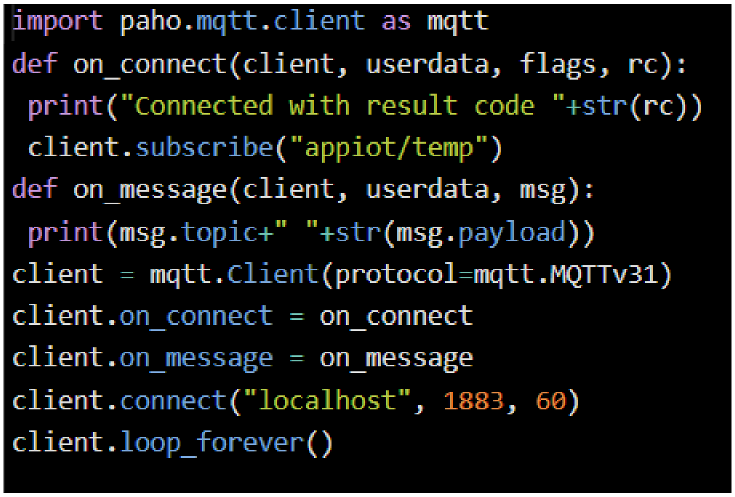
# LAB 2: MQTT

**Shubhika GARG**

## A) PART-1

**1. The following lines show the python code of a Subscriber application. Copy and save this code in "subscriber.py" file. Explain each line of the code.**



```
import paho.mqtt.client as mqtt
def on_connect(client, userdata, flags, rc):
 print("Connected with result code "+str(rc))
 client.subscribe("appiot/temp")
def on_message(client, userdata, msg):
 print(msg.topic+" "+str(msg.payload))
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_forever()
```

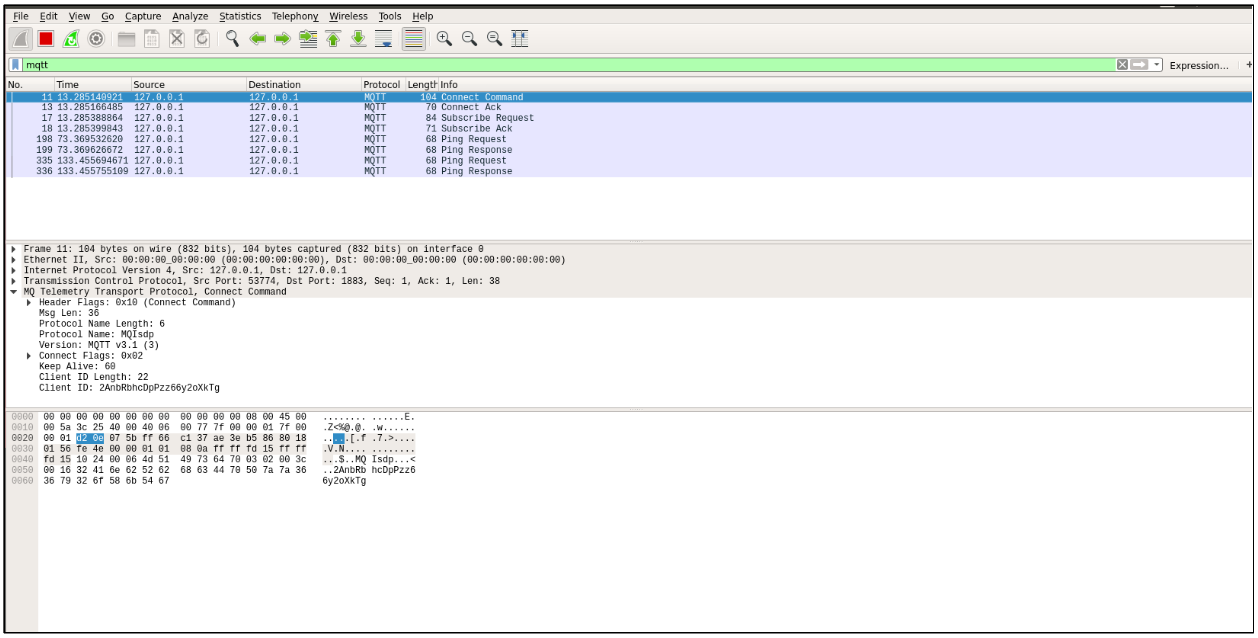
### Code Explanation:

- **import paho.mqtt.client as mqtt**
  This line imports the Paho MQTT Client library in Python and provides it a name "mqtt".
- **def on_connect(client, userdata, flags, rc):**
  **print("Connected with result code "+str(rc))**
  **client.subscribe("appiot/temp")**
  This function is called when the MQTT client connects to the broker successfully. It takes four parameters i.e., the client instance (client), any user-defined data (userdata), the connection flags (flags), and the connection result code (rc). The function prints a message indicating that the client has connected, and then subscribes to the "appiot/temp" topic stored on the server using the client's subscribe() method.
- **def on_message(client, userdata, msg):**
  **print(msg.topic+" "+str(msg.payload))**
  This function is called whenever a message is received from the broker. It takes three parameters: the client instance (client), any user-defined data (userdata), and the message object (msg). The function prints the topic and payload of the received message. This function is used to receive and process messages that are published by the broker and sent to the client (which is a subscriber)
- **client = mqtt.Client(protocol=mqtt.MQTTv31)**
  This line creates a new MQTT client instance using the mqtt.Client() constructor. It sets the protocol parameter to mqtt.MQTTv31, which specifies the MQTT version to use. The MQTT version specified is MQTTv3.1.
- **client.on_connect = on_connect**
  **client.on_message = on_message**

The above two lines tell the client library which functions to call when the client connects to the broker and receives a message. The on_connect and on_message properties of the client instance are set to the on_connect() and on_message() functions.

- **client.connect("localhost", 1883, 60)**
  This line connects the client to the MQTT broker running on the local machine. It has three parameters:
  The hostname of the server is "localhost"
  The port number to connect is 1883
  Keepalive time is 60 seconds
  The client connects to the "localhost" with the port number "1883" and keeps
  it alive for "60" seconds.
- **client.loop_forever()**
  This line starts the client's network loop, which listens for the incoming messages from the broker and then calls the appropriate callback function when a message is received. It blocks the program from exiting, and since it is called at the end of the program, it ensures that the client keeps running.

**2. Start Wireshark, and launch a capture. You may use MQTT filter to see only MQTT messages. Launch the subscriber. Stop the capture.**



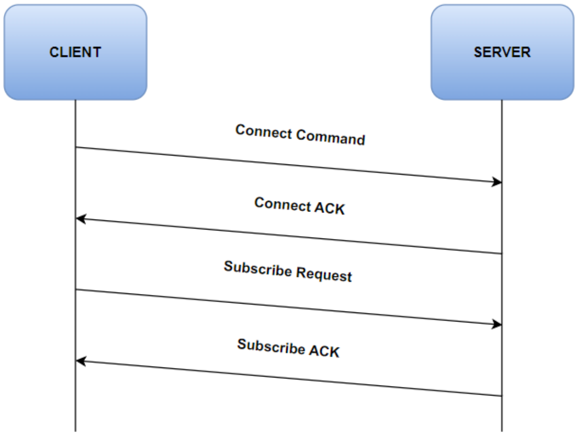
```
No.   Time          Source      Destination  Protocol  Length Info
   11 13.285140921  127.0.0.1   127.0.0.1    MQTT       104 Connect Command
   13 13.285166485  127.0.0.1   127.0.0.1    MQTT        70 Connect Ack
   17 13.285388864  127.0.0.1   127.0.0.1    MQTT        84 Subscribe Request
   18 13.285399843  127.0.0.1   127.0.0.1    MQTT        71 Subscribe Ack
  198 73.369532620  127.0.0.1   127.0.0.1    MQTT        68 Ping Request
  199 73.369626672  127.0.0.1   127.0.0.1    MQTT        68 Ping Response
  335 133.455694671 127.0.0.1   127.0.0.1    MQTT        68 Ping Request
  336 133.455755109 127.0.0.1   127.0.0.1    MQTT        68 Ping Response
```

```
▸ Frame 11: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 53774, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▾ MQ Telemetry Transport Protocol, Connect Command
  ▸ Header Flags: 0x10 (Connect Command)
    Msg Len: 36
    Protocol Name Length: 6
    Protocol Name: MQIsdp
    Version: MQTT v3.1 (3)
  ▸ Connect Flags: 0x02
    Keep Alive: 60
    Client ID Length: 22
    Client ID: 2AnbRbhcDpPzz66y2oXkTg
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a 3c 25 40 00 40 06  00 77 7f 00 00 01 7f 00   .Z<%@.@. .w......
0020  00 01 d2 0e 07 5b ff 66  c1 37 ae 3e b5 86 80 18   .....[.f .7.>....
0030  01 56 fe 4e 00 00 01 01  08 0a ff ff fd 15 ff ff   .V.N.... ........
0040  fd 15 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   ...$..MQ Isdp...<
0050  00 16 32 41 6e 62 52 62  68 63 44 70 50 7a 7a 36   ..2AnbRb hcDpPzz6
0060  36 79 32 6f 58 6b 54 67                            6y2oXkTg
```

**3. What are the exchanged messages? Draw an exchange diagram.**

From the Wireshark capture, Connect Command, Connect Ack, Subscribe Request and Subscribe Ack are the exchanged messages. The Ping request and Ping response messages will ensure the flow of the network.

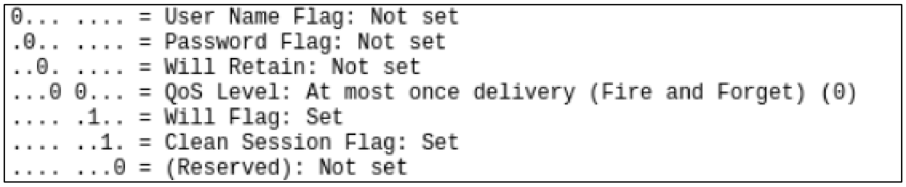


**4. What is the QoS level of the messages?**

The QoS level of the messages is set to zero. From the line:

*client.subscribe("appiot/temp")*

By default, the subscribe() method sets the QoS level to 0. This means that when the subscriber (the MQTT client) subscribes to the "appiot/temp" topic, it requests that messages published to that topic be delivered at most once.

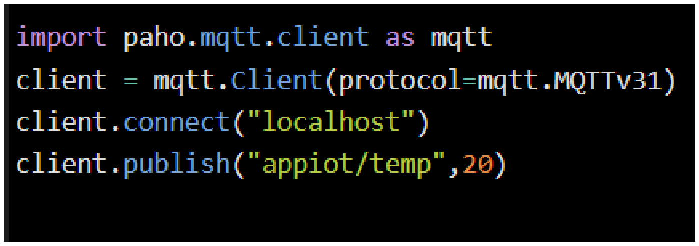


```
0... .... = User Name Flag: Not set
.0.. .... = Password Flag: Not set
..0. .... = Will Retain: Not set
...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)
.... .1.. = Will Flag: Set
.... ..1. = Clean Session Flag: Set
.... ...0 = (Reserved): Not set
```

**5. Is it a clean session? Explain.**

The QoS is set to 0 while the Clean Session Flag is “Set”. If a client with a clean session flag “Set” subscribes to a topic with a QoS level of 0, the broker will not store any messages for that client, and the client will not receive any missed messages when it reconnects.

