

Um Inspetor HTTP baseado em Proxy Server

1 Objetivo.

O objetivo desse trabalho é desenvolver um inspetor HTTP baseado em Proxy Server, aplicando o conteúdo aprendido em sala de aula sobre camada de aplicação e transporte. O servidor deverá receber requisições HTTP processando seu conteúdo de cabeçalho para verificação do endereço de destino. Deve ser utilizado um navegador web, devidamente configurado para o uso do proxy. O sistema também deve implementar duas outras funcionalidades, um Spider e um cliente recursivo. O Spider gera uma árvore hipertextual com as URLs obtidas do objeto e o cliente recursivo salva o conteúdo de uma URL.

2 Introdução.

Ao estudarmos a camada de aplicação e os protocolos de rede que existem, dois protocolos muito importantes não podem ser ignorados, dado a abrangente frequência de utilização deles na internet, esta que é usada por todos. São os protocolos HTTP, que é usado na camada de aplicação, e o protocolo TCP, que é utilizado junto ao protocolo HTTP, na camada de transporte. Uma comunicação feita entre um browser cliente e um servidor que armazena as páginas de internet será realizada com estes protocolos, e, então, estudá-los é essencial para que seja possível compreender e criar qualquer aplicação em uma rede que propõe analisar e armazenar páginas que estão disponíveis na internet, uma aplicação de exemplo é o caso de um servidor proxy.

2.1 Proxy Server Web.

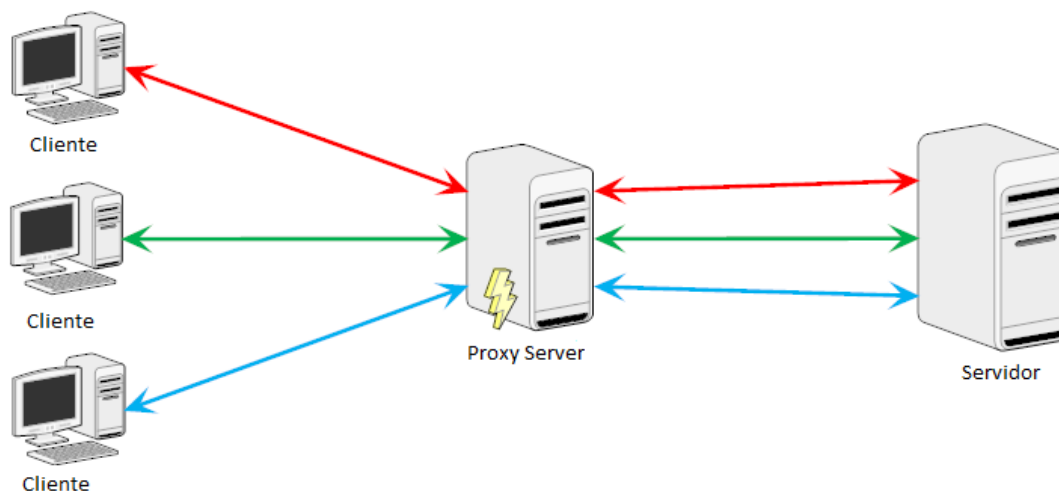


Imagem1: <https://github.com/ArashPartow/proxy> (alterado).

Um dos objetivos desse trabalho é a implementação de um servidor proxy, mas o que é um servidor proxy? Um servidor proxy, que também pode ser chamado de **cache Web**, é uma funcionalidade da rede que recebe requisições HTTP de um servidor Web. O proxy tem sua própria memória, e o mantém, junto dele, cópias dos objetos que foram requisitados. O browser

de um usuário pode ser configurado de uma maneira que suas requisições HTTP sejam direcionadas primeiro ao proxy, uma vez feito isso, cada uma das requisições de um objeto que o browser faz é direcionada primeiro ao proxy server.

