Message Queuing Telemetry Transport(MQTT)

Muhammad Moaz Amin
dept. Electronic Engineering
Hochschule Hamm Lippstadt
muhammad-moaz.amin@stud.hshl.de

Luis Cabezas Suarez
dept. Electronic Engineering
Hochschule Hamm Lippstadt
luis-david.cabezas-suarez@stud.hshl.de

Dawar Zaman
dept. Electronic Engineering
Hochschule Hamm Lippstadt
dawar.zaman@stud.hshl.de

***Abstract*—MQTT protocol specifies the quality of service, which guarantees the reliability of message delivery under different network environments. The design of QoS is the focus of the MQTT protocol. As a protocol specifically designed for IoT scenarios, MQTT's operating scenarios are not only for PC, but also in a wider range of narrow-bandwidth networks and low-power devices. In this paper relevant application areas of MQTT and its protocol will be analyzed. We will also compare its various properties and features with various MQTT implementations. At the end based upon our findings and comparison a conclusion and future researches for improvement to resolve current issues will be given.**

***Index Terms*—QoS- Quality of Service, TCP- Transmission Control Protocol, TLS- Transport Layer Security, WoT- Web of Things, M2M- Machine to machine.**

I. Introduction

There are billions of smart devices connected to the internet today. These smart devices, also called "Things" form the IoT (Internet of Things). IoT devices are everywhere from homes to schools to hospitals to factories. Communication between these devices and other internet services is possible due to the presence of various communication protocols. These devices enable us to collect monitored data from various sources over the network without any human-to-human or human-to-computer interaction that can be used to improve our life. [2]

The main purpose of these protocols is to collect, process, describe and evaluate data to realise a solution for these various problems that occur. The aim of IoT is to provide bi-directional communication between interconnected devices(actuators, sensors, micro-controller, cloud devices etc). One such communication protocol is the Message Queuing Telemetry Transport or "MQTT" for short.

A. What is MQTT?

MQTT is a platform specifically designed for the IoT. One of the biggest advantages for using MQTT for IoT applications is that it's a lightweight protocol that has high message reliability and has smooth bi-directional communication capability.

II. How MQTT works?

AS MQTT is a open messaging protocol, it provides resource constrained network clients with a simple way to distribute telemetry information in a low bandwidth environment. Its main aim is to maximise the available bandwidth as its communication model is an alternative to traditional client server architecture which communicates directly with the endpoint. A device using the MQTT platform for M2M or IoT connectivity can be divided into two main elements as described below. [3]

A. Client

Client could be a Publisher or Subscriber and it always establishes the network connection to the Server (Broker). It can do the following goods.

- Publish communications for the interested druggies.
- Subscribe in interested subject for entering communications.
- Unsubscribe to pull from the subscribed subjects.
- Detach from the Broker.

1) Publisher:

- The publisher is the initiator of the message between the devices. It sends the message to the broker.

2) Subscriber:

- The subscriber is connected to the Broker. Whenever there are any updates the broker sends the message to the subscriber.

B. Broker:

- The Broker is the central part of the MQTT communication. It constantly takes in data from the Publisher and sends it to the Subscriber.
- Accept Account requests.
- Receives Published communications by Addicts.
- Processes different requests like Subscribe and Unsubscribe from Addicts.
- After admitting communications from publisher sends it to the interested Addicts.

MQTT works on the basis of this Pub/Sub messaging technique. Its basic Model is shown in Figure 1.

III. MQTT QoS levels

The agreement between the sender of the message and the receiver of a message which defines the deliverance guarantee for a specific message is **Quality of Service**(QoS). When we talk about QoS in MQTT we need to consider both sides of the message.

1) Message delivery form the publishing client to the broker.
2) Message delivery from the broker to the subscribing client.

