On the integration of OPC UA over Wired – Wireless Time Sensitive Networking

Óscar Seijo
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
Arrasate, Spain
oseijo@ikerlan.es

Raúl Torrego
Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA)
Arrasate, Spain
rtorrego@Ikerlan.es

Iñaki Val
Max Linear Incorporated
Carlsbad, United States
ival@maxlinear.es

***Abstract*—This paper deals with the integration of OPC UA over Wireless / Wired (Hybrid) Networks with Time Sensitive Networking (TSN) capabilities. The paper overviews the current state of middleware protocols typically built over Ethernet, namely OPC UA and DDS. Focusing on OPC UA, the paper proposes a HW/SW device architecture, based on the SHARP platform to enable the integration of OPC UA and Hybrid TSN. The integration includes both the common OPC UA Client/Server mechanism and the recently standardized OPC UA PubSub. The platform is used to build a HW testbed to assess the performance of OPC UA over Hybrid TSN. The results demonstrate that the network can provide seamless application interoperability and that can satisfy the traffic needs in a typical industrial setup. However, the paper also highlights that there is still significant research to be done to achieve deep integration between OPC UA and Hybrid TSN.**

Keywords—OPC UA, DDS, middleware protocols, Wireless TSN, TSN, 802.11, time synchronization, industrial communications.

I. Introduction

Factories are going through a fast revolution caused by the Industrial Internet of Things (IIoT) paradigm, becoming increasingly digitized and interconnected [1]. In this context, it is essential to develop reliable and efficient communication solutions that meet the specific requirements of the enormous amount of data sets that are generated by the applications built on top of the networks [2]. Ethernet-based fieldbuses are currently used in these demanding applications, and Time-Sensitive Networking (TSN) is gaining traction because it supports in the same network very diverse applications with different Quality of Service (QoS) needs. Among other mechanisms, TSN defines mechanisms for clock synchronization, support of mixed-criticality traffic (i.e., real-time and best-effort traffic), and ultra-reliability by the use of redundant transmission paths.

Now that the TSN technology is being adopted by the industry, research efforts have moved to bring TSN features to wireless networks (Wireless TSN) due to the benefits that wireless technologies offer such as mobility, cost-effectiveness, and easier configuration. 5G and 802.11 are currently being explored as two potential candidates for enabling Wireless TSN [3] and there have been significant efforts into bringing their lower layers closer and closer to the performance expected by a Wireless TSN system. Even so, there are still significant research and standardization work to be done in this area to consider that 5G or 802.11 are Wireless TSN solutions [3].

Apart from the Wireless TSN development itself, two additional challenges must be addressed for the successful deployment of Wireless TSN solutions. The first challenge relates to the integration of Wireless TSN and Wired TSN (i.e., Hybrid TSN) (see Fig. 1), whereas the second challenge relates to the integration of TSN (both wired and wireless) with middleware protocols.

Regarding the integration of Wireless and Wired TSN, such integration enables a unified communications platform that allows the data to be transferred quickly and reliably across distributed devices with different needs, capacities, and communication constraints [3]. Therefore, it enables the devices connected to the networks to exchange the right data at the right time, no matter what type of network the data is being sent through. This challenge has been thoroughly explored during the last few years achieving significant progress for different wireless technologies like 5G [4], 802.11 [5] [6], custom versions of 802.11 (w-SHARP) [7], or proprietary radio solutions [8].

Regarding the integration of TSN with middleware protocols, such integration is also of relevance since it enables seamless data exchange between applications among distributed devices across the network. The middleware abstracts the application from the specific capabilities of the network interfaces and enables the communication of data between different systems, devices, and applications across diverse communication networks. Therefore, it is almost required if one wants to build a portable application that can run on devices with diverse networking and processing capabilities. In addition to the abstraction, middleware protocols typically offer QoS mapping of the traffic generated

