On using Deep Reinforcement Learning to reduce Uplink Latency for uRLLC services

Karim Boutiba
EURECOM
Sophia Antipolis, France
karim.boutiba@eurecom.fr

Miloud Bagaa
CSC-IT Center for Science Ltd.
Espoo, Finland
miloud.bagaa@csc.fi

Adlen Ksentini
EURECOM
Sophia Antipolis, France
adlen.ksentini@eurecom.fr

***Abstract*—5G networks and beyond are shifting from dominant Downlink (DL) traffic to a more equilibrate DL/UpLink (UL) and dominant UL traffic for specific emerging services. Particularly for ultra-Reliable and Low Latency Communications (uRLLC) services, the UL latency becomes an essential factor to consider. However, current UL scheduling methods are not efficient in terms of Physical Resource Blocks (PRBs) allocation, latency, or link adaptation. In this paper, we address the emerging challenge related to the UL latency in 5G networks and beyond. We introduce a solution based on Deep Reinforcement Learning (DRL) to dynamically allocate the future UL grant by learning from the dynamic traffic pattern. Simulation results demonstrate the efficiency of the proposed methodology in reducing the UL latency down to 0.25 ms and ensuring the generality by reacting to different traffic models.**

I. Introduction

Ultra-Reliable and Low Latency Communications (uRLLC) is one of the most critical services in 5G networks and beyond, requiring as stringent requirements as $1-10^{-5}$ reliability with a user plane latency of 1 ms [1]. Emerging uRLLC use cases are shifting from Downlink (DL)-dominant traffic to more equilibrate DL and Uplink (UL) traffic. Hence, it is crucial to ensure a low latency in UL and DL directions. However, achieving low latency in UL is more challenging since the next Generation Node-B (gNB) has to be aware of the User Equipment (UE)'s queue status to allocate PRBs according to the UE needs, which results in increasing the latency by the signaling overhead. Semi-Persistent Scheduling (SPS) [2] and Grant Free (GF) [3] scheduling methods are proposed to overcome the signaling overhead, hence reducing the UL latency. However, they are less efficient for dynamic traffic patterns (dynamic inter-arrivals and dynamic data sizes) and less adaptable to quick channel condition changes, especially in FR2 frequencies ($>$ 6Ghz) and high mobility users. Furthermore, GF may impact the reliability of the services since resource usage can collide in frequency between users.

To overcome the aforementioned challenges, we introduce a novel DRL-based algorithm that reduces the UL latency by predicting the subsequent arrival of data and its size. We dubbed this solution Deep Reinforcement Learning (DRL)-based Low Latency Scheduler (DRL-LLS). The latter monitors the Buffer Status Report (BSR) sent by UEs and derives the future inter-arrival time and the grant size that should be allocated to the user in the next UL opportunity. DRL-LLS is adapted to change happening in channel conditions very quickly since it dynamically sends grants to the user before the data arrival. Hence, when the data is ready to be sent, the UE will find the grant and fully transmit all the data to gNB, which decreases the overhead signaling latency while adapting to the channel conditions.

The main contributions of this work are manifolds:

- We introduce a Deep Reinforcement Learning (DRL) based framework (DRL-LLS) to reduce the UL latency of uRLLC services.
- We introduce a DRL-based agent to predict the future UL slot for a UE by leveraging its traffic inter-packets arrival history.
- We introduce a DRL-based agent to compute the grant size for the next UL slot according to the BSR history.
- We combine the above models to enhance the UL MAC scheduler. The new methodology dramatically reduces the signaling overhead. For instance, the UL latency will decrease since the UE will not send a Scheduling Request (SR) nor send BSR to get more grants for the same transmission. More details are provided in Section II-A.
- We validate DRL-LLS on different traffic models [4] used in 5G new use cases like eXtended Reality (XR) and near real-time video streaming.

The rest of the paper is organized as follows: Section II describes the UL scheduling problem in 5G New Radio (NR), the related works to reduce the UL latency, and introduces DRL algorithms. Our proposed solution is presented in Section III and evaluated in Section IV. Finally, we conclude the paper in Section V.

