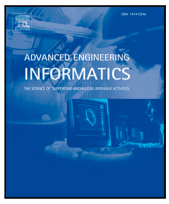
Advanced Engineering Informatics 64 (2025) 102970

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei



Full length article

Distributed digital twins on the open-source OpenTwins framework

Sergio Infante *, Julia Robles, Cristian Martín, Bartolomé Rubio, Manuel Díaz

ITIS Software, University of Málaga, Málaga, Spain

ARTICLE INFO

Keywords:
Digital Twins
Distributed systems
OpenTwins
Industry 4.0
Internet of Things

ABSTRACT

With the continuous evolution of digital twins, the requirements of interconnection and interoperability have led to the creation of the term Distributed Digital Twin, where commonly, different components of the same digital twin operate on different devices. This article addresses this new gap in the field by combining Digital Twins and Distributed systems technologies and introduces a re-definition of the architecture of OpenTwins, an open-source platform designed to develop generic next-gen 3D-IoT-AI-powered digital twins. This approach enables the distribution of digital twins across different infrastructures by seamlessly integrating multiple instances of OpenTwins working together. Distributing Digital Twins across networks and devices can offer dynamic and collaborative simulations, artificial intelligence techniques, yet poses synchronization and scalability challenges. The platform re-definition involves the definition of a lightweight, synchronized, and distributed version of the original OpenTwins architecture to tackle these issues. As the field of digital twins is closely linked to the Internet of Things environment and Industry 4.0, this architecture has been designed to be compatible in IoT devices, being compatible with ARM architectures, consuming fewer resources than the original platform as it has fewer components, thus reducing the number of messages and the bandwidth consumed.

1. Introduction

In the dynamic landscape of Industry 4.0, where technological innovations continue to reshape traditional paradigms, the concept of Digital Twins (DT) has emerged as a transformative force. A DT represents a virtual counterpart of a physical entity, providing a real-time and comprehensive representation of its behavior, status, and interactions [1]. This paradigm has proven invaluable in optimizing processes, enhancing efficiency, and fostering innovation across various domains [2,3].

The complexities of modern industries and the demand for more sophisticated and interconnected solutions have given rise to the concept of Distributed Digital Twins (DDT) [4]. This leads to the connection and combination of different DTs across various devices and networks. Consequently, this new type of DTs, while sharing challenges similar to the traditional ones, introduces a distinct set of problems and complexities associated with the field of distributed systems [5,6].

Unlike their singular counterparts, DDTs extend the capabilities of DTs by distributing their functionalities across interconnected systems, creating a networked ecosystem of virtual representations that mirrors the physical world. This can be helpful in use cases where decision-making is critical, and the latency must be minimized or in cases where the complexity of the whole DT becomes unmanageable. This distributed approach enables a more integrated and scalable representation of complex systems, unlocking new potentials for the DT field, as is the case of IoTwins project [7] or SRADI [8]. However, the research and development carried out in this field is still at an early stage and leaves many issues unresolved.

DTs need algorithms and mechanisms to face these problems and give them a solution. This is why a re-definition of the OpenTwins platform is proposed. In previous work [3], we presented the OpenTwins platform, a framework for the development of next-generation DTs (Fig. 1). This platform can manage DTs, combining them with machine learning (ML) and simulations [9], and even supports visualization not only of the data but also supports models developed in the Unity graphics engine. This re-definition is a distributed lightweight architecture, which allows it to be used in low-resource devices. This new architecture will be used to implement these solutions to the different challenges related to DDTs and transfer them to the main architecture, with the objective of creating an ecosystem of DTs.

Therefore, the main contributions of this work are the redesign of the OpenTwins architecture to make it lighter and distributed while making it compatible with ARM devices. In addition, the redesign still provides the ability to deploy simulation and ML models for the accomplishment of the system logic, removing many of the interactions

* Corresponding author.
E-mail address: sergioip@uma.es (S. Infante).

https://doi.org/10.1016/j.aei.2024.102970
Received 15 May 2024; Received in revised form 30 September 2024; Accepted 23 November 2024
Available online 7 December 2024

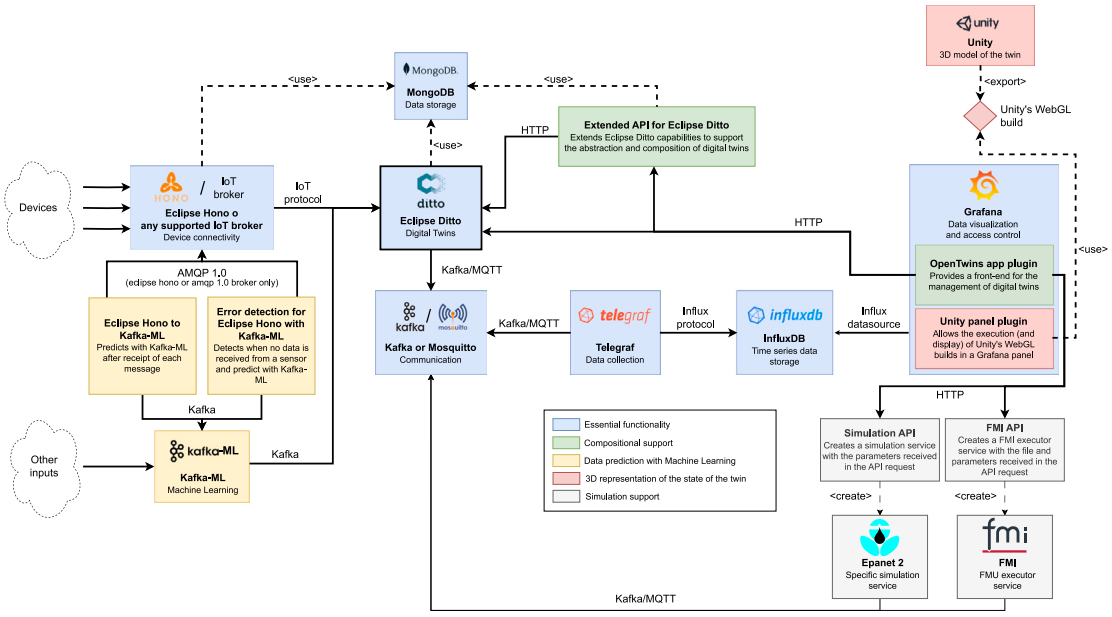


Fig. 1. Reference architecture of OpenTwins.

with a server that could slow down the model's operation. The ability to connect the different OpenTwins instances in different ways offers the possibility to manage the distribution of DTs in the way that best suits the environment. Therefore, these elements, favor and impulse the concept of DDT, which, has not been studied in depth before to the best of our knowledge. To validate this new architecture, a use case involving critical decisions for autonomous driving is proposed.

The rest of the article is structured as follows. Section 2 presents the related work. In Section 3 the architecture is presented. Then, an autonomous vehicle use case using this architecture is discussed in Section 4. An evaluation of the performance is presented in 5. Lastly, the conclusions and future work are discussed in Section 6.

