UNIVERSIDAD
COMPLUTENSE
MADRID

MQTT

NP2

MQTT

- MQTT (Message Queue Telemetry Transport) M2M communication protocol
- Developed by Dr. Andy Stanford-Clark (IBM) and Arlen Nipper (Arcom) in 1999
- Currently open standard maintained by OASIS group:
  - http://docs.oasis-open.org/mqtt/mqtt/
  - ISO standard (ISO/IEC PRF 20922)
- Public and free license
- Used by Amazon Web Services, IBM WebSphere MQ, Microsoft Azure IoT, Adafruit, Facebook Messenger...

Features

- Small code footprint
  - Ideal for memory-constrained systems
- Extremely lightweight
  - Ideal for scenarios with constraints in bandwidth
- Publish/subscribe model
- Works over **TCP**/IP
  - Usually port 1883
  - **MQTT-SN:** Variants on non-TCP/IP networks, such as Zigbee or UDP-based
  - Possible over *websocket*
- Quality of Service support
  - *(0) At most once* (no ACK*)*
  - *(1) At least once (*ACK received)
  - *(2) Exactly once (*4-way *handshake)*
- Many existing libraries/implementations
- Security: authentication using user/password
  - Optional cypher support (SSL/TLS)
- Persistence: MQTT supports messages stored in the *broker*

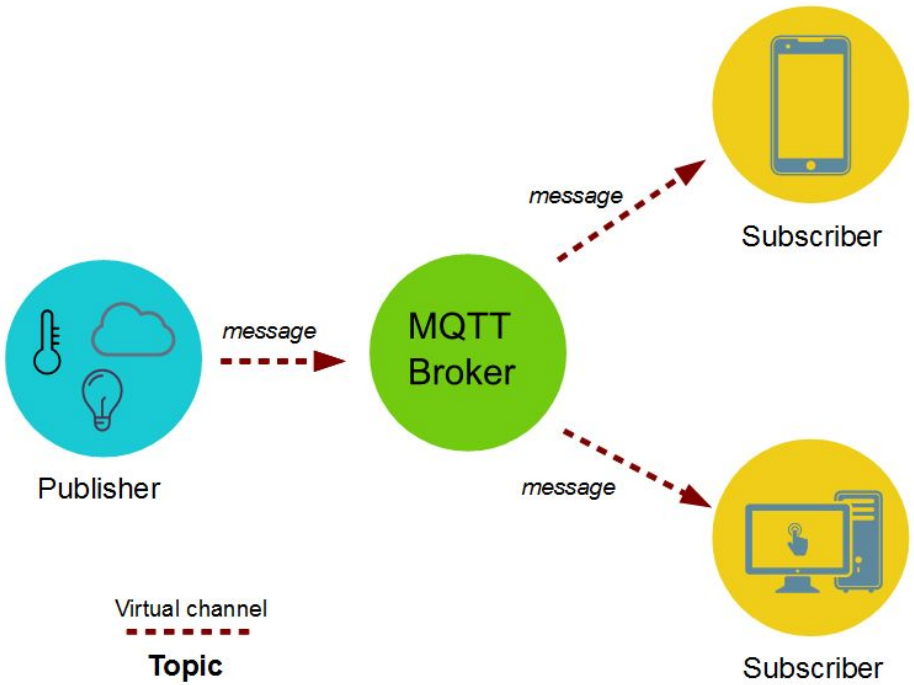
Applications

- Domotics
- Health
- Phone apps
- Industrial automation
- Automotive
- ….

