Performance comparison between OPC UA and MQTT for data exchange

Murilo Silveira Rocha, Guilherme Serpa Sestito, Andre Luis Dias,
Afonso Celso Turcato and Dennis Brandão

Department of Electrical and Computing Engineering
São Carlos School of Engineering, University of São Paulo
Email: murilo.silveira.rocha@usp.br

***Abstract*—With the new concepts brought by the Industry 4.0 and the Internet of Things (IoT), it is expected more equipment communicating with each other, using local and cloud computing servers. In order to the communication between the devices be efficient, fast and with time deterministic, it is necessary to choose a suitable network protocol. In this context, the article presents the performance comparison of quantity of used data and time spent to send and receive messages in two well-known protocols in the field of IoT and Industrial IoT (IIoT): MQTT and OPC UA. The obtained values show the advantages of using each protocol in several scenarios, in order to exploit the best data exchange between devices.**

***Keywords*—OPC UA, MQTT, M2M, IoT, Industry 4.0, Procotol Comparison, Data Exchange**

I. Introduction

Seeking more efficient and advanced manufacturing as a goal, industries are undergoing a process of digital transformation, better known as Industry 4.0. It will become common the presence of sensors, actuators, controllers and supervisory systems not in a vertical communication structure, as the usual automation pyramid, but in a mesh communication structure, where all devices would be able to communicate with each other, thus providing a way for collecting information and extract knowledge about the process through the use of intelligent systems and big data techniques [1].

The massive amount of data exchanged between devices will require the use of machine-to-machine (M2M) protocols which should be efficient, deterministic, reliable and light enough to allow use in embedded systems [2].

As well as in established industrial networks, which are robust and have several papers showing applications to improve their use and estimate communications times [3] [4] [5], the protocols that will start to appear in industrial equipment that uses the Internet to communicate with cloud servers should have well-established metrics, mainly related to time, to ensure a good communication. [6]

In this context, this work aims to compare two widely used protocols in Industry 4.0 and IoT applications: the OPC UA and MQTT. The characteristics of each protocol are discussed and time aspects for sending and receiving messages in different scenarios through publish/subscribe model is analyzed, in order to determine their advantages for applications in many cases.

The paper is divided as follows: Section 2 presents both protocols studied, showing basic operation concepts for understanding the proposed tests. Section 3 explains each proposed test ans its purpose. Section 4 analyzes tests results for each protocol. Finally, section 5 presents an overall conclusion of the comparison of OPC UA and MQTT protocols.

II. Fundamentals for the Proposed Methodology

For the understanding of the proposed tests, this section explains the operation of the publish/subscribe model employed in data exchange of both protocols, which are OPC UA and MQTT.

A. *Publish/Subscribe*

The publish/subscribe communication model facilitates communication between two or more devices by organizing data exchange using topics. The equipment which is interested in receiving the information (subscribers) subscribe to the topic, and waits for the arrival of new messages. The equipment which wishes to send information (publishers) publishes in the topic of interest. Published messages distribution to subscribers equipment are done through the server, also called broker [7].

B. *OPC UA*

The OPC Unified Architecture (OPC UA) is widely used in industry applications to interconnect equipment present in different networks or at different levels of the automation pyramid. This protocol provides object-oriented data modeling, as well as an organization of the address space of these objects inside the server [8].

It allows the creation of variables of certain data types within each object on the server, allowing clients to read variables values or to subscribe to variables which it would like to receive all updated values. [9]

Despite offering many other functionalities, the main objective of this article is restricted on data exchange in publish/subscribe model. Therefore, it will not be considered all other features provided by the protocol.

978-1-5386-2497-5/18/$31.00 ©2018 IEEE

C. MQTT

The Message Queue Telemetry Transport (MQTT) protocol is widely used in residential, office, healthcare and smartphone applications since it provides a connection with lower latency and power consumption[10]. There is no differentiation of data type or organization in the protocol, being task of its clients all the communication process with each other. Its operation is entirely based on the publish/subscribe model, in which it is not possible to read the variable at the moment desired by the client, but only when there is a publication in the topic in which it is registered [11].

There is the definition of Quality of Service (QoS) in the protocol, which is an agreement between the sender and the recipient of the message as to the certainty of its delivery. In QoS 0, there is no guarantee that the sent message will reach the destination, as seen in Fig. 1-a. In QoS 1, there is a guarantee that the sent message arrives at the destination at least once, which the recipient responds with a message back to the sender that it was received, as shown in Fig. 1-b.

