**MQTT and CoAP exercises (Part 1)**

March 7, 2024

**Setup of MQTT files (only needs to be done once):**

The commands in this section install the necessary files for the MQTT part of the exercise and are only required once. Start the course’s virtual machine, open a terminal and execute the following commands in the terminal to download and install the relevant software:

* **git clone https://github.com/lstaalhagen/C34351MQTT**
* **cd C34351MQTT**
* **sudo ./mqtt-setup.sh**

The necessary software (MQTT broker and clients) should now be installed in your VM.

**Initialize the VM for the MQTT exercise**

Every time the VM has been restarted, the following commands must be executed to setup some necessary directories:

* **cd ~/C34351MQTT**
* **sudo ./mqtt-init.sh**

**Task 0 (Verify connectivity – Does not have to be documented in the exercise report):**

Open a terminal window and execute the following commands:

* **cd ~/C34351MQTT**
* **sudo ./netgenerate.sh**

The second of these commands creates a small *virtual* network using Linux network namespaces with three hosts (H1, H2, and H3) connected to a switch, as illustrated in Figure 1. The IPv4 addresses of the hosts are **10.0.0.1** for H1, **10.0.0.2** for H2, and **10.0.0.3** for H3. The command also opens an Xterm window on each host so you can execute commands on a specific host.



Figure 1 – Switched network with three hosts.

Select an arbitrary host and (in that host’s Xterm window) check if you can ping the other two hosts with the command **ping -c1 <ip-address>**

If the virtual network is not needed anymore, it can be deleted with the script **clearnet.sh** in the C34351MQTT directory. The script requires administrative privileges, so it must be executed with the command **sudo ./clearnet.sh** (If your current directory is not C34351MQTT, you can change to it with **cd ~/C34351MQTT** ).

Note that the virtual network created with the **sudo ./netgenerate.sh** command disappears if the virtual machine is restarted, so you will need to execute the commands at the start of Task 0 to recreate it after a restart of the VM.

**Task 1 (Basic MQTT connections):**

In H1’s Xterm window, type **ip addr** and verify that one of its interfaces has an IP address in the **10.0.0.0/24** range. Type the following command in H1’s Xterm window to start the MQTT-broker on this host: **sudo mosquitto -c ./mosquitto.conf**

Start Wireshark in the VM’s normal[[1]](#footnote-1) terminal window ( **sudo wireshark** ) and start a capture on the **veth11** interface[[2]](#footnote-2). It might be a good idea to set the display filter (just below the toolbar at the top of the window) to **mqtt** as shown in Figure 2:

**A screenshot of a computer

Description automatically generated**

Figure 2 – Wireshark with **mqtt** as display filter.

In H2’s Xterm window, start an MQTT-subscriber:

* **mosquitto\_sub -v -h <H1’s IP-address> -t '#'**

Stop the client again after a few seconds with Ctrl-C in the Xterm window, switch to the Wireshark window and stop the capture. You should have captured five MQTT packets; list these packets and explain their purpose.

Start a new capture with Wireshark and restart the subscriber client again (as above), but keep it running for at least 2 minutes – what do you observe in Wireshark? Can you relate the additional messages and their intervals to one of the earlier packets?

Start a new Wireshark capture. In H3’s Xterm window, publish a message with:

* **mosquitto\_pub -h <H1’s IP-address> -t Topic1 -m "Hello World!"**

Verify that your message appears in H2’s Xterm window. Switch to Wireshark and stop the capture. List and explain the MQTT packets related to the publishing of the message. Do the MQTT “Publish Message” contents correspond to the message string of the mosquitto\_pub command? (Hint: Check the “raw” bytes of the message in Wireshark – if the ‘Packet Bytes’ section of Wireshark’s window is not shown, make sure that ‘Packet Bytes’ is selected in the View menu.)

**Task 2 (Message QoS):**

MQTT supports three different “QoS”-levels related to message delivery, but in Task 1, only the default QoS level (level 0) was used. A non-default QoS-level can be specified for both the MQTT-subscriber and the MQTT-publisher by appending the option **-q <qos-level>** (where **<qos-level>** is **1** or **2** ) to the command. For each non-default QoS level, capture MQTT traffic with Wireshark as in Task 1 and explain the difference between the different QoS levels 0, 1, and 2 (It may be helpful to consult section 4.3 in the official MQTT-specification – see https://mqtt.org).

Also, capture and explain the MQTT traffic if the publisher uses QoS-level 0 but the subscriber has requested QoS-level 2.

**Task 3 (Wills):**

MQTT permits a publisher to set up a “will”, i.e., a message to be sent in case the publisher unexpectedly disconnects from the broker. In Tasks 1 and 2, the publisher disconnected immediately after a single message, but the publisher can also be instructed to repeat a message a specified number of times with a specific interval between the messages. Execute the following command in H3’s Xterm window and verify/document with Wireshark that a) the message is sent 5 times with 2 seconds intervals between each message, and b) that the publisher disconnects after the 5th message has been sent:

**mosquitto\_pub -h 10.0.0.1 -t Topic1 -m "Hello World!" --repeat 5 --repeat-delay 2**

Consult the documentation for the **mosquitto\_pub** command (found on <https://mosquitto.org>) and determine the extra arguments that must be added to the command so that the publisher creates a “will” with the text “Help me!” to be published to the topic “Wills” if it disconnects unexpectedly.

Start a Wireshark capture. Rerun the publisher command with these extra arguments and the value 60 for the **--repeat-delay** option, so the command runs for at least 4 minutes.

Execute (in a new terminal window of the VM) the command (this will forcefully kill the publisher):  
**sudo pkill mosquitto\_pub**

Document in your report with the Wireshark capture and a screenshot of the subscriber’s Xterm window that the “will” is sent to the subscriber.

**Task 4 (Retained messages):**

Normally, an MQTT client that starts subscribing to a topic will not receive anything before a message is published to that topic. However, MQTT also supports “retained” messages. If a publisher indicates that messages published on a topic should be retained, the broker keeps a copy of the last published message on that topic, so if a client starts subscribing to this topic, it will receive the retained message immediately.

Investigate two scenarios[[3]](#footnote-3), one with and one without retained messages, illustrating the difference between the scenarios, and document these with Wireshark captures. You may also need to consult the documentation for the **mosquitto\_pub** command for this task.

Note that if you want to “unretain” a message on the broker, you must publish an empty message (also called a null message) to be retained, i.e., use **-m ''** instead of **-m 'Hello world!'** as an argument for **mosquitto\_pub.**

1. I.e., not in any of the Xterm windows of the hosts. [↑](#footnote-ref-1)
2. The **veth11** interface is the switch’s interface towards host H1 [↑](#footnote-ref-2)
3. Neither scenario should use wills, i.e., the options for wills identified in task 3 should not be used in task 4. [↑](#footnote-ref-3)