Received 10 February 2024; revised 13 March 2024; accepted 30 March 2024. Date of publication 4 April 2024; date of current version 26 April 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3385659

ODTE: A Metric for Digital Twin Entanglement

PAOLO BELLAVISTA [1] (Senior Member, IEEE), NICOLA BICOCCHI [2], MATTIA FOGLI [3] (Member, IEEE), CARLO GIANNELLI [3] (Senior Member, IEEE), MARCO MAMEI [2], AND MARCO PICONE [2]

(Invited Paper)

[1]Department of Computer Science and Engineering (DISI), University of Bologna, 40126 Bologna, Italy

[2]Department of Engineering Sciences and Methods, University of Modena and Reggio Emilia, 42121 Reggio Emilia, Italy

[3]Department of Mathematics and Computer Science, University of Ferrara, 44122 Ferrara, Italy

CORRESPONDING AUTHOR: P. BELLAVISTA (e-mail: paolo.bellavista@unibo.it)

ABSTRACT Digital Twins (DTs) have recently emerged as a valuable approach for modeling, monitoring, and controlling physical objects in Industrial Internet of Things applications. Measuring the quality of entanglement between the digital and physical counterparts plays a crucial role in the adoption of DTs. In this context, entanglement denotes how well a DT mirrors its counterpart and the extent to which the behavior of the physical counterpart aligns with the commands issued by the DT. In this paper we propose a concise yet expressive and original metric for representing the quality of entanglement, namely Overall Digital Twin Entanglement (ODTE), based on two key factors: *timeliness* and *completeness*, i.e., the freshness of the collected data and the ratio between collected and total data, respectively. In addition, the paper describes how we have built our industrial testbed implemented on top of Kubernetes, where we show practical applications of the proposed ODTE metric by highlighting and discussing its benefits in realistic use cases.

INDEX TERMS Digital twins, entanglement, industrial IoT.

I. INTRODUCTION

UNDER the Industry 4.0 innovation umbrella, Digital Twins (DTs) are adopted in the realm of Industrial Internet of Things (IIoT) applications to make easier the dynamic management and optimization of industrial equipment [1], [2]. DTs have been defined as digital entities connected to specific Physical Twins (PTs) and serving as their counterparts to interact with them as if they were, within some limitations and constraints, the original objects. While most related work focuses on DTs for PT simulation, with the objective of design improvement (thus not requiring network communications between the two counterparts [3]), here we focus on DTs designed for mediating the interactions between applications and PTs. In this scenario, because of the intrinsic relationship between DTs and their corresponding PTs, some forms of synchronization are needed.

Recent work [4], [5], [6] characterize a small set of foundational properties for DTs, among which *reflection* and *entanglement* relate to the need of keeping the DT and the PT synchronized. In particular, *Reflection* defines the digitization process keeping the DT and PT states synchronized. Indeed, state changes faced by either the DT or the PT have to be timely propagated to the counterpart. As depicted in Figure 1, users who observe a DT for the sake of interacting with an underlying PT constantly face an unclearly quantified likelihood of: *(a)* querying values that do not reflect the actual state of the PT or *(b)* sending commands that are not timely propagated to the PT. In this context, the definition of *entanglement* as a measure that represents to which extent the DT-PT synchronization process fulfills the needs of a specific application acquires a prominent role. As such, we claim that it is of paramount importance to define metrics for measuring it.

The standard set of performance indicators for measuring the quality of a network link (required to keep DTs and PTs entangled) is usually indicated with the general term of QoS. Traditionally, QoS focuses on network characteristics, such as latency, jitter, and packet loss, all of which, even if relevant, are not able to capture application-specific nuances. To mention a very basic example, as shown in Figure 1,

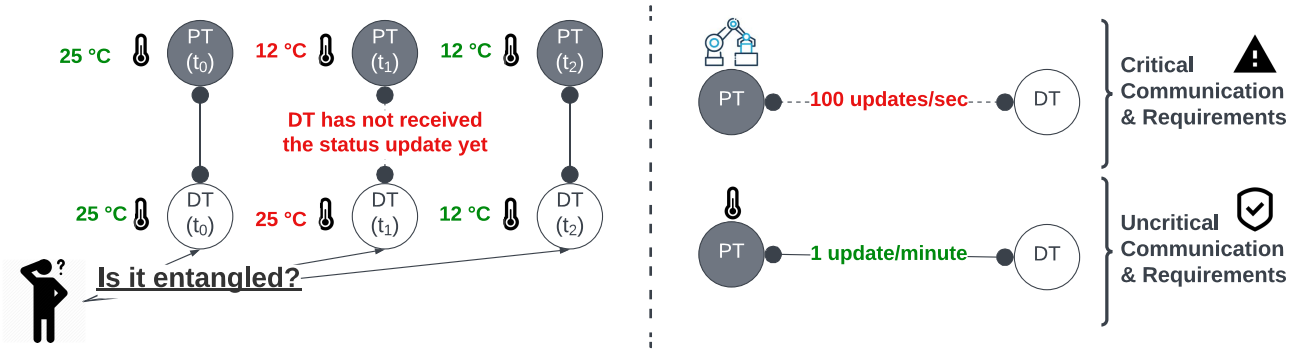


FIGURE 1. A high-level representation of the multiple aspects of the entanglement vision with respect to both the observation of external observers and applications (left side) together with the comparison between critical and uncritical communication and requirements (right side).

an increase of 1 sec in latency could be irrelevant for an application requiring sporadic updates (e.g., 1 update/minute) while dramatic for another one dealing with near real-time phenomena (e.g., 100 updates/second). To avoid such issues, alternative approaches have been proposed, e.g., Quality of Experience (QoE) [7], Quality of Information (QoI) [8], or Age of Information (AoI) [9], with the goal of evaluating the performance of applications instead of the network links they rely upon. However, existing QoE definitions focus on evaluating application quality from the lens of their users (e.g., if users are satisfied with the quality of a video-conferencing application) and are thus not well suited for unsupervised use cases (i.e., applications where human feedback is unavailable). An analysis of the existing literature led Fizza et al. [7] to conclude that measuring QoE of applications where human involvement or feedback is not readily available can be approximated by observing four key features about the collected data: *timeliness* (i.e., how fresh the collected data are for actually making decisions), *completeness* (i.e., the ratio of the amount of collected data to the total amount of required data), *accuracy* (i.e., the precision of the collected data), and *usefulness* (i.e., how useful the collected data are for the supported applications).

By concentrating our attention to manufacturing-related literature, the *Overall Equipment Effectiveness (OEE)* [10] is a widely adopted metric to evaluate production capabilities synthetically. It is defined as the product of three key factors: *availability* (i.e., the ratio of actually worked time to the total planned working time), *performance* (i.e., the ratio of the number of completed tasks to the number of tasks that could be hypothetically completed in the same period), and *quality* (i.e., the ratio of the number of tasks completed correctly to the total number of completed tasks). Despite OEE does not consider the *timeliness* aspect, which is paramount for time-dependent applications, it satisfies the need of providing a simple yet expressive metric to infer whether the production targets are actually fulfilled in terms of quantity and quality.

Building upon these considerations,

- we propose an original metric named *Overall Digital Twin Entanglement (ODTE)* capable of capturing in a concise yet expressive way the entanglement degree within a DT-PT dyad;
- we present motivations for the design of entanglement-aware DTs revolving around the following fundamental objectives: Decoupling Cyber & Physical Realms, Modularity, Flexibility, and Entanglement Awareness;
- we propose a DT architectural abstraction and its entanglement-based life cycle to model the behaviour of a DT-PT dyad;
- we outline four illustrative scenarios to emphasize the primary factors that may impact entanglement in industrial contexts;
- we present extensive experimental results that show how ODTE is responsive at quantifying the quality of entanglement under different IIoT scenarios of practical interest, thus demonstrating not only the feasibility but also the usefulness of the proposed solution.

Let us anticipate that the ODTE metric is centered around *timeliness* and *completeness* because *accuracy* and *usefulness* cannot be defined in a general manner, without an application-specific perspective. It provides, by definition, an easy-to-understand indicator (normalized between 0 and 1) that concisely represents whether the state changes happening in both the DT and the PT are effectively communicated to the counterpart; this simplicity allows anyone (or anything) to monitor the digitization process without any a-priori knowledge. This paper extends our previous work [11] as follows. First, it relevantly extends the previous mathematical formulation of the ODTE metric to fully cover the case in which the DT sends commands to the PT. Then, it proposes and gives the technical details of a blueprint architecture for entanglement-aware DTs. Moreover, it significantly surveys the related state-of-the-art by offering an extensive overview of the most recent related approaches in the literature. The remainder of the paper is organized as follows. Section II presents the related work by focusing on metrics for assessing objective/subjective qualities of networked applications. Section III outlines a reference scenario and theoretically defines the ODTE metric. Section IV discusses a blueprint DT architecture

for computing and exposing the ODTE metric to external services. Section V details the experiments that we deployed within a Kubernetes environment to quantitatively measure the ODTE metric for two real applications, in different networking contexts and with different benefits/drawbacks. Finally, Section VI concludes the paper.