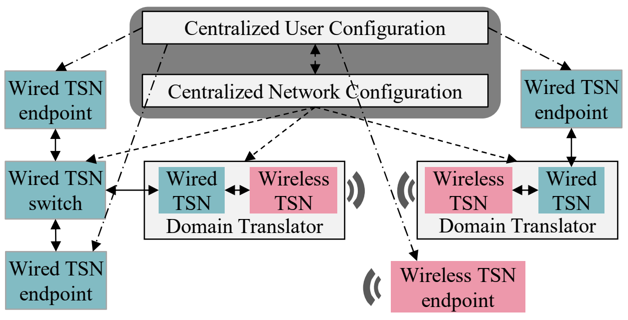

Fig. 1. Hybrid TSN Network Architecture.

by the applications. The mapping can be used to feed the TSN Centralized configuration entities (CNC/CUC) and to create TSN automatic configurations. Also, any time that a new OPC UA device is integrated into the network the device can be automatically configured based on its application needs.

The challenge of the integration of Wired TSN and middleware protocols such as OPC Unified Architecture (OPC UA) or Distribution Data Service (DDS) has been recently explored by some authors. For instance, [9], [10], [11], or [12] analyze the integration of Wired TSN and OPC UA and include assessments of the timing performance. However, these works are limited to Wired TSN networks. Some research has also been conducted for Wireless TSN using DDS [13]. However, the commercial-grade devices used in this research were not specifically designed for Wireless TSN communications, and this fact clearly limits the achievable performance. In addition to that, they analyze DDS over Wireless TSN, but other middleware protocols, such as OPC UA remain unexplored. Therefore, there is a clear gap in the state-of-the-art regarding the use of middleware protocols in Wireless TSN, and, derived from that, over Hybrid TSN networks.

In this paper, we provide first an overview of DDS and OPC UA, two of the most widely used middleware protocols for industrial applications. Then, we examine the challenges associated with the integration of these protocols and Hybrid TSN networks. Focusing on OPC UA, we discuss a possible Hardware/Software (HW/SW) architecture and components involved in the integration of OPC UA and a Hybrid TSN solution (SHARP). The discussion is completed with an experimental setup that is used to evaluate the integration of the technologies and derive some performance figures.

The rest of the paper is organized as follows. First, Section II overviews some of the most widely used middleware protocols in the industry for Ethernet-based networks: OPC UA and DDS. Then, Section III proposes a HW/SW architecture that enabled the integration of OPC UA. Section IV describes the experimental setup and its evaluation. Finally, Section V concludes the paper.

II. Middleware protocols for Industrial Networking

Amongst other protocols, two of the primary middleware protocols for industrial networking are DDS [14] and OPC UA [15]. Both protocols provide effective methods for communication between devices and systems in industrial environments. However, there are some key differences between the two protocols that should be considered to select one of them for a particular application.

On the one hand, DDS uses a publish/subscribe (PubSub) mechanism to provide lightweight, reliable, real-time communication of data across a distributed system [16]. In PubSub, publishers send broadcast/multicast messages to the network with application data. Subscribers can then receive messages from any publishers just by asking them to be on the "publisher" list. Any device in a network can be either a publisher or subscriber depending on the data they store and what they need to communicate. The PubSub model offers a decoupled communication approach, making it easier to scale and maintain applications. It also allows for sessionless, asynchronous communication, meaning publishers do not need to wait for a response from their subscribers before sending additional messages. Therefore, DDS is mainly oriented to lightweight traffic in large-scale networks with real-time demands.

On the other hand, Open Platform Communications United Architecture (OPC UA), released for the first time in 2008, is an independent-provider middleware to enable machine-to-machine communications in industrial setups [15]. OPC UA is designed to be secure, reliable, and with the capability to support both real-time and non-real-time communications. It is widely used in industrial automation systems, where it is known for its seamless interoperability. The most common mechanism of OPC UA is based on Client/Server and built over TCP/IP [17]. It means that OPC UA data access is mostly based on *Clients* polling *Servers* for gathering data. Whereas this data access approach is well suited for lightweight data consumption and interoperability, it generates more traffic than a PuSub mechanism and therefore it is less favorable for applications with real-time requirements and large-scale environments. Furthermore, a latency-optimized integration of TCP/IP on top of TSN (whether wired or wireless) is quite complex due to the nature of each protocol: whereas TCP/IP is a session-based and event-based protocol, TSN is fundamentally a time-triggered protocol [18].

