



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Course: Internet of Things - 5 CFU

Professor: Matteo Cesana

Students: Bruno Arena, Matteo Cavalleri

Personal IDs: 10806605, 10768765

Challenge 2 - Questions



Abstract

This section focuses on the empirical analysis of IoT network traffic captured in two PCAP files (A.pcapng and B.pcapng). The study is divided into three main areas of investigation:

- **CoAP Protocol Evaluation:** Identification of specific Non-Confirmable DELETE transactions and their success rates on the coap.me server. It further examines resource handling by quantifying Confirmable POST and PUT failures across local server resources.
- **Observe Mechanism and Redundancy:** A detailed audit of the CoAP Observe extension for the /dining_room/temperature resource, specifically filtering for unique notifications and identifying traffic inefficiencies or redundant data transmissions.
- **MQTT and MQTT-SN Broker Interaction:** An assessment of messaging patterns across local and public brokers. This includes measuring MQTT-SN traffic on port 1885, analysing subscriber behaviour regarding wildcards and Last Will messages, and identifying client attempts to clear retained values.
- **Comparative Topic Depth Analysis:** A comparative statistical study between the two captures, focusing on the structural complexity of MQTT topics (number of layers) and the overall volume of PUBLISH messages directed to the local broker.



Question #1 – Analysis of Non-Confirmable DELETE

Requests to coap.me

Answer CQ1a

CoAP messages are transported over UDP on the port 5683. A NON-Confirmable DELETE has Type 1 and code 4 (Delete). The CoAP.me server IP is 134.102.218.18.

The way used to identify a successful response is the code class. The code class for a successful response is 2.xx.

By using the following filter is possible to search every message:

Filter: `udp.port == 5683 && coap.type == 1 && coap.code == 4 && ip.dst == 134.102.218.18`

The screenshot shows a Wireshark capture of CoAP messages. The packet list displays 20 NON-Confirmable DELETE requests. The packet details for frame 11075 show the CoAP message structure:

```
> Frame 11075: Packet, 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 134.102.218.18
> User Datagram Protocol, Src Port: 44998, Dst Port: 5683
> Constrained Application Protocol, Non-Confirmable, DELETE, MID:30800
  01... .. = Type: Non-Confirmable (1)
  .... 1000 = Token Length: 8
  Code: DELETE (4)
  Message ID: 30800
  Token: 880f30d8b543d6f7
  > Opt Name: #1: Uri-Path: validate
    [Uri-Path: /validate]
```

The filter shows all the 20 NON DELETE requests.



To identify the successful response of a delete requests is used this filter:

Filter: `udp.port == 5683 && coap.code == 66 && ip.src == 134.102.218.18`

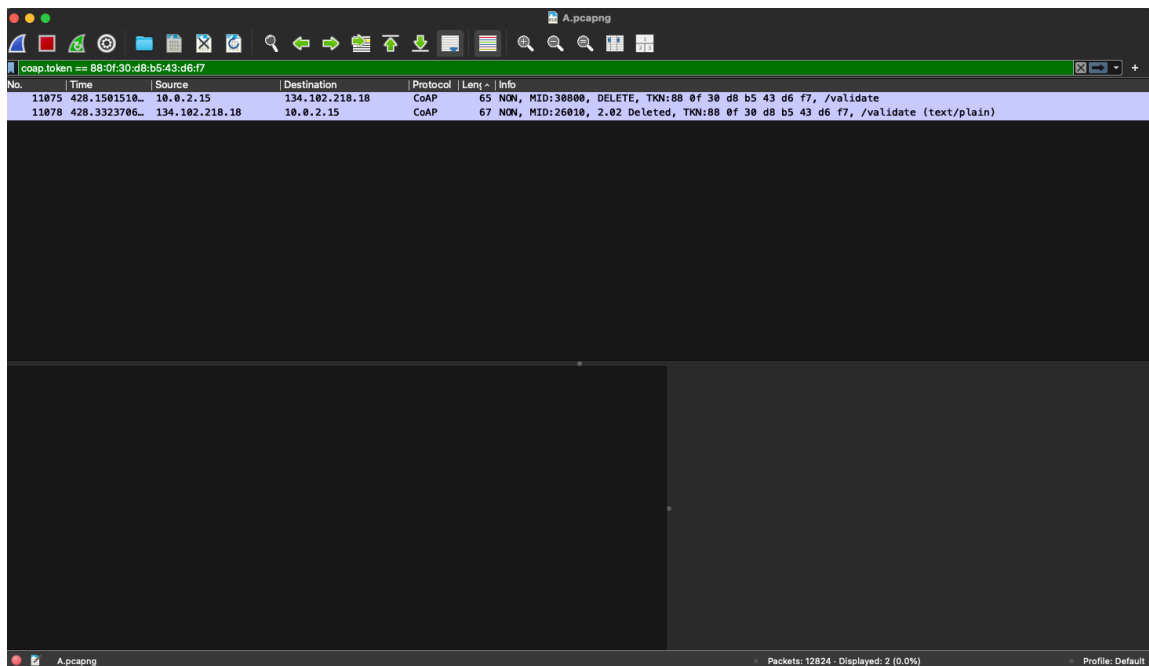
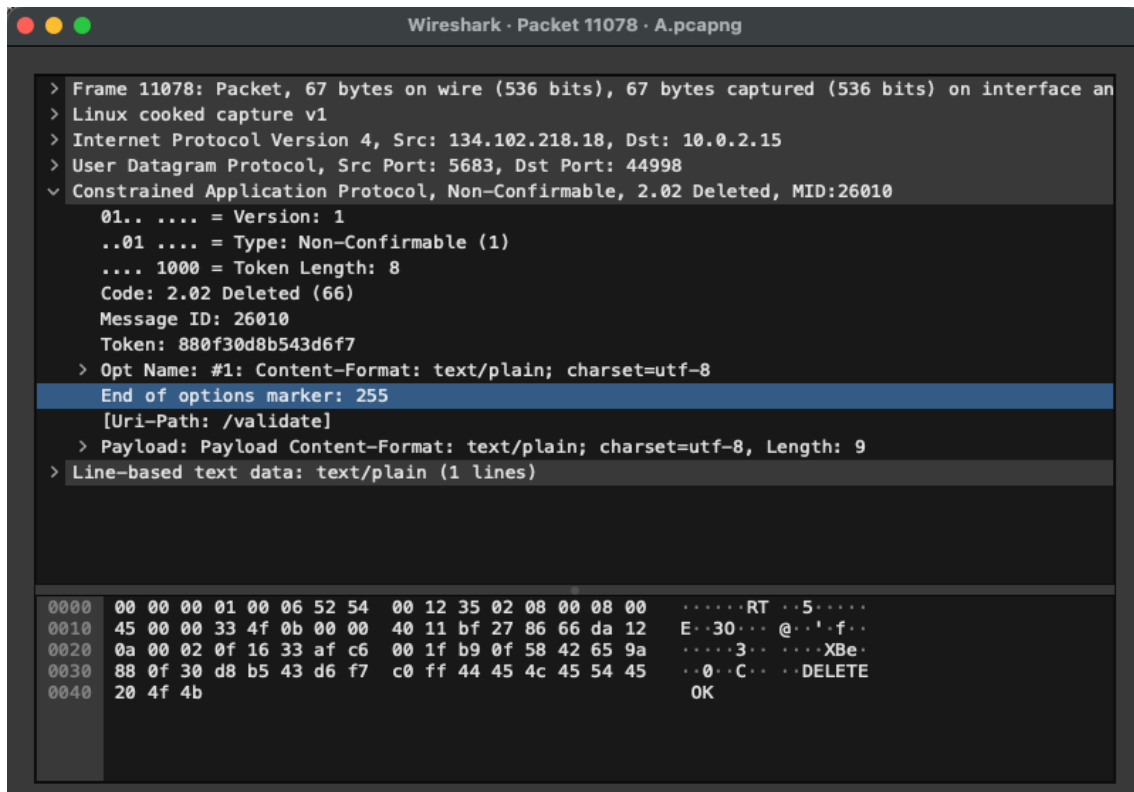
No.	Time	Source	Destination	Protocol	Length	Info
2513	126.8425082	134.102.218.18	10.0.2.15	CoAP	67	ACK, MID:24722, 2.02 Deleted, TKN:31 86 f7 11 3c a5 ab a1, /sink (text/plain)
2533	127.8028939	134.102.218.18	10.0.2.15	CoAP	67	ACK, MID:47391, 2.02 Deleted, TKN:b7 8a a8 21 b9 c6 5d 66, /validate (text/plain)
10149	369.1286711	134.102.218.18	10.0.2.15	CoAP	67	ACK, MID:5384, 2.02 Deleted, TKN:3a e7 75 df bd 67 94 1f, /sink (text/plain)
11078	428.3323786	134.102.218.18	10.0.2.15	CoAP	67	NON, MID:26010, 2.02 Deleted, TKN:88 0f 30 d8 b5 43 d6 f7, /validate (text/plain)

Frame 2513: Packet, 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface any, id 0	0000	00 00 00 01 00 06 52 54	00 12 35 02 88 b4 08 00	RT..5....
> Linux cooked capture v1	0010	45 00 00 33 3f 7a 00 00	40 11 ce 88 86 66 da 12	E..372..@...f..
> Internet Protocol Version 4, Src: 134.102.218.18, Dst: 10.0.2.15	0020	0a 00 02 0f 16 33 09 ff	00 1f b8 22 68 42 60 92	...3...hb...
> User Datagram Protocol, Src Port: 5683, Dst Port: 55807	0030	31 86 f7 11 3c a5 ab a1	c0 ff 44 45 4c 45 54 45	1...<...DELETE
> Constrained Application Protocol, Acknowledgement, 2.02 Deleted, MID:24722	0040	20 4f 4b		OK
> Line-based text data: text/plain (1 lines)				

By analysing these four requests from their token we have associated the request and the response, identifying the correct answer.



