AI and Dependency Graph–Based Digital Twins for Context-Aware 6G Robotic Networks

Kubra Duran, Mehmet Ozdem, Kaushik Chowdhury, and Berk Canberk

Abstract

6G robotic networks demand highly context-aware and intelligent management to meet extreme performance requirements. These requirements become even more challenging with real-time control and autonomous decision-making scenarios of 6G robotic applications. However, traditional methods fail to provide context-awareness and enhanced cognitive capabilities to meet these requirements in a robotic network. Therefore, in this paper, we introduce a novel dependency graph-based Digital Twin (DT) framework to provide context-awareness and cognitive decision-making capabilities in robotic applications. In this framework, we design a Data Distribution Service (DDS)-based DT layer and a context-aware mediator layer, comprising a Global What-If Engine and an AutoML-based Decision Unit. The What-If Engine constructs four types of dependency graphs based on temporal, data, control, and performance-related information to represent relationships between robots, sensor inputs, and control triggers. Moreover, the AutoML-based Decision Unit selects the most successful algorithm based on the prediction error and scenario conditions. Experimental results demonstrate that our proposed framework provides real-time DT synchronisation under even high-density twin scenarios without computationally overloading the system. The proposed framework also achieves effective learning-based decision-making for regression and classification tasks in 6G robotic applications.

Introduction

The next generation of mobile networks, namely 6G, is transforming the design and operation of the intelligent robotic networks. With key features such as Ultra-Reliable Low Latency Communication (URLLC) and massive Machine-Type Communications (mMTC), 6G enables highly dynamic interactions between autonomous robots, IoT sensors and peripherals [1].

In 6G robotic environments, customer experience translates into predictable service quality and continuity of robotic workflows. At this point, real-time perception, accurate prediction and adaptation are critical to meet these requirements. Additionally, these systems must not only sense their environment and maintain consistent communication but also reason over operational status and performance. However, traditional robotic frameworks cannot model complex interdependencies across system components, communication layers, and performance metrics. Besides that, the classical network management approaches such as *FCAPS* (Fault, Configuration, Accounting, Performance, and Security) are insufficient for handling the dynamic interactions and cognitive demands of the 6G robotic networks. In this regard, the need for context-aware models, cognitive decision capabilities, and reasoning becomes more important for the intelligent management of 6G robotic applications.

At this point, *Digital Twins (DTs)* emerge as a key enabler by providing a virtual representation of physical robots and IoT sensors by allowing for real-time analysis, and Artificial Intelligence/Machine Learning (AI/ML)-based decision-making in parallel with real-world operations [2], [3].

Challenges in 6G Robotic Networks

- *Lack of Full Context-Awareness:* It emerges as a critical requirement of the 6G robotic networks as robots must continuously perceive their physical and network contexts, interpret situational dynamics, and adapt their behaviours accordingly. In this regard, context-awareness involves (i) continuously sensing and aggregating data from heterogeneous sources, such as robots, IoT sensors, and network telemetry, (ii) interpreting these inputs accurately, and (iii) adapting decision logic accordingly. However, providing all these subtasks in such a dynamic network requires enhanced modelling and analysis capabilities while meeting the 6G's extreme KPIs.
- *Enhanced Cognitive Capabilities:* The 6G robotic network demands cognitive capabilities beyond static rule-based control. At this juncture, the intelligence should incorporate learning-based decision-making and

This work was supported by The Scientific and Technological Research Council of Türkiye (TÜBİTAK) 1515 Frontier Research and Development Laboratories Support Program for Türk Telekom 6G Research and Development Laboratory under Project 5249902.

Kubra Duran (corresponding author) and Berk Canberk are with the School of Computing, Engineering and the Built Environment, Edinburgh Napier University, EH10 5DT Edinburgh, U.K.; Mehmet Ozdem is with Turk Telekom, 06103 Ankara, Türkiye; Kaushik Chowdhury is with the Chandra Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78703 USA.

Digital Object Identifier: 10.1109/MCOMSTD.2026.3672744

adaptive coordination. This consists of learning from historical data, past actions, and system-wide impacts through the models. At this point, the key challenges regarding the intelligence of the 6G robotic network arise in selecting learning algorithms for diverse tasks according to their performance.

State of the Art

Recent studies explore the integration of DTs, AI, and graph-based representations to support emerging 6G robotics use-cases. In this context, the Internet of DTs (IoDT) survey, [4], positions DTs as a collaborative network of multiple twins and emphasises the role of intra and inter-twin communication design as key enabling directions for scalable deployments. From a DT-guided control perspective, Ho et al. [5] proposes an AI-powered DT-assisted control framework for automated warehouses over 5G URLLC. To enable real-time decision-making, this study uses AI/ML for robotic control. Similarly, Duran and Canberk [6] introduces a DT-based collaborative management framework by emphasising that DT synchronisation and interaction timeliness are constrained by end-to-end latency and bandwidth requirements. From a graph-based representation perspective, Zeng et al. [7] presents knowledge graph-driven intelligence and proposes knowledge-based robotic semantic communication, in which an edge server hosts a large-scale knowledge graph as a remote brain. In parallel, we introduce generative AI as a means of constructing adaptive twin models in our prior study [8]. A complementary research perspective focuses on DT-based evaluations. More specifically, Krieger et al. [9] integrates a real-time DT network with scaled physical testbeds by emulating radio-environment effects such as path loss and shadowing by imposing the resulting link constraints on the scaled environment. Finally, DTs are also used to accelerate AI lifecycle management. In this regard, Hou et al. [10] introduces a DT-powered online calibration model for automatic AI model selection. In this, a context-to-model mapping is optimised by generating synthetic data from multiple simulated contexts.

