Reinforcement Learning-based UL/DL Splitter for Latency Reduction in Wireless TSN Networks

Margarita Cabrera-Bean, Wenli Pan, Josep Vidal
Dept. of Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Spain
{marga.cabrera,josep.vidal}@upc.edu

***Abstract*—Reducing latency in Time-Sensitive Networking (TSN) networks is critical to fulfil real-time communication requirements, ensuring timely data delivery, and maintaining system responsiveness. Minimizing latency enhances the reliability of industrial automation, multimedia streaming, and other time-critical applications, ultimately optimizing overall network performance and user experience. One of the most critical points lies in the wireless segments, which cannot be considered as deterministic. Specially, when the traffic load between uplink and downlink is unbalanced, it is critical to allocate resources based on the volume of such traffic and on the channel state, both significantly impacting on packet latency. In this paper we present and compare a set of approaches to scheduling time slots within a wireless frame for communication between the uplink (UL) and downlink (DL) in a TSN network. The primary objective is to reduce latency in wireless transmissions, particularly in scenarios with stringent timing requirements. By optimizing the allocation of time slots between UL and DL, our proposed scheduling algorithm aims to minimize queueing delays while ensuring efficient utilization of network resources. The results highlight the significant reduction achieved in terms of queueing latency and packet loss through our scheduling strategy, thereby enhancing the reliability and timeliness of wireless links in TSN networks.**

***Index Terms*—UL/DL Scheduling, Reinforcement Learning (RL), Latency in TSN, Queueing Delay.**

I. Introduction

Time-Sensitive Networking (TSN) expands the standard Ethernet protocol to facilitate real-time synchronization and ensure deterministic, low-latency communication. TSN introduces essential elements such as time-aware shapers, schedulers, and guard bands, which are vital for applications demanding high availability, robustness, and reliability. These components enable communication latency to be deterministic and bounded. Facilitating interoperable wireless TSN capabilities that align with established wired TSN standards is crucial to foster widespread industry acceptance of TSN solutions. In systems where wireless segments are integrated with wired segments systems, TSN (Time-Sensitive Networking) traffic communications experience the inherent randomness of wireless channels in the wireless segments. Recently, significant advancements have been made in enabling TSN-based time synchronization and ensuring bounded latency over WiFi, ensuring high reliability for industrial automation systems. [1] describes ongoing TSN activities in interoperability testing and certification toward WiFi TSN, demonstrating the effectiveness of capabilities in IEEE 802.11ax and IEEE 802.11be to achieve deterministic operation. Moreover, TSN communications extend beyond just WiFi wireless segments, as demonstrated in [2], which outlines an integrated 5G and TSN network design and underscores the potential for substantial benefits in industrial use cases. An additional challenge in TSN communications arises from the coexistence of multiple types of traffic. This includes asynchronous Quality of Service (QoS) or Best Effort (BE) traffic mixed with TSN isochronous traffic. Each type of traffic is conveyed by a set of flows that arrive at the system randomly. The coexistence of the three types of traffic in the wireless segments requires smart scheduling based on the traffic type [3]. In [4], Markov chains were used to study the effect of aperiodic traffic on the queuing delay of periodic traffic, highlighting queuing delay as one of the most critical components of end-to-end delay. Additionally, some previous works, such as [5], proposes heuristic procedures to schedule multiple types of traffic, thereby supporting mixed-criticality applications in TSN.

A. Contribution

Our work focuses on the distribution of transmission time between UL and DL in a Time Division Duplex (TDD) wireless segment. The main goal is to ensure that packets are transmitted before a deadline. This is usually a critical requirement in TSN networks. Packets exceeding this waiting time are considered lost. We propose both, heuristic methods that does not require a training stage and machine learning-based techniques to achieve this objective. In both approaches, dynamic splitters are employed to distribute the time slots of each frame between DL and UL. This distribution depends on the volume of packets awaiting transmission and optionally considers the transmission bitrate determined by the current conditions of the wireless channel.

B. Previous RL based works on wireless scheduling

In recent years, intelligent communication has garnered significant research attention from both academia and industry. Reinforcement learning (RL) and deep reinforcement learning (DRL) have emerged as powerful artificial intelligence techniques capable of learning optimal decisions based on

This work is funded by the European Commission Horizon Europe SNS JU PREDICT-6G (GA 101095890) Project and the NextGeneration UNICO5G TIMING (TSI-063000-2021-145). Also by the project 6-SENSES grant PID2022-138648OB-I00, funded by MCIN/AEI/10.13039/501100011033, and by FEDER-UE, ERDF-EU *A way of making Europe*, and the grants 22CO1/008248 and 2021 SGR 01033 (AGAUR, Generalitat de Catalunya).

environmental feedback. In [6] a survey is presented specially focused to emphasize the progress on RL and DRL applied to mobile edge computing, software defined network and network virtualization in 5G. Previously in [7] a detailed survey presented some pioneering works in wireless sensor networks, working in applications such as resource and power optimization.

