EP1 - MAC0352
Broker MQTT

Nome: Bruno Pereira Campos
Nusp: 11806764
Data: 25/04/2020

Conteúdos

1. Implementação
    a. Decisões de Projeto
    b. Distinção dos Pacotes
    c. SUBSCRIBE
    d. PUBLISH
    e. CONNECT, PING e DISCONNECT
2. Testes
    a. Recursos Utilizados
    b. Especificações de Hardware
    c. Resultados
        i. 100 Clientes
        ii. 1000 Clientes
        iii. Sem clientes

Implementação

1.a Decisões de Projeto

- Não é possível manter variáveis do programa que armazenam a relação entre tópicos e inscritos, pois são utilizados muitos processos;
- Assim foi decidido a seguinte estrutura de funcionamento:
  - Para cada execução do broker, é criado um diretório temporário com *mkdtemp()*, para evitar conflitos entre execuções distintas do servidor;
  - Com o nome de cada tópico, dentro do diretório do item anterior, é gerado um diretório cujo nome é o hash MD5 do tópico;
  - O uso do hash unifica o tamanho do tamanho de todos os diretórios, tornando mais fácil sua manipulação
  - Dentro de cada diretório de um tópico, para cada um dos inscritos desse tópico é criado o pipe para comunicação com ele (inscrito), com a função *mkstemp()*;
  - *mkdtemp()* e *mkstemp()* garantem a unicidade do nome dos arquivos/diretórios criados;

1.b Distinção dos pacotes

- OBS: Todos os passos até a criação do processo para cada conexão foram mantidos do exemplo dado no enunciado;
- O primeiro byte do buffer é analisado;
- É feito um bitshift (>> 4) pois o tipo do pacote está presente no bits mais significativos do primeiro byte;
- A partir daí já é possível executar o código relativo ao tipo de cada pacote.

1.c SUBSCRIBE

- Função *get_remaning_length()* obtém o valor do campo “Remaning Length” e o a quantidade de bytes que esse campo ocupa;
- Função *get_topic_length()* obtém o valor do campo “Topic Length”, que está contido nos próximos 2 bytes;
- É obtido então a string do tópico em si (valor do campo “Topic”);
- É gerado um hash MD5 com essa string;
- Se ainda não existir, é criado um diretório do tópico, cujo nome é o hash.

1.c SUBSCRIBE

- É gerado um pipe temporário, referente a esse cliente, dentro do diretório do tópico;
- É gerado um processo filho que abre o pipe criado e fica escutando um *write* nele;
- Quando o write ocorre, o valor lido é mandado pro cliente, esse valor representa o pacote de alguma mensagem publicada;
- O processo pai manda o SUBACK para o cliente logo depois de criar o processo filho;

1.d PUBLISH

- Função *get_remaning_length()* obtém o valor do campo “Remaning Length” e o a quantidade de bytes que esse campo ocupa;
- Função *get_topic_length()* obtém o valor do campo “Topic Length”, que está contido nos próximos 2 bytes;
- É obtido então a string do tópico em si (valor do campo “Topic”);
- É gerado um hash MD5 com essa string;

OBS: Até aqui, igual ao SUBSCRIBE

1.d PUBLISH

- Com o hash é identificado o diretório referente ao tópico;
- Com as funções *opendir()* e *readdir()* da biblioteca *dirent.h* é gerado um loop com cada um dos arquivos (pipes) do diretório;
- O pacote é repassado (escrito) em cada um dos pipes que pertencem ao tópico;
- Os arquivos são fechados.

