Projeto Final - EA872

Caio Ruiz Coldebella, RA: 232621

11 de Dezembro de 2022

Este projeto foi realizado com o objetivo final de construir um servidor web, com arquivos e diretórios locais, para atender requisições HTTP de navegadores, realizando a comunicação com estes a partir de sockets que escutam uma porta TCP, no caso definida como 8080. Além disso, foram utilizadas threads, para responder múltiplas requisições paralelas, além de permitir o ganho de performance do servidor.

O sistema responde por requisições do tipo GET, HEAD, TRACE e OPTIONS, retorna para o navegador, e escreve no terminal a resposta gerada, de acordo com a RFC 2068, e exibe mensagem de erro quando o número de sockets atinge o seu valor máximo permitido, não podendo atender àquela requisição.

Para executar o programa, primeiro é necessário executar dois scripts: permissoes.sh e compilar.sh, que modificam as permissões dos arquivos e diretórios, e compilam o programa respectivamente. Após isso é necessário executar no terminal a linha de comando “./servidor /url arquivoresp.txt registro.txt”. Em que /url é o caminho local da pasta do diretório, arquivoresp é o arquivo dentro do diretório que irá salvar as respostas do servidor, e registro é o arquivo permanente que armazena o cabeçalho de todas as requisições feitas.

Fluxograma do sistema:

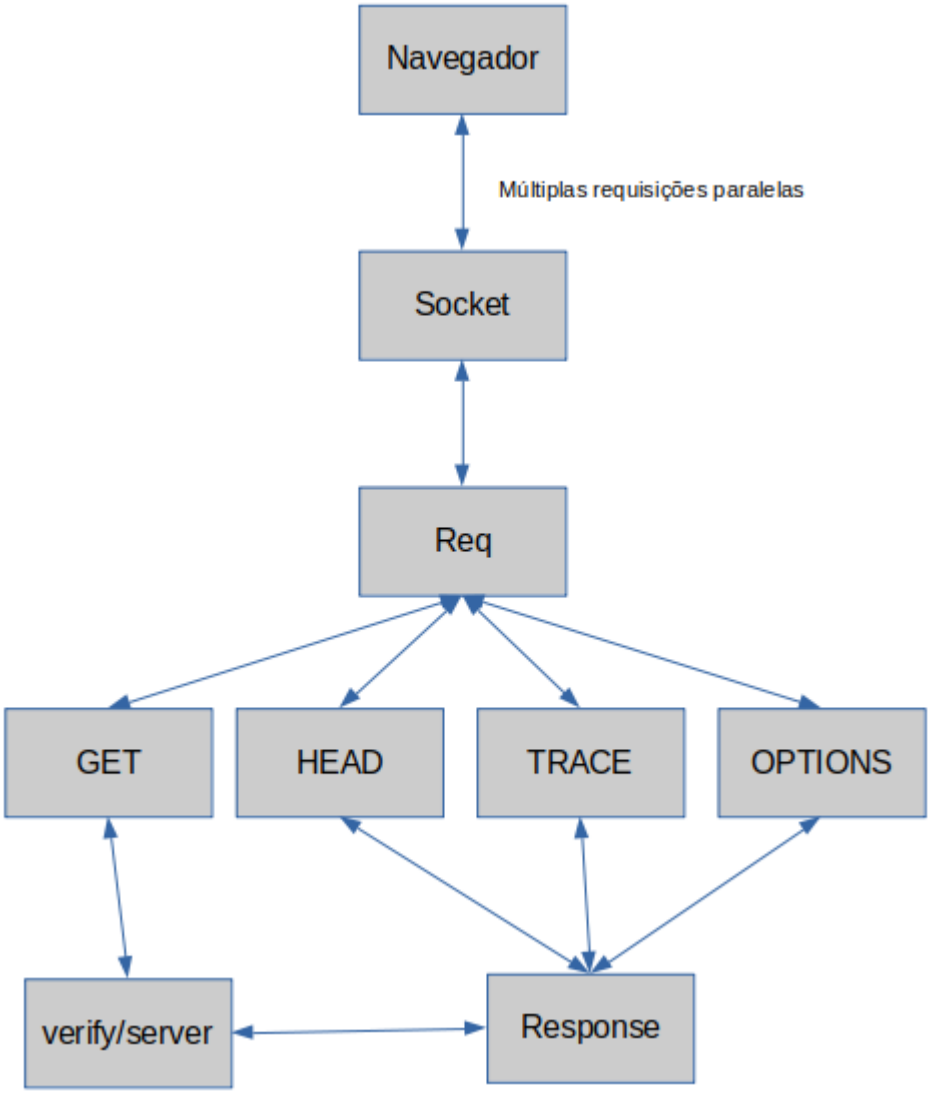


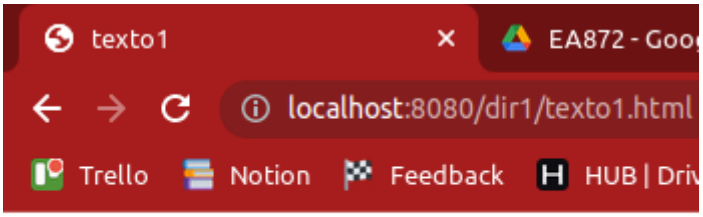
Figura 1: Fluxograma do sistema

Estrutura do webspace:

```
–
 |
 |-- index.html
 |
 |--dir1–
 |        |--texto1.html
 |        |
 |        |--dir11/
 |
 |--dir2/
```

Todos os diretórios e arquivos possuem permissão apenas de leitura

A partir de requisição GET pelo navegador obtive as seguintes respostas:



Esse é o texto 1

E a partir do http_load foram testadas múltiplas requisições paralelas, obtendo as seguintes respostas:

```
10 fetches, 2 max parallel, 818 bytes, in 0.003124 seconds
81.8 mean bytes/connection
msecs/connect: 0.1298 mean, 0.445 max, 0.019 min
msecs/first-response: 0.414 mean, 1.342 max, 0.154 min
9 bad byte counts
HTTP response codes:
  code 200 -- 1
```

```
10 fetches, 4 max parallel, 818 bytes, in 0.002841 seconds
81.8 mean bytes/connection
msecs/connect: 0.1136 mean, 0.241 max, 0.034 min
msecs/first-response: 0.5065 mean, 1.087 max, 0.195 min
1 bad byte counts
HTTP response codes:
  code 200 -- 1
```

Vemos que houveram erros de byte count wrong, estes ocorreram a partir da segunda requisição, o servidor só não obtém esse erro para a primeira requisição. Entretanto para os testes no navegador esse problema não foi enfrentado.

Portanto há essa limitação do programa quanto ao teste com o http_load, além de haver a limitação de velocidade e número de requisições do programa, pois a partir de um certo número de requisições o programa se encerra. Também há a limitação do teste do tipo da requisição do programa, pois é necessário um cliente externo para testar as requisições TRACE, OPTIONS, e HEAD.

Entre as futuras melhorias e trabalhos futuros podem ser citados a implementação de mais tipos de requisições como por exemplo POST, e a melhor investigação sobre como solucionar a questão dos erros “bytes count wrong”, que não foi possível de ser descoberto. Além disso, podem ser melhorados os arquivos do webspace, com páginas html mais adequadas.

Acerca dos comentários e críticas sobre a disciplina, acredito que uma das maiores dificuldades que os alunos tiveram que lidar foi com a falta de proporção entre o trabalho necessário para implementar coisas de redes e servidor comparada com o trabalho sobre sistemas operacionais, nesse sentido a disciplina acaba fugindo do seu objetivo e da sua ementa. Eu particularmente não tive tanta dificuldade com isso porque já havia cursado a disciplina de introdução à redes de computadores, mas essa não é pré-requisito de EA872, inclusive sendo recomendada de ser feita no semestre seguinte ao qual EA872 é recomendada. Além disso, acho que nas próximas disciplinas não seria uma boa ideia permitir aos alunos implementar os códigos em outras linguagens que não “C”, entendo que o professor fez isso de maneira a tentar ajudar os alunos e flexibilizar, mas percebi que os alunos que resolveram fazer isso tiveram mais dificuldade do que os que fizeram em C, porque a maioria das funções que mexem com threads são implementadas em C