II. Background

A. Problem description

The UL Dynamic Scheduling (DS) method is widely used among network operators due to its efficiency to allocate PRBs and high adaptation capability to channel conditions. However, it is inefficient for uRLLC services due to its negative impact on the latency. Indeed, the PRBs are allocated to a UE upon the gNB receiving an SR on the Physical Uplink Control Channel (PUCCH) channel. gNB then assigns PRBs to the UE and sends back a Downlink Control Information (DCI),

978-1-6654-3540-6/22 © 2022 IEEE

informing mainly the Modulation and Coding Scheme (MCS), the number of PRBs, and the k2 parameter to be used for that transmission. The UE will send the data after k2 slots from receiving the DCI. The first DCI contains a minimum grant in terms of the number of PRBs (since SR is just one bit and does not inform the gNB about the current queue status of the UE). If the UE needs to send more data (i.e., the minimum grant is not enough), it piggy-backs a Buffer Status Report (BSR) with the data to inform the gNB about the UL UE's queues status. Then, the gNB will send a DCI with an additional number of PRBs. Figure 1 illustrates the 3gpp compliant dynamic UL scheduling procedure. Since SR is sent each time before UE sends data, UL's dynamic schedule can adapt to changes in channel conditions very quickly. However, this method leads to a high latency which consists of the SR latency (delay between generating the SR and receiving the Physical Uplink Shared Channel (PUSCH) data) and the BSR delay (delay between sending the BSR and receiving the last PUSCH of the same transmission).

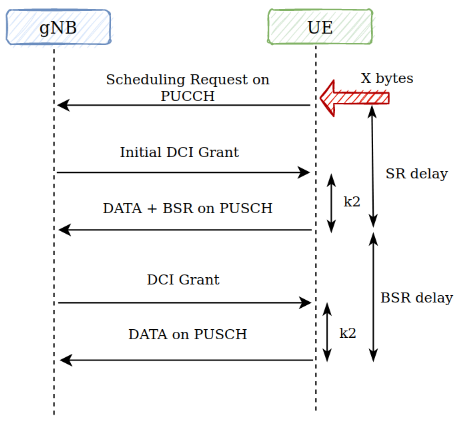

Figure 1: 3GPP dynamic UL scheduling procedure

Basically, the standard dynamic UL scheduling is unsuitable for uRLLC services requiring a latency of less than 1 ms. SPS and GF scheduling methods are proposed to reduce the latency. The gNB allocates PRBs to the UE without waiting for an SR. The PRBs are reserved for the UE periodically. However, these methods give a fixed grant size regarding PRBs and periodic slots, which is unsuitable for dynamic data patterns. Besides, they respond to changing channel conditions much slower than dynamic scheduling since the MCS is reported only once at the activation step. Hence, it reduces the reliability of transmissions, especially in very dynamic channel conditions like FR2 and high mobility users, which is unsuitable for uRLLC services.

This paper aims to overcome the shortcomings mentioned above by providing a high dynamic scheduler that has the ability to avoid the SR and BSR latency and adapt according to channel conditions. To support uRLLC, gNB may anticipate UL traffic and then dynamically send DCI grants (to avoid the SR latency) with the adapted MCS (to adapt to channel conditions) and with the right PRBs grant (to avoid the BSR latency). To achive this, we propose to predict the future traffic arrival time ΔT_{i+1} and the future traffic size X_{i+1}. Once the gNB knows ΔT_{i+1}, it will send a DCI grant, with f(X_{i+1}, MCS) PRBs grant, k2 slots before $t + \Delta T_i$, where t is last arrival time slot and $f(x, y)$ is the 3gpp compliant function [5] that returns the number of PRBs needed to transport x bytes under MCS y. The k2 parameter is computed by $k2 = t + \Delta T_i - t_c$, where t_c is the time of sending the DCI. Figure 2 illustrates the dynamic UL scheduling featuring low latency.

