RealCaPP: Real-time capable Plug & Produce communication platform with OPC UA over TSN for distributed industrial robot control

Christian Eymüller*, Julian Hanke*, Alwin Hoffmann*, Wolfgang Reif*, Markus Kugelmann†, and Florian Grätz†
*Institute for Software and Systems Engineering
University of Augsburg
Augsburg, Germany
†KUKA Deutschland GmbH
Augsburg, Germany
*{eymueller, hanke, hoffmann, reif}@isse.de, †{markus.kugelmann2,florian.graetz}@kuka.de

Abstract—**The industry of tomorrow is changing from central hierarchical industrial and robot controls to distributed controls on the industrial shop floor. These fundamental changes in network structure make it possible to implement technologies such as Plug & Produce. In other words, to integrate, change and remove devices without much effort at runtime. In order to achieve this goal, a uniform architecture with defined interfaces is necessary to establish real-time communication between the varying devices. Therefore, we propose an approach to use the combination of OPC UA and TSN to automatically configure real-time capable communication paths between robots and other cyber-physical components and execute real-time critical tasks in the distributed control system.**

I. Introduction

With Industry 4.0, manufacturing faces disruptive changes. Development and innovation cycles are becoming increasingly shorter while the products need to be individualised to the customers' requests. Hence, production processes need to be more flexible and, at the same time, need to be set up and adapted quickly. This has lead to great research efforts in intelligent manufacturing [1] to cope with these challenges. When considering robotics, especially multi-functional multi-robot cells [2] are promising because manipulators with exchangeable end-effectors can form dynamic production teams based on the needs (e. g., in aerospace production [3] or assembly [4]).

In order to set up such multi-functional robot cells in a fast and reliable way, Plug & Produce concepts [5], [6] such as skills or service-oriented architectures can be applied. Here, OPC UA [7] can be considered the de-facto standard for communication. However, because industrial control and robotics have real-time requirements [8], OPC UA over TSN was introduced [9] which allows time-deterministic communication using the Time-Sensitive Networking (TSN) standards. While in TSN networks the communication partners are known at setup and the communication channels are defined statically, Plug & Produce scenarios are highly dynamic: Devices such as robots or end-effectors can be added or removed at any time, and the communication partners can often change due to new tasks or teams.

This work partly presents results of the project *WiR Augsburg* which is funded by the German Federal Ministry of Education and Research (BMBF) and the Bavarian Government.

This discrepancy must be addressed to realise industrial grade Plug & Produce systems. Hence, it must be possible to reconfigure the real-time production system at runtime [10]. In terms of TSN, new devices and their network interfaces must be configured properly and real-time communication channels must be added or removed appropriately. To address these challenges and enable real-time capable Plug & Produce scenarios, we propose an approach which entirely builds up on the OPC UA communication models and TSN. The contributions of this paper are as follows:

1) We facilitate real-time capable Plug & Produce scenarios by using an OPC UA Discovery Server to register and find TSN capable robotic end devices.
2) We introduce a special TSN Controller to manage real-time communication channels over OPC UA between TSN End Devices.
3) With varying robotics tasks and end devices, these real-time communication channels can be newly established, reconfigured, or terminated during runtime.

Hence, we developed an integrated Plug & Produce communication platform with OPC UA (over TSN) which respects the dynamic nature of Plug & Produce scenarios and fulfils the timing requirements of distributed industrial robot control [8].

The OPC UA communication models, their current use and the OPC UA over TSN extensions are explained in Section II. Our approach for the dynamic discovery of TSN end devices as well as the configuration and establishment of real-time communication channels is described in Section III in detail. While Section IV introduces our reference implementation based on open62451 [9], Section V presents a case study to evaluate the approach. Conclusions are drawn in Section VI.

II. State of the art

As mentioned before, Plug & Produce solutions are only made possible with a uniform and vendor-independent communication standard, such as OPC UA [7]. OPC UA is a service-oriented architecture standard which ensures platform independence, security and information modelling in order to enable interoperability between machines, components and systems [11]. To cover new fields of applications with the basic client-server-model (one-to-many communication) the OPC Foundation extended the OPC UA Specification with Part 14. This adds the possibility of many-to-many communication based on the publish/subscribe (*PubSub*) mechanism without the requirement of a direct and dedicated communication channel between publisher and subscriber [12]. Data Topics define the published data, which are published and subscribed to cyclically [12].