Publish / Subscribe

- Scenario with hundreds/thousands/… devices sensing data
  - Who is the destination of the information?
  - Each node must send to all potential information consumers?
- Paradigm publisher/subscriber
  - Publishers send their information through a given channel
  - All possible consumers are subscribed to one or more channels

Publish / Subscribe



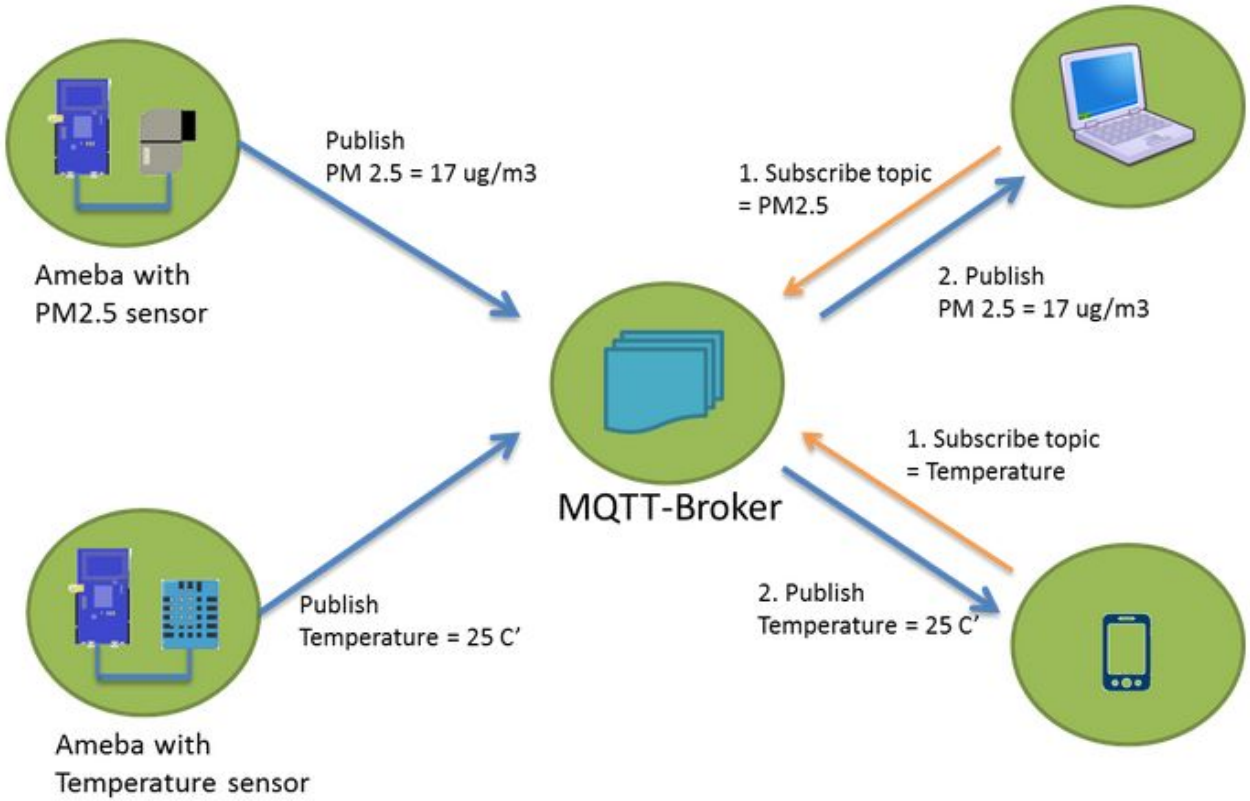
Source: https://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html

- Two types of clients: publish message and subscribe to messages
- Communicated via a *broker*
- Asynchronous: client is not blocked while waiting for a message
- Does not require that producer and subscriber are simultaneously active

