Digital Twins from a Networking Perspective

Mehrad Vaezi, Kiana Noroozi, Terence D. Todd, *Life Member, IEEE,* Dongmei Zhao, *Member, IEEE,* George Karakostas, Huaqing Wu, *Member, IEEE,* and Xuemin (Sherman) Shen, *Fellow, IEEE*

***Abstract*—Digital twin (DT) has attracted a lot of attention from both industry and academia since it was proposed over a decade ago. A DT can be viewed as a virtual implementation of a real physical system and used as a representation of the physical system for various applications. Despite the great potential of DTs in various fields, implementing DTs to obtain the desired functionality is not always straightforward. Specifically, accurate real-time synchronization between the features at a physical system and its DT is essential for the DT to represent the physical system. In this case, appropriate networking support is a key component to enable future DT development and applications. Currently, the research on DTs from a networking standpoint is still at an early stage, and only limited work has been done on DT implementation in practical systems. To fill this gap, this paper investigates networking-related issues for DTs. Based on the existing literature, a feature-based method is provided for describing the desired properties and quality of DTs from the networking perspective. A stage-based implementation framework is presented for creating large-scale DTs for complex physical systems by considering various networking constraints. Networking-related challenging issues and open research topics are discussed at the end.**

***Index Terms*—Digital twin, DT features, DT properties, stage-based DT implementation.**

I. Introduction

Digital twin (DT) is a concept where a digital/virtual representation is created for a physical system (PS) of interest. The DT can be used to complement the PS for a variety of purposes in the design, evolution, and deployment stages. The DT idea was originally introduced for "product life cycle management" [1], but since then it has been applied in many other areas such as industrial manufacturing. The first use of DTs in manufacturing was for various purposes such as improving product performance, flexibility, and competitiveness through simulation, data acquisition and communications, cyber-physical interaction, and advanced analytics [2]–[8]. DTs have been successfully used in the creation of cost- and time-effective designs of space vehicles by NASA [9] and in other areas of aviation [10], [11]. The use of DTs has also been applied to fields such as healthcare and telemedicine [12]–[14], smart homes [15], [16] and smart cities [17]–[22]. A review of DT applications in the industry is given in [23], a comprehensive survey of DTs in the IoT field is provided in [24], and [25] summarizes the existing work on applications of DTs in smart cities and DT-based smart city projects.

Although the "DT" terminology has been widely used in many applications, its use in networking scenarios is at a relatively early stage. Since the inception of the DT concept, the DT functionality has evolved along with advances in various technologies including networking techniques, as summarized in [10], [26], [27]. However, the implementation of fully functional DTs for complex physical objects or systems requires further investigation. In [4], DT was defined as a "rich representations of products that are virtually indistinguishable from their physical counterparts". Although achieving this goal is difficult, if not impossible, in most cases, a fully functional DT implementation should replicate (in software) the key aspects of its physical counterpart such as in temporal, spatial, and functional dimensions. In other words, a DT should be capable of reflecting all the features of the physical system that are required by an application. To achieve this goal, networking support is critical for future DT implementations.

In the early days, DT creation and maintenance were often limited by the capabilities of data transmission, processing, and storage [28], [29]. Early DTs, for example, may consist of an offline computer model/simulator that is used to predict the behaviour of the physical system as its inputs evolved [4]. Data collected from the real system would be manually provided to the digital (or virtual) system so that the state of the digital system can reflect the real physical system. This offline mechanism obviously limits the ability of the DT to accurately reflect the current state of the physical system. With the advancement of sensing, communications, and networking technologies, large volumes of data can be collected automatically and exchanged between physical systems and the corresponding digital systems with a low latency [3], [8], [30], [31]. The real-time bidirectional information exchange and state update between a physical system and its corresponding digital system is referred to as "synchronization" between the two.

To realize the PS-DT synchronization, there have been different proposals for DT architectural frameworks based on targeted applications and intended functionalities [10], [24], [26]. However, the study on DT framework design to achieve a desired level of performance for a given application is still in its infancy stage. In this work, we present a comprehensive review of DT from a networking perspective. This paper is different from [27], which summarizes the existing definitions, characteristics, current applications, and general supporting technologies for DT networks. DT implementation issues are also described in [24], although it emphasizes software aspects rather than networking ones.

The rest of the paper is organized as follows. The existing

Mehrad Vaezi, Kiana Noroozi, Terence D. Todd, Dongmei Zhao, and Huaqing Wu are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada. e-mail: {vaezim, noroozik, todd, dzhao, wu482}@mcmaster.ca.

George Karakostas is with the Department of Computing and Software, McMaster University, Hamilton, ON L8S 4K1, Canada. e-mail: karakos@mcmaster.ca.

Xuemin (Sherman) Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. e-mail: sshen@uwaterloo.ca.

definitions of DTs are briefly reviewed, and a feature-based method for describing DT quality is provided in Section II. The desired DT properties are discussed in Section III with an emphasis on their relationship to networking support. Section IV describes possible architectures for placing a DT in the network and Section V discusses the benefits of using DTs in different applications. A stage-based implementation framework is presented in Section VI for creating DTs of large and complex physical systems, and several implementation examples are provided in the same section. A summary of the recent networking-based research work on DTs is presented in Section VII, and some networking-related open research problems and challenging issues are listed in Section VIII. Section IX concludes the paper.

II. DT Definitions and Features

In this section, we first summarize the DT definitions from the literature, while emphasizing the importance of networking support in building and running DTs. We then introduce features that describe quality of DTs and discuss factors that affect quality of DTs.

A. Summary of Existing DT Definitions

When the concept of DT was first introduced by Michael Grieves for Product Lifecycle Management (PLM) in 2003, it was intended to be a “digital equivalent to a physical product” [4]. Using mainly manual data synchronization delivery and offline modeling, real-time electronic communications and networking were not widely used to build DTs. In 2011, the DT concept was used to create virtual sensors for predicting aircraft structural lifetime under various flight conditions [32]. This “digital representation” incorporated some important properties, e.g., structural defects or projected lifetime of the actual physical product, i.e., the aircraft. Without real-time synchronization between the physical and digital objects, DTs were mainly used for fault diagnosis, predictive maintenance, and performance analysis [33].

In 2012, a DT framework was defined to include three elements [9], i.e., the physical product, virtual (or digital) product, and the linkage between them. This linkage includes two-way interactions between the digital and physical products. The work also emphasized the benefits of such interactions, i.e., “the physical product can be made more intelligent to actively adjust its real-time behavior according to the recommendations made by the virtual product” and “the virtual product can be made more factual to accurately reflect the real-world state of the physical product”. This idea defines the mutual interactions between the physical and digital products, which implies that real-time communications are required in DT implementations.

Building the virtual or digital product of a physical product often exploits mathematical modeling, computer simulation, and other forms of data processing. In 2013, a virtual representation of the system was defined as “an integrated system of data, models, and analysis tools applied over the entire life cycle” of a product so that it can be used throughout the lifetime of the product [34]. By connecting this definition with that in [9], the importance of networking support for DTs became more prominent. This is because real-time interactions between the physical and digital products may require significant resources for communications, computations, and data storage.

In 2014, Michael Grieves formally introduced DTs in a whitepaper [4], where the DT was defined to include the same three elements as in [9], and the digital systems should be “virtually indistinguishable from their physical counterparts”. Although achieving such virtual products was impractical in many cases, various efforts have been made since then using real-time synchronization between the physical and virtual products, which help achieve this objective [6], [35].

As the DT concept was introduced into a wider range of industrial applications [36]–[38], the concept has also been extended to more general physical objects. In [33], a DT is defined as “a living model of the physical asset or system, which continually adapts to operational changes based on the collected online data and information, and can forecast the future of the corresponding physical counterpart”. According to this DT definition, a physical object and its performance can be mirrored by its virtual counterpart, which can predict potential issues and the remaining lifetime of the physical object. In other words, the physical and virtual systems should influence and synchronize with each other continuously and evolve together. The idea of mutual influence and synchronization between the physical system and the DT has been coupled with the Internet of Things for efficient sensing, data collection, and data processing [24], [33].

With all the previous DT definitions, reference [24] defined a consolidated DT as “the constant entanglement between an artifact (the PO) and its software representations (the LO)”, where PO is the physical object or system, and LO is the logical or virtual object of the PO. The “constant entanglement” is achieved through the communication linkage between the physical and virtual objects.

Consistent with the consolidated DT definition, reference [23] discusses the fundamental parts of DT modeling, including physical, virtual, and connection aspects. Physical modeling deals with issues surrounding the definition, description, and extraction of key features of the physical system. Virtual modeling builds a representation of the physical system based on these features and on the behavior data provided by the physical model. The connection model is the linkage between the physical and virtual parts which supports a frequent transfer of data between the physical and the virtual systems. This real-time data synchronization allows the DT to reflect the true status and behaviour of the physical system and represent the physical system to interact with third-party applications.

DT functionality has evolved from a “digital representation” of a static physical product to one that continuously evolves with the latter. This trend has increased importance of networking communications. The implementation of DTs often requires multiple types of network resources, e.g., computation, communication, and storage resources. Both physical and virtual modeling consumes computation resources, and the computation resource requirement increases with the increasing complexity of many modern systems. The synchronization between the DT and its physical counterpart is typically realized by communication links using network connections. In addition, communications are required to support the interactions between the DT and third-party applications. DTs may also have significant data storage requirements. In addition to storing data for modeling the physical system, the DT may also maintain historical data of the physical system for long-term behavior modeling and future state prediction [7], [32], [33], [39]. Therefore, the joint allocation of multiple types of network resources [40]–[42] is an important issue for successfully implementing and maintaining DTs.

B. DT Features

The existing DT definitions as summarized above are all descriptive, which is not convenient for evaluating quality of the DTs or analyzing their performance. In this section we describe a feature-based method that provides a consistent framework for evaluating DT quality in different situations.

Consider a physical system (PS), and its state at time t is denoted as $\mathrm{s}(t)$. Sometimes one may be interested in a particular subset of the system state. Therefore, we define multiple types of system state, $s_k(t), \forall k \in \mathcal{K}$, where $\mathcal{K}$ is a set of different state types. Note that different subsets of system states may have common system state types depending on the particular focus.

Various *features* that reflect the PS behavior can be derived from the system state. For example, a feature for a temperature sensor could provide an average of temperature readings taken over the past 24 hours. Let $i \in \mathcal{I}$ represent the ith defined feature of the PS, and $f_i(t, \mathrm{s}(t_s))$ represent the feature i output at time t based on the PS state at time t_s. In the temperature sensor case, this feature at time t would provide the average temperature over the one-day period prior to time t_s. The distinction between times t and t_s in $f_i(t, \mathrm{s}(t_s))$ provides for the case where a feature based on the system state at time t_s requires computational processing, and as a result, the feature is not available until time t, where $t - t_s$ is the processing latency. This definition also applies to the DTs, where a PS feature provided by a DT at time t may be based on a past PS state at time t_s. In this case, the time difference may also include both communications latency between the PS and DT and feature processing time at the DT.

In a more general case, for example, a substate of a manufacturing plant could include temperature, humidity, noise, etc., at different locations. Based on this, features such as whether a given machine is within a safe working condition can be provided. The PS is characterized by all its features as $\mathbf{f}(t) = [f_i(t, \mathrm{s}(t_s(i))), i \in \mathcal{I}]$. In this case, the ith feature at time t may have used state information at different times, $t_s(i)$.

A DT is intended to reflect the state and actions of its PS as closely as needed. As discussed above, because of latencies caused by communications and data processing, the current state at the DT may not be the same as that at the PS. That is, if we define $\tilde{\mathrm{s}}(t)$ to be the PS state available at time t at the DT, then typically $\tilde{\mathrm{s}}(t) \neq \mathrm{s}(t)$. For the same reasons, features offered at the DT on behalf of the PS will often not exactly reflect the current PS features. Let Δt be the age of the PS state information at the DT, i.e., $\tilde{\mathrm{s}}(t) = \mathrm{s}(t - \Delta t)$. We therefore define the features offered by the DT as $\tilde{\mathbf{f}}(t) = \left[\tilde{f}_i(t, \tilde{\mathrm{s}}(t_s)), i \in \mathcal{I}\right]$, where $\tilde{f}_i(t, \tilde{\mathrm{s}}(t_s))$ is the ith feature of the PS reflected at the DT at time t, and this feature information is based on the received and processed data at the DT at time t_s. Although the state at the PS changes in continuous time, the update of information at the DT may be initiated at discrete time instants.