1.e CONNECT e PING e DISCONNECT

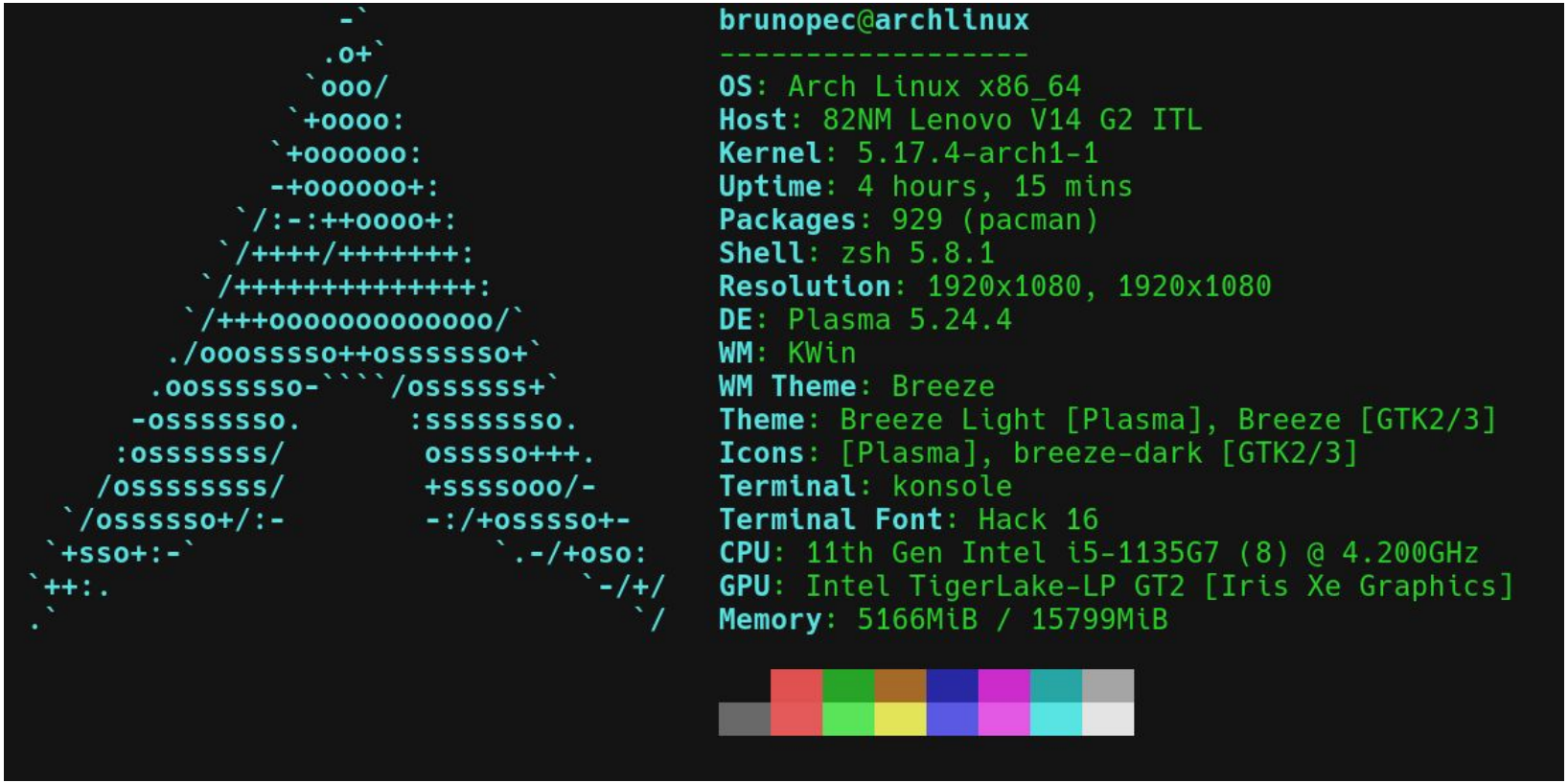
- Para o CONNECT e PING, é enviado uma resposta diretamente para o cliente;
- CONNACK para CONNECT e PINGRESP para PING;
- Os header são pré definidos, não é necessário realizar nenhuma operação;
- O DISCONNECT faz o programa sair do loop e desconecta o cliente.

