**Assignment 3**

-Yixi Rao u6826541

**Q1**

**Q1.1 Qos 0**

First, this is the subscription information for the Qos 0. I let my client subscribe to the “counter/0/#” and set the Qos to be 0.

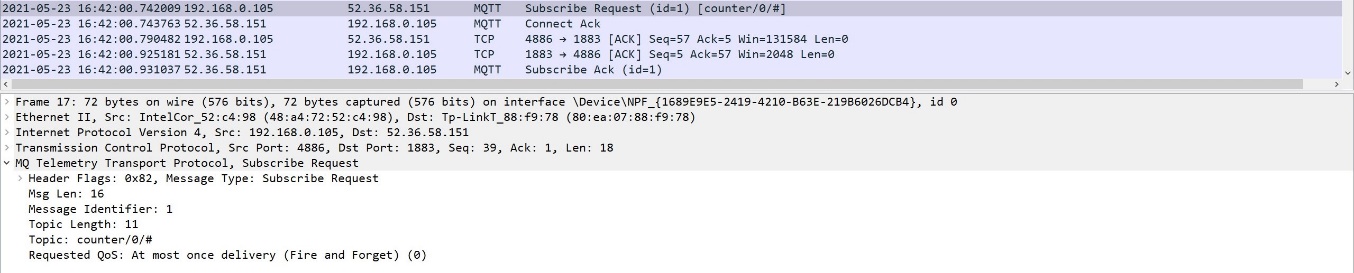


figure 1: Qos 0 subscription

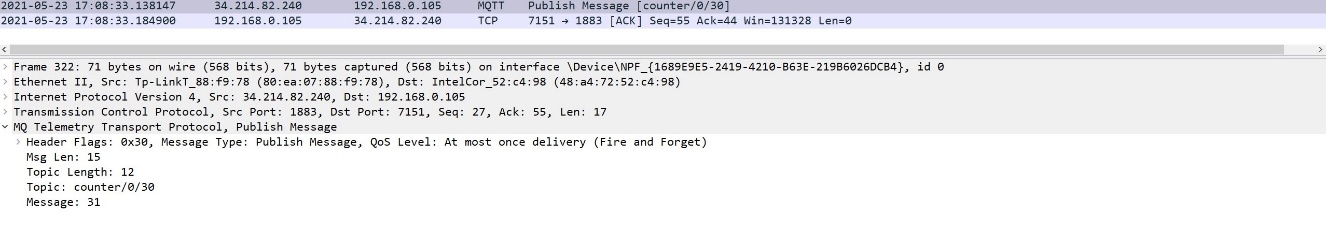


figure 2 Qos 0 handshake

For the handshake of Qos 0, if the publisher sends a message, the subscriber can receive it at most once. This means that the publisher tries to send a message to the subscriber, and if the message fails, the receiver misses it. In the figure 2, the broker sends only one PUBLISH packet containing the message data to the subscriber, regardless of the outcome, broker will discard it.

For the following circumstances, the Qos 0 can be selected [1]:

* The subscriber is acceptable to lose some messages, such as if you have a sensor that publishes data at a very short period, so even if it loses some messages, it recovers very fast.
* The connection between the broker and the subscriber (or publisher) is very stable, the bandwidth is very good, e.g. you have some Leased lines, or the connection is built on LAN.
* Suitable for data that is not very important

**Q1.2 Qos 1**

First, this is the subscription information for the Qos 0. I let my client subscribe to the “counter/1/#” and set the Qos to be 1.

