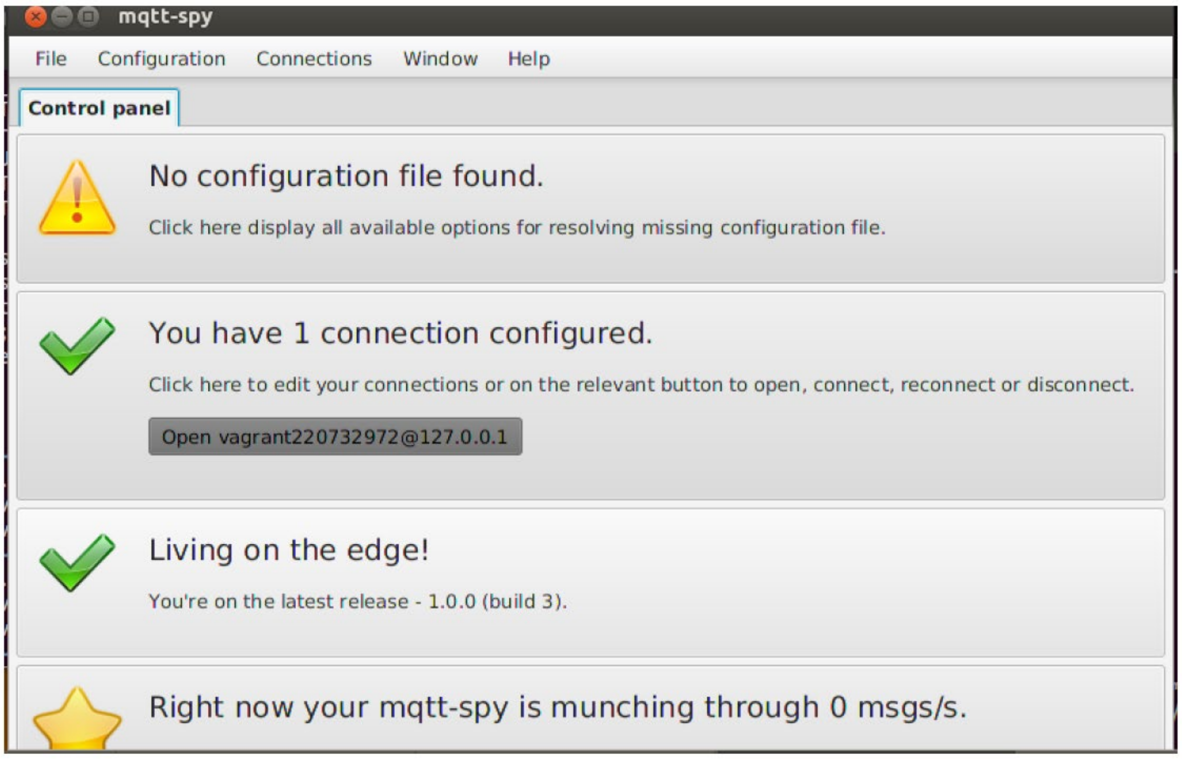
# COMP3310/6331 – Tute/Lab #7

Outline of Tute/Lab:

1. The Mid-Semester exam has been marked and released. Now is a good time to review it, with the answersheet posted to wattle.
2. The second assignment is due at the end of this week. A chance to raise any questions with the tutors about your code and results.
3. Getting into IoT with MQTT:

The main focus this week is getting familiar with MQTT, the core of the final assignment, using a couple of command-line tools and a GUI tool. This will help you to get familiar how MQTT messages are published and reported, and how topics can be used to manage messages.

- **MQTTSpy** – GUI client for connecting to an MQTT server as publisher and subscriber. You will need to install it on your computer from https://www.eclipse.org/paho/components/mqtt-spy/ and you may also need to install Java 7/8 if you don't already have it. Start it up and hopefully get a window like below.
  - If you don't see a 'connection configured' (under 'Public MQTT brokers') in the second panel below, click on it and select to install a 'default' configuration. You should then see a test.mosquitto.org option (amongst others)



No configuration file found.

Click here display all available options for resolving missing configuration file.

You have 1 connection configured.