Testes

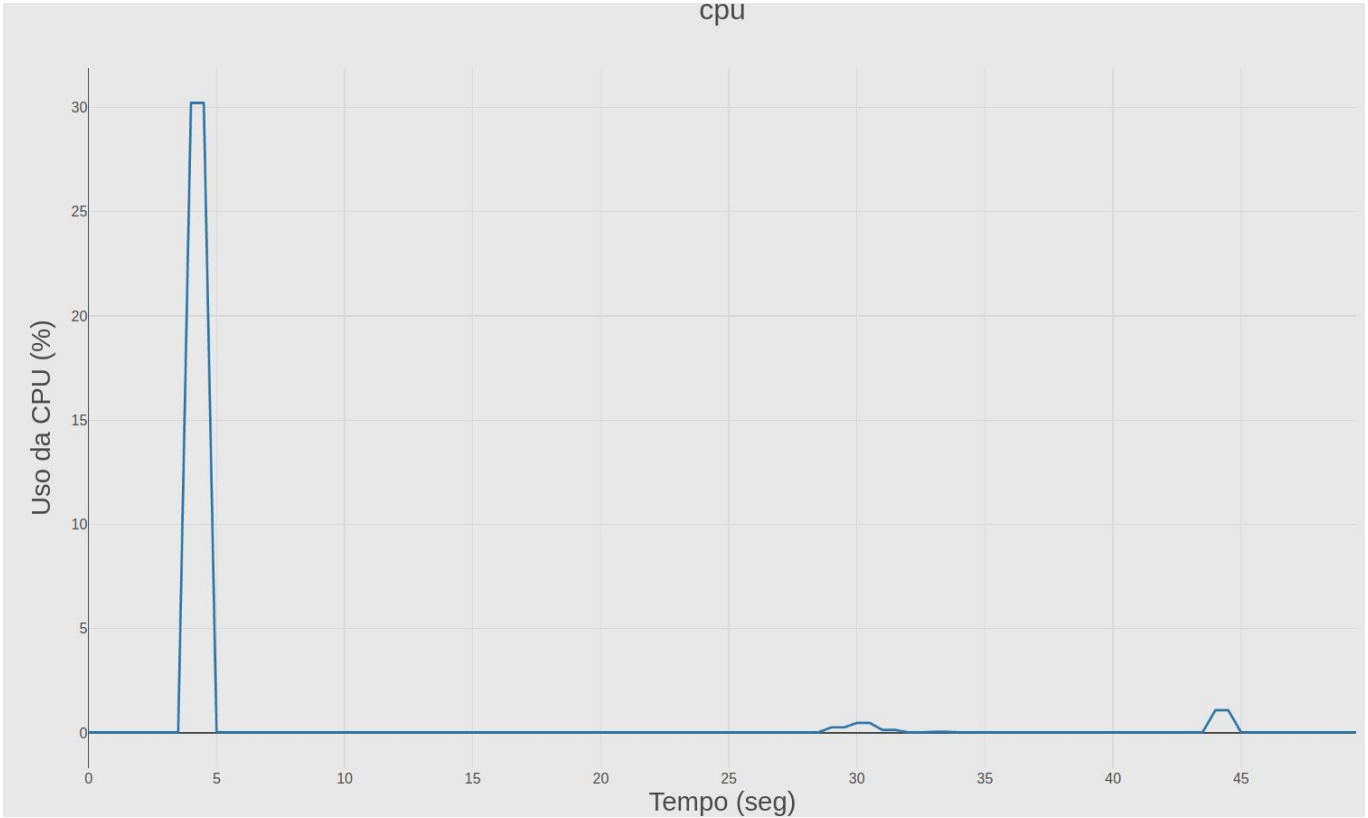
2.a Recusos Utilizados

- Utilização de um container Docker (Docker Compose) para execução do broker;
- Mais prático de configurar do que uma máquina virtual;
- Utilização o comando *Docker Stats* que permite obter o uso de CPU e memória em tempo real do container, facilitando a coleta dos dados;
- Cria uma rede virtual entre o computador e o container;
- Pacotes foram obtidos utilizando o programa Wireshark;
- Filtros: *mqtt and ip.addr == 172.18.0.1*, para pegar apenas os pacotes que chegam e partem do container;