Ideally, a feature at the DT is an exact replica of that at the PS. That means that the difference between $f_i(t, \mathrm{s}(t_s))$ and $\tilde{f}_i(t, \tilde{\mathrm{s}}(t_d))$ should be zero, i.e.,

$$f_i(t, \mathrm{s}(t_s)) = \tilde{f}_i(t, \tilde{\mathrm{s}}(t_d)) \qquad (1)$$

$\forall\, t \geq 0, i \in \mathcal{I}$, where t_s and t_d are the state information times used to create the feature at the PS and DT, respectively. Note that $t_s \leq t_d$ since the state information available at the DT is at best a delayed version of that at the PS. Equation (1) simply states that this delay has no effect under ideal conditions.

In a practical application, achieving a DT that exactly reflects the PS features at all times as defined by (1) is often difficult for a variety of reasons, such as

- the state information at the DT is normally obtained by synchronizing with the PS, which often happens periodically at discrete time instants;
- the features available at the DT incur processing delays at the DT; and
- the number of features is too large for the DT due to limited resources for data storage and processing.

Depending on its intended use, a DT can be deployed to represent only a subset of the features of the PS. A DT implementation is considered *fully functional*, if it implements all PS features with an acceptable level of deviation.

III. DT Properties

In this section, we review some specific DT properties that are dependent on networking support, including promptness, similarity, reliability, composability, scalability, and flexibility (or adaptability). Although these properties have been introduced and discussed before [3], [24], they will be described in a manner that incorporates the DT feature definition in Subsection II-B.

Promptness. Promptness assesses the value of a feature from an application viewpoint as the age of information (AoI) used to generate that feature changes. Consider a given PS feature i and time value $t_s \leq t$. If

$$d_p(f_i(t, \mathrm{s}(t_s))) \leq \epsilon_{p,i}, \qquad (2)$$

we say that an application's promptness measure of feature i (defined by $d_p(\cdot)$) is $\epsilon_{p,i}$-conformant for the given time parameters. In this expression, the function $d_p(\cdot)$ assesses the quality of feature i at time t using the PS state information at time t_s, i.e., $\mathrm{s}(t_s)$ was used to generate the feature. If, for example, the importance to an application were the time since a feature was created, the function $d_p(\cdot)$ defines the level of acceptability associated with the feature's age after a time period $t - t_s$. In general, $d_p(\cdot)$ is a non-decreasing function of $t - t_s$. When a feature's age expires at some time threshold, $d_p(\cdot)$ may exceed $\epsilon_{p,i}$, indicating the unacceptability of the feature. In a DT implementation, the state information will typically incur latencies due to the state synchronization process. A feature provided by the DT may therefore be a delayed version that is available from the PS directly. The promptness assesses the importance of this latency offset to a given application.

Fig. 1 shows the synchronization delay of information from a PS to its DT. In this example we assume that data synchronization between the PS and the DT happens periodically every T seconds. Consider the system state data generated at time t_s, and define τ_{PL} as the data transmission delay from the PS to the DT. If the communication channel capacity between the PS and the DT is sufficient, such that all the state data generated within one T interval is delivered to the DT before the end of the same interval, then the AoI of the state data at the DT is at most $T + \tau_{PL}$. Upon receiving the state data of the PS, the DT may have to process the data and extract various features as would be the case at the PS without the DT. The data processing may include data cleaning, classification, modeling, etc. Let C_i be the data processing time for feature i at the DT, then $\Delta t_i = T + \tau_{PL} + C_i$ is the maximum AoI at the DT for feature i. The DT is responsible for interacting with applications on behalf of the PS. Defining $\tau_{LA,j}$ to be the communication delay from the DT to application j, then the maximum AoI of feature i at the application is $\Delta t_i + \tau_{LA,j} = T + \tau_{PL} + C_i + \tau_{LA,j}$, which is also the upper bound of $t - t_s$. Let Δt^*_{ij} be the AoI requirement of application j for feature i of the given PS, then $\Delta t_i + \tau_{LA,j} \le \Delta t^*_{ij}$ should be satisfied for all $j \in \mathcal{J}$, where $\mathcal{J}$ is a set of the applications that depend on the DT to obtain information of feature i of the PS. If a DT is responsible for interacting with multiple applications on behalf of the PS, the AoI at the DT should satisfy the requirement of the application with the most stringent promptness requirement. That is, $\Delta t_i \le \min_{j\in\mathcal{J}}(\Delta t^*_{ij} - \tau_{LA,j})$. Based on this, we can define $\Delta t^*_i = \min_{j\in\mathcal{J}}(\Delta t^*_{ij} - \tau_{LA,j})$ as the AoI threshold of feature i of the DT for supporting the applications in $\mathcal{J}$.

Different applications may have different tolerance about the AoI. As an example, when $d_p(f_i(t, \mathrm{s}(t_s)))$ is defined as

$$d_p(f_i(t, \mathrm{s}(t_s))) = \begin{cases} 0, & \text{if } t - t_s \le \Delta t^*_{ij} \\ \infty, & \text{otherwise} \end{cases} \tag{3}$$

the feature i information provided by the DT is either acceptable or unacceptable at application j.

Note that the promptness requirement for a DT is not only determined by the application requirements but also dependent on the dynamics of the PS. More specifically, if the evolution of the system state in the real system is more abrupt and random, then the state data should be updated more frequently, i.e., the update period T should be smaller; meanwhile, the values of τ_{PL}, C_i, and $\tau_{LA,j}$ should also be smaller in order to reduce the AoI at the application.

Similarity. In contrast to promptness, similarity considers the quality of features from a functional viewpoint. For example, given a feature i, if for any $t_s \le t$,

$$d_s(f_i(t, \mathrm{s}(t_s)), \tilde{f}_i(t, \mathrm{s}(t_s)) \le \epsilon_{s,i}, \tag{4}$$

then $\epsilon_{s,i}$ sets the similarity of feature i for the given set of times. As discussed above, $\tilde{f}_i$ is the feature provided by the DT at time t. Note that if the same state information, i.e., $\mathrm{s}(t_s)$, is used by the DT and PS, and the mechanism used to create the feature at the PS and the DT is identical, then the two terms in the argument of d_s will be equal, and $d_s(f_i(t, \mathrm{s}(t_s)), \tilde{f}_i(t, \mathrm{s}(t_s)) = 0$. The reason that they are not the same may be that the DT uses a "noisy" version of PS state information or that the feature generation at the DT is different than it would be at the PS. Note that the similarity is the highest when $\epsilon_{s,i} = 0$. In this case, the DT provides identical performance from the application viewpoint.

Suppose for a given application, the required similarity for feature i is $\epsilon^*_{s,i}$, where $i \in \mathcal{I}$, and define $\boldsymbol{\epsilon}^*_s = [\epsilon^*_{s,i},\ i \in \mathcal{I}]$. The level of $\boldsymbol{\epsilon}^*_s$-similarity of a DT can be defined as the fraction of the features of the PS that have a similarity level of at least $\epsilon_{s,i}$, or

$$S(\boldsymbol{\epsilon}^*_s) = \frac{\left|\tilde{\mathcal{I}}\right|}{|\mathcal{I}|}, \tag{5}$$

where $\tilde{\mathcal{I}} = \{i \in \mathcal{I}, |\ (4) \text{ holds when } \epsilon_{s,i} = \epsilon^*_{s,i}\}$. Based on this definition, $S(\boldsymbol{\epsilon}^*_s) = 1$ if $\epsilon_{s,i} \le \epsilon^*_{s,i}$ and (4) holds for all $i \in \mathcal{I}$.

The similarity level as defined in (5) can be further extended by considering the importance of multiple features to a given application. For example, if w_{ij} represents the weight or importance of feature i to application j, then the level of $\boldsymbol{\epsilon}^*_s$-similarity of the DT to the application can be defined as

$$S_j(\boldsymbol{\epsilon}^*_s) = \frac{\sum_{i\in\tilde{\mathcal{I}}} w_{ij} I_i}{\sum_{i\in\mathcal{I}} w_{ij} I_i}, \tag{6}$$

where I_i is an indicator function with $I_i = 1$ if (4) holds when $\epsilon_{s,i} = \epsilon^*_{s,i}$ and $I_i = 0$ otherwise. Based on this definition, the same DT may have different similarity levels for different applications.

Both higher promptness and higher similarity require more network resource support, since system states must be more frequently measured, transmitted and processed, and features of the PS should be extracted with higher accuracy and lower delay. In practical implementation, since PSs vary from one another in terms of their statuses and conditions, the featured data required to characterize their status and behaviors are

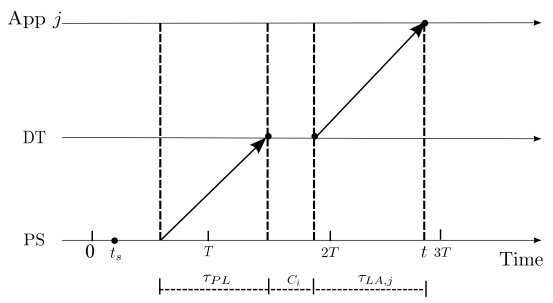

Fig. 1. AoI at DT and application

different. Therefore, the data collection and processing can be tailored for different PSs and applications to minimize resource consumption. For example, for static or low-mobility users, location information can be collected and processed at a low rate, reducing communication and computing resource consumption. On the other hand, since different types of network resources jointly affect promptness and similarity, it is possible to trade off the resource usage while maintaining an acceptable level of promptness and similarity. For example, when the network is congested, and communications between the PS and DT experience a longer delay, more computation resources can be allocated at the DT to reduce the data processing delay.

Composability. Composability is about integrating the DTs of smaller systems into a DT of a larger system or merging multiple DTs that reflect different features of the same system into a new DT [43]–[46]. Let $\mathcal{K} = \cup_{n=1}^{N} \mathcal{K}_n$, where $\mathcal{K}_n$ is the nth set of system states. For a practical system, each $\mathcal{K}_n$ can include all states of a subsystem, a subset of the states of the entire system, or a subset of the states of a subsystem. Integrating DTs may be needed in different scenarios. Scenario 1: DTs may be created for relatively independent PSs, which later on become subsystems of a larger system. Scenario 2: multiple DTs are created separately to support different applications, and the system states used by the DTs are not the same. At some point, it is found that in order to support an application, a DT should be created to use the system states that have already been used in the multiple existing DTs. In both scenarios, the ability of integrating the existing DTs helps save both time and cost.

A typical example for scenario 1 is for geographically distributed PSs, which have components or subsystems in different geographical areas. It is natural to treat each component or subsystem, such as a department in an organization or a warehouse in a store. As the inter-department businesses increase, integrating the department DTs into an organization DT helps improve the work efficiency. Similarly, by integrating the DTs of individual warehouses into a DT for the entire store, the warehouse resources may be better utilized. Another example is where multiple manufacturers along the same supply chain have increasingly more dependence on one another and want to jointly optimize their resource allocations by merging their individual DTs into one DT. For scenario 2, we use the connected vehicle network as an example. For applications such as collision avoidance and road navigation, a DT (DT 1) can be created by collecting the system states related to locations, moving speeds and direction, etc., of vehicles. Another DT (DT 2) can be created to support communication network resource allocations for entertainment applications, which require system states related to types of entertainment applications, time of using the applications, etc. Later on, it is found that DT 2 may work more efficiently in network resource allocations if the system states related to vehicle locations, moving speeds and directions are also considered.

When integrating multiple existing DTs into a new one, the mutual interactions or dependency of the corresponding physical objects should be taken into consideration and redundant information that is available in multiple existing DTs should be removed. This requires additional network resources for both data communication and data processing.