2. Related work

The advancement of DTs has significantly influenced recent years. The evolution of the field of DTs together with its proximity to the Internet of Things(IoT) device environment, ML, etc. has led to the need to distribute the functionality of DTs, as is the case of [10], where DTs, IoT, and Distributed Ledger Technology (DLR) [11], which is a secure way of conducting and recording transfers of digital assets without the need for a central authority, have been proposed for monitoring, tracking products, analyzing data and support decision-making. Although there is no mention of DDTs, the use of the mentioned technologies forms a DDT architecture. Another example is presented in [12], where a DT based on the distribution of sensors is presented.

While numerous applications offer diverse functionalities for DT development, the majority concentrate on specific tasks such as integrating DTs with simulation, ML, or pure data analysis and most of these platforms are centralized. In these platforms DTs are developed in a server and all the data related to those DTs is sent to that point where is processed and the logic is executed. The scarcity of multidisciplinary tools for the development of DTs is more noticeable in the field of DDTs.

Examples include [13], where a framework that combines simulation with ML for the development of DTs is presented, being an implementation-defined for a single use case, other example is [14], where a specific framework is developed to integrate DTs simulation into a security operations center (SOC) or [15], where a DT is developed for reconfigurable manufacturing systems.

Moshrefzadeh et al. [8] present SRADI, a conceptual platform for the development of DTs of agricultural landscapes. SRADI, based on metamodels for the interconnection of different components, is composed of different layers, each one with a different function, such as an actor layer, an application layer, sensors, analytics, etc. However, this platform focuses on a single type of DT, in addition to distributing only the devices where data is collected, unlike the functionalities offered by OpenTwins where multiple components can be distributed.

One of the latest works of DTTs is IoTwins [7], a framework for developing DTs along the IoT-edge-cloud continuum. IoTwins presents a platform capable of distributing DTs and their components, the differences between the different layers are distinctly defined. In contrast, OpenTwins offers more flexibility in deploying the different components on different devices, to better adapt to the available infrastructure.

Other general purpose framework is presented in [18]. This framework proposes the use of simulations and ML to create general purpose DTs. Although it is an interesting proposal that combines the use of DT technologies with simulation gaming technologies to create DTs, it does not allow the composition of DTs and therefore the DDTs, remaining a step behind the proposal presented in this article. A similar approach of OpenTwins is presented in [16], where a framework for the development of general purpose compositional DDTs is proposed. In this work, the importance of building DTs from other smaller DTs but in a compositional distributed approach is exposed. However, the distribution of the DTs is produced using the composition of smaller DTs whereas in OpenTwins, the user can choose whether the DDT is created using several smaller DTs or by splitting the functionalities of a DT into different devices. Although this proposal is interesting, it is still a proof of concept unlike OpenTwins, which also offers other functionalities, such as 3D representation or support for ML or simulations.

An example not close to the functionalities of OpenTwins but within the context of DDTs is also [17]. This work proposes the use of federated learning, which is a technique within ML [19], for the development of a DTs framework to create decentralized DDTs. Although it is an interesting proposal, it is not a finished implementation, in addition to the lack of components compared to other platforms already mentioned.

Scharwarz [20] studied the role of DTs in connected and automated vehicles. This paper highlights the importance of DTs in this environment, emphasizing the use of composite and DDTs in the network. One important problem of a major DT developments is the unidirectional

Table 1
Comparison of different DT frameworks functionalities.

Functionality	Distribution over networks	Composition of DTs	General purpose DTs	Simulation components	ML components	Data representation	3D visualization
OpenTwins	✓	✓	✓	✓	✓	✓	✓
Lightweight OT	✓	✓	✓	Compatible	Compatible	Compatible	Compatible
IoTwins [7]	Partial	✓	✓	✓	✓	✓	×
SRADI [8]	Partial	×	×	✓	×	✓	✓
Aziz [16]	✓	✓	✓	×	×	×	×
San [17]	✓	×	✓	×	✓	×	×
Hurkamp [13]	×	×	×	✓	✓	×	×
Lektauers [18]	×	×	✓	✓	✓	×	×

data transfer from physical entities to virtual models, also highlighted by Weifei et al. [21]. There is a notable absence of attention to environmental coupling, leading not only to inaccurate representations of virtual components in existing DT models but also to a deficiency in controlling the actual object and making informed decisions. However, in an environment where a DDT is running, a two-way communication is very important. OpenTwins, on the other hand, offers the possibility not only to receive data from the physical part of the twin but also to control and send messages to the device. This functionality is enhanced with its distributed version, as it enables to combine different and distributed DTs and executes these interactions in a shorter period of time for scenarios where a low latency is required [22].

In a previous work, we defined the OpenTwins framework [3], an open-source solution for the development of next-gen DTs. In this work, a new distributed architecture of OpenTwins has been designed to distribute the necessary computations to the edge and IoT devices, achieving a DDTs and saving in bandwidth and latency when required.

As outlined in this section, there is a challenge to discover open source frameworks designed to create general purpose DTs that seamlessly integrate visual rendering, IoT, artificial intelligence (AI) and simulation among others beyond specific implementations and therefore frameworks that offer the ability to distribute all components of the DT to work together that can be used across a wide range of devices. Table 1 summarizes the characteristics of the works studied with respect to our proposal.

3. Lightweight distributed architecture of OpenTwins

3.1. Background

OpenTwins, offers several possibilities for managing DT. These possibilities include, among others, ML functionalities, or 3D visualization of the DTs. A simulation component was recently added to support not only simulation agents developed in any programming language, but also the support of FMI standard, which is one of the most widely used simulation standards in industry nowadays.

The OpenTwins open source platform was originally designed to work on a fog-cloud server where all the necessary data is received to develop a complete DT, as well as all the elements of simulation, ML, etc. that may be needed for the development of these DT. This architecture can be observed in Fig. 1, and it is composed by:

- **Eclipse Ditto:** It acts as a central piece that shapes the DT schematic, establishes the connection of data received, etc.
  - **MongoDB:** Auxiliary DB used to store data relative to Eclipse Ditto components.
- **Ditto Extended API:** Component developed to bring extra functionalities to Eclipse Ditto.
- **InfluxDB:** Brings the possibility to store historical data from DTs.
- **Simulation support:** It gives the DTs the possibility to add simulation capabilities to their behavior.
- **ML components:** Through Kafka-ML [23], OpenTwins offers ML functionalities with data streams.
- **Grafana:** OpenTwins uses Grafana as front-end, as is open-source and gives multiple visual representation tools.
- **Unity support for Grafana:** A very important aspect of the DTs is the reliable visualization of the physical part. That is why OpenTwins supports the visualization and interaction of 3D models using Unity[1] game engine.

3.2. Need of distributing OpenTwins

The field of DT is closely associated with the realm of the IoT, emphasizing the essential need to minimize data transmission delays. Furthermore, IoT devices often face constraints in processing power, making it impractical to run resource-intensive software. Consequently, there is a compelling requirement to develop a lightweight version of the OpenTwins platform, designed to function as a distributed alternative.

