A unified method to design bridges for OPC UA PubSub networks in the industrial IoT

Quang-Duy Nguyen, Saadia Dhouib, Patrick Bellot

A Unified Method to Design Bridges for OPC UA PubSub Networks in the Industrial IoT

Quang-Duy NGUYEN*, Saadia DHOUIB*, and Patrick BELLOT†
*Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
†LTCI, Télécom Paris, Institut Polytechnique de Paris, F-91120, Palaiseau, France
Email: quang-duy.nguyen@cea.fr, saadia.dhouib@cea.fr, patrick.bellot@telecom-paris.fr

***Abstract*—Specification part 14 of the Open Platform Communication Unified Architecture (OPC UA) standard provides five different profiles to implement the publish-subscribe messaging pattern. The specification is also called OPC UA PubSub, and its profiles are called PubSub profiles. Two devices deployed with the same PubSub profile can exchange and collaborate; however, two devices deployed with two different PubSub profiles are unable to communicate. It is a limit for the Industry Internet of Things, a complex environment where there would be heterogeneous devices and networks. One approach to overcoming this issue is to use a bridge for the devices deployed with different PubSub profiles. In this sense, this paper provides a unified method to design bridges for OPC UA PubSub networks. The proof-of-concept experiment, also presented in this paper, is a use case of bridging PubSub broker-less and broker-based networks.**

***Index Terms*—Industry, IoT, OPC UA, PubSub, Bridge, Design, Broker-based, Broker-less, Interoperability**

I. Introduction

Recently, Open Platform Communication Unified Architecture (OPC UA) has become a high-demand standard for building industrial systems. Its specification part 14, also known as OPC UA PubSub, presents the details to implement the publish-subscribe messaging pattern [1]. This pattern is an asynchronous data exchange mechanism, in which some components of a system work as subscribers, and some work as publishers. A publisher manages data sources, generates data, and sends them to subscribers. A subscriber needs to subscribe to a data source once and receive newly generated data as soon as available. OPC UA PubSub proposes two communication modes: broker-less and broker-based. The broker-less mode uses the multicast approach to spread new data from publishers to subscribers. This approach relies on the UDP/IP network protocols stack and requires only common network infrastructures that support the multicast feature. In detail, when a publisher sends a message to a multicast address, the network infrastructure forwards the message to all devices in the network; however, only the devices already subscribed to the multicast address can process it. The broker-based mode requires another computing device playing as a broker. The broker's first mission is to manage topics. A topic includes information of data sources and the details required to create links between publishers and subscribers. Its second mission is to receive all data from publishers and forward them to corresponding subscribers.

OPC UA PubSub defines five profiles[1] for the two communication modes: one profile for PubSub broker-less mode and four others for PubSub broker-based mode. Each PubSub profile contains two conformance units: a message mapping and a transport protocol. A message mapping is a guide to organizing data in a structured format. It is required by the serialization process for data encoding/decoding. Two types of message mapping proposed by the OPC UA PubSub are JSON and UADP. The transport protocol specifies the protocol used for data exchange in an OPC UA PubSub system. Their supported protocols are PubSub UDP, MQTT, and AMQP. Two devices implemented by the same PubSub profile join the same network and can exchange and collaborate. This network can be called a PubSub network. Unfortunately, two devices deployed with two different PubSub profiles cannot communicate directly. It becomes a limitation for the OPC UA system to work in the Industry Internet of Things (IIoT). Indeed, the IIoT is an industrial-specific case of the Internet of Things (IoT), a scenario where "people and things are connected anytime, anyplace, with anything and anyone, ideally using any network and any services" [2]. In such complex environment, an OPC UA PubSub system in the IIoT should support heterogeneous devices communicated via multiple PubSub networks.

One approach to improving the above limitation is using a network device to bridge different PubSub networks. This network device is called a PubSub bridge. This paper aims to present a unified method to design PubSub bridges.

The organization of this paper is as follows. The second section identifies the characteristics of a bridge compared to a forwarder and a gateway. The third section presents some works related to this research. The fourth section focuses on the first contribution of this paper: a unified design method dedicated to PubSub bridges. The second contribution, presented in the fifth section, specifies a sample of using the design method: a bridge between a PubSub broker-less network and a PubSub broker-based network. Finally, a brief conclusion summarizes this paper and opens a discussion.

