REDES EP 1

- NOME: Luã Nowacki Scavacini Santilli
- NUSP: 11795492
- EMAIL: lua.til@usp.br ou luasantilli@gmail.com

- Tópicos:
  - Portabilidade
  - Tabela de Hashing e Vetor Dinâmico
  - Pacotes Implementados
  - Experimentos

Portabilidade

- O software foi desenvolvido de forma portável entre sistemas UNIX e Windows, isso foi realizado através de um mapeamento disponível no livro “Hands-On Network Programming with C” o que foi usado está disponível no arquivo “portable_header.h”.
- Ao invés do uso de processos ou threads a função “select()” foi utilizada. Ela recebe o equivalente a um vetor de sockets e permite a um processo identificar quais dos sockets abertos receberam bytes.
- O compilador utilizado para Windows foi o Microsoft (R) C/C++ Optimizing Compiler Version 19.30.30706 for x64
- O compilador utilizado para o Linux foi o gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0.

Tabela de Hashing e Vetor Dinâmico

- A implementação usa de duas tabelas de hashing. Uma cuja chave é o nome de um tópico, e cujo valor é um vetor dinâmico listando os sockets inscritos nesse tópico. Para fazer o envio de mensagens do tipo “PUBLISH” para os clientes do tipo sub. E outra que tem como chave um socket e como valor um struct com informações relevantes a esse socket, como nome do tópico inscrito, endereço e ID do cliente. Ela é usada para imprimir as mensagens na tela, indicando o nome do cliente que está enviando um pacote, e para a eventual desconexão de um cliente a um tópico.
- A tabela de hashing e o vetor dinâmico utilizados vêm da biblioteca “stb_ds.h”, que implementa estruturas de dados dinâmicas em C.

Pacotes Implementados

- Dentre os tipos de pacote descritos na versão 3.1.1 do protocolo MQTT, disponível em http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, foram implementados os seguintes:
  - CONNECT
  - CONNACK
  - PUBLISH
  - PUBACK
  - SUBSCRIBE
  - SUBACK
  - PINGREQ
  - PINGRESP
  - DISCONNECT

CONNECT e CONNACK

- Na implementação do CONNECT os dados da variable header são lidos de forma normal. Dado que a parte anterior do código já garante que os bytes foram lidos para um buffer de tamanho suficiente.
- Os dados do cliente - como o cliente id - são salvos numa tabela de hashing para posterior uso
- Caso não ocorra falhas durante o processamento do CONNECT é enviado um CONNACK para o cliente.

PUBLISH e PUBACK

- Na implementação do PUBLISH o broker recebe o tópico e a mensagem, depois acessa uma tabela de hashing para ver a lista dos processos inscritos no tópicos, enviando o pacote do PUBLISH para cada um dos sockets inscritos.
- Caso este processo não falhe, é enviado um PUBACK para o cliente que primeiro enviou o pacote PUBLISH.

SUBSCRIBE e SUBACK

- Para implementar o SUBSCRIBE o broker recebe e interpreta o pacote, então adiciona o socket no vetor dinâmico da tabela de hashing com o apropriado tópico.
- Caso este processo não falhe, o broker envia um SUBACK para o cliente.

PINGREQ e PINGRESP

- Caso o broker receba uma mensagem do tipo PINGREQ de algum dos clientes, ele envia diretamente um PINGRESP para o mesmo.

DISCONNECT

- Quando o broker recebe um pacote do tipo DISCONNECT, ele busca o nome do tópico em que o cliente foi cadastrado, e caso tenha sido, o remove do vetor dinâmico da entrada do tópico da tabela de hashing. E fecha o socket associado com o cliente.

Experimentos

- Os experimentos foram realizados em 3 máquinas virtuais usando o Virtual Box e uma imagem do Ubuntu 18.04
- Os scripts utilizados para testar a performance do broker são pub_test.py, sub_test.py e measure.py. Os scripts estão na pasta “test_scripts”
- A biblioteca psutil do Python foi utilizada para medir a utilização da CPU e dos bytes trocados, a partir de uma resolução de 0.1s. Detalhes podem ser vistos no script measure.py
- Os pacotes enviados do tipo “PUBLISH” foram testados com uma payload de 10 bytes
- Os experimentos 2 e 3 foram feitos com 10 tópicos selecionados aleatoriamente dentre os primeiros 10 da lista topics.txt
- O resultado dos experimentos está disponível na pasta “test_data”.
- Os tópicos foram selecionados a partir da seguinte requisição: https://www.randomlists.com/random-words?dup=false&qty=1000
- As máquinas virtuais utilizadas possuem 1 CPU e 1GB de RAM
- A máquina que realizou o experimento possui 8GB de RAM, e uma placa AMD Ryzen 5 5600G with Radeon Graphics 3.90 GHz, rodando Windows 10
- Além dos resultados expostos também foram gravados o número de conexões a cada “timestamp”

