COMP3310/6331 Assignment 3 – Testing MQTT

Introduction:

- This assignment is worth 15% of the final mark
- It is due by **Friday 5 June 23.59 AEST**
- Late submissions will not be accepted, except in special circumstances
- *Any extensions must be requested well before the due date, via the course convenor, with appropriate evidence.*

Assignment 3

MQTT is the most common open IoT protocol being deployed today. It provides a publisher/subscriber model, using a broker or server. It allows for an almost-arbitrary number of sources to publish information, at their own rate, and subscribers to receive information (and act on it). As such, it is designed to provide high-performance communication mechanisms, with minimal delays and excellent scaling performance. We'll use it to monitor the performance of a couple of systems: imagine a slow and steady stream of buses going through an interchange being counted, and another counting the total kilograms of coal rushing by on a conveyor belt. This assignment will look at the functionality and performance of the publishers, brokers, the network and your subscribers.

This is a coding, analysis and writing assignment. You may code in C, C++, Java or Python (hope that's enough for everyone), and yes, you may use MQTT and other helper libraries. The assessment will not rely solely on running on your code, but more on the longer-term data gathering and your analysis, but we will review the code and may briefly test it on the usual lab-type environments or similar. You will need to identify in your code any libraries you are using.

Submitting

You will be submitting your code and your analysis report, as well as reporting statistics to the broker.

1. Your code must be submitted to wattle as a zip file, packaging everything as needed, or pointing the markers at it, with a makefile for the statistics collector/publisher as appropriate.
2. Your analysis report (pdf) must be submitted via TurnItIn on the wattle site
3. Your statistics you will store on the mqtt broker, as described below

Each question has an indication of how much (rough maximum) you should write.

Outcomes

We're assessing your understanding of MQTT, your code's functionality in subscribing/publishing to a broker, dealing with high message rates, and your insight to interpret what you're seeing. You will be exploring MQTT functionality, the quality-of-service (QoS) levels, describing how they work, why you would use different levels, and how they perform in real-world deployments given various publishers.

Resources

We have set up an MQTT server (on the standard port) at comp3310.ddns.net for you to connect to as per the specifications below. There are also two (2) external publishers that count sequentially from 0 to some "large" number, before wrapping back to 0. One publishes a counter message at about 1 per second (say counting buses), and the other publishes at a much faster rate (say counting the kilograms of coal), for tackling the questions below. Both publish to the broker at all three QoS levels, with the following topics:

- counter/slow/q0, counter/slow/q1, counter/slow/q2
- counter/fast/q0, counter/fast/q1, counter/fast/q2

Note that the counters (i.e. each of slow/# and fast/#) increment at the same rate for direct comparisons. The QoS here references the QoS between the source and the broker, and is not the QoS between the broker and your subscribing client; that is handled by you.

Assignment questions

Your code needs to subscribe to the broker, authenticating with username ***students*** and password ***33106331*** (yes, it's a shared password, and the server is logging everything), and set your client-id to be ***3310-<your.uni.id>***. You can test it by subscribing to the $SYS topics, which describe the server, and help get you familiar with the information presented there.

1. Subscribe to the three slow channel counters (QoS 0,1,2) one at a time, with the matching QoS level on your client (you can use any mqtt client you like for this). Wireshark the handshake for one example of each of the differing QoS-level messages, capture screenshots and briefly explain in your report how each handshake works, and what it implies for message duplication and message order. Discuss briefly in your report in which circumstances would you choose each QoS level, and offer an example of a sensor that would use each of them. *[around 1 page]*