In [8], DRL techniques are applied to typical resource management scenarios to determine network slicing strategies, a key enabler for 5G. Similarly, in [9] and [10], DRL techniques are used to manage Massive Machine-Type Communication, enhancing random access scalability and suitability. In [11] the problem of scheduling using DRL is combined with Successive Interference Cancellation or SIC.

The problem of reducing UL latency by using DRL for uRLLC services is solved in [12] by dynamically allocating the future UL grant by learning from the dynamic traffic pattern, while in [13] transfer learning and cooperative learning mechanisms are employed to enable communication links to work cooperatively in a distributed manner, which enhances the network performance and access success probability in Ultra-Reliable Low-Latency Communications.

In [14] a dual reinforcement learning based pattern optimization scheme is proposed for dynamic TDD 5G systems. The proposed solution targets minimizing the required URLLC radio latency on a real-time basis, and accordingly, improving the achievable URLLC outage performance. The proposed scheme utilizes two nested layers, where the primary layer estimates the number of the DL and UL symbols of the upcoming radio pattern to satisfy the foreseen offered traffic and the secondary RML sub-layer determines the DL and UL radio frame structure that achieves the minimum. The proposed solution demonstrates a significant URLLC outage latency improvement compared to baseline dynamic TDD proposals. Unlike our approach, [14] does not apply deep learning techniques nor does it consider transmission bitrates in the decision-making process, which, in our work, demonstrates a positive impact on reducing both, latency and packet loss.

The rest of the paper is organized as follows: Section II presents the environment model and formulates the problem addressed in this work. Section III introduces and describes the five proposed UL/DL splitters, including one static and four dynamic splitters. Section IV evaluates and compares the performance of these splitters. Finally, conclusions are presented in Section V.

II. Environment

In the wireless segment of a TSN network, multiple users establish connections with a Base Station (BS) i.e. an Access Point in Wi-Fi or a gNB in 5G, to exchange different flows with the network, including Time Sensitive (TS), with latency constraints and best-effort (BE) flows, with no minimum rate or latency constraints. TS flows include isochronous, periodic traffic from control devices (e.g., sensors and actuators) and aperiodic bursty traffic from other TS applications such as AR/VR. The flows of periodic nature are pre-assigned to time slots in the wireless frame externally from a higher level, while BE type flows have no latency restrictions. In this work, we focus on the allocation of time slots non occupied by isochronous flows between UL and DL for the transmission of aperiodic and bursty TS packets corresponding to the users served by a single BS.

The DL/UL splitter subsystem operates at the periodicity of a wireless frame. At the onset of each frame, depending on the state of the asynchronous TS traffic queues and possibly other factors, it decides the proportion of free slot distribution between DL and UL. Once the isochronous traffic has been allocated in the wireless frame (the occupied slots in Fig. 1), the number of non-occupied slots (free slots in Fig. 1), are dynamically distributed between UL and DL.

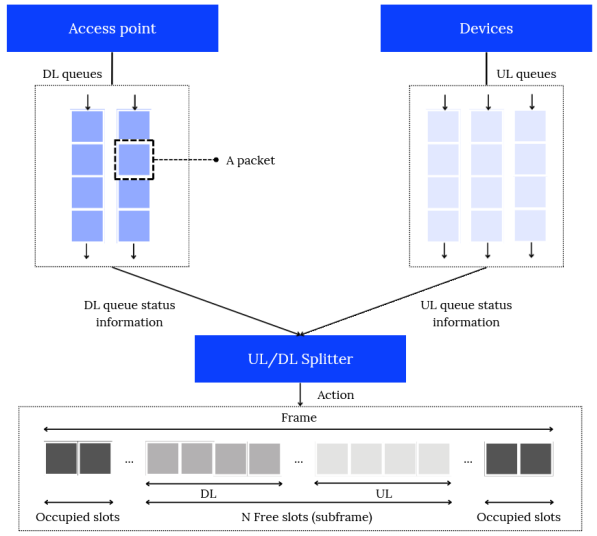

Fig. 1. UL/DL splitter scheme within Wi-Fi domain.

At every frame time (t), the splitter decides on the allocation of free time slots between UL and DL. This involves selecting an action A_t, from a set of potential actions, $\mathcal{A} := \{a_1, ..., a_{N_a}\}$, to select the UL/DL time slots distribution, where $N_a = |\mathcal{A}|$ is the size of the set of actions $\mathcal{A}$. Fig. 2 shows an example where N_a patterns are used providing N_a different actions. Action $a = 0$ means that all the free slots are assigned for UL, $a = 1$ means that a set of N_{DL} slots is assigned for DL while $N - N_{DL}$ slots are assigned for UL, and so on.