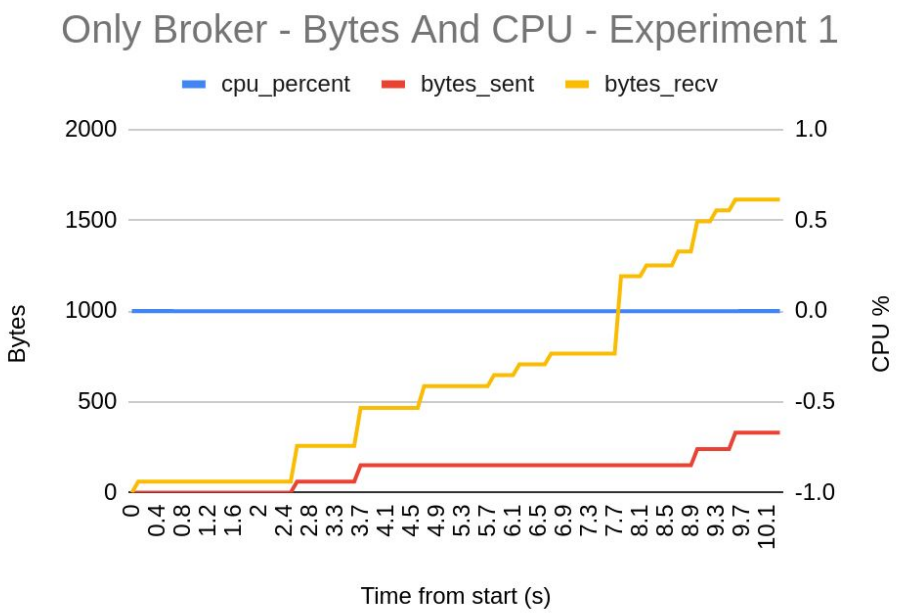
Experimento 1

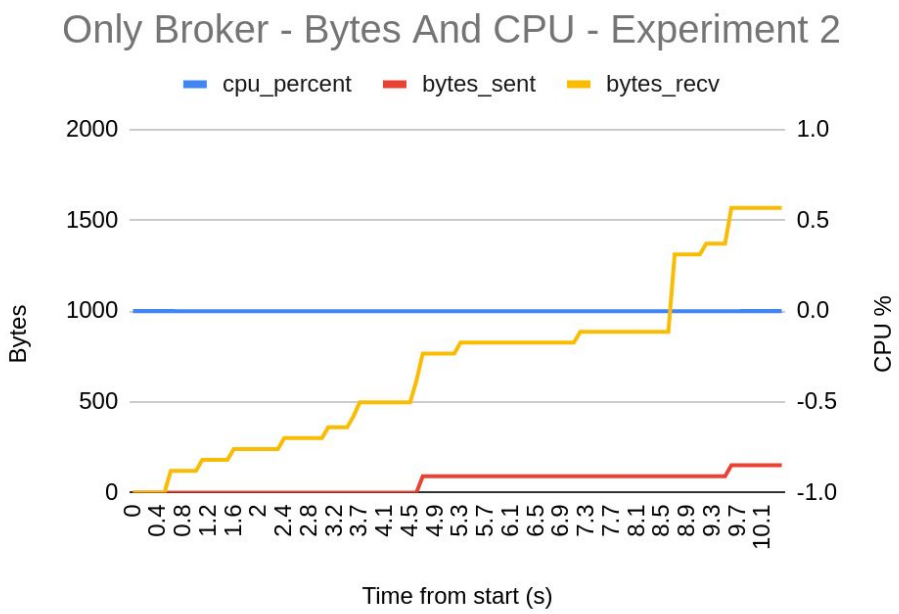
1. Rode o broker
2. Rode o script de testes
3. Espere 10 segundos
4. Desative o script de testes

Comentários:

1. O Experimento foi realizado 2 vezes
2. O Número de bytes trocados se trata de todos da máquina virtual que está rodando o broker
3. Em alguns experimentos a resolução de 0.1s não foi o suficiente para gravar a utilização da CPU