**6. The following python code shows an example of a publisher. Copy and Save it in “publisher.py”.**



```
import paho.mqtt.client as mqtt
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.connect("localhost")
client.publish("appiot/temp",20)
```

**Code Explanation:**

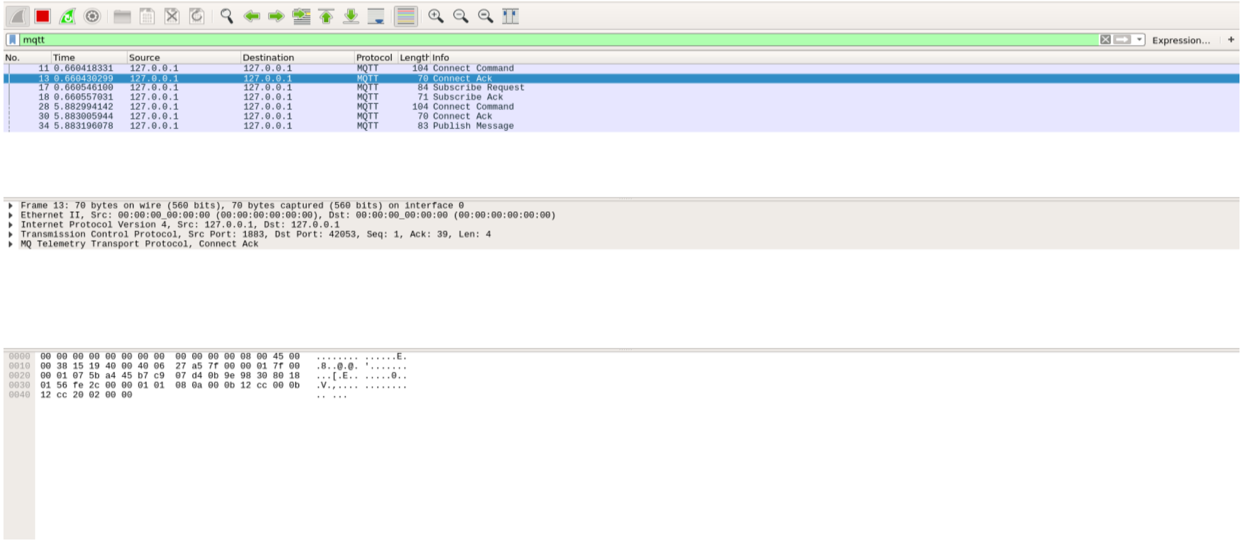
- **import paho.mqtt.client as mqtt**
  This line imports the Paho MQTT Client library in Python and provides it a name “mqtt”.
- **client = mqtt.Client(protocol=mqtt.MQTTv31)**

This line creates a new MQTT client instance using the mqtt.Client() constructor. It sets the protocol parameter to mqtt.MQTTv31, which specifies the MQTT version to use. The MQTT version specified is MQTTv3.1.

- **client.connect("localhost")**
  This line connects the MQTT client to the local MQTT broker running on the same machine.
- **client.publish("appiot/temp", 20)**
  This line publishes a message to the "appiot/temp" topic with a payload of 20.

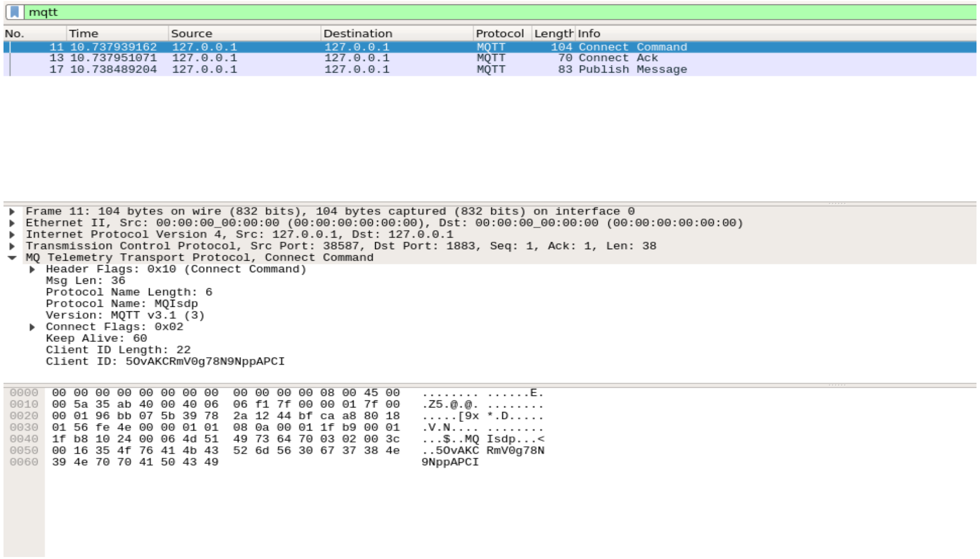
**7. Start Wireshark, and launch a capture. Launch the publisher. Stop the capture.**

The publisher sends messages to the broker, while the subscriber receives messages from the broker.



```
mqtt
No.  Time          Source       Destination  Protocol Length Info
  11 0.660418331   127.0.0.1    127.0.0.1    MQTT       104 Connect Command
  13 0.660430299   127.0.0.1    127.0.0.1    MQTT        70 Connect Ack
  17 0.660546100   127.0.0.1    127.0.0.1    MQTT        84 Subscribe Request
  18 0.660557031   127.0.0.1    127.0.0.1    MQTT        71 Subscribe Ack
  28 5.882994142   127.0.0.1    127.0.0.1    MQTT       104 Connect Command
  30 5.883005944   127.0.0.1    127.0.0.1    MQTT        70 Connect Ack
  34 5.883196078   127.0.0.1    127.0.0.1    MQTT        83 Publish Message

▸ Frame 13: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 1883, Dst Port: 42053, Seq: 1, Ack: 39, Len: 4
▸ MQ Telemetry Transport Protocol, Connect Ack

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 38 15 19 40 00 40 06  27 a5 7f 00 00 01 7f 00   .8..@.@. '.......
0020  00 01 07 5b a4 45 b7 c9  07 d4 0b 9e 98 30 80 18   ...[.E.. .....0..
0030  01 56 fe 2c 00 00 01 01  08 0a 00 0b 12 cc 00 0b   .V.,.... ........
0040  12 cc 20 02 00 00                                  .. ...
```

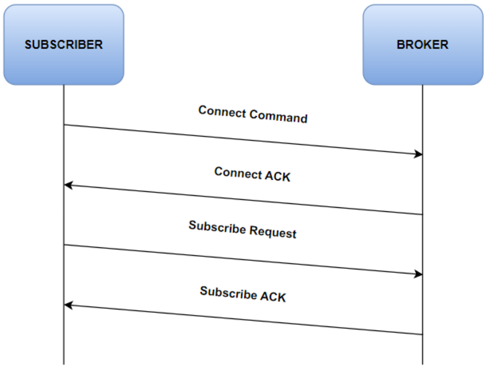
Note: Below is the snip on running only the publisher

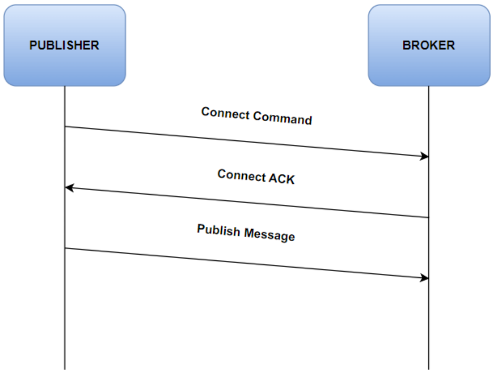


```
mqtt
No.  Time           Source       Destination  Protocol Length Info
  11 10.737939162   127.0.0.1    127.0.0.1    MQTT       104 Connect Command
  13 10.737951071   127.0.0.1    127.0.0.1    MQTT        70 Connect Ack
  17 10.738489204   127.0.0.1    127.0.0.1    MQTT        83 Publish Message

▸ Frame 11: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 38587, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▾ MQ Telemetry Transport Protocol, Connect Command
  ▸ Header Flags: 0x10 (Connect Command)
    Msg Len: 36
    Protocol Name Length: 6
    Protocol Name: MQIsdp
    Version: MQTT v3.1 (3)
  ▸ Connect Flags: 0x02
    Keep Alive: 60
    Client ID Length: 22
    Client ID: 5OvAKCRmV0g78N9NppAPCI

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a 35 ab 40 00 40 06  06 f1 7f 00 00 01 7f 00   .Z5.@.@. ........
0020  00 01 96 bb 07 5b 39 78  2a 12 44 bf ca a8 80 18   .....[9x *.D.....
0030  01 56 fe 4e 00 00 01 01  08 0a 00 01 1f b9 00 01   .V.N.... ........
0040  1f b8 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   ...$..MQ Isdp...<
0050  00 16 35 4f 76 41 4b 43  52 6d 56 30 67 37 38 4e   ..5OvAKC RmV0g78N
0060  39 4e 70 70 41 50 43 49                            9NppAPCI
```

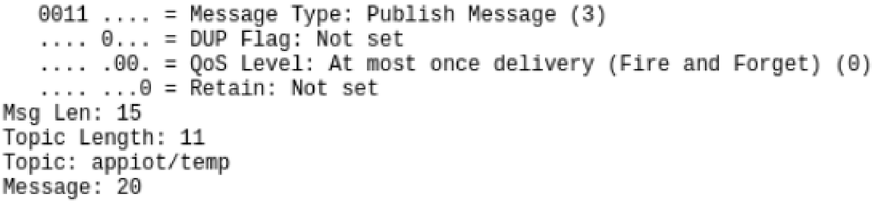
**8. What are the exchanged messages? Draw an exchange diagram.**

The exchanged messages between the broker and the publisher are Connect Command, Connect Ack and Publish Message. The publisher sends a PUBLISH packet to the broker with the topic "appiot/temp" and a payload of the message (a temperature reading of 20).





**9. What is the QoS required by this message? Does it contain a retain?**

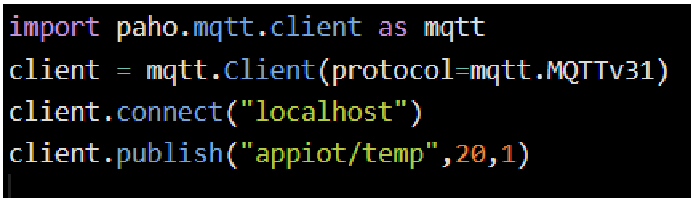


```
    0011 .... = Message Type: Publish Message (3)
    .... 0... = DUP Flag: Not set
    .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
    .... ...0 = Retain: Not set
Msg Len: 15
Topic Length: 11
Topic: appiot/temp
Message: 20
```

The QoS is 0 (at most once delivery). This implies that the message is not guaranteed to be delivered to the subscriber, and if it is lost or not delivered for any reason, the publisher will not be notified. The retain flag is ‘Not set’, and hence the message will not be retained by the broker for future subscribers.

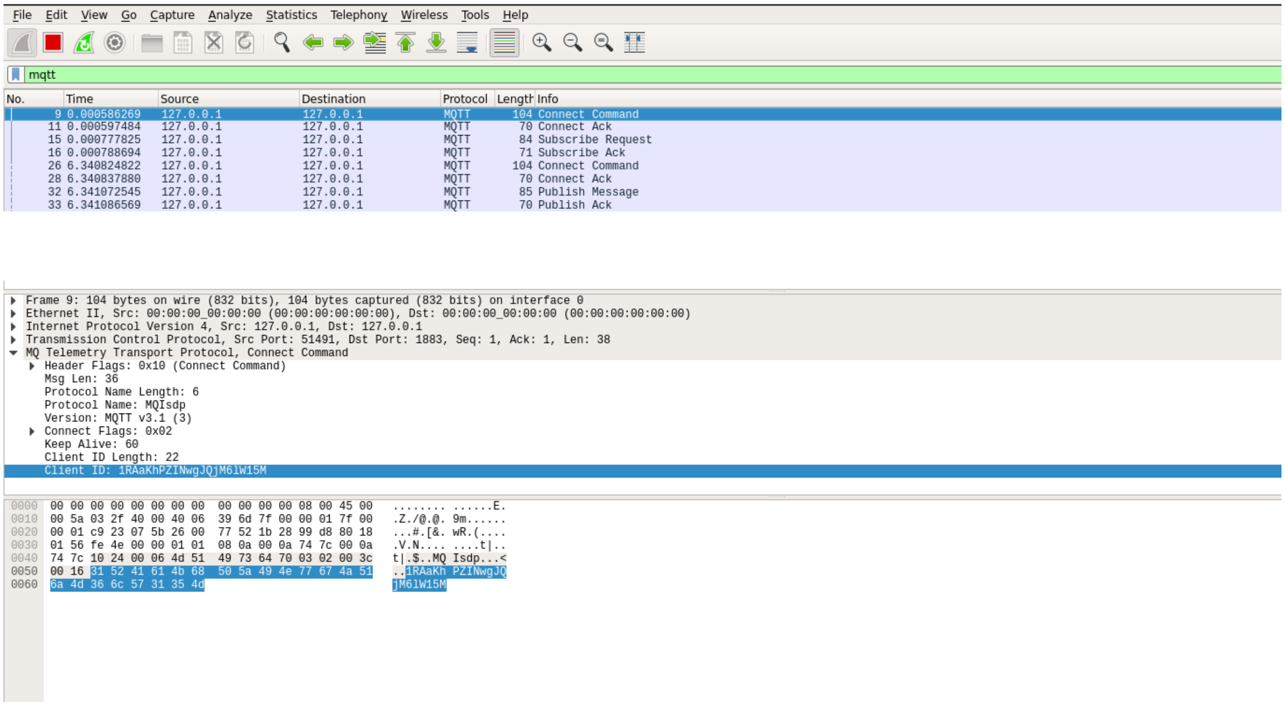
**10. Modify the last line of the publisher code with:**

