Smart De-jittering using Network Digital Twin

Marc Mollà Roselló
Technology & Innovation
Ericsson
Madrid, Spain
marc.molla@ericsson.com

Manuel Lorenzo
Technology & Innovation
Ericsson
Madrid, Spain
manuel.lorenzo@ericsson.com

***Abstract*—The evolution of mobile networks is converging with the pervasive digitalization of the industrial/ manufacturing sector, which is demanding highly reliable and deterministic wireless networking. The performance levels of wireless networks are greatly enhanced from one generation to the next, but the variability of performance, especially with regards to latency, remains an issue to be addressed since most protocols and solutions are designed with the assumption that underlying networks shall deliver performance levels with very low variability, as it is the case, for example, of Ethernet networks. In this work, we address this critical issue, proposing and validating a novel solution approach for achieving effective de-jittering in 3GPP networks that does not require the modification of the end-user traffic (e.g. timestamping), thus enabling its implementation on current 5G networks without demanding special UEs or new 5GS capabilities. This solution approach is validated over a real 5G network deployed at 5Tonic laboratory.**

Keywords—5g, time-sensitive networking, Network Digital Twin, deep reinforcement learning, 5tonic lab, non-public networks.

I. Introduction

The evolution of the mobile networks provides not only improvements in the performance but also new capabilities that allows to adapt its architecture to specific use cases. For example, using Network Slicing and other capabilities, it is possible to create Non-Public Networks [1] for Industrial use cases, which enables the integration of private segments of mobile networks with public networks. This topic has been a trend in the research and innovation of mobile networks [2], being amplified nowadays by the nice convergence of several complementary technology trends and innovations, such as deterministic communications, Digital Twin and Edge computing, for enabling robust communication solutions for meeting more and more Industry 4.0 requirements.

Deterministic communications is an essential area for providing reliable and predictable communications networks for the Industry, especially for wired networks [3]. Currently there are efforts for implementing determinism in wireless network (e.g. [4]), in the I4.0 context, as one of the issues in this context is the variability of the wireless networks performance. A major line of research in this area aims at avoiding the inherent jitter that a wireless network introduces in the end-user communications. Most of current state of art solution proposals rely on buffering the end-user traffic for apply de-jittering algorithms. In fact, it is a standard capability of 5GS called hold-forward [5], which was introduced in 3GPP Release 16 that provides a de-jitter mechanism for 5G networks. In [6] a de-jittering approach based on varying network delays for audio traffic is proposed. And in [7], they propose the use of historical information for adapting the multimedia streams to the possible delay variation. In [8] the authors propose to implement de-jittering in each hop by using different priorities and queues. All these works assume that the de-jittering mechanism is aware of either certain end-to-end parameters, or the periodic nature of the traffic (e.g. audio or video).

In this work we present a novel mechanism that relies on the modeling of 5G network provided by a Network Digital Twin for training a Deep Reinforcement Learning agent, which will de-jitter the traffic without assuming or requiring any a priori knowledge of end-to-end metrics (delays, jitter) or the nature of the traffic. This new method is not only implemented but also thoroughly validated over a real 5G network. Our proposal can be implemented without modification in the real traffic (e.g. timestamping the packets) and without external requirement in the UE. As we demonstrate in this work, this allows to implement the mechanism in a vanilla 5G network, using commercially available User Equipment without any modification.

This paper is organized in several sections as follows. Section II describes the materials and methods used for the development of the proposed smart de-jittering algorithm. Section III includes the relevant results of different flavors of smart de-jittering from the experimentation over the real 5G network previously described in section II. Section IV contains the discussion of the obtained results and, finally, Section 0 includes the relevant conclusions of this research work.

II. Method and Materials

Key methods and material used in the development of this work are described in the following sub-sections.

A. Traffic Model

In this work an industrial use case defined at [9] is used as reference. In the addressed use case, an industrial element (e.g. a robot) communicates with a Programmable Logic Controller (PLC) using a 5G network. The traffic was modelled as described in TABLE I.

In the experimental setup, both industrial element and PLC are simulated by using programmable traffic generators implemented in the Network Digital Twin platform. The PLC simulator is attached to a 5G CPE (Askey 5G NR ODU NUQ3000M), which uses the 5G network described in section B; and the Industrial Element is deployed in the Data Network (DN) of the 5G network. In Fig. 1 we describe the setup of the experiment.