Somente o Broker





Experimento 2

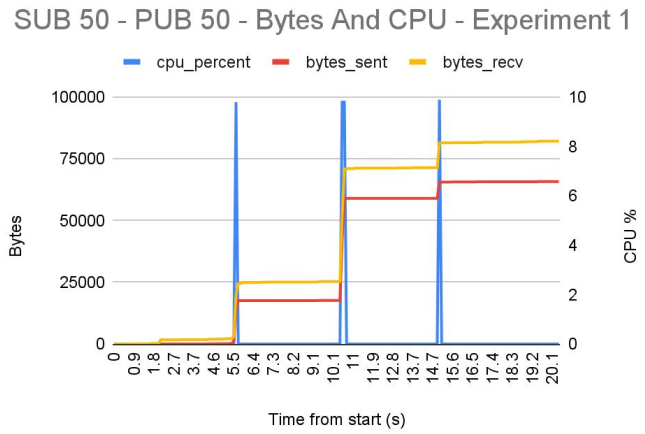
1. Rode o broker
2. Rode o script de testes
3. Espere 5 segundos e rode 50 clientes sub com 10 tópicos diferentes
4. Espere 5 segundos e rode 50 clientes pub com 10 tópicos diferentes enviando 10 bytes cada
5. Espere 5 segundos e desative os 10 clientes sub
6. Espere 5 segundos e desative o script de testes
7. Realize o mesmo com 90 clientes sub e 10 pub

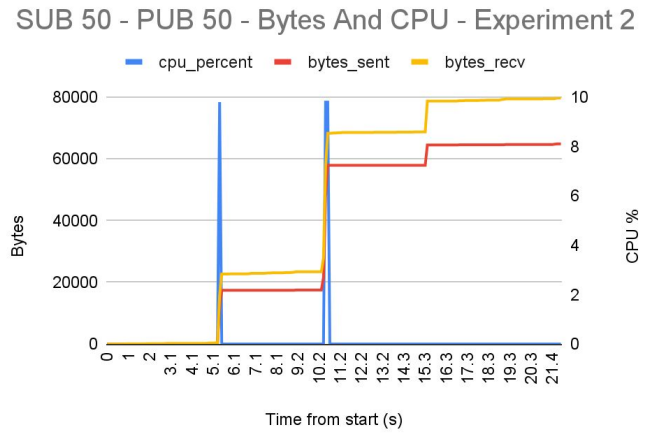
Comentários:

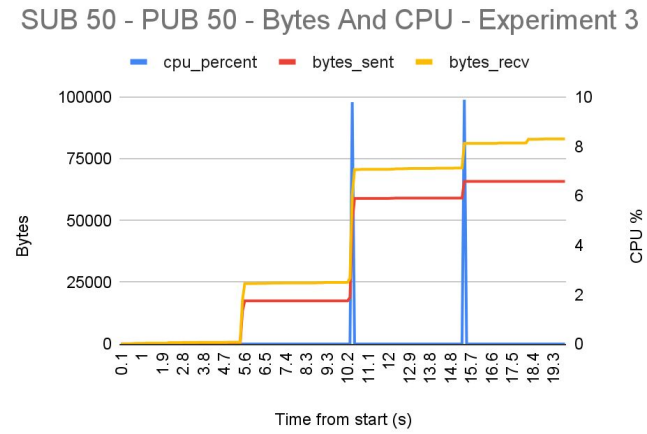
1. O Experimento foi realizado 3 vezes
2. O Número de bytes trocados se trata de todos da máquina virtual que está rodando o broker
3. Em alguns experimentos a resolução de 0.1s não foi o suficiente para gravar a utilização da CPU

