Communication Solutions for the Integration of Distributed Control in Logistics Systems

Giuliano Persico
DEMAG
Wetter, Germany
giuliano.persico@demagcranes.com

Hannes Raddatz
Universität Rostock
Rostock, Germany
hannes.raddatz@uni-rostock.de

Duy Lam Tran
ifak e.V.
Magdeburg, Germany
duylam.tran@ifak.eu

Matthias Riedl
ifak e.V.
Magdeburg, Germany
matthias.riedl@ifak.eu

Paolo Varutti
Thorsis Technologies GmbH
Magdeburg, Germany
pva@thorsis.com

Metin Tekkalmaz
ERSTE
Ankara, Turkey
metin@ersteyazilim.com

***Abstract*—This paper presents a novel decentralized control and communication system targeting diverse set of Material Handling Systems. It is an open platform and based on modular Industrial Device collaboration. The relation of proposed system with Industrial Internet of Things, which have relatively relaxed real time requirements, is also given.**

Keywords—Distributed Control, IIoT, Smart factory

I. INTRODUCTION

Today's internal material flow in logistic systems is only able to cope with challenges such as small batch sizes and increasing product individualization just to a limited extent. A flexible adaptation of the material flow process is time-consuming and expensive due to the hierarchical structure of the central factory control systems. In addition, different types of machines involved in the material flow and, therefore, their control systems have very limited or no communication with each other (Machine-to-Machine communication) due to the complexity of establishing such connections across platforms and applications. Such limited communication also results in the lack of support from machines to the operators in their work processes. Modularization and Machine-to-Machine (M2M) communication provide the basis for tomorrow's smart factories and smart Material Handling Systems (MHS). With the help of the intelligent Human-Machine-Interface (HMI) and the resulting in-depth human-technology cooperation, machines will be able to actively support the operators with "assistance functions" with regard to safety and efficiency.

A trend in industrial automation is to adapt IoT technology to preprocess data or to assist smart manufacturing creating collaborative systems [1] [2]. With IIoT, the control process is not only monitored but also analyzed and optimized. In contrast to control applications, IIoT applications have relatively relaxed real time requirements. Therefore, the non-real-time tasks should be decoupled to avoid the influence on the real-time tasks. This decoupling requires an interface for information exchange between control applications and IIoT applications.

Wireless technologies are preferred in MHS where devices and operators are constantly moving. Common solutions for wireless industrial M2M communication are based on WLAN, [3]. Typical IoT solutions rely, on the other hand, on other wireless communication technologies which are specifically designed for energy-saving, [4]-[5], short distances, [6], or reduced bandwidth, [7]-[8]. Upcoming technologies such as 5G is also considered, which allows also peer-to-peer communication and fulfils industrial requirements like determinism or different kinds of Quality of Services. Control applications need to be designed abstractly from the underlying communication in order to ease the application of new communication technologies.

This article presents a new open decentralized control and communication system that can be used by the most types of MHS – such as overhead forklifts, Automated Guided Vehicles (AGV), overhead cranes and hoists - to enable innovative assistance functions as well as a flexible and fast adaptation of the material flow. Such functions could later be complemented by specific software components running on the same or separate embedded devices without changing central components. Only links for information exchange need to be adapted. Another example is modular components in process industry according to the Namur Module Type Package concept [9]. Here, preconfigured and tested modules are put together for each process. The proposed control architecture also takes into account the interface with IIoT for the exchange of non-real-time data.

The remainder of this paper is organized as follows: Section II briefly presents the system architecture considered for this work. Section III briefly discusses the proposed concept of the Distributed Control Platform. Section IV gives an overview about the communication concepts for control level and Section V gives an overview about interaction with IIoT components running of embedded industrial devices. Section VI presents a primitive implementation of the proposed system. Finally, the last section summarizes the benefits of the presented approach, the related work and gives an outlook about the next steps on improving the found solution.