TABLE I. INDUSTRIAL USE CASE TRAFFIC MODEL

Traffic Model parameter	Value
Packet size	1440
Protocols	UDP/IP
Direction	Uplink/Downlink
Traffic Type	Periodic
Interval	10 milliseconds

B. 5G network

We use the 5Tonic[1] open laboratory, which provides a commercial 5G SA network. In particular, a portable Non-Public Network (NPN) with shared radio and control plane and private user plane, which is inspired in 5G-ACIA white paper [1], is used. The NPN part, as shown in Fig. 1, contains the radio elements (radio unit, antennas and gNB), the User Plane Function (UPF) and the transmission equipment required for the integration with 5Tonic's 5G Core and cloud applications.

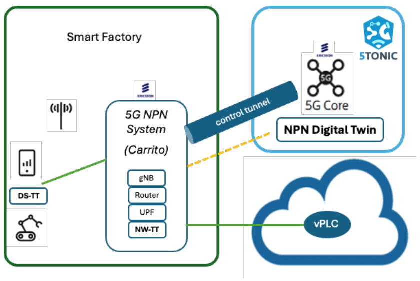

Fig. 1. High level design of Non-Public Network deployment.

The NPN is built on a portable system, designed to be easily re-located to varied premises, as shown in the Fig. 2, where different on-premises deployments are considered. The goal is to extend the experimentation environment provided by 5Tonic laboratory to the selected end-user premises for experimentation, and that includes not only the 5G network but also the experimentation tools, like for example the *traffic generator* and the *probes*, which are used in this work.

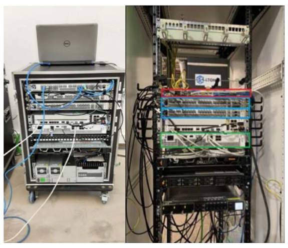

Fig. 2. NPN portable system in different on-premises deployments (photos from [10]).

The *traffic generator* uses the model defined in TABLE I. for emulating our target traffic. The *probes* are software probes deployed in different capturing points (UE, 5G Core N3 interface, 5G Core N6 interface, vPLC) and provide metrics of our traffic, such as Throughput, One-Way Delay (OWD), Packet Loss for both downlink and uplink traffic. All *probes* are synchronized in time using NTP through a local OAM network that guarantees errors of the order of microseconds (as showed in Fig. 3), which is acceptable for the accuracy required for the measurements to be obtained.

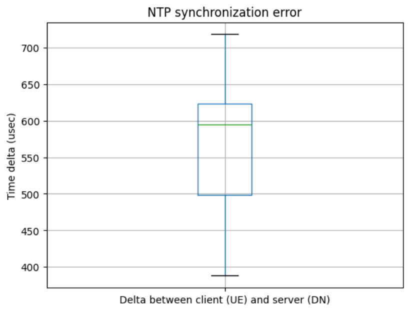

Fig. 3. NTP synchronization error during the experiments.

C. Network Digital Twin

One key element provided by the 5Tonic laboratory is a Network Digital Twin (NDT) of the Non-Public Network, as described in depth in [10]. As explained below, we propose to use the NDT to model the 5GS response in terms of latency to the use case traffic.

D. Smart de-jittering

In this work we present a novel method for performing the traffic de-jittering using Deep Reinforcement Learning (DRL), which consist in a reinforcement learning using machine learning. The main idea is to not to have to timestamp the traffic but, instead, leverage a Network Digital Twin in order to model the latency introduced by the 5GS and to use that model to train our DRL solution.

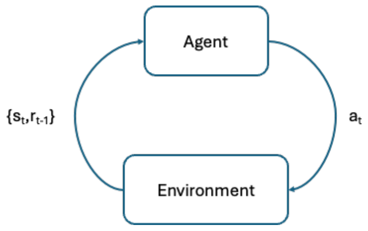

Fig. 4. Basic Deep Reinforcement Learning schema.