II. RELATED WORK

The problem of defining and measuring effectiveness and quality of equipment and devices (like the DT in our case) has always been relevant to multiple areas and domains. The already mentioned OEE is a method to quantify the overall effectiveness of industrial equipment and, although it presents similarities with ODTE, it also exhibits important differences. OEE has been designed for industrial rather than IoT environments, focusing on craft production rate and quality rather than on information transfer. Secondly, it does not consider *timeliness*, i.e., how fresh the collected data are for actually making decisions, which is paramount for time-dependent scenarios. Nevertheless, the objective of defining QoE metrics to improve traditional QoS indicators in order to effectively represent application qualities is fueling active research. For instance, two recent surveys [12], [13] investigated the key factors that influence the perceived quality of applications in both IoT and wireless domains.

In addition, some attempts have been made to measure application QoE in a subjective manner. For instance, [14], [15], [16], [17] propose QoE metrics in different contexts. However, these approaches model QoE through human evaluations, without assessing how it relates to objective features such as QoS (which is necessary anytime human evaluation is not available or desirable).

On the contrary, most recent work focused on evaluating the application QoE starting from objective metrics. In [18], the authors proposed a regression model correlating QoE and QoS indicators (after extracting their principal components), thus practically showing that QoE can be derived from QoS parameters when human feedback is not available. In [19], the authors proposed a QoE model for a communication app by identifying 5 key factors that impact QoE (i.e., integrality, retainability, availability, usability, and instantaneousness). The final QoE value is a composition of these five measurements, normalized between 0 and 1. In [20], the authors proposed a model to determine the needed amount of resources to meet the desired QoE. Their results have been further refined in [21] by adopting a two-stage deep reinforcement learning scheme allocating the resources needed for maximizing QoE. In [22], the authors identified the key stages of an IoT application life cycle and proposed a different QoE metric for each stage based on QoS features (i.e., availability, accuracy, timeliness, throughput and packet loss). Although these proposals share several aspects with ours (e.g., QoE key factors are identified, QoE is put in relation with lower-level QoS metrics, QoE values are normalized on a fixed range), they are not fine tuned for the DT-PT entanglement, as more clearly detailed in the following parts of this article.

In the field of embedded control systems, [23] proposed a metric for quantifying operational coupling by considering the topology of connections, their multiplicity, replication, frequency, and the accuracy of relationship properties. Then, individual couplings are combined into an overall coupling, where domain-specific heuristics and technology constraints are used to determine the relative weights. Since it could be possible to exploit a DT as a control system, in principle a similar approach could be applied to DTs as well: however, it has shown to be complex to adopt in practical situations because it largely takes into account structural aspects of the considered software systems, which are not needed in our entanglement-oriented approach.

Age of information (AoI) has been introduced to characterize the knowledge freshness with which a system remotely observes a given process. Several papers have shown that considering AoI is significantly different from traditional metrics, such as delay and latency [24]. By taking advantage of the AoI definition, relatively recent efforts began to investigate status update strategies to improve information freshness in the IoT context [25], [26], [27]. While efficient status update policies have been proposed in the literature, they usually assumed that update packets generated by sensors were independent; on the contrary, in many IoT applications they are correlated and contribute to the same decision making process. More recent studies [28], [29], [30] began to investigate RL/DRL-based dynamic status update schemes to improve the information freshness for correlated systems. Finally, AoI has been recently applied to the DT domain for optimizing consistency and energy management precision. In particular, [31] proposed a novel information timeliness metric named ultra-low AoI (ULAoI), which, compared with AoI, further considers the occurrence of extreme events and higher-order statistical components of the AoI value.

III. MEASURING DIGITAL TWIN ENTANGLEMENT

As already mentioned, the interactions between DTs and PTs can unfold in two distinct ways: *(a)* a state change in the PT has to be communicated to the DT; *(b)* a command from an external user or application has to be propagated to the PT and a state change confirmation sent back to the DT.

Figure 2 represents the synchronization flow required to keep aligned the DT and PT states (denoted as S_i^{PT} and S_i^{DT}) when the PT changes state together with an example associated to the variation of a temperature sensor monitoring the environment. At the beginning S^{PT} and S^{DT} are aligned at version 1 (t_0). When a new physical event occurs, it triggers a variation of the physical state (changed to S_2^{PT}) and generates a state update for the DT. At this point, there is a misalignment between the two counterparts since the physical variation has not yet been reflected on the DT (t_1). Only when the DT receives the state update and computes its new state S_2^{DT} the two counterparts are finally

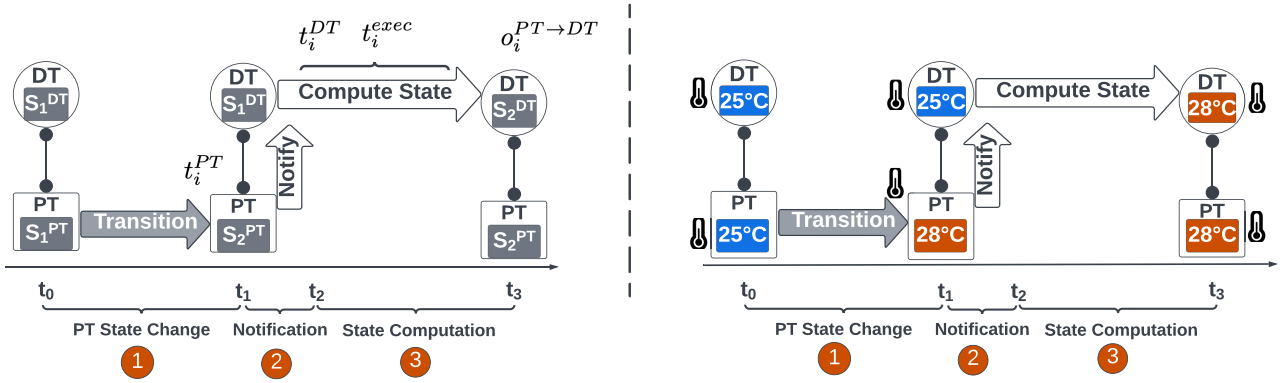


FIGURE 2. The entanglement process between PT and DT. An abstract representation with the associated references to the formulation (left) and a simplified realistic scenario with a device for temperature monitoring (right).

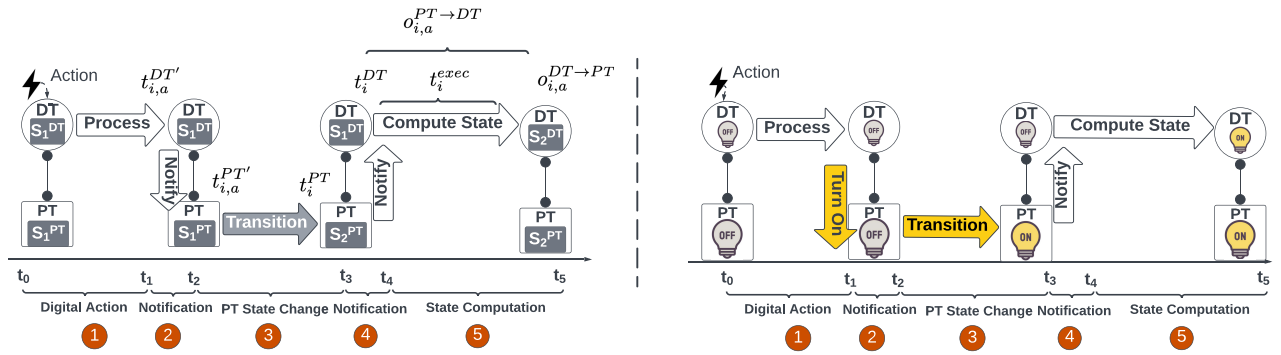


FIGURE 3. entanglement process between DT and PT. An abstract representation with the associated references to the formulation (left) and a simplified realistic scenario of an interaction with a light bulb actuator (right).

synchronized (t_2). In this first scenario, the entanglement reflects the time shift between the state of the physical entity and its digital counterpart.

Instead, Figure 3 represents the case where an action (e.g., a command to turn on or off a light bulb) is sent to the DT and has to be propagated to the PT, thus triggering a physical change. It is worth noting that a command aiming at modifying the state of the PT issued on the DT should be intended as another form of state synchronization. When the DT receives the command, it notifies the PT and waits for its state transition (from S_1^{PT} to S_2^{PT}). Then, once the state change on the PT is confirmed, the state of the DT is updated as well (from S_1^{DT} to S_2^{DT}). In this second scenario, the entanglement is even more relevant since a bidirectional exchange of information is required.

