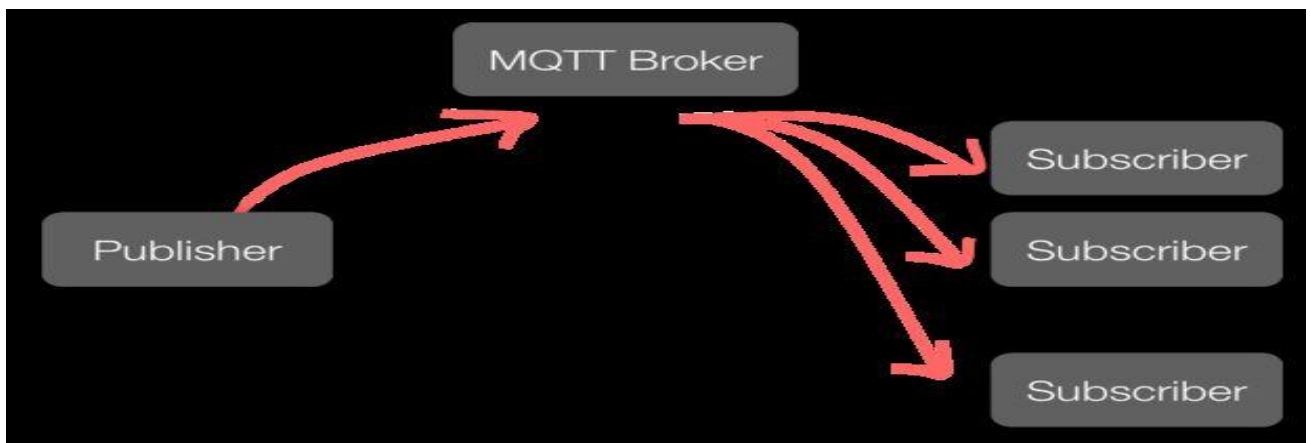


MQTT Documentation

I. Introduction

MQTT stands for “Message Queue Telemetry Transport” and is an extremely “lightweight” machine-to-machine connectivity protocol. This protocol’s first version was written by Andy Stanford-Clark and Arlen Nipper in 1999. MQTT is becoming more popular than ever before with the increasing use of mobile device and smartphone applications such as Facebook’s Messenger and Amazon Web Services. This protocol is used in WIFI or low bandwidth network. MQTT does not require any connection with the content of the message. The protocol is based on a TCP/IP protocol, and the key of MQTT is the publish-subscribe pattern. The biggest advantage is that MQTT can provide reliable messaging services based on little coding and limited bandwidth, which is highly accepted by M2M and IoT areas. It's based around a message broker, which can be highly parallelized and event-driven making a single broker easily scalable to tens of thousands of messages per second. There are several different broker implementations. In our discussion we will be using Mosquitto to implement MQTT.



II. MQTT Implementation on Ubuntu (using Mosquitto on VirtualBox)

Step 1: Installing Mosquitto clients

```
sudo apt-get install mosquitto mosquitto-clients
```

Step 2: Installing ‘paho-mqtt’ using pip and implementing MQTT on Python

```
pyenv activate ENV3  
pip install paho-mqtt  
python3
```

Code for MQTT implementation:

```
import paho.mqtt.client as mqtt  
import time  
  
def on_message(client, userdata, message):  
    print("message received " ,str(message.payload.decode("utf-8")))  
    print("message topic=",message.topic)  
    print("message qos=",message.qos)  
    print("message retain flag=",message.retain)  
  
def on_log(client, userdata, level, buf):
```

```

print("log: ",buf)

broker_address="localhost"
print("creating new instance")
client = mqtt.Client("i523") #create new instance
client.on_log=on_log
client.on_message=on_message #attach function to callback

print("connecting to broker")
client.connect(broker_address) #connect to broker
client.loop_start() #start the loop
print("Subscribing to topic","robot/leds/led1")
client.subscribe("robot/leds/led1")

print("Publishing message to topic","robot/leds/led1")
client.publish("robot/leds/led1","OFF")

time.sleep(4) # wait
client.loop_stop() #stop the loop

```

Since we already had mosquitto and paho-mqtt installed, our results will look slightly different than someone installing for the first time, the functionality will be the same.

