

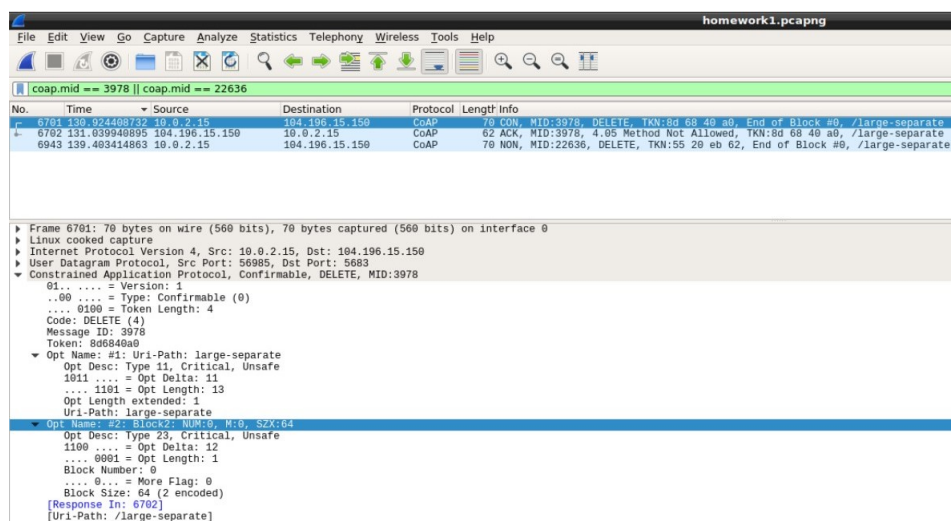
#01: CoAP-MQTT Sniffing

Andrea Calici, C. Persona 10490117, Matricola 944717

1) What's the difference between the message with MID: 3978 and the one with MID: 22636?

The message with MID = 3978 is a **Confirmable (CON)** message (if it doesn't receive an ACK, it will retransmit after a certain timeout), the message with MID = 22636 is a **Non Confirmable (NON-CON)** message. It's possible to find them with the following filter:

```
coap.mid == 3978 || coap.mid == 22636
```



2) Does the client receive the response of message No. 6949?

Yes, it receives the response in the message No. 6953 with an ACK that contains a **Method Not Allowed** error. It can be found simply by looking the message No. 6949 without using any particular filter.

No.	Time	Source	Destination	Protocol	Length	Info
6946	139.404703594	10.0.2.15	194.196.15.150	CoAP	64	NON, MID:28969, GET, TKN:09 10 17 6c, End of Block #0, /obs-large
6947	139.405027778	10.0.2.15	194.196.15.150	CoAP	64	NON, MID:28969, GET, TKN:09 10 17 6c, End of Block #0, /obs-large
6950	139.513910979	194.196.15.150	10.0.2.15	CoAP	62	NON, MID:385, 4.05 Method Not Allowed, TKN:65 cb 68 20, /large
6951	139.51784370	194.196.15.150	10.0.2.15	CoAP	62	NON, MID:37670, 4.05 Method Not Allowed, TKN:55 20 eb 62, /large-separate
6952	139.518696137	194.196.15.150	10.0.2.15	CoAP	188	NON, MID:27381, 2.05 Content, TKN:09 10 17 6c, Block #0, /obs-large (text/plain)
6953	139.519148689	194.196.15.150	10.0.2.15	CoAP	62	ACK, MID:28357, 4.05 Method Not Allowed, TKN:6f b6 3c 18, /location-query
6958	139.531119515	10.0.2.15	194.196.15.150	CoAP	64	NON, MID:28970, GET, TKN:09 10 17 6c, Block #1, /obs-large
6959	139.643322581	194.196.15.150	10.0.2.15	CoAP	92	NON, MID:27382, 2.05 Content, TKN:09 10 17 6c, End of Block #1, /obs-large (text/plain)
7055	141.569229196	10.0.2.15	194.196.15.150	CoAP	50	CON, MID:27239, DELETE, TKN:5f 12 62 69, End of Block #0, /sink

