Servidor HTTP em C - Relatório 1

Arthur Teixeira Jardim[1], Marcelo Marchioro Cordeiro[1], Rodrigo Acosta Segui[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
– Bagé – RS – Brasil

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

Abstract. *This article describes the implementation of the first version of an HTTP server, using the HTTP version that treats competitors (multiple simultaneous customers) at a controlled rate, and developed with the C language. Details of the algorithm implementation, the description of the development, and the results found.*

Resumo. *Este artigo descreve a implementação da segunda versão de um servidor HTTP, utilizando a versão HTTP 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 ainda seguindo o protocolo HTTP 1.0 como na versão anterior, porém adicionou-se o tratamento de clientes de maneira concorrente e utilização de taxa controlada no envio de objetos(exceto html) .

2. Objetivos

O trabalho tem como objetivos incentivar o exercitação de forma prática do que foi visto em sala de aula. 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 segunda versão desenvolvida o servidor deveria ser capaz de tratar requisições http conexões concorrentes (vários clientes simultaneamente), seguindo o protocolo HTTP 1.1.

3. Metodologia

Preliminarmente foi necessário realizar um levantamento sobre as funcionalidades requeridas da versão 1 na qual não foram concluídas. A partir

disso realizou-se um estudo sobre programação concorrente. Após a fase de aquisição de conhecimentos deu-se início ao processo de desenvolvimento de conexões concorrentes, utilizando semáforos para bloquear o acesso de regiões críticas.

Também implementou-se a transferência com taxa controlada a partir de IP em cache, para isso foi codificada a função *rateControl()* que realiza a leitura do arquivo info.txt, onde está os IPs em cache e suas respectivas taxas de transferência, assim se o IP do cliente está no arquivo ele envia os objetos na taxa do mesmo.

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 1. Primeiramente foi criado o socket do servidor, associou-se o socket à um endereço usando a função *bind()*, após utilizamos a função *listen()* para permitir que o socket receba conexões. O programa executa em loop a função *accept()* responsável por conectar o socket do servidor aos clientes que solicitarem uma conexão. O fluxo de receber as requisições, realizar o tratamento e enviá-las estão como detalhadas na versão 1, implementadas nas funções *treatMessage(), treatFile(), treatFileType(), sendFile()*.

O cliente é responsável somente por enviar a requisição de um arquivo, para testes optou-se pelo uso do browser a partir da URL a seguir, diferentemente do uso do telnet na versão 1.

➔ 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.

4. Resultados e Discussões

Nessa versão foi adicionado as funcionalidades de conexões concorrentes e envio por taxa controlada na versão com protocolo HTTP/1.0.

Não foi possível desenvolver o protocolo HTTP/1.1 a tempo para a segunda entrega do projeto. Essa etapa deve ser implementada para a próxima versão, pois ainda foram encontradas dificuldades relacionadas à conceitos do funcionamento do protocolo.

Foi realizada no âmbito de testes uma requisição do arquivo brasil.html utilizando o navegador (Figura 3). A Figura 2 mostra a execução do servidor.

➔ http://localhost:8080/brasil.html

```
CONEXAO:  1
----Aguardando Conexoes----- ...

CONEXAO:  2
Mensagem do Servidor | Cliente Conectado : 8
---------------------- Aguardando Requisição -----------------
Mensagem do Servidor | Cliente Conectado : 7
---------------------- Aguardando Requisição -----------------
------------------------
Requisição: GET /brasil.html HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
i/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
ication/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,pt;q=0.8


------------------------
metodo: GET***************

/brasil.html
File Found.
------------------------
Requisição: GET /imagem.jpeg HTTP/1.1
Host: localhost:8080
Connection: keep-alive
```

Figura 2: Execução do servidor.

Figura 3: Execução do cliente.

6. Conclusão

Foi possível colocar em prática e observar o comportamento de alguns assuntos vistos em sala de aula como por exemplo os sockets e o funcionamento do HTTP 1.0, assim como a área de programação concorrente. Mesmo faltando algumas funcionalidades previstas para a segunda entrega foi possível ter um resultado satisfatório em questão de aprendizagem melhorando o entendimento em alguns temas de redes de computadores. A implementação do protocolo HTTP/1.1 ainda requer por parte dos desenvolvedores maior conhecimento sobre protocolos.

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.