II. System Architecture

A. Involved Components

The components involved in MHS and their interaction are shown in Fig. 1. The machines are depicted as Industrial Devices (light orange boxes in Fig. 1). Some design aspects of the Industrial Device resemble Cyber Physical System (CPS) definitions [10][11]. A horizontal communication level allows the M2M communication, which may be wired or wireless and shall guarantee real-time data exchange. Additionally, the operator can interact with the MHS through specific HMI (Human Machine Interaction) devices. Furthermore, the components should be able to communicate according to IoT concepts with services running in edge cloud, in external cloud or offer services, which can be used remotely. There are some interesting solutions for such vertical communication either available or currently being specified. Most of such communication protocols use OPC UA [12] technology, e.g. such called Companion Specification developed at VDMA[13] or other associations, e.g. for integration of field device in modern SCADA systems [14]. In this paper, we will focus on the Distributed Control Platform (DCP) (dark blue boxes in the Industrial Device) and the horizontal interaction between software components aimed for control tasks.

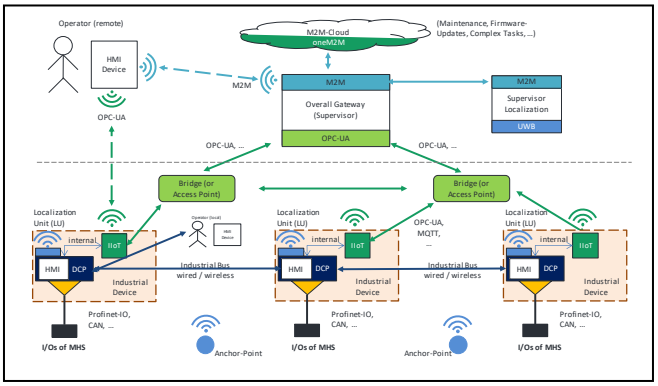

Fig. 1. Interacting Components for HMS or Smart Manufacturing

B. Software Components

Control systems, which have been applied in industrial field, are conceptually designed as centralized control systems. This implies that all inputs (sensors, HMI…) / outputs (actuators) are connected to this central controller. Hereby it is not important whether the inputs / outputs are physically arranged directly on the controller or are arranged decentrally in the automation system via fieldbus systems. The distributed control approach allows to break up the control loop and distribute it to a number of devices. For this, each device is required to be able to interact with physical process via its inputs/outputs, process the data and communicate with other devices. With all these requirements, each device in the distributed control approach can be considered as a Cyber Physical System (CPS).

Each design strategy of the control system has its advantage and limitation. Although central control system provides optimized task schedule for each task by using high performance computing power, it is difficult to satisfy a real-time control as the network size increases. In distributed control approach, real-time control can be guaranteed since each collaborating device fulfils its dedicated sub task of the overall application. However, each device, in order to accurately and safely operate, requires a larger amount of information from surrounding environment and other devices.

Fig. 2 shows the architecture of an Industrial Device. Each Industrial Device performs appropriate set of tasks based on the context. The context consists of current state of nearby components and data from environment, which are obtained via basic communication and embedded or connected hardware such as sensors. DCP is the platform on which, the distributed control applications can be deployed. It also facilitates the interactions between the control applications across devices. IIoT is responsible for vertical communication as well as the exchange of non-real-time data. The Industrial Device should have specific software components depending on the requirements. Thus, not all components shown in Fig. 2 have to be available in all device instances. Therefore, for instance, an edge cloud device may behave as an Industrial Device following the approach of this article.

The design of control application shall follow the object-oriented approach. Function Blocks are also well known and widely used in automation domain. The latest version of IEC 61131-3 [15] allows Function Blocks with high degree of abstraction, e.g. by introducing the interface concept or multiple methods per Function Block. Control Application Objects (CAO) adapt such design principles but the DCP runtime allows remote interactions between CAOs as well.