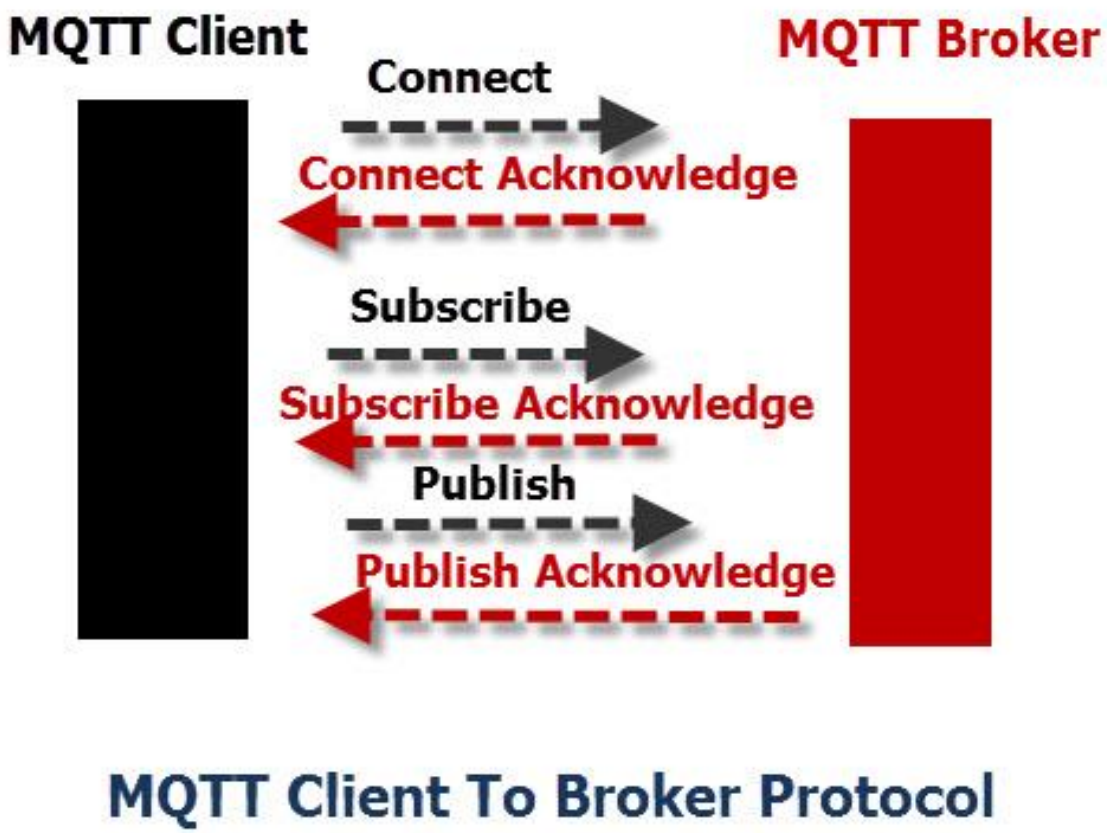
Brokers and *topics*

- The role of the *broker* is to receive messages from producers and submit them to the interested (subscribed) subscriptors
- Uses the *topic* to filter which clients will receive the message
  - The *topic* is a string of characters
  - A hierarchy of *topics* is defined using the character "/" as a separator
    - Ex: `sensors/NAME_NODE/temperature/NAME_PART`
  - Works as a virtual channel that connects producers and subscribers
    - The receiver is subscribed to a topic or to a set of topics (using wildcards (+ and #)

