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 1.0 version, 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 primeira versão de um servidor HTTP, utilizando a versão HTTP 1.0, 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 seguindo o protocolo HTTP 1.0, tratando requisições de diferentes tipos de arquivos.

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 1.0 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 versão a ser desenvolvida o servidor HTTP deve ser capaz de tratar conexões concorrentes (vários clientes simultaneamente), seguindo o protocolo HTTP 1.0.

3. Metodologia

Preliminarmente foi necessário realizar um estudo sobre as possibilidades de implementação de um servidor de HTTP 1.0, levando em conta a inexperiência neste âmbito dos desenvolvedores. Também realizou-se um estudo sobre programação de sockets pois nenhum dos membros da equipe havia tido contato com a tecnologia até o momento do desenvolvimento. Após a fase de aquisição de conhecimento deu-se início ao processo de desenvolvimento do

software, a equipe se reuniu e levantou as maneiras de desenvolver as funcionalidades necessárias para o funcionamento adequado do servidor. Todo o desenvolvimento foi realizado no sistema operacional Ubuntu 20.04, o GitHub foi usado para o versionamento do código. Foi utilizado o aplicativo Discord para a comunicação dos membros ao decorrer do desenvolvimento. A extensão LiveShare do Visual Code Studio também foi utilizada para o desenvolvimento do código em conjunto do grupo, facilitando as discussões e a implementação de ideias.

Para a criação da conexão do servidor seguiu-se uma sequência de fluxos. Primeiramente foi criado o socket do servidor, associou-se o socket ao 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 cliente é responsável somente por enviar a requisição de um arquivo, para testes foi utilizado a conexão de terminal virtual telnet, utilizando-se o padrão a seguir:

- ➔ telnet localhost 8080
- ➔ GET /"caminho_do _arquivo/nome_do_arquivo" HTTP/1.0

Ao receber uma conexão do cliente, primeiramente tratamos a mensagem da requisição através da função *treatMessage(),* no tratamento é verificado qual o método, tipo de arquivo e protocolo utilizado na requisição. Nesse primeiro momento o servidor implementado aceita o método GET e o protocolo HTTP 1.0 (conexões não persistentes), em razão disso a conexão é encerrada após o servidor enviar o objeto ao cliente. Outras versões de protocolos ainda não são suportadas.

Após a verificação de que a requisição segue o protocolo HTTP 1.0 é realizado o tratamento do arquivo, através da função *treatFile(),* assim como é realizado a verificação do tipo do arquivo na função *treatFileType()*, onde é possível verificar se o tipo de arquivo requerido é válido dentre os que são aceitos na implementação, assim como é retornado informações do objeto a ser enviado ao cliente. Por fim é utilizado a função sendFile(), responsável por enviar o objeto ao cliente.

O programa foi separado em TADS para uma melhor organização, onde a execução é separada em um arquivo principal *main.c*, arquivo de cabeçalhos *server_function.h* e um arquivo de funções *server_function.c*. Criou-se um makefile (Figura 1), arquivo que consta as instruções de como gerar um binário, para facilitar a compilação dos arquivos na linguagem C e consequentemente a execução do servidor. O servidor está disponível em: https://github.com/Rodrigo-Segui/ServidorHTTP.

```
INDEX:
    gcc -c main.c -g
    gcc -c server_function.c -g
    gcc main.o server_function.o -o exec -g
    ./exec
```

Figura 1: Arquivo makefile.

4. Resultados e Discussões

Foi implementado somente o tratamento do método GET nessa versão. Também é possível servir requisições de um usuário por vez, sendo capaz de servir arquivos do tipo: pdf, txt, html, png, jpeg, jpg e gif. O controle da taxa de envio é delimitado por um valor fixo de 1000 kbps definido no código.

Não foi possível desenvolver o mecanismo de threads a tempo para a primeira entrega do projeto, também está para ser implementado o tratamento dos outros métodos além do método GET, servir a requisição de outros tipos de arquivos que um servidor HTTP normalmente faz e ainda é preciso implementar o uso do arquivo auxiliar no qual tem salvo alguns IPs com a taxa de envio, foi encontrada uma dificuldade relacionada à identificação IP do cliente.

