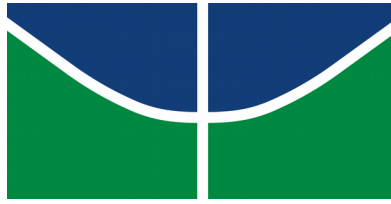


Universidade de Brasília



Trabalho de Teleinformática e Redes 2 Tomás Rosário Rosemberg 14/0087567

-Introdução Teórica

Proxy Server Web:

O Servidor de proxy web é um servidor (sistema ou aplicação) que atua Um Proxy Server Web é um servidor (sistema ou aplicação) que atua como um intermediário para requisições HTTP de clientes buscando recursos de servidores Web. O cliente se conecta ao Proxy Server requisitando um objeto de um servidor diferente e esse avalia a requisição, repassando-a para servidor de origem do objeto. Dessa forma, o Proxy Server se comporta como servidor para o cliente final e como cliente para o servidor de origem. As utilidades dessa tecnologia são várias, desde monitoramento e filtra- gem até caching e controle de acesso, porém este trabalho trata apenas de monitorar e controlar o tráfego, podendo editar o que é enviado/recebido.

Um servidor proxy pode, opcionalmente, alterar a requisição do cliente ou a resposta do servidor e, algumas vezes, pode disponibilizar este recurso mesmo sem se conectar ao servidor especificado. Pode também atuar como um servidor que armazena dados em forma de cache em redes de computadores. São instalados em máquinas com ligações tipicamente superiores às dos clientes e com poder de armazenamento elevado.

Neste trabalho foi implementado um servidor de proxy para ser possível analisar as requisições HTTP enviadas do cliente ao servidor e também de volta, para podemos entender melhor como funciona as chamadas HTTP e deveríamos, também, tornar possível a interceptação das requisições HTTP para caso fosse desejável, alterar algum valor nela.

TCP

TCP, ou seja protocolo de controle de transmissão, é um dos principais protocolos no qual a internet se baseia, por ser complementado pelo protocolo IP de internete, popularmente é chamado de TCP/IP. O protocolo TCP se baseia no principio da ordenação da informação enviada, no three way handshake para se ter confiabilidade de que o envio esta sendo para o local correto e confiabilidade, podendo ser usando varias tecnicas diferentes. Um exemplo de tecnica para garantir a confiabilidade é a Go-Back-N, para que desta forma ele receba ACKS como retorno das informações enviadas, garantindo assim que a informação foi recebida.

HTTP

O HyperText Transfer Protocol (HTTP) está no coração da Web e é responsável por definir a estrutura das mensagens trocadas entre cliente e servidor e como elas são trocadas. Essas mensagens podem ser resumidas a páginas Web, que consistem de objetos (um arquivo HTML, uma imagem, uma folha de estilo CSS ou um arquivo de áudio/vídeo) endereçados por uma Uniform Resource Locator (URL), que por sua vez é dividida em hostname do servidor e nome do caminho do objeto. Os navegadores Web implementam o cliente HTTP, cujo processo inicia uma conexão TCP com o servidor na porta 80, por padrão, e então envia uma requisição contendo o método (GET, POST, HEAD, PUT e DELETE), a URL e a versão do HTTP na primeira linha seguida pelas linhas do cabeçalho, todas separadas por uma quebra de linha (\r\n). A resposta tem uma linha de status como primeira linha, contendo a versão do HTTP, o código e a frase do status, seguida pelas linhas do cabeçalho, uma linha em branco e por fim o corpo da entidade. Vale lembrar que se trata de um protocolo stateless, ou seja, o servidor não mantém informação sobre clientes antigos, além de que a conexão utilizada por padrão é persistente, fazendo com que o servidor deixe a conexão TCP aberta após enviar o objeto para caso o mesmo cliente tenha mais requisições fazer.

Uma sessão HTTP é uma sequência de transações de rede de requisição-resposta. Um cliente HTTP inicia uma requisição estabelecendo uma conexão Transmission Control Protocol (TCP) para uma porta particular de um servidor (normalmente a porta 80. Veja isto de portas dos protocolos TCP e UDP). Um servidor HTTP ouvindo naquela porta espera por uma mensagem de requisição de cliente. Recebendo a requisição, o servidor retorna uma linha de estado, como "HTTP/1.1 200 OK", e uma mensagem particular própria. O corpo desta mensagem normalmente é o recurso solicitado, apesar de uma mensagem de erro ou outra informação também poder ser retornada.