II. Background

The most fundamental mission of a bridge is to guarantee that devices can exchange and collaborate with other devices while they have different connection mediums, use different

[1] https://profiles.opcfoundation.org/profilefolder/320

network protocols, or have different configurations. However, besides bridges, some other network devices, such as gateway or forwarder, seem to have the same goal. This paper calls these devices by the generic term "inter-connectors" and divides them into three categories as follows. First, a forwarder is a device that forwards messages from one network to another. There is no transformation process at the forwarder; in other words, the input message is the same as the output message. Second, a bridge is a device that can transform one network message into another network message and forward them from one network to another. The information carried in the two messages are unchangeable. Third, a gateway is a device that can transform messages and forward them over networks, and even more, the transformed message may contain other information different from the information received before the transformation process. In detail, one or several applications run by the gateway may process input information to produce a new output. Figure 1 illustrates the difference between a forwarder, a bridge, and a gateway.

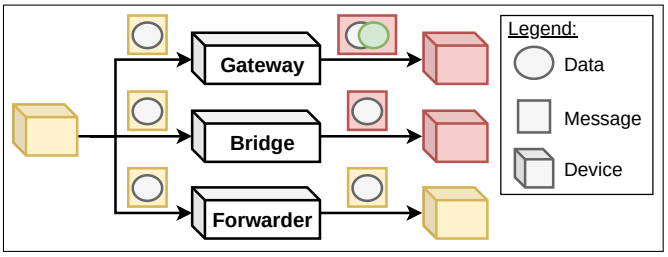

Fig. 1. Three categories of inter-connectors: forwarder, bridge, and gateway

All three types of inter-connectors contribute to the interoperability of devices by establishing the communications between them. To distinguish a bridge from a forwarder and a gateway, this paper proposes to project them on a stack of interoperability of four layers, as shown in Figure 2. This stack is inspired by the similar stack in the thesis of Nguyen [3].

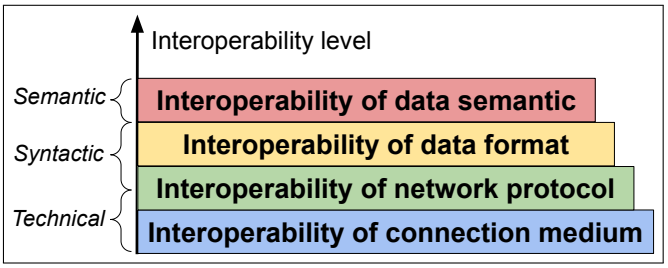

Fig. 2. Stack of Interoperability between devices

Interoperability of connection medium is when two devices exchange and process signals. The physical media carrying signals can be electrical, optical, or wave. When two devices use the same physical media, one plays the role of a sender, and the other plays the role of a receiver, or both can be receiver-sender. When the two devices use two different physical media, the inter-connector in the middle must support both physical media. For example, an inter-connector connects a device by cable and another device using its wireless antenna. Inside the inter-connector, signals are two-sides converted by software or hardware components. The forwarder, bridge, and gateway must support this interoperability level.

Interoperability of network protocol: is when two devices use a set of network protocols to ensure that data sent from one side can be received and processed on another side over the network. Each protocol has a different goal, but they all provide rules to organize data into their pre-defined protocol data unit (PDU). Some protocols even have rules not only for one data but also for data sources, such as the topic mechanism of the MQTT protocol [4]. In this sense, they contribute to syntactic interoperability. Some protocols may guarantee other network requirements, such as the quality of service and security. Thus, they contribute to technical interoperability. When the two devices use two sets of different network protocols, the inter-connector in the middle must support both the two sets. For example, an inter-connector has an interface that supports the set of protocols of the UDP/IP stack for one device and has another interface for the set of the TCP/IP stack for another device. It must establish a mechanism to redirect data flows between the two interfaces. The forwarder, bridge, and gateway must support this interoperability level.

