e. The $d_e \times 1$ output vectors $\hat{\mathbf{f}}_\ell$ of the EOT are then passed through an affine unembedding, $\mathbf{Q}_\ell := \boldsymbol{W}_u \tilde{\mathbf{f}}_\ell + \mathbf{b}_u$, where $\boldsymbol{W}_u$ is a matrix of dimensions $2 \times d_e$ and $\mathbf{b}_u$ is a vector of dimensions 2×1. Defining $\mathcal{Q} := [\mathbf{Q}_0(0), \mathbf{Q}_0(1), \dots, \mathbf{Q}_{n_L-1}(0), \mathbf{Q}_{n_L-1}(1)]$, the buffer i^* to be scheduled is then deduced by $i^* := \arg\max_i \mathcal{Q}$. Note that $\boldsymbol{W}_e$, $\mathbf{b}_e$, $\boldsymbol{W}_u$, and $\mathbf{b}_u$ are learned during the training along with the EOT parameters. Note also that due to the embedding, the size of the space vectors n_f has a very limited impact on the global complexity. Indeed, the embedding transforms the $n_f \times 1$ vectors into $d_e \times 1$ ones with $d_e > n_f$ which is a constant regardless of the state model. Thus the complexity is mainly driven by d_e.

VI. Numerical results

This section studies the performance of the different solutions, FC and EOT, by evaluating their generalization capabilities with respect to Λ, n_L, and $\mathcal{C}$. To achieve this, we

Algorithm 1: Forward pass of architecture in Fig. 1

```
Input: ${\mathbf{f}_\ell}_{\ell=0}^{n_L-1}$
Output: $i^*$
for $\ell \in \{0,\dots,n_L-1\}$ : $\tilde{\mathbf{f}}_\ell \leftarrow \boldsymbol{W}_e \mathbf{f}_\ell$
$\{\hat{\mathbf{f}}_0,\dots,\hat{\mathbf{f}}_{n_L-1}\} \leftarrow \mathrm{EOT}(\tilde{\mathbf{f}}_0,\dots,\tilde{\mathbf{f}}_{n_L-1})$
for $\ell \in \{0,\dots,n_L-1\}$ : $\mathbf{Q}_\ell \leftarrow \boldsymbol{W}_u \tilde{\mathbf{f}}_\ell$
$\mathcal{Q} \leftarrow \{\mathbf{Q}_\ell,\dots,\mathbf{Q}_{n_L-1}\}$
return $i^* = \arg\max_i \mathcal{Q}$
```

train the architectures using fixed values for some of these parameters and then evaluate their performance by varying these parameters. For each training and inference, we assume that the λs are equal for each buffer, with $\lambda = \Lambda/n_Q$.

A. Training setup

We train the proposed EOT architecture associated with the three types of states defined in Section IV-A: S-HoL, S-xHOL, S-APD that are denoted "EOT-HoL", "EOT-xHoL", and "EOT-APD", respectively in the sequel. For the sake of comparison with the SotA, we also implemented a VSI scheduler using a FC trained for the same state spaces, denoted as "FC-HoL", "FC-xHoL", and "FC-APD".

For the EOT, we select: i) a fixed number of link $n_L = 6$, since the architecture is NLI, which corresponds to $n_Q = 12$ buffers, ii) a fixed channel capacity configuration, $\mathcal{C} = \{1,1,2,2,3,3\}$. The architecture is trained using 2000 episodes of 7000 steps, with the following parameters: $H = 4$, $d_e = d_{\text{mlp}} = Hd_{\text{attn}} = 256$, and a learning rate equal to 5×10^{-4}.

For the FC, since it is not NLI, we train the architecture for $n_L \in \{4,6,8,10,12\}$, thus five different NNs. We assume that the channel capacities belong to $\{1,2,3\}$. We characterize the simulated per link-channel capacities by introducing $\mathbf{n}_b^c$, whose kth entry corresponds to the number of links with capacity k. Note that this representation is not equivalent to $\mathcal{C}$ since any arrangement of $\mathcal{C}$ leads to the same $\mathbf{n}_b^c$. The entries of $\mathcal{C}$ are drawn to satisfy the $\mathbf{n}_b^c$ indicated in Table II. Once drawn, the entries of $\mathcal{C}$ are sorted in ascending order, as done for the EOT with $n_L = 6$. Notice that the median of the $\mathcal{C}$s obtained using this procedure is always greater or equal to 2. The architectures are trained using 4000 episodes of 15 000 steps, with the following parameters: two hidden layers with ReLU activation functions, 512 neurons per layer, and a learning rate equal to 5×10^{-5}.

