

Reinforcement Learning for Adaptive Synchronization in 5G Network Digital Twins

Abdessamed Seddiki, Lemia Louail, Moufida Maimour, Eric Rondeau

Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France

abdessamed.seddiki, lemia.louail, moufida.maimour, eric.rondeau@univ-lorraine.fr

Abstract—Network Digital Twins (NDTs) require timely synchronization to remain aligned with their physical counterparts under mobility, traffic variations, and changing radio conditions. However, static or manually configured synchronization policies may be inefficient: overly frequent updates increase communication and processing cost, whereas insufficient updates can degrade the fidelity of the digital representation. This paper proposes a Reinforcement Learning (RL) framework for adaptive synchronization in 5G NDTs. The proposed controller selects the synchronization frequency according to network dynamics, physical-digital mismatch, and communication conditions. Digital Twin (DT) fidelity is evaluated through physical-digital mismatches in User Equipment (UE) position, Signal-to-Interference-plus-Noise Ratio (SINR), and throughput. The synchronization-frequency selection problem is formulated as an offline IQL task and solved using an Implicit Q-Learning (IQL)-based controller, followed by an offline-to-online refinement loop. Evaluation results show that the proposed controller improves the fidelity-cost trade-off compared with non-adaptive and offline-only baselines, mainly by reducing severe physical-digital mismatch events while maintaining a moderate synchronization rate.

Index Terms—Network Digital Twin, adaptive synchronization, reinforcement learning, Implicit Q-Learning, fidelity.

I. INTRODUCTION

Network Digital Twins (NDTs) are emerging as a promising paradigm for monitoring, analyzing, and optimizing next-generation networks. By maintaining a virtual representation of the physical network, an NDT can support performance assessment and closed-loop control without directly disturbing the real infrastructure. This capability is particularly relevant for 5G networks, where user mobility, radio conditions, traffic demand, and service requirements evolve rapidly over time [1], [2]. A central challenge in NDT systems is maintaining synchronization between the physical network and its digital counterpart. In dynamic 5G environments, User Equipment (UE) mobility, radio-channel fluctuations, and traffic variations continuously modify the physical state, which can increase the mismatch with the digital representation if updates are not performed at an appropriate rate.

The synchronization frequency therefore creates a central fidelity-cost trade-off. Frequent updates improve Digital Twin (DT) freshness but increase communication and processing overhead, whereas sparse updates reduce cost but increase physical-digital mismatch under mobility, radio variations, or changing traffic. This trade-off has motivated adaptive synchronization strategies for DTs, including Reinforcement

Learning (RL)-based approaches that aim to improve reliability and reduce physical-digital mismatch [3]–[5].

Existing studies have addressed several components required to construct and synchronize NDTs. Simulation-based approaches have shown how tools such as Simu5G can provide simulation services within Digital Twin Mobile Networks, while state-management platforms such as Eclipse Ditto have been used to represent connected entities and expose their evolving states [6], [7]. These works establish simulation engines and state repositories as important NDT building blocks, but they do not address how the synchronization frequency should be dynamically controlled according to the observed physical-digital mismatch.

Several works have formulated DT synchronization as an optimization or learning problem. Li *et al.* investigated budget-constrained synchronization in UAV-assisted mobile-edge/Internet of Things (IoT) systems, where the objective is to decide which sensor states should be refreshed under a limited update budget [8]. Cakir *et al.* combined predictive synchronization with the DTSYNC protocol to adapt the twinning frequency in resource-constrained DT communications [9]. Yang *et al.* jointly optimized device scheduling, transmit power, and data offloading to reduce DT synchronization delay over wireless networks [10]. Learning-based approaches have also considered joint synchronization and migration in mobile-edge networks [3], continual RL for device and radio-resource selection in industrial wireless DTs [4], and mobility-aware DT migration across edge servers [11].

Although these works demonstrate the relevance of optimization and RL for DT synchronization, they mainly focus on state-update selection, resource allocation, delay reduction, or DT migration. They do not jointly address explicit synchronization-frequency selection, multi-dimensional physical-digital fidelity, and cross-simulator operation in a concrete 5G NDT. In particular, the synchronization decision is rarely driven by a fidelity score that jointly captures UE position, Signal-to-Interference-plus-Noise Ratio (SINR), and throughput mismatches.

