

Servidor HTTP em C - Relatório 3

Arthur Teixeira Jardim¹, Marcelo Marchioro Cordeiro¹, Rodrigo Acosta Segui¹

¹Universidade Federal do Pampa (UNIPAMPA)

– Bagé – RS – Brasil

{arthurjardim, marcelocordeiro, rodrigosegui}.aluno@unipampa.edu.br

Abstract. *This article describes the implementation of the third version of an HTTP server, using the HTTP 1.1 version that handles concurrent connections (several clients simultaneously) with controlled rate, and developed with the C language. Details of the implementation of the algorithm are present in this writing, the description development and the results found.*

Resumo. *Este artigo descreve a implementação da terceira versão de um servidor HTTP, utilizando a versão HTTP 1.1 que trata conexões concorrentes (vários clientes simultaneamente) com taxa controlada, e desenvolvido com a linguagem C. Está presente nesta escrita detalhes da implementação do algoritmo, a descrição do desenvolvimento e os resultados encontrados.*

1. Introdução

Um servidor é um software ou computador, com sistema de computação centralizada que fornece serviços a uma rede de computadores, chamada de cliente. Um servidor HTTP é um software responsável por aceitar pedidos em HTTP de clientes, geralmente os navegadores, e servi-los com respostas em HTTP, incluindo opcionalmente dados, que geralmente são páginas web, tais como documentos em HTML com objetos embutidos (imagens, etc) ou um computador que executa um programa que provê a funcionalidade descrita anteriormente. Nessa versão do trabalho implementou-se um servidor desta vez seguindo o protocolo HTTP 1.1 diferentemente do que foi entregue na versão anterior, além disso adicionou-se a implementação correta da utilização de envio por taxa controlada, pois observou-se um equívoco na versão anterior.

2. Objetivos

O trabalho tem como objetivo incentivar de forma prática conceitos de redes de computadores. Para isso foi proposta a implementação de um servidor HTTP fazendo-se uso da linguagem de programação C. O servidor precisa ser capaz de enviar objetos (exceto arquivo html) com taxa controlada dependendo da origem da requisição, baseando-se em um arquivo auxiliar que contém IP do cliente e a taxa máxima, em kbps, que deve ser usada para atendê-lo. Caso o IP não esteja salvo no arquivo deve-se usar como padrão para a taxa máxima 1000 kbps. Na terceira versão desenvolvida, o servidor deveria ser capaz de tratar requisições HTTP com conexões concorrentes (vários clientes simultaneamente), seguindo o protocolo HTTP 1.1, assim como incorporar os mecanismos de Qualidade de Serviço (Quality of Service - QoS) como:

- 1) Envio de objetos (exceto arquivo html) com taxa controlada dependendo da origem da requisição, utilizando um arquivo auxiliar contendo IP do

- cliente e a taxa máxima, em kbps, que deve ser usada para atendê-lo. Caso o IP não esteja no arquivo, usar a taxa de 1000 kbps.
- 2) Apresentar visualmente, em tempo real, os clientes que estão sendo atendidos, informando uma estimativa de atraso fim-a-fim e largura de banda do cliente (usando estimativa de tempo baseada no Round-Trip-Time (RTT), obtida a partir de duas requisições consecutivas do mesmo cliente para referentes a uma página web – solicitação do html e a solicitação do primeiro objeto referenciado neste html).
 - 3) Controlar o número de clientes a serem atendidos ao mesmo tempo, a fim de evitar que a vazão máxima do servidor seja alcançada. Também é preciso dividir a taxa pré definida caso tenha conexões simultâneas do mesmo IP.

3. Metodologia

Preliminarmente foi necessário realizar um levantamento sobre as funcionalidades requeridas da versão 2 na qual não foram concluídas. A partir disso, realizou-se um estudo sobre como implementar o servidor seguindo o protocolo HTTP/1.1 (conexões concorrentes persistentes). Após a fase de aquisição de conhecimentos deu-se início ao processo de desenvolvimento de conexões persistentes, modificou-se o código para receber várias requisições de um cliente em uma mesma conexão, assim utilizou-se a função *close(socket)* para realizar o fechamento da conexão somente após um tempo sem requisições.

Também modificou-se a forma de implementação da transferência com taxa controlada a partir de IP em cache, verificou-se que anteriormente a função foi implementada erroneamente utilizando somente a função *sleep* para aguardar um tempo até o envio dos bytes. Após um estudo descobriu-se uma forma correta de implementação que envolve enviar uma certa quantidade de bits para o socket a cada segundo, assim utilizamos a função *gettimeofday()* para computar o tempo de escrita desses bits e posteriormente calculamos o tempo que faltava para completar um segundo. Utilizamos a função *nanosleep* para adormecer a thread pelo tempo necessário e depois retomar o laço de transmissão. Outra necessidade encontrada para essa tarefa, foi a de tratar o valor numérico do tempo em que a thread deve esperar quando atingir o máximo de transferência para o IP em questão, pois as chamadas de sistemas encontradas durante os estudos não funcionam com números decimais.

