EP1 - MAC0352
REDES DE COMPUTADORES

INTEGRANTE / NUSP:

WILLIAN HIROSHI TAKIHI | 11221755

ROTEIRO

- Comandos Broker MQTT;
- Implementação;
  - Pacotes;
  - Comunicação entre processos;
- Decisões de Projeto;
- Testes;
- Gráficos;
- Resultados e Conclusão.

Comandos Broker MQTT

- O **Broker** é executado pelo seguinte comando:
  - ./ep1-broker <porta_de_escuta_de_conexões>
- Conexão de **Subscritor**:
  - mosquitto_sub -V 311 -h <ip_do_broker> -p <porta_de_escuta_do_broker> -t <"nome_do_topico">
  - Exemplo: mosquitto_sub -V 311 -h localhost -p 8000 -t "topic"
- Conexão de **Publicador**:
  - mosquitto_pub -V 311 -h <ip_do_broker> -p <porta_de_escuta_do_broker> -t <"nome_do_topico"> -m "mensagem_publicada"
  - Exemplo: mosquitto_pub -V 311 -h localhost -p 8000 -t "topic" -m "message"

Comandos Broker MQTT - Flags

- As flags utilizadas nos comandos correspondem a:
  - -V: versão do protocolo MQTT utilizado;
  - -h: qual host o cliente deve se conectar;
  - -p: qual porta se conectar;
  - -t: tópico MQTT a se subscrever/publicar;
  - -m: mensagem a ser escrita no tópico.

Implementação - Pacotes

- É aberta uma conexão separada para cada cliente subscritor e publicador;
- Os bytes dos pacotes recebidos pelos clientes foram interpretados individualmente;
- Os bytes dos pacotes enviados são construídos individualmente e armazenados em um array;
- O broker trata dos seguintes *Control Packet Types*:
  - CONNECT e CONNACK;
  - PUBLISH;
  - SUBSCRIBE e SUBACK;
  - PINGREQ e PINGRESP;
  - DISCONNECT.

CONNECT E CONNACK

- Pacote CONNECT: requisitar uma conexão ao broker;
- Recebido um pacote CONNECT, um pacote de CONNACK é enviado;
  - Isso permite que os demais pacotes dos clientes sejam enviados.

PUBLISH

- O broker interpreta o nome do tópico e a mensagem a ser publicada;
  - Em seguida, imprime essas informações;
- Ocasionalmente, pacote DISCONNECT do cliente publicador também é recebida no pacote de PUBLISH;
  - O broker remove esses bytes do pacote DISCONNECT;
- O pacote de PUBLISH é redirecionado para os clientes subscritores do tópico em questão.

SUBSCRIBE E SUBACK

- O broker interpreta o nome do tópico;
  - Em seguida, essa informação é impressa;
- Manda um pacote de SUBACK para reconhecer a subscrição do cliente;
- Quando uma publicação é feita no tópico em questão:
  - Recebe o pacote do publicador;
  - Manda para o cliente subscritor.

PINGREQ E PINGRESP

- Pacote PINGREQ: enviado periodicamente pelo cliente subscritor para confirmar conexão com o broker;
- Recebida um pacote PINGREQ, um pacote PINGRESP é enviado.

DISCONNECT

- Recebido o pacote de DISCONNECT:
  - Socket de conexão com o cliente é fechado;
  - Processo responsável por tratar os pacotes do cliente é encerrado.

Implementação
Comunicação entre processos

- A comunicação entre processos foi feita por um descritor de arquivos;
- Recebido um pacote PUBLISH:
  - Um descritor de arquivos será aberto para escrita;
  - Escreverá pacote PUBLISH no final do arquivo responsável pelas mensagens do tópico: /tmp/<nome_do_topico>;
- Recebido um pacote SUBSCRIBE:
  - Um descritor de arquivos será aberto para leitura;
  - Caso não exista, cria um arquivo temporário com nome do tópico subscrito, no formato: /tmp/<nome_do_topico>;
  - Tentará ler pacote PUBLISH do final do arquivo em um loop.

