Application-Driven Network-Aware Digital Twin Management in Industrial Edge Environments

Paolo Bellavista, *Senior Member, IEEE*, Carlo Giannelli, *Senior Member, IEEE*, Marco Mamei, Matteo Mendula, and Marco Picone

***Abstract*—The application of Internet of Things (IoT) within industrial environments is fostering the adoption of the digital twin (DT) approach, applied at the edge of the network to handle heterogeneity stemming from siloed application management solutions and from protocols originated by different manufacturing tools and enterprise services. In this challenging context, network heterogeneity also represents a critical element that can significantly limit the design and deployment of DT-oriented applications. This article proposes the Application-driven digital twin networking middleware with the twofold objective of: 1) Simplifying the interaction among heterogeneous devices by allowing DTs to exploit IP-based protocols instead of specialized industrial ones and to enhance packet content expressiveness, by enriching data via well-defined standards. 2) Dynamically managing network resources in edge industrial environments, applying software defined networking to exploit the communication mechanisms most suitable to application requirements, ranging from native IP to more articulated based on packet content.**

***Index Terms*—Application-driven management, digital twin (DT), edge computing, industrial IoT, software defined networking (SDN).**

I. Introduction

THE adoption of the Internet of Things (IoT) is currently spreading in industrial environments. Complex Industrial IoT (IIoT) applications stem from the joint exploitation of multiple and heterogeneous devices adopting multiple protocols and data formats both co-located in the same plant and remotely spread in different locations. Such devices need to coordinate their activities to support industrial operations (e.g., manufacturing and assembly) with stringent quality of service (QoS) and safety critical requirements.

The combination of IoT with the industrial ecosystem has recently revitalized the concept of digital twin (DT) as an innovative technology expected to transform the industrial and manufacturing ecosystems. Promising improvements are expected to offer new innovative solutions reducing costs, monitoring assets, optimizing maintenance, reducing downtime, and enabling the creation of intelligent connected products. As stated in [1], a DT can be defined as a comprehensive software representation of an individual physical device including its properties, conditions, and behavior across the entire object's life-cycle. These novel twin-oriented manufacturing systems are characterized by the possibility to support and handle the massive heterogeneity of siloed distributed implementations together with protocols and data flows originated by different manufacturing and enterprise services [2], [3].

In this challenging context, network heterogeneity represents a critical element to efficiently handle complex industrial environments and may also significantly limit the design and deployment of DT-oriented architectures and applications. DTs, consumers, and services should be unaware of the complexity behind their communications and should be resilient to re-configuration and dynamic orchestration. The layering and separation of functionalities represent a key element, 1) to decouple the networking infrastructure from upper layers and, 2) to dynamically control the communications according to applications and context requirements, e.g., the creation of a segregated and secured network shared only by a group of selected DTs and target data consumers.

The approach currently adopted in industrial environments is based on the manual configuration of networks, usually consisting of several IP subnets, each one comprising a small set of strictly related equipment pieces supporting specific actions in the production flow. To allow the communication among different subnets, there is the need of properly configuring gateways interconnecting different subnets; this is a time-consuming and error-prone task that discourages the dynamic reconfiguration of industrial networks. Thus, it is generally difficult to modify industrial topologies taking into account the fact that the same industrial site may be used in different moments (or even in the

Manuscript received November 30, 2020; revised March 2, 2021; accepted March 12, 2021. Date of publication March 19, 2021; date of current version July 26, 2021. This work was supported in part by the H2020 IoTwins project (Distributed Digital Twins for industrial SMEs: A big-data platform) funded by the EU under the call ICT-11-2018-2019, under Grant 857191. Paper no. TII-20-5403. *(Corresponding author: Carlo Giannelli.)*

Paolo Bellavista and Matteo Mendula are with the Department of Computer Science and Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: paolo.bellavista@unibo.it; matteo.mendula@unibo.it).

Carlo Giannelli is with the Department of Mathematics and Computer Science, University of Ferrara, 44121 Ferrara, Italy (e-mail: carlo.giannelli@unife.it).

Marco Mamei and Marco Picone are with the Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, 41121 Modena, Italy (e-mail: marco.mamei@unimore.it; marco.picone@unimore.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3067447.

Digital Object Identifier 10.1109/TII.2021.3067447

same moment concurrently) to run different industrial applications and produce different products.

Based on these considerations, this article originally proposes the Application-driven digital twin networking (ADTN) middleware to support the twofold objective of simplifying the interaction with heterogeneous distributed industrial devices and of dynamically managing network resources (also to maximize QoS) in distributed industrial environments by adopting an application point of view. To this purpose, the ADTN middleware considerably improves current literature by

1) Supporting the adoption of the DT abstraction not only to simplify the interaction with actual devices via simple digital twins (SDTs) in charge of mediating the interaction with actual devices (e.g., SDTs serve as protocol gateways that interact with devices via field protocols such as Modbus and with remote SDTs via IP-based protocols such as MQTT [4] or CoAP [5]), but also to represent a complex industrial application as the dynamic coordination of several SDTs composing a unique composed digital twin (CDT);
2) providing the deployment of SDTs on edge nodes, where each edge node typically hosts multiple SDTs related to neighbor devices. Note that edge nodes may interact with each other, thus creating a multihop multipath topology. In other words, edge nodes not only manage traffic related to the devices they are directly connected to, but they also act as intermediary nodes by dispatching packets of devices managed by other edge nodes;
3) simplifying the management of network resources, by exploiting a high-level representation of an industrial application (including involved equipment and QoS requirements) to dynamically and autonomously manage network resources within both a single production site and multiple production sites. The adoption of such a solution can greatly facilitate the management of industrial networks, thus allowing faster and safer configuration of topologies to exploit the same production site for different industrial applications.

The rest of this article is organized as follows. Section II introduces the architectural model of the target industrial environment. Sections III and IV outline our ADTN middleware. Section V presents achieved performance based on our prototype. Section VI compares the proposed solution with the literature. Section VII concludes this article.

II. Architectural Modeling

IoT applications designed to be deployed on edge computing distributed facilities can be modeled and structured using a set of common concepts defining the roles and the responsibilities of each layer and entity. The "edge" here refers to the computing infrastructure that operates close to the origin data sources and physical devices. It is a distributed architecture where data is processed at the periphery of the network, as close to the originating source as possible.

Fig. 1 depicts a schematic and layered overview of a distributed IIoT deployment composed of multiple decentralized edge locations handled by one or multiple interconnected nodes responsible to manage local applications and DTs. Processing, storage, and networking resources, together with data and services, are carefully balanced between cloud and the edge facilities to obtain the best performance according to the target business logic, e.g., real-time decision on the edge and massive data analytics in the cloud (note that cloud management is out of the scope of the article).