Subscription to *topics*

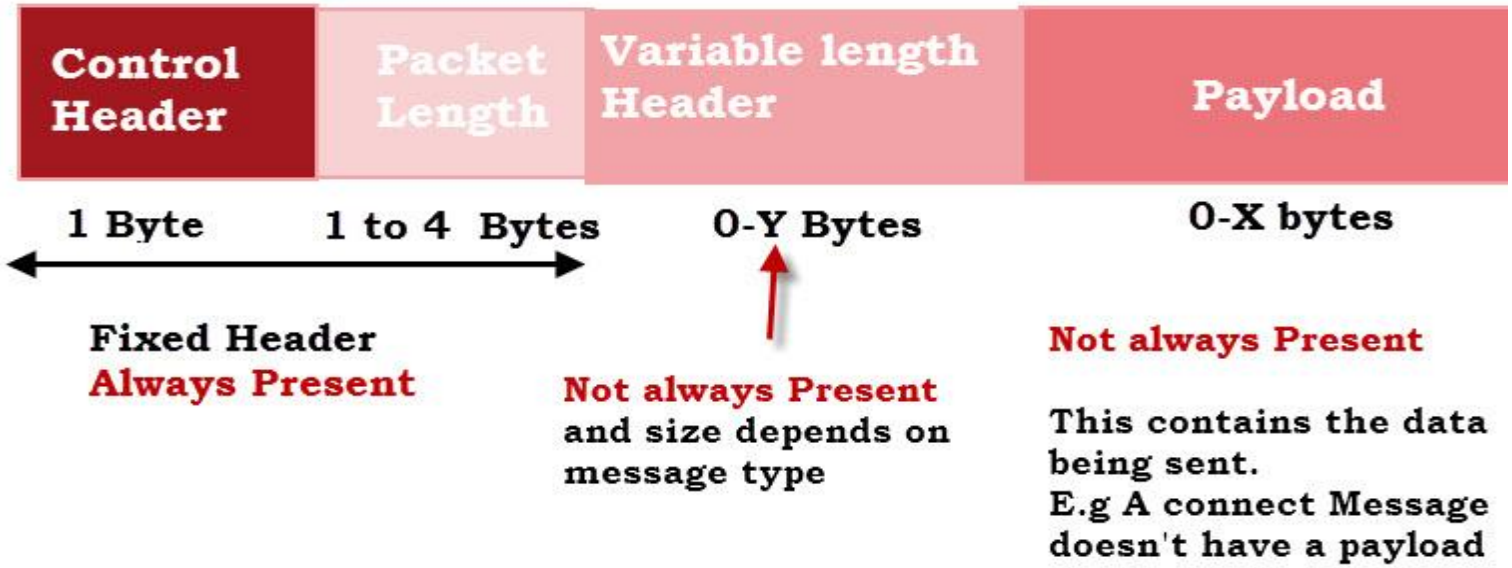


MQTT protocol



Source: http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

MQTT packets

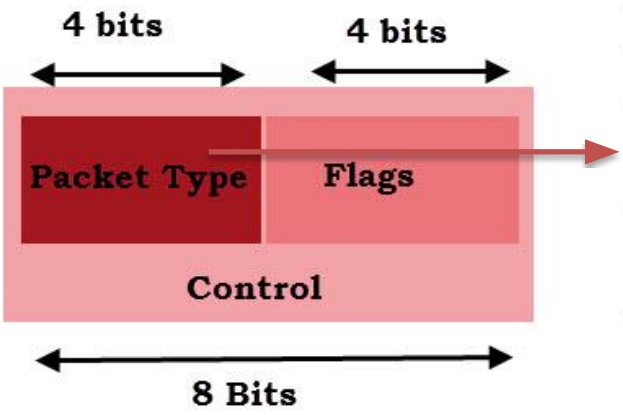


Source: http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

- Names of *topics, clientID,* usernames and *passwords* are encoded as strings (UTF-8)
- The general *payload* can be text or binary
- Maximum packet size of 256MB
  - Using a single byte for the length packet field, up to 127 bytes
  - The eight bit is to express continuity of packet

