COMP3310 Assignment 3

Analysis Report

Vesper Lin
u6828533

May 31, 2021

1 Response

1.1 Handshake of QoS levels 0, 1, 2

QoS 0

QoS 0 delivers message for at most once. Client sends a PUBLISH packet containing data of message to Broker. Then, regardless of the outcome, Broker drops the published packet, till now one message is sent. In such case, message duplication is not used, and message order is not necessary to be correct. Shown as Figure 1.

Figure 1: QoS 0 [1]

QoS 1

QoS 1 delivers message for at least once. To ensure that message has reached client, a reply mechanism is deployed.

- Client sends a Publish packet with data of message to Broker. Broker stores this Publish packet locally.
- After receiving a PUBLISH packet, Broker sends a PUBACK packet to Client. The PUBACK packet has no payload with it. There is a Packet Identifier in the Variable Header that matches the Packet Identifier in the PUBLISH packet it receives.
- After Client receives PUBACK packet, it finds the locally saved PUBLISH packet according to the Packet Identifier in the PUBACK packet and discard it. One time of message transferring is completed.
- If Client has not received a PUBACK for a Publish packet in some time, it sets the Publish packet's DUP flag to 1, which represents a re-published packet, and re-sends the Publish packet. The process will be repeated until the PUBACK packet is received, then the above step will be performed.

Shown as Figure 2.

Figure 2: QoS 1 [1]

QoS 2

QoS 2 delivers message for exactly once.

- Client sends a PUBLISH packet in QoS 2, with a Packet Identifier as P. Client saves the PUBLISH packet locally.
- After receiving the PUBLISH packet, Broker saves the Packet Identifier P of the PUBLISH packet locally, and replies to Client with a PUBREC Packet containing a Variable Header of which the Packet Identifier is P. Payload is none.
- When Client receives PUBREC, it can safely discard the PUBLISH packet with the initial Packet Identifier of P, then it will save the PUBREC packet and reply a PUBREL packet to Broker. The Packet Identifier in the Variable Header of PUBREL packet is P, and there is no payload. If Client does not receive a PUBREC within some time, it will set the PUBLISH packet's DUP flag to 1 and re-sends the PUBLISH packet.

- When Broker receives a PUBREL packet, it can drop the Packet Identifier P of the saved PUBLISH packet and reply to Client with a PUBCOMP packet, with Packet Identifier of P in the Variable Header. Payload is none.
- When Client receives a PUBCOMP packet, it regards the packet transfer process to be completed, and it drops the corresponding PUBREC packet. If Client doesn't receive a PUBCOMP packet within some time, it will re-send the PUBREL packet to Broker.

Shown as Figure 3.

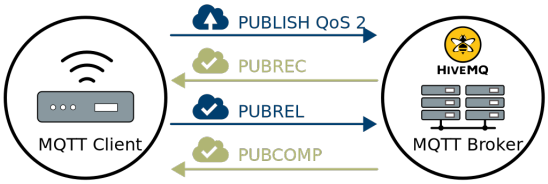

Figure 3: QoS 2 [1]

Wireshark captures

To illustrate the handshake of different QoS levels, I wireshark the handshake for each of them when running my Publisher to publish messages to localhost at QoS 0, 1, 2, and capture the screenshots to display Message Type information. Shown as Figure 4-10.

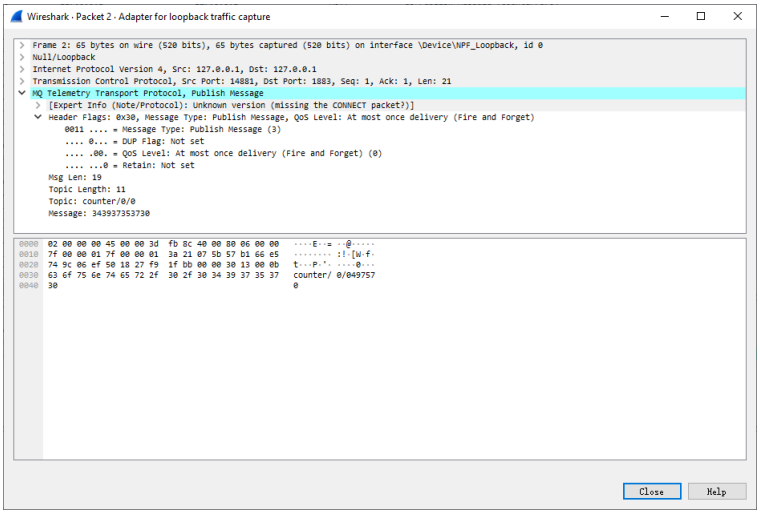

Figure 4: wireshark: PUBLISH QoS 0

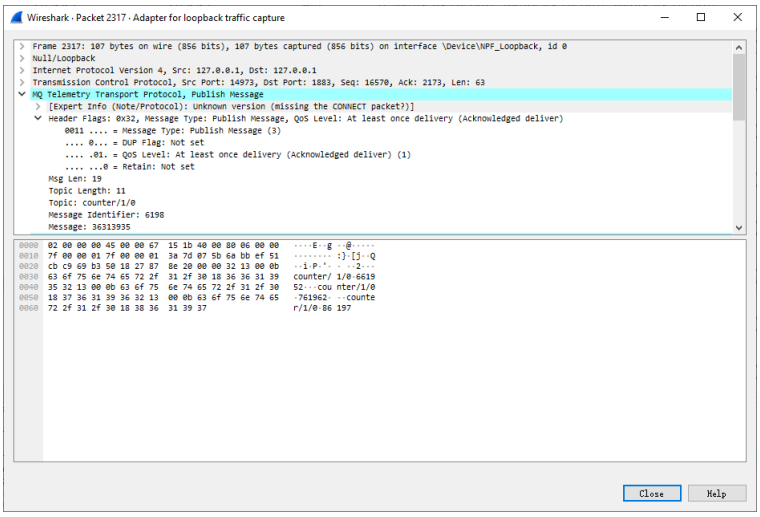

Figure 5: wireshark: PUBLISH QoS 1

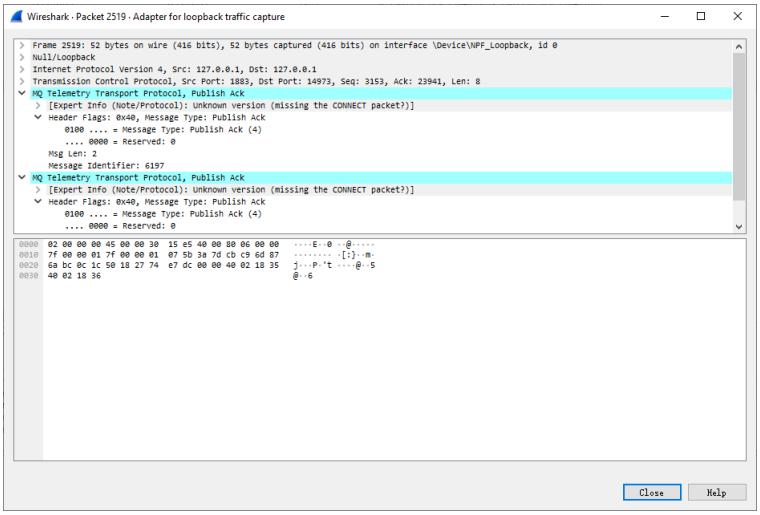

Figure 6: wireshark: PUBACK QoS 1

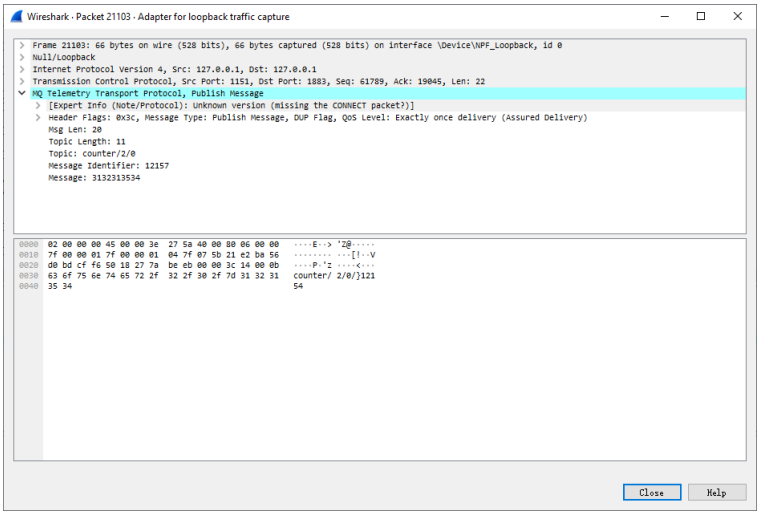

Figure 7: wireshark: PUBLISH QoS 2

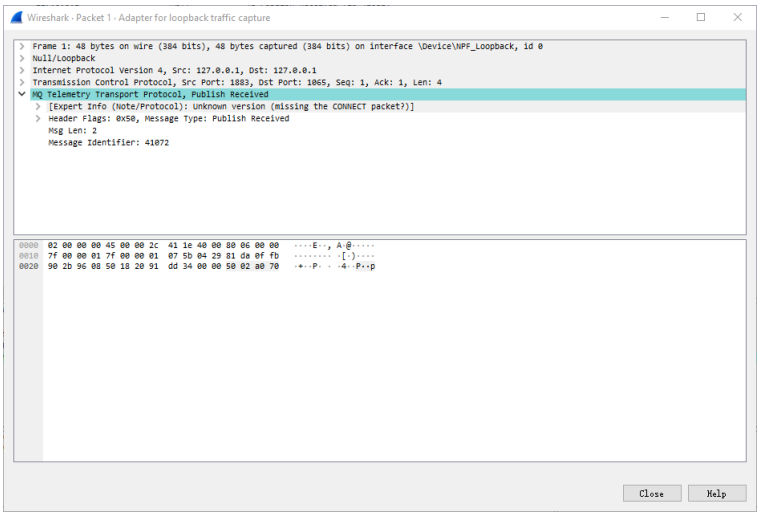

Figure 8: wireshark: PUBREC QoS 2

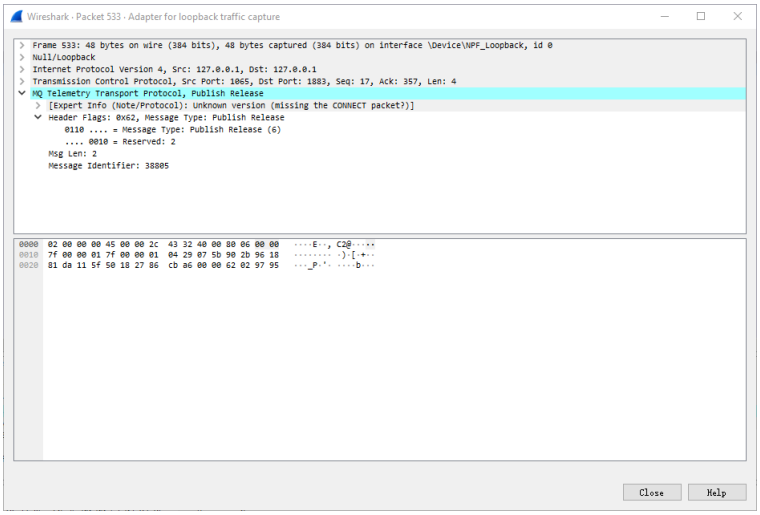

Figure 9: wireshark: PUBREL QoS 2

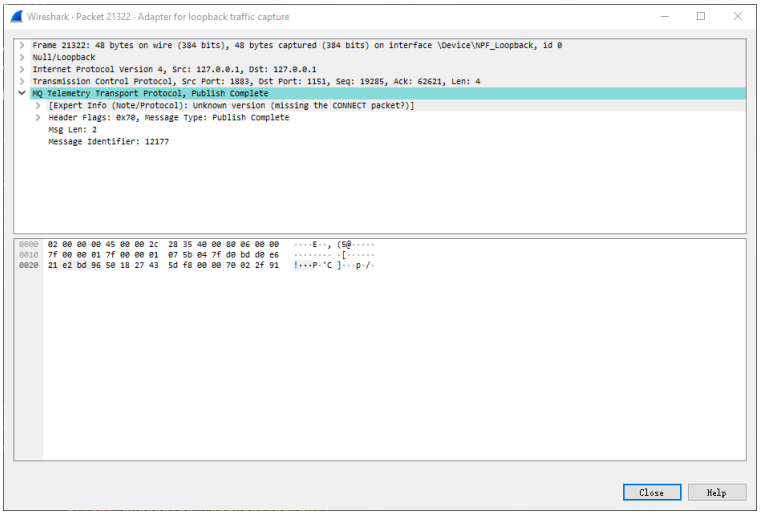

Figure 10: wireshark: PUBCOMP QoS 2

1.2 Choices of QoS

QoS 0

QoS 0 is a good choice when:

- There is a stable network environment connecting Client and Broker.
- Message loss is acceptable.
- Offline messages are not required.
- Resources are limited.
- Speed is required for most.

IoT sensor example:

Pressure sensors which need to sense pressure change immediately.

QoS 1

QoS 1 is a good choice when:

- Client needs all the messages to be received.
- Duplicate messages are unacceptable.
- Budget is not enough for choosing QoS 2.
- Speed is still regarded.

IoT sensor example:

Proximity sensors which are used to detect availability of parking.

QoS 2

QoS 2 is a good choice when:

- Client must receive all the messages.
- Application will not work well with duplicate messages.
- Budget is adequate for QoS 2.
- Safety is highly regarded.

IoT sensor example:

Chemical sensors which are used in laboratory for high precision tracking of chemical reaction. The use of QoS 2 will guarantee the safety of the laboratory environment.

2 Response

2.1 Measurement procedure

In my design of code, measuring the combinations of 3 QoS levels and 6 delays will take place at the same time. First, my partner, Carry Zhang, starts his MQTT Publisher publishing to his cloud broker on topic 'counter/0/0' by default. Then, I start my Analyser which subscribes to my partner's Publisher, and modify his Publisher's behavior to publish message to all the combinations of topics, each of which will last for a 2-minute period. In the meantime, my Analyser keeps collecting data for these 18 parts and calculating the 5 types of metrics including: the overall average rate of received messages across the period, the rate of message loss, the rate of out-of-order messages, and the mean and median inter-message-gap. Meanwhile, I use MQTT Explorer to record the $SYS topics of my partner's broker, looking for anything that seems relevant with the rate of metrics (e.g. loss rate). Traceroutes between Analyser-Broker and Publisher-Broker are recorded.

2.2 Calculation of statistics

i. average rate (messages/s)

This is calculated by the total number of packets received in a collecting period, divided by the duration of the period. In my measurement, the period for measurement is 120 seconds.

ii. loss rate (%)

This is calculated by the total number of packets received divided by the expected number of packets in a period. In my measurement, the total number of packets is the length of the list of collection in the period, and the expected number of packets can be calculated by the 'peak to peak' value from the packet IDs.

iii. out-of-order rate (%)

In theory, a current packet's ID is larger than its previous packet's ID. The out of order rate is calculated by the total number of packets which are contrary to this theory divided by the total number of packets received.

iv. mean and median inter-message-gap (ms)

Inter message gap is the time gap between a packet to its previous packet. The mean and median inter message gaps are obtained from the list of collection of each packet's timestamp.

2.3 Table of measurement results

Table 1: Measurement Statistics

	average rate	loss rate	out-of-order rate	mean inter-message-gap	median inter-message-gap
qos0 | delay0	743.8	92.5	0.0	0.4	0.0
qos0 | delay10	744.1	93.0	0.0	0.4	0.0
qos0 | delay20	743.3	93.1	0.0	0.4	0.0
qos0 | delay50	740.4	93.0	0.0	0.4	0.0
qos0 | delay100	744.7	92.9	0.0	0.4	0.0
qos0 | delay500	8.2	92.6	0.0	120.2	0.0
qos1 | delay0	100.3	39.6	25.8	7.8	0.0
qos1 | delay10	32.0	91.5	12.7	7.4	0.0
qos1 | delay20	32.1	99.9	31.4	16.3	0.0
qos1 | delay50	16.0	2.1	0.0	61.7	0.0
qos1 | delay100	9.1	1.8	0.0	108.6	1.1
qos1 | delay500	2.0	3.2	0.0	497.4	509.9
qos2 | delay0	0.8	99.1	0.0	16.3	0.0
qos2 | delay10	13.9	100.0	15.6	38.4	0.0
qos2 | delay20	24.6	2.0	0.0	39.8	0.0
qos2 | delay50	23.2	1.8	0.0	42.4	0.0
qos2 | delay100	24.3	1.7	0.0	40.4	0.0
qos2 | delay500	7.7	5.6	0.0	127.2	0.0

2.4 Data analysis

Table 1 shows the statistics of my measurement results. I will analysis each of the 5 types of metrics below.

i. average rate

Figure 11 reveals the relationship between average rate and different qos-delay combinations. We can see that no matter what delay is, QoS=0 means the highest rate of message received. Because QoS 0 ensures a best effort delivery [1]. When QoS=2, the average rate is always the lowest because it has the most complicated steps for checking received messages, which results in a slow speed in delivery. When the QoS levels are the same, the average rates tend to reduce with delay increases.

What I expect to see is QoS 0 will have a much higher average rate among other QoS levels, because QoS 0 is one-way handshake and it will keep publishing at high speed without hesitation. The expectations are matched with actual data.

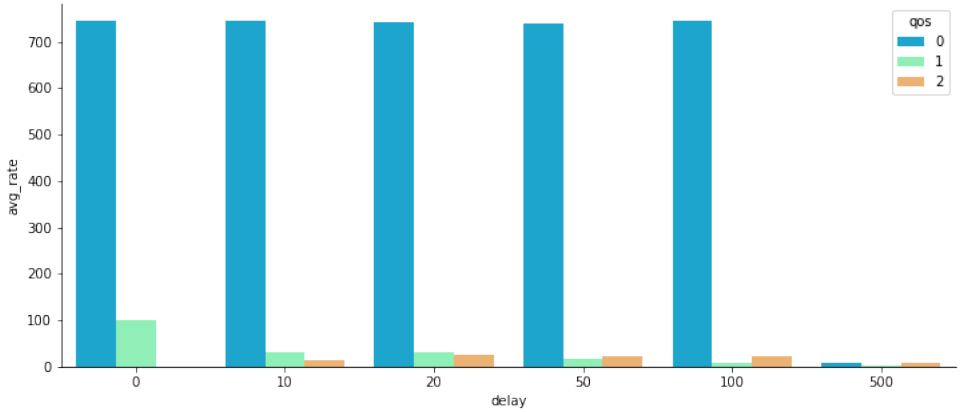

Figure 11: average rate

ii. loss rate

Figure 12 shows the loss rate among different qos-delay combinations. We observe that when QoS=0, the loss rate remains high no matter what delay is. It shows QoS 0 has no guarantee of delivery, because messages will not be acknowledged, stored or re-submitted [1]. When delay is over 20, QoS 2 obtains the lowest rate of loss. It is because QoS 2 ensures that each message is received only once by the receiver [1], thus it has high quality messages.

What I expect to see is QoS 0 will have a much higher loss rate among other QoS levels, because QoS 0 publishes tremendous messages rapidly to broker, and broker has to drop lots of messages to ensure subsequent normal operation. The expectations are matched with actual data.

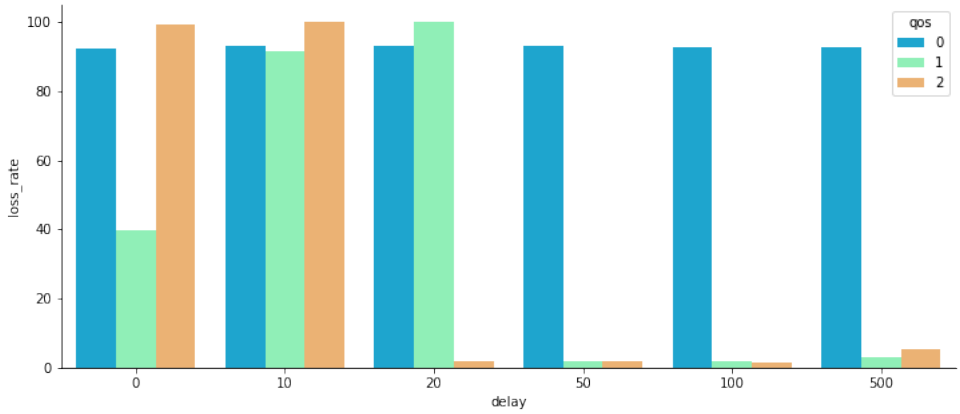

Figure 12: loss rate

iii. out-of-order rate

Figure 13 shows the out of order rate among different qos-delay combinations. It is interesting that the rate is almost none for QoS 0, but is high for QoS 1 when the delay is low. When the delay exceeds 50, no out of order packets are found.

What I expect to see is QoS 2 will have little out of order rate, because QoS 2 performs four-way handshake to ensure the there are no duplicate messages. The expectations are matched with actual data.

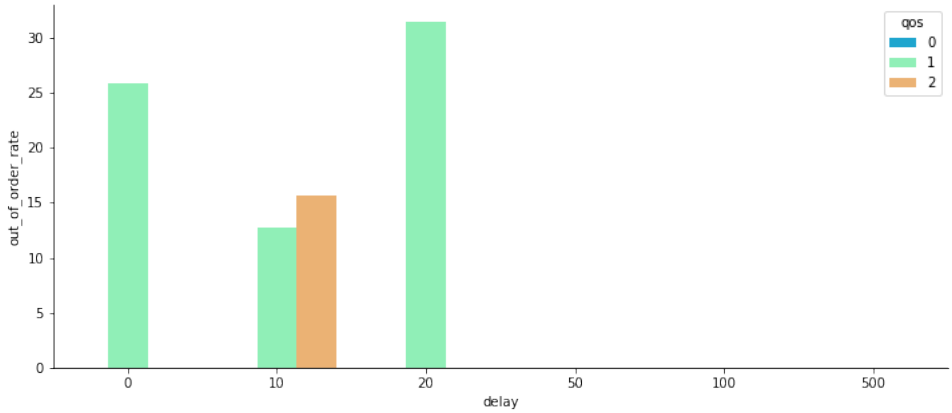

Figure 13: out-of-order rate

iv. mean inter-message-gap

Figure 14 shows the mean time of inter-message-gap among different qos-delay combinations. We observe that the inter message gap will increase when the delay increases for all QoS levels, which is reasonable for the gaps of time. The number is unreasonably high when QoS=1 and delay=500. I guess it is a network traffic issue.

What I expect to see is inter-message-gap increases with the increase of delay, this expectation is matched. However, the extremely high value when QoS=1 and delay=500 is out of expectation.

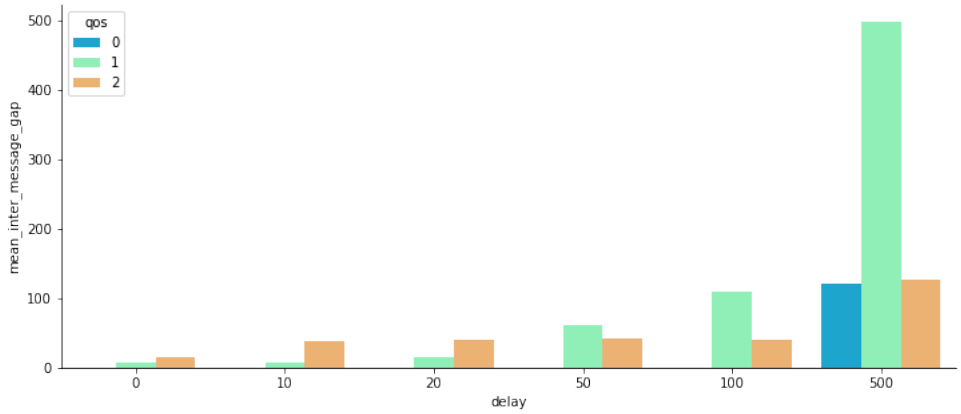

Figure 14: mean inter-message-gap

v. inter-message-gap

Figure 15 shows the median time of inter-message-gap among different qos-delay combinations. Likewise, the only circumstance that median inter-message-gap is not zero is when QoS=1 and delay=500, which I have discussed above. From the comparison of mean value and median value, we can assume that the inter-message-gap values are mostly 0 for each situation.

What I expect to see is median value of inter-message-gap will be extremely low because most of the time gaps should be close to 0, this expectation is matched. However, the extremely high median value when QoS=1 and delay=500 is out of expectation.

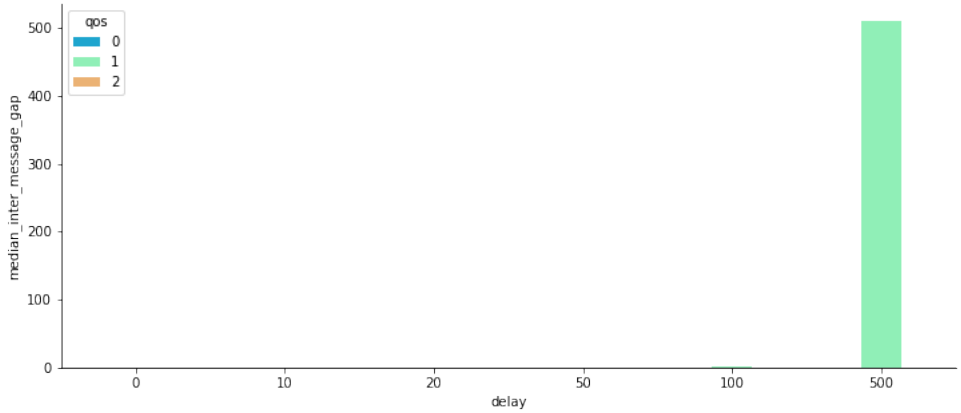

Figure 15: median inter-message-gap

2.5 $SYS

Figure 16 is a screenshot that shows lots of information about my partner's broker when running my experiments. I observe that the total number of dropped messages is 568879, the total number of received messages is 12944952. I observer that there are 2 clients connecting to the broker, which is me and my partner. From the "packets", I can observe information about the handshakes. I think the loss rate is highly correlate with metrics under 'messages', because the number of dropped messages, received messages are pretty clear from the $SYS topic. However, I can't find some specific sub-topics like 'load' under $SYS, because the broker is built on cloud server and the topic-allocation can be different from a local broker. Sadly I can't find out any clue about the unreasonable inter-message-gap value when QoS=1 and delay=500 from the screenshot below. I guess it is just an issue of network fluctuation.

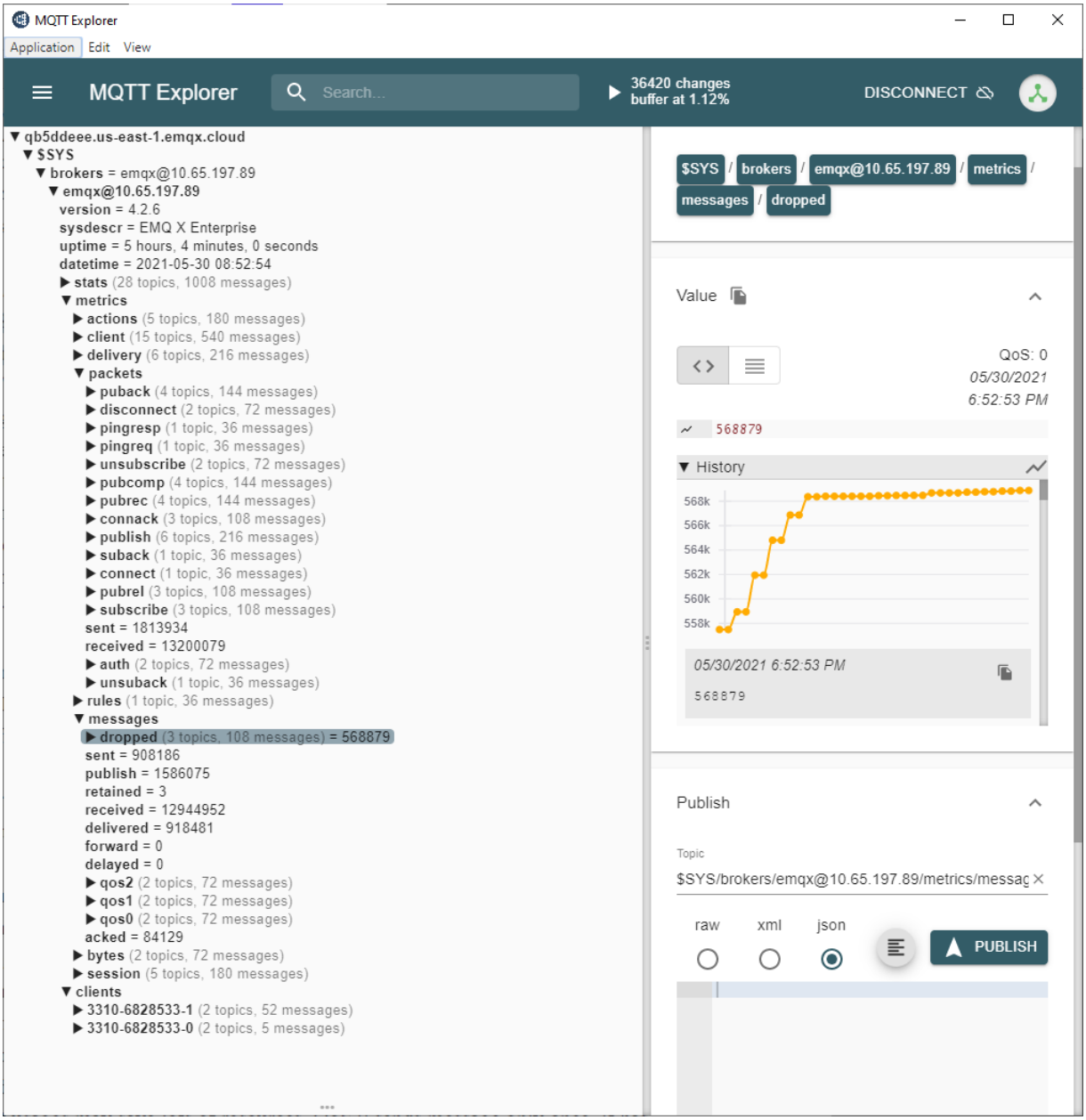

Figure 16: $SYS

2.6 Traceroutes

Unfortunately, my partner didn't provide his traceroute of the measurement. As alternative, I record a traceroute for my Publisher on ANU Linux Virtual Desktop Infrastructure to access remotely the Linux machines in the CECS labs, for simulation of partner's Publisher connecting to the cloud broker in the CECS labs. Shown as Figure 17.

As for the my Analyser connecting to the cloud broker, I record a traceroute for my Analyser on my laptop in a student accommodation. Shown as Figure 18.

```
traceroute to qb5ddeee.us-east-1.emqx.cloud (3.215.79.164), 64 hops max
 1   10.36.35.1  0.128ms  0.040ms  0.029ms
 2   192.168.10.1  0.239ms  0.113ms  0.051ms
 3   10.36.30.1  1.530ms  0.434ms  *
 4   150.203.201.34  2.337ms  0.718ms  0.358ms
 5   150.203.201.6  2.121ms  0.646ms  0.386ms
 6   138.44.161.2  1.714ms  0.440ms  0.592ms
 7   113.197.15.10  6.915ms  4.806ms  4.366ms
 8   113.197.15.157  5.673ms  4.581ms  4.360ms
 9   202.158.194.162  159.810ms  158.748ms  158.625ms
10   129.250.192.249  160.408ms  158.941ms  159.643ms
11   129.250.3.162  160.503ms  159.910ms  *
12   129.250.4.155  202.298ms  201.553ms  206.677ms
13   129.250.5.24  206.967ms  205.768ms  205.532ms
14   129.250.201.10  203.217ms  209.322ms  201.560ms
15   54.239.105.127  190.300ms  *  189.127ms
16   176.32.125.233  193.696ms  191.131ms  191.169ms
17   *  *  *
18   54.239.46.8  226.515ms  224.641ms  225.475ms
19   150.222.250.24  222.915ms  221.194ms  220.689ms
```

Figure 17: Publisher's traceroute

```
Tracing route to qb5ddeee.us-east-1.emqx.cloud [3.215.79.164]
over a maximum of 30 hops:

  1     2 ms    13 ms     1 ms  10.20.122.2
  2     2 ms     2 ms     3 ms  vlan-2111-palo.anu.edu.au [150.203.201.49]
  3    14 ms     3 ms     3 ms  vlan-2100-core.anu.edu.au [150.203.201.34]
  4     2 ms     2 ms     4 ms  huxbdr-te3-5.anu.edu.au [150.203.201.6]
  5     9 ms     3 ms     4 ms  138.44.161.2
  6     7 ms     6 ms     6 ms  et-0-3-0.pe1.rsby.nsw.aarnet.net.au [113.197.15.10]
  7     7 ms     7 ms     7 ms  113.197.15.157
  8   161 ms   162 ms   182 ms  xe-0-2-4.bdr1.a.sjc.aarnet.net.au [202.158.194.162]
  9   160 ms   161 ms   161 ms  xe-0-0-54-0.a01.snjsca04.us.bb.gin.ntt.net [129.250.192.249]
 10   161 ms   161 ms   176 ms  ae-8.r25.snjsca04.us.bb.gin.ntt.net [129.250.3.162]
 11   212 ms   213 ms   207 ms  ae-5.r25.dllstx09.us.bb.gin.ntt.net [129.250.4.155]
 12   209 ms   208 ms   208 ms  ae-5.r11.dllstx09.us.bb.gin.ntt.net [129.250.5.24]
 13   203 ms   204 ms   203 ms  ae-0.amazon.dllstx09.us.bb.gin.ntt.net [129.250.201.10]
 14   202 ms   193 ms   193 ms  54.239.105.109
 15   193 ms   193 ms   194 ms  15.230.48.44
```

Figure 18: Analyser's traceroute

From the traceroutes, I am pleased to see how the packets travelling from my laptop in Canberra, to AARNet Server in Sydney, and eventually reaching the AWS Server in the United States.

The traceroutes, which are consisted of so many global transfer stations, remind me of the challenges regarding the performance issues. Packet loss is unavoidable through such a long journey on the global network, even a minor network fluctuation on the way can cause a heavy loss of messages, which is quite similar to the butterfly effect.

3 Response

3.1 Performance challenges

End-to-end network message transferring can be quite limited by hardware performance. For instance, CPU is important for conducting message processing and computing in the message transferring process, having a low-performance CPU will result in inaccurate data. Such situation also applies for memory, since memory resource is critical for a speedy network transition. When the packet is tremendous in size for a Publisher, bandwidth issue will occur. In terms of network performance, network features such as bandwidth can greatly affect the procedure of message transferring. Also, the excessive memory usage causes a waste of resources. Network issues such as network latency or unstable network environment will heavily affect end-to-end network performances as well, since all the network operations depend on it, including IoT. For instance, a network outage rippling across Tesla's network of services will cause troubles for Tesla owners to get into their cars [2].

Apart from hardware challenges, there can also be software challenges. For instance, disadvantages of Python. Python is slow, compared to C or Java. Besides, Python is high in memory consumption, thus it is not recommended for memory intensive tasks [3].

3.2 QoS levels vs. challenges

In sum, QoS 0 is the best solution for dealing with hardware performance issues with limited resources provided, because it has the highest speed of message transferring. Although QoS 2 is safe, it requires high amount of usages in terms of CPU and memory, which can be a waste of network resources. Besides, QoS 2's four-way handshake makes the total processing time longer. I can't imagine how slow the network will be if all the services are using QoS 2. What's worth, QoS 2 is a high cost service to dispatch. Thus, QoS 2 is definitely not helpful for dealing with performance challenges.

3.3 Compare with my measurement

As I stated in data analysis of my measurement, if we look at the trend, the relationship of between values are close to the theory and common sense, for instance, QoS 0 obtains the highest rate of receiving messages while QoS 2 is the lowest. However, if we look at specific data, some of them might be unreasonably high or low, which is inconsistent with expectation. For instance, the inter-message-gap data are too high when QoS=1 and delay=500. What's more, the loss rate are generally too high for all most of the situations. It is because when I am located in Canberra, my partner is located in Beijing, and the cloud MQTT broker provided by AWS is located in US East (N. Virginia), the network latency can not be ignored and the packets are easily lost in such a long way of transfer. Besides, my laptop is not powerful in terms of configuration. Take a Python method that I use in my code for example, `time.sleep(0.01)` usually suspends the Analyser's execution for 0.005 seconds more than expected on my laptop.

Bibliography

[1] ”Quality of Service 0,1,2 - MQTT Essentials: Part 6”, *HiveMQ*, 2021. [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. [Accessed: 30- May- 2021].

[2] F. Siddiqui, ”Tesla built a network of connected cars. What happens when it goes down?”, *The Washington Post*, 2021. [Online]. Available: https://www.washingtonpost.com/technology/2020/09/23/tesla-network-outage/. [Accessed: 30- May- 2021].

[3] ”Disadvantages of Python - GeeksforGeeks”, *GeeksforGeeks*, 2021. [Online]. Available: https://www.geeksforgeeks.org/disadvantages-of-python/. [Accessed: 30- May- 2021].