**client.publish(“appiot/temp”,20,1)**



```
import paho.mqtt.client as mqtt
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.connect("localhost")
client.publish("appiot/temp",20,1)
```

### 11. Start Wireshark, and launch a capture. Launch the publisher. Stop the capture.

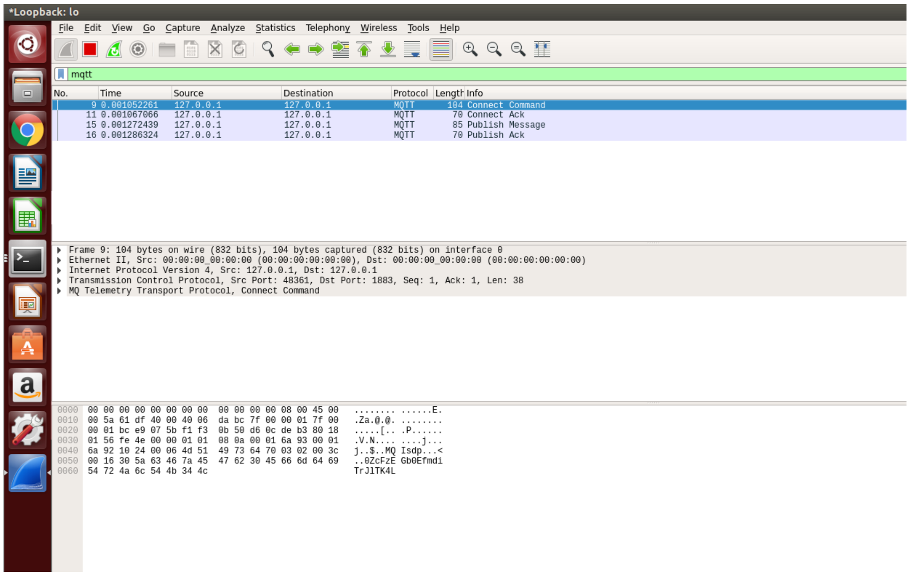


```
No.  Time          Source      Destination  Protocol  Lengtł Info
   9 0.000586269   127.0.0.1   127.0.0.1    MQTT       104 Connect Command
  11 0.000597484   127.0.0.1   127.0.0.1    MQTT        70 Connect Ack
  15 0.000777825   127.0.0.1   127.0.0.1    MQTT        84 Subscribe Request
  16 0.000788694   127.0.0.1   127.0.0.1    MQTT        71 Subscribe Ack
  26 6.340824822   127.0.0.1   127.0.0.1    MQTT       104 Connect Command
  28 6.340837880   127.0.0.1   127.0.0.1    MQTT        70 Connect Ack
  32 6.341072545   127.0.0.1   127.0.0.1    MQTT        85 Publish Message
  33 6.341086569   127.0.0.1   127.0.0.1    MQTT        70 Publish Ack
```

```
▸ Frame 9: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 51491, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▾ MQ Telemetry Transport Protocol, Connect Command
  ▸ Header Flags: 0x10 (Connect Command)
    Msg Len: 36
    Protocol Name Length: 6
    Protocol Name: MQIsdp
    Version: MQTT v3.1 (3)
  ▸ Connect Flags: 0x02
    Keep Alive: 60
    Client ID Length: 22
    Client ID: 1RAaKhPZINwgJQjM6lW15M
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a 03 2f 40 00 40 06  39 6d 7f 00 00 01 7f 00   .Z./@.@. 9m......
0020  00 01 c9 23 07 5b 26 00  77 52 1b 28 99 d8 80 18   ...#.[&. wR.(....
0030  01 56 fe 4e 00 00 01 01  08 0a 00 0a 74 7c 00 0a   .V.N.... ....t|..
0040  74 7c 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   t|.$..MQ Isdp...<
0050  00 16 31 52 41 61 4b 68  50 5a 49 4e 77 67 4a 51   ..1RAaKh PZINwgJQ
0060  6a 4d 36 6c 57 31 35 4d                           jM6lW15M
```

Note: Below is the snip on running only the publisher



```
No.  Time          Source      Destination  Protocol  Lengtł Info
   9 0.001052261   127.0.0.1   127.0.0.1    MQTT       104 Connect Command
  11 0.001067066   127.0.0.1   127.0.0.1    MQTT        70 Connect Ack
  15 0.001272439   127.0.0.1   127.0.0.1    MQTT        85 Publish Message
  16 0.001286324   127.0.0.1   127.0.0.1    MQTT        70 Publish Ack
```

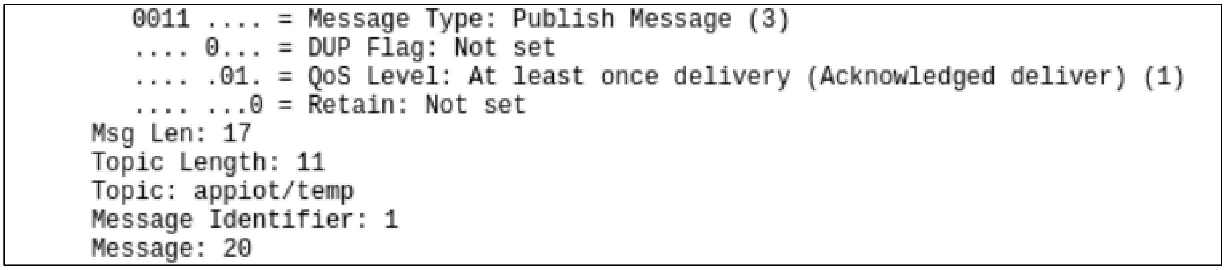
```
▸ Frame 9: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 48361, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▸ MQ Telemetry Transport Protocol, Connect Command
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a 61 df 40 00 40 06  da bc 7f 00 00 01 7f 00   .Za.@.@. ........
0020  00 01 bc e9 07 5b f1 f3  0b 50 d6 0c de b3 80 18   .....[.. .P......
0030  01 56 fe 4e 00 00 01 01  08 0a 00 01 6a 93 00 01   .V.N.... ....j...
0040  6a 92 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   j..$..MQ Isdp...<
0050  00 16 30 5a 63 46 7a 45  47 62 30 45 66 6d 64 69   ..0ZcFzE Gb0Efmdi
0060  54 72 4a 6c 54 4b 34 4c                           TrJlTK4L
```

### 12. What are the exchanged messages?

Connect Command, Connect Ack, Publish Message and Public Ack are the four exchanged messages for the above scenario.

**13. What are the differences with the precedent case? Explain.**

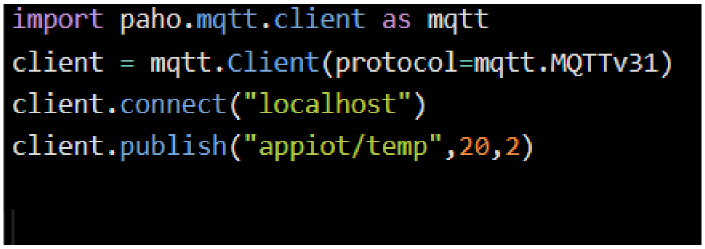


```
        0011 .... = Message Type: Publish Message (3)
        .... 0... = DUP Flag: Not set
        .... .01. = QoS Level: At least once delivery (Acknowledged deliver) (1)
        .... ...0 = Retain: Not set
    Msg Len: 17
    Topic Length: 11
    Topic: appiot/temp
    Message Identifier: 1
    Message: 20
```

The QoS level has changed from 0 to 1. This implies that the message will be delivered to the broker with an acknowledgement from the broker. The broker will send a Publish ACK packet to the publisher to acknowledge that the message has been received and stored by the broker.

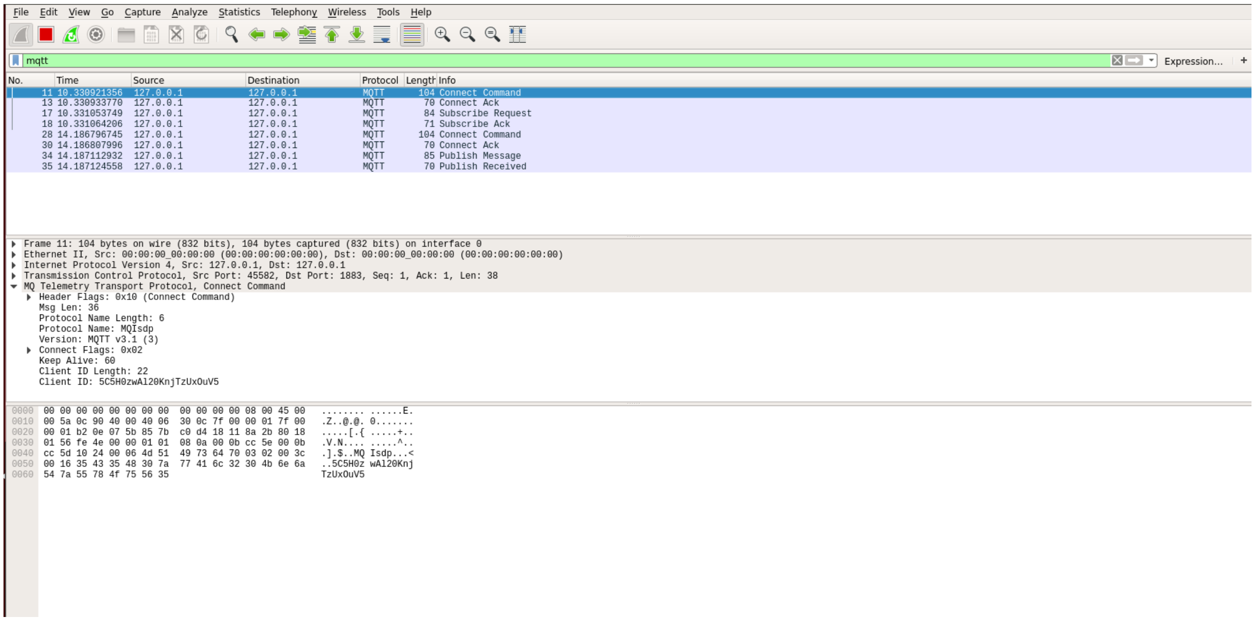
**14. Again, modify the last line of the publisher as follows:**

**client.publish(“appiot/temp”,20,2)**



```
import paho.mqtt.client as mqtt
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.connect("localhost")
client.publish("appiot/temp",20,2)
```