Overall, while existing studies demonstrate the benefits of DTs, AI, and graph-based models for 6G robotic applications, their graph abstractions are primarily used for structural representation [4], [7], [8], rather than as an operational input for system-wide what-if analysis. In addition, prior DT intelligence mechanisms are typically studied through resource optimisation within control loops [5], [6] or testbed-oriented calibration [9], [10], without explicitly modelling closed-loop dependencies. Furthermore, these works do not treat performance factors that directly affect DT timeliness, such as twinning rate. As a result, these limitations make it difficult to perform system-wide what-if analysis required for context-aware decision-making in 6G robotic networks.

At this point, our research question for this study is *"How can we design an efficient DT framework for the control and management of a 6G-enabled robotic network by considering context-awareness and enhanced intelligence capabilities?"*. To address this, we present an operational dependency-graph framework that enables system-wide what-if analysis across multiple twins. More specifically, we separate system dependencies into four categories: temporal, data, control, and performance to support low-overhead runtime updates. Here, the *time* dimension is to capture propagation paths to reason under delays and deadlines, while the *data* dimension is to capture data dependency for freshness requirements to determine which decisions become invalid under missing data. Similarly, the *control* dimension is to capture control and feedback commands to support DT-guided actuation, and the *performance* dimension is for observing the performance relation between system resources and KPIs. In our solution, the Global What-If Engine centralises scenario analysis and restricts execution to the affected dependency subgraph, thereby making the DT layer lighter.

Contributions

We list the contributions of our study below:

- We propose a DDS-based *DT Layer* and a *Context-aware Mediator Layer* to enable closed-loop operation under high-density twin workloads.
- We introduce a *Global What-If Engine* that utilises four types of dependency graphs based on temporal, data, control, and performance domains to perform context-aware management across multiple DTs by restricting execution to the affected dependency subgraph.
- We develop an AutoML-based *Decision Unit* that selects the most suitable learning model for each what-if scenario based on model error and per-sample latency threshold.

Alignment With Standardization Efforts

There are several ongoing standardization activities for DT and AI adoption in robotics and future communication networks. For instance, ISO 23247 DT Framework [11] defines the DT lifecycle, including data acquisition, modelling, and synchronisation in robotics and manufacturing environments. Our proposed DT framework aligns with this via its layered design by leveraging twinning frequency for real-time synchronisation and distinct data flows between the layers. In the context of AI-enabled networking, ITU-T Y. 3172 [12] defines an ML Sandbox as an isolated environment where ML models can be trained, tested, and evaluated without affecting the operational network. In our framework, the AutoML-based Decision Unit conceptually aligns with this, where candidate models are trained and evaluated without affecting the DT–robot communication flows. In addition, the OMG Data Distribution Service (DDS) [13] provides the standardised communication backbone. Regarding this, we design our DT framework with a DDS-based middleware to provide a reliable communication infrastructure. Overall, the proposed DT framework remains consistent with emerging standardisation activities toward 6G-enabled robotic DT deployments.

The remainder of the article is organised as follows: the "Proposed Framework" section gives the details of the proposed DT-based robotic management framework. The performance evaluation is given in the "Performance Evaluation" section.

Finally, the "Conclusion" section concludes the paper.

Proposed Framework

As given in Fig. 1, the proposed robotic DT framework consists of a Physical Space and a Virtual Space. In the Physical Space, there is a *Device Layer,* and in the Virtual Space, there are a *DT Layer* and a *Context-Aware Mediator Layer.*

Device Layer

This layer represents the set of real-world entities that interact directly with the environment and form the foundation of the 6G-enabled robotic networks. Each component in this layer can perform as a publisher or subscriber by interacting with the DDS middleware, mainly located within the DT Layer. Robotic arms execute actions based on control signals received from DT-driven decisions, while IoT sensors and robotic arms feed telemetry data, such as position, robot speed, and temperature, back to the system.

DT Layer

This layer hosts the DDS middleware core, which manages the communication between the physical and virtual spaces via a publish–subscribe mechanism.

1) DDS Topics: DDS middleware defines and manages four main topics that form the data exchange interface between the physical and virtual spaces:

- `/robot/telemetry`: This topic corresponds to the collection of state information from the physical space. It is for time-stamped observations such as robot motion, task progress, and sensor readings that are used to synchronise the DT state.
- `/dt/what-if`: This is to transfer scenario-level inputs to the Context-Aware Mediator Layer. Here, each message encodes a candidate action and constraint set consisting of latency bounds and slice selection to trigger on-demand what-if evaluation.
- `/dt/system-update`: This is to collect dependency graph updates and internal model state information. It propagates model updates so that subsequent decisions operate on the most recent dependency structure and twin state.
- `/dt/control`: This is to enable command dissemination back to the Device Layer. It delivers the selected control action for actuation and closed-loop feedback.

