Communication Middleware Technologies for Industrial Distributed Control Systems: A Literature Review

Ali Balador[1], Niclas Ericsson[1], Zeinab Bakhshi[2]
[1]RISE SICS Västerås, Sweden, {ali.balador, niclas.ericsson}@ri.se
[2]RighTel, Iran, bakhshi.zeynab@gmail.com

***Abstract*—Industry 4.0 is the German vision for the future of manufacturing, where smart factories use information and communication technologies to digitise their processes to achieve improved quality, lower costs, and increased efficiency. It is likely to bring a massive change to the way control systems function today. Future distributed control systems are expected to have an increased connectivity to the Internet, in order to capitalize on new offers and research findings related to digitalization, such as cloud, big data, and machine learning. A key technology in the realization of distributed control systems is middleware, which is usually described as a reusable software layer between operating system and distributed applications. Various middleware technologies have been proposed to facilitate communication in industrial control systems and hide the heterogeneity amongst the subsystems, such as OPC UA, DDS, and RT-CORBA. These technologies can significantly simplify the system design and integration of devices despite their heterogeneity. However, each of these technologies has its own characteristics that may work better for particular applications. Selection of the best middleware for a specific application is a critical issue for system designers. In this paper, we conduct a survey on available standard middleware technologies, including OPC UA, DDS, and RT-CORBA, and show new trends for different industrial domains.**

I. INTRODUCTION

Industrial systems are often safety critical, with end customers expecting a longevity of 10-20 years, along with an availability of up to 99.9999% in harsh environments. In an centralized design, the control algorithm is concentrated in a single entity. A single process controller operates all aspects of your process, but it becomes a single point of failure. In order to improve the reliability of control, process quality and plant efficiency, Distributed Control Systems (DCS) have emerged. Nowadays, industrial automation and control systems include a large number of distributed systems (e.g., controllers, sensors and actuators), which are often connected with each other over an Ethernet-based network. DCS systems include applications in aerospace, defense, industrial automation, automotive industry, and robotics. In these applications, data produced in one system component needs to be shared with other components. Such applications may have stringent deadlines by which the data must be delivered, in order to process it on time to make critical decisions. Increasing demand on productivity, quality, safety, and security in industrial domain leads to new challenges. One of these challenges is that DCS not only require I/O communication for every controller but also need horizontal (with other controllers on the same hierarchical level) and vertical (with other devices on different hierarchical levels) communication [1]. Another challenge is the heterogeneity [2] in various systems and system of systems, that communicates on various homogeneous networks and field-buses (e.g., DeviceNet, ProfiNET, CAN, Profibus). The different systems are normally provided by different vendors. For the systems on a homogeneous network/field-bus level, interoperability is working well. However on the system of systems level, i.e. the heterogeneous network, where all systems converge, the system capabilities, data formats, mapping schemes, and I/O interfaces varies, thereby causing a rather complex heterogeneous system to deal with. In addition, trends like, Internet of things (IoT), Cloud, and 5G are pushing to be adopted and integrated into industrial systems. Bringing new communication technologies, that is likely to cause a need for faster system evolutions. E.g., due to newly detected cybersecurity vulnerabilities, or in order to utilize new functionality provided by cloud suppliers.

To facilitate the industrial control communication in DCS and hide the heterogeneity amongst the subsystems, various middleware technologies have been proposed by international standardization organizations, industrial consortia and research groups over the last couple of decades, such as OPC UA, DDS, and MQTT. These technologies can significantly simplify the system design and integrate control devices despite their heterogeneity.

Despite all advances in the field of middleware technologies, there are still many significant challenges to meet the requirements of DCS systems. These systems are very time sensitive, meaning that transactions and decisions, must be carried out reliably within a predefined time. These systems are usually heterogeneous, complex and they must support various platforms. Some of the most important challenges and aspects for middleware where identified by the authors, together with industrial partners, such as, compatibility, changeability, reuse, timeliness (real-time), quality of service, safety and security, and Internet connectivity. In addition the maturity and activity are important for middleware selection [3].

The rest of the paper is organized as follows: Section II presents related works and scope of paper. Section III shows different middleware technologies. Evaluation and comparison of the selected middleware solutions are then presented in Section IV. Finally, section V concludes this paper.