Funcionamento:

- 1. O browser faz uma conexão com o servidor proxy e envia a ele uma requisição HTTP para o objeto.*
- 2. O servidor proxy realiza uma verificação se já existe uma cópia do objeto armazenada localmente. Se existir, envia o objeto ao browser do cliente, dentro de um response HTTP.*
- 3. Caso o objeto não exista, o servidor proxy abre uma conexão com servidor de origem, enviando uma requisição HTTP do objeto para conexão. Recebendo, o servidor de origem envia o objeto ao servidor proxy, dentro de uma resposta HTTP.*
- 4. O servidor proxy, armazena o objeto localmente e envia uma cópia dentro de uma mensagem de resposta HTTP, ao browser do cliente (Pela conexão já estabelecida entre o browser do cliente e o servidor proxy).*

Um exemplo de utilização de um proxy é que ele pode ser filtrar e bloquear requisições enviadas. É possível definir que toda requisição de uma máquina deve passar por um proxy local. Esse, por sua vez, impede que requisições HTTP, que sejam destinados a um host específico cheguem ao servidor.

2.2 Protocolo TCP.

O protocolo TCP, é um protocolo da camada de transporte, usado pelo protocolo HTTP para enviar suas mensagens. O TCP é ideal para escolha de uso junto ao HTTP diante do fato que ele é um protocolo que garante transferência confiável de dados, garante que a entrega de todos os pacotes irá ocorrer em ordem e não vai sofrer com perdas. Para uma aplicação que tem o propósito de enviar e receber objetos, é fundamental garantir a entrega total da mensagem.

O protocolo TCP é um protocolo orientado a conexão, um protocolo que faz os processos se comunicarem realizando handshake (troca de mensagens para estabelecer apenas a conexão)

Uma das funções mais importantes do protocolo TCP é realizar a multiplexação e demultiplexação de mensagens sendo enviadas para ou por processos executando em um hospedeiro. A camada de transporte tem como principal função realizar a comunicação lógica entre as aplicações, ou seja, realizar a comunicação entre processos executando em hospedeiros diferentes de tal forma que tal comunicação nunca seja de conhecimento da aplicação.

O protocolo TCP também possui um controle de congestionamento, sendo um serviço dirigido a internet como um todo, o controle de congestionamento do TCP, impede que qualquer outra conexão TCP atrapalhe os enlaces e roteadores entre os hospedeiros que estão com uma quantidade excessiva de tráfego. O TCP permite igualdade de banda entre as conexões daquele enlace, feito pela regulação da taxa de envio de tráfego do remetente.

2.3 Protocolo HTTP.

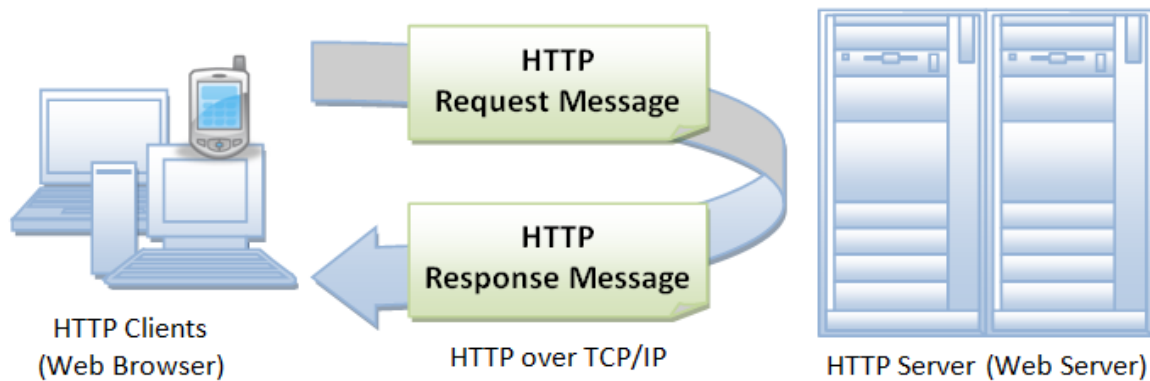


Imagem2:http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/http_basics.html

HTTP é a uma sigla usada para representar o protocolo que existe na camada de aplicação que é chamado de *Hypertext Transfer Protocol*. Protocolo de comunicação utilizado para realizar a comunicação entre servidor e cliente conectados na WEB, eles conversam entre si por meio da troca de mensagens HTTP. O Protocolo HTTP define o funcionamento dessas mensagens e o modo de como estas são trocadas.