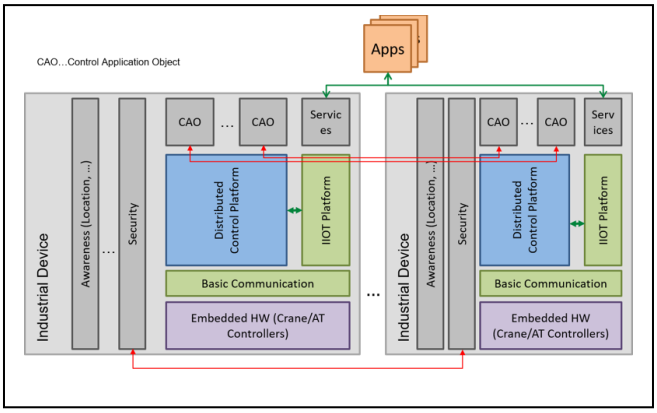

Fig. 2. Architecture of Industrial Device

Different software components running on an Industrial Device shall be deployed based on container approaches.

III. Architecture of DCP

In this section, the architecture of the DCP is presented more in detail. For the realization of the DCP, it is assumed that an operating system (OS) is available, where the DCP and its component can be installed. This may differ from hardware to hardware platform. Quite often OS with real-time features are used for embedded industrial automation devices. Fig. 3 shows the architecture of DCP. Central component of the DCP is the Node Manager. Each DCP provides exactly one Node Manager. It is the first contact point for each communication

channel established between Control Application Objects on the same node or across different nodes.

The Control Application Manager is responsible for managing tasks between Control Object Managers, especially the functionality of distributed start, stop, halt or continuation of the overall functionality, whereas the Control Object Manager handles the lifecycle of all Control Application Objects. Scheduling strategies are required for Control Object Manager to handle multiple Control Application Objects, which are running concurrently. CAOs are further grouped into execution contexts. The Control Object Execution is responsible for triggering the CAOs.

One of the main challenges in a distributed system is time synchronization. Different nodes in the network may use different time references. Therefore, the control platform provides clock synchronization to ensure that time is aligned between all nodes in the control network. In case of Ethernet based communication between the nodes, clocks can be synchronized according to IEEE 1588 (Precision Time Protocol - PTP) [16].

Logging is a very important service in different life cycle phases of the control application. The Logging Server offers services to log for instance events, Control Application Object activations or changes of the execution point of view.

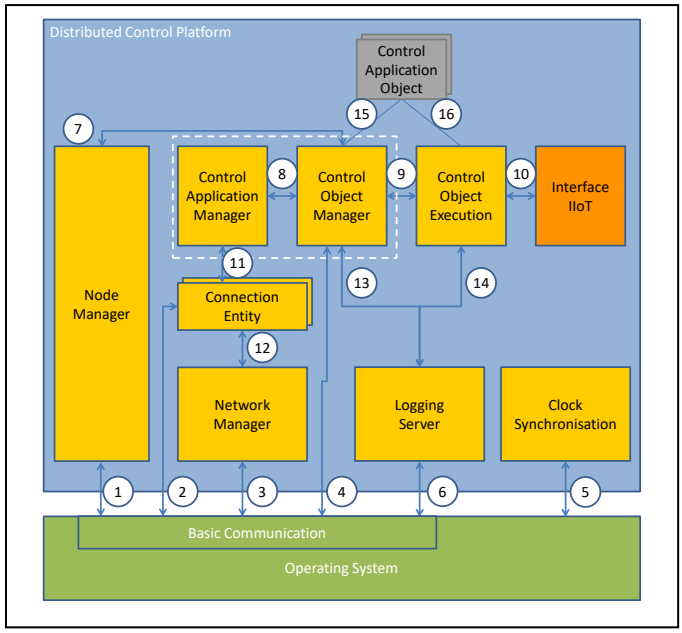

Fig. 3. Architecture of Distributed Control Platform

Generally, each component in DCP offers several services via specific interfaces. The detailed specification cannot be given in the context of this article. Examples of services are for creating / deleting of CAOs or starting / halting / stopping of the application.