To cover the use case of time-deterministic applications the OPC UA PubSub over TSN approach was introduced by Pfrommer et al. [9]. This approach combines the OPC UA PubSub mechanism with timing guarantees for time sensitive applications provided by the Time-Sensitive Networking (TSN) standard [9]. For the use of this concept there is already an open source implementation in C available in form of the open62451 library [9].

In addition to the previously mentioned challenge of a real-time capable communication channel like PubSub, a real-time capable Plug & Produce control of distributed industrial robots needs further some form of discovery mechanism and registration of newly added devices. The PubSub specification aims to solve this problem by introducing the concept of directories. With these, subscribers can inquire about currently available data. All publishers must register in the directory, whereupon the subscribers can inquire which data are published and which publisher is providing it. This concept is mentioned but not implemented yet. [12] There are also concepts that adress the automatic configuration of TSN networks for the use of OPC UA at runtime [13]. However, in this paper the focus was placed on the description of data topics and the creation of a directory using the OPC UA infrastructure [12].

The specification of OPC UA Discovery Services defines the means by which to discover new OPC UA Servers, their supported protocols and additional capabilities [14]. Three types of discovery are defined in this part of the specification. The first type is the Local Discovery Server (LDS). The LDS organises all information from the applications that have registered directly with the LDS. The second type is an extension of the LDS and is called Local Discovery Server with Multicast Extension (LDS-ME). The last type of discovery is the Global Discovery Server (GDS). The GDS allows applications to search for participants within the administration domain and thus also to find participants outside of their current subnet. [14]

Aside from the discovery and integration of new devices in order to enable a uniform real-time capable Plug & Produce communication platform, the exchange of process information in real-time for running continuous and distributed processes across multiple components is one of the main goals. In [15], we solved this challenge by combining OPC UA Programs [16] and the communication with OPC UA PubSub over TSN [15].

III. Concept

In the context of a real-time capable Plug & Produce communication platform for distributed industrial robot control, the following five technical challenges must be addressed:

1) Discovery of new communication end devices
2) Configuration of the communication channel of end devices and bridges
3) Establishment of the real-time communication channels between multiple devices
4) Real-time transmission of (control) information with OPC UA over TSN
5) Reconfiguration of the system and its communication channels with varying devices

This paper proposes a novel approach to combine the default non-real-time OPC UA communication channels (client-server-communication) with the real-time capable communication channels of OPC UA over TSN (publish-subscribe-communication) to achieve the desired infrastructure.

Because OPC UA PubSub over TSN is the basis for this work, the following terms will be used in the further course of this paper: Network switches which are TSN capable are called bridges or *TSN bridges*. Data published by a publisher is called a *data topic*. These data topics are published and subscribed to. One data topic may be subscribed to by multiple subscribers. Additionally, a distinction between the two types of members in the system is made. There are *TSN End Devices*, which can be for example a sensor, an actuator, a controller or a TSN bridge of a distributed industrial robot control system. The other type of member is the global *TSN Controller*, which is responsible for the registration of new devices and manages the establishment and configuration of real-time communication channels. The TSN Controller serves as an intermediary (broker) for the real-time communication between the TSN End Devices. The default non-real-time OPC UA client-server-communication is used for the configuration of the real-time communication between the TSN End Devices and the TSN Controller.

For example, the task is to synchronise the movement of an industrial robot with another industrial component such as a linear axis. In this case we have a Robot TSN End Device and a Linear Axis TSN End Device which know nothing of each other initially. The aim of the TSN Controller is to bring these components together so that they can perform out the collective task.

A. Discovery & registration of new communication devices

To discover new communication end devices, a customised OPC UA Discovery Server is used. Due to the restriction of TSN to merely one subnet, only the Local Discovery Server (LDS) or the Local Discovery Server with Multicast Extension (LDS-ME) of the OPC UA standard are considered. When a new device wants to join the system, it first has to register with the Local Discovery Server. Therefore two new applicationtypes where added (`TSN_END_DEVICE` and `TSN_CONTROLLER`). The following information is provided during registration: a unique name, a unique address in form of a uniform resource identifier (URI) and the applicationtype of the added device. After the registration of a device, it can query information about other devices from the Local Discovery Server. This makes it possible for a TSN End Device to get information about the TSN Controller and vice versa. Once, the participants of the system are known and information about other devices can be exchanged, the configuration of the TSN communication channels can be initiated.