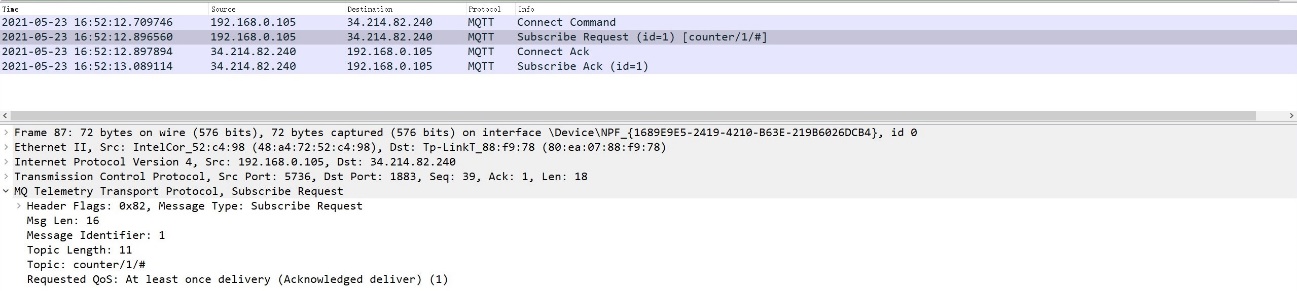


figure 3 Qos 1 subscription

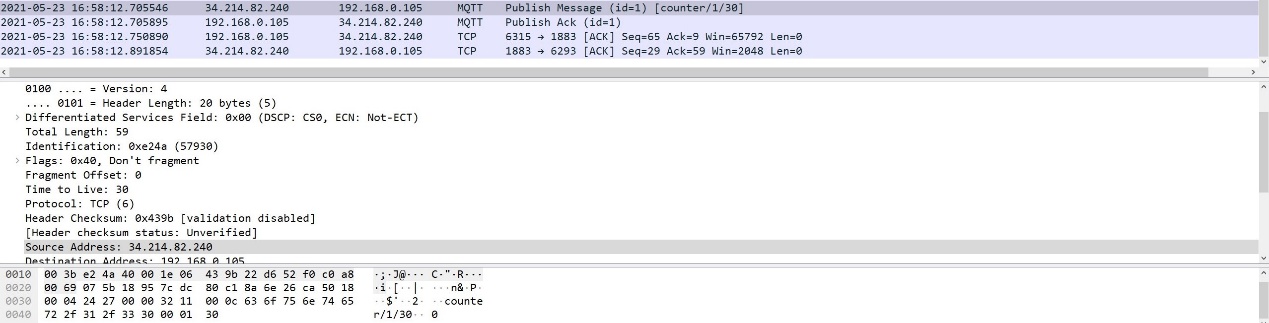


figure 4 Qos 1 handshake

For the Qos 1 handshake, Qos 1 means that the publisher (or broker) sends a message and the subscriber (or broker) can receive it at least once, which means that the sender sends a message to the receiver, and if it fails, it will try to send the message again until the receiver receives the message and return some acknowledgements to the sender, but because of the retransmission, the Receiver may receive duplicate messages. In the figure 4, the broker first sends the publish message (with id = 1) to the subscriber and does not delete it, then if the subscriber receives the message, it will send back the publish ACK containing an Identifier, afterwards the broker will receive that ACK and find the message with the same Identifier then delete it. If the broker does not receive any ACK message over a period of time, it will send the message again and identify it as duplicate.

For the following circumstances, the Qos 0 can be selected [1]:

* You need a high speed of sending the message or receiving the message, while you want to ensure the reliability of the message.
* It is not expensive or hard to handle the duplicate message in your application

**Q1.3 Qos 2**

First, this is the subscription information for the Qos 2. I let my client subscribe to the “counter/2/#” and set the Qos to be 2.