Click here to edit your connections or on the relevant button to open, connect, reconnect or disconnect.

Living on the edge!

You're on the latest release - 1.0.0 (build 3).

Right now your mqtt-spy is munching through 0 msgs/s.

  - Click on the 'open test.mosquitto.org' button, it should open a new tab that eventually turns green. If not, try broker.mqtt-dashboard.com, or iot.eclipse.org. You can right-click on the tab and 'disconnect' from a broken broker, or else it will keep trying every few seconds.
  - On the 'Connections' menu at the top, select 'Manage Connections'. On the left select the broker you are testing. For the 'Connectivity' tab include your 'ClientID' then click 'Apply'. Note also the Security tab where you may need to enter a username and

password for some brokers (e.g. the one for the assignment will need it). The public brokers do not need this, so 'close and re-open' the connection.
    - Click on the green tab at the top to select the working broker. In that tab you will see the upper half is about publishing messages, and the lower half is about subscribing. Under 'Subscriptions' click on 'New' and type in **$SYS/#** and click 'Subscribe. You'll see a new tab with that topic name appear under 'Subscriptions' and if you click on it, you should receive messages from the broker under the $SYS hierarchy. Note the number of messages counting up, the timestamps, and what the various fields tell you about the server. You can also right-click on the $SYS/# tab and view some charts.
    - Subscribe to the topic **3310/#** and focus on that tab. Without anybody publishing there should be zero content. In the 'Publish message' field enter a topic of **3310/*your-uni-ID*** and under 'Data' put in some text. When you hit 'Publish' it should appear in the messages below, along with every other student's test messages. Try creating additional subtopics below your uniID as you want, and try to explore the hierarchy other students have set up using the '#' and '+' wildcards (recall – '#' is for end-of-line-match-the-rest and '+' is for match-a-field-within-/'s). You can subscribe to overlapping topics.
- There is a perhaps more modern and more reliable client at http://mqtt-explorer.com/ but this has not been evaluated yet. Feel free to give it a go in your own time, and see if it works for you.

- **mosquitto_pub, mosquitto_sub – are** command line clients that can be used as publishers/subscribers. You will need to also install them on your computer from https://mosquitto.org/download/ and on Windows at least you'll need to add the folder to your PATH, or call the clients explicitly. Alternatively for Windows users, and perhaps a little easier for those with linux backgrounds, it's useful to install the Windows Subsystem for Linux (WSL) - https://docs.microsoft.com/en-us/windows/wsl/faq and boot a lightweight Linux instance. You're not running a VM here, but it's not as full-featured a Linux environment. Try your luck.

  As part of the installation you can also choose to install mosquitto (mqtt) as a service and run your own broker locally - look for mosquitto.conf in the install folder. That will give you something local to test against. But for the most part you will want to run against the shared, public brokers, or the assignment broker (which will be enabled next week).

  Read the man page, or see the documentation on mosquitto.org, for each of the clients. The publisher runs just once (like a sensor, it just pushes the message to the broker), while the subscriber (the consumer) will listen till stopped. Note the '-v' flag is helpful for listening, and at least you'll need to specify -h <host> and -t <topic>. Note also that + and # are special characters and may need quotes around topic.

    - Open 2-3 terminal windows, one will be for publishing, the others for subscribing.
    - Similar to the MQTTspy exercise, in one terminal subscribe to $SYS/# on a broker and check the messages come through. Note that some $SYS topics get updated more frequently than others. Think about which topics tell you the most about the current server load. If you only see numbers, try adding '-v' as an option.
    - To understand the 'retain' feature, publish a message to some topic and then subscribe to the topic. Do you see the message? Then publish it again, and see if the subscriber sees it. Then publish it again with the retain flag set – does the subscriber see it? Then stop the subscriber and start it again – does it see the earlier retained published message? How often? Try stopping and starting the subscriber – does it see the earlier retained published message?

