Internet of Things and Services

Service-oriented architectures

MQTT protocol & brokers

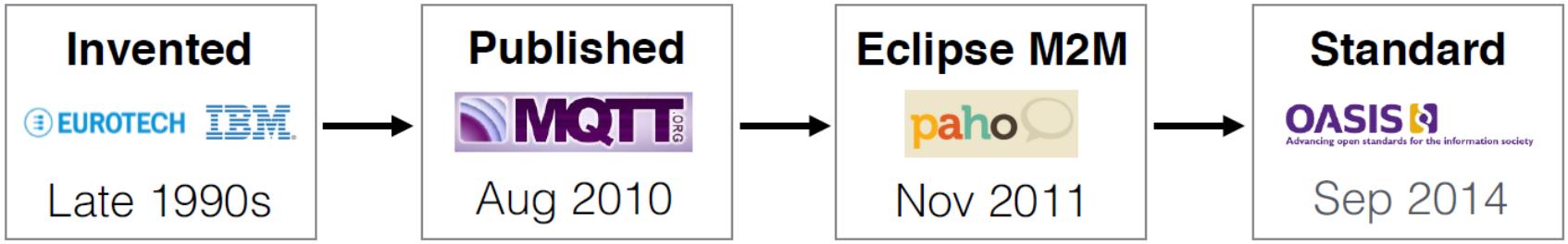
Department of Computer Science
Faculty of Electronic Engineering, University of Nis

MQTT Introduction

- Message Queuing Telemetry Transport
- Created by Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech) in 1999
- Available under a royalty free license as protocol version 3.1 since 2010
- Lightweight publish/subscribe machine-to-machine protocol on top of TCP/IP
- Near real-time communication between clients through a message broker
- Small source code footprint for embedded devices
- Protocol versions 3.1, 3.1.1 and 5.0 (released in 2018)
- MQTT-SN (MQTT for Sensor Networks) uses UDP

MQTT Introduction

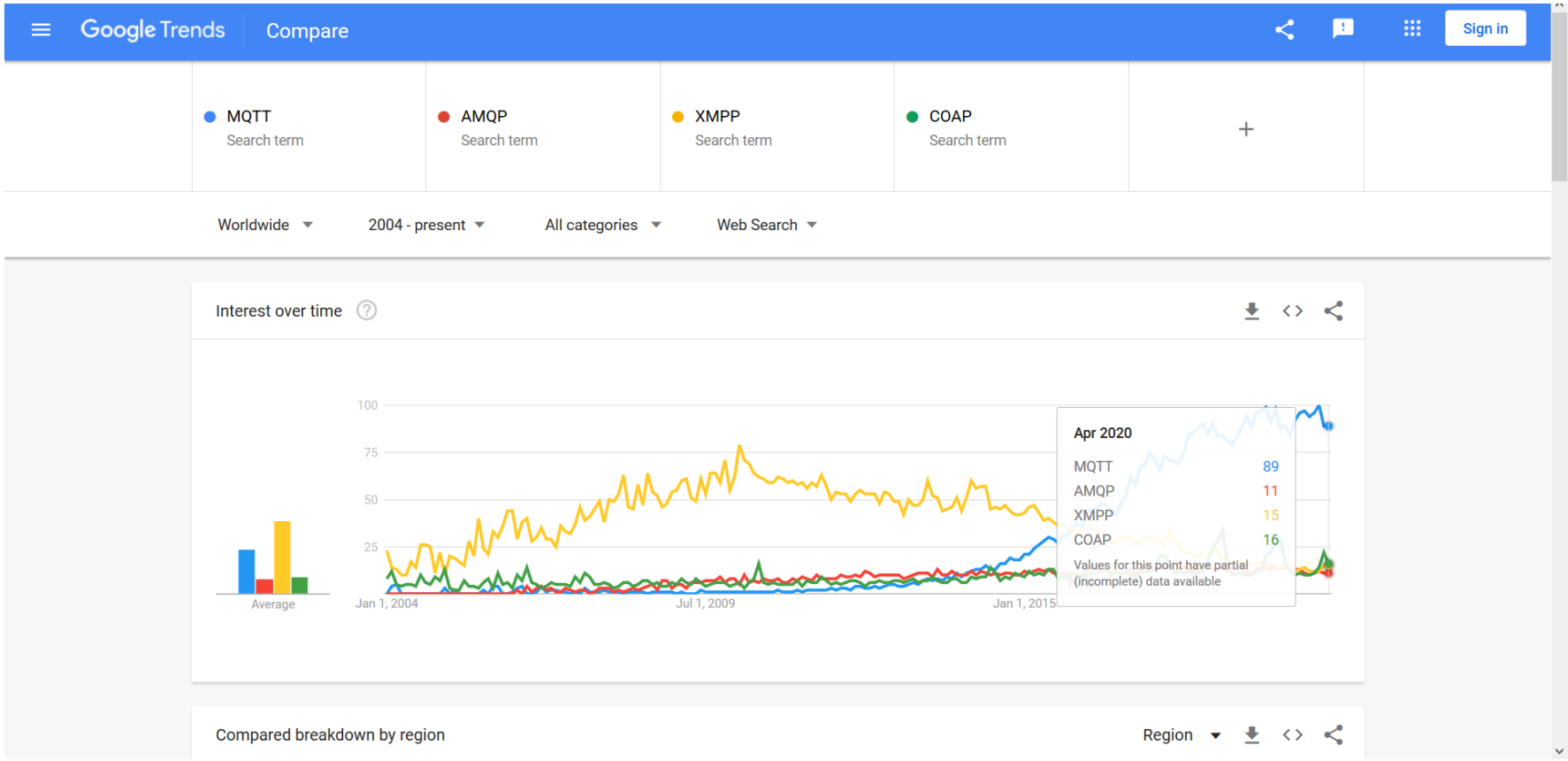
- OASIS standard (since 2014) and ISO standard (ISO/IEC 20922) since 2016
- A lightweight machine-to-machine (M2M)/IoT connectivity protocol for messaging
  - **open** - open spec, standard 40+ client implementations
  - **lightweight** - minimal overhead efficient format tiny clients (kb)
  - **reliable** - QoS for reliability on unreliable networks
  - **simple** - 43-page spec connect + publish + subscribe