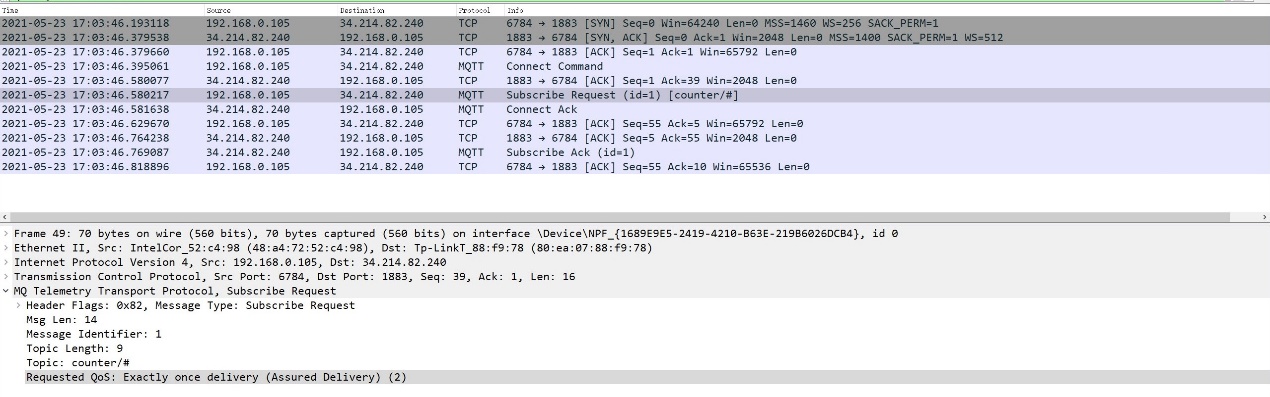


figure 5 Qos subscription

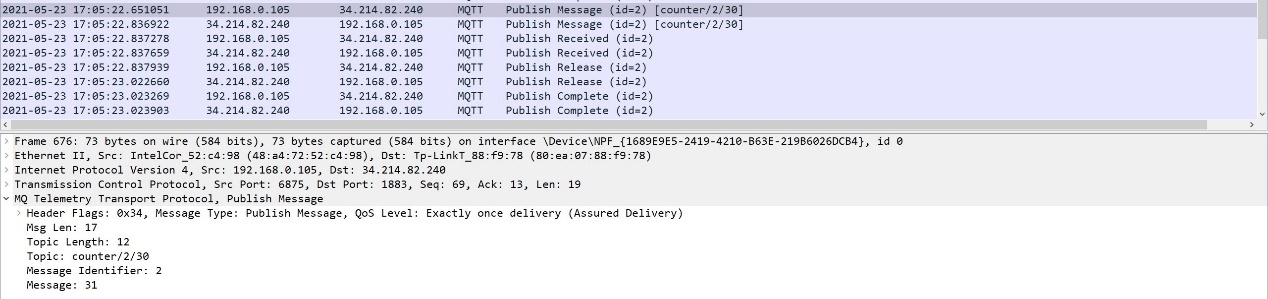


figure 6 Qos handshake

Qos 2 represents that sender sends a message to the receiver, and guarantee the receiver can receive it and only receive it once, that is, the sender tries to send a message to the receiver, and if it fails, it will try again until the Receiver receives the message. It also ensures that the Receiver will not receive duplicate messages due to retransmission of messages. In the figure 6, the broker first sends the publish message to the subscriber, and save this message locally, Second, the subscriber gets the message and save the identifier of the message locally then sends back the PUBREC with an identifier from that message, then broker can delete the publish message and save the PUBREC locally, and return a PUBREL, at last, the subscriber can delete the publish message and sends the PUBCOMP to broker, which will delete the PUBREC after broker receives it.

For the following circumstances, the Qos 2 can be selected [1]:

* Every message is very important and your application must receive all messages (The loss of a message can result in risk of life or property).
* Your application won't work with duplicate messages

**Q1.4 Summary**

Qos 0 is acceptable to lose some messages, so some sensors that sends the message frequently is more tolerable with Qos 0 such as a temperature sensor.

Qos 1 is fast and efficient, and also have reliability of message, so the highway speed sensor is applicable for this Qos because it has to record the speed of the vehicle and this data cannot be lost.

Qos 2 is suitable for some application which needs the message integrity and timeliness so Banking, firefighting, aviation or financial sensors will need it.

**Q2 Ⅰ**

The statistics I collect are the overall average rate of messages, and the rate of message loss, and the rate of out-of-order messages, the mean and median inter-message-gap for each delay and QoS level. The formula is shown below:

For the overall average rate of received messages, the expected result is all the Qos 1 level received messages rate greater than Qos 0 level and Qos 0 level will greater than Qos 2 level. This is because Qos 2 has the most overhead, so it will spend more time to receive the message, and Qos 0 will send the message very fast but it does not guarantee the subscriber to receive it, so even if has the fastest sending speed, the losing speed is also fast, and Qos 1 will ensure that the subscriber receives at least once, as a result it may receive more messages than it expected. For the different delays, it is intuitive that the short time interval will send more messages, therefore, the shorter the time interval, the faster the rate. Here is a summary of my expected result.

And this is the experiment result (reserve two decimal fractions):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Msg/s | 0.0ms | 10ms | 20ms | 50ms | 100ms | 500ms |
| Qos 0 | 88.71 | 64.07 | 32.06 | 16.02 | 9.18 | 1.97 |
| Qos 1 | 73.06 | 66.57 | 32.31 | 16.08 | 9.13 | 1.97 |
| Qos 2 | 41.25 | 38.57 | 30.69 | 15.75 | 9.12 | 1.94 |

table 1 received message rate

In the table 1, we can see that rate of Qos 1 is slightly greater than Qos 0 except for 0.0 delay. Another observation is that, as the delay increases, the difference between Qos 0 and Qos 1 will decreases, and for the 100ms and 500ms delay, three Qos levels received rate is similar, I think this is because it gives more time for the Qos 1 and Qos 2 to finish the handshake.