This paper addresses this gap by considering adaptive synchronization-frequency control in 5G NDTs. We formulate this problem as a finite-action Markov Decision Process (MDP), in which the controller observes network dynamics, communication conditions, and physical-digital mismatch at each decision interval, and then selects the synchronization frequency for the next interval. The resulting MDP is solved in

an offline RL setting using an Implicit Q-Learning (IQL)-based controller. A targeted offline-to-online refinement mechanism is subsequently activated only when difficult synchronization regimes or insufficiently covered state-action regions are identified. The objective is to balance synchronization cost against fidelity degradation, quantified by aggregating position, SINR, and throughput errors into a single fidelity measure.

The main contributions of this paper are summarized as follows:

- We design and implement a cross-simulator 5G NDT architecture for adaptive synchronization control.
- We define a multi-dimensional fidelity metric based on a normalized Minkowski distance over heterogeneous network variables, including position, SINR, and throughput.
- We formulate adaptive synchronization-frequency selection as an offline RL problem and address it using an IQL-based controller with targeted offline-to-online refinement.
- We evaluate the proposed controller against fixed and adaptive synchronization baselines using physical-digital position, SINR, and throughput errors.

The remainder of this paper is organized as follows. Section II presents the system model and formulates the synchronization-frequency control problem. Section III describes the proposed IQL-based adaptive synchronization controller and its offline-to-online refinement process. Section IV reports the experimental setup and performance evaluation. Finally, Section V concludes the paper.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider the 5G NDT architecture shown in Fig. 1, which consists of a physical network, a digital-side replica, and a synchronization-control plane. The architecture is implemented as a cross-simulator stack combining OM-Net++/INET/Simu5G for physical-network simulation [12], Eclipse Ditto for DT state management [7], and ns-3/5G-LENA for virtual network projection and performance estimation [13]–[15].

The physical side, implemented in OM-Net++/INET/Simu5G, reproduces the evolving 5G network and generates the current state of the monitored UEs. These state updates are transmitted to the digital side through a configurable communication link that may introduce latency, jitter, packet loss, and bandwidth limitations.

On the digital side, the received information updates the DT state maintained in Eclipse Ditto, while the corresponding network state is projected into ns-3/5G-LENA to estimate performance indicators. At each decision interval, the synchronization-control plane compares the physical and digital states. Their discrepancy is quantified by the fidelity metric defined in Section II-B. Based on this information and the current communication conditions, the control plane selects the synchronization frequency for the next interval.

A. Physical and Digital State Models

Let $\mathcal{U} = \{1, 2, \dots, N_u\}$ denote the set of UEs. At decision step t , the physical state of UE u is represented by

$$\mathbf{x}_{u,t}^P = [p_{u,t}^P, v_{u,t}^P, \lambda_{u,t}^P, s_{u,t}^P, r_{u,t}^P], \quad (1)$$

where $p_{u,t}^P$, $v_{u,t}^P$, $\lambda_{u,t}^P$, $s_{u,t}^P$, and $r_{u,t}^P$ denote position, mobility, traffic rate, SINR, and throughput, respectively. The corresponding digital state is

$$\mathbf{x}_{u,t}^D = [p_{u,t}^D, s_{u,t}^D, r_{u,t}^D], \quad (2)$$

where $p_{u,t}^D$, $s_{u,t}^D$, and $r_{u,t}^D$ are obtained from the digital-side state repository and virtual network projection.

The synchronization process transfers physical-state updates to the digital side. Let f_t be the synchronization frequency selected at step t . A higher f_t can improve freshness, but it also increases communication and processing overhead, thereby leading to higher energy consumption; a lower f_t saves resources but may increase the physical-digital mismatch.