Considering the OPC UA Client/Server approach limitation, the OPC foundation recently upgraded the OPC UA specification to include a PubSub mechanism in a similar fashion as DDS [19]. In comparison to traditional OPC UA, PubSub is designed to enable real-time, high-speed communication between multiple nodes. It supports data streaming (multicast and broadcast) and can be used to meet the strict demands of applications deployed over large-scale systems. Yet still, the PubSub mechanism is designed following the OPC UA principles, and to be extensible and platform independent. Finally, OPC UA PubSub is built directly on top of layer 2 (e.g., TSN) or, optionally, using UDP/IP; in contrast with Client/Server, which is built on top of TCP/IP. Therefore, it features a far simpler, latency-optimized integration with TSN than Client/Server [20]. In a nutshell, PubSub is intended to be used for time-triggered traffic and so the TSN network and OPC UA application can be configured to minimize the end-to-end latency.

OPC UA Client/Server and PubSub can be simultaneously used by devices connected to the same OPC UA network, as shown in the OPC UA / TSN stack (see Fig. 2). The unification of the mechanisms allows the OPC UA application designers to allocate different data streams to each mechanism according to the application needs. For instance, latency-demanding, high refresh rate variables, can be allocated to the PubSub mechanism. Other variables, used for monitoring and configuration purposes, can be allocated to the Client/Server mechanism and consumed based on the events that are going on in the industrial system. Since TSN

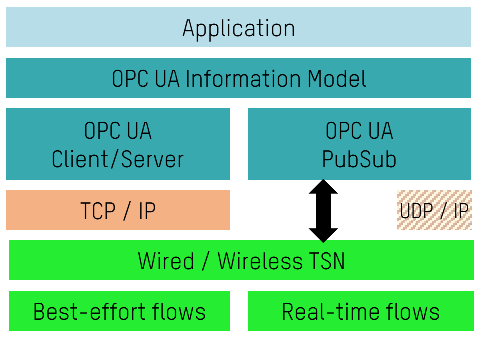

Fig. 2. OPC UA stack and its integration with Wired / Wireless TSN.

supports mixed-criticality traffic, the PubSub traffic can be naturally allocated into latency-guaranteed, real-time TSN flows (i.e., directly on top of layer 2), whereas the Client/Server traffic can be allocated into Best-Effort TSN flows through TCP/IP (Fig. 2).

III. Proposed HW/SW Hybrid TSN Architecture

This section presents the device architecture that we have designed to develop the network proof of concept. The architecture is based on the SHARP architecture, which has been comprehensively described in other works [7] [21].

The Hybrid TSN devices that we have considered for the network can be classified into three categories: domain translators, Wired TSN endpoints, and Wireless TSN endpoints. The HW/SW internal architecture of each device is depicted in Fig. 3 and Fig. 4 (a) and (b). The domain translator is equivalent to a Wi-Fi access point and provides the integration of Wired and Wireless TSN. On the other side, the Wireless TSN and Wired TSN endpoints are intended to generate or consume OPC UA data.

The HW layer of the devices integrates the communication interface, regardless they use Wired or Wireless TSN. The domain translator includes both Wired and Wireless TSN interfaces and HW entities to perform the time synchronization and traffic translation across the domains. On the other hand, the SW provides a complete Internet stack built on top of the TSN interfaces. The internet stack is completed with a standard PTP (802.1AS) stack, an OPC UA stack, and the application built on top of the OPC UA stack.

The class of traffic generated by the OPC UA stack can be registered at the TSN layers to establish its priority. In our case, the messages that are identified as PubSub messages are considered time-critical and are allocated to the real-time TSN flows, minimizing the delivery time. On the other hand, the Client/Server messages, which are based on TCP/IP, are sent through the best-effort flows. Therefore, we can provide compatibility with the whole OPC UA stack and classify/prioritize the traffic based on the application needs.