**15. Start Wireshark, and launch a capture. Launch the publisher. Stop the capture.**



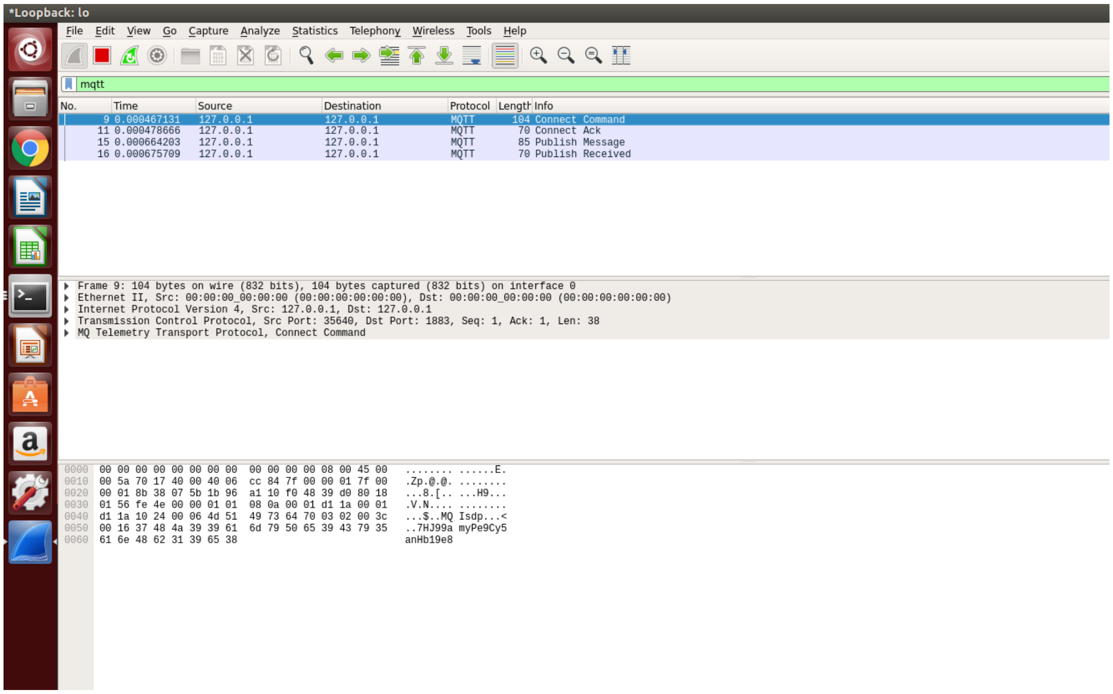
```
mqtt
```

```
No.   Time          Source      Destination  Protocol Length Info
   11 10.330921356  127.0.0.1   127.0.0.1    MQTT      104 Connect Command
   13 10.330933770  127.0.0.1   127.0.0.1    MQTT       70 Connect Ack
   17 10.331053749  127.0.0.1   127.0.0.1    MQTT       84 Subscribe Request
   18 10.331064206  127.0.0.1   127.0.0.1    MQTT       71 Subscribe Ack
   28 14.186796745  127.0.0.1   127.0.0.1    MQTT      104 Connect Command
   30 14.186807996  127.0.0.1   127.0.0.1    MQTT       70 Connect Ack
   34 14.187112932  127.0.0.1   127.0.0.1    MQTT       85 Publish Message
   35 14.187124558  127.0.0.1   127.0.0.1    MQTT       70 Publish Received
```

```
▸ Frame 11: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 45582, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▾ MQ Telemetry Transport Protocol, Connect Command
  ▸ Header Flags: 0x10 (Connect Command)
    Msg Len: 36
    Protocol Name Length: 6
    Protocol Name: MQIsdp
    Version: MQTT v3.1 (3)
  ▸ Connect Flags: 0x02
    Keep Alive: 60
    Client ID Length: 22
    Client ID: 5C5H0zwAl20KnjTzUxOuV5
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a 0c 90 40 00 40 06  30 0c 7f 00 00 01 7f 00   .Z..@.@. 0.......
0020  00 01 b2 0e 07 5b 85 7b  c0 d4 18 11 8a 2b 80 18   .....[.{ .....+..
0030  01 56 fe 4e 00 00 01 01  08 0a 00 0b cc 5e 00 0b   .V.N.... .....^..
0040  cc 5d 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   .].$..MQ Isdp...<
0050  00 16 35 43 35 48 30 7a  77 41 6c 32 30 4b 6e 6a   ..5C5H0z wAl20Knj
0060  54 7a 55 78 4f 75 56 35                            TzUxOuV5
```

Note: Below is the snip on running only the publisher


*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
mqtt

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 9 | 0.000467131 | 127.0.0.1 | 127.0.0.1 | MQTT | 104 | Connect Command |
| 11 | 0.000478666 | 127.0.0.1 | 127.0.0.1 | MQTT | 70 | Connect Ack |
| 15 | 0.000664203 | 127.0.0.1 | 127.0.0.1 | MQTT | 85 | Publish Message |
| 16 | 0.000675709 | 127.0.0.1 | 127.0.0.1 | MQTT | 70 | Publish Received |

```
▸ Frame 9: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 35640, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▸ MQ Telemetry Transport Protocol, Connect Command
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a 70 17 40 00 40 06  cc 84 7f 00 00 01 7f 00   .Zp.@.@. ........
0020  00 01 8b 38 07 5b 1b 96  a1 10 f0 48 39 d0 80 18   ...8.[.. ...H9...
0030  01 56 fe 4e 00 00 01 01  08 0a 00 01 d1 1a 00 01   .V.N.... ........
0040  d1 1a 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   ...$..MQ Isdp...<
0050  00 16 37 48 4a 39 39 61  6d 79 50 65 39 43 79 35   ..7HJ99a myPe9Cy5
0060  61 6e 48 62 31 39 65 38                            anHb19e8
```

**16. What are the exchanged messages?**

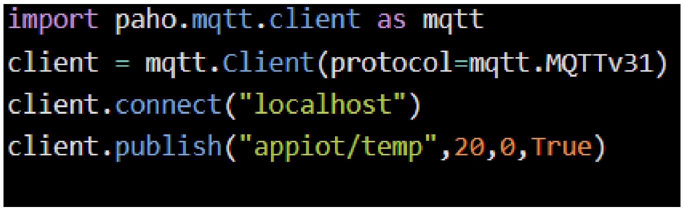
Connect Command, Connect Ack, Publish Message and Publish Received are the four exchanged messages.

**17. What are the differences with the two precedent cases? Explain.**

This case publishes the message to the MQTT broker with QoS level 2 (exactly once delivery), which implies that the message is guaranteed to be delivered to the broker exactly once. This QoS level provides the highest level of reliability and eliminates the possibility of duplicate messages being delivered to the subscriber. The publisher and the broker engage in a handshake process to ensure that the message is delivered exactly once (In this scenario, we get the message Publish Receive), and this process can be more resource-intensive compared to QoS level 0 or 1.

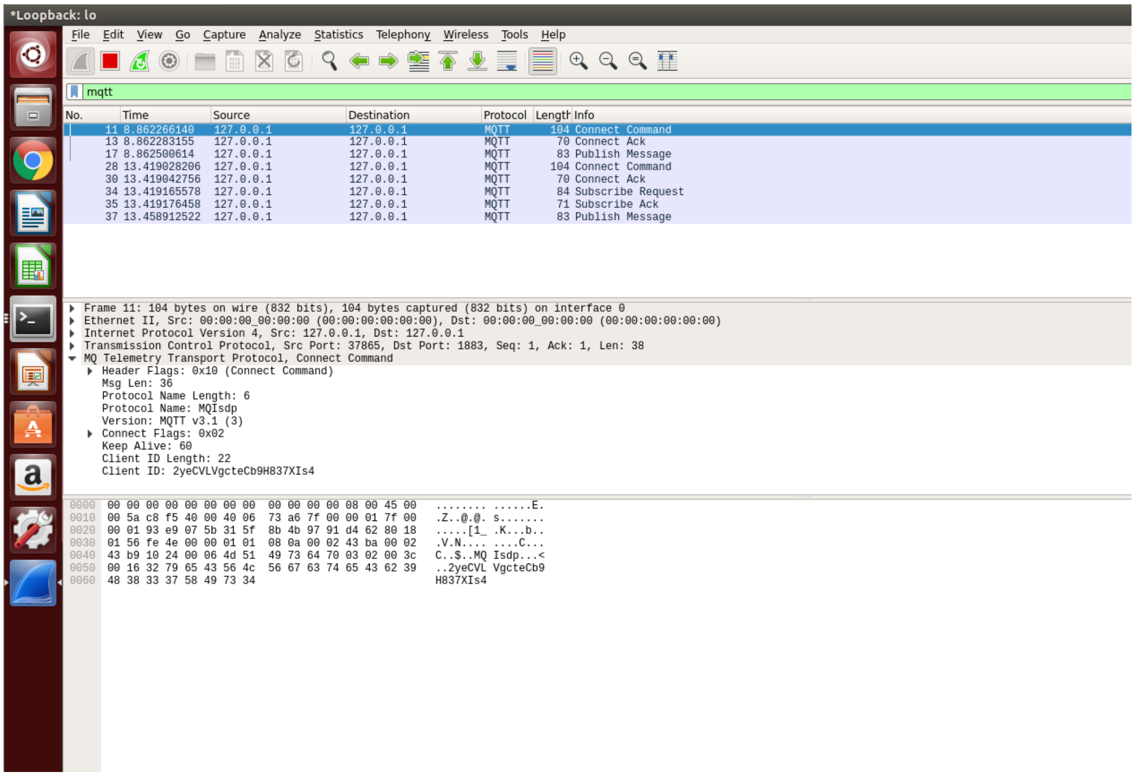
**18. Now stop both the subscriber and the publisher. Modify the last line of the publisher:**

**client.publish("appiot/temp",20,0,True)**



```
import paho.mqtt.client as mqtt
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.connect("localhost")
client.publish("appiot/temp",20,0,True)
```

**19. Start Wireshark. Start the publisher first, then the subscriber. Stop the capture.**


*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

mqtt

```
No.   Time          Source      Destination  Protocol  Length Info
   11 8.862266140   127.0.0.1   127.0.0.1    MQTT       104 Connect Command
   13 8.862283155   127.0.0.1   127.0.0.1    MQTT        70 Connect Ack
   17 8.862500614   127.0.0.1   127.0.0.1    MQTT        83 Publish Message
   28 13.419028206  127.0.0.1   127.0.0.1    MQTT       104 Connect Command
   30 13.419042756  127.0.0.1   127.0.0.1    MQTT        70 Connect Ack
   34 13.419165578  127.0.0.1   127.0.0.1    MQTT        84 Subscribe Request
   35 13.419176458  127.0.0.1   127.0.0.1    MQTT        71 Subscribe Ack
   37 13.458912522  127.0.0.1   127.0.0.1    MQTT        83 Publish Message
```

```
▸ Frame 11: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 37865, Dst Port: 1883, Seq: 1, Ack: 1, Len: 38
▾ MQ Telemetry Transport Protocol, Connect Command
  ▸ Header Flags: 0x10 (Connect Command)
    Msg Len: 36
    Protocol Name Length: 6
    Protocol Name: MQIsdp
    Version: MQTT v3.1 (3)
  ▸ Connect Flags: 0x02
    Keep Alive: 60
    Client ID Length: 22
    Client ID: 2yeCVLVgcteCb9H837XIs4
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5a c8 f5 40 00 40 06  73 a6 7f 00 00 01 7f 00   .Z..@.@. s.......
0020  00 01 93 e9 07 5b 31 5f  8b 4b 97 91 d4 62 80 18   .....[1_ .K...b..
0030  01 56 fe 4e 00 00 01 01  08 0a 00 02 43 ba 00 02   .V.N.... ....C...
0040  43 b9 10 24 00 06 4d 51  49 73 64 70 03 02 00 3c   C..$..MQ Isdp...<
0050  00 16 32 79 65 43 56 4c  56 67 63 74 65 43 62 39   ..2yeCVL VgcteCb9
0060  48 38 33 37 58 49 73 34                            H837XIs4
```

**20. What do you remark at the subscriber console?**

When the subscriber console is checked, it displays the message "20" sent by the publisher, along with the topic "appiot/temp".

The console window shows: **appiot/temp 20**

**21. Explain this concept.**

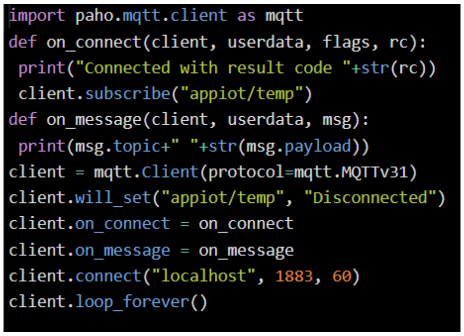
When the publisher runs first, it connects to the broker and publishes the message to the specified topic. The broker stores the message until a subscriber is available to receive it.

Then, when the subscriber is run, it connects to the broker and subscribes to the specified topic. The broker delivers the stored message to the subscriber, which then displays the received topic name and message value in its console.

