Network Communication and Network Programming

Part 1 – Open-Ended Questions

1. What are the main differences between HTTP, MQTT, and CoAP?

Basis of	COAP	MQTT	HTTP
Abbreviation	Constrained Application Protocol	Message Queuing Telemetry Transport	Hypertext Transfer Protocol
Communication Type	It uses Request-Response model.	It uses Publish-Subscribe model	It uses Request-Response model.
Messaging Mode	This uses both Asynchronous and Synchronous.	This uses only Asynchronous	Synchronous
Transport layer protocol	This mainly uses User Datagram protocol(UDP)	This mainly uses Transmission Control protocol(TCP)	Mainly uses TCP
Header size	It has 4 bytes sized header	It has 2 bytes sized header	
RESTful based	Yes it uses REST principles	No it does not uses REST principles	REST-friendly

Basis of	COAP	MQTT	HTTP
Persistence support	It does not has such support	It supports and best used for live data communication	No built-in persistence (Stateless)
Message Labelling	It provides by adding labels to the messages.	It has no such feature.	No message labelling; uses URLs/headers for identification
Usability/Security	It is used in Utility area networks and has secured mechanism.	It is used in IoT applications and is secure	Used on the Web
Effectiveness	Effectiveness in LNN is excellent.	Effectiveness in LNN is low.	Poor (not optimized for constrained networks)
Communication Model	Communication model is one-one.	Communication model is many-many	One-to-one (client–server)

Resource is https://www.geeksforgeeks.org/computer-networks/difference-between-coap-and-mqtt-protocols/
added the http coloumn myself

2. Which protocol would you choose for:

◦ Sending temperature data every second

MQTT because lightweight and small header size. in addition it is pub-sub
so easily connect multiple temperature sensors

◦ Controlling a smart bulb (on/off)

CoAP to control the smart light bulb directly. No brokers needed and also it is lightweight.

◦ Uploading a large file

HTTP is for web and is suitable for large files upload

3. Explain QoS levels (0, 1, 2) in MQTT and give one use case for each.

QoS 0 :

At most once " Fire and Forget", no confirmation, Guaranteed delivery

Example:

Imagine a weather monitoring system publishing current temperature data every minute. If a few temperature updates are lost during transmission, it's acceptable, as the system will publish new updates shortly, and older data is less relevant.

QoS 1 :

At least once , with confirmation required (msgs may delivered more than once)

Example:

Consider a home automation system where a user sends a command to turn off the lights. The system wants to ensure the lights are turned off, so it uses QoS 1 to guarantee that the command is delivered, even if it means sending the command multiple times.

QoS 2 :

Exactly once , 2-phase commit

Example:

Let's say a financial application is processing a transaction. It's crucial that the transaction is only processed once. In this case, QoS 2 ensures that the transaction request is delivered and processed exactly once.

Resource https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

And examples from : https://medium.com/@durgeshparekh381/quality-of-service-in-mqtt-ensuring-efficient-and-reliable-communication-cfc1a6e36ef7

Why does CoAP use UDP instead of TCP?

1. Small amounts of data

Small amounts of data, such as sensor readings, are ideal to send using UDP. The overhead of setting up a TCP connection to send a small amount of data can quickly outweigh the data being sent. There is no setup however for UDP which makes data usage much more efficient.

2. Suitable for low power hardware

TCP communications require setting up a TCP connection first. For a device that stays online all the time (high-power devices), this connection can be held open. However, low-power devices must power down which in turn drops any connection. So, for low-power devices to use TCP, they must negotiate a new TCP connection each time. TCP negotiation takes tens to hundreds of milliseconds and requires a few round trips between the device and the server. Modem on-time is one of the most expensive aspects of a low-power device in terms of battery consumption and is one of the first things which needs to be optimised to achieve consistent low-power behaviour.

UDP on the other hand does not require any connection setup. Once the modem is powered on and connected to the cellular network a UDP packet can be flung off into the network. Depending on the system setup, the device can either then wait for an acknowledgement from the server, or just go back to a low-power state and hope the packet was received. This will save tens to hundreds of milliseconds of each transaction, which is significant over the life of the device.

3. Connection Stability

Connection stability is another important factor for devices in areas with poor cellular coverage. With TCP, not only does the connection take effort to set up, but it also needs to be stable. If the connection drops partway through a transmission then the whole process of connecting needs to be set up again. In areas where cellular coverage is patchy, it is not guaranteed that a connection will stay open reliably even for a few seconds. This results in many attempts to connect and send data consuming both battery and cellular data.

UDP might be the right choice, because it does not have any requirements for keeping a connection open.

4. Data Efficiency

> One downside of a TCP connection is that it needs to be kept alive. The protocol dictates that if there has been no communication for a fixed period of time, then the connection should timeout and be dropped. For devices that communicate infrequently, this can be a problem. The way to get around this is to send "keep-alive" packets, typically every 5 minutes. These keep-alive packets are about 64 bytes in size, but over the course of a month account for about half a megabyte of data, which on a small IoT data plan can be a significant proportion of the data allowance simply lost in overhead keeping the connection alive.

Resource https://medium.com/@webtolife/is-udp-the-choice-for-iot-e230d069eae5

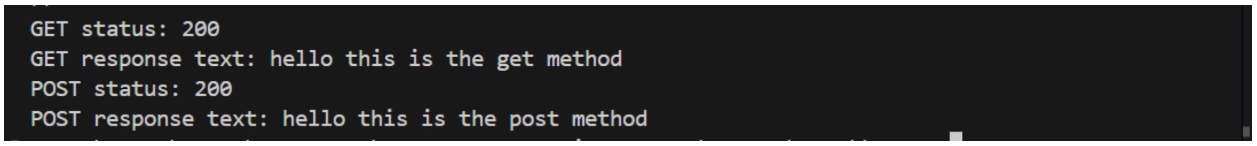
5. Why is HTTP still widely used even though MQTT and CoAP are lighter for IoT?

> The principal advantage of HTTP for use in IoT is its familiarity to developers, many of whom have implemented web solutions of one kind or another. A consequence of this is the availability of client libraries and servers.

Resource https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/#heading-the-advantages-and-disadvantages-of-http-in-io-t

Part 2 – Practical Projects (Python)

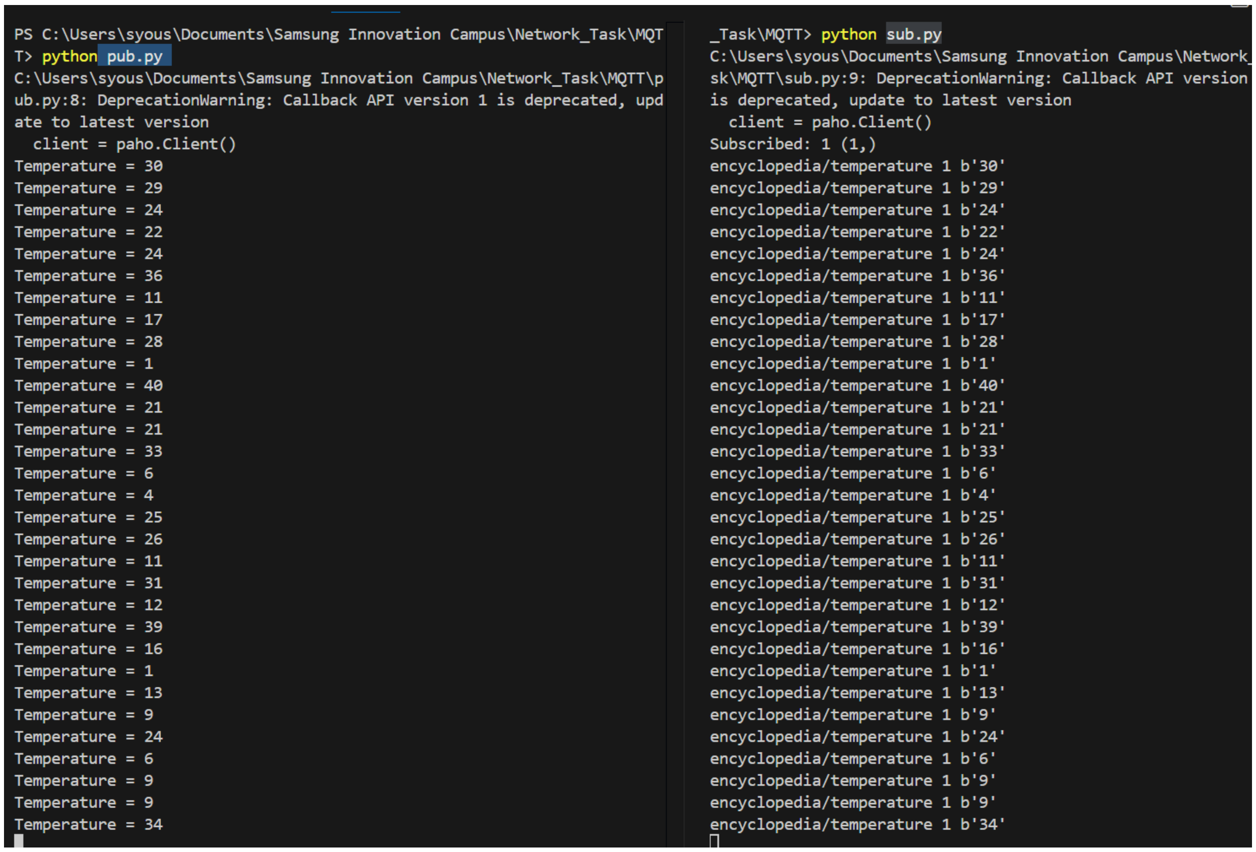
A. HTTP



```
GET status: 200
GET response text: hello this is the get method
POST status: 200
POST response text: hello this is the post method
```

Resources used to help me out
https://realpython.com/python-requests/#inspect-the-response
https://www.geeksforgeeks.org/python/flask-http-methods-handle-get-post-requests/

B. MQTT



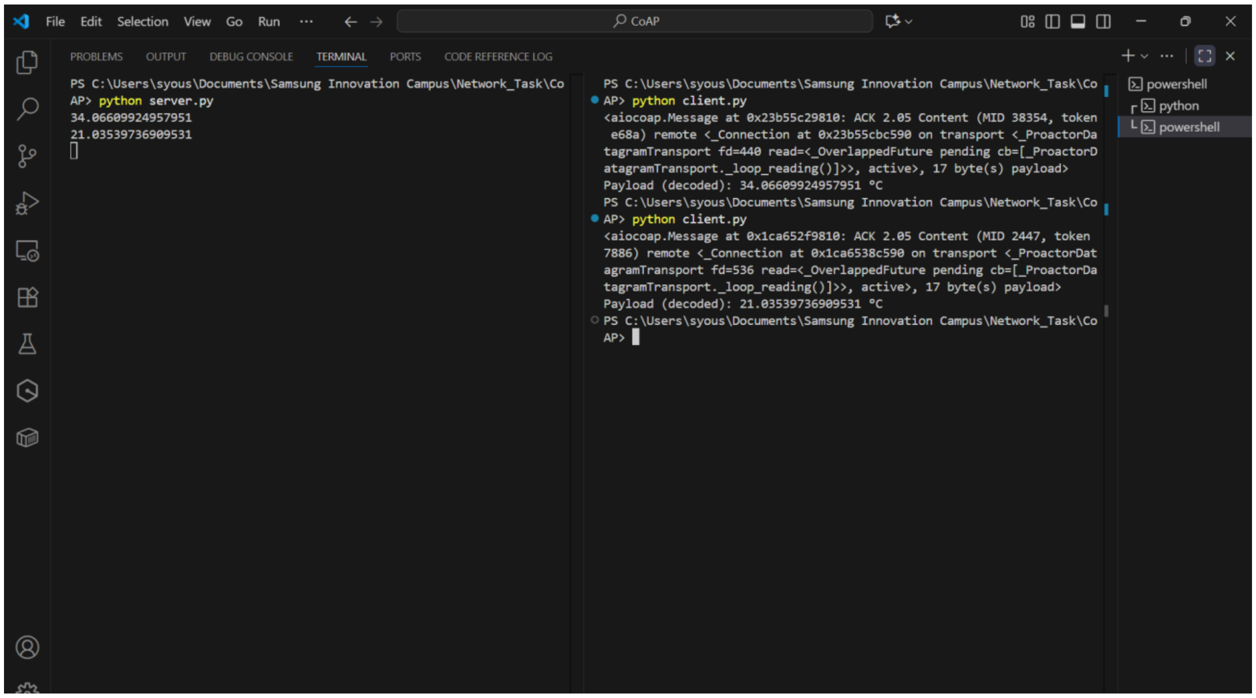
```
PS C:\Users\syous\Documents\Samsung Innovation Campus\Network_Task\MQT
T> python pub.py
C:\Users\syous\Documents\Samsung Innovation Campus\Network_Task\MQTT\p
ub.py:8: DeprecationWarning: Callback API version 1 is deprecated, upd
ate to latest version
  client = paho.Client()
Temperature = 30
Temperature = 29
Temperature = 24
Temperature = 22
Temperature = 24
Temperature = 36
Temperature = 11
Temperature = 17
Temperature = 28
Temperature = 1
Temperature = 40
Temperature = 21
Temperature = 21
Temperature = 33
Temperature = 6
Temperature = 4
Temperature = 25
Temperature = 26
Temperature = 11
Temperature = 31
Temperature = 12
Temperature = 39
Temperature = 16
Temperature = 1
Temperature = 13
Temperature = 9
Temperature = 24
Temperature = 6
Temperature = 9
Temperature = 9
Temperature = 34
```

```
_Task\MQTT> python sub.py
C:\Users\syous\Documents\Samsung Innovation Campus\Network_
sk\MQTT\sub.py:9: DeprecationWarning: Callback API version
is deprecated, update to latest version
  client = paho.Client()
Subscribed: 1 (1,)
encyclopedia/temperature 1 b'30'
encyclopedia/temperature 1 b'29'
encyclopedia/temperature 1 b'24'
encyclopedia/temperature 1 b'22'
encyclopedia/temperature 1 b'24'
encyclopedia/temperature 1 b'36'
encyclopedia/temperature 1 b'11'
encyclopedia/temperature 1 b'17'
encyclopedia/temperature 1 b'28'
encyclopedia/temperature 1 b'1'
encyclopedia/temperature 1 b'40'
encyclopedia/temperature 1 b'21'
encyclopedia/temperature 1 b'21'
encyclopedia/temperature 1 b'33'
encyclopedia/temperature 1 b'6'
encyclopedia/temperature 1 b'4'
encyclopedia/temperature 1 b'25'
encyclopedia/temperature 1 b'26'
encyclopedia/temperature 1 b'11'
encyclopedia/temperature 1 b'31'
encyclopedia/temperature 1 b'12'
encyclopedia/temperature 1 b'39'
encyclopedia/temperature 1 b'16'
encyclopedia/temperature 1 b'1'
encyclopedia/temperature 1 b'13'
encyclopedia/temperature 1 b'9'
encyclopedia/temperature 1 b'24'
encyclopedia/temperature 1 b'6'
encyclopedia/temperature 1 b'9'
encyclopedia/temperature 1 b'9'
encyclopedia/temperature 1 b'34'
```

Resource that helped me out :

https://www.hivemq.com/blog/mqtt-client-library-paho-python/

C.



```
PS C:\Users\syous\Documents\Samsung Innovation Campus\Network_Task\Co
AP> python server.py
34.06609924957951
21.03539736909531
```

```
PS C:\Users\syous\Documents\Samsung Innovation Campus\Network_Task\Co
AP> python client.py
<aiocoap.Message at 0x23b55c29810: ACK 2.05 Content (MID 38354, token
 e68a) remote <_Connection at 0x23b55cbc590 on transport <_ProactorDa
tagramTransport fd=440 read=<_OverlappedFuture pending cb=[_ProactorD
atagramTransport._loop_reading()]>>, active>, 17 byte(s) payload>
Payload (decoded): 34.06609924957951 °C
PS C:\Users\syous\Documents\Samsung Innovation Campus\Network_Task\Co
AP> python client.py
<aiocoap.Message at 0x1ca652f9810: ACK 2.05 Content (MID 2447, token
7886) remote <_Connection at 0x1ca6538c590 on transport <_ProactorDat
agramTransport fd=536 read=<_OverlappedFuture pending cb=[_ProactorDa
tagramTransport._loop_reading()]>>, active>, 17 byte(s) payload>
Payload (decoded): 21.03539736909531 °C
PS C:\Users\syous\Documents\Samsung Innovation Campus\Network_Task\Co
AP>
```

Resources that helped me out :
https://akpolatcem.medium.com/building-a-smart-thermostat-with-coap-and-lightweight-ai-413254a866af

https://aiocoap.readthedocs.io/en/latest/examples.html