For the rate of message loss, the expected result is Qos 2 level loses the fewest messages, and Qos 1 level will lose fewer than Qos 0, the reason is Qos 2 level 4 handshake processes will ensure that the subscriber will not lose a message, and also Qos 1 level ensures the subscriber can receive at least one message, Qos 0 might loses many messages since it no longer cares about the message after the message is sent. Also, the percentage of loss will decrease as the interval time increases. Here is a summary.

This is the experiment result.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| % | 0.0ms | 10ms | 20ms | 50ms | 100ms | 500ms |
| Qos 0 | 0.00 | 0.36 | 0.36 | 0.20 | 0.08 | 0.02 |
| Qos 1 | 0.37 | 0.33 | 0.35 | 0.19 | 0.09 | 0.01 |
| Qos 2 | 0.98 | 0.61 | 0.38 | 0.21 | 0.087 | 0.03 |

Table 2 message loss

The experiment result does not correspond to the expected result, it seems that Qos 0 level have the lowest message loss percentage and Qos 2 loses the greatest number of messages. My analysis is there are several possible reasons, maybe the remote delay between the broker and the subscriber, or maybe there is an error in my formula since the all the percentages of loss is less than 1%.

For the rate of the out-of-order message, the expected result is Qos 0 should not have any out-of-order message because Qos 0 prefers the latest message rather than the message that has been sent, so the messages will be added into the list in an ascending order. the percentage for the Qos 2 should greater than Qos 1 because the more time it spends, the more out of order you're likely to get and as the interval time increases, this problem will be mitigated. Here is a summary.

This is the experiment result.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Msg/s | 0.0ms | 10ms | 20ms | 50ms | 100ms | 500ms |
| Qos 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Qos 1 | 0.03 | 0.03 | 0.00 | 0.00 | 0.00 | 0.00 |
| Qos 2 | 0.05 | 0.03 | 0.04 | 0.00 | 0.00 | 0.00 |

Table 3 out-of-order message

The experiment result completely corresponds to my expected result, and another observation is that the problem of out-of-order is eliminated if the time interval is longer than 50ms.

For the mean and median inter-message-gap, the expected result is that Qos 2 level will have longest inter-message-gap time because the time-consuming handshake processes, and the Qos 1 level has the second longest, the interval time of Qos 0 should be very short but due to the high delay, the interval time will be a litter bit longer. And as the delay increases, the gap time will also increase. Here is a summary.

This is the experiment result:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | 0.0ms | 10ms | 20ms | 50ms | 100ms | 500ms |
| Qos 0 | 0.017 | 0.026 | 0.041 | 0.072 | 0.124 | 0.518 |
| Qos 1 | 0.033 | 0.039 | 0.042 | 0.072 | 0.126 | 0.517 |
| Qos 2 | 0.039 | 0.061 | 0.072 | 0.073 | 0.125 | 0.528 |

Table 4 message gap mean value

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s | 0.0ms | 10ms | 20ms | 50ms | 100ms | 500ms |
| Qos 0 | 0.013 | 0.020 | 0.040 | 0.065 | 0.104 | 0.514 |
| Qos 1 | 0.024 | 0.033 | 0.039 | 0.067 | 0.113 | 0.500 |
| Qos 2 | 0.025 | 0.058 | 0.065 | 0.066 | 0.121 | 0.517 |

Table 5 message gap median value

The experiment result corresponds to my expected result and another observation is that the interval time of three levels are similar after 50ms delay.