The architecture of this new DDT has been developed to facilitate dependable IoT applications within a distributed Edge/Fog/Cloud infrastructure. Developing a lightweight distributed version of OpenTwins can offer several advantages and address various needs depending on the context and goals of the project. Here are some of the key reasons why creating a distributed version of OpenTwins was important:

1. **Reduced Resource Consumption:** Removing different components of the main platform and reducing the load of others make the lighter version of the platform an attractive alternative, depending on the use case.
2. **Reduced Bandwidth Usage:** Since some processing is performed on the device, the bandwidth used can be reduced, making it suitable for use cases where bandwidth is limited or in areas with slower internet connections.
3. **Adaptability to Different Devices:** DDTs are not meant to be just running into the x86 CPU architecture. A considerable amount of the DDTs will run into IoT ARM devices. Knowing this, lightweight version OpenTwins should be compatible with different CPU architectures.
4. **Scalability:** The distribution of DTs is directly related to their composition, and therefore to the scalability of the platform. This re-definition of the platform is designed to achieve simple and efficient scalability, not only with light versions but with the original version of OpenTwins.

[1] https://unity.com/

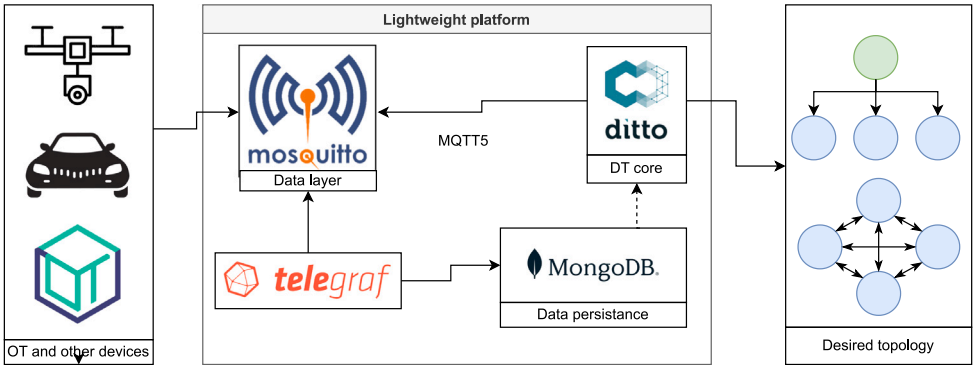


Fig. 2. Lightweight version of the OpenTwins architecture.

5. **Privacy and security:** DDTs enhance security and privacy by decentralizing sensitive data, reducing the risk of a single point of failure. It allows for granular access control, ensuring only authorized personnel can access specific components. This architecture also ensures compliance with privacy regulations, and enables data anonymization and localized processing [24].

All these features lead to the fact that the distributed version of OpenTwins must be designed to be more efficient in terms of resource utilization, leading to better performance in scenarios with limited processing power, memory, or bandwidth, being an alternative version to the original architecture not to replace it but with the objective of combining the use of these architectures to create an ecosystem of DTs covering the maximum amount of possible needs.

3.3. Development of a lightweight version of OpenTwins

Like the main platform, the core component of this new architecture remains Eclipse Ditto. Eclipse Ditto is an open-source framework that facilitates the creation of schematics and connections for the development of DTs. The main difference lies in the rest of the OpenTwins components:

- **Message broker:** The Apache Kafka messaging broker has been replaced by Mosquitto that uses MQTT5, as it is best suited messaging protocol for IoT [25].
- **Persistence:** As the platform is designed to be used in low-resource environments such as IoT devices or Raspberry Pi, for example, it has been decided not to have persistence of historical data, so InfluxDB, that is a database specifically designed for time series is not used. In the case of a persistence need, the MongoDB database that Eclipse Ditto needs to work can be used. Telegraf, that is a agent for collecting and reporting metrics, can introduce the historical data in this database. In this way, although we do not have a database optimized to store data in the form of time series, it would be possible to have persistence with a single database.
- **Visualization:** Both IoT and Edge devices are not typically used for visualization-related tasks and are sometimes lacking the power to do so. This is why it has been decided to remove this component from the new lightweight architecture.
- **Simulation components:** Due to the technical limitations mentioned above, the simulation components present in the original architecture have been eliminated.
- **ML components:** As mentioned above, some edge or IoT are not capable of handling the necessary components to run the ML module, so its installation, although it has been used in this work, is optional.

Therefore, all the points previously discussed in Section 3.2 have been covered with the lightweight version of OpenTwins. Thanks to the reduction of the number components in OpenTwins, we have managed to **reduce the resource consumption** so it can be executed on different devices. By distributing the DT using this new architecture we managed to **reduce the bandwidth** used by reducing the number of data sent and therefore **improve the scalability** of the final DT. This also **improves privacy and security** by keeping sensitive data within the device itself without having to send it to a central server. Finally, to **achieve adaptability for different devices**, the OpenTwins requirements for applications have been modified. Now, the components used by the lightweight version of OpenTwins are all **ARM compatible**. This together with having reduced the necessary resources we have achieved that OpenTwins is a tool compatible with ARM devices capable of running on a Raspberry Pi 4.

This new architecture can be observed in Fig. 2. All modules that are not present in this version of the platform are still fully available and compatible, so users can make use of any component if necessary by activating them in the platform installation process by Helm,[2] therefore the lightweight version of OpenTwins remains as an open source platform.[3]

3.4. Interaction between main architecture and lightweight version

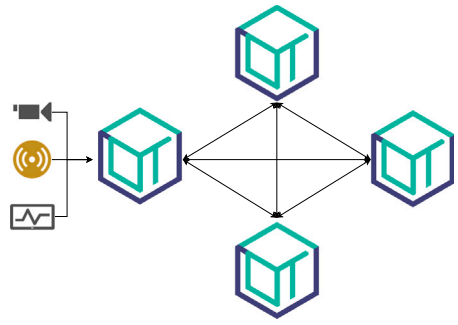
The interaction between the new architecture of OpenTwins and the original version is simple, even in scenarios where the interconnection spans beyond two instances of the platform.

The process of establishing DTs and transmitting their data to the lightweight version of OpenTwins operates seamlessly, following the same principles as in the original platform, connecting the different OpenTwins instances through the Eclipse Ditto connections so that messages are sent from one to the other. Notably, the distinction lies in the method of data transmission, which can now be accomplished through either HTTP or the MQTT broker, chosen for its lighter weight compared to previously utilized alternatives.