B. Setup & configuration of TSN communication channels

Basic configuration of newly added devices is necessary in order to establish a TSN communication between the devices. As a precondition of the TSN communication, a time-synchronisation using the precision time protocol (PTP) [17] has to be set up on each device. Each TSN End Device requests a basic TSN configuration from the TSN Controller. This basic TSN configuration contains information about the the global TSN cycle-time and a unique IP as well as multicast address for communication. These unique addresses are assigned by the TSN Controller of the system within the range of a predefined subnet. Once this information has been exchanged, almost all the prerequisites for communication are met. The only thing missing is the allocation of a transmission time slot.

For the transmission of process data in real-time, OPC UA PubSub over TSN is used which is based on the Time Aware Shaper (TAS) [18] of TSN to guarantee minimal transmission latencies. To be able to use the TAS each data topic must receive a fixed time slot for cyclic transmission. The assignment of these time slots for each TSN End Device is the sole prerogative of the TSN Controller. Two operations end in reservations of time slots: adding topics for publishing and subscribing to data. When adding a data topic, only a time slot of the publishing TSN End Device has to be added. Otherwise, by subscribing to a data topic a time slot must be added to the subscribing TSN End Device and to every further TSN End Device that is on the direct communication route between the publisher and subscriber. If the TSN End Devices time slots are already occupied by other data topics the TSN Controller tries to reorder all time slots of all TSN End Devices to find a suitable solution. Even more complex scheduling methods are conceivable here, as described by Raagaar et al. [19]. In case, all time slots have been successfully assigned, the reserved time slots are used for the deterministic transmission of the process data in real-time.

C. Setting up a data topic directory

After the concept for the discovery and registration of new communication devices as well as the setup and configuration of the TSN communication channels has been presented, a concept for a data topic directory is presented. As already mentioned in the OPC UA Specification Part 14 for PubSub [12], a directory for published data is necessary in order to enable other TSN End Devices to find the published data topics and subscribe to them. The directory stores the information about the publisher of a data topic and with which frequency the data is transmitted. The directory is stored centrally at the TSN Controller. If a member needs information on a published data topic it can query the global directory of the TSN Controller. If it needs more detailed information on a data topic it can send a request directly to its publisher. The directory is structured similar to the Discovery Service of OPC UA, so that there is not only one central directory but also distinctions between local directories for devices in the same subnet and global directories for the communication beyond the subnet boundaries. The concept of the directory allows TSN End Devices to search for publishers of data topics which they wish to subscribe to. If the publishers and subscribers are registered with the TSN Controller and the TSN Controller has reserved the required time slots in the defined time cycle, the corresponding data can be transmitted.

D. Transmission of data topics with OPC UA over TSN

Two types of real-time capable transmission of Plug & Produce data topics are provided. The first option is transmission via the default publish-subscribe mechanism. This transmission type can be used for a recurring exchange of information. For example, the provision of a sensor value for the system. The second transmission type is the transmission of data topics by distributed OPC UA Programs [15]. This type of transmission enables the synchronised starting, halting and ending of distributed processes while exchanging process data across multiple participating devices. This type can be used if the synchronisation between multiple devices is necessary. For example, synchronising the movement of two manipulators.

E. Reconfiguration of the system and its communication channels with varying devices

It must be possible to exchange devices or remove them entirely from the system. For example, in robot systems with the possibility of changing end effectors, the robot shall be capable of communicating with newly attached end effectors. In the simplest case, the device only subscribes to data topics, which means it can easily be removed or replaced. If a participating device that publishes data leaves the system all TSN End Devices subscribing to this data topic must be informed that the publisher has left the system. For this reason, the TSN Controller saves all data topic requests so that all subscribers are known for a published data topic. The more complex case is that a leaving device was participating in a distributed OPC UA Program. In this case, there are three possible options:

1) Halt all affected distributed OPC UA Programs (*cancel* a process)
2) Suspend all affected distributed OPC UA Programs, until a new TSN End Device with a similar configuration is added (*pause* a process until the process is operational again)
3) Keep the distributed OPC UA Programs running, in case the participant is not system-critical to operation (*continue* process)

Due to these three possibilities, a termination condition for each distributed OPC UA Program must be specified so that it is clearly defined what happens when a TSN End Device leaves the system.