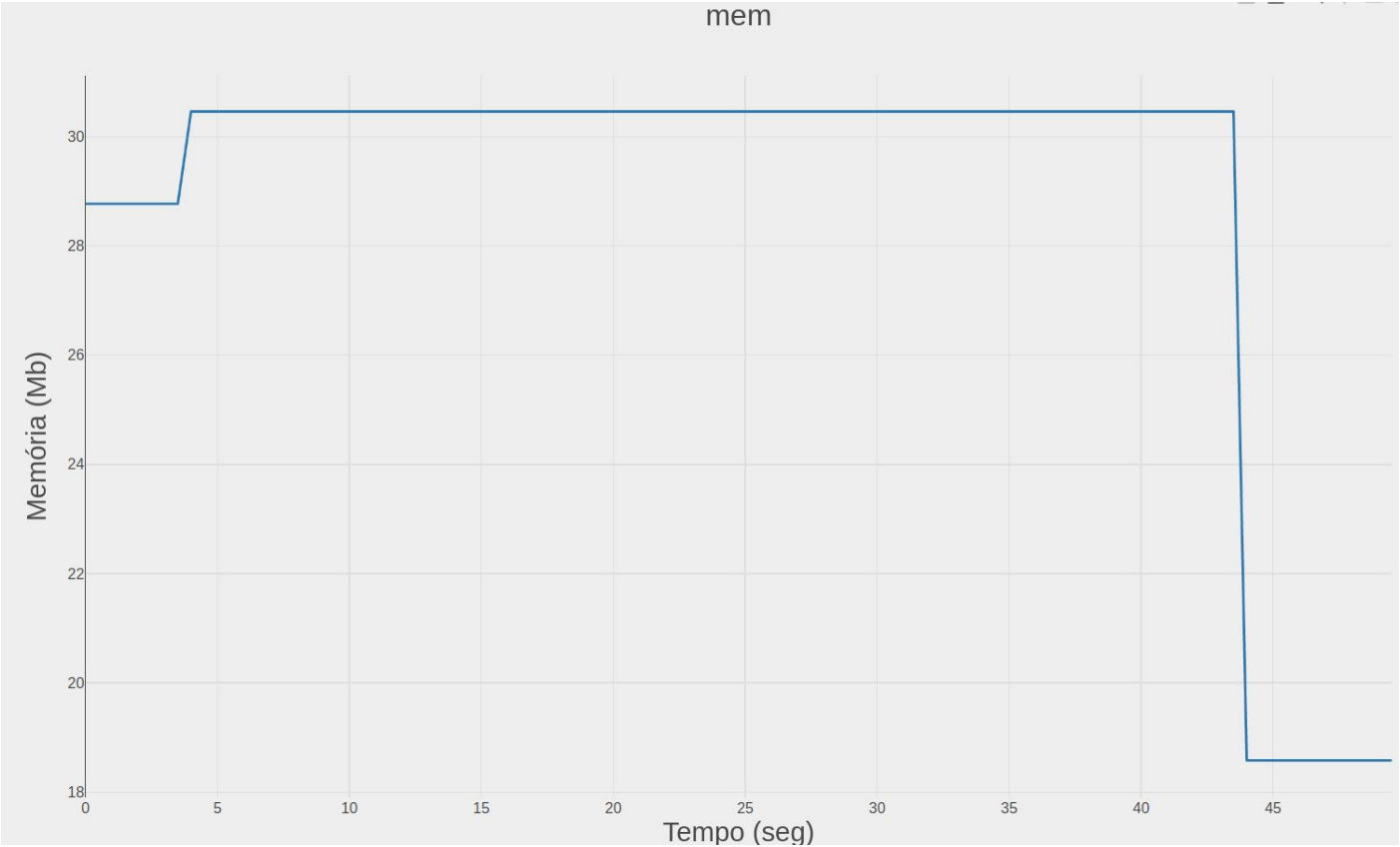
2.a Especificações de Hardware



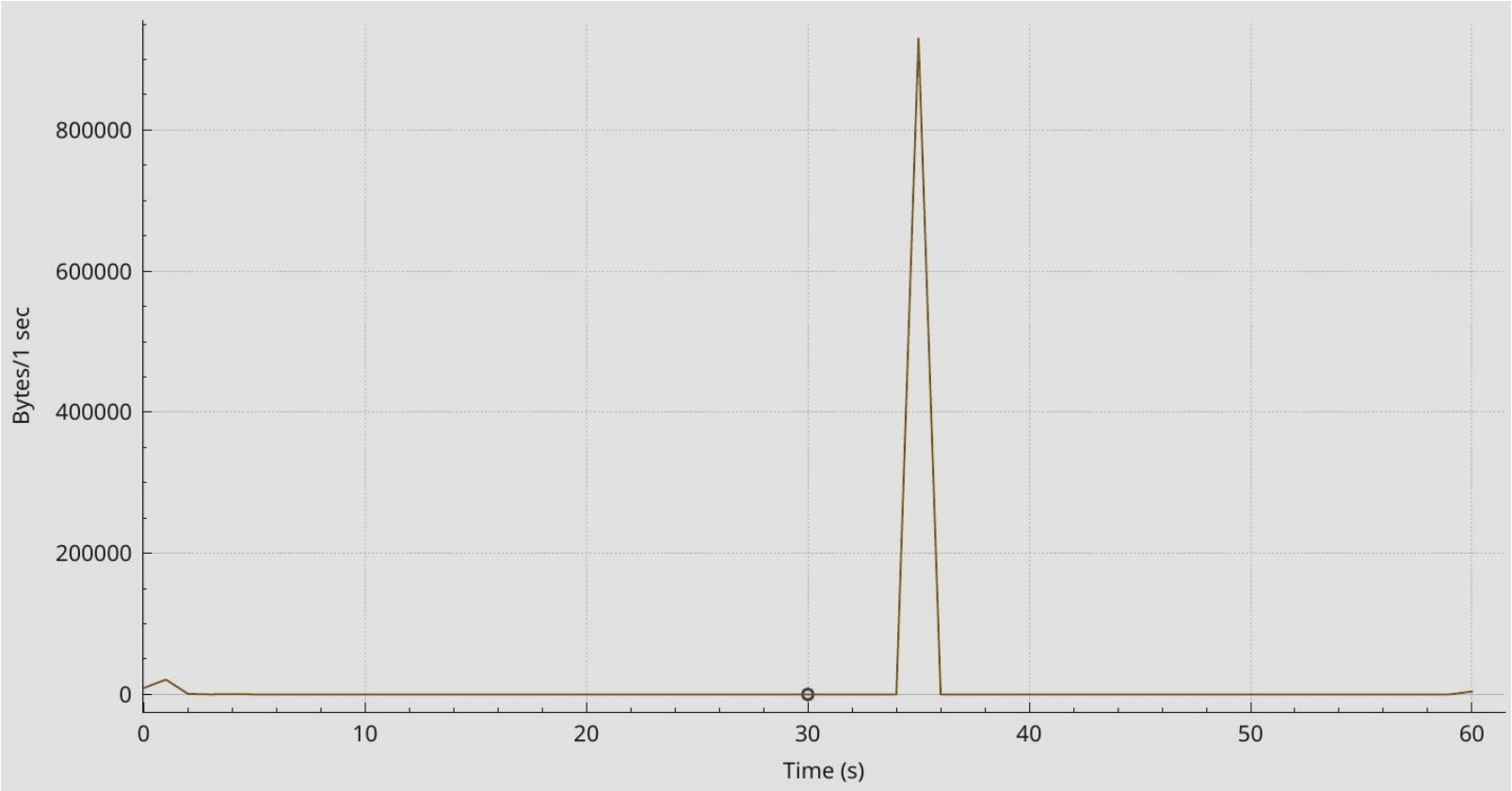
2.c.i Resultados - 100 Clientes - Uso de CPU