For all the trainings and both architectures, the schedulers are trained for $\Lambda = 1.8$, $D = 20$, and $B = 40$. The training parameters are: $\gamma = 0.99$, a batch size equal to 64, and a target network update equal to 50 steps. An ϵ-greedy approach is used for the exploration. The value of ϵ at episode k is given by $\epsilon_k = \max(0.99 \times \epsilon_{k-1}, 0.01)$, with $\epsilon_0 = 1$.

B. Inference setup

We assess the generalization capabilities of the trained architectures along two cases: i) with respect to (wrt) Λ, and ii) wrt n_L and $\mathcal{C}$.

TABLE II
$\mathbf{n}^c$ VALUES USED FOR THE TRAINING.

n_L	4	6	8	10	12
$\mathbf{n}_b^c$	$[1,2,1]$	$[2,2,2]$	$[3,2,3]$	$[3,4,3]$	$[4,4,4]$

1) Generalization wrt Λ: Inferences are conducted with $n_L = 6$ and $\mathcal{C} = \{1,1,2,2,3,3\}$ for both EOT and FC, while Λ varies within the range $[1.5, 1.75]$. This allows us to assess the generalization capability of both architectures wrt Λ. For each value of Λ, the inference is performed over a single episode consisting of 1×10^6 steps. We verified that the PLR variance is sufficiently small when using one single episode with a high number of steps.

2) Generalization wrt n_L and $\mathcal{C}$: The inferences are conducted with $\Lambda = 1.5$ for both EOT and FC, while varying the number of links, with $n_L \in \{4,6,8,10,12\}$. The value of Λ is adjusted slightly from the training setup to keep the PLR low for $n_L = 6$, allowing the dynamics of PLR degradation to be observed for $n_L > 6$. For each n_L, the channel capacities, identical for both architectures, are drawn from a uniform distribution in $\{1,2,3\}$, and thus are not sorted unlike during training. To maintain the same setting as in the training phase, we ensure that the median of $\mathcal{C}$ is greater or equal to two. Furthermore, we ensure unique channel configurations for each n_L. The number of evaluated channel configurations depends on n_L and is equal to $38, 65, 78, 86, 86$ for $n_L = 4,6,8,10,12$ links respectively. In summary, we assess the generalization capability wrt i) the channel capacities for both architectures, and ii) the number of links for the EOT, as it is trained only for $n_L = 6$. This makes the generalization task more challenging for the EOT. Additionally, the fact that the $\mathcal{C}$s used for training are sorted, while those used for inference are not, allows us to evaluate the impact of the PE property on performance.

C. Performance results

We compare the performance of the inferred schedulers in terms of PLR, defined as the ratio between the number of lost packets due to either DV or BO and the number of arrived packets in the buffers.

1) Generalization wrt Λ: Fig. 2 depicts the PLR of the different schedulers versus the sum of arrival rate Λ. Each point corresponds to the PLR obtained at the end of the episode. The following observations can be made. First, the performance of EOT-APD and EOT-xHoL is very similar, regardless of Λ. Second, for $\Lambda \in [1.5, 1.65]$, the EOT-xHoL and EOT-APD schedulers outperform their FC counterparts. For example, at $\Lambda = 1.55$, the PLR of the EOTs is approximately ten times smaller than the FCs. Last, the EOT-xHoL significantly outperforms the EOT-HoL when $\Lambda < 1.6$ and it is very close to the EOT-APD. This highlights the benefit of incorporating the multiplicity of the HoL in the state space.

2) Generalization wrt n_L and $\mathcal{C}$: Fig. 3 shows the PLR of the different schedulers averaged over the channel capacity realizations, as a function of the number of links n_L. The

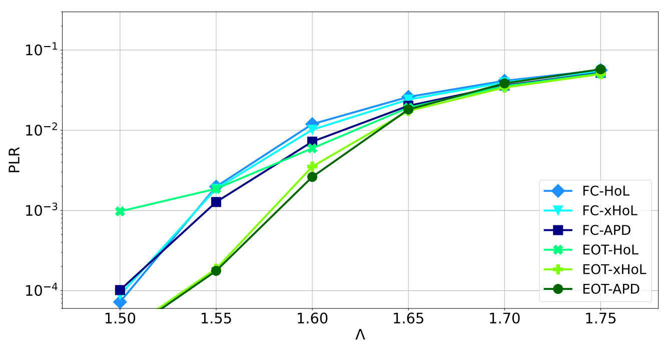


Fig. 2. PLR wrt Λ for $n_L = 6$, $\mathcal{C} = [1, 1, 2, 2, 3, 3]$, $\mathbf{n}_b^c = [2, 2, 2]$.

