# Communication and container reconfiguration for cyber-physical production systems

Patrick Denzler*, Daniel Ramsauer*,Thomas Preindl*, and Wolfgang Kastner*
*Institute of Computer Engineering, TU Wien, Vienna, Austria
Email: {patrick.denzler, daniel.ramsauer, thomas.preindl, wolfgang.kastner}@tuwien.ac.at

***Abstract*—The Fourth Industrial Revolution, or Industry 4.0, aims to advance flexibility and reconfigurability in current production systems. This paper sets cyber-physical production systems in context with Industry 4.0 concepts and architectures. The combination points out the importance of the interplay between communication and component reconfiguration when changes occur. A solution that utilizes fog computing, container-based deployment and Kubernetes functionality is presented and evaluated in simplified reconfiguration scenarios. The findings show the feasibility and the challenges of the solution and point towards further research to successfully create reconfigurable cyber-physical production systems.**

***Index Terms*—Cyber Physical Production Systems, Configuration, Fog computing, OPC UA**

## I. Introduction

From a historical perspective, the core enablers of so-called industrial revolutions were disruptive technologies and their follow-up applications. The newest, the fourth industrial revolution (4IR), is still in the initial stage, while the expectations are unprecedented in regards to technological development and societal benefits compared to its predecessors [1].

One potential core enabler for the 4IR are cyber-physical production systems (CPPSs), a new generation of embedded systems with advanced capabilities and many different definitions [2]. A commonality in all CPPS definitions is the idea of a logical/cyber embodiment linked to a physical/mechatronic aggregate, acting as its virtual representation or avatar. Applying those principles to an automation system creates a competitive edge concerning system reconfigurability [3]. The advantage lies in the modularity, and self-contained composition of cyber-physically represented equipment and systems. Instead of building statically defined and customized interconnections, CPPS focus on dynamic and on-demand interactions between the system components [4]. These principles allow dynamic and sustainable production and operations, where the industrial equipment is almost instantly reorganized to address new and ever-changing business opportunities. In such a setting, equipment is easily disconnected and moved to another site once the manufacturing task is finished. As each component has a specific set of capabilities, the possibility exists to rent or lease equipment from a machine pool provided by an external provider in a highly flexible manner. However, the potential CPPS business models are out of this paper's scope. Interested readers are encouraged to find more information in [5].

Reconfiguration of production systems looks back onto more than 20 years of multidisciplinary research. The conducted research led to several creative production paradigms and technical ideas, ranging from general applicable multi-agent systems (MASs) [6] and service-oriented architectures (SOAs) [7] to flexible and reconfigurable manufacturing systems (FMSs and RMSs) [3], [4] and specific paradigms such as holonic manufacturing systems (HMS) [8], bionic manufacturing systems (BMSs) [9] and evolvable assembly systems (EASs) [10]. However, most of those concepts never reached the industry and were not applied in actual representative operations. The reasons for the hesitant implementation have been identified as the low maturity of the related technologies [11], the absence of architectures that guide the development from conceptual phases to the actual hardware implementation [12], and missing verification and validation procedures [13]. However, most significant is the lack of know-how in establishing cyber-physical interfaces [14]. In essence, all paradigms incorporate the main principles, architectures and technologies required for a CPPS, but none has been established or realized in its totality. Another reason is the involved uncertainty when changing from conventional automation technologies to a CPPS.

This paper aims to revisit CPPS reconfiguration capabilities and presents a possibility to integrate them partly into industrial automation. By combining the concept of cyber-physical production modules (CPPMs), a subunit of CPPS, with Industry 4.0 (I4.0) [2], it becomes apparent that the interplay between communication and the software system is a crucial element for reconfiguration. The presented solution addresses this interplay by utilizing fog computing and Kubernetes capabilities in combination with a configurator to deploy and reconfigure containers and communication protocols. While investigations into containerization are being carried out in fog computing [15], little can be found that deals with the simultaneous configuration of network and container contents. Some research examines configuration issues between time-sensitive networking (TSN) and fog computing in general [16], [17]. Similarly, there is much research concerning the configuration of protocols under various aspects such as OPC Unified Architecture (OPC UA) safety [18] or automatic configuration by using auto discovery or other approaches [19], [20].

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.
978-1-7281-2989-1/21/$31.00 ©2021 IEEE

For demonstrating the feasibility of the proposed solution, the evaluation case includes OPC UA and Message Queuing Telemetry Transport (MQTT) [21] as communication protocols and an OPC UA/MQTT gateway as configurable container content. The contribution is twofold: Firstly, the use of fog computing and containerization as a potential enabler for reconfiguration. Secondly, presenting the arising challenges when configuring communication, routing, and CPPMs software systems in a dynamic environment.

In this context, this paper proceeds with the introduction and definition of the main terminology, background, and the relation between CPPM and I4.0. Section III presents the proposed solution to address reconfiguration. The evaluation of the setup is introduced in Section IV, and Section V describes the evaluation scenarios and the obtained results and identified challenges. Section VI discusses the findings and Section VII concludes the paper and points to future research directions.