A FIFO queue is associated to each flow to keep incoming packets waiting to be transmitted through the wireless link. The transmission rate of each flow, depends on the wireless channel conditions, modelled through the coherence time and the instantaneous signal to interference plus noise ratio (SINR). Depending on the BLock Error Rate (BLER) required, a Modulation and Coding Scheme (MCS) is adopted which determines the transmission rate of the corresponding flow. In this work, a range of algorithms is introduced, which adopt the action taken in each frame based on all or some of the following pieces of information at time t:

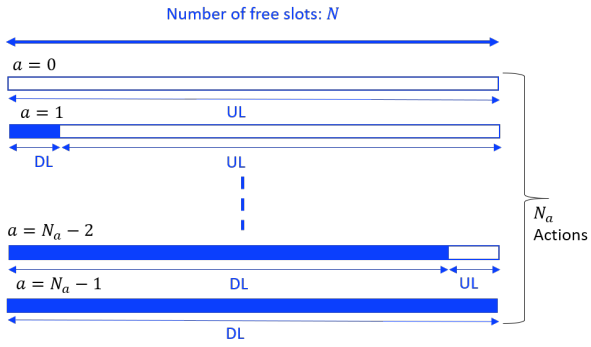

Fig. 2. Set of actions.

- $q_{DL}(t-1,n)$: instantaneous aggregated content of the DL queues associated to the DL flows at the end of each slot n^{th} in previous frame $t-1$.
- $q_{UL}(t-1,n)$: instantaneous aggregated content of the UL queues associated to the UL flows at the end of each slot n^{th} in previous frame $t-1$.
- $R_{DL}(t)$: transmission rate averaged over the users in the DL for the current starting frame.
- $R_{UL}(t)$: transmission rate averaged over the users in the UL for the current starting frame.

Note that $R_{DL}(t)$ and $R_{UL}(t)$ are computed once the MCS for each flow at the frame time periodicity is determined. To keep the queueing delay of all aperiodic TS packets within moderate limits, it is crucial to maintain a balanced state between the UL flow queues and the DL ones. To measure the average imbalance at the frame periodicity, we introduce the variables $\mu_q(t)$ and $\mu_r(t)$, measured at the end of the $(t-1)-th$ frame, by averaging the instantaneous imbalance over the N time slots of the frame:

$$\mu_q(t) = \frac{1}{N}\sum_{n=1}^{N}\frac{q_{DL}(t-1,n)}{q_{DL}(t-1,n)+q_{UL}(t-1,n)} \quad (1)$$

$$\mu_r(t) = \frac{R_{DL}(t)}{R_{DL}(t)+R_{UL}(t)}. \quad (2)$$

While average ratio $\mu_q(t)$ in (1) measures the imbalance between UL and DL queues, which is influenced by actions in previous frames and traffic arrival statistic, the ratio $\mu_r(t)$ in (2) depends solely on the conditions of the wireless channel of every use. Note that for the sake of simplicity a single flow is associated to each user. Regarding the queue management, queues are operated in FIFO mode and are managed using a circular buffer with a size of N_b packets. The size is measured in packet units for simplicity. Packets in the queue are considered lost if either their waiting time in queue exceed the maximum latency requirement (δ_{max}) for their traffic type or if a new packet arrives when the queue is full, in which case the packet at the front of the queue (the first position) is discarded. In neither of the two aforementioned circumstances is the packet transmitted.

Given the above, the problem is stated as follows. At the start of each frame, a decision must be made regarding the allocation of free time slots between UL and DL. This decision should aim to keep the flow queues as balanced as possible, with the objective of minimizing a target function specified below that aims for $\mu_q(t) = 0.5$, that is, to achieve a balance between UL and DL. Additionally, our goal is to minimize the number of lost packets in queue, whether due to excessive waiting time or maximum queue occupancy.

III. Methodology

In this section the five proposed splitter algorithms are described.

A. Static Splitter

Static Splitter (SS) consist into a fixed division that assigns same number of time slots to UL and to DL and it is included for the purpose of comparison. It does not consider any information about the environment and the only action taken at the beginning of each frame consist in allocating exactly half of the time slots to UL and the other half to DL.

