Enabling Fog-based Industrial Robotics Systems

Mohammed Salman Shaik
Václav Struhár, Zeinab Bakhshi
Van-Lan Dao, Nitin Desai
Alessandro V. Papadopoulos
Thomas Nolte
Mälardalen University, Sweden
{name.surname}@mdh.se

Vasileios Karagiannis
Stefan Schulte
TU Wien, Austria
{v.karagiannis, s.schulte}@dsg.tuwien.ac.at

Alexandre Venito
Gerhard Fohler
TU Kaiserslautern, Germany
{venito, fohler}@eit.uni-kl.de

***Abstract*—Low latency and on demand resource availability enable fog computing to host industrial applications in a cloud like manner. One industrial domain which stands to benefit from the advantages of fog computing is robotics. However, the challenges in developing and implementing a fog-based robotic system are manifold. To illustrate this, in this paper we discuss a system involving robots and robot cells at a factory level, and then highlight the main building blocks necessary for achieving such functionality in a fog-based system. Further, we elaborate on the challenges in implementing such an architecture, with emphasis on resource virtualization, memory interference management, real-time communication and the system scalability, dependability and safety. We then discuss the challenges from a system perspective where all these aspects are interrelated.**

I. Introduction

Industrial robots are widely used in different automation applications such as painting and welding in automotive facilities and packaging in the food industry [1]. More recently, the domain of robotics has evolved to support warehouse automation with mobile robots and, at the same time, emphasis on collaborative robots has also gained significant attention [2]. However, existing solutions are limited in addressing the demands of such applications due to limited computational resources and the strong coupling of the software and the computing hardware [3]. The on-demand availability of resources and reduced communication latency as offered by the fog computing paradigm [4] makes an interesting case for investigating the use of fog computing for addressing existing limitations. For example, the localization and mapping tasks of mobile robots which may be computationally intensive, have been successfully implemented using fog computing resources, improving the computation time significantly [5]. Furthermore, an optimized offloading algorithm which targets fog computing resources has been designed for robotic mission planning to show the benefits of fog computing [6]. While these approaches show the benefit of using fog computing in industrial robots, practical implementation of fog-based robotics systems remains a challenge.

To facilitate further discussions on fog-based industrial robotics systems, and to identify the challenges that should be addressed to enable fog-based control of robots, in this paper, we provide an overview of the different technical aspects that are necessary for a practical realization of a fog network based on the OpenFog reference architecture [7] in Section II. In Section III, we describe a robotic cell-based factory automation environment and a robot control application. In Section IV we use the factory automation environment to contextualize the technical aspects of the fog system and identify the potential challenges that should be addressed to enable a fog-based industrial robotics system. Here, we focus primarily on virtualization, resource orchestration, multi-core memory management and real-time communication along with a discussion on challenges from the dependability, safety and scalability perspectives. Finally, Section V concludes the paper.

II. System architecture

While several fog computing architectures have been proposed in the literature, [7], [8], [9], for our discussion, we adopt the IEEE OpenFog reference architecture and deployment model for fog computing [7]. Here, we assume that the participating devices are distributed between three layers (see Fig. 1) consisting of (i) cloud layer, (ii) fog layer, and (iii) device layer. The cloud layer provides a high computing capacity, but offers limited time predictability due to varying data transmission latencies. The fog layer provides an elastic environment in the vicinity of the origin of the data. It consists of a number of interconnected physical hardware devices (fog nodes) that are capable of hosting software applications on the shared node resources. While the processing power of the fog layer is less than that of the cloud layer, the network latency, however, is improved. The device layer consists of resource limited devices such as sensor and actuator devices that typically pre-process data from sensors and transmit it to the fog nodes, but also smart sensors and actuators, that are capable of communicating directly with the fog nodes.

Fog Software Components and Services:

We assume a fog computing architecture to be composed of a network of fog nodes, which can be viewed as a single logical entity [10]. The network is assumed to be hybrid of wired and wireless networks to exploit the benefits of both advanced wired and wireless technologies under the practical constraints of reliability, timeliness, and security for industrial environments. We briefly discuss some of the