## II. Background

Defining a CPPS is a complex endeavor. Due to the multidisciplinary nature, available definitions tend to adjust to the respective domain from which the authors look at a CPPS. Commonly agreed seems to be that a CPPS represents the sum of all involved human resources, production equipment, and accumulated products. In industrial automation, a CPPS encompasses all layers of the automation pyramid, however, focusing on the lower levels. For each element, the CPPS provides a cyber-physical representation with formulated interaction interfaces for monitoring and controlling its operation. Moreover, those interfaces allow access to the generated knowledge accumulated by the equipment and human resources during the production process and obtained by the produced products throughout their life cycle [2]. The collected knowledge continuously improves operations and resource consumption by establishing performance indicators. The knowledge generated by the products helps to improve CPPSs operations [22].

In a CPPS, a component may establish relations of different nature in rank, scope, and abstraction level with other components. A component within a CPPS therefore follows a modular architecture. Each component has one or a few distinct functions and allows interaction via simple and well-defined interfaces [23]. The internal architecture does not affect other components as they only see the provided interfaces. This circumstance allows a loose coupling between the components of a CPPS [24]. Ideally, the interfaces handle all interactions, and the system's behavior is equal to the sum of its components.

Some authors refine the CPPS components as CPPMs [22]. Such modules combine three logical entities: equipment (mechatronic system), a controller or computing platform, and a cyber representation (software system) [24]. The computing platform provides access to the physical equipment and can be shared between several cyber representations. The cyber part represents the interface and the necessary algorithms to interact with the module. In theory, the CPPMs can interact with all sorts of other modules (e.g., human resources or subsystems) without reprogramming. The reason is that the cyber part abstracts the hardware and decouples interaction and execution logic. CPPMs are either built by a functional or structural decomposition approach. The functional decomposition is a system decomposition style where the boundaries of the modules align with system functions or roles (e.g., order, feeding, supply) [10]. In the structural decomposition style, the modules' boundaries overlap with those of the physical components [25].

### *A. Reconfiguration in CPPS*

Reconfiguration in production systems has a long research tradition and is a characteristic of excellence for mid-variety/volume production [26]. Besides that, current trends indicate sustainability as the more critical excellence paradigm; reconfiguration is an essential part of long-term sustainability. The ability to reconfigure and share resources (e.g., equipment) in different sites and activities contributes directly to the total sustainability of the system. In general, a reconfigurable manufacturing system (RMS) could be defined as a system designed to accommodate rapid change related to production capacity, functionality, or new governmental regulations. This capability includes the structure of the system as well as the hardware and software components. Previous research [3], [27], [28] identified required characteristics of reconfigurable systems as:

- Scalability – the ability to sustainably increase or reduce manufacturing resources.
- Modularity – the ability to encapsulate functions to be independently used in various production activities.
- Integrability – the ability to integrate new modules into the system without disruptions.
- Customization – the system is designed to accommodate large-scale changes.
- Convertibility – the ability to ease functional changes.
- Diagnosability – the ability to react to disturbances.

Most of the research activity, therefore, focuses on enablers that address those characteristics. Such enablers are, among others, a highly universal product and systems design with mobile equipment, plug-and-play functionality, and increased compatibility between the components within the production system.

### *B. Relation to Industry 4.0*

It is not surprising that the envisioned advanced functionalities of CPPS found their way into the I4.0 and represent an essential building block for the 4IR [2]. I4.0 describes concepts to boost industrial output and competitiveness by improving efficiency, productivity, and flexibility. The aim is to create a digital enterprise that is entirely interconnected by combining people, innovative connected technologies with advanced production and operations techniques. All systems in I4.0 communicate, analyze, and use collected data to adjust intelligently the physical world [1]. The used technologies span from robotics, analytics, artificial intelligence, Internet of

Things (IoT), additive manufacturing, to advanced materials. While I4.0 roots in manufacturing, it includes all aspects of an enterprise environment and changes how organizations process and apply information to continuously improve their operations and adjust to consumer wishes.

## C. Asset Administration Shell

Analogous to CPPS, I4.0 aims at optimizing production by using reconfigurable manufacturing, in which production equipment can be reorganized or reassembled. In I4.0, assets are more than just machines, production modules, or systems; they include references to individual products, software installations, intellectual property, or human resources. Therefore, an asset must be identifiable with a type, its lifecycle stage, or its current state. Each asset communicates, apart from its real-time parameters, also its production capabilities. The combination creates a logical representation called the Asset Administration Shell (AAS) [29]. The AAS describes an asset electronically in a standardized manner and allows the exchange of asset-related data. In I4.0 terminology, the asset and the AAS form an I4.0 component [30].

The AAS can be divided into submodules that characterize the asset by describing its aspects in different domains. Examples for such domains are identification, communication, engineering, safety, or security. Moreover, the sub-modules describe the asset's functions, like drilling, milling, welding, or assembling. Each submodule has a set of well-defined attributes and a unique global identifier [29].

