Open communication protocols for integration of embedded systems within Industry 4

Peter Peniak
University of Žilina
Faculty of Electrical Engineering, Department of Control and Information Systems
Slovak Republic

Maria Franekova
University of Žilina
Faculty of Electrical Engineering, Department of Control and Information Systems
Slovak Republic

***Abstract* – This article deals with Industry 4 as a new paradigm for manufacturing of the future. Internet of Things and new communication approach can lead to next generation of automated solutions as next step from currently isolated automation systems to Cyber-Physical Systems. Article tries to create model for DDS protocol considering real time requirements with QoS features.**

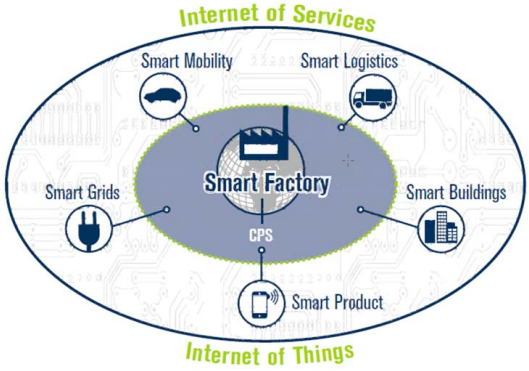
Industry4; CPS; IoT; IoS; SOAP; DDS;DCPS

I. INTRODUCTION

Industry 4.0 is „the introduction" of the Internet of Things and Internet of Services into the manufacturing environment [1], [2], [3]. In the future, businesses will establish global networks that incorporate their machinery, warehousing systems and production facilities in the shape of Cyber-Physical Systems (CPS) [4].

CPS systems are integrations of computation and physical processes. Embedded computers and networks then monitor and control the physical processes, usually with feedback loops, where physical processes affect computations and vice versa the technical integration of CPS into manufacturing and logistics requires the use of the Internet of Things and Services in industrial processes, as shown on Figure 1. This means „smart production facilities, smart grid, smart buildings, smart machines, logistic, and mobility are capable of autonomously exchanging information, triggering and controlling each other independently".

Figure 1. Smart Factory according to Industry 4



In smart factories, smart products are uniquely identifiable, may be located at all times and know their own history, current status and alternative routes to achieving their target state This will revolve around networks of manufacturing resources (manufacturing machinery, robots, conveyor and warehousing systems and production facilities) that are autonomous, capable of controlling themselves in response to different situations, self-configuring, knowledge-based, sensor-equipped and spatially dispersed and that also incorporate the relevant planning and management systems. Smart factories will make the increasing complexity of manufacturing processes manageable for the people who work there and will ensure that production can be simultaneously attractive, sustainable in an urban environment and profitable.

Smart products will be able to control the individual stages of their production semi-autonomously. Finished goods know the parameters within which they can function optimally and are able to recognize signs of wear and tear throughout their life cycle [5].

II. NEW COMMUNICATION ARCHITECTURE

Despite long-term standardization process with Industrial Ethernet and TCP/IP protocols, there are still proprietary protocols used from various vendors. Especially in embedded environments, many other communication middleware technologies simply do not fit. With very constrained memory, and often with specialized processors and networking hardware, writing customized code using sockets has been the preferred solution to embedded communications [6].

Internet of Things and Internet of Services are key-building blocks for Industry 4 [7], [8]. Devices must be able to communicate with each other via Internet protocols (TCP/IP), taking into consideration appropriate security level and support of real-time industrial applications with specified Quality of Service. Industry 4 assumes that not only device capabilities and direct communication is supposed, but overall production process control, storing of

ISBN 978-80-261-0386-8,

generated data (big data) and fast reaction time and decisions can be taken during production process as result of predictive algorithm and automatic evaluation. New communication architecture has to support the various requirements from different systems, therefore different protocols specialized for given scope are to be considered, due to implementation constrains or available device intelligence. IoT protocols are expected to generate large amounts of data from diverse locations that is aggregated very quickly, therefore increasing the need to better index, store and process such data. Device data then must be collected and sent to the server infrastructure. That server infrastructure has to share device data, possibly providing it back to devices, to analysis programs, or to people. The following IoT protocols are used:

- D2D (Device to Device) protocols for independent communication of smart devices.
- D2S (Device to Server) –protocols for transport of collected data from devices to server infrastructure and vice versa.
- S2S (Server to Server) –protocols for inter-communication of applications on servers.

