Fogification of electric drives: An industrial use case

Mohammadreza Barzegaran[1], Nitin Desai[2], Jia Qian[1], Koen Tange[1], Bahram Zarrin[1], Paul Pop[1], and Juha Kuusela[3]

[1]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
[2]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
[3]Danfoss Power Electronics A/S, Gråsten, Denmark

Abstract—**Electric drives are used to control electric motors, which are pervasive in industrial applications. In this paper we propose enhancing the electric drives to fulfil the role of fog nodes within a Fog Computing Platform (FCP). Fog Computing is envisioned as a realization of future distributed architectures in Industry 4.0. We identify the system-level requirements of such an FCP, including requirements that are extracted from the current architecture of drives, which we consider as a baseline. These requirements are then used to design a system-level architecture, which we model using the Architecture Analysis & Design Language (AADL). We identify the "technology bricks" (components such as hardware, software, middleware, services, methods and tools) needed to implement the FCP. The proposed fog-based architecture is then used to implement a Conveyor Belt industrial use case. We evaluate the resulting use case on several aspects, demonstrating the usefulness of the proposed fog-based approach. By developing the electric drives as fog nodes, that we call fogification, new offerings like programmability, analytics and connectivity to customer Clouds are expected to increase the added value. Increased flexibility allows drives to assume a larger role in industrial and domestic control systems, instrumenting thus also legacy systems by using drives as the data source.**

I. Introduction

Digitalization will affect all industries and sectors, with a potential cumulative value to society and industry of more than $100tn by 2025 [1]. This paper focuses on the industrial area, where the worldwide cumulative net value will be $1.7tn by 2020 worldwide [2]. We are at the beginning of a new industrial revolution (we will use the term Industry 4.0), which will bring increased productivity and flexibility, mass customization, reduced time-to-market, improved product quality, innovations and new business models [3].

The infrastructure of the information society is underpinned by Information Technologies (IT) such as Cloud Computing, Artificial Intelligence (AI), and Big Data. However, these technologies are not directly applicable to industrial applications [4]. The industrial area uses Operational Technologies (OT) consisting of cyber-physical systems (CPS) that monitor and control physical processes that manage, e.g., automated manufacturing, critical infrastructures, smart buildings and smart cities. These application areas are typically safety-critical and real-time, requiring guaranteed extra-functional properties, such as real-time behavior, reliability, availability, conformance to industry-specific safety standards, and security.

Industry 4.0 will only become a reality through the convergence of OT and IT [5], which are currently separated and use different computation and communication technologies. Currently, the industrial domains use OT, which is costly, completely separated from IT, and cannot support Industry 4.0 [5]. IT such as Cloud Computing has gained significant popularity and we use Cloud-based services on a daily basis as a commodity. However, Cloud Computing cannot provide dependability or quality-of-service guarantees, so it cannot be used for industrial applications. Additionally, technology paradigms such as AI and Big Data are resource-demanding and may compromise the performance of industrial applications under intensive workloads [4]. Thus, they cannot be used for these applications. Instead, a new paradigm called Fog Computing (FC), is needed as an architectural means to realize the IT/OT convergence [6]. 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*" [7].

This paper proposes the use of a Fog Computing Platform (FCP) for the implementation of industrial applications. We focus on the application areas where *electric drives* are present, since they are pervasive in industrial installations and are used in many domains, such as automotive, food and beverage, marine and offshore, hydraulics, refrigeration and air conditioning, etc. Electric drives are used to control the speed, torque and position of electrical motors (see Fig. 1), with real-time software that resides on heterogeneous safety-critical embedded systems controlling the power electronic circuits. Electric drives sit in the "control level" (see Fig. 2) and typically operate on the factory floor, working cooperatively with other devices to automate machinery. Also, data produced by each electric drive is a very critical asset because it carries vital information about the machinery it controls, so the factory owners are reluctant to expose this type of data over the Internet. In addition, data can be massive, often repetitive and not sensitive to delay. Sending it over the control network would eat capability, hence there is a need for each drive to be capable of data analytics locally. Our approach is to use a fog computing architecture (which we call *fogification*) in the implementation of electric drives, and show how such an architecture can be successfully used for the development of an industrial use case.

By developing the electric drives as fog nodes, new offerings like programmability, analytics and connectivity to customer Clouds are expected to increase the added value. Increased flexibility allows drives to assume a larger role in industrial and domestic control systems by benefiting from the ability to instrument as the data source which can help in bootstrapping the data economy. The main direct business benefit comes from the ability to instrument also legacy systems by using drives as the data source. Edge analytics will help in off-loading the network and extend the Internet-of-Things (IoT) solutions 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 will allow anyone to innovate new value-added services and will create long term benefits for all ecosystem participants.