Interoperability of data format: is when a device can extract information received from another device due to an agreement in structured data format sharing between them. The data format can be the order of pieces of information, or it follows a convention of using markup keywords that helps distinguish and recognize pieces of information. When two devices use two different message mappings, the inter-connector in the middle must support both and have a strategy to transform data from one format to another. For example, a bridge can support the UADP message mapping for one device and support the JSON message mapping for another device. Then, it can transform data from UADP into JSON format and vice versa. Only the bridge and gateway support this interoperability level.

Interoperability of data semantic: is when two devices share the same information model so that they can understand the meaning of the exchanged information and use them correctly in applications. When two devices have different information models and run different applications, an inter-connector in the middle runs applications that can understand the input information of one side, operates complex computations, such as aggregation and reasoning, and produce output usable for the other side. Many IoT gateways support these complex computations [3]. Only the gateway supports this interoperability level.

To recap: while designing a bridge, it is necessary to consider the three lower layers of the stack of interoperability.

III. Related Work

Several commercial OPC UA inter-connectors are available in the market. Some of them are products of industrial members of the OPC Foundation[2]. However, to the best of our

[2]https://opcfoundation.org/products

knowledge, no academic research analyzes and conceptualizes the characteristics of inter-connectors to produce a unified design method dedicated to OPC UA PubSub bridges.

The MQTT-SN standard, a version of the MQTT standard dedicated to sensor networks, provides a network solution for the IoT with some new-defined inter-connectors [5]. In detail, it has some modifications dedicated to constrained devices, such as reducing the protocol header's size, using UDP instead of TCP/IP as in MQTT, and shortening topics by replacing long text with two-bytes identification. To afford the above advanced, it defines two new network devices. First is the data forwarder that allows extending the distance between devices. This device is in the forwarder category of Section II. The second device is the MQTT-SN gateway. On the one hand, the MQTT-SN gateway plays as an MQTT client to maintain the connection with an MQTT broker and be a part of the corresponding MQTT network. On the other hand, it receives UDP messages from the MQTT-SN network. This inter-connector converts UDP to TCP and maps MQTT-SN topics with MQTT topics. Thus, the MQTT-SN gateway corresponds to a bridge in our definition.

As in this research, Dave et al. [6], [7] provide a concept of interoperability in the IoT and rely on it to classify the complexity of IoT systems. Our vision is different from theirs on two points. First, our interoperability stack serves as a tool to distinguish different network inter-connectors. Second, the final goal of this research is a unified method to design bridges dedicated to OPC UA PubSub networks.

IV. OPC UA PubSub Bridge Design Method

A PubSub bridge has two principles. First, it can join multiple networks, so it must have multiple interfaces. In each interface, it plays the role of both a publisher and subscriber. For example, a PubSub bridge having two interfaces, A and B, can subscribe to topics, receive and process messages at interface A, then publish processed messages to interface B. It acts the same in the reversed sense from interface B to A. The second principle is that the PubSub bridge must respect the OPC UA standard. In other words, while designing a bridge for PubSub networks, the golden rule is to adopt the network protocols (transport and security protocols) and message mappings defined in the OPC UA profiles. However, bridge designers have no constraint when choosing communication mediums to integrate into PubSub bridges.

Figure 3 presents the basic steps to develop a PubSub bridge. The steps are grouped by the three lower layers of the interoperability stack presented in Section II. The red stars are pinned at the corner of the steps that must strictly follow the golden rule. The arrows show the flow to implement the steps.

1.1 Communication Medium Verification: is the step to verify the communication mediums of the PubSub networks in which the bridge will join. When the PubSub bridge has network interface cards that support these mediums, then this step is valid.

1.2 Communication Medium Setup: is the step to set up and ensure that a PubSub bridge can exchange signals with other devices in a PubSub network using a communication medium. It can be a simple action, such as turning on the Wi-Fi. It is possible to use portable network interface cards when necessary.