978-1-5090-6505-9/17/$31.00

TABLE I
SUMMARY OF EXISTING COMMUNICATION MIDDLEWARE

OOM	MOM	SOM
RT-CORBA	DDS	OPC UA
ICE	MQTT	SOME/IP
RMI	AMQP	Thrift
COM/DCOM	CoAP	
	XMPP	
	ZeroMQ	
	Nanomsg	
	YAMI	
	JMS	

II. Related Works and Scope of Paper

In the last decades, many middleware technologies have been introduced for DCS systems and each of them have focused on different application domains. In this section, we identified 16 middleware technologies and classified them into three different groups [4], [5], including object oriented, message oriented, and service oriented middleware technologies, as shown in table I.

In order to focus more on suitable middleware technologies for future DCS systems, many of these selected middleware technologies are removed since they are outdated, though they might still be used in many systems. Moreover, several middleware technologies has been removed for other reasons. For example, Scalable service-Oriented Middleware over IP (SOME/IP) [6] was excluded from further investigations since it does not appear as a middleware and was developed by BMW for in-vehicle communication and was integrated in the AUTOSAR 4.1 specifications. In addition, ZeroMQ [7] and Nanomsg [8] were also removed because they only define messaging formats and are thus not counted as a real middleware. Moreover, those are not actually at the same level as the other solutions, which are reviewed in this paper. Communication middleware solutions like, JMS a pure java based server-to-server protocol, that only support programming languages that use just-in-time compilation or garbage collection, are omitted due to lack of deterministic behavior.

Finally, a group of three promising middleware technologies that also pointed out by many big industries and organizations, including OPC UA, DDS and RT-CORBA has been selected to be evaluated in more details.

III. Communication Middleware Technologies

In this section, we present the core architecture of each of the three selected middleware technologies and their major characteristics, including real-time capability, security, and application domains.

A. OPC UA

The OPC foundation is an independent committee that specifies and develops the OPC Unified Architecture (OPC UA) standard [9], [10]. The OPC Classic is the original version that was developed by OPC foundation in 1996. OPC UA aims to expand OPC's interoperability to the device and enterprise levels. OPC UA is a client/server protocol. It defines communications from the application to the transport layer, making it very interoperable between vendors [11]. OPC UA provides a set of standardized services with which clients can interact with servers. Servers provides access to data and functions that are structured in object-oriented information models.

1) OPC UA architecture: OPC UA is much more than a protocol. It builds on different layers shown in Figure 1. Two main components of the architecture are transport and data model. The OPC UA data model defines the rules and how to expose an information model. It includes the entry points into the address space, base types used to build a type hierarchy, and some enhancing concepts, such as state machines used in different information models. The transport layer is the other main component, which defines protocols that serializes/de-serialized the data and sends over the network. Two transport protocols that are currently defined for OPC UA are TCP and SOAP/HTTP [12]. Both of these protocols use standard TCP as their underlaying technology and it makes OPC UA less promising for hard time sensitive systems.

OPC UA basic services (base services) layer defines abstract methods descriptions that are protocol independent, and form the basis for the entire OPC UA functionality [9]. This layer provides an interface between servers and clients using transport mechanisms to exchange information. OPC UA supports all successful Classic OPC features by information models defined for the domain of process information on top of the base services layer. Different organizations can define their own specific information models on top of basic information models, which are defined by OPC UA standard to describe, configure, and monitor devices and PLCopen, a standard for PLC programming languages. Moreover, vendor specific information models can be defined using directly the UA base, the OPC models, or other OPC-UA based information models.

