ACTA IMEKO
ISSN: 2221-870X
June 2019, Volume 8, Number 2, 80 - 87

On the performance of OPC UA and MQTT for data exchange between industrial plants and cloud servers

Murilo Silveira Rocha[1], Guilherme Serpa Sestito[1], Andre Luis Dias[1], Afonso Celso Turcato[1], Dennis Brandão[1], Paolo Ferrari[2]

[1] *Department of Electrical Engineering, University of São Paulo, São Carlos, Brazil*
[2] *Department of Information Engineering, University of Brescia, Brescia, Italy*

ABSTRACT
The Internet of Things (IoT) is a key technology in the development of Industry 4.0. An increasing number of new industrial devices are expected to communicate with each other by means of local (edge) and cloud computing servers. In this article, two well-known protocols used for IoT and Industrial IoT (IIoT) are compared in terms of their performance when they are used to send/receive data to/from cloud servers. Due to their wide diffusion and suitability, the considered protocols are open platform communication-unified architecture publisher-subscriber (OPC UA PubSub) (purposely developed and maintained by industrial consortia) and message queuing telemetry transport (MQTT), the most well-known message protocol originally developed by IBM. The performance comparison is carried out considering the overall quantity of the data transferred (user payload plus overhead) and the roundtrip time required to send in data and receive a feedback message in return. The experimental results include the evaluation of several cloud computing server and application scenarios, highlighting how each protocol is particularly suitable for certain situations. Finally, conclusions about the best choice for data exchange between devices are given.

Section: RESEARCH PAPER

Keywords: open platform communication-unified architecture; message queuing telemetry transport; machine-to-machine (M2M); IoT; Industry 4.0; protocol comparison; data exchange

Citation: Murilo Silveira Rocha, Guilherme Serpa Sestito, Andre Luis Dias, Afonso Celso Turcato, Dennis Brandão, Paolo Ferrari, On the performance of OPC UA and MQTT for data exchange between industrial plants and cloud servers, Acta IMEKO, vol. 8, no. 2, article 11, June 2019, identifier: IMEKO-ACTA-08 (2019)-02-11

Section Editor: Emiliano Sisinni, University of Brescia, Italy

Received July 30, 2018; **In final form** June 17, 2019; **Published** June 2019

Funding: This work was supported by the Laboratório de Automação Industrial, LAI, from the School of Engineering of São Carlos.

Corresponding author: Murilo S. Rocha; murilo.silveira.rocha@usp.br, Paolo Ferrari; paolo.ferrari@unibs.it

1. INTRODUCTION

The process of digital transformation known as Industry 4.0 is ongoing. Its main objective is the implementation of a more advanced, flexible, and efficient manufacture [1]-[3]. In recent years, the use of sensors, actuators, controllers and supervisory systems has become common in industrial automation systems, with a typical communication infrastructure organised like a pyramid, where these layers can exchange data vertically from layer to layer. Nowadays, on the contrary, the information flows as in a mesh communication structure, where the industrial automation devices are able to communicate with each other without a strict hierarchy. This architecture is the basis for a wider collection of information, with feature extraction, and enhanced knowledge about the process by means of intelligent systems and big-data techniques [4], [5].

The huge quantity of data to be exchanged between the industrial devices deployed in the field and the cloud servers imposes the use of new machine-to-machine (M2M) protocols, oriented to be more efficient and reliable. Moreover, determinism and low complexity are two important characteristics required by embedded systems for industrial applications [6].

It is common, for the most diffused industrial automation protocols and networks to have some performance indicators that show their suitability for the demanding communication task in the industrial environment. Robustness and determinism have been studied in general terms by several researchers. A specific focus on the estimation of communication delays can be commonly found [7]-[9]. The new M2M protocols that are now

starting to be used in industrial systems for Internet connection with cloud servers should be evaluated with similar metrics, related to time, in order to estimate communication performance. A prime example of this approach is given in [10], [11].

On this subject, the current article expands the work in [12] about open platform communication-unified architecture (OPC UA) and message queuing telemetry transport (MQTT) (two widely used protocols in Industry 4.0 and IoT applications [13]) with the aim of comparing them in the publish/subscribe mode of operation. The time characteristics of each protocol are discussed, and the distribution of the round-trip delay for sending and receiving messages is analysed in different scenarios. Note that for OPC UA, only the transport part of the specification is taken into account in this article.

The article is organised in five sections. Section 2 briefly introduces the two protocols, showing basic operation concepts for the sake of understanding the proposed tests. Section 3 describes the proposed tests and their purpose. Section 4 reports the results. Finally, Section 5 shows the overall results of the comparison between OPC UA and MQTT.

2. BRIEF INTRODUCTION TO MQTT AND OPC UA

This section describes the two protocols used in the proposed tests (MQTT and OPC UA), together with a brief introduction to the publish/subscribe model, which is used for data exchange in both protocols.

2.1. The publish/subscribe general model

The publish/subscribe model has been introduced to facilitate communication between two or more devices. Data exchange is organised according to 'topics'. The publish/subscribe model is one of the most popular paradigms in the Industry 4.0 environment [14]-[17], which is the reference scenario in this article. In the publisher-subscriber model, there are two kinds of stations:

1) the subscribers, which are interested in receiving the information, may subscribe to the topic of interest and then wait for new messages to come; and
2) the publishers, which wish to send information, may publish a message identified by a topic [18].