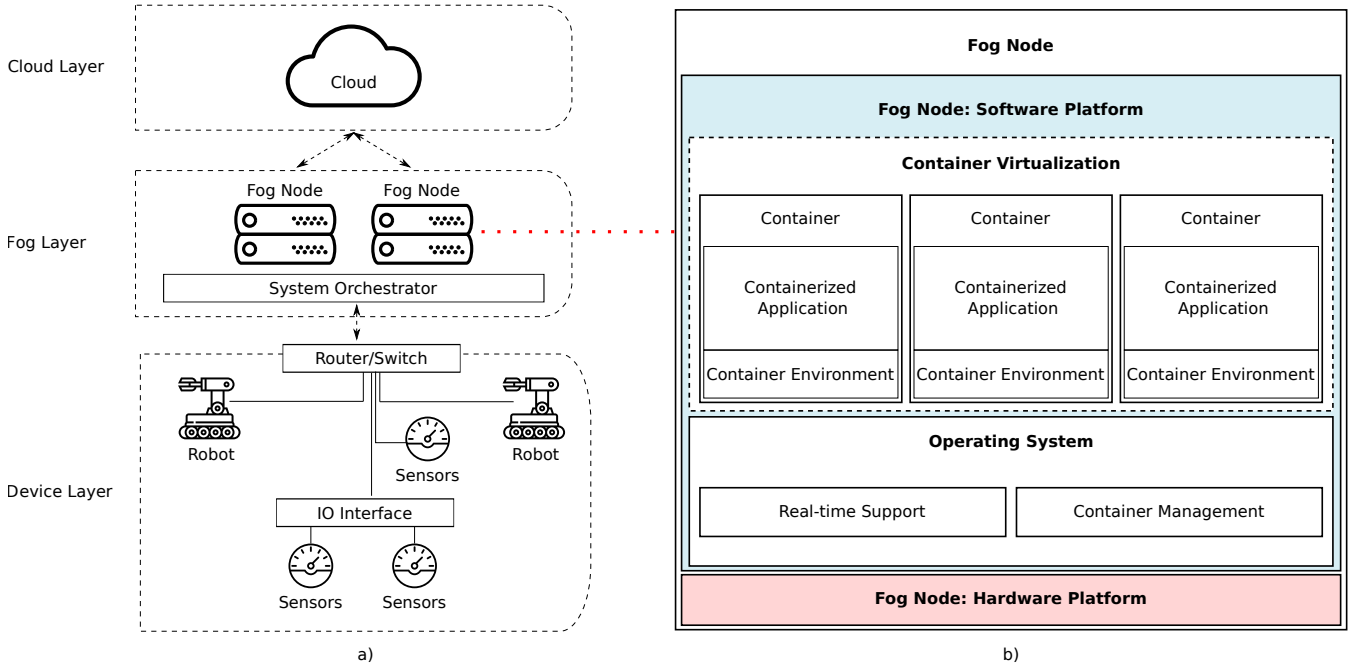

Fig. 1. a) The three-layered fog network with the device layer consisting mainly of sensors and actuators and a fog layer with a real-time orchestrator managing the distribution of workload between the fog nodes. b) An abstract view of a fog node. The fog hardware platform is supported by the software platform composed of container virtualization on top of an operating system.

software components and services necessary for such a fog architecture below:

- System Orchestration: The system level orchestrator is responsible for ensuring that application requirements such as latency and memory are met by assignment of the applications to fog nodes. It takes into account the hardware capabilities, applications already running, task schedulability and application latency requirements. Additionally, it provides interfaces that enable seamless connection and disconnection of devices (e.g., additional fog nodes, robots, sensors and actuators) as well as interfaces for application providers for deploying applications in the fog system. Moreover, it continuously monitors the status of fog nodes in terms of availability, resource usage, and communication status and assesses the quality of service provided to the applications.
- Application Virtualization: The application virtualization component provides necessary functionality that allows to co-locate multiple independent applications on a single physical device in such a way that interference between the applications is minimized. It ensures proper allocation of resources and isolation between applications on the shared hardware building virtualized environments.
- Memory Management: Memory management component is responsible for ensuring spatial isolation among the tasks running on the same hardware. Applications have bounded memory space and cannot exceed these limits. Shared memory is allowed as long as it is explicitly declared by the applications and allowed by the operating system.
- Real-time Communication: The communication component is responsible for ensuring connectivity between the nodes and the sensor and actuator interfaces to ensure real-time data freshness, correlation and separation constraints of the applications [11]. It is also responsible for non real-time communication and supports both wired and wireless communication.
- Scalability: Services related to scalability are responsible for adding new physical and virtual compute nodes (as well as sensing and actuating devices) to the system in a dynamic manner. Due to such scalability services, the system level orchestrator is agnostic of discovery and integration mechanisms, but is able to consider all the available computational resources for the execution of the applications.
- Dependability and Safety: The dependability services provide a set of functionality for ensuring system dependability including the safety aspects by providing means for fault prevention, fault tolerance, fault removal, and fault forecasting [12] .

III. Factory Automation Environment

A common factory automation setup is composed of a set of robotic cells [13]. A robotic cell consists of either a single robot or a set of robots grouped together with additional non-robotic machines to accomplish a task such as painting and welding. We categorize multi-robot cells as (i) coordinated cell, (ii) uncoordinated cell and (iii) mixed cell. In a coordinated cell, all the robots work in a synchronized manner on a single object. In an uncoordinated cell, the robot may work on

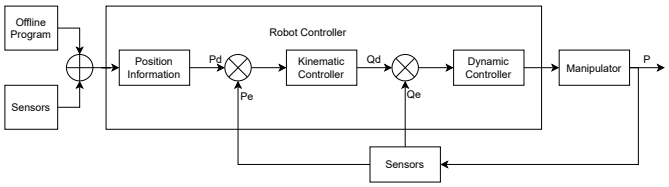

Fig. 2. Robot controller block diagram

different objects and need not be synchronized. For example, a pick and place cell with two robots operating on different objects on two different conveyors need not be in sync with each other. In many cases, a supervisory controller may control the workflow between the robots and other machines within the cell.

Each robot within a cell may be fitted with additional sensors and actuators, such as seam tracking lasers, force sensors, or vision systems, while actuators are typically end-effector tools, such as an arc torch [14]. The sensors and actuators are physically connected to an interface device, which processes the sensor data for transmission over a real-time network. It is possible that some of the sensors and actuators can be directly connected to the robot controller through a wired or a wireless connection. In some cases, sensors and actuators are still physically attached to the interface device but the interface device itself can communicate wirelessly with the controller. Fig. 1 illustrates the different possible connections within a fog network.