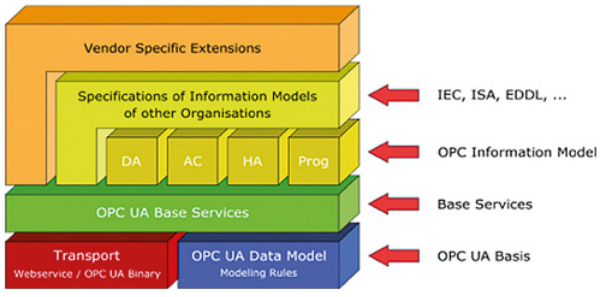

Fig. 1. OPC UA Architecture

2) Real-time capability: There was no real-time demand on OPC in the past because everything was user-driven. For example, everything was asynchronous for traditional Supervisory Control and Data Acquisition (SCADA) systems, thereby there was no need for real-time communication. As the OPC community intends to support real-time applications, real-time behavior on the software and on the network becomes a requirement. Now, OPC community is working on

the UDP based OPC UA protocol and the publish/subscribe communication model, which together with the Time Sensitive Networks (TSNs) extension to Ethernet will enable even hard real-time capabilities to OPC UA applications. During the OPC day Europe 2016 [13], OPC UA Foundation announced that the upcoming publish/subscribe communication model that is being prepared for OPC UA specification 1.04 will be released during the first half of 2017.

3) Security: OPC UA security has a multilayer structure like OPC UA architecture [14]. Different security specifications are modeled in different layers of OPC UA Structure: (a) Authentication: in application layer of OPC UA architecture, clients, user or operator can be authenticated via a password with the specific username or a certificate or combination of both. X.509 certificates are used for authentications, (b) Authorization: data access rules with defined rights in the policies, are regulated in the application layer for each node. (c) Auditing, in application layer auditing the logs and events are defined. The server can log, each operation and the value changes by users in every time frames, (d) Integrity and Confidentiality are ensured in the secure channel of the communication layer. Application authentication and authorization are also defined in communication layer. OPC UA also supports symmetric and asymmetric encryption key management systems to ensure confidentiality and data integrity using the encryption feature in transport layer. IPSec or Transport Layer Security (TLS) protocols are used to create the secure communication channel and encrypted messages between nodes in transport layer.

B. DDS

DDS [15] is a middleware whose concept was standardized and is currently managed by the Object Management Group (OMG). The first version of DDS was released by OMG in 2004. DDS simplifies software systems by streamlining the way that different applications receive and process data. It causes reductions in cost and risks through development, integration, deployment and the lifetime maintenance of distributed software systems. DDS is in charge of transferring information. The system includes different publishers (data writers) and subscribers (data readers). DDS makes data exchanges between different devices that can include different publishers and subscribers at the same time. Figure 2 generally shows how DDS connects different publishers and subscribers.

1) DDS architecture: The OMG DDS standard defines a programming model including a wire protocol and a set of standard APIs. Figure 3 shows the structure of OMG DDS standard. The DDS Interoperability Wire Protocol Specification (DDSI) defines the interoperability protocol to ensures that different vendors' implementations of DDS can interoperate. The DDS API provides the standard interface between DDS and applications, and ensures the source code portability between different vendor implementations. Currently OMG provides the standard DDS API interface in C, C++ and Java languages. Lower level Data-Centric Publish-Subscribe (DCPS) is intended for efficient delivery of information. All the activities for DDS communication, for example, defining

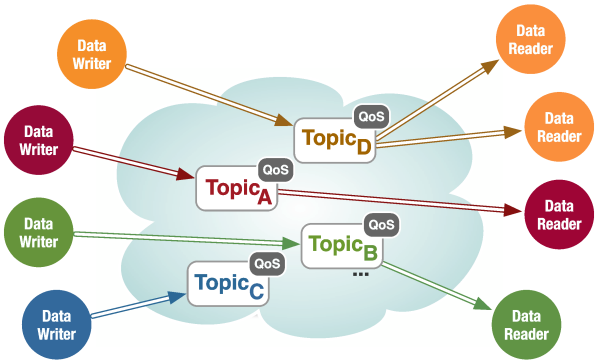

Fig. 2. DDS Concept

topics, creating publisher/subscriber entities, writing/reading data are defined in this layer. Optional higher-level Data-Local Reconstruction Layer (DLRL) provides more direct access to the exchanged data and simpler integration with the local language constructs. The light-weighted UDP/IP is used as a transport for DDS as it has the following characteristics: universal availability, connectionless, predictable behavior, scalability and multicast support. Although DDS can also be implemented using other default transport protocols, such as TCP/IP, RTPS/UDP, IP multicast, it is supposed to work best over UDP/IP [16].