The remaining components Network Manager, Connection Entity and Interface IIoT are discussed in subsequent sections.

IV. Communication between DCP Objects

As shortly mentioned in Section I, communication between Industrial Devices shall be independent from the specific communication technologies. From DCP perspective, the availability of communication interfaces offered by the OS, e.g. by means of sockets, is also important. Based on such abstraction, the network component of the DCP can establish a communication channel between CAOs running on different devices. This results in peer-to-peer communication between Industrial Devices, if the specific control application needs such information exchange. Fig. 4 shows an established communication channel between two partners.

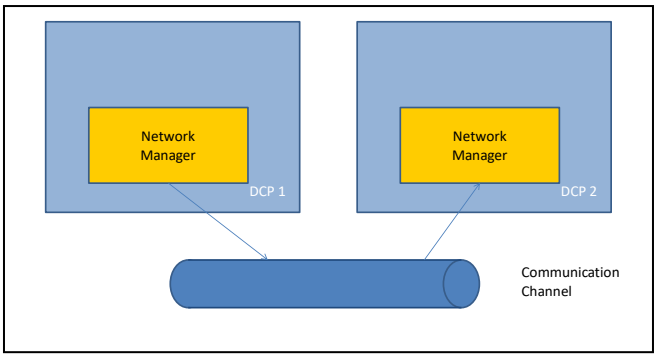

Fig. 4. Communication Channel as Basis for Device Interaction

The setup of a communication channel uses lookup functions for discovery purposes, e.g. to find the correct Industrial Device. In most cases, Industrial Devices are defined by their names. Here the approach follows well-known IT solutions as Domain Name System (DNS). Internally the Network Manager consults the Node Manager of the DCP instance.

Fig. 5 shows an application example from engineering perspective consisting of six CAOs. These objects are distributed to two Industrial Devices each with its own DCP instance. In the example, the CAOs are further distributed into two execution contexts, from OS point of view these may be processes. A CAO provides interfaces for the communication with other CAOs. The interfaces are called ports. Depending on the purpose of the port, the interaction may be blocking or non-blocking. In case of blocking interaction, the stimulating CAO waits internally up to completion of the stimulated / invoked service of another CAO. This is similar to a Remote Method Invocation know in other programming languages like Java [17] or C# [18]. In blocking interaction, the invoked service may send return values or output data. The proposed concept can be seen as adaption of Real-Time Object-Oriented Method (ROOM) or the UML-RT concepts of capsules, ports and connectors [20].

The DCP is responsible for establishing the engineered communication between the CAOs during runtime. Thus, links will be established. Depending on the context of a CAO, DCP creates the links for local interaction or for remote interaction accordingly. In each process, CAOs can be grouped into sub-contexts depending on the type of CAOs, which are active CAO (time-triggered) and normal CAO (event-triggered). The interactions between CAOs of the same

sub-context are simply method calls. In case of interactions between CAOs of different sub-contexts, the method calls need to be marshalled and registered to the sub-context of the invoked CAO. First-In-First-Out (FIFO) strategy is used to decide which message is handled first. In case of remote interaction, the described above communication channel is used. The Connection Entity, shown in Fig. 6, creates the remote links and uses the established communication channel for the real data exchange. In Fig. 6, two links share the same communication channel. At higher layer, the service calls are handled similar to the case of local interaction between CAOs of different sub-contexts.