O HTTP define como os clientes requisitam as páginas aos servidores e como estas são transferidas para o cliente utilizando protocolo da camada de transporte TCP.

- Comportamento de requisição-resposta do HTTP:



Imagem3: redes de computadores e a internet KUROSE

O cliente primeiro inicia uma conexão TCP com o servidor, com a conexão estabelecida, os processos entre o browser e o do servidor acessam o TCP por meio de suas interfaces *socket*. No lado do cliente essa interface *socket* é a porta do processo cliente e a conexão TCP, do lado do servidor ela é a porta entre o processo servidor e a conexão TCP. O cliente envia para o servidor mensagens de requisição (request HTTP) para interface *socket*, que responde para o cliente (response HTTP) que recebe em sua interface *socket*. De uma maneira semelhante, o servidor HTTP recebe mensagens de requisição de sua interface e envia mensagens de response para sua interface. Assim que o cliente envia uma mensagem, a mensagem sai de suas mãos e passa para a do TCP, que é responsável por entregar a mensagem ao destino servidor de maneira intacta. Do mesmo modo que toda mensagem no processo servidor também chegará intacta ao cliente. Podemos ver nesse ponto as vantagens de se possuir uma arquitetura baseada em camadas. Ao utilizar o protocolo TCP da camada de transporte, o HTTP não precisa se preocupar com dados perdidos ou com detalhes de como o TCP se recupera da perda de dados ou os organiza dentro da rede.

O protocolo HTTP é denominado um protocolo sem estado, pois não mantém informação alguma sobre os clientes, caso um determinado cliente solicita o mesmo objeto duas vezes em poucos segundos, o servidor não avisa que acabou de enviar, ele simplesmente manda de novo o objeto, pois já se esqueceu completamente o que havia feito antes.

O HTTP possui várias mensagens de requisição, onde a mais comum envolve o método GET, utilizado para recuperar o conteúdo de uma página web ou de um de seus objetos.

Exemplo de mensagem de requisição HTTP:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Imagem4: redes de computadores e a internet KUROSE.

Vamos entender esta requisição, a primeira linha está pedindo (GET) que o servidor HTTP envie o objeto **/somedir/page.html**, um documento escrito em html, e logo após temos a versão HTTP que está sendo utilizada nesta comunicação **HTTP/1.1**. na linha seguinte, está descrito o nome do servidor que está hospedando esse objeto requisitado, www.someschool.edu, que evidentemente será traduzida para o endereço IP que o representa. Logo após na linha 3, está definido no campo que a conexão realizada para recuperação não será persistente. A próxima linha, define o campo de cabeçalho, que diz qual o tipo do agente que está realizando a requisição, um navegador **Mozilla/5.0**, e a ultima linha, temos um campo que define que se a página possuir uma versão em francês esta versão é que deve ser enviada preferencialmente.

Exemplo de mensagem de resposta HTTP:

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
```

Imagem5: redes de computadores e a internet KUROSE

Nesta imagem perceba que a mensagem de resposta possui um formato semelhante à de requisição: contendo o estado da resposta enviada pelo servidor, indicando se a requisição foi aceita corretamente. Temos a versão do protocolo que foi utilizada novamente e o código de status com mensagem correspondente a tal código (200 OK). Temos diversos campos de cabeçalho seguidos de seus respectivos atributos, e finalmente o **payload**, da mensagem que, para o exemplo, irá conter o objeto requisitado.

3 Arquitetura desenvolvida.

Para realização deste trabalho, foi escolhido para implementação a linguagem C++. Os sistemas implementados foram: Servidor Proxy, Spider, Cliente recursivo (dump) e um manipulador de string funcionalidade extra que facilitou a implementação das funcionalidades.

3.1 Servidor Proxy.

Servidor proxy simples que age sobre a arquitetura cliente e servidor, o proxy implementado, portanto, pode receber diversas requisições HTTP vindas de diversos clientes que foram configurados para utiliza-lo. Estas requisições são filtradas pelo proxy e após isso repassadas para seus devidos servidores HTTP pelo próprio proxy, que neste caso após receber a requisição age como cliente na hora de verificar se a requisição pode ser atendida, assim podendo retornar o objeto para o cliente que solicitou a requisição.

Classe Proxy_Server:

A classe Proxy_Server implementa toda a lógica de Sockets do servidor, para receber requisições e enviar respostas ao navegador e para realizar requisições HTTP à internet e receber resposta.

Instancia e dando init para escutar na porta:

```
int porta = 8228;
Proxy_Server proxy = Proxy_Server()
proxy.init(porta);
```

Pegando requests vindos do navegador:

```
string req = proxy.get_client_request();
```

Fazendo a request para o servidor de origem e obtendo a resposta:

```
string reply = proxy.make_request(req);
```

Respondendo o navegador:

```
proxy.reply_client(response.Assembly_Response());
```

3.2 Spider e Cliente recursivo.

O Spider é uma funcionalidade feita para enumerar as URLs contidas em um site formando uma árvore hipertextual.

A funcionalidade cliente recursivo faz um dump de todo conteúdo de um site a partir de um URL específica e salva.

Classe Spider

Classe que se utiliza de todas as outras para realizar a função de cliente recursivo (dump de páginas subjacentes a uma url) e spider (construção de uma árvore hipertextual de links entre páginas a partir de uma url).

Instancia passando a url desejada :

```
string url("www.ba.gov.br");  
Spider spider = Spider(url);
```

Caso queira utilizar a url de uma request:

```
HTTP_Request request = HTTP_Request(req);  
Spider spider = Spider(request.url);
```

Verifica se a url passada é valida:

```
if(spider.valid){  
    // passou  
}else{  
    // não passou  
}
```

Gera arvore passando os numero de niveis:

```
spider.generate_tree(0);
```

ps: nível 0 só inspeciona a raiz.

3.3 HTTP_Request

Classe HTTP_Request

Classe que faz o parse da requisição HTTP e torna possível tratamentos e manipulações.

Instanciando uma Request a partir de uma string:

```
HTTP_Request request = HTTP_Request(req);
```

Acessando os parametros da Request:

```
request.method = "GET";  
request.url = "/exemplo/";  
request.version = "HTTP/1.1";  
request.fields["Host:"] = "www.host.com";
```

Monta a Request de volta para um string:

```
string req = request.assembly();
```

3.4 HTTP_Response

Classe HTTP_Response

Classe que faz o parse da resposta HTTP e torna possível tratamentos e manipulações.

Instanciando uma response a partir de uma string:

```
HTTP_Response response = HTTP_Response(resp);
```

Acessando os parametros da response:

```
response.status_code="200 OK";  
response.version = "HTTP/1.1";  
response.fields["Content-Type:"] ="text/html; charset=UTF-8" ;  
response.data = [algum html ou outro payload da resposta];
```

Monta a Response de volta para um string:

```
string res = response.assembly();
```

3.5 Extra: Manipulador de strings como facilitador.

Manipulador de String – Facilita o parser e a manipulação nos códigos.

3.6 Como executar o programa:

git clone <https://github.com/siqueiralex/aracne>