A. OVERALL DIGITAL TWIN ENTANGLEMENT (ODTE)

We propose ODTE as a metric for measuring in a concise yet expressive way the DT-PT entanglement. Similarly to OEE, it is conceived as a multiplication of factors resulting in a number between 0 and 1. The factors involved—*timeliness* and *completeness*—have been suggested by Fizza et al. [7] for measuring the QoE of applications where the features of exchanged data are available while human feedback is not. While *timeliness* (T) is expressed as a single factor, *completeness* is composed of two sub-factors: *reliability* (R), i.e., the ratio of the received state updates to the expected ones, and *availability* (A), i.e., the expected up-time of the PT from the perspective of the DT. Accordingly, ODTE is defined as:

$$ODTE = T \times R \times A \tag{1}$$

Let us note that we use multiplication of the three factors in our ODTE metric because in that way an ODTE value close to 1 concisely represents that all the three factors have high values. In case even only one factor is low, the ODTE value rapidly decreases: when ODTE is under an application/specific given threshold, our proposed framework allows users (typically IIoT technicians) to ask for the values of T, R, and A to obtain additional info on what is not working properly.

1) FROM PHYSICAL TO DIGITAL

To quantify the timeliness of a state update, the DT needs to track the rate of incoming status updates over time, the elapsed time between when the PT produces a given update and when the DT receives it, and how long the DT takes to change its state (based on the received update). A suitable way to model this phenomenon is by making use of histograms. The DT may use a histogram to sample

observations about the timeliness of the received updates. In this case, an observation o_i may be defined as follows:

$$o_i^{PT \to DT} = t_i^{DT} - t_i^{PT} + t_i^{exec} \tag{2}$$

where

- t_i^{DT} is the time at which the DT received the ith update;
- t_i^{PT} is the time at which the PT had produced the ith update;
- t_i^{exec} is the time the DT took to change state as a result of the ith update.

In Equation (2), the arrow direction indicates where the communication flow begins.

We can now express the timeliness T as a quantile over a time window:

$$T\left(\varphi, t, \overline{O}\right) \tag{3}$$

where

- $0 \leq \varphi \leq 1$ is the quantile;
- t is a time window (e.g., last 5 minutes);
- $\overline{O}$ is the set of observations about the received updates.

For example, $T(0.99, now - 5m, \overline{O}) = 0.100$ means that 99% of the observations had timeliness of at most 100 ms over the last 5 minutes. For computing a normalized metric such as ODTE, it is useful to express the timeliness as a percentage instead of in seconds. Thus, (3) may also be defined as:

$$T'\left(T_d, t, \overline{O}\right) \tag{4}$$

where T_d is the desired timeliness.

Equation (4) expresses the timeliness as a percentage and encapsulates any application-specific detail within the DT itself. It is reasonable, in fact, to assume that a DT is aware of the desired timeliness (T_d) of its physical counterpart. For example, if T_d is set to 200 ms (i.e., anything lower than 200 ms fulfills the requirement), $T'(200ms, 5m, \overline{O}) = 0.999$ means that 99.9% of the updates had the desired timeliness. By doing so, anyone (or anything) monitoring the DT can understand if the timeliness of state updates respects the entanglement requirements without any a priori, application-specific knowledge.

Timeliness itself does not account for those updates that are never received by the counterpart that, instead, are taken into account by the completeness factor. As stated above, we split the contribution of the completeness factor into two sub-factors, namely R and A.

Firstly, R measures the reliability of an entity expressed as the ratio of the received state updates to the expected ones within a specified time frame. Formally:

$$R\left(t, \overline{O}\right) = \frac{u_{measured}\left(t, \overline{O}\right)}{u_{expected}(t)} \tag{5}$$

where

- $u_{measured}(t, \overline{O})$ is the per-second average rate of the received updates based on the set of observations $\overline{O}$ over the time window t;
- $u_{expected}(t)$ is the *minimum* per-second average rate of the expected updates over the time window t. If $u_{measured}(t, \overline{O}) > u_{expected}(t)$, then $R(t, \overline{O}) = 1$.

For example, $R(now - 5m, \overline{O}) = 0.5$ indicates that the DT received half of the expected updates within the last 5 minutes.

Secondly, A measures the availability of the PT over a specified time frame. In the case of physical to digital communication, where there are no digital actions to be performed, $A = 1$. In the perspective of the DT, it does not matter whether the PT is available or not as long as the PT sends the expected status updates, whose frequency is already measured by R.

Putting the three components together, the ODTE for physical to digital communication may be defined as:

$$ODTE^{PT \to DT} = T'\left(T_d, t, \overline{O}\right) \times R\left(t, \overline{O}\right) \times A \tag{6}$$

2) FROM DIGITAL TO PHYSICAL

In contrast to $ODTE^{PT \to DT}$, where each observation concerns a status update, a digital to physical communication flow relates to an action and its consequences on the status (see Figure 3). Note that the timeliness can vary significantly from one action to another. For example, there may be actions where the DT expects near real-time PT responsiveness and, therefore, a corresponding status update. Instead, other actions may not have such strict timeliness requirements. As a consequence, observations about different action types cannot be treated as if they were identical. It is also worth pointing out that an observation should include information on which actions (if any) have played a role in the current status update. In this regard, consider a light bulb (i.e., the PT) that cycles on and off regularly. If the DT receives a status update that says "off" after it previously sent an "off" command to the PT, is the light off either because the requested action was executed correctly or due to its regular cycling?

According to the considerations mentioned above, an observation may be modeled as follows:

$$o_{i,a}^{DT \to PT} = t_{i,a}^{PT'} - t_{i,a}^{DT'} + o_{i,a}^{PT \to DT} \tag{7}$$

where

- $t_{i,a}^{PT'}$ is the time at which the PT received the action request a from the DT;
- $t_{i,a}^{DT'}$ is the time at which the DT had issued the action a;
- $o_{i,a}^{PT \to DT}$ is akin to a physical to digital observation but tailored to the action a.

In Equation (7), the arrow direction indicates where the communication flow begins.

A minor variation is then required for the timeliness factor T. Since T_d is not a single value anymore, Equation (4) becomes

$$T'\left(\overline{T_d}, t, \overline{O}\right) \tag{8}$$

where $\overline{T_d}$ contains the desired timeliness values for each possible action that may be requested.

In the context of digital to physical communication, the reliability may be defined as

$$R\left(t, \overline{O}\right) = \frac{a_{completed}\left(t, \overline{O}\right)}{a_{requested}(t)} \tag{9}$$

where

- $a_{completed}(t, \overline{O})$ is the number of completed actions based on the set of observations $\overline{O}$ over the time window t;
- $a_{requested}(t)$ is the number of the requested actions over the time window t. If $a_{completed}(t, \overline{O}) > a_{requested}(t)$, then $R(t, \overline{O}) = 1$.

The availability A, which describes the expected up-time of the PT from the perspective of the DT, plays an important role. Heartbeats, which are periodic messages generated by the PT to confirm its normal operation, provide a suitable method for measuring the actual PT up-time. Note that heartbeats are fundamentally different from status updates. In this regard, consider a PT that periodically sends a status update, turns it off for a while, and then turns it on to send the next status update. Let us also assume that the PT sends the $u_{expected}$ status updates and each of them reaches the DT with the desired timelines T_d. In this case, therefore, $ODTE^{PT \to DT} = 1$. However, this would be meaningless for the digital to physical case. In fact, since the PT remains offline except when sending a status update, the DT would never have the opportunity to deliver an action request.

Formally, the availability A may be defined as follows:

$$A\left(t, \overline{H}\right) = \frac{h_{measured}\left(t, \overline{H}\right)}{h_{expected}(t)} \tag{10}$$

where

- $h_{measured}(t)$ is the per-second average rate of the received heartbeats based on the set of heartbeats $\overline{H}$ over the time window t;
- $h_{expected}(t)$ is the *minimum* per-second average rate of the expected heartbeats over the time window t. If $h_{measured}(t, \overline{H}) > h_{expected}(t)$, then $A(t, \overline{H}) = 1$.

For example, $A(now - 5m, \overline{H}) = 0.5$ means that the PT was available for half of the expected time within the last 5 minutes.

Putting the three components together, the ODTE for physical to digital communication may be defined as:

$$ODTE^{DT \to PT} = T'\left(\overline{T_d}, t, \overline{O}\right) \times R\left(t, \overline{O}\right) \times A\left(t, \overline{H}\right) \tag{11}$$

Accordingly:

$$ODTE = \langle ODTE^{PT \to DT}, ODTE^{DT \to PT} \rangle \tag{12}$$

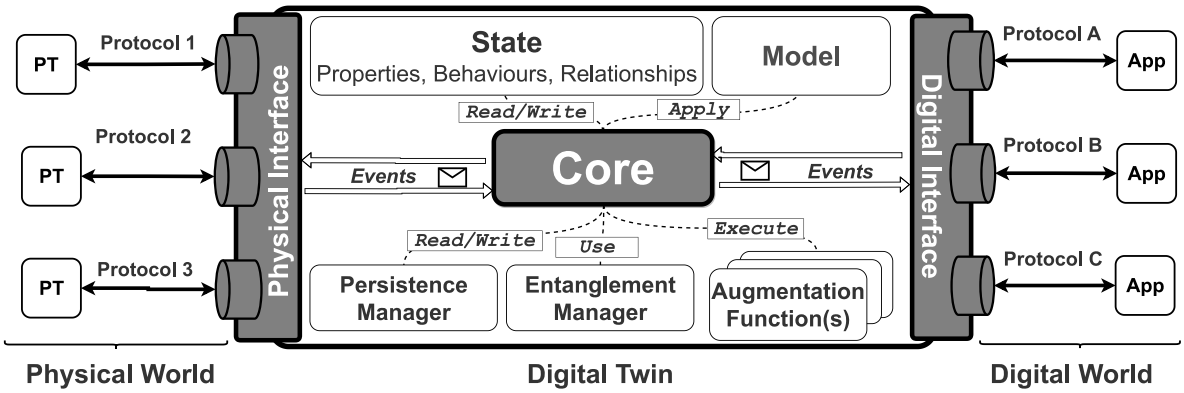
From an operational viewpoint, the DT is responsible for autonomously quantifying its own ODTE to provide either human operators or IIoT applications with a representation of the quality of entanglement. Notice that, since the proposed solution is completely decentralized (each DT computes its own ODTE in relation to the corresponding PT), the ODTE computation complexity does not depend on the number of deployed DT-PT pairs, thus without specific scalability issues. The next section discusses the DT-PT entanglement from a more operational perspective.