IV. HW Testbed and Results

In this section, we first present the HW testbed that we have developed to evaluate the integration and the experiments that we carried out over the platform, and then we present the results derived from the experiments.

A. HW Testbed

We implemented the network devices using System on Chip (SoC)-based platforms for SW-Defined Radio (SDR) purposes. These platforms are ideal for this research since they allow maximum control of the stack, from the physical layer up to the application layer. In particular, the devices were built using de ADI RF SOM platform. The core of the platform is a Zynq 7000 System-on-Chip (SoC), which comprises a microcontroller with two ARM processors and a Field Programmable Gate Array (FPGA). The platform includes an AD9361 radio chip used by the Wireless TSN modem, and an Opsero Ethernet card to connect the Wired TSN switch. The Wired TSN switch is a Multiport TSN switch developed by SoC-e for FPGAs [22] and the Wireless TSN modem is an implementation of the w-SHARP protocol [23]. The SW uses a Linux-based OS and implements the standard Linux network stack, a proprietary TSN configuration interface, a PTP stack (ptp4l [24]), an OPC UA stack (open62541 [25]), and application demos built on top of the OPC UA stack.

The network comprises a total of five devices: a Wired TSN endpoint, a Wireless TSN endpoint, a domain translator, one PC, and one 802.1AS master clock, as shown in Fig. 5.

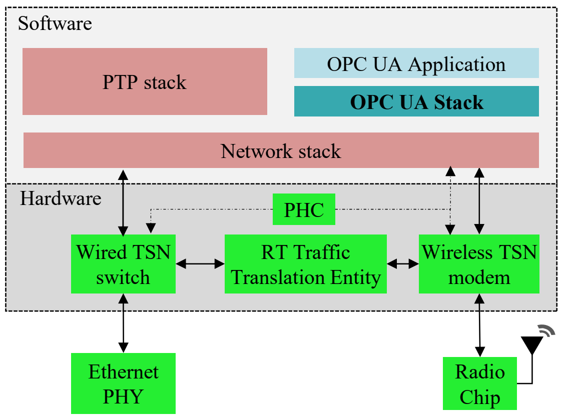

Fig. 3. Domain Translator of the Hybrid TSN architecture with OPC UA support.

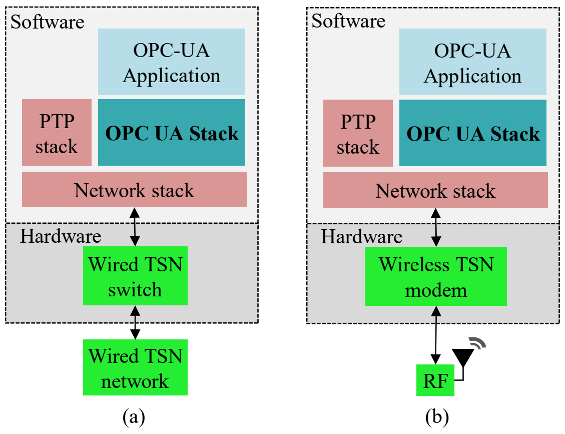

Fig. 4. (a) Wired TSN endpoint with OPC UA support, (b) Wireless TSN endpoint with OPC UA support.

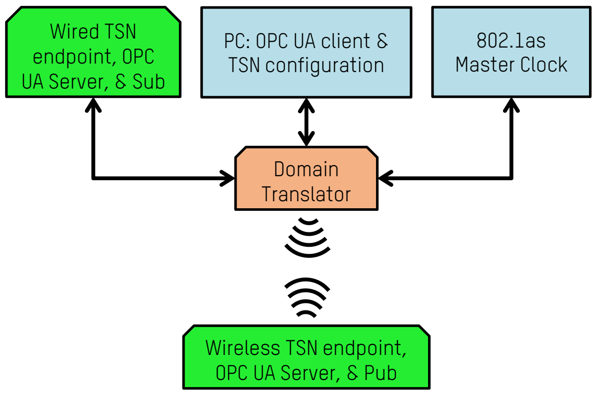
Wired TSN endpoint, OPC UA Server, & Sub
PC: OPC UA client & TSN configuration
802.1as Master Clock
Domain Translator
Wireless TSN endpoint, OPC UA Server, & Pub

Fig. 5. OPC UA Hybrid TSN network architecture.

