Universidade Federal do Espírito Santo
Centro Tecnológico
Programa de Pós Graduação em Informática

Internet das Coisas

Prof. Vinícius Fernandes Soares Mota

Laboratório MQTT

Este laboratório tem como objetivo oferecer uma experiência prática com MQTT. Você realizará experimentos que permitirão que você aprenda a "publicar" e "subscrever" dados via MQTT. Para este fim, você usará:

1. Usar um broker local
2. Usar um broker público

O objetivo é aprender:

- Instalar e configurar o MQTT broker
- Troca de dados entre Clientes MQTT
- Usar o MQTT para controlar um dispositivo ESP8266

Adicionalmente, discutiremos padronização formatos e protocolos para envio de dados

Texto puro (e os problemas que isso traz)

Ultralightweigh

JSON

Hardware

Computador ou VM com Ubuntu 20.04 (ou debian based). Usaremos o ESP8266 Virtualmente.

Básico: Instalando o MQTT broker

Nestes experimentos usaremos **Mosquitto**. Mais detalhes em: https://mosquitto.org/download/

A maioria das distros Linux tem ele e pode ser instalado facilmente:

- Debian/UBUNTU/Raspian

```
sudo apt-get update

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

Gerenciando o broker

Para inicializar o parar use o comando:

```
sudo /etc/init.d/mosquitto start/stop
```

Se necessário, para evitar que inicialize automaticamente: sudo stop mosquitto

Por padrão, após inicialização pare o Broker e reinicialize com:

```
sudo mosquitto –v
```

note: “-v” == “verbose mode” e servirá para ver o que está acontecendo. Vamos deixar o broker rodando em um terminal a parte

Verificar se o broker está rodando:

```
sudo netstat -tanlp | grep 1883
```

note: “-tanlp” == tcp, all, numeric, listening, program

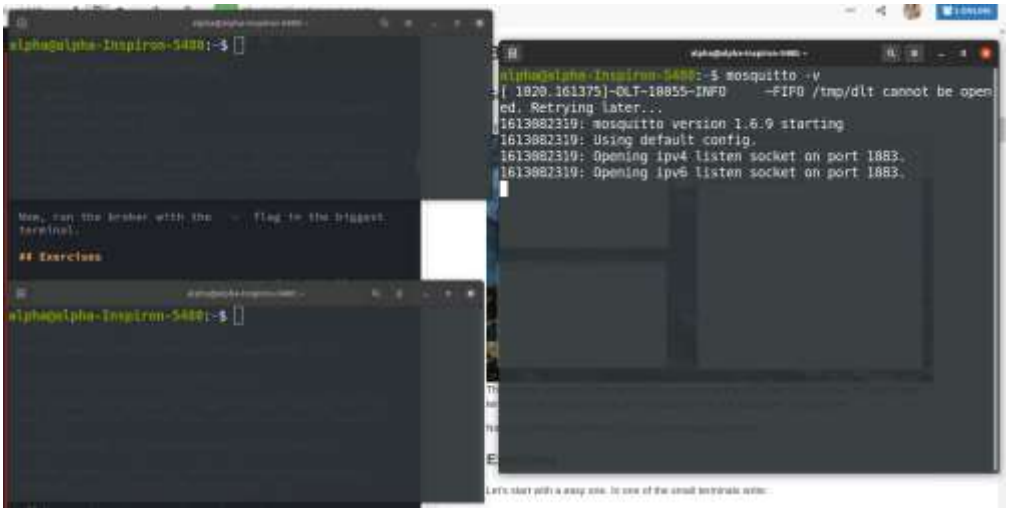
ou:

```
ps -ef | grep mosquitto
```

1: Um exemplo simples.

Set-up

Abrir 3 terminais no seu computador, como a figura abaixo:



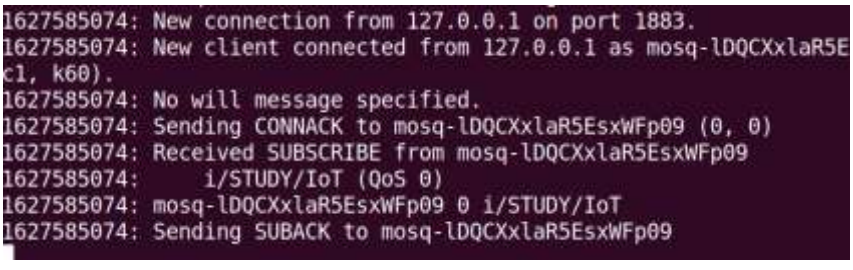
O maior terminal à direita será usado para ver a execução do Broker, os dois terminais menores serão usados para executar o publish e o Subscribe, respectivamente.

Exercícios

Em um dos terminais:

```
mosquitto_sub -t i/STUDY/IoT
```

O Broker deverá mostrar algo como a figura abaixo indicando que registrou a subscrição:



No outro terminal, publique uma mensagem usando o comando abaixo:

```
mosquitto_pub -t i/STUDY/IoT -m "Very well."
```

No broker terminal aparecerá algo como:

O terminal executando o mosquitto_sub deverá mostrar a mensagem recebida. Execute agora:

```
mosquitto_pub -t i/study/iot -m "Not so well"
```

Os tópicos são case-sensitive?

Um outra opção útil do mosquitto_pub é -l. Execute:

```
mosquitto_pub -t i/STUDY/IoT -l
```

Fez um canal unidirecional. Enviando todos os caracteres para o servidor.

QoS (Quality of Service):

A opção -q do mosquitto_pub permite definir a qualidade de serviço e qual o grau de confiança de entrega você deseja para entrega da mensagem.

