Implementando um broker MQTT

Renato Matsuda Vassão Dias

MQTT 3.1.1*

- Protocolo Cliente Servidor de transporte de mensagens
- Leve, aberto e simples
- Ideal para M2M e IoT

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

POC (Proof of concept)

- Apenas QoS 0 (Fire and forget)
- Sem autenticação
- Sem tratamento de falhas de conexão

Mensagens MQTT: Control Packet

- Fixed Header
- Variable Header
- Payload

Tipos de pacote implementados

- CONNECT
- CONNACK
- PUBLISH
- SUBSCRIBE
- SUBACK
- PINGREQ
- PINGRESP
- DISCONNECT

Todas as propriedades apresentadas foram decodificadas e estruturadas para identificação das propriedades do pacote

CONNECT

- Variable Header
  - Protocol Name (= MQTT)
  - Protocol Level (= 3.1.1)
  - Connect Flags
  - Keep Alive
- Payload
  - Client Id

CONNACK

- Variable Header
  - Session Present
  - Connect Return Code
- Payload
  - <empty>

PUBLISH

- Variable Header
  - Topic Name
- Payload
  - <Mensagem da aplicação>

SUBSCRIBE

- Variable Header
  - Packet Identifier
- Payload
  - Topic Filter
  - Requested QoS

SUBACK

- Variable Header
  - Packet Identifier
- Payload
  - Return Code

PINGREQ

- Variable Header
  - <empty>
- Payload
  - <empty>

PINGRESP

- Variable Header
  - <empty>
- Payload
  - <empty>

DISCONNECT

- Variable Header
  - <empty>
- Payload
  - <empty>

Arquitetura

- Para cada cliente são criados 2 forks do processo pai, que escuta novas conexões
- 1: Escuta novas mensagens do cliente e responde de acordo com o protocolo
- 2: Escuta uma fila e roteia a mensagem para o cliente

Arquitetura

- O nome da fila é salvo em um arquivo com o nome do tópico, e quando chega uma mensagem PUBLISH, o processo responsável por processar essa mensagem itera sobre as filas e manda as mensagens para os processos responsáveis pela comunicação com clientes inscritos no tópico

Testes

Os testes coletados foram realizados em um Arch Linux 5.14.7, utilizando um Ryzen 5 3600 3.6GHz (6 cores, 12 threads) com 16GB de Ram 2400Mhz, e SSD.

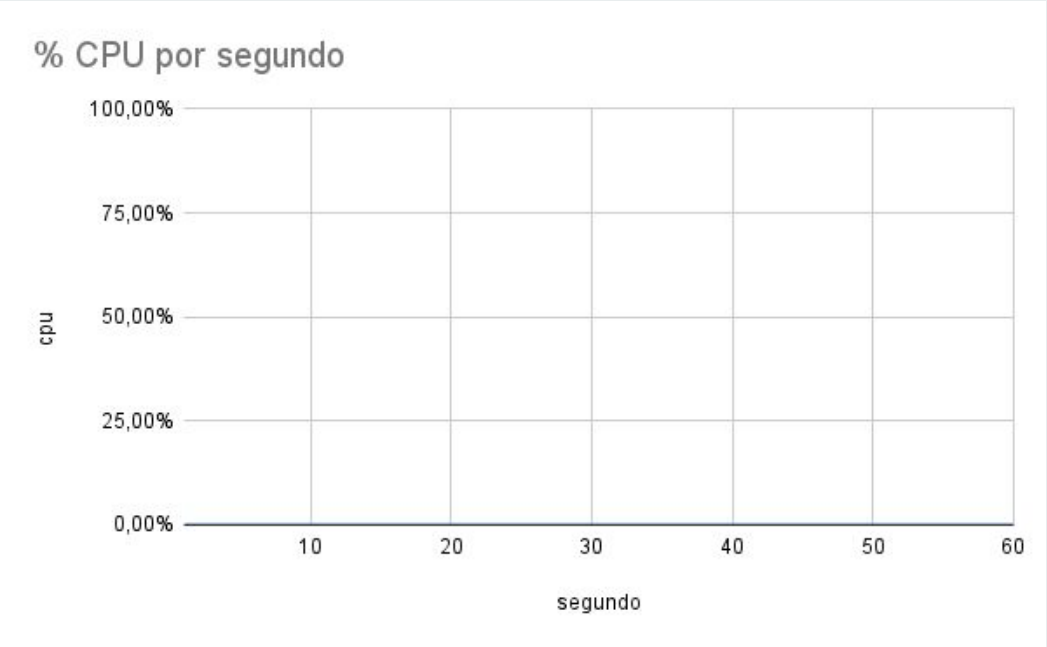
O uso de CPU foi feito utilizando a ferramenta top* somando o uso de todos os processos filhos com o processo pai; e o uso de rede foi feito utilizando a ferramenta Wireshark**

