PROJECT

Shubhika GARG

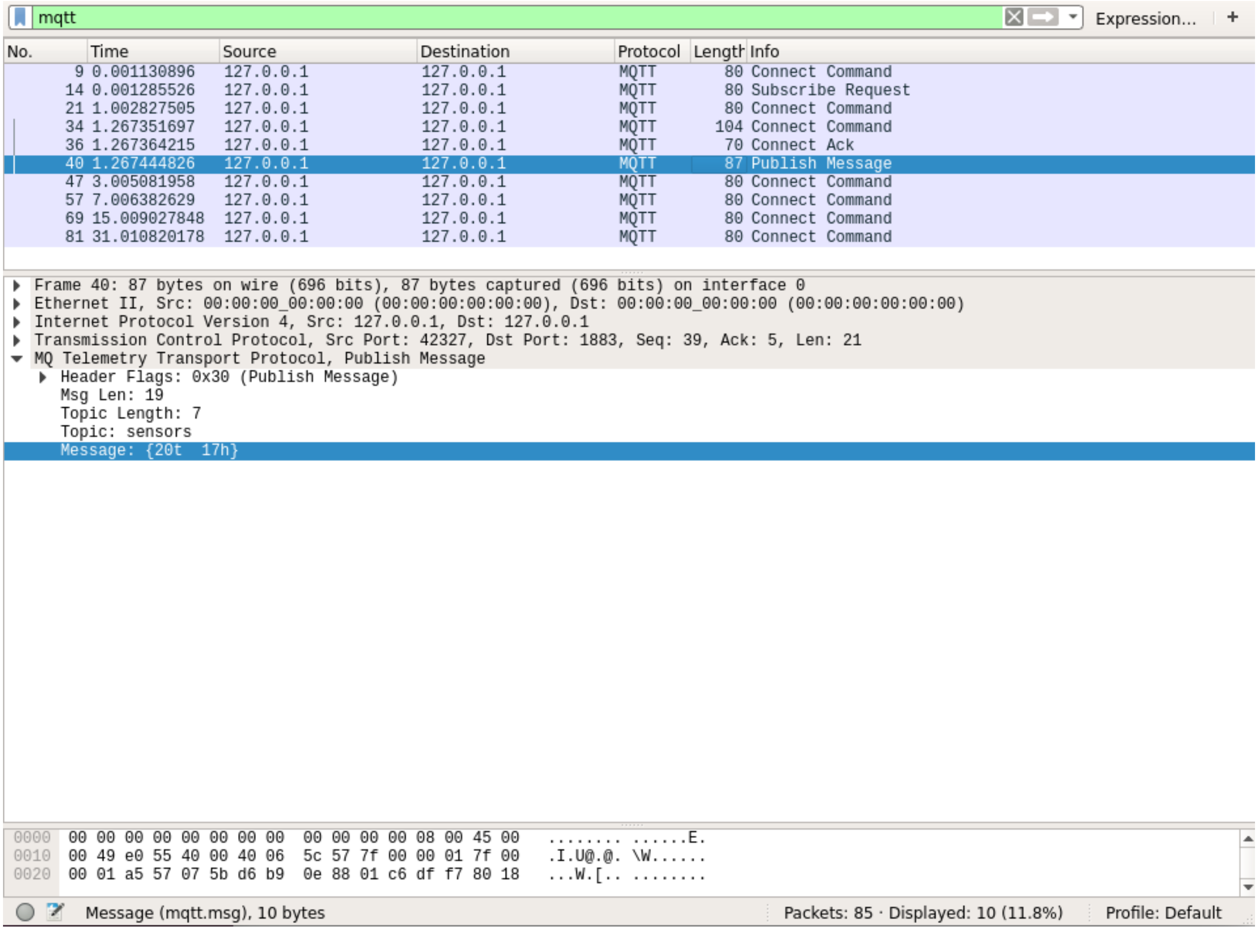
1. A company wants to deploy 10000 devices in a City. Each device (Raspberry) includes two sensors, to monitor humidity and temperature. The devices send the data to a data centre. In case the temperature and humidity are collected twice per day:

a) What is the application protocol that you advise to use? Why?

I would advise using MQTT Protocol. With MQTT, devices can easily publish their sensor data to a central broker, which can then be consumed by the data centre. MQTT also grants a flexible and extensible message format that can accommodate sensor data from multiple sources. With 10,000 devices sending data to a central data centre, scalability is an important consideration. MQTT is well-suited to handle this scale, as the broker can easily handle a large number of incoming messages, and the devices can efficiently transmit their data to the broker without requiring excessive bandwidth or processing power. MQTT supports Quality of Service (QoS) levels that allow for reliable message delivery even in the face of network disruptions or other issues. MQTT also has better support for unreliable network conditions and intermittent connectivity

b) Implement the device (device.py) and the data centre collector (datacenter.py) using Python and a library of your choice. The script "device.py" sends the data in a format of your choice (justify why you choose this format) to "datacenter.py" following the proposed protocol. The data size is 10 Bytes.

Wireshark capture:



```
No.   Time         Source      Destination  Protocol Length Info
    9 0.001130896  127.0.0.1   127.0.0.1    MQTT       80 Connect Command
   14 0.001285526  127.0.0.1   127.0.0.1    MQTT       80 Subscribe Request
   21 1.002827505  127.0.0.1   127.0.0.1    MQTT       80 Connect Command
   34 1.267351697  127.0.0.1   127.0.0.1    MQTT      104 Connect Command
   36 1.267364215  127.0.0.1   127.0.0.1    MQTT       70 Connect Ack
   40 1.267444826  127.0.0.1   127.0.0.1    MQTT       87 Publish Message
   47 3.005081958  127.0.0.1   127.0.0.1    MQTT       80 Connect Command
   57 7.006382629  127.0.0.1   127.0.0.1    MQTT       80 Connect Command
   69 15.009027848 127.0.0.1   127.0.0.1    MQTT       80 Connect Command
   81 31.010820178 127.0.0.1   127.0.0.1    MQTT       80 Connect Command
```