The publishing and the message distribution to subscribers may be done through a specific server, also called a broker [19], or by using some services offered by the transport layer (e.g. multicast communication).

2.2. Overview of MQTT

IBM designed a message protocol called MQTT for applications in the consumer market. Currently, it is widely applied, for instance, to office and home automation and healthcare applications. MQTT is also attractive for low-power and low-latency applications, especially those based on wireless devices (e.g. smartphones) [20]. The protocol is designed for 'transporting' data; hence, no differentiation/organisation of data type is provided in the protocol specification. The coding/meaning/modelling of data is demanded by the participants in the MQTT data exchange. The MQTT architecture is completely built on the publish/subscribe model. For instance, a client cannot freely read variables, but it must wait until the information (topic) is published by the system [7].

In the MQTT specification, the device's participation in the data exchange can either be as a server or a client. The server is the message broker who manages and delivers the messages. It is the centre of the architecture, and all clients are related to it. The MQTT server is also called the Broker. On the other hand, a client can act as a publisher (sender) and/or subscriber (destination) of messages.

MQTT defines the quality of service (QoS) modes. The sender and the receiver of the messages can reach an agreement about the certainty of the delivery. Three levels of QoS are defined. QoS 0 does not guarantee that the message reaches the destination, as shown in Figure 1(a). In this case, the reliability of the data exchange depends only on the underlying transport protocol (which is usually TCP). QoS 1 guarantees that the sent message arrives at the destination at least once. Since the transaction ends only when receiver responds with the confirmation message back to the sender to acknowledge receipt of the message (as shown in Figure 1(b)), it may happen that the message is sent more times to the receiver.

Furthermore, QoS 2 guarantees that the message is delivered only once to the receiver. This is also the most complex transaction, since four messages are exchanged, as shown in Figure 1(c). The message contains the data for the receiver; the acknowledgement message that informs the sender about receipt; the request for the sender to release the message; and finally, the confirmation from the receiver that the sender can release the message and close the transaction [8].

The MQTT overhead is expected to be minimal because since its beginning, IBM's goal was to minimise it. The MQTT published message has a maximum of 9 bytes of overhead plus the 'topic' string. The acknowledgement/confirmation messages used for the QoS implementation have a maximum size of 2 bytes. The MQTT is transported over TCP/IP, so another (20+20) bytes for the headers is required. The layer 2 overhead is 18 bytes for Ethernet connections. In sum, a minimum overhead of about 60 to 80 bytes per message is highly probable.

2.3. OPC UA in brief

The 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 modelling as well as organisation of the address space of these objects inside the server [21]. It allows the creation of variables of certain data types within each object on the server, allowing clients to read the variable values or to subscribe to the variables for which it would like to receive all updated values [22]. The transport layer of OPC UA is based on TCP, and it allows for the binary encoding of data or the use of HTTP with SOAP (Simple Object Access Protocol) (binary and XML).

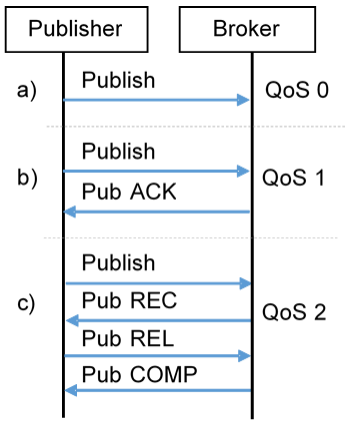

Figure 1. MQTT publish flow for different QoS (a) QoS 0; (b) QoS 1; (c) QoS 2.

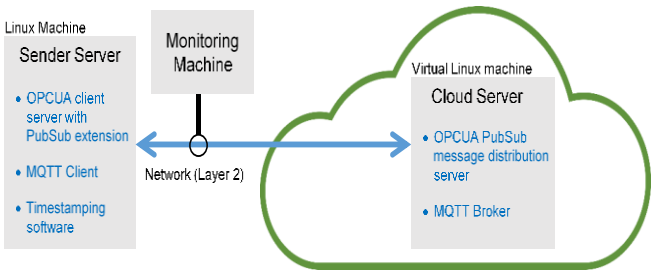

Figure 2. A proposed experimental setup for interaction with cloud servers.

The OPC UA was designed to use a client-server architecture, but recently (2017), the OPC Foundation released the OPC UA PubSub specifications that enable full support for the publisher/subscriber model. The specification defines two modes of operations depending on the way in which the publisher/subscriber mechanism is implemented: the 'Broker-based model' or 'Broker-less model'. The former is based on message protocols (like the advanced message queuing protocol [AMQP] or even MQTT) while the latter uses UDP (User Datagram Protocol) multicast. Binary or JSON (JavaScript Object Notation) encoding of data can be used.

In this paper, only the OPC UA PubSub functionalities implemented using the 'Broker-based model' have been considered in order to be directly comparable with the MQTT.

However, despite that OPC UA offers many other functionalities, the main objective of this article is data exchange in the publish/subscribe model. Therefore, it will not consider all the other features provided by the protocol. This choice, in turn, also limits the scope of the considered overhead to what is visible in the messages collected on the network at layer 2 (i.e. a black-box approach).

In literature, [12] investigated the topic of an OPC UA client server overhead, concluding that the encapsulation of data due to the structured information model and the security coding can cause an additional overhead (up to 15 times greater under some conditions) with respect to straight TCP binary encoding. For this reason, the overhead of OPC UA PubSub is expected to be higher than MQTT, since the same OPC UA data structure is maintained over the message protocol used by OPC UA PubSub.