B. Fidelity Model

The fidelity metric compares physical and digital states using normalized position, SINR, and throughput mismatches. For position, the scenario diameter is

$$d_{\max} = \sqrt{(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2}, \quad (3)$$

and the normalized position mismatch is

$$d_{u,t}^p = \frac{\|p_{u,t}^P - p_{u,t}^D\|_2}{d_{\max}}. \quad (4)$$

For SINR, the normalized mismatch is

$$d_{u,t}^s = |\hat{s}_{u,t}^P - \hat{s}_{u,t}^D|, \quad \hat{s}_{u,t} = \text{clip}\left(\frac{s_{u,t} - s_{\min}}{s_{\max} - s_{\min}}, 0, 1\right), \quad (5)$$

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum SINR values observed in the training data, and $\hat{s}_{u,t}^P$ and $\hat{s}_{u,t}^D$ denote the corresponding normalized physical and digital SINR values, respectively. Similarly, the normalized throughput mismatch is

$$d_{u,t}^r = |\hat{r}_{u,t}^P - \hat{r}_{u,t}^D|, \quad \hat{r}_{u,t} = \text{clip}\left(\frac{r_{u,t} - r_{\min}}{r_{\max} - r_{\min}}, 0, 1\right), \quad (6)$$

where $r_{\min}$ and $r_{\max}$ are the minimum and maximum throughput values observed in the training data, and $\hat{r}_{u,t}^P$ and $\hat{r}_{u,t}^D$ denote the corresponding normalized physical and digital throughput values, respectively. The clipping operation keeps normalized values bounded during evaluation.

The local mismatch of UE u is computed using a weighted Minkowski distance:

$$D_{u,t} = (w_p(d_{u,t}^p)^q + w_s(d_{u,t}^s)^q + w_r(d_{u,t}^r)^q)^{1/q}, \quad (7)$$

where $q \geq 1$, $w_p + w_s + w_r = 1$, and $w_p, w_s, w_r \geq 0$. The global mismatch and fidelity score are then defined as

$$D_t = \frac{1}{N_u} \sum_{u \in \mathcal{U}} D_{u,t}, \quad F_t = 1 - D_t. \quad (8)$$

A higher F_t indicates a more synchronized digital representation.