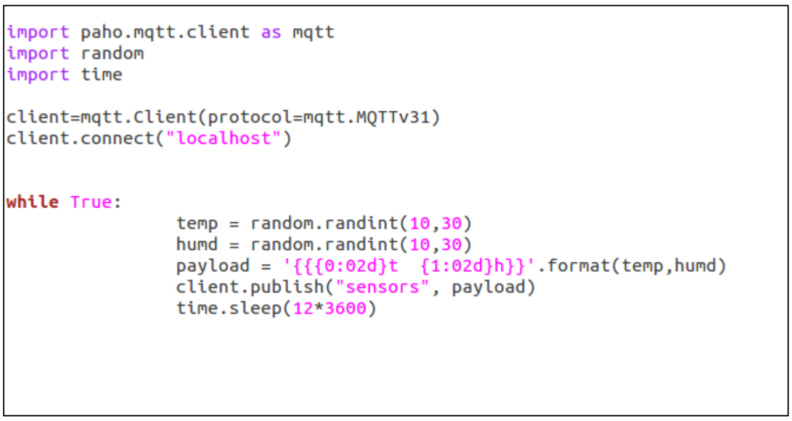
```
▸ Frame 40: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 42327, Dst Port: 1883, Seq: 39, Ack: 5, Len: 21
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 19
    Topic Length: 7
    Topic: sensors
    Message: {20t  17h}
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 49 e0 55 40 00 40 06  5c 57 7f 00 00 01 7f 00   .I.U@.@. \W......
0020  00 01 a5 57 07 5b d6 b9  0e 88 01 c6 df f7 80 18   ...W.[.. ........
```

```
Message (mqtt.msg), 10 bytes          Packets: 85 · Displayed: 10 (11.8%)          Profile: Default
```

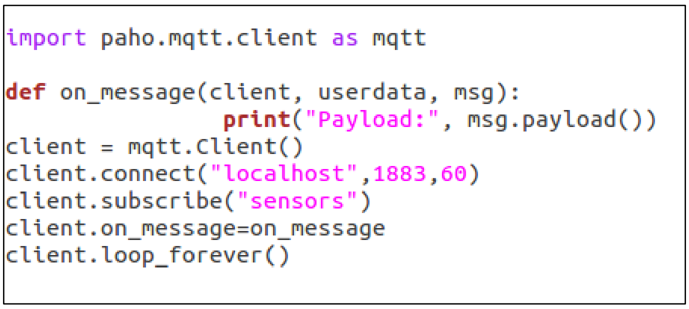
Python Codes:

device.py:



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

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


while True:
                temp = random.randint(10,30)
                humd = random.randint(10,30)
                payload = '{{{0:02d}t  {1:02d}h}}'.format(temp,humd)
                client.publish("sensors", payload)
                time.sleep(12*3600)
```

datacenter.py:

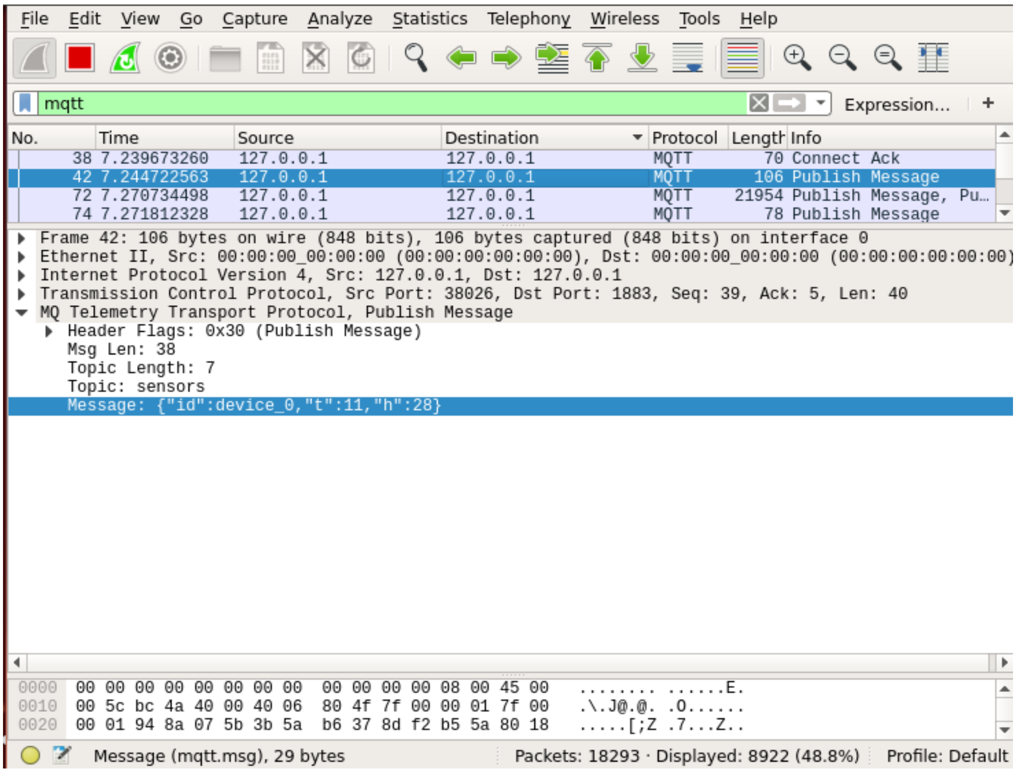