MQTT features

- Designed for minimal network traffic and constrained devices
- Small header size
  - PUBLISH 2-4 bytes, CONNECT 14 bytes
  - HTTP 0.1-1 KB
- Binary payload (not text)
- Small clients: 30 KB (C), 100 KB (Java)
- Minimal protocol exchanges
  - MQTT has configurable keep alive (2 byte PINGREQ / PINGRES)
- Efficient for battery life

IoT protocol trends



MQTT Operations

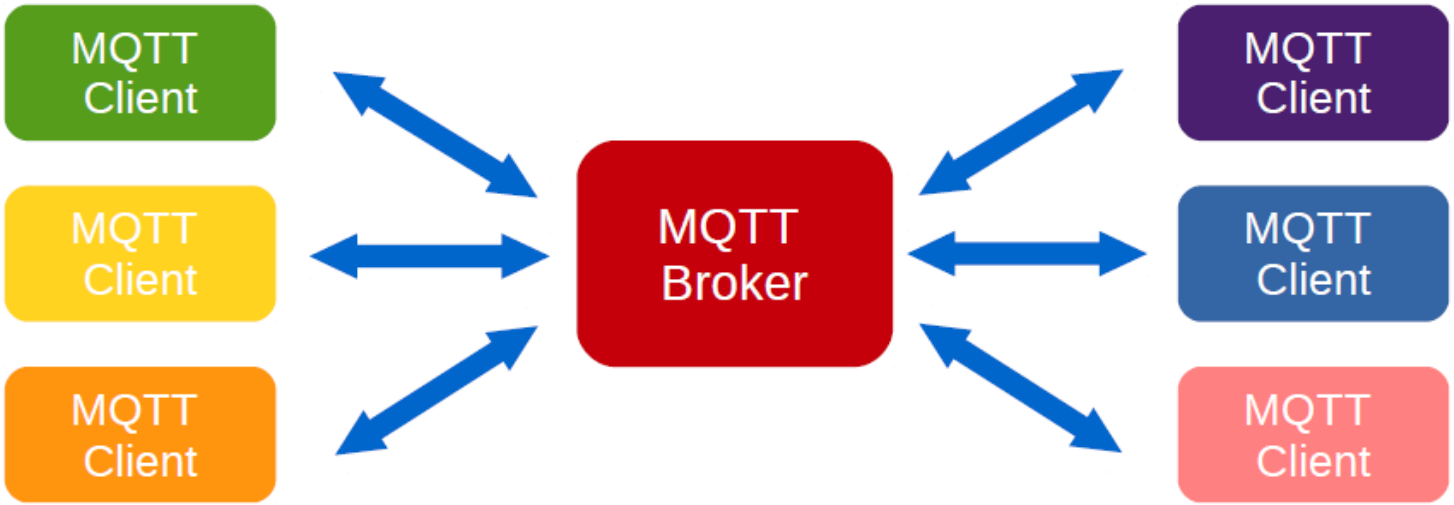
- Connect
- Disconnect
- Subscribe
- Unsubscribe
- Publish

```
client = new Messaging.Client(hostname, port, clientId)
client.onMessageArrived = messageArrived;
client.onConnectionLost = connectionLost;
client.connect({ onSuccess: connectionSuccess });

function connectionSuccess() {
     client.subscribe("planets/earth");
     var msg = new Messaging.Message("Hello world!");
     msg.destinationName = "planets/earth";
     client.publish(msg);
}

function messageArrived(msg) {
     console.log(msg.payloadString);
     client.unsubscribe("planets/earth");
     client.disconnect();
}
```