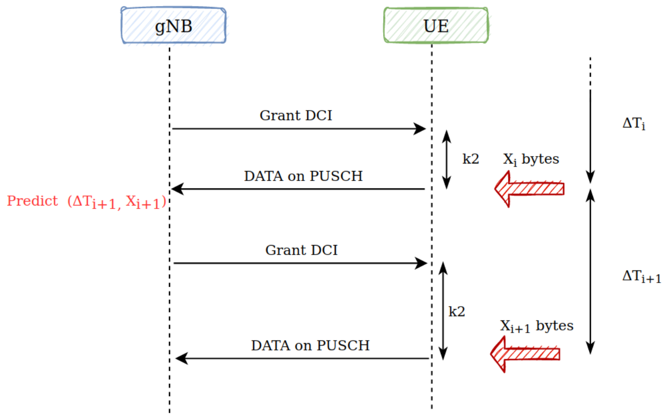

Figure 2: 3GPP dynamic UL scheduling procedure featuring low latency

B. Related work

Authors of [6] have used deep learning to predict future traffic. They show that deep learning outperforms linear statistical learning when the number of past observations is significant enough to learn the traffic pattern. Meanwhile, authors in [7] have studied the impact of NR scheduling timings (mainly the k2 parameter) on the UL traffic latency. They have proposed a heuristic approach to derive k2 value from the inter-packet arrival time considering only periodic traffic with fixed packet size. Authors in [8] have proposed an enhanced dynamic scheduling method to reduce VoIP traffic UL latency. However, they support only periodic traffic with fixed packet inter-arrival times. They have reduced the latency by removing the SR delay. However, the BSR delay was not reduced when the packet size is significant and can not be handled with the minimum grant allocated to the UE in the first grant. Authors in [9] have presented a novel framework for traffic prediction of Internet of Things (IoT) devices activated by binary Markovian events modeled by an On-Off Markov process with known transition probabilities. However, this work considered only ON-OFF traffic distribution and did not consider packet size. Authors of [10] leveraged Artificial Neural Network (ANN) to predict the ON-OFF period of burst traffic and estimate the bandwidth to be allocated to reduce the latency. However, they only considered IoT traffic following a burst pattern and over-estimated bandwidths compared to the real traffic. In fact, the bandwidth is estimated using the data rate of the network. Authors in [2] have suggested a predictive SPS scheme to predict data size during the SPS period. However, they considered only haptic data patterns, and the inter-packet arrival time is deemed static during the SPS period. Besides, the SR latency is not reduced. Authors of [11] developed a Long-Short Term Memory (LSTM) architecture to predict future traffic and minimize radio latency. However, they did not consider the size of traffic in their prediction.

All the above solutions considered a specific traffic pattern (either periodic, following Exponential distribution, or ON-OFF distribution), and most of them did not consider joint arrival prediction with data size prediction.

C. Deep Reinforcement Learning (DRL)

Machine Learning (ML) is playing an important role in 5G Networks and Beyond. Particularly DRL, a ML technique that can be used without the need for data sets. DRL can be leveraged to derive configuration or management decisions in real-time [12] (i.e., less than $1ms$) in a stochastic environment, which makes it suitable for the Radio Access Networks (RAN) domain. DRL can provide self-configured, and self-optimized network functions, such as radio resource allocation [13]. A DRL framework has two actors: An agent and an environment. The agent observes a state S_t from the environment, applies an action a_t, gets a reward r_{t+1}, and hence the environment moves to the next state S_{t+1}. In the case of S_{t+1} is not impacted by a_t, the RL problem becomes Contextual Bandit problem. For instance, DRL can be used to solve the problem by considering a continuous problem with a discount factor of zero. The agent can be in two modes: $i)$ exploration mode, where the agent explores and builds the knowledge about the environment, and $ii)$ exploitation mode, where the agent exploits the acquired knowledge by following the optimal policy π_* that gives for each state S_t the optimal action a_t^*. Accordingly, the ability of DRL to derive good decisions quickly, deal with unseen environments, and be scalable make it suitable for solving the UL scheduling problem in 5G networks.

III. Deep Reinforcement Learning Low Latency Scheduler (DRL-LLS)

In this section, we present our solution dubbed Deep Reinforcement Learning Low Latency Scheduler (DRL-LLS). In the balance of this section, we first present the DRL-LLS design and then give a detailed description of DRL-LLS.