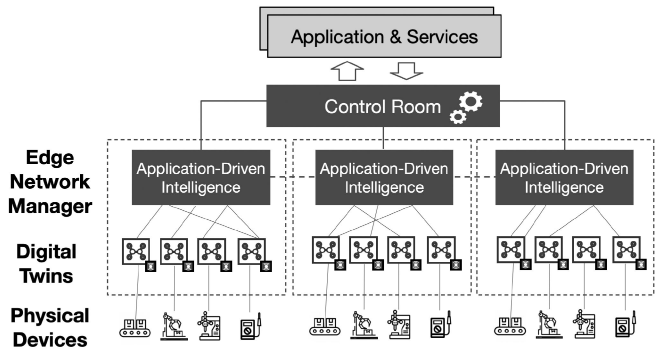

Application & Services
Control Room
Edge Network Manager
Application-Driven Intelligence
Application-Driven Intelligence
Application-Driven Intelligence
Digital Twins
Physical Devices

Fig. 1. High-level scheme of a distributed, edge-based, and intelligent IoT architecture.

At the bottom resides the *Physical Devices* layer, composed of heterogeneous sensors, controllers, and actuators.

Above the Physical Devices layer resides the *Edge* layer, divided into two sublayers: *DTs* and an *Edge Network Manager*. The former comprises digital replicas of original physical counterparts facilitating interaction with physical devices.

The latter models and implements the local application-driven intelligence to: 1) take decisions more quickly and efficiently; 2) reach decisions according to local identity management and access control policies; 3) securing the data close to its source.

Furthermore, it supports application-aware QoS maximization to optimize inter-edge node communication through dynamic network control, based on traffic management rules received from the control room (CR) layer.

The *CR* layer resides close to the Edge layer and is aware of the Edge layer topology, the location of available physical devices (and related DTs), and the current state of industrial and network resources. The CR layer allows technicians to have an overall view of the plant, also managing the Edge layer to dynamically optimize the communication among DTs taking into consideration specific characteristics and QoS requirements.

To better describe our proposal, let us consider the so-called *smartphone assembly* industrial use case where a shop floor acts as a (simplified) smartphone manufactury. It consists of: i) a conveyor (CO) carrying parts from the warehouse; ii) an assembler (AS) putting parts together; iii) a drill (DR) finalizing the assembly; and iv) a checker (CH) verifying the quality of the final product. The same shop floor can be exploited to assembly different smartphones and thus a centralized controller informs about the kind of smartphone it is currently assembled. To monitor the production, CO notifies to a centralized CR its speed, AS the type and amount of used parts, DR its current vibrations, and CH the amount of finalized products, either faulty or not.

CO, AS, DR, and CH interact with dedicated DTs, each one hosted on a different edge node. The four DTs together compose a unique industrial application with specific QoS requirements. CR exploits the knowledge about the industrial application to properly manage the network. At startup, CR configures the DR edge node to aggregate vibration data in packets containing the average and standard deviation vibration of 1 s periods. In case the amount of vibrations is slightly more than usual, CR reduces the period to 0.1 s, thus increasing bandwidth exploitation, to better assess the state of DR. Moreover, it also activates the video streaming industrial application composed of multiple DTs (one for each surveillance camera SC), to remotely verify the current situation. To this purpose, CR configures the network to provide higher priority to vibration and video stream packets, asking to edge nodes to temporarily store or even discard information related to other applications.

To enable the example above, the proposed solution originally adopts Software Defined Networking (SDN) to dynamically modify how edge nodes manage traffic flows, also considering payloads in a per-packet fine-grained manner. To this purpose, in our proposed novel solution DTs adapt and enhance packet payloads, e.g., to comply with a common syntax, making easier the adoption of advanced traffic engineering mechanisms based on the actual payload content. For instance, AS and DR can structure and model their data through the use of the IETF Sensor Measurement Lists (SenML) data format [6] by leveraging on its already available fields, e.g., measurement type, time references, and versioning, and/or by extending them to include dynamic QoS requirements, e.g., delay tolerance and bandwidth constraints, to foster the adoption of fine-grained SDN-based novel solutions.

III. The Application-Driven DT Networking (ADTN) Middleware

Based on the model presented in Section II, we designed, developed, and experimentally evaluated the original ADTN middleware that supports the dynamic aggregation and configuration of heterogeneous and sparse industrial equipment, represented as a single business unit. In other words, products and services are, respectively, crafted and supported by an aggregation of things, e.g., sensors, actuators, and simple devices, and their dynamic and flexible orchestration is optimized by an SDN-based cross-layer approach taking into consideration application-driven indications together with QoS requirements and network configuration adaptation capabilities.

The ADTN middleware is responsible to handle and effectively orchestrate scalable and reliable communications among physical devices, DTs, and modules with respect to a dynamic set of application-driven rules and indications coming from external authorized services. In particular, primary ADTN components are the *simple digital twin (SDT)* on the edge side and the *composed digital twin (CDT)* on the CR side (see Fig. 2).

SDT is a software agent running at the edge layer providing an effective one-to-one mirroring of a physical IoT device through the digitalization and cloning of all its features and functionalities. Each SDT maintains the communication and synchronization with the associated counterpart creating a standardized and uniform abstraction of the device to enable interoperability and cooperation of devices, services, and applications. Furthermore, the SDT can extend the original device's behavior in terms of supported protocols, integrating with external services, and managing how incoming and outgoing packets are exchanged and internally processed. In other words, the SDT provides the opportunity of dynamically augmenting the information coming from physical things through the conveniently reformatting/preprocessing of headers and payloads or the introduction of additional metadata.

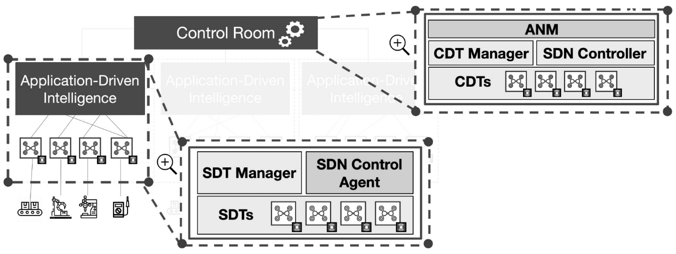

Control Room
ANM
CDT Manager
SDN Controller
CDTs
Application-Driven Intelligence
Application-Driven Intelligence
Application-Driven Intelligence
SDT Manager
SDN Control Agent
SDTs

Fig. 2. Primary components of the ADTN middleware.

