Electric Drives as Fog Nodes in a Fog Computing-based Industrial Use Case

Mohammadreza Barzegaran[1], Nitin Desai[2], Jia Qian[1], and Paul Pop[1]

[1]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
[2]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

***Abstract*—Electric drives, which are a main component in industrial applications, control electric motors and record vital information about their respective industrial processes. The development of electric drives as Fog nodes within a fog computing platform (FCP) leads to new abilities such as programmability, analytics, and connectivity, increasing their value. In this study, we use the FORA FCP reference architecture to implement electric drives as Fog nodes, which we call "fogification". We designed our fogified drive architecture and its components using Architecture Analysis and Design Language (AADL). The design process was driven by the high-level requirements that we elicited. We used both the fogified drive architecture and the current drive architecture to implement a self baggage drop system in which electric drives are the key components. We then evaluated the fog-based design using several key performance indicators (KPIs), which reveal its advantages over the current drive architecture. The evaluation results show that safety-related isolation is enabled with only 9% overhead on the total Fog node utilization, control applications are virtualized with zero input-output jitter, the hardware cost is reduced by 44%, and machine learning at the edge is performed without interrupting the main drive functionalities and with an average 85% accuracy. The conclusion is that the fog-based design can successfully implement the required electric drive functionalities and can also enable innovative uses needed for realizing the vision of Industry 4.0.**

I. Introduction

Industry 4.0 is an industrial revolution via digitalization that affects all industries and sectors. Digitization increases productivity, flexibility, and product quality. Moreover, it reduces time-to-market and supports mass-customization [1]. When machines become connected with each other, sensors, and actuators, they form the Industrial IoT (IIoT), which is expected to increase the global gross domestic product (GDP) value to USD 15 trillion by 2030 [2].

The convergence of Operational Technology (OT) and Information Technology (IT) drives this digital transformation [3]. However, OT and IT use different computation and communication technologies [3]. OT employs dedicated hardware and software to implement real-time and safety-critical applications that have stringent timing and dependability requirements. OT uses proprietary technologies, imposes information flow restrictions, and hence does not support the vision of Industry 4.0 [3]. IT uses cloud computing, artificial intelligence (AI), and big data to bring flexibility, scalability, reduced costs and faster development. However, IT is not suitable for industrial applications where non-functional properties related to timeliness and dependability must be guaranteed [4].

Because Industry 4.0 will only become a reality through the convergence of OT and IT [3], a new paradigm called fog computing has been envisioned as an architectural means to realizing the OT/IT convergence [5]. Fog computing is a "system-level architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things" [6]. A fog computing platform (FCP) brings computation and storage resources closer to the edge of the network. An FCP is composed of several interconnected fog nodes (FNs), as shown in Fig. 1. Several types of FNs from powerful high-end FNs to low-end FNs with limited resources have been proposed by researchers [7], [8] and have been developed by companies [9], [10].

This study addresses industrial application areas in which *electric drives* [11] are present. Electric drives control electric motors and are ubiquitous in industrial installations in many domains such as the automotive, food and beverage, marine and offshore, hydraulics, refrigeration, and air conditioning domains. In this paper, we propose a method for converting existing IIoT architectures to fog computing-based implementations, aiming at realizing the vision of Industry 4.0. We show how electric drives can be thus turned into FNs within a fog computing architecture (we call this process *fogification*). Extending electric drives (which are naturally at the edge of the network, near the machines, sensors, actuators, and industrial data sources) with fog computing capabilities will guarantee effective collaboration among the devices, nodes, and Cloud [12].

By developing the electric drives as FNs, new features, such as programmability, analytics, and connectivity with customer clouds, are expected to increase their value [13]. Their increased functionality allows drives to assume a more significant role in industrial and domestic control systems by leveraging their ability to instrument as the data source, which can help bootstrap the data economy. The main direct business benefit comes from the ability to also instrument legacy systems using drives as the data source. Because electric drives run real-time software to control the speed, torque, and position of electrical motors that operate cooperatively with other devices to automate machinery, they produce data that carries vital information about the machinery they control. These data compromise a critical asset that is massive, often repetitive, and often must remain on-premises for privacy

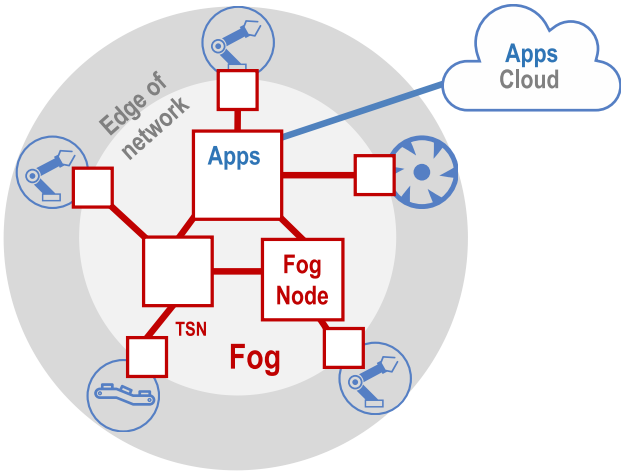


Fig. 1. An FCP consists of FNs (boxes) from high-end to low-end that are placed at the edge of the network and connected via a Deterministic Ethernet (thick lines) with each other, equipment, and the Cloud. Mixed-criticality applications are running on FNs and in the Cloud.

reasons. Hence, there is a need for drives to be capable of data analytics locally, at the edge, which cannot be realized with the current drive architecture.

Edge analytics will facilitate network off-loading and extend the Internet of Things (IoT) solution market. Digital services allow efficient service provisioning, improved uptime, and decreased overall costs. Correctly configured products and processes decrease energy consumption and improve quality. Open data ecosystems support innovations and new value-added services and will create long-term benefits for all ecosystem participants.

A. Related work

Several research projects have addressed mixed-criticality applications that share multicore-based distributed architectures. The aims of the EMC2 European Project[1] are to provide efficient handling of mixed-criticality applications under real-time conditions, scalability and maximum flexibility, and full-scale deployment and management of integrated tool chains, throughout the entire life cycle. Research on FCP architectures has made progress in recent years [8], [14]. For example, the European projects FORA [15] and mF2C[2] focus on creating open-source, standards-compliant fog platforms using COTS hardware to execute hard real-time industrial control applications such as the electric drives discussed in this paper. Companies such as TTTech Computertechnik AG [9] and Nebbiolo Technologies Inc. [10] are pioneers in the field of commercializing the fog computing paradigm with market-ready products for industrial automation. Although design paradigms for the fog are still in their early stages, there are certain generic guidelines that are followed to ensure the isolation of tasks of varying criticality. In [16], the authors describe an execution framework in which applications are isolated temporally on many-core processors.

Safety certification as proof of guarantees for the proper execution of safety functions is needed for the FCP. Classical safety controller designs such as the simplex architecture [17], [18] provide a switching mechanism between a high-performance but non-safety certified controller and a simple certified controller for safety functions. However, for complex systems such as the FCP, the simplex design is nonoptimal because of its switching latencies. Selicean et al. [19] proposed a method in which different safety-integrity levels (SILs) are assigned to the applications. In this method, applications with the same SIL are mapped to a single partition. Virtualization of control applications can then be realized by separating and scheduling the control tasks inside the partitions, similar to [20]. The modification of hypervisors provides different degrees of separation. Modification of the Xen hypervisor [21] to guarantee timing constraints was proposed by Masrur et al. [22]. The authors modified the hypervisor with a new scheduler based on a fixed-priority policy and a control loop to control the timing constraints of virtual machines. [23] addressed safety critical applications running in the fog and how the FCP must cater to these specific requirements.

One major research theme is resource management in the fog. In [24], the authors identified and classified the architectures, infrastructure, and underlying algorithms for managing resources in fog/edge computing. The authors of [25] proposed a list scheduling-based heuristics to solve this problem. The authors demonstrated the feasibility of reconfiguring the scheduled network at runtime for industrial applications within the fog. Reference [26] introduced a vulnerability-based method to quantify the security performance of communications on distributed systems. Fault-tolerant aspects were discussed in [27], where the design problem is to minimize the schedule length and security vulnerability of the application, subject to given fault-tolerant constraints. A multi-objective optimization method was then proposed to find the best solutions. Reference [28] discussed potentially contradicting design constraints: real-time capability versus scalability. This paper suggested a design methodology and architecture as a step toward perfectly scalable real-time systems, i.e., systems with deterministic timing behavior and run-time reconfiguration.

Researchers have also addressed issues with connectivity in the fog. Reference [29] addresses the challenges and issues with IEEE 802.11 MAC protocols that exchange critical messages used in industry. Barzegaran et al. in [30] proposed a constraint programming-based solution to configure communication for critical control applications running in the fog nodes. In [31] researchers proposed a solution that can be used to reduce communication costs. The solution splits the communication-related computation between fog nodes and the Cloud to avoid sending all raw data to the Cloud. Such solutions show their advantages in applications where fog nodes collect data and the FCP runs machine learning (ML).

Several approaches have been proposed for security threats

[1] www.artemis-emc2.eu
[2] https://www.mf2c-project.eu

in an FCP. Chaudhary et al. in [32] proposed an approach for finding XSS attacks in an IoT network. The approach uses Convolution Neural Network (CNN) that perform data preparation methods on the attack payload to detect the security threat. In [33] the authors proposed an authentication method for Time-Sensitive Networks (TSN) that exchanges key for secure data exchange. In [34] researchers proposed an authentication method that uses signature from sender and public/private keys for decryption. The method also considers generating optimal routing for secure transmission of messages.

Cyber-Physical Systems (CPS) in industrial infrastructures also deal with the combination of mechatronics, communication, and information technologies to control distributed physical processes and systems. They are designed as a network of interacting software and hardware devices and systems, many of them with a higher level of decision-making capability in two respects: autonomic with self-decision processes and collaborative with negotiation-based decision processes. Using CPS as emerging components in Industry 4.0 has been addressed by researchers such as [35]. Recently, several projects have focused on flexible architectures for Industry 4.0-driven CPSs as well as distributed control systems (DCS). A common design goal for various reference architectures for Industry 4.0 is to introduce dynamic and flexible interaction among components [36]. One of these initiatives is the German Industry 4.0 initiative, which specifies the Reference Architecture Model Industry 4.0 (RAMI 4.0) [DIN SPEC 91345].