In QoS 2 there is a guarantee that the message will be delivered only once to the recipient. For this purpose, four messages should be exchanged: the message which will be delivered to the recipient, the recipient acknowledgment receipt, the confirmation request which the sender could release the message, and the recipient final confirmation that sender could release the message [12], as shown in Fig. 1-c.

III. Methodology of the Proposed Tests

This section presents the proposed tests for the protocols comparison. Scenarios were created in means that both protocols have the same procedure, using only the publish/subscribe model to exchange data and the three QoS present in the MQTT.

A. Test 1: Ratio between packet and payload length

In order to measure the relationship between the number of bytes of the payload and the total number of bytes exchanged for delivering a message, this test varies the payload size which is captured by software Wireshark. This capture shows the total number of bytes exchanged to carry out the message delivery, thus allowing the calculation of the ratio presented in Eq 1, also called by overhead.

$$Test_1 = \frac{\text{Size of payload}}{\text{Total bytes transmitted}} \tag{1}$$

B. Test 2: Loopback time per telegram length

This test is intended to determine the delay generated by the message transmission and server response time in finding the clients who receive the message through the subscription. For this purpose, the loopback time is calculated, by means of the client's subscription in a topic/variable, performing a self posting on it. Once the publication is done, the T1 timestamp is saved. When the message activates the callback function of the subscribed topic, the T2 timestamp is saved, calculating the loopback time by means of Eq. 2.

$$Test_2 = T2 - T1 \tag{2}$$

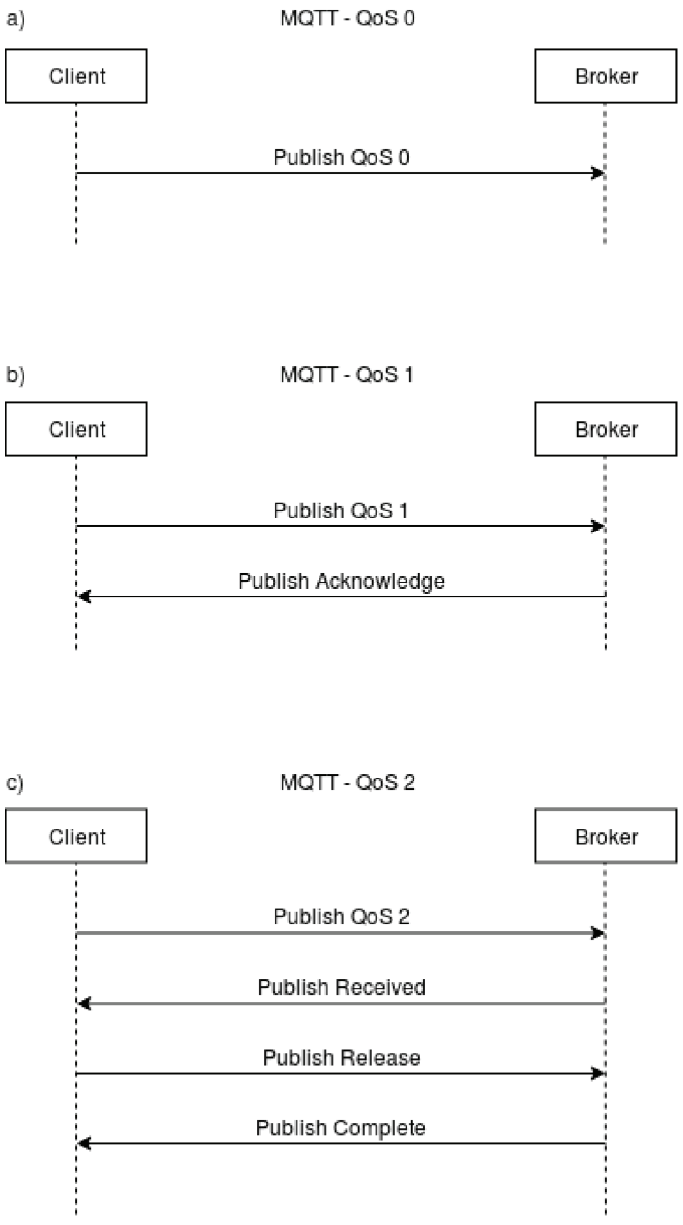

Fig. 1. MQTT publish flow for different QoS: a) QoS 0. b) QoS 1. c) QoS 2

To verify the influence of the transmission delay due to packet fragmentation and server processing time variations for sending messages to the subscribed clients, this test varies the payload length to determine the increase of the loopback time in the size of the sent message.