3. THE PROPOSED MEASUREMENT METHODOLOGY

In order to achieve this study's goal, the most recent investigation about delay estimation for data exchange across the cloud has been taken into account. Since IoT and Industry 4.0 are hot research topics, there are several important concepts that must be considered: the estimation of Internet delay is investigated in-depth in [23]-[26], and in particular, MQTT seems to be one of the most studied protocols, as shown in [27]-[30]. This section describes the measurement methodology used to compare the protocols by taking into account the previous achievements described in [31]-[34]. All the designed tests have been created to use only the publish/subscribe model to exchange data using the same procedure. Regarding MQTT, all three available QoS levels are used.

3.1. Experimental setup architecture for the tests

The experimental setup general architecture is shown in Figure 2. It is composed by a sender-server machine connected to the Internet. In the cloud, the virtual server implements the MQTT broker and the OPC UA message distribution server. The network connection is monitored (at layer 2) by means of a network tap. All the traffic to and from the machine is recorded by a monitoring machine. The round-trip delays are measured using the local clock of the sender machine. Since the duration of the measured delay is very small (a few hundreds of milliseconds) the local oscillators can be considered stable for the duration of the measurements (i.e. it is supposed that the short-term stability of crystal oscillator is better than 10^{-5}).

3.2. Test 1: Protocol overhead

This test analyses all the data transmitted by the source machine in order to estimate the number of bytes to be transferred for transmitting a given useful payload. The monitoring station captures all the network frames and calculates the total number of bytes exchanged to carry out the message delivery for the transaction under test. Then, the metric related to the protocol overhead can be computed as

$$M_1 = \frac{Total\ Bytes\ Transmitted}{Size\ of\ Payload}. \quad (1)$$

The test is carried out several times in order to obtain meaningful statistics, and the payload size can be varied in order to achieve a deeper understanding of the protocols.

3.3. Test 2: Round-trip delay per telegram length

This test aims to determine the time required by a message to reach a server/broker and then to be sent to a subscriber. Hence, the overall delay is composed by two contributions that may make it difficult to measure. For this reason, the proposed metric is the round-trip time calculated by a client that publishes a topic/variable to whom the client itself is also a subscriber. In this way, all the necessary time references are taken by the same client. The procedure requires that once the publication is done, the T1 timestamp is saved in the source machine. Then, when the message that comes back from the server/broker activates the callback function of the subscribed topic, the T2 timestamp is saved in the destination machine, which is also the source machine. Finally, the round-trip time metrics are calculated by means of equation (2):

$$M_2 = T2 - T1. \quad (2)$$

3.4. Test 3: Round-trip delay for different regions of the world

The relationship between the location of the server and the round-trip delay is studied in Test 3. The delay is expected to be dependent on the packet routing over the Internet. This test involves five servers created in different locations (i.e. in different parts of the world) with the same architecture based on Google Cloud Platform. In Test 3, the round-trip time has been calculated with a message with a payload of 16 bytes. The metric for this test is the same for Test 2, explained in Section 3.3.

The servers have been created in the following five locations: São Paulo (Brazil), Oregon (USA), Frankfurt (Germany), Tokyo (Japan), and finally, a local server in the same laboratory network, São Carlos (Brazil) for comparison.

3.5. Test 4: Round-trip delay for multiple clients participating in the publish/subscribe protocol

Test 4 is designed to highlight the effect of an increasing number of subscribers. In order to verify if the server processing time for the distribution of the message to all clients (subscribed to the same topic/variable) varies when the number of clients increases, a test with N clients has been designed. Given N clients, the first client (C_0) is a subscriber of topic A, and all other clients (C_1 to C_{N-1}) are subscribers to topic B.

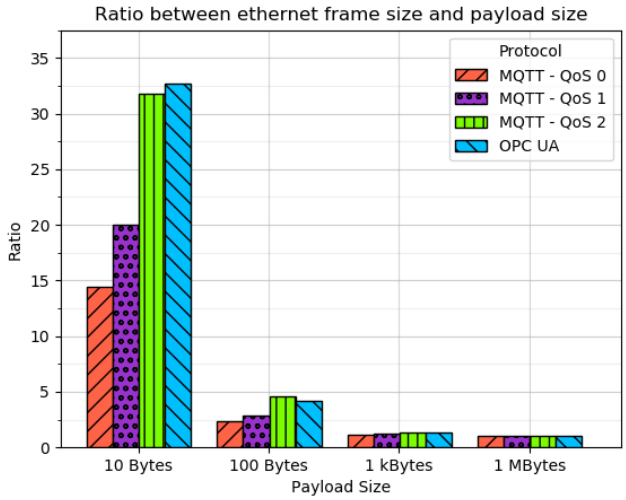

Figure 3. Overhead of the considered protocols as a function of the useful payload size.

In order to initiate one run of the experiment, client C_0 publishes a 16-byte message on topic B and at the same time saves the $T3$ timestamp. When clients C_1 to C_{N-1} receive the message from topic B, they instantly publish a message in topic A. Client C_0 waits for receiving all $N-1$ messages on topic A; then, it saves the timestamp $T4$. The total round-trip delay metrics can be calculated as:

$$M_4 = T4 - T3. \quad (3)$$

4. EXPERIMENTAL RESULTS

The experimental setup described in Section 3.1 and shown in Figure 2 is customised differently depending on the protocol that is being tested.

