## 1. Introduction

The Modbus TCP protocol is well known in the industrial environment and has been used for years in automation applications. It can be said that this protocol was an evolution of the Modbus Serial (RS485) that started to work over TCP/IP networks.

The Modbus TCP Client-Server (Master-Slave) model solves a wide range of applications. However, with the advent of concepts such as Industry 4.0, IoT and IIoT, 3G/4G and geographically distributed wireless devices, new necessities - that make difficult to use Modbus TCP and its Client-Server model – have arrived.

In order to help solving data communication in applications where the Client-Server model does not , this article presents the Publish-Subscribe model used in MQTT, a protocol widely used in IoT solutions.

This article will present characteristics, advantages and disadvantages of these two protocols and their architectures, also the kind of applications when the use of one or the other is indicated.

## 2. Client-Server model

In computer science, this model is an application framework that distributes tasks and workloads between the providers of a resource or service (**Servers**), and the service requesters (**Clients**). This model was developed by Xerox PARC in the 70's and is currently widely used in many different areas of technology.

Briefly, each client opens a direct connection to a server, forming a 1x1 (one-to-one) connection where the server waits for client requests to respond. Clients have the function of connecting to the server, making requests and receiving responses from the server. Clients never connect to other clients and do not respond to requests. Servers, in turn, wait for connections and client requests and serve them with the requested data. Servers never make requests.

## 3. The Modbus TCP protocol

Modbus TCP uses the Client-Server model, having some similarities with its "older brother" Modbus RTU. To some extent, the Master-Slave model of the Modbus RTU and the Client-Server model are alike, where the Master plays the role of the Client and the Slave is the Data Server. Unlike Modbus RTU, though, Modbus TCP allows a Slave / Server to communicate with multiple Masters / Clients.

The protocol has a Transaction ID, which is a great feature in a network that is allowed to have multiple concurrent requests, ensuring that clients do not confuse the responses sent by the server. For instance, the client can send requests with IDs 1, 2 and 3 and the slave can respond in order 3, 2 and 1. Due to the Transaction ID, the client can associate the request/response pair with no bigger problems. Another Modbus TCP protocol characteristic is the guarantee of messages delivery, due to the transit over the TCP layer, thus guaranteeing the coherence and data delivery.

As for protocol safety, there is no mechanism in Modbus TCP, it has no passwords, authorization, certificates or any other existing security mechanism. To mitigate some of these problems, many manufacturers create their own protection mechanisms for their devices, but all information is still transiting transparently in the network, which can be monitored for malicious purposes. The truth is that if you can access a Modbus device on a network, you can take over this device and can read or write any available register.

## 4. Publish/Subscribe Model

The Publish/Subscribe architecture has a different concept. In this case there are devices / entities that produce and publish data (Publishers) and devices / entities that consume this data (Subscribers). Also, there is an entity that acts as a centralizer in this data exchange, commonly called Message Broker. So, devices create new data and send them to the Broker via messages. The Broker receives and organizes these messages by topics and forwards to the Subscribers the ones with the data of their interest. This interest is formalized when a Subscriber subscribes to receive the information of a topic, which we call subscription. Once you have registered / subscribed a topic, all the information that reaches the Message Broker addressed to this topic will be sent away to subscribers.

It is important to note that in this model, Subscribers never communicate with Publishers, and the Message Broker is always the link between a Publisher and a Subscriber. Eventually, the same device might be a Publisher and a Subscriber.

In this model we have the following characters:

\* Publisher: Is the information provider and connects to the Broker to send messages in a topics format. A Publisher can publish one or more topics and, if it is able to do so, can publish on more than one Message Broker.

\* Subscriber: Is the information consumer and connects to the Message Broker to subscribe to the topics for which it needs to receive information about. The subscriber has the responsibility of maintaining the connection active with Message Broker in order to receive the information.

\* Message Broker: Is a middleware that manages topics, publishers and subscribers. Responsible for receiving messages from publishers and sending to connected subscribers, it ensures the interoperability of the application.