2) Data Flows: The proposed framework utilises four inter-layer data flows to ensure real-time synchronisation and feedback across the physical and virtual spaces, as given below:

- *Flow1:* Carries physical robots' data and sensor readings published from the Device Layer.
- *Flow2:* Carries physical space telemetry data to the Context-Aware Mediator Layer (the subscriber) to be used in the dependency graph construction and What-if simulations.
- *Flow3:* Carries the results of the decision engine and global What-If modules (the publishers) to the DT Layer.
- *Flow4:* Carries actuation and control commands from the DT Layer to the Device Layer.

All these flows are managed via the DDS middleware by ensuring deterministic message delivery within 6G latency constraints. Moreover, we kept DDS buffering bounded by utilising `KEEP_LAST` with a small history depth. Additionally, we apply a prioritisation by assigning *Reliable* QoS to time-critical control and feedback, while configuring telemetry updates as *BestEffort* to minimise QoS degradation.

3) Communication Requirements in 6G Robotic Use Case: In the 6G robotics use case, the wireless communication between the physical and virtual spaces should maintain deterministic

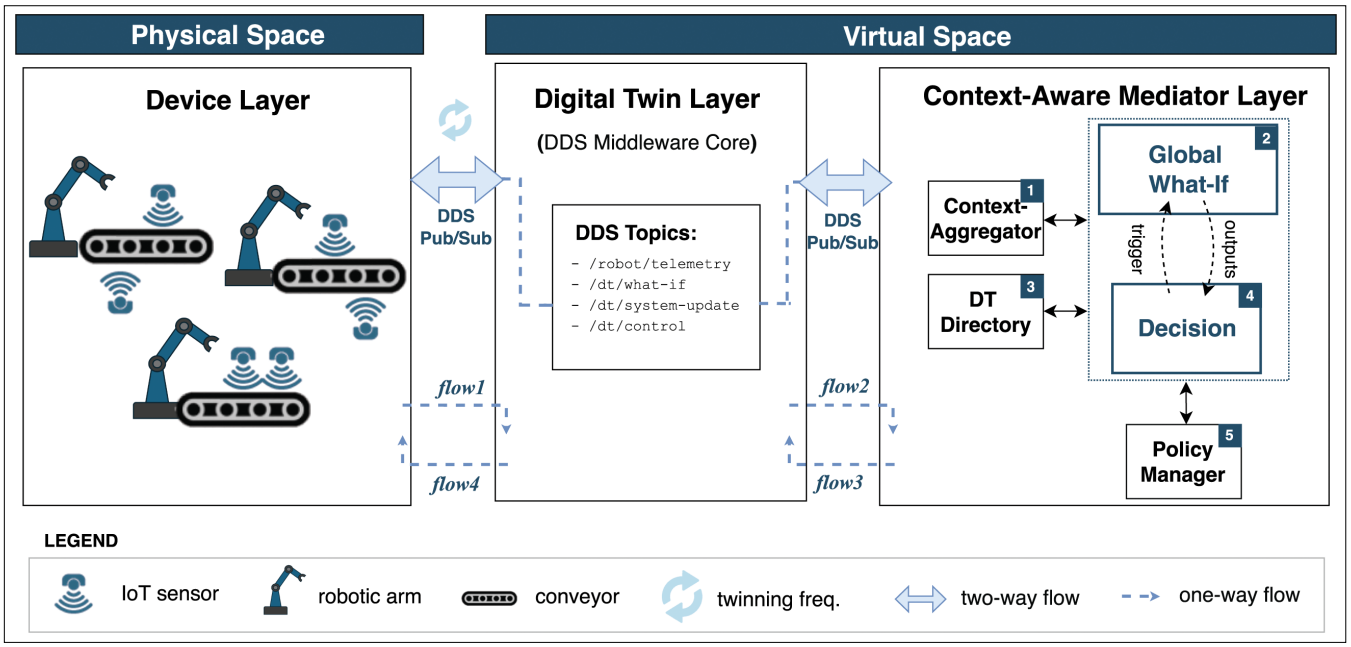

FIGURE 1. The proposed DT-based robotic applications management framework.

latency and throughput. Supporting this, an empirical evaluation of DDS-based robotic systems notes the average per-robot data rates of 0.1–0.3 Mbps for 10–50 Hz telemetry updates [14]. Based on this, in our study, each robot produces telemetry streams including position, speed, and energy consumption values at 20Hz, corresponding to ~ 0.25 Mbps per subscriber when DDS protocol overhead is considered. In this circumstance, the total throughput scales linearly with the number of robots deployed within the Device Layer. Furthermore, to guarantee responsiveness, our framework applies two latency thresholds via the twinning frequency. In this regard, the twinning frequency will be set to 50Hz to keep the control feedback latency around 20ms for real-time actuation. Also, it will be set to 20Hz to keep the synchronisation latency around 50 ms.

Context-Aware Mediator Layer

This layer acts as the central coordination unit of the proposed DT architecture. It interfaces with the DT layer to collect information and consists of five core components, including the *Context Aggregator, Global What-If Engine, DT Directory, Decision Unit, and Policy Manager* modules. We explain each of these modules below.

