INTERNET PROTOCOLS

rev1.0 20/04/2020

GOAL

Get a view of the different protocols to send data over internet.

PREREQUISITES

Software needed:

- none

Hardware used in this example:

- none

Internet protocol suite

The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed, and received.

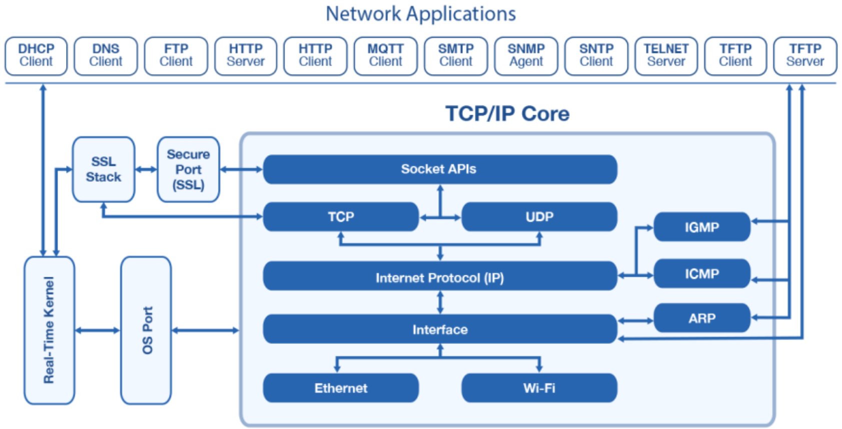
The most common internet protocols are:

- *TCP/IP*
- *MQTT*
- *HTTP*
- *FTTP*

TCP/IP

TCP/IP is a stream protocol. This means that a connection is negotiated between a client and a server. Any data transmitted between these two endpoints is guaranteed to arrive, thus it is a so-called lossless protocol. Since the TCP protocol (as it is also referred to in short form) can only connect two endpoints, it is also called a peer-to-peer protocol.

Between the network applications of the TCP/IP core, we can find the **MQTT** protocol.

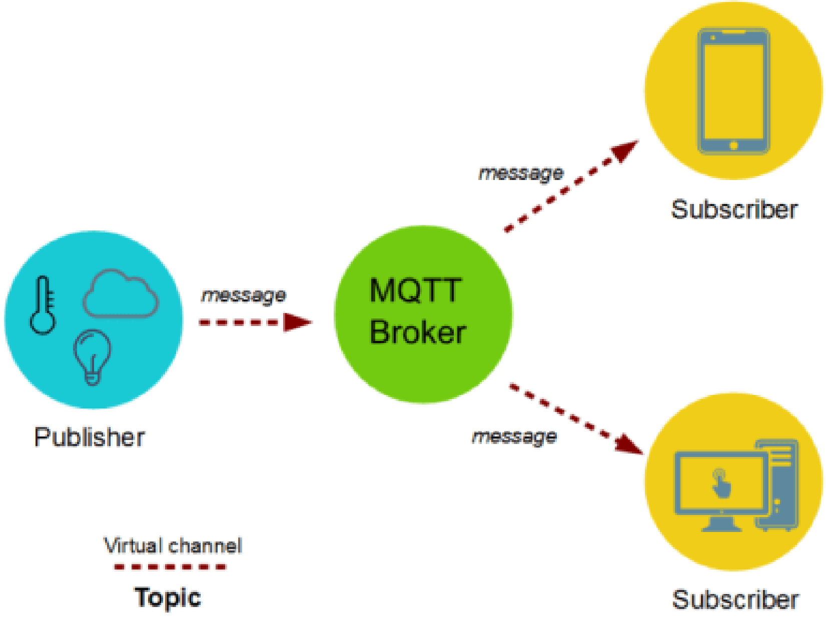


MQTT (Message Queuing Telemetry Transport)

MQTT is a lightweight, “*publish-subscribe*” network protocol: it usually runs over TCP/IP.

The MQTT protocol defines two types of network entities: the **message broker** and the **client**.

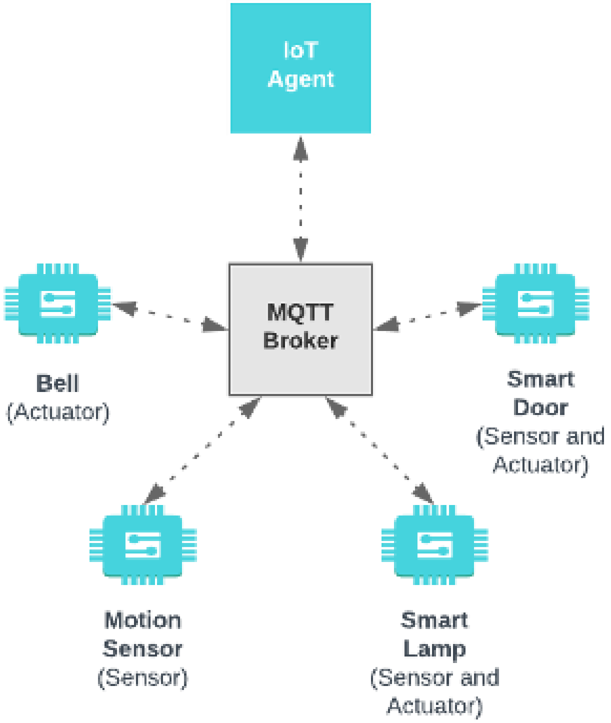
Information is organized in a hierarchy of **topics**. When a publisher has a new item of data to distribute, it sends a message with the data to the connected broker. The broker then distributes the information to any subscriber that have subscribed to that topic.