**Q2 Ⅱ**

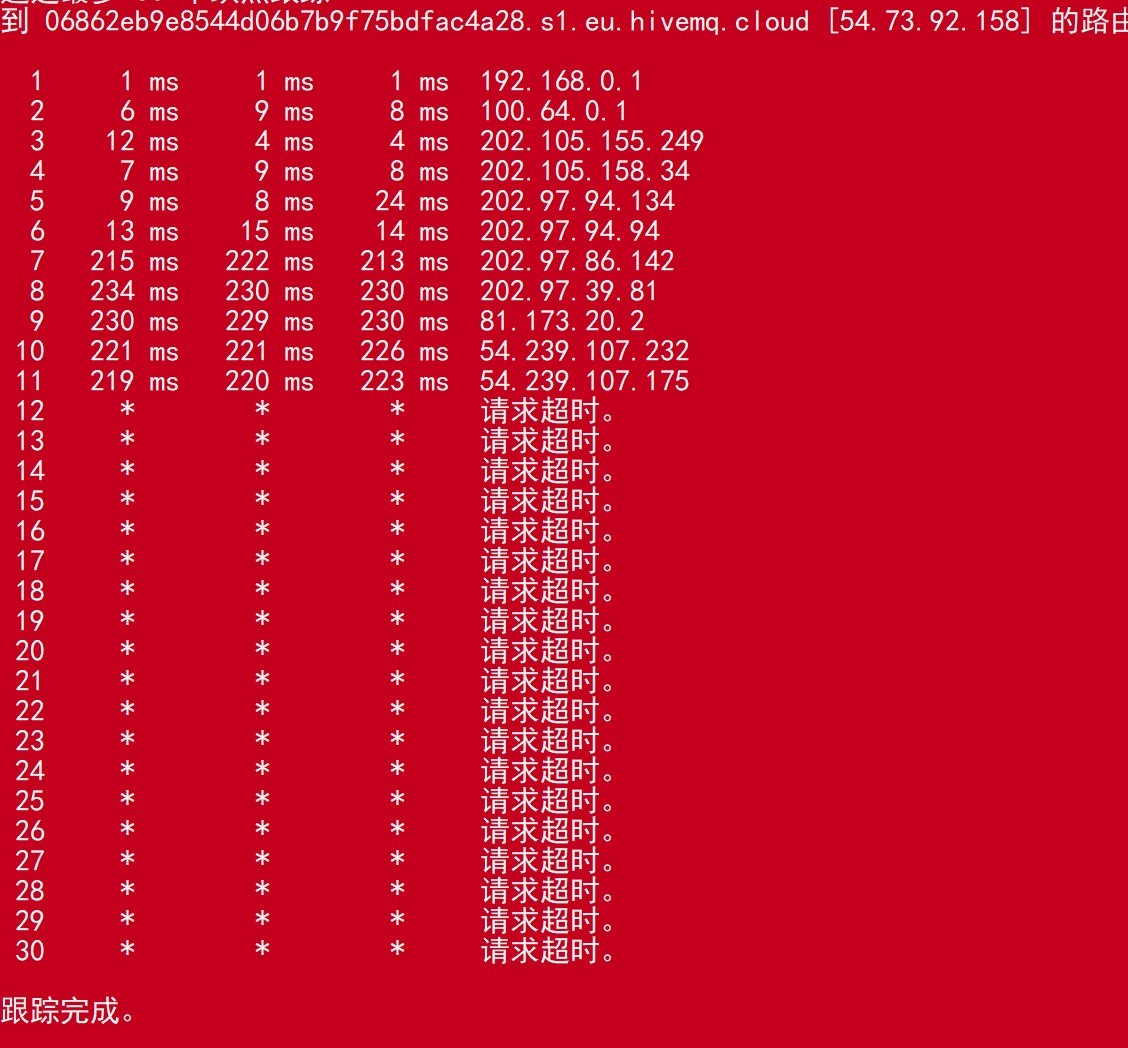
For the topic $SYS/broker/load/#, the “SYS/broker/load/messages/received/+” topic correlates with the rate of message loss because “SYS /broker/ load/ messages/ received/+” means the moving average of the number of all types of MQTT messages received by the broker over different time intervals, and when this value decreases the loss rate will increase.

For the topic “$SYS/broker/heap/#”, which has two sub-topic: “$SYS /broker /heap/ current size” returns the current heap memory of the broker and “$SYS/ broker/ heap/ maximum size” returns the largest amount of heap memory used by the broker, and this two topics correlate with the rate of message loss and rate of disorder message rate since the memory is limited and it will overflow in some situations, which causes the message loss.

For the “$SYS/broker/clients/connected” topic, which has a correlation with the rate of message loss, because when the number of subscribers grows up, the more traffic is created and messages loss might happen.

**Q2 traceroute**

It seems tracing the route to the HiveMQ broker has some issues, the traceroute between the Analyser and the broker is failed and also between the Publisher and the broker. It even cannot trace the route when I set the maximum hops to 100 and enable the ICMP Ipv4, I think this is the HiveMQ’s side problem because I can trace the EMQ X public broker’s route. Here is the one of the failures traceroute of HiveMQ.



**Q3**

**a)**

As in the broader end-to-end network environment, millions of sensors connected to a centralized broker, and also millions of subscribers subscribe to it. There is a performance challenge in the CPU and memory side because MQTT using the TCP, which is designed for some device with more memory and CPU power since it needs to establish many connections (like wake-up and handshake) before sending message. As a result, the centralized broker and the sensors still need some CPU power and memory, and the frequent connections due to the TCP will also influence the battery of sensors [3].