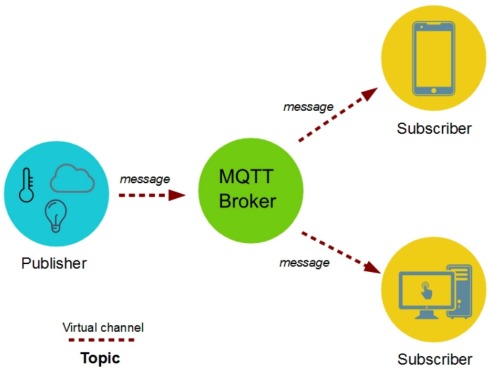

Fig. 1. MQTT Model [1]

The three levels of MQTT are the following:

- QoS 0
- QoS 1
- QoS 2

Working principle of QoS Levels

1) QoS 0 - Publish once at most: It is a "fire and forget" message sending mode: After a sender (possibly Publisher or Broker) sends a message, it no longer cares whether it is sent to the other party or sets up any resending mechanism. It is also the minimal QoS level. At QoS level 0, publication of messages depend upon the capabilities of the underlying network. In this case publisher will just publish the message only once, the receiver will not answer the message, and the publisher will not save and resend the message. The transmission efficiency is highest at this level. [2], [5]

Fig. 2. QoS level 0. delivery at most once [5]

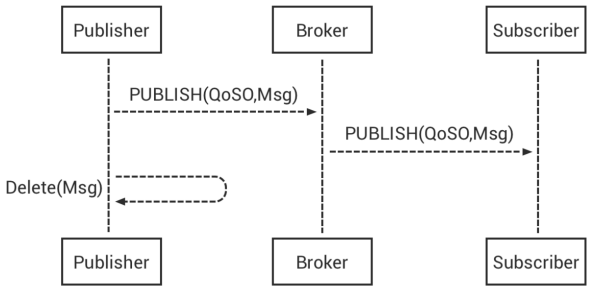

Fig. 3. QoS level 0. Sequence Diagram [2]

2) Qos 1 - Publish once at least: It includes a simple re sending mechanism. After the Sender sends a message, it waits for the receiver's ACK. If it does not receive the ACK, it resends the message. This mode can guarantee that the message can arrive at least once, but it cannot guarantee that the message is repeated. Referring the QoS level 1 as seen in Figure 5, Referring the QoS level 1, this communication can be asserted at least once. MQTT guarantees the QoS 1 level through a innocent ACK procedure. The publisher will advertise the dispatch and stay for the comeback of the receiver's which is giving as PUBACK packet. However, the publisher will concrete the dispatch's DUP to 1 and resend the dispatch, If the PUBACK reply is not admitted within the defined time. The receiver should reply to the PUBACK communication when taking a dispatch with QoS 1 level . The receiver may take over the equal dispatch numerous times, therefore as a final result the receiver will treat the message as a new dispatch and transport a PUBACK packet as a return. [2], [5]

Fig. 4. QoS level 1. delivery at least once [5]

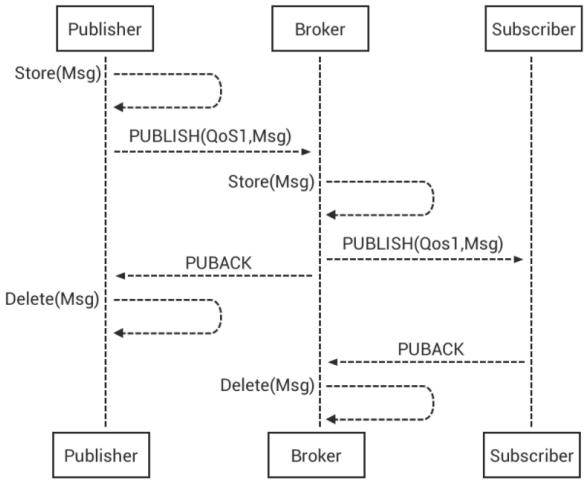

Fig. 5. QoS level 1. Sequence Diagram [2]

3) Qos 2 - Publish only once: It is designed as a resending and repeating message discovery mechanism to ensure the message will arrive only once. Publishers and subscribers work together to make sure that communications are published only once through two sessions. There's another cost about using this quality of service . This is happening when the publisher sends or display a certain communication with a QoS of 2, then as a result it would save the published dispatch and stay

and wait for the receiver to reply with the PUBREC. After the publisher receives the PUBREC message, it can safely discard the previously published message because it already knows the receiver successfully took the dispatch. Whenever packet loss occurs during transmission, the sender is responsible for resending the prior communication. This is true regardless of whether the sender is Publisher or Broker. This whole process is described in Figure 7. [2], [5]