The presented approach defines the discovery of new devices, the reconfiguration of the system, the configuration of real-time communication channels as well as the real-time transmission for industrial control with OPC UA over TSN. Consequently, all initially defined goals for our approach have been achieved.

IV. Implementation

After the concept of a real-time capable plug & produce communication platform has been presented, an introduction is given to how this approach was implemented. We implemented as much as possible with default OPC UA technologies like OPC UA Methods, OPC UA Events, OPC UA Client-Server-Communication or OPC UA Discovery Servers. The only non-standardised extensions are the concept of distributed OPC UA Programs and communication via OPC UA PubSub over TSN. As basic OPC UA implementation, we used the open source OPC UA library open62541[1]. This also includes the communication via PubSub over TSN. For the integration of distributed OPC UA Programs the standard open62541 library was forked and expanded as shown in [15].

A. Basic architecture of the TSN Controller and the TSN End Devices

First, the implementation of the two main components is presented. Figure 1 shows the activity diagram of the main processes of the TSN End Devices and the TSN Controller and interactions between both system members. First and foremost every member of the system starts an OPC UA Server and registers with the local OPC UA Discovery Server. After the registration, it becomes possible to inquire about devices and set up a non-real-time communication path to these components. Later on, this communication will be used to configure the real-time network. Once, a TSN Controller has registered successfully with the OPC UA Discovery Server, it has two parallel tasks. The first task is to receive incoming registrations of new TSN End Devices, add and reserve time slot arrays for the new devices and send them a basic TSN Configuration. The second task is to cope with removed TSN End Devices. In this regard, the time slot array of the removed TSN End Device is deleted and all TSN End Devices that have subscribed to a data topic of the removed device are notified via an OPC UA Event. The same applies to distributed OPC UA Programs in wich the removed TSN End Device participated. After the receipt of such a notification, the TSN End Device has to proceed according to one of the three defined distributed OPC UA Program termination criteria.

After the basic implementation of the TSN Controller has been described, the implementation of the TSN End Devices is presented. The TSN End Device searches for a running TSN Controller after successful registration at the OPC UA Discovery. In case a TSN Controller is found, a basic TSN configuration is requested. This basic TSN configuration contains information on whether a PTP master has to be started, the basic cycle time of the TSN communication, a unique IP address for the TSN capable network device of the TSN End Device and a multicast address for the OPC UA PubSub communication over TSN. Subsequently, a PTP Master or Slave is started on the TSN End Device and a basic configuration of the TSN communication channels is executed. In this process, the TSN capable network interface is prepared for the transmission of real-time messages by adding appropriate schedulers like the Earliest Tx-time First (ETF) scheduler to the transmission queues (Tx-queue) of the network interface and constructing different virtual networks (VLANs) for the transmission of the real-time data. This ensures that differently classified messages can be sent with different real-time priorities. At the end of the basic configuration of the communication channels, the publishers and subscribers are added by the TSN End Device to the TSN Controller via an OPC UA Method call. The OPC UA Method `RegisterPublisher` of the TSN Controller has the name, the data type and the cycle time of the data topic as input parameter and returns an offset of the reserved time slot in the time slot array of the TSN End Device. The OPC UA Method `RegisterSubscriber` takes the data topic name as input and returns the offset of the reserved time slot for the subscription of the data topic. The call to the method `RegisterSubscriber` can result in the reservation of time slots on multiple devices depending on the number of hops between the subscriber and the publisher. Accordingly, all relevant hops are notified and a corresponding forwarding of the data topic is set up on all affected devices.