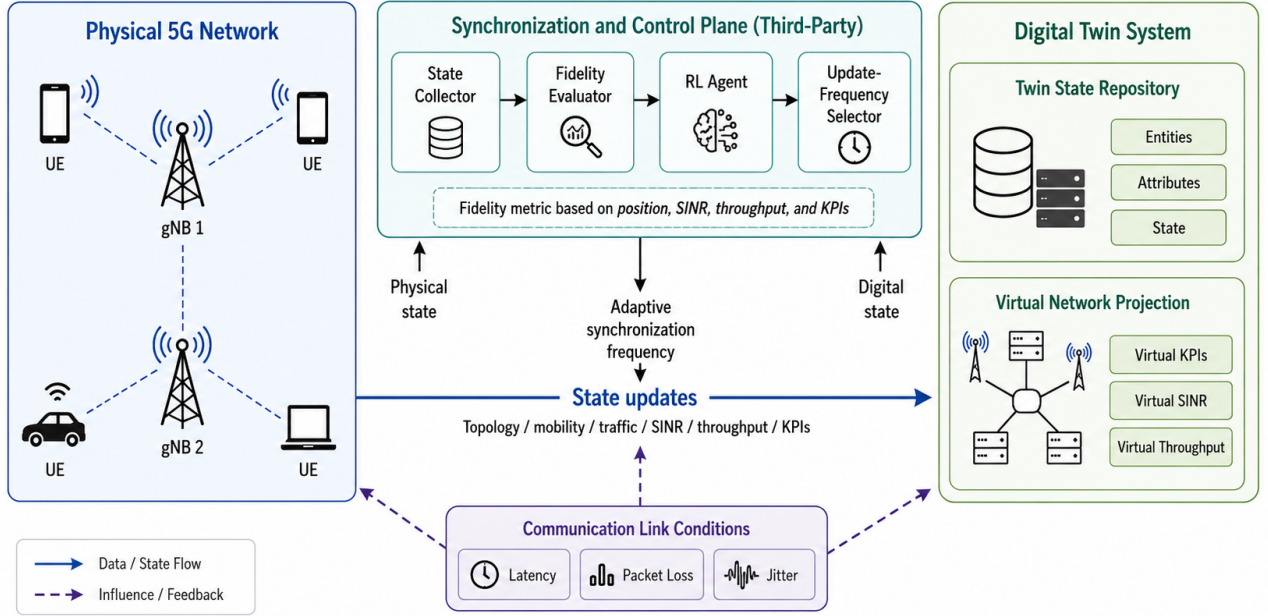


Fig. 1. The architecture of the proposed 5G NDT synchronization framework.

C. Problem Formulation

The objective is to select the synchronization frequency that minimizes the physical–digital mismatch while limiting synchronization overhead. Let $\mathcal{F}$ be the set of candidate synchronization frequencies and $C(f_t)$ the synchronization cost associated with frequency f_t . The problem is formulated as:

$$\begin{aligned} \min_{\{f_t\}_{t=1}^T} \quad & \sum_{t=1}^T (\alpha D_t + \beta C(f_t)) \\ \text{s.t.} \quad & f_t \in \mathcal{F}, \quad \forall t. \end{aligned} \quad (9)$$

where α and β control the trade-off between fidelity degradation and synchronization cost.

III. PROPOSED LEARNING-BASED SYNCHRONIZATION CONTROL

The proposed controller learns when to increase or decrease the synchronization frequency from previously collected synchronization traces. A purely online RL approach would require repeated interaction with the complete cross-simulator stack, where each action triggers physical-side execution, digital-state update, virtual projection, and metric alignment. This makes unconstrained online exploration costly and slow. Therefore, we formulate the problem in an offline RL setting, where the policy is learned from logged synchronization transitions. We then use IQL to extract a frequency-selection policy from this dataset, followed by a lightweight offline-to-online refinement stage.

A. Offline RL Formulation

The synchronization-control problem is formulated as a finite-action MDP, but the policy is first learned from an offline

dataset rather than by direct online exploration. Let $\mathcal{D}$ denote the set of logged synchronization transitions:

$$\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^N. \quad (10)$$

Each transition is obtained from a previously executed synchronization trajectory, where s_t is the observed synchronization state, a_t is the frequency action applied by the data-collection policy, r_t is the resulting reward, and s_{t+1} is the next observed state. During offline training, the proposed policy does not interact with the simulator; it only learns from the transitions stored in $\mathcal{D}$. After training, the learned policy is deployed to select the synchronization frequency online.

1) *State Space*: The state s_t summarizes the physical-network dynamics and the current digital state. It includes mobility indicators, traffic intensity, recent physical–digital mismatch, position, SINR, and throughput values from both sides, the previously applied frequency, and communication-condition features such as latency, jitter, and packet loss. These features allow the controller to adapt the update frequency according to both network dynamics and synchronization reliability.

2) *Action Space*: The action space is discrete, and each action $a_t \in \{1, \dots, K\}$ indexes one candidate synchronization frequency. We use finer granularity between 1 Hz and 10 Hz, where synchronization decisions are more sensitive, and coarser sampling at higher frequencies, where marginal gains are smaller relative to the cost:

$$\begin{aligned} f_t &= \mathcal{F}(a_t), \\ \mathcal{F} &= \{0.1\} \cup \{1, \dots, 10\} \cup \{20, 30, 40, 50\} \text{ Hz}. \end{aligned} \quad (11)$$

During offline training, a_t is the frequency action logged in the dataset.

3) *Reward Function*: The reward is derived from the fidelity-cost objective:

$$r_t(s_t, a_t) = -(\alpha D_t + \beta C(f_t)), \quad (12)$$

where D_t is the normalized physical–digital mismatch and $C(f_t)$ is the synchronization cost, under the assumption that energy overhead scales linearly with the selected update frequency. Thus, the reward favors low mismatch and discourages unnecessary high-frequency updates.