A. DRL-LLS design

DRL-LLS aims to reduce the UL latency for a set of UEs, whereby each UE can serve more than one service. DRL-LLS predicts the next UL slot for each UE u given its traffic history and schedules it before the actual data arrival. The environment is considered as a sliding window $W_{u,j}^i$ of size T containing information about the inter-arrival time intervals and BSR information history for each couple (UE u, service j) at the i^{th} data arrival. Formally:

$W_{u,j}^i = \{I_{u,j}^i, D_{u,j}^i\}$;

$I_{u,j}^i = (\Delta T_{i-T}, \Delta T_{i-(T-1)}, ..., \Delta T_i)$;

$D_{u,j}^i = (X_{i-T}, X_{i-(T-1)}, ..., X_i)$

where ΔT_i and X_i are the i^{th} inter-arrival time interval (in slot granularity) and BSR information (in bytes) of UE u, service j, respectively. We differentiate the BSR belonging to service j by the Logical Channel ID (LCID) available at the MAC header, considering that each service has a separate Logical Channel (LC) in the context of network slicing at the Radio Access Network (RAN) [13]. For instance, by giving $W_{u,j}^i$ as input, DRL-LLS schedules an UL slot for UE u following service j traffic pattern. That gives the generalization ability to our agent and makes our DRL-LLS independent from the number of UEs and the number of services per UE.

We recall that the UL latency at the RAN is composed of two parts: the SR and the BSR latency (Figure 1). In order to reduce the SR latency, DRL-LLS predicts the next arrival of UE's u data. Whereas, to reduce the BSR latency, DRL-LLS predicts the size of the future data arrival of UE u. To achieve the two goals, DRL-LLS includes two agents: $i)$ Inter-Arrival Time (IAT) Agent; $ii)$ Data Grant (DG) Agent. Since traffic size and traffic inter-arrivals are independent [4], the two agents can be independent if their states and rewards are well designed. Figure 3 illustrates the architecture of DRL-LLS.

IAT Agent takes $(\Delta T_{i-T}, \Delta T_{i-(T-1)}, ..., \Delta T_i)$ as input and generates the next arrival interval ΔT_{i+1}.

DG Agent takes $(X_{i-T}, X_{i-(T-1)}, ..., X_i)$ as input, and then generates the next arrival size X_{i+1}.

DRL-LLS combines the values of ΔT_{i+1} and X_{i+1}, and to schedule the future UL slot after $t + \Delta T_{i+1}$ slot with f(X_{i+1}, MCS) PRBs grant, where t is the current slot (last data arrival) and $f(x, y)$ is the 3gpp compliant function [5] that returns the number of PRBs needed to transport x bytes under MCS y.

1) IAT Agent:

- **State:** The observation of IAT agent at time t as normalized value is $\frac{I_{u,j}^t}{M}$ for UE u, service j. M is the maximum time interval in the system in slots.
- **Actions:** The action is ΔT_{t+1}, which takes values from 1 to M.
- **Reward:** $r_1 = \frac{L_{max} - l}{L_{max}}$, where L_{max} is the maximum latency and l is the observed latency. In order to make the IAT agent independent from the DG agent, we ignored the data size in l computation. Formally;

$$l = \begin{cases} p - c + 1 + S_{max} * \Delta f & \text{if } p > c \\ (S_{max} - c + 1) - p + S_{max} * \Delta f, & \text{otherwise} \end{cases}$$

  Where p is the predicted slot in the Time Division Duplex (TDD) pattern, $p = (t + \Delta T_{i+1} \mod S_{max})$, c is the current slot, S_{max} is the number of slots in the frame and Δf is the difference between the current frame and the predicted frame $\Delta f = \mid \lceil \frac{(c+\Delta T_{i+1})}{S_{max}} \rceil - f' \mid$, f' is the frame offset of the actual data arrival.
  The reward is negative when the predicted slot is far from the actual data arrival. It increases when the latency approaches L_{max} and becomes positive when the latency is smaller than L_{max}.

