Bridging IoT and the Metaverse: Policies for the Synchronization of Digital Twins

Claudio Marche
University of Cagliari, DIEE
Cagliari, Italy
claudio.marche@unica.it

Michele Nitti
University of Cagliari, DIEE
Cagliari, Italy
michele.nitti@unica.it

Luigi Atzori
University of Cagliari, DIEE
Cagliari, Italy
l.atzori@unica.it

Simone Porcu
University of Cagliari, DIEE
Cagliari, Italy
simone.porcu@unica.it

***Abstract*—The convergence of the Internet of Things (IoT) and the metaverse creates systems where a Digital Twin (DT) sits between physical and virtual worlds. Keeping them in sync is hard when actions arrive at the same time with different priorities and delays. We present a DT template that supports two-way, asynchronous interactions and a stack of three policies: confirm matching intents, resolve conflicts with context (domain weights and timeliness), and rollback on invariant violations. We validate the approach in a smart building with DTs representing doors, windows, and lights operating within policy domains that include security, comfort, and energy efficiency. Without policies, conflicting windows already reach 33-56% with only two users for different request rates. With the policy stack, the share of conflicts resolved by policy increases with the reliability gap and can approach all the conflicts, reducing rollbacks. Finally, under asymmetric placement, by simulating the DT at different points in the network (e.g., edge close to physical devices and cloud close to the metaverse), rollbacks shift to the slower side but leave the overall resolution essentially unchanged.**

***Index Terms*—Digital Twin, IoT, Metaverse, Synchronization Policies.**

I. Introduction

The increasing integration of Digital Twins (DTs) into real-world infrastructures is reshaping how we interact with, monitor, and influence physical environments. Born as digital representations of physical entities, DTs have rapidly evolved into autonomous socio-technical agents, capable of sensing, interpreting, and acting within the Internet of Things (IoT) landscape. Through continuous synchronization with their physical counterparts, DTs enable new forms of data-driven decision making, proactive maintenance, and adaptive control in domains ranging from smart cities to personalized healthcare [1].

At the same time, the Metaverse is emerging as a persistent, interconnected digital space that extends human activities, i.e. social interaction, commerce, entertainment, into immersive virtual environments [2]. Unlike traditional online platforms, the Metaverse is not merely a representation of digital content, but a living ecosystem where digital identities and entities evolve, communicate, and make decisions autonomously. In this vision, DTs play a central role, acting as bridges between physical and virtual dimensions, and enabling a seamless interplay between sensor data, user preferences, and social behavior [3].

Yet, while the presence of DTs in both physical and virtual worlds is becoming increasingly common, their concurrent exposure to both realities introduces new challenges. In particular, conflicts may arise when updates originate from multiple sources, e.g. the real world, the Metaverse, or internal DT logic, without a clear set of rules to govern their propagation. Unlike centralized systems with a single source of truth, DTs operating at the intersection of cyber-physical-social networks require flexible and context-aware policies to maintain coherence and responsiveness. However, current solutions are either domain-specific or rely on ad hoc synchronization mechanisms, lacking formal and reusable policy definitions to guide DT behavior across heterogeneous domains [4].

This paper addresses this gap by proposing a DT template designed to interface simultaneously in physical and Metaverse environments, supporting bidirectional interactions and state management across domains. Rather than privileging a dominant source, the proposed framework embraces synchronization strategies, enabling DTs to behave as social mediators. Through a template and a set of propagation rules, DTs can interpret incoming changes, decide when and how to apply updates, and handle conflicts.

The main contributions of this work are:

1) A DT template designed for coexistence across real and virtual worlds, enabling bidirectional and asynchronous interactions with both IoT systems and Metaverse platforms.
2) A set of synchronization policies, including confirmation mechanisms, conflict resolution, and rollback procedure, that preserve consistency under concurrent updates.
3) An experimental validation in a smart home scenario, showcasing how different policies impact consistency, responsiveness, and reliability across domains.

The rest of the paper is organized as follows: Section II reviews the background on DTs and the existing IoT–metaverse integrations. In Section III, we introduce the proposed DT template, detailing its structure and components, while Section IV formalizes the synchronization policies. Furthermore, Section V presents the use case scenario, and Section VI discusses the simulation setup and the performance results. Finally, Section VII concludes the paper.