1) Context Aggregator: It is responsible for collecting and preprocessing telemetry and behavioural data streams from all active DTs within the 6G robotic system. It handles heterogeneous data from diverse robotic and IoT sources, including sensor readings, actuation logs, status updates, and network metrics.

2) Global What-If Engine: This engine constructs dependency graphs of the entire 6G robotic network to provide context-aware knowledge on the mediator side. As it operates system-wide by considering all DTs, it makes the DT layer lighter without requiring each DT to run individual What-If scenarios. This engine constructs four types of dependency graphs as given below:

- **Temporal Dependency Graph (TDG):** This graph is a directed graph structure that models the time-based execution and timing constraints between events. Each node represents an event, and edges define their temporal relationships, such as sequence, delay, or deadline constraints. This graph type addresses the question *"What must happen when?"*. For instance, the time between a sensor reading and the robotic arm's action is traced using the edge value defined between these event nodes.
- **Data Dependency Graph (DDG):** This graph is a directed graph that represents how data values are generated, used, and propagated across different components and tasks within the 6G robotic network. In this graph, the nodes correspond to sensor readings, robotic arm positions, and actuation commands. On the other hand, the edges indicate data dependencies among them via the relations. The main target of this graph is to address *"What data is used or produced where?"*.
- **Control Dependency Graph (CDG):** This graph is the key to understanding which actions are triggered by which conditions. Therefore, it traces the flow of decisions and evaluates alternative execution paths under various inputs. This graph type addresses *"What decision causes what action?."* In this graph, the nodes represent the conditions and actions which are likely to occur within the 6G robotic scenario. Therefore, with CDGs, the impact of control nodes can be traced.
- **Performance Dependency Graph (PDG):** This graph is a directed graph that models how the performance characteristics of the 6G robotic scenario are affected by the interactions between system components and tasks. This graph type addresses *"What performance variation causes what impact?"*. For instance, to address URLLC services in 6G, the latency metric in the communication of components can be considered as one of the KPI.

As given in Algorithm 1, our proposed global what-if first initialises the dependency graphs, TDG, DDG, CDG and PDG from DT Directory metadata (line 1). During runtime, each incoming DDS message triggers an incremental update of the affected relations and the corresponding graph attributes (lines 3-9). This ensures that the dependency views remain consistent with the latest system state. Afterwards, upon a what-if request on `/dt/what-if` (lines 11-12), the engine iterates over candidate actions (lines 13-19) and extracts the impacted subgraphs from

Require: DT Directory metadata, DDS streams, what-if request
(candidate action set, constraints)
Ensure: Selected action and predicted KPIs
1 **Initialisation:** Construct TDG, DDG, CDG, and PDG
2 **Runtime update:**
3 **for all** DDS messages **do**
4 Extract affected entities and relations
5 Update TDG timing attributes using timestamps and deadline fields.
6 Update DDG links for produced and consumed data.
7 Update CDG triggers if control commands change.
8 Update PDG KPI states.
9 **end for**
10 **What-if query:**
11 Receive request on `/dt/what-if` and parse candidate action set and constraints
12 *Selected action* ← REJECT
13 **for all** candidate actions **do**
14 Extract affected subgraphs from TDG, DDG, CDG, PDG
15 **if** all subgraphs are consistent **then**
16 Predict KPIs/actions using the selected model
17 Update KPIs/actions depending on the model output
18 **end if**
19 **end for**
20 Publish predicted KPIs/actions on `/dt/control`.

Algorithm 1. Global What-If Engine.

dependency graphs by executing the selected model only when the resulting subgraphs satisfy data, timing, control, and performance consistency (line 15). Finally, the predicted outcome (KPIs/actions) is published back to the robotic system via `/dt/control` to enable closed-loop actuation (line 20).

3) DT Directory: The what-if engine maintains a centralised graph repository optimised for high-throughput access and graph-specific operations. For example, the system can query TDG to find all downstream tasks for a certain time period, in which the control actions for a particular robot are identified simultaneously via the CDG.

4) Decision Unit: This unit utilises six types of learning algorithms within the AutoML and selects the most successful one to be used in the 6G robotics scenario.

- *Model Training:* In this phase, multiple candidate prediction models are trained offline by using a bounded random search with 10 candidate configurations per model with early stopping. The following algorithms are utilised:
  - **Regression Models:** Our main target for using these is to predict the system KPIs, such as latency, especially when utilising PDGs.
  - **XGBoost:** We consider this algorithm while predicting the performance metrics through the hybrid use of dependency graphs within the What-If module.
  - **Hidden Markov Models (HMM):** We use this to model event transitions and predict the next state of the end-to-end robotic DT system, particularly when working with TDGs.
  - **Decision Tree:** We utilise this to learn the causal relationships between control inputs and corresponding robotic actions.
  - **Temporal Graph Neural Network (TGNN):** We utilise this to capture how information flows and decisions evolve across the robotic system based on both spatial and temporal dependencies.
  - **Reinforcement Learning (RL):** We utilise this to learn adaptive behaviour policies to maximise scenario objectives, such as task success rate, latency, and accurate control decisions.
- *Model Selection:* The AutoML selects the best-performing model depending on the minimum prediction error while ensuring that the model's inference latency remains below a predefined threshold. For regression, we configure Ridge (*regularisation*), XGBoost (*max_depth, n_estimators, learning_rate*), TGNN (*hidden_dimension, learning_rate, dropout*), and Q-learning (*learning_rate, discount_factor, exploration_rate*). For classification, we configure HMM (*numb_of_states*) and Decision Tree (*max_depth, min_samples_leaf*).
- *Control Actions:* The Decision Unit translates model outputs into concrete actions via rule-based actuation logic at the mediator. For this, the predicted KPI, *total_delay,* is compared against a service deadline to choose a feasible control option such as reducing twinning frequency or postponing a non-critical update before issuing actuation. In addition, the predicted task type determines the corresponding command template, which is published to the robot via the `/dt/control` topic for execution and closed-loop feedback.