Em cada teste foi randomizado a mensagem enviada e a quantidade de tópicos (no caso de 100 clientes), onde cada cliente que publica manda 1 pacote por segundo, durante 1 minuto

* https://man7.org/linux/man-pages/man1/top.1.html
**https://www.wireshark.org/

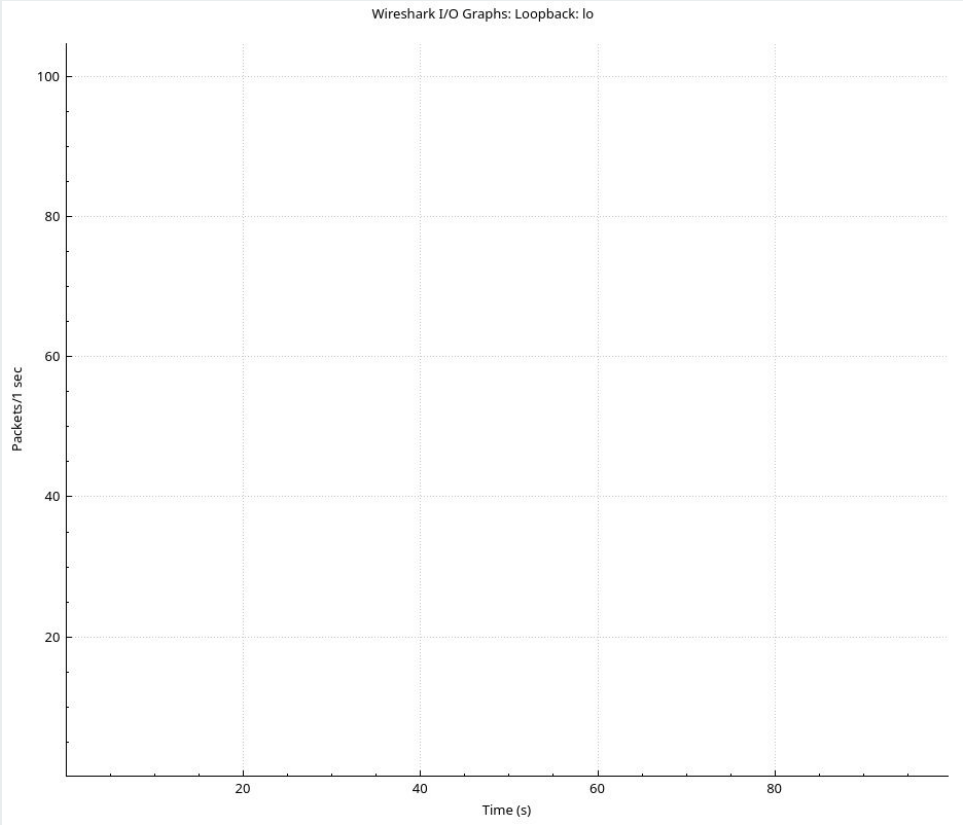
Teste 1: Broker sem conexões

Note que caso o servidor não tem que lidar com nenhuma conexão, seu uso de CPU não é alterado