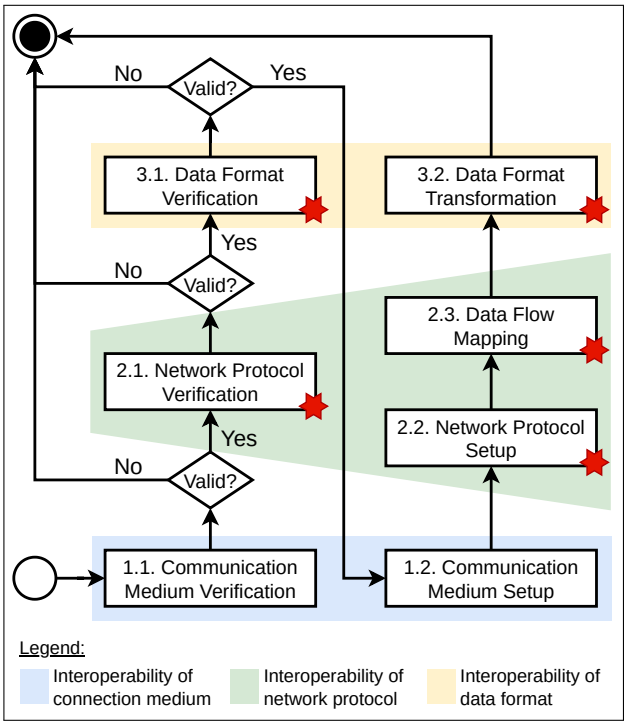

Fig. 3. Steps to design and develop a PubSub bridge

2.1 Network Protocol Verification: is the step to verify the network protocols of the PubSub networks in which the PubSub bridge will join. As defined in OPC UA PubSub, the transport protocols that the PubSub bridge can support are the protocols of TCP/IP and UDP/IP stacks, MQTT protocol, and AMQP protocol. The security protocols that the bridge can support are AES128-CTR and AES256-CTR. When the networks use the mentioned protocols for communications, this step is valid.

2.2 Network Protocol Setup: is the step to deploy one or some of the above network protocols in an interface of the bridge to ensure that the PubSub bridge can exchange data with other devices in the corresponding network.

2.3 Data Flow Mapping: is the step to redirect the data flows from an interface to other interfaces of the PubSub bridge. Bridge designers need to define mapping rules for them.

3.1 Data Format Verification: is the step to verify the message mapping methods of the networks in which the bridge will join. The message mapping methods that the PubSub bridge supports are UADP and JSON. When the PubSub networks use these two message mappings to format data, this step is valid.

3.2 Data Format Transformation: is the step to transform the data format required by the serialization at an interface into another data format required by the serialization at another interface. This step can be ignored when two sides use the same message mapping.

Following the arrows, the three verification steps, (1.1), (2.1), and (3.1), should be performed first. This approach allows bridge designers to save time by quickly recognizing if all conditions to make a PubSub bridge for their target networks are satisfied. If all are valid, they can step by step follow (1.2), (2.2), (2.3), and (3.2).

V. Bridging PubSub UDP UADP and PubSub MQTT UADP Networks

This section presents a use case of our bridge design method: a PubSub bridge for PubSub UDP UADP profile[3] and PubSub MQTT UADP profile[4] networks. This former profile is for PubSub broker-less mode, and the latter one is a profile of broker-based mode. The two first subsections briefly describe the basic concepts of the two profiles. The third subsection focuses on the proof-of-concept experiment and its results.

A. PubSub UDP UADP Profile

PubSub UDP UADP comprises the UADP message mapping for serialization and the UDP protocol for transport. While UDP is quite popular as a part of the UDP/IP stack, UADP is a specific message mapping defined in the OPC UA specification part 14. UADP stands for UA Datagram Protocol. UADP message mapping is a guide for binary data encoding/decoding. The application data after the UA binary encoding is encapsulated to become a UADP network message. Then, the UADP network message can be put into a datagram, such as a UDP datagram, to become a network message. On the receiver side, the UADP network message is extracted from the network message. The UA binary decoding uses UADP message mapping to decode the datagram for the application data. Figure 4 illustrates the relation of UADP message mapping with the UA data encoding/decoding. Note that the message-oriented middleware (MOM) in this figure can be a broker in the broker-based mode and be a network infrastructure supporting the multicast address mechanism in the broker-less mode.

