MAC0352 - Redes de Computadores e Sistemas Distribuídos (2022)

EP01 - Criação de um Broker MQTT

Lucas Moretto da Silva
NUSP: 9778602

Protocolo MQTT

Protocolo MQTT

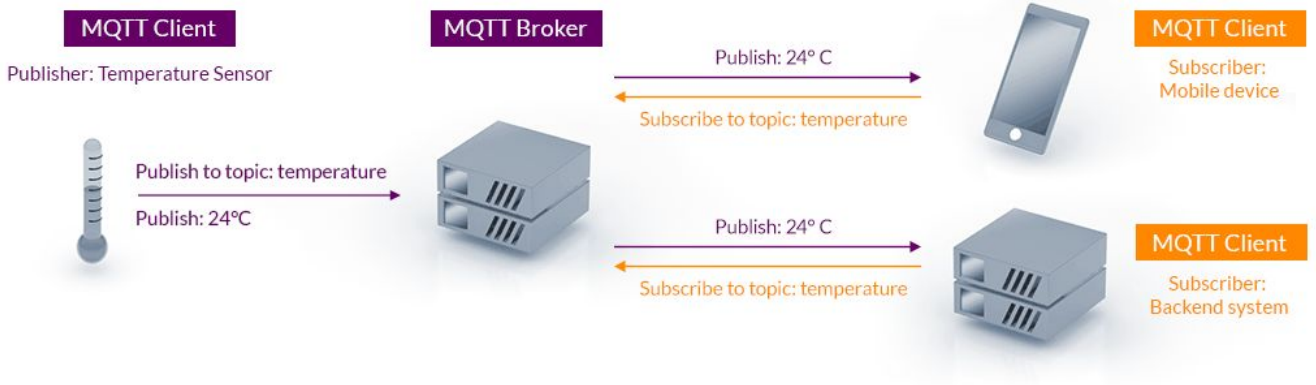
O MQTT (do inglês Message Queue Telemetry Transport) é um protocolo de mensagens projetado para baixo consumo de banda de rede e recursos de hardware, desenvolvido pela IBM e Eurotech na década de 90.

Ele foi projetado para estabelecer um forma de transporte de mensagens no padrão publisher/subscriber. Seus principais objetivos de desenvolvimento são minimizar o uso de banda de rede e uso de recursos dos equipamentos enquanto garantindo confiabilidade e algum nível de garantia de entrega das mensagens.

O MQTT hoje é usado em uma ampla variedade de indústrias, como automotiva, manufatura, telecomunicações, petróleo, gás, etc.

Protocolo MQTT

A imagem abaixo exemplifica um caso de uso em que o protocolo poderia ser utilizado. No exemplo, temos um cliente produtor (publisher) que envia dados de temperatura coletados por um sensor à um tópico no broker MQTT. Enquanto 2 consumidores (subscribers) desse tópico consomem as mensagens quando elas estiverem disponíveis. Cada consumidor processará os dados a sua maneira.



Fonte: https://mqtt.org/

Implementação

Implementação do Broker MQTT

- Toda a execução do broker é realizada no arquivo ***fonte/src/mqtt.c***
- Existem constantes armazenadas em ***fonte/includes/constants.h*** que são utilizadas por todo o código
- Cada tópico criado no broker é representado por uma pasta *<TMP_DIR>/<TOPIC_NAME>*
  - *TMP_DIR pode ser modificada em* ***fonte/includes/constants.h***

Implementação do broker MQTT

No arquivo fonte/src/message.h foi definido um struct para representar as mensagens enviadas ao broker. A estrutura segue a definida na documentação do protocolo.

```
12    typedef struct {
13        int packetType;
14        int flags;
15        int remainingSize;
16        char * remaining;
17    } Message;
```

Figure 2.2 - Fixed header format

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2…	Remaining Length							

Fonte: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html#_Toc442180832

Implementação dos comandos MQTT

Os comandos implementados no projeto estão listados abaixo. Para visualizar suas implementações acesse o arquivo ***fonte/src/mqtt.c***, método ***readMessage*** (linha 113).