A implementação novamente se baseou em estudos bibliográficos de conceitos de redes de computadores, assim como tomou-se como base códigos open-source do github dos repositórios a seguir:

<https://github.com/eduardoveiga/ServidorHTTP>

<https://github.com/ozgurhepsag/Multi-threaded-HTTP-Server>

Ademais, seguiu-se a mesma sequência de fluxos da versão 2 e para realização dos testes deve-se usar o navegador pelo endereço a seguir:

<http://localhost:8080/brasil.html>

O servidor está disponível em: <https://github.com/Rodrigo-Segui/ServidorHTTP>.

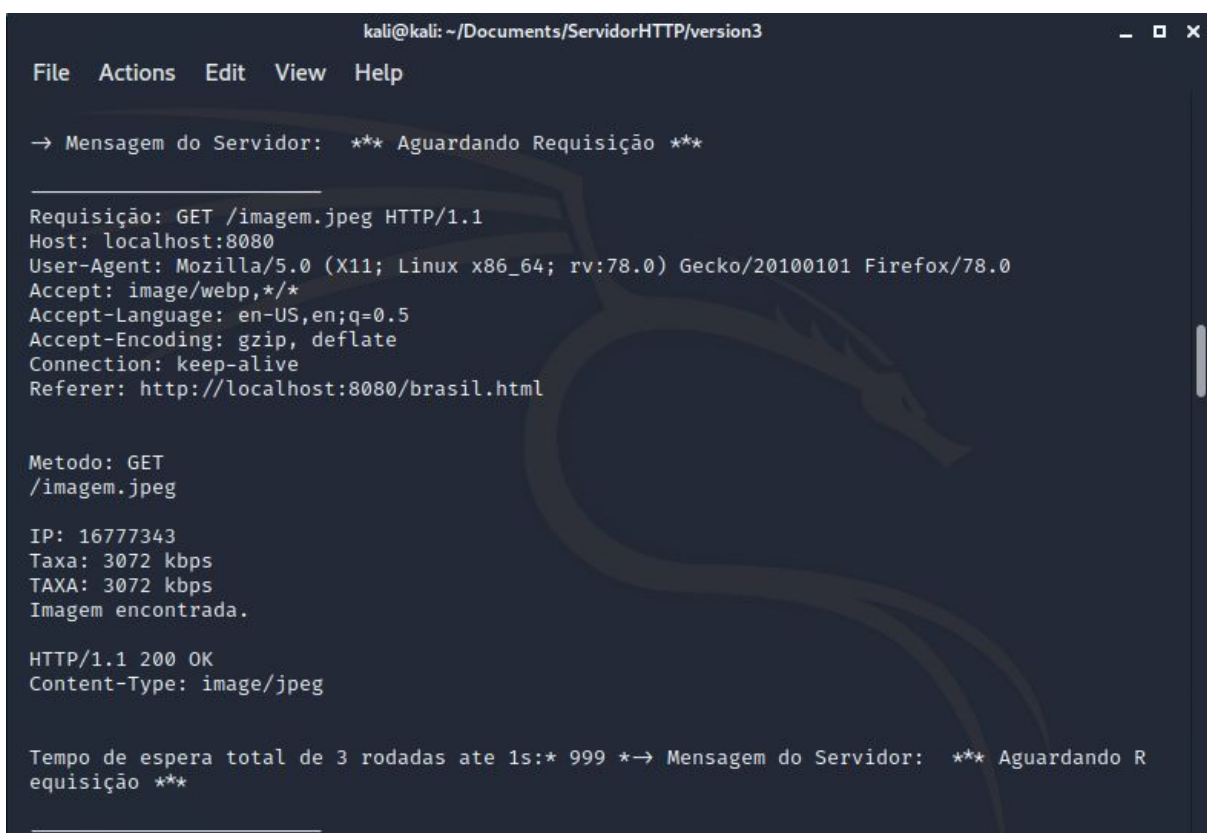
Para executar o servidor a partir do arquivo makefile deve-se colocar o comando *make* no terminal, além de mudar a constante *path* no arquivo *server_function.h* para o caminho dos arquivos do servidor.

4. Resultados e Discussões

Nessa versão foi adicionado as funcionalidades de conexões concorrentes persistentes e envio por taxa controlada na versão com protocolo HTTP/1.1.

Não foi possível incorporar alguns dos mecanismos de Qualidade de Serviço solicitados a tempo para a terceira entrega do projeto. Os mesmo devem ser implementada para a próxima versão.