The check shows that the only request that received a successful response is the request with MID = 30800.



The request receives the 2.02 code which shows the success of the message.



Answer CQ1b

The desired outcome of a DELETE request is the resource deletion.

The confirm of the operation is verified from the request's response with code "2.02 Deleted".

All the other 20 request received a "4.05 Method Not Allowed" because the deletion is not permitted on those resources on coap.me.

Filter: udp.port == 5683 && coap.code == 66 && ip.src == 134.102.218.18

The correlation is by the Token: 880f30d8b543d6f7.

Answer CQ1

CQ1a: One request, with MID = 30800 and Token: 880f30d8b543d6f7

CQ1b: One request obtained the desired outcome, the same as CQ1a with Token: 880f30d8b543d6f7



Question #2 - Analysis of Resources with Equal Counts of Unsuccessful Confirmable POST and PUT Requests

Retransmissions are detected by deduplicating on src_ip, dst_ip, MID.

An “Unsuccessful” request means the server sent a 4.xx/5.xx response.

Malformed packets are discarded by hypothesis. Packets with token = 0 are not considered because, even if they are not malformed, is impossible to distinguish. First we used a filter to select only the CON POST:

Filter: `udp.port == 5683 && coap.type == 0 && coap.code == 2 && ip.dst == 127.0.0.1`

No.	Time	Source	Destination	Protocol	Length	Info
114	48.664376562	127.0.0.1	127.0.0.1	CoAP	61	CON, MID:64880, POST, TKN:69 73 d7 69 55 6f 2f 73, /test
3350	165.8059167	127.0.0.1	127.0.0.1	CoAP	61	CON, MID:6295, POST, TKN:30 a4 fe b1 62 5f bc 30, /test
4240	218.8775396	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:42780, POST, TKN:52 c0 9d fb de 1d 2a d0, /basic
4853	235.8857534	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:11242, POST, TKN:f5 dc 29 16 3d d5 98 e7, /basic
10264	374.9928816	127.0.0.1	127.0.0.1	CoAP	63	CON, MID:26063, POST, /hello_world
7808	285.0682998	127.0.0.1	127.0.0.1	CoAP	66	CON, MID:33536, POST, TKN:7e f5 cc 34 dc 2f 44 62, /main_door
10377	383.7587762	127.0.0.1	127.0.0.1	CoAP	66	CON, MID:58468, POST, /hello_post
2815	147.7638004	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:25749, POST, TKN:bc 2a 8c bd db 5b 1e 45, /hello_world
3858	191.8868788	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:40958, POST, TKN:2e 38 56 8b 5f 47 24 47, /dining_room
12643	484.1651421	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:29078, POST, TKN:b4 90 25 bc 56 6d 56 bd, /hello_world
12659	485.1782734	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:57914, POST, TKN:6b 04 05 67 cf 9c ec b5, /hello_world
9818	344.4524598	127.0.0.1	127.0.0.1	CoAP	75	CON, MID:40964, POST, /hello_post
10215	373.3672232	127.0.0.1	127.0.0.1	CoAP	75	CON, MID:10466, POST, /dining_room/temperature
9709	341.4239818	127.0.0.1	127.0.0.1	CoAP	76	CON, MID:40534, POST, /living_room
434	65.812049250	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:21336, POST, TKN:96 d9 65 04 cc 97 da 36, /dining_room/temperature
672	72.674048287	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:53552, POST, TKN:4a d9 d0 a5 52 78 cf ad, /dining_room/temperature
1406	88.733263512	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:11409, POST, TKN:31 7f fa e4 e0 35 28 a8, /dining_room/temperature
9997	358.9118246	127.0.0.1	127.0.0.1	CoAP	83	CON, MID:40965, POST, /hello_post
10508	391.4236556	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:9853, POST, /dining_room/door?status=OPEN
10905	415.1627373	127.0.0.1	127.0.0.1	CoAP	87	CON, MID:10179, POST, /dining_room/door?status=CLOSED
9685	339.5012381	127.0.0.1	127.0.0.1	CoAP	88	CON, MID:21082, POST, /living_room/temperature
10177	369.7027761	127.0.0.1	127.0.0.1	CoAP	94	CON, MID:10465, POST, /dining_room/temperature

And the CON PUT:

Filter: `udp.port == 5683 && coap.type == 0 && coap.code == 3 && ip.dst == 127.0.0.1`

No.	Time	Source	Destination	Protocol	Length	Info
1005	79.688944884	127.0.0.1	127.0.0.1	CoAP	62	CON, MID:50477, PUT, TKN:d3 51 13 1e c6 60 2e a3, /basic
36	18.372329518	127.0.0.1	127.0.0.1	CoAP	66	CON, MID:1330, PUT, TKN:ee a1 1e 4b c3 37 76 96, /main_door
3441	171.8457455	127.0.0.1	127.0.0.1	CoAP	66	CON, MID:30514, PUT, TKN:ed 7b 30 7a c0 d5 ac 86, /main_door
11389	448.1202484	127.0.0.1	127.0.0.1	CoAP	67	CON, MID:40939, PUT, TKN:1d 96 c0 98 9a c7 7a c9, /hello_post
8051	291.0338514	127.0.0.1	127.0.0.1	CoAP	68	CON, MID:10600, PUT, TKN:85 ef bb e6 2a 94 5b 23, /dining_room
1916	99.715712183	127.0.0.1	127.0.0.1	CoAP	73	CON, MID:13760, PUT, TKN:56 ff 18 e5 68 53 2f 2e, /dining_room/door
2492	125.7486669	127.0.0.1	127.0.0.1	CoAP	73	CON, MID:6326, PUT, TKN:e1 16 0d fc 88 62 8e e1, /living_room/door
50	23.655706843	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:48174, PUT, TKN:b5 01 7e 82 79 10 26 15, /dining_room/temperature
11454	452.1158175	127.0.0.1	127.0.0.1	CoAP	80	CON, MID:40207, PUT, TKN:68 53 b6 c9 64 cb 06 c0, /dining_room/temperature
11820	423.5964220	127.0.0.1	127.0.0.1	CoAP	84	CON, MID:37991, PUT, /dining_room/door?status=OPEN
10923	416.8056771	127.0.0.1	127.0.0.1	CoAP	87	CON, MID:10181, PUT, /dining_room/door?status=CLOSED



Looking at the CON PUT is possible to identify the unsuccessful requests:

<u>Resource</u>	<u>CON POST</u>	<u>CON PUT</u>
/basic	2	1
/main_door	1	2
/hello_post	0	1
/dining_room	1	1
/dining_room/door	0	1
/living_room/door	0	1
/dining_room/temperature	3	2

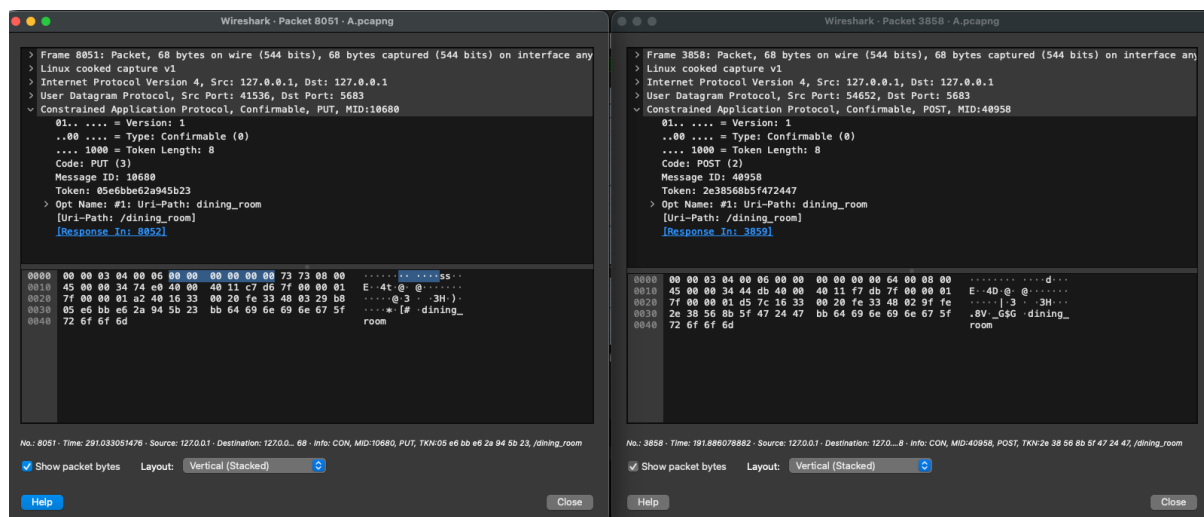


Figure 1 - Token match

Answer CO2

The only resource with X (unique CON POST unsuccessful) = Y (unique CON PUT unsuccessful) > 0 is one and is the /dining_room resource.