II. Background

DTs extend traditional simulation beyond the design phase by keeping high-fidelity digital counterparts synchronized with physical assets at run time. This enables continuous monitoring, analysis, and control without disrupting operations [5]. As observed in [6], conventional modeling tools emphasize early-stage verification and optimization, whereas DTs leverage live data feeds to mirror and predict physical behavior in operational settings.

At system scale, DTs present heterogeneous devices as interoperable services while masking protocol and network diversity. Resilient and scalable operation in IoT and cyber–physical environments therefore depends on effective discovery, orchestration, and service management [7], [8]. Prior work explores semantic and architectural mechanisms that support these needs, including modular ontology-driven directories for dynamic querying across heterogeneous assets [9] and similarity-based models that compose compatible DT functionalities [10]. Surveys situate these capabilities within broader IoT and 6G ecosystems in which elasticity and reliability are first-order concerns [11].

Within metaverse research, DTs increasingly serve as the structural layer that aligns physical and immersive virtual environments. Early contributions established DTs as real-time mirrors that enable adaptive process control and continuous optimization [12], [13], while metaverse taxonomies clarified the requirements of persistent and interactive spaces [14]. Beyond industrial settings, a recent review maps fifteen DT applications inside metaverse scenarios such as simulation and training, emergency response, mobility, and healthcare, anchoring use cases across domains [15].

For the infrastructure layer, surveys detail how DTs interoperate with cloud rendering, extended reality interfaces, and IoT sensing to realize industrial metaverse stacks [16]. Complementary work discusses how Artificial Intelligence and extended reality close perception–control loops for DT-driven metaverse interactions, with implications for interfaces and autonomy [17]. More recent strands investigate large-scale synchronization and coordination under latency and placement constraints, for example architectures that combine AI and 6G features such as ultra-reliable low-latency communication and network slicing, and hierarchical IoT-assisted updates [18]–[20].

Despite these advances, the literature still lacks a reusable and domain-agnostic model for synchronizing DTs that operate concurrently in physical and virtual domains. Most works only emphasize network and data aspects such as latency optimization, edge or cloud placement, and hierarchical propagation [18]–[20]. Explicit policy rules that govern asynchronous multi-source interactions remain under-specified, in particular when to confirm semantically aligned updates, how to arbitrate conflicting intents under context, and when to roll back inconsistent commits. This gap motivates the template and policy stack introduced in this paper.

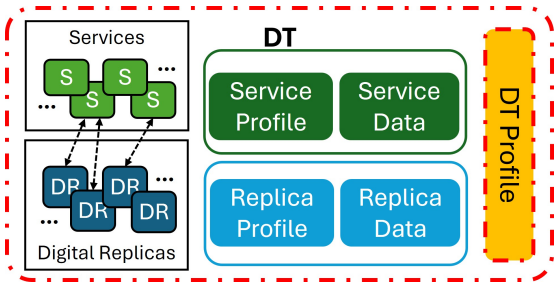

Fig. 1. Fundamental components of the DT.

III. Digital Twin: Structure and Template

As illustrated in Section II, DTs enable interaction between IoT and metaverse environments. Acting as intermediaries, DTs continuously replicate and analyze the state of physical devices while also interacting within virtual environments.

The proposed DT is modeled as a composition of interoperable components, each responsible for a specific aspect of representation and functionality. Specifically, a DT integrates Services and Digital Replicas (DRs) as primary components:

- A DR is the virtual representation of a physical entity. It includes a Replica Profile, describing identity, configuration, and capabilities, and Replica Data, maintaining real-time and historical data.
- A Service is a function associated with one or more replicas (e.g., monitoring, prediction, optimization). Each service has a Service Profile, describing operational parameters and purpose, and Service Data, containing execution results.

These components are managed by a DT Profile, which acts as the governing component. The DT Profile aggregates metadata and coordination rules, supporting management, synchronization policies, and interoperability across domains. A DT can host multiple replicas and services simultaneously, allowing different configurations depending on the application context. The overall structure is illustrated in Fig. 1.