In the remainder of this paper, we take the current architecture of electric drives, tailor an FCP-based architecture

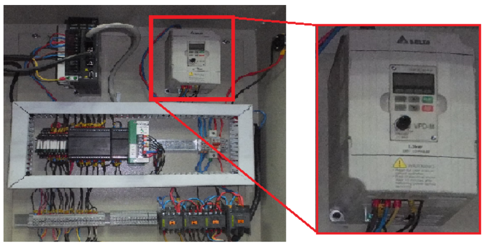

Fig. 1. An electric drive (in red) shown in an industrial setting.

for it and model them in Sect. II. We next identify the challenges related to the fogification of electric drives and collect the requirements that drive both architectures and also the "technology bricks" needed to implement the fogified electric drives in Sect. III. The proposed fog-based architecture is then used to model an industrial use case. We evaluate the resulting use case on several aspects, demonstrating the usefulness of the proposed fog-based platform in Sect. IV. We study the related work in Sect. V and conclude the paper in Sect. VI.

II. Architecture

In this section, we first introduce electric drives and how they work in Sect. II-A. Then, we describe the current architecture of the drives in Sect. II-B and take it as baseline. We have derived a set of high-level requirements (presented in detail in Sect. III-A) that will drive the definition of the fog-based drives architecture in Sect. II-C. We compare the baseline and fogified architectures in Sect. II-D. We use the Architecture Analysis & Design Language (AADL) [8], which has the ability to model large-scale architectures from many aspects in a single analyzable model via its strong syntactic and semantic support for the description of both hardware and software systems. Hence, we use AADL to model the baseline and fogified architectures.

A. *Electric Drives*

An electric motor is an electro-mechanical machine which converts electricity into mechanical rotary movement of its shaft. The mechanical rotary movement of the shaft is generated through the interaction of a magnetic field and an electric current which impacts the movement, i.e. rotation speed, rotation torque, and position of the shaft. Electric drives, alternatively called *drives*, are used to alter characteristics of the electric current such as frequency and voltage to control the motor speed, torque and position [9].

To automate and control machinery and industrial equipments, which comprise the first level of the automation pyramid (see Fig. 2) and sit on the "Machine level", a preliminary control device is used to determine the required output of the actuators: electric motors in our case. The electric drives, as secondary control equipment which are placed close to the actuators and industrial devices on the factory level, are connected to the preliminary control equipment, and control the actuators to generate the required output.

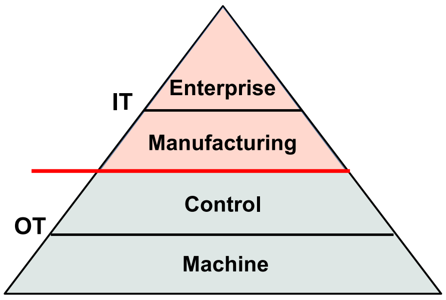

Fig. 2. Automation Pyramid

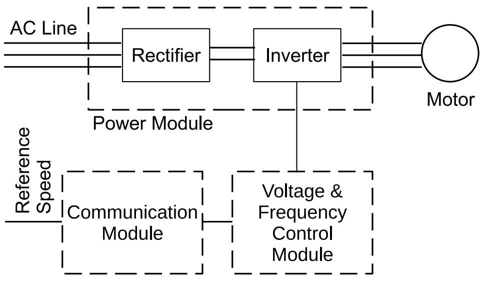

Fig. 3. Internals of an electric drive

The internals of an electric drive have a communication module, a control module, and a power module. The communication module receives the required motor output as a reference from the preliminary control equipment via industrial network. The control module runs a control application to drive the output to the given reference value via the power module which alters the frequency or voltage, based on the type of the drive, to generate the electric current that leads to the desired motor output. A drive is an embedded cyber-physical system that requires real-time response and reliability in order to meet degrees of quality in its output to be able to interact with other devices.

Electric drives are designed for either general purposes, i.e. to control certain power range motors, or specific purposes, i.e. to control a specific electric motor with specific requirements. The drives come in various types, indicated by their power modules, which use different electric current characterizations. Moreover, safety features such as a motor brake are embedded in the control module. Internals of an electric drive are depicted in Fig. 3.

B. *Baseline Architecture*

In this section, we consider a VLT drive from Danfoss Power Electronics [10], and describe its current architecture, which we call the *baseline*. The hardware platform of the baseline has four modules: the communication module, the operation module, the control module and the power module. Each of the first three modules has a dedicated single-core processor and memory unit. The software stack for each module is dedicated. We model the baseline with AADL and show it in Fig. 4.

As depicted in the figure, the communication module has a network switch to connect through the Fieldbus interface with the ProfiNet/RT [11] standard. Its software stack has a real-time operating system which runs a time-triggered application with a limited cycle time that handles the network protocol. We assume that the drive communicates with a Programmable Logic Controller (PLC) as the preliminary controller to get the desired output of the motor, and a Human Machine Interface (HMI) to set the drive parameters such as communication and motor control configurations. The control module runs a

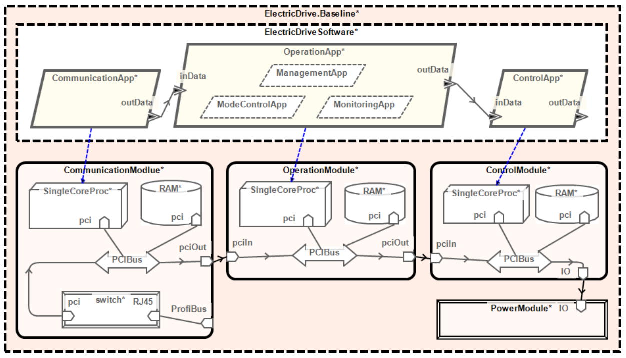

Fig. 4. AADL diagram of the baseline architecture.

feedback control application on the same real-time operating system, implemented according to the IEC61131-3 standard function blocks, once it is engaged. The control module has I/O links with the power module to get the feedback and set the control signal.