Question #3 - Analysis of Unique Observe Notifications and Redundancy for the /dining_room/temperature Resource

Answer CQ3a

First we locate the Observe registration to the resource /dining_room/temperature using this filter:

Filter: `udp.port == 5683 && coap.opt.observe && coap.type == 0 && coap.code == 1`

No.	Time	Source	Destination	Protocol	Length	Info
4343	214.7737947...	127.0.0.1	127.0.0.1	CoAP	77	CON, MID:10588, GET, TKN:89 b9, Block #0, /dining_room/temperature
11136	430.2786333...	127.0.0.1	127.0.0.1	CoAP	77	CON, MID:12425, GET, TKN:ie a4, Block #0, /living_room/temperature

The resource requested has Token = 89b9 so this token is used to find all the CON/NON responses with the observe option set. Retransmissions aren't considered.

Filter: `udp.port == 5683 && coap.opt.observe && coap.token == 89:b9`

No.	Time	Source	Destination	Protocol	Length	Info
4343	214.7737947...	127.0.0.1	127.0.0.1	CoAP	77	CON, MID:10588, GET, TKN:89 b9, Block #0, /dining_room/temperature
4628	222.5325668...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:35, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
5098	249.4948218...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:38, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
5623	273.8253609...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:43, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
8509	300.8208657...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:45, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
9414	326.4295778...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:49, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
9866	350.7696512...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:54, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
9937	353.6969101...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:54, 2.05 Content, TKN:89 b9, /dining_room/temperature [Retransmission], JSON (application/json)
10001	359.5520739...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:54, 2.05 Content, TKN:89 b9, /dining_room/temperature [Retransmission], JSON (application/json)
10193	371.2573234...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:54, 2.05 Content, TKN:89 b9, /dining_room/temperature [Retransmission], JSON (application/json)
10573	394.6692336...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:62, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
10631	397.4755014...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:62, 2.05 Content, TKN:89 b9, /dining_room/temperature [Retransmission], JSON (application/json)
11220	436.7615385...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:74, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
11917	467.5210646...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:87, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
12782	498.1668858...	127.0.0.1	127.0.0.1	CoAP	115	CON, MID:90, 2.05 Content, TKN:89 b9, /dining_room/temperature, JSON (application/json)
4344	214.7777771...	127.0.0.1	127.0.0.1	CoAP	117	ACK, MID:10588, 2.05 Content, TKN:89 b9, End of Block #0, /dining_room/temperature, JSON (application/json)
9867	350.7696686...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
9938	353.6969406...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
10002	359.5520884...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
10194	371.2573331...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
10574	394.6692447...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
10632	397.4755109...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
11221	436.7615613...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
11918	467.5210731...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)
12783	498.1669018...	127.0.0.1	127.0.0.1	ICMP	143	Destination unreachable (Port unreachable)



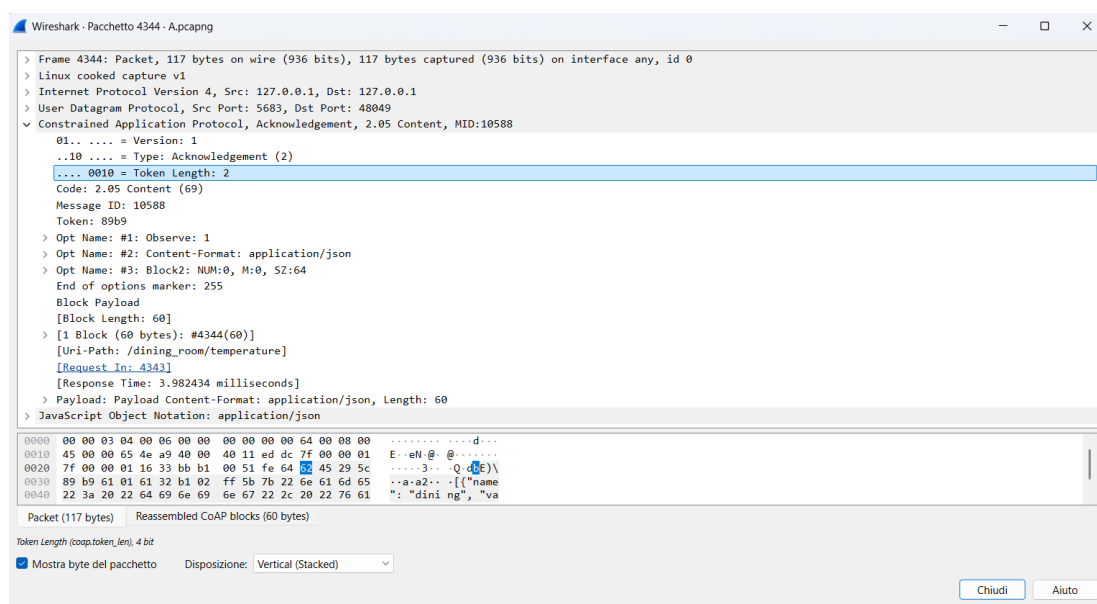
In the table are resumed all the transmissions:

<u>Number</u>	<u>MID</u>	<u>Type</u>	<u>Value</u>
1	10588	ACK	19.97
2	35	CON	22.32
3	38	CON	18.85
4	43	CON	25.98
5	45	CON	15.59
6	49	CON	25.35
7	54	CON	26.21
8	62	CON	27.53
9	74	CON	26.98
10	87	CON	25.32
11	90	CON	19.57

There are:

- 11 unique notifications
- 4 retransmissions

The ACK packet qualifies as a valid observe notification because it features a "2.05 Content" response code and includes the "Observe" option, indicating it successfully transmits the updated resource data. It is distinct from a simple acknowledgment due to its payload content, thus satisfying the criteria for separate notifications while excluding retransmissions as instructed.





Answer CQ3b

A notification is considered useless if it delivers the same value as the previous one, so there is no new information.

Comparing all the notifications, all the temperature values are distinct so there are no useless notifications.

Answer CQ3

CQ3a: *Eleven unique observe notifications*

CQ3b: *No useless notifications*



Question #4 - Analysis of MQTT-SN Message Reception from the Local Broker on Port 1885

Inspecting all 219 UDP packets on port 1885: everyone has `dst_port = 1885`, that is client to broker direction.

Filtering all UDP packets with source port 1885 (broker → client direction) yields zero results. No packet in has `srcport=1885`, confirming the broker sent no messages to any client.

Filter: `udp.srcport == 1885`

The formula counts the number of "/" characters (separators). The number of topic layers equals separators + 1. Topics with `depth=2` contain one separator (e.g., "factory/room2"), `depth=3` contain two (e.g., "hospital/area3/temperature"), etc. No single-segment or 5-segment topics were found in either capture, hence `depth=1` and `depth=5` are zero.

Answer CQ4

Zero messages received by clients from the broker



Question #5 – Last Will Delivery via Wildcard Subscription

Network Infrastructure

Broker identities were resolved by combining DNS A-record responses and link-layer analysis within the capture. One broker runs on the local IPv6 loopback interface and is not resolvable via DNS.

IP Address / Address	Hostname	Role
:::1 (IPv6 loopback)	—	Local Mosquitto broker (same host)
18.192.151.104	broker.hivemq.com	Public HiveMQ broker
35.158.43.69	broker.hivemq.com	Public HiveMQ broker (DNS round-robin)
5.196.78.28	test.mosquitto.org	Public Mosquitto broker
10.0.2.15	—	Client machine (VirtualBox NAT)

MQTT Clients Identified

Forty-six CONNECT packets were found across all streams. Twenty-one of these originate from automated Mosquitto subscriber instances (client IDs prefixed mosq-, length 23) connecting to the local broker; they are excluded from the table below as they carry no Will and issue no SUBSCRIBE relevant to this analysis. The remaining 24 named clients are listed below.

Client Identifier	Len	Broker	MQTT v	Will ?	Will Topic
pcbtjyj	7	HiveMQ 35.158.43.69	3.1	No	—
cpoejpzkhixgjiu	16	HiveMQ 18.192.151.104	3.1	No	—
tukvxesuhe	10	HiveMQ 18.192.151.104	3.1	No	—
fethvjikxjul	12	HiveMQ 18.192.151.104	3.1	No	—
giuxfijwus	10	HiveMQ 18.192.151.104	3.1.1	No	—



atqeblw	7	HiveMQ 18.192.151.10 4	3.1.1	No	—
pohujuv	7	HiveMQ 18.192.151.10 4	5.0	No	—
dzcxnwdqef	10	HiveMQ 18.192.151.10 4	5.0	No	—
rolsmcwhshlbc	13	HiveMQ 18.192.151.10 4	5.0	No	—
wikdxsrbyewyocm	15	Mosquitto 5.196.78.28	3.1.1	Yes	metaverse/room2/floor4
lstvfjwz	8	Mosquitto 5.196.78.28	5.0 ⚠	Yes	hospital/facility3/area3
kiurrrnrgj	10	Mosquitto 5.196.78.28	3.1.1	Yes	metaverse/room2/room2
(empty)	0	Local ::1	3.1.1	Yes	university/department12/room1/temperat ure
auyvhrhdudnm	12	Local ::1	3.1.1	No	—
pcdwkgeslfh	11	Local ::1	3.1	No	—
yfyxj	6	Local ::1	3.1.1	No	—
ggmqfceaplfl	12	Local ::1	3.1	No	—
sjinzrhqg	9	Local ::1	3.1	No	—
fixstgpixtt	11	Local ::1	3.1	No	—
zmjnxudohrkaegm h	16	Local ::1	5.0 ⚠	No	—
frfltwzmz	9	Local ::1	3.1.1	No	—
mjdcmjxt	9	Local ::1	3.1.1	No	—
lgmfxhwoa	10	Local ::1	5.0	No	—
ukikif	6	Local ::1	3.1	No	—

(⚠) Client `lstvfjwz` and client `zmjnxudohrkaegmh` use MQTT 5.0. Their frames include an additional **Properties Length** field not present in v3.1.1, which causes offset errors in v3.1.1-only parsers. This is discussed in detail under **Parsing Considerations**.



