Practical Activity

MQTT Protocol

Hélio Nakazato
Hirley Dayan

MQTT Last Will Implementation

```
28    def on_connect(topic, last_will_msg=LAST_WILL_DEFAULT_MSG,
29                   last_will_qos=LAST_WILL_DEFAULT_QOS):
30        """On connect."""
31        client = mqtt.Client(protocol=mqtt.MQTTv311)
32        client.will_set(topic, last_will_msg, last_will_qos,
33                        retain=LAST_WILL_DEFAULT_RETAIN)
34        client.connect(MQTT_ADDRESS, MQTT_PORT, MQTT_TIMEOUT)
35        client.loop_start()
36        return client
```

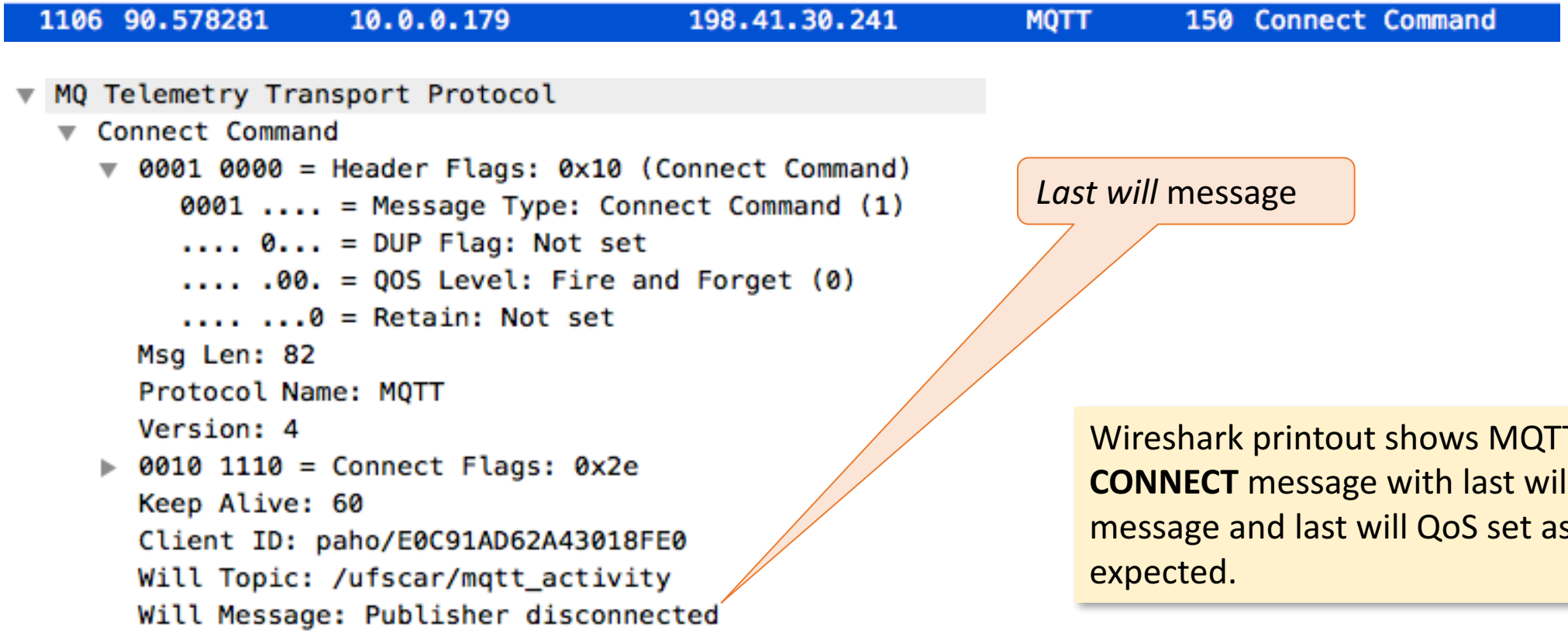
Last will definitions before CONNECT

CONNECT

Paho loop for MQTT protocol messages handling

Paho python library allows setting *last will* message and related *last will* QoS via will_set() function. As shown on the code above, **will_set()** function must be called before **connect()** function, as expected by the protocol, since last will message must be defined during **CONNECT** message.