```
cd aracne
make
./aracne <port>
```

3.7 Problemas durante implementação.

Programação baixo nível em sockets torna o funcionamento do programa instável por existirem muitos detalhes a serem considerados. (Nem sempre funciona corretamente) - Complexidade de fazer o parse HTTP e HTML, funciona pra maioria dos casos, mais existem muitas situações em que se comporta mal. -Não conseguimos criar a interface gráfica devido à grande quantidade de problemas encontrados na lógica do programa. - Muitas funções blocantes que exigem conhecimento de Programação Concorrente para implementação de Threads (não foi feito).

4 Documentação do código desenvolvido.

Código documentado encontrasse no GitHub:

<https://github.com/siqueiralex/aracne>

5 Funcionamento do programa.

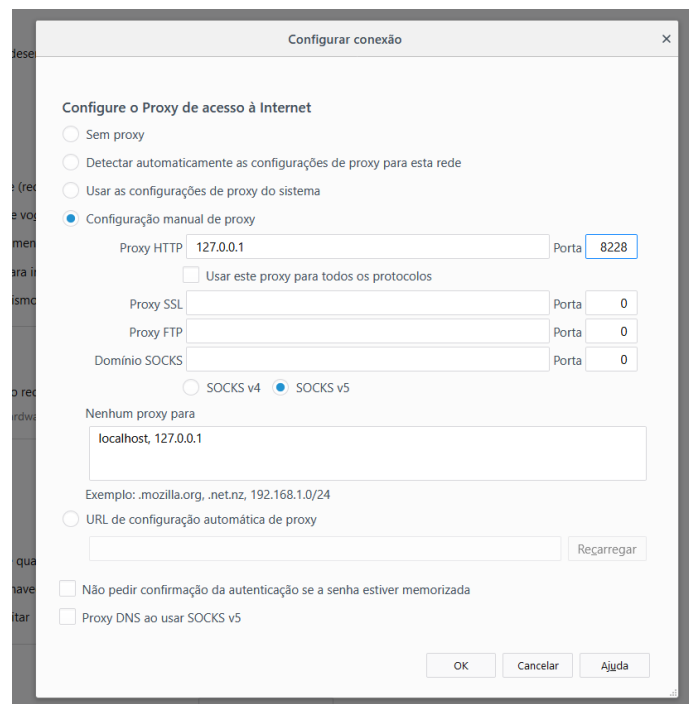
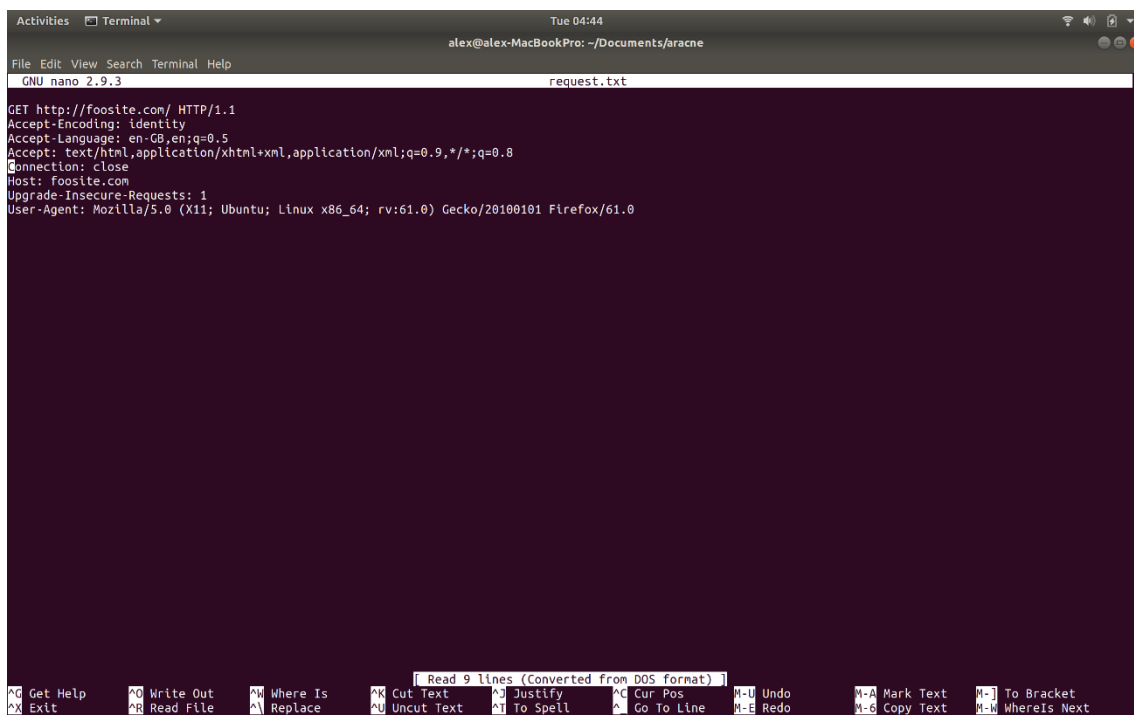


Figura 1 Configurar cliente com a porta desejada.