5) Policy Manager: This module serves as a constraint-filtering layer on top of the *Decision Engine* and interacts with *Global What-if* and the current DT states. In the context of 6G-enabled robotic systems, decision outputs must consider predefined situations, including energy constraints, emergency override rules, and latency bounds.

Performance Evaluation

In this section, we investigate the performance of our proposed robotic DT framework in terms of communication and computational cost, under changing twinning frequencies.

Dataset Description: We use an open-source industrial robot control system dataset [15]. We construct the input feature vector using: {*robot_id, task_duration, energy_consumption, data_size, position_coordinates, sensor_id, sensor_type, sensor_reading, network_latency, feedback_delay, slice_bandwidth, data_transfer_time*}. In AutoML experiments, the prediction target is *total_delay*, which captures the end-to-end delay observed for the corresponding control instance, while the classification target is *task_type*. The realisation with respect to 6G operation is provided by varying twinning frequency, the number of concurrent twins, DDS modes, and stress test network conditions. Alltogether, these capture the timeliness and scalability constraints expected in dense robotic deployments.

Experimental Setup: We use AnyLogic© for the end-to-end execution of the proposed DT-based robotic workflow. We integrate a DDS middleware into the AnyLogic setup and implement the proposed topics for message exchange. In our setting, we represent each robot and its associated DT instance as an active agent that periodically generates telemetry messages according to the configured twinning frequency (20Hz or 50Hz). We use *NetworkX* and *Alpyne* Python libraries to create dependency graphs and run the ML models exported from AnyLogic, respectively. The simulation parameters are given in Table 1. In our experiments, we consider the following two scenarios:

Parameters	Values
Number of twins	[10, 530]
Twinning frequency (Hz)	{20, 50}
DDS QoS Config.	{Reliable, BestEffort}
Number of relations/ graph	25
Payload size/ update (bytes)	300
Batch size	{128, 256}
Confidence interval	95%

TABLE 1. Simulation Parameters.

	end-to-end delay (ms)		CPU utilisation (%)		RAM usage (%)		packet loss (%)	
number of twins	Scenario-1	Scenario-2	Scenario-1	Scenario-2	Scenario-1	Scenario-2	Scenario-1	Scenario-2
10	54	57	9.8	9.8	11.4	11.5	-	0
30	72	79	21.2	21.9	27.6	27.6	-	0.2
50	96	110	28.9	29.8	35.8	36.8	-	7.9
100	142	165	41.6	42.6	49.3	50.5	-	17.8
250	208	230	64.8	64.9	70.9	72.1	-	22.4
400	264	317	82.5	82.9	88.6	89.5	-	25.1
530	305	355	91.7	91.7	98.8	98.9	-	33.6

TABLE 2. System Responsiveness and Computational Overhead.

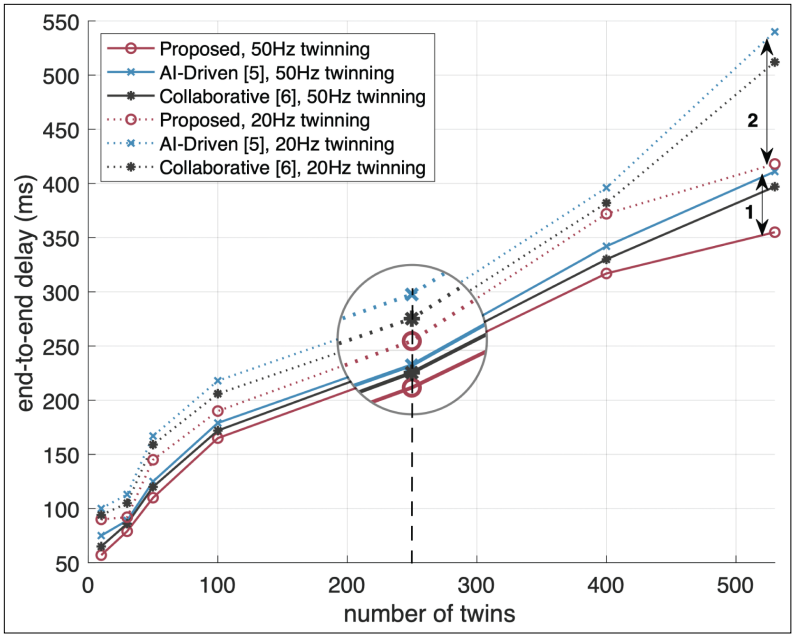

FIGURE 2. End-to-end delay comparison for proposed DT and the baseline DT approaches.