Three such architectural approaches are proposed by the PERFoRM, IMPROVE, and BaSys 4.0 projects. Production harmonizEd Reconfguration of Flexible Robots and Machinery (PERFoRM) focuses on increasing flexibility and configurability in manufacturing. The primary goal is to transform existing systems into flexible and reconfigurable systems by providing an architecture with a common infrastructure for different industries.[3] The aim of the Innovative Modeling approaches for Production systems tO Raise Validatable Efficiency (IMPROVE) project is to develop a decision support system for tasks such as diagnosis and optimization. This is realized by the creation of a virtual factory that serves as a basis for model development and validation. Therefore, data from several systems in the plant need to be aggregated and integrated.[4] BaSys 4.0 stands for Basic System Industry 4.0, and it abstracts the overall production process and allows optimization prior to making the actual changes in the system. In addition, this system architecture provides real-time capabilities for critical process control functions [37].

As mentioned, fog computing is envisioned as an architectural means for realizing the vision of Industry 4.0 [6], [7], [38]. However, very few works investigate the application of fog computing to the industrial systems. The OpenFog reference architecture from [6] has been used to design an Airport Visual Security use case. Although the authors show how a fog reference architecture can be used for the use case, they do not evaluate their design, which is only presented conceptually. A fog computing architecture has also been investigated in an industrial robotics use case [39], which has identified the main challenges and potential solutions when moving to a fog-based architecture. However, the authors do not use a systematic architectural approach, nor evaluate the proposed solutions. An initial limited investigation on extending electric drives with fog computing capabilities was carried out in [40]. However, the use case was relatively small, and the evaluation limited. The main novelty of the work in [40] is the use a semi-formal notation for the proposed design, i.e., the architecture analysis & design language (AADL), which is an architecture description language from the domain of real-time embedded systems [41]. AADL has been standardized by the Society of Automotive Engineers (SAE) and employs a component-oriented approach for modeling systems using both textual syntax and graphical notation with precise semantics [41]. AADL is supported by several tools for graphical modeling and analysis of embedded systems such as OSATE [42], which is an open-source Eclipse-based framework consisting of a modeling environment and a set of plug-ins for validating and analyzing models.

B. Contributions

This paper proposes a method for updating existing Industrial IoT architectures to fog computing-based implementations, aiming at realizing the vision of Industry 4.0. The method applies the recently proposed FORA FCP reference architecture [15] and uses AADL to formally capture the design of the architecture. The method is evaluated against several key performance indicators (KPIs) on a realistic self baggage drop system that is using electric drives to control the conveyor belts. The contributions of this paper are as follows:

- We propose a new design for electric drives as FNs using the FORA FCP reference architecture from [15]. In this design, electric drives are developed as FNs that deliver the main drive functionality and provide the benefits envisioned in Industry 4.0.
- We model the fogified drive architecture using AADL, capturing the design and the interactions between the architecture's components. We extend the AADL language annexes to realize a fog-based design, e.g., the ARINC653 AADL annex [43], which defines virtual processors and virtual buses.
- We identify the fog-based drive requirements for driving the design process, to ensure the implementation of drive functionality and to realize the vision of virtualizing the control functions on a fog-based industrial architecture.
- We apply our method to implement a fog-based architecture for a realistic use case consisting of a self baggage drop system that uses electric motors to drive its conveyor belts. We propose several KPIs, which are used to evaluate the fog-based solution.
- Using the proposed KPIs, we compare two implementations of the use case, one using the traditional design

[3]http://www.horizon2020-perform.eu
[4]http://www.improve-vfof.eu

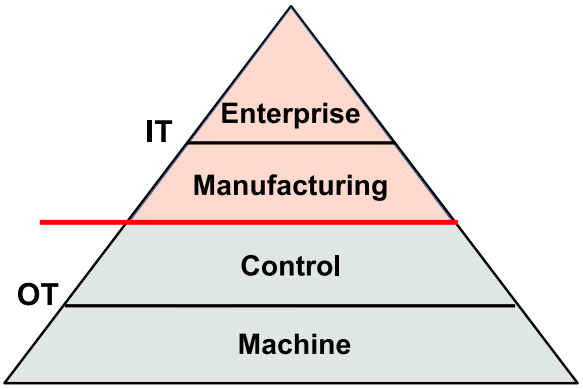


Fig. 2. Automation pyramid: the multiple levels of an industrial automation enterprise have hierarchies. The two bottom levels use OT, and the two top levels use IT.

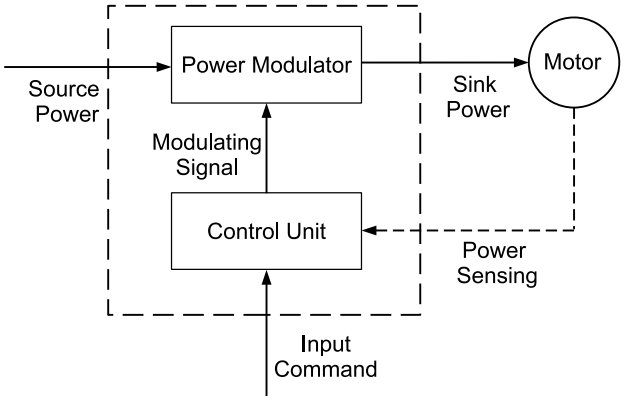


Fig. 3. Basic block diagram of an electric drive: The source power and the input command are the inputs, and the sink power is the output.

versus the proposed fog-based design. The evaluation shows that our method is capable of deriving a fog-based architecture for the use case, which improves several aspects over the traditional architecture, e.g., related to safety, control performance, data analytics, security and hardware cost.

The remainder of this paper is structured as follows. We introduce electric motors, drives, the current architecture of drives, and the drive requirements in Section II. We proposed our fogified drive architecture and its AADL model in Section III. Section IV evaluates the fog-based solution for our use case. We highlight the related work in Section I-A and conclude the paper in Section V.

II. Electric Motors and Drives

We introduce electric motors and electric drives in Section II-A and Section II-B, respectively, and describe an industrial setting in which motors and drives are used in Section II-C. We present the current drive architecture and its AADL model in Section II-D. Section II-E presents the drive requirements and KPIs.

A. Electric Motors

Electric motors are one of the main components in industrial settings in which machines are used for automation. As mentioned, they are used in many application areas, from building automation, energy systems, and industrial automation, mobile hydraulics.

An electric motor is an electromechanical machine that converts electricity into mechanical energy. The electric current is fed to the motor using a wire winding. This winding interacts with the motor's magnetic field, which applies torque to the motor shaft. Several classifications of electric motors have been introduced. For example, a well-known classification is based on the type of power source, direct current (DC) or alternating current (AC). Other classifications consider the internals of the electric motors or output motion [44].

Electric motors can generate continuous rotation and are widely used in various areas, ranging from electric watches to ship propulsion. In an industrial setting in which electric motors are operating, a specific rotation scenario is assumed. For example, an electric motor is needed to generate a fixed speed rotation, e.g., 50 revolutions per minute (rpm) for 10 s, starting at a given time. The rotation may also have a specific speed, torque, and position, which can be controlled by altering the electric current, e.g., the voltage, amperage, and frequency (in AC).

B. Electric Drives

Electric drives, alternatively called *drives*, are used to alter the electric current's characteristics such as the frequency and voltage to control the motor's rotation for a required outcome, i.e., the desired speed, torque, and position [11]. Drives are designed to be general purposes, e.g., to control motors within a certain power range, or for specific purposes, e.g., to control an electric motor with specific requirements. An example of a specific purpose is the control of a centrifuge machine's motor for spinning very dense fluid, which needs very precise control. Because using drives along with electric motors is necessary in industrial settings and products, the market for drives is huge, and it is predicted that it will grow by USD 5.11 billion between 2019 and 2023 [45]. Electric drives are a natural entry point for novel technologies that will bring significant business benefits, considering their role in the industry and their market value.

Figure 2 shows the so-called "automation pyramid" [46], which captures the multiple levels of an industrial automation enterprise. The "machine level" consists of sensors and actuators, including electric motors, which are placed in the field or on the production floor. In contrast, the "control level" consists of industrial controllers such as industrial personal computer (IPC) and programmable logic controllers (PLCs), which control and manipulate the devices in the field. The controllers obtain their inputs from the machine level, e.g., sensors, switches, and human–machine interfaces (HMIs), run a control algorithm to determine the desired outputs, and return the outputs to the actuator devices in the field. Although drives implement controllers to control the rotation of electric motors,

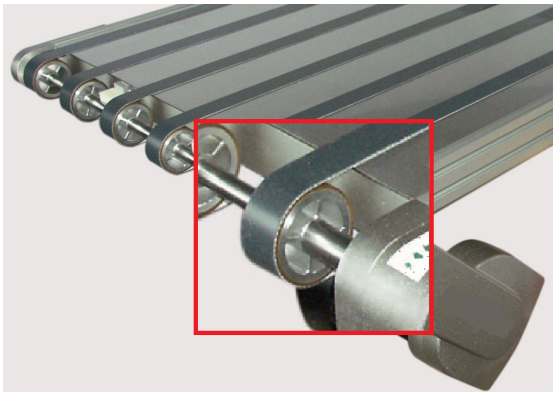


Fig. 4. A typical conveyor belt: belts are rotating around pulleys (the red box) using an electric motor that is controlled via an electric drive.

they physically reside on the machine level and are classified as secondary control devices.

Drives are embedded cyber-physical systems that must meet real-time responses and reliability guarantees in order to meet the high dependability requirements of the application areas in which they are used. They run real-time applications and have access to detailed information about the electric motors they control and the industrial processes that are implemented.

The basic block diagram of a drive is given in Fig. 3, where the source power and command are the inputs, and the sink power is the output. The drive has two internal units: (i) a power modulator unit and (ii) a control unit. The power modulator unit modulates the source power of the modulating signal and returns the sink power. The modulating signal is generated by the control unit, which performs a resource-intensive computation based on the input command and sensed value of the sink power. The control unit may have safety features embedded such as a motor brake.

C. Example Industrial Setting

We provide an example industrial setting in Fig. 4, where an electric motor is used to control the conveyor belt speed. The conveyor belt machine is a carrying medium that uses belts rotating about two or more pulleys (the red box in the figure). The conveyor belt needs to move the load it carries with a specific speed profile determined by an industrial controller residing at the control level (Fig. 2), a PLC or an IPC.

Figure 5 shows a typical implementation consisting of a PLC, a sensor, a drive, and an electric motor. The sensor reads the load position and sends the data to the PLC, which determines the electric motor's current speed. The PLC sends the desired speed value to the drive. The drive controls the speed of the electric motor so that it is the desired value at the right time.

D. Baseline Drive Architecture

We first discuss a typical generic non-fog drive implementation [47]–[50], which we refer to as the "baseline architecture". We illustrate its design using AADL, as shown in Fig. 6a.