Frame 6946: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 10.0.2.15, Dst: 194.196.15.150
- User Datagram Protocol, Src Port: 59149, Dst Port: 5683
- Constrained Application Protocol, Confirmable, GET, MID:28357
 - 01 = Version: 1
 - ..00 = Type: Confirmable (0)
 -0100 = Token Length: 4
 - 0001 1101 (1)
 - Message ID: 28357
 - Token: 8f063c18
 - Opt Name: #1: Uri-Path: location-query
 - Opt Desc: Type 11, Critical, Unsafe
 - 1011 = Opt Delta: 11
 - ... 1101 = Opt Length: 13
 - Opt Length extended: 1
 - Uri-Path: location-query
 - Opt Name: #2: Block2: NUM=0, M=0, SZX=32
 - Opt Desc: Type 23, Critical, Unsafe
 - 1100 = Opt Delta: 12
 - 0001 = Opt Length: 1
 - Block Number: 0
 -0... = More Flag: 0
 - Block Size: 32 (1 encoded)
 - [Response In: 6953]
 - [Uri-Path: /location-query]

3) How many replies of type confirmable and result code “Content” are received by the server “localhost”?

If we look for **ACK** messages that acknowledge **CON** messages and that contain the code **Content**, we can use the filter:

```
coap.code == 69 && ip.dst == 127.0.0.1
```

and find 8 messages that satisfy the request.

4) How many messages containing the topic “factory/ department*/+” are published by a client with user name: “jane”? Where * replaces the dep. number, e.g. factory/department1/+, factory/department2/+ and so on.

First of all we can use the filter:

```
mqtt.username == "jane"
```

to highlight all the **Connect** messages sent by a client with this username.

homework1.pcapng

Filter: mqtt.username == "jane"

No.	Time	Source	Destination	Protocol	Length	Info
1554	39.395386722	127.0.0.1	127.0.0.1	MQTT	122	Connect Command
1623	39.409986893	127.0.0.1	127.0.0.1	MQTT	119	Connect Command
3768	196.693147696	127.0.0.1	127.0.0.1	MQTT	149	Connect Command
5816	122.394471312	127.0.0.1	127.0.0.1	MQTT	163	Connect Command

Frame 1554: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
 Linux cooked capture
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 Transmission Control Protocol, Src Port: 42821, Dst Port: 1883, Seq: 1, Ack: 1, Len: 54
 MQ Telemetry Transport Protocol, Connect Command

We can find the source ports of this client looking at the TCP part of the 4 messages: 42821, 40989, 40005 and 50985. Then using the following filter:

```
ip.src == 127.0.0.1 && (tcp.srcport == 42821 || tcp.srcport ==
40989 || tcp.srcport == 40005 || tcp.srcport == 50985 ) &&
mqtt.msgtype == 3 && mqtt.topic matches
"factory/department[0-9]/"
```

it's possible to find that all the 6 messages doesn't match the request because are in the form:

factory/department[0-9]/+/+

so the answer is 0.

homework1.pcapng

Filter: ip.src == 127.0.0.1 && (tcp.srcport == 42821 || tcp.srcport == 40989 || tcp.srcport == 40005 || tcp.srcport == 50985) && mqtt.msgtype == 3 && mqtt.topic matches "factory/department[0-9]"

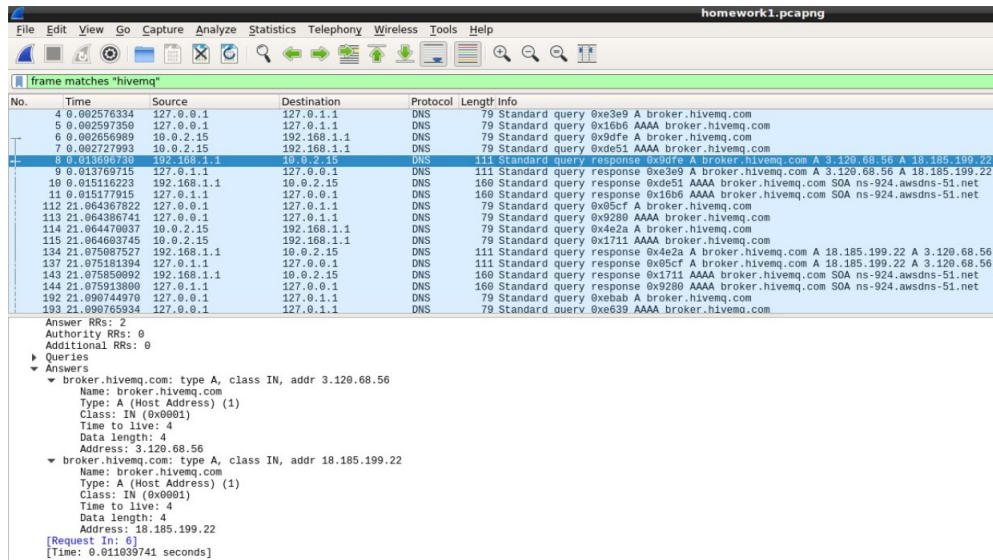
No.	Time	Source	Destination	Protocol	Length	Info
1584	39.399674378	127.0.0.1	127.0.0.1	MQTT	200	Publish Message (id=4) [factory/department2/section4/plc]
1629	39.401655935	127.0.0.1	127.0.0.1	MQTT	210	Publish Message (id=3) [factory/department2/section1/hydraulic_valve]
2540	53.418595851	127.0.0.1	127.0.0.1	MQTT	213	Publish Message (id=2) [factory/department2/section1/hydraulic_valve]
2863	78.402142281	127.0.0.1	127.0.0.1	MQTT	194	Publish Message (id=5) [factory/department2/section4/plc]
3132	98.453437832	127.0.0.1	127.0.0.1	MQTT	207	Publish Message (id=3) [factory/department2/section1/hydraulic_valve]
5836	122.397452291	127.0.0.1	127.0.0.1	MQTT	209	Publish Message (id=2) [factory/department1/section3/hydraulic_valve]