CDT is a software component in the CR layer shaping the digital representation of a new entity capable to aggregate multiple SDTs at the same time to efficiently model complex distributed applications and behaviors. In real deployment environments, physical devices are often a composition of different heterogeneous components, and things from different edge locations can participate in common application goals, such as the various tools in the smartphone assembly example. The ability - denoted as *Composability* - of grouping different objects into an aggregated one and then observing and controlling the behavior of the resulting object (as well as the individual entities) represents a strategic feature for DT design and development. Thanks to the standardization and homogeneity obtained through the use of SDTs, a CDT has the ability to easily compose and aggregate multiple twins while abstracting the complexity of a larger system and focusing only on a few application-oriented relevant status and behaviors.

The one-to-one uniformed mirroring of physical devices combined with the dynamic composability of multiple replicas, provided, respectively, by SDTs and CDTs, are a strategic and innovative application-driven feature to model complex IoT scenarios. It can be applied with multiple granularities both for devices belonging to the same equipment/plant and to aggregate twins among multiple objects and distributed locations to create new abstract digital representations. In this context, the IIoT is the perfect large scale ecosystem where the adoption of SDT and CDT can introduce effective benefits for services, data, and networking management.

In particular, CDT can change and adapt the configuration and the behavior of one or multiple SDTs according to a set of rules received by the CR. For example, by considering the smartphone assembly use case, a CDT can aggregate SDTs associated with the assembler (AS-SDT) and the security camera #1 (SC1-SDT) in the same plant area. When the AS-SDT detects an anomaly related to an increased vibration level, it immediately communicates the new context variation to the CDT. According to its application-driven rules, the CDT triggers the reconfiguration of both the AS-SDT and the SC1-SDT to increase camera image quality and to adapt the processed payload metadata requesting a personalized QoS prioritization to the underlying intelligent network layer.

In addition to SDTs, the edge layer is composed of:

1) *STD-Manager (SDT-M)*, with the responsibility of configuring, instantiating, and handling the life-cycle of SDTs. Each edge node hosts an independent SDT-M associated with the middleware to create and maintain an active virtual replica for physical devices the edge node is directly connected to. The SDT-M performs SDT advertising and receives SDT configuration commands by remotely interacting with the CDT Manager (see below);
2) *SDN Control Agent (S-CA)*, residing on host nodes together with the SDT-M. S-CA remotely interacts with the CR (and in particular the SDN Controller, see below) to send information about edge node computational, memory, networking characteristics, and its current state. Moreover, S-CA receives from the CR the traffic engineering rules to apply to the local edge node with the goal of optimizing packet management in an application-driven manner.

In addition to CDTs, the CR layer is composed of:

1) *CDT Manager (CDT-M)*, a software component orchestrating the creation and management of CDTs, according to application-driven rules and specifications. CDT-M actively communicates with deployed and active S-CAs to gather information about available physical devices and about supported features of related SDTs. The CDT-M is also responsible to push and dynamically adapt CDTs' configurations to shape their behavior and how they can react to context variations detected by active SDTs;
2) *SDN Controller (S-Ctrl)*, which i) receives per-edge node networking data from S-CAs, with the goal of generating the whole network topology, and ii) remotely distributes and activates traffic engineering rules, based on CDT QoS requirements. In particular, primary supported traffic management rules are:
   a) adoption of differentiated routing mechanisms, spanning from traditional solutions based on destination IP address to articulated ones considering the payload content;
   b) per-flow traffic priority packet scheduling, also by delaying/discarding packet dispatching to ensure delivery deadlines of higher priority traffic flows;
   c) per-packet payload management, e.g., by adding metadata to enhance payload expressiveness or by discarding part of the payload to reduce packet size.

   It is worth noting that S-Ctrl not only interacts with edge nodes hosting SDTs related to active CDTs, but also with other edge nodes acting as intermediary nodes along end-to-end inter-SDT paths.
3) *Application-driven Network Manager (ANM)*, the entry point of the whole solution allowing technicians to add/remove and de/activate CDTs (via CDT-M) and to manage the whole topology (via S-Ctrl). To this purpose, ANM actively interacts with the local CDT-M to get the set of running CDTs and related QoS requirements and with the local S-Ctrl to manage remote edge nodes and require them to properly tune network resources to activate traffic management rules, e.g., to achieve the desired delay, jitter, and/or throughput.

By exploiting the above components, the ADTN middleware can achieve the notable and original twofold objective of not only semantically enriching SDT packet content to provide more expressive information, but also exploiting the increased expressiveness to actually perform traffic flow management and packet dispatching optimization. This is done by also considering application-driven requirements, as better detailed in the following section.

IV. ADTN Design and Implementation Guidelines

The main effective and measurable advantages provided by ADTN are related to: 1) the creation of a uniformed data layer to support and handle the intrinsic IoT heterogeneity; 2) the reactive DT management and the consequent proactive adaption of QoS requirements for each relevant data stream; and 3) the dynamic and application-driven networking among devices, DTs, and services.

A. Uniformed Data Layer & Heterogeneity Management

As previously anticipated, we state that providing a uniform and standardized data format for information representation is strategic to allow the network to dynamically react to events, easily detect anomalies, and more generally support advanced management features without the need of customized and technician-driven operations. For instance, by adopting a well-known and easy-to-parse packet payload format, edge nodes can efficiently monitor the content of traversing packets and trigger a change of QoS requirements or even a network re-configuration, in case relevant content is identified. Also, note that the relevance of the content typically depends on the given application use case, and thus management rules on intermediary edge nodes not only should be able to easily monitor the content but also to effectively enforce new management rules provided by the centralized ANM in relation to activated SDTs and CDTs.

To this purpose, SDTs and CDTs use SenML [6] as a standard and uniform communication data format normalizing and neutralizing the fragmentation of data and information coming from physical devices. Each SDT is responsible to handle the bidirectional adaption with its counterpart for each involved packet and according to the supported protocols (e.g., MQTT and CoAP) and communication paradigms (e.g., Pub/Sub and RESTful). Furthermore, we enhance and extend the SenML format to allow SDTs to add contextual and operational information (denoted as "attribute" or "target") to each measurement, also allowing a simplified definition of network and QoS rules.