The drive takes as inputs (i) the source power, which is the main AC power line, and (ii) the command via a fieldbus interface that connects to a PLC, and outputs a sink power that operates an electric motor. The HMI is used as both the input and an output. As shown in Fig. 6a, the architecture consists of four components: operation component, communication component, control component, and power component. Each component uses dedicated hardware and software and has access to a shared bus, enabling it to be physically separated. Input-wise, all components are powered by the source power, but only the communication component has access to the input command. Output-wise, the power component returns the sink power, and the operation component has access to the HMI.

The **operation component** determines the operation mode of the drive. Figure 6b depicts the hardware, which has a CPU, RAM, and storage, and the software, which consists of real-time applications (Apps) that are running on a real-time operating system (OS). The operation component connects to the other components via the shared bus (Fig. 6a).

The applications running in the operation components (also shown in Fig. 6b) are as follows: (1) the mode control application (ModeControlApp), which engages and disengages the motor controller; (2) the management application (ManagmentApp), which configures the communication and controller parameters; and (3) the monitoring application (MonitoringApp), which implements safety functions. The ModeControlApp starts and stops the operation of the electric motor by engaging or disengaging the motor controller component based on the HMI or command inputs via the fieldBus interface. The ManagementApp sets the drive parameters such as the communication parameters via the HMI and motor control configurations such as the desired output (which is received from the communication component) via the shared bus. The MonitoringApp monitors the drive operation and engages safety functions when necessary.

The **communication component**, as shown in Fig. 6c, consists of the common hardware equipped with communication resources (the Switch box) to connect via the fieldBus interface with the ProfiNet/RT [52] standard. The component's software consists of a real-time application (CommunicationApp) that handles the network protocol and is running on a real-time

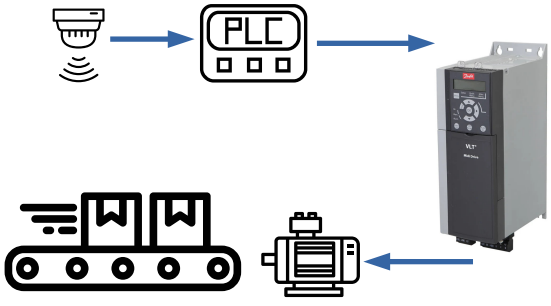
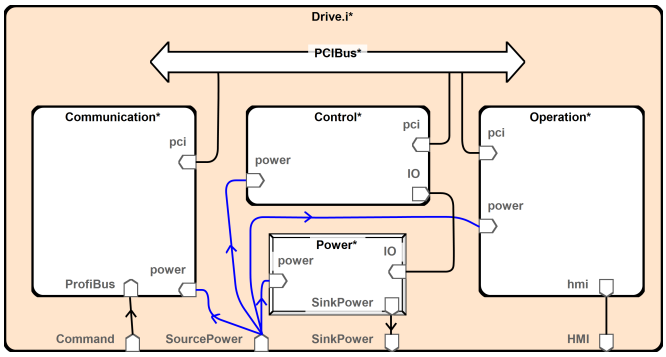
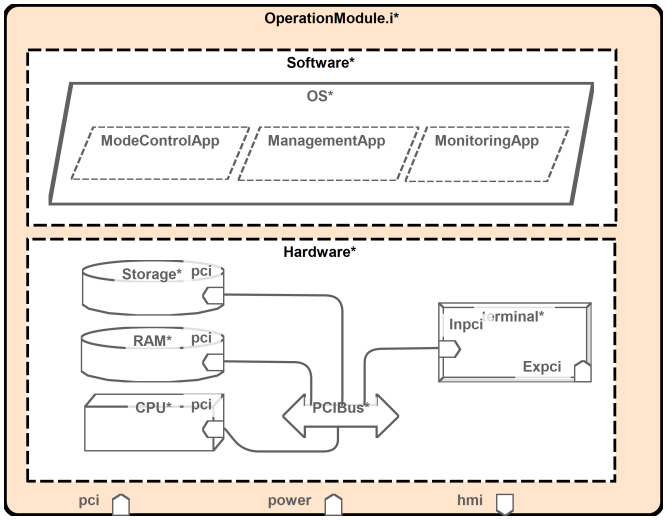


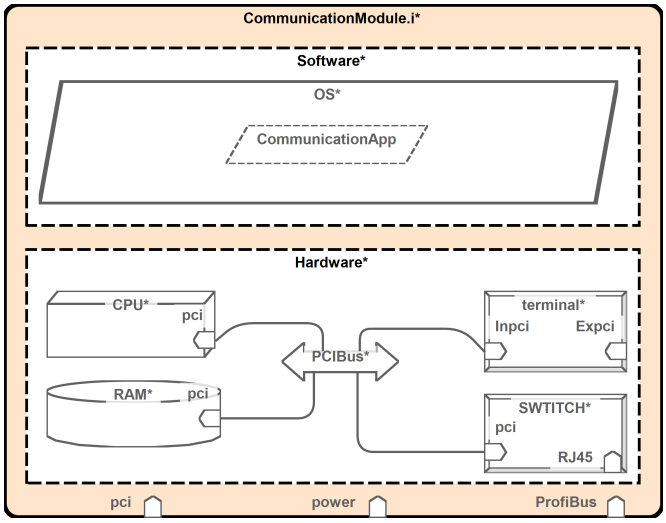
Fig. 5. Typical implementation of a conveyor belt system: The PLC uses the part present sensor to detect loads and send command to electric drive. Once the electric drive receives a run command, controls the electric motor.



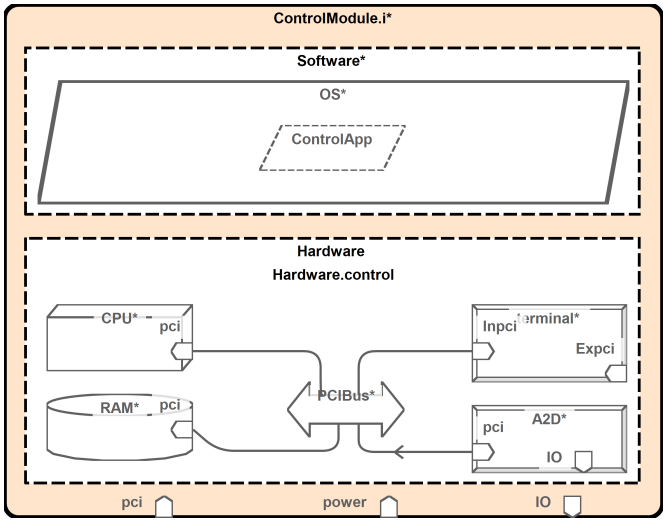
(a) Overview of the baseline architecture: Communication, Operation and Control components are connected via a shared bus. The control component uses Power module to control an electric drive.



(b) Operation component runs different applications on the shared hardware.



(c) Communication component runs the communication application on the real-time operating system.



(d) Motor control component generates the control signal for the power module to drive an electric motor.

Fig. 6. AADL diagrams of the baseline architecture: each component has dedicated hardware and software.

OS. The drive communicates with a PLC as the preliminary controller via a feildBus interface to obtain the desired motor output.

The **control component** is shown in Fig. 6d and consists of the hardware introduced earlier and software that implements a feedback control application (ControlApp) running on a real-time OS. The feedback control application is implemented according to the IEC 61131-3 standard [53] function blocks. Once the application is engaged by the input from the operation component via the shared bus, it outputs a signal to the power component via the I/O interface. The signal is generated based on the configuration received via the shared bus from the operation component and the feedback received from the power component via the I/O interface.

The **power component**, as depicted in Fig. 6a, takes the source power and the signal from the I/O interface as inputs and outputs the sink power, which operates the electric motor. This component is a power modulator that modulates the power line to the desired input reference. It also returns the current sink power setting via an I/O interface. The control component uses a feedback signal to control the sink power that operates the drive.

E. Requirements and KPIs

We elicited system-level requirements for designing a fogified drive architecture considering the vision of Industry 4.0. Table I lists the requirements and relevant rationale. The requirements specify that the fog-based drives should both have fog computing capabilities (for example, the ability to

TABLE I
SYSTEM-LEVEL REQUIREMENTS FOR FOG-BASED DRIVES

#	**Requirement**	**Rationale**	**Realization in the fog-based architecture (Section III-B)**
1	Drives shall be designed according to industrial standards.	To ensure drive compatibility with the industrial environment	IEC61800-based
2	Drives shall host mixed-criticality applications.	To virtualize critical and control applications and run IIoT applications envisioned in Industry 4.0	Virtualization and separation mechanisms
3	Drives shall perform accurate motor control.	To enable the virtualization of control applications to meet their non-functional properties (control performance)	Configuration to provide good control performance (20 ms response time)
4	Drives shall be configurable.	To enable the allocation of the necessary resources to mixed-criticality applications and to ensure portability and deployment of applications	Implemented via middleware
5	Drives shall have fault tolerant communication.	To ensure the connectivity and responsiveness of the drive	Via IEEE 802.1 TSN configuration [51]
6	Drives shall have a time-constrained communication interface.	To guarantee communication with bounded latency for critical applications exchanging messages	IEEE 802.1 TSN solution
7	Drives shall have Cloud access.	To enable deployment and communication for IIoT applications	Via Ethernet
8	Drives shall perform data analytics.	To avoid sending all data to the Cloud and to optimize resource management	Machine learning solutions
9	Drives shall enforce security policies.	To avoid malicious activity and define permissible user actions	Policies enforced via middleware
10	Drives shall be fault-tolerant.	To ensure fault-tolerant and high-integrity operation of safety-related applications	Fault detection, isolation, and recovery mechanisms

perform data analytics) and still deliver safety-related drive functions (for example, accurate motor control).

We have also defined several key performance indicators (KPIs) that, should be used to evaluate the solutions implemented using the fog-based drive architecture. Table II shows the KPIs and the relevant motivation.

III. FOG-BASED ELECTRIC DRIVES

In this section, we briefly present the FORA fog computing platform (FCP) in Section III-A, which was used to design our fogified drive architecture. We describe the proposed drive architecture and its AADL model in Section III-B.

A. FORA FCP Reference Architecture

The FORA FCP reference architecture was introduced in [15] to bring the fog computing paradigm to IIoT applications. The FCP consists of FNs connected to each other and to the machines through a deterministic communication solution, namely, IEEE 802.1 TSN [51] (Fig. 1). The aim of TSN is to provide timing guarantees for demanding applications such as critical applications. Although TSN is currently investigated in wireless networks (802.11ax/Wi-Fi 6 and 5G), we consider wired TSN in this work. The investigation of emerging 6G communication technologies in the context of Industrial IoT is an orthogonal issue to our work [55], [56].

