

R-Pi

Team Emertxe



IoT Protocols

MQTT

MQTT

Introduction

- MQTT is a lightweight publish/subscribe messaging protocol designed for M2M (machine to machine) telemetry in low bandwidth environments
- It was designed by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 for connecting Oil Pipeline telemetry systems over satellite
- MQTT stands for MQ Telemetry Transport but previously was known as Message Queuing Telemetry Transport
- MQTT is fast becoming one of the main protocols for IOT (internet of things) deployments



- There are two versions of MQTT
 - v3.1
 - v5
- The original MQTT is designed for TCP/IP networks
- MQTT-SN which was specified in around 2013, and designed to work over UDP, ZigBee and other transports

How MQTT works: Client Connections

- Is a messaging protocol
- It was designed for transferring messages
- Uses a publish and subscribe model
- This model makes it possible to send messages to 0,1 or multiple clients
- In MQTT a publisher publishes messages on a topic and a subscriber must subscribe to that topic to view the message
- There is no direct connection between the broadcaster and the viewer

MQTT

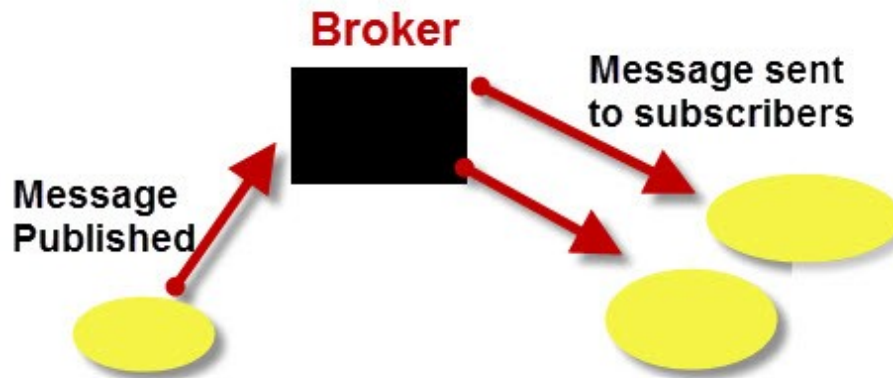
Working: Analogy

- A TV / Radio broadcaster broadcasts a TV / Radio program using a specific channel and a viewer tunes into this channel to view the broadcast

MQTT

Publish Subscribe Model

- MQTT requires the use of a **central Broker**



MQTT- Publish Subscribe Model

MQTT

Working: Imp Points

- Clients do not have addresses like in email systems, and messages are not sent to clients
- Messages are published to a broker on a topic
- The job of an MQTT broker is to filter messages based on topic, and then distribute them to subscribers
- A client can receive these messages by subscribing to that topic on the same broker
- There is no direct connection between a publisher and subscriber
- All clients can publish (broadcast) and subscribe (receive)
- MQTT brokers do not normally store messages

MQTT

Client-Broker: Connections

- MQTT uses TCP/IP to connect to the broker
- Most MQTT clients will connect to the broker and remain connected even if they aren't sending data
- Connections are acknowledged by the broker using a Connection acknowledgement message
- MQTT clients publish a keepalive message at regular intervals (usually 60 seconds) which tells the broker that the client is still connected

MQTT

The Client Name

- All clients are required to have a client name
- The client name is used by the MQTT broker to track subscriptions etc
- Client names must also be unique
 - If you attempt to connect to an MQTT broker with the same name as an existing client then the existing client connection is dropped
 - Because most MQTT clients will attempt to reconnect following a disconnect this can result in a loop of disconnect and connect

MQTT

Clean Sessions

- MQTT clients will usually by default establish a clean session with a broker
- A clean session is one in which the broker isn't expected to remember anything about the client when it disconnects
- With a non clean session the broker will remember client subscriptions and may hold undelivered messages for the client
- However this depends on the Quality of service used when subscribing to topics and the quality of service used when publishing topics

MQTT

Last will Messages

- The idea of the last will message is to notify a subscriber that the publisher is unavailable due to network outage
- The last will message is set by the publishing client on a topic
- The message is stored on the broker and sent to any subscribing client if the connection to the publisher fails
- If the publisher disconnects normally the last Will Message is not sent
- The will messages is including with the connect request