The software stack of the operation module has the same real-time operating system and runs applications which have priorities commensurate with their execution rates. The applications are: a mode application which engages and disengages the motor controller, a monitoring application which pre-analyzes the drive data, and a management application which configures the communication and controller parameters. The monitoring application collects data such as voltage and temperature and does local machine learning to predict the drive maintenance. The operation module shares separate dedicated buses with the communication module and the control module for data exchange.

C. Fogified Architecture

We propose a fogified architecture for drives, based on extending the initial designs proposed in the Fog Computing for Robotics and Industrial Automation (FORA) European Training Network [12]. The proposed fogified drives architecture can be used both within the traditional hierarchical model of industrial automation (Fig 2) and in future distributed cyber-physical systems architectures (Fig. 5) that are envisioned to be used in Industry 4.0. In such a distributed architecture, the integration of computational and storage resources into the communication devices is realized in the fog node. In many applications, including industrial automation and robotics, several layers of fog nodes with differing computation, communication and storage capabilities will evolve, from powerful high-end fog nodes to low-end fog nodes with limited resources. Researchers have started to propose solutions for the implementation of fog nodes [13] and fog node solutions have started to be developed by companies [13]. Fog nodes could be connected to each other and to the machines through a deterministic communication solution, such as IEEE 802.1 Time-Sensitive Networking (TSN) [14], see Fig. 5. In TSN, time sensitive traffic is transmitted using schedule tables called "Gate Control Lists" (GCLs). Such an FCP-based architecture allows to increase the spatial distance between the physical process and the fog nodes that controls it, allowing the control functions to be executed remotely on the fog nodes. Several initiatives are currently working towards realizing this vision [12], [13].

We model the fogified architecture with AADL and show its schematic architecture in Fig. 6. The model consists of a hardware platform which has a dual-core processor, a switch

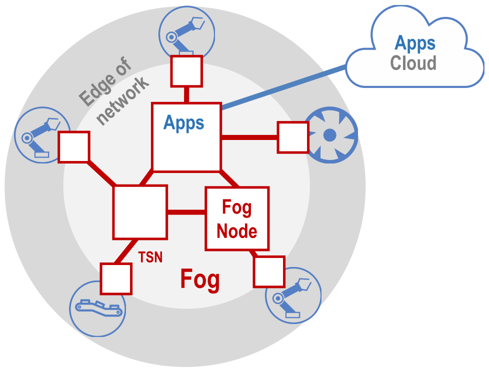

Fig. 5. Fog Computing platform. Boxes represent fog nodes, connected with each other and to the Cloud; the thick lines are the network. Applications (Apps) run on the fog and Cloud.

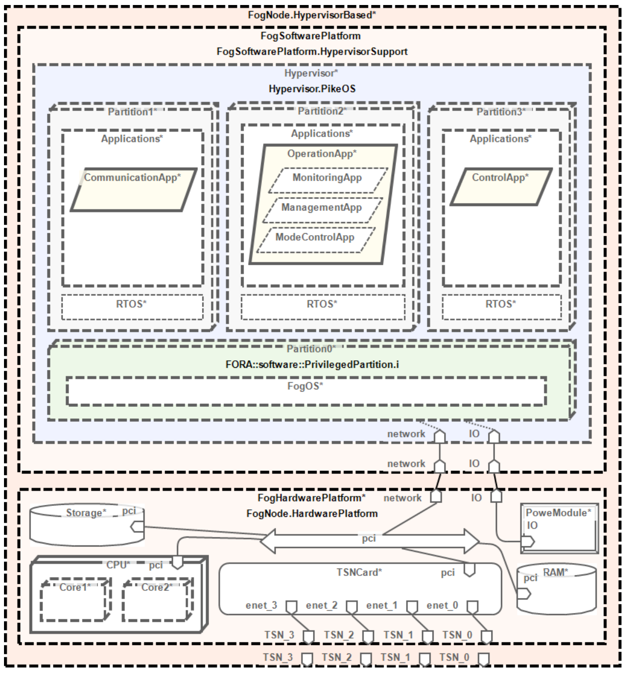

Fig. 6. AADL diagram of the fogified architecture

and a power module, and a software stack which has a hypervisor, a middleware, an OS and an application layer.

The mixed-criticality applications running on the dual-core processor are using temporal isolation enforced by the hypervisor, to prevent them from interference. The hardware platform has a TSN-enabled network switch and a power module which generates electric current according to the given control signal to drive motors, all connected through a shared bus. As depicted in the figure, the module has I/O with the processor which enables getting feedback and setting control signals.