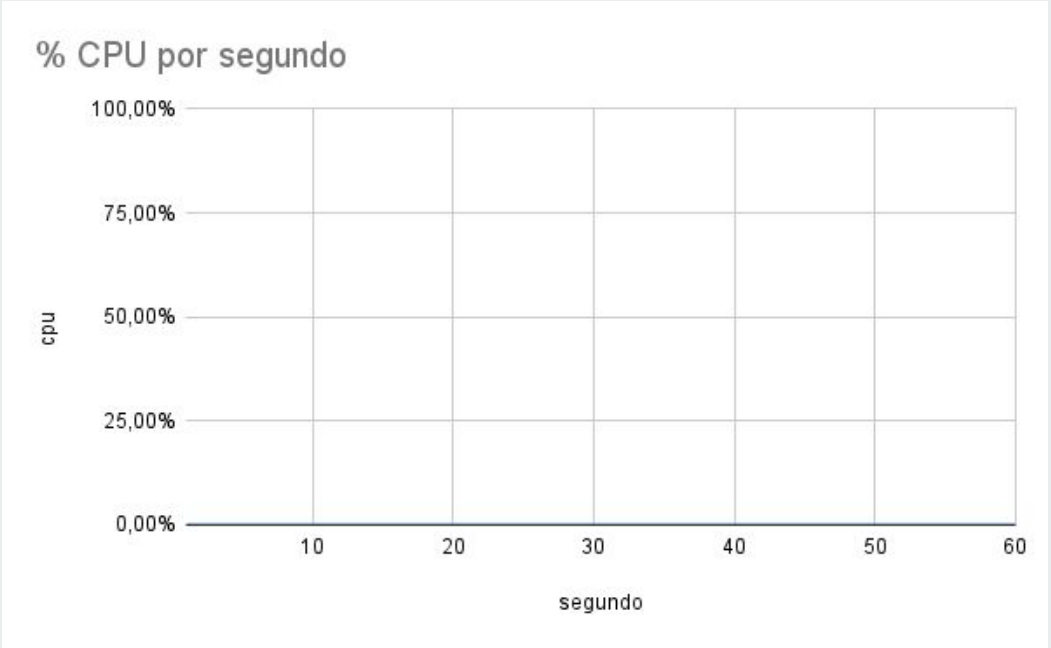
Teste 1: Broker sem conexões

No caso de pacotes MQTT, não houve nenhuma troca, o que era esperado



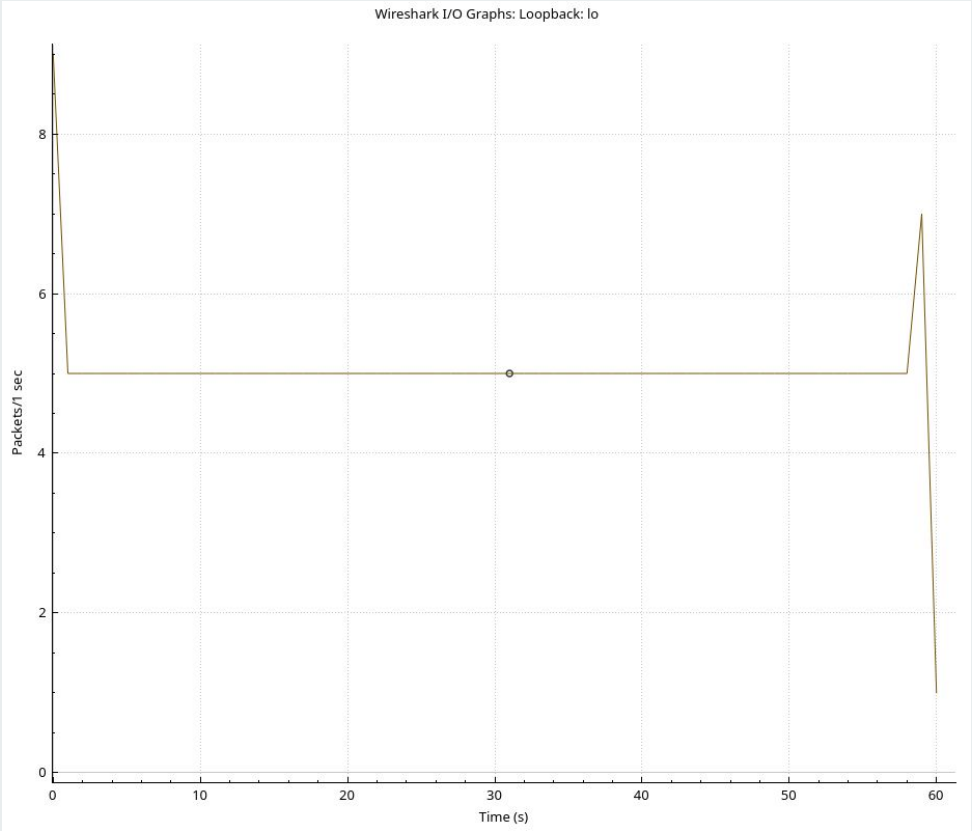
Teste 2: Broker com 2 conexões: 1 pub e 1 sub