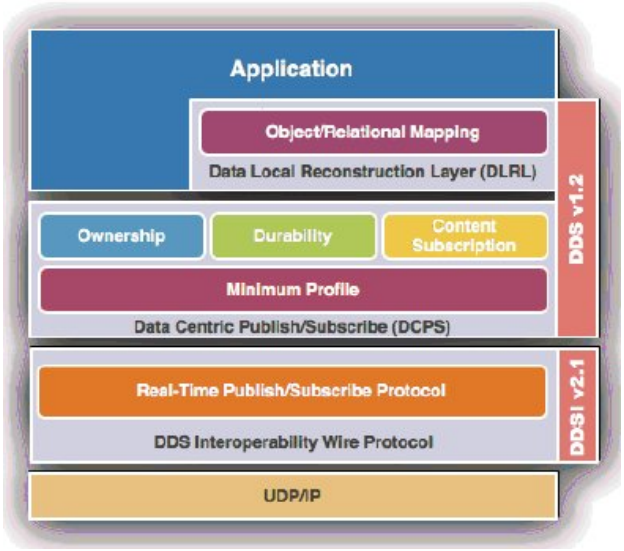

Fig. 3. Architecture of OMG DDS standard

2) Real-time capability: DDS is a publish/subscribe protocol, which focuses on communication at the edge of the network. DDS nodes can communicate directly in peer-to-peer fashion using UDP multicast. This removes the need for centralized network management. Thereby, DDS is able to provide faster and deterministic data distribution and is stated to be a good solution for reliable and extreme real-time data delivery at the edge [17]. DDS provides very low overhead, efficient processing and low latencies.

3) Quality of Service: QoS is a set of parameters to evaluate a service offered. From processing perspective, QoS represents a set of both quantitative and qualitative characteristics that is needed to achieve the required functionality. From communication perspective, QoS is defined as a set of requirements that a network must provide for the transport of traffic. Among all QoS parameters, the performance parameters (throughput, bandwidth or resource management) are the most important parameters for all distributed systems. Priority, durability, synchronization and deadline are QoS parameters that can guarantee a minimum QoS for DCS systems [18].

One of the most important characteristics of DDS is that it provides 22 different QoS policies, covering almost all aspect of communications except connection management parameters, such as connection delay, mode, status and reconnection. DDS is able to support most of the recommended parameters [18]. These QoS policies can be seen as a function that allows users to specify and control the behavior of the communication. The QoS policies can be configured to all DDS entities like publisher, subscriber, data writer, data reader, topic and so on. The DDS communication can only be established when the QoS configurations are compatible between the publisher and subscriber. In fact, the QoS policy follows the subscriber-requested, publisher-offered pattern. These QoS parameters show concerns to different aspects of DDS, including data delivery, data availability, data timeliness, configuration and resources.

4) Security: DDS provides five plug-ins to ensure Security, authentication, access control, cryptographic, data tagging, and logging plug-ins. Authentication plug-in verifies the identity of the application and/or user that call operations on DDS. X509 certificate, PKI, shared CA, RSA authentication and secret handshake establishment using Diffie-Hellman algorithm are used in this plug-in. Access control plug-in pushes and enforces a set of policies that defines what succeeded authenticated users can perform in DDS environment, like which domain a user can join or which topic can be published by a specific user. The cryptographic plug-in implements all related cryptographic features such as, encryption, decryption, digital signatures, hashing, generating and deploying keys and certificates. Data tagging plug-in provides the ability to add security label or tag to data, so it can be used to make different data classifications and data information about data reliability. Logging plug-in supports the capability of logging all the security events and necessary operation, including errors and security attacks.

C. Real-Time CORBA

The first version of CORBA was released in October 1991 by OMG. A group was formed within the OMG in 1996 to support real-time applications in the CORBA standard. Real-time CORBA provides specifications to support the QoS requirements of embedded and distributed real-time systems . The RT-CORBA specifications that extends the existing CORBA standard, providing features that allow applications to allocate, schedule, configure and control CPU, communications and memory resources. RT-CORBA adds QoS control to regular CORBA to improve application predictability, which is required for real time applications, e.g., bounding priority inversions and managing resources end-to-end.

