1. The main difference is that the packet with mid (message ID) 53533 are a

confirmable message and its ACK, while the one with mid 42804 is a

non-confirmable one. We can clearly see it on wireshark with filters:

coap.mid == 53533 || coap.mid == 42804.

Notice that the NON delete request was denied and we can check it with the following filter:

coap.token == 6d:bd:d0:20.

While the other request (the CON one) was accepted.

1. I select in the "go to the packet" the number 2428 and

it is a Non-Confirmable message, so it doesn't have any ACK replies.

Then, I wrote in the filter: coap.token == 67:c7:22:9a looking for response

to the request for DELETE. The reply is the packet number 2429 (2.02 code, a success).

1. I selected in the filter: coap.code == 69 && ip.addr == 127.0.0.1, where the former condition

implies that the code must be "Content", while the latter the address of client "localhost".

I have found 8 packets. The last condition (the replies to requests of type Confirmable)

is also satisfied because the packets are all ACK, so they are associated to a confirmable

message (which must be acknowledged).

1. First of all let's find out how many GET request packets there are excluding OBSERVE

requests with the following filters: coap.code == 1 && !(coap.opt.observe).

There are 16 packets. There are only three address and we can check them separately looking at the mid in case of CON message or at the token in case of NON.

The packets with destination 134.102.218.18 are all NON, so we can see the replies looking

at the tokens (we can easily extrapolated them and prepare a filter with Wireshark):

((((coap.token == 63:81:f3:1e) || (coap.token == 9e:24:23:75)) || (coap.token == cd:66:f1:75)) || (coap.token == 20:dd:92:35)) || (coap.token == 4a:da:09:18)

There are only successful answers 2.05.

For destination 104.196.15.150, we can apply easily the filter:

((((coap.mid == 26629) || (coap.mid == 28357)) || (coap.token == 09:10:17:6c)) || (coap.token == 09:10:17:6c)) || (coap.token == fe:d8:f1:0b)

where we select the mid for the CON messages and the token for the NON.

Looking at the answers, we can see 3 packets had a client error 4.05 "Method Not allowed", while 2 were successful 2.05.

For the address 127.0.0.1, we can use the prepare the following filter:

(((((coap.mid == 15597) || (coap.mid == 10267)) || (coap.mid == 54030)) || (coap.mid == 2441)) || (coap.mid == 57705)) || (coap.mid == 27097)

There are 6 ACK replies with error 4.04 "Not Found". So, there are only 6 get requests directed to non existing resources.

1. First of all, we need to find who are those with password "admin": mqtt.passwd == "admin".

There are 8 packets. To find the same client we can look at the IP source and the source port.

Notice that the ip address is always the same. So, we can prepare the filter considering also the fact that the topic should contain “factory/department”:

((((((((((tcp.srcport == 51565) || (tcp.srcport == 41869)) || (tcp.srcport == 60395)) || (tcp.srcport == 40989)) || (tcp.srcport == 47135)) || (tcp.srcport == 44429)) || (tcp.srcport == 60419)) || (tcp.srcport == 55953)) && ip.src == 127.0.0.1) && mqtt.topic contains "factory/department")

Notice that there are 20 packets, but there are some subscritions. So, we need to update the filter in order to avoid them:

((((((((((tcp.srcport == 51565) || (tcp.srcport == 41869)) || (tcp.srcport == 60395)) || (tcp.srcport == 40989)) || (tcp.srcport == 47135)) || (tcp.srcport == 44429)) || (tcp.srcport == 60419)) || (tcp.srcport == 55953)) && ip.src == 127.0.0.1) && mqtt.topic contains "factory/department") && mqtt.msgtype == 3

At the end, we have only 13 packets.

1. On terminal I used "ping test.mosquitto.org" in orger to get the ip address which is 5.196.95.208.

Then, I used the filter: ip.addr == 5.196.95.208 && mqtt.willmsg

There are 9 packets that specify a will message and connect to the public broker "mosquitto".

Notice that we have taken as assumption that the client is defined as source ip and source port.

All the previous packets have the same source ip and different ports, so they are all different clients, so the answer is 9. In the case we assume that a client is defined by only the ip address (because the ports may change over time), then we have only 1 client.

1. To find the PUBREL message we need the filter: mqtt.msgtype == 6. However, no packets are found. So, we can count all the publishers with qos == 2, because none received the PUBREL.

mqtt.qos == 2 && mqtt.msgtype == 3

There are 94 packets.

1. With the following filters we can find 22 packets with empty client id and

mqtt.clientid\_len == 0 && mqtt.willmsg

We have added the column for the length of the topic will. Then, we can extract the current frame of packets going to file-> extract packet dissection-> as csv. Finally, we run the following code on Rstudio (after adjusting the directories):

a <- read.csv("ex8")

mean(a[,6]) #6 is the column of the will topic length

The result is 37.77273.

1. We select the client with user.id: mqtt.clientid == "6M5H8y3HJD5h4EEscWknTD"

Then, we consider the client as a combination of ip address and port source. So, we use them as destination applying the following filter:

ip.addr == 10.0.2.15 && tcp.port == 46295 && (mqtt.msgtype == 2 || mqtt.msgtype == 4 || mqtt.msgtype == 9 || mqtt.msgtype == 11 )

Notice that msgtype 2,4,9 and 11 are all different types of ACK messages. There are 3 packets:

n°4764, n°4766 and n°4768.

However, we noticed that some packets contain more than one message.

N°4766 contains 3 subscribe acks while the other two packets contain only one ack each.

So, there are 5 ACKs in total.

1. First of all, we need to use the filters to consider only CONNECT messages that uses mqttv3.1 protocol: mqtt.ver == 3 && mqtt.msgtype == 1

We will find 47 packets. Then, we can extract the current frame of packets going to file-> extract packet dissection-> csv. Finally, I run the following code on Rstudio (after adjusting the directories):

a <- read.csv("ex10")

mean(a[,7]) #7 is the column of the message length

The result of the mean over the message length is 63.59574.

The difference is due to the variable payload. For instance, some packets have the will message and other don’t have it, so the length of the payload is different. So, also the length of the message is different.