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.**

***Index Terms*—component, formatting, style, styling, insert**

I. What is MQTT?

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. One such communication protocol is the Message Queuing Telemetry Transport or "MQTT" for short. 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.

II. How MQTT works?

A device using the MQTT platform for M2M or IoT connectivity will have the following components:

A. Publisher

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

B. Subscriber

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

C. 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.

MQTT works on the basis of this Pub/Sub messaging technique.

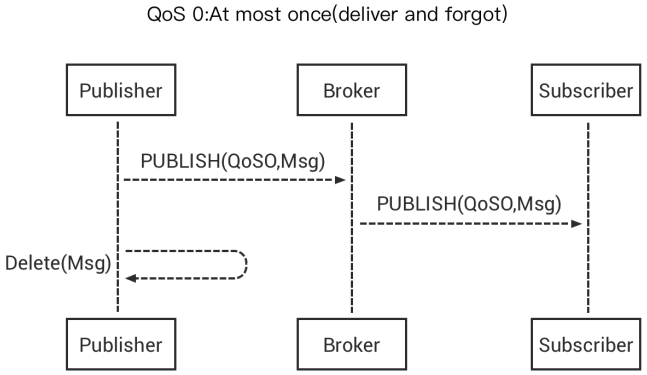
III. MQTT QoS levels

MQTT has designed 3 QoS levels.

- QoS 0
  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.
- QoS 1
  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.
- QoS 2
  designed a resending and repeating message discovery mechanism to ensure the message will arrive only once.

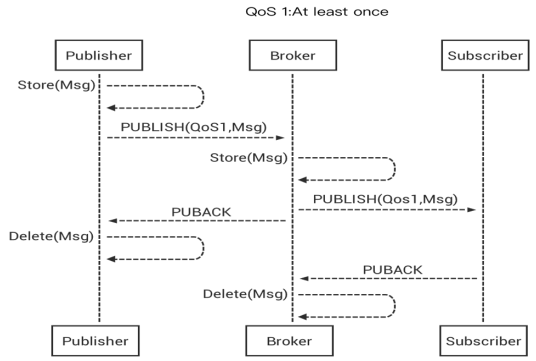
Working principle

1) QoS 0 - Publish once at most: When QoS is 0, the publish of messages depends on the capabilities of the underlying network. The publisher will publish the message only once, the receiver will not answer the message, and the publisher will not save and resend the message. Messages have the highest transmission efficiency at this level, but may not be delivered once.

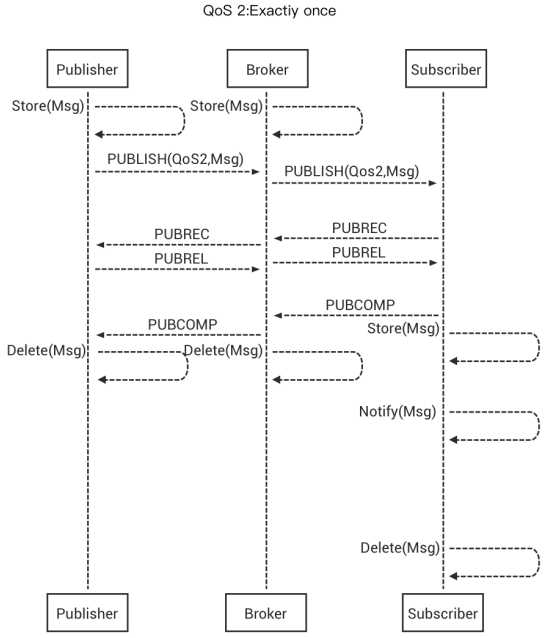


2) Qos 1 - Publish once at least: When the QoS is 1, the message can be guaranteed to be published at least once. MQTT guarantees QoS 1 through a simple ACK mechanism.

The publisher will publish the message and wait for the response of the receiver's PUBACK packet. If the PUBACK response is not received within the specified time, the publisher will set the message's DUP to 1 and resend the message. The receiver should respond to the PUBACK message when receiving a message with QoS 1. The receiver may accept the same message multiple times. Regardless of the DUP flag, the receiver will treat the received message as a new message and send a PUBACK packet as a response.



3) Qos 2 - Publish only once: When the QoS is 2, publishers and subscribers ensure that messages are published only once through two sessions. This is the highest level of service quality, and message loss and duplication are unacceptable. There is an additional cost to using this quality of service level.



After the publisher publishes a message with a QoS of 2, it will store the published message and wait for the receiver to reply with the PUBREC message. After the publisher receives the PUBREC message, it can safely discard the previously published message because it already knows the receiver successfully received the message. The publisher saves the PUBREC message and responds with a PUBREL, waiting for the receiver to reply with the PUBCOMP message. When the sender receives the PUBCOMP message, it will clear the previously saved state. When the receiver receives a PUBLISH message with a QoS of 2, it processes the message and returns a PUBREC in response. When the receiver receives the PUBREL message, it discards all saved states and responds with PUBCOMP. Whenever packet loss occurs during transmission, the sender is responsible for resending the previous message. This is true regardless of whether the sender is Publisher or Broker. Therefore, the receiver also needs to respond to each command message.

How to choose QoS

The higher QoS level corresponds to more complicated processes and the greater the consumption of system resources. The application can choose the appropriate QoS level according to their network scenarios and business requirements.

- QoS 0 can be chosen in the following cases.
  It is accepted that messages are occasionally lost.
  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 on the consumption of system resources and wish optimized performance.
  Can not lose any message, but can accept and process duplicate 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 between the client and the server. The QoS level at which subscribers receive MQTT messages ultimately depends on the QoS of the published message and the QoS of the topic subscription.

- When the QoS used by client A publish is greater than the QoS used by client B's subscription, the QoS of server forwarding messages to client B is the QoS used by client B's subscription.
- When the QoS used by client A publish 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. MQTT vs 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. 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. In MQTT devices can choose when to receive the messages.

V. Web of Things Architecture

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.

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 interactable using a fast and reliable communication technique.

Acknowledgment

References

[1] Abeta, S., 2010. Toward LTE commercial launch and future plan for LTE enhancements (LTE-advanced). Proceeding of the IEEE International Conference on Communication Systems (ICCS, 2010), pp: 146-150.

[2] Alubady, R., M. Al-Samman, A. Habbal, S. Hassan and S. Arif, 2015. Performance analysis of reactive and proactive routing protocols in MANET. J. Eng. Appl. Sci., 10(3): 1-11.

[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] Dierks, T., Allen, C.: "The TLS Protocol, Version 1.0", RFC 2246, January 1999

[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