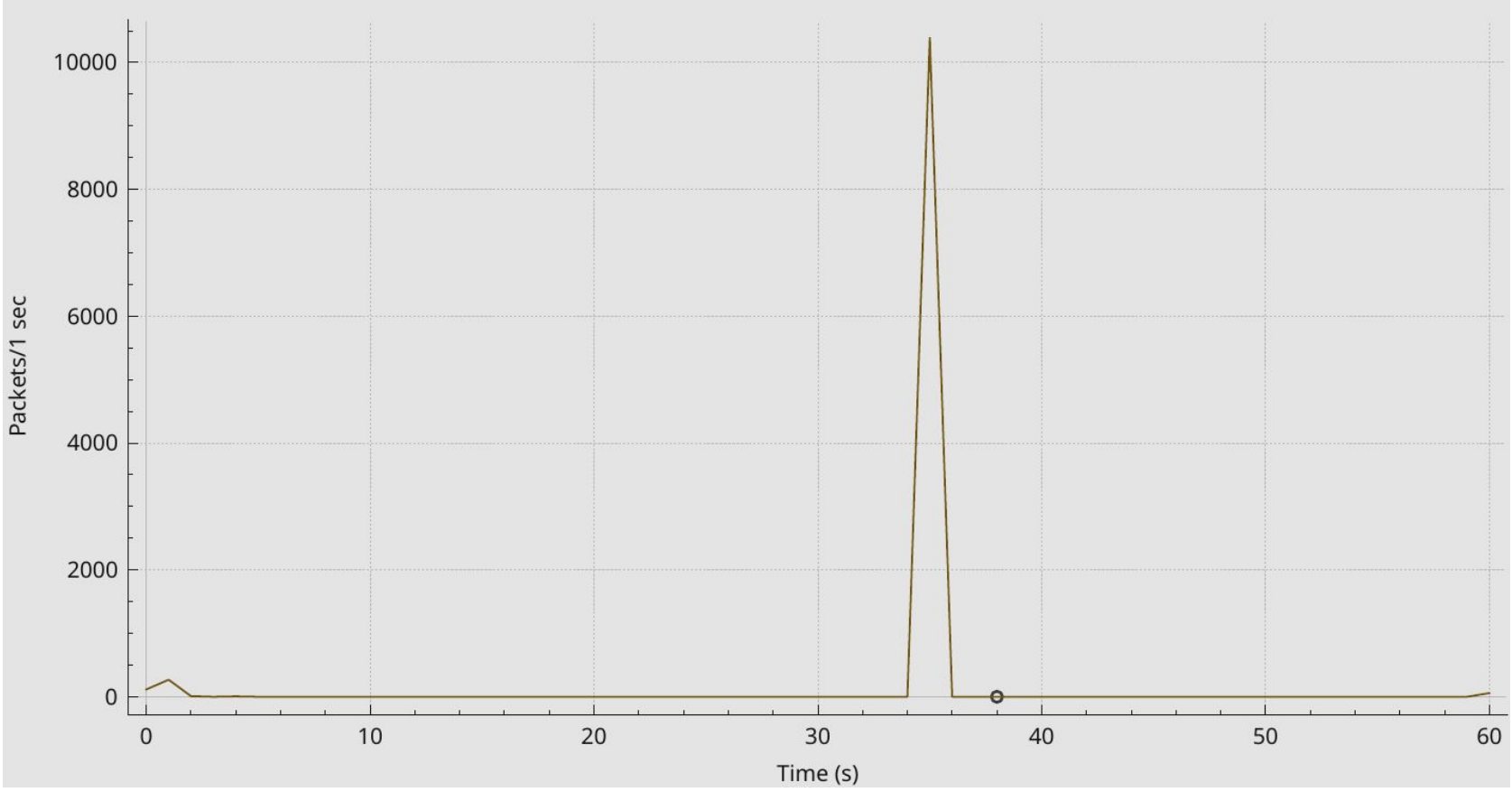
2.c.i Resultados - 100 Clientes - Uso de Memória



2.c.i Resultados - 100 Clientes - Uso da Rede (Bytes)



2.c.i Resultados - 100 Clientes - Uso da Rede (Pacotes)