- *Scenario-1:* This is the base scenario where we assume an uncongested network to isolate the DT-layer processing and DDS QoS effects while observing the scalability.
- *Scenario-2:* In this, we add bursty background traffic that shares the same DT communication resource, thereby causing queueing on the DDS message path.

System Responsiveness and Computational Overhead Under Increasing Number of Twins: In the first set of our experiments, we measure the system responsiveness and computational overhead of the proposed framework with the utilisation of multiple twins. For this, we measure the end-to-end delay with a gradually increasing number of twins. Here, end-to-end delay consists of synchronisation latency and DDS message-queuing delay from DT Layer and querying and decision delay from Context-Aware Mediator Layer. At this point, real-timeliness refers to maintaining bounded DT synchronisation latency via twinning frequency. According to simulation results, the delay shows a gradual rise from 54 ms to 305 ms under a high-density twin for *Scenario-1*. In *Scenario-2*, delay values are higher when compared to the base scenario. This is due to the queueing delay caused by bursty traffic in the DDS message path. In addition, we record the packet loss ratio as the number of twins increases. As given in Table 2, when a larger number of twins participate in message exchange, the effects of background traffic begin to appear in *Scenario-2* results. While there is no packet loss for the 10-twin case, it reaches 33.6% levels when 530 twins jointly participate in the DDS network. Moreover, despite the growth in end-to-end delay, the DT layer maintains stable synchronisation with the twinning frequency of 50Hz, meaning ~ 20 ms synchronisation latency. As we set the twinning frequency either to switch 20Hz or 50Hz, we say that the proposed DT framework is capable of real-time operation by forcing the twinning frequency to provide stable and fast synchronisation. Following this, we observe the RAM usage and CPU utilisation metrics for both scenarios. As given in Table 2, as the number of twins grows, the RAM and CPU load increase almost linearly for both scenarios. Based on these, we note that RAM usage presents a computational boundary (reaching to 98.9%) by allowing the system to utilise 530 twins concurrently. Here, we note that 25 relations for each graph refers to the size of the active dependency subgraph maintained and queried per decision cycle, rather than a full system-wide knowledge graph. As our framework operates on incrementally updated subgraphs, for larger robotic systems, it can be extended via subgraph extraction by keeping query overhead bounded. Furthermore, the framework can be extended to large-scale deployments via distributed mediation. In this, dependency graphs can be stored and queried using distributed graph backends. This perspective is an important direction for future work.

Furthermore, we compare end-to-end delay results of our proposed DT with two DT approaches: (i) AI-driven control [5], and (ii) Collaborative and per-twin what-if-based control [6], by considering *Scenario-2*. In each approach, twins generate DDS updates at two twinning frequency values, 20Hz and 50Hz, with a fixed payload size. Therefore, in the higher twinning frequency case, the total load on the network would increase when the number of twins increases. According to recorded results, when the number of twins participating in the DDS increases, the AI-driven and collaborative

methods are more affected for both 20Hz and 50Hz cases. In particular, up to 250 concurrent twins, all approaches perform in a similar manner, resulting in a gradual increase in end-to-end delay. However, as given in Fig. 2, especially after 400 concurrent twins, AI-driven and collaborative approaches start to show a sharper increase when compared to the proposed DT. At the most dense scenario (530 twins), the proposed DT results in approximately 11.9%, and 20.3% lower end-to-end delay for 20Hz, and 50Hz cases, respectively (these are shown in the figure with black arrows and labelled as 1 and 2.) The main reason for this behaviour is that the AI-driven approach does not rely on a context-aware data source for running the decision algorithms. Similarly, the collaborative approach lacks context-awareness and performs per-twin what-if execution. In this, each DT instance executes scenario evaluation locally, and the mediator aggregates outputs. In contrast, the proposed DT approach both utilises context-awareness and the global what-if centralises scenario execution (Alg. 1, lines 10-19). As a result, this restricts execution to the affected dependency subgraph. This restriction in update propagation is the main reason why the delay gap becomes more pronounced at high twin densities.