We assume using PikeOS [15] as the hypervisor which implements temporal partitioning to isolate the applications. In our model, we have three partitions, indicated by the applications assigned to them: control, communication and operation. The hypervisor schedules the execution of partitions. We also consider using a middleware on top of the hypervisor to enable data exchange between the partitions and provide features like runtime updates.

The control partition has a soft-PLC OS, and the control application is implemented using the IEC61131-3 standard function blocks. The control application is configurable via the middleware. The communication partition has a real-time OS which runs applications for controlling the network traffic, applying security mechanisms, handling the applications traffic, and setting the message schedule tables (GCLs), all via the middleware. We assume that the operation partition has an OS to run different type of applications including a machine learning application for predictive maintenance (same as the predictive maintenance application in the baseline).

D. Comparison

The most significant difference between the baseline and the fogified architecture, is the change in isolation mechanism from spatial to temporal. The multicore processor and the shared resources such as bus and memory increase the unpredictable delays and overheads in the execution of tasks, as well as the data exchange of applications. Although the fogified architecture brings more interference and unpredictability, it provides a programmable platform for monitoring purposes.

In the fogified architecture, we proposed using a hypervisor to enforce temporal isolation with the added cost of over-

TABLE I
SYSTEM LEVEL REQUIREMENTS

#	Requirement	Realization in the baseline architecture	Realization in the fogified architecture
1	Drives shall be designed according to the industrial standards	IEC61800-based design	IEC61800-based design
2	Drives shall have time–constraint interface	1 ms time–constraint ProfiNet interface	Jitter-free TSN interface
3	Drives shall be able to monitor and process data for predictive maintenance purpose	Machine Learning framework with appropriate Safety Integrity Level	Machine Learning framework with appropriate Safety Integrity Level
4	Drives shall run mixed-criticality applications according to the industrial standards	Spatial separation according to IEC61508	Temporal separation according to IEC61508
5	Drives shall control the electric motor accurately	Motor control with response time of 30 ms and good quality-of-control	Motor control with response time of 20 ms and good quality-of-control
6	Drives shall be configurable according to the industrial standards	Configurable according to IEC61508	Configurable according to IEC61131
7	Drives shall have secure access to the Cloud	Cloud connection provided by external devices	Cloud connection provided by TSN interface with security mechanisms

head to the computation. The best effort partitions provide a programmable platform to collect data from different sensors, perform sensor fusion already in the drive as edge node, run simple machine learning algorithms in the device, and finally stream data over the interface.

The other significant difference between the two architectures lies in the use of TSN instead of ProfiNet/RT which helps in Cloud connectivity and IT/OT convergence. We also consider applying selected security mechanisms to protect against possible cyber-attacks. We see that the technology bottlenecks are in the hypervisor and TSN where applications and traffic should be scheduled, isolated and protected concerning industrial grade standards.

III. CHALLENGES

In this section, we identify the system level requirements that drive the baseline and fogified architectures in Sect. III-A. We discuss the challenges inherent in the design and development of fogified drives and propose the necessary technology bricks in Sect. III-B.

A. Requirements

We identify the system level requirements and the relevant realization in Tab. I, where the requirements are shown in column 2, the baseline realization in column 3 and the fogified realization in column 4. Unlike special-purpose drives which have to satisfy certain requirements that are needed for their specific purpose, the general-purpose drives which are considered in this paper have to satisfy more generic requirements that aim to make the drive compatible with a wide-range of electric motors.

For each of the requirements, we sort the relevant architecture components by the order of flexibility, take the least-flexible one as a pivot point, apply the requirement on the component to achieve the best possible performance, refine the requirement concerning the achieved performance, and apply the refined requirement to the next component. The system level requirements are qualitative at the beginning, and become quantitative after several iterations of applying them to the components, since each requirement aims to achieve the best performance. In industry, the strongest requirement is generally the financial aspect which mostly covers the costs, and prevents achieving the best performance.

Requirement 1 implies that the drive design should comply with the industrial standards, which is met by both architectures in the same way via IEC61800 that states control strategies and performance requirements of the motor controller and converter in different operation conditions. Industrial time constraint communication is considered in requirement 2, which is realized by ProfiNet in the baseline, and TSN in the fogified architecture. We consider a machine learning approach for drive maintenance prediction as a service in requirement 3. The service is implemented as an application with lower criticality in both architectures and addressed with different isolation approaches with respect to IEC61508, as imposed by requirement 4. Requirement 5 imposes accuracy (control performance) constraints on the control application along with performance constraints (imposed by requirement 1) and integrity constraints (imposed by requirement 4 where the control application has the highest priority and highest Safety Integrity Level). The configurability of the applications for performance and operation is addressed in requirement 6, which needs communication between applications that is realized in different ways in the baseline and fogified architectures. Secure access to the Cloud (imposed by requirement 7) is applied by using external equipment such as a gateway that has cloud connection in the baseline architecture, and by using security mechanisms on the TSN interface in the fogified architecture.