Decisões de Projeto

- O broker não trata erros de pacotes enviados pelo cliente subscritor/publicador;
- O nome de tópicos está restrito a 120 caracteres;
  - Caso esse valor seja ultrapassado, o broker não funcionará corretamente.

Testes

- Especificações da máquina:
  - CPU: Intel Core i5-1135G7 @ 2.4Ghz
  - RAM: 16GB
  - Placa de rede: Realtek RTL8111/8168/8411 PCI Express Gigabit Ethernet
- Broker MQTT foi executado na máquina local;
- Os clientes foram executados em um container docker rodando na mesma máquina do Broker MQTT.
- Desempenho do broker:
  - A cada 0.5s, verificou-se o consumo de CPU pelo broker e seus processos filhos;
  - O uso da rede foi monitorado utilizando o Wireshark para contabilizar o total de pacotes trafegados entre o broker e os clientes.

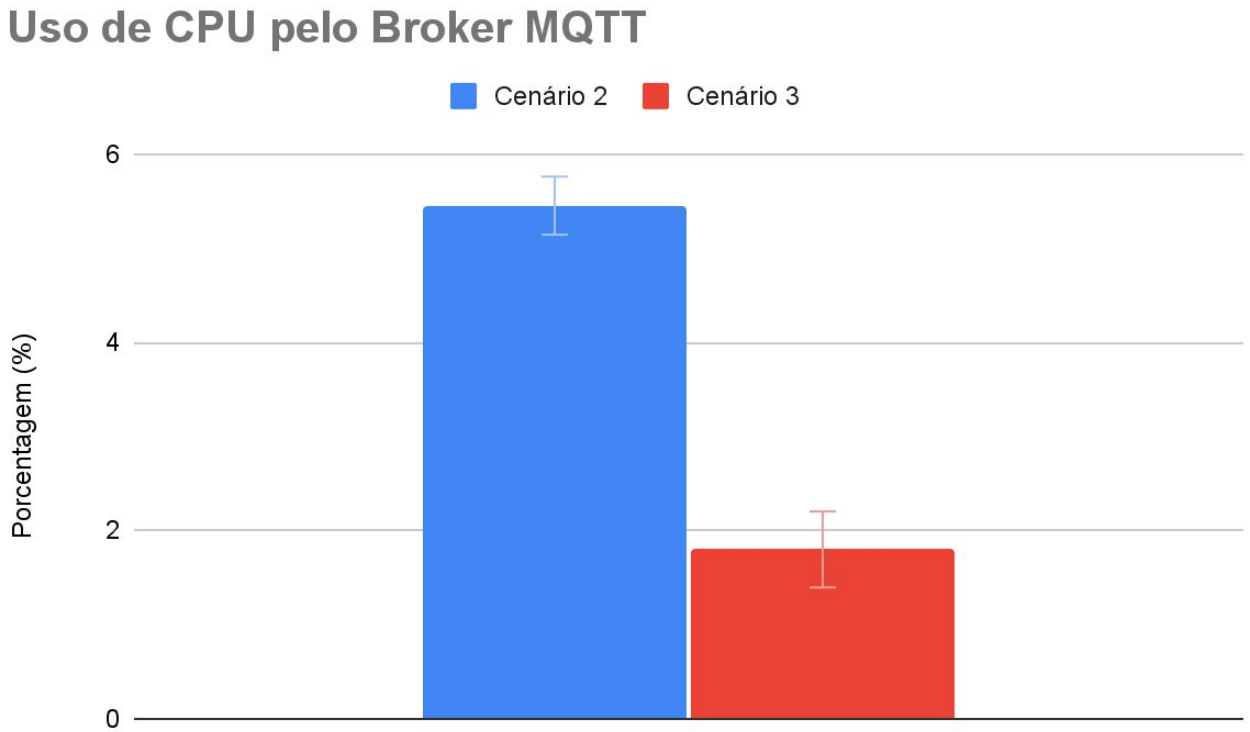
Testes

- Para os testes, foram considerados três cenários:
  - Cenário 1: broker, sem clientes conectados;
  - Cenário 2: broker, com:
    - 100 subscritores conectados a um tópico chamado “topic”;
    - 2 publicadores mandando as mensagens “Hello World!” e “FOOBAR” ao tópico “topic”, em um intervalo de 0.05 segundos;
  - Cenário 3: broker, com:
    - 1000 subscritores conectados a um tópico chamado “topic”;
    - 2 publicadores mandando as mensagens “Hello World!” e “FOOBAR” ao tópico “topic”, em um intervalo de 0.05 segundos;
- Foram realizados 10 testes para cada um dos cenários, monitorando o comportamento da CPU e rede por 30 segundos;
- Os gráficos apresentam intervalos de confiança de 95%.

