Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Introduction To

Embedded Real-Time Systems Lab
Indian Institute of Technology Bombay

IIT Bombay
July 7, 2022

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Agenda for Discussion

1. Messaging Protocols for IoT

2. MQTT Overview

3. Architecture
   - Architecture Key Terms

4. Quality of Service (QoS)

5. Wildcards

6. Advanced Features
   - Retained Message
   - Birth/Death Message
   - LWT Message
   - API to be Used

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

# MQTT Overview

- Message Queuing Telemetry Transport
- Developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Cirrus Link) in 1999
- Designed for Machine-to-Machine communication - small code size and limited network bandwidth
- Publish-Subscribe model
- Works on top of TCP/IP

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Architecture Key Terms

EMQ MQTT Architecture

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Architecture Key Terms
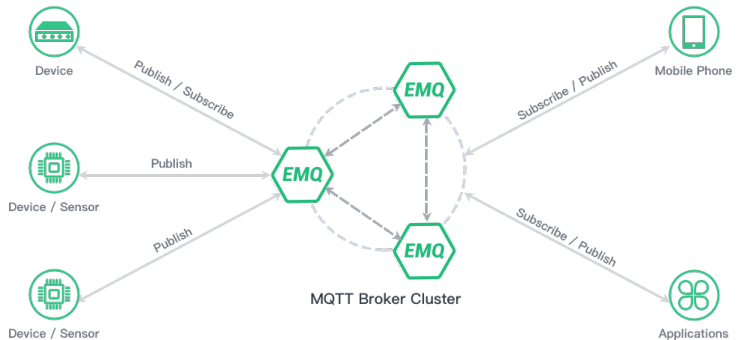
Architecture Key Terms

- ✅ **Broker**: Accepts messages from clients and then delivers them to any interested clients. (Sometimes brokers are called "servers.")
- ✅ **Client**: Thing which can connect to broker to send and receive required information. Unique ID called Client ID.
- ✅ **Topic**: Namespace (or place) for messages on the broker. Clients subscribe and publish to a topic.
- ✅ **Publish**: Client sending a message to the broker, using a topic name.
- ✅ **Subscribe**: Client tells the broker which topics it is interested. The broker sends messages published to that topic.

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Quality of Service (QoS)

- **QoS 0 (At most once)** - where messages are delivered according to the best efforts of the operating environment. Message loss can occur.
- **QoS 1 (At least once)** - where messages are assured to arrive but duplicates can occur.
- **QoS 2 (Exactly once)** - where message are assured to arrive exactly once. Safest but slowest mode.

  The higher the QoS, the lower the performance speed

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Wildcards

- ✅ Helps the developer to subscribe to multiple topics simultaneously
- ✅ Wildcard is only available for Subscription, not for Publishing
- ✅ Two wildcard character are supported:
    - '#': represents a complete sub-tree of the hierarchy.
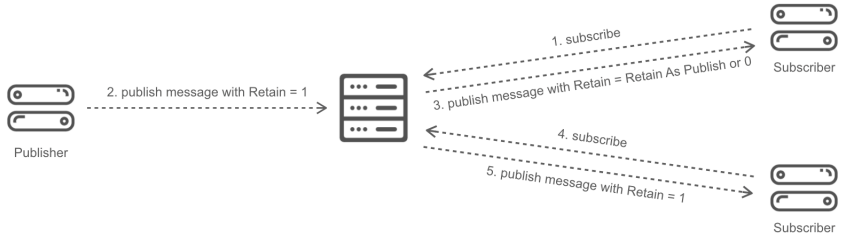   **Example Sensor/dht11/#**.
      This will match any topic starting with Sensor/dht11, such as
      Sensor/dht11/Temperature, Sensor/dht11/Humidity.
    - '+': represents a single level of the hierarchy.
      Example, Sensor/+/TEMP will match Sensor/dht11/Temperature
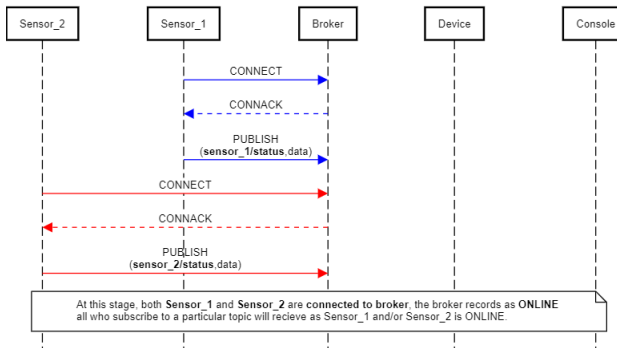   and Sensor/dht22/Temperature.
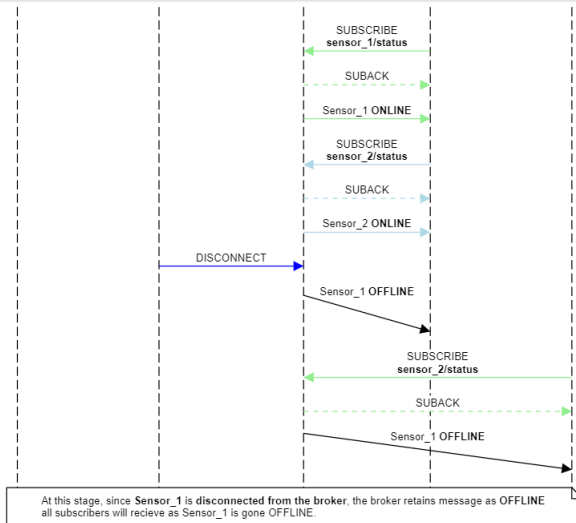
Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Retained Message
Birth/Death Message
LWT Message
API to be Used

Retained Message

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Retained Message
Birth/Death Message
LWT Message
API to be Used

Birth/Death Message

Birth/Death Messages

Outline
Messaging Protocols for IoT
MQTT Overview
Architecture
Quality of Service (QoS)
Wildcards
Advanced Features

Retained Message
Birth/Death Message
LWT Message
API to be Used

Continued..

At this stage, since **Sensor_1** is **disconnected from the broker**, the broker retains message as **OFFLINE** all subscribers will recieve as Sensor_1 is gone OFFLINE.
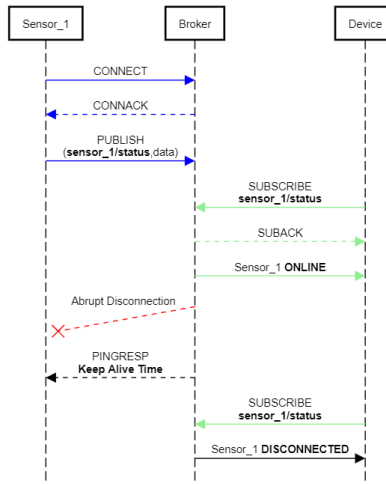
LWT Message

Last Will and Testament(LWT) Messages

API to be Used

void ey_mqtt_start(void);
int ey_mqtt_publish(char str_topic[], char str_data[], unsigned char qos);
int ey_mqtt_subscribe(char str_topic[], unsigned char qos);
int ey_mqtt_unsubscribe(char str_topic[]);