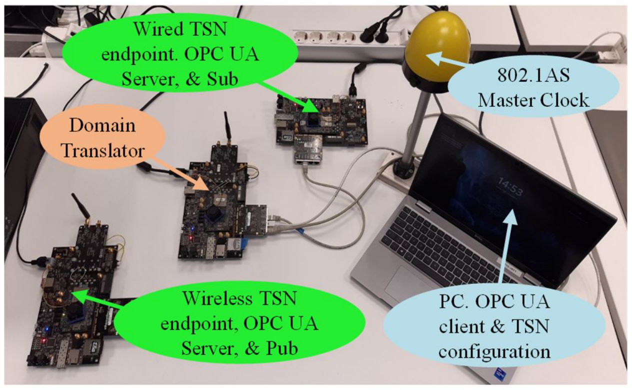
Wired TSN endpoint. OPC UA Server, & Sub
802.1AS Master Clock
Domain Translator
Wireless TSN endpoint, OPC UA Server, & Pub
PC. OPC UA client & TSN configuration

Fig. 6. Experimental testbed to evaluate the integration of OPC UA and Hybrid TSN.

Alternatively, a picture of the HW testbed can be found in Fig. 6. The functionality of each device is described as follows.

The PC is used as the network configuration and monitoring entity. It includes OPC UA and TSN clients that are respectively used to access the OPC UA and TSN configuration of the network devices. The master clock provides the reference time of the network, which is used by the devices to synchronize their network interfaces and by the OPC UA application to synchronously generate data. The domain translator provides the integration of Wired and Wireless TSN, namely: the translation of the TSN traffic flows and the time synchronization propagation from the Wired to the Wireless TSN. The Wireless TSN endpoint includes an OPC UA Server and a publisher. The Server is used for monitoring and configuration purposes. The publisher is designed to be synchronized to the network global time. Finally, the Wired TSN device includes an OPC UA server and a subscriber. The Server is also used for monitoring and configuration purposes, whereas the subscriber is prepared to receive the data from the publisher located at the Wireless TSN device.

The OPC UA servers located at the TSN devices include five parameters: an enable to either transmit/receive the application data, the transmission/reception period, the transmission/reception offset within the period, a read-only variable that measures the number of frames transmitted/received, and a variable to measure the number of lost frames (receiver only).

B. System evaluation

We have configured the network using the OPC UA and TSN clients embedded in the PC as follows.

The OPC UA publisher, located at the Wireless TSN device, is configured with a cycle of 2 ms (500 Hz) and with an offset of 1.8 ms (0.5 ms before the actual Wireless TSN transmission takes place). The data generated by the publisher comprises an increasing counter and a dummy signal (two 32-bit integers), which are used for network validation purposes. We planned to use a commercial device as the publisher but, to the best of our knowledge, there is not any available device with TSN and OPC UA PubSub support up to this date.

The data generated by the publisher is read and sent through the Wireless TSN network at every publisher Cycle. The subscriber is configured to consume the data by the publisher also with a cycle of 2 ms but with an offset of 1 ms. That is, the end-to-end delay between publisher and subscriber is configured to a maximum of 1.2 ms, which, according to our empirical measurements, is enough time to accommodate the SW stack plus network delay.

We configured the OPC UA to periodically poll every 100 ms the variables that hold the number of transmitted frames and the number of lost frames (at the Wireless and Wired TSN endpoints). However, the OPC UA client is not able to provide communication metrics. Therefore, we emulated the traffic generated by the OPC UA with the ping utility, where we generated packets with a periodicity of 100 ms and a size of 150 Bytes.