MQTT Last Will Implementation



Wireshark printout shows MQTT **CONNECT** message with last will message and last will QoS set as expected.

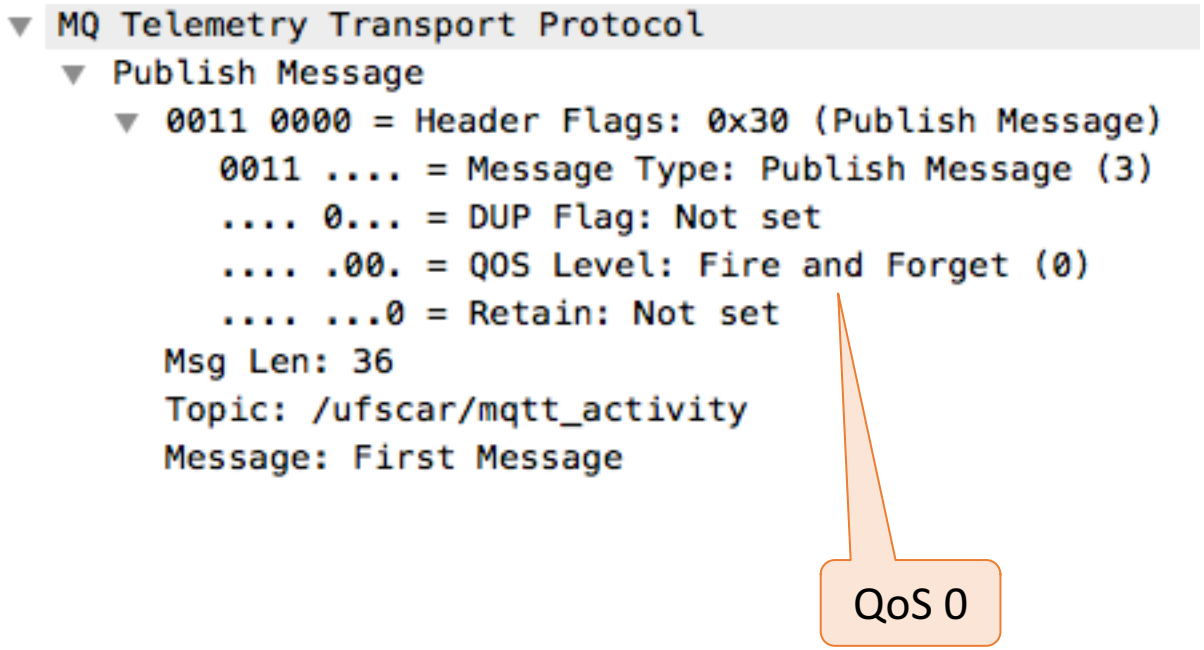
MQTT Publishing Message With QoS

```
39    def send_message(topic, client, msg, qos=MQTT_DEFAULT_QOS):
40        """Send message."""
41        result, message_id = client.publish(topic, msg, qos=qos)
42        print('Sent message - id: %d, QoS: %d' % (message_id, qos))
```

Paho python library allows setting **PUBLISH** message with the desired QoS code, as shown by the code above.

MQTT Publishing Message (QoS 0)