Eclipse Paho JavaScript MQTT client

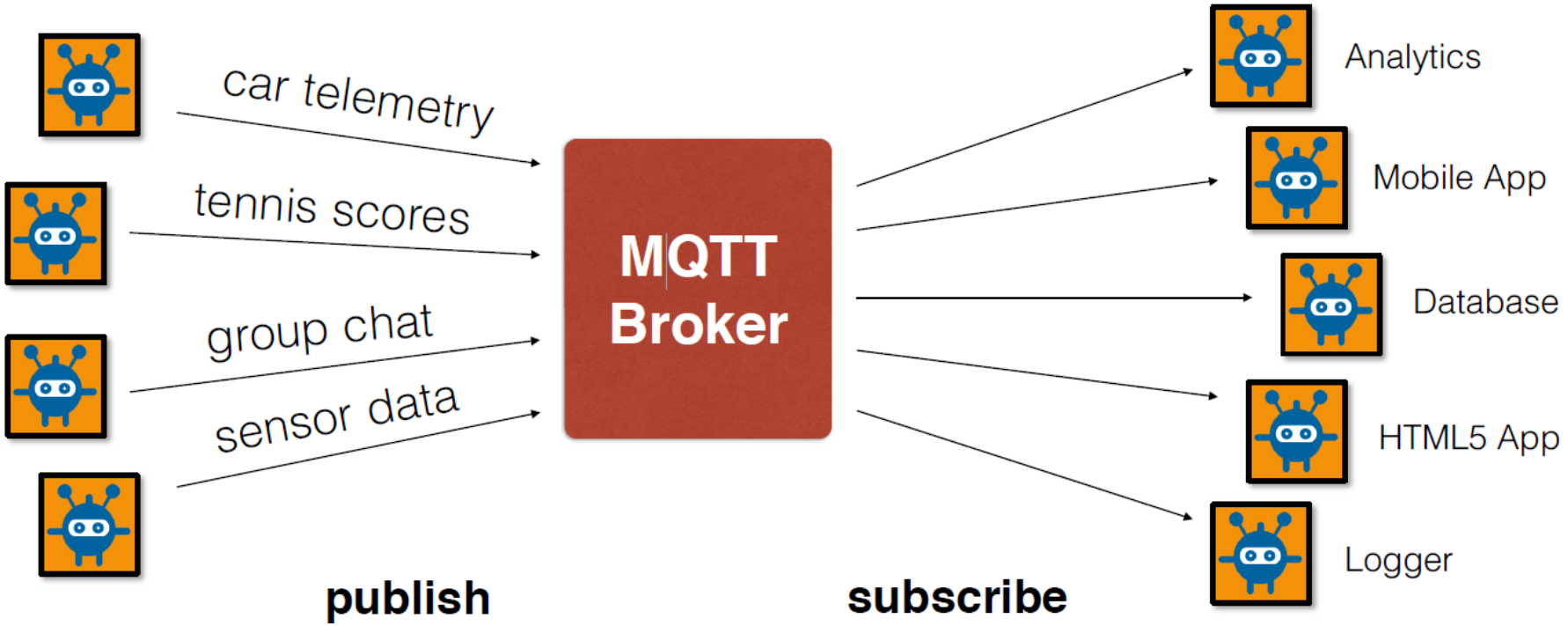
MQTT Broker

- Connect and handle multiple clients
- Deliver published messages to the subscribed MQTT clients depending on the topic of the message



Publish / subscribe

- Pub/sub decouples senders from receivers



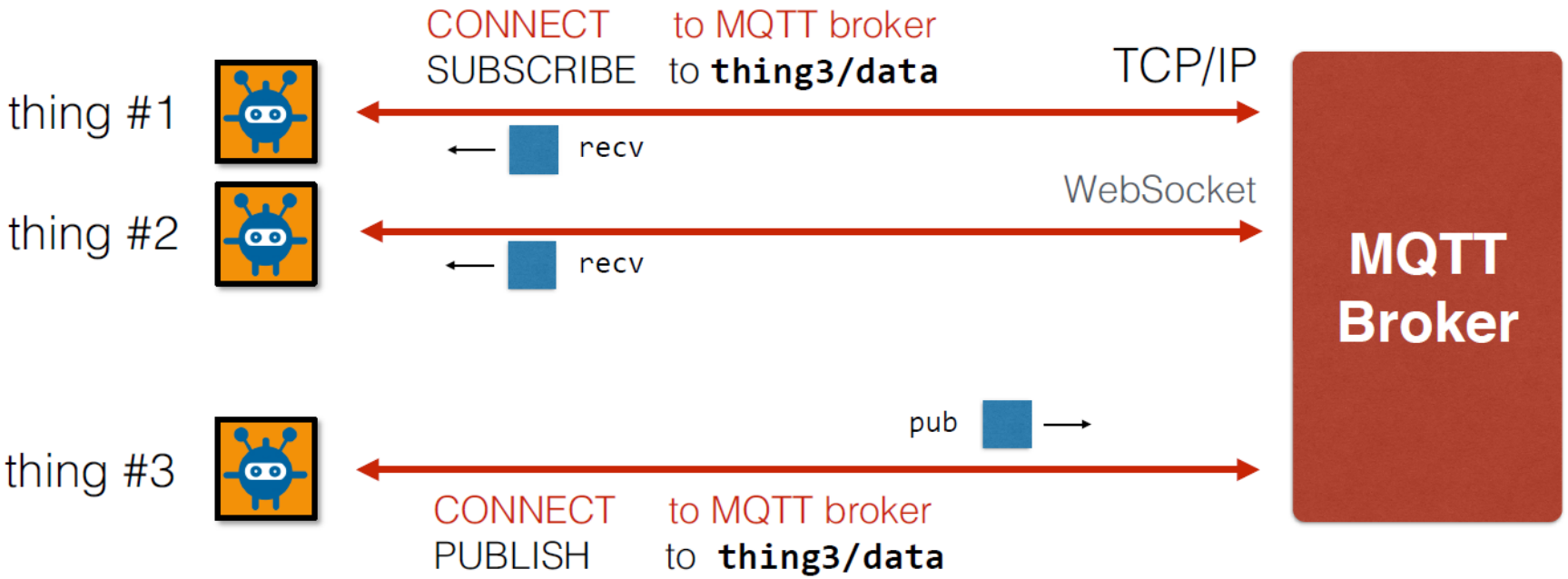
MQTT Message

- Topic
- Payload (text or binary)
- Quality of service
  - 0 - at most once (no delivery guarantee)
  - 1 - at least once
  - 2 - exactly once
- Retain (true or false)