```
import paho.mqtt.client as mqtt

def on_message(client, userdata, msg):
                print("Payload:", msg.payload())
client = mqtt.Client()
client.connect("localhost",1883,60)
client.subscribe("sensors")
client.on_message=on_message
client.loop_forever()
```

Message format: ***payload = '{{{0:02d}t {1:02d}h}}'.format(temp,humd)***

This is a compact and readable format that represents temperature and humidity values as two digits each, with a leading zero if necessary to ensure a consistent format with no separators or other overhead. This can be useful when bandwidth or message size is a concern.

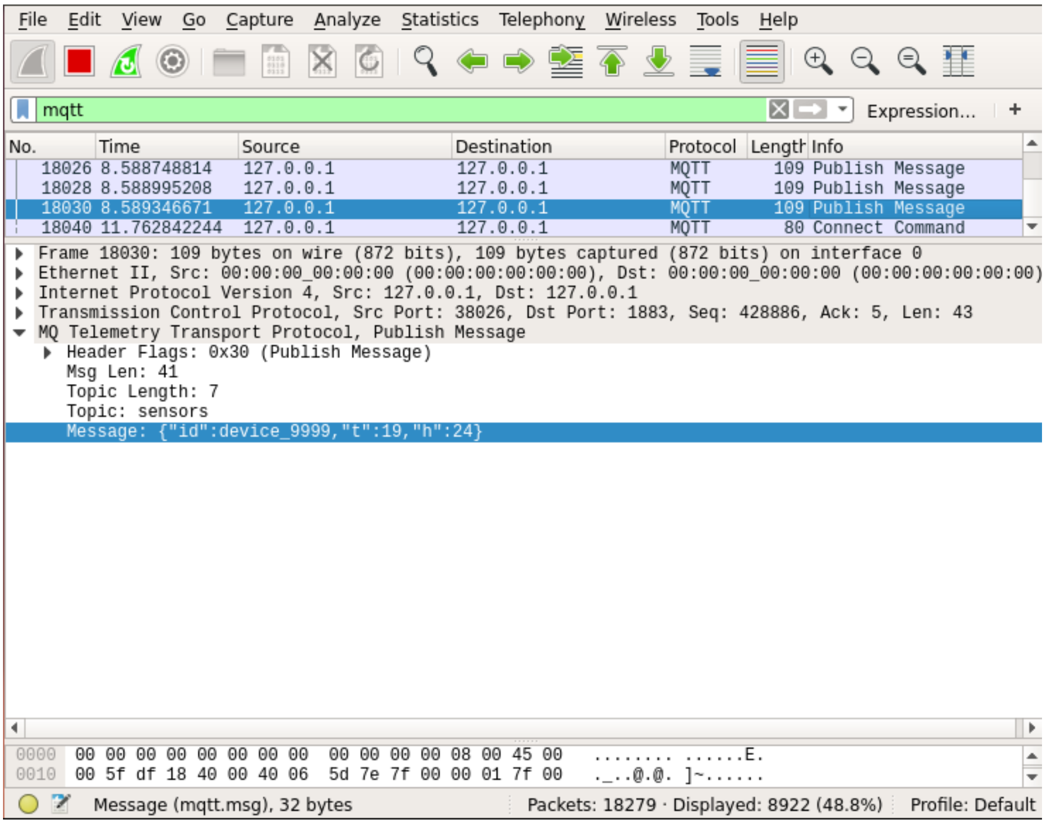
c) Write a script that simulates 10000 devices. Run the simulation (you can change the simulation time scale).



mqtt

```
No.    Time            Source         Destination      Protocol  Length Info
    38 7.239673260     127.0.0.1      127.0.0.1        MQTT          70 Connect Ack
    42 7.244722563     127.0.0.1      127.0.0.1        MQTT         106 Publish Message
    72 7.270734498     127.0.0.1      127.0.0.1        MQTT       21954 Publish Message, Pu…
    74 7.271812328     127.0.0.1      127.0.0.1        MQTT          78 Publish Message

▸ Frame 42: 106 bytes on wire (848 bits), 106 bytes captured (848 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: 38026, Dst Port: 1883, Seq: 39, Ack: 5, Len: 40
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 38
    Topic Length: 7
    Topic: sensors
    Message: {"id":device_0,"t":11,"h":28}

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5c bc 4a 40 00 40 06  80 4f 7f 00 00 01 7f 00   .\.J@.@. .O......
0020  00 01 94 8a 07 5b 3b 5a  b6 37 8d f2 b5 5a 80 18   .....[;Z .7...Z..
```

Message (mqtt.msg), 29 bytes · Packets: 18293 · Displayed: 8922 (48.8%) · Profile: Default



```
No.   Time          Source       Destination   Protocol Length Info
18026 8.588748814   127.0.0.1    127.0.0.1     MQTT        109 Publish Message
18028 8.588995208   127.0.0.1    127.0.0.1     MQTT        109 Publish Message
18030 8.589346671   127.0.0.1    127.0.0.1     MQTT        109 Publish Message
18040 11.762842244  127.0.0.1    127.0.0.1     MQTT         80 Connect Command

▸ Frame 18030: 109 bytes on wire (872 bits), 109 bytes captured (872 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: 38026, Dst Port: 1883, Seq: 428886, Ack: 5, Len: 43
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 41
    Topic Length: 7
    Topic: sensors
    Message: {"id":device_9999,"t":19,"h":24}

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 5f df 18 40 00 40 06  5d 7e 7f 00 00 01 7f 00   ._..@.@. ]~......

Message (mqtt.msg), 32 bytes      Packets: 18279 · Displayed: 8922 (48.8%)      Profile: Default
```

I chose the message format ***payload = '{{"id":{0},"t":{1},"h":{2}}}'.format(did,temp,humd)***

This format uses key-value pairs to represent the data, where the keys are strings that represent device id, temperature and humidity ("id", "t", "h") and the values are the actual data. Using this format allowed the data centre to easily parse and interpret the data received from the devices, while also allowing the devices to efficiently send only the necessary data without any unnecessary overhead.

d) How many messages are received per day ? What is the total amount of data transferred in Bytes. Hint: first compute it mathematically then confirm it in practice.

Every device gives two sensor readings per day (as temperature and humidity are collected twice per day). We have 10,000 devices.

Number of messages received per day = 2 * 10000 = 20,000 messages/day

Each message is 10 bytes.

20,000 messages/day * 10 bytes/message = 200,000 bytes/day

2. In case the temperature and humidity are sent only if a threshold is exceeded.

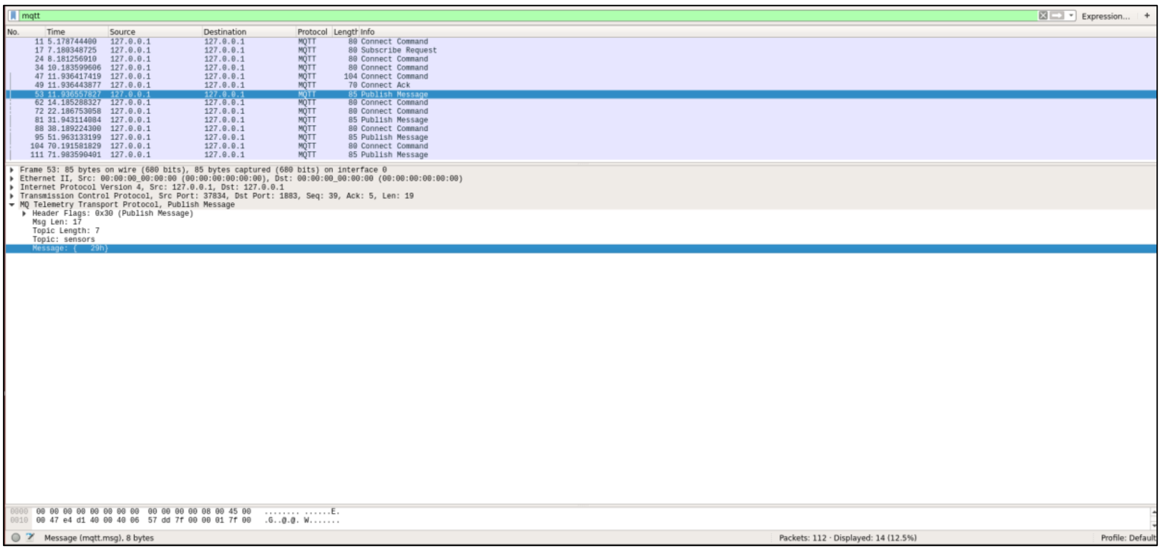
a) What is the application protocol that you advise to use? Why?

I would advise using MQTT Protocol. With MQTT, devices can publish messages to the broker only when the temperature or humidity exceeds a certain threshold, rather than sending data at fixed intervals or continuously. This selective transmission of data helps to reduce the amount of data being transmitted and processed, which can be important for large-scale IoT applications. Using MQTT in this scenario allows for efficient, reliable, and selective transmission of temperature and humidity data from the Raspberry Pi devices to the central data centre, while also minimizing the bandwidth and processing power required by the devices.

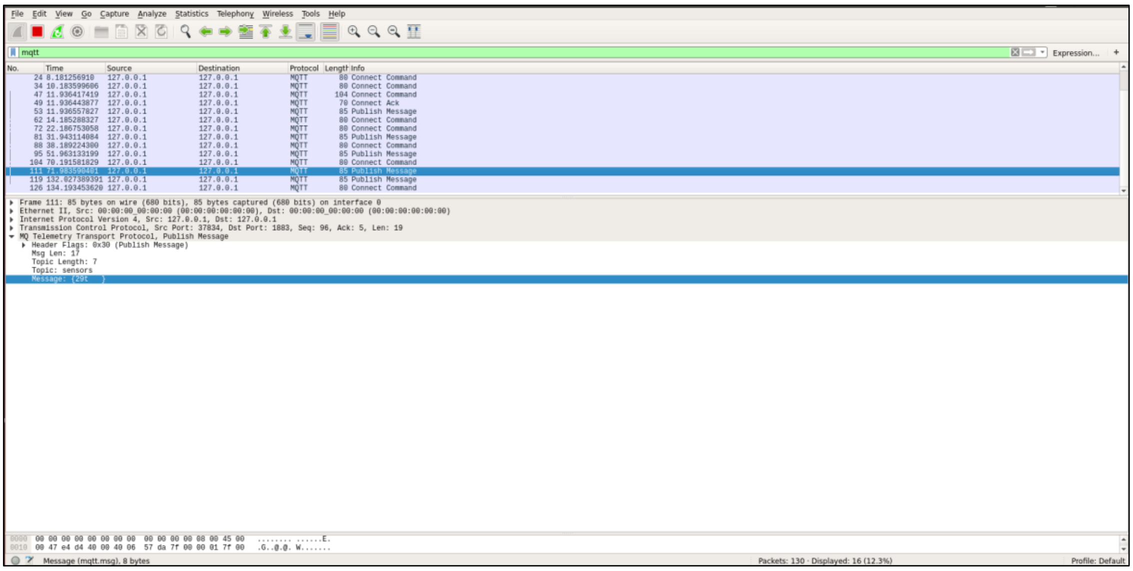
b) Implement the device (device.py) and data centre collector (datacenter.py) using Python and a library of your choice. The script “device.py” sends the data in a format of

your choice (justify why you choose this format) to "datacenter.py" following the proposed protocol. The message size is 8 Bytes if it includes humidity or temperature only or 10 Bytes if both.

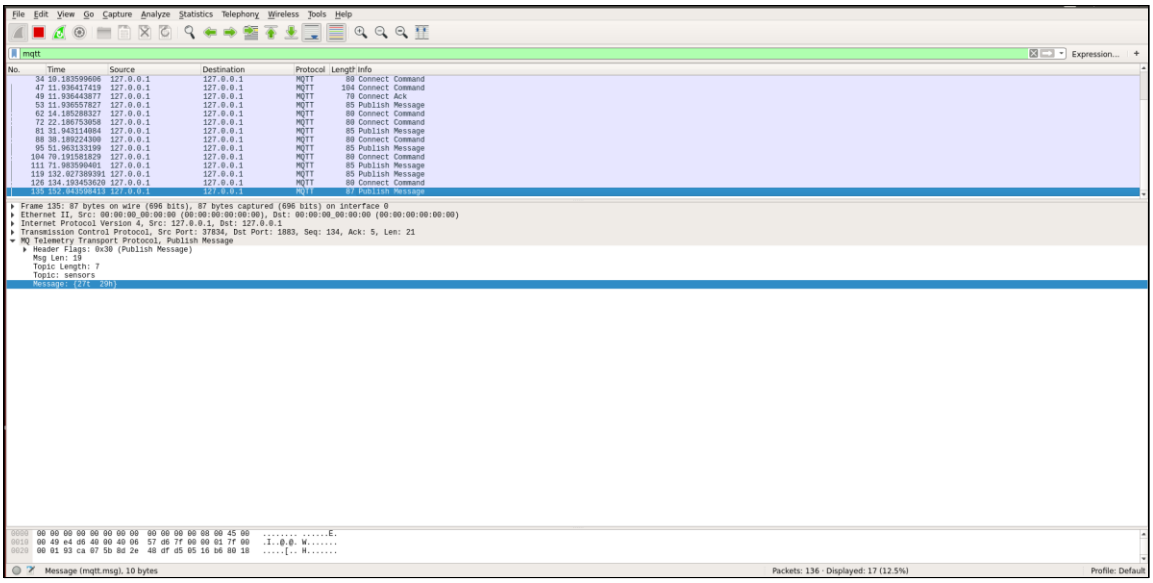
Wireshark Captures:



```
No.   Time          Source      Destination  Protocol Length Info
   11 5.178744400   127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   17 7.180348725   127.0.0.1   127.0.0.1    MQTT      80 Subscribe Request
   24 8.181256910   127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   34 10.183599606  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   47 11.936417419  127.0.0.1   127.0.0.1    MQTT     104 Connect Command
   49 11.936443877  127.0.0.1   127.0.0.1    MQTT      70 Connect Ack
   53 11.936557827  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
   62 14.185288327  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   72 22.186753058  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   81 31.943114084  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
   88 38.189224300  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   95 51.963133199  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  104 70.191581829  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
  111 71.983590401  127.0.0.1   127.0.0.1    MQTT      85 Publish Message

▸ Frame 53: 85 bytes on wire (680 bits), 85 bytes captured (680 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: 37834, Dst Port: 1883, Seq: 39, Ack: 5, Len: 19
▾ MQ Telemetry Transport Protocol, Publish Message
   ▸ Header Flags: 0x30 (Publish Message)
     Msg Len: 17
     Topic Length: 7
     Topic: sensors
     Message: {   29h}

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 47 e4 d1 40 00 40 06  57 dd 7f 00 00 01 7f 00   .G..@.@. W.......

Message (mqtt.msg), 8 bytes          Packets: 112 · Displayed: 14 (12.5%)          Profile: Default
```



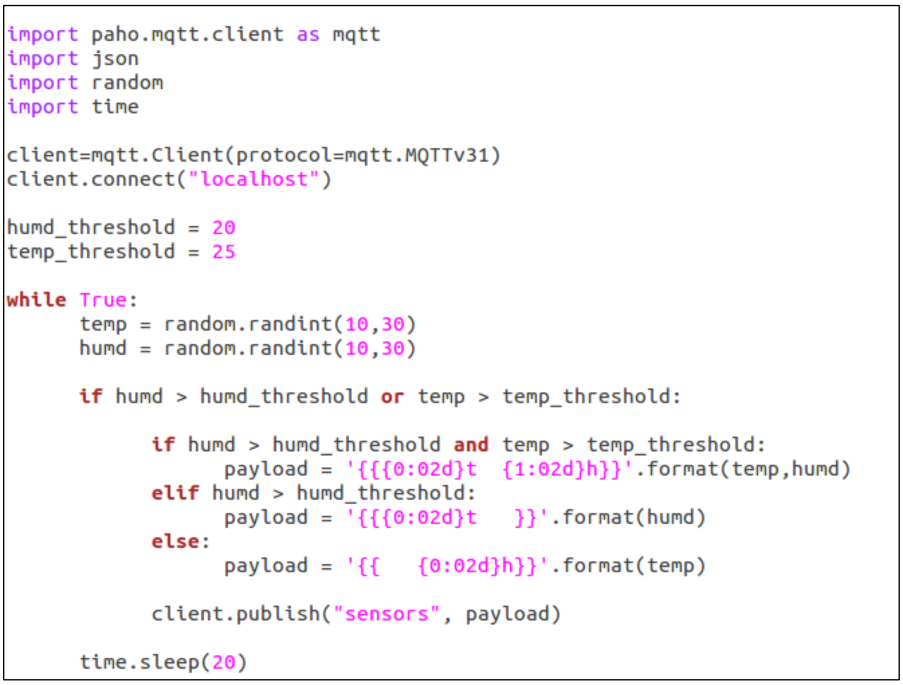
```
No.   Time          Source      Destination  Protocol Length Info
   24 8.181256910   127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   34 10.183599606  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   47 11.936417419  127.0.0.1   127.0.0.1    MQTT     104 Connect Command
   49 11.936443877  127.0.0.1   127.0.0.1    MQTT      70 Connect Ack
   53 11.936557827  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
   62 14.185288327  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   72 22.186753058  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   81 31.943114084  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
   88 38.189224300  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   95 51.963133199  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  104 70.191581829  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
  111 71.983590401  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  119 132.027389391 127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  126 134.193453620 127.0.0.1   127.0.0.1    MQTT      80 Connect Command

▸ Frame 111: 85 bytes on wire (680 bits), 85 bytes captured (680 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: 37834, Dst Port: 1883, Seq: 96, Ack: 5, Len: 19
▾ MQ Telemetry Transport Protocol, Publish Message
   ▸ Header Flags: 0x30 (Publish Message)
     Msg Len: 17
     Topic Length: 7
     Topic: sensors
     Message: {29t   }

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 47 e4 d4 40 00 40 06  57 da 7f 00 00 01 7f 00   .G..@.@. W.......

Message (mqtt.msg), 8 bytes          Packets: 130 · Displayed: 16 (12.3%)          Profile: Default
```



```
No.   Time          Source      Destination  Protocol Length Info
   34 10.183599606  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   47 11.936417419  127.0.0.1   127.0.0.1    MQTT     104 Connect Command
   49 11.936443877  127.0.0.1   127.0.0.1    MQTT      70 Connect Ack
   53 11.936557827  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
   62 14.185288327  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   72 22.186753058  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   81 31.943114084  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
   88 38.189224300  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
   95 51.963133199  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  104 70.191581829  127.0.0.1   127.0.0.1    MQTT      80 Connect Command
  111 71.983590401  127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  119 132.027389391 127.0.0.1   127.0.0.1    MQTT      85 Publish Message
  126 134.193453620 127.0.0.1   127.0.0.1    MQTT      80 Connect Command
  135 152.043598413 127.0.0.1   127.0.0.1    MQTT      87 Publish Message

▸ Frame 135: 87 bytes on wire (696 bits), 87 bytes captured (696 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: 37834, Dst Port: 1883, Seq: 134, Ack: 5, Len: 21
▾ MQ Telemetry Transport Protocol, Publish Message
   ▸ Header Flags: 0x30 (Publish Message)
     Msg Len: 19
     Topic Length: 7
     Topic: sensors
     Message: {27t  29h}

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 49 e4 d6 40 00 40 06  57 d6 7f 00 00 01 7f 00   .I..@.@. W.......
0020  00 01 93 ca 07 5b 8d 2e  48 df d5 05 16 b6 80 18   .....[.. H.......

Message (mqtt.msg), 10 bytes          Packets: 136 · Displayed: 17 (12.5%)          Profile: Default
```

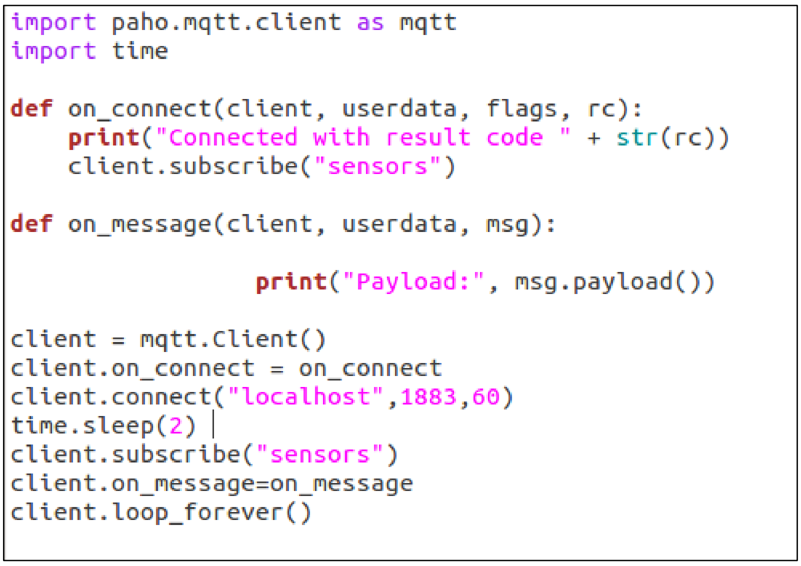
Python codes:

device.py:



```python
import paho.mqtt.client as mqtt
import json
import random
import time

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

humd_threshold = 20
temp_threshold = 25

while True:
      temp = random.randint(10,30)
      humd = random.randint(10,30)

      if humd > humd_threshold or temp > temp_threshold:

            if humd > humd_threshold and temp > temp_threshold:
                  payload = '{{{0:02d}t  {1:02d}h}}'.format(temp,humd)
            elif humd > humd_threshold:
                  payload = '{{{0:02d}t   }}'.format(humd)
            else:
                  payload = '{{   {0:02d}h}}'.format(temp)

            client.publish("sensors", payload)

      time.sleep(20)
```

datacenter.py:

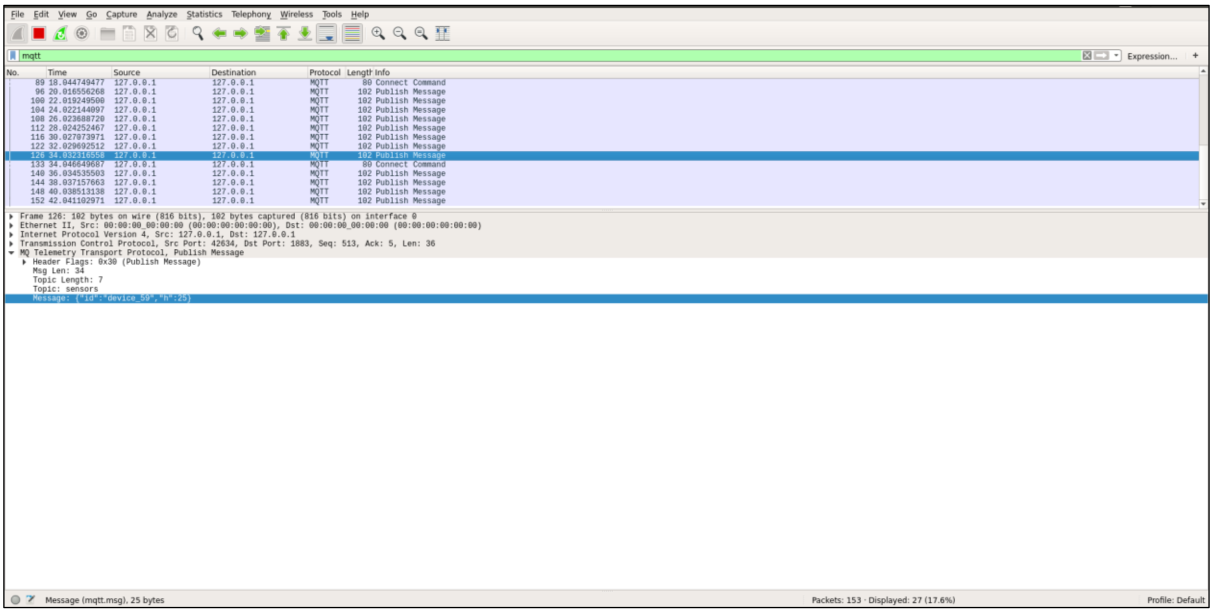


```python
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe("sensors")

def on_message(client, userdata, msg):

                 print("Payload:", msg.payload())

client = mqtt.Client()
client.on_connect = on_connect
client.connect("localhost",1883,60)
time.sleep(2)
client.subscribe("sensors")
client.on_message=on_message
client.loop_forever()
```

Choice of message format:

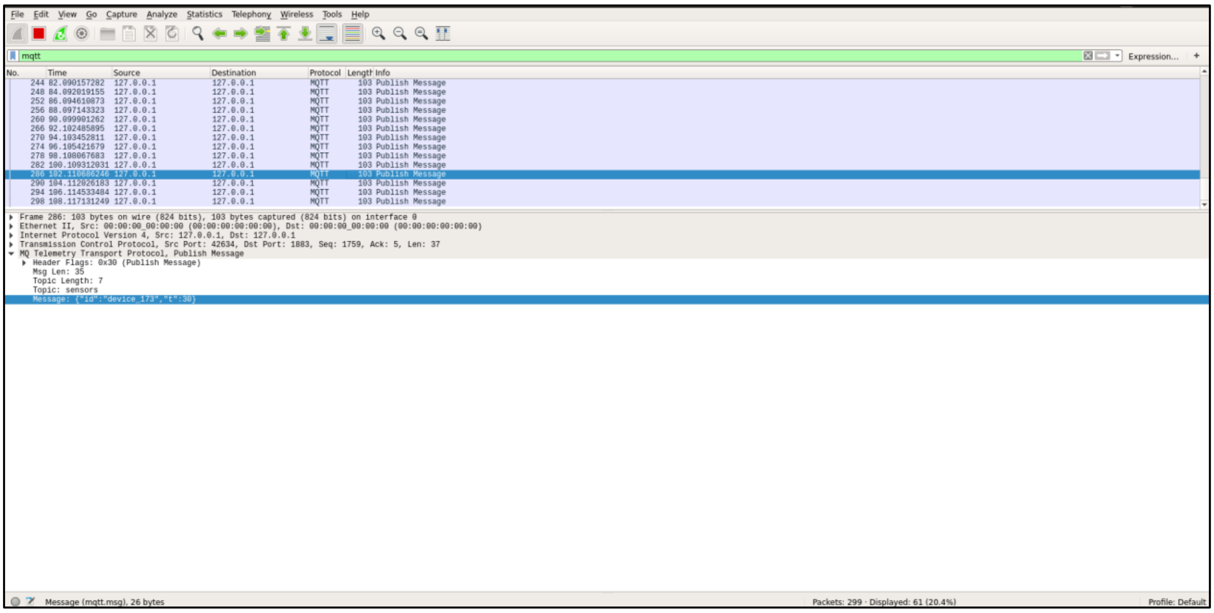
The format is compact, taking up only 10 bytes of data if both temperature and humidity are transmitted, or 8 bytes if only one value is transmitted. It is designed to use the minimum amount of data necessary to transmit the temperature and humidity values. It can be easily extended to include additional data fields or metadata (We can include a device ID field to identify the source of the data).

c) Write a script that simulates 10000 devices. Run the simulation (you can change the simulation time scale). Knowing that the probability that humidity exceeds the threshold is 0.5 and temperature exceeds the threshold is 0.2. The events are independent.



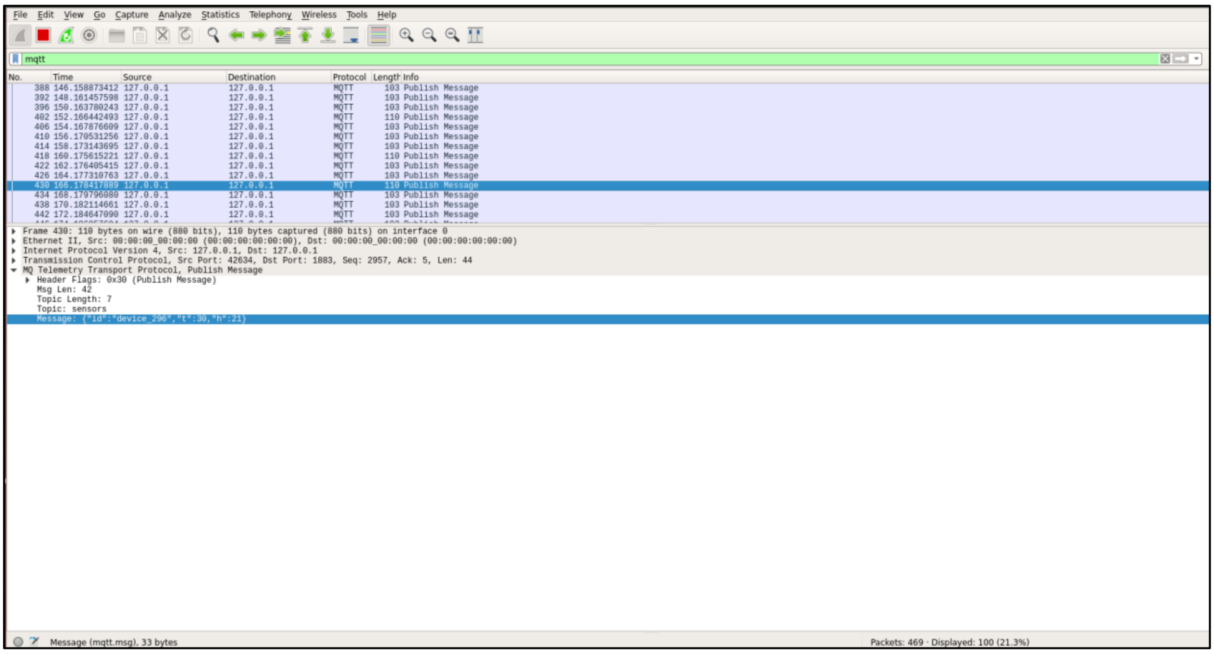
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
mqtt
Expression…
+

```
No.  Time          Source     Destination  Protocol Length Info
 89 18.044749477 127.0.0.1  127.0.0.1    MQTT      80 Connect Command
 96 20.016556268 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
100 22.019249500 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
104 24.022144097 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
108 26.023688720 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
112 28.024252467 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
116 30.027073971 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
122 32.029692512 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
126 34.032316558 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
133 34.046649687 127.0.0.1  127.0.0.1    MQTT      80 Connect Command
140 36.034535503 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
144 38.037157663 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
148 40.038513138 127.0.0.1  127.0.0.1    MQTT     102 Publish Message
152 42.041102971 127.0.0.1  127.0.0.1    MQTT     102 Publish Message

▸ Frame 126: 102 bytes on wire (816 bits), 102 bytes captured (816 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: 42634, Dst Port: 1883, Seq: 513, Ack: 5, Len: 36
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 34
    Topic Length: 7
    Topic: sensors
    Message: {"id":"device_59","h":25}

Message (mqtt.msg), 25 bytes        Packets: 153 · Displayed: 27 (17.6%)        Profile: Default
```



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
mqtt
Expression…
+

```
No.  Time           Source     Destination  Protocol Length Info
244 82.090157282  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
248 84.092019155  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
252 86.094610873  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
256 88.097143323  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
260 90.099901262  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
266 92.102485895  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
270 94.103452811  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
274 96.105421679  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
278 98.108067683  127.0.0.1  127.0.0.1    MQTT     103 Publish Message
282 100.109312031 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
286 102.110686246 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
290 104.112026183 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
294 106.114533484 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
298 108.117131249 127.0.0.1  127.0.0.1    MQTT     103 Publish Message

▸ Frame 286: 103 bytes on wire (824 bits), 103 bytes captured (824 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: 42634, Dst Port: 1883, Seq: 1759, Ack: 5, Len: 37
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 35
    Topic Length: 7
    Topic: sensors
    Message: {"id":"device_173","t":30}

Message (mqtt.msg), 26 bytes        Packets: 299 · Displayed: 61 (20.4%)        Profile: Default
```



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
mqtt

```
No.  Time           Source     Destination  Protocol Length Info
388 146.158873412 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
392 148.161457598 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
396 150.163780243 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
402 152.166442493 127.0.0.1  127.0.0.1    MQTT     110 Publish Message
406 154.167876609 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
410 156.170531256 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
414 158.173143695 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
418 160.175615221 127.0.0.1  127.0.0.1    MQTT     110 Publish Message
422 162.176405415 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
426 164.177310763 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
430 166.178417889 127.0.0.1  127.0.0.1    MQTT     110 Publish Message
434 168.179796080 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
438 170.182114661 127.0.0.1  127.0.0.1    MQTT     103 Publish Message
442 172.184647090 127.0.0.1  127.0.0.1    MQTT     103 Publish Message

▸ Frame 430: 110 bytes on wire (880 bits), 110 bytes captured (880 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: 42634, Dst Port: 1883, Seq: 2957, Ack: 5, Len: 44
▾ MQ Telemetry Transport Protocol, Publish Message
  ▸ Header Flags: 0x30 (Publish Message)
    Msg Len: 42
    Topic Length: 7
    Topic: sensors
    Message: {"id":"device_296","t":30,"h":21}

Message (mqtt.msg), 33 bytes        Packets: 469 · Displayed: 100 (21.3%)
```

d) How many messages are received per day ? What is the total amount of data transferred in Bytes. Hint: first compute it mathematically then confirm it in practice.

P(humidity exceeds threshold) * P(temperature exceeds threshold) = 0.5 * 0.2 = 0.1

P(humidity exceeds threshold) * P(temperature does not exceed threshold) = 0.5 * 0.8 = 0.4

P(temperature exceeds threshold) * P(humidity does not exceed threshold) = 0.2 * 0.5 = 0.1

The expected number of messages per day per device is:

(0.1 * 1 + 0.4 * 1 + 0.1 * 1) = 0.6

For 10,000 devices:

The expected total number of messages per day for 10,000 devices is:

10,000 * 0.6 = 6,000

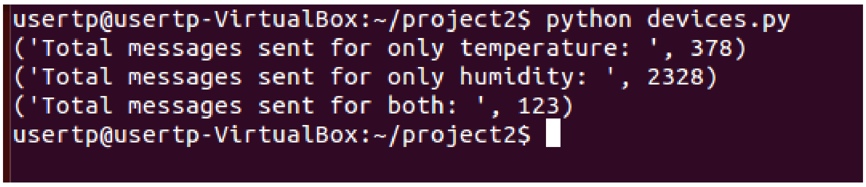
The expected total amount of data transferred per day can be calculated as:

(0.1 * 10 + 0.4 * 8 + 0.1 * 8) * 6,000 = 30,000 Bytes

The expected total amount of data transferred per day for 10,000 devices is 30,000 Bytes

Note:

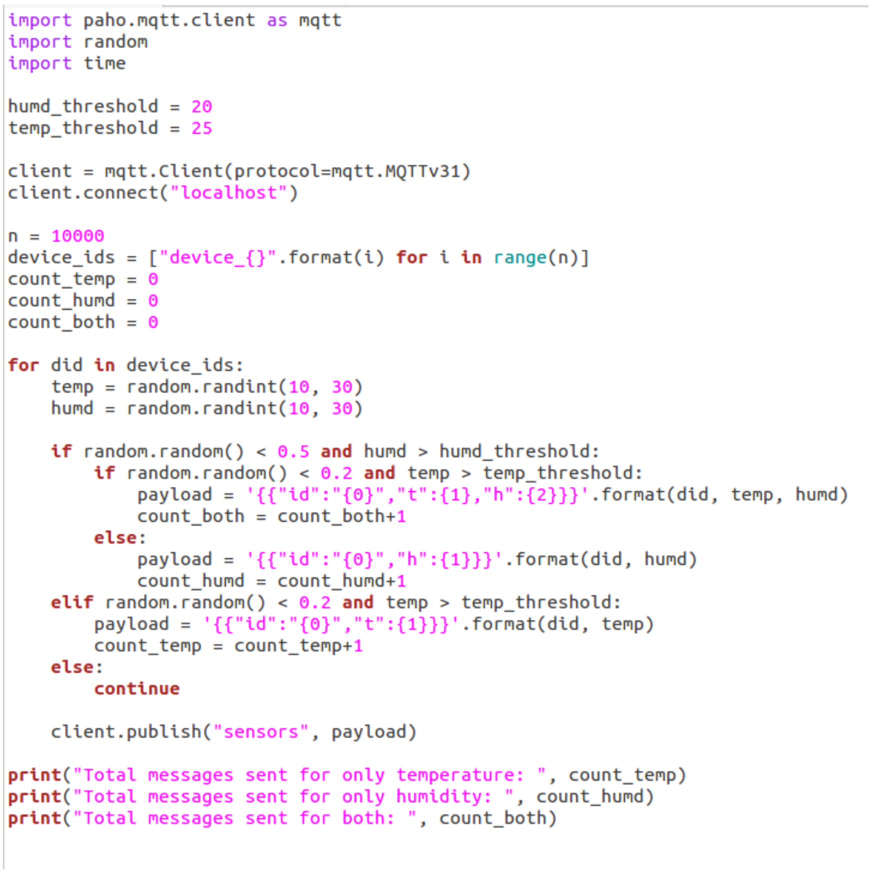
To compute this, I added three counter variables for all three cases in my Python code:



```
usertp@usertp-VirtualBox:~/project2$ python devices.py
('Total messages sent for only temperature: ', 378)
('Total messages sent for only humidity: ', 2328)
('Total messages sent for both: ', 123)
usertp@usertp-VirtualBox:~/project2$ 
```

So, when only either temperature or humidity is sent, the message size is 8 bytes while if both are sent, the size is 10 bytes:

Therefore, (378*8)+(2328*8)+(123*10) = 22,878 Bytes for 10,000 devices



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

humd_threshold = 20
temp_threshold = 25

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

n = 10000
device_ids = ["device_{}".format(i) for i in range(n)]
count_temp = 0
count_humd = 0
count_both = 0

for did in device_ids:
    temp = random.randint(10, 30)
    humd = random.randint(10, 30)

    if random.random() < 0.5 and humd > humd_threshold:
        if random.random() < 0.2 and temp > temp_threshold:
            payload = '{{"id":"{0}","t":{1},"h":{2}}}'.format(did, temp, humd)
            count_both = count_both+1
        else:
            payload = '{{"id":"{0}","h":{1}}}'.format(did, humd)
            count_humd = count_humd+1
    elif random.random() < 0.2 and temp > temp_threshold:
        payload = '{{"id":"{0}","t":{1}}}'.format(did, temp)
        count_temp = count_temp+1
    else:
        continue

    client.publish("sensors", payload)

print("Total messages sent for only temperature: ", count_temp)
print("Total messages sent for only humidity: ", count_humd)
print("Total messages sent for both: ", count_both)
```