The different scenarios use OPC UA and MQTT clients executed on a Linux computer with a high-speed Internet connection located at the University of São Paulo in the city of São Carlos (approximately 200 km from São Paulo, Brazil). On the contrary, the servers of both protocols are implemented by means of a virtual Linux machine inside the Google Cloud Platform. Most of the experiments are on the Google Cloud Server located in São Paulo, Brazil. However, for some tests, other locations of the Google Cloud Server have been used (see the section on Test 3 for further details).

For the software implementation, Python 3 has been used as a programming language. This article has used open-source libraries. In particular, the OPC UA server and client (required to create the publisher-subscriber architecture) are based on the Python-OPC UA library with PubSub extensions. The MQTT server (Broker) uses the HBMQTT library, while the MQTT client is built with the PAHO-MQTT library.

The monitoring machine is a Linux machine running the network capture software Wireshark. The network uses Ethernet layer 2.

4.1. Test 1: Experimental results

Test 1 requires assessing the overhead of the protocols varying the payload length. For the implementation, four different payloads (10 bytes, 100 bytes, 1 kB and 1 MB) have been considered. A graphical comparison of the results is shown in Figure 3. The main point is that in the case of messages with small payloads, the protocol overhead is dominant. Most of the transmitted bytes on the network (Ethernet in the experiment cases) are used by other supporting protocols (e.g. Ethernet, IP, and TCP headers) or by headers/trailers and acknowledgement/confirmation messages of MQTT and OPC UA.

MQTT QoS 0 and 1 have clear advantages over the quantity of the transmitted bytes compared to MQTT QoS 2 and OPC UA. This is highly relevant in scenarios in which the main limit of the connection is the amount of data transmitted; for instance, when stations are using mobile network operators that charge depending on the total exchanged bytes.

However, there are situations in which the behaviour of OPC UA is very similar to that of the MQTT. It is in the case of MQTT with QoS 2 in which the MQTT guarantees the message delivery only once to the subscriber. Since this is the normal behaviour of OPC UA, it is reasonable to expect the same value for the considered metric. As matter of fact, the ratio of the payload and the total size of the Ethernet frame is practically the same in the two protocols.

Consequently, if the application scenario has poor quality connection between stations, and messages can be lost, the use of MQTT QoS 2 or OPC UA is strongly recommended, and both methods are equivalent from the point of view of their

Table 1. Results of Test 1 with the cloud servers located in São Paulo. All values are in ms.

Payload size	Protocol	Mean	Median	St. Dev	Max	Min
100 bytes	MQTT QoS 0	6.08	6.08	0.57	28.63	5.42
	MQTT QoS 1	54.50	54.07	9.02	636.33	51.97
	MQTT QoS 2	65.91	65.50	5.76	512.41	62.78
	OPC UA	67.22	67.07	1.06	553.90	60.31
10 kB	MQTT QoS 0	8.62	8.34	1.46	47.10	7.70
	MQTT QoS 1	14.14	14.09	0.89	52.11	12.87
	MQTT QoS 2	25.63	25.47	3.19	331.98	23.63
	OPC UA	42.23	40.11	2.72	60.65	17.27
100 kB	MQTT QoS 0	35.60	35.21	5.00	385.32	33.56
	MQTT QoS 1	34.79	34.30	2.55	89.07	31.10
	MQTT QoS 2	46.57	46.15	2.44	88.47	43.72
	OPC UA	58.01	55.40	5.86	85.04	37.97

measured performance. A typical situation in which these conditions are met are the wireless connections (or even mobile networks) with low received signal strength indicator (RSSI) and unpredictable interference.

4.2. Test 2: Experimental results

Test 2 aims to calculate the round-trip time required by the protocols. In this test, the round-trip time for a self-published topic is calculated by changing the payload size between 100 bytes, 10 kB, and 100 kB. The round-trip times for the calculation of the statistics are shown in Table 1, while the probability density function estimates are reported in Figure 4, Figure 5, and Figure 6. Each test considers 10000 samples, which means 10000 published messages with a 5 s wait interval between transmissions.

The results show that the MQTT with QoS 0 has advantages when transmitting smaller and medium packets of up to 10 kB. With larger packets, MQTT QoS 0 and QoS 1 perform the same, while MQTT QoS 2 are beneficial for transmitting larger packets (more than 10 kB). The OPC UA protocol is slower than the MQTT in all situations in terms of the round-trip time, and its distribution has a long tail and several peaks, demonstrating the presence of several sampling intervals and timeouts [35] inside the OPC UA stacks. It should also be noted that the results for OPC UA PubSub are in the same order of magnitude of the results obtained in [36] and [37] for the OPC UA client-server architecture, demonstrating the inherent complexity of OPC UA protocol stacks.

In conclusion, for situations in which the speed in updating variables is a decisive factor in the process, and the size of this variable is small, the use of the MQTT protocol with QoS 0 is suggested. It has round-trip time statistics with a lower mean, a smaller standard deviation, and (most importantly) a distribution with a single and narrow peak.

4.3. Test 3: Experimental results

Test 3 evaluates the influence of the cloud server location on communication performance. As in Test 2, the same code was used to create the OPC UA architecture and the MQTT broker on different locations by means of virtual machine instances hosted by Google Cloud Platform. The test measured the round-trip time for each one of these cloud servers, as shown in Table 2, while the probability density function estimates are reported in Figure 7, Figure 8, Figure 9, Figure 10, and Figure 11.