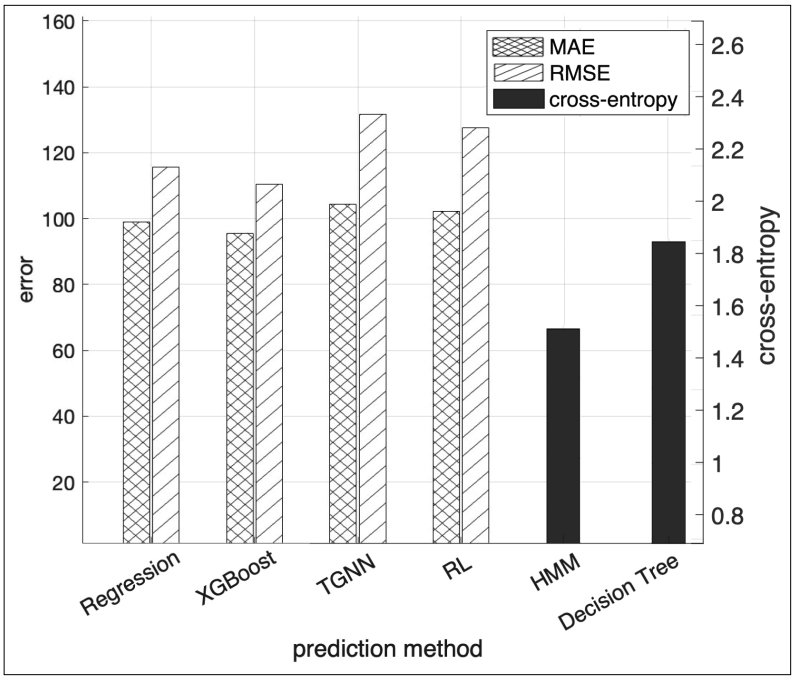

FIGURE 3. Prediction success of models for regression and classification tasks.

Performance of Learning Algorithms Within AutoML-Based Decision Unit: In the last set of our experiments, we evaluate the performance of the AutoML-based Decision Unit. We start by partitioning the data into two: 80% for training and 20% for testing. Afterwards, we perform regression and classification tasks by using the four types of dependency graphs. For the regression task, we configure Ridge Regression (regularisation strength set to 3), XGBoost (maximum depth, number of estimators, and learning rate are set to 4, 50, 0.8, respectively), TGNN (hidden dimension, learning rate and dropout are set to 2, 0.8 and 0.2, respectively), and Q-Learning (learning rate, discount factor, and exploration rate are set to 0.8, 0.9, 0.95, respectively) algorithms to identify candidate KPIs over PDGs. For classification, we configure HMM (with a number of states as 3), and Decision Tree (with maximum depth, and minimum number of samples in leaf set to 4, and 3, respectively). Furthermore, we set a persample inference latency threshold of 10 ms and limit the AutoML search to 10 candidate configurations per model, utilising a random search with early stopping. Afterwards, we select the best feasible models based on validation error under this latency constraint. We evaluate regression performance using MAE and RMSE. For classification performance, we measure cross-entropy. As shown in Fig. 3, in the regression task where we request the prediction total delay, XGBoost results in the most successive values. On the contrary, TGNN and Q-Learning perform worse than Ridge Regression and XGBoost. Due to the limited dataset size, complex models tend to overfit rather than generalise, which degrades their KPI prediction performance. Here, TGNN underperformance is due to limited training data; therefore, the Decision Unit selects lighter models when they provide better generalisation under the latency threshold. To mitigate this, future works can incorporate more diverse traces and constraint-aware synthetic augmentation, such as preserving timing-consistency among delay components to improve the generalisation of our framework. Besides that, we observe the success of classification tasks in which the actions of the robots are determined. According to our simulation results, the HMM performs better than the Decision Tree model at predicting whether the robot will perform pick-and-place, welding, or assembly actions. As a result, XGBoost and HMM can be used as the main prediction algorithms of the Context-Aware Mediator Layer.

Conclusion

In this paper, we propose a robotic DT framework consisting of a DDS-based DT Layer and a Context-Aware Mediator Layer. These layers consist of two main components: (i) Global What-If Engine, and (ii) AutoML-based Decision Unit. In this framework, we model four distinct types of interdependencies based on temporal, data, control, and performance information. Furthermore, we use AutoML to select the most appropriate model based on prediction errors and scenario constraints. The experimental results demonstrate that the proposed framework is capable of providing real-time DT synchronisation even under high-density twin scenarios without overloading the system, thanks to the adaptable twinning frequency and context-aware Global What-If. Moreover, the proposed DT framework achieves effective learning-based decision-making in 6G robotic applications.

In future work, we aim to extend the framework by integrating real-time edge-cloud coordination to model end-to-end network effects in distributed robotic applications. In addition, we will extend our evaluations based on a robotic emulation environment by considering larger graphs and interference in a multi-robot scenario.

References

[1] C.-X. Wang et al., "On the road to 6G: Visions, requirements, key technologies, and testbeds," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 905–974, 2nd Quart., 2023.

[2] K. Duran and B. Canberk, "Digital twin enriched green topology discovery for next generation core networks," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 4, pp. 1946–1956, Dec. 2023.

[3] M. Picone, M. Mamei, and F. Zambonelli, "A flexible and modular architecture for edge digital twin: Implementation and evaluation," *ACM Trans. Internet Things*, vol. 4, no. 1, pp. 1–32, Feb. 2023.

[4] R. Liu et al., "Internet of digital twin: Framework, applications, and enabling technologies," *IEEE Commun. Surveys Tuts.*, vol. 28, pp. 3870–3910, 2026.

[5] T. M. Ho, K.-K. Nguyen, and M. Cheriet, "AI-powered digital twins for robotic control in 5G-enabled industrial automation," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 10, pp. 3347–3361, Oct. 2025.

[6] K. Duran and B. Canberk, "Digital twin-based collaborative management for energy-aware 6G IoT systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2025, pp. 1–6.