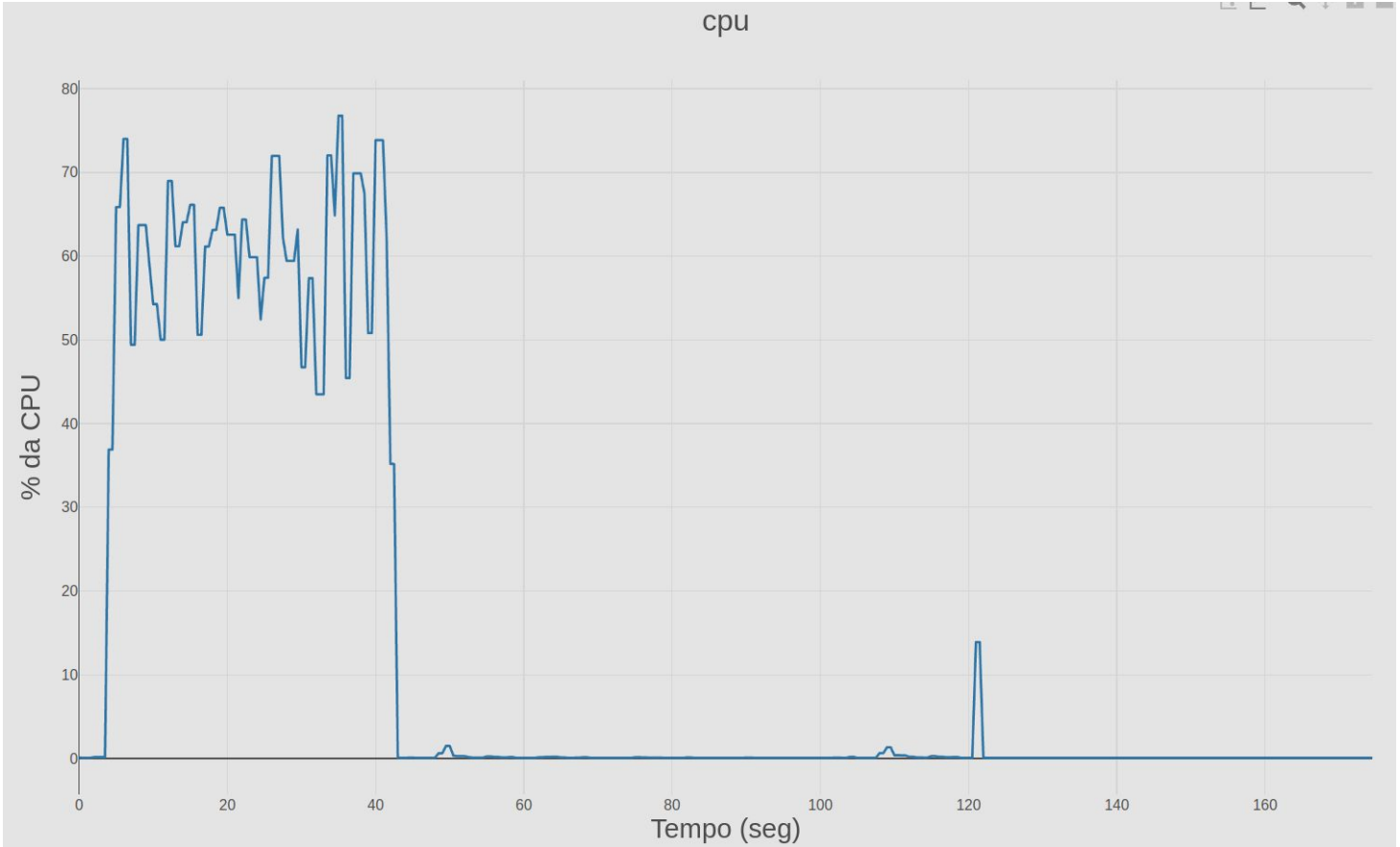


2.c.i Resultados - 100 Clientes - Uso da Rede (Geral)