- CONNECT
- PINGREQ
- PUBLISH
- SUBSCRIBE
- DISCONNECT

Comando CONNECT

Quando um cliente tenta realizar uma conexão ao servidor (enviando o código CONNECT no packet), então o broker simplesmente responde com CONNACK.

```
switch (message.packetType) {
    case CONNECT:
        response.packetType = CONNACK;
        response.remainingSize = 2;
        response.remaining = malloc( size: 2 * sizeof(char));
        bzero( s: response.remaining,  n: 2);
        break;
```

Comando PINGREQ

Quando um cliente realiza uma requisição de PINGREQ, então o servidor (broker) simplesmente responde com PINGRESP.

```
case PINGREQ:
    response.packetType = PINGRESP;
    break;
```

Comando SUBSCRIBE

Para cada subscriber o broker cria um arquivo de pipe com o padrão *<TMP_DIR>/<TOPIC_NAME>/<PID>*. O processamento das mensagens é então divido em duas threads para facilitar a sincronia do processo:

- Uma thread responsável por continuar a troca de mensagens com o subscriber (retorno de pings, retorno do SUBACK, retorno do DISCONNECT).
- Uma thread responsável por ouvir mensagens escritas ao tópico e propagar aos subscribers.

Comando PUBLISH

O broker identifica o tópico vindo na mensagem, buscando por *<TMP_DIR>/<TOPIC_NAME>,* e escreve a conteúdo da mensagem em todos os arquivos de pipe desse diretório.

```
case PUBLISH:;
    Topic topicPub = getTopic(message);
    handlePublish( topic: topicPub,  packet: messageToPacket(message));
    response.packetType = PUBACK;
    break;
```

Comando DISCONNECT

Ao receber uma requisição de DISCONNECT, optou-se por apenas retornar uma resposta com o código RESERVED (15). O tópico continuará existindo (caso o cliente seja um subscriber), e nenhum outro efeito será propagado pelo broker.

```
case DISCONNECT:;
    response.packetType = RESERVED;
    break;
```

Experimentos

Ambiente de testes

- Computador Acer Predator Helios 300
- 32 GB RAM
- 2 GHZ
- 8 núcleos
- Ubuntu 18.04 LTS
- Chip Realtek PCIe GBE

Experimentos

- O broker implementado foi executado dentro de um contêiner do Docker.
- A porta 1883 do contêiner foi vinculada a porta 1883 do host.
- Os clientes do mosquitto foram executados no host.
- Com o broker executando dentro de um contêiner do Docker, pode-se obter as métricas de CPU (CPUPerc) e Rede (Net I/O) de forma mais simples através do comando docker stats.

Cenários de teste

1. Broker rodando sem nenhum cliente conectado.
2. Broker rodando com 100 clientes subscribers recebendo mensagens e 100 clientes publishers enviando mensagens.
3. Broker rodando com 1000 clientes subscribers recebendo mensagens e 1000 clientes publishers enviando mensagens.
    a. Durante os experimentos com grande quantidade de subscribers e publishers, notou-se que muitos dos clientes executando sofriam com starvation e suas conexões eram encerradas.
    b. Para contornar o problema considerou-se que apenas 1 cliente irá publicar por vez, de forma blocante, em 1 tópico.

