Broker server + user application

Andrew Martins, Daniel Saldanha, Davi Simite, Hilton Costa, Laísa Paiva, Letícia Balbi e Ligia Marcia

Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Otávio, 6200 – Coroado I – CEP: 69077-000 – Manaus – AM – Brasil

1. Introdução

Atualmente, o MQTT (*Message Queuing Telemetry Transport*) é o mais usado no ramo de IoT. Esse protocolo foi desenvolvido pela IBM com o objetivo inicial de auxiliar na comunicação entre sensores e satélites. Entre suas vantagens podemos citar: ser um protocolo leve, o que permite a implementação em redes com pouca largura de banda e alta latência, e ser um protocolo que se adequa à maioria dos dispositivos e serviços de IoT. Além disso, o MQTT implementa um sistema específico de conexão, o modelo publicação/assinatura (*publisher/subscriber*).

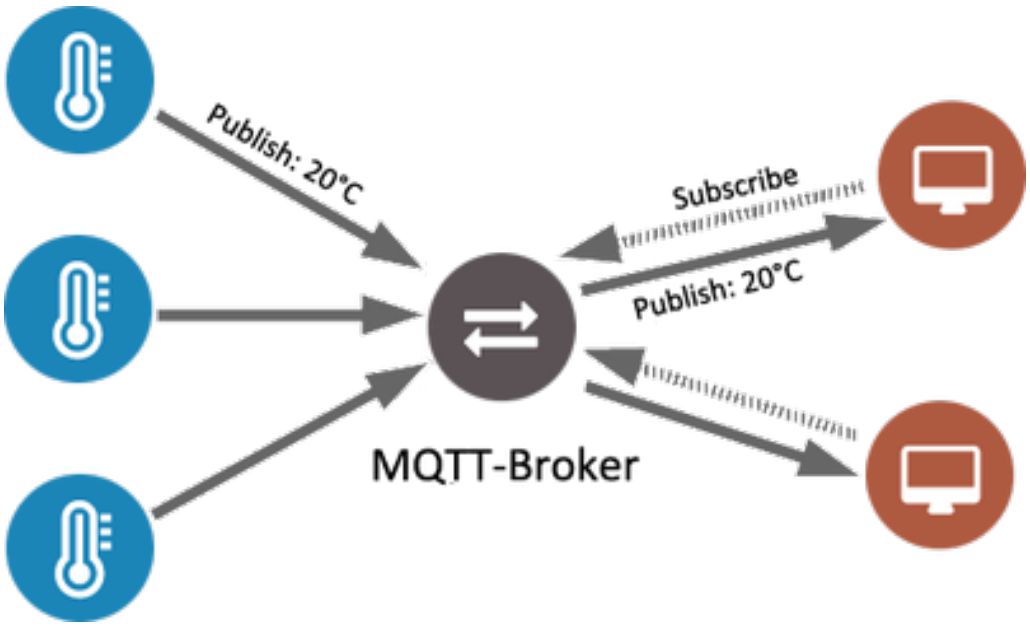

Figura 1. Modelo Publisher/Subscriber.

Este modelo tem 3 elementos principais:

- **Publicadores (*Publisher*)**: elementos que enviam mensagens para o MQTT *broker* em um determinado tópico.
- **Assinantes (*Subscriber*)**: elementos que estão dispostos a receber essas mensagens especificadas pelo tópico de interesse.
- ***Broker***: elemento responsável por gerenciar as publicações e assinaturas. Recebe e repassa mensagens de um publicador para todos os assinantes do tópico da mensagem enviada.

Neste trabalho usamos um computador Linux como MQTT *broker*, um sensor de PH junto com um Raspberry Pi como Publicador e uma aplicação web como Assinante.

2. Objetivo

A equipe ficou responsável pela criação e configuração do *broker* e a implementação de uma interface de usuário. As funções básicas da implementação são: efetuar a leitura dos dados de um sensor de PH através do protocolo MQTT e ligar e desligar um LED para representar um atuador em Iot.