5) How many clients connected to the broker “hivemq” have specified a will message?

We can use the filter

frame matches "hivemq"

in order to highlight the IP addresses of the broker: 3.120.68.56 and 18.185.199.22.



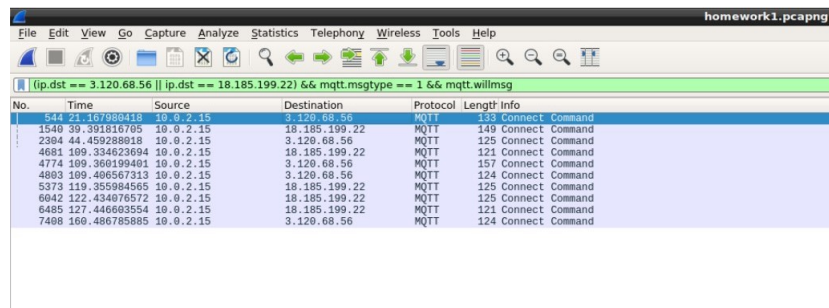
The screenshot shows a Wireshark capture of a pcapng file named 'homework1.pcapng'. The filter bar at the top is set to 'frame matches "hivemq"'. The packet list shows 193 packets, with the 8th packet selected. The packet details pane shows the selected packet is a DNS Standard query response from 3.120.68.56 to 18.185.199.22. The packet bytes pane shows the raw data of the DNS response.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.002576334	127.0.0.1	127.0.1.1	DNS	79	Standard query 0xe3e9 A broker.hivemq.com
5	0.002597350	127.0.0.1	127.0.1.1	DNS	79	Standard query 0x16b6 AAAA broker.hivemq.com
6	0.002659689	10.0.2.15	192.168.1.1	DNS	79	Standard query 0x9dfe A broker.hivemq.com
7	0.002727993	10.0.2.15	192.168.1.1	DNS	79	Standard query 0xde51 AAAA broker.hivemq.com
8	0.013696730	192.168.1.1	10.0.2.15	DNS	111	Standard query response 0x3dfe A broker.hivemq.com A 3.120.68.56 A 18.185.199.22
9	0.013709715	127.0.1.1	127.0.0.1	DNS	111	Standard query response 0xe3e9 A broker.hivemq.com A 3.120.68.56 A 18.185.199.22
10	0.015116223	192.168.1.1	10.0.2.15	DNS	160	Standard query response 0xde51 AAAA broker.hivemq.com SOA ns-924.awsdns-51.net
11	0.015177015	127.0.1.1	127.0.0.1	DNS	160	Standard query response 0x16b6 AAAA broker.hivemq.com SOA ns-924.awsdns-51.net
112	21.064367822	127.0.0.1	127.0.1.1	DNS	79	Standard query 0x85cf A broker.hivemq.com
113	21.064386741	127.0.0.1	127.0.1.1	DNS	79	Standard query 0x9280 AAAA broker.hivemq.com
114	21.064470837	10.0.2.15	192.168.1.1	DNS	79	Standard query 0x4e2a A broker.hivemq.com
115	21.064693745	10.0.2.15	192.168.1.1	DNS	79	Standard query 0x1711 AAAA broker.hivemq.com
134	21.075087527	192.168.1.1	10.0.2.15	DNS	111	Standard query response 0x4e2a A broker.hivemq.com A 18.185.199.22 A 3.120.68.56
137	21.075181394	127.0.1.1	127.0.0.1	DNS	111	Standard query response 0x95cf A broker.hivemq.com A 18.185.199.22 A 3.120.68.56
140	21.075500892	192.168.1.1	10.0.2.15	DNS	160	Standard query response 0x1711 AAAA broker.hivemq.com SOA ns-924.awsdns-51.net
144	21.075913800	127.0.1.1	127.0.0.1	DNS	160	Standard query response 0x9280 AAAA broker.hivemq.com SOA ns-924.awsdns-51.net
192	21.090744970	127.0.0.1	127.0.1.1	DNS	79	Standard query 0xebab A broker.hivemq.com
193	21.090765934	127.0.0.1	127.0.1.1	DNS	79	Standard query 0xe639 AAAA broker.hivemq.com

