Demonstration of Eclipse Mosquitto

Abhishek Kumar (181CO201)

Before We Begin

- ➢ I expect *some* background knowledge of MQTT.
- ➢ I will *not* focus on packet structure and packet flow.
- ➢ Instead, I *will* focus on the functionality provided by Eclipse Mosquitto and its use in client libraries.

Example Application

- ➢ RngProducer simulates an IoT device, publishing a random floating number between 0 and 50 every five seconds.
- ➢ RngConsumer reads the readings and calculates the running average.
- ➢ Since both producer and consumer are on same machine, I am using Process ID as a unique identifier.

Refer to my repository for complete source code

Installing Mosquitto

- ➢ Can be installed as an application or using Docker:
  - ○ How to Install The Mosquitto MQTT Broker on Linux
  - ○ Running the eclipse-mosquitto MQTT Broker in a docker container
- ➢ With Mosquitto 2 and later, Mosquitto accepts **only** authenticated connections from localhost.
- ➢ Allow anonymous connections by adding the following lines to configuration file:

```
listener 1883

allow_anonymous true
```

Installing Mosquitto

- ➢ I have used the docker image method.
- ➢ Created a folder mosquitto in project root with three subfolders - config, data and logs.
- ➢ The important lines in configuration file are:

```
listener 1883

password_file /mosquitto/config/password
```

Authentication and Authorization

- ➢ Authentication can be managed by password files and or the plugin mosquitto-go-auth, which offers a variety of storage backends.
- ➢ Dynamic Security Plugin provides control over role based authentication and access control features.

Authentication and Authorization

➢ I have used password files as they are easier to set up.

➢ To create a password file, use:

```
mosquitto_passwd -c <passfile> <username>
```

➢ You add new users or overwrite existing password using the same command as well.

➢ Use username_pw_set to pass username, password to the broker.

Topics

- ➢ Topics are used by the broker to filter messages for each connected client. For example: ‘home/groundfloor’.
- ➢ Clients can publish or subscribe to topics without prior initialization.
- ➢ Wildcards ‘+’ and ‘#’ can be used to subscribe to multiple topics simultaneously.
- ➢ Topics beginning with $ are reserved for internal statistics of broker.

Topics

- ➢ RngProducers are publishing to ‘random_numbers/{pid}’ and ‘status/{pid}’ topics.
- ➢ Note how topics are specific to one particular producer.
- ➢ RngConsumer use wildcards to subscribe to producer-topics.
- ➢ RngConsumer also subscribe to‘$SYS/broker/clients/connected’ to monitor the number of connected clients.

Quality of Service

- ➢ Quality of Service is an agreement between clients and broker to guarantee delivery of messages:
  - At most once (0)
  - At least once (1)
  - Exactly once (2)
- ➢ Both publisher and subscriber can define their QoS levels, leading to a downgrade if subscriber sets a lower level.

Quality of Service

- ➢ Readings are published with QoS 0 - some loss is acceptable.
- ➢ Status updates are published with QoS 1 - they should reach broker at least once (idempotent, so multiple deliveries is okay).
- ➢ Status updates are subscribed with QoS 2 - should reach exactly once as multiple disconnects will try to clear buffer multiple times.

Persistent Sessions

- Clients can avoid re-subscribing by creating a persistent session.
- Broker stores:
  - Client id
  - Subscriptions
  - Messages with QoS 1 or 2 that have not been acknowledged
  - Messages with QoS 1 or 2 published while the client was disconnected.
- Client start or end a persistent session using **cleanSession** flag.

Persistent Sessions

- ➢ Client ID must be provided during connection for persistent session.
- ➢ RngProducers connect with a clean session - they only publish messages.
- ➢ RngConsumers connect with a persistent session - no need to subscribe again if done already.
- ➢ Also need QoS 1 and 2 messages while the consumer was offline.

```
mqtt.Client(client_id='...', clean_session=True)
```

Retained Messages

- ➢ Helps newly-subscribed clients get a status update immediately rather than wait for the next update.
- ➢ Broker stores the retained message for a topic until overwritten.
- ➢ Retained message can be deleted by publishing a retained message with payload of zero bytes.

Retained Messages

- ➢ The status messages are retained, so a consumer is always updated.
- ➢ The last reading is also retained.

```
publish(topic, payload, retain=True)
```

Last Will And Testament (LWT)

- ➢ A special message that is sent when a client disconnects *disgracefully* (that is, without DISCONNECT message).
- ➢ LWT can be used by other clients to handle unexpected failures.
- ➢ If the client disconnects gracefully, LWT is discarded - meaning it must be re-established even with a persistent connection.

Last Will And Testament (LWT)

- RngProducer set their LWT message same as the disconnected status update to ensure that consumers are notified of the change.

Further Reading

- ➢ Documentation | Eclipse Mosquitto
- ➢ MQTT Essentials | HiveMQ
- ➢ Steve's Internet Guide - a website dedicated to MQTT, IoT and python
- ➢ paho.mqtt.python - A client library for MQTT by Eclipse