3. Materiais

- Computador (MQTT *broker* e servidor Web)
- Raspberry Pi (Publicador)
- Sensor de ph (Gerador de dados)
- Computador, celular, tablet e etc. (Usuário da aplicação)

4. Experimento

Para o desenvolvimento do experimento foram implementados:

- **MQTT *broker***: um servidor para manipular as requisições do servidor web (*subscriber*) e do Raspberry Pi (*publisher*);
- **Banco de Dados**: foi criado um banco de dados em PostgreSQL para controle de dados gerados pelo publisher do MQTT *broker*;
- **Aplicação para usuário**: foi implementada uma aplicação web para usuário usando as ferramentas Node.js e ReactJs para criar um gráfico de leituras do sensor de PH e manipular um LED.

4.1. Cenário

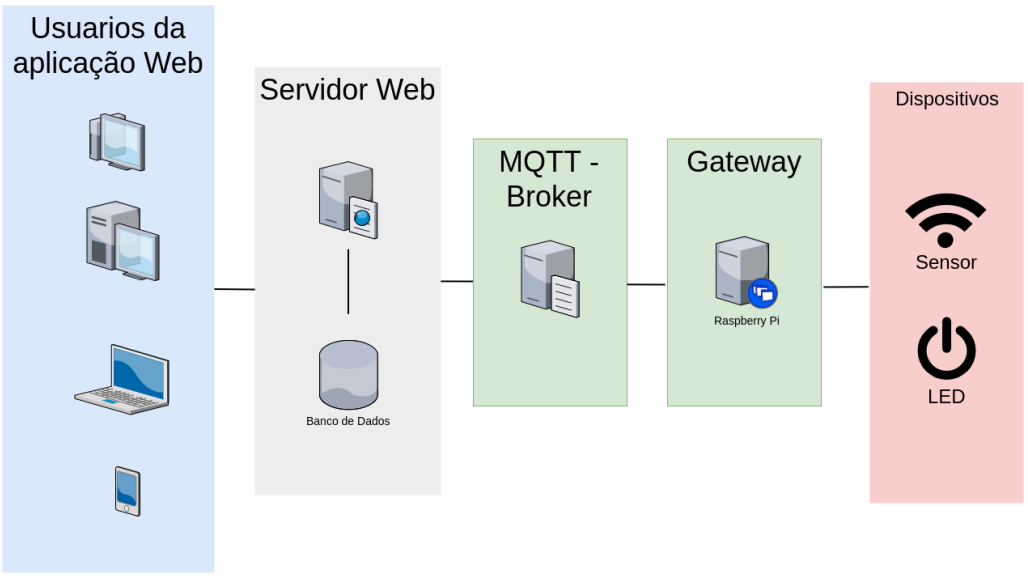

Figura 2. Etapas da comunicação de todo o sistema.

O sistema todo consiste na comunicação entre 5 sistemas diferentes:

4.1.1. Dispositivos

É a parte que representa os dispositivos que compõem a Iot. O sensor representa todos os aparelhos que geram dados e enviam-os na rede. Já o led simboliza os atuadores que exercem alguma função dependendo de como são projetados. Neste exemplo há apenas a ação de ligar e de desligar um led, mas poderia ser desligar/ligar equipamentos industriais, alternar estados de máquinas de construção automatizada, recolher informações de sensores etc.

4.1.2. *Gateway*

No experimento, sua função é executada por um Raspberry Pi. Ele tem a responsabilidade de receber todas as mensagens vindas dos dispositivos e mandar ao MQTT *broker* para serem tratadas pelo servidor web. Sua principal tarefa é executar esse trabalho sem perdas. Desse modo, é imprescindível o uso de um protocolo para o tratamento de corrupção de dados, demora de chegada da mensagem, entre outros fatores.

4.1.3. MQTT *broker*

Através da instalação do Eclipse Mosquitto foi criado o MQTT *broker* em um dos computadores do Laboratório de Redes (GRCM). Sempre que o computador é iniciado o programa do MQTT *broker* é executado e torna o computador disponível para novas conexões. Utilizamos a biblioteca **paho-mqtt 1.5.0** do python para criar clientes MQTT e estabelecer conexões entre cliente e servidor MQTT.