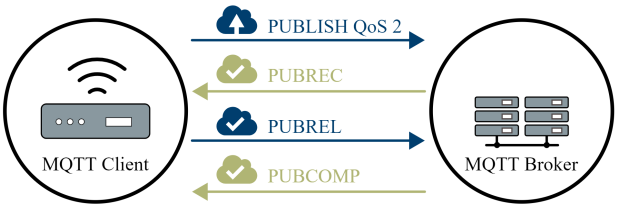

Fig. 6. QoS level 2. delivery exactly once [5]

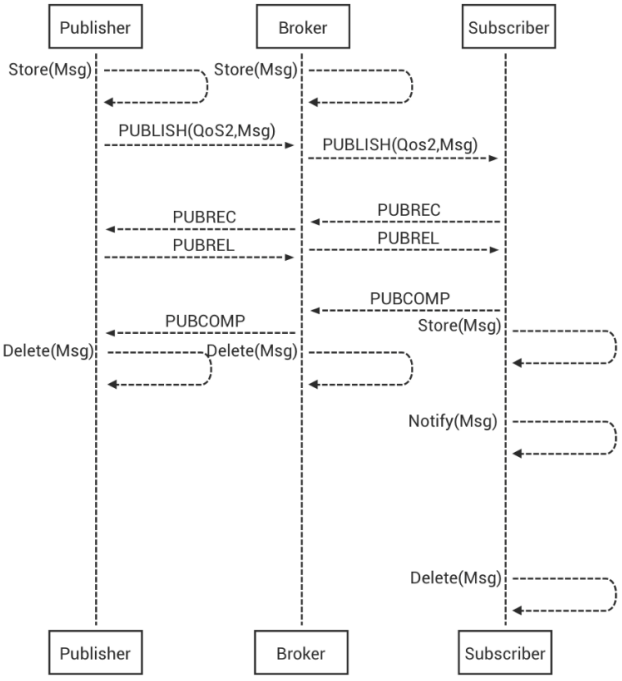

Fig. 7. QoS level 2. Sequence Diagram [2]

How to choose QoS

If the QoS level would be high it would in turn consume more system resources as it corresponds to more complicated processes. The application itself chooses it on the basis of network scenarios and business requirements. [2], [5]

- QoS 0 can be chosen in the following cases.
  occasionally lost messages scenario is accepted
  In the scenario that the message interaction between the internal services in the same subnet, or the network of other client and server is very stable.
- QoS 1 can be chosen in the following cases
  Focus in on the utilization of framework assets and wish streamlined performance.
  Can not lose any message, yet can acknowledge and deal with copy messages.
- QoS 2 can be chosen in the following cases
  It is unacceptable that lost message(the loss of message may result in loss of life or property), and do not want to receive duplicate messages.
  For some industries such as a bank, firefight, aviation, etc that require high completeness of data and timeliness.

The difference of QoS in publishing and subscribing

QoS of MQTT publishing messages is not end-to-end, but it is between the client and the server. It is totally dependent on the level of QoS of the published messages and the Qos of the topic subscription that at which level do the subscribers receive MQTT messages. [2]

- if the client A's publish utilizes more QoS than client B's subscription, the QoS of server forwarding messages to client B is the QoS used by client B's subscription.
- and if Client A's publish utilization of QoS is less than the QoS used by client B's subscription, the QoS of server forwarding messages to client B is the QoS used by client A's publishing.