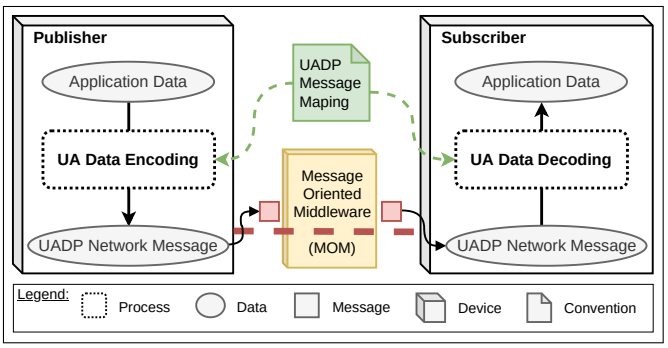

Fig. 4. Data exchange using UADP message mapping of OPC UA PubSub

Three fundamental parameters that a UADP network message requires are *PublisherId*, *WriterGroupId*, and *DataSetWriterId*. From now on, this paper uses the term "triple of UADP IDs" to mean these three parameters. A subscriber can further process a UADP network message based on the triple of UADP IDs. In detail, *PublisherId* is the identifier of a publisher, that is, a device that sent the UADP network message. *DataSetWriterId* is the identifier of a dataset writer, that is, the data source. *WriterGroupId* is the identifier of an abstracted group of dataset writers sharing one or several similar features.

PubSub UDP is about reusing the multicast addresses to publish data. For example, a group of subscribers can listen to the multicast address *224.0.0.2* to wait for data from a data source. The publisher managing this data source can send data to the subscribers by publishing the data to *224.0.0.2*.

B. PubSub MQTT UADP Profile

PubSub MQTT UADP profile comprises the UADP message mapping for serialization and the MQTT protocol for transport. Since Subsection V-A has already introduced the principle of the UADP message mapping, the following focuses on the concept of the MQTT protocol. It is a lightweight open messaging protocol for the broker-based mode [4]. In MQTT specification, the broker is an MQTT server that serves its MQTT clients, which are publishers and subscribers. It defines several messages to manage the communication between a broker, and publishers, subscribers. Each message has a meaning and contains different information. The communication between a broker and MQTT clients relies on the TCP/IP standard. The seven basic MQTT messages are CONNECT, CONNACK, PINGREQ, PINGRESP, PUBLISH, SUBSCRIBE and DISCONNECT. The couple CONNECT and CONNACK messages are for creating a connection between a broker and MQTT clients. This connection is called the MQTT connection. The couple PINGREQ and PINGRESP are to update the keep-alive value to maintain the MQTT connection between a broker and its clients. The SUBSCRIBE message is from a subscriber towards the broker to subscribe to topics. The PUBLISH message sent from a publisher to the broker contains published data. The DISCONNECT message is used to end a MQTT connection.

C. Use Case

This use case relies on the need for the LocalSEA testbed of CEA List [8]. All the robots and devices of LocalSEA connect locally inside an OPC UA PubSub broker-less network. The testbed cannot satisfy the new working scenarios which require the devices to publish data over the Internet using MQTT protocol. Thus, the strategy is to develop a bridge for the PubSub UDP UADP and PubSub MQTT UADP networks.

Our system developers then follow the method in Section IV. In step (1.1), our developers verify that all devices in LocalSEA have a port RJ45 for Ethernet cable connections and a standard Wi-Fi antenna for wireless connections. We choose a laptop to work as a PubSub bridge. Since the laptop supports the two types of connection mediums, the first step is valid. Steps (2.1) and (3.1) are also valid as, from the beginning, the development strategy of the testbed is to adopt