Scalability. Scalability is about the increase in complexity in building and maintaining a DT when the number of the system states (i.e., $|\mathcal{K}|$) increases. As a simple example, if the PS includes N devices, K state types from each device, and each state type may take V different values, then the total number of system states is V^{NK}. Therefore, the number of system states increases exponentially with the number of devices and state types. The PS size, in terms of number of components or types of data to be collected, may increase with time as the system expands. Given this, one goal in the DT designs is that the amount of network resources required does not increase exponentially as the system size increases.

Scalability is also about the quality of the DT when the PS extends in geographical area. In this case, the communication quality, e.g., communication delay, tends to deteriorate as the distance increases. How to maintain good quality DTs in this case can be a challenging issue.

Reliability. Since a DT works as a replica of its PS, its reliability is important to both its PS and the applications. The reliability includes both data reliability and operation reliability. The data reliability refers to the completeness, accuracy and consistency of the data of the physical system states available at the DT, and the operation reliability is about how well the DT can support the applications.

As discussed in Section III, system features that are offered by the DT may not exactly reflect that of the original PS due to a lack of *similarity*. This can be caused, for example, by inaccurate state information transmitted during the PS-DT synchronization process. An example of this is where PS state information is compressed before synchronizing with the DT. Therefore, the level of data reliability largely depends on the quality and quantity of data transmitted during synchronization. In general, higher data reliability makes it possible to achieve higher operational reliability. However, achieving higher data reliability requires more network and computational resources. The reliability required by the applications can be used to determine the required level of data reliability at the DT and plan the network resource allocation.

In practical scenarios, random channel conditions, network congestion, and unexpected link and/or server failures, can all affect the data reliability and the operation reliability. Achieving higher reliability may require additional network resources, such as backup communication links, servers, etc. When network resources are limited, there may be a tradeoff between reliability and promptness/similarity. In this case, the achieved promptness and/or similarity may have to be reduced for some features.

Flexibility (adaptability). Being a replica of its physical counterpart, a DT should adaptively reflect changes in the PS, including not only state changes, but also hardware and software upgrades, changes to the operation modes, etc. This may result in an adjustment to data collection, e.g., types of the system states and frequencies at which different types of state data are sensed. This can change the mathematical modeling, computer simulations, etc., and further affect the requirements on the networking resources. For example, for building a DT of a school, changing from in-person exams to online exams at the school may involve changes to data collection, data processing, and data storage partially because of the significant differences between online and in-person proctoring [47]–[49]. Achieving smooth transitions for the DT to adapt to changes in the PS may require the designers to know possible future directions of the PS and plan the resource deployment and allocation accordingly, which can be difficult in practical applications. In terms of network resource allocations, two traditional ways for accommodating possible changes are resource reservation and prioritization. When applied to supporting DTs, the former is always reserving a certain amount of resources for a given DT, regardless of its current requirement. The reserved resources help absorb some changes to the resource requirements when the PS experiences changes. However, this can result in significant waste in resource utilization. In contrast, prioritization means that DTs requiring higher quality are given a higher priority to use the network resources when the network does not have sufficient resources to support all DTs, and this is at the price of sacrificing the performance of DTs with low priorities.

IV. DT Placement Architectures

A DT is a digital replica of its PS. Depending on the structure of the PS, the application requirements, and the network resource availability, the DT of a PS can be hosted at a single server, distributed among multiple servers, organized in a primary-secondary architecture, or formed by a cybertwin or a network of DTs of the individual components/subsystems of the PS.

A. *Centralized DT*

In this case, the DT is assigned to and executed on a single server. When the PS is geographically small, the DT may be placed at an edge server adjacent to the same network as the PS. This will help to reduce latencies by minimizing the path lengths needed for communication. The quality of the DT will be affected by the conditions within the network and the capacity of the edge server that hosts it. Since the PS and the DT communicate through the same local area network, this setting is especially suitable when the DT interacts solely with its PS, or it interacts with third-party applications through the same network. Examples of this type of DTs are described in [50] for small size electromechanical products, [51] for small-scale production lines, and [52] for individual network devices and servers.

Alternatively, the DT can be located at a remote server, in which case the communications latency between the PS and the DT may be much higher. The additional delay helps the PS to access a more powerful cloud server to host its DT. This helps reduce the computation time, which can be prohibitively long for complicated PSs if the DT is hosted by an edge server. Routing, traffic scheduling, and resource reservation can be made to minimize the end-to-end communication delay, such as the machine learning-based routing scheme with mobility prediction in [53] and the delay-aware routing and scheduling algorithm in [54].

B. *Distributed DT*

In contrast to the centralized DT case, the DT can be distributed across multiple servers, each hosting the digital representation of a subset of the PS features. This is a more suitable arrangement for large scale PSs that have many components or subsystems. For example, when building the DT of a network of connected vehicles, one server could be used to collect system state data in one region, such as a community or a city. Based on the collected data, features such as traffic density, possibility of traffic accidents, need for car maintenance, etc., can be extracted at the DT by different servers. Examples of this type of DTs also include the cyber-physical system in [55] and the edge computing networks in [56]. In [55] individual physical devices have their separated DTs, and in [56] each end device and edge server are individually twined in the virtual space.

Implementing the digital representation of the subsystems or reflecting the subsets of features of the PSs at different servers helps distribute network loads, which makes it possible to extract more accurate PS feature information at the DT. However, this requires the PS to communicate with multiple servers that host the DT, and may also require the servers to interact with each other when the features hosted at different servers have mutual dependencies. The information exchanges among the servers incur additional delays in the DT operation.

In addition, since the PS has to communicate with multiple servers that host the DT, quality of the DT is affected by the communication conditions between the PS and the multiple servers and the processing capabilities of these servers.

C. *Primary-Secondary DT*

A special case of the distributed DT implementation is organizing the DT operation in a hierarchical fashion. A primary DT component directly interacts and synchronizes with the PS. The primary DT component also interacts with one or multiple secondary DT components, which do not interact directly with the PS. Such a scheme was introduced in [24]. It has the advantages of a centralized DT implementation for the primary DT component, while it also provides the flexibility of a distributed DT implementation for the secondary DT components. However, it has the disadvantage of extra latencies due to the updating between the DT components.

D. *Network of DTs or DT Network*

As DTs proliferate, multiple DTs may communicate with each other, i.e., DTs of different PSs may interact on behalf of their respective PSs, which creates a network of DTs [57]–[59]. As an application, networks of DTs can be used to develop new network architectures and protocols. In [60], a cybertwin-based network architecture was proposed to replace the traditional end-to-end communication model with a cloud-to-end communication version. The *cybertwin* is similar to a DT and is a representation of a physical object, e.g., a person or a thing in the Internet of Things. Cybertwins “serve as a communications assistant, network data logger, and digital asset owner of humans and things” [60]. The proposed architecture aims to ensure that next generation networks are more flexible, scalable, reliable, and secure. In [61], cybertwins located at edge servers are used to collect user-level information, aggregate end user traffic into different service groups, and map quality of experience at the application level into the service-level QoS demands. This helps achieve fine grain resource orchestration for virtual networks and enhance quality of service (QoS) provisioning in sliced virtual networks.

The cybertwin or network of DTs can serve as a centralized space that collects, stores, and processes global information of the PS. In [62], a cybertwin is considered for a space-ground integrated network, where the cybertwin consists of a DT of a satellite and DTs of base stations and vehicles. With the mutual information exchanges among the DTs of the network components, all the DTs can obtain the global information of the network. One example application of the cybertwin is for individual BSs to make beamforming decisions that are optimum to the global network. A similar network of DTs is considered in [63] for an air-ground network. In this network of DTs, DTs of ground devices within a certain area are maintained at a central ground server, and a DT of a drone is maintained by having the drone contact the central server instead of individual devices in the ground network. This allows the DT of the drone to efficiently obtain the global network information. In [64], a DT network is created by integrating DTs of individual satellites in a network. The DT network reproduces the operation state of a satellite network, predicts the dynamic network topology, and performs virtual routing at the DT network before assigning it to the real network. A DT network can also play an important role in collaborative driving, which requires to make time sensitive decisions based on status of other vehicles. By creating DTs of individual vehicles, each vehicle only updates its information to its own DT. The DTs form a network that supports efficient information exchange among the DTs so that each DT can make a globally optimum driving decision for its vehicle. This helps reduce frequent information exchanges among autonomous vehicles and improve reliability and efficiency of autonomous driving [65] .

V. Benefits of using DTs

Although networking support is a necessary aspect for implementing DTs and DT-based applications, a network DT, which is a DT created as a digital replica of a real network, helps improve network performance in various ways. This section starts with discussing the general benefits of using DTs, then focuses on the benefits of using network DTs.

A. General applications

PS as an application. A DT can be used for management, optimization, and prediction for its own PS. The benefits of using DTs in this context include:

- With DTs emulating their PSs in dedicated servers, the design, operation, and maintenance of the PSs can be simplified, allowing for their real-time monitoring and control.
- Through smart analysis of data in DTs, the status of the PS can be predicted to enable more intelligent maintenance and avoid potential failures.

Third party applications. A DT can interact with applications on behalf of the PS. A DT may be especially useful when the PS information is required by multiple applications and/or when the direct interaction with the PS is costly/noisy. For example, in 5G networks, both handover and navigation tasks need to receive location-based information from the end users. To alleviate duplicated information transmission in wireless networks, the user location information can be retrieved from the DT instead of the PS. Furthermore, in handover management, considering that end users may have a high level of mobility and fluctuating channel conditions, making handover decisions based on the PS information may suffer from the ping-pong effect, which increases the numbers of handovers and reduces the quality of the mobile user's connection. When using a DT, user features (e.g., user's daily routine and mobility patterns) can be utilized to conduct user trajectory prediction, which facilitates more efficient handover decisions with reduced ping-pong effect and decreased handover resource consumption.

The main benefits of using DTs in this context include:

- When a user has multiple devices (e.g., mobile phones, tablets, wearable devices, etc.), these devices possess similar behavior patterns and can be characterized by the same user DT. On this account, when multiple devices interact with the same application, the required information can be retrieved from the user DT only once, which tremendously mitigates the network traffic burden.
- Powered with data processing capabilities, DTs can help reduce the noise from the data generated at the PS and filter out unnecessary data. Moreover, instead of utilizing the tremendous raw data, the user features and/or behavior patterns can be further extracted in DTs with reduced data dimension before forwarded to applications, leading to reduced network traffic load. These features/behavior patterns also provide better characterization of PSs, enabling more efficient, flexible, and precise resource management with enhanced resource utilization.
- DT implementation can be customized for different applications by constructing different DT components, each of which is tailored to one or multiple applications. This helps simplify the PS by outsourcing the application-related tasks to the DT.

Consider a supplier-distributor-retailer chain as an example. The suppliers manufacture goods, which are distributed by the distributors to the retailers. Customers order the goods through the retailers. A DT can be created for each retailer, so that the state of the retailer, such as information of goods provided from different distributors and information of orders placed by customers, is updated at the DT periodically. The state information is then processed at the DT to extract features such as the demands of different goods, which assist the retailer to make decisions about what goods and from which distributors it should purchase next. In addition, complicated learning models can be built in the DT so that it can use the historical data to predict future demands and supplies in order to optimize its long-term revenue. This is an example of using the DT to optimize the performance of its own PS (i.e., the retailer). Meanwhile, the DTs of the retailers can interact with different distributors by providing retailing information of goods from the individual distributors. In such interactions, the DTs of the retailers can help optimize operations of the distributors, which are considered as third party applications. In addition, the individual distributors and suppliers may each build their own DTs, and the DTs of the suppliers, distributors, and retailers can be integrated and form a distributed DT of the full supply chain.

