

MQTT (Message Queing Telemetry Transport)

- developed by IBM in 1999
- lightweight protocol
- Publish-Subscribe architecture (Pub-Sub)

Terminologies

- Client
 - client is endpoint (devices which are going to send and receive data)
 - clients are of two types
 - Publisher
 - going to generate data or to send data
 - Subscriber
 - going to consume data or to receive data
- Server
 - intermediate between multiple clients
 - server is also known as broker
- Topic
 - string or path which will be used to differentiate your messages
 - to create multilevel topics strings are separated by '/'
 - eg Home automation
 - home/room1/temperature
 - home/room2/temperature
 - home/room3/temperature
 - home/room4/temperature
 - home/room1/fan
 - home/room2/fan
 - home/room3/fan
 - home/room4/fan
 - home/+ /temperature -- 4 topics of temperature
 - home/+ /+ -- all topics of temperature and fan
 - home/+ -- not valid
 - '+' - single
 - home/# -- all topics of temperature and fan

- '#' - all remaining and must be last in hierarchy
- wild card characters can be only used for subscriptions
- level1/level2/level3/level4
- level1+/level3/level4
- level1+/+/level4
- level1+/+/+
- +/+/+/+
- level1/#
- level1/#/level4 -- not valid
- level1+/level4 -- not valid

MQTT methods

- connect() - this method is used to connect with broker
- disconnect() - this method is used to disconnect from broker
- publish() - this method is used to publish data on some topic
- subscribe() - this method is used to subscribe the topic

Message

- message structure is lightweight
- depending on message type, structure is going to change
- message consists of 3 parts
 - fixed header (2 bytes)
 - variable header (optional)
 - variable payload (optional)
- Message format types
 - fixed header
 - fixed header + variable header
 - fixed header + variable header + payload
- Message types
 - CONNECT
 - DISCONNECT
 - PUBLISH
 - SUBSCRIBE
 - CONNACK, DISCONNECTACK, PUBACK, SUBACK

- Header
 - MQTT version
 - is_to_retain
 - 0 - message is not retained on broker
 - 1 - message is retained on broker
 - Quality of Service (QoS) (integer)
 - message deliver guarantee is decided by QoS level
 - there are 3 QoS levels
 - 0 - only once with no ACK
 - 1 - at least once with ACK
 - 2 - only once with 4 handshake ACK mechanism
 - by default QoS level is 1
- by default broker retains last message send on topic

Installation

- To install MQTT broker (mosquitto - Eclipse)

```
sudo apt-get install mosquitto
```

- To install MQTT clients

```
sudo apt-get install mosquitto-clients
```

- To start mosquitto as service in background

```
sudo systemctl start mosquitto
```

- Options of mosquitto
 - -c : to do configurations
 - -v : to see debug/log messages
- Options of mosquitto_pub
 - -d : to see debug messages
 - -h : to give host
 - -p : to give port
 - -u : to give username
 - -P : to give password
 - -q : to give QoS level
 - -r : to give message to be retained or not
 - -t : to specify topic name

- -m : to specify message to be published
- Options of mosquitto_sub
 - -d : to see debug messages
 - -h : to give host
 - -p : to give port
 - -u : to give username
 - -P : to give password
 - -q : to give QoS level
 - -t : to specify topic name
- To publish message in MQTT

```
mosquitto_pub -h localhost -p 1883 -t "topic/hello" -m "Hello MQTT"
mosquitto_pub -h localhost -p 1883 -t "topic/hello" -m "Hello MQTT" -d
```

- To subscribe for topic in MQTT

```
mosquitto_sub -h localhost -p 1883 -t "topic/hello"
mosquitto_sub -h localhost -p 1883 -t "topic/hello" -d
```

- To install MQTT module(library) in python

```
sudo pip3 install paho-mqtt
```

paho-mqtt

- To create client mqttc = Client()
- Methods
 - connect()
 - host
 - port=1883
 - keepalive=60
 - bind_address=""
 - disconnect()
 - loop()
 - loop_start()/loop_stop()
 - loop_forever()
 - publish()
 - topic
 - payload=None
 - qos=0
 - retain=False

- subscribe()
 - topic
 - qos=0
- callback function
 - on_connect()
 - on_connect(client, userdata, flags, rc)
 - rc = 0 : Connection successful
 - on_publish()
 - on_publish(client, userdata, mid)
 - on_message()
 - on_message(client, userdata, message)
 - message
 - an instance of MQTTMessage.
 - This is a class with members topic, payload, qos, retain.