Spider

Um spider ou Web crawler é um rastreador de páginas muito utilizado para indexação Web que sistematicamente identifica todos os links de uma página e os adiciona a uma lista de URLs a serem visitadas recursivamente. No contexto deste trabalho, por fins de simplicidade, o resultado será uma árvore de páginas HTML contidas no host da página raiz e que podem ser acessadas a partir da mesma. A parte de dump fica a cargo do cliente recursivo que salvará uma cópia local dos objetos acessíveis pela árvore gerada, se limitando a arquivos HTML e imagens porém mantendo uma estrutura de diretórios semelhante à do servidor remoto.

-Arquitetura do Sistema

O Sistema feito se baseia na orientação ao objeto utilizando c++, desta forma criamos grandes classes e separamos muito bem suas responsabilidades para que fosse possível obter o melhor desempenho e facilidade de programação acerta do trabalho.

Foi criada um classe Sockets, onde nela possuem apenas as funções responsáveis por pelos sockets, onde se cria as sockets, se conectam, aceitam as conexão, se le o resultado dos sockes e se verifica se esta tudo certo com os mesmos.

Foi criada um classe Servidor, responsavel por rodar todo o codigo.

Foi criada uma classe HTTP, onde se é responsavel por interpretar o HTTP, tanto enviado como recebido, e por ultimo uma classe CRAWLER, onde se foi implementado o spider primitivo, que consegue apenas algumas paginas com base no href dos html e a geração local da página da web requisitada e suas sucessoras com base no href dentro dela.

-Documentação

A documentação do código se encontra no próprio, o qual pode ser encontrado no repositório do Github https://github.com/trosemberg/TR2_Trab

-Funcionamento

O programa funciona utilizando o comando `./aracne`, podendo também selecionar a porta a ser utilizada, conforme foi requisitado pelo professor nas especificações. Depois disto deve-se fazer uma requisição no Browser para que o programa a intercepte e possamos ver o formato dela e também escolher entre as funcionalidades disponíveis, que é a de utilizar o spider ou inspetor HTTP.

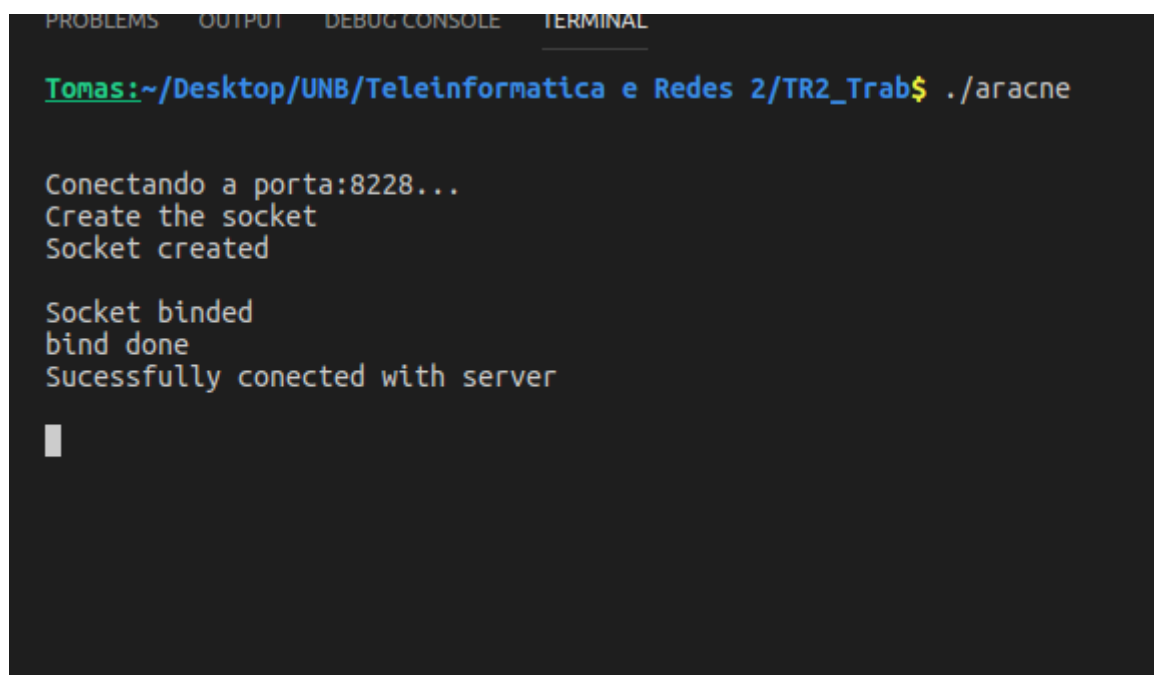
Caso escolha a opção de utilizar o inspetor de HTTP, precisa-se selecionar ele diversas vezes para que se complete todas as requisições e se possa visualizar o site por completo.