At the robot level, the runtime robot behaviour is directed through a task specification interface, where a user typically specifies the way-points that the robot should pass through, the maximum speed the robot is allowed to take, along with other attributes such as, if the robot should pass through the way-points or just within a range of the way-points. The user is also able to define logical behaviour such as to wait until a specific signal is set or a timer has expired before moving to the next way-point. Finally, depending on the configuration and the user task specification, the robot software determines the trajectory of the robot using motion planning and trajectory generation algorithms [15], [16]. The information from the trajectory generation is fed to a low-level controller that runs periodically, usually, with a fixed cycle time having a typical value of 1 millisecond to control the joints of the robots [16]. To achieve this, a robot controller software, composed of diverse components, is systematically put together to provide a coherent mechanism for manipulator control supported by a real-time operating system. Fig. 2 illustrates the block diagram of a robot controller. The position information component of the robot controller reads the desired values either from sensors or a pre-defined program and forwards it to the kinematic controller. The kinematic controller generates the necessary reference values which are used by the dynamic controller to generate the required actuator commands for each joint. The sensors component provides the feedback information to the kinematic controller and the dynamic controllers.

IV. Use-case Aspects

Cyber physical systems such as industrial robots are dependent not only on the logical correctness of the computations but also the time at which the computations are completed. For example, the dynamic controller as shown in Fig. 2 is dependent on new reference values from the kinematic controller to be available during each new cycle of the control loop for better control. Using stale information may result in the robot deviating from the expected path. In order to cover such scenarios, as a fog-based industrial robotic system, we denote a fog computing system enhanced with real-time capabilities, i.e., capabilities to guarantee that a computational task is finished and its result is transmitted within a predefined amount of time. Further, such a system should have the following properties: timeliness (the results of the computation must be available and transmitted within a predefined time), predictability (the system must be analyzable to guarantee performance of the applications), efficiency (the system should efficiently utilize available resources), scalability (the system should be able to grow in size dynamically when new cloud/fog nodes are discovered) and fault tolerance (the system should provide mechanisms to deal with unpredictable failures). We note that security is a critical property for enabling fog-based robotic systems, however, it is beyond the scope of this paper, and interested readers are referred to [17], [18], [19], [20], [21]. In the following sections, we elaborate in detail on the elementary set of aspects of fog computing in the context of factory automation environments.

A. Virtualization

Fog computing allows co-location of independent applications on a shared fog node in a fog network. To host such applications, for example, the different kinematic controllers of different robots, the fog nodes must provide virtual environments that ensure a proper isolation, resource allocation and the fulfilment of the demands of the applications. Virtualization abstracts physical hardware from the applications running on that hardware, and thus, emulates computing environments in such a way that it appears for the applications that they are executed exclusively on a dedicated hardware. Virtualization allows to host multiple isolated applications and their software dependencies on a single physical device, and thus, reducing resource wastage. However, sharing resources may lead to (time) unpredictability, and consequently, the timing constraints may be violated.

There are two main classes of virtualization technologies: Hypervisor-based virtualization and container-based virtualization [22]. Hypervisor-based virtualization utilizes a hypervisor that distributes resources among virtual machines that behaves like independent virtual computers containing dedicated operating systems and scheduling mechanisms. This solution requires support from hardware and, additionally introduces non-negligible overheads [23] and performance degradation. The container-based virtualization is provided purely by the host OS and it offers near-native performance, rapid startup

times and low overhead, however the resource isolation may be weaker [24].

The solutions addressing real-time in hypervisor-based virtualization, e.g., RT-Xen [25] or PikeOS[1]. However, time predictability in container-based virtualization is a novel topic. As summarized in [26], real-time behavior of container-based virtualization must be supported by a predictable host operating system and real-time scheduling policies that are container-aware. The former is addressed by the application of a real-time patch that makes the Linux Kernel fully preemptive [27] or by the use of a real-time co-kernel that runs side-by-side with a standard Kernel. Real-time aware scheduling policy for containers is addressed by utilization of hierarchical scheduling that provides temporal isolation of containers [28].

For a full adoption of real-time virtualization, we see the following challenges: (i) Minimizing the interference between virtualized applications (e.g., co-located memory or cache-intensive applications may experience performance degradation of the physical fog nodes). (ii) Lack of real-time communication mechanisms between virtualized applications (that may be executed on different devices) and enabling the communication in a time-predictable, secure, and safe manner. (iii) The possibility of supporting a mixture of both hypervisor-based virtualization (to satisfy hard real-time requirements) and container-based virtualization (to satisfy less stringent requirements) in a single fog computing system should be explored. (iv) Dealing with unpredictability of communication between virtualized applications in a fog computing architecture due to network performance.

Multi-core Platforms: Additional set of problems hampering timing predictability of virtualized applications is caused by the use of a Multi-Core Processor (MCP). General Commercial Off-The-Shelf (COTS) MCPs share hardware resources like cache and main memory. Sharing such resources is one of the primary sources of Worst Case Execution Time (WCET) unpredictability [29] of a task. In an MCP, the task execution time not only depends on the task itself, but it is also significantly influenced by the applications running on the other cores (intercore interference). Nowotsch *et al.* [30] showed that the latency of a single memory store request can increase up to 25.82 times when the number of active cores increases from 1 to 8.