For an asset to communicate and interact with information technology (IT) systems, it requires a standardized communication architecture [31]. The Reference Architecture Model Industry 4.0 (RAMI 4.0) includes next to communication several other aspects of I4.0 and has gained broad support in significant companies and foundations [32]. The architecture depicts itself as a three-dimensional cube promoting a Service-Oriented Architecture (SOA) that combines services and data in the sense of I4.0. Each dimension represents different aspects, relevant to establish I4.0. The horizontal integration covers aspects from the product perspective to the connected world, while the life cycle & value stream axis represents lifecycle management covering development, production, and maintenance. The vertical integration spans across the functional layers and tackles different types of interoperability (i.e., business, functional, information, communication, integration, and asset).

## D. Combining CPPM and I4.0

There are different approaches on how to connect the CPPM concept with I4.0 [2]. In this paper, the CPPM is simplified as part of the vertical integration axis of RAMI 4.0 as depicted in Figure 1. Depending on the degree of the asset's existing instrumentation, automation, and communication abilities, the integration can reach up to the functional layer. The AAS supplements the CPPM with additional functionality depending on its integration and acts as its virtual representation. In I4.0, this construct is an I4.0 component [1]. The AAS and the CPPM software system could also be located anywhere on the horizontal integration axis.

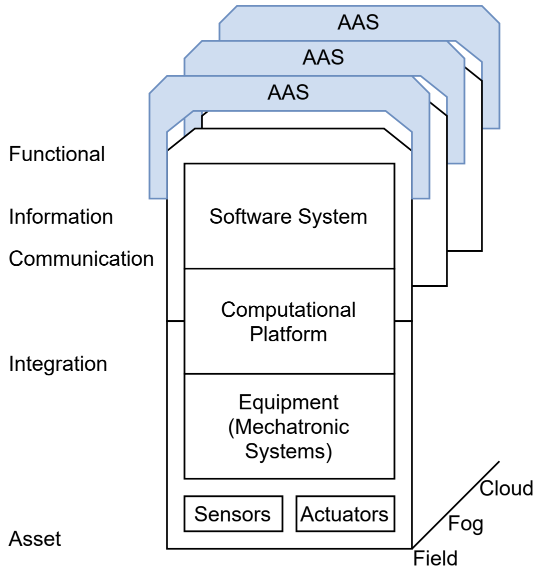

Fig. 1. Connecting CPPM with RAMI 4.0 (I4.0)

An advantage of combining these two concepts is that it shows the importance of communication to enable reconfiguration. For example, moving a piece of equipment to another location, reconfiguring the network and internal CPPM software system is inevitable to ensure that the data reaches the cloud. RAMI 4.0 addresses this circumstance partly in the communication and information layers [32]. The communication layer describes a unified I4.0 transmission mechanism, component discovery, and data format, while the information layer takes care of the transmitted service descriptions and the data model. RAMI 4.0 favors for that task OPC UA and MQTT as communication protocols, as they possess the necessary abstraction and modeling capabilities. However, RAMI 4.0 does not consider (re-)configuration of the communication protocols and assets.

## III. Addressing reconfiguration

Combining the previously mentioned CPPS requirements (II-A) with the identified importance of communication and the internal CPPM software system reconfiguration raises the question of how to address these challenges. A possible first step is to create a simplified arrangement as depicted in Figure 2. The CPPMs (including the AAS) represent any type of equipment. OPC UA is used to connect the CPPMs with a gateway that transfers the data via MQTT to a cloud application. For simplicity reasons, the arrangement assumes that the software part of the CPPM contains an OPC UA server. Moving a CPPM to another gateway requires several changes on the gateways, MQTT and OPC UA simultaneously. This paper proposes a solution that utilizes fog computing and container-based deployment to address the described interplay between the various parts.

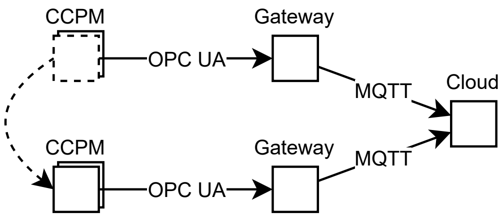

Fig. 2. Simplified reconfiguration

### A. Fog computing

Fog computing brings cloud computing functionality such as virtualization and advanced networking closer to the equipment in the field. While cloud computing locates itself in the higher IT environments, fog computing acts as a bridge to the traditionally separated operational technology [33]. Due to this proximity, fog computing enables applications that would not be possible in the cloud. An essential part of fog computing is the so-called fog node with specific non-functional properties. Such non-functional features include, for example, real-time behavior, reliability, availability, safety, and security. Due to the industrial setting, fog nodes usually consist of verified software and hardware to fulfill industry-specific safety standards (e.g., IEC 61508). The virtualization capabilities of fog nodes build the foundation for the deployment of containerized applications in the field. In combination with TSN [33], containerization enables developers to create distributed applications for industrial automation. Recent research on fog computing focuses on implementing fog computing platforms, tools, implementation strategies, and suggestions for efficient container deployment to handle the limited fog node resources [34].