For a subscriber to receive a Last Will and Testament (LWT) message through a wildcard subscription, three conditions must hold simultaneously:

- A client must have registered a Will Topic in its CONNECT packet (`willflag == 1`).
- Another client must have issued a SUBSCRIBE whose topic filter contains at least one wildcard (+ or #) on the same broker.
- The wildcard filter must match the Will Topic according to MQTT rules: + matches exactly one topic level; # matches one or more remaining levels and must appear last in the filter.

All three conditions must be satisfied on the same broker. A will registered on broker A is never delivered to subscribers of broker B: MQTT brokers are independent and do not bridge messages between each other.

Wireshark Filters

Step 1 — Collect Will Topics

```
mqtt.msgtype == 1 && mqtt.confmsg.willflag == 1
```

This filter is intentionally left without an IP-version restriction. The local broker runs on IPv6 loopback (::1), so constraining to `ip.version == 4` would exclude the CONNECT packet carrying the will topic `university/department12/room1/temperature`, which travels over IPv6.

mqtt.msgtype == 1 && mqtt.confmsg.willflag == 1						
No.	Time	Source	Destination	Protocol	Length	Info
4	0.000117188	::1	::1	MQTT	176	Connect Command
196	2.116585177	10.0.2.15	5.196.78.28	MQTT	126	Connect Command
352	5.034840089	10.0.2.15	5.196.78.28	MQTT	123	Connect Command
557	7.043177949	10.0.2.15	5.196.78.28	MQTT	120	Connect Command



```
▶ Frame 4: 176 bytes on wire (1408 bits), 176 bytes captured (1408 bits) on interface any, id 0
▶ Linux cooked capture
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▼ Transmission Control Protocol, Src Port: 38083, Dst Port: 1883, Seq: 1, Ack: 1, Len: 88
  Source Port: 38083
  Destination Port: 1883
  [Stream index: 0]
  [TCP Segment Len: 88]
  Sequence number: 1 (relative sequence number)
  Sequence number (raw): 73159594
  [Next sequence number: 89 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  Acknowledgment number (raw): 1785466748
  1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x018 (PSH, ACK)
  Window size value: 512
  [Calculated window size: 65536]
  [Window size scaling factor: 128]
  Checksum: 0x0080 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
  TCP payload (88 bytes)
  [PDU Size: 88]
▼ MQ Telemetry Transport Protocol, Connect Command
  ▶ Header Flags: 0x10, Message Type: Connect Command
  Msg Len: 86
  Protocol Name Length: 4
  Protocol Name: MQTT
  Version: MQTT v3.1.1 (4)
  ▶ Connect Flags: 0x0e, QoS Level: At least once delivery (Acknowledged deliver), Will Flag, Clean Session Flag
  Keep Alive: 60
  Client ID Length: 0
  Client ID:
  Will Topic Length: 41
  Will Topic: university/department12/room1/temperature
  Will Message Length: 29
  Will Message: 6572726f723a20612056495020436c69656e74206a757374...
```

Step 2 — Collect Wildcard Subscriptions on the local broker

```
mqtt.msgtype == 8 && ipv6.dst == ::1
```

From the results, only topic filters containing + or # characters are retained.

Wireshark does not expose a wildcard count field; the count is evaluated programmatically on the parsed filter strings.



B.pcapng						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
mqtt.msgtype == 8 && ipv6.dst == ::1						
No.	Time	Source	Destination	Protocol	Length	Info
121	1.083118347	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/tempe..
154	2.082593293	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/tempe..
155	2.082587507	::1	::1	MQTT	135	Subscribe Request (id=1) [metaverse/department3/section0/poll..
162	2.083847671	::1	::1	MQTT	124	Subscribe Request (id=1) [house/room0/+//hydraulic_valve]
226	3.087146589	::1	::1	MQTT	117	Subscribe Request (id=1) [university/+//deposit]
228	3.087240982	::1	::1	MQTT	118	Subscribe Request (id=1) [university/fxlugs/room1]
230	3.087766995	::1	::1	MQTT	106	Subscribe Request (id=2) [metaverse/#]
232	3.087868409	::1	::1	MQTT	115	Subscribe Request (id=1) [house/department3/+]
300	4.096365250	::1	::1	MQTT	127	Subscribe Request (id=2) [hospital/+//section0/temperature]
304	4.097040463	::1	::1	MQTT	136	Subscribe Request (id=1) [university/department12/room1/tempe..
312	4.103392541	::1	::1	MQTT	127	Subscribe Request (id=2) [university/department3]
314	4.103500816	::1	::1	MQTT	114	Subscribe Request (id=2) [house/department3/+]
316	4.103557104	::1	::1	MQTT	110	Subscribe Request (id=2) [house/building2]
356	5.089853407	::1	::1	MQTT	115	Subscribe Request (id=3) [university/facility0]
362	5.106595370	::1	::1	MQTT	116	Subscribe Request (id=3) [metaverse/department3]
363	5.106636971	::1	::1	MQTT	113	Subscribe Request (id=3) [hospital/+//light]
460	6.092938713	::1	::1	MQTT	114	Subscribe Request (id=4) [metaverse/+//room1/#]
473	6.099235260	::1	::1	MQTT	117	Subscribe Request (id=3) [house/department3/+//+]
488	6.106750118	::1	::1	MQTT	120	Subscribe Request (id=3) [factory/department3/room1]
490	6.106824619	::1	::1	MQTT	117	Subscribe Request (id=4) [university/facility0/#]
495	6.106849816	::1	::1	MQTT	112	Subscribe Request (id=4) [house/facility0/+]
563	7.094231915	::1	::1	MQTT	127	Subscribe Request (id=5) [metaverse/department3/section0/#]
566	7.100800848	::1	::1	MQTT	112	Subscribe Request (id=4) [metaverse/fxlugs]
569	7.102517118	::1	::1	MQTT	112	Subscribe Request (id=1) [house/room0/area0]
573	7.107008608	::1	::1	MQTT	112	Subscribe Request (id=4) [metaverse/+//area0]
574	7.107052527	::1	::1	MQTT	132	Subscribe Request (id=5) [university/room0/section0/temperatu..
626	8.095348548	::1	::1	MQTT	111	Subscribe Request (id=6) [factory/fxlugs/+]
629	8.109974300	::1	::1	MQTT	128	Subscribe Request (id=5) [factory/fxlugs/section0/pollution]
630	8.109139506	::1	::1	MQTT	112	Subscribe Request (id=6) [house/department3]
631	8.109532963	::1	::1	MQTT	110	Subscribe Request (id=5) [hospital/fxlugs]
686	9.096122375	::1	::1	MQTT	120	Subscribe Request (id=7) [factory/building2/room1/+]
689	9.101062415	::1	::1	MQTT	113	Subscribe Request (id=5) [factory/building2]
694	9.103623384	::1	::1	MQTT	114	Subscribe Request (id=2) [factory/department3]
700	9.110386346	::1	::1	MQTT	118	Subscribe Request (id=6) [metaverse/facility0/+//+]
1136	10.102492445	::1	::1	MQTT	108	Subscribe Request (id=6) [university/#]
1174	10.110242880	::1	::1	MQTT	121	Subscribe Request (id=7) [house/room0/area0/humidity]
1177	10.112498473	::1	::1	MQTT	114	Subscribe Request (id=7) [hospital/fxlugs/+//#]
1178	10.112541241	::1	::1	MQTT	106	Subscribe Request (id=6) [metaverse/#]
1773	11.101273432	::1	::1	MQTT	110	Subscribe Request (id=8) [house/building2]
1776	11.103183884	::1	::1	MQTT	125	Subscribe Request (id=7) [factory/fxlugs/floor0/deposit]
1786	11.111286779	::1	::1	MQTT	110	Subscribe Request (id=3) [factory/room0/+]

Step 3 — Cross-check broker and topic matching

Each wildcard filter is evaluated against each Will Topic using the MQTT topic matching algorithm, considering only filters registered on the same broker as the corresponding will.