Virtualization aims for resource optimization and allows co-execution of independent applications such as the control tasks of different robots on the same hardware. Since sharing hardware resources leads to execution time unpredictability, bounding the WCET is necessary for the use of schedulability analysis and admission tests by the orchestrator. Such analysis allows the orchestrator to optimally allocate the resources for the applications while meeting the timing constraints imposed by the applications.

There are some proposed solutions based on a memory bandwidth management system to improve, and also to guarantee the tasks' WCET on an MCP in the context of real-time systems, like in Yun *et al.* [31] and Agrawal *et al.* [32]. However, it is necessary to know the WCET of a task to schedule a set of tasks, and since the WCET depends on the inter-core interference, and which in turn is dependent on the schedule, we need solutions that address both memory bandwidth and scheduling problems together.

The solutions based on time and memory bandwidth management are well suited for time-triggered applications in a context where we know the number of active cores at the same time, and the maximal inter-core interference introduced by these cores. It becomes unrealistic for a industrial robotic system where the number of applications and their requirements change dynamically and during runtime. As an example, unlike in regular real-time systems, where the execution behaviour can be modelled via different task models based on WCETs, modelling the execution time of motion planning tasks is complicated. The reason for this variability is twofold. One, the user of the robotic system is free to program the robot motion as desired. Two, the non-deterministic nature of the motion planning and trajectory generation algorithms. Some commonly used planning algorithms such as the Probabilistic Road Map (PRM) sample the joint space to find a collision free path [15]. In the best case, if a connection between initial point and the target point is established without collisions in the first iteration, no further computation is required. The execution time in this case can be minimal. While in the worst case, possibly in the presence of multiple obstacles, the number of samples that need to be checked for connectivity and collision can be huge, requiring larger computational time. This makes the motion and trajectory planning algorithms non-deterministic [33]. Given such a scenario, using traditional WCET-based analysis and design can result in significant wastage of resources. Therefore, to achieve real-time guarantees in a system running on a MCP, we have (i) to ensure the temporal isolation of tasks and containers taking the platform resource contentions into account, and (ii) to design a resource sharing mechanisms for managing access to shared resources, such as bus and memory controller, taking dynamic applications scenarios into account.

The worst-case number of memory accesses that an application can issue depends on many different factors, for instance, hardware like in 32 or 64 bits platform, system configuration, and operating system. However, it is also unrealistic to assume that the maximum memory bandwidth necessary for each critical task is known. To address the lack of this data, a solution is to bind the containers that run critical tasks to a specific core and monitor their execution progress in predefined checkpoints along the time, to check if the execution is going according to the schedule. In case execution is late, the other non-critical containers can be paused to reduce the inter-core interference preventing the critical one from missing a deadline.

COTS MCP are designed primarily for the average-case performance and that is not enough to meet the real-time requirements of robotic applications. Therefore, to address the loss of predictability in such platforms, we need to apply new

[1]https://www.sysgo.com/products/pikeos-hypervisor

mechanisms and know the application resource needs better.

B. Real-Time Aware Orchestration

The role of the orchestrator in the fog-based industrial computing systems is to deploy and maintain virtualized applications in the shared fog and cloud computing environment in such a way that the resource and timing requirements are fulfilled. Although there has been extensive research on orchestration, taking into account various resources and optimization goals, and there are several mature orchestrator systems available[2], none address timing-related requirements [34]. Therefore, we envision the following real-time enhancements of the orchestrator that can serve in fog computing systems. The real-time orchestrators should provide functionality for resource selection, real-time deployment, real-time aware service monitoring and real-time resource control.

a) Resource Selection: The orchestrator must be aware of timing requirements of applications as well as real-time capabilities provided by fog nodes. The orchestrator must perform schedulability tests that ensure that the virtualized applications will be granted enough CPU time for performing time-critical actions. Additionally, the orchestrator should be aware of interference between virtualized applications and try to minimize such impacts during the resource selection phase.

b) Real-time Deployment: The orchestrator should provide a bounded time for the deployment of virtualized applications. It should take into account transmission times of the applications from the repository to the fog node and the startup time of the application. This enhancement is important for safety and dependability aspects, e.g., during the redeployment of a failed application.

c) Real-time Aware Service Monitoring: Due to imperfections of the underlying operating systems (e.g., Linux) that may not provide accurate temporal isolation for virtualized environments, the orchestrator must monitor the quality of service delivered by the virtualized applications (e.g., deadline misses or lateness[3]). The orchestrator should use this information during the resource selection phase.

d) Real-time Aware Resource Control: Based on real-time related metrics mentioned previously, the orchestrator should perform migration of virtualized applications in order to improve their real-time behavior. This can be a case of a memory or cache-intensive application that may experience performance degradation when it is co-located with another memory or cache-intensive application on a single node.

C. Timely and Reliable Communication

The fog nodes should communicate with a number of sensors, actuators and other devices in different layers, using both wired and wireless connections to ensure the smooth operation of the robot tasks. In many instances, the communication should be in real-time. For example, the sensors providing the feedback information as shown in Fig. 2, may be wireless, and for the feedback based dynamic control algorithms, the sensor data should be available before the beginning of the computation of next cycle of the control loop. However, when the probability of losing a packet goes up, especially in a wireless network, for example, due to the noise in industrial environments combined with the Doppler effect, multi-path fading, and dynamic wireless channel [35], a re-transmission technique may be necessary to improve communication reliability. This can lead to an increase of end-to-end latency resulting in unavailability of the sensor values in a timely manner. This in turn may affect the path accuracy of the robots.