[3]http://opcfoundation.org/UA-Profile/Transport/pubsub-udp-uadp
[4]http://opcfoundation.org/UA-Profile/Transport/pubsub-mqtt-uadp

all network protocols of the OPC UA standard. Step (1.2) is to branch an Ethernet cable to the laptop. Its Wi-Fi antenna is always on. Next, in step (2.2), our developers install the library and develop programs so that the laptop can work as both a publisher and a subscriber for both required PubSub profiles. The network interface for the PubSub UDP UADP profile communications has wired connections (eth0), and the network interface for the PubSub MQTT profile communications has wireless connections (wlan0). In order to redirect the data flows from wlan0 to eth0 and vice versa, it is necessary to map MQTT topics to triples of UADP IDs. The ideal candidate to standardize the mapping rules is to use the Sparkplug B standard [9]. It defines the rules to define MQTT topics and payloads for use in the industry. A Sparkplug B topic is a text composed of five elements. In which, Sparkplug version (`namespace`) and `message type` are selective elements. In other words, users can only select terms in the vocabulary defined in Sparkplug B but cannot create a new one. The other three: `groud id`, `edge node id`, and `device id`, are open for users to define. It is possible to associate them with the triple of UADP IDs. In detail, our developers map *PublisherId* to `edge node id`, *WriterGroupId* to `group id`, and *DatasetWriterId* to `device id`. In this sense, a triple of UADP IDs can be converted into an MQTT topic following the SparkPlug B topic format, and vice versa. In the final step (3.2), since both PubSub profiles use UADP data format, no data format transformation is required. Figure 5 illustrates our PubSub bridge. Note that the PubSub bridge and MQTT broker can run on one device, as in our scenario, or can run on two separate devices. In this paper, we call the device in the first case a BnB (Bridge & Broker) device. The PubSub bridge subscribes to triples of UADP Ids in the network interface against the local wireless network, and subscribes to topics in the interface connecting to the MQTT broker.

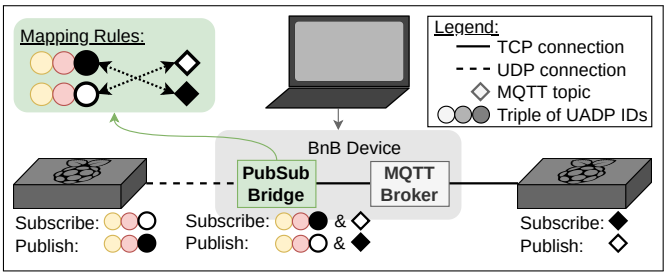

Fig. 5. PubSub bridge for PubSub UDP UADP and MQTT UADP networks

In order to evaluate the PubSub bridge, our developers design a test case as in Figure 6. In this test case, there are five main elements. First, an OPC UA server plays the role of a publisher that publishes a data value periodically every five seconds. The data value increases by 0.1 after each publication. Next, three devices, called echoer 1, echoer 2, and echoer 3, have three different paths to connect to the OPC UA server. Their mission is to receive and modify the data value and repeat the modified data value back to the OPC UA server. To realize the idea, they perform the following jobs: (1) listen to receive network messages, (2) extract the data value from each received network message, (3) add a number to it to produce a new data value, (4) put the new data value and a new creation timestamp in a new message, and (5) send the new network message back to the OPC UA server. In job (3), three echoers add three different numbers to the received data. The final element is a BnB device that guarantees communications between the OPC UA server and the three echoers.

Technically, the OPC UA server publishes to the triple of UADP Ids *{101,1001,10002}* and subscribes to the three triples of UADP Ids: *{101,1001,10001}*, *{101,1001,10011}*, *{101,1001,10021}*. Inside the BnB device, the PubSub bridge maps the four mentioned triples of UADP Ids to four topics. It subscribes to the four topics on the connection side with the MQTT broker and the four triples of UADP Ids on the side with the OPC UA server. Echoer 1 is an external device of the LocalSEA's network but is an internal device of the CEA List's network. Echoer 2 is an internal device of the LocalSEA's network. Both echoers use PubSub MQTT UADP profile and subscribe to the topic mapped to the triple of UADP Ids *{101,1001,10002}* at the BnB device's MQTT broker. Echoer 3 is an internal device of LocalSEA, and it connects directly to the OPC UA server through an Ethernet cable connection. It uses the PubSub UDP UADP profile and can process UADP messages containing the triple of UADP Ids *{101,1001,10002}*. In other words, Echoer 3 subscribes directly to the triple of UADP Ids *{101,1001,10002}*.

