Universidade de Brasília
Departamento de Ciência da Computação
Transmissão de Dados

Trabalho de Transmissão de Dados
Implementação de uma Aplicação de Servidor Proxy

Integrantes:
Eric Gil 14/0095993
Rodrigo Flores Mendes 14/0050868

Data: 26/06/2017

1. Introdução Teórica:

1.1. Servidor Proxy Web

Proxy é um termo utilizado para definir os intermediários entre o usuário e o servidor. O proxy desempenha a função de estabelecer a conexão do computador local com a rede externa que no caso será a Internet, como os endereços do computador não são válidos para acesso externos cabe ao proxy enviar a solicitação do endereço local receber as requisições traduzindo e repassando-a para o computador[1].

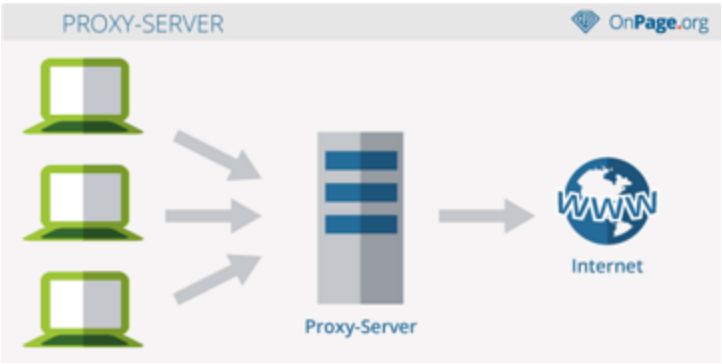

Figura 1 - Imagem Ilustrado um Servidor Proxy.

A figura (1) ilustra o funcionamento de um servidor proxy, todas as requisições feitas ao servidor deverão necessariamente passar pelo servidor proxy, ao acessar o site o *IP* (Protocolo de Internet) que fica registrado é o do proxy não o do usuário. Isso é um benefício do proxy pois o *IP* é a informação que hackers utilizam para invadir computadores, com a utilização do proxy é possível manter um certo nível de sigilo aumentando a segurança na navegação pela Internet.

Um segundo benefício dos proxies vem do fato que eles diminuem o tempo de acesso a um determinado site devido a presença da cache. A cache faz o registro de cada página anteriormente acessada, portanto se um usuário acessou certa página recentemente o próximo acesso a página fica mais rápido pelo fato de que não será necessário fazer o reconhecimento feito inicialmente.

Os web proxies que permite navegar anonimamente são utilizados em redes fechadas como universidades e ambientes de trabalho para burlar uma determinação de bloqueio a alguns sites da internet como por exemplo: *Youtube, Facebook e Twitter.*

[1] https://www.tecmundo.com.br/navegador/972-o-que-e-proxy-.htm - [Acessado em 26/06/2017]

1.2. Protocolo HTTP

O protocolo *HTTP(Hypertext Transfer Protocol)* é um método utilizado para enviar e receber informações na internet. Atualmente a versão 1.1 está sendo a mais utilizada, definida pela especificação da *RFC 2616*. Um navegador fará a requisição solicitando um certo recurso envia um pacote de informações contendo alguns cabeçalhos para um *URL*, o servidor recebe estas informações e envia uma resposta que pode ser um recurso ou simplesmente outro cabeçalho[2].

O protocolo *HTTP* é stateless,ou seja, ele não é capaz de reter informações entre requisições diferentes, para isso é preciso utilizar cookies, sessões entre outras ferramentas. No momento da requisição é preciso especificar que método do protocolo *HTTP* será utilizado, isso irá definir que ação deve ser executada para um determinado recurso.

- *GET*

O método *GET* solicita a representação de um determinado recurso. É definido como um **método seguro** e ***não*** deve ser usado para disparar uma ação (remover um usuário, por exemplo).

- *POST*

No método *POST* as informações enviadas no corpo (body) da requisição são utilizadas para criar um novo recurso. Também é responsável por fazer processamentos que não são diretamente relacionados a um recurso.