Parsing Considerations — MQTT v5 Compatibility

Two clients in the capture use MQTT 5.0: lstvfjwz (CONNECT to 5.196.78.28) and zmjnxudohrkaegmh (CONNECT and SUBSCRIBE to ::1). In MQTT 5.0, the CONNECT and SUBSCRIBE packets include a Properties section that is absent in v3.1.1. Even when no properties are set, a one-byte Properties Length field (0x00) is always present, inserted immediately after the Keepalive in CONNECT and after the Message ID in SUBSCRIBE.

A parser written exclusively for MQTT v3.1.1 reads this extra 0x00 byte as the high byte of the next length field (Client ID length in CONNECT, Topic Filter length in SUBSCRIBE). This shifts all subsequent field offsets by one byte, producing:



- In CONNECT: Client ID length reads as 0 (empty), then the actual Client ID bytes are misinterpreted as the Will Topic length, producing a value far beyond the remaining payload. The will topic is never read.
- In SUBSCRIBE: Topic Filter length reads as 0 (empty topic), and the following byte is misread as QoS. Both outcomes (empty topic, out-of-range QoS such as 15 or 21) are syntactically legal in MQTT, so no error is raised — the parser silently discards the data and moves on.

The fix requires detecting the Protocol Level byte in each client's CONNECT (0x04 for v3.1.1, 0x05 for v5.0) and applying the appropriate parsing logic for all subsequent packets on that TCP flow. Wireshark handles this automatically; a custom parser must do it explicitly. Without this correction, the will topic of `lstvfjwz` and the topic filters of `zmjnxudohrkaegmh` — including the decisive `university/#` — would not be recovered.

Results

Four Will Topics were registered across the capture:

Client	Broker	Will Topic
wikdxsrbyewyocm	test.mosquitto.org (5.196.78.28)	metaverse/room2/floor4
lstvfjwz (MQTT 5.0)	test.mosquitto.org (5.196.78.28)	hospital/facility3/area3
kiurrrnrgj	test.mosquitto.org (5.196.78.28)	metaverse/room2/room2
(empty) ← key	Local broker ::1	university/department12/room1/temperature

For the three Will Topics on test.mosquitto.org (5.196.78.28), no SUBSCRIBE message directed to that broker appears in the capture: the three connected clients exclusively sent CONNECT and PUBLISH packets. Because brokers are independent, any subscriber on a different broker cannot receive those wills regardless of filter content.

For the Will Topic on the local broker ::1, the wildcard subscriptions registered on the same broker were evaluated. The relevant match is:



Subscriber	Topic Filter	Will Topic	Match?
zmjnxudohrkaegmh (MQTT 5.0)	university/#	university/department12/room1/temperature	YES
(other local subscribers)	university/+/+/deposit, university/department3/#, etc.	university/department12/room1/temperature	No — level mismatch or final level mismatch

The filter `university/#` matches because the `#` wildcard covers any number of remaining topic levels. The topic `university/department12/room1/temperature` starts with `university/` and has further levels — the condition is satisfied. All other wildcard filters on the local broker fail due to a fixed-level mismatch in the second or last level (e.g., `university/department3/#` does not match `department12`; `university/+/+/deposit` does not match the last level `temperature`).

The subscription `university/#` was only recovered after applying the correct MQTT v5 parser to the frames from client `zmjnxudohrkaegmh`. With a v3.1.1 parser, all topic filters from this client appear as empty strings and are silently dropped (see Parsing Considerations above).

Answer to CQ5: 1



Question #6 – Clients Erasing Retained Values on HiveMQ

In MQTT, a retained message is stored by the broker and delivered immediately to any future subscriber on that topic. The only way to erase a retained value is to publish to that exact topic with the RETAIN flag set to 1 and a zero-byte payload. The broker then deletes the stored message.

The analysis identifies PUBLISH packets satisfying all the following:

- MQTT message type = 3 (PUBLISH)
- RETAIN flag = 1
- Payload length = 0 bytes (mqtt.len == 0)
- Destination IP is a HiveMQ broker (18.192.151.104 or 35.158.43.69)
- Direction is Client to Broker (ip.dst is the broker address)

Following the instruction to ignore retransmissions, each packet is deduplicated based on its (TCP stream, sequence number) pair. In this capture, no TCP retransmissions were detected among the erase packets: the count before and after deduplication is identical (50 packets in both cases, 0 removed). Each stream is associated with a client identifier by matching the source port and broker IP to the CONNECT packet seen on the same TCP connection.

Wireshark filters

Erase PUBLISH packets — unique only

```
mqtt.msgtype == 3 && mqtt.retain == 1 && mqtt.len == 0  
&& (ip.dst == 18.192.151.104 || ip.dst == 35.158.43.69)  
&& ip.version == 4 && !tcp.analysis.retransmission
```

CONNECT packets to HiveMQ — to retrieve client identifiers

```
mqtt.msgtype == 1 && ip.version == 4  
&& (ip.dst == 18.192.151.104 || ip.dst == 35.158.43.69)
```



B.pcapng						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
[mqtt.msgtype == 1 && ip.version == 4 && (ip.dst == 18.192.151.104 ip.dst == 35.158.43.69)]						
No.	Time	Source	Destination	Protocol	Length	Info
19	0.030931634	10.0.2.15	35.158.43.69	MQTT	79	Connect Command
43	0.111075257	10.0.2.15	18.192.151.104	MQTT	88	Connect Command
130	1.107593686	10.0.2.15	18.192.151.104	MQTT	81	Connect Command
335	4.147888260	10.0.2.15	18.192.151.104	MQTT	82	Connect Command
507	6.122099094	10.0.2.15	18.192.151.104	MQTT	84	Connect Command
553	7.033325651	10.0.2.15	18.192.151.104	MQTT	80	Connect Command
622	8.044595049	10.0.2.15	18.192.151.104	MQTT	78	Connect Command
1747	11.070681277	10.0.2.15	18.192.151.104	MQTT	77	Connect Command
1752	11.072194429	10.0.2.15	18.192.151.104	MQTT	84	Connect Command

CQ6b — client identifier length filter

mqtt.msgtype == 1 && mqtt.clientid_len > 7 && ip.version == 4
&& (ip.dst == 18.192.151.104 || ip.dst == 35.158.43.69)

Results — CQ6a

Fifty unique erase-publish packets were found, sent by three distinct client identifiers:

Client Identifier	Broker	Erase Messages Sent
pcbtjyj	HiveMQ 35.158.43.69	27
atqeblw	HiveMQ 18.192.151.104	12
giuxfjwus	HiveMQ 18.192.151.104	11

Answer to CQ6a: 3

Results — CQ6b

The three clients from CQ6a are evaluated against the strict length threshold (> 7, i.e., at least 8 bytes):

Client Identifier	Length (bytes)	Strictly > 7?
pcbtjyj	7	No — equal to 7, threshold requires strictly greater
atqeblw	7	No — equal to 7, threshold requires strictly greater
giuxfjwus	10	Yes



Only giuxfijwus, with a 10-byte identifier, satisfies the strict inequality. Both pcbtjyj and atqeblw are exactly 7 bytes and therefore do not qualify.

```
MQ Telemetry Transport Protocol, Connect Command
  Header Flags: 0x10, Message Type: Connect Command
  Msg Len: 22
  Protocol Name Length: 4
  Protocol Name: MQTT
  Version: MQTT v3.1.1 (4)
  Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget), Clean Session Flag
  Keep Alive: 52
  Client ID Length: 10
  Client ID: giuxfijwus
```

Answer to CQ6b: 1



Question #7 – SUBSCRIBE Requests to the Local Broker with At Least Two Wildcards

The local broker in this capture is the Mosquitto instance running on the IPv6 loopback address `::1` (TCP port 1883). A topic filter with at least two wildcards is one where the combined count of `+` and `#` characters is greater than or equal to 2 (e.g., `hospital/+/+/light` or `metaverse/+/room1/#`). The `#` wildcard, if present, must appear as the last level.

Two conditions must hold: the SUBSCRIBE must be addressed to the local broker (`::1`), and at least one topic filter within that SUBSCRIBE must contain two or more wildcard characters.

Wireshark Filters

SUBSCRIBE packets to the local broker with at least two wildcards per filter string

```
mqtt.msgtype == 8 && ipv6.dst == ::1
```

The filter uses `ipv6.dst == ::1` rather than `ip.dst` because the local broker is reachable only over IPv6 loopback. A filter restricted to IPv4 (`ip.version == 4`) would return zero results and incorrectly suggest no SUBSCRIBE packets were sent to the local broker. Wireshark does not expose a wildcard count field; an additional check must be performed programmatically on the parsed filter strings: only filters where $\text{count}('+') + \text{count}(\#) \geq 2$ are retained.