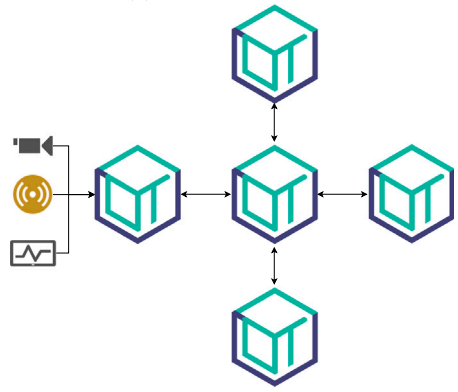
The interconnection and data transmission between distinct instances can be accomplished through three distinct methods:

- **P2P type architecture:** Peer-to-peer (P2P) networks [26] are a type of decentralized network where participants, called peers, communicate and share resources directly with each other without relying on a central server or authority. This type of connection requires configuring the different OpenTwins instances so that the data received from the different sensors is forwarded to each and every other instance. This type of architecture can be observed in Fig. 3(a).

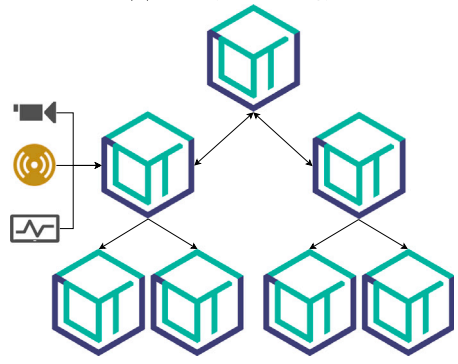
[2] https://helm.sh/
[3] https://github.com/ertis-research/opentwins



(a) P2P type topology.



(b) Star type topology.



(c) Tree type topology.

Fig. 3. Types of connection topologies in OpenTwins distributed architecture.

- **Star architecture:** Star networks [27] are a type of topology where all nodes are connected to a central hub. In a star network, the central hub acts as a repeater to transmit data between the connected devices. Each device in the network communicates directly with the central hub, and communication between devices occurs through the hub. In this type of topology, an OpenTwins instance, usually one that has the original architecture, acts as a hub, allowing the exchange of messages from other instances through it. This type of architecture can be observed in Fig. 3(b).

- **Tree architecture:** Tree networks [28] are a network topology in which the nodes are arranged in the form of a tree. From a topological point of view, it is similar to a series of interconnected star networks except that it does not have a central hub. In this type of topology, the main OpenTwins architecture and distributed architecture instances alternate in the most appropriate way for the transmission of data through the tree. This type of architecture can be observed in Fig. 3(c).

The selection between these architectures, or a hybrid approach, depends on the specific use case. For instance, in the context of creating a DT for a factory, opting for a P2P topology is advisable when real-time status updates between process components are crucial. Alternatively, for a global structure where fast message exchange is not required, it may be advantageous to use a star or tree architecture as it is simpler to deploy. This requires interconnecting various process twins, forming a cohesive combined architecture.

4. Use case: Distributed digital twin for critical decisions on autonomous driving

In the field of autonomous driving, DTs can serve as virtual replicas of physical vehicles, replicating their behavior and characteristics in a virtual environment. For instance, the insurance company Mapfre investigates the behavior of autonomous vehicles for risk assessment through DTs.[4] These digital replicas are essential for simulating, monitoring and analyzing real-world scenarios, thus ensuring the safety and efficiency of autonomous systems. In a city where a fleet of autonomous vehicles operates, each vehicle is equipped with a set of sensors, cameras and processing units, continuously collecting data about the vehicle's environment. This data is essential for the vehicle to make decisions and drive safely on the road.

However, the challenge emerges when this data has to be processed to ensure safe navigation. Traditionally, data is sent to a centralized DT server for analysis, where algorithms evaluate the information and send instructions to the vehicle. However, this process can introduce latency, which can be an impediment in critical response situations (e.g. detecting a hazard while driving). In addition, increasing the number of vehicles can increase the computational capacity required to operate this DT and can lead to a bottleneck on the server side, breaking the operation of the DT and causing major problems in the operation of the physical counterpart of the DT.

In this scenario, DDTs can minimize latency while ensuring the safety of autonomous vehicles. Instead of relying on a centralized server for processing, each vehicle has its own onboard DT. This DT on board is essentially a virtual representation of the vehicle, capable of processing sensor data and making decisions in real time . By exploiting DDTs, autonomous vehicles can react to hazards through data-driven models instantly, without the need for data to go back and forth to a centralized server, thus increasing safety in autonomous driving scenarios. In this aspect, data processing is carried out inside the vehicle, reducing possible security risks, an important aspect highlighted in [20], by only sending the required information to the central server and keeping the critical data private, such as in the case of federated learning in the field of ML. Moreover, the other part of the DDT in the cloud allows to have an overview of the twin (vehicle) situation, as for example to asset their interaction, as in the example of the insurance sector.

Reducing communication latency between devices and servers is critical in autonomous driving, so bringing these elements to the device itself is essential to reduce latency, a potential bottleneck if the number of vehicles increases and possible data loss. Furthermore, in some cases, it might be necessary to send data to the central server for other purposes, such as training ML models, data visualization, etc.

In this paper, a use case concerning autonomous driving is proposed to compare the difference between the use of a centralized DTs and the use of a DDTs with this architecture, demonstrating the feasibility of the platform.

To achieve these requirements for the development of this DDT the aforementioned star architecture has been used, where an instance of OpenTwins, located on a central server acts as a central hub and a distributed lightweight instance of OpenTwins is located in the autonomous vehicle. A schema of this use case can be observed in Fig. 4.

[4] https://www.mapfre.com/en/insights/innovation/investigating-autonomous-vehicles/

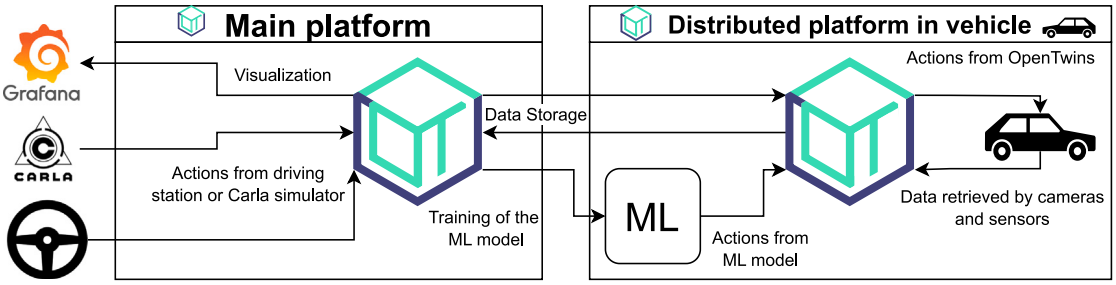


Fig. 4. Schema of the digital twin for autonomous driving.