B. Implementation of the time slot arrays

The example shown in Figure 2 illustrates how the time slot arrays are implemented. As a first step, the global TSN cycle time is divided into simultaneous time slots. A power of two is used as a divisor of the global cycle time ($divisor = 2^m$). In the example, the global TSN cycle time is divided into 16 time slots ($m = 4$). It is specified that the cycle time of a publisher corresponds at most to the global TSN cycle time and the global TSN cycle time must be divided into power of two time intervals for the publisher cycle time ($cycletime_{publisher} = \frac{cycletime_{global}}{2^n}$ with $\forall n \in [0, m]_{\mathbb{N}}$). These restrictions make it easier to fully allocate the time slots. With new reservations of time slots, the first free time slot in the time slot array is used. As shown in the example Publisher 1 uses the first time

[1]www.open62541.org

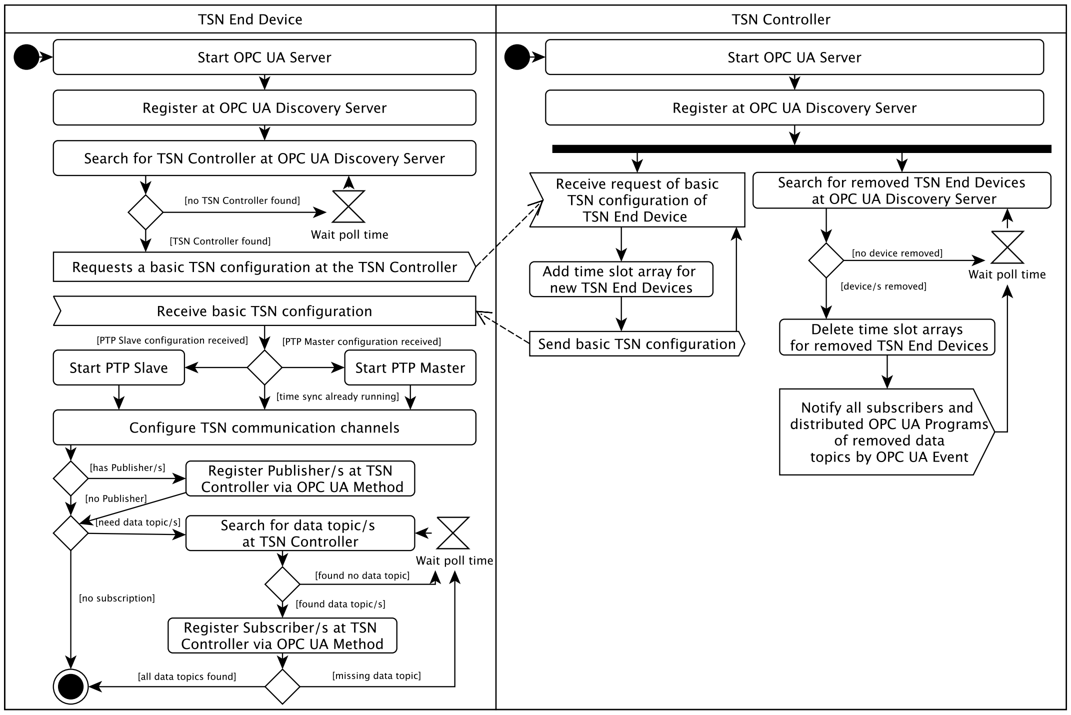
TSN End Device
TSN Controller
Start OPC UA Server
Register at OPC UA Discovery Server
Search for TSN Controller at OPC UA Discovery Server
[no TSN Controller found]
Wait poll time
[TSN Controller found]
Requests a basic TSN configuration at the TSN Controller
Receive basic TSN configuration
[PTP Slave configuration received]
[PTP Master configuration received]
Start PTP Slave
Start PTP Master
[time sync already running]
Configure TSN communication channels
[has Publisher/s]
Register Publisher/s at TSN Controller via OPC UA Method
[no Publisher]
[need data topic/s]
Search for data topic/s at TSN Controller
Wait poll time
[found no data topic]
[no subscription]
[found data topic/s]
Register Subscriber/s at TSN Controller via OPC UA Method
[all data topics found]
[missing data topic]
Start OPC UA Server
Register at OPC UA Discovery Server
Receive request of basic TSN configuration of TSN End Device
Search for removed TSN End Devices at OPC UA Discovery Server
Add time slot array for new TSN End Devices
[no device removed]
Wait poll time
[device/s removed]
Send basic TSN configuration
Delete time slot arrays for removed TSN End Devices
Notify all subscribers and distributed OPC UA Programs of removed data topics by OPC UA Event

Figure 1: Activity diagram of the management of TSN End Devices by the TSN Controller. It shows how the startup routine of the TSN End Device is combined with the main sequence of the TSN Controller

slot and fills in the other required time slots. Publisher 2 now gets the second time slot and accordingly fills in the time slots needed. In case an overlap with other reservations occurs, an attempt is made to redistribute the entire array. Through the defined division of the time slot array, it does not have to be altered for all subsequent cycles.