C. Test 3: Loopback time for different regions of the world

The delay due to packet routing over the internet in different parts of the world is checked by means of having 5 servers created in different locations, thus calculating the loopback time for a 16-byte payload. The loopback time was calculated as Test 2, explained in Section III-B.

The five locations chosen were: São Paulo (Brazil), Oregon (USA), Frankfurt (Germany), Tokyo (Japan) and a local server in the same laboratory network, located in São Carlos, Brazil.

D. *Test 4: Response time for multiple clients participating in publish/subscribe*

In order to verify the increase of the server processing time for the distribution of the message for all clients enrolled in a topic/variable, a test with the presence of N clients is proposed, where the first client (C_0) is subscribed in topic A and all other clients (C_1 to C_{N-1}) are subscribed in topic B.

At the beginning of the experiment, client C_0 publishes a 16-byte message in topic B and saves the T3 timestamp. Clients C_1 to C_{N-1} receive the message from topic B, instantly posting a 16-byte message in topic A. Client C_0 waits for receiving N-1 messages in topic A, thus saving the timestamp T4, to calculate the response time by means of the Eq. 3.

$$Test_4 = T4 - T3 \tag{3}$$

IV. Results

The developed scenario for the tests were characterized by OPC UA and MQTT clients executed on a computer with internet connection at University of São Paulo, located in the city of São Carlos, approximately 200 kilometers from São Paulo, Brazil. For the servers of both protocols, a Linux instance of Google Cloud Platform was employed, located in São Paulo, Brazil.

All the codes were created in Python 3. The OPC UA server and client were elaborated with the Python-OPCUA library. MQTT server used the HBMQTT library and MQTT client employed the PAHO-MQTT library.

A. *Results from Test 1*

Fig. 2 shows the results of the tests in four different scenarios concerning payload length: 10 bytes, 100 bytes, 1kBytes and 1MBytes. It could be seen that for messages with small payloads length, most of the transmitted bytes from the complete Ethernet frame are used for other protocols involved with the transmission (Ethernet, IP, TCP, and MQTT / OPC UA).

It is noted that using QoS 0 or 1 of MQTT has advantages over the number of bytes transmitted compared to QoS 2 and OPC UA. This could be relevant in scenarios where the connection is limited concerning the amount of data transmitted, a common situation in mobile network operators.

It is however, important to highlight data exchange scenario of OPC UA is very similar to QoS 2 of MQTT, where both confirm message delivery only once to the client. The ratio of number of bytes transmitted between the two protocols, the ratio of payload and total size of the ethernet frame is practically the same. Therefore, for scenarios where the connection does not present good stability, as in wireless connections or mobile networks with low Received Signal Strength Indicator (RSSI), the use of QoS 2 or OPC UA is highly recommended, so there is no significant difference between protocols.

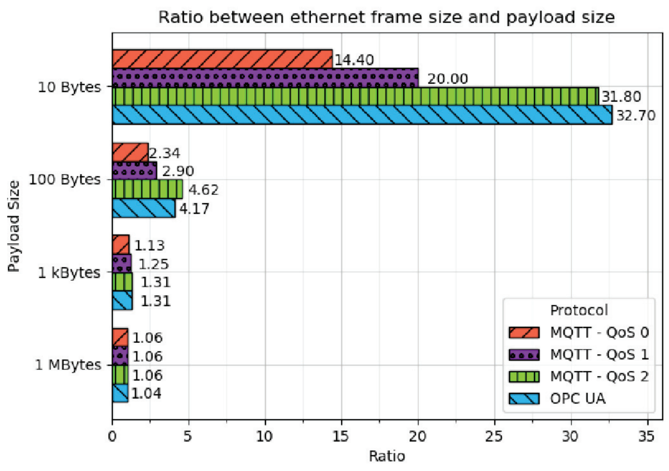

Fig. 2. Results from test 1

B. *Results from Test 2*

For the second test, the payload was changed between 100 bytes, 10KBytes and 100Kbytes, thus achieving the loopback time for statistics calculation, as shown in Tab. I. For all the tests, 10000 samples were collected, having a 5-second wait between transmissions. In order to avoid interference with the network use, the Internet is not used on the server and on the client for no other purpose.

