# COMP3310/ENGN3539/etc Assignment 3 – Testing MQTT

## Introduction:

- This assignment is worth 15% of the final mark
- It is due by **Tuesday 24 May 23.55 AEST (Canberra time)**
- Late submissions will not be accepted, except in special circumstances.
- *Any extensions should be requested well before the due date, 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-unbounded number of sources to publish information, each 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 some imaginary system: say counting the total kilograms of minerals rushing by on a conveyor belt, that you can control. This assignment will look at the functionality and performance of the publishers, brokers, the network and the subscribers.

This is a coding, analysis and writing assignment. You may code in any programming language (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 data gathering and your analysis. However, we will review the code and may briefly test it against our own broker running in the usual lab-type environments or similar. You will need to identify in your README/code any libraries you are using.

*Please note, **you will be working in pairs or small groups** for a key part of this assignment, to test each other's systems across the Internet. If you don't have somebody to work with, ask your tutors NOW to identify somebody in your tutorial group. You do not need to be in the same location at any time.*

## Submitting

You will be submitting two things: your code and your analysis report. Note that there will be two submission links on the course-site in wattle:

1. Your code must be submitted to wattle as a zip file, with instructions on how to compile/run the components as appropriate.
2. Your analysis report (pdf) must be submitted via TurnItIn on the wattle site, so ensure you quote and cite sources properly.

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

You will need to set up your own MQTT server (broker) for you to connect to as per the specifications below. The Mosquitto broker is perhaps the best choice to run on your own computer, it has binaries and/or packages for most operating systems, but if you find a better one you can use that. It must be at least MQTT v3.1.1 compliant (i.e. MQTT v5 is not necessary).

A key consideration is that you will need to allow your assignment partner to connect to it over the Internet for part of the assignment. If you have a home modem/router that does NAT you will need to work out how to configure it to allow remote connections (think about the port(s) you need to forward, and how to manage any changing IP addresses). If you have a firewall running on your computer, you may need to unblock the appropriate ports there too There are a few other ways to solve this, e.g. a VPN service on your modem or a local computer. If it proves too difficult you might consider setting up a cloud server with a public IP address on e.g. Amazon, Azure, Alibaba, etc. and installing a broker there. Yet another option is to set up a (free) fully-hosted broker through a provider such as EMQX.io – see the week 8/9 tutorials.

Your broker needs to be configured for some basic authentication and access control:

- username ***student*** and password ***33102021***
- require the client-id to start with ***3310-***

*Public brokers may limit your ability to configure this, so you will need to explain in your report how you manage security/protection of the information, or make yourself robust against somebody overwriting your data.*

You can test the broker works by subscribing to the $SYS topics, which describe the server, and help get you familiar with the information presented there - you will be using them for your analysis later.

## Assignment programming:

You need to write three small programs, the first two of which could be combined into one:

- **A Publisher**: It will send a simple message to the broker, an incrementing counter (0, 1, 2, 3, …). It will send those messages at a specific QoS level (0, 1 or 2), and with a specific delay between messages (0ms, 1ms, 2ms, 10ms, 20ms, 100ms, 200ms). It needs to publish to the topic 'counter/<qos>/<delay>', so e.g. 'counter/1/100' implies qos=1 and delay=100. It must only publish one stream at any given time, based on input from the Controller. *At 0ms delay and qos=0 it should be able to publish very quickly, potentially up to several thousand messages per second.*

- **A Controller**: This will listen to the broker on two topics 'request/qos' and 'request/delay'. Whenever it gets a message on either of them it needs to modify the Publisher's behaviour accordingly. One way is to stop any current publisher process (e.g. send a KILL signal, or at worst put a 2.5-minute time limit on it running) and start a new one with the new parameters, say via command line arguments. Another approach might be through event-handling and multi-threading within a combined publisher/controller, but it is up to you. Please explain in a README file or comments what approach you are taking.

- **An Analyser**: This will listen to the counter on the broker and take measurements (below) to report statistics on the performance of the broker/publisher/network combination. Those measurements should be taken across the range of qos and delay values as above, so that you can compare them. Your Analyser will publish to the 'request/qos' and 'request/delay' topics to get the Publisher to deliver what your Analyser needs at that time. For the QoS, subscribe at the same QoS as requested (i.e. if publisher->broker qos=2 then Analyser->broker should have qos=2 as well). The Analyser does not need to take all (7 delay * 3 qos) measurements sequentially in one run, but the closer in time the measurements occur the better, as network conditions do change all the time.

You can test all of these at home on your local broker. It is only in the final measurements that you will allow another student's Analyser to access your broker, and vice-versa.

## Analysis and Reporting

In your report, please address the following questions:

1. Subscribe to some broker, you can use any mqtt client you like for this, to retrieve some $SYS/# value. Wireshark the handshake for one example of each of the differing QoS-levels (0,1,2), include screenshots in your report that show the wireshark capture (filter for mqtt), and briefly explain 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. **Run your Analyser against your partner's broker**, tell it what you want the publisher to send and record information for 2 minutes. (*Tips: (i) only print individual messages to screen for debugging, otherwise it will slow your code down a lot. (ii) Check with the broker owner what delay there might be between your request and their publisher responding. (iii) Use the counter values to tell you what messages are arriving, or are not arriving as expected, and where they start/stop when you subscribe.)*

    a. Collect statistics, for each delay and QoS level, and measure over the 2-minute period:
        i. The overall average rate of messages you actually 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 any out-of-order messages you see *[percentage]*
            *(i.e. how often do you get a smaller number after a larger number)*
        iv. The mean and median inter-message-gap you see *[milliseconds]*.
            *Only measure for actually consecutive counter-value messages, ignore the gap if you lose any messages in between.*

    b. While measuring the above also
        i. Subscribe to and record the $SYS topics, and identify what, if anything, on the broker do any loss/misordered-rates correlate with. *(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 )*
        ii. Record a traceroute between the Analyser and the broker, and between the Publisher and the broker *(you'll need the broker owner to provide the second one)*

    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.
        iii. Note whether you were able to constrain the ClientID, username/password combinations on your broker, or if you arranged other ways to protect your information, or how you made your code robust against intruders.

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 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.

## Amendments and Assessment

Any amendments to the assignment specification will be published in the wattle forum.

Your assignment will be assessed on

- Your code clarity, and documentation/comments (15%)
- your code subscribing and properly publishing/listening to the broker (15%), and
- your analysis report addressing the questions above (70%)