1) RT-CORBA Architecture: In this middleware, Object Request Broker (ORB) is the central component that is responsible for making communications between clients and servers transparent and for allowing interoperability between applications in hetero- and homogeneous networks and environments. An application sends requests to an ORB, which directs the request to an appropriate object that provides the desired service. ORB acts as an intermediary, which allows the object request or to access multiple remote or local objects.

Over the last decade, RT-CORBA has been gradually deployed and used in many domains, such as aerospace, telecommunications, medical systems and many more, that have high level of QoS requirements. The acceptance of RT-CORBA has mainly two factors: maturation of patterns and frameworks, and standards. Several design patterns, frameworks, and specifications has been introduced considering high-performance and real-time systems.

The RT-CORBA specification defines capabilities that must be managed by ORB endsystems to provide end-to-end predictable communications between CORBA clients and servers. Figure 4 shows the ORB endsystem features for RT-CORBA. These capabilities include communication infrastructure resource management, OS scheduling mechanisms, real-time ORB endsystem, and real-time services and applications. To manage these capabilities, RT-CORBA defines standard interfaces and QoS policies that can be used by applications to (1) manage processor resources via thread pools, priority mechanisms, intra-process mutexes, and a global scheduling services, (2) memory resources via buffering requests in queues and bounding the size of thread pool, and (3) communication resources via protocol properties.

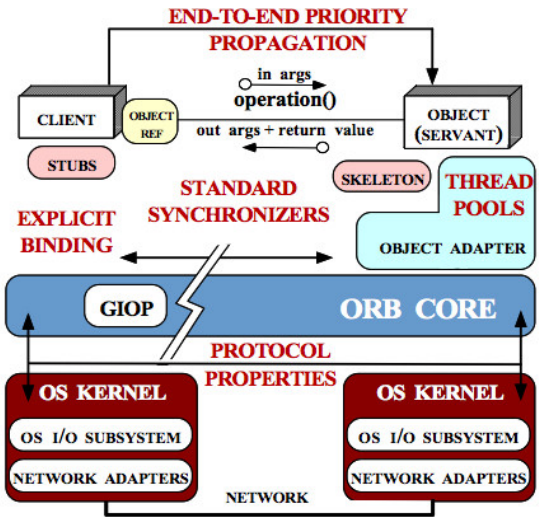

Fig. 4. ORB Endsystem Features of RT-CORBA

2) Quality of Service: RT-CORBA defines 13 different QoS policies that includes many QoS areas, such as connection management, message flow, and time management. However, CORBA is focused on the message flow management parameters including delivery order, max hops, priority, routing, and synchronization.

3) Security: Different security protocols are included in the RT-CORBA specifications, such as TLS, its predecessor, Secure Sockets Layer (SSL), Security Attribute Service (SAS) and Internet Inter-ORB Protocol over SSL (SSLIOP) which uses the SSL standard for authentication and encryption over CORBA's Internet Interoperability Protocol. SAS protocol [19] provides message security and protection as well as client to server authentication. The SAS protocol is divided into two layers: (1) the authentication layer, (2) the attribute layer, SAS provides Inter-operable Authentication, Delegation and Privileges. The SSLIOP provides certificates and encryption keys. Certificates and security tags are written in CORBA Interface Definition Language (IDL). The encryption also supports wide range of algorithms, such as RC4, DES, 3DES, and IDEA.

IV. Evaluation and Comparison

Table II summarizes the results of analysis of different characteristics of the middleware technologies that has been presented in previous sections.

OPC UA and DDS have many similarities, but they actually target different applications that make them neither competitive nor orthogonal, but complementary. OPC UA provides client-server interaction between components such as devices or applications. DDS is a data-centric bus for integration and peer-to-peer data distribution. Many applications would need both of these characteristics, or they might propose two different solutions to the same problem. Currently, OPC UA is widely deployed in automation for manufacturing, process control, and power. DDS applications are more focused on medical, transportation, power, and defense domains.