The Wired/Wireless TSN network flows are accordingly configured to accommodate the OPC UA traffic with the appropriate latency. That is, we have configured a TSN flow with the origin and the destination the OPC UA publisher and subscriber respectively.

The TSN flow is configured at the Wireless TSN, domain translator, and Wired TSN device with a cycle of 2 ms and the TSN window starts with an offset of 0.3 ms.

The Wireless TSN network has been configured to operate at the 2.4 GHz band and with a bandwidth of 2.4 GHz. The experiments were done under conditions without significant interferences from other wireless devices. We placed the wireless devices next to each other since we wanted to focus the analysis on the integration instead of on the performance of the wireless link. More details on the w-SHARP performance can be found in [7], [23].

With this setup, we monitored the network status for 2 hours, which includes a total of $3.6 \cdot 10^5$ real-time frames, and around $1.4 \cdot 10^5$ best-effort frames. During the experiment, we collected the statistics of the network latency in each network hop and the Packet Error Rate (PER) of the Wireless link. The main performance figures are summarized in Table. 1. Also, a timing diagram of the real-time traffic going from the OPC UA publisher to the OPC UA subscriber is depicted in Fig. 7.

TABLE. 1. MEASURED LATENCY AND PER.

Traffic Type	Wireless link measured PER	Latency [ms]
PubSub traffic (real-time)	0%	0.9 – 1.2
Emulated OPC UA client traffic (Best-effort)	0%	4-8 (Typ.) 50 (worst-case)

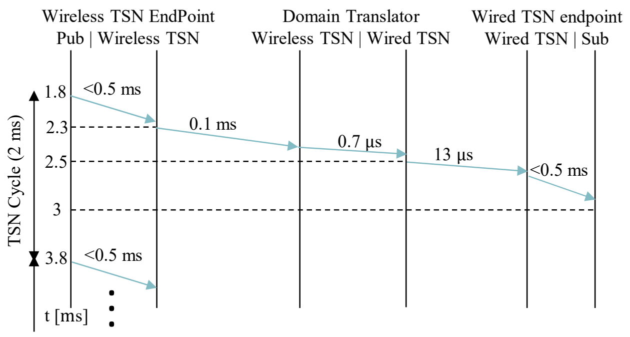

Fig. 7. Representation of a real-time packet moving along the network using a TSN flow.

Regarding the performance of the real-time traffic, we have seen that we are able to guarantee (including network stack) an end-to-end latency in the range of 0.9 to 1.2 ms, which is equivalent to the configuration that we applied to the network. In other words, the maximum latency deviation that we found was $\pm 150\ \mu s$. Since the network is totally deterministic by design and has no jitter, the root cause of the jitter is the software stack of the subscriber (Linux network and OPC UA stack). In addition to that, we have seen that no frames were lost during the experiment, which is in line with other results obtained in experiments described in other works [23].

Regarding the performance of the best-effort traffic, we have seen that the delay between the PC and Wireless TSN endpoint is most of the time stable and ranges between 4 to 8 ms of round-trip time. However, the delay of the best-effort traffic is not guaranteed by the network design and therefore we have (rarely) seen pings with latency up to 50 ms. There is one main phenomenon that explains the increased latency. The w-SHARP best-effort frames are transmitted using the standard IEEE 802.11 mechanism based on random access. This mechanism is based on the possibility that frame collisions can occur and includes a retransmission scheme. Therefore, any frame that requires one or more retransmissions due to collisions will have a significant increase in latency. That is also the reason why the PER of the ping is still 0% even when a collision can occur: the retransmissions ensure that there are no packet losses. On the other hand, the latency between the PC and the Wired TSN endpoint is very stable and in the order of 1 ms in any case.

V. CONCLUSIONS

Through this paper, we discussed the integration of OPC UA, a middleware protocol, on top of a Hybrid TSN deployment. We have proposed a HW/SW architecture and traffic mapping that allows seamless integration of OPC UA and Hybrid TSN while satisfying the application QoS needs. We developed a HW testbed to demonstrate the integration. The HW testbed was based on SHARP and the OPC UA open62541 stack. The results show that the integration of OPC UA and Hybrid TSN is feasible and that the proposed testbed can serve as a platform for further development of OPC UA over Hybrid TSN networks. In particular, we showed that OPC UA over the network can achieve a latency of around 1 ms even with a Linux-based SW stack.

