COMP3310/6331 – Tute/Lab #4 – Week 10 – 7-8 May

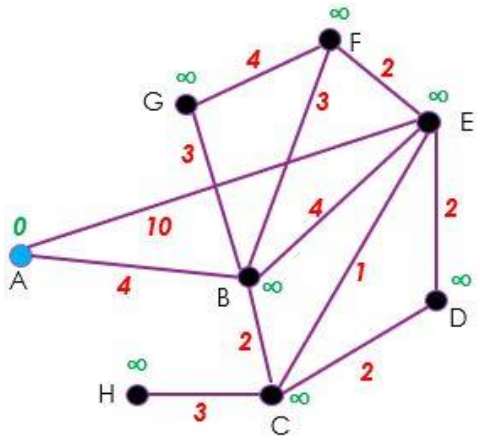
Outline of Tute/Lab:

- **MQTT familiarity**
  - **MQTTSpy** – GUI client for connecting to MQTT server as publisher and subscriber. See separate notes for installing MQTTSpy on the lab machines and setting up a connection to listen to $SYS/# and the counter queues. (There may be an update to the notes shortly, there have been some small gremlins causing grief).
  - **Android/iOS clients –** there are a few smartphone MQTT clients available in the respective Play/iTunes stores. It's useful to install one and test it out, though a little trickier to wireshark the traffic…
  - **mosquitto_pub, mosquitto_sub –** command line clients that can be used as publishers/subscribers. You need to install them locally (sudo apt install mosquitto_clients). Read the man page for each of them and understand the connection requirements (client-index of 3310-<your.uni.id>, need to provide the username and password). The publisher runs just once, while the subscriber will listen till stopped.

    - Open 2-3 terminal windows on your machine, one will be for publishing, the others for subscribing.
    - Similar to the MQTTspy exercise, in one terminal subscribe to $SYS/# on 3310exp.hopto.org 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.
    - 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, if you do then clear them all afterwards.

- **Packet Sniffing:**
  - We'll use wireshark (with admin privileges, via gksudo) 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. This is also good practice for understanding filters. 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 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) to confirm which packets you see/should be seeing.

- **Questions from lectures**
  - This week we briefly covered **routing** and **congestion**
    - Can you explain the difference between unicast routing, broadcast routing and multicast routing? i.e. what are they trying to achieve?
    - Why does routing tackle fixed parameters about links, and ignore dynamic things like congestion and load?
    - What is a sink tree, and how does it differ from a source tree?
    - Can you explain the path optimality property?
    - 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?
    - What's the benefit of regional route aggregation? What's the downside?
    - Why do we use different routing algorithms internally versus globally?
    - What is the difference between a transit and a peering arrangement?
    - Talking congestion, what does having a receive-window achieve?
    - Where does congestion usually happen?
    - Why do we get a collapse in goodput as we approach congestion?
    - Why does TCP traffic (often) have a sawtooth pattern?
    - How can we detect congestion? (3 ways, at least)
    - Why does TCP Slow Start help improve throughput?
    - How much loss can Fast Retransmit handle?
    - What's the assumption underpinning Fast Recovery?