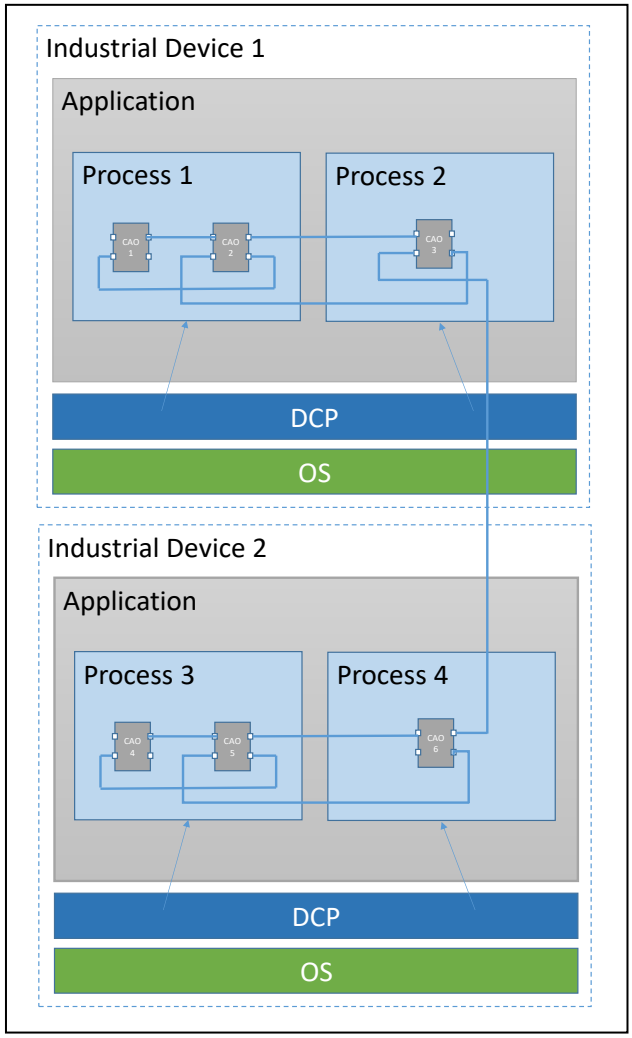

Fig. 5. Interaction between Control Application Objects

By means of this concept, special communication channels could be established to fulfil dedicated QoS or to be used for redundancy.

The communication protocol used by DCP is designed as a protocol at application level, independent from the lower level protocols, which provide a specific header and a payload for the data. DCP messages are embedded into the payload of the underlying protocols e.g. TCP. DCP messages are categorized into request and response messages. The structure of the message for the interaction between management components in DCP is defined as following:

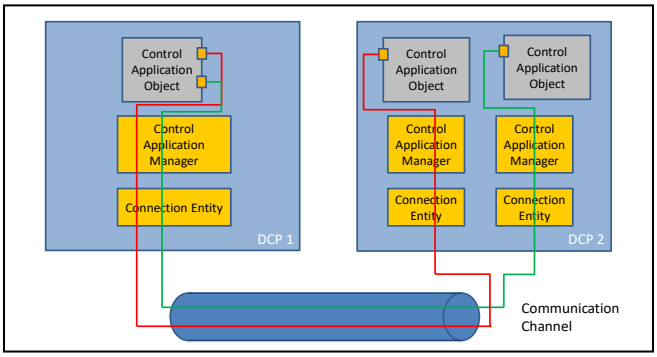

Fig. 6. Links sharing the Communication Channel

The request contains a Transaction Number (TAN) to identify a transaction, see Fig. 7. After the transaction number is the length of the data of the request. Each request contains an Opcode for identification of the required service. Opcode from 0x0000 to 10 (0x000A) are reserved for response (0x0000) and for other special messages. The rest is used for other service requests.

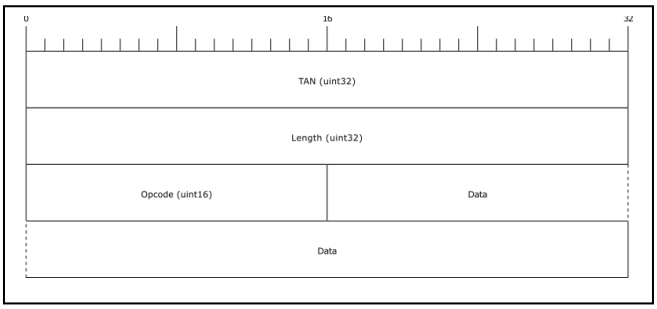

Fig. 7. DCP message for service request

TABLE I. shows the value scheme for the opcode in DCP. Depend on the request, Data part can be empty or can contain the required data. In DCP, the data is converted to network byte order before being sent over the network. For a string, the length of the string is inserted into the message before the content of the string.