### B. Container deployment and Kubernetes

For containerization, the open-source project Docker emerged as an enabling technology to build and scale containerized applications. The combination of Docker and a resource management system that deploys the containers to available fog node resources, enables a highly flexible environment as needed for reconfiguration. A prominent example of an orchestration service for the deployment and management of the containerized applications is Kubernetes. The Google-incubated open-source container orchestration tool became the de facto standard for managing large container deployments in cloud environments. Kubernetes supports all types of deployment functionalities, e.g., automatic redeployment by failure or overload. However, the distributed nature, heterogeneity, and limited resources within a fog environment, demand an adjusted approach to benefit from the flexibility. Current research looks into the challenges arising with the deployment and management of IoT applications on fog nodes [15].

### C. OPC UA and MQTT

OPC UA is the successor of the open platform communications (OPC) protocol and is regarded as a new standard to homogenize communication in the industrial domain. The architecture of OPC UA is one of two pillars [35]. The first pillar defines a meta-model to enable information modeling, while the second pillar describes the transport mechanisms responsible for encoding data and exchanging messages between devices. In OPC UA, the primary communication paradigm follows a client-server pattern. However, newer additions to the standard support publish-subscribe (OPC UA PubSub) communication. The client-server mechanism can invoke complex services like browsing the information model and calling methods. Most of the configuration in OPC UA is static, and changes require manual intervention.

In contrast to OPC UA, MQTT is a TCP/IP based publish-subscribe protocol, developed for constrained devices in unreliable network environments [21]. MQTT is built upon topics to which clients can publish data and subscribe for updates. Technically, a client publishes a payload-agnostic message to the MQTT-server (broker), which takes care of the distribution to every subscribed client. The flexibility of MQTT makes it also suitable for IT environments, like other protocols that follow a publish-subscribe paradigm (e.g., Data Distribution Service (DDS)). Bridging OPC UA and MQTT as foreseen in RAMI 4.0 requires gateways that handle the data conversion.

### D. Bringing it together

While fog computing provides the platform for containerization, Kubernetes takes care of the container deployment and the routing. However, the missing link to establish the intended functionality is the combination of container content configuration (e.g., CPPM, MQTT, OPC UA, gateway) and deployment. In essence, the system needs to get all relevant configuration information before deploying and starting the containers. An external entity can take over this role. The process depicted in Figure 3 assumes a configurator to take over this task while receiving the change request by a user. Based on the intended change, the configurator provides configuration files for the container content and Kubernetes. In this case, the user provides the intended changes; however, those could also come from simulation environments previously testing the new configuration. Section IV contains the implementation details.

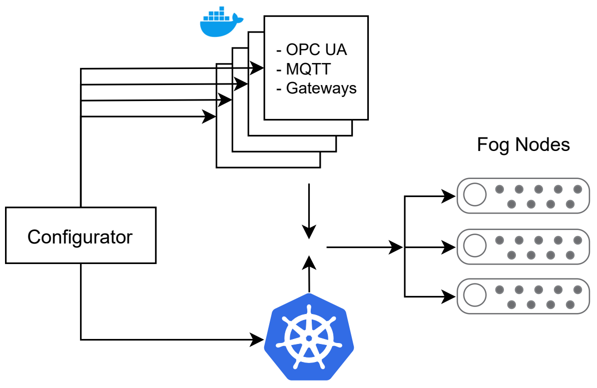

Fig. 3. Docker and Kubernetes configuration

## IV. Evaluation setup

A Kubernetes cluster with Raspberry Pi computers as nodes acts as an evaluation environment to demonstrate the feasibility of the proposed solution. The cluster simulates a fog environment with containerization deployment capabilities. Such a cluster is capable to accommodate several reconfiguration scenarios (e.g., adding/removing CPPMs functionality such as adding a sensor or moving CPPMs to another location).

### A. Gateway architecture

An essential part of the evaluation setup is a bidirectional OPC UA/MQTT gateway, leaning on the Object Management Group (OMG) gateway specification [36]. The gateway processes requests and responses from OPC UA servers and clients on one side, and the other side handles the MQTT topics. Inside the gateway an OPC UA client and server as well as an MQTT client take care of the bridging (cf. Figure 4). A specialty of OPC UA/MQTT gateway is that it consists of two bridges, each responsible for one bridging direction, (i.e., OPC UA to MQTT and versa). The modular architecture allows the adding and removing of bridges if one bridge, for example, reaches its capacity limit. Before starting the gateway, a configuration file specifies the internal gateway mappings and routings. Readers interested to know more about the gateway are encouraged to consult [37].

### B. Configurator and XML files

Configuring the gateway is a complex task that requires following strict structures and naming conventions to ensure the correct functioning of the gateway's services. Included within the configuration are types used by the gateway and connected OPC UA servers, URLs specifying these servers and mappings to define OPC UA nodes and corresponding MQTT topics, on which the gateway will publish changes to before mentioned OPC UA nodes. In order to reduce potential errors, a Python script was developed which generates the necessary XML-encoded configuration files. The configurator enables the creation of these XML files with minimal knowledge about the internal structures of the configuration. For example, in order to map an OPC UA node to the MQTT topic hierarchy, the Python script requires only the following information. First, the OPC UA node and the server which provides the node in its address space; and second, the MQTT topic on which node changes should be published as well as the corresponding type and name of the node's variable. The output configuration contains all internal references necessary for the gateway's mapping process. Additionally, the configurator generates a second configuration file that includes the necessary MQTT environment information so that the gateway can establish a connection to the specified MQTT broker. The git repository provides the entire source code [38].

