COMP3310/6331 Assignment 3 – Testing MQTT

Introduction:

- This assignment is worth 10% of the final mark
- It is due by **Thursday 30 May 17.00 AEST**
- Late submissions will not be accepted, except in special circumstances
    - 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, 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, say a slow and steady stream of buses going through an interchange, and another counting the total kilograms of coal rushing by on a conveyor belt.

This is a coding, analysis and writing assignment. You may code in C, Java or Python, and yes, you may use MQTT and other helper libraries. The assessment will not rely solely on running on your code, since it involves longer-term analysis, but we will review it and may briefly test it on the usual lab machines. You need to identify any libraries you are using.

You will be submitting to wattle your code and your analysis report, as well as reporting statistics to the broker. Your submission must be a zip file, packaging everything as needed, with a makefile for the statistics collector/publisher. Each question has an indication of how much (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.

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. One publishes a counter message at about 1 per second (counting buses), and the other publishes at a much faster rate (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 (each of slow and fast) increment at the same rate. The QoS here references the QoS between the source and the broker, and is not the QoS between the broker and your subscribing client, which 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 command-line client 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. *[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 five minutes. Tip: only print individual messages to screen for debugging, otherwise it will slow your code down a lot.
   a. Collect statistics over the run, for each slow/fast and QoS level, and measure:
      i. The rate of messages you receive *[messages/second]*.
      ii. The rate of message loss you see *[percentage]*.
      iii. The rate of duplicated messages you see *[percentage]*
      iv. The rate of out-of-order messages you see *[percentage]*
      v. For the slow counter, the mean inter-message-gap and gap-variation (standard deviation) you see *[milliseconds, milliseconds]*.
         1. Only measure for actually consecutive messages, ignore the gap if you lose any messages in between.
      vi. For the fast counter, also measure the same gap and gap-variation, and see if it impacts the rate of messages received and/or lost.
   b. While measuring the above also subscribe to and record the $SYS topics, and identify what, if anything, do the loss/duplicate/misordered-rates correlate with.
      i. Look at measurements (over some period) under e.g. 'load', as well as the 'heap' and 'active clients'. See https://mosquitto.org/man/mosquitto-8.html
   c. In your report 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) when measuring message-gaps/variation for the fast counter *[around 1 page]*

3. 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 of **slow** and **fast**:

a.	**/language**	what language is your code in and which mqtt libraries you used. (a string)
b.	**/network**	what kind of network you are connecting over
c.	**/slow|fast/<qos>/recv**	rate of messages received (messages/second)
d.	**/slow|fast/<qos>/loss**	loss rate (%)
e.	**/slow|fast/<qos>/dupe**	duplicate rate (%)
f.	**/slow|fast/<qos>/ooo**	out-of-order rate (%)
g.	**/slow|fast/<qos>/gap**	inter-message-gap (milliseconds)
h.	**/slow|fast/<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.

In your report summarise your measurements, reflecting what you've hopefully also published, and explain what your measurements tell you about the performance of the broker/server for handling this level of traffic. *[around 1 page]*

4. Consider the broader end-to-end network environment, in a situation with millions of sensors publishing regularly 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 the different QoS levels may help, or not, in dealing with the challenges, and
   c. how it compares (or not) with the actual quantified differences between QoS levels you measured as part of this assignment.

Assessment

Your assignment will be assessed on

- Your code clarity, style and documentation,
- your code effectively subscribing and properly publishing to the server, and
- your analysis report addressing the questions above.