This test turns it possible to observe that using MQTT with QoS 0 has advantages when transmitting smaller and medium packets, until 10kBytes, and QoS 1 and 2 have advantages in transmitting larger packets, having more than 10kBytes. OPC UA protocol was slower than MQTT in all situations in terms of loopback time.

It can be inferred that for situations where the updating speed of a variable is a determining factor for the process, it is suggested to use the MQTT procotolo with QoS 0, which showed a lower mean and a smaller deviation from the loopback time.

TABLE I. Measures of loopback time to São Paulo server, varying payload size

Payload Size [Bytes]	Protocol	Mean [ms]	Median [ms]	St. Dev. [ms]	Max. [ms]	Min. [ms]
100	mqtt_0	6.08	6.08	0.57	28.63	5.42
	mqtt_1	54.50	54.07	9.02	636.33	51.97
	mqtt_2	65.91	65.50	5.76	512.41	62.78
	opcua	67.22	67.07	1.06	553.90	60.31
10k	mqtt_0	8.62	8.34	1.46	47.10	7.70
	mqtt_1	14.14	14.09	0.89	52.11	12.87
	mqtt_2	25.63	25.47	3.19	331.98	23.63
	opcua	42.23	40.11	2.72	60.65	17.27
100k	mqtt_0	35.60	35.21	5.00	385.32	33.56
	mqtt_1	34.79	34.30	2.55	89.07	31.10
	mqtt_2	46.57	46.15	2.44	88.47	43.72
	opcua	58.01	55.40	5.86	85.04	37.97

C. *Results from Test 3*

The same code was used to create the OPC UA server and a MQTT broker on different locations using Google Cloud Platform virtual machine instances. The test measured the

TABLE II. MEASURES TO DIFFERENT LOCATIONS AROUND THE WORLD

Location	Protocol	Mean [ms]	Median [ms]	St. Dev. [ms]	Max. [ms]	Min. [ms]
Local Network	mqtt_0	5.33	5.31	0.46	8.19	4.87
	mqtt_1	15.82	15.77	0.50	22.05	13.47
	mqtt_2	25.44	25.51	0.53	27.51	22.91
	OPC UA	29.12	29.27	0.61	26.83	26.83
São Paulo	mqtt_0	33.80	31.69	1.82	204.50	25.89
	mqtt_1	58.93	54.12	2.57	382.85	50.52
	mqtt_2	62.80	57.78	2.82	357.82	59.50
	OPC UA	62.11	60.01	2.64	515.79	59.21
Oregon	mqtt_0	175.39	173.79	3.44	610.21	152.44
	mqtt_1	210.02	211.52	3.17	759.37	189.17
	mqtt_2	238.90	232.16	4.85	749.84	215.75
	OPC UA	251.61	241.11	4.11	812.40	222.65
Frankfurt	mqtt_0	243.27	243.12	2.88	437.77	218.94
	mqtt_1	311.94	307.06	3.30	811.85	288.09
	mqtt_2	372.61	365.61	5.53	756.41	352.76
	OPC UA	389.17	388.49	5.26	899.63	333.71
Tokyo	mqtt_0	327.79	317.27	7.79	760.25	295.48
	mqtt_1	408.04	395.88	8.29	1025.10	385.04
	mqtt_2	416.40	401.98	9.76	1135.43	398.56
	OPC UA	433.13	419.78	12.16	1235.61	401.58

loopback time to each one of these servers, as seen on Table II.

It was found that MQTT with QoS 0 presented the best result for all locations, since it doesn't need any kind of confirmation about the receiving. The MQTT with QoS 2 presented a similar result to OPC UA, showing that for this case the delay is caused by the routing of the messages across the world, increasing the time with the distance and the number of routers between the server and client.

D. Results from Test 4

Performing the fourth test proposed, setting the payload to 16 bytes and varying the number of clients connected to the topics / variables A and B, it was obtained the graph shown in Fig. 3, grouped into 7 different number of clients tested. All tests employed capture of 1000 samples.

It was found that for 10 clients, the time between posting on a topic and having the response of all clients on a second topic was practically the same for both protocols. However, it was observed that from 100 connected clients, the response time for all messages to be delivered began to increase faster in the UA OPC, achieving response time of up to 1.2 seconds for 1000 connected clients, while in MQTT the worst case was 471 milliseconds.

In this way, it is observed that the MQTT protocol is more suitable for the distribution of messages in the publish / subscribe model when there is a large number of clients connected to the same topic, not having a considerable difference between their quality of services.