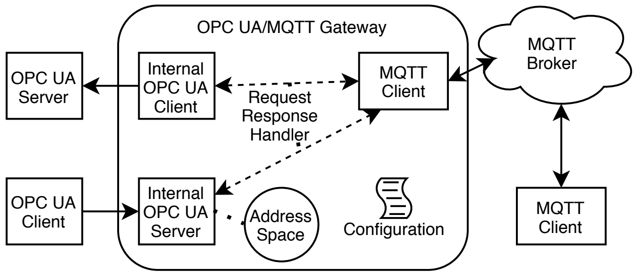

Fig. 4. Gateway architecture overview [37]

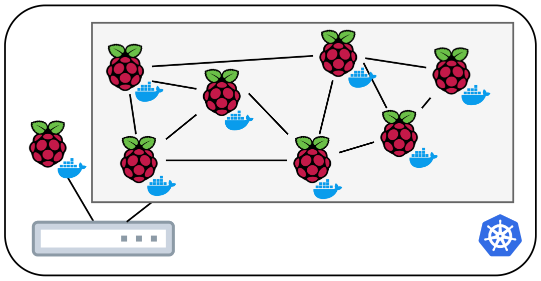

Fig. 5. Raspberry Pi cluster setup

### C. Configuration and service deployment

In case of a deployment, several OPC UA servers and gateways may be involved. These components have to be configured and instantiated on the target computing resource. A Kubernetes cluster is the deployment target, which uses the Raspberry Pi computers for the evaluation setup. The gateways and OPC UA servers are represented by Kubernetes Deployments. Similarly, one Kubernetes Deployment represents the MQTT broker using the official *eclipse-mosquitto* container image. The configuration files are stored in Kubernetes ConfigMaps and linked to their associated Deployments. For allowing the gateways to communicate with the OPC UA servers and MQTT broker, Kubernetes Services provide endpoints for each of the Deployments. In order to ease the description of the deployment, the Python script was extended to generate the description of the whole deployment.

### D. Implementation details software

The gateway implementation uses the SDKs, Eclipse Paho v1.2.1 (MQTT) and Eclipse Milo v0.2.5 (OPC UA) and an XML-encoded configuration file. As part of the evaluation, an OPC UA server provides nodes for attaching the sensors and uses Eclipse Milo v0.2.5 as well [37].

### E. Implementation details cluster

The Kubernetes cluster comprises 24 Raspberry Pi Model 4 single-board computers, of which three are equipped with 4GB RAM and the rest with 2GB RAM. For the evaluation, one 4GB Raspberry Pi orchestrates the other 23 cluster nodes. The Kubernetes K3s (v1.17) distribution was chosen to manage the cluster [39]. A gigabit Ethernet 24-Port MikroTik Cloud Router Switch connects the individual nodes. The 10-gigabit internet up-link of the switch enables fast downloads of container images. Ubuntu 18.04 is used as operating system for each Raspberry Pi.