Results:

```

gabriel@gabriel-VirtualBox:~$ sudo apt-get install mosquitto mosquitto-clients
[sudo] password for gabriel:
Reading package lists... Done
Building dependency tree
Reading state information... Done
mosquitto is already the newest version (1.4.8-1ubuntu0.16.04.2).
mosquitto-clients is already the newest version (1.4.8-1ubuntu0.16.04.2).
0 upgraded, 0 newly installed, 0 to remove and 82 not upgraded.
gabriel@gabriel-VirtualBox:~$ pyenv activate ENV3
pyenv-virtualenv: prompt changing will be removed from future release. configure `export
PYENV_VIRTUALENV_DISABLE_PROMPT=1` to simulate the behavior.
(ENV3) gabriel@gabriel-VirtualBox:~$ pip install paho-mqtt
Requirement already satisfied: paho-mqtt
in ./pyenv/versions/3.6.2/envs/ENV3/lib/python3.6/site-packages
(ENV3) gabriel@gabriel-VirtualBox:~$ python3
Python 3.6.2 (default, Sep 1 2017, 15:00:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import paho.mqtt.client as mqtt
>>> import time
>>>
>>>
>>> def on_message(client, userdata, message):
...     print("message received " ,str(message.payload.decode("utf-8")))
...     print("message topic=",message.topic)
...     print("message qos=",message.qos)
...     print("message retain flag=",message.retain)
...
>>> def on_log(client, userdata, level, buf):
...     print("log: ",buf)
...
>>> broker_address="localhost"
>>> print("creating new instance")
creating new instance
>>> client = mqtt.Client("i523") #create new instance
>>> client.on_log=on_log
>>> client.on_message=on_message #attach function to callback
>>>
>>> print("connecting to broker")
connecting to broker
>>> client.connect(broker_address) #connect to broker
log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'i523'
0
>>> client.loop_start() #start the loop
>>>
>>> print("Subscribing to topic","robot/leds/led1")
Subscribing to topic robot/leds/led1
>>> client.subscribe("robot/leds/led1")
(0, 1)
>>>
>>> print("Publishlog: Received CONNACK (0, 0)

```

Here, we can observe the messaging process through the broker Mosquitto using the subscribe and publish commands.

III. MQTT Implementation on Windows (using a public broker)

Step 1: Installing 'paho-mqtt' using pip and implementing MQTT on Python

```
pip install paho-mqtt
python
```

Code for MQTT implementation:

```
import paho.mqtt.client as mqtt
import time

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

def on_log(client, userdata, level, buf):
    print("log: ",buf)

broker_address="iot.eclipse.org"
print("creating new instance")
client = mqtt.Client("i523") #create new instance
client.on_log=on_log
client.on_message=on_message #attach function to callback

print("connecting to broker")
client.connect(broker_address) #connect to broker
client.loop_start() #start the loop
print("Subscribing to topic","robot/leds/led1")
client.subscribe("robot/leds/led1")

print("Publishing message to topic","robot/leds/led1")
client.publish("robot/leds/led1","OFF")

time.sleep(4) # wait
client.loop_stop() #stop the loop
```

Since we already had paho-mqtt installed, our results will look slightly different than someone installing for the first time, the functionality will be the same.

Results:

```
C:\Users\gabejone>python
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul 5 2016, 11:41:13) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import paho.mqtt.client as mqtt
>>> import time
>>> def on_message(client, userdata, message):
...     print("message received ",str(message.payload.decode("utf-8")))
...     print("message topic=",message.topic)
...     print("message qos=", message.qos)
...     print("message retain flag=",message.retain)
...
>>> def on_log(client, userdata, level, buf):
...     print("log: ",buf)
...
>>> broker_address="iot.eclipse.org"
>>> print("creating new instance")
creating new instance
>>> client = mqtt.Client("i523") #create new instance
>>> client.on_log=on_log
>>> client.on_message=on_message #attach function to callback
>>> print("connecting to broker")
connecting to broker
>>> client.connect(broker_address) #connect to broker
log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'i523'
0
>>> client.loop_start() #start the loop
>>> log: Received CONNACK (0, 0)

>>> print("Subscribing to topic","robot/leds/led1")
Subscribing to topic robot/leds/led1
>>> client.subscribe("robot/leds/led1")
(0, 1)
>>> log: Received SUBACK

>>> print("Publishing message to topic","robot/leds/led1")
Publishing message to topic robot/leds/led1
>>> client.publish("robot/leds/led1","OFF")
log: Sending PUBLISH (d0, q0, r0, m2), 'b'robot/leds/led1', ... (3 bytes)
<paho.mqtt.client.MQTTMessageInfo object at 0x00000244637DFA48>
>>> log: Received PUBLISH (d0, q0, r0, m0), 'robot/leds/led1', ... (3 bytes)
message received OFF
message topic= robot/leds/led1
message qos= 0
message retain flag= 0

>>> time.sleep(4) #wait
```

Here, we can observe the messaging process through the public broker "iot.eclipse.org" using the subscribe and publish commands.