**22. Copy the subscriber.py, and name it subscriber2.py. Modify the subscriber.py by adding the following line (after client…):**

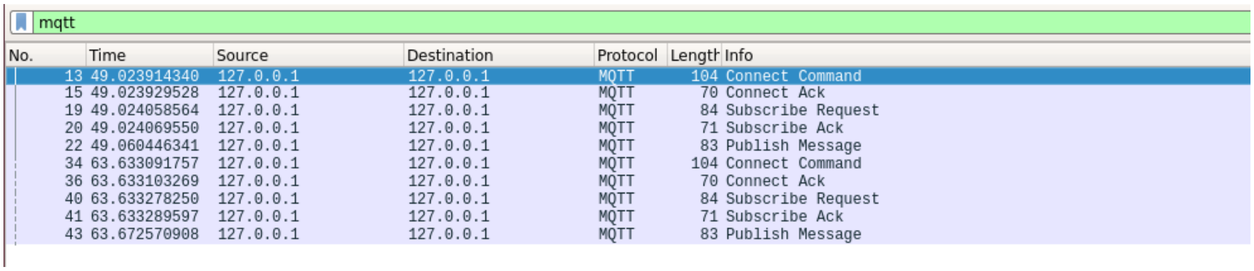
**client = mqtt.Client(protocol=mqtt.MQTTv31)**

**client.will_set("appiot/temp", “Disconnected”)**

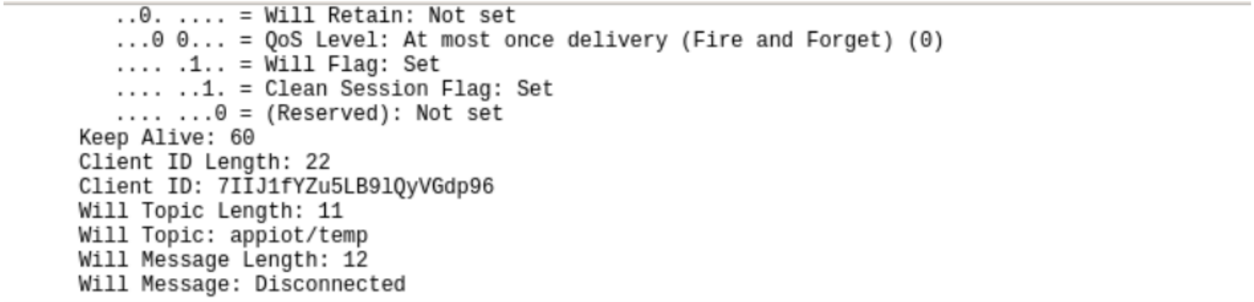


```
import paho.mqtt.client as mqtt
def on_connect(client, userdata, flags, rc):
 print("Connected with result code "+str(rc))
 client.subscribe("appiot/temp")
def on_message(client, userdata, msg):
 print(msg.topic+" "+str(msg.payload))
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.will_set("appiot/temp", "Disconnected")
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_forever()
```

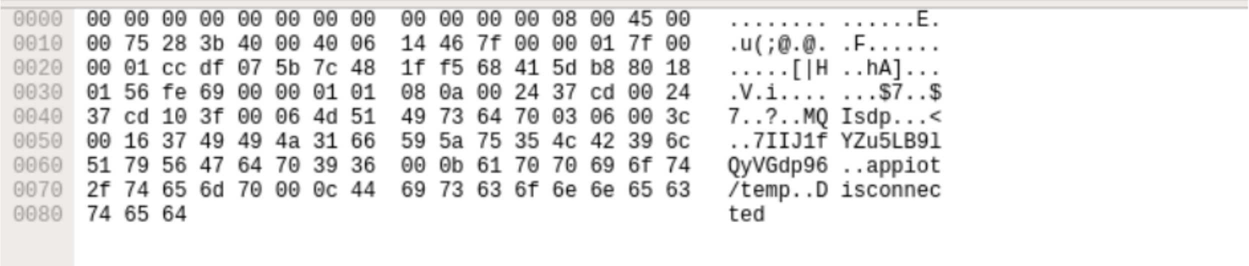
**23. Start Wireshark. Start both subscribers. In the subscriber terminal enter “ctrl+c”. Stop Wireshark.**



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 13 | 49.023914340 | 127.0.0.1 | 127.0.0.1 | MQTT | 104 | Connect Command |
| 15 | 49.023929528 | 127.0.0.1 | 127.0.0.1 | MQTT | 70 | Connect Ack |
| 19 | 49.024058564 | 127.0.0.1 | 127.0.0.1 | MQTT | 84 | Subscribe Request |
| 20 | 49.024069550 | 127.0.0.1 | 127.0.0.1 | MQTT | 71 | Subscribe Ack |
| 22 | 49.060446341 | 127.0.0.1 | 127.0.0.1 | MQTT | 83 | Publish Message |
| 34 | 63.633091757 | 127.0.0.1 | 127.0.0.1 | MQTT | 104 | Connect Command |
| 36 | 63.633103269 | 127.0.0.1 | 127.0.0.1 | MQTT | 70 | Connect Ack |
| 40 | 63.633278250 | 127.0.0.1 | 127.0.0.1 | MQTT | 84 | Subscribe Request |
| 41 | 63.633289597 | 127.0.0.1 | 127.0.0.1 | MQTT | 71 | Subscribe Ack |
| 43 | 63.672570908 | 127.0.0.1 | 127.0.0.1 | MQTT | 83 | Publish Message |

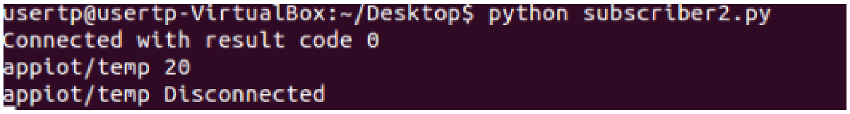


```
        ..0. .... = Will Retain: Not set
        ...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)
        .... .1.. = Will Flag: Set
        .... ..1. = Clean Session Flag: Set
        .... ...0 = (Reserved): Not set
     Keep Alive: 60
     Client ID Length: 22
     Client ID: 7IIJ1fYZu5LB91QyVGdp96
     Will Topic Length: 11
     Will Topic: appiot/temp
     Will Message Length: 12
     Will Message: Disconnected
```



```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 75 28 3b 40 00 40 06  14 46 7f 00 00 01 7f 00   .u(;@.@. .F......
0020  00 01 cc df 07 5b 7c 48  1f f5 68 41 5d b8 80 18   .....[|H ..hA]...
0030  01 56 fe 69 00 00 01 01  08 0a 00 24 37 cd 00 24   .V.i.... ...$7..$
0040  37 cd 10 3f 00 06 4d 51  49 73 64 70 03 06 00 3c   7..?..MQ Isdp...<
0050  00 16 37 49 49 4a 31 66  59 5a 75 35 4c 42 39 6c   ..7IIJ1f YZu5LB9l
0060  51 79 56 47 64 70 39 36  00 0b 61 70 70 69 6f 74   QyVGdp96 ..appiot
0070  2f 74 65 6d 70 00 0c 44  69 73 63 6f 6e 6e 65 63   /temp..D isconnec
0080  74 65 64                                            ted
```

**24. What is the message displayed in the subscriber2 console?**



```
usertp@usertp-VirtualBox:~/Desktop$ python subscriber2.py
Connected with result code 0
appiot/temp 20
appiot/temp Disconnected
```

The console window shows: **appiot/temp Disconnected**

**25. What is this concept?**

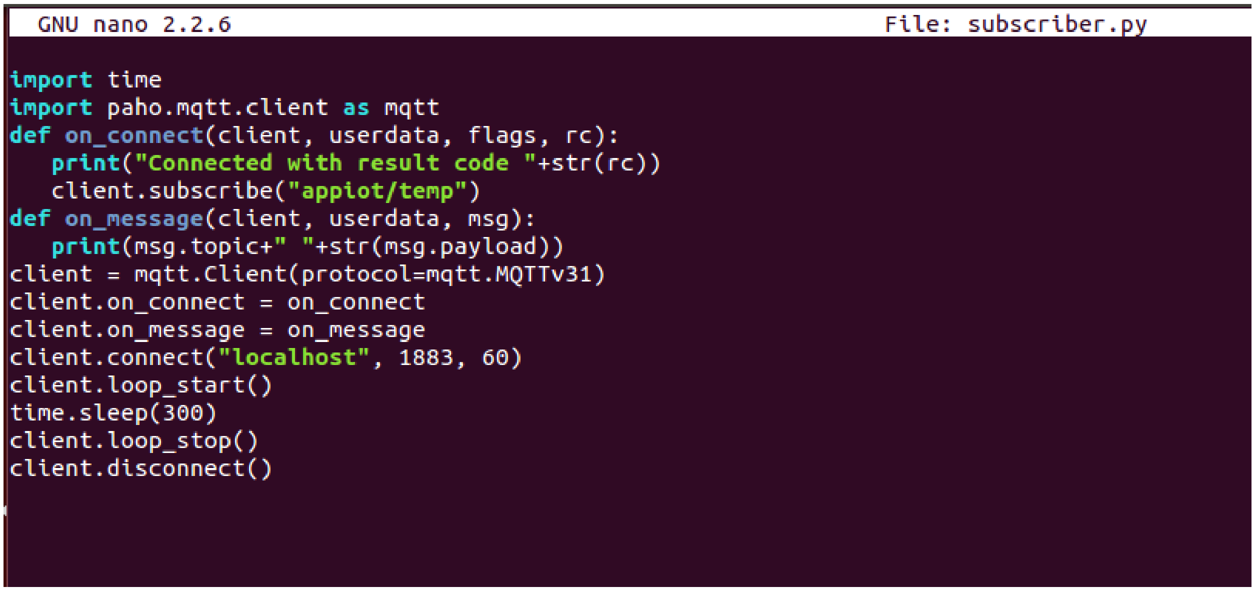
When the first subscriber is disconnected, the broker publishes the will message on its behalf, and the second subscriber receives the message in its on_message callback function and prints it on the console. When we interrupt the execution of subscriber using ctrl+c, the on_disconnect callback function is triggered on the client side. If the client has a will message set using the will_set() function, then the broker will publish the will message on behalf of the client to indicate that it has disconnected.

**26. Which packet is carrying this information? Refer to your capture.**

The disconnected message is sent by the broker as a Will message when a client disconnects unexpectedly and the broker is configured to send Will messages for that client. When a subscriber disconnects, it sends a message to the broker with the "Last Will and Testament" (LWT) message, which was set by the client when it connected. The LWT message is carried in a separate control packet called "Will Message" packet.

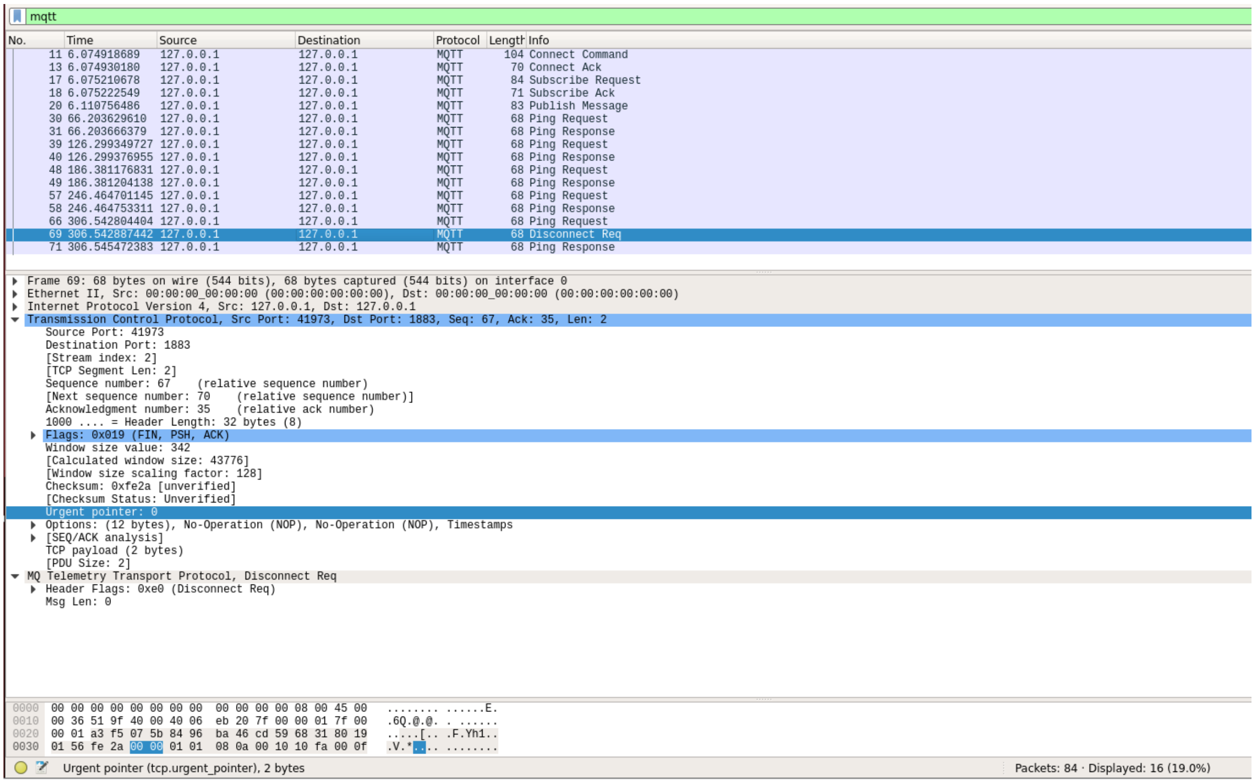
## B) PART-2

### 1. Modify the subscriber code to connect for a duration of 5 min, then gracefully disconnect.



```
  GNU nano 2.2.6                                            File: subscriber.py

import time
import paho.mqtt.client as mqtt
def on_connect(client, userdata, flags, rc):
   print("Connected with result code "+str(rc))
   client.subscribe("appiot/temp")
def on_message(client, userdata, msg):
   print(msg.topic+" "+str(msg.payload))
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_start()
time.sleep(300)
client.loop_stop()
client.disconnect()
```