We can then use the filter

```
(ip.dst == 3.120.68.56 || ip.dst == 18.185.199.22) &&  
mqtt.msgtype == 1 && mqtt.willmsg
```

to retrieve all the **connect** messages directed to the broker, with the **will flag** true and that provide a **will message**.



The screenshot shows the same Wireshark capture with the filter '(ip.dst == 3.120.68.56 || ip.dst == 18.185.199.22) && mqtt.msgtype == 1 && mqtt.willmsg'. The packet list shows 10 filtered packets, all of which are MQTT Connect Command messages. The packet details pane shows the selected packet is a MQTT Connect Command from 18.185.199.22 to 3.120.68.56.

No.	Time	Source	Destination	Protocol	Length	Info
544	21.167980418	10.0.2.15	3.120.68.56	MQTT	133	Connect Command
1540	39.391816705	10.0.2.15	18.185.199.22	MQTT	149	Connect Command
2304	44.459288018	10.0.2.15	3.120.68.56	MQTT	125	Connect Command
4681	109.334623604	10.0.2.15	18.185.199.22	MQTT	121	Connect Command
4774	109.368199401	10.0.2.15	3.120.68.56	MQTT	157	Connect Command
4803	109.406567313	10.0.2.15	3.120.68.56	MQTT	124	Connect Command
5373	119.355984505	10.0.2.15	18.185.199.22	MQTT	125	Connect Command
6042	122.434076572	10.0.2.15	18.185.199.22	MQTT	125	Connect Command
6485	127.446603554	10.0.2.15	18.185.199.22	MQTT	121	Connect Command
7498	166.486785885	10.0.2.15	3.120.68.56	MQTT	124	Connect Command

There are 10 messages:

- 2 of them have an unique **Client ID** (No. 1540 and No. 4774);

- 2 of them have the same **Client ID**, but different **TCP port** (No. 4083 and No. 7408);
- the last 6 don't provide a **Client ID**, but have the **Clean Session Flag**.

So there are at least 3 different clients and at maximum 9 (10 if we consider different the clients with the same ClientID, but different TCP port).

6) How many publishes with QoS 1 don't receive the ACK?

It's possible to interpret the question in 2 different ways:

1) If we consider all the published messages with QoS 1 sent by the clients and the broker that satisfy the filter:

```
mqtt.msgtype == 3 && mqtt.qos == 1
```

are 124. The **PUBACK** are 74, due to:

```
mqtt.msgtype == 4
```

If we count also the duplicates with the filter:

```
mqtt.msgtype == 3 && mqtt.dupflag == 1
```

there are 2 messages. It's possible to find $124 - 74 - 2 = 48$ messages.