following observation can be made. First, the PLR of the EOT schedulers is very similar for the three considered state spaces as soon as $n_L > 4$, and increases monotonically with n_L. Second, the PLR of the FC schedulers varies within the range $[1 \times 10^{-2}, 1 \times 10^{-1}]$, regardless of n_L, and is consistently worse than that of the EOT schedulers, despite the fact that the FC schedulers are specifically trained for each n_L. This highlights the strong generalization capability of the proposed EOT schedulers. For $n_L = 6$, both FC and EOT architectures generalize only wrt the channel capacities. In this case, the EOT significantly outperforms the FC, which can be attributed to the PE property of the EOT. For $n_L \neq 6$, the EOT continues to outperform the FC, demonstrating its ability to generalize with respect to both the channel capacities and the number of links.

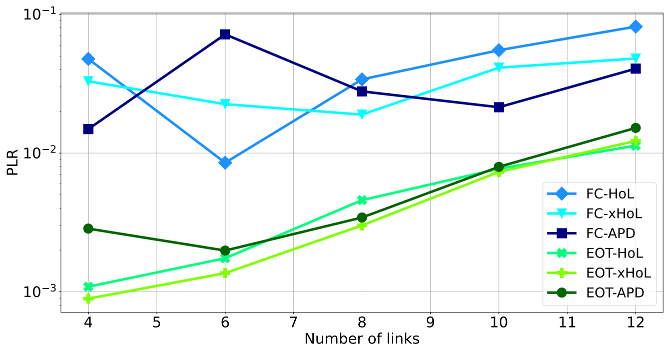


Fig. 3. PLR wrt n_L for $\Lambda = 1.5$ and $n_i^c \in \{1, 2, 3\}$.

VII. Conclusion

In this paper, we addressed the scheduling problem with the objective of minimizing the PLR, where packet loss may result from DV and BO. We considered two types of traffic: packets with strict DC, and BE packets. As alternatives to the conventional HoL state space, we proposed the xHoL, which incorporates the multiplicity of the oldest packet in the buffer, and the APD, which accounts for the WT of all packets in the buffer. We introduced a scheduler based on DQL using an EOT architecture. This architecture is capable of handling a variable number of links, considers the entire buffer information, and is permutation equivariant.

Through simulations, we demonstrated that the proposed EOT scheduler outperforms FC schedulers from the SotA. Notably, this is true even when comparing the EOT, trained with a specific number of links, to a FC scheduler specifically trained for the same number of links. This underscores the advantages of PE architectures. Training the EOT with a variety of link counts is expected to further enhance its generalization performance. Additionally, we observed that the xHoL state space improves upon the conventional HoL state space, achieving performance close to that of the APD.

In the future, it would be interesting to study: i) the effect of training the EOT with several values of n_L, ii) the robustness of the proposed scheduler to random channel, iii) solutions to perform the scheduling for different resource blocks.

References

[1] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[3] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and B. Vucetic, "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, 2021.

[4] V. H. L. Lopes, C. V. Nahum, R. M. Dreifuerst, P. Batista, A. Klautau, K. V. Cardoso, and R. W. Heath, "Deep reinforcement learning-based scheduling for multiband massive MIMO," *IEEE Access*, vol. 10, 2022.

[5] C. Xu, J. Wang, T. Yu, C. Kong, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Buffer-aware wireless scheduling based on deep reinforcement learning," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[6] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[7] A. Anand, R. Balakrishnan, V. S. Somayazulu, and R. Vannithamby, "Model-assisted deep reinforcement learning for dynamic wireless scheduling," in *Asilomar Conference on Signals, Systems, and Computers*, 2020.

[8] A. Robinson and T. Kunz, "Downlink scheduling in LTE with deep reinforcement learning, LSTMs and pointers," in *IEEE Military Communications Conference (MILCOM)*, 2021.

[9] F. AL-Tam, A. Mazayev, N. Correia, and J. Rodriguez, "Radio resource scheduling with deep pointer networks and reinforcement learning," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020.

[10] J. S. Shekhawat, R. Agrawal, K. G. Shenoy, and R. Shashidhara, "A reinforcement learning framework for QoS-driven radio resource scheduler," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[11] A. Paz-Pérez, A. Tato, J. J. Escudero-Garzás, and F. Gómez-Cuba, "Flexible reinforcement learning scheduler for 5G networks," in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2024, pp. 566–572.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] H. Xu, L. Xiang, H. Ye, D. Yao, P. Chu, and B. Li, "Permutation equivariance of transformers and its applications," 2024. [Online]. Available: https://arxiv.org/abs/2304.07735

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

[16] M. Phuong and M. Hutter, "Formal algorithms for transformers," *arXiv preprint arXiv:2207.09238*, 2022.