Therefore, to enable real-time robot control, the wireless communication protocols should be designed to fulfil both strict timeliness and reliability requirements. For example, the wireless protocols should be able to guarantee an upper bound end-to-end latency of 1 millisecond along with a probability that a packet does not reach its destination before the deadline to not exceed 10^{-7} [36].

For wired networks, more recently, Time Sensitive Networking (TSN) has emerged as a front-runner and a competing technology to the well established field-bus standards that have been the staple of industrial and automotive networking [37]. TSN is a set of standards that provide real-time guarantees over standard Ethernet. Hence, current IEEE 802.1Q standards [38] come integrated with TSN so that vendors can directly provide properties such as timeliness, fault-tolerance, reliability and availability to their networking products.

Since wired networks can offer deterministic communications, wireless ones should also guarantee deterministic reliable communications at the same level. In this context, Medium Access Control (MAC) protocols and relaying strategies are central in achieving the desired requirements. Rajandekar *et al*. [39] concluded that the hybrid MAC protocols can meet the stringent requirements of reliability and timeliness. Furthermore, since the fog nodes need to support for a large number of IoT devices in a massive IoT scenario such as in a factory-to-factory communication (Section IV-D), the proposed MAC protocols must support a large number of simultaneous connections. Li *et al*. [40] proposed a hybrid Time Division Multiple Access (TDMA) Non Orthogonal Multiple Access (NOMA) scheme for cellular-enabled machine-to-machine communications. With NOMA, multiple nodes can be served simultaneously utilizing the same time-frequency resources but different power levels. The proposed time-sharing scheme is introduced to deal with the massive deployment of devices, while improving the total transmission time and energy efficiencies. This solution is suitable for fog networks where a NOMA transceiver may be deployed at the fog nodes [41]. Another approach, Hoang *et al*. [42] proposed a relaying sequence that considers all cases that can happen in each time slot. Hence, the probability of an error is an exact value compared to an upper bound value as is the case in other solutions related to multi-hop communication. Moreover, the authors introduced a method of group based relaying on a hybrid TDMA-CSMA (Carrier-Sense Multiple Access)

[2]e.g., Kubernetes (https://kubernetes.io/), Docker Swarm (https://docs.docker.com/engine/swarm/), or OpenStack(https://www.openstack.org/)

[3]The delay of a task completion with respect to its deadline.

protocol to address the drawback in relaying sequencing where a relayer keeps silent if it does not have any correct copy of the packet [43]. Therefore, the relaying strategies combined with packet aggregation are practical techniques that can be implemented on the fog nodes. However, further research on these techniques based on a hybrid protocol with NOMA is needed to minimize the end-to-end latency, especially in the case of the large number of connections.

To obtain timely and reliable communication necessary for the robot control, we observe several challenges including: (i) the placement of fog nodes' transceiver must minimize the probability of error and end-to-end latency, (ii) hybrid protocols with NOMA should be further studied and improved to deal with the massive connection, and (iii) in industrial environments, such as those of the factory automation environments, there are strict constraints, i.e., a limited number of relay nodes and re-transmissions are possible, and therefore, specific relaying strategies should be defined.

D. Scalability

Another aspect of the proposed use case is scalability [44]. In an industrial setting (as discussed in Section III), the number of participating compute nodes can become very large depending on the size of the factory, because compute nodes may include cloud and fog nodes as well as the interface devices. Furthermore, apart from interconnecting the compute nodes of one factory, there is also the possibility of interconnecting various factories with each other. This can be beneficial when, for instance, the production of an item in a factory is reduced or halted due to faults in the hierarchical or vertical communication of the nodes (see Section IV-E). In such cases, another factory which produces the same product (and is able to increase production) can change its production plans in order to compensate for the faults. This can be achieved by coordinating the function of multiple factories through fog and cloud nodes.

Along with the potential benefits of interconnected factories, and the advantages that fog computing will bring to industrial environments, there are also related concerns. For example, the communication overhead from the interactions among the distributed cloud and fog nodes, needs to remain limited so that it does not interfere with the operation of the applications [44], and it does not compromise the functionality of nodes which execute tasks with strict deadlines (see Section IV-A). Most applicable fog computing approaches aim at enabling computing close to the edge of the network [45]. However, they do not consider scalability metrics (e.g., communication overhead) which show that scaling a fog computing system to a large degree (i.e., adding many compute nodes), does not compromise the performance of the applications [46].

In an industrial fog computing setting (as discussed in Section III), when adding new compute nodes (e.g., new controllers, fog, and cloud nodes) these nodes need to be discovered and integrated in the system [47]. This means that the existing nodes need to store the new nodes' information (e.g., IP addresses, amount of computational resources, etc.) so that they can communicate (e.g., using widely-used communication protocols [48]). This creates the problem of determining which nodes a new node should connect to [49]. A simple solution to this problem would be that each node maintains a global view of the system, i.e., stores the information of all the other nodes. However, this might be impractical since the compute nodes at the edge (e.g., the fog nodes) may be able to store only a small number of other compute nodes due to having limited computational resources [50]. For this reason, scalability still poses a challenge in the proposed use case.