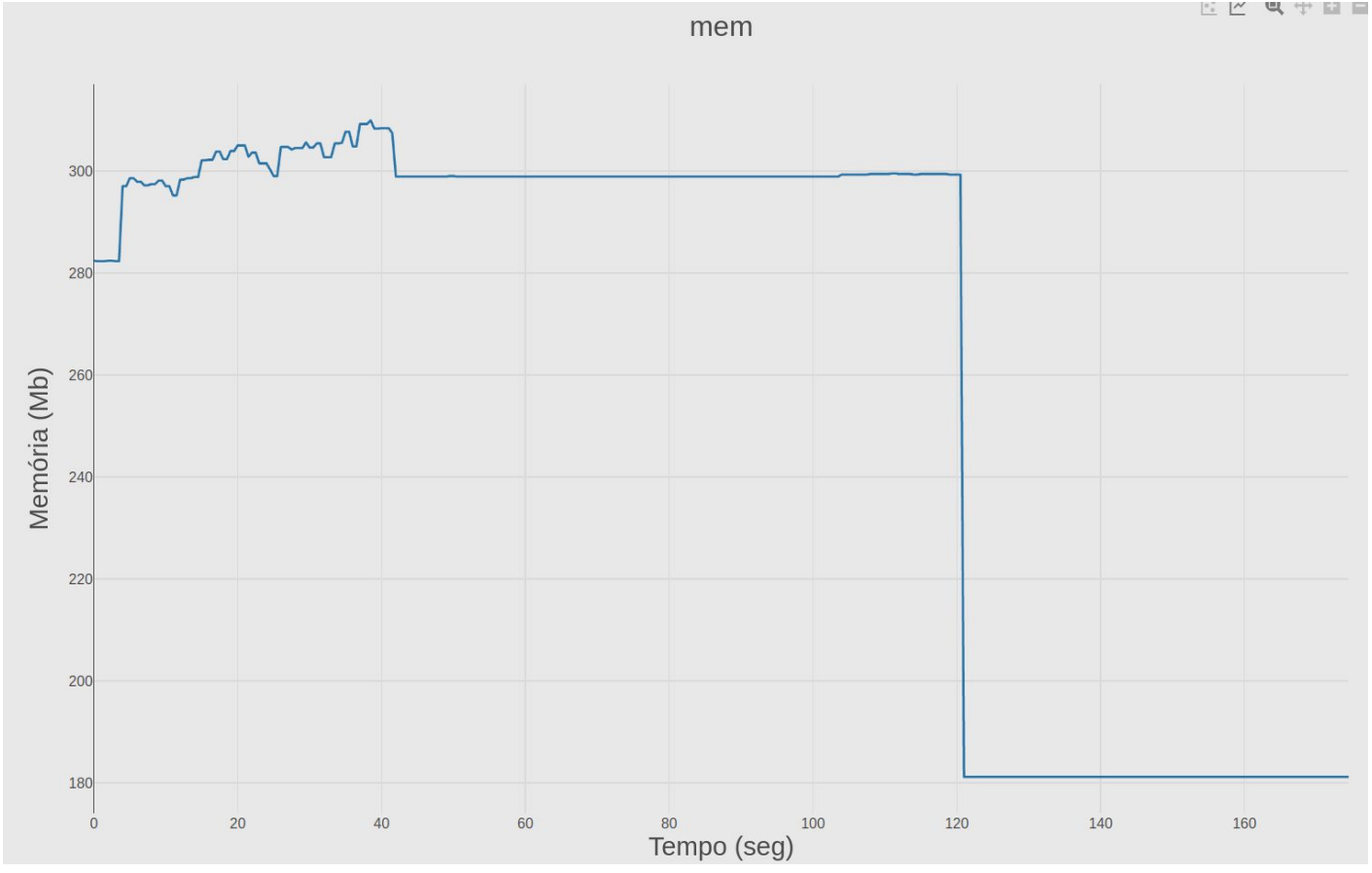
Statistics

Measurement	Captured	Displayed
Packets	68782	10855 (15.8%)
Time span, s	60.515	60.514
Average pps	1136.6	179.4
Average packet size, B	85	89
Bytes	5837588	965648 (16.5%)
Average bytes/s	96 k	15 k
Average bits/s	771 k	127 k

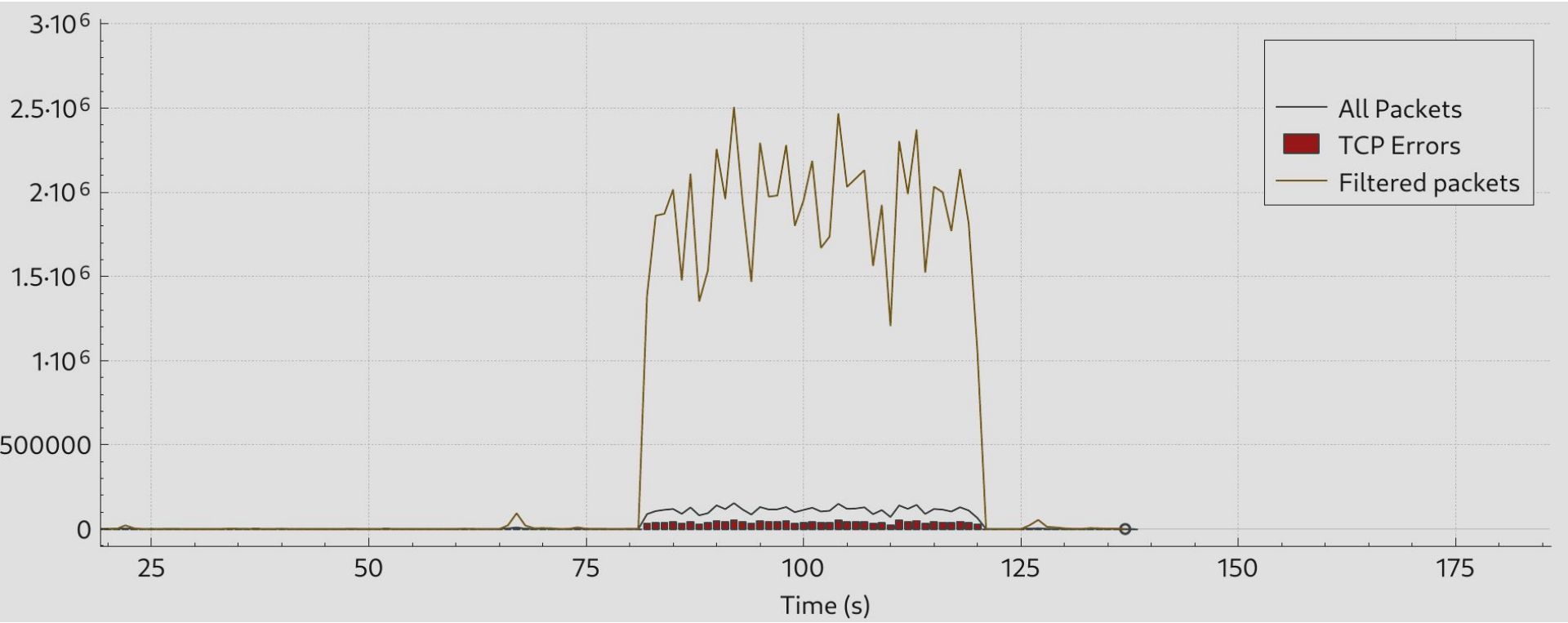
2.c.ii Resultados - 1000 Clientes - Uso de CPU