Example message	
Topic	"hello/1"
Payload	{ "temperature": 20 }
QoS	2
Retain	false

MQTT Communication

- Bi-directional, async “push” communication

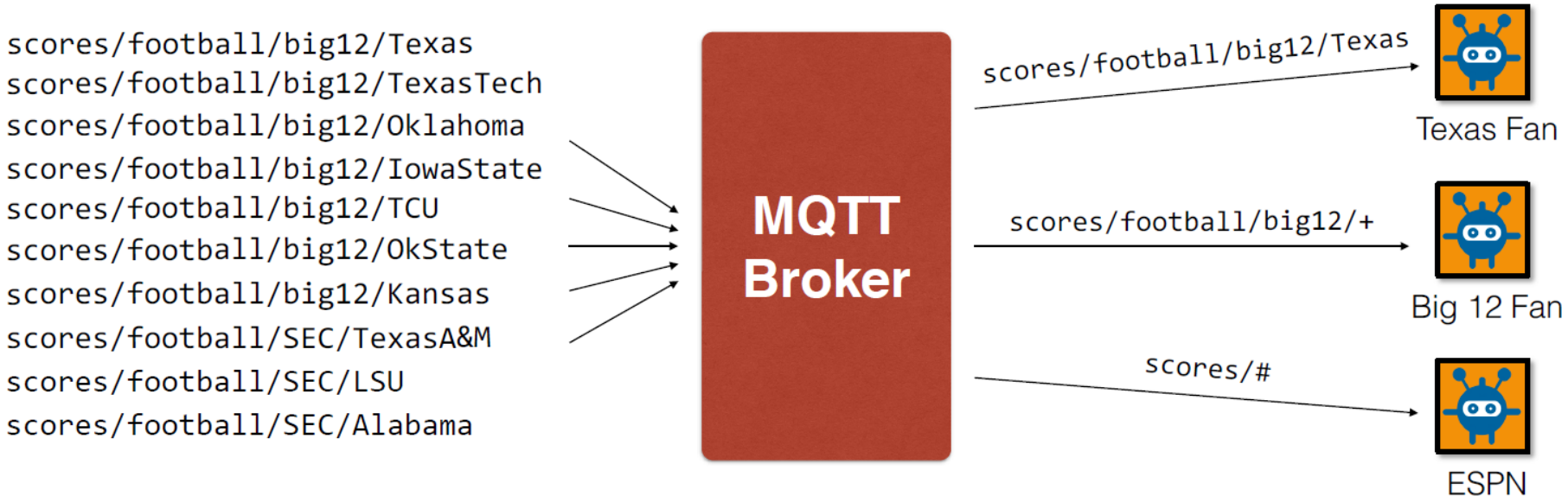


MQTT Topics & Wildcards

- Topic
  - home / bedroom / temperature
- Single level wildcards
  - home / + / temperature
- Multiple levels wildcards
  - home / #