V. Case Study

The movement synchronisation between a linear axis and an industrial-6-axis-manipulator was carried out as a case study. Figure 3 shows the experimental setup with a spindle linear axis DGE from FESTO and a KR16 6-axis-manipulator from KUKA. Since there are few or no devices with an explicit TSN interface the two devices are controlled by an upstream control unit in form of an industrial PC (IPC) with a TSN network interface. These upstream control units represent the TSN End Devices. The linear axis is controlled by the upstream controller via CAN Open. The 6-axis-manipulator is controlled via the KUKA Device Connector (KDC), an OPC UA interface for KUKA robots. Both IPCs are equipped with an Intel i210 network card and operate with a real-time capable Preempt-RT Linux Kernel (4.19.37-rt20) with ETF-drivers. The two IPCs are connected via a 1 Gbps link for the TSN communication directly with each other. In addition, the two devices are connected to a non-real-time network for exchanging configurations. Another IPC is used as TSN Controller which is only connected to the non-real-time network.

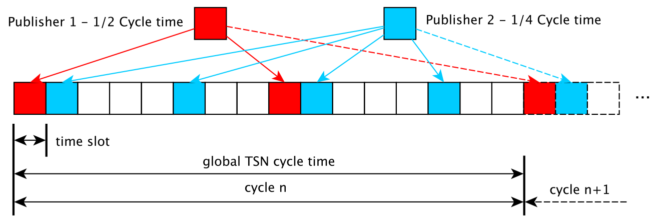

Figure 2: Example of a time slot array

The aim of the two devices is to configure themselves without any pre-configuration, establish a TSN real-time communication and then synchronise their movements. As data basis for the movement synchronisation, only the current position of the linear axis is available as a data topic. This data topic is published with a cycle time of 5 milliseconds. Since the calculation of the robot trajectory relies solely on the current actual position of the linear axis, hard real-time requirements must be satisfied.

After the system has configured itself and has automatically set up a real-time connection between the two upstream controllers, the robot subscribes to the position of the linear axis and starts synchronising the position of its tool centre point to the movement of the linear axis. After successful position synchronisation, no deviation in the movements can be seen at first glance. However, the comparison of the

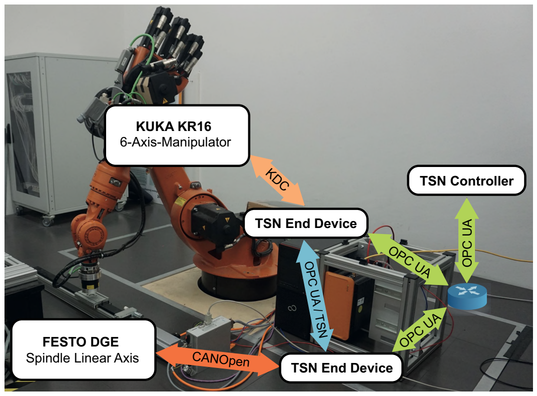

Figure 3: Hardware setup of the prototype implementation

measurements of the current positions of the two devices shows small deviations. These minimal deviations can be explained by possible delays in the communication between the upstream control unit and the controller of the robot via the KDC interface. The measurements of the transmission time between the two upstream control units and between the upstream controller and the linear axis controller show no delay anomalies.

In addition to the evaluation of the synchronisation, the configuration time was measured. Measurements with different settings were executed as shown in Table I. The results showed that the maximum configuration time for the establishment of a TSN communication channel is approximately 24 seconds. While the system is running and the clocks of the devices are already synchronised, publishers and subscribers can be added in less than 7 seconds. It can be seen that the time synchronisation via PTP has a non-negligible influence on the configuration time.