B. DT of Networks

In its initial use cases, a DT was meant to span the entire life-cycle of a product, from its design to manufacturing and eventually to its disposal stage [1], [4], [66]. When a DT is created for a network, it should not only reflect the network features, but also evolve with the real network in terms of both software and hardware upgrades, until the disposal of the network. During this process, the DT of a network helps optimize network operations and management. As an example, using DTs can provide the network orchestrator with a reliable simulation environment, which is especially important before applying large-scale decisions on the actual physical network. As a direct use case of building DTs, the virtual model of a real network can be duplicated and placed inside routers. This provides the routers with the current states of the surrounding nodes and helps them make better routing and forwarding decisions [67]. By analyzing the past and current work conditions, diagnosing potential issues, predicting the future performance, and helping make decisions on changes to network deployment, operations, and other aspects, the DT of a network not only influences the network operations in real-time, but also evolves with the real network throughout its lifetime. With all the network entities mapped to the cyber space using DTs, communications and applications can be provided across the physical and cyber spaces and more services can be provided across the physical and cyber spaces. For example, an end-to-end connection in the physical space can be replaced with an end-DT-DT-end connection. Such cross-space services benefit from the use of DTs and the cross-space resource management [68].

Using the DT of networks is a natural way to achieve software defined networks (SDNs). The basic principle of software defined networking is to decouple the network control and the data forwarding functions, making the network more suitable for dynamic computing and storage needs in various application scenarios. The network control is directly programmable and centrally managed through a software-based SDN controller. The central controller has a global view of the network operation by collecting the status of network devices and other operational data. With powerful computation power, the controller can run complicated learning algorithms to optimize routing decisions, balance traffic load, and help the network adapt to dynamically varying topology, traffic and propagation conditions. Although the current SDNs or SDNs with artificial intelligence also make control decisions based on present and past collected data, creating a DT of an SDN allows the network to benefit from the intelligence built in the DT. For example, possible changes in the network architecture can be applied first to the DT in order to observe the eventual outcome caused by these changes. This helps network operators skip through the transient states and plan the network operations more effectively. Performance of any new function models, such as routing strategies or load balancing criteria, can be simulated at the DT to examine the performance of such models based on possible behaviors of the network components. This not only helps predict the network performance but also diagnoses potential problems. In addition, the prediction results can be used to influence the network evolvement, such as software and hardware upgrade, resource deployment and resource allocations [69]. In addition, the SDN controller can work with DTs of individual network elements, such as network users, user devices, and communication channels, to optimize the network operations [56], [70]–[72].

DT and network slicing are the two key elements that form the holistic network virtualization for 6G [73]. By creating DTs of end users, extensive historical and real-time data can be collected that characterize the service demands and QoS satisfaction of the users. With the powerful processing capability of the DTs to emulate complicated network management strategies, adaptive and customizable resource allocations can be achieved for network slices based on various user and network conditions.

VI. Stage-Based DT Implementations

A DT may need to operate on many different levels of granularity [74] in the temporal, spatial, and functional dimensions. This can be difficult to enforce for large and complicated systems that have high device heterogeneity, cover a wide geographical area, and support diverse applications. In this case, the number of system states needed to represent the system condition, and the number of features needed to support the applications may be extremely large. In addition, the timing constraints imposed on the DT when producing an accurate reflection of the PS features, or when supporting applications, may vary from very loose to very tight. The amount of networking resources required for building DTs of such PSs can also be prohibitively high. As a result, the straightforward implementation of a fully functional DT of a complex PS operating within a complex interactive environment may be extremely hard or even impossible. Therefore, incremental transitional stages of a DT implementation are necessary and practical, allowing some applications to take advantage of the (partially functional) DT. Depending on the physical system and applications to be supported, the transition can happen in different ways. In this section, we first present the general ideas for stage-based DT implementations, based on which a five-stage implementation framework is described. The implementation ideas for two specific PSs, a space-air-ground integrated network and a network of connected vehicles, are then provided. Finally, the generic implementation framework is integrated with two existing DT structures.

A. General Ideas

A stage-based DT implementation starts from implementing a partially-functional DT based on the DT features given in Subsection II-B and the properties defined in Section III. Multiple partially-functional DTs are integrated to reflect more complete behavior of the PS and support more applications.

The stage-based implementation of a DT can evolve along the following paths:

- In the temporal dimension, the DT implementation can start from reflecting features with relatively loose promptness, i.e., larger Δt_i^*. In this case, the data sensing and collecting frequency, the PS-DT and DT-application communication data rates, and/or the data processing speed at the DT can be relatively lower. This helps reduce the amount of communication and computation resources needed to implement the DT.
- In the functional dimension, the DT implementation can start from smaller subsets of features and gradually integrate more features, i.e., from smaller $|\tilde{\mathcal{I}}|$ to larger $|\tilde{\mathcal{I}}|$ until $\tilde{\mathcal{I}} = \mathcal{I}$.
- In the spatial dimension, DTs can be implemented for smaller sets of $\mathcal{K}_n$'s, and then integrated gradually, until the DT based on the full set of $\mathcal{K}$ is implemented.

The three implementation dimensions are not independent of one another, and their combinations also depend on the application requirements. For example, if a DT of a city is to be built for infrastructure planning, the designer should first identify what features of the city are needed (such as energy consumption per capita, household income, etc.). Based on the analysis of required features, the system states can be incrementally determined; the incremental implementation of the DT is allowed by a sufficient relaxation of requirements imposed by the application. Note that, in this particular case, the functional and spatial dimensions are directly dependent. Another example is collision avoidance in vehicular networks. Collision avoidance requires real-time information, such as the location, moving speed, and moving direction of vehicles within a limited geographic area. These requirements can be satisfied by a bare-bones implementation of the DT, and thus, collision avoidance can be served by the DT during all stages of its implementation, and not necessarily by only the (final) fully-functional DT implementation.

Several implementation examples are given in Table I. For the smart city DTs, the system state includes many types of data from a large number of system components distributed across large geographical areas. The current implementations of smart city DTs, such as [75]–[80], focus on some specific applications, such as energy efficiency [75], [77], infrastructure planning [76], [78], [79], and disaster management [80], which are all time insensitive. For supporting these applications, the delay for collecting data from a large scale system and the time needed to process the large amount of data from different sources is less of a concern, compared to the accuracy of the model when it is used for predicting future state of the city.

On the other hand, the DT applications for smart manufacturing, such as product design, production planning, fault diagnosis, and predictive maintenance [50], [51], [81]–[85], are designed for small scale PSs. The DTs can be positioned close to the PSs, which reduces the data transmission delay between the PS and the DT. Also, the volume of data is typically small for small scale PSs, which leads to reasonably small data processing delays at the DTs. Therefore, tight synchronization can be achieved between the DTs and their PSs, which is important to optimize manufacturing operations. Similarly, the DTs used in mobile computation offloading are also for small size PSs, such as end devices [52], [56], [86]–[88], base stations and edge servers [70], [71]. These DTs are used for supporting computation offloading decisions, resource allocations, and secure and reliable computation in the edge networks, which require short delay in making the decisions.

In the functional dimension, the number of features extracted from the current smart city DTs is a small portion of the features that can possibly be extracted due to the complicated nature of the PSs. For the smart manufacturing and edge computation network examples, the current implementation examples emphasize extracting features of the PSs that help improve or optimize the short-term PS operation, while more features can possibly be extracted in future DT implementations for similar PSs. For example, in edge computing networks, the current emphasis is to make better computation offloading decisions, which involves features of the networks such as device mobility, traffic conditions, channel variations, etc. Additional features, such as server lifetimes, satisfaction level of end users, etc., may be extracted to help network owners consider changes to resource deployments and long-term resource allocation policies.

TABLE I
DT IMPLEMENTATION EXAMPLES

PS DT example	Temporal Δt_i^*	Functional $\lvert\tilde{\mathcal{I}}\rvert$	Spatial $\lvert\tilde{\mathcal{K}}_n\rvert$
Smart city [75]–[80]	Large	Small	Large
Edge network [52], [56], [70], [71], [86]–[89]	Small	Small - medium	Small
Smart manufacture [50], [51], [81]–[85]	Small	Small - medium	Small

A stage-based implementation of a DT also provides designers with time and opportunities to better understand the physical system and/or application requirements, such as the types of states needed to accurately represent the system and the features used by applications.

B. Five-Stage Implementation Framework

Following the ideas in the previous subsection, we present one specific framework for the stage-based DT implementation for PSs that span large geographical areas. In the spatial dimension, we divide the types of the system states into the component level, subsystem level, and global level, and use sets $\mathcal{K}^C$, $\mathcal{K}^S$, and $\mathcal{K}^G$, respectively, to represent the types of system states at these levels. For example, in a connected metropolitan vehicular network, the moving speed of a car is at the component level, while its departure and destination locations may belong to either a subsystem or at the global level. At each spatial level, the implementation is divided into stages that start from a low promptness level for a subset of the features, and gradually transit to achieving the target promptness of all features. This framework is given in Table II. It includes five stages, and is explained below.

TABLE II
EXAMPLES OF STAGE-BASED DT IMPLEMENTATIONS

Stages	Components $k \in \mathcal{K}^C$	Subsystems $k \in \mathcal{K}^S$	Global system $k \in \mathcal{K}^G$
Stage 1	$\Delta t_i = \infty, \forall i \in \mathcal{I}$		
Stage 2	$\Delta t_i^* < \Delta t_i \leq \infty$ for $i \in \tilde{\mathcal{I}} \subseteq \mathcal{I}$	$\Delta t_i = \infty$ $\forall i \in \mathcal{I}$	
Stage 3	$\Delta t_i \leq \Delta t_i^*$ $\forall i \in \mathcal{I}$	$\Delta t_i^* < \Delta t_i < \infty$ for $i \in \tilde{\mathcal{I}} \subseteq \mathcal{I}$	$\Delta t_i = \infty$ $\forall i \in \mathcal{I}$
Stage 4	$\Delta t_i \leq \Delta t_i^*$ $\forall i \in \mathcal{I}$		$\Delta t_i^* < \Delta t_i < \infty$ for $i \in \tilde{\mathcal{I}} \subseteq \mathcal{I}$
Stage 5	$\Delta t_i \leq \Delta t_i^*, \forall i \in \mathcal{I}$		

From the second to the fourth columns, each represents a level of physical (or spatial) size.

- The component level represents the lowest spatial size. Higher promptness should first be achieved at this level, since the volume of data sensed at the same components is relatively small and can be collected and processed at an edge server with shorter communication and computation delay.
- At the subsystem level, a larger volume of data should be collected and processed as the number of components in a subsystem increases; furthermore, the mutual dependence of components in the subsystem should also be reflected in the digital representation. As the size of the subsystem increases in terms of both the number of components and the geographical space, the complexity to model the system increases. In addition, the larger communication delay and delay variation between individual components and the DT makes it difficult to achieve a high level of promptness.
- The third column represents the entire system that integrates all the subsystems and components. The complexity for achieving high promptness at this level for a large system can be prohibitively high.

Each row in Table II represents an implementation stage that is represented by the level of promptness and similarity of the DT.

- In stage 1, DTs do not exist, and there are only physical systems at all levels in the spatial dimension.
- In stage 2, a certain level of synchronization is achieved at the component level but the promptness is still below the target for some or all the features; and there is no synchronization at the subsystem and global levels in the spatial dimension. Depending on the volume of data and the availability of communication bandwidth and computing resources, this stage may be omitted.
- In stage 3, the target promptness is achieved for all the features at the component level; a certain level of synchronization is achieved for some but not all the features at the subsystem level, and not all features have achieved the target promptness. There is no synchronization at the global level.
- In stage 4, the target promptness is achieved for all the features at the component and subsystem levels. At the global level, not all the features have achieved their target promptness.
- In stage 5, a fully functional DT is finally implemented. All features have achieved their target promptness at all spatial levels.

This stage-based implementation includes system decomposition and DT integration. Decomposition refers to the division of the physical system into a series of consecutively finer (smaller) subsystems, all the way down to individual components. Integration refers to the creation and consecutive integration into ever larger subsystem-level DTs, starting with individual component-level DTs, and going all the way up to a fully-functional DT for the full system.

The division of a system (or subsystem) into smaller subsystems can be performed in various ways. Some systems can be naturally divided into relatively independent subsystems based on geographical locations or functionalities. For example, a city can be divided into different communities, a university consists of multiple campuses, and a company has multiple departments. On the other hand, the division of some systems may not be straightforward, especially when the components in the system are interrelated or coupled in complicated ways.