It also guarantees bounded communication latency between nodes in the FCP. The key components of the FORA FCP are (1) deterministic virtualization, (2) middleware, and (3) mechanisms for resource management and orchestration [15].

Because the FCP hosts mixed-criticality applications with differing requirements, an isolation mechanism is required to prevent low-criticality applications from interfering with high-criticality applications. In the FORA FCP, each FN utilizes a hypervisor that provides spatial and temporal isolation among the mixed-critically applications.

The FORA FCP uses middleware to support the implementation of distributed critical applications and non-critical applications such as IIoT applications that implement data analytics, updates, and regular checks. It also offers services for dependability, such as resource monitoring, safety and security monitoring, and machine learning services.

The FORA FCP employs resource management techniques for supply and demand alignment in an FCP and support configuration for optimizing resource utilization. With the resource management mechanisms, the available resources of each FN and the fog landscape are monitored at runtime, and services for the placement, deployment, and support of future applications and dynamic applications that may migrate across the FNs of the FCP are provided.

These services enable real-time decision making, security services, and resource prioritization. More information on the FORA FCP middleware is presented in [15].

Resource management techniques align resource supply and demand in an FCP and support FCP configuration for optimizing resource utilization. With the resource management mechanisms, the available resource of each FN and the fog landscape is monitored at runtime, which allows the placement, deployment, and support of IIoT applications. More information on the resource management mechanisms for the FORA FCP is presented in [15].

TABLE II
KPIs

#	Criteria	Motivation
1	Safety	Safety-critical industrial applications should be able to be hosted with no interference from less- criticality applications.
2	Security	Platform-level security services will be provided and the threat of successful attacks would be reduced.
3	Virtualized critical control performance	Using the configuration mechanisms, the FN should be able to configure its resources and the TSN switches according to the operational needs of the virtual control tasks such that the control performance can be guaranteed.
4	Hardware costs	Hardware spending should be reduced because the platform enables virtualization that converges various functions into one FN.
5	Data analytics	The FNs will implement the OPC unified architecture [54] and connect directly to the equipment, sensors, and actuators. The data can then be analyzed locally using analytics applications at the edge. A subset of production data is then transferred securely to the Cloud for big data analytics.

B. Fogified Drive Architecture

This section describes our proposed fogified drive architecture, which is based on the FORA FCP [15]. We assume that using our proposed fogified architecture, drives are developed as FNs that are connected to each other and to the machines through TSN [51]. The fogified drives implement both the typical drive functionality (from the baseline architecture) and extra functionalities, as envisioned in the FORA FCP in an FN. We provide an overview of such a drive developed as an FN and modeled using AADL in Fig. 7a. The architecture consists of a hardware component, a software component, and a power modulator unit for operating electric motors. The fogified drive takes as inputs (1) the source power and (2) the network connection via the TSN interface. It outputs the sink power and uses the network connection to send data.

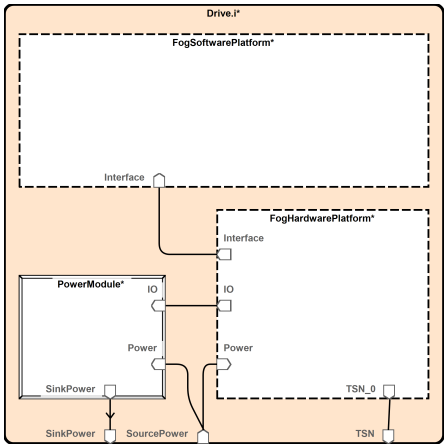
The **hardware component** is depicted in Fig. 7b, which is equipped with a commercial of-the-shelf (COTS) multicore processor (CPU), RAM module, storage resources, an analog to digital converter module (A2D), and a network switch (TSNCard) for TSN capabilities. These resources are shared for running applications that sit on the software component. Similar to [15], we consider a fog-based design that uses TSN because it supports mixed-criticality bounded-latency communication via multiple traffic types. For the scheduled traffic, which requires synchronized schedule tables, TSN employs a network-wide clock synchronization protocol, namely, IEEE 802.1AS [57] with sub-microsecond precision. The FN utilizes its advanced networking capabilities to interact with the environment, including sensors, actuators, other FNs, and remote Cloud facilities.

An example of a COTS multicore processor is the Intel Atom processor, which implements hardware virtualization extensions, such as Intel's VT-x and VT-d, second level address translation (SLAT), and single-root I/O virtualization (SR-IOV) [58]. Hardware virtualization extensions allow the hypervisor to host virtual machines with dedicated operating systems.

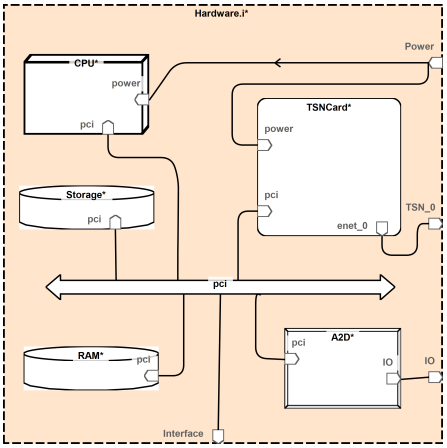
The **software component** has a software stack that consists of a hypervisor, middleware, partitions with dedicated operating systems, and an application layer, as shown in Fig. 7c. As mentioned in [15], the software stack can take advantage of existing open-source software stacks for the edge, e.g., OpenStack [59]. The middleware can use application layer protocols such as MQTT-SN [60] or CoAP [61] for higher-level component communication and TSN and OPC UA for lower-level component communication. Mixed-criticality applications that share the same hardware resources are separated into different virtual machines (partitions) enforced using hardware-supported virtualization [62] based on hypervisors such as PikeOS [63], ACRN [64], or Xen [21].

A hypervisor partitions its resources, such as processor cores and time, main memory, and I/O devices, to achieve strict temporal and spatial isolation of applications with mixed criticalities. The internals of our proposed hypervisor component are depicted in Fig. 7e, where we assume PikeOS [63] is used as the hypervisor. PikeOS can also make use of static partition schedules decided at design time to enforce temporal isolation. A static partition schedule consists of several partition slices where the partition is running based on a partition table that captures the start and end of partition slices (see [20] for more information on partitions and partition slices).

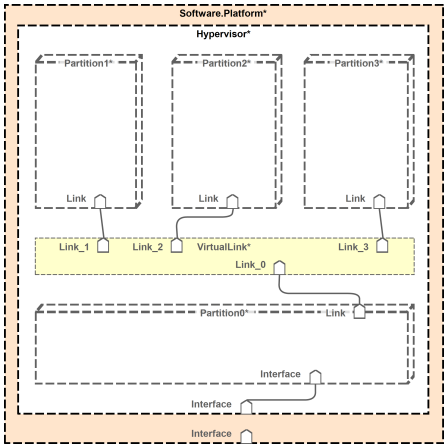
The fogified drives can run all the drive applications from the baseline architecture, all of which are critical and modeled as periodic hard real-time tasks and messages [65]. They can also run IoT applications that are non-critical, may migrate in and out of the drives, and implement tasks such as data analytics. Each IoT application was also implemented as tasks and messages. Our model assumes three types of applications—control, communication, and operation—which are assigned to separate partitions. The model for each partition is illustrated in Fig. 7d, where an OS runs the applications on the application layer. The OS runs the applications and uses the AppServices and AppSupport for using shared resources and assuring dependability. The control partition has a soft-PLC OS, and the control application was implemented using the IEC 61131-3 standard [53] function blocks. We also define a Quality-of-Control (QoC) for control applications that captures the control performance; see [66] for more information about the QoC. The communication partition has a real-time OS



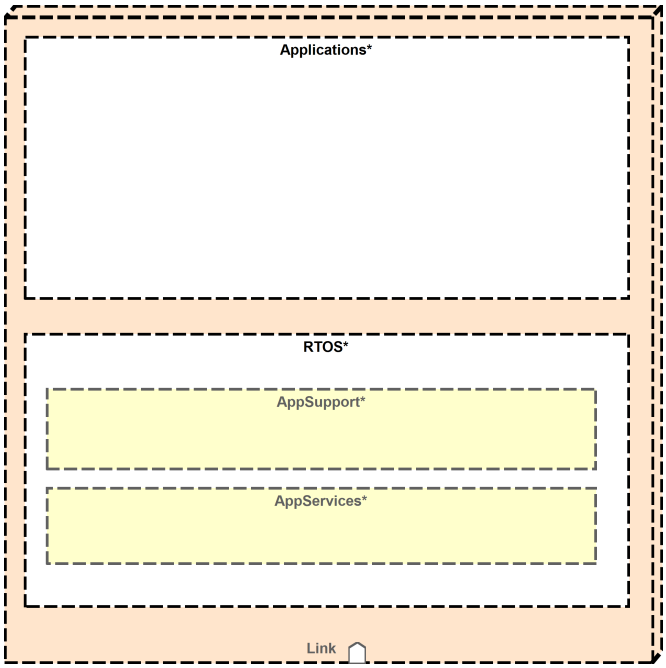
(a) Overview of a fogified drive: the power module drives an electric motor.



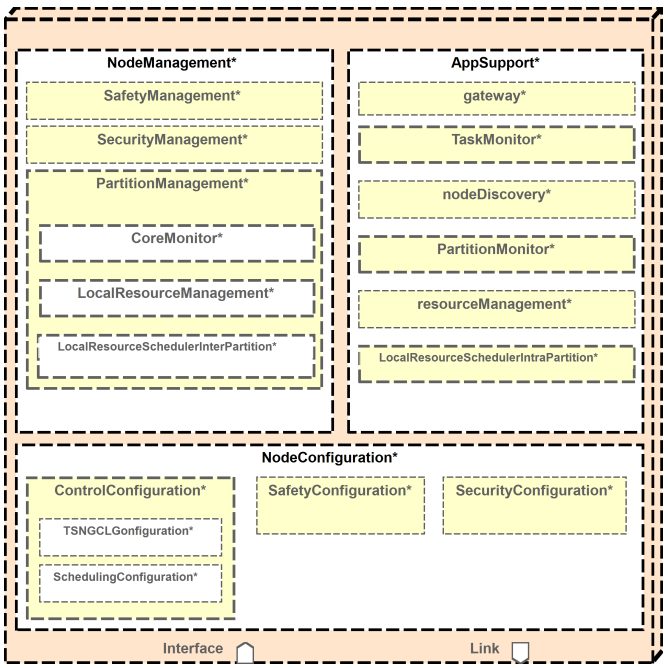
(b) Hardware platform.