IV. ENTANGLEMENT-AWARE DIGITAL TWINS

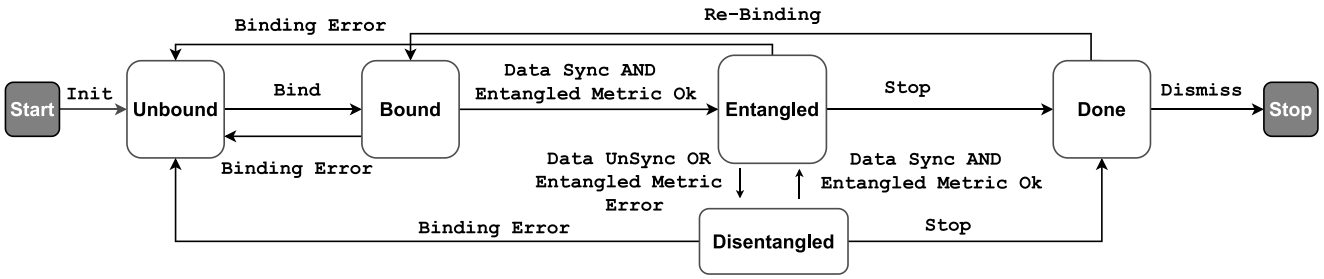
In the realm of distributed architectures, where entities can play an active role, the concept of DTs evolves from passive representations to active software components that influence and may be influenced by their physical counterparts. In contrast to DTs passively collecting data received from their associated PTs, this work envisions DTs as active entities not only capable of collecting data but also of analyzing their operational context and making decisions accordingly. Such DTs, designed according to the recognized state-of-the-art principles in the field [5], [6], [32], can be updated, re-configured, and migrated based on their needs and the ones of the currently supported applications, such as an adequate (application/specific) entanglement value. This shift necessitates a novel approach to DT modeling, development, and deployment; we claim that this shift has to be entanglement-aware.

In this context, our motivation for the design of entanglement-aware DTs revolves around the following fundamental objectives:

- *Decoupling Cyber & Physical Realms:* Our foremost goal is to effectively decouple the cyber and physical realms. Their more traditional integration often poses challenges in adaptability and maintenance. By decoupling these realms, our DT framework ensures that changes in the digital domain do not intrinsically impact the physical counterpart and vice versa. This separation facilitates efficient maintenance, upgrades, and evolution of both realms independently.
- *Modularity:* Recognizing the diverse structures and functionalities of physical entities, the proposed modeling facilitates the seamless integration of modular components. Each component serves for a specific purpose, allowing for the creation of tailored DTs and simplifying the modeling process, thus also enhancing adaptability and scalability.
- *Flexibility:* In response to the intrinsically dynamic nature of the physical world, our framework is designed to ensure that DTs can evolve alongside changes in the associated physical environment and/or evolving user needs, without forcing the framework administrators to extensive reconfiguration.
- *Entanglement Awareness:* We claim that the proper design of the targeted DT frameworks calls for robust support for entanglement. Acknowledging the intrinsic connection between the physical and digital realms, our approach facilitates the measurement and understanding of entanglement through our original ODTE metric. This support enhances the accuracy of digital



(a) DT abstraction.



(b) Life cycle.

FIGURE 4. DT architectural abstraction and its life cycle based on entanglement.

representations and enables efficient monitoring and management of the related dynamic relationships.

From a technical standpoint (shown in Figure 4a), the *Digital Twin Model* is responsible for determining how and when changes in the physical world should be mapped into the digital replica, as well as propagating inputs and actions to the PT. The model closely works with the *Digital Twin State* component, storing *attributes* (e.g., physical properties), *behaviors* (e.g., actions that can be performed on the DT), and *relationships* (e.g., modeling how PTs are linked in the physical space). The interaction with the physical and digital layers builds upon the *Physical Interface* (PI) and the *Digital Interface* (DI), each composed of different *Adapters* (implementing protocols and data formats). On the one hand, the PI serves as the conduit through which a DT communicates and engages with its physical counterpart (i.e., the PT). This interface entails the integration of diverse communication protocols and hardware components: its primary purpose is to establish a seamless connection with physical assets, sensors, and actuators in the real world. The PI ensures that the DT can efficiently receive real-time data, can exert control over the associated physical processes, and can synchronize its state with the PT.

We propose a specific type of framework components (i.e., the Adapters) to act as intermediaries that facilitate bidirectional communication between the DT and the PT. These Adapters can support (if needed) a multitude of communication protocols, including MQTT, CoAP, Modbus, and others, alongside various interaction patterns such as publish/subscribe or request/response. This adaptability empowers the DT to effectively engage with diverse types of physical entities. For example, an adapter may seamlessly convert sensor data from a proprietary format to a standardized one or navigate the translation from an event-driven interaction to a synchronized approach.

This approach to physical communication yields several benefits. Firstly, it enables the DT to efficiently receive and process real-time data from sensors and other physical devices. Secondly, it facilitates the execution of actions and control commands that exert influence over the behavior of the PT. Lastly, the PI support for multiple communication protocols enhances the DT adaptability, by allowing it to seamlessly interface with an array of physical assets and devices.

On the other hand, the DI assumes the responsibility of linking the DT with the broader digital ecosystem, encompassing other DTs, digital services, and applications. Its role is to facilitate seamless, collaborative, and interoperable data exchange within the digital realm. Adapters play a crucial role within the DI, managing communication protocols, data formats, and interactions in the digital space. For instance, a DI might facilitate communication through diverse protocols like RESTful APIs, MQTT, or WebSocket, by enabling the DT to share information with external applications via standardized interoperable protocols and data formats

tailored to the specific use case and deployment environment. This approach provides significant benefits to the digital side, first of all in terms of interoperability.

By relying on this modular framework, the model receives inputs from the physical layer. Such inputs are reflected in the digital representation either immediately or after various transformations to align them with the DT model (e.g., resampling signals, and changing metric units). Given that modifying the functionalities of PTs might be costly and complex, a physical asset can be functionally expanded through its DT, using a collection of *Augmentation Functions* introducing additional attributes, behaviors, or relationships. The DT also integrates an *Entanglement Manager* responsible for monitoring the entanglement. This module adjusts the DT state and generates contextual metrics. All internal DT modules are supported by a *Persistence Manager* component handling the memorization and retrieval of past states and events.

Each DT is in charge of monitoring its entanglement and evaluates it in light of the context where it operates and of application requirements. In particular, with respect to the extraction, collection, and pre-processing of the values for ODTE determination, the PI, together with its Adapters, plays a crucial role in decoupling the DT core from the complexity of retrieving this information, by favoring the definition and deployment of reusable components and DT instances. The PI can adopt, through its Adapter, any protocol or interaction pattern (e.g., publish/subscribe, event-driven, or request/response) required to communicate with the PT and to support ODTE parameter retrieval. In this context, on the one hand, some protocols and communication techniques may directly support or provide target-relevant values (e.g., packet timestamp for delay computation or heartbeats monitoring). On the other hand, in other cases these parameters should be inferred by the Adapters of the PI, for example, via local timestamp reference caching or via proactively checks of PT availability to compensate for missing heartbeats information. The specific focus of this paper is not to list and discuss all the communication and protocol options that our design approach simplifies to support, but rather to push towards the idea of the suitability of a general modular framework to enable entanglement measurements through the ODTE metric such a framework can be applied to different application scenarios and use cases, without any bound to any specific protocol or interaction pattern. Section V presents our prototype implementation together with the details of the specific protocols employed within the identified reference testbed.