IV. Supported Network Protocols

As MQTT is quite lightweight, it works well with applications that involve remote monitoring including the following:

- synchronization of sensors, e.g Fire detectors
- health monitoring conditions which make use of sensors
- sensors for alerting people in danger

It is preferred to use TCP/IP stack for running MQTT nevertheless other protocols such as SSL/TLS can also be used. Theoretically, MQTT can utilize any network protocol that provides ordered, lossless and bi-directional connections. Since MQTT is mostly used for IoT related applications therefore using SSL/TLS is not always preferred. If a secure connection is required then SSL/TLS has to be used as MQTT on its own is unencrypted. [13]

A typical MQTT session has the following four phases:

1) Connection
2) Authentication
3) Communication
4) Termination

In order for MQTT to work on SSL/TLS all of these have to be followed.In a normal TCP/IP setting the client initiates the connection by making a TCP/IP connection with the broker using standard or custom ports. However, as SSL/TLS is a secured protocol, the client and broker send and receive certificates from each other and unless these certificates are validated from each side the connection is not possible. [13]

There might always be drawbacks when using MQTT inSSL/TLS for IoT applications. For example, if authentication

is required by the server from the client in clear text format then the client has to send the information such as the username and password in clear text format. Furthermore,SSL/TLS is also not a lightweight protocol which ignores the purpose of using it with MQTT in the first place. MQTT was not designed to be the most secure protocol and in cases where security was concerned it would be used in secure back-end networks for application-specific purposes [13], [14]

Although MQTT is considered to be compatible with SSL/TLS the extent of its functionality is questionable. Therefore, it is recommended to use MQTT with TCP/IP.

TCP/IP is the best solution when working with MQTT because rather than going to the application layer again for SSL encryption the connection saves time by jumping directly into the Transmission Layer. [13], [14]

A. Other Commonly used Protocols

1) Constrained Application Protocol (CoAP: It uses a request/response communication pattern and is quite suitable for IoT.

2) Advanced Message Queuing Protocol (AMQP): It uses publish/subscribe communication pattern and is similar to MQTT.

3) Simple/ Streaming Text Oriented Messaging Protocol (STOMP): It does not deal with ques and topics instead it uses a send semantic with a destination string. Overall it is a text based protocol.

4) Simple Media Control Protocol (SMC): It is Constrained Application based protocol which is C based and is used in embedded environments.

5) SSI (Simple Sensor Interface): This is a communication protocol between multiple computers and sensors and is for the data transfer.

6) Data Distribution Service (DDS): This protocol is generally a middle ware standard that can directly publish or subscribe communications in real time in embedded systems. [12]

V. MQTT vs HTTP

What is HTTP?

First of all if we are comparing HTTP with MQTT we need to understand that is a protocol based on the client-server principle: requests are sent by one entity: the user agent (oracle proxy). Most of the time the user agent (client) is a Web browser, but it could be any other program, such as a robot program, that scans the Web, to acquire data on its structure and content for use by an Internet browser. Every request is sent to a server, which handles and responds to it. Between each request and response, there are several intermediaries, usually called as proxies, which perform different functions, such as gateways or caches. In real ambience ,there are more intermediary elements, between a browser and the server that manages its request: there are other types of devices: such as routers, modems...etc. It is thanks to the layered architecture of the Web that these intermediaries are transparent to the browser and the server, since HTTP relies on network and transport protocols. HTTP is an application protocol, and therefore relies on the above. Although for diagnosing problems in communication networks, the lower layers are irrelevant to the definition of the HTTP protocol. [9]

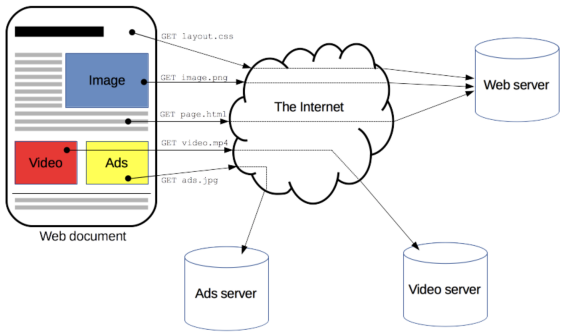

Fig. 8. Communication behavior of HTTP [9]

HTTP

- In HTTP IoT devices communicate directly with each other.
- HTTP works using the Request/Response technique.
- It is a document centric messaging protocol that also uses a lot of power.
- In HTTP the devices must always be ready to receive the messages.
- HTTP protocol sends multiple small packets to the server to get connected. This may lead to operation of high business which may encourages high network resource utilisation and may lead to network hold back.
- HTTP works on TCP/IP which provides a steady communication.
- Ordinarily, all HTTP calls are stateless which lead to doing authentication every time it connects as it's connected to IP or URL to do Rest API calls so the session isn't saved. So, after getting the response the device closes the connection. In turn, IoT causes serious price in network communication.