For every implementation stage of the framework, the following issues arise:

- How many component DTs and subsystem DTs should be created for each PS?
- Where should the component, subsystem, and global system DTs be located?
- How should each (cloud/edge) server allocate communication, computing, and storage resources for each DT implementation?

Answering these questions requires solving complicated network resource allocation problems, if the DTs are supported by existing network resources. When deploying additional network resources is an option, such as adding servers or communication links to the network, answering the above questions requires solving joint resource deployment and resource allocation problems. In addition, domain knowledge related to the PS is often required to build its DT. More discussions about the DT implementation issues are provided in Section VIII.

Following the above framework, two implementation examples are given in the following subsections.

C. DT Implementation for a Space-Air-Ground Integrated Network

Consider a space-air-ground integrated network (SAGIN) [90]–[94]. In the spatial dimension, the entire system is naturally divided into the ground, air, and space networks, which form the three top-level subsystems of the SAGIN. Each of the three spatially separated networks can be further divided into multiple subnetworks that form the next-level subsystems. The subsystems can be further divided into components, such as base stations (BSs), routers, and user equipment in the ground network, drones and balloons in the air network, and satellites in the space network.

Building the DT for the entire SAGIN can start from building DTs for the individual components. The state at each component should be synchronized with that at the DT, which processes the state information and extracts the features of the component. In addition, profiling the individual components may also help the system to predict possible malfunctions and lifetime of the components. Consider a cellular BS as an example. The state information may include the number of busy channels, number of associated users, number of active connections, types of traffic carried by active connections, transmission power and data transmission speed to each user. Based on this information, the DT of the BS can extract features of the BS, such as whether the BS is overloaded.

The DTs of the components belonging to the same subsystem are integrated into the DT of the subsystem. For the ground network, its DT is built by integrating the DTs of the BSs, user equipment (UE) devices, gateways, and routers. Note that the DT of the ground network can be located in a different server as the DTs of the individual components. In this case, additional communication resources are required for the component DTs to forward state information to the subsystem DT. For example, the DT of a BS can pass information including possible user handoffs and co-channel interference conditions to the DT of the cellular network, which uses the information to make decisions on channel allocations, handoffs, and traffic load balancing.

This type of information aggregation is repeated until reaching the highest level DT, which is the DT of the entire system. At this level, data collected from the entire system can be used to coordinate network-wide operations, e.g., end-to-end routing, long-term traffic profile modeling, and network resource deployment. The system DT can also be utilized to make strategies about possible system configuration changes, such as adding or removing some drones in specific areas.

D. DT Implementation for a Network of Connected Vehicles

In this section we consider the case of constructing the DT for a network of connected vehicles in order to help facilitate network management and planning. From the temporal dimension, the DT can be created to support network management functions that require information with different time scales [95], [96]. Examples of real-time functions include collision avoidance and road traffic navigation. This requires the DT to track real-time locations, moving speeds and directions for vehicles, etc., which form a subset of the network states. Data belonging to this subset of the system state should be transmitted to and processed at the DT with very low delay, e.g., less than 10ms, so that the extracted feature information, such as distances to neighboring vehicles, can be useful for collision avoidance, assisted driving [97], [98], and other time-critical applications.

The DT can also be used to prevent road accidents caused by mechanical or electrical failures [98]. To support this function, the DT should keep tracking battery levels for electric vehicles, mechanical conditions of vehicles on roads, etc., which form a different subset of the network state. In order to extract features such as whether the car brake would be safe for the next few minutes or how long the battery can keep the car running, the DT should not only process the current state data but also the battery energy consumption and other vehicle running data in the past few minutes or hours.

The DT can also be constructed to reflect behavior of the network over a longer time period. For example, the DT stores system state data, including users' positions and service requests. After accumulating the data over days or months, the DT can extract system features including individual users travel patterns, usage of various mobile applications among road users, etc. This allows the DT to effectively predict user locations and content requirements in social-aware networks and make optimum decisions on mobile edge caching [99]. In vehicular networks, historical data of drivers can be used to model and predict driver behaviors, based on which optimum decisions can be made for personalized driving guidance or insurance pricing [98]. Furthermore, by processing the system state data over months or years, the DT can extract features of the system including traffic density, service demand distribution, and resource utilization, which helps the network operators to make decisions on the network infrastructure deployment. In this case, supporting the delay-tolerable functions does not require high data transmission rates or short data processing times. However, a large storage space and high computational power are required for storing and processing the historical system state data.

E. DT Structures

The generic five-stage framework can be integrated with different design and implemention structures when building DTs of complicated PSs. Below we introduce two DT implementation structures, a layered structure [74] and a modular structure [100].

The concept of "key-components" is proposed in [74] for modeling and building DTs. The key-components for a PS can be defined at different levels to enable different applications, and the different levels provide a layered structure for DT implementations. For instance, in a smart city or intelligent vehicular network, a set of "low-level" key-components can be defined for building the DTs that are used to model and predict the mechanical performance of vehicles. A different set of "high-level" key-components can be defined for building DTs that are used to support Car-as-a-Service. Building the latter DTs directly on top of the "high-level" components avoids processing unnecessary "low-level" details, which helps reduce the implementation complexity of the "high-level" DTs. This layered structure also allows a subset of the key-components at the same or different granularity levels to be integrated into a DT based on specific services and applications. When integrating the layered design with the stage-based framework, key-components can be defined from the spatial dimension. For example, the granularity levels vary from individual physical components and subsystems to global systems.

With a modular structure, the DT of a complicated PS can be broken into smaller parts, each of which can be developed and improved independently. The modular structure of DTs in [100] is based on the basic building blocks referred to as Digital Maps (DMaps). Each DMap implements a specific functionality of the DT. Higher level tasks and applications can be achieved by integrating the related DMaps that are needed by the tasks or applications. Using such a structure, different DMaps can be individually developed and updated without affecting other DMaps. A DMap, once implemented, can be used in multiple DTs or support different applications. This structure is used to design the *5G and beyond networks* in [100], where the DMaps are implemented based on subsets of network components or ranges of IP addresses. The DMap-based modular architecture easily fits the stage-based framework from the functional dimension. Starting from stage 2, new DMap modules can be built incrementally toward achieving the temporal objectives of the DT in the respective spatial level of the stage.

Both the layered and modular structures provide flexibility for building DTs of complicated PSs in an increasing manner. Meanwhile, both structures require interactions of different building blocks, i.e., DMaps or key-components. Therefore, defining standardized building blocks and communication protocols is essential for the composability, scalability, and adaptation of the DTs.

VII. Summary of Recent Research

For networking-related research on DTs, existing literature falls into two general categories, the first is creating DTs for different types of physical systems based on available networking support, and the second is taking advantage of the DTs for improving the network performance. This section starts from the first category by summarizing the existing DT solutions in intelligent manufacturing and different prototype implementations of DT-based smart cities, and then introduces recent work on applying DTs for network performance improvement. At the end, combination of machine learning (ML) and DT-related networks is discussed.

A. Intelligent Manufacturing and Cyber-Physical Systems

Since it is first introduced in manufacturing, the DT concept has been gaining substantial attention from manufacturing companies. One important DT application is building virtual workshops for intelligent manufacturing. Real-time mutual interactions between the physical and the virtual systems are important not only for real-time synchronization between the two systems but also for continuous evolvement of both systems. The convergence between the physical and virtual systems is important for realizing various smart operations in the manufacturing process, such as automatic control and fault prediction [5]. In [81], a DT solution is developed for optimizing operations of production lines. A virtual model that imitates real-world behavior of production lines is built by using a simulation software model. The virtual model is then coupled with the real production system through asynchronous connections that allow the virtual model and real system to exchange data periodically (e.g., every 10ms). The DT can be used to predict the performance of the real system in different scenarios, and the information is used to help improve the production efficiency. Since the PS is relatively “small” in the spatial dimension, it is possible to use high-speed communication links to achieve fast data exchange between the PS and the DT. The product DTs designed in [50] for electromechanical products and in [51] for a production line are also for PSs that span small areas so that Ethernet or other high speed local area communication links can be used to achieve real-time PS-DT communications. In addition, the volume of system state data is also relatively small, given the size of the PSs, and this leads to relatively short data processing delay at the DTs. Different from the above manufacturing DTs, the DT designed in [101] is for energy management in manufacturing systems. Since the application is not time sensitive, there is less pressure on real-time data communications and processing. This makes it possible to run the DT for large systems, where collecting and processing the system state data may be time consuming.

DTs and Cyber-physical systems (CPSs) are both important in integrating the cyber (or virtual) and physical systems to achieve smart manufacturing. CPSs are a new trend in the IoT research community, where PSs act as sensors collecting real-world data, which go through further analysis in the dedicated computation modules, and the result is fed back to the PSs. Both DT and CPS rely on real-time sensing at the physical system, data processing at the cyber system, and mutual interactions between the cyber and physical systems. However, CPS emphasizes more on monitoring and controlling functions of the cyber system on the physical system, and DT requires the virtual system to be an exact replica of the physical system and keep evolving with and influencing the physical system. For this reason, DT can be used not only for real-time optimization and control but also for error prediction and longer term designs and optimization of the physical system. DT is considered as a seamless integration between the cyber and physical spaces in [2], where more detailed discussions about the relationship and differences between CPS and DT are provided. As cyber-systems have become increasingly cloud-based, DTs can be used as a bridge between the cyber and physical systems, so that the real-time state and feature information at the DT helps the cyber system to make more accurate and optimum decisions and support more time-critical applications [55]. Large scale CPSs and DTs usually suffer from networking issues such as poor synchronization between the cyber (virtual) and physical systems due to communication and/or data processing delay. A common DT platform is needed so that various simulators, simulation engines, and physical machines that are used to build the DT components can share data, interwork, and support applications using standard interfaces [102]. A common DT platform also provides standard tools for testing, deploying, managing, updating, and integrating the DTs.

B. DT-based Smart Cities

The use of DTs in smart cities has been increasingly popular for planning infrastructure deployment, anticipating public problems, assisting communities in need, adjusting government policies, etc. Although building a DT to comprehensively reflect all aspects of a city is not practical in a short term, DTs are a critical tool towards smart cities. As such, prototype DTs have been created for specific applications of smart cities. Examples of these prototype DTs can be found in [75], [76], [103]. The virtual Singapore designed by the Dassault Systmes [75] helps achieve more efficient energy consumption by collecting and processing city state data and guiding operations of the city. The DT-based smart city for Dublin, Ireland [76] is created for helping the city planning, such as adding or removing buildings and blue spaces, by collecting information from the citizens and processing the information at the DT. The Vision Zero [103] program monitors road traffic conditions, collisions, pedestrian injuries, etc. and makes use of the information for managing traffic and improving road safety.

Healthcare is an important type of applications for smart cities. A DT-based ecosystem is designed in [104] for healthcare and well-being. In the proposed DT framework, health-related data is collected from personal health devices, such as smart phones; the data is processed and analyzed through an AI-inference engine; and the results are fed back to the user for personalized healthcare and well-being. The healthcare DT framework is based on the ISO/IEEE 11073 standard, which facilitates communications between personal devices and servers that host the DTs.

The road system is an important part of a city infrastructure. Implementation of the DT of the roads for a city is considered in [19], where different road condition data is collected and sent to cloud servers for extracting features of the road system. The data includes live videos of the road traffic, GPS data, and environmental measurements. When processed at the cloud servers, information can be extracted to identify and track persons and vehicles for various purposes, such as preventing crimes, solving accident conflicts, or resolving accident conflicts. DT-assisted road traffic data prediction is used in [20] for traffic management, where cameras on road crossings are used to collect road traffic, which is then processed to predict the traffic condition in nearby road crossings. Since the geographical area considered in this work is relatively limited, compared to other smart-city DTs, data can be collected and processed within relatively short time delay. For example, the traffic prediction in [20] can be performed on hourly basis.

