Data and Information Quality

## ID Project: 3

## Dataset: 1

Luca Mattana – ID Person: 10104067

# CoAP and MQTT Traffic Analysis

## CoAP

### 1a.

All GET requests to a temperature resource (.../temperature) are from the source 127.0.0.1 to the destination 127.0.0.1 (localhost). There are 9 GET requests from 8 different ports. Assuming that each client when started opens a unique port, the number of distinct clients is: **8**



Wireshark Filter:

* coap.code==1 and coap.opt.uri\_path=="\temperature"

### 1b.

Considering the assumption above, the table below shows the MID and the longest CoAP response for each client found in 1a.

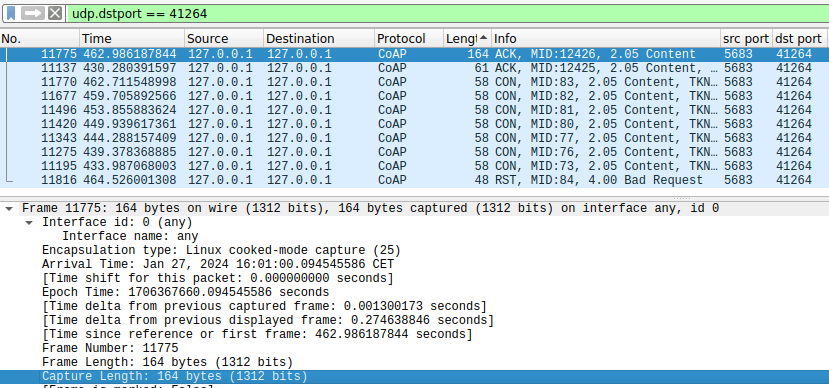
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **src port (client)** | **lenght** | **MID** |
| 1 | 55898 | 62 | 25 |
| 2 | 33677 | 62 | 26 |
| 3 | 51812 | 119 | 29 |
| 4 | 48049 | 189 | 10589 |
| 5 | 52276 | 62 | 7365 |
| 6 | 48645 | 62 | 63 |
| 7 | 52247 | 113 | 47747 |
| 8 | 41264 | 164 | 12426 |

Wireshark Filter:

for each client port found in 1a I filtered the destination UDP Port, ordered for length and chose the longest response:

udp.dstport == <XXXXX>

*Example of applied filter with Wireshark to the udp destination port of a client.*