However, there is still plenty of research to do in this area. We think that the next aspects must be addressed to tackle the current limitations. First, the Linux classic network stack may not be reliable enough for the sub-ms operation of OPC UA or other middlewares, especially under high network/processor load conditions. Therefore, tailor-made solutions for real-time communications over Linux, such as eXtreme Data Path (XDP) and Linux Traffic Control (qdisc), can be considered and evaluated. Another alternative to improve the latency guarantees is using a co-processor running a real-time SW to generate and consume the OPC UA PubSub data.

From the control plane perspective, OPC UA PubSub configuration can be used to feed TSN Centralized entities and to automatically configure/adapt the Wired/Wireless TSN traffic flows during network operation. Having automatic configuration is very relevant because it simplifies the deployment of the networks and eases maintenance and upgrading.

Finally, to the best of our knowledge, there are no commercially available devices that integrate OPC UA PubSub over TSN up to the day we did the experiments. Therefore, it could be interesting to test those devices over a Hybrid TSN network.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Spanish MINECO and the European Union through the Next Generation UNICO5G TIMING (TSI-063000-2021-145/148/149) project.

REFERENCES

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial Communication," *IEEE Industrial Electronics Magazine*, vol. 12, no. 4, pp. 370–376, 2017, doi: 10.1021/ie50124a022.

[2] Z. Pang, M. Luvisotto, and D. Dzung, "Wireless High-Performance Communications: The Challenges and Opportunities of a New Target," *IEEE Industrial Electronics Magazine*, vol. 11, no. 3, pp. 20–25, 2017, doi: 10.1109/MIE.2017.2703603.

[3] M. K. Atiq, R. Muzaffar, O. Seijo, In. Val, and H. P. Bernhard, "When IEEE 802.11 and 5G Meet Time-Sensitive Networking," *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, pp. 14–36, 2022, doi: 10.1109/OJIES.2021.3135524.

[4] A. Larrañaga, M. C. Lucas-Estañ, I. Martínez, I. Val, and J. Gozalvez, "Analysis of 5G-TSN Integration to Support Industry 4.0," *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020.

[5] D. Cavalcanti, J. Perez-Ramirez, M. M. Rashid, J. Fang, M. Galeev, and K. B. Stanton, "Extending Accurate Time Distribution and Timeliness Capabilities Over the Air to Enable Future

Wireless Industrial Automation Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1132–1152, Jun. 2019, doi: 10.1109/JPROC.2019.2903414.

[6] S. Sudhakaran, K. Montgomery, M. Kashef, D. Cavalcanti, and R. Candell, "Wireless Time Sensitive Networking Impact on an Industrial Collaborative Robotic Workcell," *IEEE Trans Industr Inform*, vol. 18, no. 10, pp. 7351–7360, Oct. 2022, doi: 10.1109/TII.2022.3151786.

[7] O. Seijo, X. Iturbe, and I. Val, "Tackling the Challenges of the Integration of Wired and Wireless TSN With a Technology Proof-of-Concept," *IEEE Trans Industr Inform*, vol. 18, no. 10, pp. 7361–7372, Oct. 2022, doi: 10.1109/TII.2021.3131865.

[8] C. Cruces, R. Torrego, A. Arriola, and I. Val, "Deterministic Hybrid Architecture with Time Sensitive Network and Wireless Capabilities," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2018-Septe, pp. 1119–1122, 2018, doi: 10.1109/ETFA.2018.8502524.

[9] C. Eymuller, J. Hanke, A. Hoffmann, M. Kugelmann, and W. Rif, "Real-time capable OPC-UA Programs over TSN for distributed industrial control," *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020.