Caso escolha spider, ele gerará os arquivos locais e a árvore hipertextual até um máximo de profundidade 3.

Primeiro o programa nos forneceria uma mensagem de que a requisição foi interceptada, desta forma caso queira alterar algo nela, basta abrir o arquivo “request”, onde estará localizada a requisição, e após alterar algo, salvar o arquivo e escrever qualquer coisa + enter no terminal.

Então apareceria as opções previamente citadas, onde ao escolhermos spider, o programa gera o arquivo local da request, e lê os links dentro deste arquivo local para fazer novas requests e salvar um arquivo local com base nessas requests até um máximo de profundidade de 3.

Ao escolhermos o inspetor HTTP, veremos mensagens relacionadas ao request como amarelo, relacionados ao response em azul, desta forma facilita a identificação do que é request e do que é response. Se é gerado um arquivo response que seria responsável pela possibilidade de edição da response, porém houve problemas ao interpretá-lo como binário o que gerava problema no proxy pois o programa não continuava as requisições.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Tomas:~/Desktop/UNB/Teleinformatica e Redes 2/TR2_Trab$ ./aracne

Conectando a porta:8228...
Create the socket
Socket created

Socket binded
bind done
Sucessfully conected with server

█
```

Figura 1 – Mensagem inicial

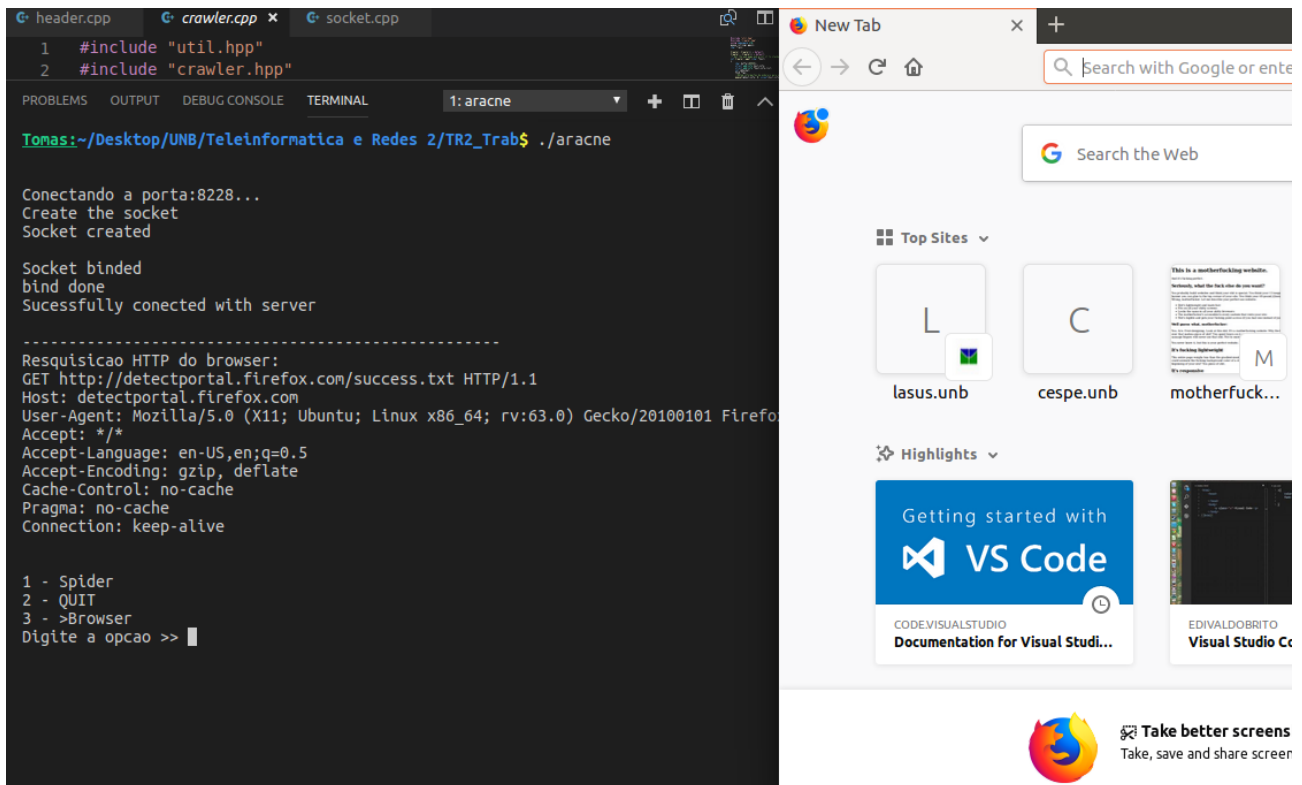


Figura 2 – Tela inicial ao abrir o navegador

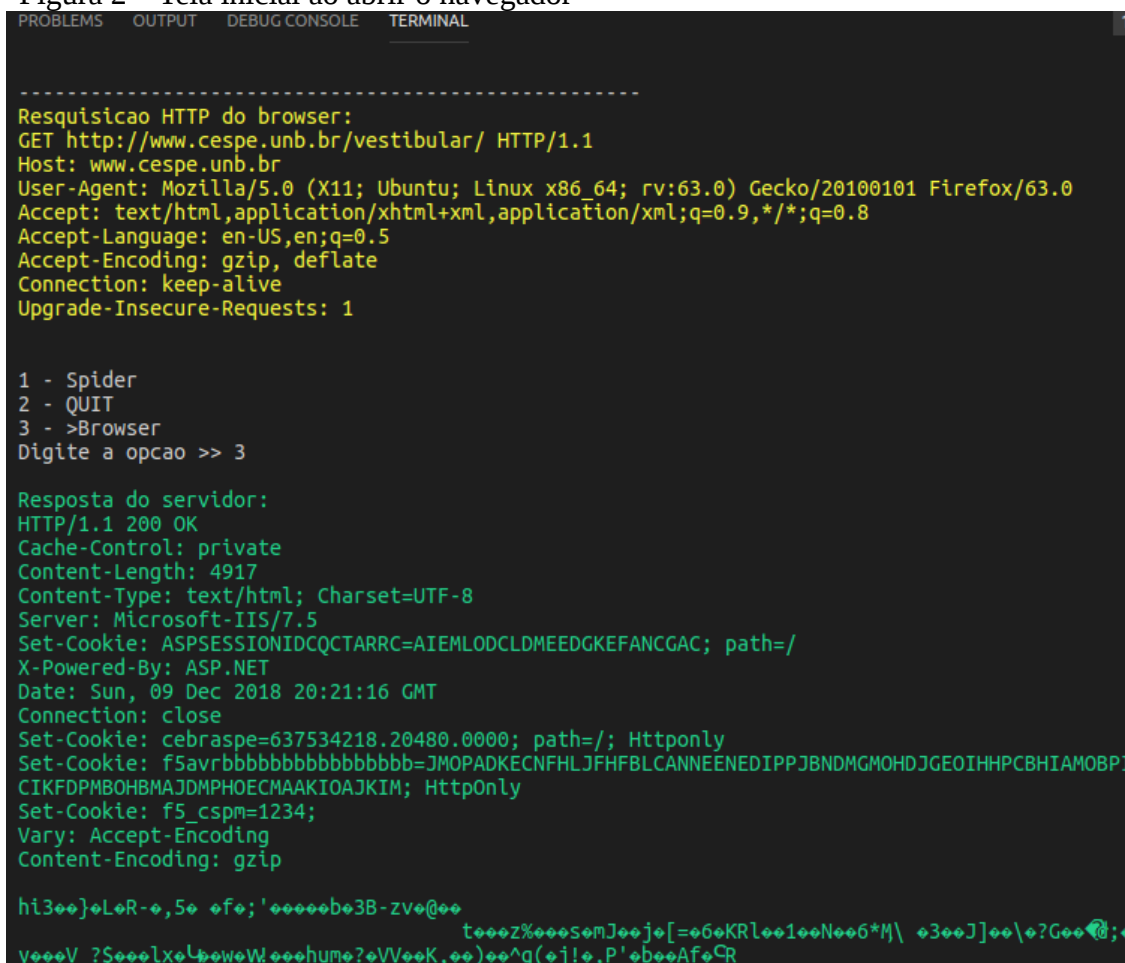


Figura 3 – Fazendo requisição da pagina, diferenciando requisição e resposta

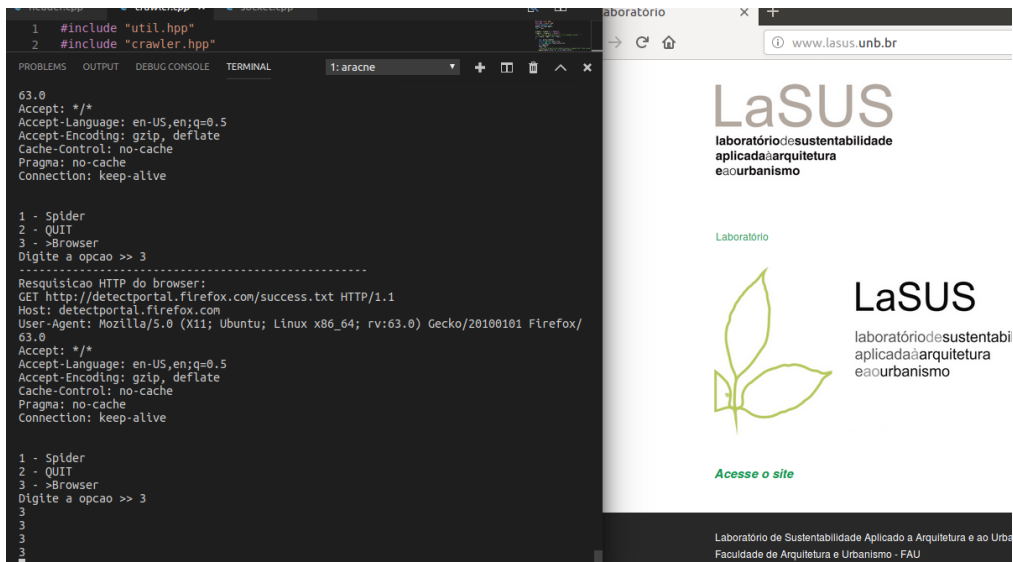


Figura 4 - após conseguir abrir o site

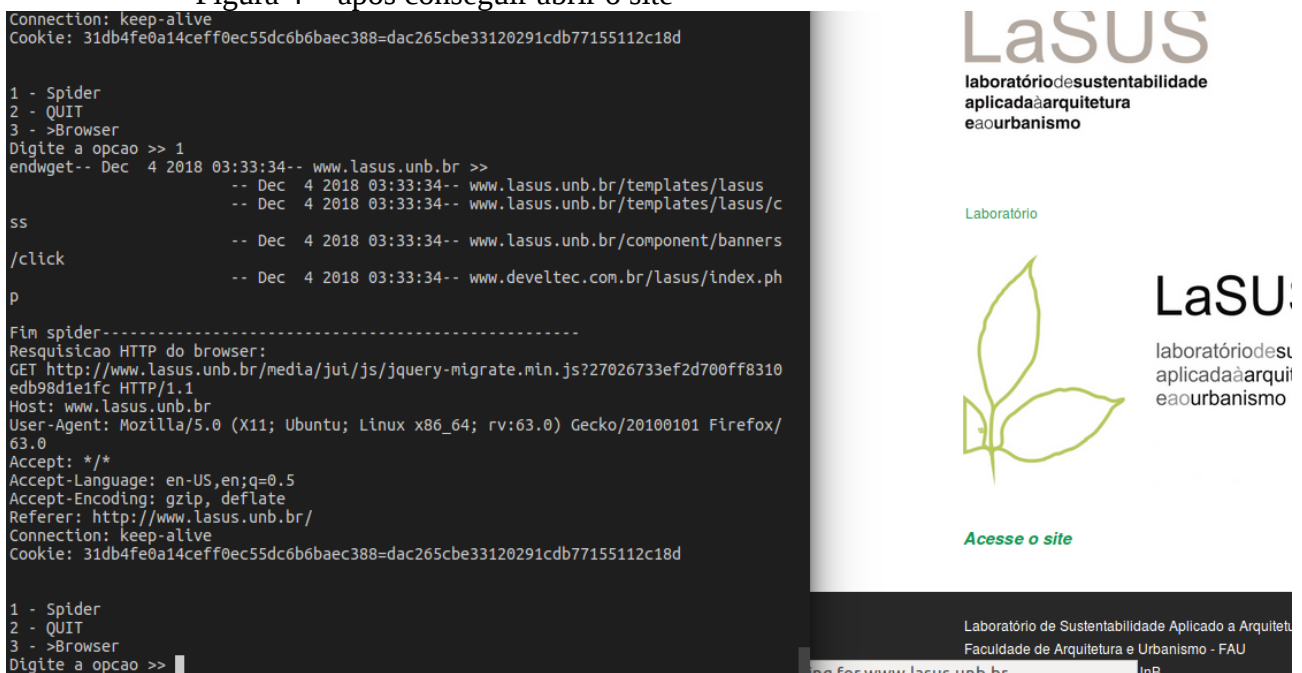


Figura 5 – executando o spider.

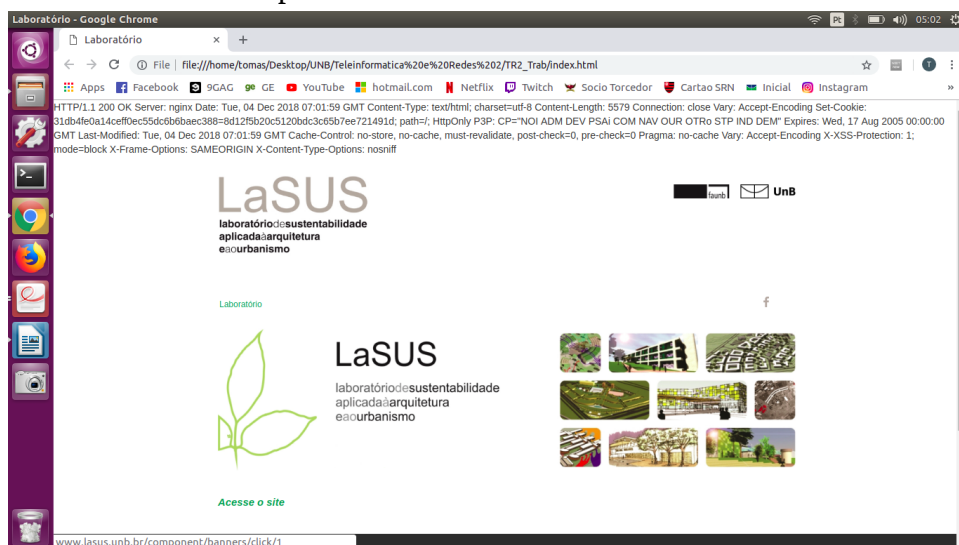


Figura 6 - Arquivo local gerado.