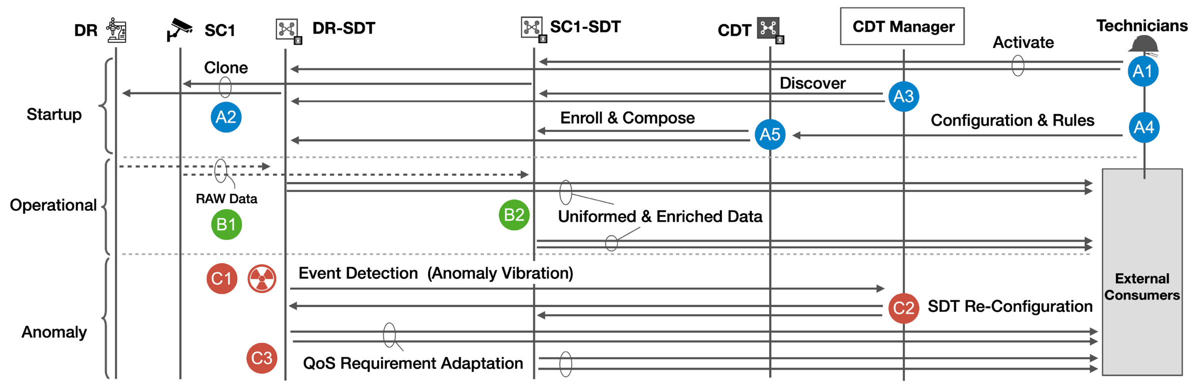
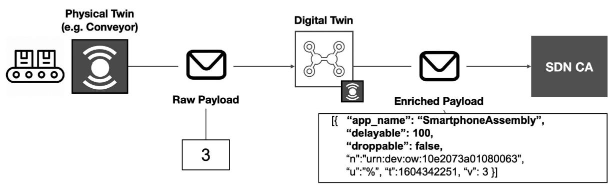


Fig. 3. Primary steps of Smartphone Assembly use case, with SDT enrollment and distributed coordination in case of anomaly.



```
[{  "app_name": "SmartphoneAssembly",
    "delayable": 100,
    "droppable": false,
    "n":"urn:dev:ow:10e2073a01080063",
    "u":"%", "t":1604342251, "v": 3 }]
```

Fig. 4. Example of SenML-based dynamic payload enrichment with delayable, droppable, and app name attributes.

As illustrated in Fig. 4, SDT is responsible to transform raw data incoming from the physical device into a SenML structured payload, comprising the timestamp (t), the current measurement value (v), the name associated with the device or data source (n) and the data unit (u), by following the IANA units specifications.[1] By following the specifications of the SenML standard, we have extended the data format to add three new additional metadata useful to support the proposed dynamic application-driven network management. The *droppable* attribute specifies if the current packet can be safely dropped whenever required (thus packet drop does not cause any major service misbehavior), the *delayable* field provides the maximum amount of time (in ms) that a packet can be delayed, and the *app_name* uniquely identifies the industrial application associated with the generated packet. These metadata can be used to dynamically model packet prioritization at the network level according to application requirements and current context. QoS requirements flags are then used by the SDN CA on intermediary edge nodes to autonomously determine if (according to the global network status and the specified packet constraints) it is required to increase or decrease packet priority, shape bandwidth, change packet delay, or even drop it (more details in Section IV-C).

B. Dynamic DT Management & QoS Requirements

As previously illustrated in Section III, SDTs, and CDTs actively collaborate to shape the target application behavior and proactively detect and react to events and context variations. This is meant to provide to the network management layer all the required information to improve the performance and properly prioritize the traffic. CDTs receive events and notifications from SDTs and, according to their application management rules, command the re-configuration of one or more SDTs. These dynamic changes can involve how SDTs process datastreams and how they adapt and enrich the exposed payload. For instance, in the case of anomalous vibration detection, CDT may require the drill-related SDT to tag packets with vibration data as not droppable or delayable. Listing 1 and Listing 2 report two examples of data management rules sent by CDT to the target DR-SDT (Drill SDT) to control how it sets data enrichment and QoS requirements metadata. The Standard rule specifies that packets can be safely dropped and delayed up to 200 ms; the Anomaly rule that packets cannot be dropped or delayed. Thanks to this distributed coordination, applications can modify their QoS requirements in an effective and fine-grained way, according to the target goal and the current deployment context.

Listing 1 Drill Standard data management rule.

```
name: urn:dev:ow:10e2073a01080063
unit: %
data_type: numeric
app_name: Smartphone Assembly
droppable: true
delayable: 200
```

Listing 2 Drill Anomaly rule update.

```
droppable: false
delayable: 0
```

For the sake of clarity, Fig. 3 illustrates the primary steps of this dynamic adaptation for a simplified version of the Smartphone Assembly industrial use case, by focusing only on the drill (DR-SDT) and the security camera (SC1-SDT) parts. The sequence diagram is broken down into three main phases: A) *Startup*, identifying the network location of SDTs related to

[1]IANA SenML Units – https://www.iana.org/assignments/senml/senml.xhtml

physical devices; B) *Operational*, when physical devices provide raw data and related SDTs generate SenML-compliant enriched packets sent back and forth to the CR; and C) *Alerting*, triggered by an anomaly detection and involving the dynamic adaptation of data and network management.

In details, the Startup phase involves: A1) technicians activate DR-SDT and SC1-SDT to mirror the IoT resources on their physical counterparts; A2) SDTs mirror the physical devices by discovering and cloning their resources into a digital replica; A3) the CDT Manager autonomously discovers the two newly activated SDTs and store their network location and capabilities; and A4) technicians interact with CDT Manager to provide the "Smartphone Assembly CDT" as the composition of the two SDTs together with the data management rules that specify how to enrich packets with QoS requirements. During the Operational phase: B1) technicians activate the Smartphone Assembly CDT, by enabling the associated SDTs and by triggering the activation of physical devices; and B2) SDTs process the received packets to make the data format compliant with SenML, by also enriching them with QoS requirements parameters. The Anomaly phase consists of: C1) the detection of an anomaly vibration by DR-SDT and the corresponding notification to CDT; C2) according to its internal rules, CDT reacts to the alert by pushing an updated configuration to both AS-SDT and SC1-SDT, e.g., to switch from Listing 1 to Listing 2 thus modifying the value of delayable/droppable QoS parameters of traffic flows coming from the drill; and C3) the involved SDTs adapt their behaviors by changing the generated and exchanged packets. Let us stress that CDTs (by reconfiguring SDTs) only modify the value of QoS parameters, while their proper management is delegated to S-CAs running on edge nodes. This highly decoupled approach allows, on the one hand, to create a dynamic, scalable, and uniformed environment able to detect and react to anomalies and events, and on the other hand, to efficiently distribute responsibilities among application-aware semantically informed CDTs/SDTs and network layer components that efficiently manage packets in a (mostly) application-agnostic manner.

