Universidade Federal de Itajubá

Segundo Trabalho de TELC03: Protocolo MQTT

Alvaro Munhoz Mota – 20847

6 de dezembro de 2018

Sobre a aplicação

O intuito desta aplicação foi simular a comunicação por protocolo MQTT. Para isso, foram criados dois tipos de cliente: um publisher, responsável por fazer as publicações, e um subscriber, que permanece atento a qualquer publicação em um dos tópicos selecionados.

Os programas foram codificados em python, por ser uma linguagem que incentiva um código mais limpo, de fácil leitura e também por ser uma linguagem com a qual o aluno está acostumado a usar.

O broker utilizado foi o mosquitto. Ele pode ser instalado em um sistema Linux pelo comando:

sudo apt-get install mosquitto

Também foi utilizada a biblioteca ***paho*** que fornece as funcionalidade do mqtt para o python. Informações de como a ***paho*** pode ser instalada em um ambiente virtual podem ser encontradas seguindo este link.

O Publisher

Ao executar o client-publisher, a primeira coisa que o programa irá fazer é solicitar o endereço IP da máquina onde o MQTT broker estiver rodando:

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

client = mqtt.Client()
loop = True
broker = input("IP do Broker: ")      # Aqui é solicitado o endereço IP da máquina rodando
                                      # o Broker
```

Logo após, o programa entra em um loop, que informa o usuário das opções possíveis de comando. Ao entrar com **bye**, o client-publisher irá enviar nos tópicos **temperatura** e **pressao** uma mensagem que fará com que os subscribers desliguem, e sai do loop.

Ao entrar com **temp**, o client-publisher irá simular a leitura de temperatura, enviando dez vezes uma mensagem para o tópico **temperatura** a cada cinco segundos.

Ao entrar com **pres**, o client-publisher irá simular a leitura de pressão atmosférica, enviando dez vezes uma mensagem para o tópico **pressao** a cada cinco segundos.

```
while(loop):
    print("Comandos:")
    print("bye - desliga os subscribers")
    print("temp - inicia sensoriamento de temperatura")
    print("pres - inicia sensoriamento de pressao")
    message = input("Entrada: ")

    if (message == "temp"): # se a opção escolhida é a de temperatura
        i = 10
        client.connect(broker,1883,60)
        while(i > 0):
```

```
            client.publish("topic/temperatura", "Temperatura: 25°C")
            time.sleep(5)
            i = i-1
        client.disconnect()

    if (message == "pres"): # se a opção escolhida é a de pressão
        i = 10
        client.connect(broker,1883,60)
        while(i > 0):
            client.publish("topic/pressao", "Pressão: 1 atm")
            time.sleep(5)
            i = i-1
        client.disconnect()

    elif (message == "bye"): # se a opção escolhida é de desligar os subscribers
        client.connect(broker,1883,60)
        client.publish("topic/temperatura", "bye")
        client.publish("topic/pressao", "bye")
        client.disconnect()
        loop = False
```

O comando que o client usa para a conexão é o **connect()**. No código, temos *client.connect(broker,1883,60)*, onde o primeiro parâmetro é o endereço IP de onde o broker está rodando, o segundo parâmetro é a porta do broker (no caso do mosquitto, a porta padrão é 1883) e o terceiro parâmetro é o *keepalive*, que representa o tempo de timeout da conexão do client, em segundos.

O Subscriber

Assim como o client-publisher, o client-subscriber inicia por perguntar o endereço IP da máquina onde está rodando o MQTT Broker, e mostrando os tópicos nos quais é possível fazer o subscribe. No caso, são usados os tópicos **temperatura**, **pressão** e a *wildcard* **#** que pode ser selecionada através da entrada **all**.

```
import paho.mqtt.client as mqtt

broker = input("IP do Broker: ")
print("Topics:")
print("temperatura")
print("pressao")
print("all")
message = input("Entrada: ")

def on_connect(client, userdata, flags, rc):  # o método on_connect lida com a conexão e
    print("Conectado! Código: "+str(rc))      # com o tópico em que ocorrera o subscribe
    print("Tópico Escolhido: "+message)
    if (message == "temperatura"):
        print("(topic/temperatura)")
        client.subscribe("topic/temperatura")
```

```
    elif (message == "pressao"):
        print("(topic/pressao)")
        client.subscribe("topic/pressao")

    elif (message == "all"):
        print("(topic/#)")
        client.subscribe("topic/#")

    else:
        print("Nenhum topic com esse nome encontrado!")
        print("Desconectando")
        client.disconnect()


def on_message(client, userdata, msg):      # o método on_message trata o que irá ocorrer
    print(msg.payload.decode())             # ao receber uma mensagem
    if msg.payload.decode() == "bye":
        print("Desconectando")
        client.disconnect()


client = mqtt.Client()
client.connect(broker,1883,60)

client.on_connect = on_connect
client.on_message = on_message

client.loop_forever()  # ao chamar loop_forever(), o programa continua executando e
                       # reconectando ao Broker indefinidamente
```

O programa irá, basicamente, se manter recebendo qualquer mensagem do tópico escolhido por tempo indefinido. Isso é realizado através do *client.loop_forever()*. A função *loop_forever()* garante que o client continue rodando e reconecte automaticamente com o MQTT Broker.

A única forma do programa parar é forçando interrupção pelo ambiente utilizado (fechando o terminal ou entrando com Ctrl+C no terminal Linux, por exemplo), ou quando o client-subscriber recebe a mensagem **bye** por um dos tópicos que está acompanhando.

Como Executar

Para tornar mais fácil a demonstração, sem a necessidade de instalar pacotes, o python, ou a criação de um ambiente virtual, foi utilizada a ferramenta Pyinstaller. Informações sobre a instalação e uso da ferramenta podem ser encontradas aqui.

No caso, foram gerados então dois executáveis, o client-publisher e o client-subscriber.

Eles só podem ser executados no Linux. Para criar executáveis para outros sistemas operacionais, é necessário usar o Pyinstaller no sistema operacional desejado (pode-se fazer uso de uma máquina virtual).

Os clientes são executados via terminal:

./client-publisher

./client-subscriber