Below is a Wireshark capture:

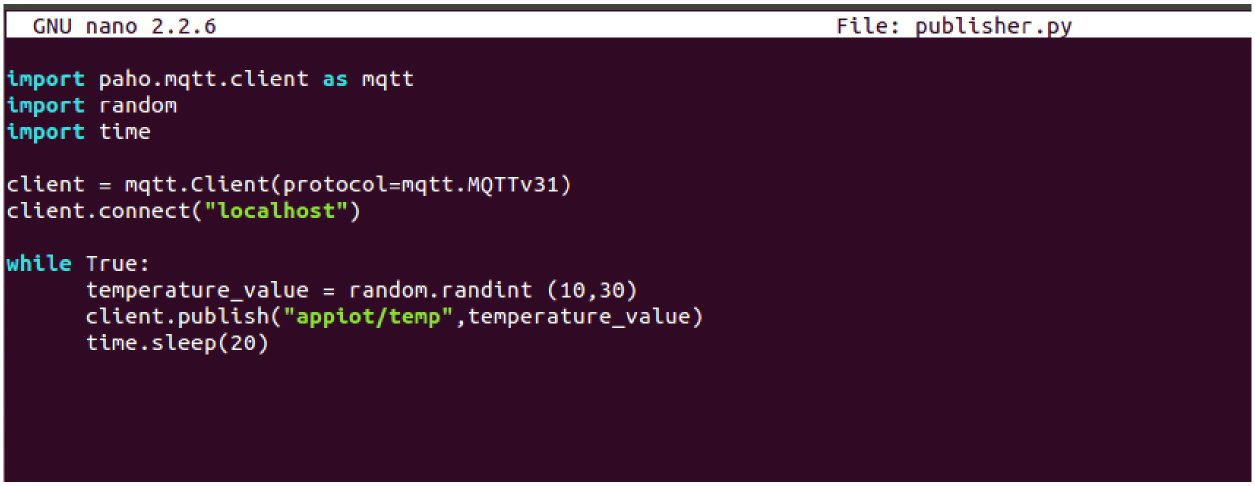


```
mqtt
No.   Time          Source      Destination  Protocol Length Info
   11 6.074918689   127.0.0.1   127.0.0.1    MQTT      104 Connect Command
   13 6.074930180   127.0.0.1   127.0.0.1    MQTT       70 Connect Ack
   17 6.075210678   127.0.0.1   127.0.0.1    MQTT       84 Subscribe Request
   18 6.075222549   127.0.0.1   127.0.0.1    MQTT       71 Subscribe Ack
   20 6.110756486   127.0.0.1   127.0.0.1    MQTT       83 Publish Message
   30 66.203629610  127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   31 66.203666379  127.0.0.1   127.0.0.1    MQTT       68 Ping Response
   39 126.299349727 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   40 126.299376955 127.0.0.1   127.0.0.1    MQTT       68 Ping Response
   48 186.381176831 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   49 186.381204138 127.0.0.1   127.0.0.1    MQTT       68 Ping Response
   57 246.464701145 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   58 246.464753311 127.0.0.1   127.0.0.1    MQTT       68 Ping Response
   66 306.542804404 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   69 306.542887442 127.0.0.1   127.0.0.1    MQTT       68 Disconnect Req
   71 306.545472383 127.0.0.1   127.0.0.1    MQTT       68 Ping Response

▸ Frame 69: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▾ Transmission Control Protocol, Src Port: 41973, Dst Port: 1883, Seq: 67, Ack: 35, Len: 2
    Source Port: 41973
    Destination Port: 1883
    [Stream index: 2]
    [TCP Segment Len: 2]
    Sequence number: 67    (relative sequence number)
    [Next sequence number: 70    (relative sequence number)]
    Acknowledgment number: 35    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x019 (FIN, PSH, ACK)
    Window size value: 342
    [Calculated window size: 43776]
    [Window size scaling factor: 128]
    Checksum: 0xfe2a [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
    TCP payload (2 bytes)
    [PDU Size: 2]
▾ MQ Telemetry Transport Protocol, Disconnect Req
  ▸ Header Flags: 0xe0 (Disconnect Req)
    Msg Len: 0

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 36 51 9f 40 00 40 06  eb 20 7f 00 00 01 7f 00   .6Q.@.@. . ......
0020  00 01 a3 f5 07 5b 84 96  ba 46 cd 59 68 31 80 19   .....[.. .F.Yh1..
0030  01 56 fe 2a 00 00 01 01  08 0a 00 10 10 fa 00 0f   .V.*.... ........

Urgent pointer (tcp.urgent_pointer), 2 bytes                 Packets: 84 · Displayed: 16 (19.0%)
```

### From the above code:

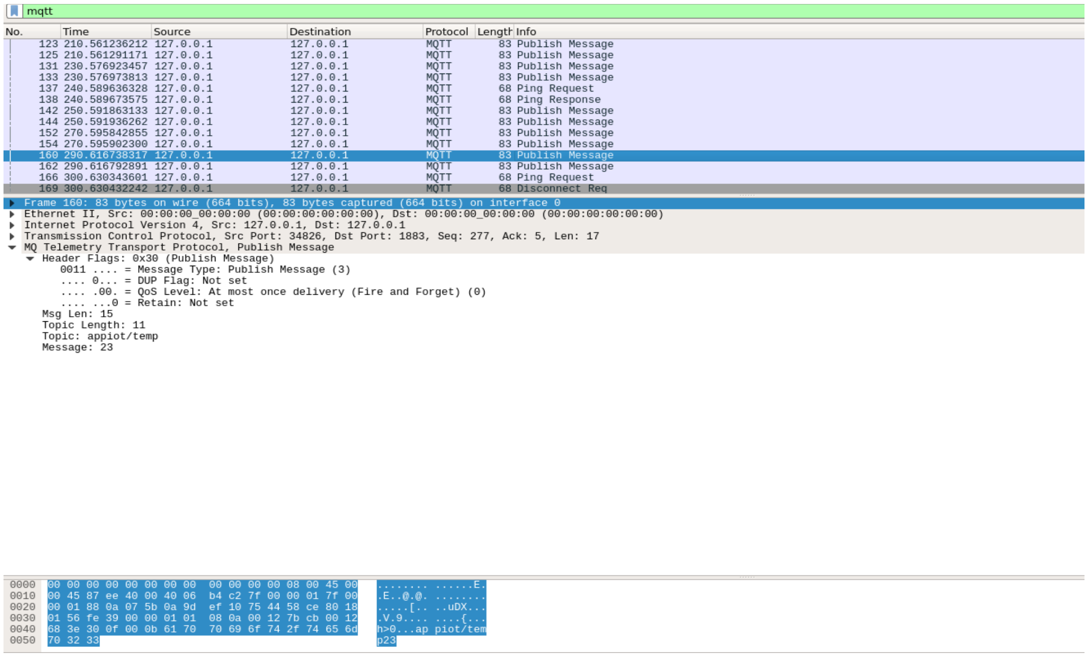
| | |
|---|---|
| **client.loop_start()** | Starts the network loop in a separate thread |
| **time.sleep (300)** | Pause the main thread for 5 minutes |
| **client.disconnect()** | Gracefully disconnect from the broker |
| **client.loop_stop()** | Stop the network loop |

**2. Modify the publisher to generate a temperature value between 10 and 30 periodically.**



```
  GNU nano 2.2.6                                          File: publisher.py

import paho.mqtt.client as mqtt
import random
import time

client = mqtt.Client(protocol=mqtt.MQTTv31)
client.connect("localhost")

while True:
      temperature_value = random.randint (10,30)
      client.publish("appiot/temp",temperature_value)
      time.sleep(20)
```

Below is a Wireshark capture:

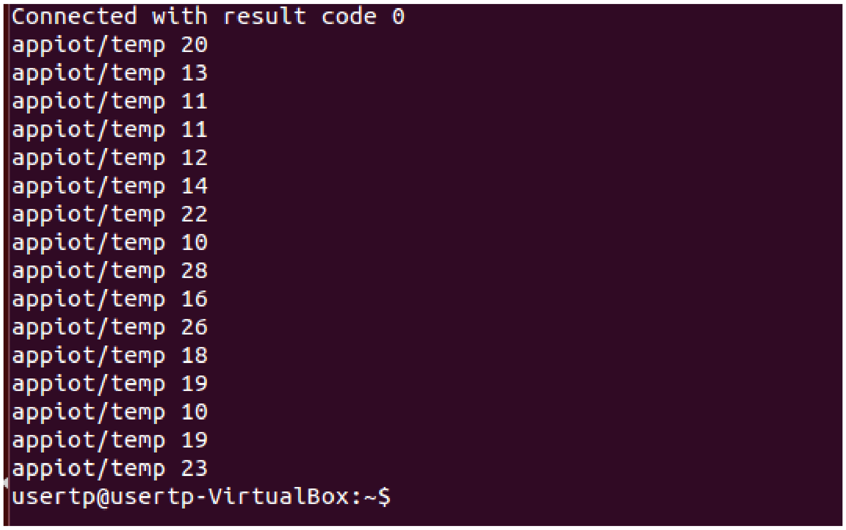


```
mqtt
No.    Time           Source       Destination  Protocol Length Info
   123 210.561236212 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   125 210.561291171 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   131 230.576923457 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   133 230.576973813 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   137 240.589636328 127.0.0.1    127.0.0.1    MQTT      68 Ping Request
   138 240.589673575 127.0.0.1    127.0.0.1    MQTT      68 Ping Response
   142 250.591863133 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   144 250.591936262 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   152 270.595842855 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   154 270.595902300 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   160 290.616738317 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   162 290.616792891 127.0.0.1    127.0.0.1    MQTT      83 Publish Message
   166 300.630343601 127.0.0.1    127.0.0.1    MQTT      68 Ping Request
   169 300.630432242 127.0.0.1    127.0.0.1    MQTT      68 Disconnect Req

▶ Frame 160: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 34826, Dst Port: 1883, Seq: 277, Ack: 5, Len: 17
▼ MQ Telemetry Transport Protocol, Publish Message
   ▼ Header Flags: 0x30 (Publish Message)
        0011 .... = Message Type: Publish Message (3)
        .... 0... = DUP Flag: Not set
        .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
        .... ...0 = Retain: Not set
     Msg Len: 15
     Topic Length: 11
     Topic: appiot/temp
     Message: 23

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 45 87 ee 40 00 40 06  b4 c2 7f 00 00 01 7f 00   .E..@.@. ........
0020  00 01 88 0a 07 5b 0a 9d  ef 10 75 44 58 ce 80 18   .....[.. ..uDX...
0030  01 56 fe 39 00 00 01 01  08 0a 00 12 7b cb 00 12   .V.9.... ....{...
0040  68 3e 30 0f 00 0b 61 70  70 69 6f 74 2f 74 65 6d   h>0...ap piot/tem
0050  70 32 33                                           p23
```

**From the above code:**

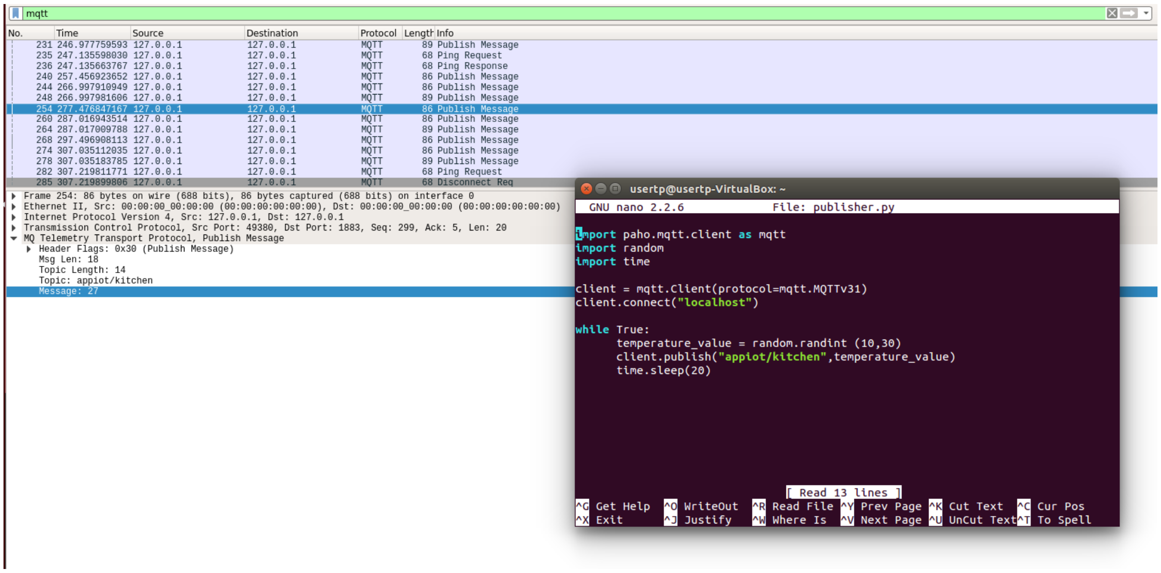
The **while** loop will generate a new temperature value randomly between 10 and 30 using the **random.randint ()** function, publishes it to the appiot/temp topic using **client.publish()**, and then waits for 20 seconds using **time.sleep()** before generating the next value.