(c) Software platform consists of several partitions managed by a hypervisor.



(d) A partition has an operating system that runs different applications.



(e) Hypervisor isolates the mixed-criticality applications using partitions.

Fig. 7. AADL diagram of the fogified architecture: mixed-criticality applications run on shared hardware.

that runs applications for controlling network traffic, applying security mechanisms, handling application traffic, and deciding the TSN message schedule tables, called Gate Control Lists (GCLs). We assume that the operation partition has an OS to run different types of applications, including a machine learning application.

The **power modulator unit**, as depicted in Fig. 7a operates the same way as the power component in the baseline architecture. It takes the source power and the signal from the I/O interface as inputs and outputs the sink power, which operates the electric motor. The I/O interface is connected to the hardware platform where an analog to digital converter is accessible to control the partition that uses the I/O signal to operate the electric motors.

IV. Evaluation

In this section, we first provide the details of the use case (UC), which we used for evaluation in Section IV-A. We evaluate the UC in Section IV-B on several aspects using the KPIs in Table II, discussing the suitability of the fog-based drive design for industrial applications.

A. UC Description

We used the proposed fogified drive to model a **self baggage drop system**. A self baggage drop system is a well-known and widely used machine in airports to automatically collect and distribute passenger baggage (Fig. 8). The machine is realized with several conveyor belts (see [40] for details of a conveyor belt), and it collects baggage from passengers and delivers

Fig. 8. Self baggage drop system in Brisbane airport: baggage is collected and carried to determined destinations using several conveyor belts.

the baggage to special vehicles (that carry the baggage to an airplane according to the destination). In this UC, we consider a typical machine, as depicted in Fig. 8, which is fed with baggage from every input location, weighs the baggage, reads the tag of the received baggage, and determines the destination of the baggage according to the tag by accessing a database. It then conveys the baggage toward the destination from one of the output locations, and then reports the machine usage and baggage delivery to the costumer's cloud.

Figure 9 shows a schematic of the machine. As depicted in the figure, the machine has four input conveyor belts (green), one main distribution conveyor belt (yellow), two actuators (blue) that push baggage, and two output conveyor belts (red). Each input conveyor belt has an electric motor that drives a belt, two part-present sensors that sense the presence of a load, a tag reader for reading baggage tags, and a weight sensor for weighing the load. The main distribution conveyor belt has two electric motors for driving the belt and two actuators for pushing the loads. Each output conveyor belt also has an electric motor and two part-present sensors.

We consider two different implementations for the UC: (1) a baseline architecture and (2) our proposed fogified architecture. We also modeled the fogified implementation of the UC with AADL and show its diagram in Fig. 10. Table III summarizes the hardware equipment costs and number of units used in the two implementations of the UC.

Drives in the fogified architecture cover the functionality of the baseline drives and can run all drive applications from the baseline architecture. We show example UC applications running on FN N_1 in Table V, which shows the applications and relevant mapping of these applications to the processing elements of the baseline architecture.

TABLE III
SUMMARY OF THE UC HARDWARE EQUIPMENT

Equipment Name	**Unit Price (USD)**	**No. of units in the fogified Architecture**	**No. of units in the baseline Architecture**
Part present sensor	100	12	12
Electric motor	200	8	8
Tag reader	50	4	4
Weight sensor	50	4	4
Push actuator	100	2	2
Belt	50	7	7
PLC	500	0	6
IPC	1,000	0	1
Switch	100	3	2
Fogified drive	1,000	2	0
Electric drive	350	0	8

B. Assessing the KPIs

In this section, we address each KPI separately and consider it as an evaluation criterion, using the evaluation method mentioned in Table. IV to evaluate it on the UC. The results of the evaluation are as follows.

Safety: As mentioned, the FNs in the UC host mixed-criticality applications. High-criticality applications should be protected from low-criticality applications. This is achieved by spatial and temporal separation mechanisms implemented via the hypervisor. Consider the UC applications mapped to FN N_1 in Table V, where column 1 shows the criticality of the application, ranging from 3 for the highest criticality to 0 for the lowest criticality.

The baseline architecture employs spatial separation that uses dedicated processing elements, i.e., drives, PLCs, and IPCs, to isolate applications of different criticalities. For example, high-criticality applications are assigned to run on PLCs, which guarantee deterministic execution, and the less-critical applications are assigned to be run on IPCs. Column 7 in Table V shows the mapping of the applications to processing elements.

The same level of separation should be achieved in the fog-based solution. Hence, the fogified architecture uses a hypervisor and dependable middleware to achieve spatial partitioning. The hypervisor provides deterministic access to shared resources (spatial partitioning) and the temporal isolation of mixed-criticality applications via a static configuration table.

TABLE IV
EVALUATION METHODS OF KPIS

Criterion	**Evaluation**
#1	Provision of isolation via partitioning and evaluation of the applied overhead
#2	Protection of high-criticality applications and provision of authentication mechanisms for communication
#3	Optimization of the control performance for control applications
#4	Comparison of the hardware cost of a UC implementation using the baseline architecture and the fogified architecture
#5	Provision of a decentralized machine learning solution

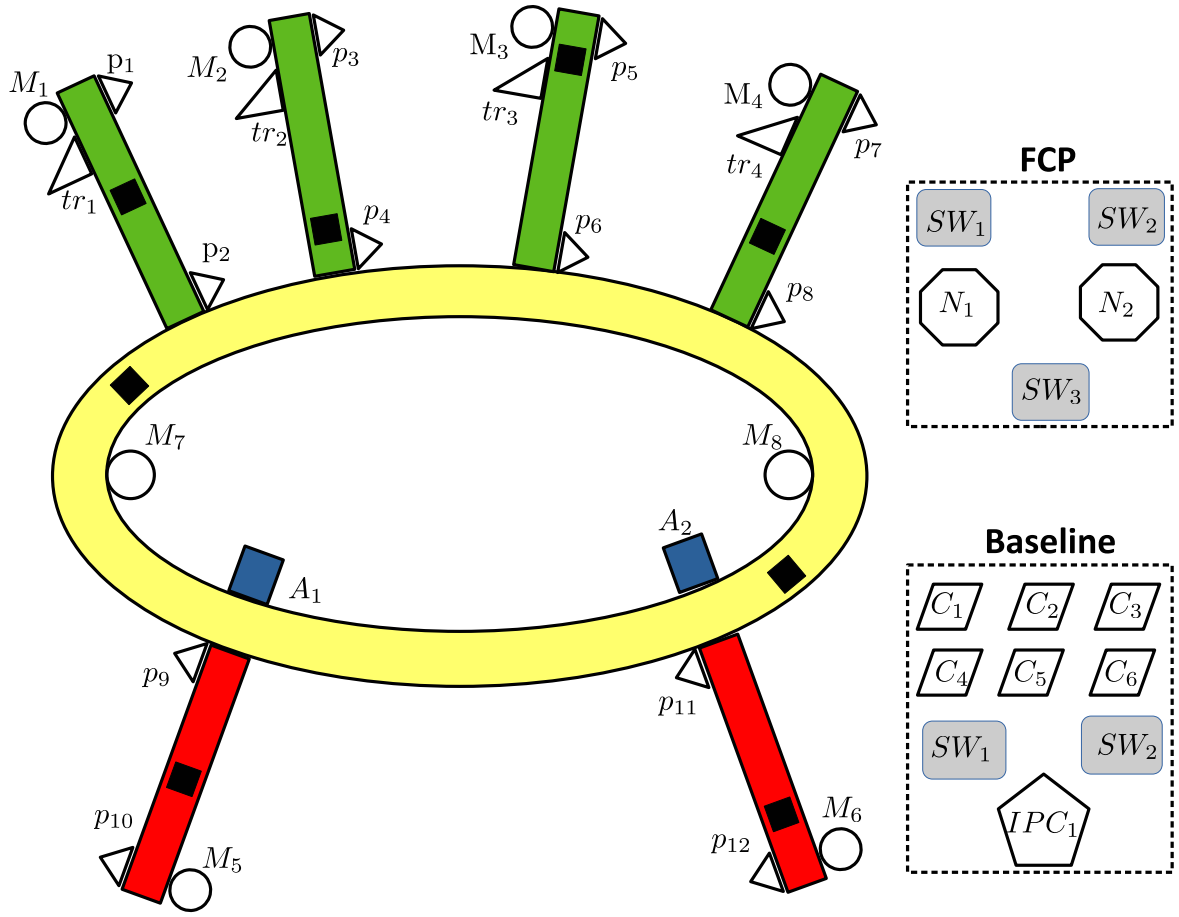


Fig. 9. UC Schematics: The green belts are inputs, and the red belts represent the output. The round yellow belt is the main distribution belt. There are eight electric motors (M), 12 part present sensors (p), and four tag reader sensors (tr). The small black boxes represent the baggage. Network switches are indicated by SW in both the FCP and baseline architectures, whereas N denotes a fogified drive in the FCP architecture and C denotes a PLC in the baseline architecture. The baseline architecture has an IPC.

The configuration of spatial and temporal isolation can be achieved using approaches such as the one proposed in [20], which relies on a heuristic algorithm for determining partitions and assigning tasks to the partitions. In the proposed approach, each partition is dedicated to a criticality level and consists of a set of partition slices. The start and end of each partition slice is captured in partition tables that are determined statically. The fogified architecture implements the proposed approach in the node configuration component (Fig. 7e). The configuration generates an optimized partition table for each FN, the mapping of applications to partitions based on their criticality levels, and the schedule tables for the critical tasks.

The advantage of virtualization and partitioning is that it reduces hardware costs, i.e., more equipment and mixed-criticality functionality can be hosted as software tasks on the FNs. However, we were interested in determining whether we could achieve a level of performance equal to that of the baseline architecture, that is, whether the virtualization would introduce much overhead. We evaluate this via a performance index defined as the number of processing element types and the overhead introduced by the separation mechanisms. The proposed approach in [20] considers the partition overhead in the partition tables. As shown in Table V, the applications are mapped to the FN N_1 in the fogified architecture, whereas on the baseline architecture, the processing elements are drives, PLCs, IPCs, and the Cloud; this result demonstrates that the performance on the fogified architecture is increased.

We evaluated the performance of our solution employed on the fogified architecture. As shown in Table V, the example applications running on FN N_1 are separated using the proposed solution via partitioning. Our solution successfully determined the partitions and partition slices where the applications implemented as tasks are running. Moreover, it generated partition schedules that capture the start and end of partition slices.