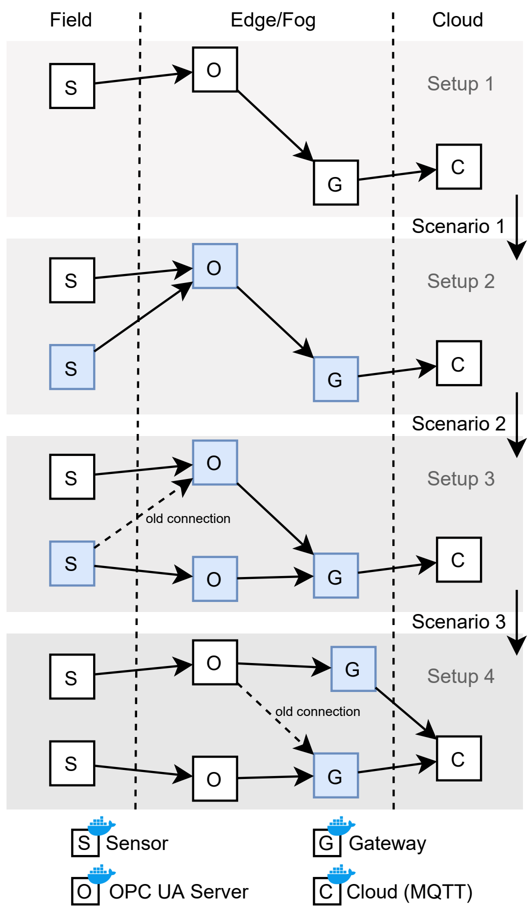

Fig. 6. Setup scenarios

## V. Evaluation Cases

In order to evaluate how a CPPS reconfiguration would impact a running system, the evaluation setup utilizes three scenarios. This evaluation process measures lost messages while doing reconfiguration of the containers within the cluster setup. As a starting point, setup one provides the initial configuration, with a sensor $\boxed{\text{S}}$, one OPC UA server $\boxed{\text{O}}$, an OPC UA/MQTT gateway $\boxed{\text{G}}$ and one MQTT broker $\boxed{\text{C}}$ (cf. Figure 6). After deploying, the sensor writes its measured value to the OPC UA server on fixed intervals. For forwarding this new sensor value to the cloud, the OPC UA/MQTT gateway has a subscription on the relevant nodes of the OPC UA server. Due to this subscription, the gateway will publish any changes made to monitored nodes to the corresponding MQTT topics.

### A. *Scenarios*

The first scenario simulates the adding of components to the cluster setup and builds upon setup one. After redeployment, an additional sensor node of a new type writes values on a fixed interval to the existing OPC UA server. Part of the redeployment involves restarting the OPC UA/MQTT gateway, which will load the required configuration files generated along with the deployment files. Subsequently, the gateway maintains an additional subscription for the node of the OPC UA server to which the new sensor node writes its value. Similarly to setup one, the gateway publishes new values to the corresponding MQTT topics. Figure 6 highlights the configuration changes of the various containers as blue boxes. In addition to the already mentioned configuration change of the gateway's subscription, the new type of the sensor requires changes in the type configuration of the gateway and changes in the OPC UA server configuration. This change is necessary because publishing to the cloud requires strict typing.

Scenario two introduces a second OPC UA server and demonstrates the movement of equipment. Consequently, several configuration changes are necessary. Indicated by the dotted line (tagged as *old connection* in Figure 6), the new deployment intends changing the connection of the second sensor node to the newly created OPC UA server. This change requires removing the unused type information of the first OPC UA server. Additionally, the gateway's configuration receives a change as well, as it has to establish a connection and change its existing subscription to the newly created OPC UA server. Therefore, in order to realize these changes, the gateway restarts alongside the redeployment process.

For moving software components to another fog node, scenario three introduces a second OPC UA/MQTT gateway within the cluster. Additionally, in this scenario, one of the existing OPC UA servers and a sensor node change their location. Therefore, after redeploying the containers and adding the new gateway on another fog node, a restart of the existing gateway for loading new configuration files removes the subscription to the moved OPC UA server.

### B. *Measurements*

Each scenario change requires a redeployment of individual containers and a new set of configuration files. During those changes, some sensor values may be lost during container creation and termination. To measure the number of lost messages, a specifically created Python script monitors all available MQTT topics in the cloud. The script was started before any changes were issued on the cluster to ensure that all messages were logged.

Regarding the container redeployment, Kubernetes first starts a new container instance and waits until it is fully booted before terminating the old one. This process reduces the downtime of containers that require replacement. Additionally, the time it takes to create and terminate can vary. Hence the Kubernetes deployment was extended to log the startup and replace timestamps.

As mentioned, each scenario change requires replacing the existing gateway container, as a configuration reload is necessary. Figure 7, depicts the lost messages during this switch and the timestamps of creation and termination of the gateway container. The vertical lines illustrate when the new gateway container starts and the execution of the replacement process. The gaps of 0,21s and 1,14s respectively indicate the gateway's downtime.

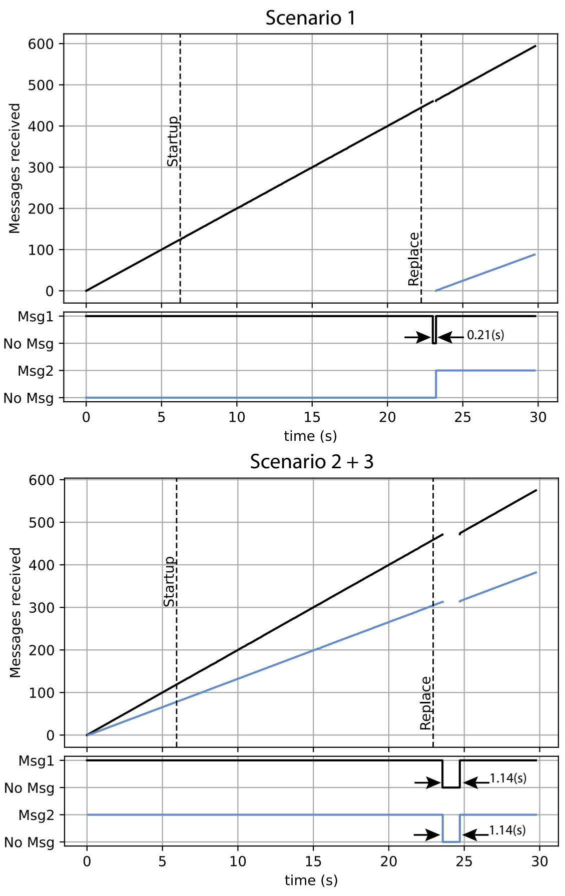

Fig. 7. Scenario measurements

### C. Challenges

The scenarios point out challenges in the chosen solution, mainly when reconfiguring and redeploying the gateway during runtime. Foremost, as the configuration of the current implementation is static, to add or remove subscriptions, a restart of the gateway's container is necessary. The need for restarting introduces the potential for race conditions between the containers, particularly between the gateway and the OPC UA server. If the server is not fully booted before the OPC UA/MQTT gateway, the connection establishment is unsuccessful.