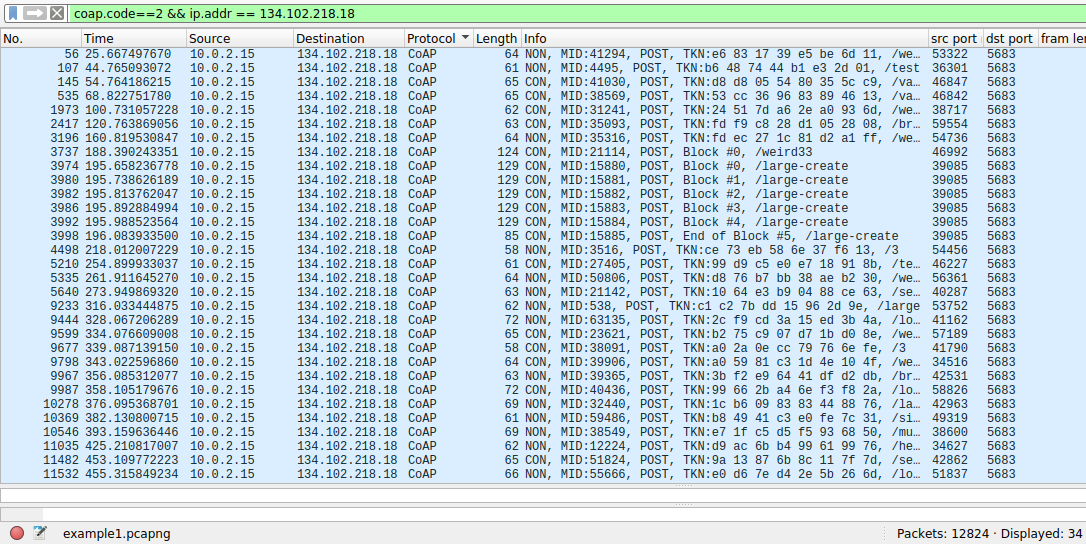
## CoAP

### 2a.

IP *coap.me* server: 134.102.218.18

1. I found all POST request sent to the *coap.me* server

Wireshark Filter: coap.code==2 && ip.addr == 134.102.218.18



1. I filtered all response from the *coap.me* server

Wireshark Filter: udp.srcport==5683 and ip.addr == 134.102.218.18

For each result of the filters at step 1 and 2 I exported a csv file and used Excel to cross reference for each POST request the corresponding response, using the token as key. For 2 different blocks of really long requests (first: MID:21114; second: MID from 15880 to 15885) I cross referenced with MID because there are CON requests (so the response definitely has the same MID) and I have not the token immediately visible to me after filtering.

The table below shows the result:



There is a POST request to *coap.me* with MID:12224 and coap.token==d9:ac:6b:b4:99:61:99:76 that it is not counted because it has no response at all.

Number of CoAP POST request to the *coap.me* server with a not successful response: **17**

### 2b.

Considering the CoAP POST request from 2a, I filter those targeting a “weird” resource



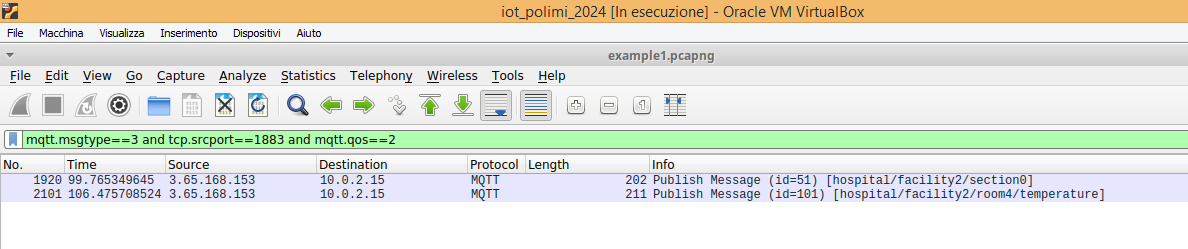
In the table above there are all the request to a “*/weird.....”* resource.

Number of CoAP POST req with not successful response directed to a resource of type “/weird…..”: **8**

## MQTT

### 3a.

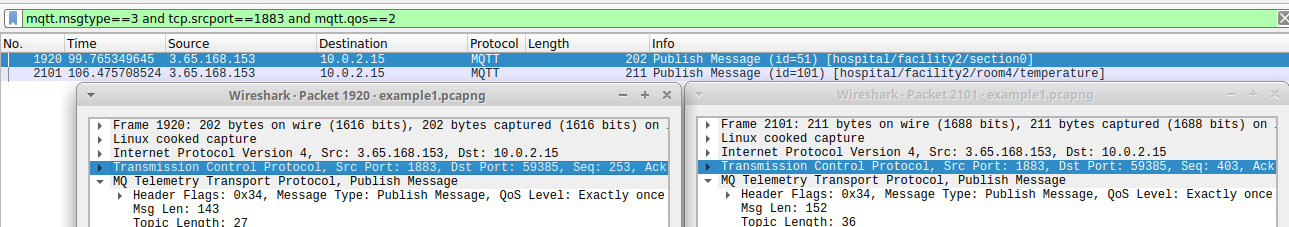
To seek clients, I filter MQTT publish message (msgtype=3) from port 1883 and QoS=2



Wireshark Filter: mqtt.msgtype==3 and tcp.srcport==1883 and mqtt.qos==2

Applying the filter above, the Publish Messages “Received” by the clients with QoS=2 are **2**

### 3b.



Number of clients involved: **1**

The *Destination Port* for both message is 59385 🡪 Assuming that each client on the machine is using the same port we can ensure that the same client is involved in the messages above.