B. Heuristic Strategies

The first heuristic strategy does not use averaged transmission rates while the second one incorporates information about the radio channel over time.

a) Queue-size proportional splitter: QSPS is proposed to fulfil the objective of having a set of free time slots allocated to UL (DL) that is proportional to the queue size $q_{UL}(q_{DL})$, so that we can accommodate traffic excess with low outage. QSPS consists on assigning to DL a number of slots, $N_{DL}(t)$, proportional to the size of the aggregated DL queues by using a quantified value of $\mu_q(t)$ in (1). If N_a is the size of the action set, time slots are assigned in groups of size $\Delta_a = \frac{N}{N_a-1}$ as in (3):

$$N_{DL}(t) = (i-1)\Delta_a \quad \text{if} \quad i-1 \le N\mu_q(t) < i \quad (3)$$

b) Waiting-time proportional splitter: WTPS takes advantage of the average transmission bit rates knowledge, which are chosen per flow i as the maximum UL and DL transmission rates that meet the BLER requirement of the wireless physical link, $R_i^{DL}(t)$, $R_i^{UL}(t)$. This strategy aims to allocate time slots proportionally to the average time required to empty the UL (DL) queue. Therefore, the percentage of time slots devoted to DL, at the start of frame t results:

$$\mu_T(t) = \frac{T_{DL}(t)}{T_{DL}(t)+T_{UL}(t)} \quad (4)$$

where $T_{DL}(t)$ and $T_{UL}(t)$ remain constant throughout the entire frame and are defined as

$$T_{DL}(t) = \sum_{i=1}^{F_{DL}}\frac{q_i^{DL}(t)}{R_i^{DL}(t)}; \quad T_{UL}(t) = \sum_{i=1}^{F_{UL}}\frac{q_i^{UL}(t)}{R_i^{UL}(t)} \quad (5)$$

where $F_{DL}(F_{UL})$ is number of flows with corresponding transmission rate $R_i^{DL}(R_i^{UL})$ in the DL(UL).

So, the allocation of time slots for the DL is determined at the beginning of each frame as in (3) by using $\mu_T(t)$ given in (4) instead of $\mu_q(t)$ given in (1).

C. RL based methods

Reinforcement Learning (RL) is the machine learning branch devoted to decision making, by learning the optimal behavior in an environment that generates feedback in response to decisions, with the goal of maximizing the long-term expected reward. RL follows the mathematical framework of the Markov decision process followed in [7] and in [9], where the learning outcomes are partially random and tightly related to the environment. An RL agent in a given state S_t selects an action A_t for the environment. The state changes after the environment accepts the action. Meanwhile, a reward feedback R_{t+1} is generated to the agent. The agent selects the next action A_{t+1} according to the current state of the environment with the goal of maximizing the long-term reward, G_t, also named discounted cumulative return and defined as:

$$\begin{aligned} G_t = \sum_{k=1}^{\infty} \gamma^{k-1} \cdot R_{t+k} = \\ = 1 \cdot R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + ... + \gamma^{\infty} \cdot R_{\infty}. \end{aligned} \tag{6}$$

In (6) $\gamma \in (0, 1]$ is the discount factor through which immediate rewards are valued more than future ones. Accordingly, the goal of an RL agent is to learn an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$, which determines an action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$, thus, to optimally maximize or minimize a pre-defined value function V^{π} which is typically expressed in terms of G_t as:

$$V_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] \tag{7}$$

To better determine the optimal policy, the action-state value function defined as:

$$\begin{aligned} Q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] = \\ = \mathbb{E}\left[R_t + \gamma V_{\pi}(s) | S_t = s, A_t = a\right] \end{aligned} \tag{8}$$

denotes the expected long-term return when at state $S_t = s$ the RL agent performs action $A_t = a$ following the policy π. The goal of training an RL agent is to find an optimal strategy, that is, the policy that gets the most return. All the optimal policies share the same optimal action-value function, denoted Q^*, and defined as:

$$Q^*(s, a) \triangleq \max_{\pi} Q^{\pi}(s, a) \tag{9}$$

a) QL, without use of averaged transmission rates: The Q-Learning (QL) approach is one of the most effective model free approaches to rapidly learn an optimal policy π^* by estimating the function $Q^*(s, a)$ iteratively, according to the Bellman equation-based iteration:

$$Q(S_t, A_t) \leftarrow (1-\alpha) Q(S_t, A_t) + \alpha [r_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')] \tag{10}$$

Where α is the learning rate which affects the learning speed of the Q-table $Q(s, a)$. The convergence of the previous iteration is guaranteed by the fixed-point theorem. In this work the QL based UL/DL splitter is applied by defining state S_t at the beginning of frame t as the quantified version of the queue imbalance rate $\mu_q(t)$,

$$S_t = i \quad \text{if} \quad i - 1 \leq N_a \mu_q(t) < i \tag{11}$$

for $i = 1, ..., N_s$.

The action space is given by (3), i.e. $A_t = i$ implies $N_{DL}(t) = (i-1)\Delta_a$. The reward function r_t is designed to maintain a balance between the aggregated queues while also penalising packet loss:

$$r_t = -|\mu_q(t) - 0.5| - L(t), \tag{12}$$

where $L(t)$ is a counter of loss queueing packets during the past frame $t-1$. Packet loss occurs either when the queue buffer is full or when a packet at the queue's output has been waiting longer than the maximum permitted queueing delay, previously defined as δ_{max}.

