COMP3310/6331 Assignment 3 – Testing MQTT

Introduction:

- This assignment is worth 10% of the final mark, with up to another 5% bonus points on offer (see BONUS POINT items below)
- It is due by **Tuesday 22 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, 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. Any libraries that are not on the lab machines need to be packaged with your code.

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. The report should be around 1200 words long total across all questions (a bit more if you tackle the bonus questions), and the use of charts/graphs/tables is encouraged.

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 3310exp.hopto.org 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 issues a counter message at about 1 per second (counting busses), useful for testing your code, and the other 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 this references the QoS between the source and the broker, and is not the QoS between the broker and your subscribing client, which is handled separately, by your code.

Assignment questions

Your code needs to subscribe to the broker, authenticating with username ***3310student*** and password ***comp3310*** (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. That becomes important later on in the analysis.

1. Subscribe to the three slow channel counters (QoS 0,1,2) one at a time, with the matching QoS level on your client. Wireshark the handshake for one example of each of the differing QoS-level messages (include 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.

2. *You should run your code for this next part of the assignment as often as you can during the assignment period, to get better statistics at different times of day and with different user loads.* Subscribe to each of the three fast channel counters (QoS 0,1,2), and listen for at least ten minutes each time – or longer if you have an appropriate host to run it on.
    a. Collect statistics for the duration of each analysis, for each QoS level, over 1-minute intervals and measure the following (to be reported under item #3 below):
        i. What rate of messages you receive
        ii. What rate of message loss you see
        iii. What rate of duplicated messages you see
        iv. [BONUS POINTS] What rate of out-of-order messages you see
    b. While running the above measurements, also subscribe to and record the $SYS topics, and identify what, if anything, do the loss/duplicate/misordered-rates correlate with.
        i. Focus on just the 1min samples under 'load', as well as the 'heap' and 'active clients', *[BONUS POINTS] or more.*
        ii. See https://mosquitto.org/man/mosquitto-8.html for explanations.
        iii. In your report describe what correlations with $SYS topics you expect to see and why, and whether you do, or do not.

3. For each run, your code should publish your current results at 10-minute intervals under **studentreport/<your.Uni.ID>/** with the 'retain' flag set and QoS=2. Use the sub-topics structure as follows:

    a. **/language** message = what language is your code written in
    b. **/timestamp** message = time of report in *unixtime* (seconds since 1 Jan 1970)

    Then use the timestamp, and your QoS setting, as an index in the topics below.

    c. **/<timestamp>/<qos>/recv** message = best 1-min rate of messages received
    d. **/<timestamp>/<qos>/loss** message = worst 1-minute loss rate (%)
    e. **/<timestamp>/<qos>/dupe** message = worst 1-minute duplicate rate (%)
    f. ***/<timestamp>/<qos>/ooo*** *message = worst 1-minute out-of-order rate (%)*

    So for example: ***studentreport/u9876543/1524062563/2/recv*** *=* ***61000***

    You must check that your reports are being properly published and retained.

4. *[BONUS POINTS] Towards the end of the assignment period when there are a 'good number' (100+) of reports from the class held on the server, subscribe to **studentreport/#** to download what others have been seeing (you can use any command-line client/script for this if you want), and compare results. In your analysis report discuss:*
    a. *How consistent is the loss/dupe/out-of-order rate across the class, at similar times?*
    b. *What correlations in performance do you see with the language used for the client?*

5. Consider the broader end-to-end network environment, in a situation with many thousands of sensors publishing regularly to some large number of subscribers. Explain in your report
    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 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 publishing to the server, and
- your analysis report addressing the questions above.