QTT

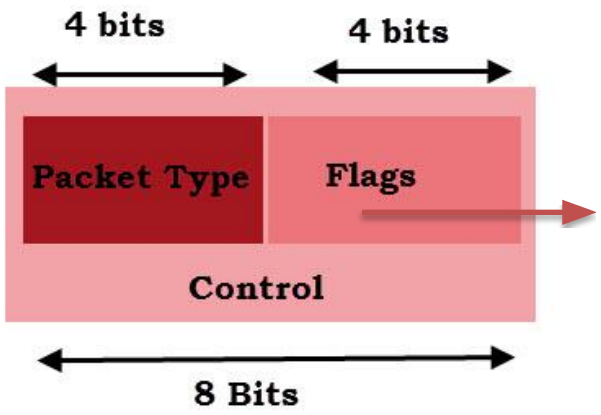
MQTT Control Filed



Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

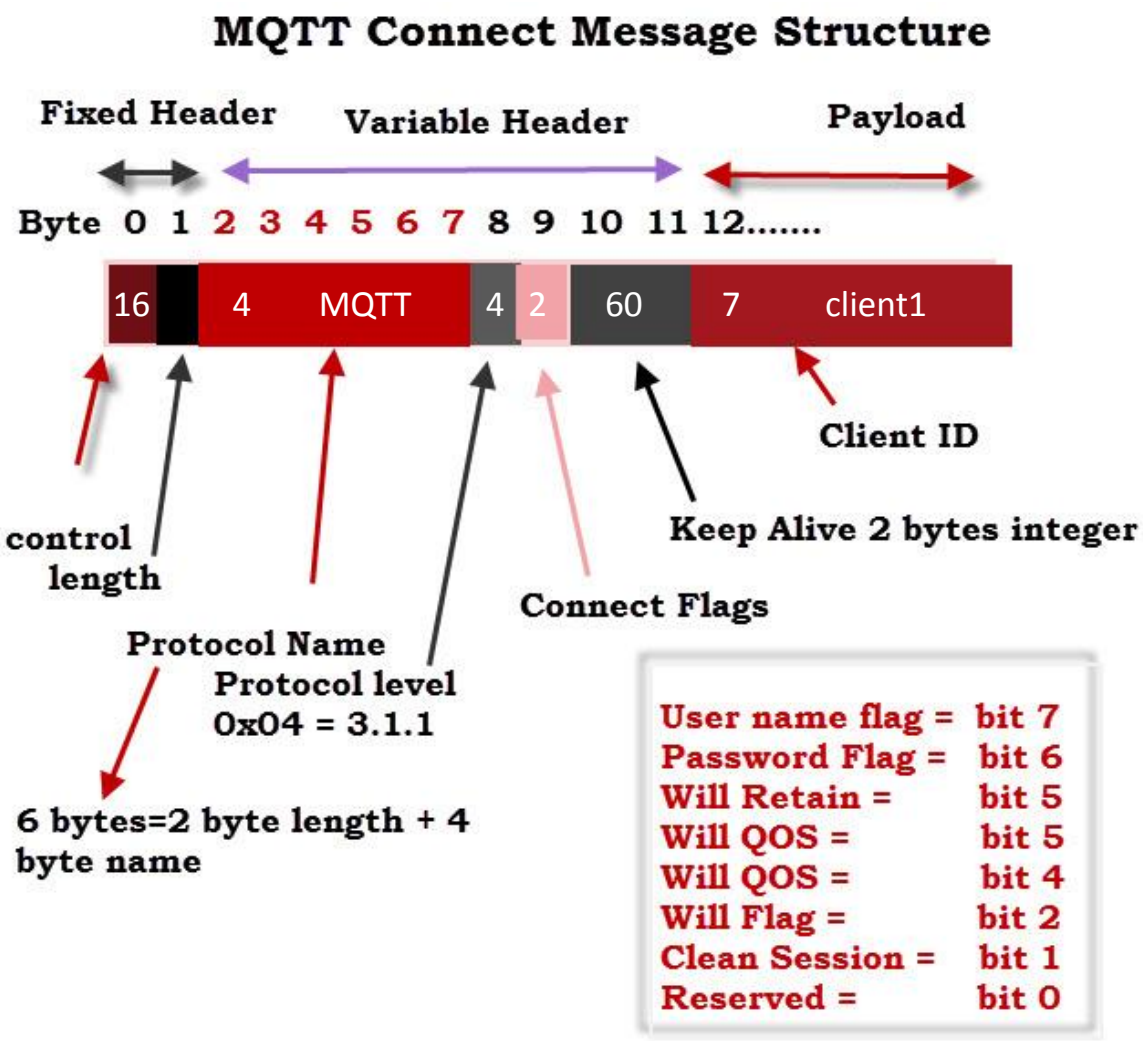
MQTT control field

MQTT Control Filed



Control Packet	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

Connection package example



Fuente:
http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

Some brokers

HiveMQ: **http://www.mqtt-dashboard.com** (can be used online)
ActiveMQ: **http://activemq.apache.org/**
RabbitMQ: **https://www.rabbitmq.com/**
Mosquitto: **http://mosquitto.org/**
Flespi: **https://flespi.com/mqtt-broker**
Mosca & **Aedes**: **http://www.mosca.io/**
Eclipse IoT: **http://iot.eclipse.org/**
VerneMQ: **https://verne.mq/**
Solace: **https://dev.solace.com**
CloudMQTT: **https://www.cloudmqtt.com**
Emqttd: **https://github.com/emqx/emqx**
Wave: **https://github.com/gbour/wave**
vertx-mqtt-broker: **https://github.com/GruppoFilippetti/vertx-mqtt-broker**
JoramMQ: **http://www.scalagent.com/en/jorammq-33/products/overview**
Moquette MQTT: **https://github.com/andsel/moquette**

IoT Platforms and Dashboards

- A *dashboard* allows a rich visualization of data submitted by nodes and to perform simple data modification