For smart cities, the geographical size of the PS makes it unlikely to achieve a fully functional DT. Since collecting system state data in real-time is not practical, most applications supported by the existing smart city DTs have been time-insensitive. Meanwhile, extracting features needed by these applications requires processing data from a large number of places, people, or other components of a city over a long period of time. The amount of network resources required to communicate, store, and process the high volume of data is extremely high. In addition, the DTs for supporting different applications collect data based on different types of city states, use different data processing methods, and adopt different simulation models. This makes it difficult to extend the existing smart city DTs for supporting other applications. In addition, building the DT of a city or other large systems requires large volumes of data in different formats, multiple simulation models, different data fusion platforms, etc., which may not be interoperable without consistent guidelines to follow. A T-Cell framework developed by the digital urban European Twins (DUET) project provides a common environment that allows models, data, and simulations to dynamically interact in order to facilitate decision making in DT-based smart cities [18].

C. *DTs for Network Performance Improvement*

Recent enhancements in mobile network infrastructure have facilitated the support for strict performance requirements and new functions, and brought new intelligent services to the edge of the networks. Although making the optimal decisions and prediction is challenging due to the dynamic and heterogeneous nature of the networks, deployment of DTs provides real-time monitoring of the networks and perception data for the decision-making module. In [52], [56], [70], [71], DT networks are formed for making offloading decisions in mobile edge computing (MEC) networks. The DT networks are created by integrating DTs of different network elements, such as DTs of individual IoT devices [52], [105], DTs of individual end devices and edge servers [56], DTs of mobile users, base stations, and edge servers [70], or DTs of vehicles and RSUs [95], [96]. The DTs collect real-time state information of their PSs and extract the information that is needed to make task offloading decisions for mobile devices. The DT network in [71] includes a DT for each edge server and a DT for the entire MEC system. The DT of the MEC system reflects the interactions in the MEC system, such as offloading decisions of mobile users.

Deviation often exists between information reflected at a practical DT and the true information at the PS. This can be caused by many factors, such as varying network conditions and limited network resource availability. Such discrepancy affects the performance of the applications that depend on DTs to obtain information of real systems. The effects of this discrepancy are considered in [71], [106] when making decisions for mobile computation offloading. The work in [107] further considers different factors, including packet loss during data synchronization between the PS and the DT and biases in communication, computation, and storage resources, that cause the information discrepancy, analyzes the information deviations, and uses the derived deviation information to make computation offloading decisions and allocate network resources.

By synchronizing the system state at the physical and the digital systems, the DT networks enable network providers to estimate and predict the dynamic changes of the device locations, traffic demands, communication channels, and resource availability, and make decisions on computation offloading and resource allocations. However, maintaining the DTs requires real-time communications and data processing. The additional amounts of network resources for hosting the DTs are often not considered in the existing work on DT networks. In addition, the amount of network resources for running the DTs is directly related to the quality of the DTs, such as the promptness and similarity, which further affects the operation reliability of the applications. DT-enabled metaverse is achieved in [108], where multi-dimensional optimization, including communication and computation resources, offloading decisions, caching policies, BS allocation, transmit power, etc., is performed in order to achieve the low delay requirement of a metaverse.

DTs have also been used in vehicular networks to help catch the highly dynamic and unpredictable nature of the network topology and transmission conditions. DTs are used in an aerial-assisted vehicular network in [109], where the dynamically changing network topology poses high challenges to efficient resource allocations. In this work, one DT is maintained for the RSUs and another DT is for the vehicles. The former is to track real-time states of the RSUs, including their traffic load and available resources; and the latter is to track the real-time topology of the vehicles and required traffic load. The DTs are hosted in a UAV, which has a wide coverage and can communicate with the vehicles and RSUs and update the information at the DTs in real-time. The DTs are then used to make intelligent offloading decisions regarding which vehicle should offload its tasks to which RSU. A social-aware vehicular edge caching network is studied in [99], where a DT network is implemented at the RSUs and collects network state information, including vehicle locations, driving status, and data requirement. The information is then processed at the RSUs in order to extract features needed to build the vehicular social model. This helps design cache resource scheduling schemes that satisfy requirements of the vehicular users and reduce energy consumption of the RSUs. In [110], DT-enabled software-defined vehicular network (SDVN) is studied for improving the routing performance. Different routing schemes are run at the DT-based virtual network, which keeps refining the routing methods based on received routing-related reports from the real vehicular network. The central controller of the SDVN collects the status of individual vehicles and the roads and builds a real-time DT of the physical network. In addition, the controller maintains multiple future virtual SDVNs based on different predicted states of the vehicles and roads and adopts the optimum control scheme. Achieving the optimum routing requires accurately predicting the road traffic and network conditions, which requires collecting system state data in very small time granularity and processing the data at very high speed. The high computation load also consumes high processing energy, which should be taken into consideration when making the control decisions.

D. DT Combining with ML

Machine learning is one of the most powerful methods to achieve the intelligence needed for DTs to learn, model, and predict the behavior of the PSs. For example, based on historical and real-time data collected from the production lines and other manufacturing systems, supervised learning is used to estimate work conditions of the equipment, optimize equipment operations, predict possible faults, and locate causes of faults [50], [101]. For healthcare, a convolutional neural network (CNN) is used in [104] that takes input data collected from healthcare devices worn by the users and classifies health-related activities of users. In the DT of network function platform studied in [111], a neural network is trained for the DT to estimate and predict key performance indicators of the service system and optimize CPU resource sharing among virtual network functions.

DTs have also been combined with ML algorithms to enhance network management performance in data analytics and improve decision-making efficiency. In [69], a DT-empowered Industrial IoT architecture is considered, where DTs capture the characteristics of industrial devices and the dynamic changes of the network. Based on knowledge from DTs, training efficiency of federated learning is improved under resource constraints. With the aid of reinforcement learning, DTs can help track the dynamic changes of high dimensional states of mobile networks and make optimum decisions for network resource allocations, user scheduling, and other network operations [52], [89]. In [99], optimum caching decisions for a vehicular social network are made through a deep deterministic policy gradient (DDPG) learning approach. With the assistance of the social-aware DT that captures the dynamic and long-term changes of the network, the caching decisions are optimized in the diverse vehicular networking environments. In [112], a graph neural network helps build scalable DTs by modelling the intertwined relationships among multiple network slices that share the same physical infrastructure. Collaboration of DT and ML methods has been applied in MEC systems for making better task offloading decisions [56], [70], [89], [113], [114], [115]. With the support of DTs, complicated ML algorithms can be run to keep tracking states of the devices and the system and predict the future states. However, due to the limited network resources, the state information at the DTs may be inaccurate, delayed, or incomplete. This should be taken into consideration in the DT-based ML algorithms.

DT can predict the network behavior by running trained models based on current and historical data collected from the real network. This allows the network to proactively adapt to dynamically changed network conditions. However, processing the large volume of data to satisfy the requirements of real-time applications poses significant challenges to network resource deployment and management. Some key design requirements are presented in [116] in order to achieve a scalable and reliable architecture to build the DT of a 6G wireless system. Using distributed deep learning is one method that allows multiple models to be trained at different places with reduced computation load. However, such locally-trained models should then be integrated, which consumes additional communication and computation resources.

As one of the distributed learning methods, federated learning has been used to build DT-empowered communication networks [52], [89]. In mobile wireless networks, synchronizing real-time data between mobile user devices and the DTs that are usually located at edge servers requires a considerable amount of wireless resources. Using federated learning allows the DT of a mobile user to learn the user's behavior through its communications with other users. This helps reduce the communication loads between end users and the edge servers, which is especially helpful when such communication links are in poor quality. As a distributed ML algorithm, federated learning has enabled distributed advanced analysis while considering privacy protection and data security. However, designing scalable and efficient federated learning algorithms is still a challenging research topic [117].

Training practical ML models requires a huge volume of data. Often, there is a lack of data from real network due to data encryption or protection of user privacy. Instead of using data collected from real networks, generative adversarial networks (GANs) [118] can be used to generate synthetic data [100]. In addition, data can be generated using computer simulation or testbeds. It has been shown that network models trained using Graphical neural networks (GNNs) can be more scalable. That is, models trained for small networks are still valid for large networks or for networks with different configurations or traffic distributions [119].

VIII. Implementation Issues and Open Research Topics

Despite the great potentials of DTs, there are many challenges associated with developing and incorporating DTs in practical applications. In this section, we briefly discuss and identify some networking-related open issues in DT research and implementation.

Curse of dimensionality. By definition, a fully functional DT should reflect the physical system in temporal, spatial, and functional dimensions from small to large granularities. This means that in the temporal dimension, a DT is expected to reflect the features of its physical system from very small to very large time scales, e.g., from microseconds to years; in the spatial dimension, a DT should be built “from the micro atomic level to the macro geometrical level” [66]; and in the functional dimensions, a DT should reflect all features of the physical system. Thus, a fully functional DT implementation involves representing the physical system with an extremely large number of state types and collecting and processing a prohibitively high volume of data. This curse of dimensionality problem in DT implementation [120] may go beyond the capability of any existing methods and computing devices. In addition to using modern high speed CPUs and more efficient big-data technologies, another way to break the curse of dimensionality is to split the fully functional DT of a given physical system into multiple DTs, each of which reflects a subset of the features and states of the physical system in certain temporal and spatial ranges. The stage-based implementation described in Section VI is one approach to solving this problem.

Real-time synchronization. Keeping tight synchronization between the physical system and its DT is one of the fundamental requirements of high-quality DTs. To achieve this, it is important to minimize both the communication and computation time. Data transmission delay can affect the DT quality considerably if PS-DT communications or communications among multiple DT components of a PS should go through the Internet. Time-sensitive networking (TSN) [121] and deterministic networking DetNet [122] are protocols that help achieve bounded end-to-end transmission delay. However, TSN is a layer-2 protocol that has limited scalability, while providing ultra-low latency end-to-end using DetNet requires the network infrastructure to support DetNet services. In addition, complicated control and management technologies are required in order to provide reliable ultra-low end-to-end communication delay based on DetNet and TSN [123]–[125]. However, reliably delivering ultra-low end-to-end delay also depends on more efficient network architecture that facilitates efficient network management for routing, addressing, resource allocation, etc. In addition to communication delay, computation time in building and maintaining DTs, including simulation, modeling, and data processing, can also be significant. When the amount of network resource is limited, reducing the computation time may be temporarily achieved through the stage-based DT implementation.

Optimum resource deployment and management. A DT should keep tracking accurate real-time status of the PS and constantly update the features of the system. Implementing and maintaining DTs requires the support of network resources, including communication, computing, and caching. For complicated or large-scale PSs, how much each type of the network resources is needed, where to deploy the resources, and how to allocate the resources to build and maintain the DTs with desired quality requires joint heterogeneous resource management, which is a very challenging issue for large size networks [126]. In addition, different applications may require different features to be extracted from the system states and have different promptness and similarity requirements. This will result in the optimization of multiple objectives, some of which are likely contradictory. Given the limited amount of network resources, how to coordinate the application requirements in the network resource allocations is a complicated issue.

Standardization. In the last decade, DT technology has been used in a variety of applications and the use cases are growing rapidly. However, due to the lack of a standardized prototype of DTs, every application is using a definition of the DT specific to its domain. A standardized protocol is imperatively needed for proper usage of DT technology. The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) have been developing DT-related standards. The ISO 23247 series provide the general guidelines for building DTs in manufacturing environments. In October 2021, the organization published ISO 23247-1 [127], which is the first part of the ISO 23247 series and provides the terms and definitions related to DTs in manufacturing. Later parts of the series are expected to provide the reference models and framework views from different perspectives. In particular, the ISO 23247-4 is expected to provide guidelines for information exchange and protocols from a networking perspective, which would facilitate the information exchange of DTs built for different components and subsystems in the same system or even DTs of different systems. In addition, the ISO/IEC JTC 1/SC 41 [128] is for standardization for developing IoT and DT-related applications.

Addressing. Addressing is another important issue, especially in a network of DTs [24], [129]. First of all, the DT of a given PS should have a unique identity in a network. This means that given the identity of the PS, its DT should be located in the network; and given the identity of a DT in the network, the identity of its PS should be located. Second, as a virtual representative of its PS, a DT should be addressable directly when it is working with third-party applications on behalf of its PS. That is, communications between the DT of a given PS and a third-party application should not go through the PS. This can be difficult unless the information about the identity mapping between the PS and the DT is available at the third-party application before the communication starts. As a transitional step, a third-party application can always communicate with the PS first, which then directs the application to its DT. Alternatively, trusted DT mapping servers may be built to help applications locate the DTs of given PSs. The identity mapping information can also be cached by intermediate network devices. However, the communication re-direction and outdated caching information can result in a considerable initial delay in DT-involved communications and cause security issues.