```
File Edit View Search Terminal Help
(1) Traffic Inspector
(2) Generate Tree of Url's
(3) Dump website
(4) Exit
Select your option ->> █
```

Figura 2: Ao rodar o programa, selecionar a opção desejada



The screenshot shows a terminal window titled 'alex@alex-MacBookPro: ~/Documents/arcane'. Inside, the GNU nano 2.9.3 editor is open, editing a file named 'request.txt'. The file contains an HTTP GET request to 'http://foosite.com/'. The request headers include 'Accept-Encoding: identity', 'Accept-Language: en-GB,en;q=0.5', 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8', 'Connection: close', 'Host: foosite.com', 'Upgrade-Insecure-Requests: 1', and 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0'. The nano editor's status bar at the bottom shows 'Read 9 lines (Converted from DOS format)' and various keyboard shortcuts for editing and navigation.

```
Activities  Terminal  Tue 04:44
alex@alex-MacBookPro: ~/Documents/arcane
GNU nano 2.9.3 request.txt
GET http://foosite.com/ HTTP/1.1
Accept-Encoding: identity
Accept-Language: en-GB,en;q=0.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Connection: close
Host: foosite.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0

Read 9 lines (Converted from DOS format)
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    ^U Undo       ^-A Mark Text  ^-] To Bracket
^X Exit      ^R Read File  ^I Replace    ^_ Uncut Text ^T To Spell   ^G Go To Line  ^-E Redo      ^-C Copy Text ^-_ WhereIs Next
```

Figura 3: Traffic Inspector - Após requisição do browser

```
Activities Terminal Tue 04:44
alex@alex-MacBookPro: ~/Documents/paracne
File Edit View Search Terminal Help
GNU nano 2.9.3 response.txt

HTTP/1.1 200 OK
Date: Tue, 10 Jul 2018 07:44:15 GMT
Server: Apache
Last-Modified: Fri, 29 Jun 2018 12:01:59 GMT
ETag: "3936-56fc69d7496b5"
Accept-Ranges: bytes
Content-Length: 14646
Vary: Accept-Encoding,User-Agent
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
  <!-- Site made with Mobirise Website Builder v3.11.1, https://mobirise.com -->
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="generator" content="Mobirise v3.11.1, mobirise.com">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="shortcut icon" href="assets/images/foosite-144x144-128x128-1.png" type="image/x-icon">
  <meta name="description" content="One webmaster's experimental playground. I've already wrecked this site a few times, this is the latest incarnation.">
  <title>foo site</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic&subset=latin">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Montserrat:400,700">
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Raleway:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i">
  <link rel="stylesheet" href="assets/bootstrap-material-design-font/css/material.css">
  <link rel="stylesheet" href="assets/tether/tether.min.css">
  <link rel="stylesheet" href="assets/bootstrap/css/bootstrap.min.css">
  <link rel="stylesheet" href="assets/soundcloud-plugin/style.css">
  <link rel="stylesheet" href="assets/dropdown/css/style.css">
  <link rel="stylesheet" href="assets/animate.css/animate.min.css">
  <link rel="stylesheet" href="assets/socicon/css/styles.css">
  <link rel="stylesheet" href="assets/theme/css/style.css">
  <link rel="stylesheet" href="assets/mobirise/css/mbr-additional.css" type="text/css">