[10] P. Denzler, T. Fruhwirth, D. Scheuchenstuhl, M. Schoeberl, and W. Kastner, "Timing Analysis of TSN-Enabled OPC UA PubSub," in *2022 IEEE 18th International Conference on Factory Communication Systems (WFCS)*, Apr. 2022, pp. 1–8. doi: 10.1109/WFCS53837.2022.9779177.

[11] O. Konradi, A. Mankowski, L. Wisniewski, and H. Trsek, "Towards an Industrial Converged Network with OPC UA PubSub and TSN," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2022, pp. 1–4. doi: 10.1109/ETFA52439.2022.9921646.

[12] A. Eckhardt, S. Muller, and L. Leurs, "An Evaluation of the Applicability of OPC UA Publish Subscribe on Factory Automation use Cases," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2018, pp. 1071–1074. doi: 10.1109/ETFA.2018.8502445.

[13] S. Sudhakaran, V. Mageshkumar, A. Baxi, and D. Cavalcanti, "Enabling QoS for Collaborative Robotics Applications with Wireless TSN," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, Jun. 2021, pp. 1–6. doi: 10.1109/ICCWorkshops50388.2021.9473897.

[14] P. Bellavista, A. Corradi, L. Foschini, and A. Pernafini, "Data Distribution Service (DDS): A performance comparison of OpenSplice and RTI implementations," in *2013 IEEE Symposium on Computers and Communications (ISCC)*, Jul. 2013, pp. 000377–000383. doi: 10.1109/ISCC.2013.6754976.

[15] "OPC-UA Foundation," *https://opcfoundation.org/about/opc-technologies/opc-ua/*, Feb. 2023.

[16] C. Scordino, A. G. Marino, and F. Fons, "Hardware Acceleration of Data Distribution Service (DDS) for Automotive Communication and Computing," *IEEE Access*, vol. 10, pp. 109626–109651, 2022, doi: 10.1109/ACCESS.2022.3213664.

[17] L.-D. Chiorean, M.-F. Vaida, and H. Hedesiu, "Modelling an OPC UA client application for predictive maintenance support," in *2020 International Symposium on Electronics and Telecommunications (ISETC)*, Nov. 2020, pp. 1–4. doi: 10.1109/ISETC50328.2020.9301058.

[18] Z. Li *et al.*, "Time-Triggered Switch-Memory-Switch Architecture for Time-Sensitive Networking Switches," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 185–198, Jan. 2020, doi: 10.1109/TCAD.2018.2883996.

[19] P. Denzler, M. Ashjaei, T. Fruhwirth, V. N. Ebirim, and W. Kastner, "Concurrent OPC UA information model access, enabling real-time OPC UA PubSub," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2022, pp. 1–4. doi: 10.1109/ETFA52439.2022.9921438.

[20] S. Gruner, A. E. Gogolev, and J. Heuschkel, "Towards Performance Benchmarking of Cyclic OPC UA PubSub over TSN," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2022, pp. 1–8. doi: 10.1109/ETFA52439.2022.9921503.

[21] I. Val, O. Seijo, R. Torrego, and A. Astarloa, "IEEE 802.1AS Clock Synchronization Performance Evaluation of an Integrated Wired–Wireless TSN Architecture," *IEEE Trans Industr Inform*, vol. 18, no. 5, pp. 2986–2999, May 2022, doi: 10.1109/TII.2021.3106568.

[22] SoC-e, "SoC-e Multiport Time-Sensitive Networking (MTSN) switch." https://soc-e.com/mtsn-multiport-tsn-switch-ip-core/ (accessed Sep. 01, 2020).

[23] Ó. Seijo, J. A. López-fernández, I. Val, and S. Member, "w-SHARP : Implementation of a High-Performance Wireless Time-Sensitive Network for Low Latency and Ultra-Low Cycle Time Industrial Applications," *IEEE Trans Industr Inform*, 2020.

[24] "ptp4l stack." https://github.com/richardcochran/linuxptp (accessed Feb. 15, 2023).

[25] "open62541 OPC UA stack website." https://www.open62541.org/ (accessed Feb. 15, 2023).