B. Technology Bricks

The fogified architecture proposed in Sect. II-C has been driven by and meets the requirements from Sect. III-A. To implement such an architecture, we identify in this section the needed "technology bricks", which can be methods, models, hardware, software, tools, mechanisms, etc. We have identified the following technology bricks:

The **FCP Configuration** provides a configuration for temporal separation and scheduling of mixed-criticality tasks. The configuration also considers the extra functional requirements of the tasks such as the quality-of-control (QoC) for control applications (see [16] for more details). The configuration consists of partition tables, the task schedule tables inside each partition and GCLs of TSN switches; which addresses requirements 2, 4, 5 and 6. A **Machine Learning Framework** brings the capability to predict when the drive maintenance is needed, by accessing the drive data via middleware. We propose a decentralized framework concerning the TSN capability of the fogified architecture which leads to more accurate prediction. We also propose a fault detection, isolation, and recovery (**FDIR**) method to be applied on TSN communication and tasks execution. The method is implemented on the middleware and monitors tasks execution and network communication traffic. It applies detection and identification techniques, and provides recovery mechanisms in case of faults. Furthermore, we propose to deploy various **Security Mechanisms** which protect TSN from cyber-attacks and unauthorized access to the node. The mechanisms are implemented on the hypervisor where low-level access to hardware is possible.

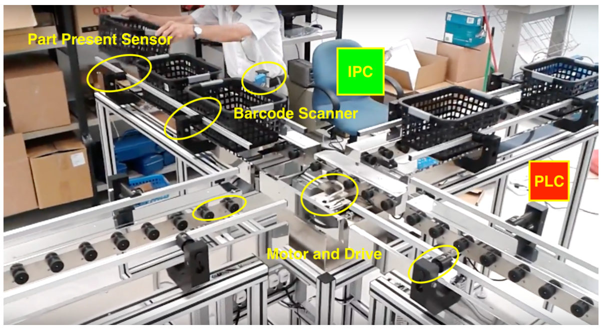

Fig. 7. Conveyor Belt Use Case

IV. Evaluation

We have used the proposed fogified architecture to model a **Conveyor Belt** Use Case (UC): a conveyor belt is used to distribute packages from an inventory to different destinations based on the package. The conveyor belt is well-known and widely used in inventories for the automatic distribution of packages, and is realized using electric motors and drives. In the UC we consider a typical machine as depicted in Fig. 7. The machine is fed with packages from one side and reads the tag of the received package. It gets the destination of the package by accessing a database with the read tag; and drives the package towards the destination from one of the other sides of the machine. The UC has been realized using the proposed fogified architecture from Sect. II-C and the technology bricks from Sect. III-B. In this section we evaluate the resulting implementation on several aspects to address the requirements (showing the suitability of the implementation for industrial applications) and exhibit the new offerings such as programmability, analytics and connectivity to customer Clouds (showing the added value of the implementation for industrial applications) as shown in the subsections.

A. Network configuration for QoC

Since the fogified architecture shares the same communication medium for hard and soft real-time, non-critical and best effort communication, we propose the scheduling of the traffic on TSN to guarantee timing requirements of the streams. We assume that the communication between drives and other industrial equipment is achieved via TSN and also the control applications are a set of streams which have hard real-time requirements. All messages are scheduled using the schedule-based time-sensitive traffic type in TSN, which, as mentioned, uses schedule tables in the switches (GCLs) to schedule the transmission of messages. The streams are prioritized in accordance with their criticality and scheduled with respect to their requirements. Recent works that address scheduling of TSN traffic can be found in [17].

We employ the constraint programming-based schedule synthesis strategy aiming at maximizing the QoC and satisfying the deadlines of real-time messages, proposed in [18] to schedule the traffic.

TABLE II
Streams in the Conveyor Belt Use Case

#	Size (bytes)	Period (μs)	Routing	Max. delay (μs)
1	500	3000	$C_1 \to N_4 \to A_1$	4
2	100	1000	$C_2 \to N_5 \to N_2 \to N_4 \to A_1$	8
3	150	3000	$C_1 \to N_4 \to N_2 \to A_2$	4
4	250	4000	$C_2 \to N_5 \to A_3$	2
5	1200	10000	$A_2 \to N_2 \to N_5 \to C_2$	57
6	300	4000	$A_1 \to N_8 \to N_6 \to C_1$	3
7	400	3000	$S_1 \to N_1 \to N_3 \to C_1$	20
8	400	6000	$S_2 \to N_1 \to N_3 \to C_1$	20
9	1500	15000	$S_3 \to N_2 \to N_5 \to C_2$	59

TABLE III
Tasks in the Conveyor Belt Use Case

Applications	Tasks	WCET (μs)	P (ms)
γ_1	τ_1	500	10
	τ_2	2500	10
	τ_3	2500	10
γ_2	τ_4	1500	12
	τ_5	3000	12
γ_3	τ_6	15000	50
γ_4	τ_7	2000	20
	τ_8	3000	20
	τ_9	1500	20
	τ_{10}	1500	20

We have evaluated our proposed network configuration strategy on our Conveyor Belt UC. In the given example, the five switches (denoted with N) connect three sensors (denoted with S) to two controllers (denoted with C), and transmit the messages from controllers to four actuators (denoted with A). The details of the streams are shown in Tab. II where the stream name, size, period and routing are shown in columns 1 to 4 respectively. We assume that one of the controllers runs a motor control application for speed control of electric motors. The motor speed controller running on the node C_1 receives message 7 which is the sensor data S_1, and sends message 1 to our proposed fogified drive A_1, which controls an electric motor. We assume that all links have 1 Gbps bandwidth.