Centralized broker is also another challenge when the number of sensors or subscribers growing rapidly. Under normal circumstances, the capability of centralized broker is fixed, which means that the broker’s memory only can support a fixed number of devices, as in the broader end-to-end network environment, there are many devices connected to broker, so when the number of devices exceeds the threshold, the memory performance challenge happens. Centralized broker does not have a good scalability [3].

MQTT is a message protocol not the data protocol, so MQTT allows any type of message, and the message parsing is the job on the subscriber sides or the broker sides,

As the number of devices increase, more and more data types emerge some light-weight controller devices have to install some interpretive program to help it parse the data, so it is no longer light-weight and it is a challenge for memory and CPU [4].

For the network performance challenge, MQTT is not intelligent for routing or publishing message because MQTT does not check the payload of the message, which results in MQTT can not distinguish the duplicate message or previously sent message, this is very inefficient and cost lots of bandwidth and CPU power. In the broader end-to-end network environment, the traffic is heavy and busy, this drawback will aggravate the traffic and cause delay [4].

**b)**

For the performance challenge of CPU, memory and network, the Qos level can help to mitigate it. As long as we select the proper Qos level for different kinds of message, we can achieve better data rate and more message. So, it is important to distinguish the messages according to the objective of the message. However, in some areas, the three levels of Qos may not help.

As we mentioned above in Q1, Qos 0 level is applicable for some high-speed publisher, and losing some messages is acceptable for it. So, for the broader end-to-end network environment, we can let all the publishers and subscribers under the same sub-net using the Qos 0, on the other hand, some applications that is tolerable with message lost or can infer the missed messages like the temperature sensors or video application. It is fast and consumes very tiny amount of memory and CPU power. Qos 1 is a decent choice because it guarantees the message timeliness and reliability, it is suitable for some applications that messages had to be received and losing it will cause some problems, like some event-handlings or command receiver [5]. For the Qos 2, it has the most overhead, and consume many memories and CPU power, so in the broader end-to-end network environment, we should use the Qos 2 as less as possible, only using it in some transactional systems. Another approach to deal with the challenges is to replace some Qos 2 messages exchange to Qos 1 [5], this method can increase the bandwidth, reduce the memory used and raise performance, however, the negative effect is that the installation of elimination of duplicate messages has to be applied in the subscriber side, so it is a trade-off between footprint and bandwidth.

Meanwhile, some application cannot boost performance by using these three levels Qos, one example is the IIoT (Industrial IoT) [6]. In IIoT, Qos 0 will cause a fatal flaw due to the Qos 0’s property that is QoS 0 will favor old messages over new ones [6], for Qos 1, the problem will also emerge when the publisher sending rate is quicker than the subscriber receiving rate [6], for Qos 2, it is similar to Qos 1, but more severe [6]. So, none of these three Qos levels satisfy IIoT, the IIoT wants some different - guaranteed consistency [6], which means it does not care about the intermedia messages, it only cares about the current message, it is desire to receive the current status information, and really pay attention to the timestamp, if it misses a value that is already superseded by a new measurement, that is not generally a problem [6] so the publisher should guarantee the successful publishment the final “off” message [6].

**c)**

I think in the broader end-to-end network environment, the actual quantified differences between QoS levels will be bigger than differences I measured because there are millions of sensors and thousands of subscribers, and the sensors (or subscriber) are even very far away the broker, As the result, the delay of Qos 0 would grow as the distance and the number of client increases, it could be even worse for the Qos 1 and Qos 2 levels since the Round-Trip Time (RTT) of the handshake will aggravate the delay which will longer than the delay I measured. Theoretically speaking, if the Qos 2 level delay I measured is , then the delay in the broader end-to-end network environment would be where n is the difference between RTT that I measured and RTT in the broader end-to-end network environment. So, the delay will be terrible if the is so big.

**Reference:**

[1]<https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/>

[2] <https://en.wikipedia.org/wiki/MQTT>

[3]<https://www.oreilly.com/library/view/analytics-for-the/9781787120730/e86ff73d-7e8c-4eda-9890-0ceebbadcf78.xhtml>

[4] <https://skkynet.com/mqtt-answer-iiot/>

[5] <https://blog.usejournal.com/how-to-optimize-data-usage-over-mqtt-792abebd2cd1>

[6] <https://skkynet.com/qos-quality-service-right-iiot/>