MQTT broker

The broker is a software usually running on a computer or on a cloud. Multiple clients can receive the message from a single broker (*one to many capability*). Similarly, multiple publishers can publish topics to a single subscriber (*many to one*).

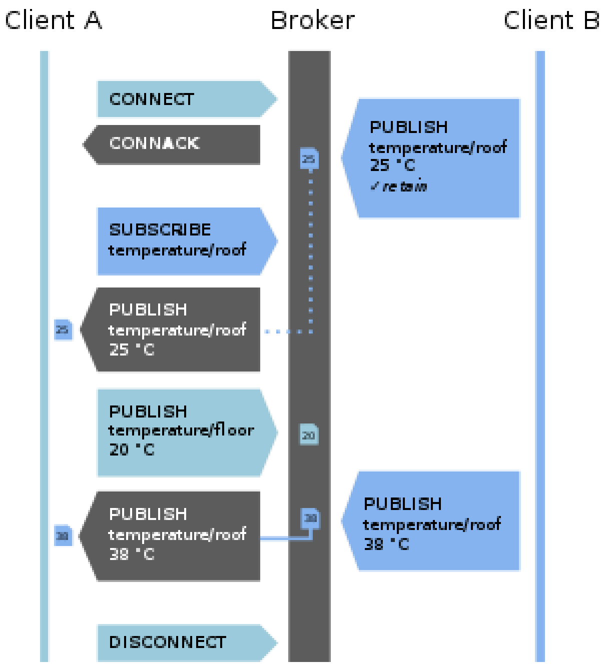
With MQTT protocol, the devices can publish sensor data and still be able to receive the configuration information or control commands.


IoT Agent
MQTT Broker
Bell (Actuator)
Smart Door (Sensor and Actuator)
Motion Sensor (Sensor)
Smart Lamp (Sensor and Actuator)

Message types

The MQTT protocol is very simple, there are only tree kind of messages:

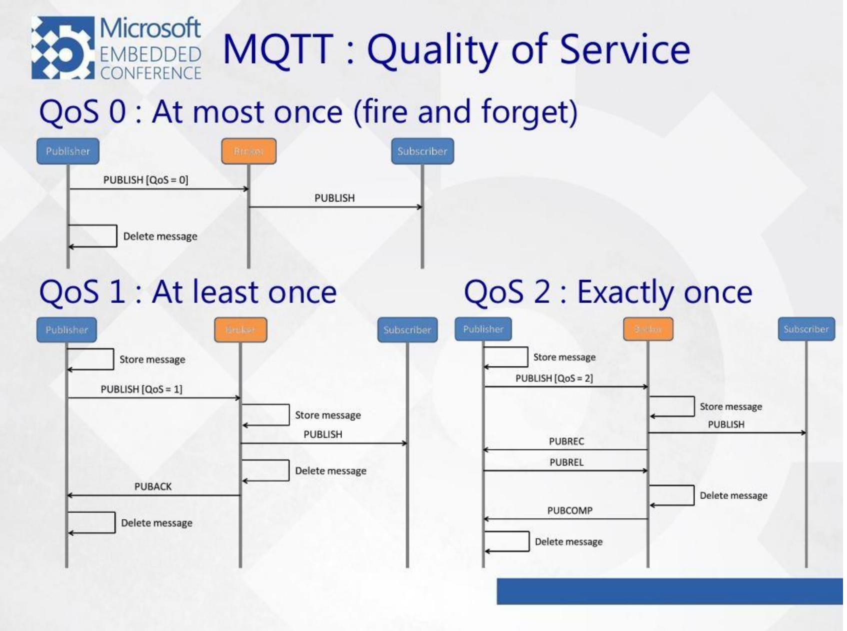
- ***Connect***: Waits for a connection to be established with the server and creates a link between the nodes.
- ***Disconnect***: Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.
- ***Publish***: Returns immediately to the application thread after passing the request to the MQTT client.