During training, the $Q(s, a)$ function is learnt over N_e episodes, with each episode consisting of N_f steps, with each step corresponding to one wireless frame. In this work, the RL agent follows the standard $\epsilon-$greedy policy, where a random action is selected with probability ϵ and the action with the highest estimated reward is chosen with probability $1 - \epsilon$, balancing exploration and exploitation. ϵ can either be fixed throughout the entire training period or be time-decaying, starting with a high degree of exploration and gradually shifting towards more exploitation.

b) DQN, with use of averaged transmission rates: Deep Q-Network (DQN) emerges as an extension of QL [15], in which a neural network, $Q(s, a; \theta)$, plays the role of Q-table, or in other words, it provides the Q-value function for any state-action pair and solves the problem of Q-table dimensions. During training, after frame t, the neural network parameters θ_t are updated based on a training batch of size L, resulting from a random sampling over a replay buffer that, at time-step or frame t, has a maximum of H experiences $(S_i, A_i, R_{i+1}, S_{i+1})$, for $i = t-H, ..., t$, to break the temporal correlation of training samples. Beyond the experience buffer, a secondary neural network, named target NN, is also introduced to generate a less fluctuating reference signal, which is updated through the primary neural network. The target NN, with same structure as the trained one and parameters θ', is used during training, to evaluate the Q-value function and minimize the Mean Square Error $\mathcal{L}_{MSE}(\theta_t)$ given in:

$$\mathcal{L}_{MSE}(\theta_t) = \frac{1}{L} \sum_{i \in \mathcal{H}_t} (y_i - Q(S_i, A_i; \theta_t))^2 \tag{13}$$

where $\mathcal{H}_t$ comprises the L indices of the batch experiences randomly drawn from the replay buffer at frame t, and $\forall i \in \mathcal{H}$:

$$y_i = R_{i+1} + \gamma \cdot \arg\max_{a'} Q(S_{i+1}, a'; \theta'_t). \tag{14}$$

Periodically, or using a smoothing-over-time strategy, the parameters of the training network, θ, are transferred to the target network into θ'. The DQN training algorithm pseudocode is shown in the annex (section VI). As in QL, DQN follows an $\epsilon-$greedy policy during episodic training. At the beginning of frame t, state S_t is defined as the continuous vector:

$$S_t = \begin{bmatrix} \mu_q(t) \\ \mu_r(t) \end{bmatrix} \in \mathcal{R}^2 \tag{15}$$

Accordingly, the DQN architecture consists of an input layer (two entries as the state dimension), followed by two hidden layers with 64 and 32 neurons respectively, and an output layer with $N_a = 11$ neurons, each corresponding to a possible action. As in QL, action is given by (3) and reward by (12). In summary, employing neural networks instead of a Q-table permits introducing continuous states with more than one variable for the splitter to act.

IV. Results

A. Generated traffic

DL packets originated at the access point and UL packets at user devices. All packets are of a fixed size of 720 bits. Packets from different flows enter a FIFO queue, since they cannot be sent instantly. In the experiments, we consider a single aggregated queue for UL and another for DL to formulate the problem. Each aggregated queue contains packets from N_{flow} different flows. The variable N_{flow} is generated randomly in the range $[1, N_{MF}]$ and is different in DL and UL. The time interval, between two consecutive packets from the same flow (UL or DL) that enter the queue is modelled as Poisson of parameter λ. The channel status, distinct for each flow, determines the transmission rate using an appropriate Modulation Coding Scheme (MCS) that adapts to varying channel conditions. We have considered a WiFi physical layer based on the 802.11a/g standard (non-HT format) [16]. Each MCS mode delivers a different number of bits that can be transmitted per slot, as shown in Table I. In the simulations, the MCS for UL and DL are generated randomly according to the probabilities in Table I with a periodicity equal to the channel coherence time. These probabilities have been computed within the framework of the TIMING project [17], by generating many realizations of a typical WiFi fading channel and ensuring a block error rate below 10^{-4}.

TABLE I
MCS table for 802.11a/g.

MCS	MOD	COD	bits/slot	Rate (Mbps)	Probability
0	BPSK	1/2	24	6	0.000104
1	BPSK	3/4	36	9	0.000270
2	QPSK	1/2	48	12	0.006419
3	QPSK	3/4	72	18	0.000382
4	16-QAM	1/2	96	24	0.156602
5	16-QAM	3/4	144	36	0.087424
6	64-QAM	2/3	192	48	0.179512
7	64-QAM	3/4	216	54	0.569287

Table II contains the most relevant parameters used in the experiments for the wireless scenario and for the algorithms.