Following the schema described in Fig. 4, we create:

- An **environment** that simulates the 5GS response to the use case traffic. The simulation is based on the statistical information collected by NDT, and the **environment** is capable of generating a pattern of One-Way Delay (OWD) that matches with the probability distribution of OWD of the real 5G NPN system.
- An **agent** that, based on the input observation provided by the **environment**, identifies the best action to be performed on it.

In our solution the **environment** provides the following information to the **agent:**

- **State:** The *state* (or observation) is the current state of the *environment*, and it is used by the *agent* to take decisions. It contains the *Inter-Packet Arrival* (IPA), in milliseconds, of the current packet compared to the previous one, and a *delta,* in milliseconds,

[1] https://www.5tonic.org

corresponding to the forwarding time of the last packet.

- **Reward**: The *reward* defines how well the *agent* performs in the previous step. Different methods for calculating the *reward* are used, being described in section *E*.

The *agent* interacts with the *environment* by the defined *action,* which consist in the buffer time (in milliseconds) to apply to current packet.

The goal of the proposed smart de-jittering approach is to reduce the baseline jitter (Fig. 5) by using simple information that can be calculated in any of the 5GS endpoints without having to modify the end-user traffic.

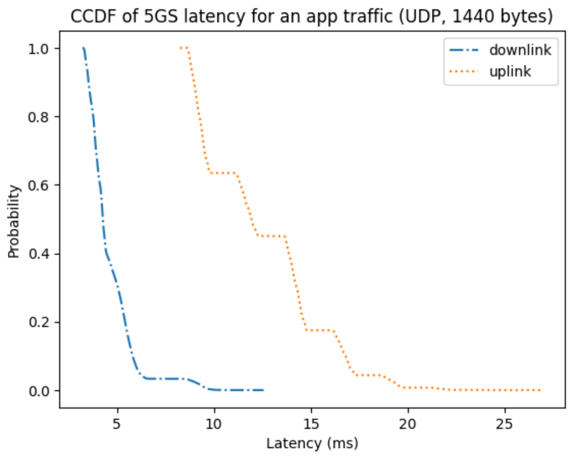

Fig. 5. Complementary Cumulative Distribution Function of latency using 5Tonic's NPN deployment for use case traffic model.

E. *Reward*

Based on our experience ([10], [11], [12]), the **reward** is key to obtain good results in DRL. In this work three different methods were defined for calculating the *reward*:

- Jitter method: Calculates the jitter between two consecutives packets.
- Average OWD: Calculates the difference between current packet OWD with the average OWD.
- Packet Delay Budget (PDB): Defines a packet delay budget and calculates the difference between the PDB and the current OWD for each packet.

In all cases, if the difference is less than 1 millisecond, the step reward will be positive (*+1*), and if not, the reward will be negative *(-difference*). For the *PDB* method a strict hyperparameter that reduces the threshold to 0.5 milliseconds is also used.

F. *Traffic model generation*

In order to validate our approach, we use in the training of the agent not only the traffic generated using the NDT platform but also real traffic. With that, the accuracy of our NDT approach can also be estimated.

G. *Hyperparameters*

During our experimentation, we define the following hyperparameters:

- *Reward method*, as described in section E.
- *Reward modifier*: weight to apply to the reward.
- *Evaluation frequency*: Number of steps to evaluate the agent performance.
- *Time steps*: Number of steps for the training.
- *PDB*: target PDB for this reward method.
- *Strict*: threshold for the rewards (0.5 or 1.0 millisecond)
- *Traffic model*: model of traffic used during training.

H. *Agent*

Both *agent* and *environment* are implemented using Python. In case of the *agent*, we use *stable baselines3* [13] as our DRL library, and we use a Proximal Policy Optimization [14] implementation.

I. *Training and Validation*

With all the setup described in this paper, different *agents* are trained, using a combination of the hyperparameters for Uplink and Downlink scenarios. One of the hyperparameters define how many steps are used for the training, which order of magnitude is 10 million steps. The environment executes episodes of 1024 packets, so the maximum *reward* is 1024. Every *Evaluation Frequency* steps we evaluate the performance of the agent, as showed in Fig. 6. In each evaluation the obtained performance is evaluated vs. the previous results, and the best model is saved. That allowed us to obtain the best performance model even if we over trained the agent.