Nesse caso, também temos que o uso de CPU continuou próximo de 0%



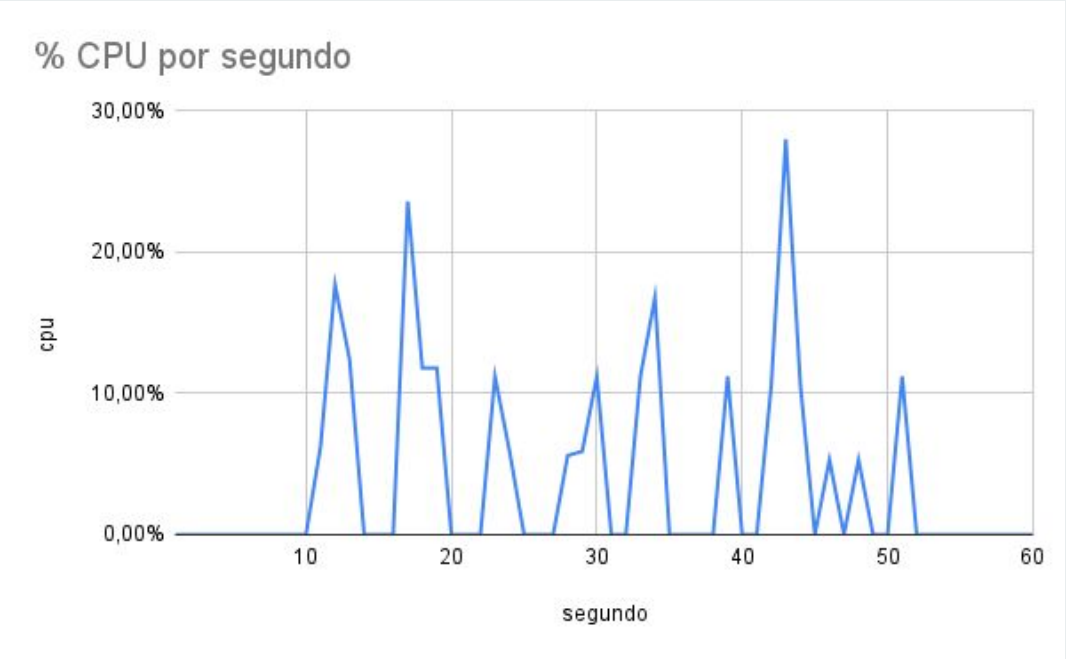
Teste 2: Broker com 2 conexões

Note que no começo dos testes caso temos um aumento na troca de pacotes MQTT, sendo estes pacotes de SUBSCRIBE e PUBLISH, e ao longo do tempo uma troca de 5 pacotes por segundo, o que faz sentido uma vez que temos apenas 1 cliente publicando e 1 cliente inscrito. No final temos dois pacote de DISCONNECT publicados, o que aumenta a quantidade de pacotes enviados.



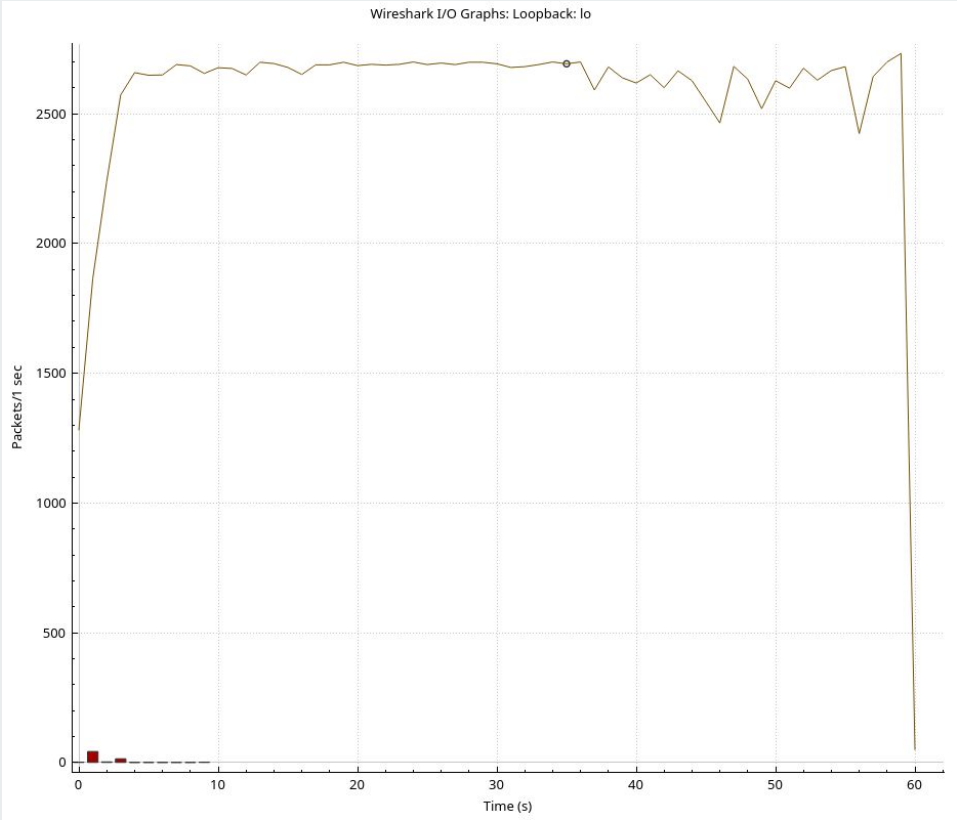
Teste 3: Broker com 100 conexões: 50 pub e 50 subs

Aqui temos alguns picos de uso de CPU, uma vez que o servidor teve que processar diversas mensagens em paralelo



Teste 3: Broker com 100 conexões: 50 pub e 50 subs

Nesse caso, temos um grande aumento de troca de pacotes entre o servidor e os clientes. Praticamente acima de 2500 por segundo, o que é esperado devido a alta quantidade de clientes conectados.



Conclusão

- MQTT é um protocolo simples e leve
- Demanda pouco uso de CPU e rede
- O servidor poderia ter um aumento de performance caso não utilizasse dois processos para cada cliente