.

MQTT

- In MQTT devices can choose when to receive the messages.
- MQTT uses a broker to communicate with the IoT devices therefore there is an indirect communication between the devices.
- It uses the Pub/Sub messaging technique.
- It is a data centric messaging protocol that uses low power and low resources.
- MQTT protocol is designed to transfer a dispatch to one or fresh device with smaller inactivity. The quantum of dispatch can be 0¬256 Mb but it 's not recommended to transfer a large measure of data.

- In IoT usages, the quantum of data is too low so that we can use this protocol for messaging over fragile networks.
- MQTT consists of two communication sets on a connection, Publish and Subscribe. The Dispatch is transfer using publish and the dispatch can be took by subscribing it. The dispatch is linked by the subject registered by Subscribe dispatch, in advance.

Figure 9 and 10 show the result of a survey carried out for comparing response time over one connection of cycle using either the protocol.

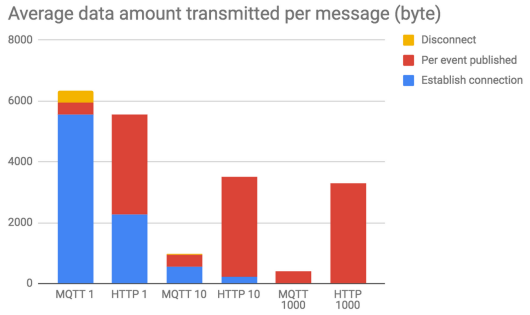

Fig. 9. Average amount of data transmitted per message for different numbers of messages transmitted over the same connection [10]

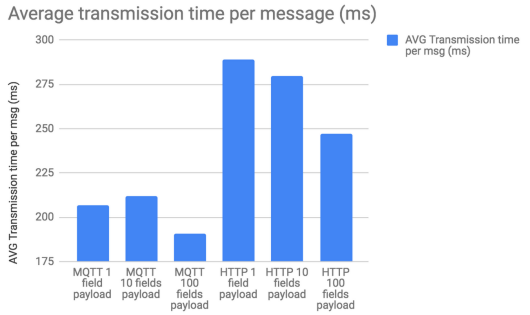

Fig. 10. Transmission time for different message sizes [10]

As we represented in the previous graphs. The idea of this experiment was about changing the calculation of communications. It was done by checking the response time for packing 1, 100, and 1000 communications over a single connection cycle each for the MQTT, as well as captured the packet sizes that were packed over the lace. Therefore, we measured the response time for packing a single communication with 1, 10, and 100 property fields over a single connection cycle each and capture the packet size packed. And then for the HTTP the idea was to measure the average response time for packing a freight with 1, 10, and 100 property fields and capture the packet size over the lace. Showing as a result the behavior of the HTTP and MQTT in a example [10]

VI. Web of Things Architecture

Nowadays, describing the essence of Internet of things in a single sentence is nearly impossible because of it's presence in almost all the sectors present. Nevertheless, in simple words and in a broader prospective IoT is simply a world in which with the help of embedded devices internet becomes more than just multimedia pages it is today but extends to the real time world. We have come very far away from the point of just imagining about feeling the world through sensors and then evaluating that information. Nowadays, such concepts are rapidly becoming reality because of which exchanging information and getting the desired results, all has become very much possible. It is because of colossal progress in embedded devices which have brought a new class of objects in the world, Smart Things. Internet of Things is a network of Things which are anything that can be connected in some form of Internet. However to interact with these Things we still have to interact with their own separate interfaces. There is no single universal application to run all the Things with a single interface. [11]

So instead of creating and interacting with each individual Thing, why should we not reuse something and utilize something that already exists in the world. Reusing and leveraging is the main concept behind Web of Things. The Web of Things architecture can be considered as a successor to the current IoT architecture. The WoT uses "Thing Descriptions" which are used as a sort of metadata to connect devices that are not recognized as IoT to the IoT environment. The thing descriptions ease the process of making devices IoT capable. MQTT is also supported for the WoT architecture which is given in Figure 11.