B.pcapng					
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help					
mqtt.msgtype == 8 && ipv6.dst == ::1					
No.	Time	Source	Destination	Protocol	Length Info
121	1.083118347	:::1	:::1	MQTT	136 Subscribe Request (id=1) [university/department12/room1/tempe..
154	2.082593293	:::1	:::1	MQTT	136 Subscribe Request (id=1) [university/department12/room1/tempe..
155	2.082587507	:::1	:::1	MQTT	135 Subscribe Request (id=1) [metaverse/department3/section0/poll..
162	2.083847671	:::1	:::1	MQTT	124 Subscribe Request (id=1) [house/room0/+/hydraulic_valve]
226	3.087146589	:::1	:::1	MQTT	117 Subscribe Request (id=1) [university/+/+/deposit]
228	3.087240982	:::1	:::1	MQTT	118 Subscribe Request (id=1) [university/fxlugs/room1]
230	3.087766995	:::1	:::1	MQTT	106 Subscribe Request (id=2) [metaverse/#]
232	3.087868409	:::1	:::1	MQTT	115 Subscribe Request (id=1) [house/department3/+]
300	4.096365250	:::1	:::1	MQTT	127 Subscribe Request (id=2) [hospital/+/section0/temperature]
304	4.097849463	:::1	:::1	MQTT	136 Subscribe Request (id=1) [university/department12/room1/tempe..
312	4.103392541	:::1	:::1	MQTT	117 Subscribe Request (id=2) [university/department3]
314	4.103500816	:::1	:::1	MQTT	114 Subscribe Request (id=2) [house/department3/+]
316	4.103557104	:::1	:::1	MQTT	110 Subscribe Request (id=2) [house/building2]
356	5.089853407	:::1	:::1	MQTT	115 Subscribe Request (id=3) [university/facility0]
362	5.106595370	:::1	:::1	MQTT	116 Subscribe Request (id=3) [metaverse/department3]
363	5.106636971	:::1	:::1	MQTT	113 Subscribe Request (id=3) [hospital/+/+/light]
460	6.092938713	:::1	:::1	MQTT	114 Subscribe Request (id=4) [metaverse/+/room1/#]
473	6.099235260	:::1	:::1	MQTT	117 Subscribe Request (id=3) [house/department3/+/+]
488	6.106750118	:::1	:::1	MQTT	120 Subscribe Request (id=3) [factory/department3/room1]
490	6.106824619	:::1	:::1	MQTT	117 Subscribe Request (id=4) [university/facility0/#]
495	6.106849816	:::1	:::1	MQTT	112 Subscribe Request (id=4) [house/facility0/+]
563	7.094231915	:::1	:::1	MQTT	127 Subscribe Request (id=5) [metaverse/department3/section0/#]
566	7.100808048	:::1	:::1	MQTT	112 Subscribe Request (id=4) [metaverse/fxlugs]
569	7.102517118	:::1	:::1	MQTT	112 Subscribe Request (id=1) [house/room0/area0]
573	7.107008608	:::1	:::1	MQTT	112 Subscribe Request (id=4) [metaverse/+/area0]
574	7.107052527	:::1	:::1	MQTT	132 Subscribe Request (id=5) [university/room0/section0/temperatu..
626	8.095348548	:::1	:::1	MQTT	111 Subscribe Request (id=6) [factory/fxlugs/+]
629	8.109074300	:::1	:::1	MQTT	128 Subscribe Request (id=5) [factory/fxlugs/section0/pollution]
630	8.109139506	:::1	:::1	MQTT	112 Subscribe Request (id=6) [house/department3]
631	8.109532963	:::1	:::1	MQTT	110 Subscribe Request (id=5) [hospital/fxlugs]
686	9.096122375	:::1	:::1	MQTT	120 Subscribe Request (id=7) [factory/building2/room1/+]
689	9.101062415	:::1	:::1	MQTT	113 Subscribe Request (id=5) [factory/building2]
694	9.103623384	:::1	:::1	MQTT	114 Subscribe Request (id=2) [factory/department3]
700	9.110386346	:::1	:::1	MQTT	118 Subscribe Request (id=6) [metaverse/facility0/+/+]
1136	10.102492445	:::1	:::1	MQTT	108 Subscribe Request (id=6) [university/#]
1174	10.110242880	:::1	:::1	MQTT	121 Subscribe Request (id=7) [house/room0/area0/humidity]
1177	10.112498473	:::1	:::1	MQTT	114 Subscribe Request (id=7) [hospital/fxlugs/+/#]
1178	10.112541241	:::1	:::1	MQTT	106 Subscribe Request (id=6) [metaverse/#]
1773	11.101273432	:::1	:::1	MQTT	110 Subscribe Request (id=8) [house/building2]
1776	11.103183884	:::1	:::1	MQTT	125 Subscribe Request (id=7) [factory/fxlugs/floor0/deposit]
1786	11.111286779	:::1	:::1	MQTT	110 Subscribe Request (id=3) [factory/room0/+]

Note on MQTT v5: the 21 mosq- automated subscriber instances and client zmjnxudohrkaegmh all connect to the local broker. The mosq- clients issue no SUBSCRIBE and are irrelevant to this question. Client zmjnxudohrkaegmh uses MQTT 5.0 and its SUBSCRIBE frames include the extra Properties Length byte described in the Parsing Considerations section of CQ5. Without the corrected parser, the two multi-wildcard filters from this client are invisible.

Results

Eighty-three SUBSCRIBE messages were found directed to the local broker ::1, each carrying exactly one topic filter. Nine topic filters contain two or more wildcard characters:

#	Topic Filter	Wildcards	Subscriber Client	MQTT
1	university/+/+/deposit	+, +	(port :44613)	3.1.1
2	hospital/+/+/light	+, +	(port :43797)	3.1.1
3	metaverse/+/room1/#	+, #	(port :42777)	3.1.1



4	metaverse/facility0/+/+	+, +	(port :53121)	3.1.1
5	hospital/fxlugs/+/#	+, #	(port :53121)	3.1.1
6	metaverse/+/+/hydraulic_valve	+, +	(port :53121)	3.1.1
7	factory/+/#	+, #	(port :43797)	3.1.1
8	house/department3/+/+	+, +	zmjnxudohrkaegmh (port :51743)	5.0 ⚠
9	house/+/room1/#	+, #	zmjnxudohrkaegmh (port :51743)	5.0 ⚠

Entries 8 and 9 were recovered only after applying the correct MQTT v5 parser to client zmjnxudohrkaegmh. With a v3.1.1 parser, this client's Properties Length byte (0x00) is absorbed into the Topic Filter length field, yielding length = 0 and an empty topic string, followed by the next byte being misread as QoS (obtaining out-of-range values such as 15 or 21). Both outcomes are syntactically valid in MQTT, so the parser discards the entries silently. The actual filters `house/department3/+/+` and `house/+/room1/#` — each containing two wildcard characters — are therefore invisible without version-aware parsing.

Answer to CQ7: 9



Question #8 – MQTT PUBLISH

First, we apply the following filter to search for base packets:

Filter: tcp.port == 1883 && ip.dst == 127.0.0.1 (A.pcapng)

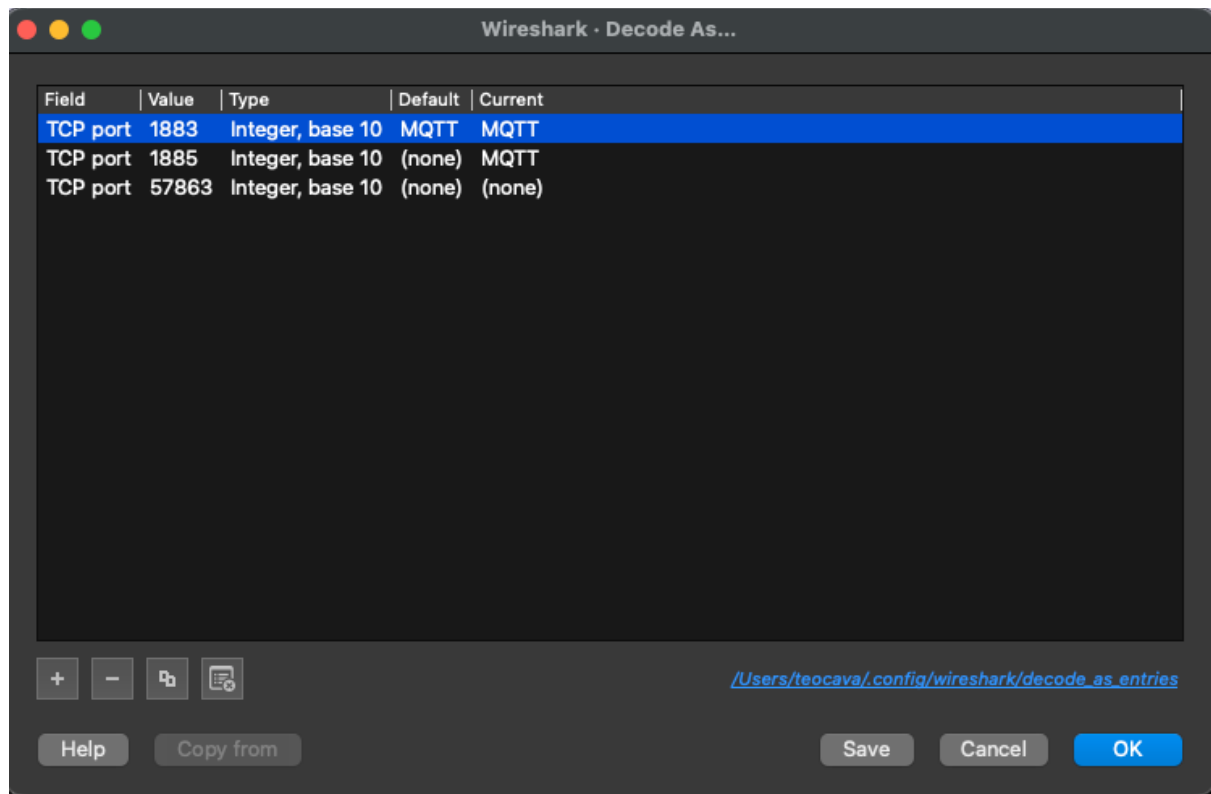
Filter: tcp.port == 1883 && ipv6.dst == ::1 (B.pcapng)