After the training the best evaluated model is selected and then validated using real traffic over the real 5G network, as described above. All results included in the next section are obtained in this way. That allowed us to verify the real performance of the agent in a real scenario.

III. Results

In this section we include the relevant results of our work.

As example, the Fig. 6 shows the training and performance evaluation of a model. In this case, the training step is set to *1e7* steps, and the evaluation is based on the result of executing 50 episodes each *Evaluation Frequency* steps. The average *reward is shown in Red*, and the minimum and maximum *reward* is shown in Blue.

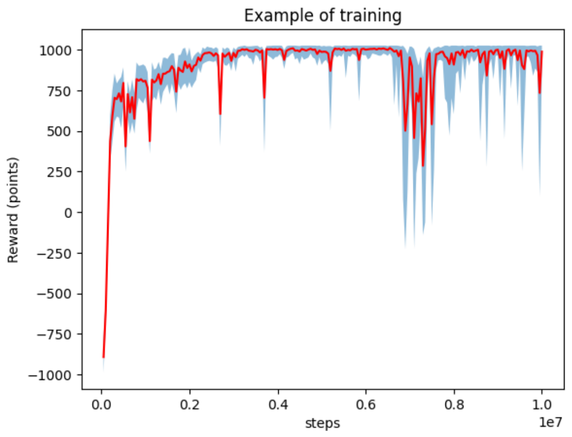

Fig. 6. Example of agent training.

The Fig. 7 shows the validation of the agent using different reward methods for the training. For the comparison, the original One-Way Delay is shown, along with the output of the smart de-jittering using different reward methods:

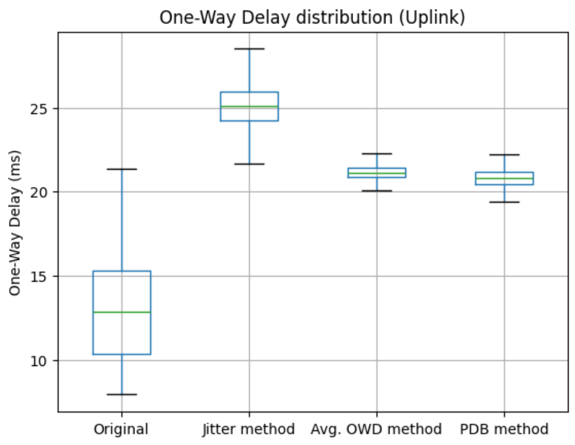

Fig. 7. De-jittering using different methods for Uplink traffic.

TABLE II. presents the details of the results of the different methods for uplink. The jitter is calculated by not including the *outliers*, that is, the difference between the minimum value of the OWNand the $Q_3 + 1.5 * IQR$, where Q_3 is the third quartile and IQR is the Interquartile range ($Q_3 - Q_1$). The *precision* is the percentage of packets which OWD are included in the jitter.

TABLE II. UPLINK TRAFFIC DETAILS

Case	Jitter (ms)	Precision
Original	19.9	0.999
Jitter method	7.24	0.999
Avg. OWD	2.23	0.993
PDB	2.73	0.995

The Fig. 8 and TABLE III. provide the same comparison of validation for downlink traffic. In both figures the best combination of hyperparameters using the minimum average One-Way Delay was selected.

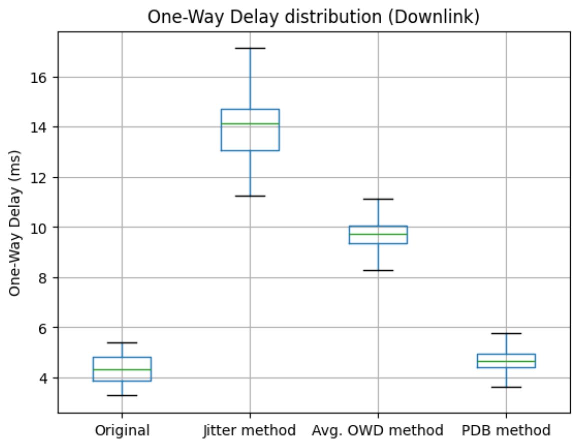

Fig. 8. De-jittering using different methods for Downlink traffic.