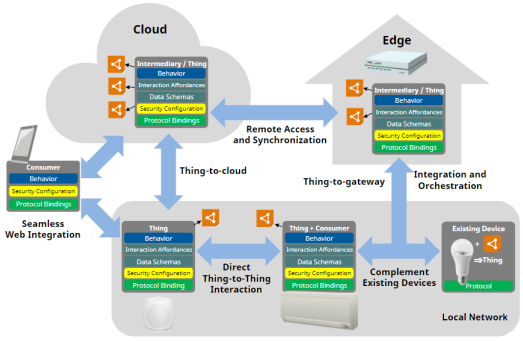

Fig. 11. Abstract Architecture of Web of Things [11]

By combining the power of Reliable messaging using MQTT with the Web of Things Architecture we can define new devices using Thing Descriptions (TD) or make new TD's for existing non-IoT devices and then make these devices discoverable and intractable using a fast and reliable communication technique.

VII. Conclusion

The aim of this paper was to dive into this system architecture called MQTT, After having done research we ended

up in a idea where MQTT is considered the best option to have as the IoT protocol. Admiring the architecture based on keeping the lightweight communication which is achieved with the MQTT protocol itself.

However, We have seen a few hints of MQTT, a very interesting protocol for IoT (as we mentioned before) but nevertheless MQTT has the advantages of a pub/sub pattern and stands out for the simplicity of communication. This make it as an interesting tool like one of the standards for both commercial and maker IoT applications. Of course, there are many more aspects that we could talk much more about MQTT that we cannot forget such as advanced security features, permanence of messages in the broker, configuration of multiple brokers. Having some features that makes this MQTT famous for.

Therefore, as we could see MQTT will become a platform for connecting many devices over the internet safely, quickly and reliably. As there are also different versions of MQTT . Hence there is also a chance that newer MQTT versions will have better support for Web based applications and will also have increased functionality in this direction.

REFERENCES

[1] Bernstein, C., Brush, K.,&. amp; Gillis, A. S. (2021, January 27). What is MQTT and How Does it Work? IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport.

[2] Gubbi, J., Buyya, R., Marusic, S., &. amp; Palaniswami, M. (2013, February 24). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems. https://www.sciencedirect.com/science/article/pii/S0167739X13000241.

[3] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," in IEEE Access, vol. 8, pp. 201071-201086, 2020.

[4] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh and R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT protocol," 2017 International Conference on Engineering and MIS (ICEMIS), 2017.

[5] Team, T. H. M. Q. (n.d.). Quality of Service 0,1 &.; 2 - MQTT Essentials: Part 6. https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/.

[6] Kent, S., Atkinson, R.: "Security Architecture for the Internet Protocol", RFC 2401, November 1998

[7] Fumy, Walter: "Internet Security Protocols", State of the Art in Applied Cryptography, B. Preenel, V. Rijmen (Eds.) COSIC'97 Course, Springer LNCS 1528, pp. 186-208, 1998

[8] Stallings, William: "Cryptography and Network Security", 2nd Edition, Prentice Hall, 1998

[9] Generalidades del protocolo HTTP - HTTP: MDN. (n.d.). Retrieved from https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[10] Google. (n.d.). HTTP vs. MQTT: A tale of two IoT protocols — Google Cloud Blog. Google. https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols.

[11] Web of Things (WoT) Architecture. W3C. (n.d.). https://www.w3.org/TR/wot-architecture/#sec-wot-architecture.

[12] Bernstein, C., Brush, K., & Gillis, A. S. (2021, January 27). What is MQTT and How Does it Work? IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport.

[13] Grigorik, Ilya. "Networking 101: Transport Layer Security (TLS) - High Performance Browser Networking (O'Reilly)." High Performance Browser Networking, O'Reilly, 10 Dec. 2017, hpbn.co/transport-layer-security-tls/

[14] Steve, et al. "Mosquitto SSL Configuration -MQTT TLS Security." —, 30 Sept. 2020, www.steves-internet-guide.com/mosquitto-tls/.