</head>
<body>
<section id="menu-0">
  Read 195 lines (Converted from DOS format)
  ^C Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^U Undo ^M-A Mark Text ^M-J To Bracket
  ^X Exit ^R Read File ^N Replace ^L Uncut Text ^T To Spell ^A Go To Line ^M-E Redo ^M-B Copy Text ^M-W WhereIs Next
```

Figura 4 Traffic Inspector: Response antes de enviar parar o cliente

```
Inspecting /node/859
Inspecting /noticias
Inspecting /noticias/detran-reforca-fiscalizacao-no-sao-joao-de-cruz-das-almas-e-santo-antonio-de-jesu
Inspecting /noticias/elba-ramalho-levanta-publico-com-grandes-sucessos-do-forro-no-pelourinho
Inspecting /noticias/governo-entrega-nova-ciclovia-com-12-km-de-extensao-na-paralela
Inspecting /noticias/milhares-de-pessoas-se-reunem-na-festa-da-independencia-da-bahia
Inspecting /servidor
Inspecting /sitemap
Inspecting /themes/secom/favicon.ico
Inspecting /turismo
Saving file: index.html
Saving file: cidadao.html
Saving file: galeria-multimidia.html
Saving file: investidor.html
Saving file: node232.html
Saving file: node859.html
Saving file: noticias.html
Saving file: noticiasdetran-reforca-fiscalizacao-no-sao-joao-de-cruz-das-almas-e-santo-antonio-de-jesu
Saving file: noticiaselba-ramalho-levanta-publico-com-grandes-sucessos-do-forro-no-pelourinho.html
Saving file: noticiasgoverno-entrega-nova-ciclovia-com-12-km-de-extensao-na-paralela.html
Saving file: noticiasmilhares-de-pessoas-se-reunem-na-festa-da-independencia-da-bahia.html
Saving file: servidor.html
Saving file: sitemap.html
Saving file: themessecomfavicon.ico
Saving file: turismo.html
Saved files in a new directory named www.ba.gov.br
File "index.html" is the root.
Press any key to go back to main menu...
```

Figura 5 - Opção dump, passando como parâmetro uma URL, arquivos sendo baixados.

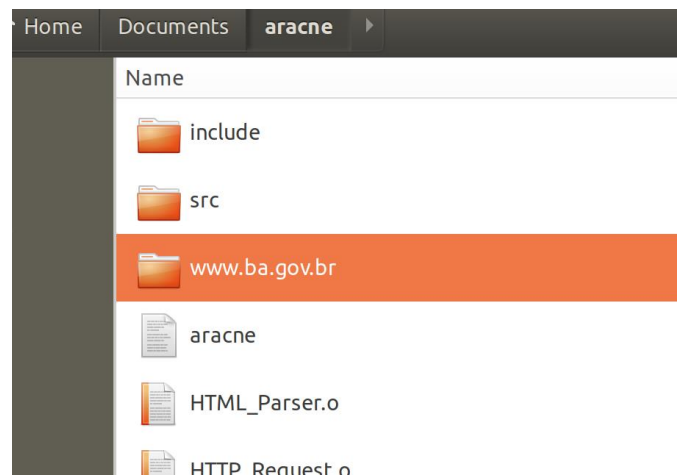


Figura 6: Arquivo criado após o dump.

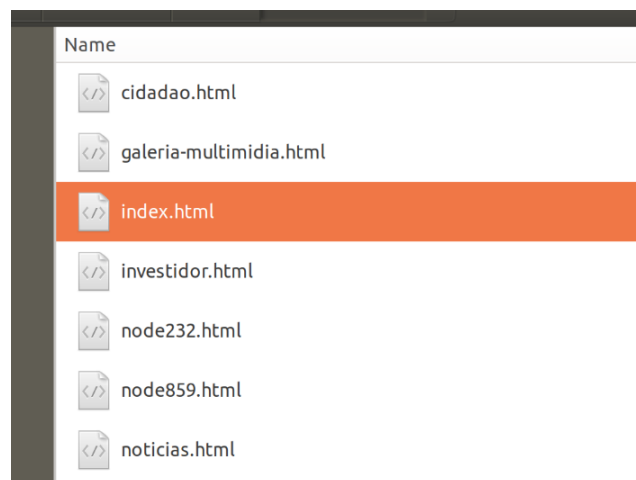


Figura 7: Arquivos baixados.