B. Training of RL algorithms

The learning phase for both QL and DQN solutions relies on interaction characterized by random initial exploration. Following hyperparameter optimization stages, training was conducted over $N_e = 800$ episodes, with each episode comprising $N_f = 900$ frames or steps. In each step, the agent takes an action that consists of planning a frame according to the information provided by the current state and receives feedback from the environment. The QL algorithm has been trained considering $N_s = N_a = 11$, with discount factor $\gamma = 0.9$, learning parameter $\alpha = 0.1$ and following an $\epsilon-$greedy policy with ϵ decaying from 1 to 0.1 (decaying factor was 0.95).

TABLE II
Simulation Parameters

Parameter	Notation	Values
Number of Episodes [Train Val]	N_e	[800 100]
Frames per Episode [Train Val]	N_f	[900 100]
Frame Duration	T_f	10 ms
Slots per Frame	N_s	2500
Number of Actions	N_a	11
Poisson Traffic Parameter	λ	5 ms
DL Queue Size in Packets	N_b	1000
Packet Size	N_p	720 bits
Max Number of Flows [UL DL]	N_{MF}	[150 150]
Maximum Waiting Time	δ_{max}	10, 1 Frames
Learning Rate [QL DQN]	α	[0.1 1e-3]
Replay Buffer Size (DQN)	H	10000
Batch Size (DQN)	L	32
Exploration Parameter	ϵ	Decaying
Discount Factor	γ	0.9

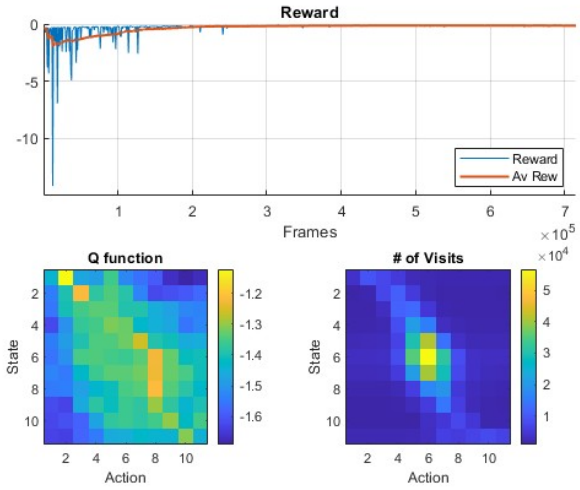

Fig. 3. QL Training; Per frame reward and averaged over a 50-frame window reward (12) (Top). Learnt QL table in terms of state-action (Left-Bottom). Number of visits per state-action pair (Right-Bottom)

As shown in Fig. 3, since the reward in (12) takes into account packets losses and these occur at the beginning of training, we initially get low reward values. Once the Q-function is learnt, the reward stabilizes at high values with no packet losses. Regarding the learnt Q-function, it is observed that in states corresponding to more loaded UL queues compared to DL queues, higher actions are chosen, which correspond to a greater allocation of time slots to the UL. In Fig. 3 the number of visits per state-action pair is also provided to validate the conditions under which the Q-table

has been computed.

The DQN algorithm was trained using parameters γ, ϵ as with QL. Furthermore, a smoothing strategy was used to transferring parameters from the training network to the target one. Mini-batch size was 32, weights were initialized using the Glorot procedure [18], and updated with the Adam optimizer [19], with learning parameter of 0.001. In Fig. 4 it is shown

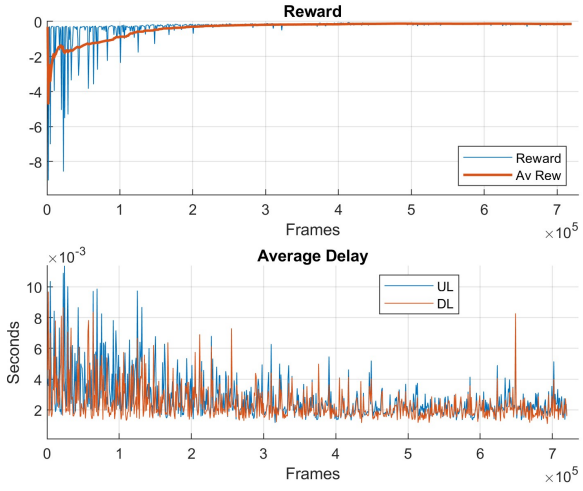

Fig. 4. DQN Training; Per frame reward and averaged over a 50-frame window reward (12) (Top). UL and DL averaged queueing delay (Bottom)

that the training has been successful: the reward exhibits an increasing trend over the epochs, denoting that there are no lost packets after 40000 frames (500 episodes) while the average queueing delay, i.e., the mean waiting time in the queue for all packets that were successfully transmitted, decreases.