- *DELETE*

O método *DELETE* remove um recurso. Deve retornar o status 204 caso não exista nenhum recurso para a URI especificada.

- *PUT*

O método *PUT* atualiza um recurso na URI especificada. Caso o recurso não exista, ele pode criar um. A principal diferença entre POST e PUT é que o primeiro pode lidar não somente com recursos, mas pode fazer processamento de informações, por exemplo.

---

[2] https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http - [Acessado em 26/06/2017]

- HEAD

O método *HEAD* retorna informações sobre um recurso. Na prática, funciona semelhante ao método GET, mas sem retornar o recurso no corpo da requisição. Também é considerado um método seguro.

Além dos métodos citados acima existem ainda os métodos *OPTIONS, TRACE e CONNECT*. Na teoria os servidores devem implementar os métodos *GET* e *HEAD* e quando possível também o método *OPTIONS*.

O status do protocolo *HTTP* é um código de resposta que é recebido após toda requisição, com ele é possível saber se uma operação foi realizada com sucesso, se ela foi movida e agora existe em outro local ou se ela foi removida e não existe mais. Existem vários status em diversas categorias eles são representados por números, é possível ver alguns exemplos a seguir[3]:

200: OK

A requisição foi bem sucedida.

301: Moved Permanently

O recurso foi movido permanentemente para outra URI.

302: Found

O recurso foi movido temporariamente para outra URI.

304: Not Modified

O recurso não foi alterado.

401: Unauthorized

A URI especificada exige autenticação do cliente. O cliente pode tentar fazer novas requisições.

403: Forbidden

O servidor entende a requisição, mas se recusa em atendê-la. O cliente não deve tentar fazer uma nova requisição.

404: Not Found

O servidor não encontrou nenhuma URI correspondente.

405: Method Not Allowed

O método especificado na requisição não é válido na URI. A resposta deve incluir um cabeçalho Allow com uma lista dos métodos aceitos.

503: Service Unavailable

O servidor não é capaz de processar a requisição pois está temporariamente indisponível.

[3] https://tools.ietf.org/html/rfc2616#section-10.2.1 - [Acessado em 26/06/2017]

Existem vários outros Status que a requisição pode retornar como resposta esses são apenas alguns, o protocolo *HTTP* é bem vasto aqui foram mostradas apenas uma visão geral do seu funcionamento e algumas funcionalidades.

1.3. Transporte orientado para conexão: TCP

O protocolo *TCP* é conhecido como um protocolo de transporte confiável da camada de transporte da Internet. É dito que o protocolo *TCP* é orientado para conexão porque antes que um processo de aplicação possa começar a enviar dados a outro os dois processos precisam estabelecer o *"handshake"* , ou seja, eles devem se apresentar. Uma conexão *TCP* provê um serviço *full-duplex* isso significa se houver uma conexão *TCP* entre um processo A em um hospedeiro e um processo B em outro hospedeiro, então os dados da camada de aplicação poderão fluir de A para B ao mesmo tempo em que os dados da camada de aplicação fluem de B para A[1].

Outra característica da conexão *TCP* é o fato que ela sempre é ponto a ponto, isto é ela sempre é estabelecida entre um único remetente e um único destinatário. A estrutura do segmento *TCP* consiste de um cabeçalho e um campo de dados, conforme a figura (2):

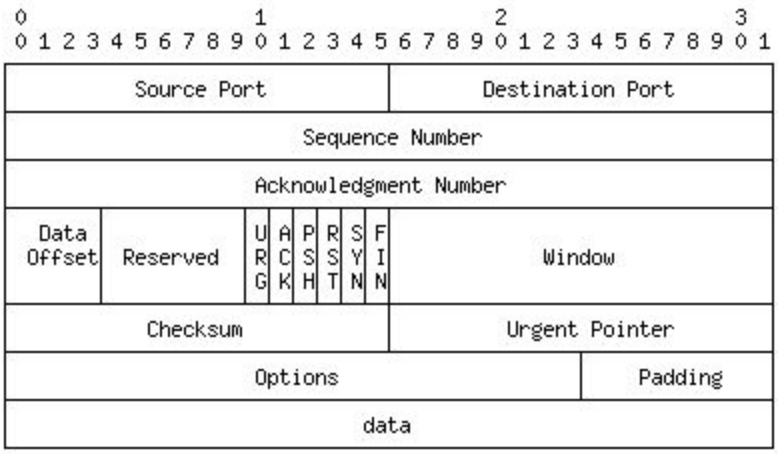

Figura 2 - Estrutura do Cabeçalho TCP.