To design a generic and interoperable template, we build upon the GOIoTP ontology introduced in [21], created to enhance interoperability among heterogeneous IoT platforms. GOIoTP defines core entities such as devices and services, as well as their relationships. Since the original ontology focuses on IoT interoperability, we extend it to capture metaverse requirements. In our proposal, the device concept is generalized into the DR, and the user entity is generalized into an ”owner”, representing the actor responsible for deploying or managing the DT. We introduce ”geometry”, ”data”, and ”platform” elements to describe spatial configuration, information produced by replicas and services, and the execution environment.

The DT acts as the central element, linking all components (Fig. 2). The owner represents the actor (individual or organization) responsible for control. The platform defines the environment, specifying where the DT and its components are deployed and how they communicate with external systems.

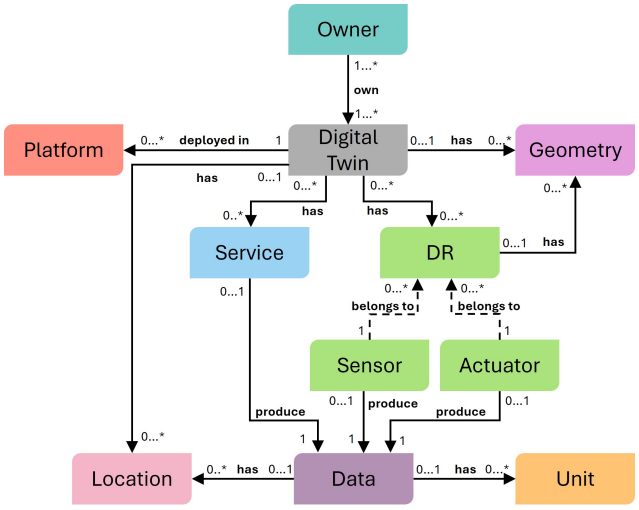

Fig. 2. Template structure of the DT.

The geometry describes the appearance and spatial configuration of the twin within the metaverse, while the location provides its geographical position in the physical domain. The DR captures the representation of physical devices; each DR may include sensors and actuators that produce or manipulate data. The data module aggregates measurements, processed results, and historical information, with unit defining measurement references. Finally, the service component encapsulates logic linked to one or more replicas, enabling monitoring, analysis, or optimization tasks across physical and virtual counterparts.

IV. Synchronization Policies

While the proposed DT template integrates the two worlds, this very interoperability exposes the system to conflicting updates and concurrent actions from multiple sources. To preserve consistency, the DT Profile enforces a set of synchronization policies that govern all communications between the real world and the metaverse. We formalize the behavior of these policies by modeling a DT instance DT_i as:

$$DT_i = \{S_i(t), A_i(t), P_i, \Omega_i\} \tag{1}$$

where each element captures dynamic aspects relevant to policy execution (as opposed to the structural template). Specifically, $S_i(t)$ is the system state at time t, $A_i(t)$ the set of actions applied to physical and virtual entities, P_i the set of active policies, and Ω_i the contextual parameters (e.g., domain, latency, confidence, user identity).

Each action $a \in A_i(t)$ carries an emission timestamp t_a and is considered valid only if its delay $\delta_t = t - t_a$ does not exceed the synchronization tolerance τ_{th}. Actions targeting the same DT and arriving within τ_{th} are treated as concurrent and thus subject to policy evaluation. A policy $p \in P_i$ is expressed as a mapping

$$p : (\Delta S_i, \Omega_i) \to a_i \tag{2}$$

that selects the action a_i to execute given the observed state variation ΔS_i and context Ω_i.

The policy stack is applied in order: (i) confirmation of semantically aligned actions, (ii) context-aware arbitration of conflicting actions, and (iii) rollback if an inconsistency is detected after execution.

1) Confirmation Policy: Let $\text{sem}(a)$ map an action to a normalized intention (for a door: *open*, *close*, *lock*). Two actions a_r and a_v (real vs. virtual) are executed jointly iff $\text{sem}(a_r) = \text{sem}(a_v)$:

$$\text{Execute}(a_r, a_v) = \begin{cases} 1, & \text{if } \text{sem}(a_r) = \text{sem}(a_v), \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Aligned actions are propagated to both worlds and the new state is acknowledged in each context. If $\text{sem}(a_r) \neq \text{sem}(a_v)$, arbitration is required.