E. Dependability and Safety

Dependability: While fog-based systems address some of the limitations of existing architectures [3], they introduce new challenges for ensuring dependability attributes, for instance, reliability and availability. For example, the fog-based software architecture demands frequent data exchange between the different nodes of the fog layer to accomplish a functionality such as trajectory generation and control if we assume that the trajectory generators are placed on a different fog nodes. This puts more focus on ensuring reliability and availability of the fog platforms.

The attributes of the fog services, such as compute and communication, are challenged by dependability threats viz., errors, faults and failures [12], which in turn might disrupt the entire functionality of the system. Threats related to computational resources include the occurrence of faults in fog nodes. Dependability threats associated with the orchestrator may lead to either a performance failure or wastage of resources. Another example is A failure in data storage that might result in data loss. This can result in loss of user-specified tasks or the configuration settings mapping different sensors to the user task specifications and data variables. Additionally, data stored for future analysis in the cloud may not be accessible. Threats related to communication are faults that might disrupt or prevent the connectivity between the different nodes as well as the different layers in the architecture. Loss of communication between the nodes and the edge devices can result in stoppage of robots as the new trajectory parameters are not available for the low level controllers. Such failures can have a cascading effect within a robotic cell. For example, if a robot stops in the worskpace of another robot, the safety system may force the independent robot to take mitigating actions such as slowing down or stopping altogether. This situation may be undesirable It is therefore critical to preserve a tight end-to-end communication, while providing suitable fault tolerance mechanisms.

Dependability approaches for fog computing in the literature are mainly proposed to address dependability requirements using fault management solutions and redundancy techniques [51]. Fault management solutions proposed are mainly considering threat detection tools like monitoring [52] for fault prevention and fault removal, or failure recovery solutions like reconfiguration upon failure [53]. Redundancy techniques proposed in the literature focus on addressing different dependability requirements of the fog platform. For instance, im-

TABLE I
OVERVIEW OF THE ASPECTS AND THEIR INTER-RELATION AND CHALLENGES.

Aspect/Relation	Virtualization & Orchestration	Multi-Core Platform	Communication	Safety & Dependability	Scalability
Virtualization & Orchestration	–	Platform abstraction and efficient resource management.	Resource allocation for achieving tight end-to-end communication.	Provide predictability and fault tolerance.	Complexity.
Multi-Core Platform	Provide real-time guarantees and predictability.	–	Virtualization & Orchestration isolates the platform.	Intra-core isolation and predictability.	Challenges wrt. hierarchical architectures & legacy.
Communication	Heterogeneity, combination of wired and wireless networks.	Virtualization & Orchestration isolates the platform.	–	Trade-off between reliability and safety.	Large numbers along with big data communication.
Safety & Dependability	Heterogeneous safety levels.	Certification.	Tight end-to-end communication and fault tolerance.	–	Redundancy overhead.
Scalability	Runtime reconfiguration.	Hardware architecture.	Congestion.	Increased complexity.	–

proving reliability [54], and availability [55], of the system by introducing redundancy in the form of a redundant node [54], in a network, redundant communication channels [56] and task offloading [57] or application migration [58].

The proposed dependability solutions are mainly tied to redundancy methods and replication techniques, for instance, using passive/ active replicas of the system components. However, using replicas for each component will result in overhead of cost and consumption of resources [59], and it can also impact the scalability of the overall system. Therefore, we need further research to improve the system dependability.

Safety: Safety in robotics involves multiple domains such as the design of the manipulator arm and the layout of the cell. We focus on safety from the software perspective. We define safety in the present use-case as the property of the system which guarantees the timely and correct execution of safety-critical tasks (with hard real-time deadlines) under *all* operational conditions [60] [61]. This encompasses scenarios wherein the system reverts to a safe state in the event of a safety violation or hazard. For instance, applications using motion control algorithms for robot arm movements need safety guarantees in terms of the range of motion that is allowed to prevent hazardous motions. Typically a safety function (or task) is responsible for ensuring the said guarantees. Catastrophic consequences (such as loss of life, danger to the surrounding environment) can ensue in case of a failed execution of safety-critical tasks occur [62].

To ensure that such failure scenarios are well handled, it is necessary that timing analysis, schedulability tests and the network schedule are designed considering the potential threats to safety. The fact that a single failure scenario can pose a threat to safety is the principal challenge for the verification and validation of multi-core platforms, and more in general for fog architectures.

Therefore to ensure safety of users and of the equipment, we need to ensure such threats are handled at the design phase itself, even in presence of heterogeneous safety levels. Additionally, as in all safety systems, there needs to be no single point of failure. Along with these challenges, it is necessary to investigate if real-time applications running in the fog provide the same level of safety guarantees that the existing safety-certified robotics systems provide.

V. CONCLUSION

Fog-computing brings cloud-like capabilities to low latency applications but the practical implementation of a fog architecture for real-time applications such as robot control is non-trivial. To move a step forward in this direction, we need a holistic approach that considers different technical aspects independently but also in conjunction with each other. We summarize the relationship between the different aspects in Table IV-D.