2. With your own code subscribe to each of the three slow and fast channel counters (QoS 0,1,2), and listen for at least 5 minutes. *Tips: only print individual messages to screen for debugging, otherwise it will slow your code down a lot. Measuring longer than 5 minutes is better, and doing it more than once over the assignment period is also better, for comparisons by time-of-day, time-till-due, etc. I suggest you only subscribe to one of the 6 channels at any one time, to avoid your client's performance becoming an issue, but doing all three at a given speed at once may be informative, as you can directly compare the exact same set of published messages via different QoS channels.*

    a. Collect statistics over the run, for each slow/fast and QoS level, and measure over the 5+ minute period:
        i. The average rate of messages you receive across the period *[messages/second]*.
        ii. The rate of message loss you see *[percentage]*.
            (i.e. how many messages should you have seen, minus how many you saw)
        iii. The rate of duplicated messages you see over any 10 seconds *[percentage]*
            (i.e. keep a window of 10 seconds of messages and see if any are repeated)
        iv. The rate of out-of-order messages you see *[percentage]*
            (i.e. how often do you get a smaller number after a larger number, ignoring counter wraps)
        v. For the slow counter, the mean inter-message-gap and gap-variation (standard deviation) you see *[milliseconds, milliseconds]*.
            Only measure for actually consecutive messages, ignore the gap if you lose any messages in between, but watch the wrap.
        vi. For the fast counter, also measure the same gap and gap-variation, and see if it correlates with the rate of messages received and/or lost.
            Again, only measure for actually consecutive messages, ignore the gap if you lose any messages in between, but watch the wrap.

    b. While measuring the above also subscribe to and record the $SYS topics, and identify what, if anything, on the broker do the loss/duplicate/misordered-rates correlate with.
        i. Look at measurements under e.g. 'load', as well as the 'heap' and 'active clients', anything that seems relevant. See https://mosquitto.org/man/mosquitto-8.html

c. In your report *[around 2-3 pages of text, plus any charts]*
   i. Summarise your measurements, and note when you took them. Explain what you expected to see, especially in relation to the different QoS levels, and whether your expectations were matched.
   ii. Describe what correlations of measured rates with $SYS topics you expect to see and why, and whether you do, or do not. Also indicate whether you saw any impacts (client-side cpu, memory, etc.) when measuring the fast counter

3. Consider the broader end-to-end (internet-wide maybe) network environment, in a situation with millions of sensors publishing regularly/irregularly to thousands of subscribers. Explain in your report *[around 1-2 pages]*
   a. what (cpu/memory/network) performance challenges there might be, from the source publishing its messages, all the way through the network to your subscribing client,
   b. how an irregular publisher has different issues to a very-frequent publisher,
   c. how the different QoS levels may help, or not, in dealing with the challenges, and
   d. how it compares (or not) with the actual quantified differences between QoS levels you measured as part of this assignment.

4. For your last run when all is working well, your code should publish your measurements under **studentreport/<your.Uni.ID>/** with the 'retain' flag set and QoS=2. Use the sub-topics structure as follows, replacing **<qos>** with 0,1 and 2 in each case, and report for each **<speed>** of **slow** and **fast**:

   a. **/language** what language your code is in and which mqtt libraries you used. (free text string)
   b. **/network** what kind of network(s) your client is connecting over (free text string, e.g. wifi, dsl, 4g, what speeds, etc.)
   c. **/location** from which city/town, state/province and country in the world you are running your client (free text string, to whatever level of detail you feel comfortable with)
   d. **/<speed>/<qos>/recv** rate of messages received (messages/second)
   e. **/<speed>/<qos>/loss** loss rate (%)
   f. **/<speed>/<qos>/dupe** duplicate rate (%)
   g. **/<speed>/<qos>/ooo** out-of-order rate (%)
   h. **/<speed>/<qos>/gap** inter-message-gap (milliseconds)
   i. **/<speed>/<qos>/gvar** inter-message-gap variation (milliseconds)

   For example: ***studentreport/u9876543/fast/2/recv 2000***

   You must check that your reports are being properly published and retained. You can overwrite the statistics if you need to repeat the exercise, but ideally only report once you are about to submit. Again, we will log publications to the server.

Assessment

Your assignment will be assessed on

- Your code clarity, efficiency, and documentation/comments (20%)
- your code subscribing and properly publishing to the server (10%), and
- your analysis report addressing the questions above (70%)