Wireshark Filter: mqtt.msgtype==3 and tcp.srcport==1883 and mqtt.qos==2

### 3c.

Applying a filter on MQTT Subscribe Request and inspecting the source IP, source port and requested resources it is possible to detect a unique plausible Subscribe Request message from the client at 3b, to receive the messages found in 3a:

Topic of Publish Messages:

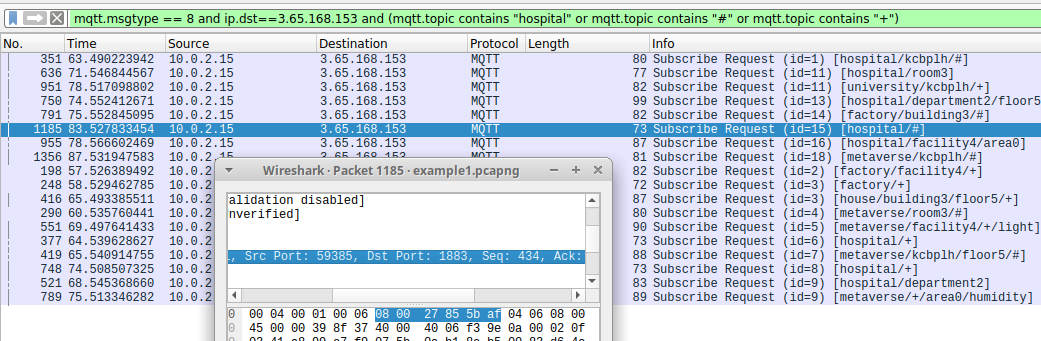
* hospital/facility2/section0
* hospital/facility2/room4/temperature

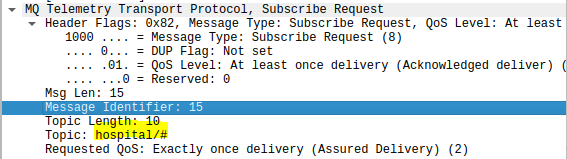
Message Identifier: **15**  
 Subscribed topic: **hospital/#**

Wireshark Filter:

mqtt.msgtype == 8 and ip.dst==3.65.168.153 and (mqtt.topic contains "hospital" or mqtt.topic contains "#" or mqtt.topic contains "+")

*The images below show the result with Wireshark*





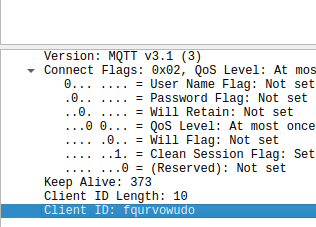
## MQTT

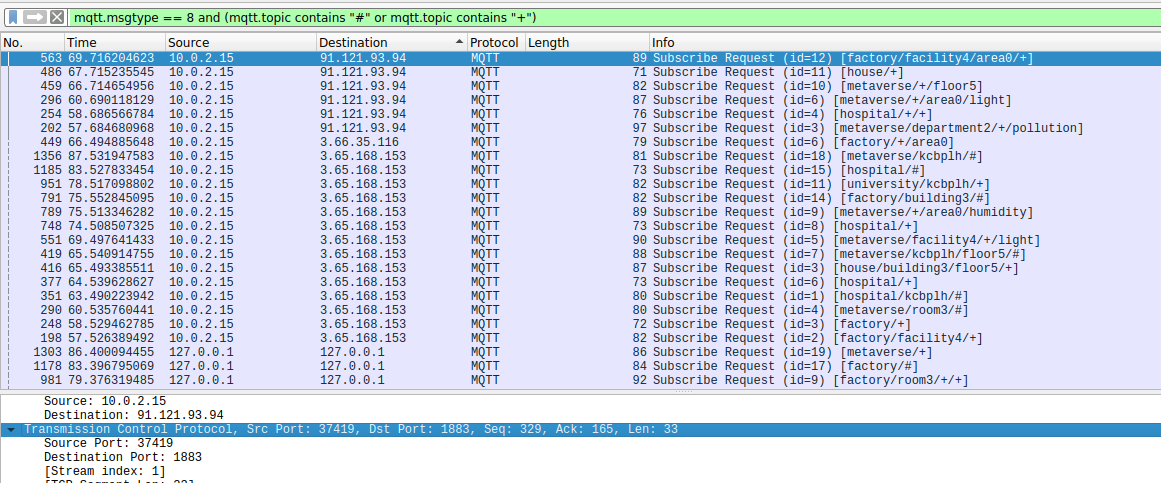
4a.