Understanding MQTT Topics



MQTT TOPICS

Introduction



- MQTT Topics are structured in a hierarchy similar to folders and files in a file system using the forward slash (/) as a delimiter
- Using this system you can create a user friendly and self descriptive naming structures of your own choosing
- Topic names are:
 - Case sensitive
 - use UTF-8 strings
 - Must consist of at least one character to be valid

MQTT TOPICS

Introduction

- Except for the \$SYS topic there is no default or standard topic structure
- That is there are no topics created on a broker by default, except for the \$SYS topic
- All topics are created by a subscribing or publishing client, and they are not permanent
- A topic only exists if a client has subscribed to it, or a broker has a retained or last will messages stored for that topic

MQTT TOPICS

The \$SYS topic

- This is a reserved topic and is used by most MQTT brokers to publish information about the broker
- They are read-only topics for the MQTT clients. There is no standard for this topic structure

MQTT TOPICS

Subscribing to Topics

- A client can subscribe to individual or multiple topics
- When subscribing to multiple topics two wildcard characters can be used
 - # (hash character) - multi level wildcard
 - + (plus character) -single level wildcard
- Wildcards can only be used to denote a level or multi-levels
 - i.e /house/# and not as part of the name to denote multiple characters
 - e.g. hou# is not valid

MQTT TOPICS

Topic naming Examples



- Valid Topic subscriptions: Single topic subscriptions

/

/house

house/room/main-light

House/room/side-ligh

Using Wildcards

Subscribing to topic house/#	Subscribing to topic house+/main-light
Valid: house/room1/main-light house/room1/alarm house/garage/main-light house/main-door	Valid: house/room1/main-light house/room2/main-light house/garage/main-light Invalid: house/room1/side-light house/room2/side-light

MQTT TOPICS

Publishing the Topics



- A client can only publish to an individual topic. That is, using wildcards when publishing is not allowed

- Example

To publish a message to two topics you need to publish the message twice

MQTT TOPICS

Creation: Topics

- Topics are created dynamically when,
 - Someone subscribes to a topic
 - Someone publishes a message to a topic with the retained message set to True

MQTT TOPICS

Removal: Topics

- Topics are removed when,

When the last client that is subscribing to that broker disconnects, and clean session is true

When a client connects with clean session set to True

MQTT TOPICS

Republishing: Topics

- Topics are republished when,

This is likely to be done when changing or combining naming schemes

The idea is that a client would subscribe to a topic

- Example: `hub1/sensor1` and republish the data using a new topic naming of `house1/main-light`

MQTT Publish and Subscribe

MQTT Publish and Subscribe

Introduction



- The process of sending messages is called publishing
- To receive messages an MQTT client must subscribe to an MQTT topic
- **Basics**
 - A client is free to publish on any topic it chooses.
 - Brokers can restrict access to topics
 - A client cannot publish a message to another client directly and doesn't know if any clients receive that message
 - A client can only publish messages to a single topic, and cannot publish to a group of topics
 - However a message can be received by a group of clients if they subscribe to the same topic