New communication architecture is illustrated on Figure 2. Open and independent communication of end devices is allowed by means of D2D protocols. The devices can exchange data without limitation and dependency on central gateways (brokers). In addition, all collected data can be delivered from end-devices to applications via D2S protocols, for next processing and final transformation to information (big data). Service oriented architecture is proposed to exchange data and information among distributed application via S2S protocols, such as SOAP. Typical protocols that are currently used for IoT are listed in Table 1, including mapping of protocols to defined classes (D2D, D2S, S2S).

Figure 2. New communication architecture for IoT

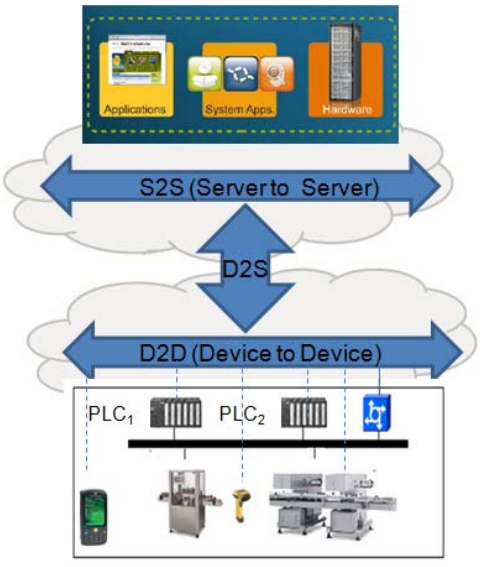


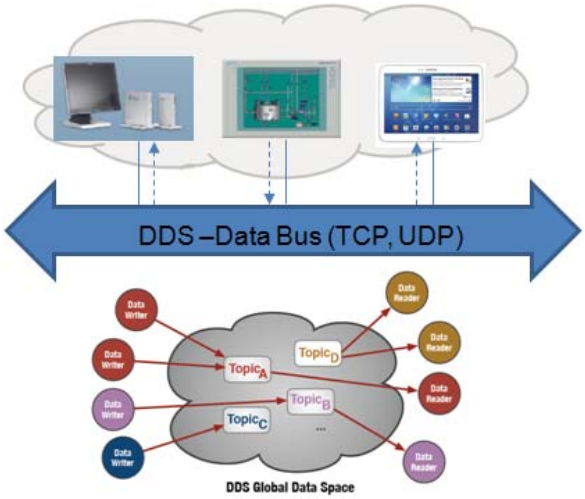
TABLE I. INTERNET OF THINGS -PROTOCOLS

No	IoT protocols		
	Protocol	*Class*	*Remark*
MQTT	Message Queue Telemetry Transport	D2S	Broker based
DDS	Data Distribution Service	D2D	Data bus based
XMPP	Extensible Messaging and Presence Protocol	D2S	Messaging
AMPQ	Advanced Message Queuing Protocol	D2S	Messaging with Queuing
SOAP	Service Oriented Architecture Protocol	S2S	Web Services

III. DDS PROTOCOL AND DCPS MODEL

DDS uses the Data-Centric Publish-Subscribe (DCPS) model, based on the concept of a "global data space" that is accessible to all interested applications. The application can write data to data-bus via specific service called "Data writer". Data are stored in specified data objects, which are addressed by unique identifier, called "Topic". Data, which are written to data objects, can be read by other nodes via "Data Reader" service, based on specified "Topic" id. The basic concept of DDS Data bus is shown on Figure 3.

Figure 3. DDS –Data bus system



Even though DDS models all interactions as reads and writes to the global data space, data flows directly from publishers (producers) to subscribers (consumers). There is no broker. Instead, publishers and subscribers connect over a "databus." this data space declares their intent to become "Publishers." Similarly, applications that want to access portions of this data space declare their intent to become "Subscribers." Each time a Publisher posts new data into this "global data space," the middleware propagates the information to all interested Subscribers. Underlying any data-centric publish subscribe system is a data model. This model defines

ISBN 978-80-261-0386-8, © University of West Bohemia, 2015

the "global data space" and specifies how Publishers and Subscribers refer to portions of this space. The data-model can be as simple as a set of unrelated data structures, each identified by a topic and a type. The topic provides an identifier that uniquely identifies some data items within the global data space. The type provides structural information needed to tell the middleware how to manipulate the data and also allows the middleware to provide a level of type safety. However, the target applications often require a higher-level data model that allows expression of aggregation and coherence relationships among data elements.