C. Fine-Grained Application-Driven QoS Management

Each SDT hosted on an edge node is uniquely identifiable and reachable through a dedicated logical name or a distinctive identifier, e.g., logic-name:port such AS:1234), able to hide its actual IP address, thus allowing to route traffic flows and packets while reducing networking complexity. To this purpose, the developed ADTN middleware exploits the SDN-based multilayer routing (MLR) approach, specifically supporting network management in edge-based multihop deployment environments [7]. MLR allows to exploit, even at the same time, different routing strategies and mechanisms suitable for applications with heterogeneous features and requirements. Based on its centralized point of view, S-Ctrl dynamically determines and configures the proper MLR forwarding mechanism, ranging from traditional IP and sequence-based overlays to more articulated forwarding solutions based on the inspection of payload content types and values. In particular, the SDN-enabled MLR approach allows to exploit the same network topology even at the same time by different industrial applications (represented by different CDTs) to dispatch traffic flows based on native IP, overlay networking information, and payload content. Moreover, MLR allows the exploitation of the multihop and multipath network by supporting an overlay network that distinguishes edge nodes based on fixed unique identifiers rather than with time-varying private IP addresses.

As specified in Section IV-B, CDT-M takes as input a high-level representation of an industrial application and provides as output the dynamic reconfiguration of CDTs and SDTs. In addition, S-Ctrl interacts with S-CAs on edge nodes to tune how traffic flows are actually managed, based on per-packet metadata as well as the knowledge of the network current state.

Taking into consideration again the smartphone assembly use case, S-Ctrl adopts four different network management approaches, by also considering the payload type. To this purpose, S-Ctrl and S-CAs identify three packet types: *Video*, carrying frames provided by surveillance cameras, *Vibration*, representing the data provided by the drill, and *Info*, logs generated by the conveyor, the assembler, and the checker.

In case there is *no network congestion*, network management rules are not activated and thus packets are dispatched in a best effort manner.

In case S-Ctrl identifies *low network congestion*, S-Ctrl enables on S-CAs a network management rule dropping packets with probability $(\text{percentage}/100)^{\text{hopCount}}$ (with hopCount > 0) and percentage set to 25%, 33%, and 33% for Info, Video, and Vibration packets, respectively. Of course, this rule only applies to packets tagged as droppable. For instance, according to the Smartphone Assembly policies and Listings 1 and 2, Vibration packets can be dropped only in case no anomalies have been detected. Let us note that by dropping packets based on the hop count, the probability of dropping a packet along a multihop path is

$$\sum_{\text{hopCount}=1}^{\text{pathLength}} \left(\frac{\text{percentage}}{100}\right)^{\text{hopCount}} \tag{1}$$

and with pathlength $\to \infty$ the overall probability a packet is dropped is 33% and 50% if the *percentage* is set to 25% and 33%, respectively. In this manner, we achieve the notable effect of adopting a simple dropping mechanism that can be applied on each edge node in a stateless manner, while imposing a limit to the percentage of packets that can be dropped.

In addition, at each step the network management rule delays packets with delayable>0 by temporarily queuing them in case other not delayable packets are currently dispatched by the edge node, thus ensuring additional networking resources to packets tagged as not delayable. In particular, at each step Video packets are delayed by 10 ms, Vibration packets by 25 ms, and Info packets by 5 ms. In addition, packets are delayed only if *current time - timestamp* $<$ *delayable:* by setting an upper bound to delayable packets the network management rule allows to ensure that they are not indefinitely buffered on edge nodes. For instance, without vibration anomaly Info packets are not delayed since the data management rule sets delayable to 0, while Vibration/Video packets are delayed up to 150 ms.

In case S-Ctrl identifies *medium network congestion*, the same considerations apply but with higher values for dropping percentages and delays for droppable and delayable packets. In case S-Ctrl identifies *high network congestion*, it interacts with edge nodes generating traffic flows to switch not droppable and not delayable packets (e.g., Vibration packets in case of anomaly detection) to OS routing, thus minimizing network management overhead and achieving the best performance.

Finally, let us stress that edge nodes applying network management rules are aware neither of the current application state (either "regular" or "anomaly") nor of the semantic of the smartphone assembly use case. They only monitor traversing packets and, based on the current traffic situation and the packet type, they apply the associated delayable and droppable rules as much efficient as possible.

V. Experimental Evaluation

This section presents our experimental testbed environment and the performed evaluations of our ADTN middleware. The aim is to measure and analyze the ADTN performance in terms of communication delay and processing overhead, as well as to discuss its capability to efficiently and dynamically adapt to context and application variations.

A. Experiment Environment

Achieved performance results are based on our working Java prototype of the ADTN middleware that we have developed not only to demonstrate the feasibility and the efficiency of the presented model, but also to provide the community with a working solution to foster the research in this field. The source code of the adopted libraries and implementations has been released as Open Source projects.[2]

In particular, we analyzed two different aspects. The first one relates to the investigation of SDT performance in terms of 1) SenML data enrichment delay and 2) its impact on the twin forwarding delay between data producers and consumers. The second aspect is about how our ADTN middleware dynamically manages network configurations in relation to application requirements and network state.

In the former case, involved tests have been conducted on ten independent runs considering an uncongested network, a target set of 10 000 messages with IoT smart objects and DTs using MQTT as protocol, and an average payload size of 100 bytes. SDTs are implemented using Java and the WLDT library, a modular software stack based on a shared multithread engine able to effectively implement DT behavior and to define its mirroring procedures, data processing, and the interaction with external applications [8]. Implemented SDTs are executed as independent processes, but can also be easily packed as containers to run on virtualized environments and microservices. In order to evaluate the SDT's suitability to different hardware profiles, we tested the implementation both on a high specs Linux edge node (i7 Intel CPU and 32 GB of RAM) and on a Raspberry Pi (RPi) Model B board with 700 MHz CPU, 128 MB of RAM and a 10/100 Mbps Ethernet connectivity.

[2]1) [Online]. Available: https://github.com/DSG-UniFE/ramp and 2) [Online]. Available: https://github.com/wldt

In the latter case, we focused on the capacity of the ADTN middleware to dynamically introduce new networking policies depending on the congestion level of the network, by activating different traffic management rules on an intermediary edge node, as proposed in Section IV-C. To this purpose, we made measurements over a test-bed composed of N1, N2, and N3, three high specs RPis (Model 3B+ with 1 Gb RAM and 1.4 GHz 64-bit quad-core processor). N1 and N2 are connected via 100 Mbps Ethernet, N2 and N3 via 10 Mbps IEEE 802.11. N1 acts as SDT and sends data (with 100 bytes payload), while N3 is the receiver and N2 plays the role of the intermediary edge node. These three nodes interact via peer-to-peer communication channels, negotiated and instantiated during the discovery phase. When a new node joins the network, it looks for a node acting as S-Ctrl. In our test-bed, S-Ctrl is placed on N1 with the sender SDT. Once each node has notified its presence in the network, S-Ctrl performs parameter negotiation (e.g., IP addresses and ports) required for peer-to-peer socket instantiation between sender and receiver. The high-level API offered by our middleware allows S-Ctrl to activate networking rules on S-CAs not only by using the information about the congestion level, but also by taking into account the content of a specific packet. In this way packets with prioritized data can be delivered while respecting more demanding QoS requirements.