The proposed strategy has successfully scheduled all streams, i.e., none of the deadlines are missed, and minimized delay and jitter of streams, resulting in an optimized control performance. The results show that all streams have zero jitter, which improves control. The column 5 in Tab. II shows the maximum delay of streams. We used JitterTime [19] to simulate the behavior of the control application which reports a value of 0.008 for the QoC (see [18] for the exact cost function), i.e., a good control performance.

B. Configuration of hypervisor partitions and task schedules

Since the real-time applications are virtualized and implemented as tasks, the FCP configuration (e.g., task scheduling) has impact on the performance of control applications. We assume the use of deterministic hypervisors for virtualization of applications on the fog node similar to [20], where hypervisors provide a deterministic access to shared resources via a static configuration table and provide spatial and temporal isolation of mixed-criticality applications via "partitioning". We propose a metaheuristic solution to optimize the hypervisor partition tables, map the tasks to the processing cores of the multi-core processors of fog nodes, assign the tasks to partitions and schedule the tasks inside the partition tables.

Our proposed solution provides temporal separation of tasks similar to [21], [22] and assignment of the tasks to the cores, and scheduling of the tasks inside the partition slices, similar to the optimization strategy presented in [16] where the static scheduling of tasks considers the QoC of control applications. We have evaluated our solution, while ignoring the temporal isolation of tasks, on the UC in which four applications, including a control application denoted with γ_1, are running on a fog node that has two cores. Each application has number of tasks and each task has a worst-case execution time (WCET) and a period (P). The control application is the drive's controller for controlling electric motors. The details of the applications are shown in Tab. III.

Our proposed optimization strategy has successfully scheduled all the tasks and decided the task mapping to the cores. The results show that none of the tasks has missed its deadline. Furthermore, the control application has a good control performance which is evaluated with JitterTime [19] that calculates a value of 0.011 for the QoC (cf. the cost function from [16]).

C. Addressing security mechanisms in TSN

In order to adequately protect the system against adversaries, security mechanisms are required. A compromised system may lead to safety requirements being violated, meaning that security services must run with at least the same priority as critical tasks. We briefly discuss security solutions that are enabled by fogification of the drive in our UC. These should all be deployed in parallel, as an instance of a defense-in-depth approach. The various mechanisms proposed here are summarized in Table IV.

Firstly, the drive will communicate sensitive data over the Internet, such as usage statistics for predictive maintenance. To ensure confidentiality and integrity of the data, as well as authenticity of the remote party, secure communication standards such as TLS should be used to provide confidential authenticated communication channels. The FCP should block any attempt at communication to endpoints it cannot authenticate. To further limit the attack surface, a firewall should be active on the hypervisor level, ideally making use of the predictable nature of machine-to-machine communication by using whitelists for known addresses and services. These measures protect against attacks known as the Man-in-the-Middle (MitM) attacks, where an adversary sits between the legitimate sender and receiver, capable of snooping on and/or modifying data in transit.

Services that communicate with the Internet (including Cloud service) form a major attack vector of Internet-connected devices, and should be placed in separate partitions by the FCP, isolating them from other services. This makes it more difficult for attackers to pivot to other parts of the system, should they break into an Internet-facing service, thus limiting the impact of an intrusion.

Additionally, a security monitoring service should run in a separate highly-privileged partition, capable of detecting anomalous behavior in the system, while also improving forensic possibilities. These are important factors in a fast detection of system intrusion, impact analysis, and attacker attribution.

Because of the time-critical nature of TSN, the protocol itself provides only minimal security, it is necessary to isolate this as much as possible from the rest of the system. In this UC, the architecture is perfectly positioned to isolate all TSN traffic from the rest of the physical network using Software Defined Networks (SDN), or similar techniques. For further protection within TSN, per-stream filtering as described in the IEEE 802.10qw [14] standard can be applied as a light-weight monitoring technique. In order to mitigate Denial of Service (DoS) attacks as much as possible, careful consideration to the network topology should be given during the design phase, so that if a single link in the network were to fail, traffic can be routed over different paths.

Finally, within industrial networks, physical access to machines is a relatively common attack vector, therefore, configuration changes of the electric drive should not be possible without some form of authentication of the operator, such as a secure hardware element.

D. Distributed predictive maintenance in the fog

Here we propose a distributed Machine Learning framework where the distributed drives and the centralized server jointly (collaboratively) train one global model. Typically, the distributed drives placed in different locations, generate data that captures the local information instead of global information and they train their local models based on the partial knowledge. The aggregation step at the server-side enables the information sharing between drives and server to obtain one model with overall knowledge. Consecutively, the server sends the aggregated model back to drives. The whole procedure may iterate several times. In a nutshell, as depicted in Fig. 8, it is divided into four steps: (i) local model training, (ii) model (or gradient) transmission, (iii) aggregation and (iv) sending aggregated model back to decentralized devices.