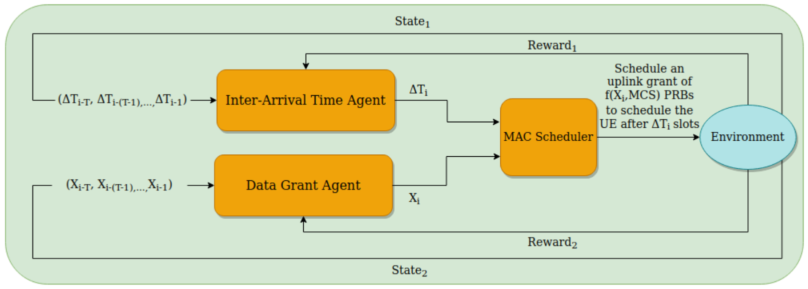
State$_1$
Reward$_1$
(ΔT_{i-T}, $\Delta T_{i-(T-1)}$,...,ΔT_{i-1})
Inter-Arrival Time Agent
ΔT_i
Schedule an uplink grant of f(X_i,MCS) PRBs to schedule the UE after ΔT_i slots
MAC Scheduler
Environment
(X_{i-T}, $X_{i-(T-1)}$,...,X_{i-1})
Data Grant Agent
X_i
Reward$_2$
State$_2$

Figure 3: DRL-LLS design

Reward and States are normalized since it is well-known that the activation functions in the neural network work well for small values, which positively impacts the model convergence.

2) DG Agent:

- **State:** The observation of IAT agent at time t as normalized value is $\frac{D^t_{u,j}}{N}$ for UE u, service j. N is the maximum data size in the system in bytes.
- **Actions:** The action is X_{t+1}, which takes values from 1 to N. The unit is kbytes instead of bytes to reduce the action space.
- **Reward:** $r_2 = -\alpha * \mid \lceil \frac{X_{t+1}-d_r}{N} \rceil \mid + (1-\alpha) * l'$, whereby, d_r is the real $(t+1)^{th}$ data arrival size. In order to make the DG agent independent from the IAT agent, we ignored the time slot prediction in l' computation, Formally,

$$l' = \begin{cases} 0 & \text{if } X_{t+1} < d_r \\ 1 & \text{otherwise} \end{cases}$$

The reward increases when the predicted size is slightly bigger than the actual size. It decreases when the predicted size is smaller than the actual size or the predicted size is much bigger than the actual size (over allocated PRBs). α controls the contribution of each term of the reward. For instance, it influences how much we tolerate the gap between actual and predicted sizes.

B. DRL-LLS detailed description

For both agents, DRL-LLS leverages the Deep Q-Network (DQN) algorithm [14], which is one of the most efficient DRL algorithms for continuous state space and discrete actions. DRL-LLS executes two steps: decision making and updating the Q-Networks. In DQN, two networks are used: a local Q-Network and a target Q-Network. The latter is the same as the local network except that its parameters are updated every τ^{-1} step. They are combined to help the convergence and stabilization of the learning.

1) Decision making: IAT agent observes a state $I^t_{u,j} \div M$ and feeds it to the local QNetwork to get the discrete action distribution of ΔT_{t+1}. While DG agent observes a state $D^t_{u,j} \div N$ and feeds it to the local QNetwork to get the discrete action distribution of X_{t+1}. Then, we apply an ϵ-greedy approach to choose an action from each distribution, which means IAT and DG agents will choose a random action over the possible actions with ϵ probability and the best action over the action distribution with a 1-ϵ probability. ϵ will decrease over time during the learning pushing the agent to explore the environment at the beginning of the training and driving it to exploitation over time.

2) Updating the Q-Networks: At each step, the current state, the action, the next state, and the reward are stored in a buffer known as the replay buffer. The local Q-Network is updated using a random sample from the replay buffer, which reduces the correlation between the agent's experiences and increases the stability of the learning. Using mean square error (MSE) and ADAM optimizer [15], the parameters of the local Q-Network are optimized at every step by considering the local and target values. In contrast, the parameters of the target Q-Network are updated every τ^{-1} step to stabilize the algorithm's convergence.

IV. Performance Evaluation

In this section, we will introduce the simulation environment and parameters used for training DRL-LLS agents. Then, we will evaluate the trained agents in a 5G simulated environment using different 5G traffic models from [4].