O tamanho máximo do segmento (*MSS*) é estabelecido normalmente determinando o tamanho do maior quadro de camada de enlace que pode ser enviado pelo hospedeiro remetente local. O *MSS* limita o tamanho máximo do campo de dados de um segmento, quando o *TCP* envia um arquivo grande como por exemplo uma imagem de uma página Web ele comumente fragmenta

o segmento em pedaços de tamanho *MSS* exceto o último pedaço que muitas vezes é menor do que o *MSS.*

2. Materiais Utilizados

2.1. Materiais Utilizados na Implementação do Servidor Proxy

2.1.1. Python

Para a implementação do Servidor Proxy foi escolhida a linguagem de programação *Python 3* devido a sua vasta aplicabilidade e a facilidade de produzir código utilizando a mesma. Na implementação do trabalho utilizou-se bibliotecas típicas da linguagem como a *socket, thread, sys e time*.

2.1.2. Controle de Versionamento Github

De acordo com a especificação do trabalho foi utilizada uma ferramenta de controle de versionamento no caso o *GitHub* para manter registro de versões do trabalho desenvolvidas por cada integrante do grupo. No repositório foi adicionado também o monitor da disciplina, de modo que foi possível para ele manter registra da contribuição de cada membro do grupo.

2.2. Materiais Utilizados teste do Servidor Proxy

2.2.1. Linux Ubuntu

Para a verificação do correto funcionamento do Servidor Proxy desenvolvido foi utilizado o Sistema Operacional *Linux Ubuntu* para a execução do código.

2.2.2. Mozilla FireFox

O *Mozilla FireFox* é um navegador *open-source* gratuito que é utilizado por muitos usuários, esse navegador foi o escolhido para executar os testes de verificação do correto funcionamento do servidor proxy desenvolvido.

3. Procedimentos Adotados

O desenvolvimento do servidor proxy foi feito em conjunto por ambos os integrantes do grupo por meio da linguagem de programação *Python 3*, foi utilizado o fluxograma proposto no roteiro do trabalho como base para o desenvolvimento do código e a implementação das funcionalidades do proxy.