The applications in Table V are implemented as tasks, and column 8 shows the utilization of each task when implemented on its respective core, i.e., the fraction of the 100% core utilization. Column 9 in the table shows the total utilization for each application without considering the overhead introduced by the partitioning-based virtualization; note that because the processors are dual-core, the total capacity of the two cores is 200%. Column 10 shows the total utilization when virtualization is used, accounting for the overheads. Three

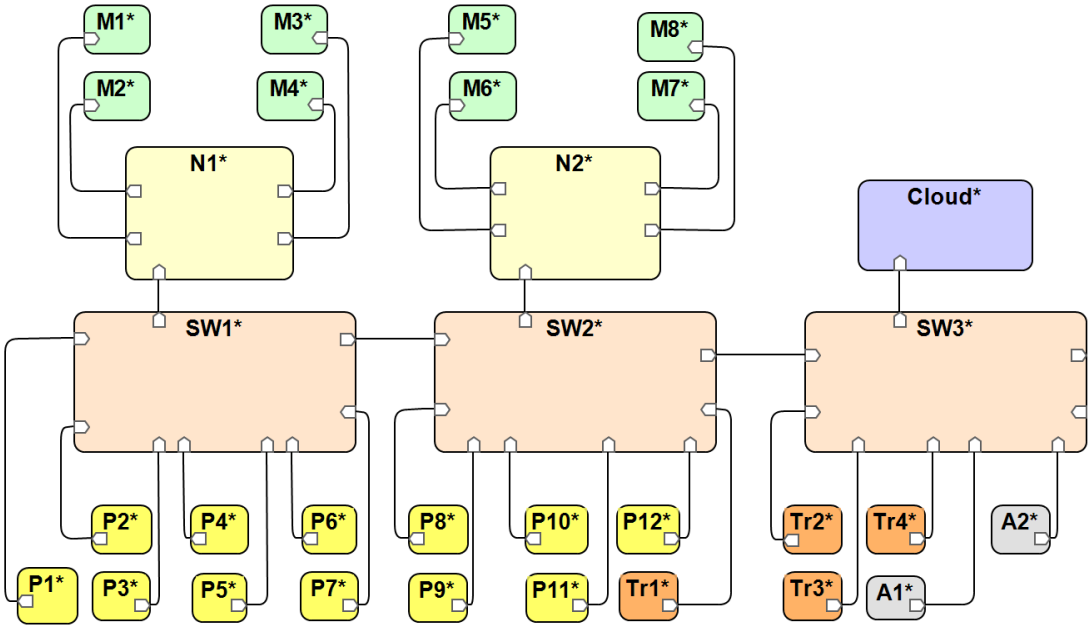


Fig. 10. UC AADL model: part present sensors (P), tag reader sensors (tr), and actuators (A) are connected to fogified drives (N) via network switches (SW). The fogified drives (N) are connected to motors (M) to control them.

partitions were generated, and the core utilization of each partition increased by an average of 9% compared with the total utilization of applications without partitioning. In other words, the results show that to enable the safety-related separation of mixed-criticality applications, our proposed solution introduces only a 9% overhead on the total FN utilization.

Security: Security mechanisms are required to adequately protect the system against adversaries. A compromised system may allow the safety requirements to be violated. We briefly discuss security solutions that can be used in the UC's fogified architecture. These should all be deployed in parallel, as an instance of a defense-in-depth approach [67]. The various mechanisms proposed here are summarized in Table VI, see [68] for details of the mitigation.

As shown in the table, security attacks can be divided into two categories: execution-based and communication-based. The execution-based attacks target the configuration of the FN to interrupt the execution of the assigned applications. The FNs in the fogified architecture employ user access policies and apply configurations that are determined by the node configuration component. Partitioning introduces another protection mechanism because safety-critical applications are isolated in separate partitions. The isolation is enforced both spatially and temporally such that not only each partition is not accessible to other partitions but also its execution does not interrupt the execution of other partitions.

Communication-based attacks affect the operation of a system by tampering with the network communication. Because the fog-based solution uses TSN, we focus here on the security vulnerabilities of TSN networks. TSN relies on clock synchronization for some of its mechanisms, e.g., the scheduled traffic type implemented with GCLs via 802.1Qbv. Hence, clock synchronization can be a security weakness for TSN [69]–[71].

We now consider execution-based attacks. We assume that an attacker has control of FN N_1 and tries to interrupt the execution of high-criticality applications. The malware, which we denote as task γ_{36}, runs on a partition with criticality level 0 (Table V), and targets the interruption of the partition with criticality level 3, which runs motor control applications. The security configuration component implemented in the node configuration component (Fig. 7e) monitors the execution of tasks according to their pre-computed schedule tables, identifies tasks with suspicious activity (e.g., unresponsive tasks or unauthorized access to resources), and performs predefined actions on the task (e.g., stopping the task and recovering the application).

Performance of virtualized control: In the baseline architecture, critical control applications are assigned to run on dedicated processing elements, i.e., PLCs, which are configured to meet the non-functional performance requirements of the control applications, such as deadlines and QoC. The QoC captures different aspects of the performance of a controller, i.e., rapidity and accuracy of the controller, see [66] for more information.

The fogified architecture uses deterministic hypervisors to virtualize applications on FNs, similar to [58], where hypervisors provide deterministic access to shared resources via a static configuration table and provide spatial and temporal isolation of mixed-criticality applications via partitioning. Because critical control applications are virtualized and implemented as tasks, the configuration of the fogified architecture

TABLE V
UC'S APPLICATIONS RUNNING ON THE FN N_1

Criticality	Apps.	No. of tasks	No. of streams	Bandwidth utilization	Routing	Mapping (baseline)	App. core utilization	Total utilization w/o partitioning	Total utilization incl. partitioning
3	MC[1](γ_1)	2	1	0.2%	$p_1 \rightarrow SW_1 \rightarrow N_1$	D_1	15%	126%	139%
	MC[1](γ_2)	2	1	0.2%	$p_3 \rightarrow SW_1 \rightarrow N_1$	D_2	15%		
	MC[1](γ_3)	2	1	0.2%	$p_5 \rightarrow SW_1 \rightarrow N_1$	D_3	15%		
	MC[1](γ_4)	2	1	0.2%	$p_7 \rightarrow SW_1 \rightarrow N_1$	D_4	15%		
	MM[2](γ_9)	4	2	0.3%	$N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow N_2$ $N_2 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$	C_1	12%		
	MM[2](γ_{10})	4	2	0.3%	$p_9 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$ $p_{11} \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$	C_2	12%		
	MM[2](γ_{11})	4	2	0.3%	$N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow SW_3 \rightarrow A_1$ $N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow SW_3 \rightarrow A_2$	C_3	12%		
	SS[3](γ_{15})	2	0	0%	–	N/A	10%		
	SS[3](γ_{16})	2	0	0%	–	N/A	10%		
	SS[3](γ_{17})	2	0	0%	–	N/A	10%		
2	SR[4](γ_{21})	1	1	0.1%	$p_2 \rightarrow SW_1 \rightarrow N_1$	C_1	1.5%	18%	20%
	SR[4](γ_{22})	1	1	0.1%	$tr_1 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$	C_1	1.5%		
	SR[4](γ_{23})	1	1	0.1%	$p_4 \rightarrow SW_1 \rightarrow N_1$	C_2	1.5%		
	SR[4](γ_{24})	1	1	0.1%	$tr_2 \rightarrow SW_3 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$	C_2	1.5%		
	SR[4](γ_{25})	1	1	0.1%	$p_6 \rightarrow SW_1 \rightarrow N_1$	C_3	1.5%		
	SR[4](γ_{26})	1	1	0.1%	$tr_3 \rightarrow SW_3 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$	C_3	1.5%		
	DA[5](γ_{33})	4	3	0.5%	$N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow SW_3 \rightarrow$ Cloud Cloud$\rightarrow SW_3 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$ $N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow SW_3 \rightarrow$ Cloud	IPC_1	9%		
0	ML[6](γ_{34})	6	2	0.4%	$tr_1 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$ $N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow SW_3 \rightarrow$ Cloud	N/A	8%	16%	17%
	ML[6](γ_{35})	6	2	0.4%	$tr_2 \rightarrow SW_3 \rightarrow SW_2 \rightarrow SW_1 \rightarrow N_1$ $N_1 \rightarrow SW_1 \rightarrow SW_2 \rightarrow SW_3 \rightarrow$ Cloud	N/A	8%		
	Sum							160%	175%

[1] Motor Control
[2] Machine Management
[3] Safety Service
[4] Sensor Reading
[5] Database Access
[6] Machine Learning

(e.g., task and communication scheduling) has an impact on the QoC of control applications [72], [73].

The control functions are monolithic in the baseline architecture and do not exchange critical messages over the Profinet fieldbus. However, in the fog-based solution, critical control tasks exchange messages with each other as industrial "things", e.g., as sensors and actuators. Furthermore, the fogified architecture shares the same communication medium, i.e., TSN, for hard and soft real-time, non-critical, and best-effort communication. TSN has mechanisms to guarantee the timing requirements of critical streams, e.g., via scheduling enforced by GCLs [73], but these must be properly configured.

TABLE VI
THREATS AND THEIR MITIGATION

Threat	Mitigation
Man-in-the-middle, impersonation	Confidential, authenticated com. channels
Attack impact	Service isolation (e.g., partitions)
Remote attacks	Firewalls, endpoint whitelisting
DoS	Redundant network topologies
TSN security	Isolation of the TSN protocol, per-stream filtering
Physical attacks	Hardware token for configuration changes
Detection	Security monitoring services

The node configuration component determines the configuration needed to guarantee good control performance, measured via the QoC metric. The component uses a metaheuristic solution proposed in [20] to optimize the hypervisor partition tables, map the tasks to the processing cores of the multi-core processors of the FNs, assign the tasks to partitions, and schedule the tasks inside the partition tables, optimizing the QoC of control applications. The proposed approach uses JitterTime [66] to evaluate the QoC of control applications and uses the QoC value as an objective for optimizing the schedule tables. The best QoC is achieved when control tasks experience zero jitter and the end-to-end delay of control applications are minimum.

The node management component also employs the con-

straint programming-based schedule synthesis strategy, which aims to maximize the QoC and satisfy the deadlines of real-time messages, as proposed in [30] to schedule the traffic. The proposed approach schedules each frame individually and generates static schedule tables. Optimizing schedules for QoC means minimizing end-to-end delay and jitter for real-time messages and maximizing the execution slice for control applications. The execution slice is the time interval between reception of the control input messages and transmission of the control output messages, see [30] for more information.