In the current implementation, this would result in a restart of the gateway, as the successful connection to the configured OPC UA server is a requirement. Moreover, such a condition creates more extended downtimes and lost messages. Error handling of containers introduces another challenge. Kubernetes is able to detect if a container enters an internal loop or gets unresponsive, however, the containerized software has to support this feature. Hence there may be a need for external supervision.

Specifically to the gateway, is the challenge that defining the correct and required mappings between a CPPM and cloud applications requires external knowledge. The configurator can provide the correct format, but the knowledge needs to be provided by a domain expert. Important here is that advanced applications that access the sensor data require strict typing from the OPC UA server up to the gateway to avoid misinterpretations.

Apart from the configuration of the Kubernetes deployment, every involved entity, e.g., the fog node, has to have enough computational power and resources to route the messages within reasonable time frames.

## VI. Discussion

The proposed solution to address reconfiguration in CPPS shows promising results but also presents further challenges. Each evaluation scenario represents simplified reconfiguration activities that allow the assessments of the solution regarding the reconfiguration requirements (II-A). The results indicate that requirements of scalability, integrability, and modularity are possible to address. Containers can be scaled, added and removed, changed, and redeployed as long there is a computing platform available. Moreover, the setup is resilient to disturbances, as in case of a failure (e.g., fog node break down), Kubernetes detects the failure and redeploys the container to another fog node. Customization and convertibility are system-level requirements and, therefore, not assessable.

A benefit of the CPPM concept is that the software part could also be separated and hosted on a fog node. Separating the software part would create more deployment flexibility and the possibility to execute more advanced algorithms. A fog node has more computing resources as an embedded system located in a machine. Such functionality is very similar to traditional programmable logic controllers (PLC) environments and conforms to the general ideas of how to use fog computing. A limitation of the proposed solution is that a CPPS assumes that CPPMs are self-contained and interact with other modules without the need for reconfiguration. Such functionality would require auto-configuration or interfaces that provide the CPPM's functions to other modules. In the simplified setup, this is yet not attainable. After deploying the OPC UA/MQTT gateway, it is not possible to add or remove subscriptions, to publish value changes to other MQTT topics, or establish new or change existing connections without redeployment. The gateway's architecture needs to be changed to support auto-configuration or implement an interface that provides the gateway services to others. The changes would require changing the configuration during runtime instead of the static configuration that defines the mappings between OPC UA nodes and MQTT topics. A possibility would be exposing gateway endpoints (interfaces) that allow creating subscriptions to OPC UA nodes, define MQTT topics, and create the mappings between them. With such a change, the gateways internal services handling these endpoints, would create the required instances, e.g., new OPC UA clients or OPC UA data space structures, to send data to the cloud. No restart of the gateway would be necessary for scenario three, and no messages would be lost.

## VII. Conclusion

The paper addresses reconfiguration in CPPS, by combining the CPPM concept with I4.0 and points out the difficulties to configure communication and software parts of CPPMs simultaneously. A potential solution that utilizes fog computing, containerization, and Kubernetes functionality was evaluated in simplified reconfiguration scenarios. The findings indicate the potential of the solution and its challenges while laying the ground for further research that aims to address the CPPMs interfaces. Another aim is connecting the configurator with an information model to generate configurations.

## References

[1] G. Culot, G. Nassimbeni, G. Orzes, and M. Sartor, "Behind the definition of Industry 4.0: Analysis and open questions," *International Journal of Production Economics*, vol. 226, p. 107617, 2020.

[2] S. Karnouskos, P. Leitao, L. Ribeiro, and A. W. Colombo, "Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems: Multiagent Systems Entering Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 14, no. 3, pp. 18–32, 2020.

[3] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable manufacturing systems: Key to future manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, no. 4, pp. 403–419, 2000.

[4] R. M. Setchi and N. Lagos, "Reconfigurability and reconfigurable manufacturing systems: state-of-the-art review," in *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN '04. 2004*, 2004, pp. 529–535.

[5] A. Maffei, "Characterisation of the business models for innovative, non-mature production automation technology," Ph.D. dissertation, KTH Royal Institute of Technology, 2012.

[6] L. Monostori, J. Váncza, and S. Kumara, "Agent-Based Systems for Manufacturing," *CIRP Annals*, vol. 55, no. 2, pp. 697–720, 2006.

[7] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 62–70, 2005.

[8] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: PROSA," *Computers in Industry*, vol. 37, no. 3, pp. 255–274, 1998.