```
File Edit View Search Terminal Help
/---> /atores/---> /atores/download_csv/1
                    /atores/download_csv/16
                    /atores/download_csv/20
                    /atores/download_csv/23
                    /atores/download_csv/25
                    /atores/download_csv/28
                    /atores/download_csv/34
                    /atores/download_csv/35
                    /atores/download_csv/36
                    /atores/download_csv/37
                    /atores/download_csv/38
                    /atores/download_csv/39
                    /atores/download_csv/40
                    /atores/download_csv/41
                    /atores/download_csv/42
                    /atores/download_csv/43
                    /atores/download_csv/76
                    /atores/download_csv/77
                    /atores/download_csv/78
                    /atores/download_csv/79
                    /atores/download_csv/80
                    /atores/download_csv/81
                    /atores/download_csv/82
                    /atores/download_csv/83
                    /atores/download_csv/84
                    /atores/download_csv/85
                    /atores/download_csv/86
                    /atores/download_csv/87
                    /atores/download_csv/88
                    /atores/download_csv/89

/relacoes/---> /relacoes/download_csv/1
                /relacoes/download_csv/2
                /relacoes/download_csv/3
                /relacoes/download_csv/4
                /relacoes/download_csv/5
                /tlrelacoes/download_csv/1
                /tlrelacoes/download_csv/2
                /tlrelacoes/download_csv/3
                /tlrelacoes/download_csv/4

/tweets/view/342Artes---> /tweets/download_csv/14
                           /tweets/download_csv/147
                           /tweets/download_csv/210
                           /tweets/download_csv/308
                           /tweets/download_csv/374
```

Figura 8: Opção Tree of Url's, gerando arvore.

6 Conclusão.

A realização do trabalho facilitou o entendimento de muitos conceitos vistos em sala de aula, visto que muito da teoria envolvida pode ser visto de forma mais clara após a realização da aplicação do conteúdo e exemplos práticos.

Neste trabalho é implementado um servidor proxy simples, apesar de não ser o mais correto que se pode ter, abre portas para entendimentos mais complexos em futuras implementações das aplicações em rede. O trabalho exigiu o estudo e a compressão dos protocolos que foram definidos e a interação com um cliente já implementado. A implementação dos módulos capazes de interagir com outras aplicações e programas diferentes não é uma tarefa trivial, mas, é uma tarefa necessária e que exige ser praticada quando é se falado de aplicações em rede, o cliente e o servidor nem sempre são desenvolvidos no mesmo processo mas isso não deve ser um impedimento para que eles se comuniquem. Descobrir o funcionamento dos protocolos e suas funcionalidades foram de suma importância para esse trabalho.

7 Referências.

KUROSE, James F; ROSS, Keith W. Transport Layer. In: **Computer Networking – A Top-Down Approach**. Sexta edição. Estados Unidos da América: Pearson Education, 2013. Cap. 3, p. 185-283.

KUROSE, James F; ROSS, Keith W. Application Layer. In: **Computer Networking – A Top-Down Approach**. Sexta edição. Estados Unidos da América: Pearson Education, 2013. Cap. 2, p. 83-184.