IV. MQTT Implementation on MAC OSX (using Mosquitto)

Step 1: Installing Mosquitto clients

Open a terminal and use homebrew to install mosquito.

```
brew install mosquitto
```

The install script finishes by providing the instructions to start the MQTT server on startup.

```
ln -sfv /usr/local/opt/mosquitto/*.plist ~/Library/LaunchAgents
```

Finally, to save a restart, the server can be started now by running the following:

```
launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mosquitto.plist
```

Now you can test the installation and ensure the server is running successfully. Open a new command window and start a listener.

```
mosquitto_sub -t topic/state
```

In another window, send a message to the listener.

```
mosquitto_pub -t topic/state -m "Hello World"
```

This ensures the server is running.

Step 2: Implementing MQTT on Python

Code for MQTT implementation:

```
import paho.mqtt.client as mqtt
import time

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

def on_log(client, userdata, level, buf):
    print("log: ",buf)

broker_address="localhost"
print("creating new instance")
client = mqtt.Client("i523") #create new instance
client.on_log=on_log
client.on_message=on_message #attach function to callback

print("connecting to broker")
client.connect(broker_address) #connect to broker
client.loop_start() #start the loop
print("Subscribing to topic","robot/leds/led1")
client.subscribe("robot/leds/led1")

print("Publishing message to topic","robot/leds/led1")
client.publish("robot/leds/led1","OFF")

time.sleep(4) # wait
client.loop_stop() #stop the loop
```

Results:

```
[(2.7.13) 149-161-210-172:~ joshlipe-melton$ python
Python 2.7.13 (default, Sep 1 2017, 10:15:06)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import paho.mqtt.client as mqtt
>>> import time
>>>
>>>
>>> def on_message(client, userdata, message):
...     print("message received " ,str(message.payload.decode("utf-8")))
...     print("message topic=",message.topic)
...     print("message qos=",message.qos)
...     print("message retain flag=",message.retain)
...
>>> def on_log(client, userdata, level, buf):
...     print("log: ",buf)
...
>>> broker_address="localhost"
>>> # broker_address="test.mosquitto.org"
... # broker_address="broker.hivemq.com"
... # broker_address="iot.eclipse.org"
...
>>> print("creating new instance")
creating new instance
>>> client = mqtt.Client("i523") #create new instance
>>> client.on_log=on_log
>>> client.on_message=on_message #attach function to callback
>>>
>>> print("connecting to broker")
connecting to broker
>>> client.connect(broker_address) #connect to broker
('log: ', 'Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=i523')
0
>>> client.loop_start() #start the loop
>>>
('log: ', 'Received CONNACK (0, 0)')

>>> print("Subscribing to topic","robot/leds/led1")
('Subscribing to topic', 'robot/leds/led1')
>>> client.subscribe("robot/leds/led1")
(0, 1)
>>>
>>> ('log: ', 'Received SUBACK')
print("Publishing message to topic","robot/leds/led1")
('Publishing message to topic', 'robot/leds/led1')
>>> client.publish("robot/leds/led1","OFF")
('log: ', "Sending PUBLISH (d0, q0, r0, m2), 'robot/leds/led1', ... (3 bytes)")
<paho.mqtt.client.MQTTMessageInfo object at 0x10d8edd60>
>>>
>>> ('log: ', "Received PUBLISH (d0, q0, r0, m0), 'robot/leds/led1', ... (3 bytes)")
('message received ', 'OFF')
time.sleep(4) # wait(
'message topic=', 'robot/leds/led1')
('message qos=', 0)
('message retain flag=', 0)
client.loop_stop() #stop the loop

>>> client.loop_stop() #stop the loop
>>>
>>> █
```

We observe the same results here as well.

V. MQTT Implementation on Raspberry Pi

This is a tutorial guide on how to install MQTT on Raspberry Pi. Due to the unavailability of Raspberry Pi with any of the group members, we could only read through Online tutorials and come up with this. We have not implemented it.

There are two ways of installing MQTT on Raspberry Pi. The first one is straightforward: Directly connecting the keyboard, mouse and monitor to a Pi, turning it on and going to the terminal of the raspberry pi and installing it from there. The second one is to access the Raspberry Pi through a computer using Putty, logging into the root and installing it. One thing that needs to be taken care of while doing this is that both, the MQTT client (Publisher/Subscriber Client) and broker(Mosquitto) needs to be installed on Pi. Following steps need to be implemented to do this:

Step 1: Installation of the MQTT Broker on Raspberry Pi

To use the new repository, the repository package signing key should be imported by running the following commands:

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

```
sudo apt-key add mosquitto-repo.gpg.key
```

And then the repository should be made available to apt:

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

Then one of the following commands should be executed, depending on which version of debian is being used:

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

OR

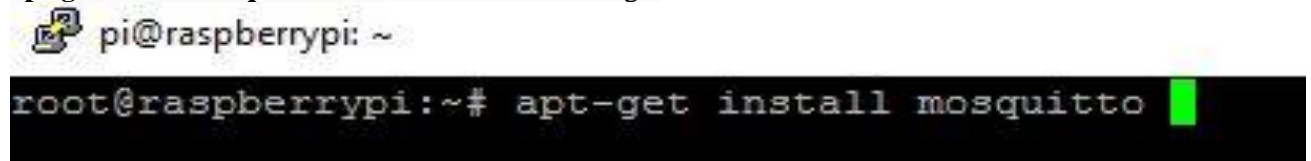
```
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

Then update apt information to just get the latest versions of the software:

```
apt-get update
```

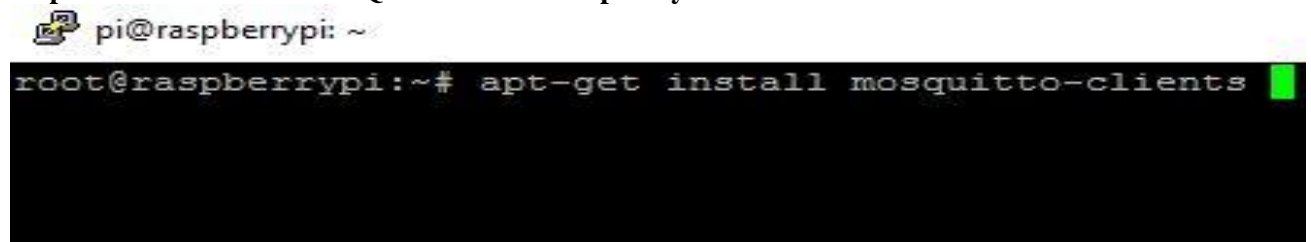
Mosquitto is installed by implementing the following command:

apt-get install mosquitto as shown in the below image:



```
pi@raspberrypi: ~  
root@raspberrypi:~# apt-get install mosquitto
```

Step 2: Installation of the MQTT Client on Raspberry Pi



```
pi@raspberrypi: ~  
root@raspberrypi:~# apt-get install mosquitto-clients
```

The MQTT client needs to be installed on raspberry pi by running the following command:

apt-get install mosquitto-clients

Step 3: Testing the Working of MQTT

To test whether the Mosquitto broker is running successfully, Open two putty programs, say Putty1 and Putty2 on windows which are connected to Pi via ssh.

In putty1 window type the command with any topic name, say for example 'armtronix_mqtt':

```
mosquitto_sub -d -t armtronix_mqtt
```

This will subscribe Pi to MQTT broker as shown in the figure:

```
pi@raspberrypi:~$ mosquitto_sub -d -t armtronix_mqtt
Client mosqsub/7111-raspberryp sending CONNECT
Client mosqsub/7111-raspberryp received CONNACK
Client mosqsub/7111-raspberryp sending SUBSCRIBE (Mid: 1, Topic: armtronix_mqtt, QoS: 0)
Client mosqsub/7111-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/7111-raspberryp received PUBLISH (d0, q0, r0, m0, 'armtronix_mqtt', ... (15 bytes))
Hello armtronix
Client mosqsub/7111-raspberryp received PUBLISH (d0, q0, r0, m0, 'armtronix_mqtt', ... (11 bytes))
Test passed
```

In putty2 window type the following command with the same topic i.e armtronix_mqtt and a message, say **Hello Armtronix** like:

mosquitto_pub -d -t armtronix_mqtt -m "Hello armtronix"

In window putty1, the message **Hello armtronix** will be visible, thus making sure that the Mosquitto broker is running successfully.

Step 4: Testing MQTT From Another System

MQTT can also be connected through the network/internet from another system by just typing the IP address of the host

```
naren@Naren-PC:~$ mosquitto_sub -h 192.168.1.10 -t armtronix_mqtt
```

Window 1

```
naren@Naren-PC:~$ mosquitto_pub -h 192.168.1.10 -t armtronix_mqtt -m "Hi this is Armtronix_server"
```

Window 2

```
naren@Naren-PC:~$ mosquitto_sub -h 192.168.1.10 -t armtronix_mqtt
Hi this is Armtronix_server
```

Window 1

As can be seen from the image 1, the following command should be implemented in window 1.

mosquitto_sub -h 192.168.1.10 -t armtronix_mqtt

In the terminal window 2 we are publishing by entering the command as shown in the image 2

mosquitto_pub -h 192.168.1.10 -t armtronix_mqtt -m "Hi this is Armtronix_server"

Now in the window 1 we will be able to see the message **Hi this is Armtronix_server** as shown in the image 3, thus ensuring that it is running successfully.