A. Simulation parameters and training phase

We have trained both IAT and DG agents using 5000 independent episodes. For each episode, a different traffic inter -arrival and data size patterns are used in a circular way from the list: Exponential distribution with a random λ; $0 < \lambda < \frac{1}{M}$ for IAT agent and $0 < \lambda < \frac{1}{N}$ for DG agent, Static distribution with a random value m ($0 < m < M$ for IAT agent and $0 < m < N$ for DG agent), On-Off distribution (For IAT agent only, λ, β_1 and β_2 are the parameters of the exponential distributions for traffic inter-arrival during the ON period, the intervals of ON period and the intervals of OFF period, respectively) and the truncated Pareto distribution (random α, $1 < \alpha < 3$, $0 < m < M$, $m < max < M$ for IAT agent and $0 < m < N$, $m < max < N$ for DG agent). The value of M was fixed by 100 slots and the value of N by 10^3 kbytes. We have fixed the maximum number of steps at each episode by 200 and the window size T by 1000. Before starting the learning process, the window is filled by arrivals generated by the current episode distribution. The different considered parameters for both agents are presented in Table II

Table I: Traffic patterns parameters

Traffic model index	packet inter-arrival	packet size
0	Exponential distribution ($\lambda = 1/20$)	Exponential distribution ($\lambda = 1/10k$)
1	Exponential distribution ($\lambda = 1/5$)	Static (20k)
2	Static (10)	Static (10k bits)
3	On-Off distribution ($\lambda = 1/10$, $\beta_1 = 1/20$, $\beta_2 = 1/20$)	Static (15k bits)
4	On-Off distribution ($\lambda = 1/10$, $\beta_1 = 1/20$, $\beta_2 = 1/10$)	Exponential distribution ($\lambda = 1/20k$)
5	Truncated Pareto distribution ($\alpha = 1.1$, $m = 5$, $max = 30$)	Truncated Pareto distribution ($\alpha = 1.2$, $m = 10k$ bits, $max = 30k$ bits)
6	Truncated Pareto distribution ($\alpha = 1.2$, $m = 10$, $max = 40$)	Static (50k bits)

To evaluate DRL-LLS in a 5G environment, we extended the 5G Simulator developed in [16] to support UL scheduling, and we fixed the numerology by 2 (Sub-Carrier Spacing (SCS) = 60 khz) which is available in both FR1 and FR2. We used a TDD period of 1.25 ms, which is the smallest TDD period for numerology 2. We used a random MCS for each episode in the range of 20..26, which maps to medium or good channel condition. Since DRL-LLS is independent of the number of UEs and services, we focused on training and testing the framework on a single UE with different traffic patterns to show the ability of DRL-LLS to reduce the UL latency. We have implemented our simulation environment using Python and Pytorch library. We have used a machine with 32 CPUs, an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz (2.7 GHz with Turbo Boost technology), and 128 GBs of RAM.

Table II: DRL-LLS parameters

Parameter	Value
α	0.2
L_{max}	4 slots (1 ms)
Number of hidden layers	3
Hidden layer size	64 nodes
discount factor γ	0
Batch size	128
Learning rate	$5 * 10^{-4}$
Replay buffer size	10^9
Soft update coefficient τ	0.001
Optimizer	ADAM [15]
ϵ-start	1
ϵ-decay	0.9991
ϵ-end	0.01
Number of training episodes	5000

Figure 4 depicts the convergence evaluation of DRL-LLS score (sum of rewards during an episode) averaged every 100 episodes. We observe that the DRL-LLS agents converge after 4800 episodes since the curve tangents tend toward 0.

B. Inference phase

We have evaluated the DRL-LLS framework in terms of: i) The average UL latency between data chunk arrival and the transmission of the whole chunk; ii) The number of PRBs allocated during the test; iii) The average inference time of both IAT and DG agents executed sequentially (one after the other).