In contrast to OPC UA, which exposes the network, DDS hides the network, making device types and topology transparent. DDS delivers the data to the receiver without the receiver needs to deal with how the data got there. There is a huge amount of data collected from sensors and other inputs that need to be managed. OPC UA turns lower-level data into information, that generally means raising the level of abstraction and taking us away from the low level data points. But, DDS provides access to low-level data points through filtering. Presumably aggregation and abstraction are possible too, but they simply become part of the data model, in the same manner that database tables can handle aggregation and calculations. OPC UA has two main characteristics: (1) it was designed for non time-critical applications and (2) it shows a relatively static configuration with data flowing up and out for use by application programmers. On the other hand, DDS was designed for applications that require short-loop, real-time, reliable, and distributed exchange of data. DDS can be used for setups having a complex data model and a dynamic network that supplies and consumes the data. Therefore, DDS provides more dynamic configuration compared to OPC UA.

TABLE II
Evaluation and comparison of studied middleware technologies

	OPC UA	DDS	RT-CORBA
Sponser	OPC (2009)	OMG (2004)	OMG (1996)
Real-time Mode	Yes	Yes (extreme RT)	Yes (hard RT)
QoS	No	22 Levels	Limited
Security	Integrated encryption, integrity verification and authentication services. Custom access control can be implemented.	Five Security Plug-in Interfaces (SPIs) for authentication, access control, cryptographic, logging, and data tagging service plug-ins.	TLS/SSL and SSLIOP for encryption and integrity verification. SAS protocol for authentication. Custom access control.
Architecture	Request/Reply Publish/Subscribe	Publish/Subscribe	Request/Reply
Transport	TCP, optionally; SOAP/HTTP or UA-specific protocol on top	UDP, TCP	TCP
Portability and platform independence	Yes	Yes	Yes
Main Features	Full support for all gateway and control network services. Interoperability with existing applications and availability of APIs for different programming languages.	Efficient distribution of data and rich set of QoS policies.	Ability to control and management of resources.
Disadvantages	Designed for specific and non-real time applications. Weak support of systems with dynamic configurations. Weak support of QoS.	Difficulty in guarantee of data delivery and ordering events.	Weak support of security, scalability, and fault tolerance. Not suitable for highly dynamic environments.

Although DDS is a good candidate for use in industrial automation domain, specially since it supports real-time data production and consumption, but OPC UA dominates in this area. The reason is that OPC UA is not being used in the control feedback loops and instead it is used for sharing non time critical information. PLCs have handled that role for a long period of time before the IoT. New trends in Industry 4.0 and the IoT show many companies recently

selected DDS for real-time applications, especially those that start a manufacturing floor completely from scratch. In total, both OPC UA and DDS are important to the future of the industrial IoT (IIoT). They are currently working together to be integrated. Two methods have already been proposed: (1) a "OPC UA/DDS gateway" specifications that will permit independent implementations to work together more smoothly and (2) a "OPC UA DDS profile" will provide integration of different use cases [20].

RT-CORBA is another middleware developed by the OMG to support real-time applications. RT-CORBA and DDS are complementary middleware standards. Similar to OPC UA, CORBA is based on the client-server architecture and is best suited to applications in which, one software component (the server) is supplying a service to one or more other components (clients), such as remote command processing, file transfer and synchronous transactions. DDS is based on the publish-subscribe architecture and is best suited to applications in which one or more data sources (publishers) need to communicate information to one or more data users (subscribers). Because publishers and subscribers require no knowledge of each other, DDS provides a powerful integration framework for large or dynamic distributed systems. With DDS, new components can be added (as publishers and/or subscribers) without any changes to the rest of the system. RTI offers an integrated DDS and CORBA solution for many applications that support both CORBA and DDS middleware standards.