Security. The use of DTs poses significant security issues. With a DT as the replica of the physical system, keeping the system safe requires not only protecting the PS, but also the DT and the connection between them. With DTs being used to interact with third party applications on behalf of the PS, the DTs are more exposed to the outside environment than the PS, and therefore, their security is even more crucial. Any security breach to the DTs results in security breach to the PSs. However, discovering security breach to a DT can be delayed due to the fact that the DT and the PS are physically apart. In addition, DTs can influence or control over their PSs, and keeping DTs from infiltrators and hackers is important to ensure the PS to work normally.

Specialization. In a large system consisting of multiple subsystems, each subsystem can use its own DTs. These DTs can all be specialized or customized for different applications of their corresponding subsystems. In this type of implementation, the central management unit needs to deal with multiple types of DTs with different structures and has to convert their outputs to a unified type of data to facilitate the communication among different DTs. Alternatively, a unified DT structure can be used in all subsystems to make the inter-DT communications easier. However, this may not achieve the same quality of DTs as in the specialized case, since the DTs are crafted for general usage for all subsystems.

IX. Summary

This paper has presented a comprehensive review on the networking-related research and technological development in DT implementation. In particular, the existing definitions of DT with key features and characteristics have been summarized. After that, desired properties of DT implementation have been discussed, highlighting the relationship between the quality of DTs and the networking support. Moreover, a stage-based DT implementation framework has been presented to demonstrate how DTs can be practically implemented for large scale and complicated systems. Finally, the existing research on networking-related DT implementation have been thoroughly reviewed and the technical challenges and open research issues have been elaborated.

References

[1] M. Grieves, *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. New York, NY, USA: McGraw-Hill Education, 2005.

[2] F. Tao, Q. Qi, L. Wang, and A. Nee, "Digital twins and cyberphysical systems toward smart manufacturing and industry 4.0: Correlation and comparison," *Engineering*, vol. 5, no. 4, pp. 653–661, 2019.

[3] Q. Qi and F. Tao, "Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison," *IEEE Access*, vol. 6, pp. 3585–3593, 2018.

[4] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," *Digital Twin White Paper*, 2014.

[5] F. Tao and M. Zhang, "Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, p. 2041820427, 2017.

[6] J. C. F. Tao, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9, p. 35633576, 2018.

[7] J. Lee, E. Lapira, B. Bagheri, and H. Kao, "Recent advances and trends in predictive manufacturing systems in big data environment," *Manufacturing Letters*, vol. 1, no. 1, pp. 38–41, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213846313000114

[8] B. A. Talkhestani, N. Jazdi, W. Schloegl, and M. Weyrich, "Consistency check to synchronize the digital twin of manufacturing automation based on anchor points," *Procedia CIRP*, vol. 72, pp. 159–164, 2018, 51st CIRP Conference on Manufacturing Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221282711830324X

[9] E. H. Glaessgen and D. S. Stargel, "The digital twin paradigm for future NASA and U.S. air force vehicles," in *the 53rd Structures, Structural Dynamics, and Materials Conference*, 2012.

[10] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167 653–167 671, 2019.

[11] C. Ananda, "General aviation aircraft avionics: Integration & system tests," *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 5, pp. 19–25, 2009.

[12] N. Wickramasinghe, P. P. Jayaraman, J. Zelcer, A. R. M. Forkan, N. Ulapane, R. Kaul, and S. Vaughan, "A vision for leveraging the concept of digital twins to support the provision of personalised cancer care," *IEEE Internet Computing*, pp. 1–1, 2021.

[13] T. Erol, A. F. Mendi, and D. Doan, "The digital twin revolution in healthcare," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2020, pp. 1–7.

[14] A. Ricci, A. Croatti, and S. Montagna, "Pervasive and connected digital twins a vision for digital health," *IEEE Internet Computing*, 2021.

[15] S. Chakrabarty, D. W. Engels, and L. Wood, "Consumer frameworks for smart environments," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, Berlin, Germany, 2020.

[16] L. Cascone, M. Nappi, F. Narducci, and I. Passero, "Dtpaal: Digital twinning pepper and ambient assisted living," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1397–1404, 2022.

[17] A. Lee, J. Kim, and I. Jang, "Movable dynamic data detection and visualization for digital twin city," in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, 2020, pp. 1–2.

[18] L. Raes, P. Michiels, T. Adolphi, C. Tampere, T. Dalianis, S. Mcaleer, and P. Kogut, "Duet: A framework for building secure and trusted digital twins of smart cities," *IEEE Internet Computing*, 2021.

[19] O. E. Marai, T. Taleb, and J. Song, "Roads infrastructure digital twin: A step toward smarter cities realization," *IEEE Network*, vol. 35, no. 2, pp. 136–143, 2021.

[20] C. Hu, W. Fan, E. Zeng, Z. Hang, F. Wang, L. Qi, and M. Z. A. Bhuiyan, "Digital twin-assisted real-time traffic data prediction method for 5g-enabled internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2811–2819, 2022.

[21] W. Serrano, "Digital systems in smart city and infrastructure: Digital as a service," *Smart Cities*, vol. 1, no. 1, p. 134154, 2018.

[22] N. Mohammadi and J. E. Taylor, "Smart city digital twins," *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–5, 2017.

[23] F. Tao, H. Zhang, A. Liu, and A. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, April 2019.

[24] 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.

[25] G. Mylonas, A. Kalogeras, G. Kalogeras, C. Anagnostopoulos, C. Alexakos, and L. Muoz, “Digital twins from smart manufacturing to smart cities: A survey,” *IEEE Access*, vol. 9, pp. 143 222–143 249, 2021.

[26] A. Rasheed, O. San, and T. Kvamsdal, “Digital twin: Values, challenges and enablers from a modeling perspective,” *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[27] Y. Wu, K. Zhang, and Y. Zhang, “Digital twin networks: A survey,” *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[28] Y. He, J. Guo, and X. Zheng, “From surveillance to digital twin: Challenges and recent advances of signal processing for industrial internet of things,” *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 120–129, 2018.

[29] H. Park, A. Easwaran, and S. Andalam, “Challenges in digital twin development for cyber-physical production systems,” *Cyber Physical Systems. Model-Based Design. Cham, Switzerland: Springer*, pp. 28–48, 2018.

[30] H. Zipper and C. Diedrich, “Synchronization of industrial plant and digital twin,” in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1678–1681.

[31] D. Kiritsis, “Closed-loop PLM for intelligent products in the era of the internet of things,” *Computer-Aided Design*, vol. 43, no. 5, pp. 479–501, 2011.

[32] E. J. Tuegel, A. R. Ingraffea, T. G. Eason, and S. M. Spottswood, “Reengineering aircraft structural life prediction using a digital twin,” *International Journal of Aerospace Engineering*, pp. 1–14, 2011.

[33] Z. Liu, N. Meyendorf, and N. Mrad, “The role of data fusion in predictive maintenance using digital twin,” in *AIP Conference Proceedings*, vol. 1949, no. 1. AIP Publishing LLC, 2018, pp. 020–023.

[34] Global Horizons Final Report, *United States Air Force Global Science and Technology Vision*. CreateSpace Independent Publishing Platform, 2013.

[35] R. Sderberg, K. Wrmefjord, J. S. Carlson, and L. Lindkvist, “Toward a digital twin for real-time geometry assurance in individualized production,” *Cirp Annals Manufacturing Technology*, vol. 66, no. 1, pp. 137–140, 2017.

[36] S. Boschert, C. Heinrich, and R. Rosen, “Next generation digital twin,” in *Twelfth International Symposium on Tools and Methods of Competitive Engineering (TMCE)*, May 2018.

[37] “Digital twin,” https://www.plm.automation.siemens.com/global/zh/our-story/glossary/digital-twin/24465, 2020.

[38] “The digital twin: Compressing time to value for digital industrial companies,” https://www.ge.com/digital/applications/digital-twin, 2020.

[39] B. Smarslok, A. Culler, and A. Mahadevan, “Error quantification and confidence assessment of aerothermal model predictions for hypersonic aircraft,” *Proc. AIAA*, p. 1817, 2012.

[40] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, “Energy aware offloading for competing users on a shared communication channel,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 87–96, 2017.

[41] H. Chen, D. Zhao, Q. Chen, and R. Chai, “Joint computation offloading and radio resource allocations in small-cell wireless cellular networks,” *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 3, pp. 745–758, 2020.

[42] A. Hekmati, P. Teymoori, T. D. Todd, D. Zhao, and G. Karakostas, “Optimal mobile computation offloading with hard deadline constraints,” *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2160–2173, 2020.

[43] Q. Qi, D. Zhao, T. W. Liao, and F. Tao, “Modeling of cyber-physical systems and digital twin based on edge computing, fog computing and cloud computing towards smart manufacturing,” *13th International Manufacturing Science and Engineering Conference*, vol. 1, p. 17, 2018.

[44] D. Hofheinz and V. Shoup, “Gnuc: A new universal composability framework,” *Journal of Cryptology*, vol. 28, no. 3, p. 423508, July 2015.

[45] G. T. Heineman and W. T. Councill, *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Professional, 2001.

[46] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. da Mota Silveira Neto, Y. C. Cavalcanti, and S. R. de Lemos Meira, “Twenty-eight years of component-based software engineering,” *Journal of Systems and Software*, vol. 111, pp. 128–148, 2016.

[47] Y. Mitrofanova, A. Sherstobitova, and O. Filippova, *Modeling Smart Learning Processes Based on Educational Data Mining Tools*. Springer, 01 2019, pp. 561–571.

[48] J. Autiosalo, “Platform for industrial internet and digital twin focused education research, and innovation: Ilmatar the overhead crane,” *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, p. 241244, 2018.

[49] M. Abdel-Basset, G. Manogaran, M. Mohamed, and E. Rushdy, “Internet of things in smart education environment: Supportive framework in the decision-making process,” *Concurrency and Computation: Practice and Experience*, vol. 31, p. e4515, 05 2018.

[50] Y. Lu, X. Qiu, and Y. Xing, “Digital twin-based operation simulation system and application framework for electromechanical products,” in *2021 International Conference on Computer, Control and Robotics (ICCCR)*, 2021, pp. 146–150.

[51] H. Zhang, Q. Liu, X. Chen, D. Zhang, and J. Leng, “A digital twin-based approach for designing and multi-objective optimization of hollow glass production line,” *IEEE Access*, vol. 5, pp. 26 901–26 911, 2017.

[52] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Communication-efficient federated learning and permissioned blockchain for digital twin edge networks,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2276–2288, 2021.

[53] Y. Tang, N. Cheng, W. Wu, M. Wang, Y. Dai, and X. Shen, “Delay-minimization routing for heterogeneous VANETs with machine learning based mobility prediction,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3967–3979, 2019.

[54] Z. Ji, Y. Wang, and J. Lu, “Delay-aware resource control and routing in multihop wireless networks,” *IEEE Communications Letters*, vol. 19, no. 11, pp. 2001–2004, 2015.

[55] K. M. Alam and A. El Saddik, “C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems,” *IEEE Access*, vol. 5, pp. 2050–2062, 2017.

[56] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, “Deep reinforcement learning for stochastic computation offloading in digital twin networks,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4968–4977, 2020.

[57] Z. Khalid, N. Fisal, and M. Rozaini, “A survey of middleware for sensor and network virtualization,” *Sensors*, vol. 14, no. 12, p. 2404624097, 2014.

[58] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, “Network function virtualization: Challenges and opportunities for innovations,” *IEEE Commun. Mag.*, vol. 53, no. 2, p. 9097, 2015.

[59] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, “Wireless sensor network virtualization: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, p. 553576, 2016.

[60] Q. Yu, J. Ren, Y. Fu, Y. Li, and W. Zhang, “Cybertwin: An origin of next generation network architecture,” *IEEE Wireless Communications*, vol. 26, no. 6, pp. 111–117, 2019.