Comparing the different servers, it is clear that for this test, the delay is caused by the routing of the messages across the world. The round-trip time increases with the distance and the number of routers between the server and client.

Multiple peaks appear in the distribution of all the protocols because the cyclical behaviour of Internet routers along the path is now relevant. The presence of multimodal distributions is an impairment in the use of such architectures in high-speed industrial control loops.

It is clear that an MQTT with QoS 0 again has the best performance for all the locations. The reason is that it does not need any kind of confirmation of receipt. The MQTT with QoS 2 presented a similar result to OPC UA, but the latter has the longest tails when the distributions are compared.

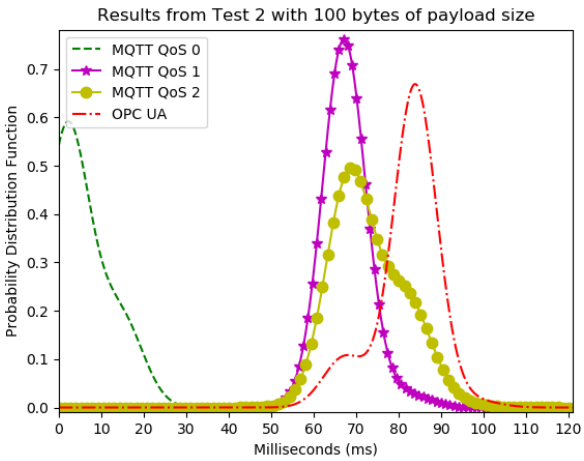

Figure 4. Distribution estimate for the round-trip time of a single variable with a size of 100 bytes.

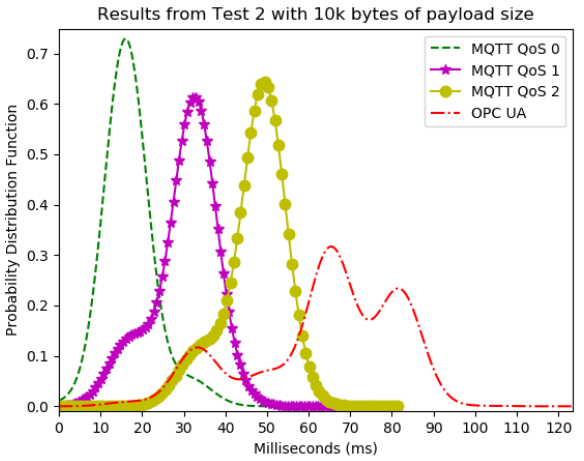

Figure 5. Distribution estimate for the round-trip time of a single variable with a size of 10 kB.

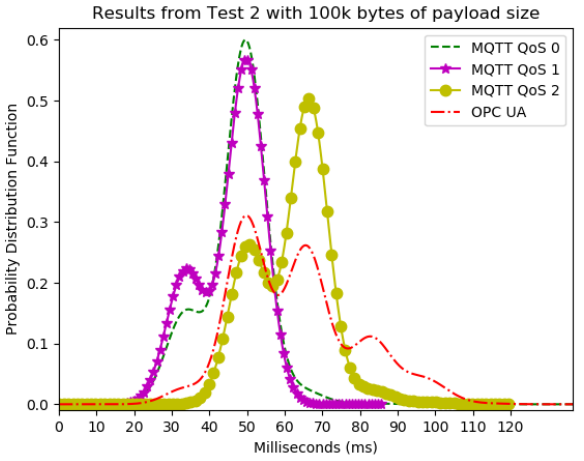

Figure 6. Distribution estimate for the round-trip time of a single variable with a size of 100 kB.

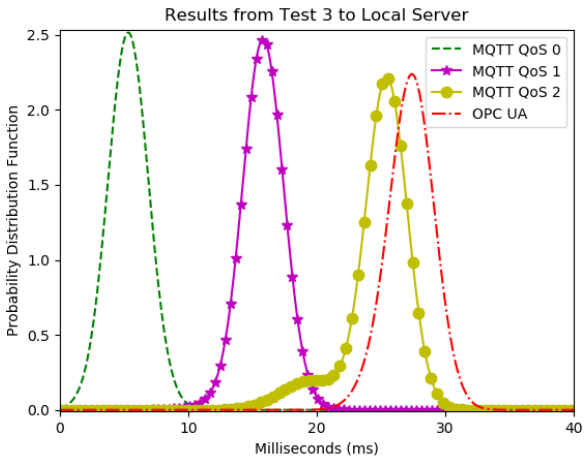

Figure 7. Distribution estimate for the round-trip time of a server located in the same local area network.

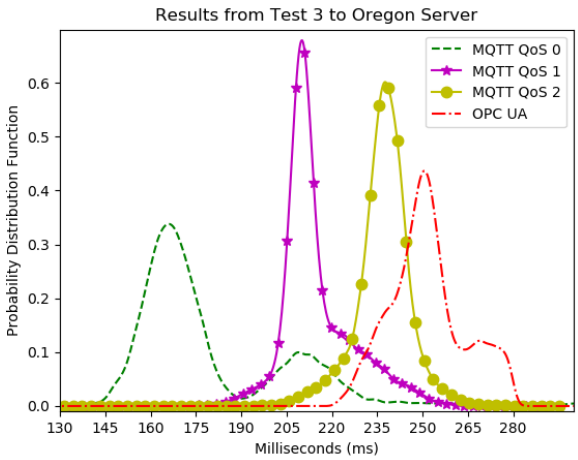

Figure 9. Distribution estimate for the round-trip time of a server located in Oregon (USA).