Finally, the DT life cycle (shown in Figure 4b) is crucial to model the behaviour of a DT-PT dyad and inform its computational environment about its evolution. Upon its start, the DT is *Unbound* and ready to bind to the PT. Once the binding is completed (a network channel with the PT is established and the DT is ready to initiate the digitization process), the DT moves to the *Bound* state. If binding errors occur, the state reverts back to *Unbound* and the DT tries to recover the channel. In the *Entangled* state, the entanglement is measured. Networking or computational resource issues involving the DT-PT synchronization and degrading the level of entanglement below a target threshold bring the DT into the *Disentangled* state. In this state, the DT becomes unable to provide its intended functionality. From the *Disentangled* state, the DT can transition to either the *Unbound* or *Done* state in case of an error during the binding procedure or if it is explicitly stopped by the middleware. Upon successful error recovery, the DT reverts back to the *Entangled* state. In the *Done* state the DT remains accessible to external applications as a software component detached from the PT, retaining its memory and exposing collected historical data, events, and metrics together with the last DT state until it is dismissed, by transitioning to the *Stop* state.

V. EXPERIMENTAL EVALUATION AND DISCUSSION

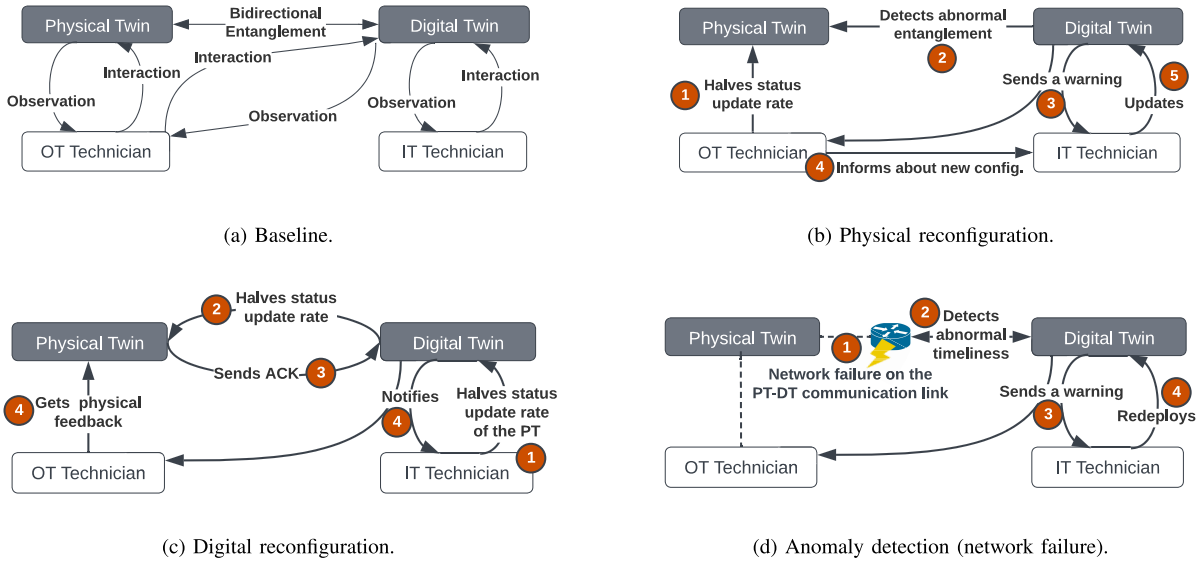
This section *(i)* outlines four illustrative scenarios to emphasize the primary factors that may impact entanglement in industrial contexts, *(ii)* details the testbed where we performed the reported experiments, and *(iii)* discusses the performance results collected in those experiments.

A. FOUR ILLUSTRATIVE SCENARIOS FOR ENTANGLEMENT

We elaborated on four illustrative scenarios that point out the main factors that may affect entanglement. Such scenarios illustrate the interactions between Information Technology (IT) and Operation Technology (OT) in IIoT environments and how those interactions affect the entanglement. Specifically, each scenario involves an OT technician working on a PT and an IT technician working on a DT. For instance, the former might be a shop floor operator working in a production line (i.e., where industrial automation takes place), while the latter might be a software architect deploying a DT as a microservice through a container orchestration system.

1) BASELINE

The baseline scenario describes the interactions between IT and OT in a DT-based industrial environment (see Figure 5(a)). Under the baseline scenario, we assume that those interactions do not disrupt the current entanglement characterizing the communication relationship between the PT and the DT. The OT technician interacts with the PT (e.g., a production line) to craft goods and observes the PT status to oversee what is going on. Additionally, the OT technician may request a simulation to the DT and, based on the simulation results, decide on the subsequent actions to send to the PT. Instead, the IT technician only interacts with the DT. Such interactions may relate, for example, to the deployment of the DT. It is worth remarking that different layers of IIoT environments provide different network performances, thus influencing the entanglement. Therefore, the IT technician should plan the DT deployment carefully and re-plan it dynamically according to the network conditions.



(a) Baseline.

(b) Physical reconfiguration.

(c) Digital reconfiguration.

(d) Anomaly detection (network failure).

FIGURE 5. Illustrative scenarios showing the main interaction patterns between cyber-physical industrial entities.

As the number of DTs grows, so does the complexity of making effective decisions about their deployment. Thus, quantifying the entanglement in a concise yet expressive way becomes even more critical.

2) PHYSICAL RECONFIGURATION

The physical reconfiguration scenario sketches the case where an action of the OT technician on the PT disrupts the entanglement (see Figure 5(b)). For instance, the OT technician may change the configuration of the PT, which may result in a different status update rate, say halving the status updates per second. In turn, the DT detects an abnormal entanglement because it still expects double the status updates it is actually receiving. As soon as the DT detects that the entanglement got disrupted, it notifies the OT/IT technicians. At this time, the IT technician can only infer that something is not going as expected. Therefore, the OT technician, whose initial interaction caused the misalignment between the PT and the DT, should notify the IT technician about the change they made to the PT. Then, the IT technician can update the configuration of the DT accordingly, thus bringing the system back to a steady-state phase.

3) DIGITAL RECONFIGURATION

The digital reconfiguration scenario sketches the case where an action of the IT technician on the DT disrupts the entanglement (see Figure 5(c)). At first glance, this scenario might seem symmetrical to the physical reconfiguration one, but it is not. In particular, the IT technician uses the DT to change the configuration of the PT. For example, the IT technician may halve the status update rate of the PT through the DT. As soon as the DT receives the instructions issued by the IT technician, it sets the PT accordingly. At this time, the DT must wait until the PT reports a status update reflecting a status change meeting the request(s) of the DT. Then, the DT can notify the OT/IT technicians back. Note that the OT technician might have already noticed that the PT changed status because of physical feedback from the PT, e.g., a robot part of the production line where the OT technician is operating changed position.

4) ANOMALY DETECTION

Under the physical and digital reconfiguration scenarios, an intentional action triggered the course of action affecting the entanglement. In contrast, the anomaly detection scenario is about things that could go wrong unpredictably (see Figure 5(d)). In particular, this scenario takes into account anomalies striking either the PT (e.g., crash of the production line), the environment (e.g., poor network connectivity between the PT and the DT), or the DT (e.g., hardware fault of the server hosting the DT). Let us assume an outburst of latency upon the communication link that connects the PT and the DT. The DT can detect such an anomaly by looking at the timeliness of the received status updates from the PT. If we instead assume a crash of the PT, the DT can detect that something is not as expected because of a drop in the status update rate. Note that the OT technician may also detect the crash of the PT through physical feedback, e.g., the production line stops working. The recovery phase is started by the actor that detects the anomaly first. In the former case, the DT would start the recovery phase by notifying the IT technician, who might decide, e.g., to redeploy the DT somewhere else or fix the network. In the latter case, the OT technician would start the recovery phase, e.g., by fixing the PT. The outcome of the recovery phase is to bring the system back to a steady-state phase.

B. TESTBED AND EXPERIMENTS

An emerging trend in industrial environments is to adopt cloud-oriented technologies, such as Virtual Machines (VMs) and containers. In particular, a microservice approach makes software development, deployment, and management easier. In fact, microservices are also gaining momentum in Industry 4.0 [33], [34] and represent a valuable option for managing the lifecycle of DTs. Accordingly, we relied on Kubernetes to build a representative testbed in line with current industry trends. Specifically, Kubernetes—the de facto industry standard container-orchestration system—is a platform for automating the deployment, scaling, and management of containerized applications. On top of Kubernetes, we deployed Prometheus and Chaos Mesh. The former was used to scrape metrics from DTs (deployed as containers through Kubernetes), store such metrics in a time-series database, and query the database to extrapolate aggregate insights. The latter is a cloud-native chaos engineering platform for Kubernetes that allows injecting a broad spectrum of faults into a target. Chaos engineering techniques have been shown to be effective in assessing DT resilience [35]. Through Chaos Mesh, we reproduced a broad spectrum of network conditions under which we tested the effectiveness of the proposed ODTE metric in quantifying the entanglement.