B. Achieved Results

Histograms (left y-axis) in Fig. 6 depict the processing delay required on the SDT to enrich the incoming payload (as illustrated in Section IV-A) into a standard and structured SenML packet and to serialize it using JSON data format. As expected, the i7 edge node is able to efficiently process packets with a negligible delay of about 0.56 ms, while the RPi delay is higher with an average of 4.35 ms. The second graph (right y-axis) in Fig. 6 reports the forwarding delay (with respect to the message rate) due to SDT forwarding. Illustrated results clearly show how the high specs edge node is able to handle an increasing message rate ranging from 1 $\mathrm{msg/sec}$ to 1000 msg/sec with an average overall delay (considering also the SenML processing) of 6 ms. In contrast, the very limited capabilities of the RPi allows to keep up good performance until 50 $\mathrm{msg/sec}$ before saturating its resources.

At the beginning, N2 is configured with *no network congestion* rule and N1 sends data at 150 $\mathrm{msg/sec}$. Then, N1 increases the message rate by 500 $\mathrm{msg/sec}$ every 12 s. As Fig. 5 shows, at about 12 s, the receiver notices a packet latency greater than the 150 {\rm ms} threshold {\rm ms} and infor{\rm ms} S-Ctrl about it. Then, S-Ctrl infor{\rm ms} S-CA on N2 to activate the *low network congestion*, i.e., the intermediary edge node has to discard or delay part of video and vibration messages. Network congestion lowers for a while and N3 receives packets at reduced latency. However, the message rate keeps increasing, and at about 25 s N3 notices a latency greater than 300 {\rm ms} and again it interacts with S-Ctrl to trigger the activation of the *medium network congestion*. This rule further increases

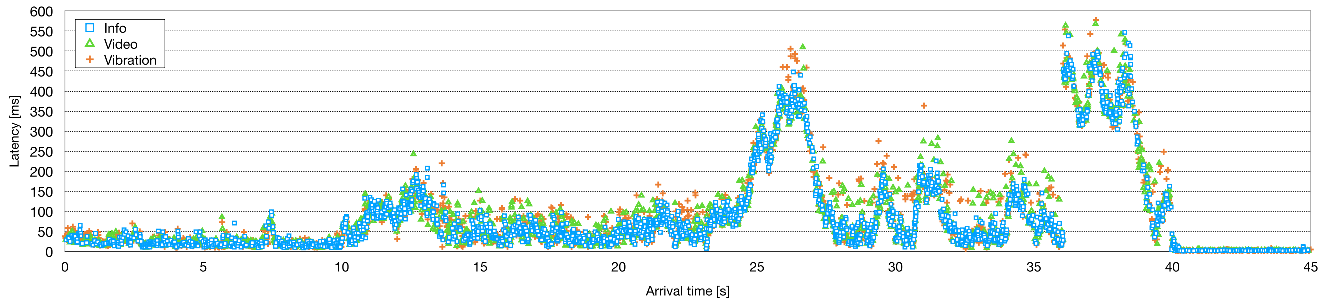


Fig. 5. Message delay at increasing message rate while applying the proposed QoS management solution.

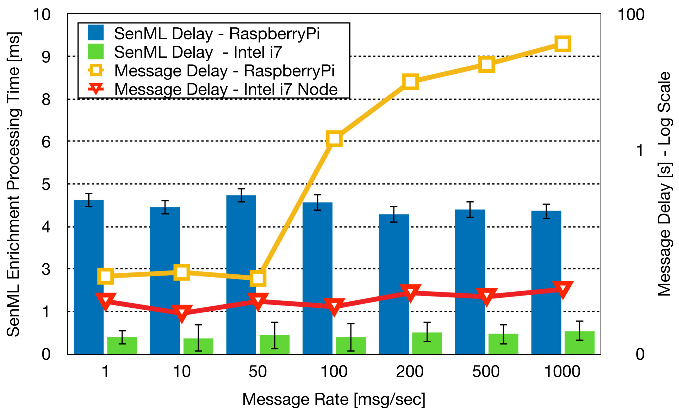


Fig. 6. SDT SenML data enrichment delay and message forwarding with respect to the variation of the message rate.

the amount of dropped and delayed packets, limiting the overall packet latency. Finally, at about 36 s the latency becomes greater than 450 {\rm ms}. The receiver infor{\rm ms} S-Ctrl about the current situation and the latter activates the *high network congestion* rule, thus imposing sender and receiver to communicate via OS routing instead of the overlay network. In this case, the network latency considerably lowers, but with the drawback of reduced traffic management at the application level, since N2 is not able to access the payload of traversing packets anymore.

C. Discussion

The results presented above demonstrate that the ADTN middleware is able, on the one hand, to enrich packets in a very efficient manner also providing a uniform standard data layer and, on the other hand, to dynamically manage traffic flows in an application-driven manner also considering the current state of the network. In particular, we found that the additional overhead imposed by SDTs to format and enrich the packet payload in the SenML syntax is very limited, largely justified by the advantages of greatly improving the QoS management features of traffic flows via articulated and fine-grained network rules on intermediary edge nodes.

However, it is worth noting that while experimental results prove that the ADTN middleware provides an efficient high level interface for DT management, they do not show how our solution behaves in ter{\rm ms} of other highly demanding QoS requirements, e.g., scalability, privacy awareness, and reliability. Therefore, based on the encouraging results already obtained, we are now working on the development of a full-fledged industrial pilot.

VI. Related Work

Traditionally, SDN emerged primarily to manage switches of closed and geographically centralized datacenters. However, its adoption has quickly proven its benefits also in scenarios characterized by more limited computational and networking capabilities. Considering fog and edge computing, [9] proposes to exploit SDN to deliver and deploy new services in IoT environments in a faster and more cost-effective manner. The SDN approach can be also adopted together with Blockchain, e.g., to deliver a fully distributed Cloud architecture based on Fog nodes [10] or to improve the credibility and authenticity of nodes while addressing the issues associated with the fact that the SDN controller represents a single point of attack [11]. Finally, SDN is fruitfully adopted to support load balancing in Fog environments [12], [13]. Interested readers can refer to [14] and [15] for comprehensive studies on the adoption of SDN in Edge and Fog environments.