tcp.port == 1883 && ip.dst == 127.0.0.1									
No.	Time	Source	Destination	Protocol	Length	Info			
2677	130.9328988...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=161 Ack=4546 Win=65536 Len=4	TSval=749415891 TSecr=749413869	
2685	130.9395643...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=165 Ack=4546 Win=65536 Len=4	TSval=749416798 TSecr=749415891	
2737	140.8864376...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=169 Ack=4675 Win=65536 Len=4	TSval=749417845 TSecr=749417521	
869	76.979923308...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=17 Ack=635 Win=65536 Len=4	TSval=749353939 TSecr=749353939	
2771	142.8634132...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=173 Ack=4810 Win=65536 Len=4	TSval=749419822 TSecr=749419520	
2773	142.8634398...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=177 Ack=4810 Win=65536 Len=4	TSval=749419822 TSecr=749419822	
2775	142.8634644...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=181 Ack=4810 Win=65536 Len=4	TSval=749419822 TSecr=749419822	
2889	146.8293604...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=185 Ack=4959 Win=65536 Len=4	TSval=749423787 TSecr=749428261	
2888	150.8911846...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=189 Ack=5260 Win=65536 Len=4	TSval=749427849 TSecr=749427849	
2974	156.8424781...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=193 Ack=5397 Win=65536 Len=4	TSval=749433800 TSecr=749432320	
3108	157.8481893...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=197 Ack=5397 Win=65536 Len=4	TSval=749434806 TSecr=749433800	
3144	159.8213732...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=201 Ack=5556 Win=65536 Len=4	TSval=749436779 TSecr=749434935	
3189	160.8130899...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=205 Ack=5782 Win=65536 Len=4	TSval=749437772 TSecr=749437772	
3259	161.9188798...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=209 Ack=5782 Win=65536 Len=4	TSval=749438876 TSecr=749437772	
923	78.293581478...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=21 Ack=924 Win=65536 Len=4	TSval=749355253 TSecr=749355252	
3332	163.9048213...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=213 Ack=5852 Win=65536 Len=4	TSval=749440863 TSecr=749440461	
3334	163.9048365...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=217 Ack=5852 Win=65536 Len=4	TSval=749440863 TSecr=749440863	
3336	163.9048499...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=221 Ack=5852 Win=65536 Len=4	TSval=749440863 TSecr=749440863	
3417	167.8640689...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=225 Ack=5876 Win=65536 Len=4	TSval=749444822 TSecr=749444110	
3443	171.8616170...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=229 Ack=5876 Win=65536 Len=4	TSval=7494448819 TSecr=749444822	
3555	177.8361345...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=233 Ack=6034 Win=65536 Len=4	TSval=749454794 TSecr=749452163	
3577	178.9175845...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=237 Ack=6034 Win=65536 Len=4	TSval=749455875 TSecr=749454794	
3670	180.8965511...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=241 Ack=6179 Win=65536 Len=4	TSval=749457854 TSecr=749456781	
3676	181.9086283...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=245 Ack=6179 Win=65536 Len=4	TSval=749458866 TSecr=749457854	
3680	182.9182457...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=249 Ack=6179 Win=65536 Len=4	TSval=749459876 TSecr=749458866	
1821	79.816661204...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=25 Ack=1218 Win=65536 Len=4	TSval=749356776 TSecr=749356775	
3782	184.8178842...	127.0.0.1	127.0.0.1	TCP	72	1883 → 32965 [PSH, ACK]	Seq=253 Ack=6179 Win=65536 Len=4	TSval=749461775 TSecr=749459876	

tcp.port == 1883 && ipv6.dst == ::1									
No.	Time	Source	Destination	Protocol	Length	Info			
1	0.000000000	::1	::1	TCP	96	38083 → 1883 [SYN]	Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM TSval=2654789845 TSecr=0 WS=128		
2	0.000011734	::1	::1	TCP	96	1883 → 38083 [SYN, ACK]	Seq=0 Ack=1 Win=65464 Len=0 MSS=65476 SACK_PERM TSval=2654789845 TSecr=2654789845		
3	0.000018419	::1	::1	TCP	88	38083 → 1883 [ACK]	Seq=1 Ack=1 Win=65536 Len=0 TSval=2654789845 TSecr=2654789845		
4	0.000117188	::1	::1	MQTT	176	Connect Command			
5	0.000119441	::1	::1	TCP	88	1883 → 38083 [ACK]	Seq=1 Ack=89 Win=65408 Len=0 TSval=2654789845 TSecr=2654789845		
6	0.000252220	::1	::1	MQTT	92	Connect Ack			
7	0.000256578	::1	::1	TCP	88	38083 → 1883 [ACK]	Seq=89 Ack=5 Win=65536 Len=0 TSval=2654789845 TSecr=2654789845		
27	0.077643226	::1	::1	TCP	96	39551 → 1883 [SYN]	Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM TSval=2654789923 TSecr=0 WS=128		
28	0.077652648	::1	::1	TCP	96	1883 → 39551 [SYN, ACK]	Seq=0 Ack=1 Win=65464 Len=0 MSS=65476 SACK_PERM TSval=2654789923 TSecr=2654789923		
32	0.081922575	::1	::1	MQTT	114	Connect Command			
33	0.081957524	::1	::1	TCP	88	1883 → 39551 [ACK]	Seq=1 Ack=27 Win=65536 Len=0 TSval=2654789927 TSecr=2654789927		
34	0.082832242	::1	::1	MQTT	92	Connect Ack			
35	0.082835932	::1	::1	TCP	88	39551 → 1883 [ACK]	Seq=27 Ack=5 Win=65536 Len=0 TSval=2654789927 TSecr=2654789927		
90	1.001972511	::1	::1	MQTT	128	Publish Message (id=1) [factory/facility3/floor2/pollution]			
91	1.002062383	::1	::1	TCP	88	1883 → 38083 [ACK]	Seq=5 Ack=129 Win=65536 Len=0 TSval=2654790847 TSecr=2654790847		
92	1.002291442	::1	::1	MQTT	92	Publish Ack (id=1)			
93	1.002300527	::1	::1	TCP	88	38083 → 1883 [ACK]	Seq=129 Ack=9 Win=65536 Len=0 TSval=2654790847 TSecr=2654790847		
96	1.008370957	::1	::1	TCP	96	43797 → 1883 [SYN]	Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM TSval=2654790925 TSecr=0 WS=128		
97	1.008377872	::1	::1	TCP	96	53557 → 1883 [SYN]	Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM TSval=2654790925 TSecr=0 WS=128		
98	1.008384878	::1	::1	TCP	96	1883 → 43797 [SYN, ACK]	Seq=0 Ack=1 Win=65464 Len=0 MSS=65476 SACK_PERM TSval=2654790925 TSecr=2654790925		
99	1.008396883	::1	::1	TCP	96	1883 → 53557 [SYN, ACK]	Seq=0 Ack=1 Win=65464 Len=0 MSS=65476 SACK_PERM TSval=2654790925 TSecr=2654790925		
100	1.008392271	::1	::1	TCP	88	43797 → 1883 [ACK]	Seq=1 Ack=1 Win=65536 Len=0 TSval=2654790925 TSecr=2654790925		
101	1.008467861	::1	::1	TCP	88	53557 → 1883 [ACK]	Seq=1 Ack=1 Win=65536 Len=0 TSval=2654790925 TSecr=2654790925		
102	1.008504346	::1	::1	MQTT	133	Connect Command			
103	1.008593773	::1	::1	TCP	88	1883 → 53557 [ACK]	Seq=1 Ack=46 Win=65536 Len=0 TSval=2654790925 TSecr=2654790925		
104	1.008609548	::1	::1	MQTT	108	Connect Command			



Then is necessary to decode all the packets from TCP to MQTT and, by applying the following filters, is possible to identify the local broker messages distribution:



File: A.pcapng

Filter: mqtt.msgtype == 3 && ip.dst == 127.0.0.1

No.	Time	Source	Destination	Protocol	Length	Info
492	68.002314091	127.0.0.1	127.0.0.1	MQTT	204	Publish Message (id=1) [factory/facility2]
653	72.010881447	127.0.0.1	127.0.0.1	MQTT	214	Publish Message (id=1) [hospital/facility2/section0]
656	72.011049255	127.0.0.1	127.0.0.1	MQTT	214	Publish Message (id=1) [hospital/facility2/section0]
606	71.008846065	127.0.0.1	127.0.0.1	MQTT	215	Publish Message (id=1) [metaverse/building5/section0]
614	71.009883554	127.0.0.1	127.0.0.1	MQTT	215	Publish Message (id=1) [metaverse/building5/section0]
1497	98.455153837	127.0.0.1	127.0.0.1	MQTT	225	Publish Message (id=10) [factory/room2/section0/hydraulic_valve]
1837	88.054949794	127.0.0.1	127.0.0.1	MQTT	106	Publish Message (id=10) [hospital/building5/area2/deposit]
944	78.417396182	127.0.0.1	127.0.0.1	MQTT	218	Publish Message (id=10) [metaverse/facility2/area2/light]
803	75.819349837	127.0.0.1	127.0.0.1	MQTT	222	Publish Message (id=10) [metaverse/facility2/area2/pollution]
1000	79.622578550	127.0.0.1	127.0.0.1	MQTT	224	Publish Message (id=11) [hospital/building5/area2/temperature]
1515	91.199817973	127.0.0.1	127.0.0.1	MQTT	205	Publish Message (id=11) [hospital/building5]
899	77.656246551	127.0.0.1	127.0.0.1	MQTT	209	Publish Message (id=11) [metaverse/room2/floor1]
973	79.243614195	127.0.0.1	127.0.0.1	MQTT	216	Publish Message (id=12) [hospital/department5/section0]
1158	82.897817809	127.0.0.1	127.0.0.1	MQTT	201	Publish Message (id=12) [hospital/room2]
1523	91.493471383	127.0.0.1	127.0.0.1	MQTT	200	Publish Message (id=13) [factory/room2]
1129	81.886388278	127.0.0.1	127.0.0.1	MQTT	228	Publish Message (id=13) [metaverse/room2/section0/hydraulic_valve]
978	79.268757872	127.0.0.1	127.0.0.1	MQTT	219	Publish Message (id=13) [university/facility2/area2/light]
1203	84.323709321	127.0.0.1	127.0.0.1	MQTT	224	Publish Message (id=13) [university/facility2/floor1/pollution]
1064	80.509178603	127.0.0.1	127.0.0.1	MQTT	115	Publish Message (id=14) [factory/department5/area2/hydraulic_valve]
1751	95.708831103	127.0.0.1	127.0.0.1	MQTT	211	Publish Message (id=14) [factory/facility2/floor1]
1143	82.256894632	127.0.0.1	127.0.0.1	MQTT	213	Publish Message (id=14) [metaverse/facility2/floor1]
1766	96.301505985	127.0.0.1	127.0.0.1	MQTT	204	Publish Message (id=15) [factory/building5]
1286	86.151131174	127.0.0.1	127.0.0.1	MQTT	99	Publish Message (id=15) [factory/department5/area2]
1120	81.791068770	127.0.0.1	127.0.0.1	MQTT	206	Publish Message (id=15) [metaverse/building5]
1193	84.055048578	127.0.0.1	127.0.0.1	MQTT	206	Publish Message (id=15) [metaverse/facility2]
1945	100.1204485...	127.0.0.1	127.0.0.1	MQTT	92	Publish Message (id=16) [hospital/facility2]
1198	84.227213714	127.0.0.1	127.0.0.1	MQTT	220	Publish Message (id=16) [metaverse/building5/room4/deposit]



File: B.pcapng

Filter: mqtt.msgtype == 3 && ipv6.dst == ::1

No.	Time	Source	Destination	Protocol	Length	Info
98	1.001972511	::1	::1	MQTT	128	Publish Message (id=1) [factory/facility3/floor2/pollution]
204	2.873994259	::1	::1	MQTT	233	Publish Message (id=2) [university/building4/room0]
242	3.088025157	::1	::1	MQTT	239	Publish Message [metaverse/building4/area0/humidity]
244	3.088031546	::1	::1	MQTT	231	Publish Message [metaverse/building4/floor4]
246	3.088037163	::1	::1	MQTT	226	Publish Message [metaverse/department3]
248	3.088043019	::1	::1	MQTT	235	Publish Message (id=1) [metaverse/department3/floor1]
250	3.088048557	::1	::1	MQTT	243	Publish Message [metaverse/department3/section2/deposit]
252	3.088053898	::1	::1	MQTT	238	Publish Message [metaverse/department3/room0/light]
254	3.088059549	::1	::1	MQTT	244	Publish Message (id=2) [metaverse/department3/floor4/humidity]
256	3.088064738	::1	::1	MQTT	244	Publish Message [metaverse/department3/area0/temperature]
258	3.088070039	::1	::1	MQTT	224	Publish Message [metaverse/building5]
260	3.088075270	::1	::1	MQTT	238	Publish Message [metaverse/building5/room1/deposit]
262	3.088080842	::1	::1	MQTT	224	Publish Message [metaverse/facility3]
264	3.088087472	::1	::1	MQTT	230	Publish Message [metaverse/facility3/room1]
266	3.088092817	::1	::1	MQTT	236	Publish Message [metaverse/facility3/area0/light]
268	3.088098149	::1	::1	MQTT	241	Publish Message [metaverse/facility3/section0/deposit]
270	3.088103375	::1	::1	MQTT	232	Publish Message [metaverse/department4/area0]
272	3.088108990	::1	::1	MQTT	238	Publish Message [metaverse/department4/area0/light]
273	3.088146333	::1	::1	MQTT	244	Publish Message (id=1) [university/department3/floor1/deposit]
275	3.088154395	::1	::1	MQTT	240	Publish Message [university/facility1/area0/deposit]
284	3.131438443	::1	::1	MQTT	4166	Publish Message [metaverse/department4/area0/humidity], Publish Message (id=3) [metaverse/department4/area0/humidity]
296	4.014006443	::1	::1	MQTT	231	Publish Message [metaverse/facility3/floor4]
298	4.014196626	::1	::1	MQTT	231	Publish Message [metaverse/facility3/floor4]
324	4.104730732	::1	::1	MQTT	227	Publish Message [university/department3]
339	4.680443102	::1	::1	MQTT	223	Publish Message [factory/facility3]
369	5.106091691	::1	::1	MQTT	235	Publish Message [hospital/facility1/room6/light]
371	5.106094918	::1	::1	MQTT	238	Publish Message [hospital/building2/section1/light]
373	5.106097281	::1	::1	MQTT	236	Publish Message [metaverse/department3]



The depth is defined by the numbers of “/” separated layers. Extracting the packets in a .CSV file, filtering the Topic Name with:

“=LUNGHEZZA(G632)- LUNGHEZZA (SOSTITUISCI(G632;"/";"")) + 1”

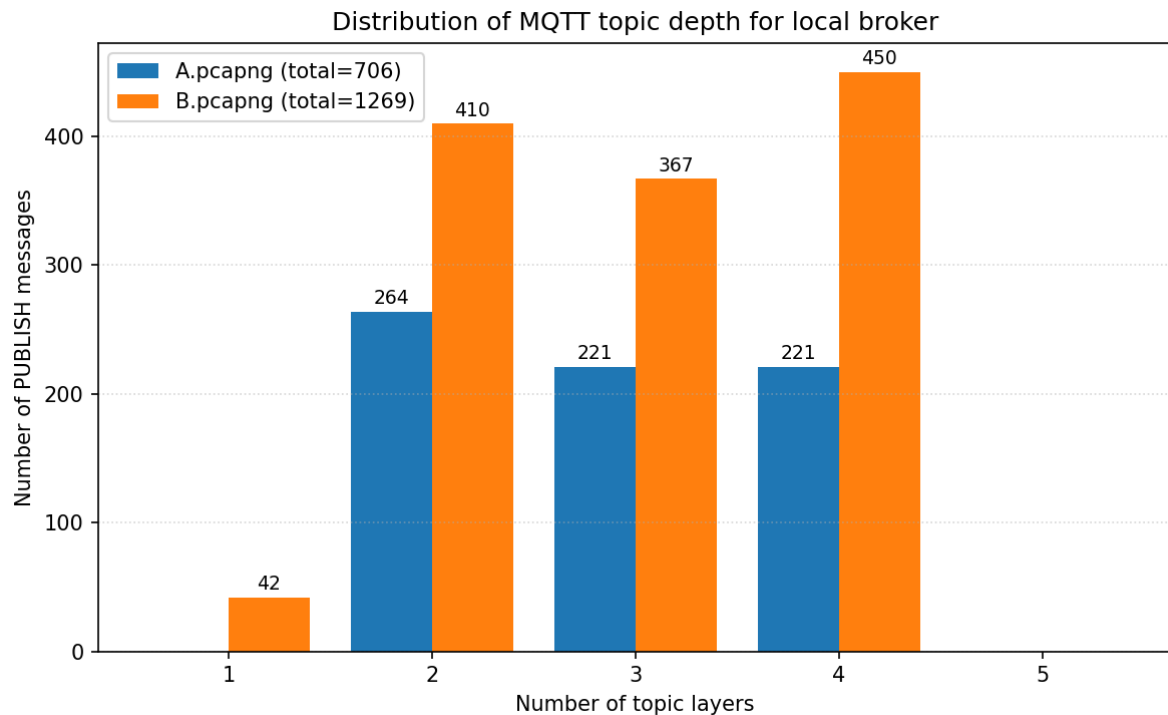
is possible to identify the value of the depth.

In the table are resumed all the values:

<u>Depth</u>	<u>A.pcapng</u>	<u>B.pcapng</u>
1	0	42
2	264	410
3	221	367
4	221	450
5	0	0



In the figure is shown the distribution of MQTT topic depth for local broker:



Answer CQ8

Total PUBLISH to local broker in **A.pcapng = 706**

Total PUBLISH to local broker in **B.pcapng = 1269**