The testbed consisted of a Kubernetes cluster of four nodes, each within its own VM. A single node acted as the master (i.e., the node running the control plane) while the others joined as workers (i.e., regular cluster nodes). Each VM was equipped with 2 vCPU and 2 GB of RAM, each one based on Ubuntu. The testbed was automatically configured in a reproducible manner using Ansible, a well-known configuration management tool that configures a set of target nodes over SSH through a control node. In this way, we made up the Kubernetes cluster (i.e., Kubernetes along with the ancillary software required by Kubernetes to run successfully, such as cri-o and Flannel) as well as deployed Prometheus and Chaos Mesh.

Through Kubernetes, we orchestrated the deployment of DTs, PTs, and message brokers as containerized applications. The physical layer operates on the shop floor and consists of PTs, emulating the behavior of IIoT devices within the target demo use case. PTs were emulated to streamline the experimental evaluation with a more configurable approach and were implemented in Java, by following standard protocols and data formats. Each emulated industrial machine publishes its status information and is updated via JSON (JavaScript Object Notation) on an MQTT [36] message broker. Each emulated PT manages a set of configurable resources composed by three sub-elements (energy consumption, battery level, and temperature) constituting the overall PT state: any sub-element is published on independent topics, together with a topic to update the entire physical state with a configurable message rate (ranging from 10 ms to 100 ms) and an average payload size of 100 Bytes. Similar to the DT, each PT exposes interfaces to configure its behavior at runtime. The chosen MQTT message broker is Eclipse Mosquitto[1] and PTs transmit status updates to the DT as MQTT messages on a topic to which the DT subscribes. In addition, the DT issues commands to the PT, by publishing on another topic to which the PT subscribes.

The implementation of the proposed and designed DTs leverages a Java DT engine, characterized by built-in modularity and a microkernel-oriented structure. This implementation exclusively relies on open-source technologies. Entanglement-aware DTs were implemented by using the White Label Digital Twin (WLDT) library,[2] a modular Java stack built on a shared multi-thread engine. This library simplifies the implementation of DT behavior, as well as the definition of mirroring procedures, data processing, and interaction with external applications [37]. To accommodate DT-PT entanglement, we originally extended that library according to the requirements discussed in Section III and the architectural specifications of Section IV. Then, container images of the implemented DTs were generated and hosted in a dedicated container registry to be used within the configured testbed.

Each DT has been configured and equipped with dedicated MQTT Adapters for both PI and DI. As already stated, this ensures proper communication with the PT and the broader digital ecosystem, by using interoperable standard protocols and data formats. Each DT is tasked with the digitalization of a target PT by: i) processing and adapting received payloads to adhere to the standard Sensor Measurement Lists (SenML) [38] data format; ii) evaluating and maintaining the internal status; and iii) handling potential incoming commands and reconfiguration requests sent by applications. Notably, each DT publishes its status variations (using SenML) to the MQTT message broker.

To validate the ODTE metric, we first focused on the timeliness factor in the context of an industrial environment based on the Purdue model [39]. The Purdue model recommends a hierarchical approach that splits the industrial network into five layers, with the first three layers related to OT (focusing on industrial machine management) and the last two dedicated to IT (related to crafting planning, Web servers, email servers, and databases). From a network perspective, each layer of the Purdue model is supposed to provide different performance. As a rule of thumb, the lower, the better. For example, since the goal of layer 0/1 is to meet safety-critical requirements, the network is expected to provide high reliability and low latency (e.g., within 10 ms). However, layer 0/1 does not provide as abundant computing resources as the upper layers. In this regard, layer 2 tends to provide worse network performance (e.g., within 25 ms) than layer 0/1 but potentially more computing resources and less stringent security requirements. In this regard, Table 1 details the network conditions we injected using Chaos Mesh to emulate such an industrial deployment environment. Specifically, we established plausible regular

[1]Eclipse Mosquitto MQTT Broker: https://mosquitto.org/
[2]WLDT GitHub: https://github.com/wldt/

TABLE 1. Experiments based on the Purdue model layers.

Purdue Model	Network conditions	Latency ± Jitter	Loss
Layer 0/1	Regular (R)	$2.5\ ms \pm 2.5\ ms$	-
	Deteriorated (D)	$5\ ms \pm 5\ ms$	5 %
	Critical (C)	$12.5\ ms \pm 12.5\ ms$	15 %
Layer 2	R	$12.5\ ms \pm 7.5\ ms$	-
	D	$25\ ms \pm 15\ ms$	5 %
	C	$50\ ms \pm 30\ ms$	15 %
Layer 3	R	$35\ ms \pm 15\ ms$	-
	D	$70\ ms \pm 30\ ms$	5 %
	C	$175\ ms \pm 75\ ms$	15 %

(R), deteriorated (D), and critical (C) network conditions for each OT layer of the Purdue model (i.e., layer 0/1, layer 2, and layer 3). The regular network conditions represent those expected under normal operations. In contrast, the deteriorated and critical conditions represent scenarios that are worse than typical but still likely to occur during real-life operations. For example, a DT deployed at layer 0/1 under regular network conditions would experience a one-way latency of $2.5\ ms \pm 2.5\ ms$ (with a correlation between consecutive packets of 25%) and no packet loss. These experiments were conducted over a time window of 5 minutes. Let us note that not only these experiments aim to practically show how our timeliness factor operates, but they also illustrate some criteria to determine where to deploy a DT based on its desired timeliness.

Then, we instantiated the illustrative scenarios described in Section V-A to conduct an experimental evaluation of the ODTE metric. These experiments are designed to validate the ODTE metric in practical scenarios where entanglement is influenced by a variety of factors. The physical reconfiguration scenario was emulated by halving the status update rate sent by the PT to the DT, i.e., from 1 to 0.5 status updates per second. Then, we instantiated the digital reconfiguration scenario by forcing the DT to calculate 17.5K prime numbers while performing a state transition. Lastly, we produced two instances of the anomaly detection scenario to investigate the responsiveness of the ODTE metric to latency (i.e., $50\ ms \pm 50\ ms$) and the combined effect of latency (as before) and packet loss (i.e., 10%). We performed these experiments in three phases (5 minutes each) over a time window of 15 minutes overall. The first phase resembled the baseline scenario, the second put into action a given scenario, and the third consisted of rolling back what had been injected to reproduce the scenario (thus bringing the system back to the baseline).

A repository hosting any relevant software artifacts developed for this paper is publicly available on GitHub, also to foster result reproducibility and the full understanding of our proposed framework.[3]

C. RESULTS

Figure 6 shows the results we collected from the experiments focusing on timeliness. As expressed in Equation (3), the timeliness factor T takes as arguments a quantile φ, a time interval t, and a set of observations $\overline{O}$. Specifically, Figure 6

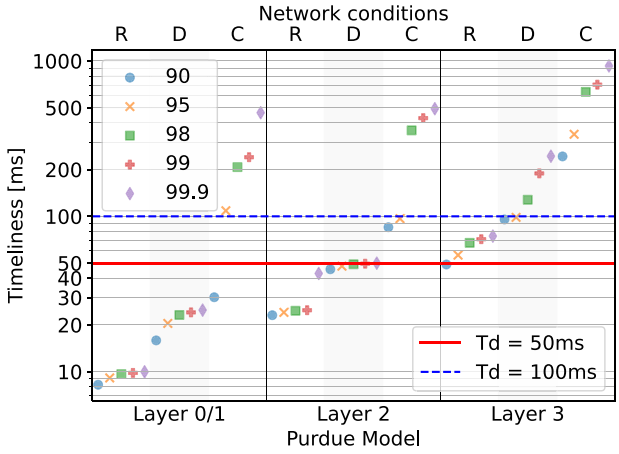


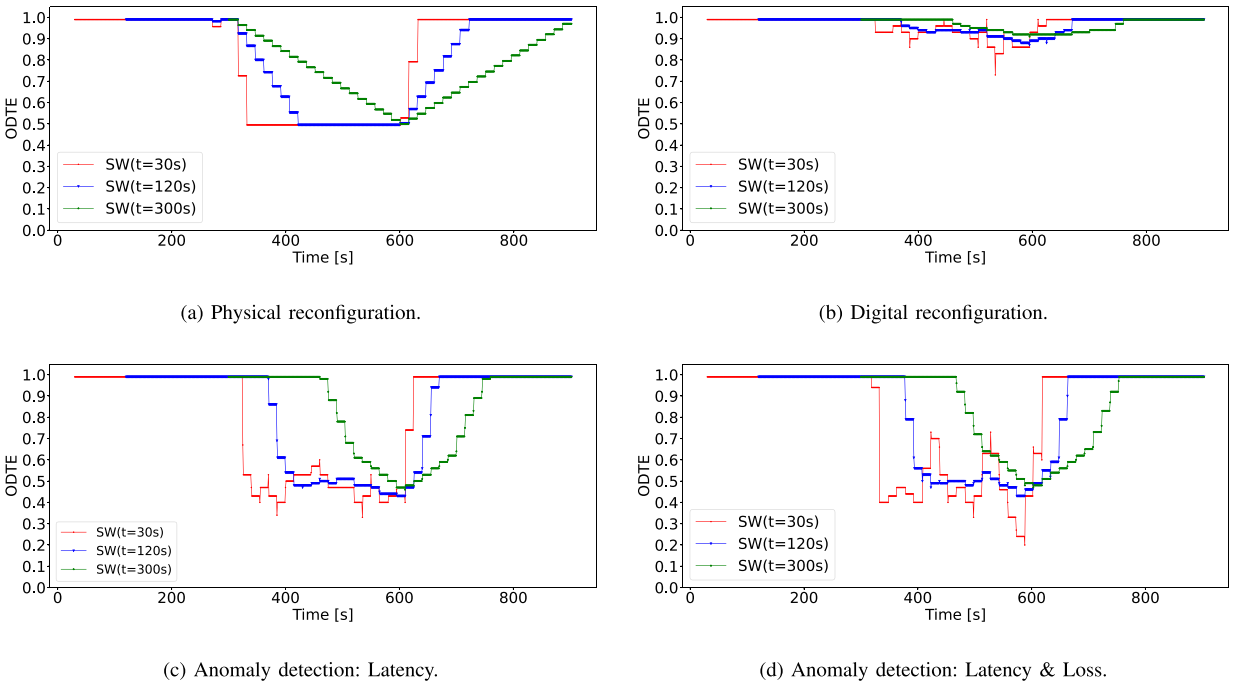
FIGURE 6. Timeliness performance with network effects resembling the Purdue model layers.