  - Can include an MQTT *broker*
  - An *IoT Platform usually includes more functionality: device control, updates, …*
- Examples
  - https://thingsboard.io (very rich)
    - http://www.steves-internet-guide.com/thingsboard-mqtt-dashboard/
  - https://thingspeak.com (allows Matlab processing)
  - https://thingstream.io
  - https://www.ptc.com/en/products/iot/thingworx-platform
  - https://wolkabout.com
  - http://iotgo.iteadstudio.com
  - https://www.kaaproject.org
  - https://www.elastic.co

Mosquitto

- Mosquitto (https://mosquitto.org/) is an open source MQTT implementation
- Implements both the broker and the clients (producer and consumer)
- Once installed, *broker* and clients can exchange messages (in the same or different machines)
  - Command *mosquitto_sub* to subscribe to a topic
  - Command *mosquitto_pub* to publish in a *topic*

Examples

Suscripción

```
marco@ubuntu:~$ mosquitto_sub -h localhost -t test
hallo
goodbye
```

Publicando

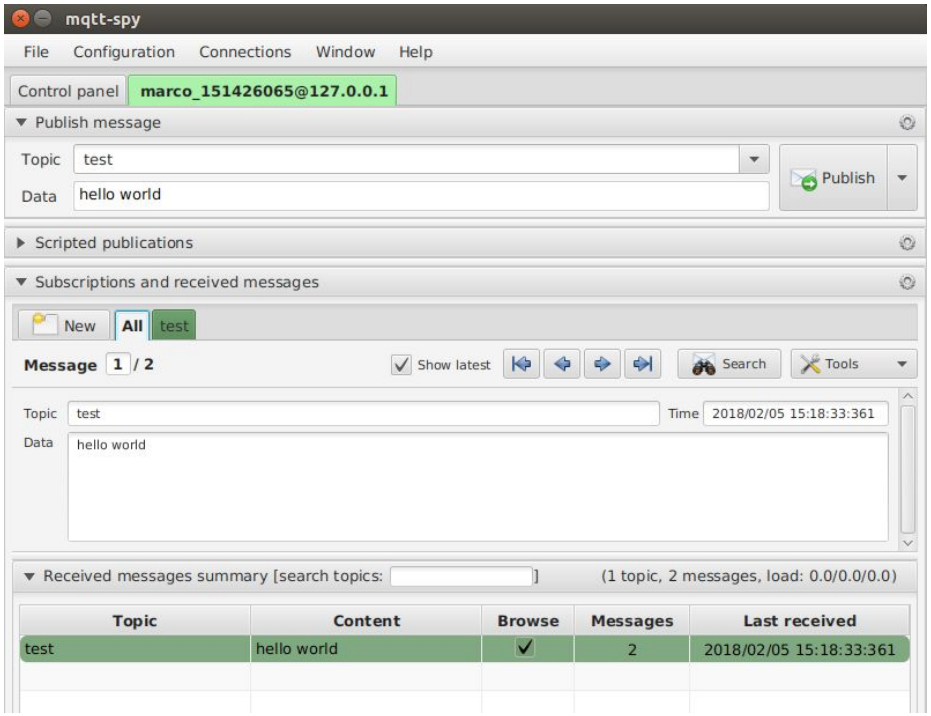
```
marco@ubuntu:~$ mosquitto_pub -h localhost -t test -m "hallo"
marco@ubuntu:~$ mosquitto_pub -h localhost -t test -m "goodbye"
marco@ubuntu:~$
```