In Figure 5, The y-axis represents the UL latency (in ms), which is measured by $t_t^c - t_a^c$, where t_a^c is the arrival time of a data chunk c and t_a^c the transmission time of the last part of the data chunk c. The x-axis represents the traffic pattern index introduced in Table I. Each pattern has a packet inter-arrival time and data size distribution. We used different distributions with different parameters to show that DRL-LLS is able to predict the pattern of different distributions with different parameters. We recall that these distributions correspond to 5G use case patterns [4]. We compared DRL-LLS with two methods: a) DS, where the UE sends an SR to gNB when it has data to transmit; b) always Dynamic Scheduling (aDS), where gNB schedules a minimal grant in all the UL slots to avoid the SR procedure. The minimum grant for both DS and aDS is 5 PRBs. We should note that the time unit in packet-inter arrival is slots. For instance, $\lambda = \frac{1}{20}$ for an Exponential distribution means that the mean of the inter-arrival intervals is 20 slots. We run the test over 1000 packet arrival for each pattern configuration. We observe that DRL-LLS offers a lower latency compared to aDS and DS. Indeed, DS latency includes both SR and BSR latency, while aDS only consists of the BSR latency. However, DRL-LLS removes both SR and BSR latency since it predicts the subsequent arrival and the size of the next arrival. Hence, the UE sends the data directly without sending the SR and does not need a BSR to extend the transmission. The smallest achieved latency by DRL-LLS is 0.25 ms (1 slot in numerology 2) in the case of periodic traffic (traffic pattern index 2). This is due to the simplicity of the pattern, which is predictable ideally by DRL-LLS. The highest latency achieved by DRL-LLS is 1.1 ms, which makes

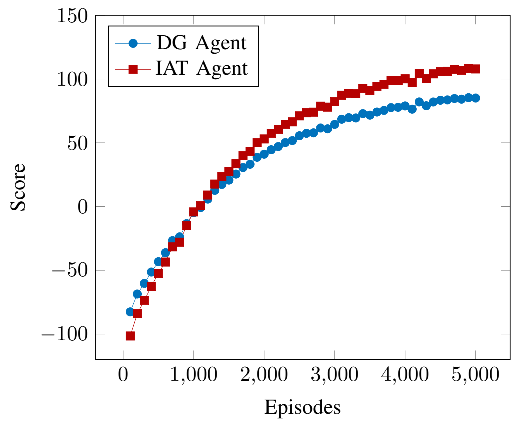

Figure 4: Convergence evaluation of DRL-LLS agents during the training mode

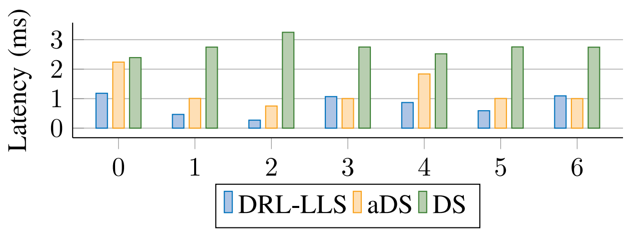

Figure 5: Average UL latency comparison between DRL-LLS, aDS and DS

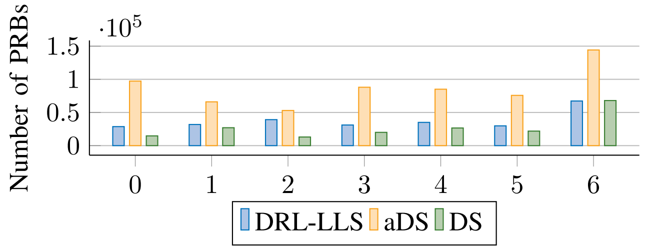

Figure 6: Number of the allocated PRBs comparison between DRL-LLS, aDS and DS

the framework suitable for uRLLC services with a latency requirement of 1 ms.

Figure 6 depicts the number of allocated PRBs (for UL) during the 1000 packet arrivals for each traffic pattern (Table I). We notice that aDS consume much more PRBs compared to DRL-LLS and DS. This is since aDS schedules 5 PRBs in every UL slot even if there is no transmission. We should shed light that DS is optimal in terms of resource allocation since it allocates only the needed amount of PRBs to transmit the data. We also observe a smaller difference between DRL-LLS and DS in terms of resource allocation while achieving a much better latency by DRL-LLS (Figure 5).

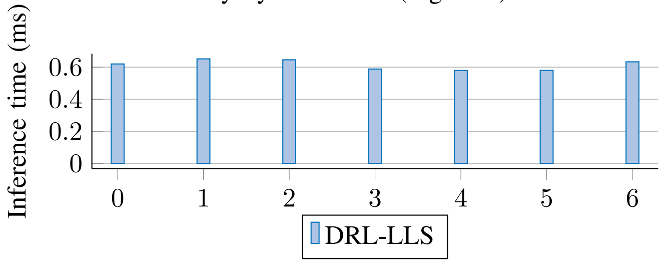

Figure 7: Average inference execution time of DRL-LLS

Finally, Figure 7 shows the average execution time of DRL-LLS inference over the 1000 data arrival for each traffic pattern (Table I). We notice that the execution time does not exceed 0.62 ms, which is suitable for real-time scheduling and hence, better adaptation with channel conditions and better latency.