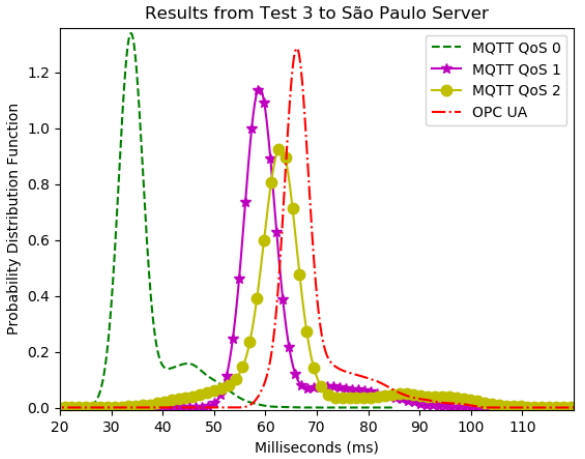

Figure 8. Distribution estimate for the round-trip time of a server located in São Paulo (BRA).

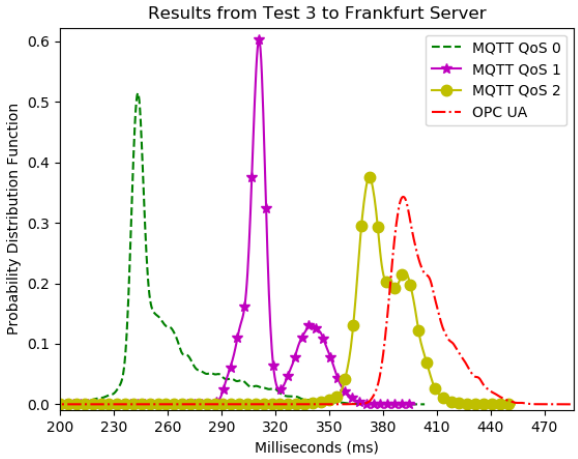

Figure 10. Distribution estimate for the round-trip time of a server located in Frankfurt (GER).

Table 2. Results of Test 3 with cloud servers around the world. All the results are in ms.

Location of the Cloud Server	Protocol	Mean	Median	St. Dev	Max	Min
Local Area Network	MQTT QoS 0	5.33	5.31	0.46	8.19	4.87
	MQTT QoS 1	15.82	15.77	0.50	22.05	13.47
	MQTT QoS 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 QoS 0	33.80	31.69	1.82	204.50	25.89
	MQTT QoS 1	58.93	54.12	2.57	382.85	50.52
	MQTT QoS 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 QoS 0	175.39	173.79	3.44	610.21	152.44
	MQTT QoS 1	210.02	211.52	3.17	759.37	189.17
	MQTT QoS 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 QoS 0	243.27	243.12	2.88	437.77	218.94
	MQTT QoS 1	311.94	307.06	3.30	811.85	288.09
	MQTT QoS 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 QoS 0	327.79	317.27	7.79	760.25	295.48
	MQTT QoS 1	408.04	395.88	8.29	1025.10	385.04
	MQTT QoS 2	416.40	401.98	9.76	1135.43	398.56
	OPC UA	433.13	419.78	12.16	1235.61	401.58

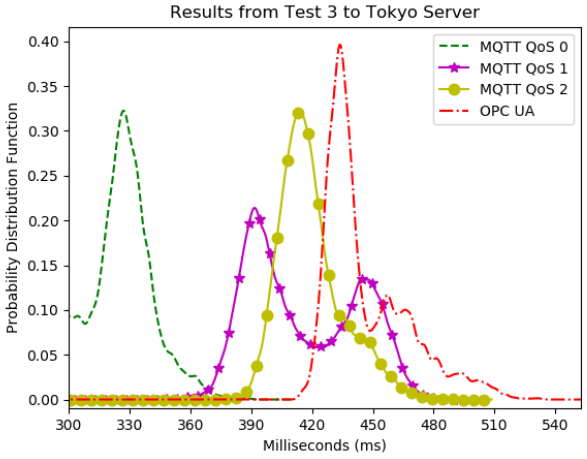

Figure 11. Distribution estimate for the round-trip time of a server located in Tokyo (JAP).

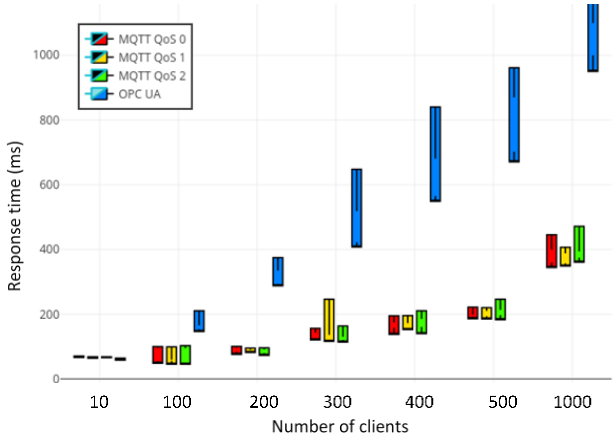

Figure 12. Comparison of performance with the multi-client test. OPC UA performance quickly worsens when the number of clients increases.