MQTT subscription

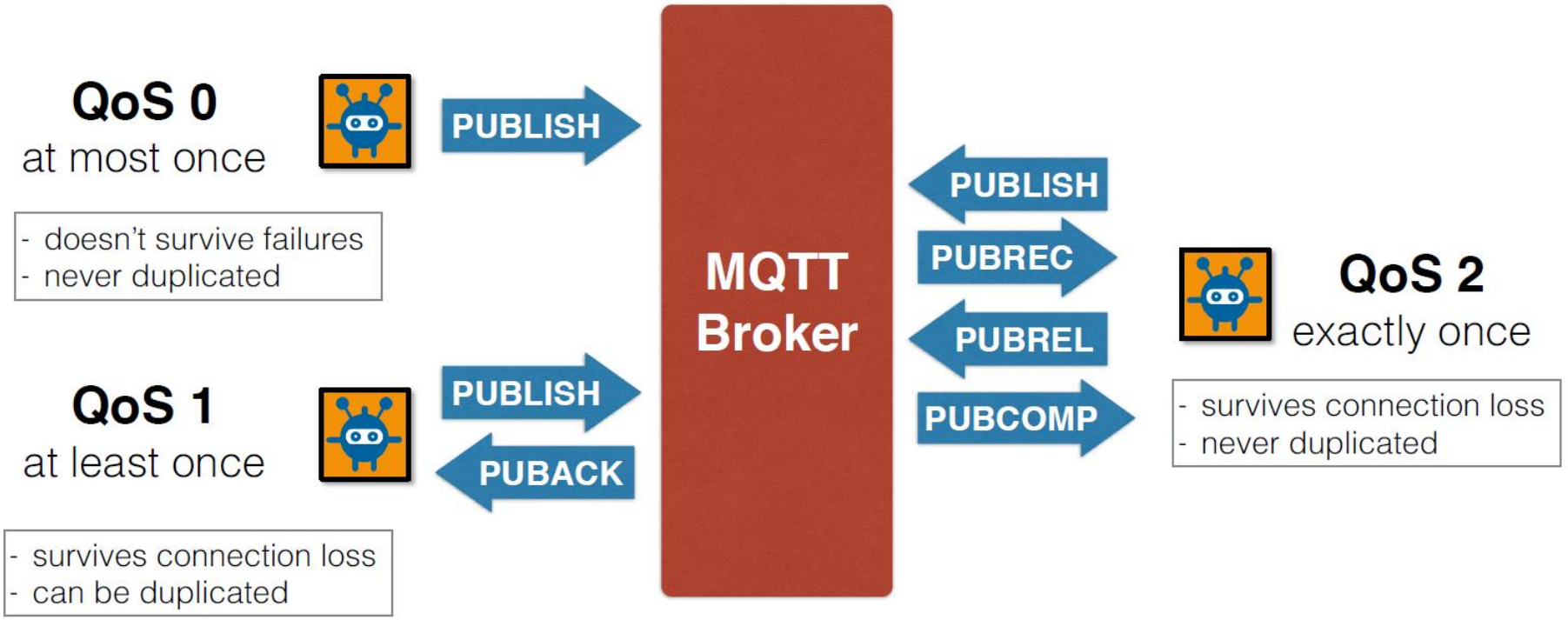
- Allows wildcard subscriptions



single level wildcard: **+** multi-level wildcard: **#**

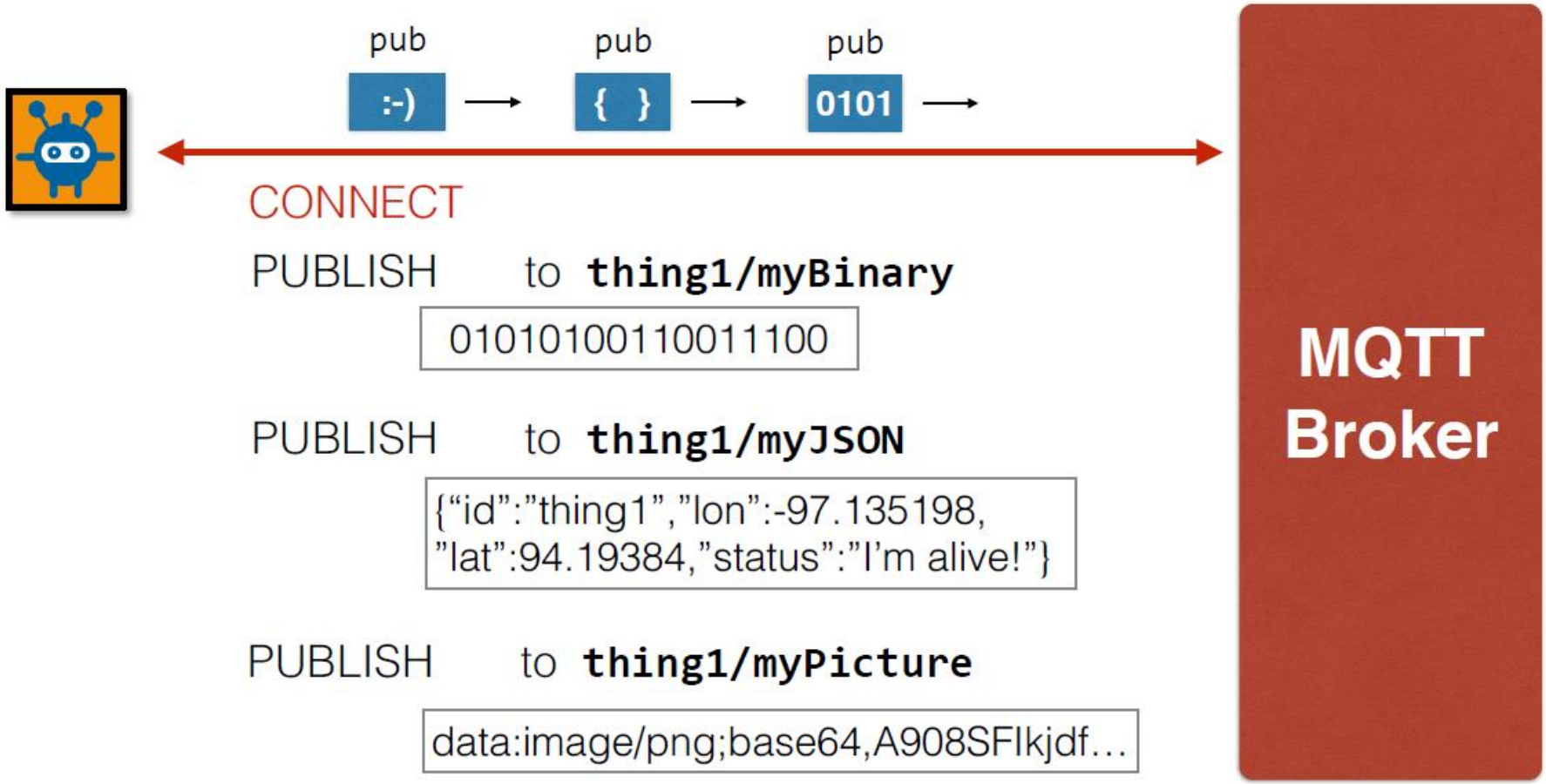
MQTT Quality of Service

- Quality of Service for reliable messaging



MQTT Payload

- Agnostic payload for flexible delivery

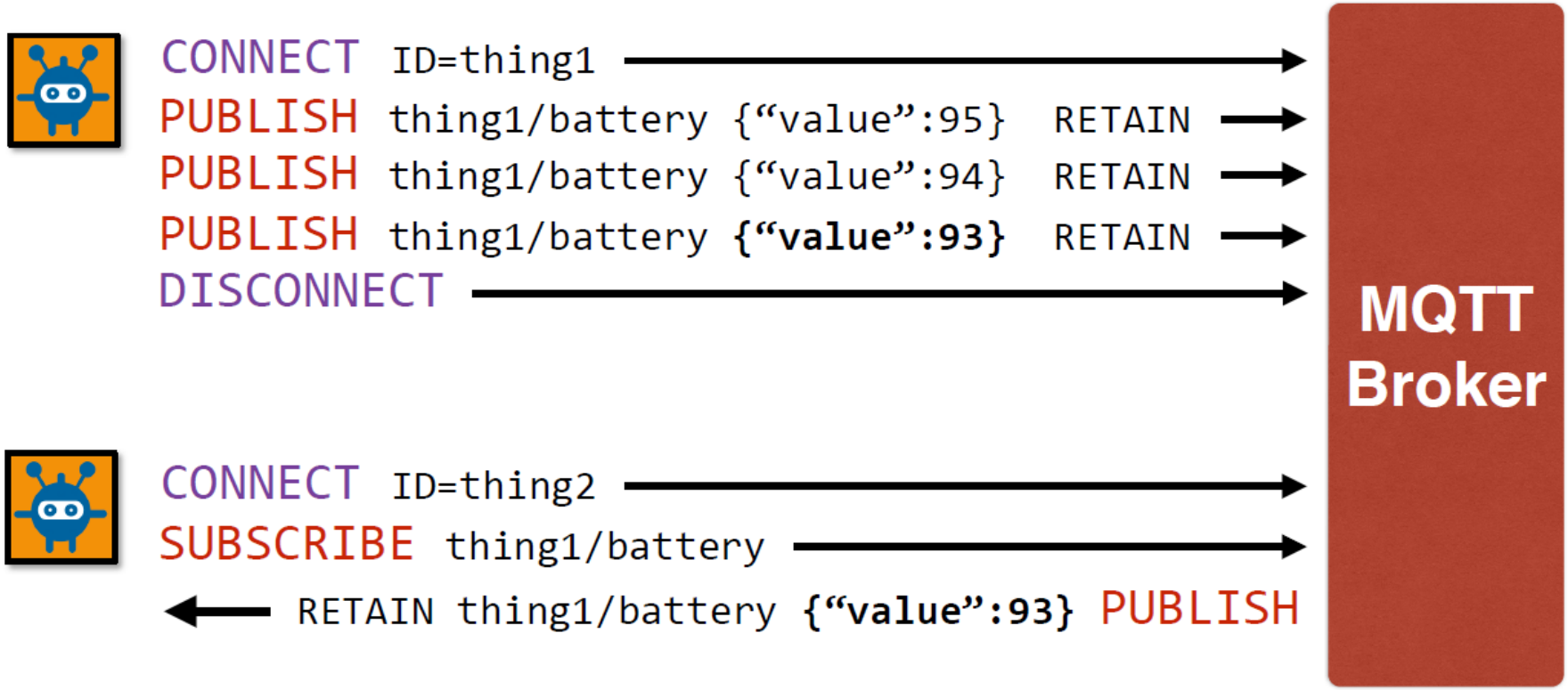


MQTT Retained messages

- Message published with retain fag set to true that is used to store the last know good value
- The MQTT broker is responsible for transmitting the retained message to all newly-subscribed for this topic MQTT clients
- To delete a retained message publish another message with the same topic and an empty payload