- For this example, it is necessary to have a broker executing in the local machine

Example

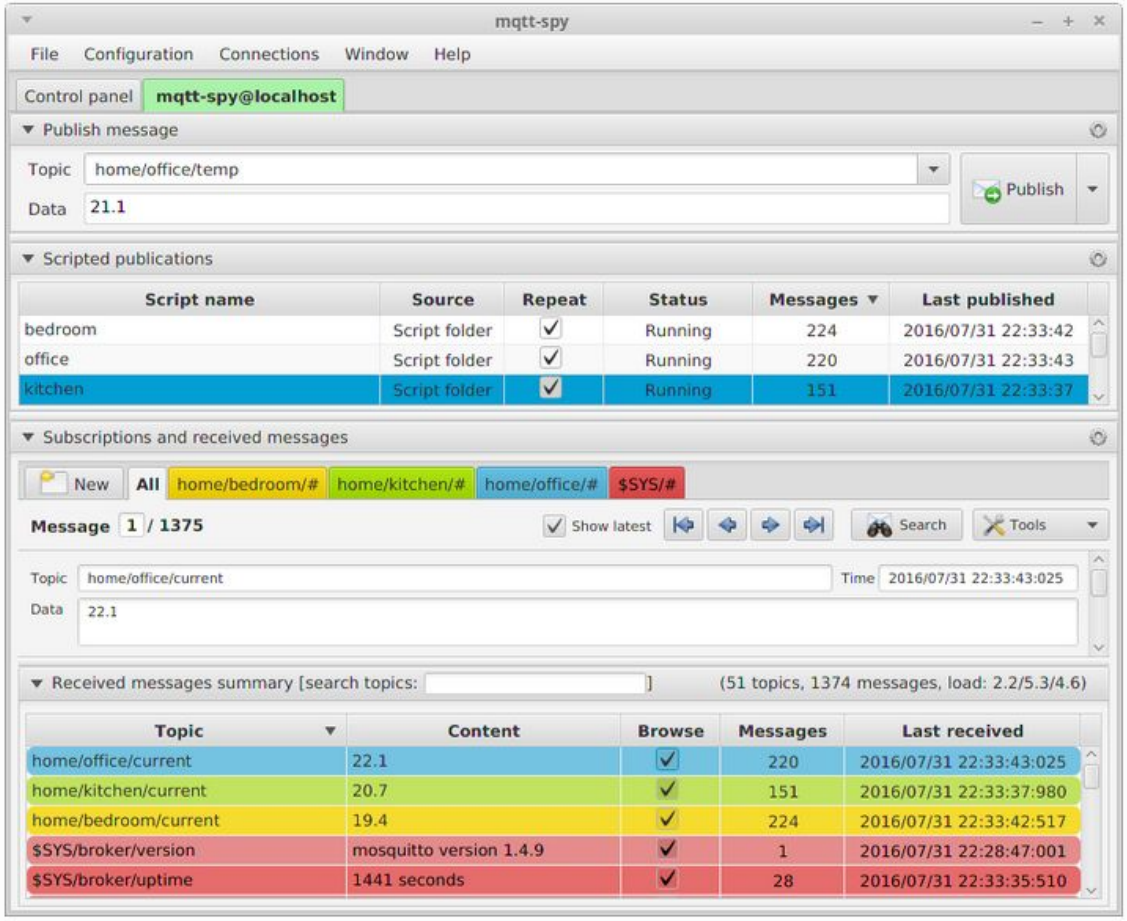
Suscripción

```
marco@ubuntu:~$ mosquitto_sub -h localhost -t test
123
hello world
```