DPCS model is shown on Figure 4. DDS creates filtering capabilities that can be used to separate end-devices each other based on logical domains. Domain is logical area of data-bus, where defined data objects are grouped for filtering purposes based on assigned Topic-id$_s$ (1):

$$Domain_i \sim \{Topic_{i1}, Topic_{i2}, \ldots, Topic_{in}, \} \quad (1)$$

In addition domains are used to associate defined entities of publishers and subscribers, which are selected for exchange of data among end nodes, with defined data objects and assigned Topics, as expressed in formulas (2-3):

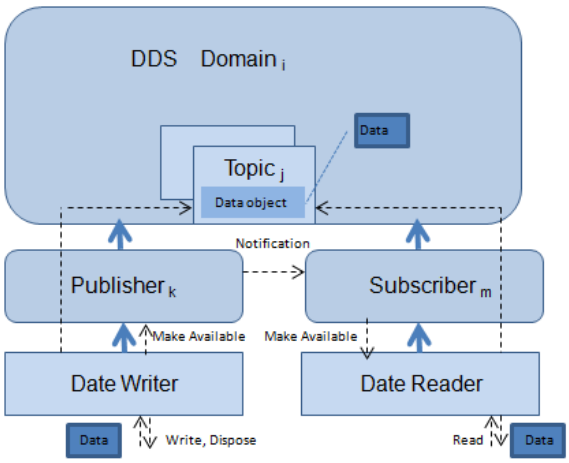
$$Domain_i \sim \{Publisher_{i1}, Publisher_{i2}, .., Publisher_{ik}\} \quad (2)$$

$$Publisher_k \sim \{DataWriter_{i1}, DataWriter_{i2}, \ldots\} \quad (3)$$

$$Subscriber_m \sim \{DataReader_{i1}, DataReader_{i2}, \ldots\} \quad (4)$$

The proposed DPCS model is shown on Figure 4:

Figure 4. DDS –DPCS model



The communication between applications that wants to publish data via DDS protocol is illustrated on Figure 5. An application creates a publisher for an assigned domain via DDS services. When the publisher is created by DDS services, the application can request assignment of a "Data writer" to the created publisher. Since then, application is able to write data to "Data writer". As soon as data is written to data object, which is associated with Topic, publisher notifies other nodes about new data. Figure 6 shows process of DDS subscriber. Application demanding reading of data via DDS protocol has to create entity of subscriber for given domain and associate "Data reader" to it. After "Data reader" had been created, the process of reading data could start as well. When a notification is triggered concerning update in a data object with assigned Topic-id, the subscriber sends information to "Data reader" about event. Afterwards the listener process is notified that reading can start.

Figure 5. –DDS Publisher process

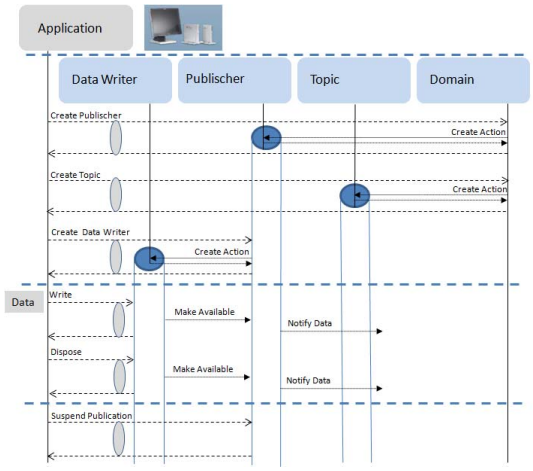
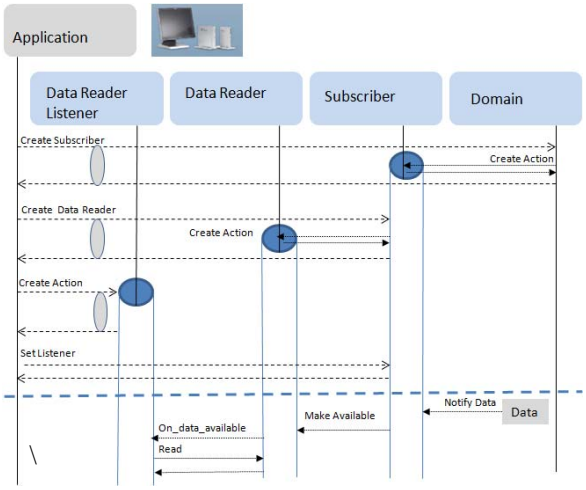