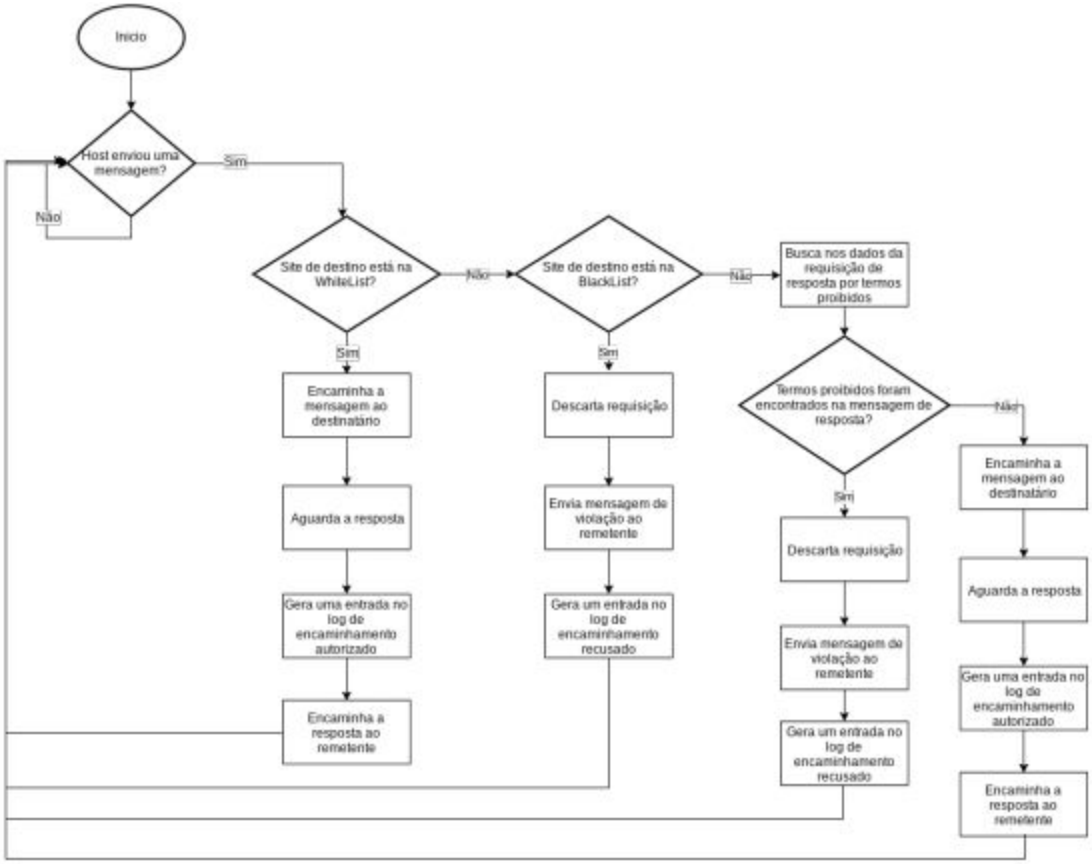

Figura 3 - Fluxograma proposto para Servidor Proxy.

O código foi dividido em duas funções, uma função *main()* que contém a parte principal do código como a abertura do socket, o estabelecimento da conexão do socket criado com a porta e o host e também a função *listen()* que escuta na porta para verificar se há alguma requisição sendo feita. Dentro da *main()* também temos a criação das *threads* para cada um das conexões.

A outra função que foi criada é a função *proxy()* que recebe um requisição e separa o seu cabeçalho para obter o url, após isso é feita a comparação para verificar se o site que está sendo requisitado se encontra na *Whitelist, Blacklist* ou nos *Deny Terms* se o url não se encontra dentro da parte proibida a conexão é estabelecida normalmente, porém se o url se encontra proibido ou possui um termo proibido a conexão é descartada. Dentro da função *proxy()* se encontra também o trecho de código que escreve no log todas as requisições, tanto as proibidas como as liberadas.

4. Resultados Obtidos e Análise

Com o desenvolvimento do servidor pronto foram feitos alguns testes em sites para verificar o correto funcionamento do proxy, inicialmente foi feito um teste no site

"*www.falstad.com*" por meio do navegador *Mozilla FireFox*. O site falstad se encontra na *Whitelist* de acordo com o código desenvolvido, com isso após o servidor proxy verificar que o site está no *Whitelist* o seu acesso será liberado sem que ele passe pelo *Blacklist* e *Deny terms*. A figura (4),(5) e (6) mostra o url para cada uma das requisições enviadas no acesso ao site. É possível perceber que acessar um site relativamente simples como o do *falstad* foram enviadas 7 requisições diferentes, sendo que uma foi da imagem que estava presente no site. A figura (7) mostra o acesso de forma bem sucedida no site.

O próximo teste foi realizar o acesso do site "*www.aprender.unb.br*" o site não se encontra nem na *Blacklist* nem na *Whitelist*, com isso foi realizado o teste para verificar se existe algum termo proibido, como não há nenhum termo proibido o site foi acessado com sucesso. A figura (8) mostra o url da requisição, e a figura (9) mostra que o acesso no site foi bem sucedido.

O último teste foi realizado na tentativa de acessar o site "*www.gaia.cs.umass.edu*" nesse site existe um termo que se encontra no *Deny Terms* que é o nome "Jim Kurose" com isso a figura (10) mostra o url da requisição na tentativa de acessar o site porém como o site está dentro dos deny terms o acesso não é permitido, a figura (11) mostra o bloqueio do site e a mensagem 403 Forbidden. A figura (12) exibe uma imagem do log com os registros de url de requisição e de sites acessados pelo usuário.