4.4. Test 4: Experimental results

Test 4 has been designed to verify the impact of the number of clients on the publisher/subscriber architecture of the two protocols. In order to perform the test, the message payload has been set to 16 bytes, and the number of clients connected to the topics/variables *A* and *B* has been varied. The results are shown in Figure 12 in the form of a box-plot graph. There are seven groups depending on the number of clients used in the tests. Each test uses 1000 samples, which means 1000 round-trip transactions as explained in Section 3.5.

It is interesting to note that for ten clients, the time between publishing on a topic and receiving the response of all clients on a second topic is practically the same for the MQTT (any QoS) and OPC UA. However, the behaviour changes as soon as the number of clients (subscribers) increases. Starting from 100 subscribed clients, the considered metric begins to increase faster with the OPC UA. The maximum value for OPC UA is a response time of up to 1.2 seconds for 1000 subscribed clients, while for the MQTT, the worst case was less than 500 ms.

In can be concluded that the current implementations of MQTT the protocol are more efficient for the distribution of messages in the publish/subscribe model when there is a large number of clients subscribed to the same topic. Moreover, there are no considerable differences between the available QoSs. On the other hand, the available OPC UA implementations of the PubSub architecture suffer when the number of subscribers increases, showing once again the higher complexity of OPC UA protocol stacks.

5. FINAL REMARKS AND RECOMMENDATIONS

The results of the proposed tests highlight that the MQTT has the advantage of using less data to transmit the same payload and slightly lower transmission times compared to OPC UA.

The most interesting result is the large difference between the two protocols when the transmission of same message to multiple clients is considered: The MQTT protocol has a great advantage (at least three times faster) in sending the messages to a large number of clients. When the topic/variable is small, the MQTT is better. When a small delay is important for an application, the MQTT has more advantages than OPC UA.

However, this paper has evaluated only the publish/subscribe model. The view of the OPC UA protocol is therefore limited only to the data exchange part. The OPC UA protocol stack is complex, and it has several other services besides data exchange (such as data modelling; address space; alarm and event management; variable history; and access control), which will be evaluated in future works.

On the contrary, the MQTT is basically unstructured, and it implies the use of additional tools for the development of methods, for the definition of data types sent between devices, for the sequencing of messages, and for the creation of historical data services.

Even if it is clear that the comparison of complete solutions based on the MQTT or OPC UA must take into account all the components and tools used, the results related to the data exchange presented in this article are fundamental, since they may constitute a reference for the best performance that is obtainable.

6. CONCLUSIONS

Today, IoT is the most important technology for the implementation of Industry 4.0. New industrial devices communicate by means of local (edge) and cloud computing servers. In this article, OPC UA and the MQTT (two well-known protocols used for IoT and industrial IoT) are compared in terms of performance when they are used to send/receive data to/from cloud servers. The performance comparison is carried out considering the overall quantity of data transferred (user payload plus overhead) and the round-trip time required to send in data and receive a feedback message in return. The measurement methodology was fully described, and the experimental results, including the evaluation of several cloud computing server and application scenarios, were reported. Considering specific use case studies, the MQTT protocol was found to be faster than OPC UA for pure data exchange. OPC UA pays the complexity of its stack, which is also designed for ancillary services. Generally speaking, the data exchange between industrial plants and cloud servers may take from less than 100 ms to more than 1 s depending on the Internet path length, with lower values obtained by the MQTT QoS 0 and the small size of the exchanged variables.

7. REFERENCES

[1] L.D.Xu, W.He, S.Li, Internet of things in industries: a survey, IEEE Transactions on Industrial Informatics, 10(4) (2014) pp. 2233-2243.

[2] F.Tao, Y.Zuo, L.D.Xu, L.Zhang, IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing, IEEE Transactions on Industrial Informatics, 10(2) (2014) pp. 1547-1557.

[3] Y.Liu, X.Xu, 'Industry 4.0 and cloud manufacturing: A comparative analysis', J. Manuf. Sci. Eng., 139(3), (2016).

[4] Y.T.Chou, An integrated cloud-based smart home management system with community hierarchy. IEEE Transactions on Consumer Electronics, 62(1), (2016), pp. 1-9.

[5] S.Corbellini, E.Di Francia, S.Grassini, L.Iannucci, L.Lombardo, M.Parvis, Cloud based sensor network for environmental monitoring, Measurement: Journal of the International Measurement Confederation (2017) (in press).

[6] H.D.Deventer, 'Protocol interoperability of OPC UA in service oriented architectures', Proc. of the 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), pp. 44-50.

[7] F.A.Fernandes, G.Serpa Sestito, A.Luis Dias, D.Brandão. P. Ferrari, Influence of network parameters on the recovery time of a ring topology PROFINET network, IFAC-PapersOnLine (2016) pp. 278-283.

[8] Y.C.Han A survey of emerging M2M systems: context, task, and objective. IEEE Internet of Things Journal, 3(6), (2016), pp. 1246-1258.

[9] L.Dürkop, B.Czybik, J.Jasperneite, 'Performance evaluation of M2M protocols over cellular networks in a lab environment', Proc. of the 2015 18th International Conference on Intelligence in Next Generation Networks, 2015, pp. 70-75.

[10] P.Ferrari, E.Sisinni, D.Brandao, M.Rocha, 'Evaluation of communication latency in industrial IoT applications', Proc. of the IEEE International Workshop on Measurements and Networking (M&N), 27-29 Sept., 2017, Naples, Italy, pp. 17-22.

[11] P.Ferrari, A.Flammini, E.Sisinni, S.Rinaldi, D.Brandão, M.S.Rocha, Delay estimation of industrial IoT applications based on messaging protocols, IEEE Trans. on Instrumentation and Measurement, 67(8), (2018), pp. 2188 – 2199.