Foi realizada no âmbito de testes uma requisição do arquivo *brasil.html* utilizando o navegador (Figura 5), onde podemos observar uma página html com três imagens de tamanho diferentes e dois hiperlinks. As Figuras 1 à 4 mostram a execução do servidor.



```
kali@kali: ~/Documents/ServidorHTTP/version3
File Actions Edit View Help

→ Mensagem do Servidor: *** Aguardando Requisição ***

Requisição: GET /imagem.jpeg HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost:8080/brasil.html

Metodo: GET
/imagem.jpeg

IP: 16777343
Taxa: 3072 kbps
TAXA: 3072 kbps
Imagem encontrada.

HTTP/1.1 200 OK
Content-Type: image/jpeg

Tempo de espera total de 3 rodadas ate 1s:* 999 *→ Mensagem do Servidor: *** Aguardando R
equisição ***
```

Figura 1: Execução do servidor parte 1.

```
kali@kali: ~/Documents/ServidorHTTP/version3
File Actions Edit View Help

→ Mensagem do Servidor:  *** Aguardando Requisição ***

Requisição: GET /imagem.jpeg HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost:8080/brasil.html

Metodo: GET
/imagem.jpeg

IP: 16777343
Taxa: 3072 kbps
TAXA: 3072 kbps
Imagem encontrada.

HTTP/1.1 200 OK
Content-Type: image/jpeg
```

Figura 2: Execução do servidor parte 2.

```
kali@kali: ~/Documents/ServidorHTTP/version3
File Actions Edit View Help

Requisição: GET /imagem.jpeg HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost:8080/brasil.html

Metodo: GET
/imagem.jpeg

IP: 16777343
Taxa: 3072 kbps

Requisição: GET /brasil2002.jpeg HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost:8080/brasil.html

Metodo: GET
/brasil2002.jpeg

Requisição: GET /brasilsil.jpeg HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://localhost:8080/brasil.html
```

Figura 3: Execução do servidor parte 3.

```
kali@kali: ~/Dc
File Actions Edit View Help

Metodo: GET
/brasilsil.jpeg
TAXA: 3072 kbps
Imagem encontrada.

HTTP/1.1 200 OK
Content-Type: image/jpeg

Nº Total de bytes enviados 86665:
IP: 16777343
Taxa: 3072 kbps
TAXA: 3072 kbps
Imagem encontrada.

HTTP/1.1 200 OK
Content-Type: image/jpeg

Nº Total de bytes enviados 70781:
IP: 16777343
Taxa: 3072 kbps
TAXA: 3072 kbps
Imagem encontrada.

HTTP/1.1 200 OK
Content-Type: image/jpeg
```

Figura 4: Execução do servidor parte 4.

Servidor HTTP

TODO MUNDO TENTA, MAS SO O BRASIL E PENTA.



HISTORIA DO BRASIL EM COPAS DO MUNDO

É a seleção mais bem-sucedida da história do futebol mundial, sendo a recordista em conquistas em Copas do Mundo, com cinco títulos invictos:

- 1958
- 1962
- 1970
- 1994
- 2002

O BRASIL É a única seleção que venceu a Copa do Mundo em três continentes distintos, na área de quatro confederações continentais de futebol (Europa, América, do Norte/CONCACAF e do Sul/CONMEBOL, e Ásia), e a ter jogado todas as primeiras vezes em 1958, 15 edições depois da de 1938, disputando todas as seguintes.

Mais um motivo de orgulho para os torcedores brasileiros com nossa trajetória pelas Copas do Mundo da FIFA[®], entre os 4 maiores artilheiros de todas as Copas, temos 2 brasileiros: o primeiro lugar é ocupado pelo lendário Ronaldo, com Desde o título do Pentacampeonato, em 2002, no Japão, até a edição de 2014, no Brasil, a seleção brasileira viveu mais momentos de brilho do que de altos. As performances nas Copas de 2006 e 2010 não atenderam às expectativas e, em



Hyperlinks

- [História do Brasil](#)
- [Ronaldo da Silva](#)

Figura 5: Execução do servidor.

6. Conclusão

Foi possível colocar em prática e observar o comportamento de alguns assuntos vistos em sala de aula como o funcionamento do HTTP 1.1 e envio de requisições baseadas em ip em cache. Mesmo faltando algumas funcionalidades previstas para a terceira entrega foi possível ter um resultado satisfatório em questão de aprendizagem melhorando o entendimento em alguns temas de redes de computadores. A implementação do controle do admissão, atendendo um total de clientes simultâneos que não exceda a vazão máxima do servidor, assim como a funcionalidade de apresentar visualmente, em tempo real, os clientes que estão sendo atendidos, informando uma estimativa de atraso fim-a-fim e largura de banda do cliente (usando estimativa de tempo baseada no Round-Trip-Time (RTT) devem ser funcionalidades adicionadas ao servidor para a próxima versão.

7. Referências

G, Gracioli.(2011). Sockets em C, <http://www-usr.inf.ufsm.br/~giovani/sockets.html>, Outubro.

Socket Programming in C, <https://bitwordblog.wordpress.com/2017/01/17/socket-programming-in-c/>, Outubro

R, Lopes, (2011), Simples servidor http com concorrência feito em C, <https://www.vivaolinux.com.br/script/Simples-servidor-http-com-concorrencia-feito-em-C/>, Outubro

F, Junior. (2011), Web Server em C, <https://www.codigofonte.com.br/codigos/web-server-em-c>, Outubro.

H, Costa. (2012), Programação de Sockets em C/C++,<https://student.dei.uc.pt/~hpcosta/rc/rcMod7BAula2.pdf>, Outubro.

FOROUZAN, Behrouz A. Redes de computadores uma abordagem top-down. 1. Porto Alegre AMGH 2013 1 recurso online ISBN 9788580551693.