2) Conflict Resolution Policy: When confirmation fails, we arbitrate using time-varying domain weights. Let $\mathcal{D} = \{d_1, \ldots, d_m\}$ be the active domains. Each domain d_k has a contextual weight $w_{d_k}(t, \Omega_i) \in [0, 1]$. If domains propose concurrent actions $\{a_{d_k}(t)\}$ for the same DT, we execute

$$a_i = \arg \max_{a_{d_k} \in A_i(t)} \left[w_{d_k}(t, \Omega_i) \cdot R(a_{d_k}) \right], \tag{4}$$

where $R(a_{d_k}(t)) \in [0, 1]$ quantifies action reliability (timeliness, consistency, source trust).

3) Rollback Policy: All actions that fail to reach consensus in the Conflict Resolution Policy are redirected to the rollback mechanism. Rollback is triggered only if the previous policy cannot select a winner or if a post-execution check reveals an inconsistency. When a newly issued action $a_{i,new}$ invalidates a previous action $a_{i,old}$, the DT triggers a recovery procedure to restore consistency. Formally, the rollback process reverts the DT to the last valid state:

$$\begin{aligned} &S_i(t) \leftarrow S_i(t - \Delta t^*), \\ &\Delta t^* = \min\{\Delta t \mid C(S_i(t - \Delta t)) = 1\}, \end{aligned} \tag{5}$$

where $C(S)$ is a consistency function returning 1 for valid states and 0 otherwise. Once the inconsistency is detected, the DT notifies both the real and virtual worlds that the corresponding action has been deemed invalid. Consequently, neither world finalizes the change, ensuring that no conflicting update is permanently applied.

V. Use Case Scenario

We evaluate the proposed DT template and synchronization policies in a smart building where several DTs coexist and interact across real and virtual domains. Each physical object is modeled as a DT governed by multiple, possibly overlapping, policy domains: Security (access control and anomaly detection), Comfort (environmental conditions), and Energy (power optimization). Representative DTs include doors, windows, and lights. Each DT primarily inherits the policy set most relevant to its function (e.g., Security for the door, Comfort for the window, Energy for the light) but can also be affected by cross-domain rules. For instance, the window DT normally operates under Comfort policies that regulate ventilation, yet

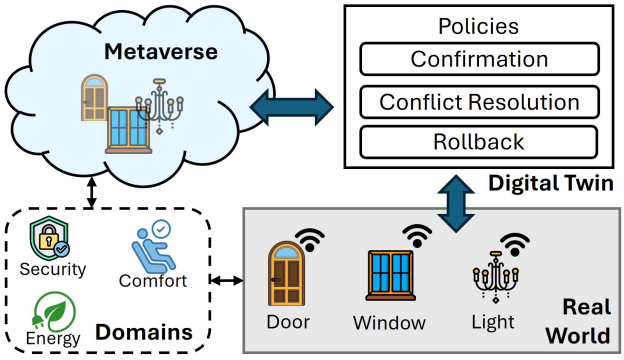

Fig. 3. Use case scenario.

if a forced opening is detected, the Security domain overrides local control and triggers an alert. Such dynamic prioritization allows policies to adapt according to real-time context and environmental state.

User interactions occur simultaneously in both the real world and the metaverse. A physical user may open a window manually, while another user performs the same action through the metaverse interface. Both actions generate concurrent events with contextual metadata (domain, timestamp, latency, confidence) that are processed by the DT profile. This manager evaluates whether updates should coexist, be serialized, or rolled back based on synchronization policies and contextual priority. The same principle applies across domains: a security alert on the door DT can temporarily suspend comfort-driven actions such as lighting or ventilation control, ensuring consistent behavior between physical and virtual spaces. The profile component, embedded in each DT, exchanges context and dependency information with peers, enabling distributed conflict resolution without central coordination. When the Security domain raises an alert, the Comfort and Energy domains dynamically adjust their configurations (e.g., reducing lighting or disabling ventilation) to preserve a coherent and situationally aware state across both worlds.

