# IoT 2023 CHALLENGE 1

**Referring to the file challenge2023_1.pcap, answer the following questions**:

1. How many CoAP GET requests are directed to non-existing resources in the local CoAP server? How many of these are of type Non confirmable? **(0.2 pts)**
   First, we first use the following filter to obtain get messages in the local host:
   Coap.code == 1 && ip.dst == 127.0.0.1
   Then we save the CSV file corresponding to this outcome.

   Second, we filter messages to the non-existing resources by this filter:
   Coap.code == 132 → 4.04 not found responses
   Then again, we save the CSV file corresponding to this outcome.
   We separated the information related to token and message ID and type of the messages (confirmable or non-confirmable) by adding them to column to be transparent in python for comparison.

   The rest of the solution is obtained by Python. We use the Pandas library to open CSV files and save them as DataFrame.

```
import pandas as pd
not_found_messages = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/IOT/CH1/q1_2.csv')
not_found_messages

get_messages = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/IOT/CH1/q1_1.csv')
get_messages
```

   Then we compare lists with messages Token to find those that are common between these 2 lists and count them. The problem here is that some of the messages have null Token and in this case we compare their messages ID. Here the answer to the first part is 11.

```
count = 0
for item in range(len(get_messages)):
  for token in range(len(not_found_messages)):
    if get_messages.iloc[item]['Token'] ==
not_found_messages.iloc[token]['Token']:
      print(get_messages.iloc[item]['Token'])
      count +=1
      break
    elif get_messages.iloc[item]['Token']!= 'NaN':
        if get_messages.iloc[item]['Message ID'] ==
not_found_messages.iloc[token]['Message ID']:
          print(get_messages.iloc[item]['Message ID'])
          count +=1
          break
count
```

For the second part, we also checked their Type to find those are Non-confirmable. Here the answer is 6.

```
count = 0
for item in range(len(get_messages)):
  for token in range(len(not_found_messages)):
    if get_messages.iloc[item]['Token'] ==
not_found_messages.iloc[token]['Token']:
      if get_messages.iloc[item]['Type'] == 'Non-Confirmable':
        print(get_messages.iloc[item]['Token'])
        count +=1
        break
    elif get_messages.iloc[item]['Token']!= 'NaN':
        if get_messages.iloc[item]['Message ID'] ==
not_found_messages.iloc[token]['Message ID']:
          if get_messages.iloc[item]['Type'] == 'Non-Confirmable':
            print(get_messages.iloc[item]['Token'])
            count +=1
            break
count
```

2. How many CoAP DELETE requests directed to the "coap.me" server did not produce a successful result? How many of these are directed to the "/hello" resource? **(0.2 pts)**

First we need to enable resolved network address from view →Name resolution → resolved network addresses. Then we filtered all the messages using coap protocol and contains a delete request directed to coap.me using following filter:
Coap.code == 4 && ipdst_host == coap.me

Here we obtained 115 messages, that all are from IP source 10.0.2.15 to coap.me.
Then we filtered all the response messages from coam.me to IP destination = 10.0.2.15 which contained successful deleted content by following filter:

Coap.code == 66 && ip.src_host == coap.me && ip.dst == 10.0.2.15

Here we obtained 10 messages with code 2.02 deleted, so 105 is the final answer (115 – 10 = 105).

For the second part, we have to find the delete requests of the first part which are also directed to “/hello” resource. Just we need to modify a bit the first part filter:

Coap.code == 4 && ip.dst_host == coap.me && coap.opt.uri_path == “hello”

Here we obtain 5 messages.Due to the low number of obtained messages, I picked their tokens and compared them with those of successfully deleted responses. There were no common tokens between them, so all the 5 delete requests to the “/hello” resource don’t have successful results.
Also to be smarter we could use the below query:

ip.dst == 134.102.218.18 && coap.code == 4 && ! coap.response_in in {1561 13071 15994 20613 21781 32084 33515 34573 35201 38756} && coap.opt.uri_path == "hello"

in this way, among delete requests to “/hello” resource in coap.me, we filter those with successful response using the frame number of successful responses.

3. How many different MQTT clients subscribe to the public broker mosquitto using single-level wildcards? How many of these *clients* **WOULD** receive a publish message issued to the topic "hospital/room2/area0“ **(0.2 pts)**

The filter is:

mqtt && ip.dst == 91.121.93.24 && mqtt.topic contains “+”

This filter returns 13 messages, but we have just 3 unique clients by considering the port number. These 3 ports are 35239,43133 and 51531. Then the answer to the first part is 3. For the second part, the answer is 2 (port numbers are 35239 and 43133). One was subscribed to hospital/+/area0 (43133) and the other one was subscribed without wildcard which we found by giving the topic without wildcard.

4. How many MQTT clients specify a last Will Message directed to a topic having as first level "university"? How many of these Will Messages are sent from the broker to the subscribers? **(0.2 pts)**

   The filter is:

   mqtt.willtopic contains “university”

   This filter returns these 2 messages:
   58887 → university/facility2/floor2
   41083 → university/building2/room6
   As we see university is at the first level, then the answer in the first part is 2.
   Both messages were sent by the subscriber to the broker, then the answer to the second part is 0, and IP_source is 10.0.2.15.

5. How many Publish messages with QoS = 1 are received by the MQTT clients connected to the HiveMQ broker with MQTT version 5? **(0.1 pts)**
   First, we want to find the ports that are connected to broker.hivemq.com, because while the client is connecting to the broker sends a connect message to the broker. We are doing this filtering as follows:

   Mqtt.msqtype == 1 && mqtt.ver == 5 && ip.dst_host == broker.hivemq.com

   And we obtain 9 messages with these ports:
   47723, 60609, 57265, 36665, 45635, 37401, 46967, 47549, and 42827.
   Then we want to find those messages that were sent to these ports,

Mqtt.msgtype == 3 && mqtt.qos == 1 && ip.src_host == “ broker.hivemg.com” && tcp.dstport in { 47723 60609 57265 36665 45635 37401 46967 47549 42827}

And the remained packets are 51.

6. How many MQTT-SN (on port 1885) publish messages sent after the hour 3.16PM (Milan Time) are directed to topic 9? Are these messages handled by the server? **(0.1 pts)**

   First of all, we changed the port number of MQTT-SN to the mentioned one(1885). Then using the below code in Wireshark we filtered the messages with mentioned specifications.

   frame.time >= “march 14,2023 15:16:00 “ && mqttsn.topic.id == 9 && mqttsn.msg.type == 0xc && !ICMP

   and the result is 15 packets.
   Because the destinations of these messages are unreachable then all of them are not handled by the server.