Conclusão

As partes possíveis de fazer do trabalho foi interceptação das http requests editando-as e fazer o proxy delas, também foi possível interceptar as responses mas não foi possível editá-las.

Na parte de proxy, não foi possível baixar as imagens, nem fazer a interação local para navegar localmente entre as páginas, mas foi possível fazer um árvore textual de tamanho limitado e baixar dos links que funcionavam seus arquivos locais.

A parte de interface gráfica se baseou basicamente no terminal e partes do texto teriam cores específicas para facilitar o manuseio do programa. Desta forma o texto das requests seriam amarelos, das responses azul e da árvore sintática magenta, para facilitar a visualização de todos eles.

Bibliografia e Referencias

- [1] James Kurose e Keith Ross. Computer Networking. A Top-Down Approach. 7a ed. Pearson, 2017. Cap. 2.
- [2] Proxy server. Wikimedia Foundation, Inc. Jul. de 2018. url: [https :
//en.wikipedia.org/wiki/Proxy_server](https://en.wikipedia.org/wiki/Proxy_server) (acedido em 09/07/2018).
- [3] Web crawler. Wikimedia Foundation, Inc. Jun. de 2018. url: [https:
//en.wikipedia.org/wiki/Web_crawler](https://en.wikipedia.org/wiki/Web_crawler) (acedido em 09/07/2018).
- [4] Wget. Recursive download. Wikimedia Foundation, Inc. Jun. de 2018. url: https://en.wikipedia.org/wiki/Wget#Recursive_download (acedido em 09/07/2018).
- [5]<https://github.com/RobertN/HTTProxy>
- [6]<https://github.com/sameer2800/HTTP-PROXY>
- [7]<https://www.youtube.com/watch?v=LtXEMwSG5-8>
- [8]<https://www.youtube.com/watch?v=mStnzIEprH8&feature=youtu.be>
- [9]<https://gist.github.com/berlinbrown/4583728>
- [10]<http://professor.ufabc.edu.br/~francisco.massetto/sd/03-Aula3-Sockets.pdf>
- [11]https://www.example-code.com/cpp/spider_simpleCrawler.asp