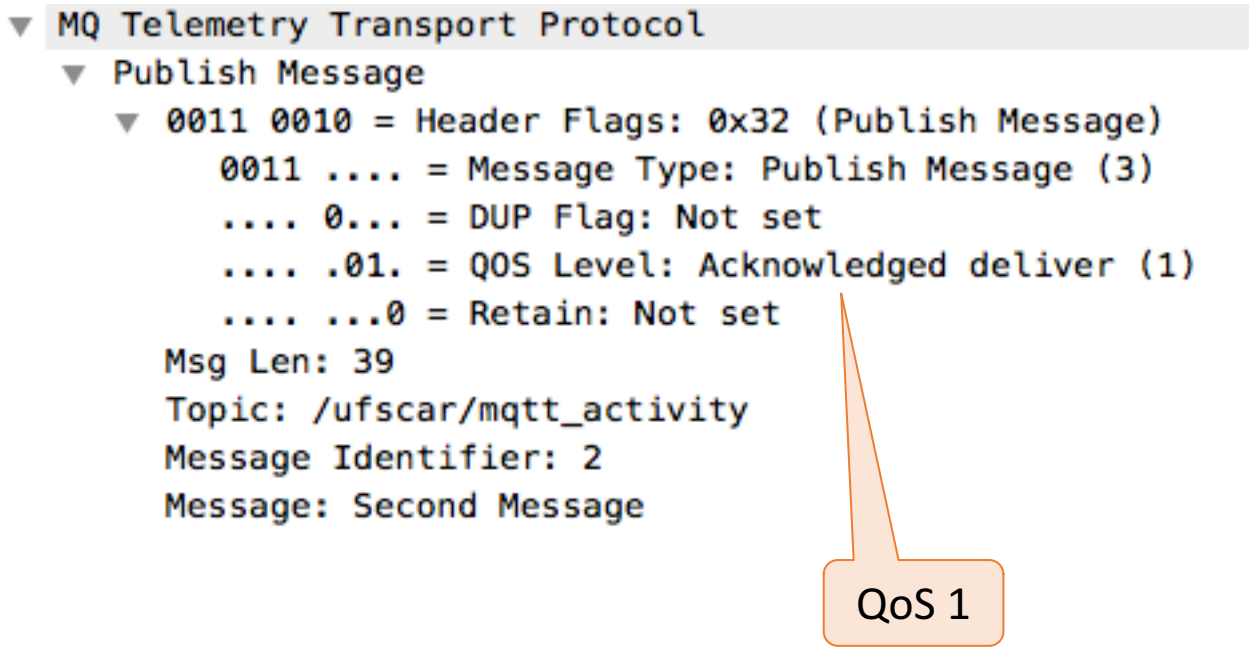
1142	106.028020	10.0.0.179	198.41.30.241	MQTT	104	Publish Message
1145	106.615246	198.41.30.241	10.0.0.179	MQTT	104	Publish Message



During **PUBLISH** message, QoS 0 is set as shown by the decoded message. For QoS 0 it is not expected any confirmation back from the broker as shown by the protocol flow above.

MQTT Publishing Message (QoS 1)

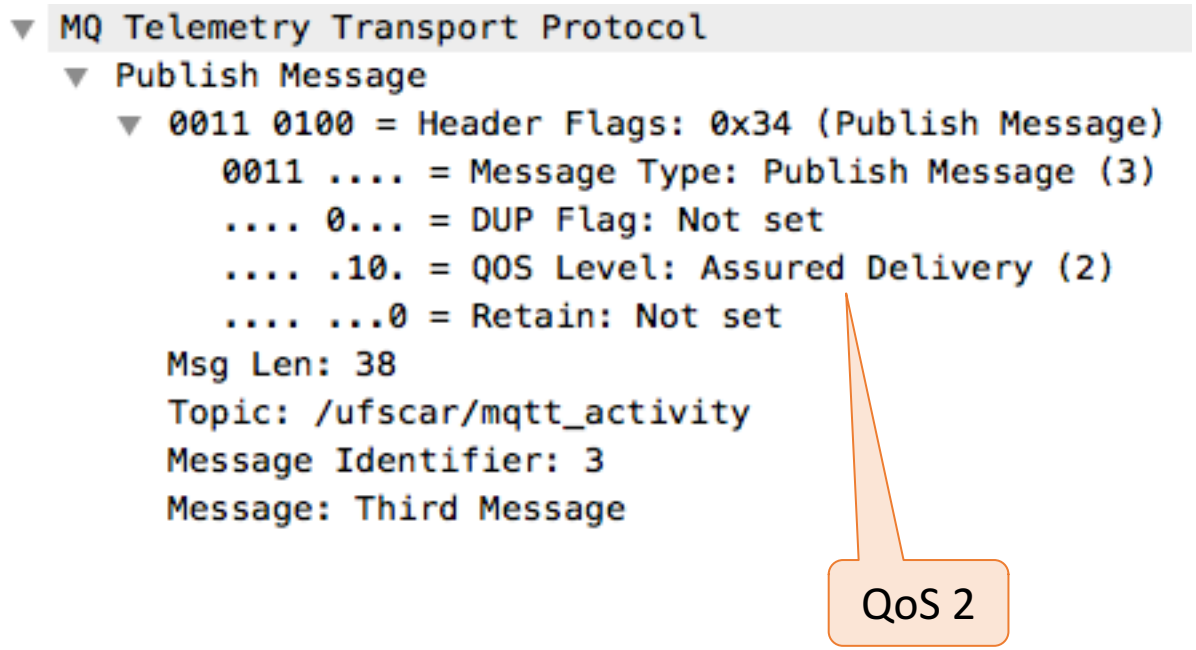
6844	277.672752	10.0.0.179	198.41.30.241	MQTT	107	Publish Message
6846	277.928560	198.41.30.241	10.0.0.179	MQTT	70	Publish Ack
6847	277.928567	198.41.30.241	10.0.0.179	MQTT	105	Publish Message