[7] Q. Zeng et al., "Knowledge-based ultra-low-latency semantic communications for robotic edge intelligence," *IEEE Trans. Commun.*, vol. 73, no. 7, pp. 4925–4940, Jul. 2025.

[8] K. Duran et al., "GenTwin: Generative AI-powered digital twinning for adaptive management in IoT networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 11, no. 2, pp. 1053–1063, Apr. 2025.

[9] C. Krieger et al., "Integration of scaled real-world testbeds with digital twins for future AI-enabled 6G networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2023, pp. 2037–2042.

[10] Q. Hou et al., "Automatic AI model selection for wireless systems: Online learning via digital twinning," *IEEE Trans. Wireless Commun.*, vol. 24, no. 4, pp. 3088–3102, Apr. 2025.

[11] *Automation Systems and Integration—Digital Twin Framework for Manufacturing—Part 1: Overview and General Principles*, document I. 23247-1, 2020.

[12] *Architectural Framework for Machine Learning in Future Networks Including IMT-2020*, document ITU-T Y.3172, ITU-T, 2020. [Online]. Available: https://www.itu.int/rec/T-REC-Y.3172-201906-I

[13] OMG. (2015). *Data Distribution Service (DDS) for Real-Time Systems*. [Online]. Available: https://www.omg.org/spec/DDS/1.4

[14] J.-B. Castillo-Sánchez, E. González-Parada, and J.-M. Cano-García, "Swarm robot communications in ROS 2: An experimental study," *IEEE Access*, vol. 12, pp. 142930–142943, 2024.

[15] (2025). *Industrial Robot Control System Dataset*. Accessed: Aug. 27, 2025. [Online]. Available: https://www.kaggle.com/datasets/ziya07/industrial-robot-control-system-dataset

Biographies

Kubra Duran (Graduate Student Member, IEEE) (kubra.duran@napier.ac.uk) received the B.Sc. degree in electrical and electronics engineering from Eskisehir Osmangazi University in 2016 and the M.Sc. degree in computer engineering from Istanbul Technical University in 2021. She is currently pursuing the Ph.D. degree with the School of Computing, Engineering and the Built Environment, Edinburgh Napier University. She has five years of experience in the industry, performing as a Research and Development Specialist and a Software Engineering roles. Her research interests are digital twins, the Internet of Things, and AI-enabled communication and networking.

Mehmet Ozdem (Member, IEEE) (mehmet.ozdem@turktelekom.com.tr) received the Ph.D. degree in electrical and electronic engineering from Gazi University, Türkiye. He is the Director of the Innovation and the Vice President of the Product and Service Development at Turk Telekom, Türkiye, and International Telecommunication Union SG12 and Quality of Service Development Group Organisations.

Kaushik Chowdhury (Fellow, IEEE) (kaushik @utexas.edu) received the M.S. degree from the University of Cincinnati, Cincinnati, OH, USA, in 2006, and the Ph.D. degree from Georgia Institute of Technology, Atlanta, GA, USA, in 2009. He is a Chandra Family Endowed Distinguished Professor in electrical and computer engineering at The University of Texas at Austin. His current research interests involve systems aspects of machine learning for agile spectrum sensing/access, autonomous autonomous systems, programmable and open cellular networks, and largescale experimental deployment of emerging wireless technologies. He was a recipient of the Presidential Early Career Award for Scientists and Engineers (PECASE), in 2017, ONR Director of Research Early Career Award, in 2016, and the NSF CAREER Award, in 2015.

Berk Canberk (Senior Member, IEEE) (B.Canberk@napier.ac.uk) received the B.Sc. degree in electrical engineering from Istanbul Technical University (ITU), the M.Sc. degree in telecommunications engineering from Chalmers University of Technology, Sweden, and the Ph.D. degree in computer science from ITU. He is a Full Professor at Edinburgh Napier University, U.K. Prior to his current position, he was a Full Professor at ITU, where he served as an Associate Professor from 2016 to 2021 and a Full Professor from 2021 to 2022 at the Department of Computer Engineering. He was previously a Post-Doctoral Researcher at Georgia Institute of Technology, USA. As a Leading Expert of Digital Twin Technologies, he specializes in real-time synchronization, AI-driven orchestration, and semantic communication frameworks. He leads the Broadband Communication Research Group (www.bcrg.uk) with five Ph.D. students, one postdoctoral researcher, two research associates, one visiting professor, and one visiting M.Sc. student. He has previously supervised seven Ph.D. and 21 M.Sc. graduates. His publication record includes more than 98 journal articles, 103 conference papers, and 23 patents, of which 46 journal articles, 43 conference papers, and 18 patents are specifically focused on Digital Twins. Since 2015, he has secured over £3.6M in research funding as a Principal Investigator. His research focuses on AI-powered digital twins, IoT communication, and smart wireless networks, spanning technology readiness levels (TRL) 1–6. He serves as a Lead Editor of the IEEE ComSoc Best Readings in Digital Twins and as an Area Editor for IEEE Communications Surveys and Tutorials. He has served as an Associate Editor for IEEE Transactions on Vehicular Technology, *Elsevier Computer Networks*, and IEEE Communications Letters, and has taken on key roles as the General Chair, the TPC Chair, and an Organizing Committee Member at several IEEE conferences.