Portanto é possível verificar o correto funcionamento do servidor proxy quanto algumas funcionalidades como *Whitelist, Blacklist e Deny Terms* desenvolvidas. Não foi possível implementar o cache do proxy, com isso não teve nenhum exemplo com a funcionalidade do cache sendo exibida.

5. Conclusão

No trabalho foi feito o desenvolvimento de um Servidor Proxy simples com funcionalidades como a *Blacklist*, *Whitelist* e *Deny terms*, e um catálogo de log que registra todos os sites acessados. Para o desenvolvimento dessa aplicação foram utilizados os conhecimentos adquiridos na disciplina de *Transmissão de Dados* como o tipo de requisição que é recebida pelo navegador e o modo de separar essa requisições de modo a obter o *URL*.

A partir da implementação do servidor foi possível colocar em prática a teoria vista em sala de aula, assim como também o funcionamento servidor simples que age como um intermediário para requisições de clientes solicitando recursos de outros servidores. A cache que é responsável por armazenar requisições já solicitadas de modo a minimizar o tempo que é gasto verificando a *Blacklist, Whitelist* e *Deny terms*, foi uma das funcionalidades que não foi possível implementar.

6. Referência Bibliográfica

[1] - Kurose, James F., Keith W. Ross, and Wagner Luiz Zucchi. *Redes de Computadores e a Internet: Uma abordagem top-down*. Pearson Addison Wesley, 2007.

[2] - Caetano, Marcos F., *Material de Apoio,* primeira edição, 2017.

7. Apendice

7.1. Imagens dos Resultados Obtidos

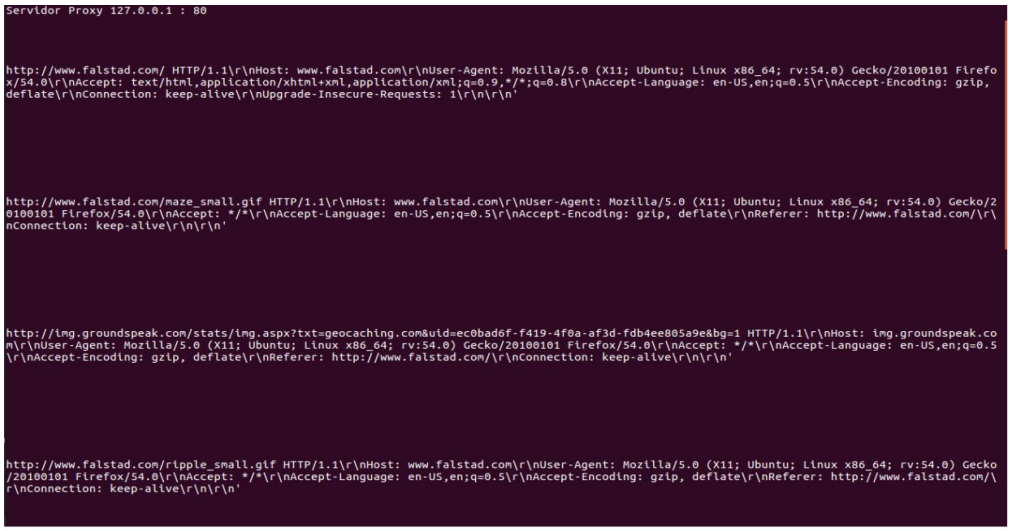

Figura 4 - Teste para acessar o site www.falstad.com.