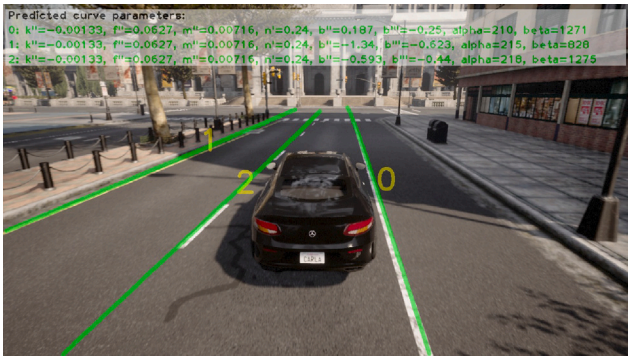


Fig. 5. Predicted image from the autonomous vehicle use case.

This DDT is powered by a ML model. The ML model has been developed with ailia SDK[5] using PyTorch 1.8 and trained with a dataset of real images.[6] This model, that is a LSTR[7] [29] (Lane Shape Prediction with Transformers) is a ML model that uses transformers [30] to detect the trajectory of the road lines to keep the car on a correct trajectory. A predicted image using this LSTR can be observed in Fig. 5.

5. Evaluation

To assess the functionality and performance of the new OpenTwins architecture, a series of tests will be conducted. These tests will focus on two key aspects: latency measurement and resource consumption comparison. These two tests are fundamental to evaluate the performance and efficiency of the proposed new architecture and the importance of DDTs.

The first test, which measures latency, is crucial for evaluating the speed and responsiveness of the platform and the distribution of DTs. Therefore, this test will allow us to determine whether the new architecture is capable of delivering fast and consistent response times under various load conditions.

The second test focuses on the resource consumption and allows us to understand how the new platform performs in terms of utilization of resources such as memory, processing power and bandwidth. This comparison will help in determining whether the new architecture achieves more efficient resource utilization compared to the main architecture.

All the necessary to replicate the experiment can be found in a github repository.[8]

5.1. Experimental setup

The following components, both hardware and software, have been considered to carry out the evaluation:

- **Hardware.** The experiment has been performed on a 7-node Kubernetes cluster. Each node has an Intel Xeon Gold 6230R CPU, two Nvidia Tesla V100 GPUs, and 384 GB of RAM. All the tests from which the measurements are collected and the distributed architecture is running, were launched from a different PC on the same wired network with an Intel Core i9-9900k and 64 GB of RAM.
- **Software.** Each of the nodes runs Kubernetes v1.21.6 and Docker 20.10.8 over Ubuntu 20.04.3 LTS.

5.2. Test 1 - Latency comparison between distributing DTs and executing on server

To evaluate the performance of the platform and provide insights into the improvement obtained by distributing the DTs and thus their functions, a demonstration of decision-making in critical environments as in the proposed use case was developed.

This test aims to measure the difference between distributing the DTs and keeping the functions in the central server, also taking into account the difference in the ML model's inference time execution from the test PC and the cloud. Two scenarios were considered:

1. **First scenario:** Collection and sending of data through the entire platform instance in the central server to the ML module and back to the device to measure latency. This is the scenario of the OpenTwins original platform without the distributed scenario considered in this paper.
2. **Second scenario:** In this case, the distributed approach of OpenTwins presented in this article is considered, where decision-making is executed in the device itself. Afterward, data is sent to the central platform for visualization and data storage. This data flow can be observed in Fig. 6.

The data sent to OpenTwis are images generated by the CARLA simulator[9] (CARLA is a car simulator created to support development, training, and validation of autonomous driving systems). Images is a new type of data that had not been considered so far, so it was necessary to adjust the parameters until finding the right configuration. For example, Eclipse Ditto needs to store the data in JSON format, so the image had to be in a serializable JSON format. This forced the image to be transformed to base64 in order to get a string to include in JSON.

However, when sending the images to the MQTT broker it was found that the length of the message, even adjusting the broker to use the maximum possible, was greater than allowed, so it was necessary to rescale the images obtained from CARLA. Once this problem was

[5] https://github.com/axinc-ai/ailia-models/tree/master/road_detection/lstr

[6] https://www.kaggle.com/code/hikmatullahmohammadi/road-lane-line-tusimple-dataset-preparation

[7] https://github.com/liuruijin17/LSTR

[8] https://github.com/sergioinf/ddt-opentwins

[9] https://carla.org/

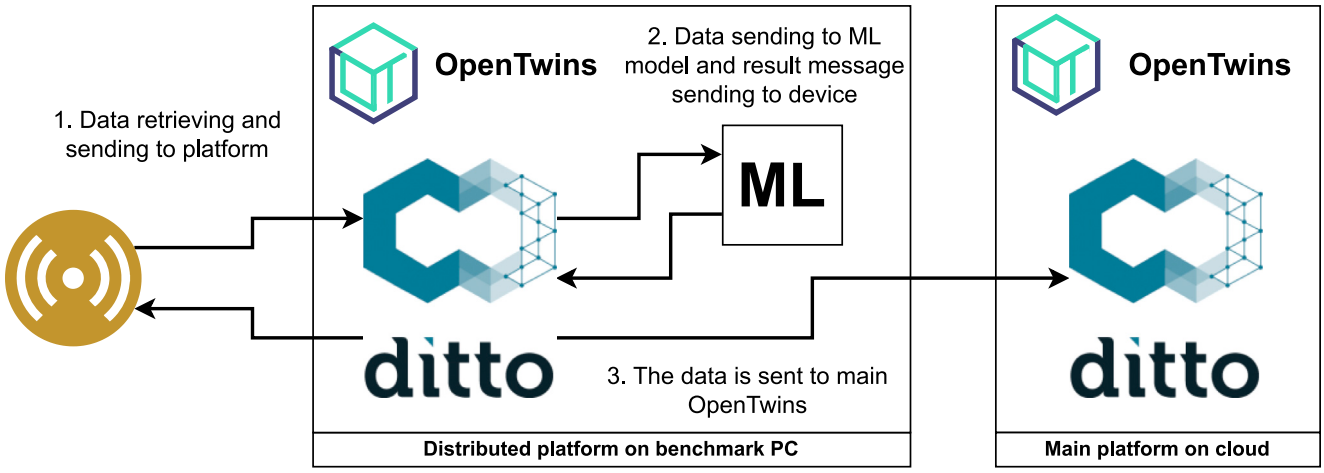


Fig. 6. Dataflow of DDTs interactions carried out in test 2.

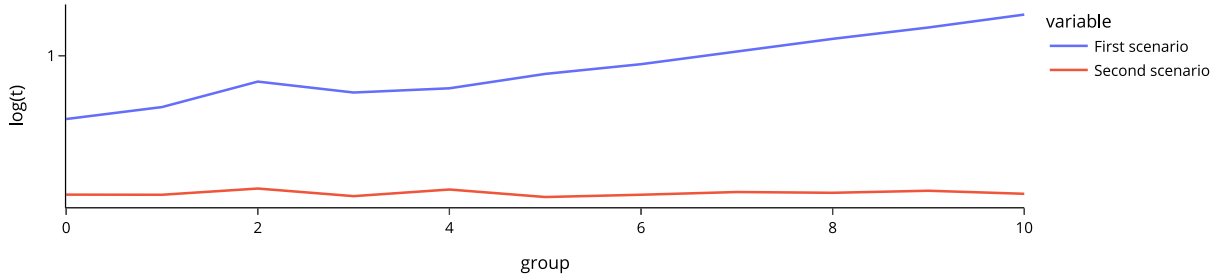


Fig. 7. Elapsed time from the data retrieving to response arrival.

solved, we found that Eclipse Ditto also has a schema length limit of a twin, which forced us to reduce the size of the image until we could include it as a DT parameter in Eclipse Ditto.