A classe do cliente paho mqtt possui vários métodos. Os principais são:

```
mqtt.Client(client_name)
```

Para criar uma instância de usuário utilizou-se esse comando atribuído a sua saída por uma variável chamada client. Para poder publicar e se inscrever a fim de receber mensagens é necessário se conectar ao *broker*. Dessa forma, usa-se o método *connect* do cliente mqtt do Python. O método pode ser chamado com até 4 parâmetros, mas na sintaxe geral é preciso somente fornecer o nome/endereço IP do *broker*.

```
connect(host,port=1883,keepalive=60,bind_address="")
```

Função da classe mqtt.client necessária para se conectar ao MQTT *broker*. Após a sua execução é possível publicar mensagens em algum tópico.

```
subscribe(topic, qos=0)
```

Essa função inscreve o computador a um tópico (topic) o qual ele quer receber mensagens. Durante o cadastro é possível definir a qualidade de serviço (QoS-quality of service) que a sua conexão requer com o broker. O QoS é 0 por padrão, mas pode variar entre 0 (00),1 (01) e 2 (10).

- **00 garante no máximo uma entrega** : trata-se de uma entrega de melhor esforço, visto que não há garantia sobre a verdadeira recepção da mensagem;
- **01 garante pelo menos uma entrega**: envia várias vezes uma mensagem até que uma chegue no broker;
- **10 garante exatamente uma entrega**: a recepção da mensagem é única.

```
publish(topic, payload=None, qos=0, retain=False)
```

Utiliza-se o método publish para enviar uma mensagem em algum tópico especificado no parâmetro. Os parâmetros obrigatórios são topic e payload. Payload é a mensagem que se deseja publicar, topic é o tópico em questão no qual a mensagem será publicada. Os tópicos utilizados para estabelecer conexão entre subscriber/publisher foram:

1. **Arduino/led**: tópico usado para toda a comunicação relacionada ao led. Foi definido com a equipe 2 que os valores de payload para a mensagem será "ON" e "OFF";
2. **Arduino/sensor**: tópico usado para o envio de mensagens do *gateway* para o broker, as quais contêm os valores coletados do sensor.

```
on_message(client, userdata, message)
```

Após as configurações realizadas anteriormente, já é possível assinar um tópico e enviar uma publicação. Para visualizar as mensagens é preciso ativar e processar o retorno de chamada *on_message*, como mostrado no exemplo a seguir. A função *on_message* permite processar qualquer mensagem.

Para que as mensagens sejam identificadas, o cliente precisa escutar o canal. Logo é necessário iniciar uma thread (segundo a biblioteca utilizada) para monitorar o canal e caso identifique alguma mensagem, o programa irá requerer uma interrupção do sistema operacional e irá executar a função "*on_message*".

O relacionamento final entre o servidor broker e os inscritos ficou da seguinte forma:

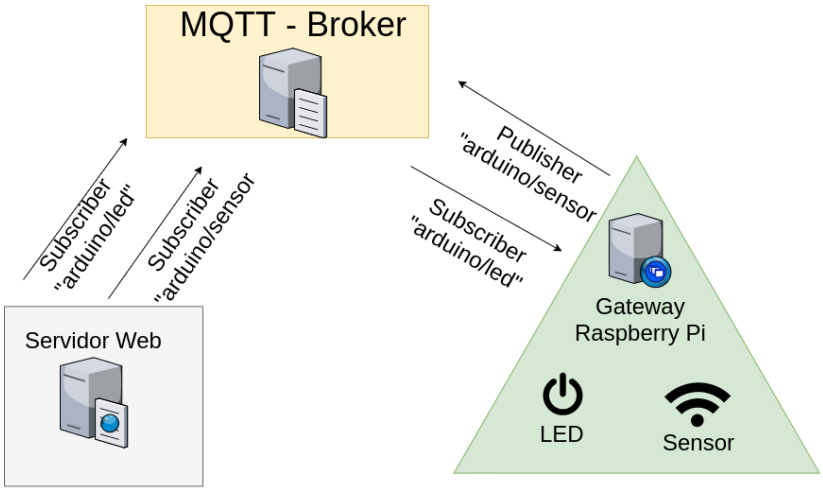

Figura 3. Relacionamento broker e clientes.