uses percentiles, i.e., 90th, 95th, 98th, 99th, and 99.9th, computed based on the metrics ($\overline{O}$) scraped by Prometheus over a 5-minute window (t).

Note that the left y-axis depicts the timeliness expressed in ms on a logarithmic scale (base 10). The bottom x-axis divides the figure into three vertical macro-sections, each representing a layer of the Purdue model. The top x-axis further divides those macro-sections based on plausible network conditions, i.e., regular, deteriorated, and critical, that might affect each layer of the Purdue model (see Table 1). For example, the yellow cross on the second column means that 95% of the status updates received by the DT had timeliness of at most 20 ms over the observed 5-minute window. The solid red horizontal line distinguishes the layers of the Purdue model that fit a DT with desired timeliness of 50 ms between those that do not. If the target is the 90th percentile, then layer 0/1 represents a suitable option under any network condition. Layer 2 is also a suitable option but only up to the deteriorated network conditions (the 90th percentile almost doubled the desired timeliness while critical network conditions occurred). Instead, the dashed blue horizontal line refers to a DT whose desired timeliness is 100 ms. If we still assume that the target is the 90th percentile, then both layers 0/1 and 2 are suitable deployment options under any network condition.

Figures 7(a), 7(b), 7(c), and 7(d) show the responsiveness of the ODTE metric over a 15-minute time window concerning the experiments instantiating the (a) physical reconfiguration scenario, (b) digital reconfiguration scenario, (c) the anomaly detection scenario where the anomaly was an outburst of latency, and (d) where two anomalies were in place simultaneously, i.e., latency and loss. As there are no actions involved in the illustrative scenarios, ODTE equals to Equation (6). The fixed factors were the desired timeliness T_d, which was set to 50 ms, and the desired availability A, which was set to 1. Then, t was varied from 30 s to 2 min to 5 min. For example, if $t = 30$, then the

[3]https://github.com/fglmtt/ojcoms-2024-artifacts



(a) Physical reconfiguration.

(b) Digital reconfiguration.

(c) Anomaly detection: Latency.

(d) Anomaly detection: Latency & Loss.

FIGURE 7. ODTE performance with varied sliding windows (SWs) in the illustrative scenarios.

set of observations $\overline{O}$ consists of the observations collected over the previous 30 s. Since the PT was supposed to send 1 status update per second, then $u_{expected} = 1$. Accordingly, Figures 7(a), 7(b), 7(c), and 7(d) depict three lines, i.e., red, blue, and green, each plotting the ODTE computed on a different sliding window (see the abbreviation SW in the legend), i.e., 30 s, 2 min, and 5 min, respectively. On the one hand, a shorter sliding window makes the ODTE metric more responsive (the red line reacts faster than the others to the scenario). On the other hand, a shorter sliding window makes the ODTE metric more sensitive to noise. The wide fluctuations depicted in Figure 7(d) (see the red line) make evident the impact of a shorter sliding window on the ODTE metric. However, this does not mean that longer sliding windows are always better than shorter ones. The sliding window width choice should reflect the target DT sensitivity to short-lived variations of the entanglement over time. Finally, Figure 8 shows the ODTE (sliding window of 30 s) concerning two DTs whose desired timeliness was 50 ms (red line) and 100 ms (blue line), and both were performing under the same anomaly detection scenario with latency and loss as described above. The DT with 100 ms of desired timeliness was much less influenced (blue) than the one with 50 ms of desired timeliness (red).

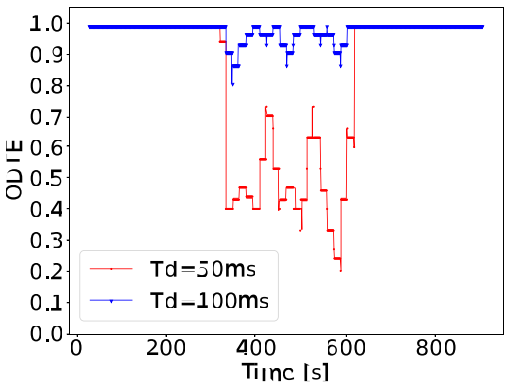


FIGURE 8. Comparison between two applications with different T_d in the anomaly detection scenario (latency & loss).

D. DISCUSSION

The reported results demonstrate the effectiveness and usability of the proposed and original ODTE metric. The metric has been extensively tested in illustrative target scenarios of practical interest, properly selected for their differentiated characteristics (i.e., physical reconfiguration, digital reconfiguration, and anomaly detection). Each of these scenarios jeopardizes the entanglement differently. For example, the physical reconfiguration scenario changes the frequency of updates that the PT sends without notifying the DT; the digital reconfiguration scenario introduces a CPU-intensive computation on the state to be considered for the DT; and the anomaly detection scenario injects network faults over the communication link connecting the PT and the DT. Some of the effects associated with and introduced by these scenarios are at the application level (e.g., variation in the number of status updates or introduction of a CPU-intensive computation on state transition), while others are at the network level (e.g., injected latency on a communication link). Notice that most traditional QoS metrics in the literature could have captured network-level issues but could not possibly capture application-level

nuances. It is also worth mentioning that the proposed ODTE metric not only captures network and application-level nuances but also evaluates their impact on the overall entanglement of a specific DT. In this regard, it is important to underlie also that the ODTE metric embeds domain-specific knowledge directly into the DTs: indeed, it is reasonable that this knowledge (i.e., T_d, $u_{expected}$, $h_e xpected$, and the ODTE threshold at which the DT stops working properly) is known by the DT developers. In this way, system administrators or manufacturing line technicians can be alerted whenever needed without the need to have application-specific knowledge regarding the deployed DTs, their nature, design or implementation specifications. For example, manufacturing line technicians could be instructed to perform further analysis only when ODTE deteriorates below a DT-developer-defined threshold of 0.8. In this case, the dynamic evolution of the timeliness, reliability, and availability indicators can be independently investigated to provide further elements about the cause of the possible problem and about how to solve it, e.g., whether it relates to either network issues or the DT performing a very CPU-intensive task that saturated its local resources.

Through our conducted experimental evaluation, we illustrated the robust performance of the introduced ODTE metric. Across the identified target illustrative scenarios, we have demonstrated how the metric detects variations in context, effectively quantifying the entanglement between a DT and its associated physical counterpart. Our experiments have encompassed diverse configuration parameters, providing a comprehensive investigation of the metric performance including variations in terms of desired timeliness, latency, loss, and processing capability. We have highlighted the metric versatility and applicability across different use cases and multiple domains. This uniform and homogeneous representation and measurement of entanglement offer valuable insights for external observers, facilitating a deeper understanding of the cyber-physical relationship between the DT and its physical counterpart.

VI. CONCLUSION AND FUTURE WORK

In this work, we have presented ODTE—an innovative metric for quantifying the quality of entanglement between DTs and PTs. The metric is based on factors, such as timeliness and completeness, for computing QoE without relying on subjective evaluations. More specifically, while timeliness has been represented with a single term, completeness has been split into two terms, namely reliability and availability. ODTE provides technicians with a concise yet expressive value normalized between 0 and 1, not requiring any application-specific knowledge to be correctly understood. Experimental results show that ODTE is responsive at quantifying the quality of entanglement under IIoT scenarios of practical interest. In future work, we plan to generalize the ODTE concept and develop additional metrics that account for other aspects of DTs, such as their interaction with external services and emergent properties of ensembles or hierarchies of DTs.

REFERENCES

[1] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020.

[2] K. Hribernik, G. Cabri, F. Mandreoli, and G. Mentzas, "Autonomous, context-aware, adaptive digital twins—State of the art and roadmap," *Comput. Ind.*, vol. 133, Dec. 2021., Art. no. 103508.