TABLE III. DOWNLINK TRAFFIC DETAILS

Case	Jitter (ms)	Precision
Original	3.76	0.974
Jitter method	6.65	0.9993
Avg. OWD	2.88	0.992
PDB	2.19	0.979

Fig. 9 shows the result of using strict training (PDB variation less than 0.5 ms) for PDB method:

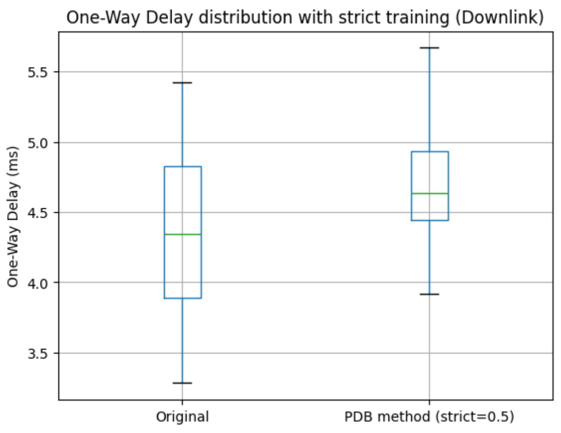

Fig. 9. De-jittering using PDB strict method for Downlink traffic.

Fig. 10 and TABLE IV. show the comparison of agent performance using NDT modelled traffic and real traffic (captured from 5G network, but simulated. Please refer to II.A) for the training. We also include the precision of the jitter, which is calculated as percentage of packets which OWD is less than $Q_3 + 1.5 * IQR$, where Q_3 is the third quartile and IQR is the Interquartile range ($Q_3 - Q_1$).

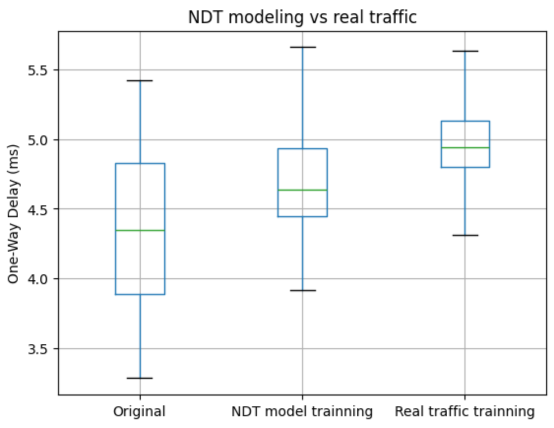

Fig. 10. NDT modeling versus Real Traffic for training.

TABLE IV. PRECISION WITH DIFFERENT TARGET PDB AND MODELS FOR TRAFFIC

Traffic	Average OWD (ms)	Jitter (ms)	Precision
NDT model	4.81	1.95	0.95
Real traffic	5.12	1.33	0.90
Real traffic	8.9	1.77	0.96
Real traffic	9.7	1.68	0.98
Real traffic	11.5	2.33	0.94

IV. Discussion

For uplink traffic (TABLE II.), the result of all methods presents improvements over the original Jitter, performing effective de-jittering with different degrees and precision. *Jitter* method is the one performing worsts, as this method focuses on the "local" jitter of two consecutives packets, which is able to reduce to less than *0.5 ms*. However, if we compare all packets, we notice that the OWD fluctuates, and the resulting jitter is still high. The other methods perform better, and the jitter is close to *2 milliseconds* with a precision similar to the original one.

For downlink we see (TABLE III.) a similar result: *Jitter method* is not performing well (in general) and the other two present improvements in the jitter with similar (or even better) precision than the original. Focusing in the *PDB* method, we see at Fig. 9 that being stricter with the reward provides small improvements in the jitter, at the expense of the precision.

Finally, we compare the result of the agent trained with the NDT modeled traffic versus the agent trained with real traffic. We can see that the jitter improves in this case but the precision decreases, and in this case *1* of each *100* packets *OWD* are not in the limit of *1.33 ms.* of *jitter*. As stated in TABLE IV. , using real traffic for traffic requires an increase of the PDB for improving both *jitter* and *precision*.