During a **PUBLISH** message, QoS 1 is set as shown by the decoded message.
For QoS 1 an **PUBACK** message is received by the publisher, from the broker, as shown by the protocol flow above.

MQTT Publishing Message (QoS 2)

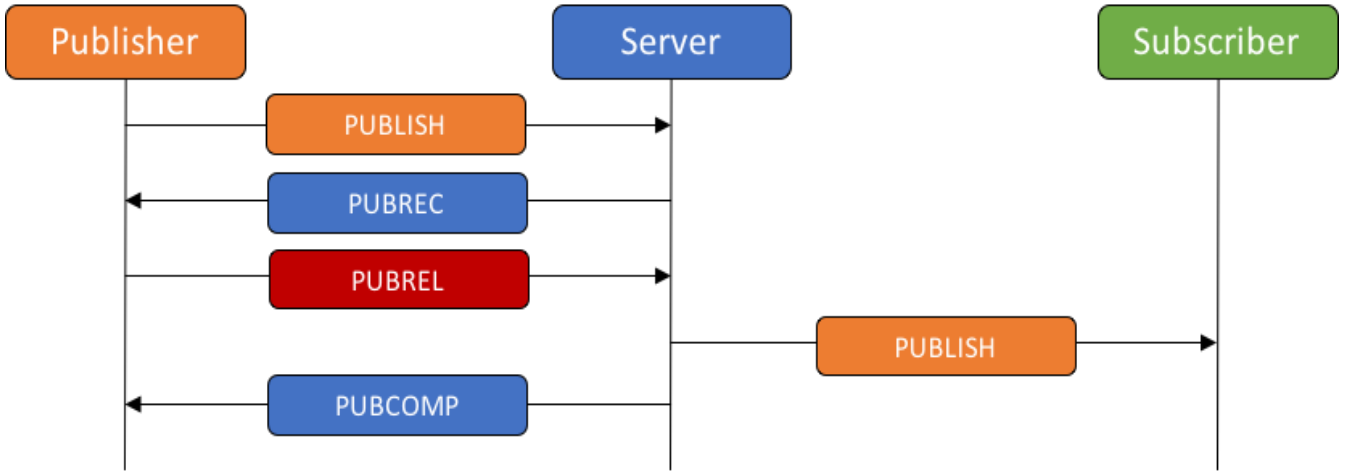
94	9.694709	10.0.0.179	198.41.30.241	MQTT	106	Publish Message
103	9.945402	198.41.30.241	10.0.0.179	MQTT	70	Publish Received
105	9.946026	10.0.0.179	198.41.30.241	MQTT	70	Publish Release
113	10.156050	198.41.30.241	10.0.0.179	MQTT	70	Publish Complete
115	10.162182	198.41.30.241	10.0.0.179	MQTT	104	Publish Message



During **PUBLISH** message, QoS 2 is set as shown by the decoded message. For QoS 2, a **PUBREC** message is received by the publisher, from the broker. Once a **PUBREC** is received, the publisher releases the message on the broker sending a PUBREL message **PUBCOMP** is also received by the publisher once broker releases the message to the subscribers

MQTT Publishing Message (QoS 2)

94	9.694709	10.0.0.179	198.41.30.241	MQTT	106	Publish Message
103	9.945402	198.41.30.241	10.0.0.179	MQTT	70	Publish Received
105	9.946026	10.0.0.179	198.41.30.241	MQTT	70	Publish Release
113	10.156050	198.41.30.241	10.0.0.179	MQTT	70	Publish Complete
115	10.162182	198.41.30.241	10.0.0.179	MQTT	104	Publish Message



The picture on the left shows the message flow between publisher, broker and subscriber as captured by Wireshark messages above.