To operationalize the Conflict Resolution Policy in Eq. 4, we instantiate its two key components: the contextual weight $w_{d_k}(t, \Omega_i)$ and the reliability function $R(a_{d_k}, t)$. First, w_{d_k} models the priority of each domain based on the system's state. In the smart building, the context Ω_i determines a state $S_{\text{i}}(t)$ and

$$w_{d_k}(t, \Omega_i) = \text{Map}(d_k, S_i(t)), \tag{6}$$

where the mapping is given in Table I. As described, the system primarily operates in a Normal state (Comfort prioritized) but transitions to SecurityAlert upon anomaly detection.

Second, the reliability function $R(a_{d_k}, t)$ evaluates the trustworthiness of an action based on its source and timeliness:

$$R(a_{d_k}, t) = V(a_{d_k}, S_i(t)) \left[A\, e^{-\lambda\, \delta_t} \right], \tag{7}$$

where $\delta_t = t - t_{a_{d_k}}$. Furthermore, λ is anchored to the concurrency window τ_{th} by setting $e^{-\lambda \tau_{th}} = \alpha$, yielding $\lambda = \ln(1/\alpha)/\tau_{th}$ (specifically, $\tau_{th} = 0.4\,\text{s}$ and $\alpha = 0.2$ give $\lambda \approx 4.0\,\text{s}^{-1}$); and $V(a_{d_k}, S_i(t)) \in \{0, 1\}$ is a validation check.

TABLE I
CONTEXT–WEIGHT MAPPING FOR THE SMART BUILDING

$S_{\text{i}}(t)$	Description	w_{Sec}	w_{Comf}	w_{En}
Normal	Standard operations	0.3	**0.5**	0.2
SecurityAlert	Intrusion detected	**1.0**	0.1	0.1

Finally, we define a consistency function $C : \mathcal{S} \to \{0, 1\}$ that returns 1 if the tentative post-action state S' is defined under the action sets used in our simulations (door: *open/close/lock*; window: *open/close*; light: *on/off*) and does not contradict the current context $S_i(t)$; specifically, $C(S') = 0$ when an action–state pair is invalid (e.g., attempting to open a locked door) or when an action conflicts with a context where a different domain has priority (e.g., comfort overriding `SecurityAlert` effects), otherwise $C(S') = 1$.

VI. PERFORMANCE EVALUATION

A. Simulation Setup

In this simulation setup, the DT is instantiated with an autonomous entity, according to the template definition, that implements the three policies (confirmation, conflict resolution, rollback) in the profile component, which acts as the ingress point for all requests. The DT hosts three DRs representing door (Security), window (Comfort), and light (Energy). The DT Profile is the only entrypoint for actions coming from the two worlds, i.e., real and virtual; it timestamps arrivals, buffers them within the concurrency window τ_{th}, and applies the policies in order. Feasibility checks are enforced before commit, and any violation triggers a rollback. The reliability term R used in the conflict resolution (according to Eq. 4) is evaluated as defined in Eq. 7, using the latency actually observed for each received action at the DT.

Arrivals from the two worlds are modeled as Poisson processes with rates ν_r (real) and ν_v (virtual), i.e., the inter-arrival times are $\Delta t_r \sim \text{Exp}(\nu_r)$ and $\Delta t_v \sim \text{Exp}(\nu_v)$ in line with recent IoT traffic measurements [22]. For each incoming request, a domain is selected by assigning a probability p per world and per domain; e.g., for Security two parameters are configured, $p^{\text{real}}_{\text{Security}}$ and $p^{\text{virtual}}_{\text{Security}}$, and analogously for Comfort and Energy. Within the selected DR, the action label (e.g., *open/close/lock* for the door) is sampled from a set so as to generate both semantically aligned updates and conflicts within the concurrency window τ_{th}. Two channels are emulated to account for placement effects: (i) real-DT and (ii) DT-metaverse. Three configurations are exercised: symmetric case (same average delay on both segments), faster real-DT and DT-metaverse. These latency settings are varied independently of the arrival rates, so that effects and propagation effects can be differentiated, as will be illustrated in the next subsection.