Conclusion

This paper introduced DRL-LLS, a Deep Learning Reinforcement (DRL)-based solution that reduces the UL latency in 5G NR. DRL-LLS will be leveraged by the 5G base station before the UE scheduling process to derive i) the next UL slot to schedule and ii) the number of PRBs to allocate to the user. Without knowing the UEs traffic model, DRL-LLS can learn the traffic inter-arrival and data size patterns. DRL-LLS uses the available BSR information history to derive the following UL grant. Simulation results clearly showed that DRL-LLS is able to reduce the UL latency down to 0.25 ms. Our future focus is on implementing DRL-LLS on top of OpenAirInterface (OAI) [17] 5G to validate real uRLLC use cases.

Acknowledgment

This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program under 5G!Drones project (Grant No. 857031) and MonB5G project (Grant No. 871780).

References

[1] Adlen Ksentini et al. "Providing Low Latency Guarantees for Slicing-Ready 5G Systems via Two-Level MAC Scheduling". In: *IEEE Network* 32.6 (2018), pp. 116–123.

[2] Ye Feng, Ampalavanapillai Nirmalathas, and Elaine Wong. "A Predictive Semi-Persistent Scheduling Scheme for Low-Latency Applications in LTE and NR Networks". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019.

[3] Nurul Huda Mahmood and al. "Uplink Grant-Free Access Solutions for URLLC services in 5G New Radio". In: *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. 2019.

[4] Jorge Navarro-Ortiz et al. "A Survey on 5G Usage Scenarios and Traffic Models". In: *IEEE Communications Surveys Tutorials* (2020).

[5] 3GPP. "5G NR; Physical layer procedures for data". In: *TS 38.214* Release 15 (2018).

[6] Amin Azari et al. "User Traffic Prediction for Proactive Resource Management: Learning-Powered Approaches". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019.

[7] Natale Patriciello et al. "The Impact of NR Scheduling Timings on End-to-End Delay for Uplink Traffic". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019.

[8] Ahmet Gizik, Ozgun Alkin Sensoy, and Engin Masazade. "Enhanced Dynamic Scheduling for Uplink Latency Reduction in Broadband VoLTE Systems". In: *2021 55th Asilomar Conference on Signals, Systems, and Computers*. 2021.

[9] Mohammad Shehab and al. "Traffic Prediction Based Fast Uplink Grant for Massive IoT". In: *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. 2020.

[10] Lihua Ruan, Maluge Pubuduni Imali Dias, and Elaine Wong. "Machine Learning-Based Bandwidth Prediction for Low-Latency H2M Applications". In: *IEEE Internet of Things Journal* (2019).

[11] Eslam Eldeeb, Mohammad Shehab, and Hirley Alves. "A Learning-Based Fast Uplink Grant for Massive IoT via Support Vector Machines and Long Short-Term Memory". In: *IEEE Internet of Things Journal* (2022).

[12] N. C. Luong and al. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey". In: *IEEE Communications Surveys Tutorials* (2019).

[13] Karim Boutiba, Miloud Bagaa, and Adlen Ksentini. "Radio resource management in multi-numerology 5G new radio featuring network slicing". In: *ICC 2022*. Ed. by IEEE. Seoul, 2022.

[14] Volodymyr Mnih and al. "Playing Atari with Deep Reinforcement Learning". In: (2013).

[15] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017.

[16] Karim Boutiba et al. "NRflex: Enforcing network slicing in 5G New Radio". In: *Computer Communications* (2021).

[17] Florian Kaltenberger et al. "The OpenAirInterface 5G new radio implementation: Current status and roadmap". In: *WSA 2019, 23rd ITG Workshop on Smart Antennas, Demo Session, 24-26 April 2019, Vienna, Austria*. 2019.