We evaluated the control performance of the proposed UC solution. We assume that the applications in Table V are running on FN N_1, which has a dual-core processor and exchanges network streams via TSN. Columns 3 to 6 in the table show the task and stream details, which are the total number of tasks, total number of streams, total bandwidth utilization of streams on the 1 Gbps link, and the the routing, respectively. We assume that applications γ_1 to γ_4 are motor control applications that control the speed of the electric motors. Each motor control application exchanges a message with a part-present sensor, calculates the control function similar to the function presented in [72], and applies the respective output signal to electric motors.

Our proposed strategy for communication scheduling in TSN successfully scheduled all streams, i.e., none of the deadlines were missed, and optimized the schedules for the QoC. We also evaluated the performance of our proposed optimization strategy for task scheduling, and the proposed system successfully scheduled all the tasks and determined the task mapping to the cores.

The results show that all streams have zero jitter, which improves the QoC. We present the results in Table VII, where the I/O jitter, maximum end-to-end delay of the streams, and QoC values are reported using JitterTime [66], which simulates the behavior of the control application with respect to execution timing. We achieved a good control performance with an average QoC value of 0.092 using the objective defined in [20].

Hardware cost: From a monetary perspective, the fogified architecture provides incentives concerning reducing the hardware cost. The common equipment cost for both the architectures C_{cost} of implementing the system shown in Fig. 9 is the sum of the equipment costs that are common for both, i.e., the first six items of Table III. We compute the common equipment costs as C_{cost} = 3.750 (all values are in USD). The architecture-specific costs can be divided into baseline cost B_{cost} and fogified cost F_{cost}. Hence, the total costs of the baseline and fogified architectures are T^B_{cost} and T^F_{cost}, respectively. Therefore, $T^B_{cost} = B_{cost} + C_{cost}$ and $T^F_{cost} = F_{cost} + C_{cost}$. We computed the total costs for the baseline architecture and the fogified architecture as $T^B_{cost} = 10,750$ and $T^F_{cost} = 6,050$, respectively.

TABLE VII
QoC OF CONTROL APPLICATIONS RUNNING ON FN N_1

App.	IO Jitter	Max E2E delay (ms)	QoC
γ_1	0	31	0.089
γ_2	0	29	0.081
γ_3	0	33	0.101
γ_4	0	32	0.096

Furthermore, we consider scenarios in which the UC is updated with new features. For example, the UC employs an application that stops the conveyor belts when they do not carry a load to save energy. In the fogified architecture, this application is deployed on FNs via the Cloud connection. However, in the baseline architecture, the application is deployed on each processing element individually, increasing the management costs.

Data analytics: In a Cloud-based AI implementation, distributed participants upload their data to the Cloud, where the collected historical data is processed in a centralized fashion. The downsides of this approach are that (1) it is bandwidth- and time- consuming to upload this data and (2) it impairs privacy via potential data leakage. The solution is to utilize the edge device resources to carry out data analytics at the edge. A recent study [74] analyzed the minimal requirements of a reconstruction attack and provided insight into choosing the model size and architecture during the design of a ML algorithm. With the release of the General Data Protection Regulation (GDPR) [75], data privacy protection has become a legal requirement.

We assume a scenario in which four airlines have decided to predict passenger satisfaction. As part of this survey, passenger baggage information must be processed. Implementing an ML approach on the UC implemented with the baseline architecture is impossible because processing elements such as the PLCs and IPC do not provide sufficient resources, services, and connectivity to deploy such an application. Thus, a decentralized machine learning approach is implemented to run on the FNs of the fogified architecture.

The airlines aim to collaboratively train a predictor for passenger satisfaction. First, the airlines locally train their own models based on their own data. Second, they share the updated model with the Cloud based on their own data. Finally, the Cloud aggregates and updates the global model and sends it back to the airlines. The entire procedure can be repeated multiple times. The features we used in this experiment are passenger features such as gender and age, and flight features such as flight class, departure time, baggage weight, flight destination, number of bags, and baggage size. We cast all categorical features into numerical ones and normalized them to a range between zero and one during the preprocessing step. Here, we employ logistic regression for the binary predictor. Given $X \in \mathbb{R}^{N \times d}$, N is the total number of data and d is the dimension. Logistic regression maps the linear product of features to the range between zero and one using a sigmoid function, defined as $S(z) = \frac{1}{1+e^{-z}}$, where z is a linear combination of features defined as $z_i = \sum_j^d w_j x_{ij} \quad \forall i \in [1, N]$.

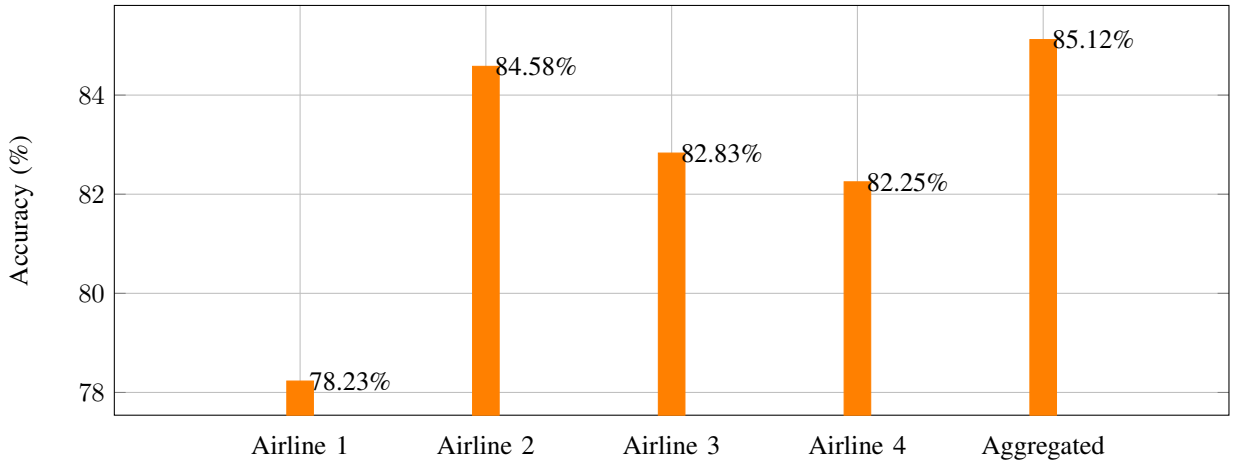


Fig. 11. Distributed ML: four airlines jointly train a global model. The aggregated model outperforms a single entity after one-shot communication and demonstrates a good performance.

In Fig. 11, we illustrate the advantage of the aggregation step using one-shot communication, where the aggregated accuracy of models created by individual airlines.

The implementation significantly decreases the upload bandwidth. For example, it may save $(B-1)\times 8d$ bytes for logistic regression model, where B is the batch size for one shot and, d is the data dimension. This will save $n\times(B-1)\times 8d$ for n-shot communications, and the advantage is more obvious with high-dimensional data (d is large). Moreover, the FN can reply to customer queries without a large delay, which is the main drawback of Cloud-based AI. The data remaining in the generation location significantly reduce the risk of privacy leakage.

V. Conclusions

In this study, we addressed electric drives, which are widely used in industrial applications. We proposed a way to re-engineer them as FNs, a process that we called fogification, based on the recently proposed FORA FCP reference architecture. We modeled and designed the fogified drive architecture using the AADL, capturing the main components and their interconnections. The design was driven by a set of requirements that we created that consider both the baseline functionality of the drives and their envisioned role as FNs. Fog-based drives are naturally located at the edge of the network, close to the machines, sensors, and actuators. Using the proposed architecture, we identified a solution for a self baggage drop system UC. We defined several KPIs for evaluating the suitability of our fog-based architecture for the UC. The evaluation results show improved performance, reduced hardware cost, and an increased analytics capability, without jeopardizing safety and security or the performance of virtualized critical control applications. As the evaluation shows, the fog-based drive architecture is a promising approach to implement the functionalities envisioned in Industry 4.0. In our future work, we will further integrate required technology components for the implementation of the use case.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie grant agreement No. 764785 (FORA—Fog Computing for Robotics and Industrial Automation), and from the European Union's Horizon 2020 Research and Innovation Programme under the ECSEL Joint Undertaking (JU) grant agreement No. 101007273 (DAIS—Distributed Artificial Intelligent System).

REFERENCES

[1] H. Bauer, C. Baur, D. Mohr, A. Tschiesner, T. Weskamp, K. Alicke, and D. Wee, "Industry 4.0 after the initial hype–where manufacturers are finding value and how they can best capture it," *McKinsey Digital*, 2016.

[2] P. Daugherty, P. Banerjee, W. Negm, and A. E. Alter, "Driving unconventional growth through the industrial internet of things," *accenture technology*, 2015.

[3] D. R. Harp and B. Gregory-Brown, "IT/OT convergence bridging the divide," *NEX DEFENSE*, 2014.

[4] M. García-Valls, T. Cucinotta, and C. Lu, "Challenges in real-time virtualization and predictable cloud computing," *Journal of Systems Architecture*, vol. 60, no. 9, pp. 726–740, 2014.

[5] W. Steiner and S. Poledna, "Fog computing as enabler for the Industrial Internet of Things," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 310–314, 2016.

[6] IEEE, "IEEE 1934-2018 - IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing," 2018.

[7] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.

[8] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A Survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–41, 2019.

[9] TTTech Computertechnik AG, "Nerve," http://tttech.com/products/industrial/industrial-iot/nerve, 2019 (accessed October 7, 2019).

[10] Nebbiolo Technologies, Inc, "Nebbiolo," https://www.nebbiolo.tech/, 2021 (accessed April 1, 2021).

[11] I. Boldea and S. A. Nasar, *Electric drives*. CRC press, 2016.

[12] V. Karagiannis, S. Schulte, J. Leitao, and N. Preguica, "Enabling fog computing using self-organizing compute nodes," in *International Conference on Fog and Edge Computing (ICFEC)*, 2019, pp. 1–10.

[13] Fog Computing for Robotics and Industrial Automation (FORA), "Fog Computing Platform: requirements and initial designs," https://drive.google.com/file/d/1QwBfcqij72ZdeMWmhwAwm-MdSHePElUy/view, 2019 (accessed March 25, 2020).

[14] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog Computing: Platform and Applications," in *Proc. of IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2015, pp. 73–78.

[15] P. Pop, B. Zarrin, M. Barzegaran, S. Schulte, S. Punnekkat, J. Ruh, and W. Steiner, "The FORA Fog Computing Platform for Industrial IoT," *Information Systems*, vol. 98, p. 101727, 2021.