This test has been split into several simulation batches, from batch 0 to batch 10. Every batch contains a total of $(batchNumber * 10)$ messages simultaneously, and stores the elapsed time of travel of every message. When all the data produced has returned, all stored timestamps are taken to calculate the elapsed time of the execution. The result of the test is an average of 10 executions per batch. Those results can be seen in Fig. 7. This figure shows, on a logarithmic scale, the latency in both proposed scenarios.

The first scenario shows the latency obtained in the case of not distributing the DT functionalities, where the data travels from the device to a central server where the necessary functionalities are executed, in this case, a critical detection model, and back to the device.

On the other hand, the second scenario shows the latency obtained in the case of distributing the DT, where the data travels through the OpenTwins platform within the Edge device itself and executes the decision-making process on-site.

In scenario 1, corresponding to the classic DT architecture, it is observed that the latency in the process is higher than in scenario 2. This is mainly due to the higher number of information transmissions made in this scenario and to the larger size of the data being processed. In addition, a progressive increase in latency is observed as the test load increases. At the beginning, the classical architecture offers acceptable performance, but with increasing requests, the latency starts to increase. This sustained increase suggests that the centralized nature of this type of DT presents bottlenecks as the number of devices grows, which negatively affects its responsiveness.

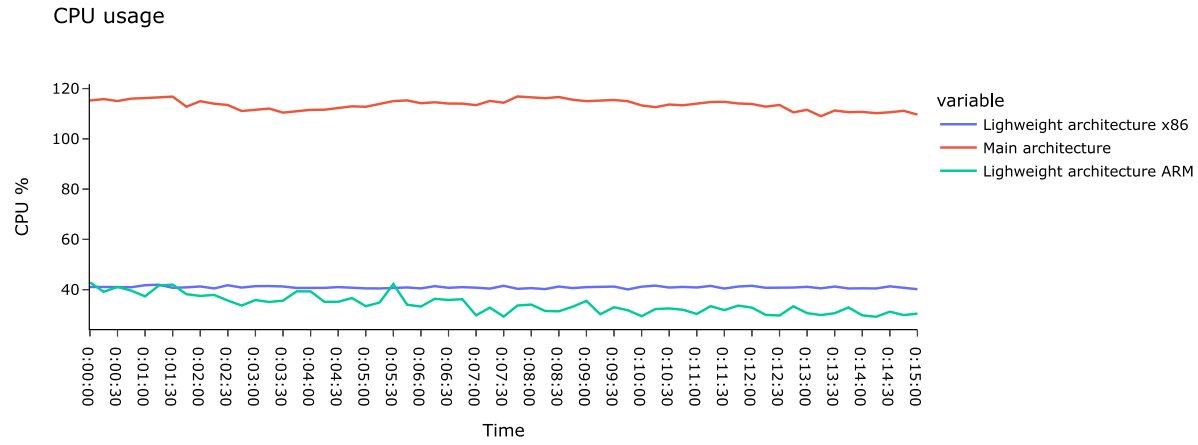
In contrast, scenario 2, based on a distributed architecture, shows much more stable behavior. Latency remains consistently below the levels observed in the classic DT architecture, even under higher loads. Due to the decentralized nature of the distributed architecture, data processing is performed more efficiently at different DDT nodes, which avoids single-point saturation and improves overall response time.

Although both scenarios have low latency under light loads, even though they were lower in the distributed version, the differences become noticeable when the number of requests increase. The classical architecture does not scale so effectively, resulting in a considerable increase in latency, while the distributed architecture manages to keep its performance under control. This not only highlights the advantage of latency reduction, load distribution and scalability that DDTs offer, but also exposes the better performance of the lightweight, distributed version of OpenTwins.

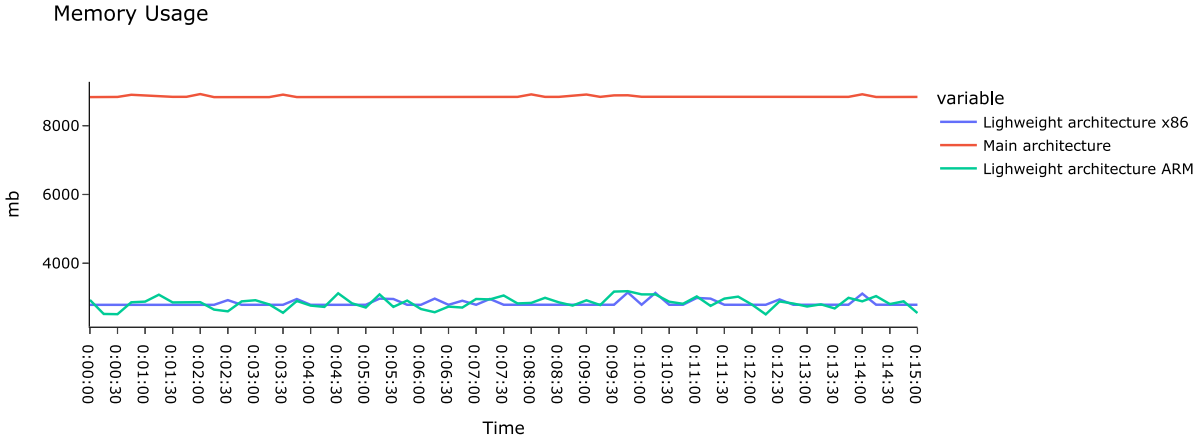
5.3. Test 2 - Comparison between consumption of two architectures

To demonstrate that the new OpenTwins architecture really consumes fewer resources than the original one, the consumption of each architecture has been measured. All metrics were taken on the same equipment under the same circumstances and the result of the test is an average of 5 different data collections. These metrics are generated by the cluster itself thanks to the Google cAdvisor tool integrated in Kubernetes, a open source platform for managing containerized workloads and services, and then stored in Prometheus for further study. Prometheus is an open-source technology designed to provide monitoring and alerting functionality for cloud-native environments. These metrics provide detailed data on the resource consumption of running containers, covering aspects such as CPU, memory, network and storage usage. In addition, the consumption of the lightweight platform has been measured on a Raspberry Pi 4 of 4 GB.