Situation	Min [s]	Avg [s]	Max [s]
Add Publisher, time sync needed	20.1	21.5	23.3
Add Publisher, time synced	6.4	6.7	7.0
Add Subscriber, time sync needed	19.7	20.8	22.4
Add Subscriber, time synced	4.0	4.0	4.0

Table I: Configuration time for different situations in seconds

VI. Conclusion

In this paper we have presented the main components for a real-time capable Plug & Produce communication platform with OPC UA over TSN. This approach shows how the OPC UA infrastructure can be used to achieve the discovery of devices, the establishment and configuration of real-time communication channels and subsequently use these communication channels for a distributed industrial system. The prototypical implementation of the approach has shown that the concepts also work and meets the the technical challenges a functioning distributed real-time system could be set up.

In future experiments, we plan to combine the concept of the general automated TSN configuration as shown by Pop et al. [13] with the concepts of our approach to further combine the advantages of the technologies OPC UA and TSN. In addition, the reconfiguration of systems and the explicit replacement of devices without losing real-time guarantees will be examined.

References

[1] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: a review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[2] A. Angerer, M. Vistein, A. Hoffmann, W. Reif, F. Krebs, and M. Schönheits, "Towards multi-functional robot-based automation systems," in *Intl. Conf. on Inform. in Control, Autom. & Robot.*, 2015.

[3] R. Glück, A. Hoffmann, L. Nägele, A. Schierl, W. Reif, and H. Voggenreiter, "Towards a tool-based methodology for developing software for dynamic robot teams," in *Intl. Conf. on Inform. in Control, Autom. & Robot.*, 2018.

[4] L. Nägele, A. Schierl, A. Hoffmann, and W. Reif, "Multi-robot cooperation for assembly: Automated planning and optimization," in *Informatics in Control, Automation and Robotics, 2019 Revised Selected Papers*, ser. LNEE. Springer, 2021.

[5] M. Onori, N. Lohse, J. Barata, and C. Hanisch, "The IDEAS project: plug & produce at shop-floor level," *Assembly automation*, 2012.

[6] J. Pfrommer, D. Stogl, K. Aleksandrov, S. E. Navarro, B. Hein, and J. Beyerer, "Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems," *at-Automatisierungstechnik*, vol. 63, no. 10, pp. 790–800, 2015.

[7] T. Burke, "OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things," *Opc Foundation*, vol. 1, pp. 01–44, 2017.

[8] A. Hoffmann, A. Angerer, F. Ortmeier, M. Vistein, and W. Reif, "Hiding real-time: A new approach for the software development of industrial robots," in *IEEE/RSJ Intl. Conf. on Intell. Robots and Systems*, 2009, pp. 2108–2113.

[9] J. Pfrommer, A. Ebner, S. Ravikumar, and B. Karunakaran, "Open source OPC UA PubSub over TSN for realtime industrial communication," in *23rd IEEE Intl. Conf. on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2018, pp. 1087–1090.

[10] M. Vistein, A. Hoffmann, A. Angerer, A. Schierl, and W. Reif, "Towards re-orchestration of real-time component systems in robotics," in *IEEE Intl. Conf. on Robotic Computing (IRC)*. IEEE, 2017, pp. 60–68.

[11] "OPC Unified Architecture Part 1: Overview and Concepts (OPC 10000-1)," OPC Foundation, v 1.04, Nov. 2017, https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/.

[12] "OPC Unified Architecture Part 14: PubSub (OPC 10000-14)," OPC Foundation, v 1.04, Feb. 2018, https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub.

[13] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (TSN)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[14] "OPC Unified Architecture Part 12: Discovery and Global Services (OPC 10000-12)," OPC Foundation, v 1.04, Feb. 2018, https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-12-discovery-and-global-services/.

[15] C. Eymüller, J. Hanke, A. Hoffmann, M. Kugelmann, and W. Reif, "Real-time capable OPC-UA Programs over TSN for distributed industrial control," in *25th IEEE Intl. Conf. on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2020, pp. 278–285.

[16] "OPC Unified Architecture Part 10: Programs (OPC 10000-10)," OPC Foundation, v 1.04, Nov. 2017, https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-10-programs/.

[17] S. Schriegel and J. Jasperneite, "Investigation of industrial environmental influences on clock sources and their effect on the synchronization accuracy of ieee 1588," in *2007 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2007, pp. 50–55.

[18] IEEE Standards Association *et al.*, "IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic," *Amendment to IEEE Std*, vol. 802, pp. 1–57, 2016.

[19] 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.