TABLE I. OPCODES USED IN DCP

Value	**Description**
0x0000	Response
0x0001	Interface info request
0x0002	Remote connection status
0x0003	Connection echo
0x000A-0xFFFF	Service requests

The response message uses the same TAN number as the corresponding request, see Fig. 8. The length of the data of the response is specified next to the TAN. The Opcode of all responses have value 0. The status of the response is shown in

the Exception type. TABLE II. shows the list of exceptions used in DCP. Depending on the request, the response can be merely an acknowledgment or can contain the required data. In case there is an exception, the returned data contains a string, which gives detail about the error.

The communication between CAOs for exchanging data is based on the presented structure above. For such communication it must be differentiated, if the communication between CAOs is blocking or non-blocking. The blocking interaction requires that a return message is sent back to the sender.

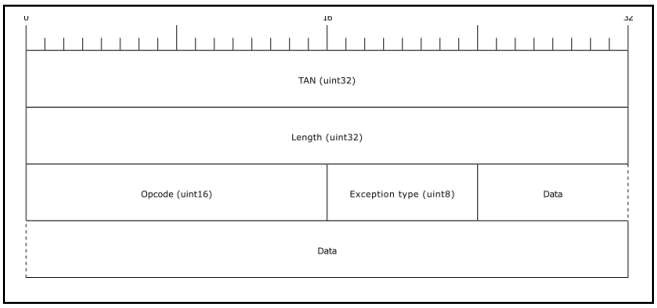

Fig. 8. DCP message for service response

TABLE II. EXCEPTION TYPES

Value	Description
0x00	No exception
0x01	Runtime error
0x02	Logic error

The first 16 octets of the data request message are the same as a configuration / management message, where a transaction number TAN and length of the data are presented. The opcode of a message is always 0x000E. Each remote communication is assigned with a Link handle when the link is created. This handle is included in the request message for the receiver to select the corresponding receiving endpoint (port) of the remote link, see Fig. 9.

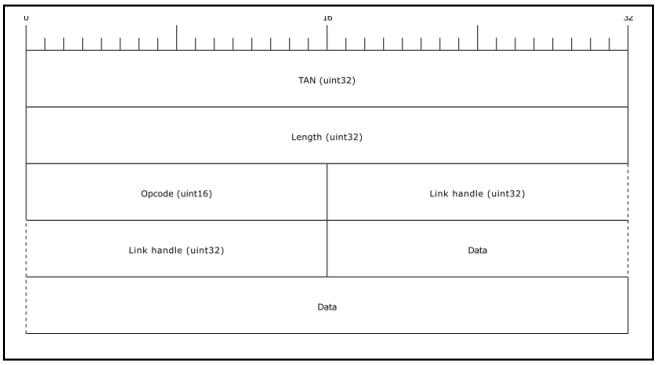

Fig. 9. DCP message for CAO data sending

Each port is defined with data (parameters from point of view of a function call), which needs to be transmitted. This data is embedded after the Link handle in the Data section. If the data is a structure, each structure element is inserted subsequently in the Data section of the message. The encoding of the data is the same as described above. On the other side, the port of receiving CAO is defined with the same data type, therefore the message will be decoded accordingly.

The CAO response message is similar to the service response message described above, where Opcode has value 0 and Exception type has value as in TABLE II. The Data section contains the data corresponding to the return data type defined in the CAO port.

V. COMMUNICATION BETWEEN DCP AND IIOT

The communication between CAOs in DCP can be classified as event driven RMIs. The coupling between CAOs more or less tight regarding to the real-time constraints of the automation application. Often also, data should be provided for vertical information flow in plant hierarchy. Here normally no real-time requirements has to be fulfilled and such data transfer should not influence the control application. Therefore, real-time and non-real-time tasks shall be decoupled. This decoupling will be done by means of the component Interface IIoT as shown in Fig. 3. The communication should be as lightweight as possible. Performance measurements show, the lightweight communication approaches such as MQTT [21] can be used for such decoupling.