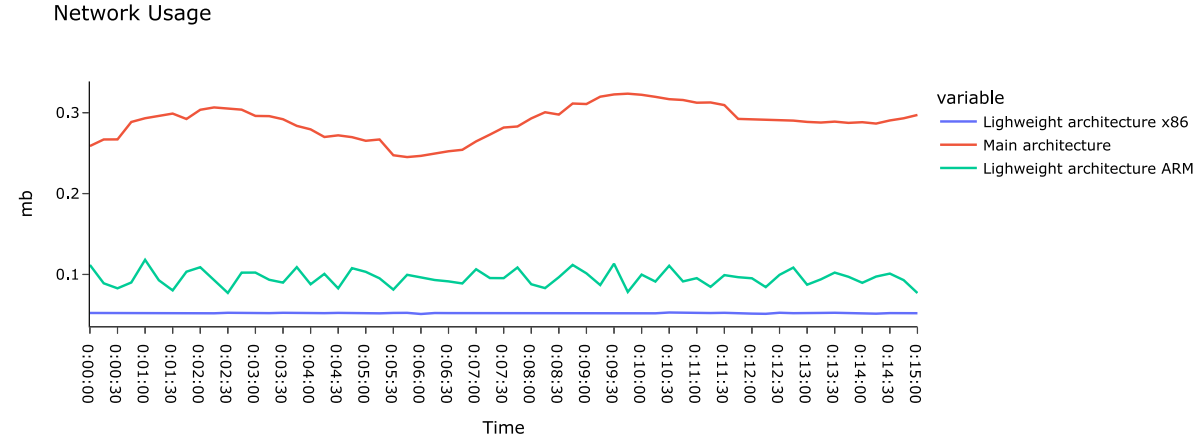
1. **CPU consumption.** In Fig. 8(a) the comparison of CPU consumption of both architectures can be observed. This test measures the percentage of CPU used every 15 min being 100% 1 core, 200% 2 cores, etc. It can be observed that the main architecture is composed of many more components than the



(a) Comparison of CPU consumption.



(b) Comparison of memory consumption.



(c) Comparison of bandwidth consumption

Fig. 8. Comparison of resource consumption between main and lightweight architectures.

lightweight version. Consequently, the consumption increases. Based on the results obtained, it can be seen that the lightweight architecture consumes 3 times less CPU than the original one.

2. **Memory consumption.** In this test, memory usage in mb has been measured. As mentioned before, as number of components increases, the memory consumption increases to. In this case, the most consuming component in both architectures is MongoDB, but in the main architecture, MongoDB consumes more memory because there are more components that store information in this database. This result can be observed in Fig. 8(b).
3. **Bandwidth consumption.** In this test, bandwidth usage in mb has been measured. In this case, the difference between the two

architectures is even greater. In Fig. 8(c), we can see how the bandwidth consumption of the lightweight architecture is up to 5 times lower than the original architecture.

4. **ARM consumption.** As an additional test, the lightweight architecture has been deployed on a 4 GB Raspberry Pi 4 to evaluate the platform's consumption in an ARM environment. As can be observed in Figs. 8(a), 8(b) and 8(c), the consumption of the lightweight version of OpenTwins remains very similar to the consumption on an x86 architecture, remaining below the consumption of the original architecture.

These results reveal how, in general, the new architecture is lighter than the original architecture, reducing CPU, memory and bandwidth consumption up to 3 times. This is due to the reduction of elements present in the platform. While this condition limits OpenTwins functionalities, such as data visualization or historical data storage, it makes this new architecture perfect for IoT or edge environments, where resources are more limited.

6. Conclusions and future work

In this article an architecture for distributed digital twins based on the open-source OpenTwins framework is presented. This architecture enables the seamless integration of distributed and autonomous digital twins that can collaborate each other to create high-level entities. This new architecture is not only lighter but also focused on the creation of DDTs and their deployment in IoT and edge systems, which sometimes do not have the necessary resources to properly run all the original OpenTwins components, highlighting the benefits of DDT.

To achieve seamless distribution of DTs, this new platform is fully compatible with the original platform, creating an ecosystem of DTs capable of running on a multitude of devices and making the interaction of twins easier.

To validate this new platform we have presented a use case where decision-making in critical environments should be carried out as fast as possible, thus avoiding unnecessary latencies, as is the case of safety in autonomous driving. Thanks to the tests performed, we have been able to confirm that the new version of OpenTwins, combined with the distribution of DTs, not only provides a low latency for scenarios where critical decision is needed and a reduction of server bottleneck risk but also a reduction in resource consumption.

During the development of this work we have observed different challenges to be addressed in the future, such as the correct coordination of the DDTs parts or the resistance to failures and disconnection, therefore, it is necessary to continue researching and developing to refine and extend the new proposed architecture. In future work, we have outlined the following challenges and improvements to be carried out:

- Testing in a larger number of devices and connection difficulties, in order to test the scalability and resilience of the platform. Through the development of a DDT of an array of configurable antennas as part of a starting project.
- Study and incorporate a functionality within OpenTwins that autonomously coordinates the data sent when more than 2 instances of the platform come into play. Receiving outdated messages can cause malfunction in the DTs.
- As part of a project, the development of a DDT of an irrigation field, combining the main platform and the lightweight version in different agricultural process to study the behavior of the combinations of different types of architectures.
- Transform the simulation service to be executed in different devices being coordinated over the whole DT, allowing to execute joint simulations of the same DT from different devices.

CRediT authorship contribution statement

Sergio Infante: Writing – review & editing, Writing – original draft, Investigation, Conceptualization. **Julia Robles:** Writing – review & editing, Writing – original draft, Investigation, Conceptualization. **Cristian Martín:** Writing – review & editing, Writing – original draft, Conceptualization. **Bartolomé Rubio:** Writing – review & editing, Writing – original draft, Conceptualization. **Manuel Díaz:** Writing – review & editing, Writing – original draft, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Manuel Diaz reports financial support was provided by Spain Ministry of Science and Innovation.

Acknowledgments

This work is funded by the Spanish projects: Grant TSI-063000-2021-116 ('5G+TACTILE_2: Digital vertical twins for B5G/6G networks') funded by MICIU/AEI/ 10.13039/501100011033 and by 'European Union NextGenerationEU/PRTR'. Grant PID2022-141705OB-C21 ('DiTaS: A framework for agnostic compositional and cognitive digital twin services') funded by MICIU/AEI/10.13039/501100011033/ and by 'FEDER, Spain'. Grant TED2021-130167B ('GEDIER: Application of Digital Twins to more sustainable irrigated farms'), funded by MICIU/AEI/10.13039/501100011033/ and by 'European Union NextGenerationEU/PRTR'. Grant MIG-20221022 ('GEDERA: Intelligent Flexible Energy Demand Management in Coupled Hybrid Networks'), funded by MICIU/AEI/10.13039/501100011033/ and by 'European Union NextGenerationEU/PRTR'. Funding for open access charge: Universidad de Malaga/CBUA.

Data availability

The data used for this research is available on github. A link is include in the article.