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

➔ telnet localhost 8080
➔ GET /home/user/Documentos/github/ServidorHTTP/teste.txt HTTP/1.0

```
----- Servidor HTTP ------

Mensagem do Servidor | Status      : Online!
Mensagem do Servidor | Taxa Maxima : 1000
Mensagem do Servidor | Porta       : 8080
Recebeu conexão cliente: 0
Mensagem do Servidor | Cliente Conectado : 7
---------------------- Aguardando Requisição ----------------------
-
Requisição: GET /home/user/Documentos/github/ServidorHTTP/arquivos
/teste.txt HTTP/1.0
-> Método: GET
-> Arquivo: /home/user/Documentos/github/ServidorHTTP/arquivos/tes
te.txt
-> Protocolo: HTTP/1.0

-> Protocolo Aceito
-> Tipo Arquivo: txt
-> Tamanho: 10

Date: Mon, 26 Oct 2020 02:35:42 GMT
-----------------

Last-Modified: Sun, 09 Aug 4433987 05:53:07 GMT
-----------------
-> Dados:

HTTP/1.0 200 OK
Connection: close
Date: Mon, 26 Oct 2020 02:35:42 GMT
Server: SERVER 2020
Last-Modified: Sun, 09 Aug 4433987 05:53:07 GMT
Content-Length: 10
Content-Type: text/txt
```

Figura 2: Execução do servidor.

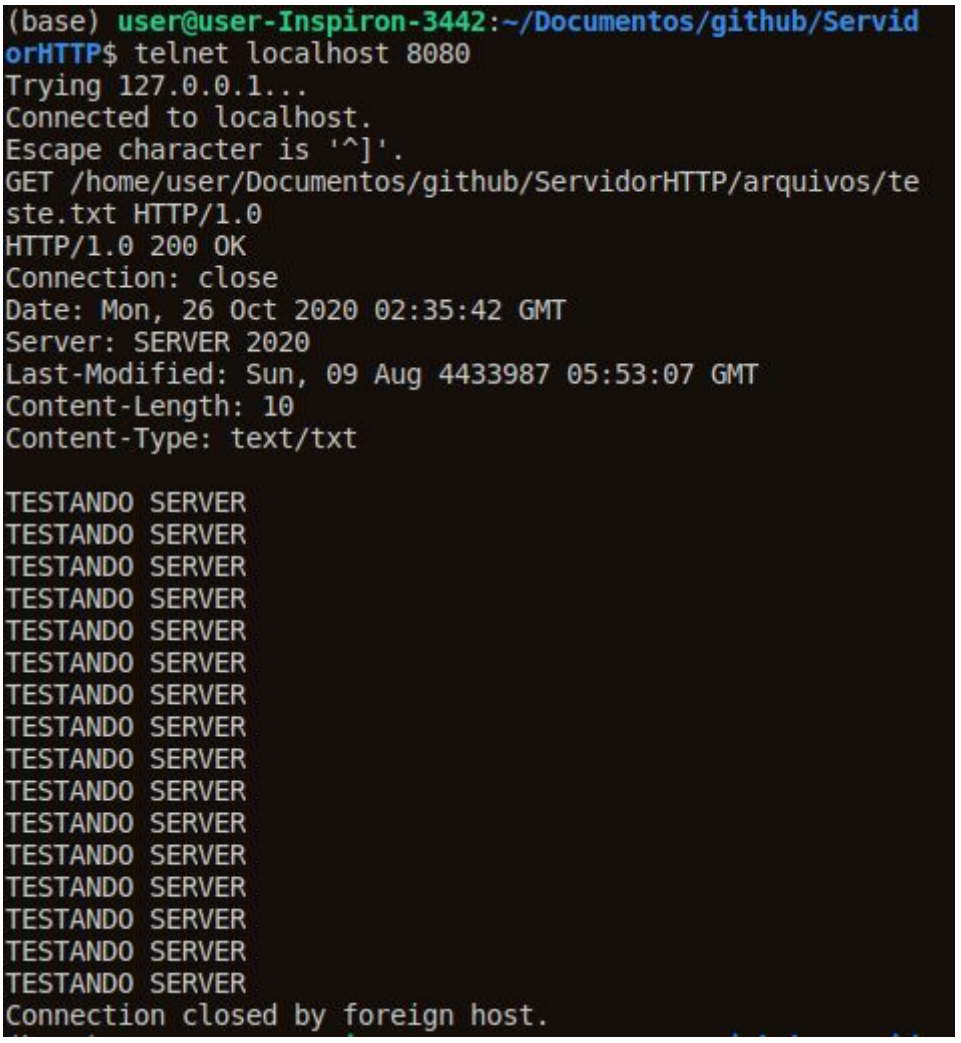

Figura 3: Execução do cliente.

6. Conclusão

Foi possível pôr 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. Mesmo faltando algumas funcionalidades previstas para a primeira entrega foi possível ter um resultado satisfatório em questão de aprendizagem melhorando o entendimento em alguns temas de redes de computadores. Os imprevistos que impossibilitam a implementação de algumas funcionalidades serão analisados com cuidado para que os mesmos não interfiram nas próximas etapas do projeto.

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.