Checkpoint 3 MQTT

Part 1 Environment Set up

MQTT: Message Query Telemetry Transport protocol is used to exchange messages between devices with constrained resources. It uses the Publish/Subscribe (pub/sub) model and a Broker-Client architecture.

MQTT components:

Topic: The subject of interest.
Publisher: A device or system that publishes messages on a topic
Subscriber: A device or system that subscribes to messages on a topic
*Client***:** A device or system that fills the role of a publisher and/or a subscriber
The Broker: The system that routes messages between MQTT clients. Because the clients are decoupled, the messages are never exchanged directly and must pass through the broker first.

Important Note: Due to project scope, if a group chooses to implement the MQTT protocol for the course project, the broker would also be a client.

Environment set up:

1. **Broker**

Set up a local mosquitto broker on the machine that you have been using as the HTTP server. Mosquitto by Eclipse is one of many open source brokers. It's free and available on all platforms. The initial setup is straightforward. By default, the broker listens on port 1883.
The link below contains platform-specific download and installation instructions for the Mosquitto broker.

https://mosquitto.org/download/

2. **Client**

Once the broker is set, install the client on a virtual server (Raspberry pi). Follow these steps on the virtual machine to download the mosquitto client:

Important: We need to download ONLY clients on the virtual machine not the broker.

a. To use the new repository you should first import the repository package signing key:

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
```

b. Then make the repository available to apt:

```
cd /etc/apt/sources.list.d/
```

c. We installed debian-buster in checkpoint 1. Therefore we use:

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list
```

d. Then update apt information, install the mosquitto client and reboot the VM:

```
apt-get update

sudo apt-get install mosquitto-clients

reboot
```

Using the terminal test for the successful installation.

At the broker:

Change the directory to where the mosquitto broker is installed. Using cd command. From this directory use:

```
mosquitto_pub -h <IP address of the broker> -t 'test' -m 'message'
```

At the client:

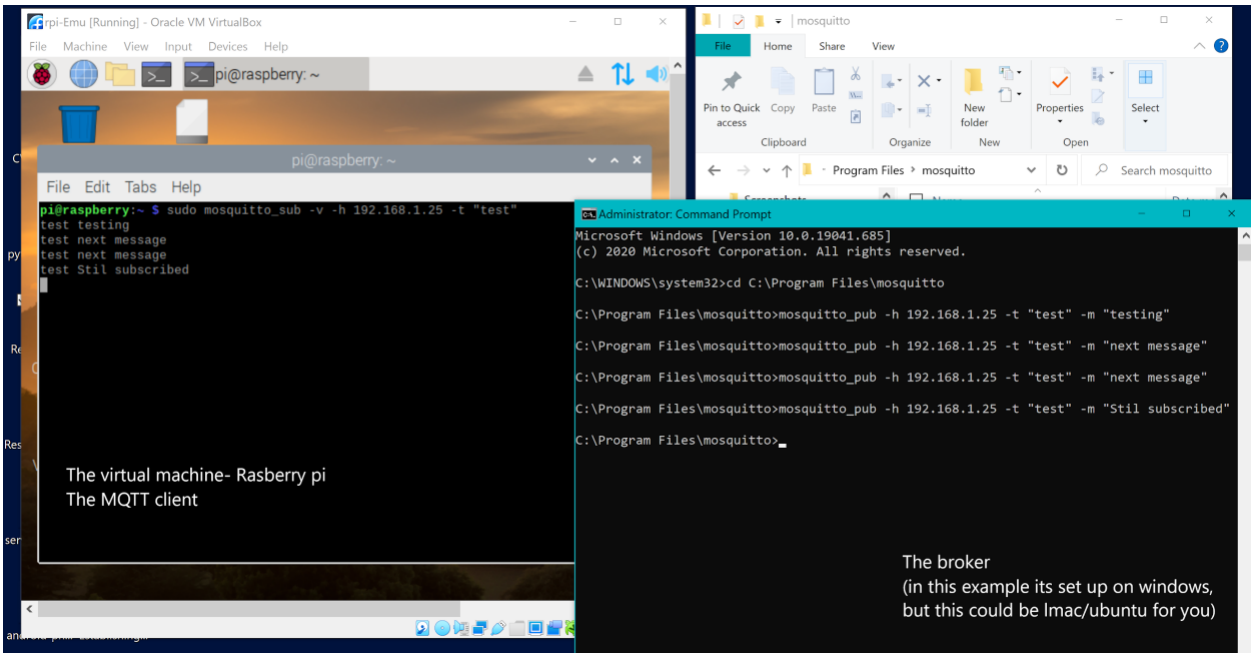
At the raspberry pi you don't need to change the directory. Open the terminal and type:

```
mosquitto_sub -h <IP address of the broker> -t 'test' -d
```

Out of the many parameters for defining a MQTT pub/sub model we are using the following:

-h = host
-t = topic
-m = message
-d = debug

The screenshot of the example window shows the following steps to test the installation:



Part 2 Exploring MQTT functionalities.

On completion of the set up, install python paho library using the following link:
https://pypi.org/project/paho-mqtt/
The boilerplate for a simple publisher and subscriber are provided to you in the project folder. Using them as the starting point, implement the following:

1. Only authorised users can publish and subscribe to a topic “status”.
    a. Publisher :
        i. User name : sam
        ii. Password: sam
    b. Subscriber :
        i. User name: alice
        ii. Password : alice
2. Quality of service
    Hint: In order to define QoS >0, the connection has to be persistent.
    a. Implement pub/sub using quality of service 0, 1 and 2
    b. Observe the different control packets being exchanged. Use debug mode for this.
    c. Observe the packet size and time latency that is caused due to the change in quality of service.
    d. What happens when the publisher posts messages on a topic with a QoS > than the QoS defined by the subscriber of that topic? Why?
3. Keep alive

a. Implement a pub/sub message exchange that has a keep alive value of 5 seconds and another message exchange for 25 seconds.
b. Observe and report the differences in terms of network usage.
c. What should be the ideal keep alive value for a topic that monitors the health of you smart lock? Justify your answer

4. Last will testament
   a. Implement a last will message for both publisher and subscriber on a the topic "status"
   b. Stimulate the network-outage environment by closing the IDE
   c. Observe and report the behavior when the connection is closed by the broker vs when the connection is lost due to unforeseen circumstances such as network outage.

Reference:

https://mosquitto.org/man/mosquitto_sub-1.html
https://mosquitto.org/man/mosquitto_pub-1.html
http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/
http://www.steves-internet-guide.com/topic-restriction-mosquitto-configuration/
http://www.steves-internet-guide.com/understanding-mqtt-qos-2/
http://www.steves-internet-guide.com/mqtt-keep-alive-by-example/
http://www.steves-internet-guide.com/mqtt-clean-sessions-example/
http://www.steves-internet-guide.com/mqtt-last-will-example/
https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals/
https://openlabpro.com/guide/mqtt-packet-format/

Understand paho to add QoS and last will :
https://www.hivemq.com/blog/mqtt-client-library-paho-python/