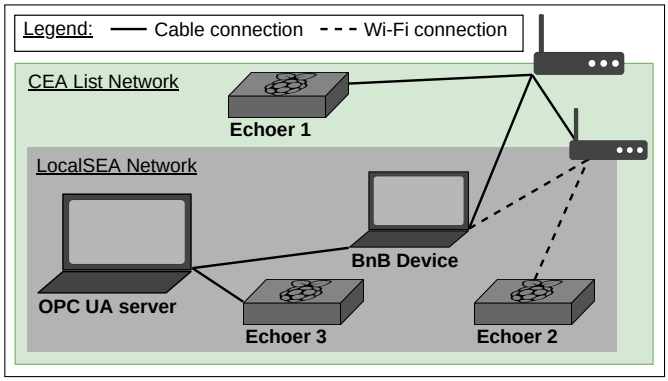

Fig. 6. Physical Architecture of the experimentation

After launching the test scenario, we can access the address space of the OPC UA server using UaExpert[5]. The three OPC UA variable nodes *data_echo_1*, *data_echo_2*, *data_echo_3* are created to hold the data returned respectively from echoer 1, echoer 2, and echoer 3. The historical values of each variable node are recorded in the History Data window, as in Figure 7. With each variable node, there are nine instances. Each instance has a data value projected on the vertical Y-axis and a message creation timestamp projected on the horizontal X-axis. Table I shows the detail of these timestamps.

The above result has three meanings. First, it is obvious to conclude that the PubSub bridge produced from our design method works. Second, time records in Table I show that the delay of the echoer 3 is the smallest. The delay of echoer

[5]https://unified-automation.com/products/development-tools/uaexpert.html

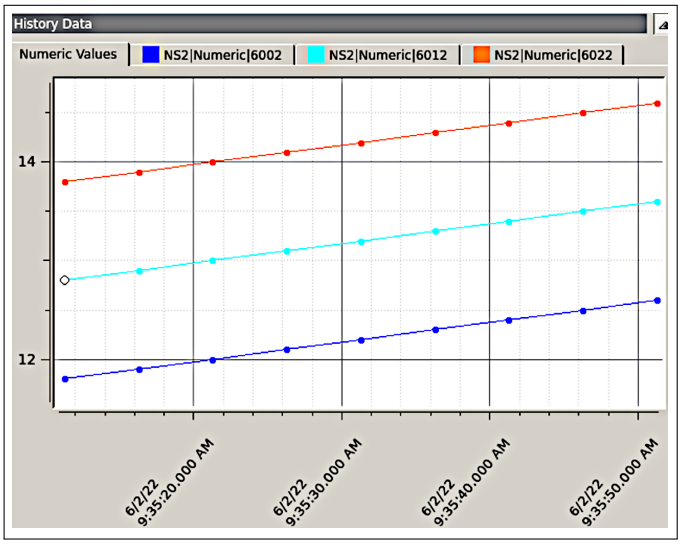

Fig. 7. History data received from the three echoers shown on UaExpert

TABLE I
TIME WHEN THE ECHOERS CREATE NEW A MESSAGE TO SEND TO THE OPC UA SERVER (SOURCE TIMESTAMP)

No.	t(Echoer 1)	t(Echoer 2)	t(Echoer 3)
1	09:35:11.264	09:35:11.272	09:35:11.264
2	09:35:16.265	09:35:16.272	09:35:16.264
3	09:35:21.265	09:35:21.272	09:35:21.264
4	09:35:26.265	09:35:26.272	09:35:26.264
5	09:35:31.265	09:35:31.272	09:35:31.264
6	09:35:36.265	09:35:36.272	09:35:36.264
7	09:35:41.265	09:35:41.272	09:35:41.265
8	09:35:46.265	09:35:46.273	09:35:46.265
9	09:35:51.266	09:35:51.273	09:35:51.265

1 is nearly the same as echoer 3; sometimes, they have a tiny difference of 0.001 seconds. The delay of the echoer 2 is the largest. Its gap from the delay of the echoer 3 is about 0.008 seconds. The reason for such differences is that connection mediums between echoer 1 and the OPC UA server, and between echoer 3 and the OPC UA server are wired cables, while the connection mediums between echoer 2 and the OPC UA server are a mix of wired and wireless. Then, it is possible to conclude that the transmission of wired networks is still slightly better. Third, with the same conditions of communication mediums and a small-scale scenario, there is nearly no transmission difference between a PubSub UDP UADP network and a PubSub MQTT network.