4.1.4. Servidor Web

O servidor web é um inscrito no MQTT *broker* no tópico '**arduino/sensor**'. Com o banco de dados alimentado, uma API foi desenvolvida em Node.js. Suas funções são:

1. **Povoar o banco de dados**: as mensagens enviadas neste tópico são números reais gerados pelo sensor e encaminhados pelo gateway. Através desses valores é alimentado um banco de dados relacional que segue o seguinte esquema:

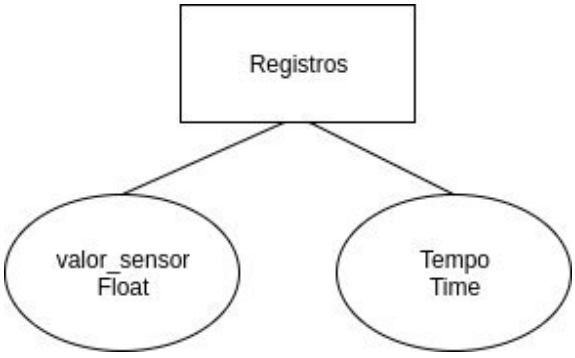

Figura 4. Banco de dados relacional.

   para executar consultas SQL e retornar uma lista de tuplas contendo o valor do sensor e o tempo no qual foi coletado.
2. **Gerar comando para led**: por meio da entrada fornecida por usuários o servidor deverá ser capaz de receber e enviar o comando para o gateway seguindo o protocolo MQTT e contendo o tópico "Arduíno/Led" conforme foi definido na convenção das equipes 3 e 2. Através da mensagem enviada pelo Raspberry este fará a comunicação segura com arduíno utilizando a comunicação LoRa.

3. **Criar interface de usuário**: através da conexão com o usuário através do endereço principal da API (**http://10.208.3.226:3000/completo**), o servidor fará uma consulta ao bancos de dados e exibirá um gráfico contendo os dados retornados, os quais foram povoados com as mensagens coletadas pelo sensor e o horário da captura. Além dos dados, o usuário terá a opção de ligar e desligar o led pela interface sem precisar saber utilizar todo o processo de comunicação por trás da interface. Optamos por entregar os dados em forma de gráfico onde o eixo X é referente ao tempo de coleta do dado e o eixo Y o valor coletado do sensor mostrado na figura abaixo:

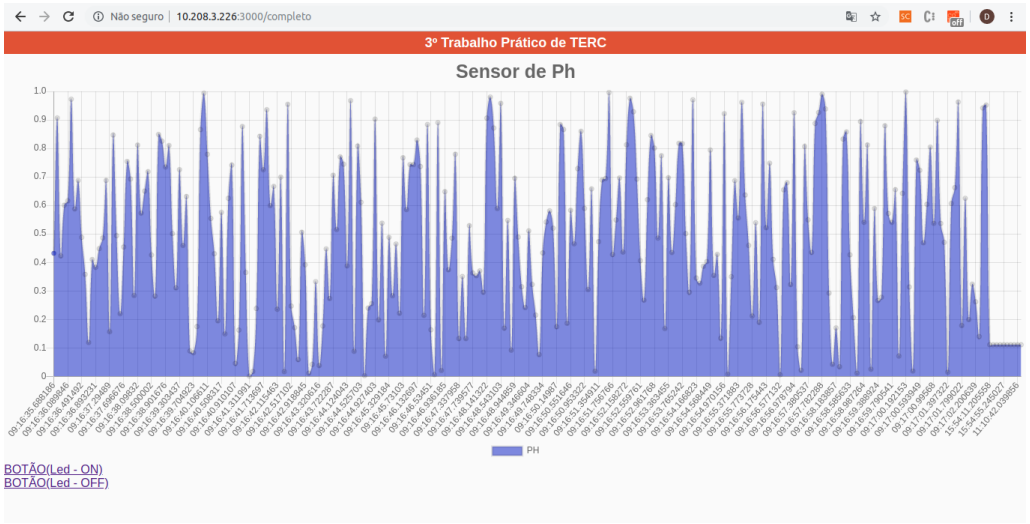

Figura 5. Gráfico da aplicação para usuário.

5. Dificuldades

Falta de experiência com programação Web: devido todos os integrantes do grupo não saberem sobre programação web, o desenvolvimento do trabalho foi afetado. Desse modo, a implementação do código levou muito tempo e esforço dos integrantes para conseguir utilizar os frameworks Node.js e ReactJs e gerar um servidor funcional para a aplicação.

6. Referências

https://pypi.org/project/paho-mqtt/

https://mosquitto.org/

https://medium.com/@onur.dundar1/mqtt-part-i-understanding-mqtt-aade455baec9

https://appcodelabs.com/introduction-to-iot-build-an-mqtt-server-using-raspberry-pi

http://www.steves-internet-guide.com/into-mqtt-python-client/