Below is the terminal capture:



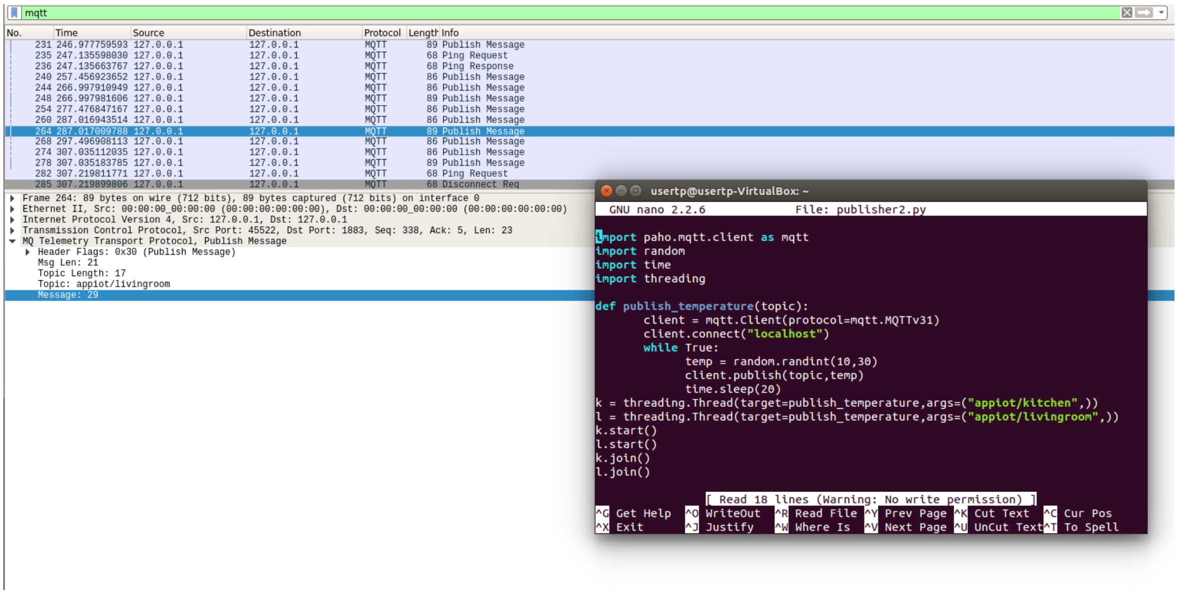
```
Connected with result code 0
appiot/temp 20
appiot/temp 13
appiot/temp 11
appiot/temp 11
appiot/temp 12
appiot/temp 14
appiot/temp 22
appiot/temp 10
appiot/temp 28
appiot/temp 16
appiot/temp 26
appiot/temp 18
appiot/temp 19
appiot/temp 10
appiot/temp 19
appiot/temp 23
usertp@usertp-VirtualBox:~$
```

### 3. Add another publisher that generates temperature for living room, while the first one on the kitchen.



```
mqtt
No.    Time           Source      Destination  Protocol Length Info
   231 246.977759593 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   235 247.135598030 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   236 247.135663767 127.0.0.1   127.0.0.1    MQTT       68 Ping Response
   240 257.456923652 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   244 266.997910949 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   248 266.997981606 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   254 277.476847167 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   260 287.016943514 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   264 287.017009788 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   268 297.496908113 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   274 307.035112035 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   278 307.035183785 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   282 307.219811771 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   285 307.219899806 127.0.0.1   127.0.0.1    MQTT       68 Disconnect Req
▶ Frame 254: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 49380, Dst Port: 1883, Seq: 299, Ack: 5, Len: 20
▼ MQ Telemetry Transport Protocol, Publish Message
  ▶ Header Flags: 0x30 (Publish Message)
    Msg Len: 18
    Topic Length: 14
    Topic: appiot/kitchen
    Message: 27
```

```
usertp@usertp-VirtualBox: ~
  GNU nano 2.2.6              File: publisher.py

import paho.mqtt.client as mqtt
import random
import time

client = mqtt.Client(protocol=mqtt.MQTTv31)
client.connect("localhost")

while True:
      temperature_value = random.randint (10,30)
      client.publish("appiot/kitchen",temperature_value)
      time.sleep(20)

                              [ Read 13 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```



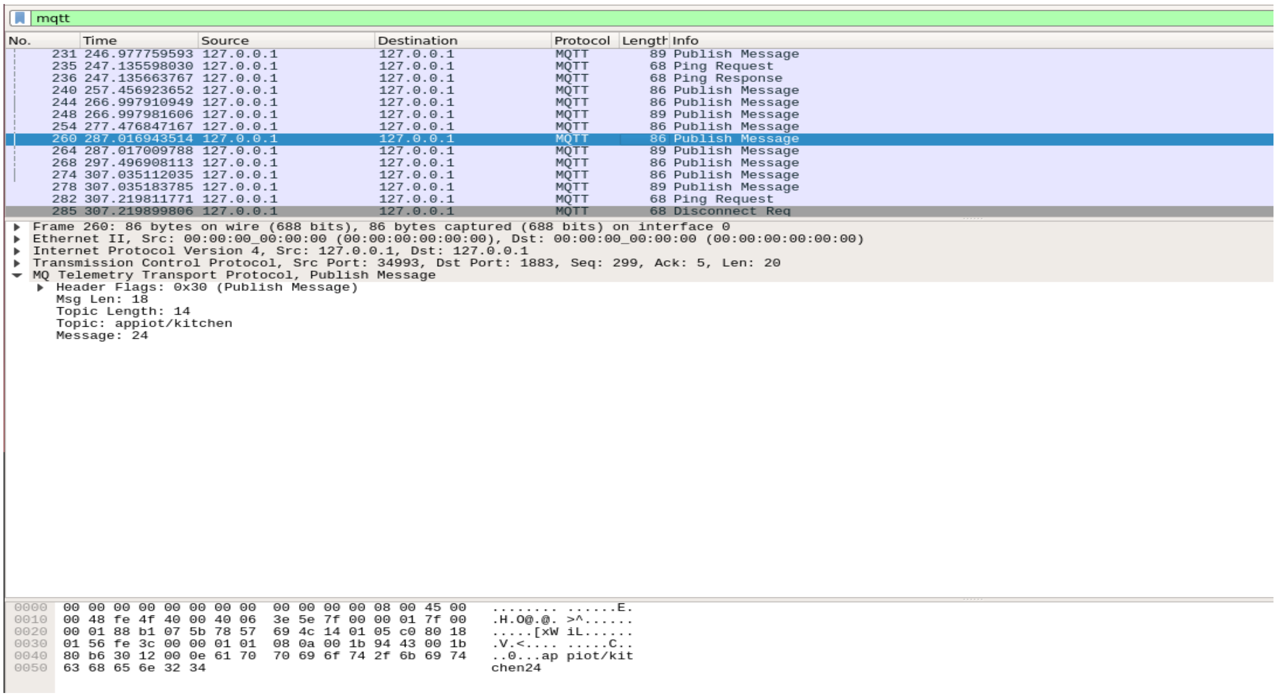
```
mqtt
No.    Time           Source      Destination  Protocol Length Info
   231 246.977759593 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   235 247.135598030 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   236 247.135663767 127.0.0.1   127.0.0.1    MQTT       68 Ping Response
   240 257.456923652 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   244 266.997910949 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   248 266.997981606 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   254 277.476847167 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   260 287.016943514 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   264 287.017009788 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   268 297.496908113 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   274 307.035112035 127.0.0.1   127.0.0.1    MQTT       86 Publish Message
   278 307.035183785 127.0.0.1   127.0.0.1    MQTT       89 Publish Message
   282 307.219811771 127.0.0.1   127.0.0.1    MQTT       68 Ping Request
   285 307.219899806 127.0.0.1   127.0.0.1    MQTT       68 Disconnect Req
▶ Frame 264: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 45522, Dst Port: 1883, Seq: 338, Ack: 5, Len: 23
▼ MQ Telemetry Transport Protocol, Publish Message
  ▶ Header Flags: 0x30 (Publish Message)
    Msg Len: 21
    Topic Length: 17
    Topic: appiot/livingroom
    Message: 29
```

```
usertp@usertp-VirtualBox: ~
  GNU nano 2.2.6              File: publisher2.py

import paho.mqtt.client as mqtt
import random
import time
import threading

def publish_temperature(topic):
       client = mqtt.Client(protocol=mqtt.MQTTv31)
       client.connect("localhost")
       while True:
             temp = random.randint(10,30)
             client.publish(topic,temp)
             time.sleep(20)
k = threading.Thread(target=publish_temperature,args=("appiot/kitchen",))
l = threading.Thread(target=publish_temperature,args=("appiot/livingroom",))
k.start()
l.start()
k.join()
l.join()

                [ Read 18 lines (Warning: No write permission) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

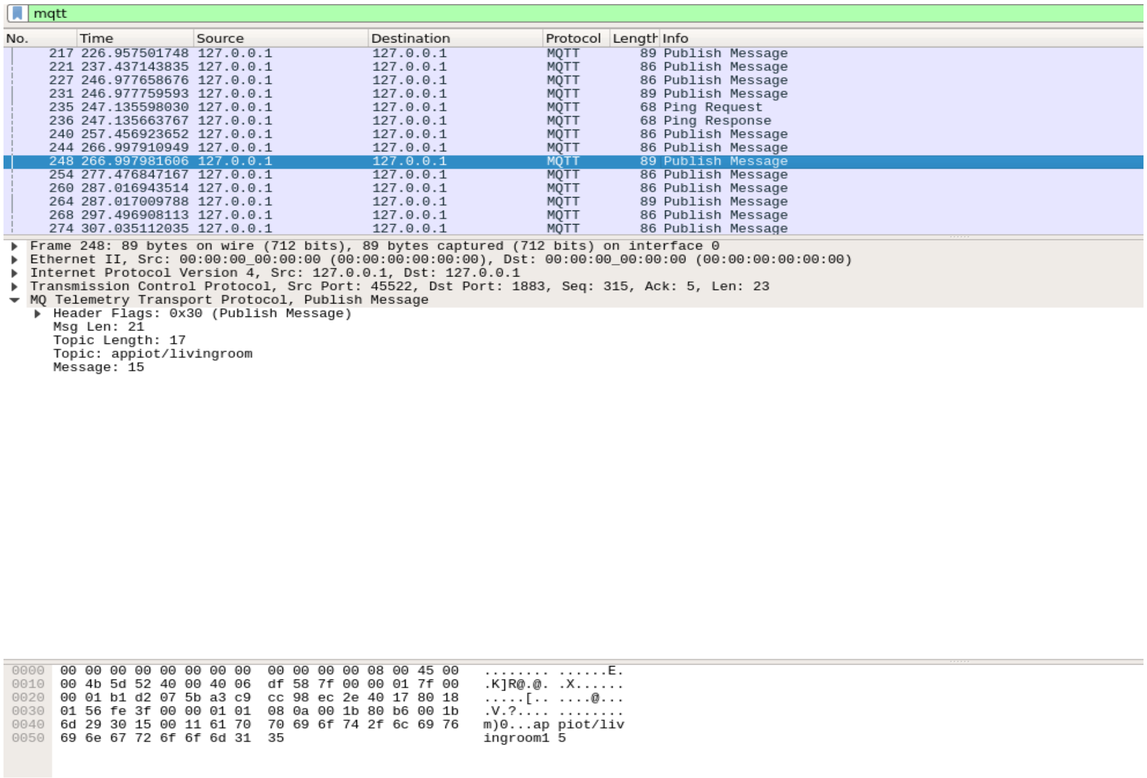
### From the above publisher2.py code:

I have created two Thread objects “k” and “l”, one for each MQTT topic, using the Thread constructor from the threading module. Each thread calls the publish_temperature function with a different topic argument i.e., appiot/kitchen and appiot/livingroom. Both the threads are started using start() method. Then, we can wait for both threads to complete using the join() method.