2) Otherwise if we consider only the messages sent by the clients we find that 77 of them are sent to the broker. If we use the filter:

```
mqtt.msgtype == 4
```

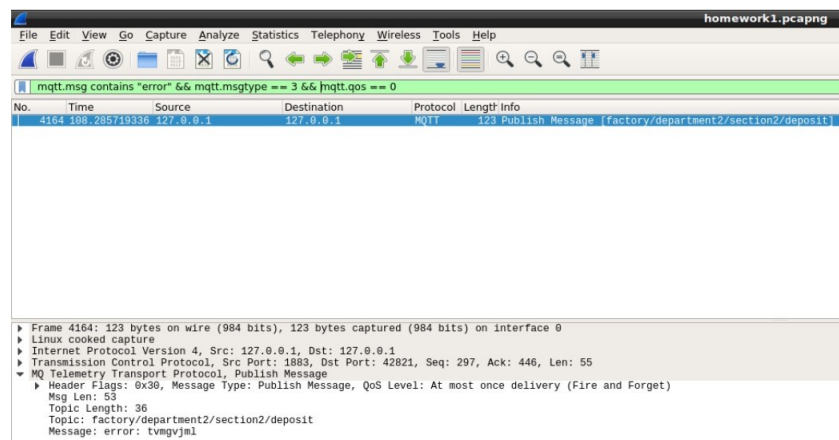
there are 74 **PUBACK** messages. So only 3 messages with QoS 1 don't receive the ack.

7) How many last will messages with QoS set to 0 are actually delivered?

We can see that all the **Last Will messages** contain the string 'error'. Then it's possible to use the filter:

```
mqtt.msg contains "error" && mqtt.msgtype == 3 && mqtt.qos == 0
```

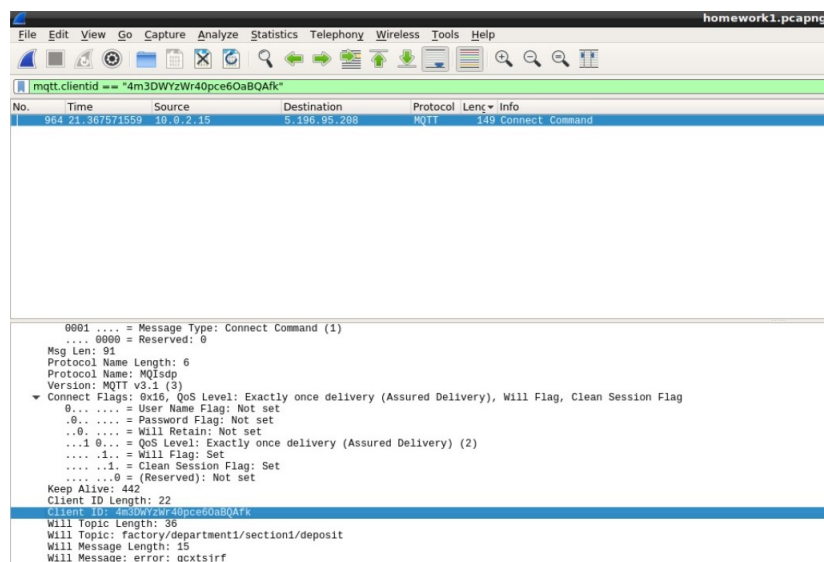
to find a single message that satisfy the query: No. 4164.



8) Are all the messages with QoS > 0 published by the client “4m3DWYzWr40pce6OaBQAfk” correctly delivered to the subscribers?
Using the filter:

`mqtt.clientid == "4m3DWYzWr40pce6OaBQAfk"`

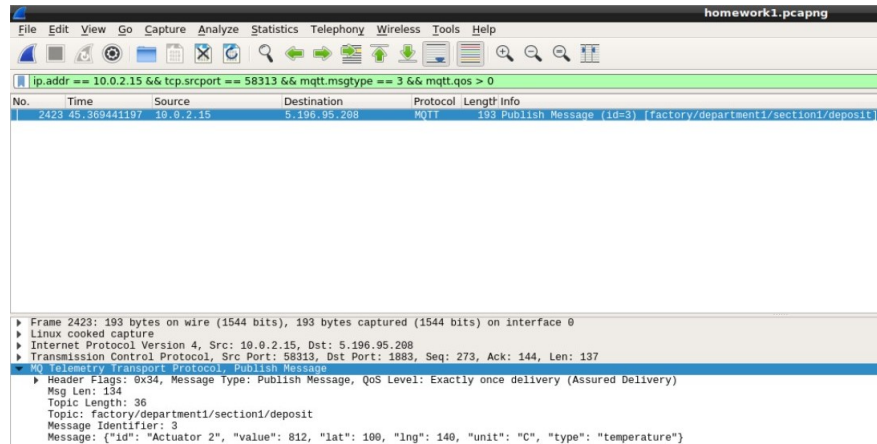
it's possible to discover the **IP address** (10.0.2.15) and the **TCP port** (58313) of the requested client.