[61] J. Li, W. Shi, Q. Ye, S. Zhang, W. Zhuang, and X. Shen, “Joint virtual network topology design and embedding for cybertwin-enabled 6g core networks,” *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 313–16 325, 2021.

[62] Z. Yin, N. Cheng, T. H. Luan, and P. Wang, “Physical layer security in cybertwin-enabled integrated satellite-terrestrial vehicle networks,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4561–4572, 2022.

[63] W. Sun, N. Xu, L. Wang, H. Zhang, and Y. Zhang, “Dynamic digital twin and federated learning with incentives for air-ground networks,” *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 321–333, 2022.

[64] L. Zhao, C. Wang, K. Zhao, D. Tarchi, S. Wan, and N. Kumar, “Interlink: A digital twin-assisted storage strategy for satellite-terrestrial networks,” *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–1, 2022.

[65] Y. Hui, X. Ma, Z. Su, N. Cheng, Z. Yin, T. H. Luan, and Y. Chen, “Collaboration as a service: Digital twins enabled collaborative and distributed autonomous driving,” *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[66] M. Grieves and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. Springer, August 2017, pp. 85–113.

[67] Z. Wei, S. Wang, D. Li, F. Gui, and S. Hong, “Data-driven routing: A typical application of digital twin network,” in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021, pp. 1–4.

[68] B. Tan, Y. Qian, H. Lu, D. Hu, Y. Xu, and J. Wu, "Toward a future network architecture for intelligence services: A cyber digital twin-based approach," *IEEE Network*, vol. 36, no. 1, pp. 98–104, 2022.

[69] J. Deng, Q. Zheng, G. Liu, J. Bai, K. Tian, C. Sun, Y. Yan, and Y. Liu, "A digital twin approach for self-optimization of mobile networks," in *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2021, pp. 1–6.

[70] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1427–1444, 2022.

[71] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 240–12 251, 2020.

[72] P. Krishnan, K. Jain, R. Buyya, P. Vijayakumar, A. Nayyar, M. Bilal, and H. Song, "Mud-based behavioral profiling security framework for software-defined iot networks," *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6611–6622, 2022.

[73] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 1–30, 2022.

[74] C. Steinmetz, G. N. Schroeder, R. N. Rodrigues, A. Rettberg, and C. E. Pereira, "Key-components for digital twin modeling with granularity: Use case car-as-a-service," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 23–33, 2022.

[75] Dassault systmes builds a second singapore. https://www.hannovermesse.de/en/news/news-articles/dassault-systemes-builds-a-second-singapore.

[76] G. White, A. Zink, L. Codecá, and S. Clarke, "A digital twin smart city for citizen feedback," *Cities*, vol. 110, p. 103064, 2021.

[77] A. Francisco, N. Mohammadi, and J. E. Taylor, "Smart city digital twin–enabled energy management: Toward real-time urban building energy benchmarking," *Journal of Management in Engineering*, vol. 36, no. 2, 2020.

[78] G. Schrotter and C. Hürzeler, "The digital twin of the city of zurich for urban planning," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 88, no. 1, pp. 99–112, 2020.

[79] T. Deng, K. Zhang, and Z.-J. M. Shen, "A systematic review of a digital twin city: A new pattern of urban governance toward smart cities," *Journal of Management Science and Engineering*, vol. 6, no. 2, pp. 125–134, 2021.

[80] D. N. Ford and C. M. Wolf, "Smart cities with digital twin systems for disaster management," *Journal of management in engineering*, vol. 36, no. 4, p. 04020027, 2020.

[81] S. M. Jeon and S. Schuesslbauer, "Digital twin application for production optimization," in *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, Singapore, 2020, pp. 542–545.

[82] Q. Qi, F. Tao, Y. Zuo, and D. Zhao, "Digital twin service towards smart manufacturing," *Procedia Cirp*, vol. 72, pp. 237–242, 2018.

[83] J. Wang, L. Ye, R. X. Gao, C. Li, and L. Zhang, "Digital twin for rotating machinery fault diagnosis in smart manufacturing," *International Journal of Production Research*, vol. 57, no. 12, pp. 3920–3934, 2019.

[84] K. Xia, C. Sacco, M. Kirkpatrick, C. Saidy, L. Nguyen, A. Kircaliali, and R. Harik, "A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence," *Journal of Manufacturing Systems*, vol. 58, pp. 210–230, 2021.

[85] F. Tao, Y. Zhang, Y. Cheng, J. Ren, D. Wang, Q. Qi, and P. Li, "Digital twin and blockchain enhanced smart manufacturing service collaboration and management," *Journal of Manufacturing Systems*, 2020.

[86] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6g," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 219–16 230, 2021.

[87] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5709–5718, 2020.

[88] D. Wang, B. Li, B. Song, Y. Liu, K. Muhammad, and X. Zhou, "Dual-driven resource management for sustainable computing in the blockchain-supported digital twin iot," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[89] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098–5107, 2021.

[90] N. Cheng, W. Quan, W. Shi, H. Wu, Q. Ye, H. Zhou, W. Zhuang, X. Shen, and B. Bai, "A comprehensive simulation platform for space-air-ground integrated network," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 178–185, 2020.

[91] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2714–2741, 2018.

[92] Z. Niu, X. Shen, Q. Zhang, and Y. Tang, "Space-air-ground integrated vehicular network for connected and automated vehicles: Challenges and solutions," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 142–169, 2020.

[93] H. Wu, J. Chen, C. Zhou, W. Shi, N. Cheng, W. Xu, W. Zhuang, and X. Shen, "Resource management in space-air-ground integrated vehicular networks: SDN control and AI algorithm design," *IEEE Wireless Communications*, vol. 27, no. 6, pp. 52–60, 2020.

[94] H. Cui, J. Zhang, Y. Geng, Z. Xiao, T. Sun, N. Zhang, J. Liu, Q. Wu, and X. Cao, "Space-air-ground integrated network (sagin) for 6g: Requirements, architecture and challenges," *China Communications*, vol. 19, no. 2, pp. 90–108, 2022.

[95] K. Zhang, J. Cao, and Y. Zhang, "Adaptive digital twin and multi-agent deep reinforcement learning for vehicular edge computing and networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1405–1413, 2022.

[96] Y. Dai and Y. Zhang, "Adaptive digital twin for vehicular edge computing and networks," *Journal of Communications and Information Networks*, vol. 7, no. 1, pp. 48–59, 2022.

[97] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–6.

[98] Z. Wang, R. Gupta, K. Han, H. Wang, A. Ganlath, N. Ammar, and P. Tiwari, "Mobility digital twin: Concept, architecture, case study, and future challenges," *IEEE Internet of Things Journal*, 2022.

[99] K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, "Digital twin empowered content caching in social-aware vehicular edge networks," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 239–251, 2022.

[100] A. Mozo, A. Karamchandani, S. Gómez-Canaval, M. Sanz, J. I. Moreno, and A. Pastor, "B5gemini: Ai-driven network digital twin," *Sensors*, vol. 22, no. 11, p. 4106, 2022.

[101] F. Pires, B. Ahmad, A. P. Moreira, and P. Leito, "Digital twin based what-if simulation for energy management," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2021, pp. 309–314.

[102] S. Yun, J.-H. Park, and W.-T. Kim, "Data-centric middleware based digital twin platform for dependable cyber-physical systems," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, Milan, Italy, 2017, pp. 922–926.

[103] Vision zero. https://resources.esri.ca/news-and-updates/the-role-digital-twins-in-smart-cities/.

[104] F. Laamarti, H. F. Badawi, Y. Ding, F. Arafsha, B. Hafidh, and A. E. Saddik, "An iso/ieee 11073 standardized digital twin framework for health and well-being in smart cities," *IEEE Access*, vol. 8, pp. 105 950–105 961, 2020.

[105] Z. Lv, L. Qiao, and R. Nowak, "Energy efficient resource allocation of wireless energy transfer for internet of everything in digital twins," *IEEE Communications Magazine*, pp. 1–7, 2022.

[106] T. Do-Duy, D. Van Huynh, O. A. Dobre, B. Canberk, and T. Q. Duong, "Digital twin-aided intelligent offloading with edge selection in mobile edge computing," *IEEE Wireless Communications Letters*, vol. 11, no. 4, pp. 806–810, 2022.

[107] T. Liu, L. Tang, W. Wang, X. He, Q. Chen, X. Zeng, and H. Jiang, "Resource allocation in dt-assisted internet of vehicles via edge intelligent cooperation," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[108] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Wireless Communications Letters*, pp. 1–1, 2022.

[109] W. Sun, P. Wang, N. Xu, G. Wang, and Y. Zhang, "Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5839–5852, 2022.

[110] L. Zhao, G. Han, Z. Li, and L. Shu, "Intelligent digital twin-based software-defined vehicular networks," *IEEE Network*, vol. 34, no. 5, pp. 178–184, 2020.

[111] P. Krmer, P. Diederich, C. Krmer, R. Pries, W. Kellerer, and A. Blenk, "D2a: Operating a service function chain platform with data-driven

scheduling policies," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[112] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1367–1376, 2022.

[113] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692–4707, 2019.

[114] X. Xu, B. Shen, S. Ding, G. Srivastava, M. Bilal, M. R. Khosravi, V. G. Menon, M. A. Jan, and M. Wang, "Service offloading with deep q-network for digital twinning-empowered internet of vehicles in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1414–1423, 2022.

[115] S. Chen, J. Chen, Y. Miao, Q. Wang, and C. Zhao, "Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 364–375, 2022.

[116] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6g: Vision, architectural trends, and future directions," *CoRR*, vol. abs/2102.12169, 2021. [Online]. Available: https://arxiv.org/abs/2102.12169

[117] P. Kairouz *et al.*, *Advances and Open Problems in Federated Learning*. Foundations and Trends in Machine Learning, 2021, vol. 14, no. 1-2.

[118] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2014, p. 26722680.

[119] P. Almasan, M. Ferriol-Galms, J. Paillisse, J. Surez-Varela, D. Perino, D. Lpez, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Digital twin network: Opportunities and challenges," 2022. [Online]. Available: https://arxiv.org/abs/2201.01144

[120] M. Sokolov, M. von Stosch, H. Narayanan, F. Feidl, and A. Butté, "Hybrid modelinga key enabler towards realizing digital twins in biopharma?" *Current Opinion in Chemical Engineering*, vol. 34, p. 100715, 2021.

[121] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The ieee TSN and IETF DetNet standards and related 5G ULL research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[122] "Deterministic networking," https://datatracker.ietf.org/wg/detnet/about/.

[123] E. Kim, Y. Ryoo, B. Yoon, and T. Cheung, "Active control and management system for providing the ultra-low latency service on deterministic networks," in *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2021, pp. 70–74.

[124] S. Tschke, F. Lynker, H. Buhr, F. Schreiner, A. Willner, A. Vick, and M. Chemnitz, "Time-sensitive networking over metropolitan area networks for remote industrial control," in *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2021, pp. 1–4.

[125] J. Lv, Y. Zhao, X. Wu, Y. Li, and Q. Wang, "Formal analysis of tsn scheduler for real-time communications," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 1286–1294, 2021.

[126] S. Manap, K. Dimyati, M. N. Hindia, M. S. Abu Talip, and R. Tafazolli, "Survey of radio resource management in 5g heterogeneous networks," *IEEE Access*, vol. 8, pp. 131 202–131 223, 2020.

[127] Iso 23247-1: 2021 automation systems and integration digital twin framework for manufacturing part 1: Overview and general principles. https://www.iso.org/standard/75066.html/.

[128] "Iso/iec jtc 1/sc 41 internet of things and digital twin," https://www.iec.ch/dyn/www/f?p=103:7:614288923263590::::FSP_ORG_ID,FSP_LANG_ID:20486,25/.

[129] S. O. Erikstad, "Merging physics, big data analytics and simulation for the next-generation digital twins," in *11th Symp. High-Perform. Mar. Vehicles (HIPER)*, Sept. 2017, p. 140150.