C. Performance analysis

After training the RL-based methods, in this section we validate them alongside the heuristic techniques to compare their performance. The five algorithms presented in section III were executed over 100 episodes of 100 frames each one. To this end, we have utilized various metrics to evaluate the model's performance thoroughly. These metrics include:

- Average reward
- Average, Variance, 95% confidence interval and 90% percentile of the queueing delay measured in ms.
- Average lost packet rate, calculated as the number of lost packets divided by the total number of packets generated over the duration of the runs.

In validation, RL-based algorithms choose the action following exclusively a greedy policy, always selecting the one that provides a higher Q-value, and Q-functions are frozen, that is, the Q-table in QL or network parameters in DQN are no longer updated. Given that the remaining algorithms are already predefined, they follow the corresponding pre-established strategy as presented in section III.

The environmental conditions during validation are similar to those during the training of the RL-based algorithms, as detailed in table II. According to results shown in table III for $\delta_{max} = 10$ frames, the DQN algorithm performs the best overall achieving a higher average reward and exhibiting superior delay properties. The confidence interval is narrow enough in all cases to conclude that the average delay are precise and reliable. DQN shows significantly better variability and percentile than the rest of algorithms.

TABLE III
ALGORITHMS PERFORMANCE; $\delta_{max} = 10$ FRAMES

Measures	*SS*	*QSPS*	*WTPS*	*QL*	*DQN*
Avg. reward	-0.1472	-0.2171	-0.2048	-0.1487	**-0.1042**
Avg. delay (ms)	2.6841	2.7186	2.7102	2.5778	**2.2411**
$CI_{95\%}$ Inf	2.6806	2.7155	2.7071	2.5746	2.2386
$CI_{95\%}$ Sup	2.6875	2.7219	2.7135	2.5811	2.2436
Var of delay	8.9061	7.772	7.8494	8.1114	**4.8347**
90% percentile	5.1271	6.6671	6.6057	5.7855	**4.8823**
Avg. lost packets	3.42e-07	**0**	**0**	**0**	**0**

Table IV shows results of an scenario where a more restrictive maximum delay has been imposed ($\delta_{max} = 1$ frame). In it, SS achieves the highest average reward, but this is outweighed by its packet loss issue. On the other hand, when stringent latency requirements are imposed, WTPS also experiences occasional packet losses. Among the remaining algorithms, DQN exhibits superior properties by avoiding packet loss and achieving lower latency values alongside significant rewards.

TABLE IV
ALGORITHMS PERFORMANCE; $\delta_{max} = 1$ FRAME

Measures	*SS*	*QSPS*	*WTPS*	*QL*	*DQN*
Avg. reward	**-0.1671**	-0.2101	-0.2468	-0.2155	-0.2021
Avg. delay (ms)	2.8488	2.9044	3.0513	2.8985	**2.8263**
$CI_{95\%}$ Inf	2.8456	2.9012	3.0476	2.8952	2.8231
$CI_{95\%}$ Sup	2.8521	2.9077	3.0551	2.9019	2.8296
Var of delay	8.7687	8.7758	11.6035	9.2331	**8.7484**
90% percentile	5.9486	6.9311	7.1202	6.9632	6.6282
Avg. lost packets	6.28e-06	**0**	1.63e-04	**0**	**0**

Fig. 5 illustrates the reward obtained in each frame alongside the average reward during validation. The DQN algorithm achieves the highest reward, followed by QL and SS. However, SS is not competitive due to the occurrence of packet loss.

V. CONCLUSIONS

Latency remains a major challenge for TSN networks, specially in the wireless sectors. Throughout this study, various strategies have been explored for slot allocation between DL and UL, aiming to optimize available resources and reduce queueing latency.

Based on the results obtained, we can conclude that DQN consistently demonstrated superior performance in minimizing delay and packet loss, due to its lookahead capabilities, and that the use of channel information based on the transmission rate has improved performance. Furthermore, it has been generally observed that RL-based algorithms outperform

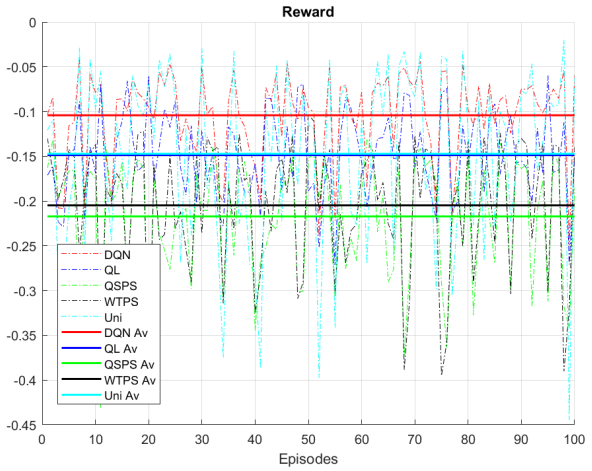

Fig. 5. Per-frame and averaged reward over a 50-frame window reward (12) in validation. $\delta_{max} = 10$ frames.