V. CONCLUSIONS

In this survey paper, we have analyzed three key middleware technologies for industrial control systems, including OPC UA, DDS, and RT-CORBA. Several parameters, such as response time, QoS, and security were studied and summarized. This survey shows that both OPC UA and DDS are critical standards for ensuring interoperability between a broad set of manufacturing processes and equipment. Therefore, both of them are important for Industry 4.0 that also mentioned by Industrial Internet Consortium (IIC). Microsoft works with the OPC Foundation to expand support of the OPC UA in their products. They are announcing a new open-source reference stack that supports OPC UA. Moreover, Azure IoT uses upcoming Publisher/Subscriber OPC specification extension and contains a sample application called OPC Publisher, therefore, existing OPC UA servers can be connected directly to Azure IoT and send OPC UA telemetry data for analysis and machine learning in the cloud. On the other hand, DDS specification is able to handle many of the problems and requirements of distributed control systems, thereby having a publish/subscribe communication style. Efficient distribution of data with minimal overhead and the ability to control QoS are the most important benefits of this middleware. In addition, a rich set of QoS policies provided by this middleware solves most of the existing problems in distributed control systems. However, current trends shows that none of these middleware technologies can support all requirements and QoS attributes of distributed control systems, and each one only focuses on some QoS parameters and requirements. Recently, both OPC foundation and OMG are working on methods to integrate OPC UA and DDS to combine the benefits of both technologies.

REFERENCES

[1] Calvo, F. Pérez, I. Etxeberria and G. Morán, "Control Communications with DDS using IEC61499 Service Interface Function Blocks", IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-4.

[2] B. Almadani, S. Khan, M. N. Bajwa, T. R. Sheltami, and E. Shakshuki, "AVL and monitoring for massive traffic control system over DDS," Mobile Information Systems, vol. 2015, Article ID 187548, 9 pages, 2015.

[3] N. Ericsson, T. Lennvall, J. Åkerberg and M. Björkman, "Challenges from research to deployment of industrial distributed control systems," 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), Poitiers, 2016, pp. 68-73.

[4] L. Qilin and Z. Mintian, "The State of the Art in Middleware," 2010 International Forum on Information Technology and Applications, Kunming, 2010, pp. 83-85.

[5] Wolfgang Emmerich. 2000. Software engineering and middleware: a roadmap. In Proceedings of the Conference on The Future of Software Engineering (ICSE '00). ACM, New York, NY, USA, 117-129.

[6] "SOME/IP". Available: http://some-ip.com.

[7] "iMatix ZeroMQ". Available: http://www.zeromq.org.

[8] "Nanomsg". Available: http://nanomsg.org.

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

[10] M. Stopper and B. Katalinic, "Service-oriented Architecture Design Aspects of OPC UA for Industrial Applications", IMECS 2009, mARCH 18-20, 2009, Hong Kong.

[11] "OPC Unified Architecture, Interoperability for Industrie 4.0 and the Internet of Things, OPC UA Foundation". Available: https://opcfoundation.org/wp-content/uploads/2016/05/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN-v5.pdf.

[12] "Why OPC UA matters". Available: http://www.ni.com/white-paper/13843/en.

[13] "OPC Day Europe 2016". Available: https://www.prosysopc.com/blog/opc-day-europe- 2016/

[14] P. Jafary, S. Repo, M. Salmenpera and H. Koivisto, "OPC UA security for protecting substation and control center data communication in the distribution domain of the smart grid," 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, 2015, pp. 645-651.

[15] Data Distribution Service (v1.4), Object Management Group, 2015. Available: http://www.omg.org/spec/DDS/1.4/.

[16] Object Management Group (OMG), The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification, OMG, 2014.

[17] Oh, Sangyoon, Jai-Hoon Kim, and Geoffrey Fox. "Real-time performance analysis for publish/subscribe systems." Future Generation Computer Systems 26.3 (2010): 318-323.

[18] J.-L. Poza-Luján, J.-L. Posadas-Yagüe, J.-E. Simó-Ten, "A Survey on Quality of Service Support on Middleware-Based Distributed Messaging Systems Used in Multi Agent Systems", Proc. International Symposium on Distributed Computing and Artificial Intelligence ser. Advances in Intelligent and Soft Computing, pp. 77-84, 2011.

[19] "3GPP TS 32.373, "3rd Generation Partnership Project - Technical Specification Group Services and System Aspects - Telecommunication management;Security services for Integration Reference Point (IRP) - Common Object Request Broker Architecture (CORBA)solution(Release 8),"", 2008.

[20] "The Inside Story: How OPC UA and DDS Can Work Together in Industrial Systems". Available: http://www.slideshare.net/RealTimeInnovations/the-inside-story-how-opc-ua-and-dds-canwork-together-in-industrial-systems.