Number of clients that sent a subscribe message to a public broker using at least one wildcard: **4**

They are identified applying a filter to detect which subscribe request used a wildcard, then selecting only those directed to a not local IP address. Finally checking the client ID (filtering the Connect packet) and the source port, it is possible to count the number of clients.

Wireshark Filter: mqtt.msgtype==8 and (mqtt.topic contains "#" or mqtt.topic contains "+")

 mqtt.msgtype == 1 and tcp.srcport==<XXXXX>



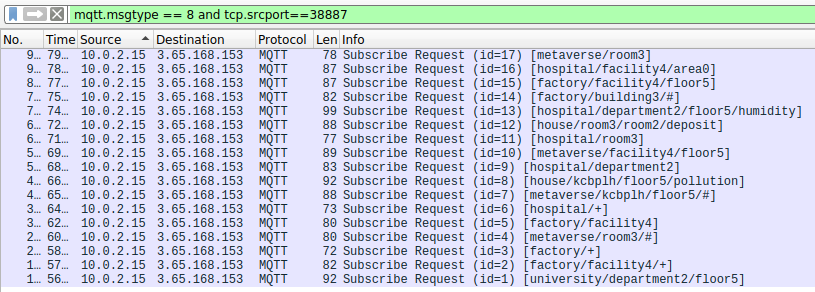
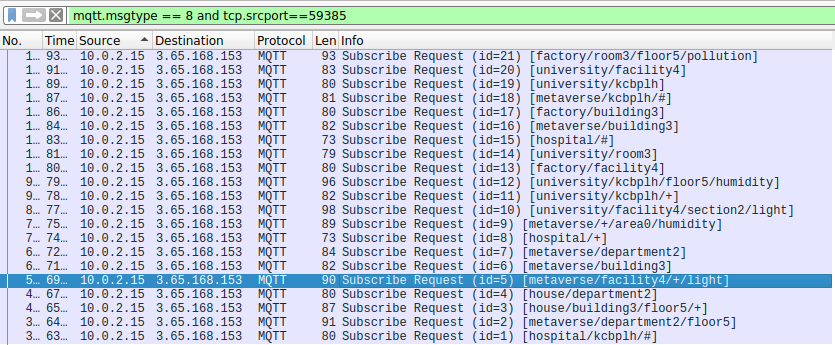
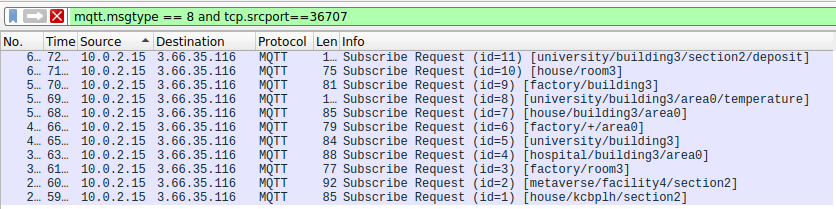
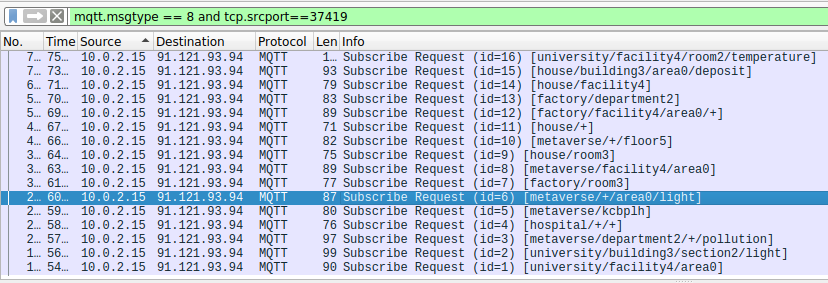
4b.