Then, with the filter:

```
ip.addr == 10.0.2.15 && tcp.srcport == 58313 && mqtt.msgtype ==
3 && mqtt.qos > 0
```

it's possible to find a single message delivered to the broker with IP address 5.196.95.208.



Then, we can check if this message was correctly delivered by the broker to the subscribers, with the filter:

```
mqtt.msg == '{ "id": "Actuator 2", "value": 812, "lat": 100,
"lng": 140, "unit": "C", "type": "temperature"}' && ip.src ==
5.196.95.208 && mqtt.msgtype == 3
```

and find that there are no messages that satisfy the request.

9) What is the average message length of a connect msg using mqttv5 protocol? Why messages have different size?

There are 63 connect messages using **mqttv5** protocol. We can distinguish between 2 cases (message length and packets length):

Message Length: they range from 13 to 86 because some of them add more properties (User Name, Password, ClientID...) and others are malformed. The length of the majority is 13. If we calculate the average message length including also the malformed packets we obtain 30.22. If we exclude the malformed packets we obtain 16.93.

homework1.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

mqtt.ver == 5

No.	Time	Source	Destination	Protocol	Length	Info
228	21.102715180	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
359	21.134113278	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
387	21.141490460	10.0.2.15	3.120.68.56	MQTT	71	Connect Command
520	21.163928484	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
715	21.209264261	10.0.2.15	137.135.83.217	MQTT	71	Connect Command
789	21.214789505	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
854	21.255097339	10.0.2.15	137.135.83.217	MQTT	71	Connect Command
893	21.304987705	10.0.2.15	137.135.83.217	MQTT	71	Connect Command
978	21.369032844	10.0.2.15	5.196.95.208	MQTT	71	Connect Command
1106	26.178722566	10.0.2.15	5.196.95.208	MQTT	71	Connect Command
1116	26.185137882	10.0.2.15	3.120.68.56	MQTT	71	Connect Command
1134	26.195694817	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
1160	26.205818293	10.0.2.15	137.135.83.217	MQTT	71	Connect Command
1256	31.215780635	10.0.2.15	5.196.95.208	MQTT	71	Connect Command
1760	39.423552738	10.0.2.15	5.196.95.208	MQTT	71	Connect Command
1816	39.429866326	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
1950	39.457333909	10.0.2.15	18.185.199.22	MQTT	71	Connect Command
2390	44.525389430	10.0.2.15	137.135.83.217	MQTT	71	Connect Command

▼ MQTT Telemetry Transport Protocol, Connect Command

▼ Header Flags: 0x10, Message Type: Connect Command

0001 = Message Type: Connect Command (1)

.... 0000 = Reserved: 0

MQTT Length: 13

Protocol Name Length: 4

Protocol Name: MQTT

Version: MQTT v5.0 (5)

▼ Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget), Clean Session Flag

0 = User Name Flag: Not set

..0 = Password Flag: Not set

...0 = Will Retain: Not set

...0 = QoS Level: At most once delivery (Fire and Forget) (0)

.... 0.. = Will Flag: Not set

.... ..1 = Clean Session Flag: Set

.... ...0 (Reserved): Not set

Keep Alive: 554

▼ Properties

Total Length: 0

Client ID Length: 0

Client ID:

homework1.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

mqtt.ver == 5

No.	Time	Source	Destination	Protocol	Length	Info
3388	106.668445982	127.0.0.1	127.0.0.1	MQTT	102	Connect Command
4385	109.298431882	127.0.0.1	127.0.0.1	MQTT	102	Connect Command
1322	36.275328646	10.0.2.15	137.135.83.217	MQTT	123	Connect Command[Malformed Packet]
2002	39.468899958	10.0.2.15	18.185.199.22	MQTT	123	Connect Command[Malformed Packet]
2054	39.508123657	10.0.2.15	137.135.83.217	MQTT	123	Connect Command[Malformed Packet]
6083	122.443709707	10.0.2.15	3.120.68.56	MQTT	123	Connect Command[Malformed Packet]
770	21.211984200	10.0.2.15	3.120.68.56	MQTT	127	Connect Command[Malformed Packet]
879	21.291163737	10.0.2.15	137.135.83.217	MQTT	127	Connect Command[Malformed Packet]
2266	44.443554102	10.0.2.15	3.120.68.56	MQTT	127	Connect Command[Malformed Packet]
2481	49.447979967	10.0.2.15	18.185.199.22	MQTT	127	Connect Command[Malformed Packet]
3973	106.780842771	10.0.2.15	137.135.83.217	MQTT	135	Connect Command[Malformed Packet]
5461	119.448695740	10.0.2.15	137.135.83.217	MQTT	135	Connect Command[Malformed Packet]
6464	127.440668823	10.0.2.15	3.120.68.56	MQTT	135	Connect Command[Malformed Packet]
3508	106.670878457	127.0.0.1	127.0.0.1	MQTT	147	Connect Command[Malformed Packet]
4608	109.324885356	127.0.0.1	127.0.0.1	MQTT	148	Connect Command[Malformed Packet]
5916	122.405482925	127.0.0.1	127.0.0.1	MQTT	153	Connect Command[Malformed Packet]
4273	109.286902050	127.0.0.1	127.0.0.1	MQTT	156	Connect Command[Malformed Packet]

► Frame 4273: 156 bytes on wire (1248 bits), 156 bytes captured (1248 bits) on interface 0

► Linux cooked capture

► Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

► Transmission Control Protocol, Src Port: 49649, Dst Port: 1883, Seq: 1, Ack: 1, Len: 88

▼ MQTT Telemetry Transport Protocol, Connect Command

▼ Header Flags: 0x10, Message Type: Connect Command

0001 = Message Type: Connect Command (1)

.... 0000 = Reserved: 0

MQTT Length: 13

Protocol Name Length: 4

Protocol Name: MQTT

Version: MQTT v5.0 (5)

▼ Connect Flags: 0xd6, User Name Flag, Password Flag, QoS Level: Exactly once delivery (Assured Delivery), Will Flag, Clean Session Flag

1.. = User Name Flag: Set

..1 = Password Flag: Set

...0 = Will Retain: Not set

...1 0.. = QoS Level: Exactly once delivery (Assured Delivery) (2)

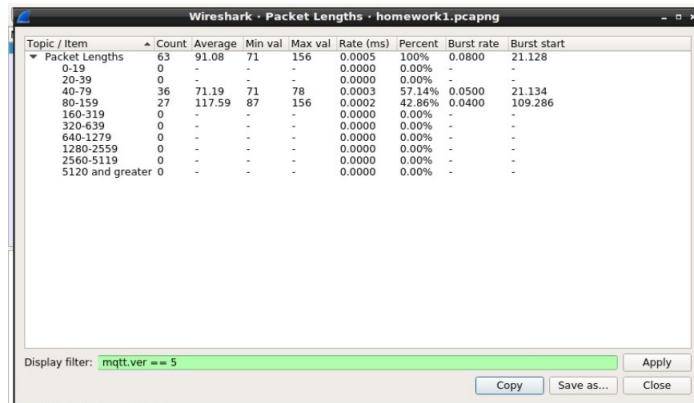
.... 1.. = Will Flag: Set

.... ..1 = Clean Session Flag: Set

.... ...0 (Reserved): Not set

Keep Alive: 211

Packet Length: if the question requests the average **packet length** it's possible to find it in the *Statistics/Packet Lengths* panel and it's equals to 91.08.



The image shows the 'Wireshark - Packet Lengths - homework1.pcapng' window. It displays a table of packet length statistics. The table has columns for Topic / Item, Count, Average, Min val, Max val, Rate (ms), Percent, Burst rate, and Burst start. The data is categorized by packet length ranges. The display filter is set to 'mqtt.ver == 5'.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	63	91.08	71	156	0.0005	100%	0.0800	21.128
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	36	71.19	71	78	0.0003	57.14%	0.0500	21.134
80-159	27	117.59	87	156	0.0002	42.86%	0.0400	109.286
160-319	0	-	-	-	0.0000	0.00%	-	-
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	0	-	-	-	0.0000	0.00%	-	-
1280-2559	0	-	-	-	0.0000	0.00%	-	-
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter: `mqtt.ver == 5` [Apply] [Copy] [Save as...] [Close]

10) Why there aren't any REQ/RESP pings in the pcap?

If there are no REQ/RESP pings it means that all the interactions between the clients and the broker occurred before the expiration of the keep alive time, or the broker disconnect the client due to the connection drop.