[16] Q. Perret, P. Maurere, E. Noulard, C. Pagetti, P. Sainrat, and B. Triquet, "Temporal Isolation of Hard Real-Time Applications on Many-Core Processors," in *Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2016, pp. 1–11.

[17] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety," in *Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009, pp. 99–107.

[18] Lui Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[19] D. Tamas-Selicean and P. Pop, "Design optimization of mixed-criticality real-time systems," *ACM Transaction on Embedded Computing*, vol. 14, no. 3, pp. 50–78, May 2015.

[20] M. Barzegaran, A. Cervin, and P. Pop, "Performance Optimization of Control Applications on Fog Computing Platforms Using Scheduling and Isolation," *IEEE Access*, vol. 8, pp. 104 085–104 098, 2020.

[21] The Linux Foundation®, "Xen Project," https://xenproject.org, 2019 (accessed October 7, 2019).

[22] A. Masrur, S. Drossler, T. Pfeuffer, and S. Chakraborty, "VM-Based Real-Time Services for Automotive Control Applications," in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2010, pp. 218–223.

[23] N. Desai and S. Punnekkat, "Safety of Fog-Based Industrial Automation Systems," in *Proc. of the Workshop on Fog Computing and the IoT*, 2019, p. 6–10.

[24] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms," *ACM Computing Surveys*, vol. 52, no. 5, 2019.

[25] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling Fog Computing for Industrial Automation Through Time-Sensitive Networking (TSN)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[26] W. Jiang, P. Pop, and K. Jiang, "Design Optimization for Security- and Safety-Critical Distributed Real-Time Applications," *Microprocessors and Microsystems*, vol. 52, no. C, p. 401–415, 2017.

[27] W. Jiang, H. Hu, J. Zhan, and K. Jiang, "Work-in-Progress: Design of Security-Critical Distributed Real-Time Applications with Fault-Tolerant Constraint," in *Proc. of International Conference on Embedded Software*, 2018, pp. 1–2.

[28] P. Priller, W. Gruber, N. Olberding, and D. Peinsipp, "Towards perfectly scalable real-time systems," in *Proc. of International Conference on Computer Safety, Reliability, and Security*. Springer, 2014, pp. 212–223.

[29] S. K. Memon, K. Nisar, M. H. A. Hijazi, J. J. Rodrigues, A. H. Sodhro, and S. Pirbhulal, "A Review on 802.11 MAC Protocols Industrial Standards, Architecture Elements for Providing QoS Guarantee, Supporting Emergency Traffic, and Security: Future Directions," *Journal of Industrial Information Integration*, p. 100225, 2021.

[30] M. Barzegaran and P. Pop, "Communication Scheduling for Control Performance in TSN-Based Fog Computing Platforms," *IEEE Access*, vol. 9, pp. 50 782–50 797, 2021.

[31] A. Al-Qerem, M. Alauthman, A. Almomani, and B. Gupta, "IoT transaction processing through cooperative concurrency control on fog–cloud computing environment," *Soft Computing*, vol. 24, no. 8, pp. 5695–5711, 2020.

[32] P. Chaudhary, B. B. Gupta, X. Chang, N. Nedjah, and K. T. Chui, "Enhancing big data security through integrating XSS scanner into fog nodes for SMEs gain," *Technological Forecasting and Social Change*, vol. 168, p. 120754, 2021.

[33] N. Reusch, P. Pop, and S. S. Craciunas, "Work-in-progress: Safe and secure configuration synthesis for TSN using constraint programming," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 387–390.

[34] F. Mirsadeghi, M. K. Rafsanjani, and B. B. Gupta, "A trust infrastructure based authentication method for clustered vehicular ad hoc networks," *Peer-to-Peer Networking and Applications*, pp. 1–17, 2020.

[35] A. Ahmadi, A. H. Sodhro, C. Cherifi, V. Cheutet, and Y. Ouzrout, "Evolution of 3C cyber-physical systems architecture for industry 4.0," in *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, 2018, pp. 448–459.

[36] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry*, vol. 81, pp. 11 – 25, 2016.

[37] T. Terzimehic, M. Wenger, A. Zoitl, A. Bayha, K. Becker, T. Müller, and H. Schauerte, "Towards an industry 4.0 compliant control software architecture using IEC 61499 OPC UA," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–4.

[38] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[39] M. S. Shaik, V. Struhár, Z. Bakhshi, V.-L. Dao, N. Desai, A. V. Papadopoulos, T. Nolte, V. Karagiannis, S. Schulte, A. Venito *et al.*, "Enabling fog-based industrial robotics systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 61–68.

[40] M. Barzegaran, N. Desai, J. Qian, K. Tange, B. Zarrin, P. Pop, and J. Kuusela, "Fogification of electric drives: An industrial use case," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 77–84.

[41] P. H. Feiler and D. P. Gluch, *Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language*. Addison-Wesley, 2012.

[42] S. A. Team *et al.*, "An extensible open source AADL tool environment (OSATE)," *Software Engineering Institute*, 2006.

[43] Y. Wang, D. Ma, Y. Zhao, L. Zou, and X. Zhao, "An AADL-based modeling method for ARINC653-based avionics software," in *IEEE 35th Annual Computer Software and Applications Conference*, 2011, pp. 224–229.

[44] A. Hughes and B. Drury, *Electric motors and drives: fundamentals, types and applications*. Newnes, 2019.

[45] TechNavio, "Electric Drives Market by End-users, Product, Power Rating, and Geography - Global Forecast 2019-2023," https://www.technavio.com/report/electric-drives-market-industry-analysis, 2019 (accessed December 1, 2020).

[46] T. J. Williams, "The Purdue enterprise reference architecture," *Computers in industry*, vol. 24, no. 2-3, pp. 141–158, 1994.

[47] J. Berthing and A. Danfoss, "D5. 6 drive controller," *ARTEMIS JU RECOMP Project*, 2012.

[48] J. L. Gutiérrez-Rivas, S. Holmbacka, M. Míndez-Macías, W. Lund, S. Lafond, J. Lilius, and J. Díaz-Alonso, "Safe motor controller in a mixed-critical environment with runtime updating capabilities," *Journal of Universal Computer Science*, vol. 21, no. 2, pp. 177–205, 2015.

[49] J. L. Gutiérrez, J. Berthing, D. Fernández, and J. Dıaz, "Safety-critical platform model based on certification standards," *III Jornadas de Computación Empotrada, JCE*, 2012.

[50] J. Berthing and T. Maier, "Formalised implementation of safety related hw/sw architectures in compliance with functional safety requirements," in *2007 2nd Institution of Engineering and Technology International Conference on System Safety*. IET, 2007, pp. 153–158.

[51] IEEE, "Official Website of the 802.1 Time-Sensitive Networking Task Group," http://www.ieee802.org/1/pages/tsn.html, 2016 (accessed March 5, 2020).

[52] Siemens Simatic, "Profinet system description–system manual," *Issue A5E00298288-04*, vol. 6, 2008.

[53] IEC, "TIEC 61131–3 2nd Ed. Programmable Controllers-Programming Languages," International Electrotechnical Commission, Tech. Rep., 2003.

[54] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer Science & Business Media, 2009.

[55] A. H. Sodhro, N. Zahid, L. Wang, S. Pirbhulal, Y. O. Ouzrout, A. Sekhari, A. V. L. Neto, A. R. L. De Macedo, and V. H. C. De Albuquerque, "Towards ML-based Energy-Efficient Mechanism for 6G Enabled Industrial Network in Box Systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7185–7192, 2020.

[56] C. L. Stergiou, K. E. Psannis, and B. B. Gupta, "IoT-based big data secure management in the fog over a 6G wireless network," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5164–5171, 2020.

[57] IEEE, "Official Website of the 802.1 Audio Video Bridging Task Group," https://www.ieee802.org/tsn/802-1as-rev, 2013 (accessed April 1, 2021).

[58] J. Ruh and W. Steiner, "The need for deterministic virtualization in the Industrial Internet of Things," in *Proc. of the Workshop on Fog Computing and the IoT*, 2019, pp. 26–30.

[59] O. I. Foundation, "Open Source Edge Computing Architecture," https://www.openstack.org, 2012 (accessed April 1, 2021).

[60] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTT-SN) protocol specification," *International business machines (IBM) Corporation version*, vol. 1, p. 2, 2013.

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

[62] K. Sandström, A. Vulgarakis, M. Lindgren, and T. Nolte, "Virtualization technologies in embedded real-time systems," in *IEEE 18th Conference on Emerging Technologies & Factory Automation*, 2013, pp. 1–8.

[63] R. Kaiser and S. Wagner, "The PikeOS concept: History and design," *SysGO AG White Paper. Available: http://www.sysgo.com*, 2007.

[64] ACRN, "Official Website of the Project ACRN™," http://projectacrn.org/, 2019 (accessed December 8, 2019).

[65] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.

[66] A. Cervin, P. Pazzaglia, M. Barzegaran, and R. Mahfouzi, "Using JitterTime to Analyze Transient Performance in Adaptive and Reconfigurable Control Systems," in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation*, 2019, pp. 1025–1032.

[67] C. L. Smith, "Understanding concepts in the defence in depth strategy," in *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003. Proceedings.* IEEE, 2003, pp. 8–16.

[68] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of industrial internet of things security: Requirements and fog computing opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[69] E. Itkin and A. Wool, "A security analysis and revised security extension for the precision time protocol," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 1, pp. 22–34, 2017.

[70] M. Pahlevan, B. Balakrishna, and R. Obermaisser, "Simulation framework for clock synchronization in time sensitive networking," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, 2019, pp. 213–220.

[71] Y. Xu and X. Xie, "Modeling and analysis of security protocols using colored Petri nets." *JCP*, vol. 6, no. 1, pp. 19–27, 2011.

[72] M. Barzegran, A. Cervin, and P. Pop, "Towards Quality-of-Control-Aware Scheduling of Industrial Applications on Fog Computing Platforms," in *Proceedings of the Workshop on Fog Computing and the IoT*. ACM, 2019, pp. 1–5.

[73] M. Barzegaran, B. Zarrin, and P. Pop, "Quality-Of-Control-Aware Scheduling of Communication in TSN-Based Fog Computing Platforms Using Constraint Programming," in *2nd Workshop on Fog Computing and the IoT*, vol. 80. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020, pp. 3:1–3:9.

[74] J. Qian and L. K. Hansen, "What can we learn from gradients?" *arXiv preprint arXiv:2010.15718*, 2020.

[75] Proton Technologies AG., "General Data Protection Regulation," https://gdpr.eu, 2021 (accessed April 1, 2021).