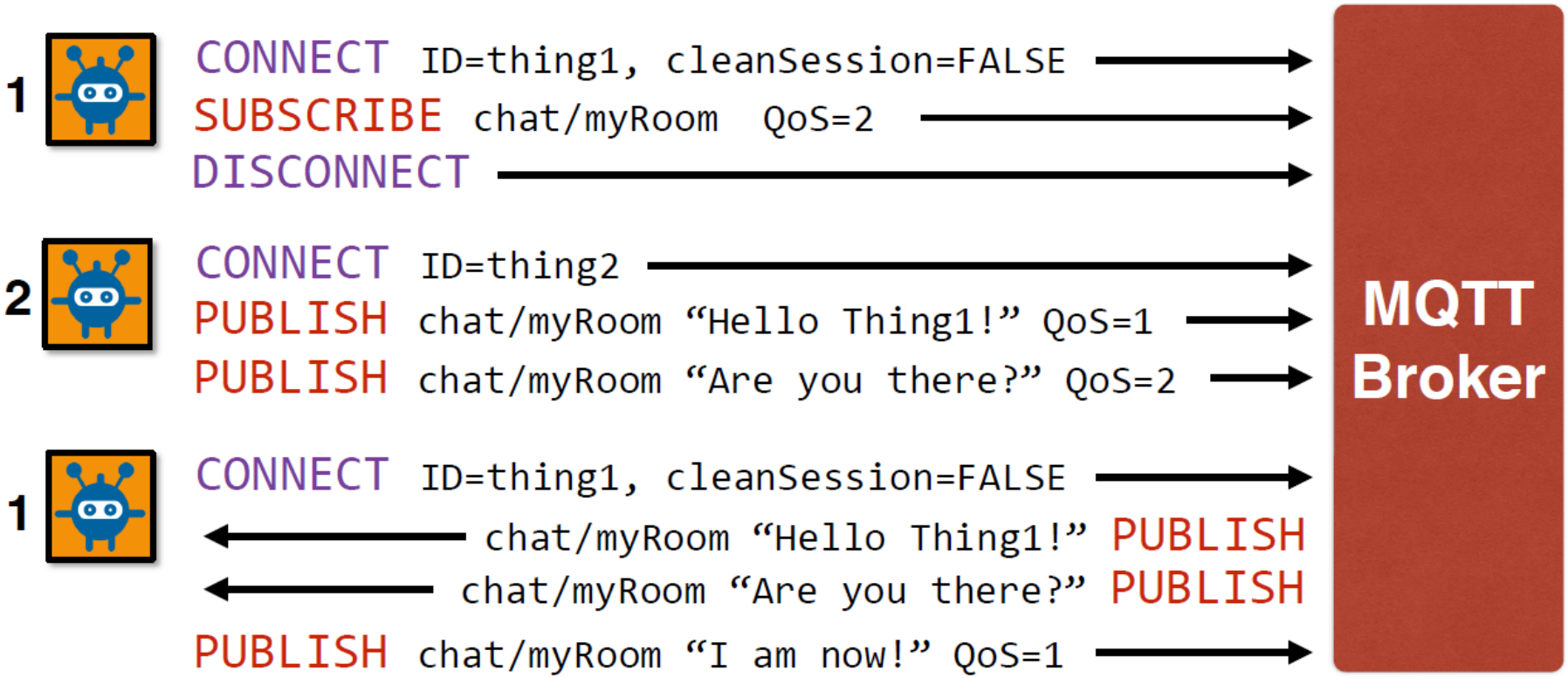
MQTT

- Retained messages for last value caching



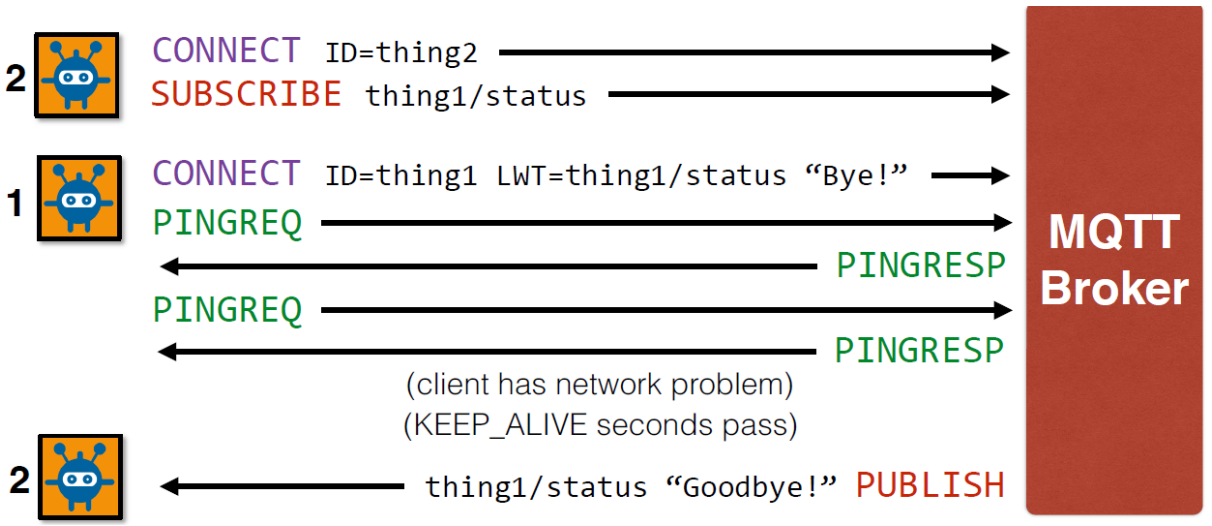
MQTT

- Client id and *cleanSession* for session state



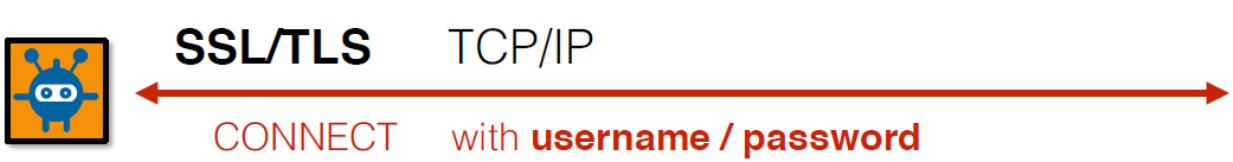
MQTT LWT

- Last Will & Testament (LWT) allows the broker to notify interested clients about an ungracefully disconnected client by publishing a message on his behalf
- The MQTT client should register the will message when connecting to the MQTT broker
- The minimum requirement for a will message is to specify at least a topic