Figure 6. DDS –Subscriber process



ISBN 978-80-261-0386-8, © University of West Bohemia, 2015

For better understanding of DDS usage, a simplified C++ programming code is attached to indicate process of writing and reading data from a sensor with a measured temperature (TempSensor), via publisher (pub) , subscriber (sub), topic (tts) on domain (dp). Example of DDS API with C++ programming:

```
auto dp= DomainParticipant(domainId);
//Publisher C++
auto pub=Publisher(dp);
auto tts=Topic<Temperature_Var1>(dp,"TempSensor");
auto dw=DataWriter<TemperatureVar1>(pub,tts);
dw.write(TemperatureVar1(101,23.5F,0.55F));
//Subscriber C++
auto sub=Subscriber(dp);
auto tts=Topic<Temperature_Var1>(dp,"TempSensor");
auto dr=DataReader<TemperatureVar1>(sub,tts);
auto data=reader.read();
```

IV. CONCLUSION

Having taken into consideration the previous aspects, DDS protocol is proposed as the best candidate to meet high expectations and enable overall opens communication among embedded systems. In contrast to popular protocol MQTT, DDS protocol does not require "central node" approach and has no need for network configuration to enable communication of all devices. In spite of very limited hardware resources and capabilities of embedded systems, this protocol can offer appropriate communication services that meet requirements of the critical applications, such as a defined quality of service with appropriate bandwidth and low response times. DDS provides low-latency, high-throughput data communications, similar to the performance that can be achieved with raw Sockets.

- Content and Temporal Filtering.
- Queries.
- QoS to control.
- Low latency (~50 μsec).
- High throughput (7M mes./sec node-to-node).

A very extensive set of quality-of-service (QoS) parameters controls exactly how information flows from publishers and subscribers through the bus. The middleware matches publishers to subscribers based on their types, topics, and QoS. The result is fast, direct, controlled communications.

ACKNOWLEDGMENT

This work has been supported by the Educational Grant Agency of the Slovak Republic (KEGA) Number: 008ŽU-4/2015: Innovation of HW and SW tools and methods of laboratory education focused on safety aspects of ICT within safety critical applications of processes control.

REFERENCES

[1] R. Drath and A. Horch, "Industrie 4.0: Hit or Hype ?", Industry Forum, IEEE Industrial Electronics Magazine 8(2), pp. 56-58, 2014

[2] J. Mroz, "Shared Data of the Internet of Things and in Idustry 4.0," Detecon Management Report BLUE, Vol. 1 2015, pp. 24-27.In: http://www.detecon.com/ap/ap/files/DMR_blue_IoT_Industry_4_0_Mroz_01_2015_E.pdf

[3] Kagermann at all., "Recommendation of implementing the strategic initiative Industrie 4.0. Final report of the Industrie 4.0 Working Group", 2014

[4] E. A. Lee, "Cyber Physical Systems:Design Challenges. 11 IEEE Symposium on Object Orientated Real-Time Distributed Computing (OSORC)", pp. 363-369, 2008

[5] Mary J. Cronin, "Smart Products, Smarter Services: Strategies for Embedded Control", Cambridge University Press, New York, USA, 2010, ISBN 052114750697805221147507

[6] S. Schneider, "Understanding of Internet Things Protocols", In: http://www.slideshare.net/RealTimeInnovations/

[7] Ling Li, Shancang Li, Shansan Z., „QoS – Aware Scheduling of Services – Orientated of Internet Things", Industrial Informatics, IEEE Transactions on (Volume:10 , Issue: 2), pp. 1497 – 1505, 2015, ISSN 1551-3203

[8] Thiago Teixeira et all, „Service Orientated Middleware for the Internet of Things: A Perpective", Service Wave 2011, LNCS 6994, pp. 220-229, Springer-Verlag Berlin Heidelberg 2011