V. Conclusions

In this work we presented a novel mechanism for implementing effective de-jittering without having to modify the end-user traffic, which allows to implement this mechanism on top of existing 5G networks without demanding special support on the UE side. We demonstrated that a well-trained DRL agent can perform the de-jittering by just using the information available in one of the endpoints, and without knowing the end-to-end jitter or delay. For that, we successfully relied on a Network Digital Twin that models the behavior of a particular architecture of a 5G System, and we used such information to train the DRL agent. All these claims are backed by the results presented and discussed in this paper, which provide real validation of smart de-jittering with significant results, especially for uplink traffic, where the original jitter was worse. The NDT modeling is also validated, as it provides even a better way of training the smart de-jittering.

There are two limitations that this work did not address. The first one is the *long tail problem*, which especially impacts downlink traffic. In our opinion this issue must be addressed in lower layers, as the solution in transport layer only consist in increasing the target PDB. The second one is regarding the nature of the traffic that is not well captured by the agent. The use case model is periodic, and we estimate that a recurrent neural network could produce better results.

For future work we plan to: (i) improve the NDT traffic model, which will allow us to (ii) validate the Smart De-jittering with other kind of traffic (e.g. burst traffic) and (iii) use recurrent model for improving the final jitter.

Acknowledgement

This work has been partially funded by the European Commission Horizon Europe SNS JU PREDICT-6G (GA 101095890) Project and the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D 6G-EDGEDT (TSI-063000-2021-124), 6G-DATADRIVEN (TSI-063000-2021-132) and 6G-DAWN (under references TSI-063000-2021-54, TSI-063000-2021-55) projects.

References

[1] 5G-ACIA, "5G Non-Public Networks for Industrial Scenarios," 5G ACIA, White Paper WP_5G_NPN_2019_01., Jul. 2019. [Online]. Available: https://5g-acia.org/wp-content/uploads/2021/04/WP_5G_NPN_2019_01.pdf

[2] M. Lorenzo *et al.*, "Innovation Trends in I4.0 enabled by 5G and Beyond Networks." Zenodo, Oct. 2023. doi: 10.5281/zenodo.8367578.

[3] "Time-Sensitive Networking (TSN) Task Group |." Accessed: Dec. 05, 2022. [Online]. Available: https://1.ieee802.org/tsn/

[4] János Farkas, Balázs Varga, György Miklós, Joachim Sachs, "5G-TSN Integration for Indrustrial Automation." Ericsson Technology Review, Aug. 27, 2019. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-tsn-integration-for-industrial-automation

[5] 3GPP, "System architecture for the 5G System (5GS)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, Sep. 2022. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[6] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of INFOCOM '94 Conference on Computer Communications*, 1994, pp. 680–688 vol.2. doi: 10.1109/INFCOM.1994.337672.

[7] C. J. Sreenan, J.-C. Chen, P. Agrawal, and B. Narendran, "Delay reduction techniques for playout buffering," *IEEE Trans. Multimed.*, vol. 2, no. 2, pp. 88–100, 2000, doi: 10.1109/6046.845013.

[8] B. M. Nyambo, Gerrit. K. Janssens, H. Marufu, M. Munyaradzi, B. Mapako, and K. Dzinavatonga, "Queue Modelling and Jitter Control in Mobile Ad Hoc Networks," in *2022 1st Zimbabwe Conference of Information and Communication Technologies (ZCICT)*, 2022, pp. 1–6. doi: 10.1109/ZCICT55726.2022.10046032.

[9] P.-6G Consortium, "D1.1 Analysis of use cases and system requirements." Zenodo, Jul. 2023. doi: 10.5281/zenodo.8138548.

[10] M. M. Roselló, J. V. Cancela, I. Quintana, and M. Lorenzo, "Network Digital Twin for Non-Public Networks," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 495–500. doi: 10.1109/WoWMoM57956.2023.00086.

[11] M. M. Roselló, "Multi-path Scheduling with Deep Reinforcement Learning," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 400–405. doi: 10.1109/EuCNC.2019.8802063.

[12] C. Villasante Marcos, "Artificial Intelligence Edge Applications in 5G Networks," in *Proceedings of Sixth International Congress on Information and Communication Technology*, X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi, Eds., Singapore: Springer Singapore, 2022, pp. 269–279.

[13] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms." 2017.