MQTT security

- Transport encryption with TLS/SSL
- Authentication: username/password
- Authorization: Access Control Lists (ACL)



- MQTT spec doesn't define security model aside from username/password authorization on connection
- Brokers *can* implement support for SSL/TLS and policies for connection and messaging

ex. organize topic space by "group"
username associated with a group

bboyd is in group "IBM" and can pub/sub `IBM/bboyd/#`

MQTT Broker

MQTT BROKERS

Mosquitto

- Free and open source MQTT broker written in the C programming language
- Supports MQTT protocol version 3.1 and 3.1.1
- Supports QoS 0, 1 and 2
- Supports Web sockets
- Available for Windows, FreeBSD, Mac OS and GNU/Linux distributions
- Also provides simple command line MQTT clients called mosquitto_pub and mosquitto_sub
- https://mosquitto.org/

Mosca

- Free and open source MQTT broker written in JavaScript
- Can be used standalone or embedded in another Node.js application
- Supports MQTT protocol version 3.1 and 3.1.1
- Supports Web sockets
- Supports QoS 0 and 1
- Available for all platforms on which you can run Node.js: MS Windows, Mac OS and GNU/Linux distributions
- https://github.com/moscajs/mosca
- http://www.mosca.io/
- Aedes (improved version) https://github.com/moscajs/aedes

EMQ (emqttd, emqx)

- Free and open source MQTT broker written in Erlang/OTP
- Supports MQTT protocol version 3.1, 3.1.1 and 5.0
- Supports QoS 0, 1 and 2
- Supports Web sockets, MQTT-SN, CoAP, STOMP and SockJS
- Available for Windows, FreeBSD, Mac OS and GNU/Linux distributions
- https://emqtt.io/
- https://www.emqx.io/

VerneMQ

- Free and open source MQTT broker written in Erlang/OTP
- Supports MQTT protocol version 3.1, 3.1.1 and 5.0
- Supports QoS 0, 1 and 2
- Supports Web sockets
- Available for GNU/Linux distributions and Mac OS
- **Not** working on Windows due to the LevelDB code
- https://vernemq.com/

Apache ActiveMQ

- Free and open source message broker written in JAVA
- Supports large number of transport protocols: MQTT, OpenWire, STOMP, AMQP & others
- Supports MQTT protocol version 3.1
- Supports QoS 0, 1 and 2
- Supports Web sockets
- Available for GNU/Linux distributions, UNIX compatible systems and Windows
- https://activemq.apache.org/

RabbitMQ

- Free and open source message broker written in Erlang
- Supports large number of transport protocols: MQTT, STOMP, AMQP, HTTP
- Supports MQTT protocol version 3.1 via a plugin
- Supports QoS 0 and 1
- Supports Web sockets
- Available for GNU/Linux distributions, BSD & UNIX compatible systems, Mac OS and MS Windows
- https://www.rabbitmq.com/

HiveMQ

- MQTT broker implement in the Java programming language
- Supports MQTT protocol version 3.1, 3.1.1 and 5
- Supports Web sockets
- Commercial license, owned by dc-square GmbH, as well as Community Edition (open source)
- Lead developer Dominik Obermaier
- Open source plugins available at GitHub under Apache-2.0
- https://github.com/HiveMQ
- https://www.hivemq.com/

More open source MQTT brokers

- Apache ActiveMQ Artemis (written in JAVA, based on HornetQ)
  - https://activemq.apache.org/components/artemis/
- Moquette (JAVA)
  - https://github.com/moquette-io/moquette
- Vertx-mqtt-broker (written in JAVA with Vert.x)
  - https://vertx.io/docs/vertx-mqtt/java/
- MQTTnet (written in C#, the library is available as a nuget package https://www.nuget.org/packages/MQTTnet/)
  - https://github.com/chkr1011/MQTTnet
- mqttools (Python)
  - https://github.com/eerimoq/mqttools

Commercial MQTT brokers

- HiveMQ
  - https://www.hivemq.com/
- IBM IoT MessageSight
  - https://www.ibm.com/support/knowledgecenter/SSWMAJ_5.0.0
- Flespi
  - https://flespi.com/
- JoramMQ
  - http://www.scalagent.com/en/jorammq-33/products/overview
- PubSub+
  - https://solace.com/products/event-broker/
- CloudMQTT – Hosted message broker for IoT (cloud)
  - https://www.cloudmqtt.com/

References

- MQTT home
  - http://mqtt.org/
- OASIS MQTT Version 5.0
  - http://docs.oasis-open.org/mqtt/mqtt/v5.0/cs02/mqtt-v5.0-cs02.html
- Eclipse Paho MQTT clients
  - http://www.eclipse.org/paho/
- Comparison of MQTT implementations
  - https://en.wikipedia.org/wiki/Comparison_of_MQTT_implementations