By focusing on industrial environments, SDN has emerged in the communication research and industrial fields of IIoT [16] primarily to manage switches of closed environments such as datacenters and department networks via the OpenFlow protocol. More recently, [17] focused on the adoption of the SDN paradigm in the context of IIoT to dispatch packets with different delay constraints in a per-flow tailored manner, by considering time deadlines, traffic load balances, and energy consumption. Similarly, [18] adopted SDN to efficiently manage the interplay between edge and cloud environments by considering energy efficiency, bandwidth, and latency. Finally, some efforts have focused on edge IoT scenarios with industrial wireless sensor networks. For instance, [19] manages both transmission scheduling and node mobility allowing to ensure bounded end-to-end delays. [20] ai{\rm ms} at improving industrial performance by adopting end-to-end QoS control. To this purpose, it adopts a unique SDN instance for IoT environments consisting of wireless and wired segments, by exploiting 6TiSCH as industrial

IoT and open network platform allowing to orchestrate every network segment. Compared with this previous work, our solution adopts SDN to dynamically exploit the communication mechanis{\rm ms} most suitable to current application requirements, ranging from native IP to more articulated ones based on packet content [7]. In addition, it takes advantage of SenML enriched packet payload to efficiently enforce fine-grained content-based traffic flow rules, allowing to better satisfy per-application QoS requirements.

The great interest in DT-based solutions from both academic and industrial points of view is demonstrated by the vast recent literature. In fact, during the last years, several researches, organizations, and consortiu{\rm ms} worked to define and shape the role of DTs and their responsibilities among different architectural and technological infrastructures and layers. In particular, DTs show a strong and renewed relationship with IoT and IIoT [21] since the peculiar properties of DTs can bring actual benefits in the context of large distributed deployments, factories, and smart cities. An extensive, clear, and detailed survey of DTs is reported in [22].

The DT research area is attracting a wide interest involving a multitude of diverse and approaches related, e.g., to big data analytics [23], behavioral modeling [24], ontology definition [25], specific device mirroring [26]. In this fragmented context, the industrial world and in particular the Industrial Internet of Things Consortium is proposing a shared reference architecture [27], [28] taking into account DT relationships, composition, and main services (e.g., prediction, maintenance, safety). It also covers different production stages and use cases, in particular related to manufacturing [29] and product design [30]. However, the limitations of proposed solutions are mainly related to the adoption of a centralized DT management (often deployed only on Cloud infrastructure) where a unique entry point is responsible to maintain twin instances and by moving the integration responsibilities to external modules or connectors. Furthermore, DTs are not in charge of supporting or handling the heterogeneity associated with the connected devices. This missing role feeds the creation of unnecessary substrates of domain specific technologies and legacy interaction for{\rm ms}. In this context, edge processing has already shown its fundamental role to effectively and efficiently handle IoT heterogeneity through the introduction of intermediate proxies and hubs, responsible to manage objects and data strea{\rm ms} through centralized approaches [31], [32]. Despite these advancements, the adoption of DTs on the edge is still under-explored and represents a novel research area, including the seamless integration of data and services in heterogeneous syste{\rm ms}. Authors in [33] describe the importance of bringing DTs on the edge by highlighting the relevant interest in this new research field and its experimentation. In [34] and [35] the authors present two interesting initial evaluations of edge DTs in the specific contexts of blockchain technologies and the social internet of things (SIoT). In [36], instead, the authors propose a different and significant point of view by incorporating DTs into edge networks to support real-time federated learning with reduced communication costs.

The above-reviewed projects and research activities provide excellent solutions in their operational fields and show how DTs can be flexible and fundamental for next-generation applications in the area. The primary differences with our approach are mainly related to the fact that they keep DTs as separated and supplementary architectural entities, by addressing domain-specific contexts without generalizing and investigating the real role of DTs on the edge. On the contrary, we deeply integrated DTs in the edge architecture and measured how they can be active and fundamental components to enable interoperability and intelligent networking. To the best of our knowledge, our work is the first one adopting the DT approach to support application-aware dynamic management of edge network resources in industrial environments. In fact, it originally adopts awareness of the industrial environment to perform application-driven network management and software component coordination. To this purpose, it exploits a DT abstraction layer to make easier the interaction with actual devices and to enable the adoption of semantically rich standards, while allowing the dynamic reconfiguration of network devices to maximize QoS in the multihop multipath industrial edge topology. Our middleware adopts DTs not only to more easily interact with actual devices, e.g., by exploiting traditional IP-based protocols instead of specialized industrial ones, but also to enhance packet content expressiveness, e.g., by reformatting monitored information by means of well-defined data syntax standards (in particular the SenML format).

VII. Conclusion

This article presented the novel ADTN middleware composed of semantically enriching SDTs distributed on edge nodes and centralized CDTs performing their flexible orchestration, to significantly lower the management complexity deriving from heterogeneous industrial environments. In addition, the adoption of an application-aware SDN solution based on multiple traffic forwarding techniques allowed to maximize the tradeoff among packet expressiveness and their efficient dispatching.

While the obtained encouraging results demonstrated both the feasibility and the efficiency of the proposed solution, there are still some aspects that could be considered and investigated to further foster its adoption in real-world industrial environments. First of all, it could be useful to offer a wide set of preconfigured SDTs able to manage a plethora of industrial protocols, thus allowing our middleware to easily interact with several industrial devices with no additional effort by application domain developers. In addition, developing CDTs for complex and articulated industrial applications can be very challenging for industrial technicians. Thus, we plan to investigate how to develop CDTs based on high-level representation of applications and required QoS, e.g., by adopting semantically enriched solutions to make their development (semi) automatic. Moreover, network rules can be hard to enforce in a safe manner when several competing industrial applications may coexist (emergence of conflicting requirements). In conclusion, we believe that one of the paramount challenges that should be addressed as future research is to find a proper tradeoff among traditional single-application static solutions (under-exploiting available industrial resources) and novel highly dynamic solutions allowing to fully exploit every resource by running multiple competing applications based on automatic mechanis{\rm ms} (with the risk of compromising the full-functionality and safety requirements of the plant).

References

[1] S. Haag and R. Anderl, "Digital twin - Proof of concept," *Manuf. Lett.*, vol. 15, pp. 64–66, 2018.

[2] 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, Jan. 2018.

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

[4] "MQTT Version 3.1.1," Sep. 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[5] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," RFC 7252, Jun. 2014.

[6] C. Jennings, Z. Shelby, J. Arkko, A. Keranen, and C. Bormann, "Sensor measurement lists (senml)," Internet Requests for Comments, RFC Editor, RFC 8428, Aug. 2018.

[7] P. Bellavista, C. Giannelli, and D. D. P. Montenero, "A reference model and prototype implementation for sdn-based multi layer routing in fog environments," *IEEE Trans. Netw. Serv. Manage.*, vol. 17, no. 3, pp. 1460–1473, Sep. 2020.