To make effective use of the fog resources, we need virtualization techniques such as hypervisors and containers with real-time capabilities to support the timing requirements of robotic cells and robot motion. To provide temporal isolation, hypervisors and containers need memory management techniques to limit the interference of shared caches and buses within the multi-core architectures. By providing bounds on the interference, we can enable the orchestrator with the capabilities to use real-time schedulability analysis and appropriate scheduling algorithms to allocate applications to fog nodes, to ensure timing predictability. Since the fog platform is a distributed system involving fog nodes and edge devices such as low-level controller and sensors, we need real-time communication mechanisms. Such communication can be wired or wireless. The constraints imposed by communication technologies further guide the scheduling and allocation of resources by the orchestrator. For robotic applications, dependability and safety are important attributes to prevent any damage to the equipment and more importantly, to safeguard the health of the operators working in close proximity to such robots. To this end, we need solutions that consider the requirements of the applications as well as the new challenges imposed by the fog platforms. In this paper, we briefly discussed a robotic cell environment to highlight the usefulness of fog-based solutions and discussed key aspects such as resource orchestration and network scalability, virtualization and memory management techniques supported by real-time communication paradigms. Further we discussed the dependability and safety issues that need to be considered when moving towards the fog-based architectures for robotic applications.

Acknowledgement: This research has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreements No.764785, FORA—Fog Computing for Robotics and Industrial Automation.

REFERENCES

[1] M. Hägele *et al.*, "Industrial robotics," in *Springer Handbook of Robotics*, 2016.

[2] V. Villani *et al.*, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, 2018.

[3] S. M. Salman *et al.*, "Fogification of industrial robotic systems: Research challenges," in *Proceedings of the Workshop on Fog Computing and the IoT*, 2019.

[4] F. Bonomi *et al.*, "Fog computing and its role in the internet of things," in *Workshop on Mobile Cloud Comp.*, 2012.

[5] S. Dey and A. Mukherjee, "Robotic slam: a review from fog computing and mobile edge computing perspective," in *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services*, 2016.

[6] A. Kattepur *et al.*, "A-priori estimation of computation times in fog networked robotics," in *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2017.

[7] IEEE, "Ieee adoption of openfog reference architecture for fog computing," 2018.

[8] F. Bonomi *et al.*, *Fog Computing: A Platform for Internet of Things and Analytics*, 2014.

[9] A. Dastjerdi *et al.*, "Chapter 4 - fog computing: principles, architectures, and applications," in *Internet of Things*, R. Buyya and A. V. Dastjerdi], Eds. Morgan Kaufmann, 2016.

[10] E. M. Tordera *et al.*, "What is a fog node a tutorial on current concepts towards a common definition," *arXiv preprint arXiv:1611.09193*, 2016.

[11] R. Gerber *et al.*, "Guaranteeing real-time requirements with resource-based calibration of periodic processes," *IEEE Trans. on Softw. Eng.*, 1995.

[12] A. Avizienis *et al.*, "Basic concepts and taxonomy of dependable and secure comput." *IEEE Trans. on Dep. and Secure Comp.*, 2004.

[13] J. Zhang and X. Fang, "Challenges and key technologies in robotic cell layout design and optimization," *J. Mech. Eng. Science*, 2017.

[14] P. Kah *et al.*, "Robotic arc welding sensors and programming in industrial applications," *International Journal of Mechanical and Materials Engineering*, 2015.

[15] S. M. LaValle, *Planning Algorithms*, 2006.

[16] T. Kröger, *On-Line Trajectory Generation in Robotic Systems*, 2010.

[17] I. Stojmenovic *et al.*, "An overview of fog computing and its security issues," *Concurrency Computation*, 2016.

[18] S. Yi *et al.*, "Security and privacy issues of fog computing: A survey," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015.

[19] M. Mukherjee *et al.*, "Security and privacy in fog computing: Challenges," *IEEE Access*, 2017.

[20] P. Zhang *et al.*, "Security and trust issues in fog computing: A survey," *Future Generation Computer Systems*, 2018.

[21] A. Khalid *et al.*, "Security framework for industrial collaborative robotic cyber-physical systems," *Computers in Industry*, 2018.

[22] R. Morabito *et al.*, "Hypervisors vs. lightweight virtualization: A performance comparison," in *IEEE Int. Conf. on Cloud Eng.*, 2015.

[23] S. Hoque *et al.*, "Towards container orchestration in fog computing infrastructures," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 2017.

[24] M. G. Xavier *et al.*, "A performance isolation analysis of disk-intensive workloads on container-based clouds," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*.

[25] S. Xi *et al.*, "RT-Xen: Towards real-time hypervisor scheduling in Xen," in *ACM Int. Conf. Embedded Software*, 2011.

[26] V. Struhár *et al.*, "Real-time containers: A survey," in *Workshop on Fog Comput. and IoT*, 2020.

[27] A. Moga *et al.*, "Os-level virtualization for industrial automation systems: Are we there yet?" in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016.

[28] L. Abeni *et al.*, "Container-based real-time scheduling in the Linux kernel," *ACM SIGBED Review*, 2019.

[29] S. Schliecker *et al.*, "Bounding the shared resource load for the performance analysis of multiprocessor systems," in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2010.

[30] J. Nowotsch *et al.*, "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement," in *Euromicro Conf. on Real-Time Systems (ECRTS)*, 2014.