## 5. The MQTT Protocol

Message Queue Telemetry Transport (MQTT) is a lightweight application protocol that is based on the publish / subscribe architecture for sending messages, despite having "message queue" in its name. It has become the standard protocol for communications in IoT.

As the Modbus TCP protocol, MQTT transits over the TCP / IP stack. Created by IBM in the late 1990s, its original application was to link sensors in oil pipelines and satellites. In 2014, MQTT became an open standard, with support to several programming languages ​​and using several free software implementations.

The protocol principles are:

\* Minimize device resource requirements by providing an extremely lightweight, simple and low overhead (between 2 and 4 bytes) interface.

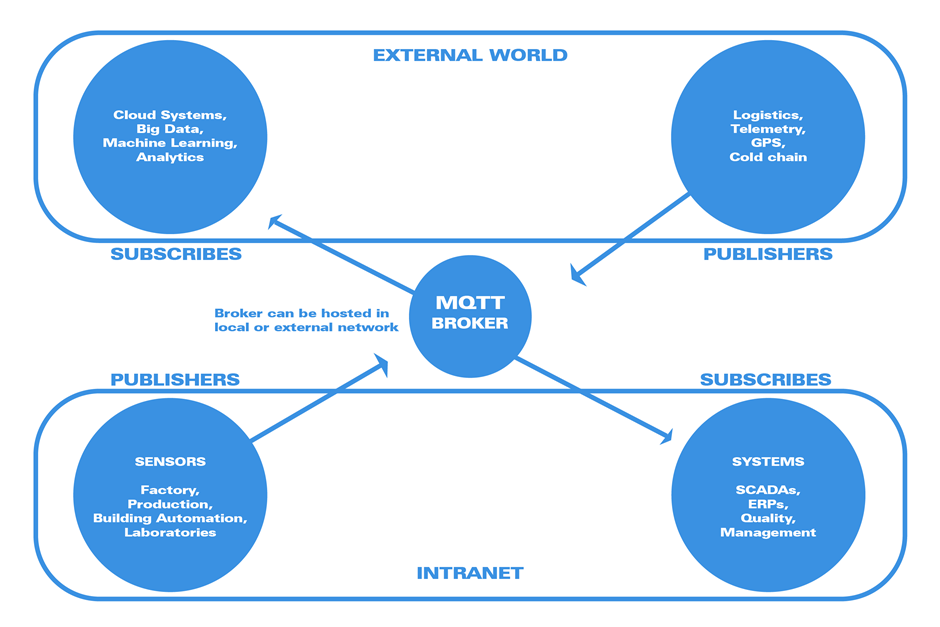
\* Optimize traffic on unreliable, low-latency networks.

\* Perform communication asynchronously between the parties, so that the sender and receiver are decoupled both in space and time.

Due to the use of the publish / subscribe architecture, the MQTT protocol guarantees high scalability in its applications, where numerous devices can publish messages and several other subscribers can consume their information.

Despite being a simple protocol, MQTT does not fail to contemplate security features in the message sending. Connections between clients and Broker can be encrypted (TLS) to protect data in transit. In addition, Brokers may require a login and password to accept a connection. Because MQTT does not restrict the formatting of your message data, payload can still be encrypted by the publisher and then decrypted when the subscribers receive the messages.

The protocol has also mechanisms such as quality of service (QoS), which defines how the message exchanging between Broker and the clients (publishers and subscribers) is performed. The higher the quality of service, the more secure the messaging, but. However, increasing security turns out to slower messaging, as well as increased data traffic in the network.

Due to its interoperability features and ease of implementation, the MQTT protocol is a great solution for IIoT applications and industrial automation systems that seek to have adherence to the "Industry 4.0" concept. The protocol should be considered for the ease with which the data generated by devices can be accessed simultaneously by several systems such as SCADA, ERP, Big Data or Machine Learning systems. In the same way, it is possible to connect the external world (logistics, suppliers, climate, Cloud services) to the internal systems of the organizations.