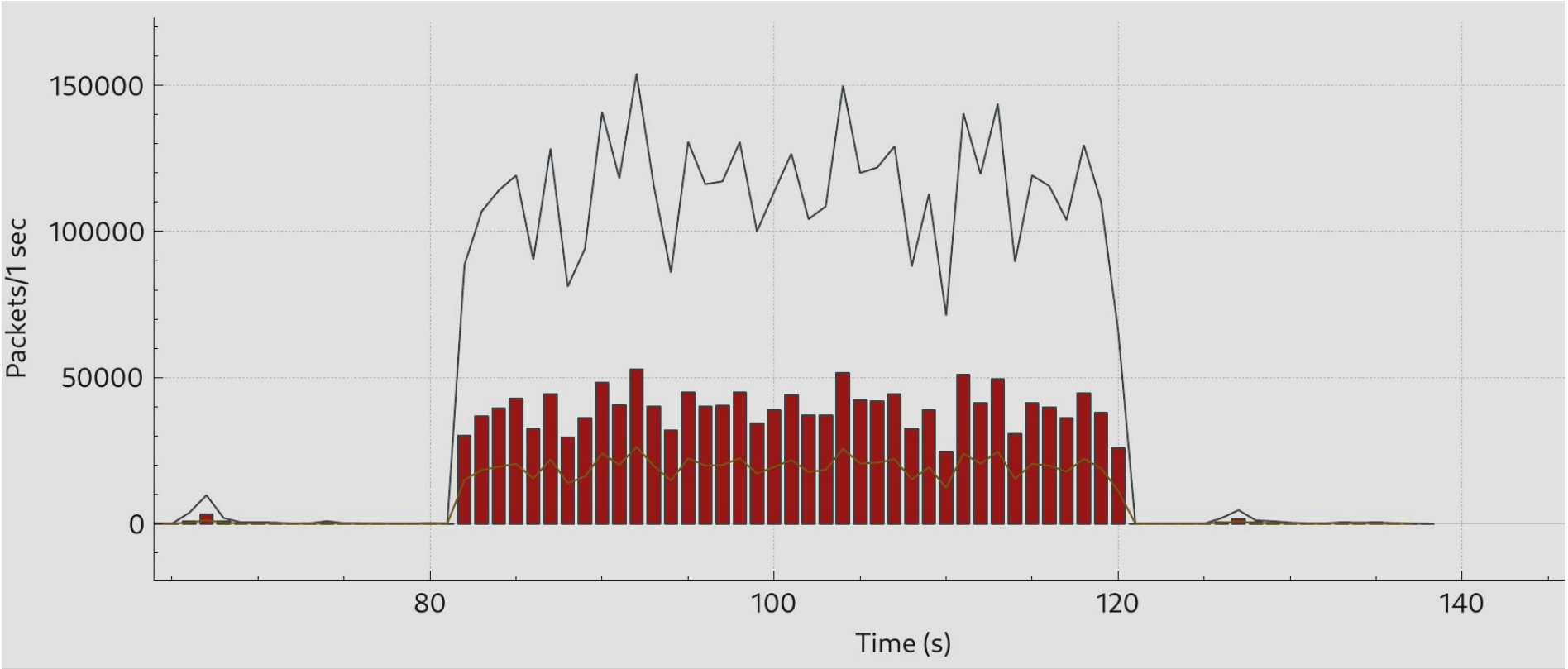
2.c.ii Resultados - 1000 Clientes - Uso de Memória



2.c.ii Resultados - 1000 Clientes - Uso da Rede (Bytes)



2.c.ii Resultados - 1000 Clientes - Uso da Rede (Pacotes)



2.c.ii Resultados - 1000 Clientes - Uso da Rede (Geral)

Measurement	Captured	Displayed
Packets	4492313	763358 (17.0%)
Time span, s	138.054	130.496
Average pps	32540.3	5849.7
Average packet size, B	89	98
Bytes	401914580	74570943 (18.6%)
Average bytes/s	2.911 k	571 k
Average bits/s	23 M	4.571 k

2.c.iii Resultados - Sem Clientes - Uso de CPU

- Ficou estável em 0%.

2.c.ii Resultados - Sem Clientes - Uso de Memória

- Ficou estável em 584Kb.

2.c.ii Resultados - Sem Clientes - Rede

- Nenhum pacote foi recebido ou enviado.