mqtt-spy is an open source tool to monitor activity in MQTT topics

Client tool



mqtt-spy publish/subscribe (source: https://github.com/eclipse/paho.mqtt-spy)

Ejercicio

- Try to find an accessible MQTT broker and publish on it:
  - https://github.com/mqtt/mqtt.github.io/wiki/public_brokers
- Using mosquitto and/or mqtt-spy (https://github.com/eclipse/paho.mqtt-spy/releases)
- Generate synthetic data and visualize them on a dashboard (optional)

Example Wireshark

Wi-Fi: en0

mqtt

No.	Time	Source	Destination	Protocol	Length	Info
38	3.854971	192.168.1.34	18.194.184.181	MQTT	93	Connect Command
40	3.897232	18.194.184.181	192.168.1.34	MQTT	70	Connect Ack
189	36.999484	192.168.1.34	18.194.184.181	MQTT	78	Subscribe Request
190	37.035808	18.194.184.181	192.168.1.34	MQTT	71	Subscribe Ack
220	43.895463	192.168.1.34	18.194.184.181	MQTT	86	Publish Message
221	43.931336	18.194.184.181	192.168.1.34	MQTT	86	Publish Message
551	103.8969…	192.168.1.34	18.194.184.181	MQTT	68	Ping Request
552	103.9352…	18.194.184.181	192.168.1.34	MQTT	68	Ping Response
25…	163.9005…	192.168.1.34	18.194.184.181	MQTT	68	Ping Request
25…	163.9338…	18.194.184.181	192.168.1.34	MQTT	68	Ping Response
34…	223.9013…	192.168.1.34	18.194.184.181	MQTT	68	Ping Request
34…	223.9375…	18.194.184.181	192.168.1.34	MQTT	68	Ping Response
35…	234.3984…	192.168.1.34	18.194.184.181	MQTT	77	Unsubscribe Request
35…	234.4853…	18.194.184.181	192.168.1.34	MQTT	70	Unsubscribe Ack
35…	234.4861…	192.168.1.34	18.194.184.181	MQTT	68	Disconnect Req

```
▸ Frame 38: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface 0
▸ Ethernet II, Src: Apple_0c:dc:e4 (8c:85:90:0c:dc:e4), Dst: Mitrasta_4e:bc:ed (98:97:d1
▸ Internet Protocol Version 4, Src: 192.168.1.34, Dst: 18.194.184.181
▸ Transmission Control Protocol, Src Port: 55359, Dst Port: 1883, Seq: 1, Ack: 1, Len: 2
▾ MQ Telemetry Transport Protocol, Connect Command
  ▾ Header Flags: 0x10 (Connect Command)
      0001 .... = Message Type: Connect Command (1)
      .... 0000 = Reserved: 0
    Msg Len: 25
    Protocol Name Length: 4
    Protocol Name: MQTT
    Version: MQTT v3.1.1 (4)
  ▸ Connect Flags: 0x02
    Keep Alive: 60
    Client ID Length: 13
    Client ID: luis173204484
```

Frame (frame), 93 bytes | Packets: 4213 · Displayed: 15 (0.4%) | Profile: Default

Links of interest

- http://mqtt.org
- http://docs.oasis-open.org/mqtt/mqtt/
- http://www.steves-internet-guide.com/mqtt/
- https://github.com/mqtt/mqtt.github.io/wiki/public_brokers