[12] M.Silveira Rocha, G.Serpa Sestito, A.Luis Dias, A.Celso Turcato, D.Brandão, 'Performance comparison between OPC UA and MQTT for data exchange', Proc. of the Workshop on Metrology for Industry 4.0 and IoT, 2018, Brescia, Italy, pp. 175-179.

[13] B.C.Yin, Smart factory of Industry 4.0: key technologies, application case, and challenges, IEEE Access (2017) p. 99.

[14] J.Wan et al., Software-defined industrial Internet of Things in the context of Industry 4.0, IEEE Sensors Journal, 16(20) (2016) pp. 7373-7380.

[15] T.H.Szymanski, Supporting consumer services in a deterministic industrial Internet core network, IEEE Communications Magazine, 54(6) (2016) pp. 110-117.

[16] H.Derhamy, J.Eliasson, J.Delsing, IoT interoperability—on-demand and low latency transparent multiprotocol translator, IEEE Internet of Things Journal, 4(5) (2017) pp. 1754-1763.

[17] C.Yang, W.Shen, X.Wang, 'Applications of Internet of Things in manufacturing', Proc. of the IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2016, Nanchang, pp. 670-675.

[18] P.Bellagente, P.Ferrari, A.Flammini, S.Rinaldi, E.Sisinni, 'Enabling PROFINET devices to work in IoT: Characterization and requirements', Proc. of the IEEE Instrumentation and Measurement Technology Conference, I2MTC 2016, 2016.

[19] Industrial Internet reference architecture [Online] Available: http://www.iiconsortium.org/IIRA.htm [Access date: 11/2017].

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

[21] W.a-H.Mahnke, OPC Unified Architecture, Springer Publishing Company, Inc., 2009.

[22] T.Mizuya, M.Okuda, T.Nagao, 'A case study of data acquisition from field devices using OPC UA and MQTT', Proc. of the 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, 2017, pp. 611-614.

[23] C.J.Bovy, H.T.Mertodimedjo, G.Hooghiemstra, H.Uijterwaal, 'Analysis of end-to-end delay measurements in Internet', Proc. of the ACM Conference Passive and Active Measurements, 2002.

[24] A.Pathak, H.Pucha, Y.Zhang, Y.C.Hu, Z.M.Mao, 'A measurement study of Internet delay asymmetry', Proc. of the International Conference on Passive and Active Network Measurement PAM, 2008, pp. 182-191.

[25] P.Mahadevan, D.Krioukov, M.Fomenkov, B.Huffaker, X.Dimitropoulos, K.Claffy, A.Vahdat, The Internet AS-level topology: three data sources and one definitive metric, ACM SIGCOMM Computer Communication Review (CCR), 36(1) (2006) pp. 17-26.

[26] M.Collina, M.Bartolucci, A.Vanelli-Coralli, G.E.Corazza, 'Internet of Things application layer protocol analysis over error and delay prone links', Proc. of the 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2014, Livorno, pp. 398-404.

[27] N.Naik, 'Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP, and HTTP', Proc. of the 2017 IEEE International Systems Engineering Symposium (ISSE), 2017, Vienna, pp. 1-7.

[28] S.Lee, H.Kim, D.K.Hong, H.Ju, 'Correlation analysis of MQTT loss and delay according to QoS level', Proc. of the International Conference on Information Networking (ICOIN), 2013, Bangkok, pp. 714-717.

[29] Y.Xu, V.Mahendran, W.Guo, S.Radhakrishnan, Fairness in fog networks: Achieving fair throughput performance in MQTT-based IoTs, Proc. of the 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2017, Las Vegas, Nevada, pp. 191-196.

[30] K.Govindan, A.P.Azad, 'End-to-end service assurance in IoT MQTT-SN', Proc. of the 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), 2015, Las Vegas, Nevada, pp. 290-296.

[31] S.Mijovic, E.Shehu, C.Buratti, 'Comparing application layer protocols for the Internet of Things via experimentation', Proc. of the 2nd IEEE International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), 2016, Bologna, Italy, pp. 1-5.

[32] G.S.Sestito, et al. A method for anomalies detection in real-time Ethernet data traffic applied to PROFINET, IEEE Transactions on Industrial Informatics, 14(5) (2018) pp. 2171-2180.

[33] C.Pereira, A.Pinto, D.Ferreira, A.Aguiar, Experimental characterization of mobile IoT application latency, IEEE Internet of Things Journal, 4(4), (2017), pp. 1082 - 1094.

[34] S.Savage, A.Collins, E.Hoffman, J.Snell, T.Anderson, The end-to-end effects of internet path selection, SIGCOMM Comput. Commun. Rev., 29(4) (1999) pp. 289-299.

[35] P.Ferrari, A.Flammini, D.Marioli, A.Taroni, F.Venturini, 'Experimental analysis to estimate jitter in PROFINET IO class 1 networks', Proc. of the IEEE Conference on Emerging Technologies and Factory Automation ETFA, 2006, Prague, Czech Republic, pp. 429-432.

[36] S.Cavalieri, F.Chiacchio, Analysis of OPC UA performances, Computer Standards & Interfaces, 36(1) (2013) pp. 165-177.

[37] S.Cavalieri, Evaluating overheads introduced by OPC UA specifications, in Advances in Intelligent and Soft Computing, 98 (2012) pp. 201-221