All components are containerized: the DT Profile service and the three DRs run as separate Docker containers, together with supporting services (e.g., storage/messaging). The stack

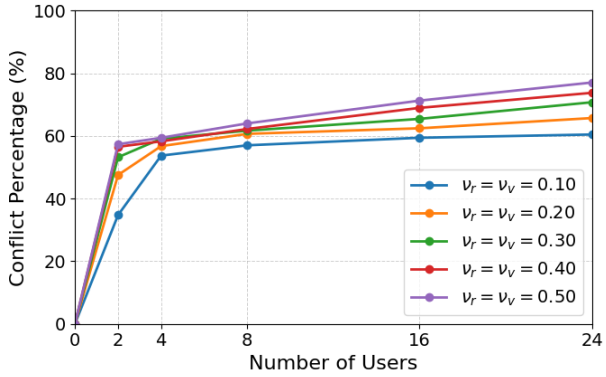

Fig. 4. Conflict rate for different numbers of users and for different request generation rates.

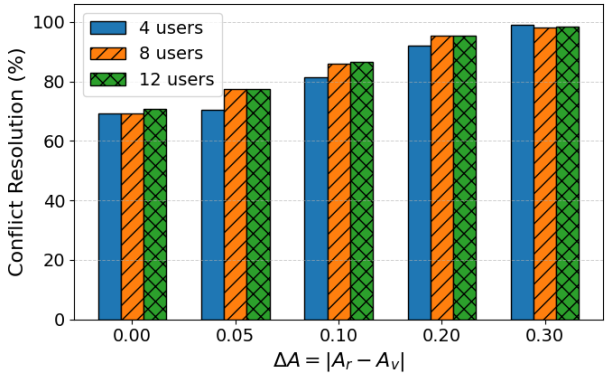

Fig. 5. Conflict resolution percentage over different reliability gaps ΔA.

is orchestrated via a single docker-compose specification and instantiated on a single host node. The two worlds (real and virtual) are emulated by issuing API calls to the DT Profile entrypoint, which receives and processes requests according to the arrival models and latency configurations described above.

B. Simulation Results

The first set of simulations analyzes how conflicts emerge when no synchronization policies are applied. Arrivals from the real world and the metaverse are modeled as independent Poisson processes with rates ν_r and ν_v, according to Section VI-A; a conflict is counted whenever, within the same concurrency window $\tau_{th} = 0.4$ s and on the same domain, updates originating from the two worlds carry different intentions. Figure 4 reports the fraction of conflict windows for each pair of generation rates; results are averaged over runs of duration 600 seconds. Specifically, the figure shows a monotonic trend along both axes. For a fixed rate, increasing the number of users raises the probability that actions from the two worlds fall within the same concurrency window, thereby increasing the percentage of conflicts. At the same time, for a fixed number of users, larger rates produce higher traffic, which in turn yields more concurrent updates and higher conflict rates. The growth is already evident at very small populations: with only 2 users, the conflict percentage spans $\approx 33\%$ to $\approx 56\%$ as rates increase from 0.1 to 0.4. These results underline the importance of synchronization policies: without them, frequent inconsistencies would arise in the DT, and the choice of which world's update to prioritize would remain ambiguous.

The second set of simulations is focused on the proposed policies since the full stack is enabled and conflicts are handled as follows. For each window containing divergent actions, Confirmation and Conflict Resolution are applied; if no action can be selected, or if a post-execution check fails, the Rollback policy restores the last consistent state. In this sense, Figure 5 reports the percentage of Conflict Resolution over conflicting windows, where the Rollback could be calculated as the complement to 100% of Conflict Resolution. Request arrivals from the two worlds are generated with $\nu_r = \nu_v = 0.2$, while the synchronization tolerance is fixed to $\tau_{th} = 0.4$ s, chosen as the largest latency recorded during setting the simulations, as the previous simulations, so that practically concurrent actions are evaluated within the same window. To evaluate the role of source reliability A, the reliability gap $\Delta A = |A_r - A_v|$ is introduced, where $A \in [0, 1]$ is the source reliability coefficient in Equation. 7, with r and v denoting the real and virtual worlds. The results, where the legend reports the total number of users, indicate a clear trend: the larger the reliability gap, the higher the fraction of conflicts solved without rollback. When $\Delta A = 0$, the two sources are equally credible and many conflicting windows fall within the indecision margin, yielding less resolutions and more rollbacks. As ΔA increases, one source dominates the reliability factor R and conflicts are resolved by policy rather than reverted, pushing resolutions toward 100% (and thus rollback toward 0%). Moreover, the resolution percentage is essentially invariant across user populations, since increasing the number of users raises the absolute number of conflicts but does not alter the success ratio, indicating that policy effectiveness is insensitive to the overall conflict number. The number of conflicting actions remains governed by traffic and τ_{th}; what changes with ΔA is the outcome of the policy, which increasingly prioritizes the more reliable source and consequently reduces rollbacks.