References

[1] A. Rasheed, O. San, T. Kvamsdal, Digital twin: Values, challenges and enablers from a modeling perspective, Ieee Access 8 (2020) 21980–22012.
[2] A.J. Chaves, C. Martín, M. Díaz, The orchestration of machine learning frameworks with data streams and GPU acceleration in kafka-ML: A deep-learning performance comparative, Expert Syst. (2023) e13287.
[3] J. Robles, C. Martín, M. Díaz, OpenTwins: An open-source framework for the development of next-gen compositional digital twins, Comput. Ind. 152 (2023) 104007, http://dx.doi.org/10.1016/j.compind.2023.104007.
[4] S. Barykin, I. Kapustina, S. Sergeev, O. Kalinina, V. Vilken, E. De la Poza, Y. Putikhin, L. Volkova, Developing the physical distribution digital twin model within the trade network, Acad. Strateg. Manag. J. 20 (1) (2021) 1–24.
[5] U.A. Bukar, M. Othman, Architectural design, improvement, and challenges of distributed software-defined wireless sensor networks, Wirel. Pers. Commun. 122 (3) (2022) 2395–2439.
[6] M. Eddoujaji, H. Samadi, M. Bohorma, Data processing on distributed systems storage challenges, in: Networking, Intelligent Systems and Security: Proceedings of NISS 2021, Springer, 2022, pp. 795–811.
[7] A. Costantini, G. Di Modica, J.C. Ahouangonou, D.C. Duma, B. Martelli, M. Galletti, M. Antonacci, D. Nehls, P. Bellavista, C. Delamarre, et al., Iotwins: Toward implementation of distributed digital twins in industry 4.0 settings, Computers 11 (5) (2022) 67.
[8] M. Moshrefzadeh, T. Machl, D. Gackstetter, A. Donaubauer, T.H. Kolbe, Towards a distributed digital twin of the agricultural landscape, J. Digit. Landsc. Archit. 5 (2020) 173–118.
[9] S. Infante, C. Martín, J. Robles, B. Rubio, M. Díaz, R.G. Perea, P. Montesinos, E.C. Poyato, Integrating FMI and ML/AI models on the open-source digital twin framework OpenTwins, Softw. - Pract. Exp. (2024).

[10] J.L. Vilas-Boas, J.J. Rodrigues, A.M. Alberti, Convergence of distributed ledger technologies with digital twins, IoT, and AI for fresh food logistics: Challenges and opportunities, J. Ind. Inf. Integr. 31 (2023) 100393, http://dx.doi.org/10.1016/j.jii.2022.100393.

[11] Y. Zhou, A.N. Manea, W. Hua, J. Wu, W. Zhou, J. Yu, S. Rahman, Application of distributed ledger technology in distribution networks, Proc. IEEE 110 (12) (2022) 1963–1975.

[12] M.F. Bado, D. Tonelli, F. Poli, D. Zonta, J.R. Casas, Digital twin for civil engineering systems: An exploratory review for distributed sensing updating, Sensors 22 (9) (2022) 3168.

[13] A. Hürkamp, S. Gellrich, T. Ossowski, J. Beuscher, S. Thiede, C. Herrmann, K. Dröder, Combining simulation and machine learning as digital twin for the manufacturing of overmolded thermoplastic composites, J. Manuf. Mater. Process. 4 (3) (2020) 92.

[14] M. Dietz, M. Vielberth, G. Pernul, Integrating digital twin security simulations in the security operations center, in: Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, pp. 1–9.

[15] B. Leng, S. Gao, T. Xia, E. Pan, J. Seidelmann, H. Wang, L. Xi, Digital twin monitoring and simulation integrated platform for reconfigurable manufacturing systems, Adv. Eng. Inform. 58 (2023) 102141, http://dx.doi.org/10.1016/j.aei.2023.102141, URL https://www.sciencedirect.com/science/article/pii/S1474034623002690.

[16] A. Aziz, S.S. Chouhan, O. Schelén, U. Bodin, Distributed digital twins as proxies-unlocking composability & flexibility for purpose-oriented digital twins, IEEE Access (2023).

[17] O. San, S. Pawar, A. Rasheed, Decentralized digital twins of complex dynamical systems, Sci. Rep. 13 (1) (2023) 20087.

[18] A. Lektauers, J. Bikovska, V. Bolsakovs, An agent-directed digital twin framework for simulation-based training, in: 2022 63rd International Scientific Conference on Information Technology and Management Science of Riga Technical University, ITMS, 2022, pp. 1–6, http://dx.doi.org/10.1109/ITMS56974.2022.9937103.

[19] A.J. Chaves, C. Martín, M. Díaz, Towards flexible data stream collaboration: Federated learning in Kafka-ML, Internet Things 25 (2024) 101036.

[20] C. Schwarz, Z. Wang, The role of digital twins in connected and automated vehicles, IEEE Intell. Transp. Syst. Mag. 14 (6) (2022) 41–51, http://dx.doi.org/10.1109/MITS.2021.3129524.

[21] W. Hu, T. Zhang, X. Deng, Z. Liu, J. Tan, Digital twin: A state-of-the-art review of its enabling technologies, applications and challenges, J. Intell. Manuf. Special Equip. 2 (1) (2021) 1–34.

[22] K. Kušić, R. Schumann, E. Ivanjko, A digital twin in transportation: Real-time synergy of traffic data streams and simulation for virtualizing motorway dynamics, Adv. Eng. Inform. 55 (2023) 101858, http://dx.doi.org/10.1016/j.aei.2022.101858, URL https://www.sciencedirect.com/science/article/pii/S1474034622003160.

[23] C. Martín, P. Langendoerfer, P.S. Zarrin, M. Díaz, B. Rubio, Kafka-ML: connecting the data stream with ML/AI frameworks, Future Gener. Comput. Syst. 126 (2022) 15–33.

[24] Y. Wang, Z. Su, S. Guo, M. Dai, T.H. Luan, Y. Liu, A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects, IEEE Internet Things J. 10 (17) (2023) 14965–14987, http://dx.doi.org/10.1109/JIOT.2023.3263909.

[25] M. Singh, M. Rajan, V. Shivraj, P. Balamuralidhar, Secure MQTT for internet of things (IoT), in: 2015 Fifth International Conference on Communication Systems and Network Technologies, 2015, pp. 746–751, http://dx.doi.org/10.1109/CSNT.2015.16.

[26] S. Hong, P2P networking based internet of things (IoT) sensor node authentication by blockchain, Peer-to-Peer Netw. Appl. 13 (2) (2020) 579–589.

[27] C. Li, S. Lin, S. Li, Structure connectivity and substructure connectivity of star graphs, Discrete Appl. Math. 284 (2020) 472–480.

[28] S. Sørensen, D. Pisinger, Reconstructing the tree topology in telecommunication networks, in: 32nd European Conference on Operational Research, 2022.

[29] R. Liu, Z. Yuan, T. Liu, Z. Xiong, End-to-end lane shape prediction with transformers, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 3694–3702.

[30] T. Lin, Y. Wang, X. Liu, X. Qiu, A survey of transformers, AI open 3 (2022) 111–132.