V. CONCLUSION

Considering the results presented by the tests proposed in this work, it is inferred that MQTT has the advantage of using less data to transmit the same payload, in addition to having a slightly smaller transmission times. The biggest difference was in the transmission of a message to multiple clients, where the MQTT protocol presented a great advantage in sending the messages to a large number of clients.

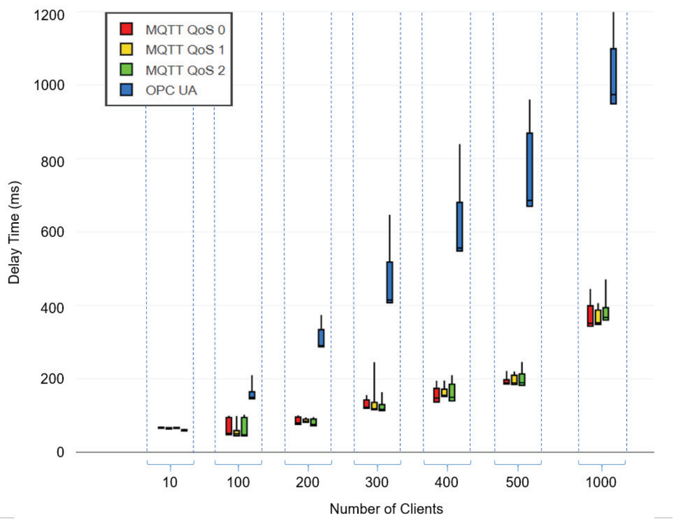

Fig. 3. Results from test 4

However, one can not depreciate the magnitude of the OPC UA protocol, which has several other services besides the data exchange, which in this paper was still limited only to the publish/subscribe model. Services such as data modeling, address space, alarm and event management, variable history, access control, among many others are offered.

The application of MQTT implies in the use of tools or the development of methods and conventions to deal with the data types sent between devices, the ensure sequencing of messages, the creation of historical data and other services that may be necessary, thus allowing the usage of just what is necessary to obtain the advantages of data exchange.

REFERENCES

[1] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[2] Y. Cao, T. Jiang, and Z. Han, "A survey of emerging m2m systems: Context, task, and objective," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1246–1258, Dec 2016.

[3] G. S. Sestito, A. C. Turcato, A. L. Dias, M. S. Rocha, M. M. M. da Silva, P. Ferrari, and D. Brandao, "A method for anomalies detection in real time ethernet data traffic applied to profinet," *IEEE Transactions on Industrial Informatics*, 2017, article in Press. [Online]. Available: www.scopus.com

[4] F. A. Fernandes, G. S. Sestito, A. L. Dias, D. Brandão, and P. Ferrari, "Influence of network parameters on the recovery time of a ring topology profinet network," *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 278–283, 2016, cited By :1. [Online]. Available: www.scopus.com

[5] S. Rinaldi, P. Ferrari, D. Brandao, and S. Sulis, "Software defined networking applied to the heterogeneous infrastructure of smart grid," in *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS*, vol. 2015-July, 2015, cited By :9. [Online]. Available: www.scopus.com

[6] P. Ferrari, E. Sisinni, D. Brandão, and M. Rocha, "Evaluation of communication latency in industrial iot applications," in *2017 IEEE International Workshop on Measurement and Networking, M and N 2017 - Proceedings*, 2017. [Online]. Available: www.scopus.com

[7] L. Durkop, B. Czybik, and J. Jasperneite, "Performance evaluation of m2m protocols over cellular networks in a lab environment," in *2015 18th International Conference on Intelligence in Next Generation Networks*, Feb 2015, pp. 70–75.

[8] T. Mizuya, M. Okuda, and T. Nagao, “A case study of data acquisition from field devices using opc ua and mqtt,” in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Sept 2017, pp. 611–614.

[9] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[10] Y. t. Lee, W. h. Hsiao, C. m. Huang, and S. c. T. Chou, “An integrated cloud-based smart home management system with community hierarchy,” *IEEE Transactions on Consumer Electronics*, vol. 62, no. 1, pp. 1–9, February 2016.

[11] H. Derhamy, J. Rönnholm, J. Delsing, J. Eliasson, and J. van Deventer, “Protocol interoperability of opc ua in service oriented architectures,” in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, July 2017, pp. 44–50.

[12] OASIS Standard, *MQTT Version 3.1.1.*, 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html