heuristic algorithms, suggesting that the optimal dynamic slot allocation strategy can be learnt through interaction with the environment. Heuristic methods have also become competitive for higher maximum queueing delay, leveraging knowledge of transmission speeds. However, in more stressed scenarios, the best results are obtained by applying DQN-based algorithms. Further work includes to test the algorithms in other wireless environments such as 5G and 6G. Given the scalability of the model, both in terms of state definition and actions, we expect the results to be similar to those presented here.

VI. Annex: DQN Algorithm

Algorithm 1 DQN pseudocode

Initialize the critic network θ
Copy from critic to target the parameter values, $\theta' \leftarrow \theta$
for each episode **do**
  Initialize state s_0
  for each t of the episode **do**
    Select a_t from $Q(s_t, a, \theta_t)$; e.g. $\epsilon - greedy$ policy
    Take action a_t, and observe r_{t+1} and s_{t+1}
    Store $(s_t, a_t, r_{t+1}, s_{t+1})$ in the experience buffer
    Sample mini-batch of L exp. $(s_i, a_i, r_{i+1}, s_{i+1})$
    Compute MSE $\mathcal{L}_{MSE}(\theta_t)$ as in (13)
    Update θ by single-step minimization of the MSE
    Update target parameters θ' if needed
  end for
end for

References

[1] D. Cavalcanti, C. Cordeiro, M. Smith, and A. Regev, "WiFi TSN: Enabling deterministic wireless connectivity over 802.11," *IEEE Communications Standards Magazine*, vol. 6, no. 4, pp. 22–29, 2022.

[2] G. A. for Connected Industries and Automation, "Integration of 5G with time-sensitive networking for industrial communications," 2021, accessed on May 11, 2024. [Online]. Available: https://5g-acia.org/whitepapers/integration-of-5g-with-time-sensitive-networking-for-industrial-communications/

[3] L. Velasco, G. Graziadei, Y. El-Kaisi-Rahmoun, J. Villares, O. Muñoz-Medina, J. Vidal, and M. Ruiz, "Provisioning of time-sensitive and non-time-sensitive flows: from control to data plane," in *4th International Workshop on Time-Sensitive and Deterministic Networking (TENSOR)*, 2024.

[4] R. Jurdi, J. Guo, K. J. Kim, P. Orlik, and Y. Nagai, "Queueing delay analysis of mixed traffic in time sensitive networks," in *2021 6th International Conference on Control, Robotics and Cybernetics (CRC)*, 2021, pp. 327–332.

[5] V. Gavriluţ and P. Pop, "Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[6] Y. Qian, J. Wu, R. Wang, F. Zhu, and W. Zhang, "Survey on reinforcement learning applications in communication networks," *Journal of Communications and Information Networks*, vol. 4, no. 2, pp. 30–39, 2019.

[7] M. Abu Alsheikh, D. T. Hoang, D. Niyato, H.-P. Tan, and S. Lin, "Markov decision processes with applications in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1239–1267, 2015.

[8] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," pp. 74 429–74 441, 2018.

[9] Z. Liu, X. Chen, Y. Chen, and Z. Li, "Deep reinforcement learning based dynamic resource allocation in 5G ultra-dense networks," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, 2019, pp. 168–174.

[10] M. A. Jadoon, A. Pastore, M. Navarro, and A. Valcarce, "Learning random access schemes for massive machine-type communication with marl," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 95–109, 2024.

[11] E. Mete and T. Girici, "Q-learning based scheduling with successive interference cancellation," *IEEE Access*, vol. 8, pp. 172 034–172 042, 2020.

[12] K. Boutiba, M. Bagaa, and A. Ksentini, "On using deep reinforcement learning to reduce uplink latency for urllc services," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 407–412.

[13] H. Yang, Z. Xiong, J. Zhao, D. Niyato, C. Yuen, and R. Deng, "Deep reinforcement learning based massive access management for ultra-reliable low-latency communications," *IEEE Transactions on Wireless Communications*, vol. 20, no. 5, pp. 2977–2990, 2021.

[14] A. A. Esswie, K. I. Pedersen, and P. E. Mogensen, "Online radio pattern optimization based on dual reinforcement-learning approach for 5G URLLC networks," *IEEE Access*, vol. 8, pp. 132 922–132 936, 2020.

[15] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.06461

[16] *IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, High-speed Physical Layer in the 5 GHZ Band*, Std., 1999.

[17] "TIMING Project," https://timing.upc.edu/home.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980