[8] M. Picone, M. Mamei, and F. Zambonelli, "Wldt: A general purpose library to build IoT digital twins," *SoftwareX*, vol. 13, Art. no. 100661, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711021000066

[9] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[10] P. K. Sharma, M. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, Sep. 2017.

[11] Y. Gao, Y. Chen, X. Hu, H. Lin, Y. Liu, and L. Nie, "Blockchain based IIoT data sharing framework for SDN-enabled pervasive edge computing," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2020.3012508.

[12] X. He, Z. Ren, C. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles," *China Commun.*, vol. 13, no. Supplement2, pp. 140–149, 2016.

[13] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for iot applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1159–1166, May 2019.

[14] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2359–2391, Oct.–Dec. 2017.

[15] F. Y. Okay and S. Ozdemir, "Routing in fog-enabled iot platfor{\rm ms}: A survey and an SDN-based solution," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4871–4889, Dec. 2018.

[16] J. Wan *et al.*, "Software-defined industrial Internet of Things in the context of industry 4.0," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7373–7380, May 2016.

[17] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in SDN-based industrial internet of things with edge computing," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1351–1360, Jun. 2018.

[18] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge computing in the industrial Internet of Things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 44–51, Feb. 2018.

[19] L. L. Bello, A. Lombardo, S. Milardo, G. Patti, and M. Reno, "Experimental assessments and analysis of an SDN framework to integrate mobility management in industrial wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 16, no. 8, pp. 5586–5595, Aug. 2020.

[20] N. B V and R. M. R. Guddeti, "Fog-based intelligent machine malfunction monitoring system for industry 4.0," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2021.3056076.

[21] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Inform.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.

[22] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167 653–167671, May 2019.

[23] D. Riemer, "Feeding the digital twin: Basics, models and lessons learned from building an IoT analytics toolbox (invited talk)," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 4212–4212.

[24] J. Sleuters, Y. Li, J. Verriet, M. Velikova, and R. Doornbos, "A digital twin method for automated behavior analysis of large-scale distributed IoT syste{\rm ms}," in *Proc. 14th Annu. Conf. Syst. Syst. Eng.*, 2019, pp. 7–12.

[25] C. Steinmetz, A. Rettberg, F. G. C. Ribeiro, G. Schroeder, and C. E. Pereira, "Internet of Things ontology for digital twin in cyber physical syste{\rm ms}," in *Proc. 8th Braz. Symp. Comput. Syst. Eng.*, 2018, pp. 154–159.

[26] E. Y. Song, M. Burns, A. Pandey, and T. Roth, "IEEE 1451 smart sensor digital twin federation for IOT/CPS research," in *Proc. IEEE Sensors Appl. Symp.*, 2019, pp. 1–6.

[27] S. Malakuti and S. Grüner, "Architectural aspects of digital twins in IIoT systems," in *Proc. 12th Eur. Conf. Softw. Architecture: Companion*, New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–2. [Online]. Available: https://doi.org/10.1145/3241403.3241417

[28] V. Souza, R. Cruz, W. Silva, S. Lins, and V. Lucena, "A digital twin architecture based on the industrial internet of things technologies," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2019, pp. 1–2.

[29] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," in *Proc. 16th IFAC Symp. Inf. Control Proble{\rm ms} Manuf.*, 2018, pp. 1016–1022.

[30] R. Wagner, B. Schleich, B. Haefner, A. Kuhnle, S. Wartzack, and G. Lanza, "Challenges and potentials of digital twins and industry 4.0 in product design and production for high performance products," *Procedia CIRP*, vol. 84, pp. 88–93, 2019.

[31] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[32] S. Cirani, G. Ferrari, N. Iotti, and M. Picone, "The iot hub: A fog node for seamless management of heterogeneous connected smart objects," in *Proc. 12th Annu. IEEE Int. Conf. Sens., Commun., Netw. Workshops*, 2015, pp. 1–6.

[33] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, May 2020.

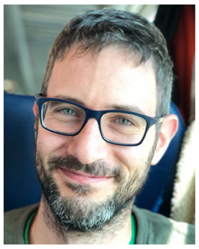
[34] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning and permissioned blockchain for digital twin edge networks," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021.

[35] O. Chukhno, N. Chukhno, G. Araniti, C. Campolo, A. Iera, and A. Molinaro, "Optimal placement of social digital twins in edge IoT networks," *Sensors*, vol. 20, no. 21, pp. 1–7, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6181

[36] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial IoT," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2020.3010798.

Paolo Bellavista (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science engineering from the University of Bologna, Bologna, Italy.

He is now a Full Professor of distributed and mobile syste{\rm ms} with the University of Bologna. His research activities span from pervasive wireless computing to online big data processing under quality constraints, from edge cloud computing to middleware for Industry 4.0 applications. He has served on several Editorial Boards, including IEEE CO{\RM MS}T (Associate EiC), ACM CSUR, and Elsevier JNCA and PMC. He is the scientific coordinator of the H2020 BigData project IoTwins.



Carlo Giannelli (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Bologna, Italy, in 2008.

He is currently an Associate Professor in computer science with the University of Ferrara, Ferrara, Italy. His primary research interests include Industrial Internet of Things, software defined networking, Blockchain technologies, location/based services, heterogeneous wireless interface integration, and hybrid infrastructure/ad hoc and spontaneous multi-hop networking environments.

Marco Mamei received the Ph.D. degree in computer science from the University of Modena and Reggio Emilia, Modena, Italy, in 2004.

He is Full Professor in computer science with the University of Modena and Reggio Emilia, since 2019. His work focuses on data mining applied to mobile phone data and Internet of Things applications. In these areas we published more than 100 papers, eight patents, and won several best paper awards.

Matteo Mendula received the M.Sc. degree in computer engineering with the University of Bologna, Bologna, Italy, in 2020. He is currently working toward the Ph.D. degree in distributed learning and local data processing in edge environments with the Department of Computer Science and Engineering, University of Bologna.

His main research interests include are big data processing and distributed learning on the edges of the network. In particular, his research relates to architectural aspects and machine Learning enhanced techniques in fog computing scenarios.

Marco Picone received the M.Sc. (cum Laude) degree in computer engineering and the Ph.D. degree in information technology from the University of Parma, Parma, Italy, in 2008 and 2012, respectively.

He is currently a Senior Postdoctoral Researcher with the University of Modena and Reggio Emilia, Modena, Italy. He published several research papers in international journals/conferences and the main research interests include Internet of Things, edge/fog computing, and digital twins.