[3] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

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

[5] R. Minerva and N. Crespi, "Digital twins: Properties, software frameworks, and application scenarios," *IT Prof.*, vol. 23, no. 1, pp. 51–55, Jan./Feb. 2021.

[6] A. Ricci, A. Croatti, S. Mariani, S. Montagna, and M. Picone, "Web of digital twins," *ACM Trans. Internet Technol.*, vol. 22, no. 4, pp. 1–30, Nov. 2022. [Online]. Available: https://doi.org/10.1145/3507909

[7] K. Fizza et al., "QoE in IoT: A vision, survey and future directions," *Discov. Internet Things*, vol. 1, no. 1, pp. 1–14, 2021.

[8] V. Sachidananda, A. Khelil, and N. Suri, "Quality of information in wireless sensor networks: A survey," in *Proc. Int. Conf. Inf. Qual.*, 2010, pp. 1–15.

[9] Y. Shu, Z. Wang, H. Liao, Z. Zhou, N. Nasser, and M. Imran, "Age-of-information-aware digital twin assisted resource management for distributed energy scheduling," in *Proc. IEEE Global Commun. Conf.*, 2022, pp. 5705–5710.

[10] B. Dal, P. Tugwell, and R. Greatbanks, "Overall equipment effectiveness as a measure of operational improvement–a practical analysis," *Int. J. Oper. Prod. Manage.*, vol. 20, no. 12, pp. 1488–1502, 2000.

[11] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "Measuring digital twin entanglement in Industrial Internet of Things," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 5897–5903.

[12] B. Saovapakhiran, W. Naruephiphat, C. Charnsripinyo, S. Baydere, and S. Özdemir, "QoE-driven IoT architecture: A comprehensive review on system and resource management," *IEEE Access*, vol. 10, pp. 84579–84621, 2022.

[13] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, "Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2230–2254, 4th Quart., 2022.

[14] D.-H. Shin, "Conceptualizing and measuring quality of experience of the Internet of Things: Exploring how quality is perceived by users," *Inf. Manage.*, vol. 54, no. 8, pp. 998–1011, 2017.

[15] Y. Ikeda, S. Kouno, A. Shiozu, and K. Noritake, "A framework of scalable QoE modeling for application explosion in the Internet of Things," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, 2016, pp. 425–429.

[16] M. Suryanegara, D. A. Prasetyo, F. Andriyanto, and N. Hayati, "A 5-step framework for measuring the quality of experience (QoE) of Internet of Things (IoT) services," *IEEE Access*, vol. 7, pp. 175779–175792, 2019.

[17] W. Robitza et al., "Challenges of future multimedia QoE monitoring for Internet service providers," *Multimedia Tools Appl.*, vol. 76, no. 21, pp. 22243–22266, 2017.

[18] L. Li, M. Rong, and G. Zhang, "An Internet of Things QoE evaluation method based on multiple linear regression analysis," in *Proc. 10th Int. Conf. Comput. Sci. Educ. (ICCSE)*, 2015, pp. 925–928.

[19] I. de la Torre Díez, S. G. Alonso, E. M. Cruz, and M. A. Franco, "Measuring QoE of a teleconsultation app in mental health using a pentagram model," *J. Med. Syst.*, vol. 43, no. 7, pp. 1–5, 2019.

[20] M. Aazam and K. A. Harras, "Mapping QoE with resource estimation in IoT," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, 2019, pp. 464–467.

[21] I. Alqerm and J. Pan, "DeepEdge: A new QoE-based resource allocation framework using deep reinforcement learning for future heterogeneous edge-IoT applications," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 3942–3954, Dec. 2021.

[22] K. Fizza, P. P. Jayaraman, A. Banerjee, N. Auluck, and R. Ranjan, "IoT-QWatch: A novel framework to support the development of quality-aware autonomic IoT applications," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 17666–17679, Oct. 2023.

[23] D. Chen and M. Törngren, "A metrics system for quantifying operational coupling in embedded computer control systems," in *Proc. 4th ACM Int. Conf. Embed. Softw.*, 2004, pp. 184–192.

[24] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE INFOCOM*, 2012, pp. 2731–2735.

[25] I. Kadota, A. Sinha, and E. Modiano, "Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1359–1372, Aug. 2019.
[26] C. Xu, Y. Xie, X. Wang, H. H. Yang, D. Niyato, and T. Q. S. Quek, "Optimal status update for caching enabled IoT networks: A dueling deep R-network approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8438–8454, Dec. 2021.
[27] C. Xu, H. H. Yang, X. Wang, and T. Q. Quek, "Optimizing information freshness in computing-enabled IoT networks," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 971–985, Feb. 2020.
[28] B. Yin, S. Zhang, and Y. Cheng, "Application-oriented scheduling for optimizing the age of correlated information: A deep-reinforcement-learning-based approach," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8748–8759, Sep. 2020.
[29] B. Zhou and W. Saad, "On the age of information in Internet of Things systems with correlated devices," in *Proc. IEEE Global Commun. Conf.*, 2020, pp. 1–6.
[30] A. E. Kalor and P. Popovski, "Timely monitoring of dynamic sources with observations from multiple wireless sensors," *IEEE/ACM Trans. Netw.*, vol. 31, no. 3, pp. 1263–1276, Jun. 2023.
[31] H. Liao et al., "Ultra-low AoI digital twin-assisted resource allocation for multi-mode power IoT in distribution grid energy management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3122–3132, Oct. 2023.
[32] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "Requirements and design patterns for adaptive, autonomous, and context-aware digital twins in industry 4.0 digital factories," *Comput. Ind.*, vol. 149, Aug. 2023, Art. no. 103918.
[33] M. Azarmipour, H. Elfaham, C. Gries, T. Kleinert, and U. Epple, "A service-based architecture for the interaction of control and MES systems in industry 4.0 environment," in *Proc. IEEE Int. Conf. Ind. Inform. (INDIN)*, 2020, pp. 217–222.
[34] F. Siqueira and J. G. Davis, "Service computing for industry 4.0: State of the art, challenges, and research opportunities," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–38, Oct. 2021.
[35] M. Fogli, C. Giannelli, F. Poltronieri, C. Stefanelli, and M. Tortonesi, "Chaos engineering for resilience assessment of digital twins," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 1134–1143, Feb. 2024.
[36] (OASIS, Woburn, MA, USA). *MQTT Version 3.1.1*. Sep. 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
[37] M. Picone, M. Mamei, and F. Zambonelli, "WLDT: A general purpose library to build IoT digital twins," *SoftwareX*, vol. 13, Jan. 2021, Art. no. 100661.
[38] C. Jennings, Z. Shelby, J. Arkko, A. Keränen, and C. Bormann, "Sensor measurement lists (SenML)," Internet Res. Task Force, RFC 8428, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8428.txt
[39] T. J. Williams, "The purdue enterprise reference architecture," *Comput. Ind.*, vol. 24, nos. 2–3, pp. 141–158, 1994.

PAOLO BELLAVISTA (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science engineering from the University of Bologna, Italy, where he is currently a Full Professor of Distributed and Mobile Systems. His research activities span from pervasive wireless computing to online big data processing under quality constraints, from cloud continuum middleware to digital twins for industry 4.0 applications. He serves on several editorial boards, including IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (Associate EiC), *ACM Computing Surveys*, and *Journal of Network and Computer Applications* and *Pervasive and Mobile Computing* (Elsevier). He was recently a Scientific Coordinator of the H2020 BigData Project IoTwins.

NICOLA BICOCCHI is an Associate Professor with the University of Modena and Reggio Emilia, Italy. His research activity is focused on Internet of Things and pervasive computing. He mostly investigates new forms of interaction between humans, personal devices, and data in increasingly pervasive environments. In these areas, he published more than 60 papers in international venues.

MATTIA FOGLI (Member, IEEE) received the Ph.D. degree in computer engineering from the University of Ferrara, Italy, in 2024. He served as a Research Intern with the Florida Institute for Human and Machine Cognition, USA, from 2019 to 2020. He is currently a Postdoctoral Researcher with the University of Ferrara. His research interests include digital twins, service orchestration in the compute continuum, and federated cloud infrastructures for tactical edge networks.

CARLO GIANNELLI (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Bologna, Italy, in 2008. He is currently an Associate Professor of Computer Science with the University of Ferrara, Italy. His primary research activities focus on industrial Internet of Things, digital twin management, software defined networking, blockchain technologies, and cybersecurity in industry 4.0. He serves on the editorial boards of *ACM Computing Surveys*, *Computer Communications* (Elsevier), and *EURASIP Journal on Wireless Communications and Networking* (Springer).

MARCO MAMEI received the Ph.D. degree in computer science from the University of Modena and Reggio Emilia in 2004, where he has been a Full Professor of Computer Science since 2019. His work focuses on data mining applied to mobile phone data and Internet of Things applications. In these areas, he published more than 100 papers, eight patents, and won several best paper awards.

MARCO PICONE received the Ph.D. degree in information technology from the University of Parma. He is a Senior Assistant Professor with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia. His research interests are distributed systems, Internet of Things, edge/fog computing, and digital twins.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement