MQTT
Message Queuing Telemetry Transport

Pete Keefe

April 2019

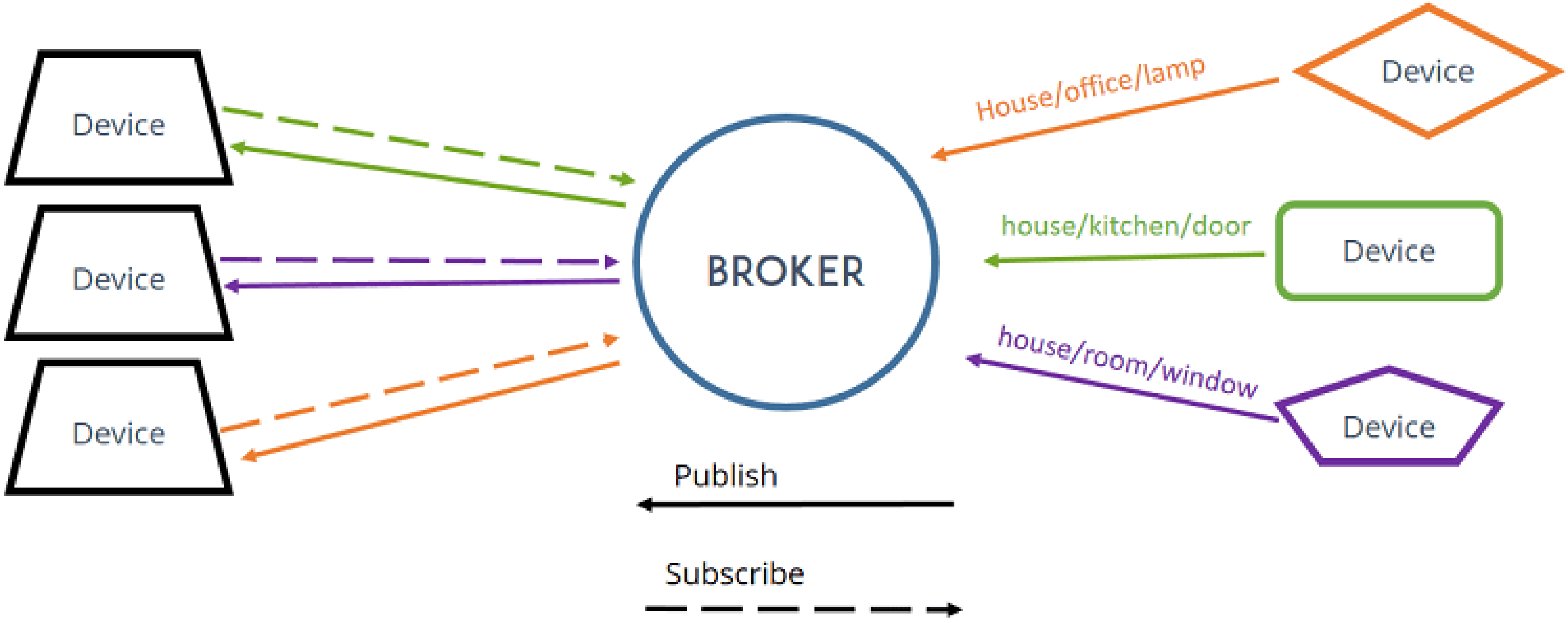
MQTT

- Open standard for IoT communications between devices and controllers
  - No formal standards organization for IoT traffic thus many different messaging types in use
- Originally designed by IBM in 1999 for it's MQ Series message queuing product line
  - Node-Red – MQTT imbedded into (Node-Red was also originally IBM)
  - Now an open standard
- Publish-subscribe-based messaging protocol
  - Publisher does not know subscribers to a message
  - Subscriber(s) do not know publisher of a message
- Small foot-print (not much code or overhead)

Components

- Broker - Software running on “server” that receives messages from publishers, stores messages, sends messages to subscribers
  - Can also be configured to exchange messages with other brokers – i.e., home broker to cloud broker
  - Options access rules are possible
- Clients – Software that starts and maintains connection to broker, might be a:
  - Publisher(s): sends messages to broker along
  - Subscriber(s): receives requested messages from broker

Message Flow



MQTT Message Parts

- Message
  - Topic
  - Payload
    - simple data
    - JSON object

    { "employee":{ "name":"John", "age":30, "city":"New York" } }

    { "employees":["John", "Anna", "Peter"] }
  - Retain sent by publisher – true / false
  - Quality of Service (QOS) – 0,1,2

Message Topic

- Topic names are:
  - Case sensitive
  - use UTF-8 strings
  - Must consist of at least one character to be valid
- i.e.,
  - home/sensor1/temp/33
  - home/sensor1/light/0
  - $sys/ (prefix reserved for broker status)
- Wildcard subscriptions
  - $sys/#
  - home/# (# = match all with home/ prefix)
  - home/sensor1/*/* (* = match to all home/sensor1)
- Brokers can optionally change topic if to/from another broker

Retain Flag

- Broker will only keeps (retains) the last message with the same topic
- If broker receives another message with same topic but no payload, it removes the retained message
- Allows for publisher to send message to subscriber that hasn’t started yet

Quality of Service (QOS)

=0 At most once - the message is sent only once and the client and broker take no additional steps to acknowledge delivery (fire and forget).

=1 At least once - the message is re-tried by the sender multiple times until acknowledgement is received (acknowledged delivery).

=2 Exactly once - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received (assured delivery)

Last Will & Testament

- When Clients connect to broker they can optionally send an LWT which is a standard format message to be published on unexpected loss of the client's keep alive signal.
- LWT message will normally have a Retain = true

Installing on Raspberry Pi

- Installing mosquitto broker and test clients

  ```
  Sudo apt-get update
  sudo apt –get install -y mosquitto mosquitto-clients
  ```

- Set broker to start automatically

  ```
  sudo systemctl enable mosquitto.service
  sudo systemctl start mosquitto.service
  ```

- If you want to write Python clients

  pip install paho-mqtt **-or-** pip3 install paho-mqtt

- Default mosquitto configuration
  - File: /etc/mosquitto/mosquitto.conf
  - Broker on port 1883
  - Allows anonymous clients

Demos

- Using mosquitto_pub & mosquitto_sub
- Node-Red on Raspberry Pi publishes message(s) to Node-Red on another Raspberry Pi
- Python mqtt client
  - Publish message
  - Subscribe to messages
- ESP8266 with ESPEasy for controlling lights