```
mosquitto_pub -t i/STUDY/IoT -q 2 -m testing
```

Compare a saída usando q igual 0, 1 e 2 no terminal que está executando o broker.

Retained messages:

> Normalmente, se um publicador publica uma mensagem em um tópico e ninguém está inscrito nesse tópico, a mensagem é simplesmente descartada pelo broker. Se você quiser que seu corretor lembre a última mensagem publicada, você terá que

usar a opção reter. Apenas uma mensagem é retida por tópico. A próxima mensagem publicada nesse tópico substitui a mensagem retida para aquele tópico.

Para isto basta adicionar a flag -r

Abra o subscriber apenas após publicar a mensagem em cada uma das situações e diga o que observa ao abrir o subscriber:

1. Publicar uma mensagem sem a flag -r
2. Publicar uma mensagem com a flag (-r).
3. Publicar mensagens diferentes com a flag -r antes de abrir o subscriber .

Para remover uma mensagem retida, envie uma mensagem em branco.

Mas aqui é aula de redes.. Vamos ver o broker em detalhes

O Mosquitto permite utilizar a própria ideia do MQTT para analisar métricas de rede internas. Para isso, basta subscrever nos tópicos das métricas que deseja. Maiores detalhes em:

```
man mosquitto
```

Repita o exercício dos retained message acima mas agora você deverá monitorar:

1. Quantos bytes o broker recebeu;
2. Número de mensagens recebidas.
3. Número de mensagens enviadas;

Para cada um destes, você deverá abrir um terminal separado.

Estrutura de Tópicos (Wildcard)

O # é usado para corresponder a todos os níveis subsequentes de hierarquia. Com um tópico de "a / b / c / d", pode ser subscrito como:

- a/b/c/d
- #
- a/#
- a/b/#
- a/b/c/#
- +/b/c/#

Simulando uma casa inteligente

Considere uma casa inteligente com ar condicionado e lampadas inteligentes em todos os 3 ambientes desta casa. Modele uma estrututa de tópicos que permita voce ligar e desligar todos os equipamentos ou por tipo (lampada e ar) ou especifico. Emule esse ambiente usando o Mosquitto broker e um terminal para cada dispositivo.

Dica: o aplicativo chamado **Terminator** pode ser útil aqui :)

1. Qual a estrutura de tópicos utilizadas?
2. Como e quais dados são enviados;
3. Qual a quantidade de byte enviado por sua aplicação?

Brokers Públicos

Existem diversos brokers públicos chamados sandbox para testar MQTT

- `test.mosquitto.org`
    - http://test.mosquitto.org/
- `iot.eclipse.org`
    - : https://iot.eclipse.org/getting-started#sandboxes
- `broker.hivemq.com`
    - http://www.hivemq.com/
        - http://www.mqtt-dashboard.com/ <-- Vamos usar este:

Todos acessam pela porta `1883`.

Faça o exercício da casa inteligente mas agora use o broker público. Tente notar as diferenças.

2: Show me the code - Ligando e desligando o LED ESP8266 via MQTT

Existem blibliotecas MQTT para qualquer linguage. No geral, funcionam como conectar a um broker, subscrever ou publicar. Neste exercício, vamos utilizar a biblioteca mqtt para Arduino.

Vamos adaptar o código em https://gitlab.com/vmota/ensino-iot-ufes/-/snippets/1856875 para que acenda o LED interno de um ESP8266.

Devemos então, substituir as informações de redes sem fio e coordenar os tópicos como feito no exercício anterior.

3: Show me the code - Temperatura, pressão e umidade via ESP8266 via MQTT

Agora considere que o ESP8266 possui um sensor de temperatura, pressão e umidade. Sua missão é definir a estrutura de tópicos para obter a informação de um sensor específico e com qual o padrão protocolo os dados serão enviados.

Existem diversos padrões como exemplificado abaixo. Fica a escolha do desenvolver utilizar o modo com mais verbosidade para dar semântica aos campos, útil em aplicações que compartilham dados e facilita o entendimento para quem recebe os dados, por outro lado, gastam mais bytes de transmissão.

Exemplos Formatos típicos:

CSV: temperatura, umidade, pressao ou com verbosidade: temp, valor, umi, valor, pressao, valor

Ultralight: Similar ao CSV mas usa pipe '|' como separador: temp|umidade|pressao

JSON: Formato que permite hierarquia entre os membros (https://json.org/)

Exemplo abaixo utiliza verbosidade

```
{
  ″sensor″: {
        “temp” : valor,
        “umidade” : valor,
        “pressao” : valor,
  }
}
```

XML. Just Don't use for IoT app, please....

Considerações finais

Concluindo, MQTT é uma ferramenta poderosa para envio de dados para múltiplos sensores em IoT. Segurança não foi discutida neste laboratório. O MQTT permite autenticação por tópico, dessa forma apenas os que possuem a senha para um tópico poderiam receber as mensagens.

Por se tratar de um protocolo de fila, ele precisa ser eficiente em memória. Nosso próximo trabalho consiste em fazer uma análise de carga para o MQTT.

Este laboratório foi inspirado e adaptado no contexto de IoT de https://hackmd.io/@ramonfontes/mqtt_basics#