TABLE IV
THREATS AND THEIR MITIGATIONS

Threat	Mitigations
MitM, 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 TSN protocol, per-stream filtering
Physical attacks	Hardware token for configuration changes
Detection	Security monitoring service

We applied this method combing with active learning [23] in the work [24], which experimentally proves its effectiveness. This collaborative learning scheme mainly has two virtues. Firstly, it may save the cost of bandwidth by avoiding the transmission of the massive training dataset (transmit model parameter or gradient instead). Secondly, it may preserve user privacy by keeping data in the generation place, and at the same time train a model that has comprehensive knowledge. Yet, recently some researchers argue the gradient may breach the privacy by reverse engineering work, but it can be defended by plugging noises to the gradient before the transmission, e.g., noises generated from Laplace distribution. As we mentioned before, data produced by electric drives is a very critical asset because it carries precious information about the machinery it controls, and we believe it can be addressed by the proposed collaborative model.

We experimented on a public simulated engine run-to-failure events dataset [25] to demonstrate our method, since a public drive failure dataset was not available. We assume four edge devices and one fog node in the experiment. The dataset is composed of 24 features and the binary labels where zero represents failure within one preset period (30 days), one otherwise. For the sake of a more elaborated result, we can chunk time-to-failure into more periods to convert it to a multiple-classification problem. For instance, label zero indicates time-to-failure less than two weeks, label one indicates the period between 2 weeks and one month, etc. We employ Logistic Regression [26] as the model to carry out the binary classification task. The one-shot binary classifier result is depicted in Fig. 9, where the accuracy of the devices and the aggregated model is shown. The aggregation step improves the overall performance and the accuracy of the devices. Note

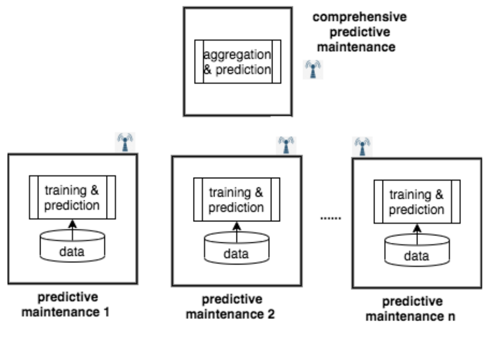

Fig. 8. Distributed ML Diagram

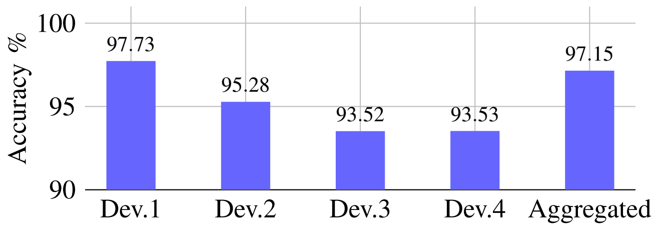

Fig. 9. One-shot Predictive Maintenance Performance

that here we only demonstrate the one-shot result, but it can be repeated multiple times according to the requirement.

E. Fault detection, identification and recovery for the UC

In this section, first, we list the safety requirements of the IEC61508 standard, and then we propose a way to provide safety assurance for the safety functions in the proposed UC. The primary safety requirement for electric drives, whether fogified or not, is to shutdown and stop the motor in case of emergency situations such as voltage/current surges, which can result in uncontrolled motor speeds, short-circuits and possible danger to human life. Additionally, we present below a set of safety requirements that needs to be included in the design of the fogified electric drives. The identification of failure modes for fogified electric drives shall be documented at design time. A safe timer is a common practice since functions related to drives require safe timers. The safety of the drive is directly relying on the fact that the timer is correct. Deadlock prevention among safety and non-safety functions in the drives is one of the critical requirements since a deadlock brings non-deterministic behaviors and is the major cause of deadline misses for safety tasks. All emergency and process shutdown functions (ESD, PSD) need to be executed regardless of concurrent processes running in the FCP. Their criticality must always be the highest. A failure event should have a persistence parameter which specifies the duration of the failure detection before a failure is declared. The parameter should specify either a time duration or a repeated detection threshold. The persistence time may be zero so that any detection is immediately treated as a failure. Adequate redundancy measures for safety functions are a standard practice as well as temporal and spatial isolation of safety and non-safety functions. The system shall have a defined behavior on detection of a fault or a failure event. This may be either a safe state or a well-defined consequence or behavior. Last but not least, any run-time changes (such as over-the-air firmware updates) in the fogified drives should only be done to the non-critical parts or should undergo a validation for safety.

To provide safety function guarantees, we have turned the safety instructions into drive operational states by introducing a marginal behavior. Therefore, the values of the operational state of the drive are decided based on the satisfactory behavior as a tool. We propose to deploy the tool on the middleware of the architecture, which has access to all the partitions that have applications with different criticality assigned to them.