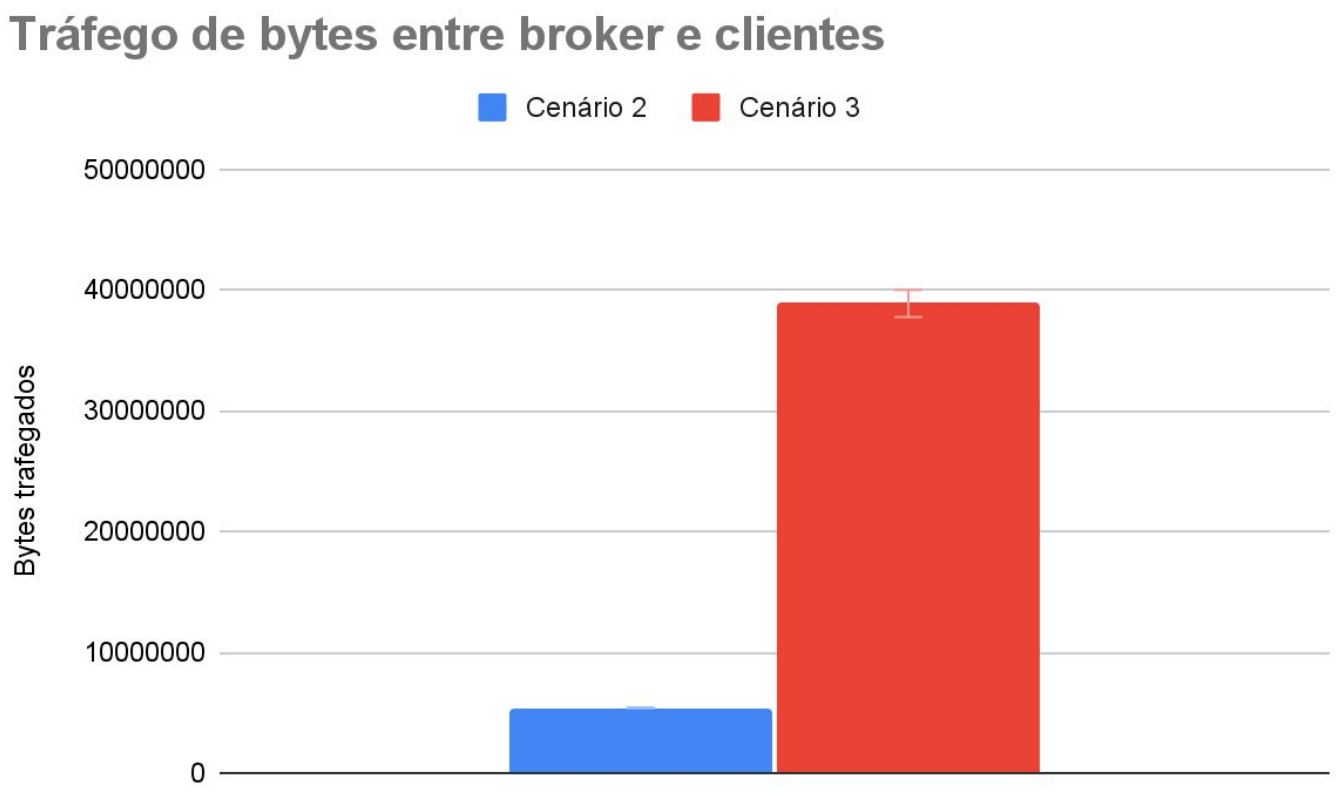
Cenário 1

- O Broker MQTT não utilizou porcentagem significativa de CPU;
- Como não existem clientes conectados nesse cenário, não houve tráfego de pacotes. Por isso, não foi possível monitorar a rede.