## 6. MQTT vs. Modbus TCP: When to Use Them

The decision whether or not to use a new technology often raises questions. At such times, it must be verified that the technology that has been used until then continues to supply all the new application necessities. A new technology often can solve problems better and cheaper than the one currently used, or even open up new possibilities for integration among the sectors of the organization. On the other hand, we should not give up an analysis and blindly adopt a new technology just because it is new!

Here are some pros and cons of each of the protocols / architectures:

MQTT protocol advantages:

\* There is no need to know the network, address or physical location of the devices generating data (publishers), subscribers just need to connect to the broker and subscribe the device topics in order to start receiving data.

\* Submitting publisher data does not require a request. So, the publisher is able to manage the best time to send its data to the broker, which enables a more efficient energy management.

\* With a broker hosted outside the company's local network, some issues involving IT security and AT can be avoided, such as opening entry ports for a SCADA to receive data from the devices.

\* Ease of deploying geographically distributed applications with the use of 2G, 3G or 4G networks, due to the problem of device IP being changed with each new connection, which makes  access from other parties much more complicated. In MQTT, all publishers and subscribers have the connection initiative to the broker, never the opposite.

\* It is a protocol with implemented layers of security, login and password for access to brokers and TLS for messages exchange.

\* Possible applications where many subscribers can receive information from many publishers.

\* The vast majority of cloud services support this protocol: Amazon AWS, Microsoft Azure, IBM Bluemix and Google Cloud, among the best known.

* Modbus TCP Protocol advantages:

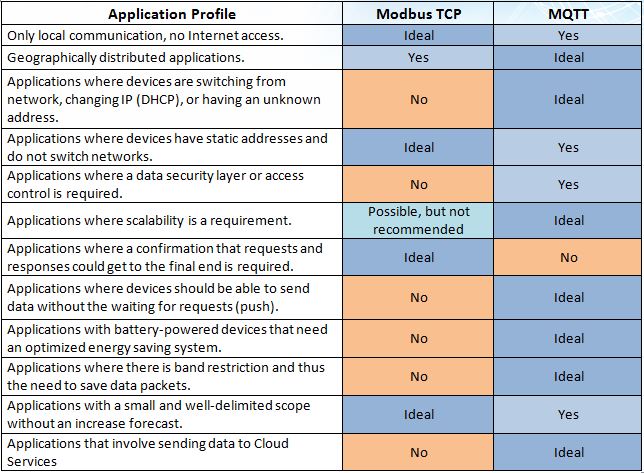
\* It does not need an intermediary (broker) since it is a point-to-point connection.

\* Ensures that information will reach its final destination, i.e. the client knows that its request has reached the server and the server knows that the response has reached the client.

\* Servers that allow multiple connections can serve multiple clients simultaneously, which makes possible that, for example, both a SCADA software and a mobile supervisory app communicate with the same device to read its data.

\* Clients can obtain information from the server at any time without having to wait for the spontaneous sending of the clients.

## 7. Application profile for each protocol

The following table has several applicationn profiles and, for each one, a suggestion of which protocol is best suited to it. In many cases it will be possible to use any of the protocols and - in these cases – factors such as implementation costs and solution complexity should be taken into account in order to choose the best option. Table 1 - Application profile x Ideal protocol

## 8. Conclusion

With all that was presented, we can conclude that there is no better or worse protocol, but rather more suitable applications for one or another architecture / protocol.

Why should I use MQTT if Modbus has always met my needs? The novelty of MQTT and the tradition of Modbus TCP must always be evaluated case-by-case: in some cases the advantage in changing is evident, in others there is no need at all.

**9. Reference**

<https://www.novusautomation.com/site/default.asp?Idioma=1&TroncoID=053663&SecaoID=0&SubsecaoID=0&Template=../artigosnoticias/user_exibir.asp&ID=939463>