Finally, the impact of DT placement in the network is analyzed by creating asymmetric τ_r and τ_v and analyzing their gap $\Delta\tau = |\tau_r - \tau_v|$. While the policies are set using the $\tau_{th} = max(\tau_r, \tau_v)$, actions are simulated with 4 users per world and with $\nu_r = \nu_v = 0.4$. Source reliabilities are set to $A_r = A_v = 0.85$, and results are averaged over runs of 600 s with 15 repetitions, as the previous simulations. The Figure 6 reports the percentage of rollbacks, where the curve labeled "DT to metaverse" measures rollbacks when DT is placed closer to the metaverse and so to the cloud; the curve labeled "DT to real-world" measures rollbacks when DT is closer to the real world and so placed in the edge, with low latencies toward physical devices. As $\Delta\tau$ increases, a rise of

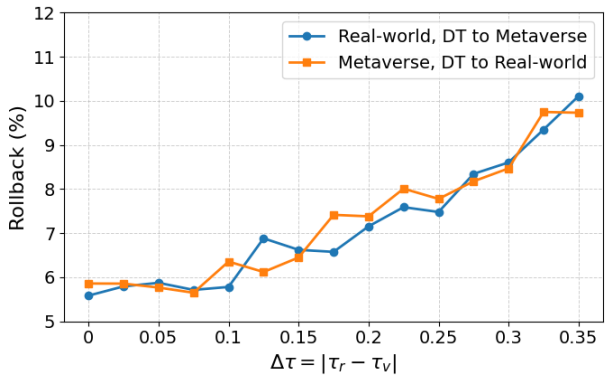

Fig. 6. Rollback percentage for two placement strategies of the DT.

both series is observed, with the DT placement favoring one world at the expense of the other. With $A_r = A_v$, the overall conflict-resolution capability does not fundamentally improve with placement; instead, rollback is redistributed between sources. Consequently, when reliabilities are comparable, DT placement should be driven by the application: a smaller tolerance should be assigned to the world that deserves priority (for example, the real world in security domains, or the metaverse in immersive interactions), accepting the corresponding degradation for the other world.

VII. Conclusions

We address the problem of keeping a DTs consistent when the physical world and the metaverse issue actions at the same time. We introduced a simple template and a three-step policy stack: confirmation of matching intents, conflict resolution using domain weights and a reliability term, and rollback when invariants are violated. In a smart-building case, without policies, the share of conflicting windows grows from about 35% to 57% even with only two users as request rates increase. With our policies enabled, the fraction of conflicts solved by policy increases with the reliability gap ΔA and can approach around 100% for large gaps. Changing DT placement (edge vs. cloud) does not change the total number of resolved conflicts when the two sources are equally credible; it mainly shifts rollbacks toward the slower side, so placement should favor the world that needs a faster response. Overall, these results turn the stack into a practical approach that can be reused in other domains. Future work will expand the evaluation at scale, learn domain weights and source reliabilities online, and study how simple explanations affect user trust.

Acknowledgment

This works has been supported by Fondazione di Sardegna, research project “DITTO - DIgital Twin as The bridge between meTaverse and internet Of things”, CUP F23C25000300007, call 2023.

References

[1] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, “Digital-twin-enabled 6g: Vision, architectural trends, and future directions,” *IEEE Communications Magazine*, vol. 60, no. 1, pp. 74–80, 2022.

[2] S. Mystakidis, “Metaverse,” *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.

[3] K. Li, Y. Cui, W. Li, T. Lv, X. Yuan, S. Li, W. Ni, M. Simsek, and F. Dressler, “When internet of things meets metaverse: Convergence of physical and cyber worlds,” *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4148–4173, 2022.