We take the Conveyor Belt UC as an example to define

TABLE V
OPERATIONAL STATES AND SAFETY ACTIONS

Operational state	**Safety action**
Switch failure	Re-route stream on alternate switch
Part presence sensor failure	Stop conveyor belt if item is fragile
Emergency brake failure	Trigger emergency shutdown function
Motor over-heating	Inject coolant
Controller malfunction	Switch to safety controller (redundant)

several operational states for the drive and also proportional safety actions to take. An unsafe safety state is one wherein the safety state variables breach their threshold values. The objective is that we have no overlapping conditions and the drive always falls into one of the conditions which has a safety action to take. For example, as shown in Tab. V, a safety critical operational state such as an emergency brake failure will trigger a safety action like an emergency shutdown that is executed in a separate system partition.

V. Related work

Several research projects have addressed mixed-criticality applications on that share multicore-based distributed architectures. The EMC2 European project[1], aims to provide efficient handling of mixed criticality applications under real-time conditions, scalability and utmost flexibility, full scale deployment and management of integrated tool chains, through the entire life cycle. Research on Fog computing platform architectures has made progress in recent years [13], [27]. For example, the European projects FORA[2] and mF2C[3] 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 and Nebbiolo Technologies, Inc. are pioneers in the field of commercializing the Fog computing paradigm with market ready products for industrial automation. While design paradigms for the fog are still in their early stages, there are certain generic guidelines that are followed to ensure isolation of tasks of varying criticality. In [28] the authors describe an execution framework wherein 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 design such as the simplex architecture [29], [30] 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 non-optimal due to the switching latencies. Selicean et al. [21] propose 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 be realized through separation and scheduling the control tasks inside the partitions similar to [22]. The modification of hypervisors provides different degrees of separation. Modification of the Xen hypervisor to guarantee timing constraints are proposed by Masrur et al. [31]. The authors modify the hypervisor with a new scheduler based on a fixed-priority policy and a control loop to control timing constraints of virtual machines. [32] addresses safety critical applications running in the Fog and how the FCP must cater to these specific requirements.

One of the major research themes is resource management in the Fog. In [33], the authors identify and classify the architectures, infrastructure, and underlying algorithms for managing resources in fog/edge computing. [34] proposes a list scheduling-based heuristic to solve this problem. The authors demonstrate the feasibility of reconfiguring the scheduled network at runtime for industrial applications within the fog. [35] introduces a vulnerability-based method to quantify

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

the security performance of communications on distributed systems. Fault tolerant aspects are discussed in [36] 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 to find the best solutions is then proposed. [37] discuss potentially contradicting design constraints: real-time capability versus scalability. This paper suggests a design methodology and architecture as a step towards perfectly scalable real-time systems, i.e. systems with deterministic timing behavior and run-time reconfiguration.

VI. Conclusions

In this paper, we have addressed a novel fog-based architecture that is a key enabling technology for Industry 4.0. We have proposed to re-engineer electric drives and turn them into fog nodes. We take the current drives architecture as baseline, apply our proposed fog computing architecture to it, and compare the two architectures. The proposed architecture is driven by the stringent safety and performance requirements of industrial applications. In addition, we have identified fog-specific requirements and challenges.

We have modeled our proposed architecture with AADL and studied the interaction of the components, identified the needed "technology bricks" and bottlenecks, and mapped the proposed architecture to a computing platform for the realization of an industrial use case, a Conveyor Belt application. We have evaluated the use case in relation to the proposed technology bricks. As the evaluation shows, a fog-based implementation of industrial applications is a promising approach to realize the vision of Industry 4.0.

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.

References

[1] World Economic Forum, "Digital Transformation of Industries," http://reports.weforum.org/digital-transformation/wp-content/blogs.dir/94/mp/files/pages/files/wef-digital-transformation-2016-exec-summary.pdf, 2016 (accessed March 15, 2020).

[2] D. Floyer, "Defining and sizing the industrial internet," http://wikibon.org/wiki/v/Defining_and_Sizing_the_Industrial_Internet, 2013 (accessed March 15, 2020).

[3] 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.

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

[6] 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.

[7] OpenFog Consortium, "OpenFog reference architecture for fog computing," https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, 2017 (accessed January 5, 2020).

[8] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, Tech. Rep. CMU/SEI-2006-TN-011, 2006.

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

[10] Danfoss, "Danfoss Electric Drives," https://www.danfoss.com/en/products/ac-drives/?sort=default_sort, 2020 (accessed March 15, 2020).

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

[12] 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).

[13] 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.

[14] 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).

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

[16] 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.

[17] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks," in *Proc. of the International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[18] 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.

[19] 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*. IEEE, 2019, pp. 1025–1032.

[20] 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*. ACM, 2019, pp. 26–30.

[21] 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.

[22] 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.

[23] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proc. of the International Conference on Machine Learning*. JMLR. org, 2017, pp. 1183–1192.

[24] J. Qian, S. Sengupta, and L. K. Hansen, "Active learning solution on distributed edge computing," *arXiv preprint arXiv:1906.10718*, 2019.

[25] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set. NASA Ames Prognostics Data repository, NASA Ames Research Center, Moffett Field," 2008.

[26] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.