100 Clientes

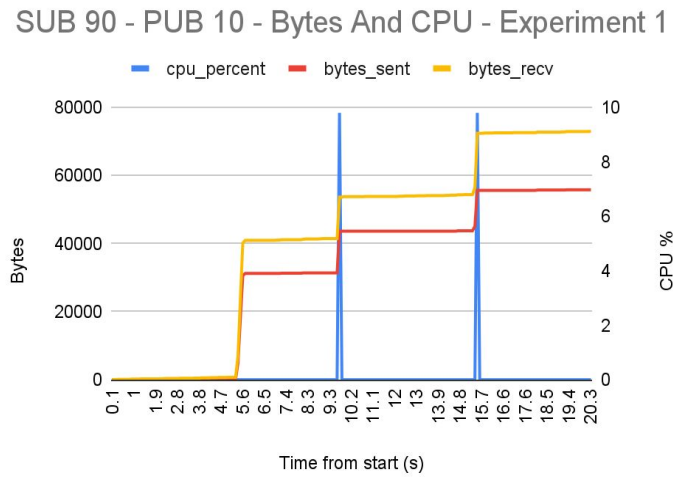
50% SUB - 50% PUB

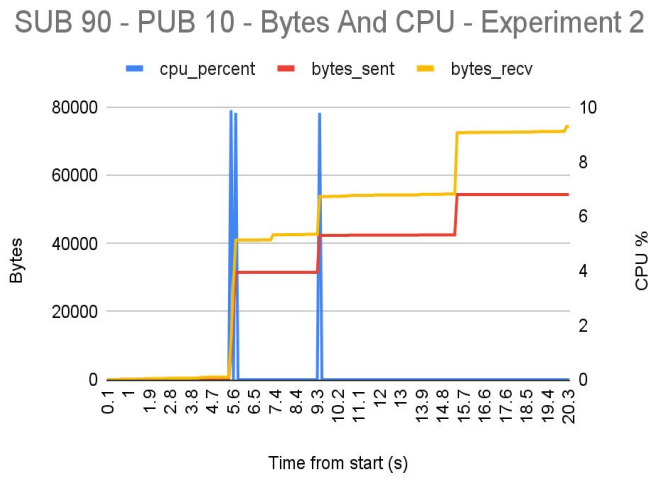


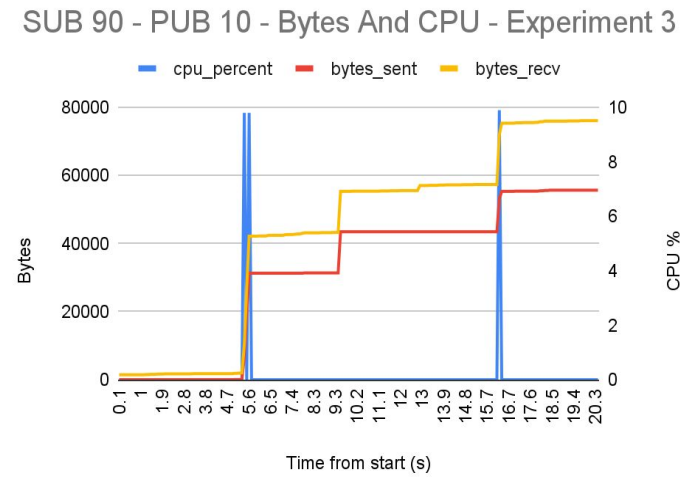




90% SUB - 10% PUB







Experimento 3

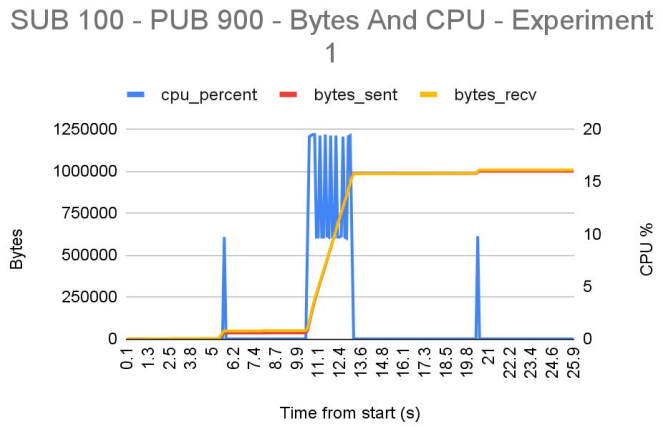
1. Rode o broker
2. Rode o script de testes
3. Espere 5 segundos e rode 100 clientes sub com 10 tópicos diferentes
4. Espere 5 segundos e rode 900 clientes pub com 10 tópicos diferentes enviando 10 bytes cada
5. Espere 5 segundos e desative os 900 clientes sub
6. Espere 5 segundos e desative o script de testes
7. Realize o mesmo com 500 clientes sub e 500 pub e depois com 900 clientes sub e 100 pub