[4] R. Minerva, G. M. Lee, and N. Crespi, “Digital twin in the iot context: A survey on technical features, scenarios, and architectural models,” *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785–1824, 2020.

[5] V. Hassija, V. Chamola, R. De, S. Das, A. Chakrabarti, K. S. Sangwan, and A. Pandey, “A Survey on Digital Twins: Enabling Technologies, Use Cases, Application, Open Issues, and More,” *IEEE Journal of Selected Areas in Sensors*, vol. 2, pp. 84–107, 2025.

[6] M. Liu, S. Fang, H. Dong, and C. Xu, “Review of digital twin about concepts, technologies, and industrial applications,” *Journal of Manufacturing Systems*, vol. 58, pp. 346–361, 2021, digital Twin towards Smart Manufacturing and Industry 4.0.

[7] B. Pourghebleh, V. Hayyolalam, and A. A. Anvigh, “Service discovery in the Internet of Things: review of current trends and research challenges,” *Wireless Networks*, vol. 26, no. 7, pp. 5371–5391, 2020.

[8] H. Zorgati, R. B. Djemaa, and I. A. B. Amor, “Service discovery techniques in Internet of Things: a survey,” in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 1720–1725.

[9] M.-O. Pahl and S. Liebald, “A modular distributed IoT service discovery,” in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 448–454.

[10] C. Marche and M. Nitti, “Towards trustworthy digital twins collaboration in the internet of things: An overview of essential design guidelines,” *Computer Networks*, p. 111733, 2025.

[11] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, “Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions,” *Comm. Mag.*, vol. 60, no. 1, p. 74–80, Jan. 2022.

[12] A. El Saddik, “Digital twins: The convergence of multimedia technologies,” *IEEE MultiMedia*, vol. 25, no. 2, pp. 87–92, 2018.

[13] F. Tao and M. Zhang, “Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing,” *IEEE Access*, vol. 5, pp. 20 418–20 427, 2017.

[14] S.-M. Park and Y.-G. Kim, “A metaverse: Taxonomy, components, applications, and open challenges,” *IEEE Access*, vol. 10, pp. 4209–4251, 2022.

[15] S. Sai, P. Sharma, A. Gaur, and V. Chamola, “Pivotal role of digital twins in the metaverse: A review,” *Digital Communications and Networks*, 2024, in press.

[16] Z. Lyu and M. Fridenfalk, “Digital twins for building industrial metaverse,” *Journal of Advanced Research*, vol. 66, pp. 31–38, 2024.

[17] M. Tang, M. Nikolaenko, A. Alrefai, and A. Kumar, “Metaverse and digital twins in the age of ai and extended reality,” *Architecture*, vol. 5, no. 2, 2025. [Online]. Available: https://www.mdpi.com/2673-8945/5/2/36

[18] M. Aloqaily, O. Bouachir, F. Karray, I. Al Ridhawi, and A. E. Saddik, “Integrating digital twin and advanced intelligent technologies to realize the metaverse,” *IEEE Consumer Electronics Magazine*, vol. 12, no. 6, pp. 47–55, 2023.

[19] F. Tang, X. Chen, M. Zhao, and N. Kato, “The roadmap of communication and networking in 6g for the metaverse,” *IEEE Wireless Communications*, vol. 30, no. 4, pp. 72–81, 2023.

[20] Y. Han, D. Niyato, C. Leung, D. I. Kim, K. Zhu, S. Feng, X. Shen, and C. Miao, “A dynamic hierarchical framework for iot-assisted digital twin synchronization in the metaverse,” *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 268–284, 2023.

[21] G. Fortino, C. Savaglio, C. E. Palau, J. S. De Puga, M. Ganzha, M. Paprzycki, M. Montesinos, A. Liotta, and M. Llop, “Towards multi-layer interoperability of heterogeneous iot platforms: The inter-iot approach,” in *Integration, interconnection, and interoperability of IoT systems*. Springer, 2017, pp. 199–232.

[22] D. E. Ruiz-Guirola, O. L. López, and S. Montejo-Sánchez, “Modeling iot traffic patterns: Insights from a statistical analysis of an mtc dataset,” *arXiv preprint arXiv:2409.01932*, 2024.