VI. CONCLUSION AND DISCUSSION

In conclusion, this paper presents a unified method to design bridges dedicated to OPC UA PubSub networks. To determine the requirements of a bridge, we projected it into a stack of interoperability and compared it to other similar network device types. A use case of designing a PubSub bridge for two PubSub profiles: UDP UADP and MQTT UADP, and using it in the LocalSEA testbed, is proof of our concept. This bridge becomes a part of the testbed's toolset.

This paper has two points to discuss further. First, we plan to upgrade the LocalSEA testbed with the 5th generation mobile network (5G) technologies. In theory, our design method can help communicate new 5G devices with the old ones. However, since the OPC Foundation and the 5G Alliance for Connected Industries and Automation are collaborating to produce new specifications for OPC UA over industrial 5G, it is necessary to consider their updates before taking action.

Second, the implementation of the PubSub bridge, presented in Section V-C, profits from some open-source libraries, including open62541[6] and Eclipse Mosquitto [7]. Since several implementations and libraries for the AMQP transport protocol are also available; thus, it is possible to reuse such tools to realize this conformance unit. Unfortunately, to the best of our knowledge, there is not yet any tool that supports the data format transformation step (3.2) between OPC UA PubSub UADP and JSON. Then, it can be challenging to implement the PubSub bridges that require this feature.

Our near future goal is to automate the network protocol setup (3.2) step. Following the SysML-based model-driven approach, we aim to model communications between devices using SysML internal block diagrams (IBD). The connectors between the blocks in the IBD will be refined to specify network protocols. Theoretically, when the system model has all network protocols required by a bridge, it is possible to generate the bridge automatically from the SysML model. This work will be a plugin for Eclipse Papyrus[8], an open-source model-based engineering tool developed by CEA List.

ACKNOWLEDGMENT

This work is partially funded by DIMOFAC, an EU Horizon 2020 research and innovation programme under grant agreement N°870092.

REFERENCES

[1] OPC Foundation, "OPC Unified Architecture - Part 14: PubSub," Industry Standard Specification OPC 10000-14, 2018.
[2] International Telecommunication Union, "ITU Internet Reports - The Internet of Things," Nov. 2005.
[3] Q.-D. Nguyen, "Interoperability and Upgradability Improvement for Context-Aware Systems in Agriculture 4.0," Ph.D. dissertation, Université Clermont Auvergne, Aubière, France, 2020.
[4] Oasis, "MQTT Version 3.1.1," OASIS Standard v3.1.1, Sep. 2014.
[5] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN) Protocol Specification," IBM Corporation, Technical report v1.2, 2013.
[6] M. Dave, M. Patel, J. Doshi, and H. Arolkar, "Ponte Message Broker Bridge Configuration Using MQTT and CoAP Protocol for Interoperability of IoT," in *Computing Science, Communication and Security*. Singapore: Springer Singapore, 2020, vol. 1235, pp. 184–195.
[7] M. Dave, J. Doshi, and H. Arolkar, "MQTT- CoAP Interconnector: IoT Interoperability Solution for Application Layer Protocols," Oct. 2020, pp. 122–127.
[8] Q.-D. Nguyen, F. Tmar, Y. Huang, and S. Dhouib, "Early lessons learned from the development of a local OPC UA-based robotic testbed for research," in *IEEE 31st International Symposium on Industrial Electronics*, Anchorage, Alaska, United States, Jun. 2022, pp. 1–4.
[9] Eclipse Foundation, "Sparkplug : MQTT Topic & Payload Definition," Technical report v2.2, 2019.

[6]http://www.open62541.org/
[7]https://mosquitto.org/
[8]https://www.eclipse.org/papyrus/