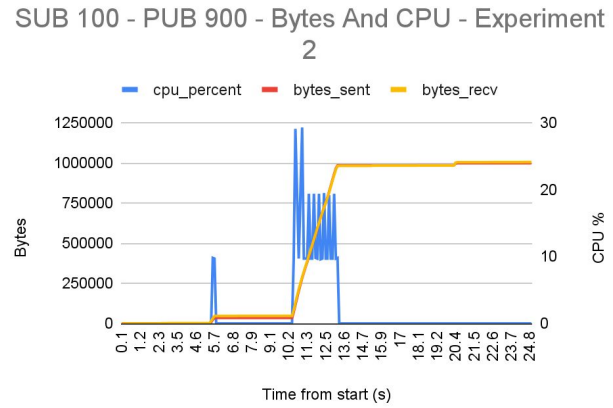
Comentários:

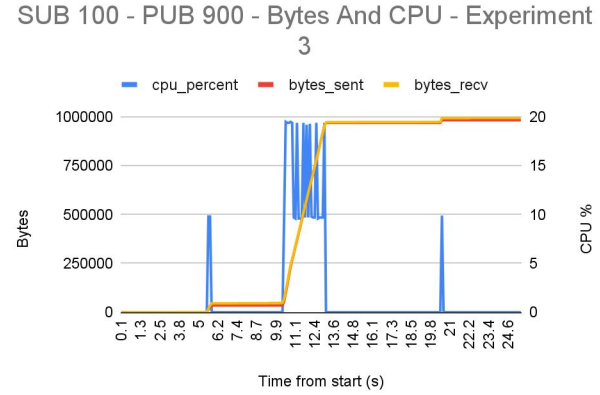
1. O Experimento foi realizado 3 vezes
2. O Número de bytes trocados se trata de todos da máquina virtual que está rodando o broker

1000 Clientes

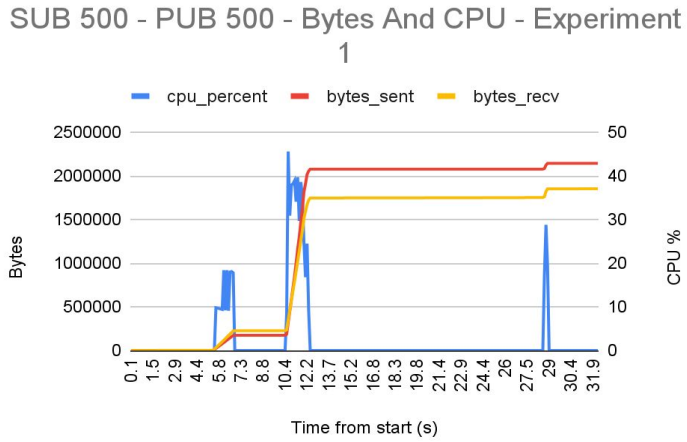
10% SUB - 90 % PUB

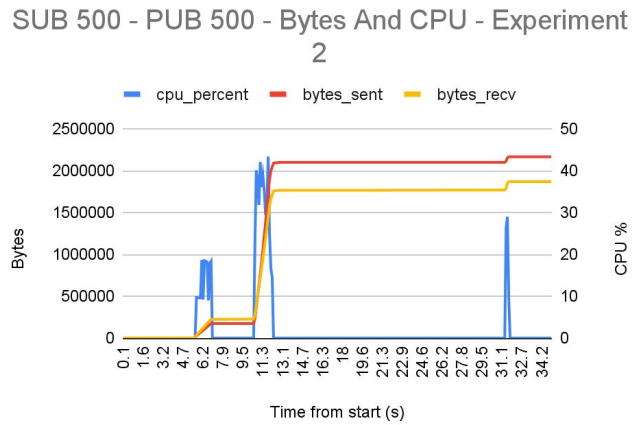


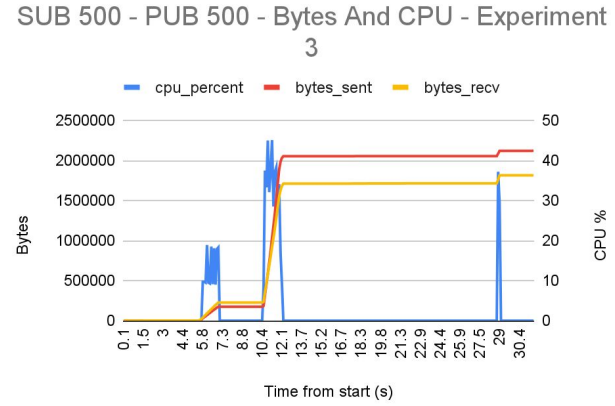




50% SUB - 50% PUB







90% SUB - 10% PUB