The clients of point 4a have ports and Client ID: 37419 (twsjob), 36707 (gubiekhhgloyqod), 59385 (fqurvowudo), 38887 (hxrutjdrvecur).

Applying the filter on Subscribe Request and source port it is possible to detect all the subscription  
for the four clients because there are not too many sub req.

Clients with a subscription that let them eventually receive a public message directed to the topic:  
"metaverse/facility4/area0/light“ : **2** (*port 37419* and *port 59385*)

Wireshark Filter: mqtt.msgtype == 8 and tcp.srcport==<*xxxxx>*

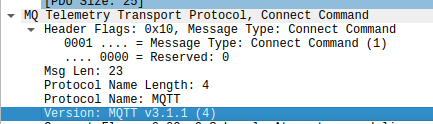


## MQTT ACK

## 

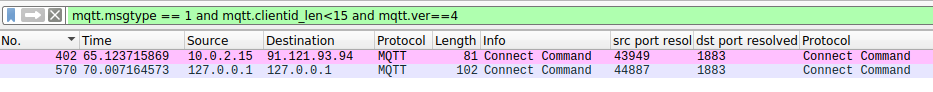
5.

MQTT version 3.1.1 corresponds to mqtt.ver==4



I proceed in 2 steps:

1. I seek clients who connected to brokers specifying a client identifier shorter than 15 bytes
2. I detect all MQTT ACK messages in total received by the clients found at step 1
3. Applying the filter 1) (Wireshark filter below) 2 clients are found.

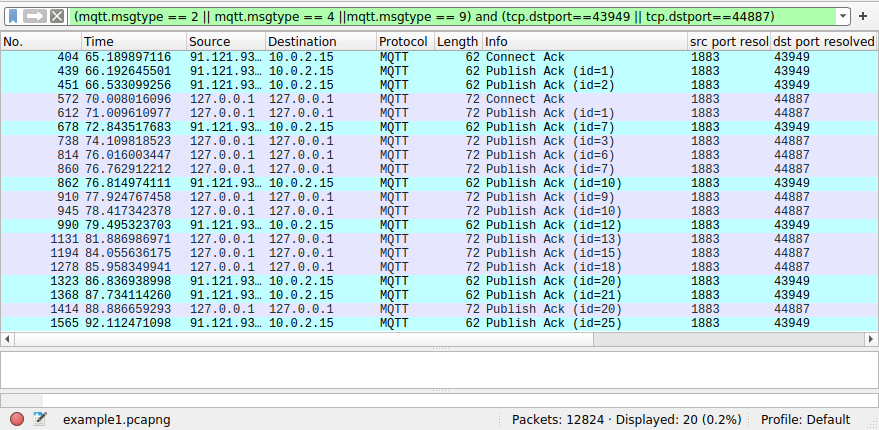


|  |  |  |
| --- | --- | --- |
| **tcp.srcprt** | **Client ID** | **Client ID length** |
| 43949 | poypeinlcvu | 11 |
| 44887 | apzsfux | 7 |

1. The table below shows what in general, depending on the “Quality of Service (QoS)”, we can consider ack messages and the correspondent mqtt.msgtype. In this specific case we will filter only the message type containing “ACK”, so mqtt.msgtype 2, 4 and 9 .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **mqtt.msgtype** | **msg** | **QoS 0** | **QoS 1** | **QoS 2** |
| 2 | Connect ACK | X | X | X |
| 4 | Publish ACK |  | X |  |
| 5 | Publish Received |  |  | X |
| 7 | Publish Complete |  |  | X |
| 9 | Subscribe ACK | X | X | X |

In the image below is shown the result of filtering the mqtt.type with values 2, 4 and 9 for the two clients found in step 1, using their destination ports (tcp.dstport).