CAO can publish data to MQTT Broker via its output port. Messages from CAOs can vary from simple type (integer, float, string…) to complex type (array, struct). The data from CAOs are then encoded using format that IIoT applications can decode. A text-based format is preferable in this case e.g. JSON. The JSON object is then converted to JSON string before publishing to broker.

In addition, CAOs may also provide properties. This can be seen as public member variables in object oriented languages. Properties allow dynamic adaptions of CAO behavior. Thus, properties of CAO may also be published. Important for publishing is a topic name. The topic name can be defined during engineering. If not defined, a topic name is generated automatically and consists of the instance name of the Industrial Device, the CAO instance name and the port name, see TABLE III.

TABLE III. TEMPLATE BUILDING TOPIC NAMES

Template name	Function	Default value
Instance template	DCP specific part	%I = Name of the process which contains the CAO where the port/property is defined
Object template	CAO specific part	%O = Name of the object where the port/property is defined
Port template	Port/Property specific part	%P = Name of the port/property

In addition, topics can also be subscribed. Here also both variants – automatic or engineered – subscription will be supported.

VI. Implementation

First implementation exists for a demonstration of two cranes see Fig. 10. In this demonstrator, Industrial Devices are two IPCs, each of which controls one crane. Each IPC is equipped with Linux based OS and physical I/O systems. The DCP is ported to embedded hardware. Control application is developed to run on top of DCP, which implements tandem and come-to-position function. In tandem function, the control signals come from a pendant switch via CAN interface to one IPC. The control commands are then forwarded to the other IPC so that two cranes have the same movement. For automatic function e.g. come-to-position function, a joystick is employed in order to provide more inputs to the control application i.e. to change mode (manual/auto) and to trigger come-to-position movement. A Raspberry PI 3 is used to get data from the joystick since the IPC does not provide USB interface. DCP is also ported to the Raspberry PI, so that data can be transferred and interpreted by the control application on the IPCs.

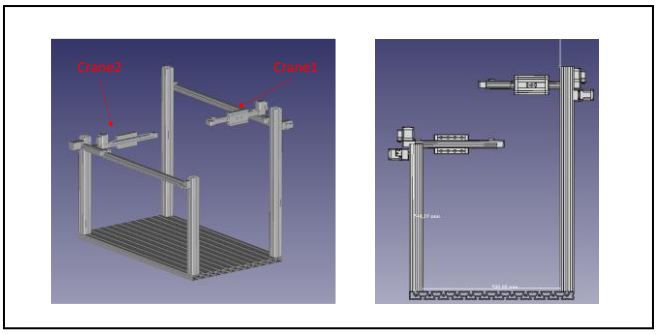

Fig. 10. Demonstrator of MHS with two cranes

VII. Conclusions and Related Work

This paper discussed an approach to provide solutions supporting distributed application for control tasks. First implementation showed that the proposed architecture is able to satisfy the real-time requirement for the control task, especially in tandem function where two cranes are able to move concurrently without any significant delay. As visible in Fig. 2, further points need to be considered for the DCP. Firstly, IIoT platform is required in order to evaluate the interaction with the DCP. Secondly, security aspects shall be considered. Furthermore, the concept must take particular account of practicability, because the concept must be implemented in engineering tools and maintenance personnel has to handle it at operation of the automation system. Safety concepts are not explicitly mentioned, but they are important for the interaction of automation applications. Unfortunately, safety solutions are more complex and cannot be solved by research projects. Current implementation of DCP is based on Ethernet, which does not provide prioritization on the transmitted data. This can be solved by using Time-Sensitive Network (TSN). With the use of TSN-capable Ethernet hardware and the integration of TSN in Data Link layer, prioritization on transmitted data can be controlled by application and this feature will be considered for DCP in the future.