B. Offline IQL Synchronization Controller

We adopt IQL to learn the synchronization policy from the offline dataset $\mathcal{D}$. This method is well suited to our setting because it improves the policy from logged state-action pairs, without requiring costly online exploration on the full cross-simulator stack. It also limits the risk of overestimating actions that are poorly represented in the dataset.

IQL relies on three learned functions: an action-value function $Q_\theta(s, a)$, a state-value function $V_\psi(s)$, and a policy $\pi_\omega(a|s)$. Here, $V_\psi(s)$ estimates the value of the current synchronization state, while $Q_\theta(s, a)$ evaluates the frequency action applied in that state. For each transition $(s_t, a_t, r_t, s_{t+1}) \in \mathcal{D}$, the Q-function is trained using the target

$$y_t = r_t + \gamma V_\psi(s_{t+1}), \quad (13)$$

and the temporal-difference loss

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{\mathcal{D}} \left[(Q_\theta(s_t, a_t) - y_t)^2 \right]. \quad (14)$$

The policy is then extracted by giving more importance to dataset actions that are better than the estimated state value. This is measured by the advantage:

$$A(s_t, a_t) = Q_\theta(s_t, a_t) - V_\psi(s_t). \quad (15)$$

The value function is trained by expectile regression with parameter τ , while the policy is learned by advantage-weighted regression, where the temperature controls the emphasis on high-advantage actions.

To improve stability in the offline setting, we use reward clipping and scaling, conservative Q-regularization (CQL), a pessimistic two-head Q estimate, gradient clipping, and learning-rate scheduling.

C. Offline-to-Online Active Refinement

The initial dataset may miss difficult synchronization regimes, especially under fast mobility, degraded links, or rarely selected update frequencies. Therefore, the trained IQL policy is evaluated online, and useful transitions are retained for refinement. This includes cases where fidelity remains low, the learned policy is unstable, the policy collapses toward a narrow subset of frequencies, or difficult states are insufficiently covered by the offline dataset.

The new transitions are added to the offline dataset, and the next training cycle continues from the previously learned parameters whenever the previous checkpoint is stable. This avoids restarting the learning process from scratch while progressively improving the dataset coverage and the robustness

Algorithm 1 Triggered Offline-to-Online IQL Refinement

Require: Offline dataset $\mathcal{D}_0$ and offline-trained policy π_0

```

1:  $\pi^* \leftarrow \pi_0$ 
2: Diagnose  $\pi^*$  using validation metrics and training indicators
3: Compute fidelity, risk, selected-frequency distribution, and state-frequency coverage
4: if fidelity is insufficient or risk is high or coverage is limited then
5:   Identify the corresponding difficult synchronization regimes  $\Omega$ 
6:   Collect targeted online transitions  $\tau_\Omega$  under these regimes
7:    $\mathcal{D} \leftarrow \mathcal{D}_0 \cup \tau_\Omega$ 
8:   Continue IQL training from  $\pi^*$  on  $\mathcal{D}$  to obtain  $\pi_{\text{ref}}$ 
9:   Evaluate  $\pi_{\text{ref}}$  using the same indicators
10:  if  $\pi_{\text{ref}}$  is stable and improves the fidelity–risk trade-off then
11:     $\pi^* \leftarrow \pi_{\text{ref}}$ 
12:  else
13:    Retain  $\pi^*$ 
14:  end if
15: else
16:   Retain  $\pi^*$ 
17: end if
18: return  $\pi^*$ 

```

of the synchronization controller. The refinement process is summarized in Algorithm 1.

IV. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

This section presents the experimental setup and performance evaluation of the proposed IQL-based synchronization controller. We describe the offline dataset, training configuration, held-out true-online test scenario, evaluation metrics, and baseline comparison. All reported values correspond to raw physical–digital errors; therefore, lower values indicate better synchronization quality.

A. Dataset, Training, and Validation Setup

The offline dataset was generated from eight scenarios executed with the proposed cross-simulator 5G NDT stack. Overall, it includes 95,845 transitions, corresponding to approximately 12,000 s of simulated traces. These scenarios involve up to five monitored UEs and one Next Generation Node B (gNB), and cover heterogeneous conditions in terms of mobility, traffic, and physical–digital communication quality. The mobility configurations include stationary, linear, and Gauss-Markov movements, while the traffic configurations include uplink and downlink flows with on/off, uniform, and exponential packet-generation patterns. The physical–digital communication profiles range from a quasi-ideal link to degraded links with delay, loss, and bandwidth limitation.

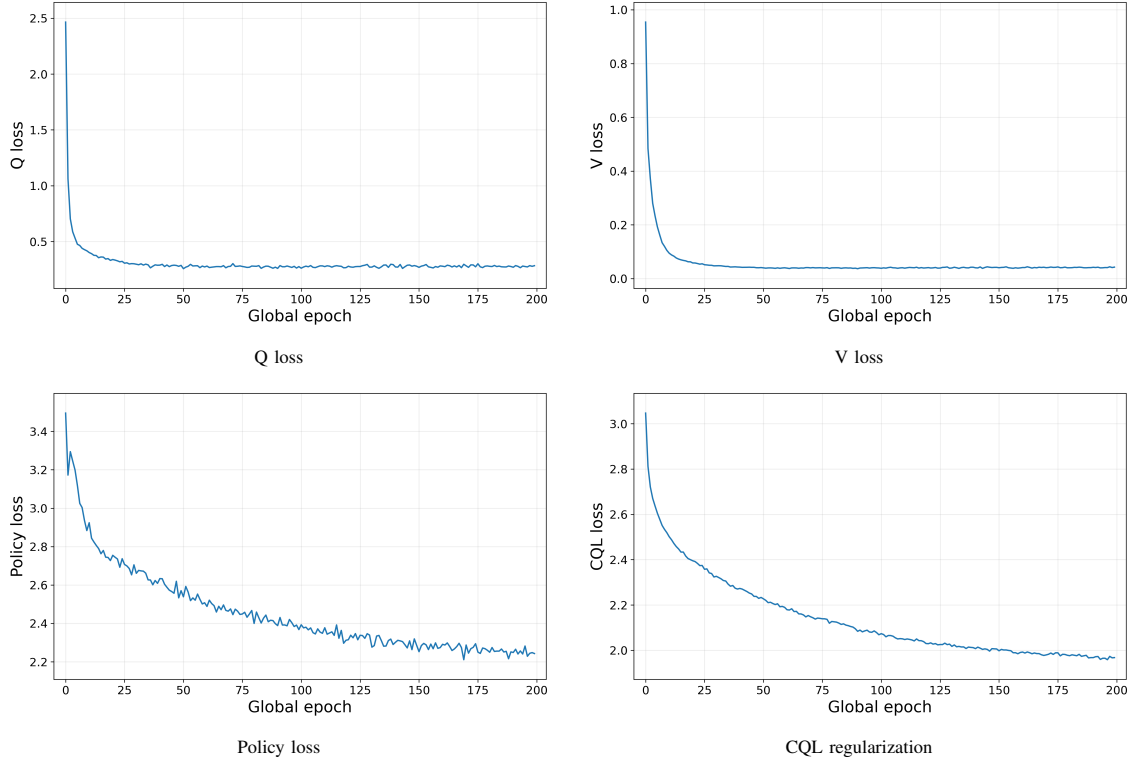


Fig. 2. Training behavior of the proposed offline IQL controller.

TABLE I
MAIN EXPERIMENTAL AND LEARNING PARAMETERS.

Parameter	Value
Minkowski order q	2
Fidelity weights (w_p, w_s, w_r)	(0.55, 0.15, 0.3)
Reward weights (α, β)	(1.0, 0.5)
Cost function	$C(f) = f/f_{\max}$
Discount factor γ	0.90
Expectile τ	0.70
Advantage temperature	0.5
Batch size	256
Learning rates	10^{-4} for Q/V, 10^{-5} for P
Optimizer	Adam
Network architecture	MLP, 2 hidden layers, 256 units, ReLU

The final evaluation is conducted on a held-out true-online stress scenario that is not used during offline training or validation. This scenario contains one fixed gNB and 20 UEs. The traffic configuration includes 60 server applications. UE mobility follows a Gauss-Markov model, with speeds uniformly sampled between 30 m/s and 70 m/s.

The IQL controller is trained at the decision level using logged synchronization transitions, and the main training parameters are summarized in Table I.

B. Evaluation Metrics

We evaluate raw physical-digital errors for position, SINR, and throughput. Position error is measured as the Euclidean distance between the physical and digital locations of each UE, while SINR and throughput errors are computed as absolute differences. For each error type e , we report three

complementary indicators. The mean error captures the average synchronization accuracy, while CVaR_{99} focuses on the average severity of the largest 1% mismatch events. However, the mean may hide rare but severe errors, whereas CVaR_{99} focuses on the extreme tail and ignores most regular states. To capture both effects, we also use a concave-distortion risk score:

$$\rho_h(e) = \int_0^{+\infty} h(\mathbb{P}(e > u)) du, \quad h(p) = \sqrt{p}. \quad (16)$$

This metric considers the full error distribution through the exceedance probability $\mathbb{P}(e > u)$, while the concave function h gives additional weight to rare exceedance events.

C. Training Convergence

Fig. 2 reports the training behavior of the offline IQL controller. The Q and V losses decrease rapidly during the first epochs and then reach a stable regime, indicating stable critic and value estimation. The policy loss decreases more gradually because the policy is extracted from advantage-weighted dataset actions rather than optimized through direct online interaction. The CQL regularization remains bounded, which helps limit overestimation on poorly represented actions. Overall, the curves indicate a stable offline training process before online deployment.

D. Comparison with Baseline Synchronization Policies

Table II compares the proposed controller with random, cyclic, threshold-based heuristic, and offline IQL baselines.

TABLE II
PROPOSED IQL CONTROLLER VS. BASELINE SYNCHRONIZATION POLICIES.

Method	Freq. mean (Hz)	Position error (m)			SINR error (dB)			Throughput error (Mbps)		
		Mean	CVaR99	$\rho \sqrt{\cdot}$	Mean	CVaR99	$\rho \sqrt{\cdot}$	Mean	CVaR99	$\rho \sqrt{\cdot}$
Random	13.39	11.79	140.77	28.94	18.43	27.93	20.93	11.79	379.51	66.95
Cyclic	11.92	10.02	42.88	16.11	18.20	28.00	20.87	3.41	235.32	45.32
Heuristic	6.37	10.89	50.46	18.86	18.61	28.35	21.25	1.41	28.17	7.16
Offline IQL	2.0	9.81	41.33	11.88	19.05	41.31	25.01	2.87	191.37	41.58
Our controller	3.15	9.21	22.52	12.44	18.54	27.42	20.19	1.41	26.17	5.16

The proposed controller achieves the lowest position mean error, reducing it from 10.02 m for the cyclic baseline and 9.81 m for offline IQL to 9.21 m. In addition, it reduces severe mismatch events, with position CVaR₉₉ decreasing from 41.33 m for offline IQL to 22.52 m.

For SINR, the mean errors remain close across methods, while the proposed controller obtains the lowest tail-sensitive errors. For throughput, it matches the best mean error achieved by the heuristic baseline, while reducing CVaR₉₉ from 28.17 Mbps for the heuristic baseline and 191.37 Mbps for offline IQL to 26.17 Mbps. Compared with offline IQL, the proposed refinement increases the average frequency from 2.0 Hz to 3.15 Hz, but significantly improves robustness under the held-out stress scenario. Overall, these results show that offline-to-online refinement improves the fidelity-cost trade-off by reducing severe physical-digital mismatches while maintaining a moderate synchronization rate.

V. CONCLUSION

This paper proposed an adaptive synchronization-frequency control framework for a cross-simulator 5G NDT. The framework combines OMNeT++/INET/Simu5G, Eclipse Ditto, and ns-3/5G-LENA, and uses a fidelity metric based on position, SINR, and throughput mismatches to guide synchronization decisions. The update-frequency selection problem was formulated as an offline RL task and solved using an IQL-based controller with offline-to-online refinement. The results show that the resulting controller improves the fidelity-cost trade-off compared with random, cyclic, threshold-based, and offline-only baselines, mainly by reducing severe physical-digital mismatch events while maintaining a moderate synchronization rate.