In total the clients who connected to brokers specifying a client identifier shorter than 15 bytes and using MQTT version 3.1.1 received **20** MQTT ACKmessages

Wireshark Filter:

1. mqtt.msgtype == 1 and mqtt.clientid\_len<15 and mqtt.ver==4
2. (mqtt.msgtype == 2 or   
    mqtt.msgtype == 4 or  
    mqtt.msgtype == 9)  
   and (tcp.dstport==43949 or tcp.dstport==44887)

## MQTT

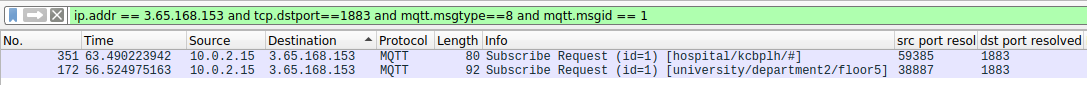
### 6a.

The ip address for hiveMQ Broker in this case is 3.65.168.153.

I filtered all subscription requests (mqtt.msgtype -> 8) with message ID=1, destination ip 3.65.168.153 and destination port 1883

Wireshark Filter:

ip.addr == 3.65.168.153 and tcp.dstport==1883 and mqtt.msgtype==8 and mqtt.msgid == 1



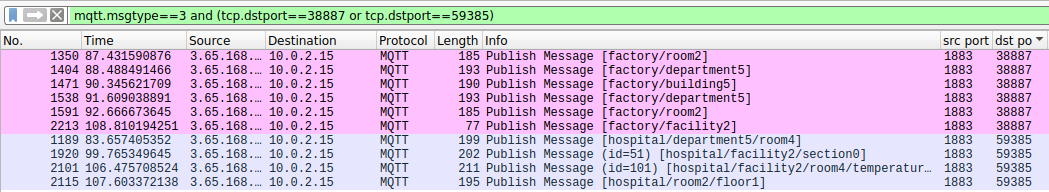
MQTT subscribe requests with message ID=1 directed to the HiveMQ: **2**

### 6b.

At point 6a two subscriptions were found:

Client on port 59385 🡪 “hospital/kcbplh/#”  
 Client on port 38887 🡪 “university/department2/floor5”

The image below shows all Publish Messages received by the 2 clients.



Wireshark Filter: mqtt.msgtype==3 and (tcp.dstport==38887 or tcp.dstport==59385)

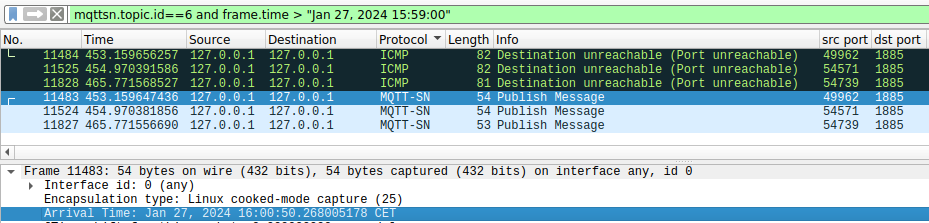
There are only few messages so I do not need even to filter the topic and I can read them directly.  
 There are no publish messages generated by the subscriptions in point 6a.

The clients received **0** messages.

## MQTT-SN

### 7a.

Wireshark Filter: mqttsn.topic.id==6 and frame.time > "Jan 27, 2024 15:59:00"



Number of MQTT-SN Publish Messages sent after the hour 3.59PM (Milan Time) directed to topic 6: **3**

### 7a

There are some possible reasons for which the messages at 7a are not handled by the server.

* The destination port the messages are sent to may not be correct. This could be a likely reason considering that the destination is 127.0.0.1 (localhost) and the destination port is 1885 that is the default port for MQTT-SN brokers. The localhost broker may be set on a different port. In the image above (ordered by protocol, not by time) it is also possible to detect the info sent back to the server with the ICMP messages: (Port unreachable).
* Another possible reason it could be that the server is down or not active for some reason.