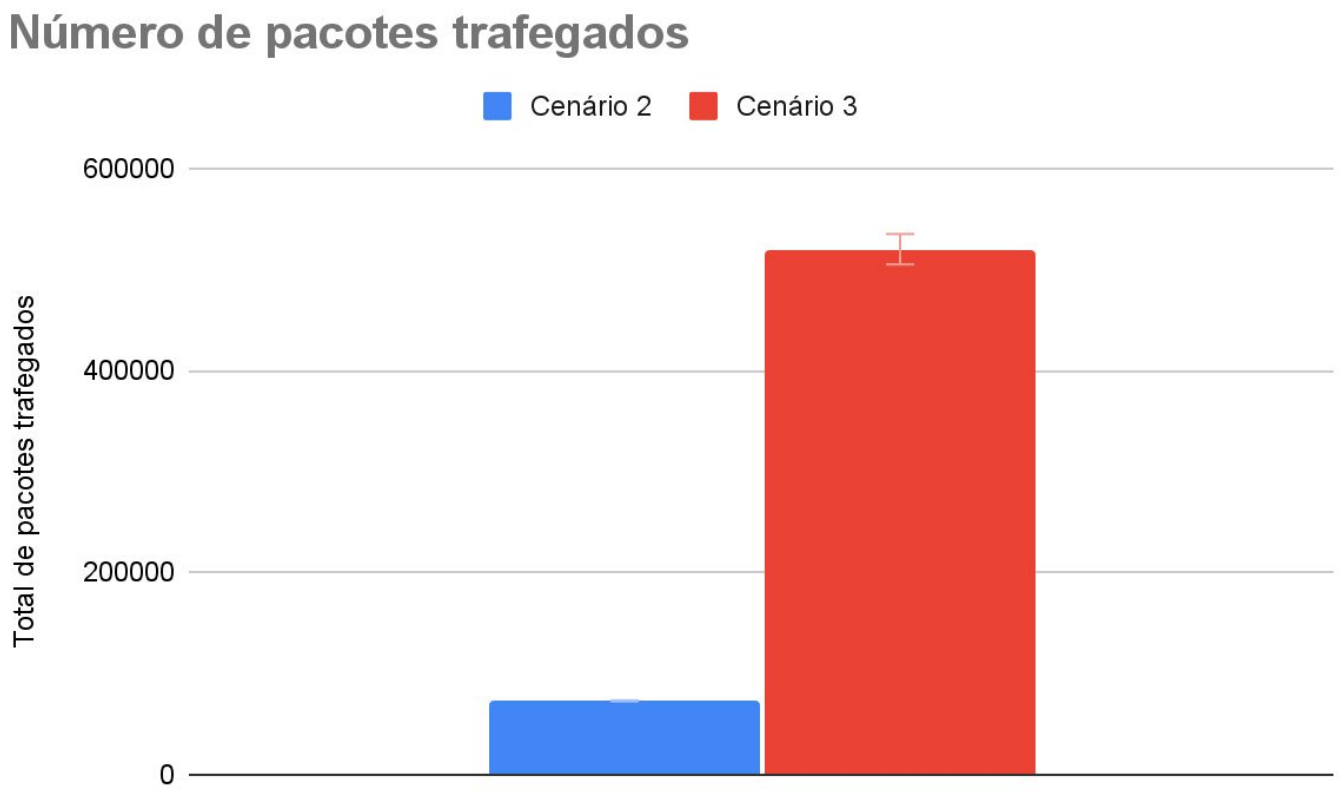
Gráfico CPU - Cenários 2 e 3



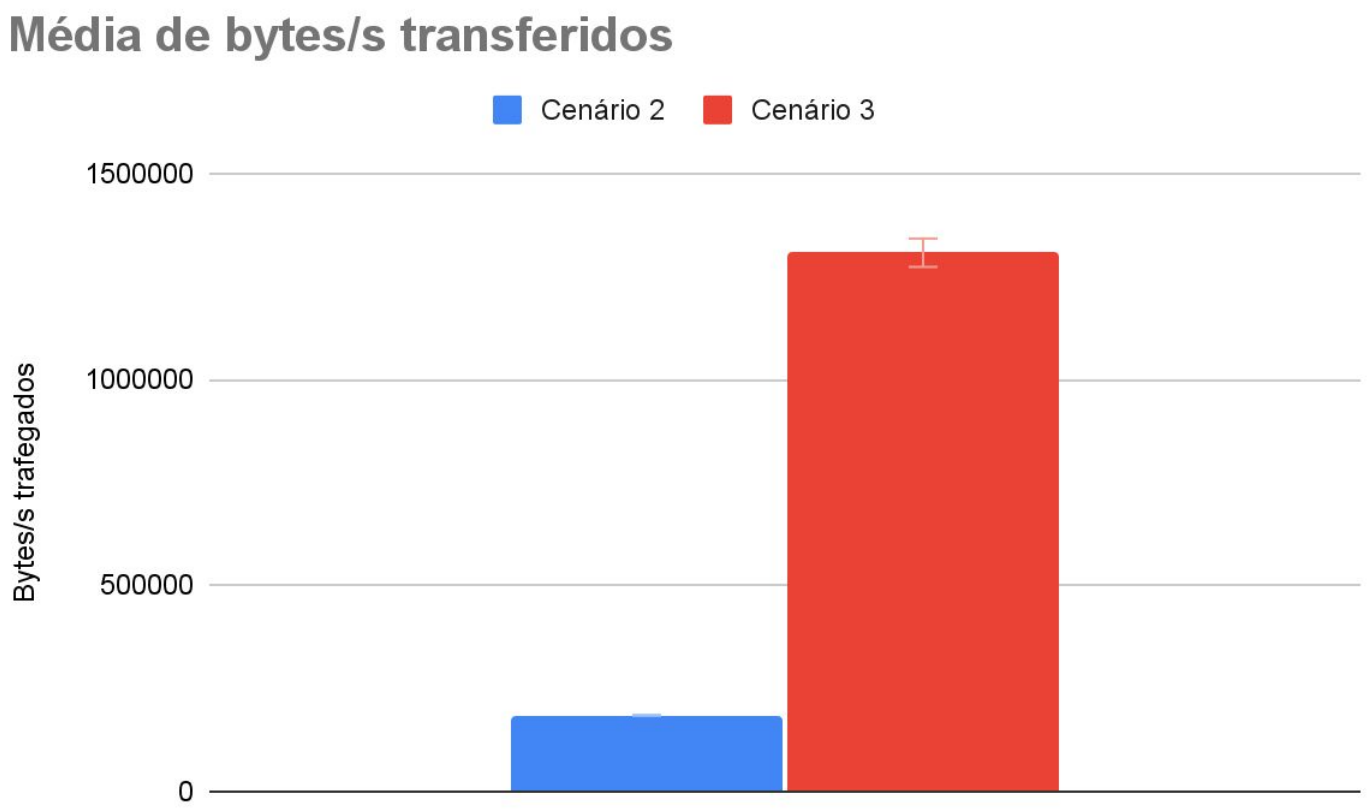
Gráficos Rede - Cenários 2 e 3



Gráficos Rede - Cenários 2 e 3



Gráficos Rede - Cenários 2 e 3



Resultados e Conclusão

- O uso de CPU pelo broker foi maior no cenário 2 em relação ao 3, o que contradiz o esperado, dado que existem mais processos de clientes subscritos:
  - Isso pode ser devido ao fato dos clientes subscritores estarem gastando mais tempo lendo do arquivo do que esperando que algo seja escrito;
  - Como há um loop para ler do tópico, esperar acaba por usar mais da CPU;
- Os resultados de rede coincidiram com o esperado:
  - Total de bytes e pacotes trafegados entre broker e clientes foram maiores no cenário 3;
  - Quanto maior número de clientes conectados, maior o número de pacotes que devem ser trocados para realizar a comunicação.