Below are the Wireshark captures:

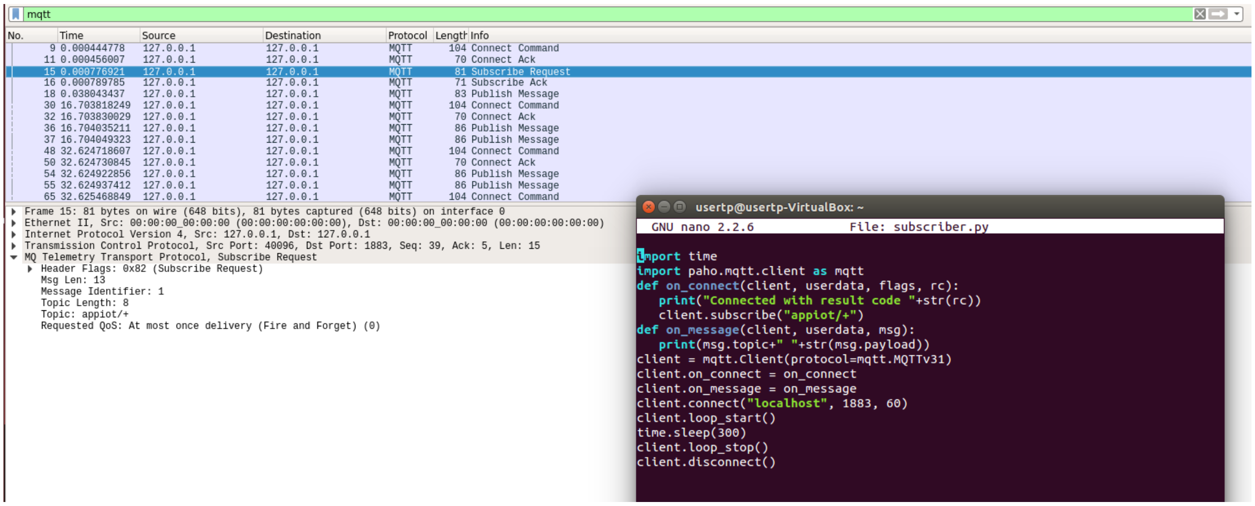


```
mqtt
No.      Time           Source      Destination  Protocol  Length Info
    231 246.977759593 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    235 247.135598030 127.0.0.1   127.0.0.1    MQTT        68 Ping Request
    236 247.135663767 127.0.0.1   127.0.0.1    MQTT        68 Ping Response
    240 257.456923652 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    244 266.997910949 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    248 266.997981606 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    254 277.476847167 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    260 287.016943514 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    264 287.017009788 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    268 297.496908113 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    274 307.035112035 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    278 307.035183785 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    282 307.219811771 127.0.0.1   127.0.0.1    MQTT        68 Ping Request
    285 307.219899806 127.0.0.1   127.0.0.1    MQTT        68 Disconnect Req
▶ Frame 260: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 34993, Dst Port: 1883, Seq: 299, Ack: 5, Len: 20
▼ MQ Telemetry Transport Protocol, Publish Message
  ▶ Header Flags: 0x30 (Publish Message)
    Msg Len: 18
    Topic Length: 14
    Topic: appiot/kitchen
    Message: 24

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 48 fe 4f 40 00 40 06  3e 5e 7f 00 00 01 7f 00   .H.O@.@. >^......
0020  00 01 88 b1 07 5b 78 57  69 4c 14 01 05 c0 80 18   .....[xW iL......
0030  01 56 fe 3c 00 00 01 01  08 0a 00 1b 94 43 00 1b   .V.<.... .....C..
0040  80 b6 30 12 00 0e 61 70  70 69 6f 74 2f 6b 69 74   ..0...ap piot/kit
0050  63 68 65 6e 32 34                                  chen24
```



```
mqtt
No.      Time           Source      Destination  Protocol  Length Info
    217 226.957501748 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    221 237.437143835 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    227 246.977658676 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    231 246.977759593 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    235 247.135598030 127.0.0.1   127.0.0.1    MQTT        68 Ping Request
    236 247.135663767 127.0.0.1   127.0.0.1    MQTT        68 Ping Response
    240 257.456923652 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    244 266.997910949 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    248 266.997981606 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    254 277.476847167 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    260 287.016943514 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    264 287.017009788 127.0.0.1   127.0.0.1    MQTT        89 Publish Message
    268 297.496908113 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
    274 307.035112035 127.0.0.1   127.0.0.1    MQTT        86 Publish Message
▶ Frame 248: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 45522, Dst Port: 1883, Seq: 315, Ack: 5, Len: 23
▼ MQ Telemetry Transport Protocol, Publish Message
  ▶ Header Flags: 0x30 (Publish Message)
    Msg Len: 21
    Topic Length: 17
    Topic: appiot/livingroom
    Message: 15

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 4b 5d 52 40 00 40 06  df 58 7f 00 00 01 7f 00   .K]R@.@. .X......
0020  00 01 b1 d2 07 5b a3 c9  cc 98 ec 2e 40 17 80 18   .....[.. ....@...
0030  01 56 fe 3f 00 00 01 01  08 0a 00 1b 80 b6 00 1b   .V.?.... ........
0040  6d 29 30 15 00 11 61 70  70 69 6f 74 2f 6c 69 76   m)0...ap piot/liv
0050  69 6e 67 72 6f 6f 6d 31  35                        ingroom1 5
```

### 4. Modify the subscriber to obtain information on both the kitchen and living room using only one subscription.



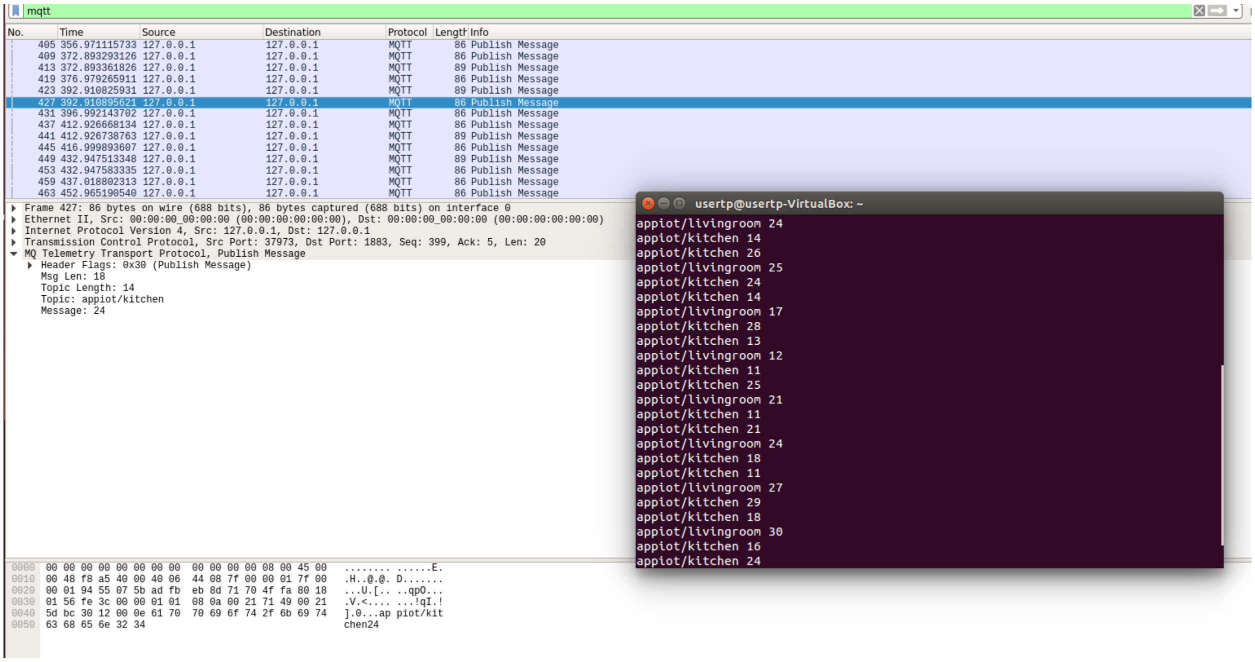
```
mqtt
No.  Time         Source      Destination  Protocol Length Info
   9 0.000444778  127.0.0.1   127.0.0.1    MQTT     104 Connect Command
  11 0.000456007  127.0.0.1   127.0.0.1    MQTT      70 Connect Ack
  15 0.000776921  127.0.0.1   127.0.0.1    MQTT      81 Subscribe Request
  16 0.000789785  127.0.0.1   127.0.0.1    MQTT      71 Subscribe Ack
  18 0.038043437  127.0.0.1   127.0.0.1    MQTT      83 Publish Message
  30 16.703818249 127.0.0.1   127.0.0.1    MQTT     104 Connect Command
  32 16.703830029 127.0.0.1   127.0.0.1    MQTT      70 Connect Ack
  36 16.704035211 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
  37 16.704049323 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
  48 32.624718607 127.0.0.1   127.0.0.1    MQTT     104 Connect Command
  50 32.624730845 127.0.0.1   127.0.0.1    MQTT      70 Connect Ack
  54 32.624922856 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
  55 32.624937412 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
  65 32.625468849 127.0.0.1   127.0.0.1    MQTT     104 Connect Command
▸ Frame 15: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 40096, Dst Port: 1883, Seq: 39, Ack: 5, Len: 15
▾ MQ Telemetry Transport Protocol, Subscribe Request
  ▸ Header Flags: 0x82 (Subscribe Request)
    Msg Len: 13
    Message Identifier: 1
    Topic Length: 8
    Topic: appiot/+
    Requested QoS: At most once delivery (Fire and Forget) (0)
```

```
usertp@usertp-VirtualBox: ~
  GNU nano 2.2.6              File: subscriber.py

import time
import paho.mqtt.client as mqtt
def on_connect(client, userdata, flags, rc):
   print("Connected with result code "+str(rc))
   client.subscribe("appiot/+")
def on_message(client, userdata, msg):
   print(msg.topic+" "+str(msg.payload))
client = mqtt.Client(protocol=mqtt.MQTTv31)
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
client.loop_start()
time.sleep(300)
client.loop_stop()
client.disconnect()
```

## From the above code:

The subscriber receives messages for the "appiot/temp" and "appiot/livingroom" topics.

Below are the Wireshark and terminal captures:



```
mqtt
No.  Time          Source      Destination  Protocol Length Info
 405 356.971115733 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 409 372.893293126 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 413 372.893361826 127.0.0.1   127.0.0.1    MQTT      89 Publish Message
 419 376.979265911 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 423 392.910825931 127.0.0.1   127.0.0.1    MQTT      89 Publish Message
 427 392.910895621 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 431 396.992143702 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 437 412.926668134 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 441 412.926738763 127.0.0.1   127.0.0.1    MQTT      89 Publish Message
 445 416.999893607 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 449 432.947513348 127.0.0.1   127.0.0.1    MQTT      89 Publish Message
 453 432.947583335 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 459 437.018802313 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
 463 452.965190540 127.0.0.1   127.0.0.1    MQTT      86 Publish Message
▸ Frame 427: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 37973, Dst Port: 1883, Seq: 399, Ack: 5, Len: 20
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 18
    Topic Length: 14
    Topic: appiot/kitchen
    Message: 24

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 48 f8 a5 40 00 40 06  44 08 7f 00 00 01 7f 00   .H..@.@. D.......
0020  00 01 94 55 07 5b ad fb  eb 8d 71 70 4f fa 80 18   ...U.[.. ..qpO...
0030  01 56 fe 3c 00 00 01 01  08 0a 00 21 71 49 00 21   .V.<.... ...!qI.!
0040  5d bc 30 12 00 0e 61 70  70 69 6f 74 2f 6b 69 74   ].0...ap piot/kit
0050  63 68 65 6e 32 34                                  chen24
```

```
usertp@usertp-VirtualBox: ~
appiot/livingroom 24
appiot/kitchen 14
appiot/kitchen 26
appiot/livingroom 25
appiot/kitchen 24
appiot/kitchen 14
appiot/livingroom 17
appiot/kitchen 28
appiot/kitchen 13
appiot/livingroom 12
appiot/kitchen 11
appiot/kitchen 25
appiot/livingroom 21
appiot/kitchen 11
appiot/kitchen 21
appiot/livingroom 24
appiot/kitchen 18
appiot/kitchen 11
appiot/livingroom 27
appiot/kitchen 29
appiot/kitchen 18
appiot/livingroom 30
appiot/kitchen 16
appiot/kitchen 24
```