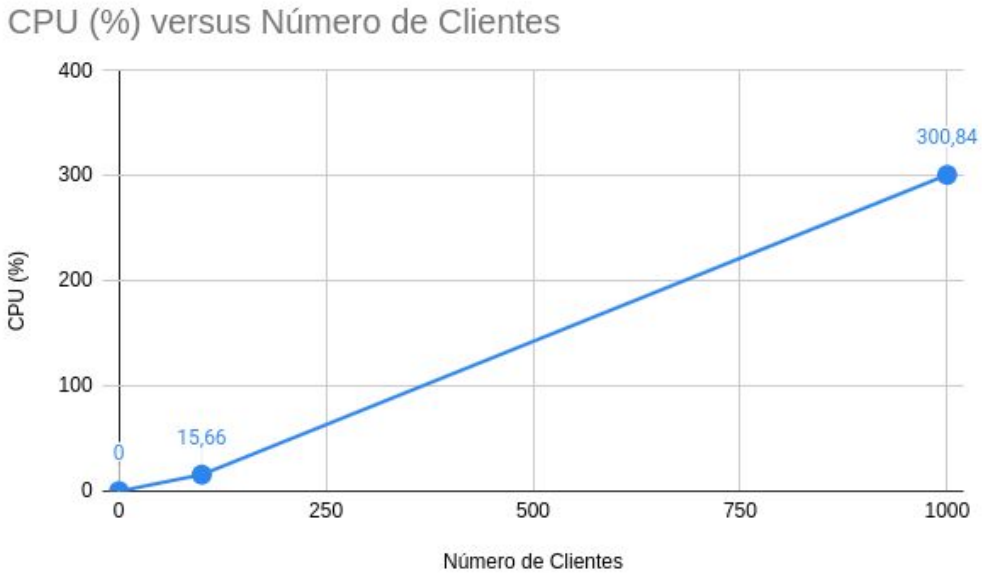
Execução dos testes

Foram criados alguns scripts em bash para executar os testes nos cenários descritos. Podemos resumir seu funcionamento da seguinte forma:

Para cada cenário descrito:

1. Cria-se um contêiner com o broker MQTT.
2. Executa-se o script *startup-clients.sh* para rodar os clientes do mosquito no host.
3. Coleta-se as métricas de CPU e Rede do contêiner através do método ***read_metrics*** em run-tests.sh.
4. Após os testes serem finalizados, o ambiente do contêiner é finalizado e deletado.

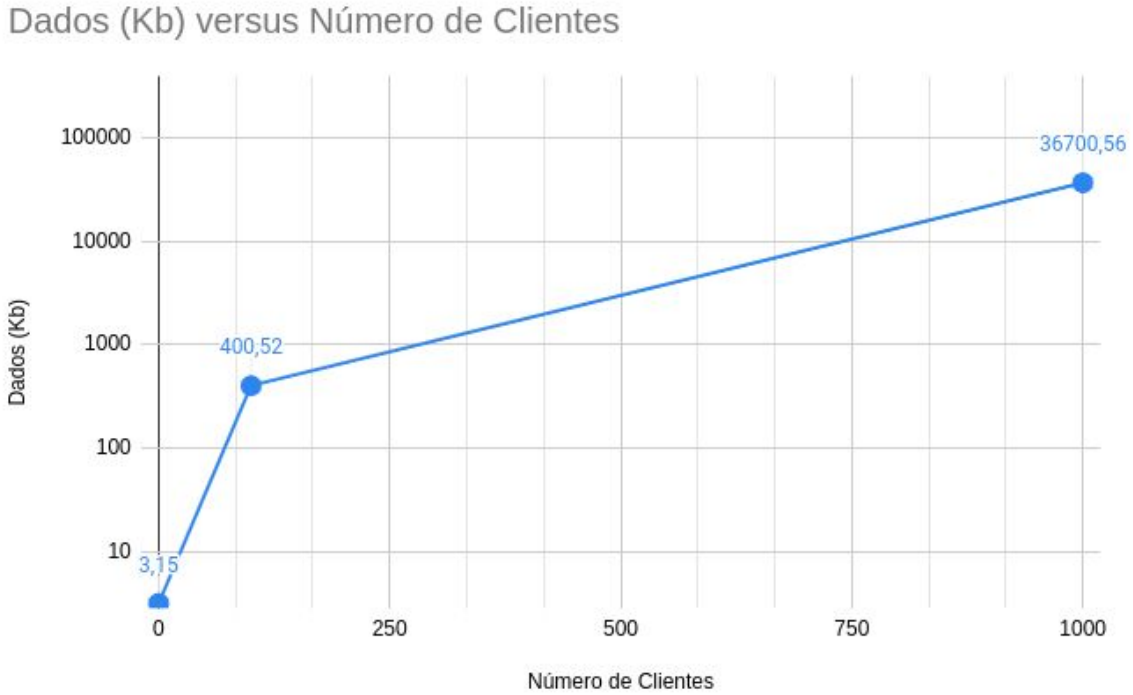
Resultados para uso de CPU



Intervalos de confiança:

[0; 0]
[0; 33.83]
[172.58; 398.85]

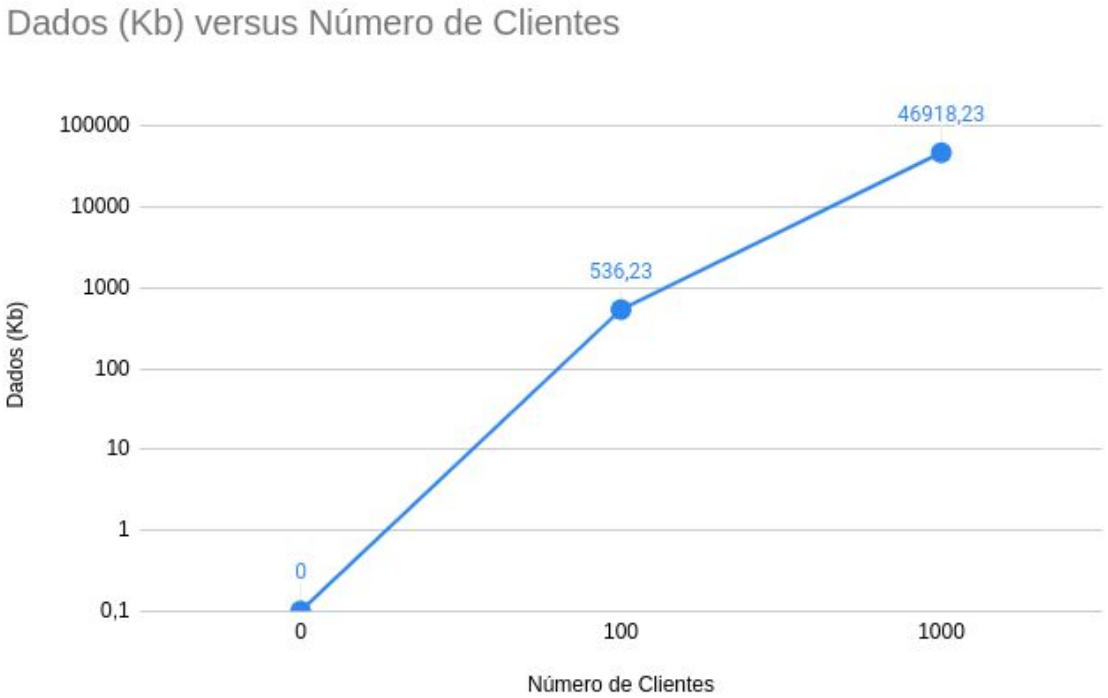
Resultados para recebimento de pacotes de rede



Intervalos de confiança:

[0.75; 4.53]
[0; 1037.65]
[25865.62; 46959.56]

Resultados para envio de pacotes de rede



Intervalos de confiança:

[0; 0]
[0; 1317.56]
[32760.56; 63526.23]

Resultados

- O aumento da CPU teve uma tendência linear com relação ao aumento do número de clientes.
- Interessante notar as diferenças entre dados recebidos e enviados pela rede.
    - O broker MQTT é responsável por fazer o broadcast das mensagens para todos os consumidores de um tópico. Por conta disso, a quantidade de dados enviados é muito maior do que os dados recebidos.

Referências

- http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
- http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/#:~:text=The%20MQTT%20packet%20or%20message,)%20%2B%20Variable%20Header%20%2DExample%20PUBACK
- https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/