[31] H. Yun *et al.*, "Memory bandwidth management for efficient performance isolation in multi-core platforms," *IEEE Trans. on Comp.*, 2016.

[32] A. Agrawal *et al.*, "Contention-Aware Dynamic Memory Bandwidth Isolation with Predictability in COTS Multicores: An Avionics Case Study," in *Euromicro Conf. on Real-Time Syst.*, 2017.

[33] M. Alcon *et al.*, "Timing of autonomous driving software: Problem analysis and prospects for future solutions," in *IEEE Real-Time and Embedded Technol. and Appl. Symp.*, 2020.

[34] M. A. Rodriguez and R. Buyya, "Container-based cluster orchestration systems: A taxonomy and future directions," *Software: Practice and Experience*, 2019.

[35] A. Willig *et al.*, "Wireless technology in industrial networks," *Proceedings of the IEEE*, 2005.

[36] R. Candell and M. Kashef, "Industrial wireless: Problem space, success considerations, technologies, and future direction," in *Resilience Week*, 2017.

[37] N. Finn, "Introduction to Time-Sensitive Networking," *IEEE Commun. Standards Mag.*, 2018.

[38] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks - Redline," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014) - Redline*, 2018.

[39] A. Rajandekar and B. Sikdar, "A survey of MAC layer issues and protocols for machine-to-machine communications," *IEEE Internet of Things J.*, 2015.

[40] Z. Li and J. Gui, "Energy-efficient resource allocation with hybrid tdma–noma for cellular-enabled machine-to-machine communications," *IEEE Access*, 2019.

[41] H. Tezuka *et al.*, "A UL-NOMA system providing low E2E latency," in *IEEE VTS Asia Pacific Wireless Commun. Symp.*, 2019.

[42] L.-N. Hoang, "Relaying for timely and reliable applications in wireless networks," Ph.D. dissertation, Halmstad University, 2017.

[43] L. Hoang *et al.*, "Relay grouping to guarantee timeliness and reliability in wireless networks," *IEEE Commun. Lett.*, 2019.

[44] V. Karagiannis *et al.*, "Enabling fog computing using self-organizing compute nodes," in *IEEE Int. Conf. Fog and Edge Comput.*, 2019.

[45] V. Karagiannis and A. Papageorgiou, "Network-integrated edge computing orchestrator for application placement," in *Int. Conf. Netw. and Service Manage.*, 2017.

[46] V. Karagiannis and S. Schulte, "Comparison of alternative architectures in fog computing," in *IEEE Int. Conf. Fog and Edge Comput.*, 2020.

[47] V. Karagiannis *et al.*, "Addressing the node discovery problem in fog computing," in *Workshop Fog Comput. and IoT*, 2020.

[48] ——, "A survey on application layer protocols for the internet of things," *Trans. on IoT and Cloud comput.*, 2015.

[49] V. Karagiannis, "Compute node communication in the fog: Survey and research challenges," in *Workshop on Fog Comput. and IoT*, 2019.

[50] R. Ranjan *et al.*, "The next grand challenges: Integrating the internet of things and data science," *IEEE Cloud Computing*, vol. 5, no. 3, pp. 12–26, 2018.

[51] Z. Bakhshi and G. Rodriguez-Navas, "A preliminary roadmap for dependability research in fog computing," *ACM SIGBED Review*, 2020.

[52] X. Yuan *et al.*, "Cyber-physical systems for temporary structure monitoring," *Automation in Construction*, 2016.

[53] Y. Xiao *et al.*, "A novel task allocation for maximizing reliability considering fault-tolerant in VANET real time systems," in *IEEE Int. Symp. on Personal, Indoor, and Mobile Radio Communications*, 2017.

[54] A. Aral and I. Brandic, "Quality of service channelling for latency sensitive edge applications," in *IEEE Int. Conf. Edge Computing*, 2017.

[55] X. Chen *et al.*, "A fault-tolerant data acquisition scheme with mds and dynamic clustering in energy internet," in *IEEE Int. Conf. Energy Internet*, 2018.

[56] K. E. Benson *et al.*, "Ride: A resilient IoT data exchange middleware leveraging SDN and edge cloud resources," in *IEEE/ACM Third Int. Conf. Internet-of-Things Design and Implementation*, 2018.

[57] M. T. Saqib and M. A. Hamid, "FogR: A highly reliable and intelligent computation offloading on the internet of things," in *IEEE Region 10 Conf. (TENCON)*, 2016.

[58] O. Osanaiye *et al.*, "From cloud to fog computing: A review and a conceptual live vm migration framework," *IEEE Access*, 2017.

[59] Z. Bakhshi *et al.*, "Dependable fog computing: A systematic literature review," in *Euromicro Conf. Software Eng. and Adv. Appl.*, 2019.

[60] R. Dobrin *et al.*, "On fault-tolerant scheduling of time sensitive networks," in *Int. Workshop on Security and Dependability of Critical Embedded Real-Time Syst.*, 2019.

[61] N. Desai and S. Punnekkat, "Safety of fog-based industrial automation systems," in *Proceedings of the Workshop on Fog Computing and the IoT*, 2019.

[62] N. Desai and S. Punnekkat, "Safety-oriented flexible design of autonomous mobile robot systems," in *2019 International Symposium on Systems Engineering (ISSE)*, 2019.