- How do you get rid of a retained message? What happens if you retain messages under multiple topics in a tree, how do you get rid of them collectively.
- To help understand the '+' and '#' wildcards, consider a hierarchy of switches like <uniid>/light/<room>/ceiling and <uniid>/light/<room>/wall, and <uniid>/heater/<room> (pick 2-3 room names). Think about which wildcard patterns you use to receive messages about
  - All the lights.
  - All the switches of a certain type in a particular room.
  - All the switches in a particular room, regardless of type
  - All the switches you publish
  - And any other combinations
- Test your guesses. You might want to 'retain' your published messages, and if you do then clear them afterwards.

- **Packet Sniffing:**
  - We'll use wireshark to sniff the traffic to/from your computer. Reminder: Wireshark runs a 'capture' from go to stop and provides a report – there's a lot of traffic, and it goes by quickly! You can then filter the report to identify the packets you're interested in and examine them at your leisure. You can set up a filter in advance before the capture, but they have to be 'right' before you start.
  - Identify packets to/from the MQTT server at different QoS levels ('-q' option) as you publish/subscribe with mosquitto_pub/sub and identify the (MQTT) handshake packets at each QOS level. You may need to look at the MQTT standard (see OASIS website or HiveMQ https://www.hivemq.com/mqtt-essentials/) to confirm which packets you see/should be seeing.

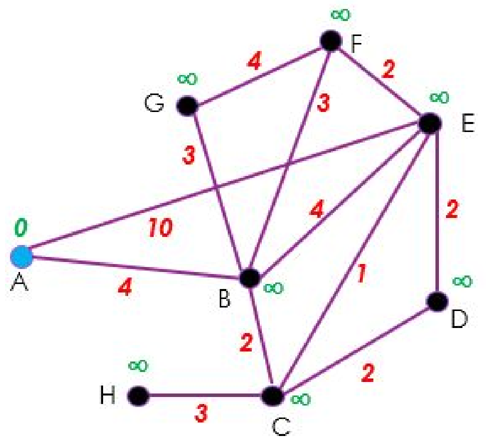
- **Android/iOS clients –** there are a few MQTT clients available for smartphones. It's useful to install one and test it out if you have the time, though trickier to wireshark the traffic…

- The assignment server will require a username/password and also a structured ClientID on your subscription. Check that you can find where to put those in mqttspy and mosquitto_sub/pub as they will be useful for debugging your code in assignment 3.

**Questions from lectures**

- This week we covered **routing**
  - Can you explain the difference between unicast routing, broadcast routing and multicast routing? i.e. what are they trying to achieve?
    - Unicast – get to a single particular endpoint. Multicast – get to all interested 'subscribers' or 'group members' listening on 224/8. Broadcast – get packets to all endpoints within my broadcast domain, and no further!
  - Why does routing tackle fixed parameters about links, and ignore dynamic things like congestion and load?
    - Routing algorithms take time and energy to establish forwarding tables across all routers. The algorithms cannot respond very quickly to changes like congestion/load, nor would you want them to.
  - What is a sink tree, and how does it differ from a source tree?

        - Sink tree is the combination of all shortest paths towards the sink for any given topology. Source tree is the same, but fanning outwards. They may be different if the costs on a path are different by direction.
    - Can you explain the path optimality property?
        - Any series of path elements along the shortest path are themselves the shortest path for nodes along it. If there were a better path element, it would be on the shortest path!
    - Walk through the following network with Dijkstra's algorithm, starting from A, and identify the shortest path A to E



    - What's the downside of using Dijkstra's algorithm? Why is Distance-Vector (Bellman-Ford) better?
        - The need to (a) know the **entire** topology up front, and (b) to have **all** nodes know that.
    - What's the benefit of regional route aggregation? What's the downside?
        - Benefit is significant shortening of forwarding/routing tables across the network, and a reduction in processing load on the routers to maintain them. The downside is that you will probably end up with some sub-optimal paths.
    - Why do we use different routing algorithms internally versus globally?
        - Internally we should/will have perfect knowledge, and a shorter network radius, so Link-state or Distance-Vector will (a) work, and (b) converge within a reasonable time. Outside of that *we* will have imperfect knowledge, so easier to just throw it outside and let the Internet handle it.
    - What is the difference between a transit and a peering arrangement?
        - Transit – you can get to other networks through me (for a fee). Peering – you can get to devices within my network, only (for usually no fee).