Acknowledgment

The authors would like to thank ITEA 3 and the National funding authorities: The Federal Ministry of Education and Research; Ministerio de Economia y Competitividad; TÜBITAK; Korea Institute for Advancement of Technology for their support, and the partners of the ITEA 3 project OPTIMUM - OPTimised Industrial IoT and Distributed Control Platform for Manufacturing and Material Handling (https://www.optimum-itea3.eu/) for their work and contributions that enabled this paper.

References

[1] D. Zuehlke, “SmartFactory-Towards a factory-of-things,” Annual reviews in control, vol. 34(1), pp. 129–138, 2010.

[2] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, “Industrial internet of things and cyber manufacturing systems,” In Industrial Internet of Things, pp. 3–19, 2017.

[3] Hascher, Wolfgang: Wireless LAN auf einen Blick. In: Elektronik Scout 2008, 2008

[4] Lora Alliance: https://lora-alliance.org/

[5] IETF, “IPv6 over Networks of Resource-constrained Nodes (6lo)”

[6] PaloWireless: Bluetooth. http://www.palowireless.com/bluetooth, 2008

[7] Kupris, Gerald; Kremser, Hans-Günter: Reichweitenuntersuchungen in ZigBee-Netzwerken. In: D&E ZigBee&Co-Entwicklerforum, 2005

[8] Kupris, Gerald; Sikora, Axel: ZigBee. Franzis Verlag, 2007. – ISBN 978–3–7723–4159–5

[9] Bernshausen, J. et al. Namur Modul Type Package – Definition. atp magazin, [S.l.], v. 58, n. 01-02, p. 72-81, Jan. 2016. ISSN 2364-3137. Available at: <http://ojs.di-verlag.de/index.php/atp_edition/article/view/554>. Date accessed: 08 aug. 2019. doi: https://doi.org/10.17560/atp.v58i01-02.554.

[10] M. Broy, “Cyber-Physical Systems - Innovation durch softwareintensive eingebettete Systeme“, ISBN 978-3-642-14901-6, Springer, 2010

[11] John Eidson, Edward A. Lee, Slobodan Matic, Sanjit A. Seshia, Jia Zou, “Distributed Real-Time Software for Cyber-Physical Systems”, Proceedings of the IEEE (special issue on CPS), Vol. 100, p 45 – 59, January 2012

[12] OPC Foundation: OPC Unified Architecture Specification, https://opcfoundation.org/developer-tools/specifications-unified-architecture

[13] VDMA: OPC UA Companion Specifications for Robotics and Machine Vision released, https://industrie40.vdma.org/en/viewer/-/v2article/render/26418188

[14] Großmann, D.; Braun, M.; Danzer, B.; Kaiser, A.; Riedl, M.: FDI - Field Device Integration, Handbook for the unified Device Integration Technology, ISBN 978-3-8007-3630-0, E-Book: ISBN 978-3-8007-4010-9, 2016

[15] IEC TC65/WG6: IEC 61131-3: Programmable controllers Part 3: Programming languages, 3rd Edition, IEC, Genf, 2014

[16] The Institute of Electrical and Electronics Engineers, Inc., IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE Std. 1588–2002. ISBN 0-7381-3369-8, New York 2002

[17] M. Rouse, “Remote Method Invocation (RMI)”, https://www.theserverside.com/definition/Remote-Method-Invocation-RMI

[18] “R. Wiener, “Remoting in C# and .NET”, Journal of Object Technology, http://www.jot.fm, Vol. 3, No. 1, January-February 2004

[19] Selic, B., Gullekson, G., Ward, P. T.: Real-Time Object-Oriented Modelling, John Wiley & Sons, 1994

[20] Pezzé, M.: Fundamental Approaches to Software Engineering, 6th International Conference, p. 121-122, FASE 2003 as part of ETAPS 2003, Warsaw, Poland, April 2003, Proceedings

[21] OASIS:, "Message Queuing Telemetry Transport (MQTT)", https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt