1.

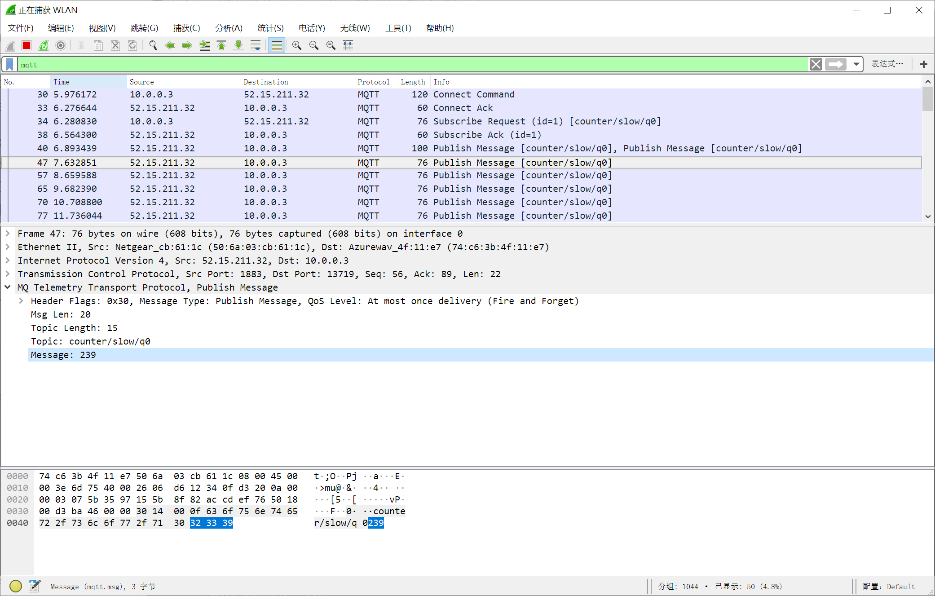
In the MQTT protocol, the QoS has three different flags which 0 stands for at most send once, 1 stands for at least sent once and 2 stands for sent once only. All three of QoS level has initial handshake which are

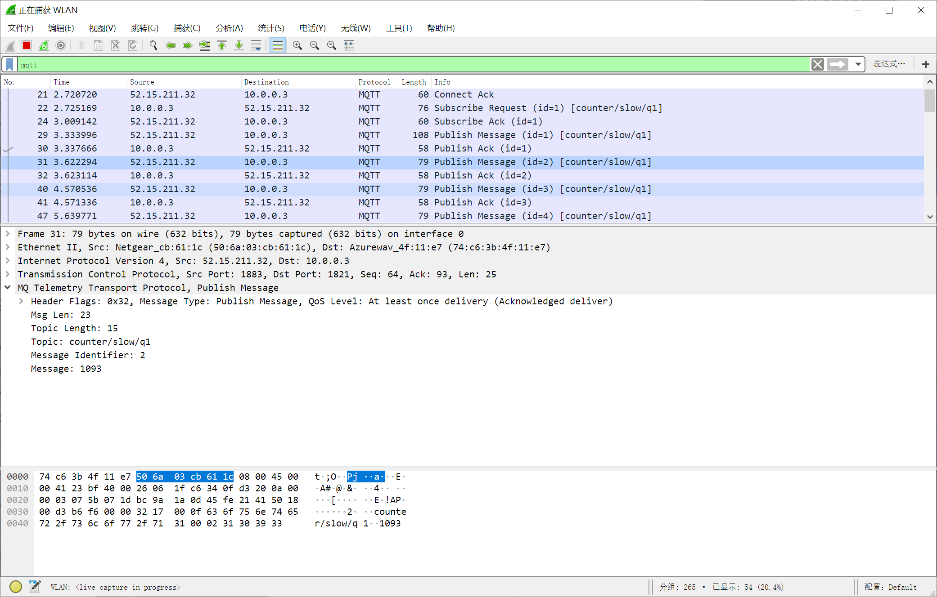
- Sent the connect command from client,

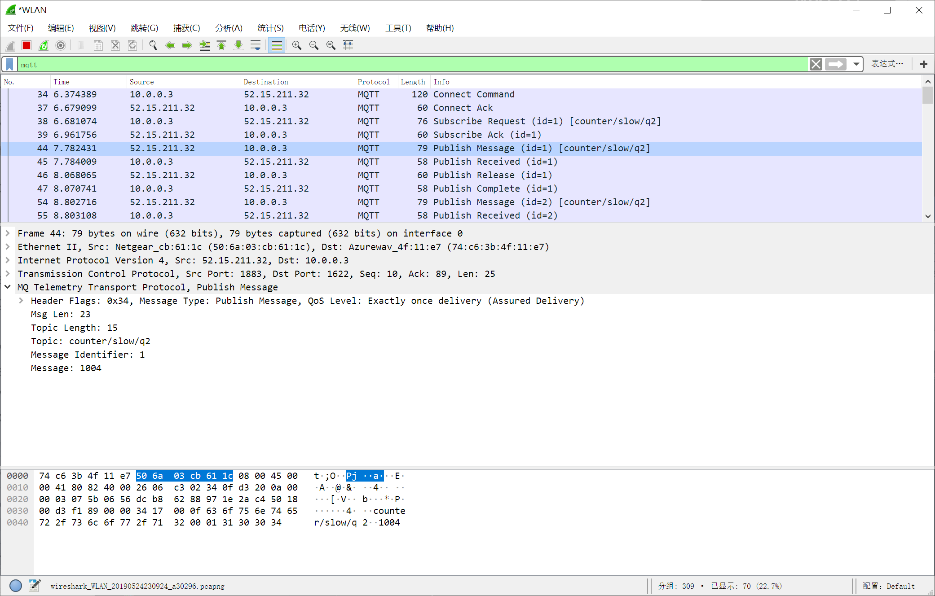
- Server sent the acknowledgement back to client,

- The client sent the subscribe request,

- Subscribe acknowledgement send from Server.

However, each QoS level has different message handshakes. When the QoS level is 0 each message at most sends once. There is no extra handshake, the server will send the message only. Thus, the server will not get any acknowledgement from clients. In this case, some message could be lost if the connection between the server and clients is not stable. Nowadays, this level could be used for the environment sensor data transportation, lost some packages is not critical in this case because the newer data will be sent very soon.

When the QoS level is 1, each message will be sent at least once. In this case, the client will send an extra acknowledgement back to sever to ensure the client has received that message successfully. If the server did not receive that acknowledgement on time, the server will send that message again. Compare with level 0 and level 2, this level is more reliable then level 0 and faster than level 2. However, if the client communication jammed to cause the receiving acknowledgement not sent out on time the server will send the same message as well. Thus, that may cause duplicate packages. This level could be used for the fire alarm system. Because for the system both response speed and reliability are important. Furthermore, in that case, duplicate package is not affect anything. Thus, the level 1 is most suitable for that system.

When the QoS level is 2, which send one package exactly once. In this case, there are 4 steps between the server and the client for each package. Firstly, the server will send the message to clients. Secondly, the client will send the receiving acknowledgement to the server. Thirdly, after the server receives that acknowledgement, that server will send to the client again about that publish is released. Finally, the client will send back to the server about the publish is complete. Compare with level 0 and 1, this level is the most complicate QoS level. It requires 4 communications for each message. Due to that, this level also cost more time than level 0 and 1. However, level 2 is the most reliable QoS level. It will guarantee the message arrived successfully and without duplicate in most time. Due to the reliability, the level 2 could use for paying system. Usually, the duplicate and lost message might cause an incorrect amount to pay. But the level 2 will not generate the lost and duplicate message in most of time.

2.

a. For fast counter channels, the standard deviation impacting the lost rate. The standard deviation is the square root of the arithmetic mean of the square of the mean difference, expressed as σ. The standard deviation is the arithmetic square root of the variance. Standard deviation can reflect the degree of dispersion of a data set. In my statistics, fast counter with the QoS level 0 does not lose any message also has a low standard deviation about 23 which means the time between each message is close to the average. QoS level 1 losing about 88% message in my testing but also with a huge standard deviation which is 103.14. Furthermore, the QoS level 2 lost rate is higher than QoS level 1, around 93.98%. That level also has a higher standard deviation like level 1, around 207.12, which near twice of level 1. The average gap is impacting the receive rate. In my testing, the QoS level 0 receive 272 messages per second, the average gap is 3.69 milliseconds. The QoS level 1 should receive 60729 messages but only receiver 7000 messages and the receive rate is 23.65 messages per second. The average gap between each message for level1 is 42.43 milliseconds. For level 2, the receive rate is 11.66 message per second and the average gap is 86.1 milliseconds. Compare with level 1 and level 2, the receive rate of level 1 is faster twice than level 2 and the average gap is lower twice than level 2. Furthermore, level 2 has more 2 steps communication for each message than level 1. Thus, the receive rate of level 1 should be twice of level 2 and the average gap also should lower twice than level 2 as fact.

b. Topic “clients/active” correlate with duplicate rate. Topics “publish/messages/dropped”, “heap/current” and “heap/maximum” are correlate with lose rate. The mis-ordered rates usually caused by client side.

c. According to the $SYS topic, the topic “clients/active” is potentially impacting the duplicate rate on QoS level 1. The level 1 is requiring the client sent an acknowledgement back to server otherwise the server will send the same message again. If the server did not receive that acknowledgement on time, the duplicate will appear. Thus, if the number of active clients is too high to handle on the server. Then all connecting clients will cause the internet jammed for each other, that will cause the server unable to receive the clients’ acknowledgement back, which correlate with the duplicate rate.

Moreover, the topic “publish/messages/dropped” also showed how many messages the server had dropped. That correlated with the lose rate caused by server side. The “heap/maximum” and “heap/current” topics also correlate with the lose rate. The heap is the memory of the server if the server run out of memory it also will drop messages.

However, I did not see any topics correlate with the mis-ordered rate. That because the mis-ordered happens on the client side. It usually caused by the connection between the server and the client is unstable or the client’s computation resource is insufficient. The mis-order usually on QoS level 2 and it may not happen on level 0. Because the level 0 will release the message instantly. Thus, if the client did not receive that will only lose it. Otherwise, assume message A and message B. On the level 2, the server needs client response to server firstly and reply to the client again to finish one message transport. Thus, if A is sending firstly and B sending secondly. But the network latency appears to lead the server to receive the acknowledgement about B firstly and then receive A secondly. In this case, message B is finished firstly on the client but should finish secondly, that is the mis-ordered.

On the client side, the performance also affects the result of message-gaps and variation for the fast counter. I had tested the fast counter with QoS level 2 on two different computers. According to the previous result, the QoS level 2 for the fast counter is the most sensitively affected by the client side. To ensure my testing result not effect ed by other factors, I used the same code and same network environment and did multiple times. According to my test, the higher performance computer got a lower gap and lower variation. It means the more powerful computer able to process the same message faster and more stable than the lower powerful computer. That indicates the client’s performance correlates with the process of the message in extreme condition. In this test, I use one computer with 18 cores 64GB memories and another one with 4 cores 16GB memories.

3.

For this assignment, I used Java to program it and use an extra library which is the org.eclipse.paho.client.mqttv3. The LAN is 5GHz wireless, WAN is the nbn100.

In my code, I use the HashSet to check the duplicate rate. Because the HashSet does not the same data twice. I use one ArrayList to storage all messages I received and put every message into a HashSet. Then I compare the size of two ADT to determinate the duplicate rate. For the out of order, I compare the current message with 10 previous messages before that message. Due to the counter is continuous increased. Thus, out of order will be recorded if anyone of previous messages greater than the current message.

On the one hand, I test the lost rate, duplicate rate and out of order rate both 0 and the received rate is 0.98 message per second on three QoS level on the slow counter channel. That shows the network between the client and the server is stable also the client and the server is powerful enough to handle those messages. Because the loss rate is 0% on level 0 which send no more than once. If the connection is unstable or client and server not powerful, there are lost package on level 0. The level 1 and level 2 will not lose message as I mentioned in Q1. However, level 1 might contain duplicate message if the connection is unstable. If the server does not receive the acknowledgement from the client on time, it will send the same message to the client again. Moreover, the server counter is continuing to increase. Thus, the message out of order issue should not cause by the server and connection. On level 2 might be a message out of order, because the level 2 need time to make 4 steps for each message. Thus, if the client is not fast enough, the message might be out of order.

On the other hand, in the fast counter channel, I got no lost on QoS level 0, but over 80% lost on level 1 and level 2. As I mentioned in the Q1 level 1 and 2 should not lose any package. However, in the situation which the server sending a lot of messages fast. There is a latency between the broker and the client. Thus, there might be many messages pending for QoS level 1 and 2, before the server receives the acknowledgement from the client about clear to send the next message. Moreover, the CPU usage and memory are not infinity big on both the client machine and server. Thus, in that situation, the server or client will drop messages due to there not fast enough to handle all message in that speed or insufficient memory. That dropping packages will show as lost packages in the sequence of MQTT message. There is no message out of order because the server dropped all message which unable to handle.

4.

In the scenario with millions of sensors and thousands of subscribers. The minimum length of each MQTT message is 3 bytes include 2 bytes fixed header and 1-byte message payload plus variable header (IBM, 2010). Assume each sensor only send ten minimum messages to the broker per second, the broker has to handle 30MB different message from the different source per second. Assume that the broker has 200 cores CPU. The CPU has to process each message within 0.0067 milliseconds. That is difficult to handle by the broker. The CPU performance is critical in this scenario. Furthermore, the CPU also need a process to send messages to clients and each client might requesting different messages. Thus, the CPU also required able to process at least 1000 tasks concurrently.

Moreover, if this scenario using QoS 1 and 2, the broker also needs to store all message from each sensor. That could be a challenge because the server will release the message only after it receives an acknowledgement from clients for both QoS level 1 and 2. Furthermore, clients receiving message is not synchronized. Thus, the broker must storage all messages a while. That could consume a lot of memory. In this scenario, assume the number of clients is 1000 and the server will only storage 10 minutes messages. That could consume 18 gigabytes memories to storage messages only.

For this case, the network also is a challenge. Especially for the part of the broker. In this scenario, all sensors will generate 30MB data per second as I calculated above. Thus, the broker needs at least 30MB/s bandwidth for accepting messages from publishers which faster 3 times than NBN tire 100. The broker also needs another broadband at least 30MB/s bandwidth for subscribers. Moreover, if all subscribers want every message from all sensors, that could be 30000MB/s bandwidth requested.

In this scenario, QoS level 1 and level 2 might not be useful. Both level 1 and level 2 requiring extra feedback from clients, that not only consume extra broker’s CPU resource but also consume a lot of memory. Those two levels might cause the lost package like the MQTT broker with the fast counter channel in this assignment. As I mentioned before, the CPU resource and memory size on the broker server are not unlimited. Thus, any extra communication between clients and broker might be jammed by network latency or speed. In this case, the broker handling very huge amount of data and any latency time might cause extra memory space usage. Run out of memory will lead the system dropping package. Thus, clients will lose a lot of packages while they are subscribing with QoS level 1 and 2.

However, in this case, level 0 might be helpful. There is not any extra communication between the broker and clients. Thus, that will reduce a lot of pressure for the broker. Moreover, sensors will send data continuously. Thus, lost one or two of the packages does not matter, the updated data will be sent soon. The QoS level 1 and level 2 are losing packages over 80% during my testing. Compare with that, level 0 losing some packages is more acceptable.

According to the result from this assignment. Compare with QoS level 1 and level 2, level 0 is losing packages less than level 1 and 2 in the scenario which the broker handling lots of data at a fast speed. Apply that result in this scenario, the QoS level 0 is better than level 1 and level 2 not only on losing rate but also consume less system resource for broker server.

# Bibliography

IBM. (2010, 8 23). *MQTT V3.1 Protocol Specification.* Retrieved from MQTT V3.1 Protocol Specification: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html