Client A
Broker
Client B
CONNECT
CONNACK
PUBLISH temperature/roof 25 °C ✓retain
SUBSCRIBE temperature/roof
PUBLISH temperature/roof 25 °C
PUBLISH temperature/floor 20 °C
PUBLISH temperature/roof 38 °C
PUBLISH temperature/roof 38 °C
DISCONNECT
25
25
20
38
38

Quality of service (QoS)

Each connection to the broker can specify a **quality of service** measure. These are classified in increasing order of overhead:

- *At most once* - the message is sent only once and the client and broker take no additional steps to acknowledge delivery (fire and forget).
- *At least once* - the message is re-tried by the sender multiple times until acknowledgement is received (acknowledged delivery).
- *Exactly once* - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received (assured delivery).



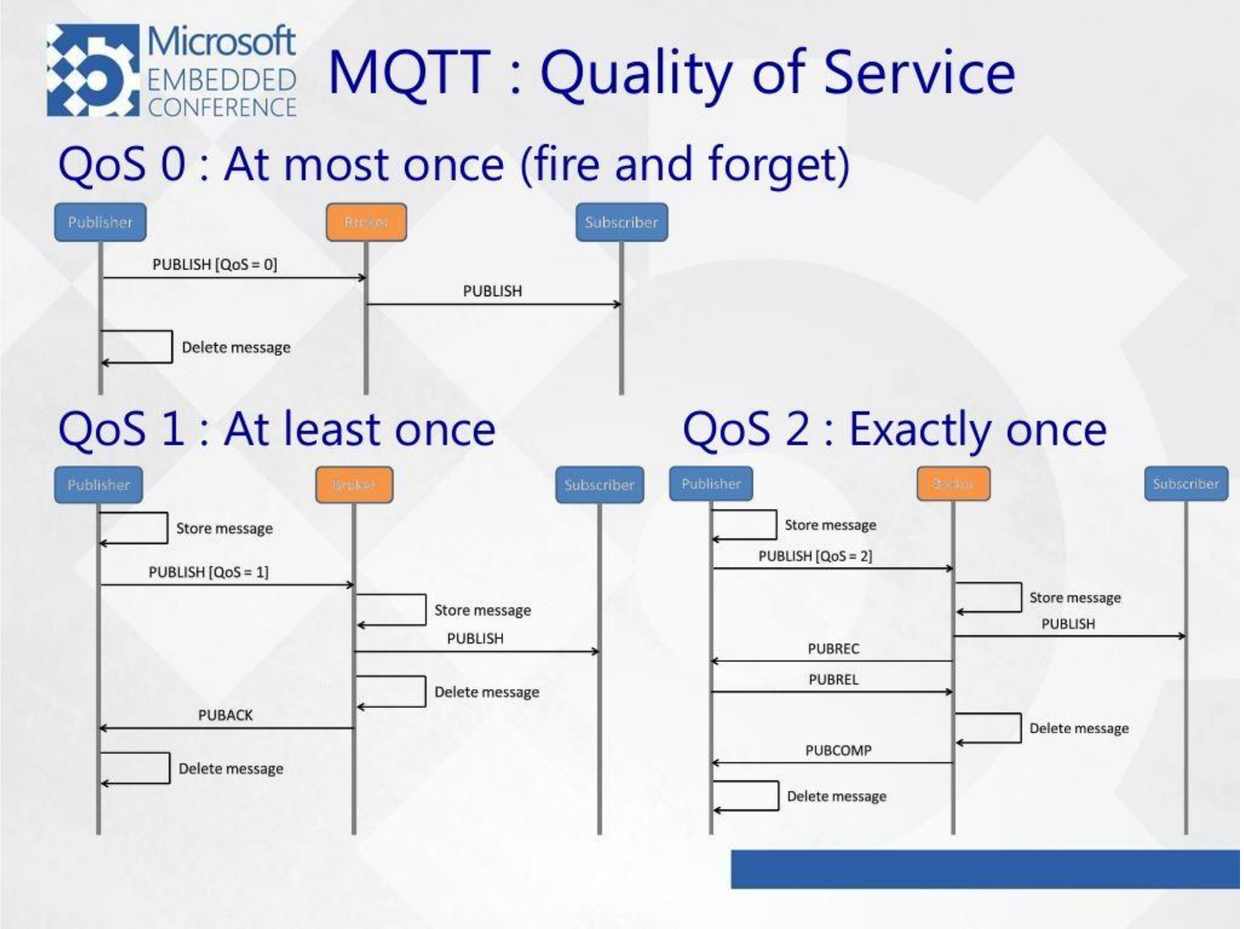
Microsoft EMBEDDED CONFERENCE

MQTT : Quality of Service

QoS 0 : At most once (fire and forget)

QoS 1 : At least once

QoS 2 : Exactly once



MQTT : Quality of Service

QoS 0 : At most once (fire and forget)

QoS 1 : At least once

QoS 2 : Exactly once

Advantages of MQTT

A few desirable features of such a protocol are:

- Small code footprint (to make it easy to implement in small devices)
- Low power consumption
- Low bandwidth consumption
- Low latency
- Use of a *publish/subscribe* pattern

For these reasons is perfect for embedded applications and IoT Devices.