REFERENCES

- [1] X. Lin, L. Kundu, C. Dick, E. Obiodu, T. Mostak, and M. Flaxman, "6G Digital Twin Networks: From Theory to Practice," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 72–78, 2023.
- [2] S. M. Raza, R. Minerva, N. Crespi, M. Alvi, M. Herath, and H. Dutta, "A Comprehensive Survey of Network Digital Twin Architecture, Capabilities, Challenges, and Requirements for Edge-Cloud Continuum," *Computer Communications*, vol. 236, p. 108144, 2025.
- [3] W. Liu, Y. Fu, Y. Guo, F. L. Wang, W. Sun, and Y. Zhang, "Two-Timescale Synchronization and Migration for Digital Twin Networks: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Transactions on Wireless Communications*, vol. 23, no. 11, pp. 17294–17309, 2024.
- [4] H. Tong, M. Chen, J. Zhao, Y. Hu, Z. Yang, Y. Liu, and C. Yin, "Continual Reinforcement Learning for Digital Twin Synchronization Optimization," *IEEE Transactions on Mobile Computing*, 2025, early Access.
- [5] A. Seddiki, L. Louail, and M. Maimour, "Deep Reinforcement Learning for Adaptive Synchronization Control in Digital Twins," in *4e Congrès annuel de la Société d'Automatique, de Génie Industriel et de Productique (SAGIP 2026)*, Bordeaux, France, 2026.
- [6] G. Nardini and G. Stea, "Enabling simulation services for digital twins of 5G/B5G mobile networks," *Computer Communications*, vol. 213, pp. 33–48, 2024.
- [7] M. Kherbache, M. Maimour, and E. Rondeau, "Digital Twin Network for the IIoT Using Eclipse Ditto and Hono," *IFAC-PapersOnLine*, vol. 55, no. 8, pp. 37–42, 2022.
- [8] Y. Li, W. Liang, Z. Xu, W. Xu, and X. Jia, "Budget-Constrained Digital Twin Synchronization and Its Application on Fidelity-Aware Queries in Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 24, no. 1, pp. 165–182, 2025.
- [9] L. V. Cakir, C. J. Thomson, M. Özdem, B. Canberk, V. L. Nguyen, and T. Q. Duong, "Intelligent Digital Twin Communication Framework for Addressing Accuracy and Timeliness Tradeoff in Resource-Constrained Networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 3, pp. 1954–1965, 2025.
- [10] Z. Yang, M. Chen, Y. Liu, and Z. Zhang, "Optimizing Synchronization Delay for Digital Twin over Wireless Networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2024, pp. 9106–9110.
- [11] Y. Zhang, L. Wang, and W. Liang, "Deep Reinforcement Learning for Mobility-Aware Digital Twin Migrations in Edge Computing," *IEEE Transactions on Services Computing*, vol. 18, no. 2, pp. 704–717, 2025.
- [12] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Viridis, "Simu5G: An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.
- [13] N. Patriciello, S. Lagen, L. Giupponi, and B. Bojovic, "An Improved MAC Layer for the 5G NR ns-3 Module," in *Proceedings of the Workshop on ns-3*. ACM, 2019.
- [14] B. Bojovic, Z. Ali, and S. Lagen, "ns-3 and 5G-LENA Extensions to Support Dual-Polarized MIMO," in *Proceedings of the Workshop on ns-3*. ACM, 2022.
- [15] A. Larrañaga, M. C. Lucas-Estañ, S. Lagén, Z. Ali, I. Martinez, and J. Gozalvez, "An Open-Source Implementation and Validation of 5G NR Configured Grant for URLLC in ns-3 5G-LENA: A Scheduling Case Study in Industry 4.0 Scenarios," *Journal of Network and Computer Applications*, vol. 215, p. 103638, 2023.