MQTT Publish and Subscribe

Message Flow and QOS on Published Messages

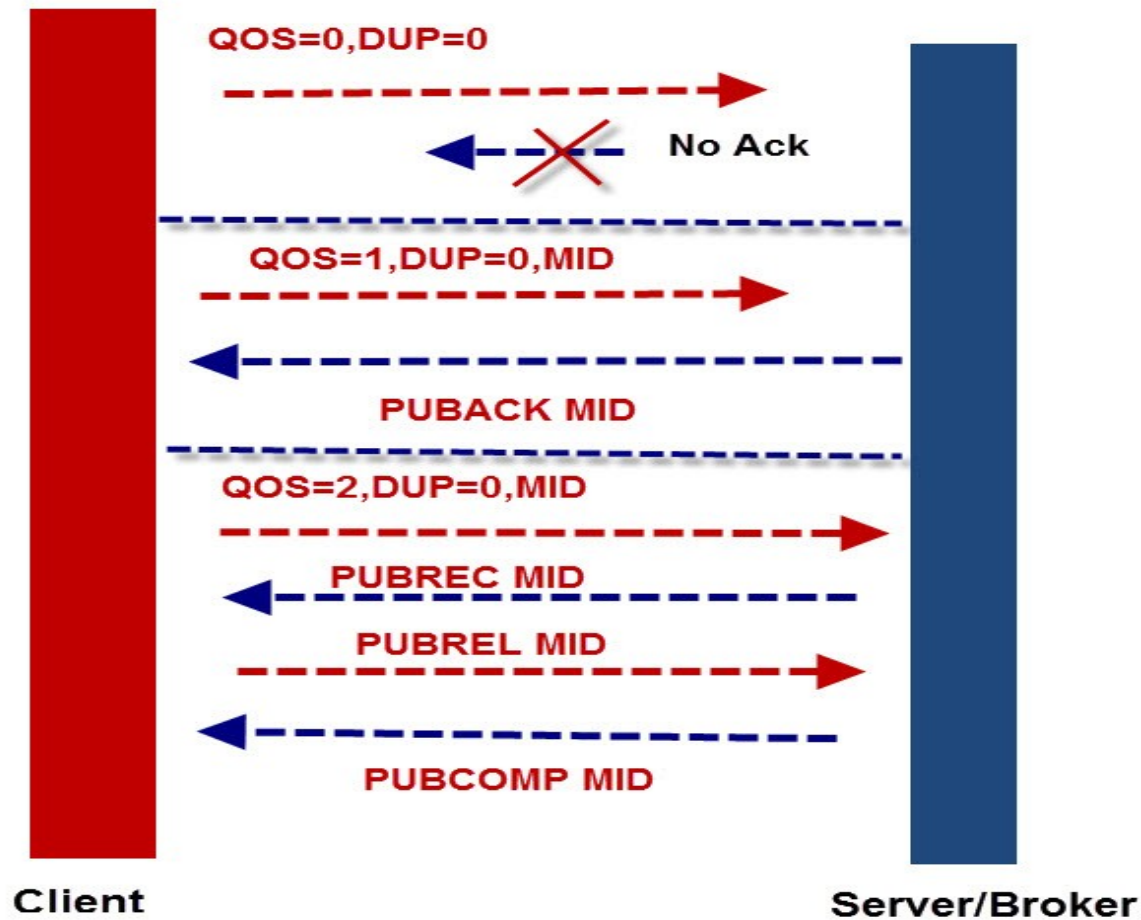


- MQTT supports 3 QOS levels 0,1,2
 - QOS -0: Default and doesn't guarantee message delivery.
 - QOS -1: Guarantees message delivery but could get duplicates.
 - QOS -2: Guarantees message delivery with no duplicates.
- A message is published using one of these levels with QOS level 0 being the default
- If you want to try and ensure that the subscriber gets a message even though they might not be online then you need to publish with a quality of service of 1 or 2

MQTT Publish and Subscribe

Message Flow and QOS on Published Messages

MQTT Message Publishing Message Flow



MQTT Publish and Subscribe

What Happens?



- What happens to the published message after the subscriber receives it?
- What happens to the published message if there are no subscribers?

MQTT Publish and Subscribe

Subscribing To Topics



- To receive messages on a topic you will need to subscribe to the topic or topics
- When you subscribe to a topics you also need to set the QOS of the topic subscription
- The QOS levels and their meaning are the same as those for the published messages
- When you subscribe to a topic or topics you are effectively telling the broker to send you messages on that topic
- To send messages to a client the broker uses the same publish mechanism as used by the client
- You can subscribe to multiple topics using two wildcard characters (+ and #)
- All subscriptions are acknowledged by the broker using a subscription acknowledge message that includes a packet identifier that can be used to verify the subscription

MQTT: Clients



MQTT Clients

Introduction

- Because MQTT clients don't have addresses like email addresses, phone numbers etc. you don't need to assign addresses to clients like you do with most messaging systems

MQTT: Brokers / Servers



MQTT Brokers

Introduction



- There are many MQTT brokers available that you can use for testing and for real applications
- There are free self hosted brokers , the most popular being Mosquitto and commercial ones like HiveMQ
- If you don't want to install and manage your own broker you can use a cloud based broker from Cloud service providers like IBM, Microsoft (Azure) etc
- Eclipse has a free public MQTT broker and COAP server that you can also use for testing. The address is iot.eclipse.org and the port is 1883 or 8883(SSL)

MQTT: Security



MQTT Security

Introduction

- MQTT supports various authentications and data security mechanisms
- It is important to note that these security mechanisms are configured on the MQTT broker, and it is up to the client to comply with the mechanisms in place

THANK YOU