[9] K. Ueda, "A concept for bionic manufacturing systems based on DNA-type information," in *Human Aspects in Computer Integrated Manufacturing*, G. OLLING and F. KIMURA, Eds. Amsterdam: Elsevier, 1992, pp. 853–863.

[10] M. Onori, "Evolvable assembly systems: A new paradigm?" in *33rd international symposium on robotics*, 2002.

[11] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, 2009, distributed Control of Production Systems.

[12] A. M. Farid and L. Ribeiro, "An Axiomatic Design of a Multiagent Reconfigurable Mechatronic System Architecture," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1142–1155, 2015.

[13] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *IEEE Intelligent Systems*, vol. 20, no. 1, pp. 27–35, 2005.

[14] L. Ribeiro and P. Linder, "Hardware Abstraction Layer for JAVA-based agents," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 4896–4901.

[15] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 351–359.

[16] M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner, "Runtime reconfiguration of time-sensitive networking (TSN) schedules for Fog Computing," in *2017 IEEE Fog World Congress (FWC)*, 2017, pp. 1–6.

[17] M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin, and S. Punnekkat, "Self-configuration of IEEE 802.1 TSN networks," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8.

[18] D. Etz, T. Frühwirth, and W. Kastner, "Flexible Safety Systems for Smart Manufacturing," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1123–1126.

[19] J. M. Gutierrez-Guerrero and J. A. Holgado-Terriza, "Automatic Configuration of OPC UA for Industrial Internet of Things Environments," *Electronics*, vol. 8, no. 6, 2019.

[20] M. Bennulf, F. Danielsson, and B. Svensson, "Identification of resources and parts in a Plug and Produce system using OPC UA," *Procedia Manufacturing*, vol. 38, pp. 858–865, 2019, 29th International Conference on Flexible Automation and Intelligent Manufacturing ( FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[21] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, "The Evolution of Factory and Building Automation," *IEEE Industrial Electronics Magazine*, vol. 5, no. 3, pp. 35–48, 9 2011.

[22] L. Ribeiro, "Cyber-physical production systems' design challenges," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1189–1194.

[23] E. Crawley, O. De Weck, C. Magee, J. Moses, W. Seering, J. Schindall, D. Wallace, D. Whitney *et al.*, "The influence of architecture in engineering systems (monograph)," 2004.

[24] N. Chiriac, K. Hölttä-Otto, D. Lysy, and E. Suk Suh, "Level of Modularity and Different Levels of System Granularity," *Journal of Mechanical Design*, vol. 133, no. 10, 10 2011, 101007. [Online]. Available: https://doi.org/10.1115/1.4005069

[25] M. Merdan, G. Koppensteiner, A. Zoitl, and B. Favre-Bulle, "Distributed agents architecture applied in assembly domain," *Proc. 8th Int. Symp. Knowl. Syst. Sci.*, 2007.

[26] H. A. ElMaraghy, "Flexible and reconfigurable manufacturing systems paradigms," *International Journal of Flexible Manufacturing Systems*, vol. 17, no. 4, pp. 261–276, 2005.

[27] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable Manufacturing Systems," *CIRP Annals*, vol. 48, no. 2, pp. 527–540, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007850607632326

[28] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems," *Journal of Manufacturing Systems*, vol. 29, no. 4, pp. 130–141, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612511000021

[29] P. Adolphs, S. Auer, H. Bedenbender, M. Billmann, M. Hankel, and R. Heidel, "Structure of the Administration Shell," April 2016.

[30] H. Bedenbender, U. Billmann, U. Epple, T. Hadlich, M. Hankel, and R. Heidel, et al., "Examples of the Asset Administration Shell for Industrie 4.0 Components - Basic Part ZVEI," *German Electrical and Electronic*, 4 2017.

[31] C. Diedrich and M. Riedl, "Engineering and integration of automation devices in I40 systems," *at-Automatisierungstechnik*, vol. 64, no. 1, pp. 41–50, 2016.

[32] DIN SPEC 91345:2016-04, "Reference Architecture Model Industrie 4.0 (RAMI4.0)," *DIN Deutsches Institut für Normung*, 2016.

[33] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 13–16.

[34] P. Pop, B. Zarrin, M. Barzegaran, S. Schulte, S. Punnekkat, J. Ruh, and W. Steiner, "The FORA Fog Computing Platform for Industrial IoT," p. 101727, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437921000053

[35] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[36] Object Management Group. OPC UA/DDS Gateway Version 1.0. (2020, October 15). [Online]. Available: https://www.omg.org/spec/DDS-OPCUA/1.0

[37] P. Denzler, D. Ramsauer, and W. Kastner, "Tunnelling and Mirroring Operational Technology Data with IP-based Middlewares," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, 2021, pp. 1205–1210.

[38] D. Ramsauer, T. Preindl, and P. Denzler. Configurator Gitlab Repository. (2021, April 29). [Online]. Available: https://git.auto.tuwien.ac.at/fora/opc-ua-dds-gateway-configuration

[39] Cloud Native Computing Foundation. Lightweight Kubernetes K3s. (2021, May 01). [Online]. Available: https://k3s.io/