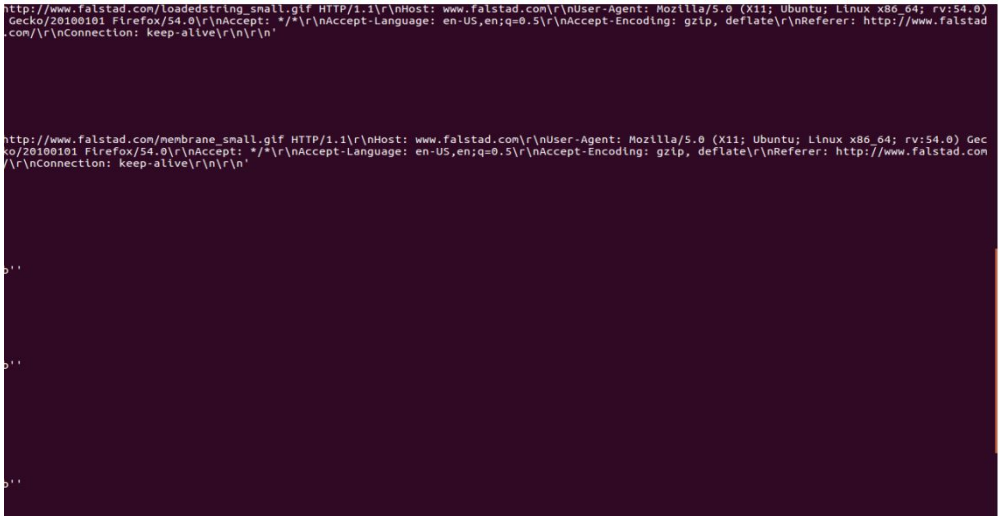

Figura 5 - Teste para acessar o site www.falstad.com.

Figura 6 - Teste para acessar o site www.falstad.com

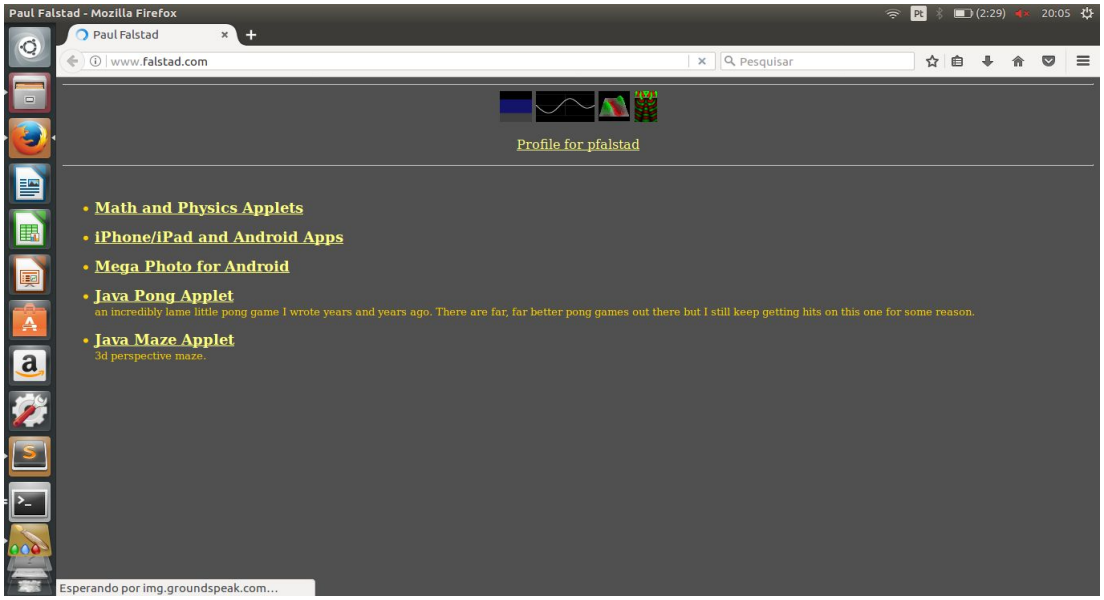

Figura 7 - Resultado do acesso a página www.falstad.com.

```
http://aprender.unb.br/ HTTP/1.1\r\nHost: aprender.unb.br\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefo
x/54.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip,
deflate\r\nCookie: _ga=GA1.2.429653183.1496064709\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\n\r\n'
```

Figura 8 - Teste para acessar o site www.aprender.unb.br.

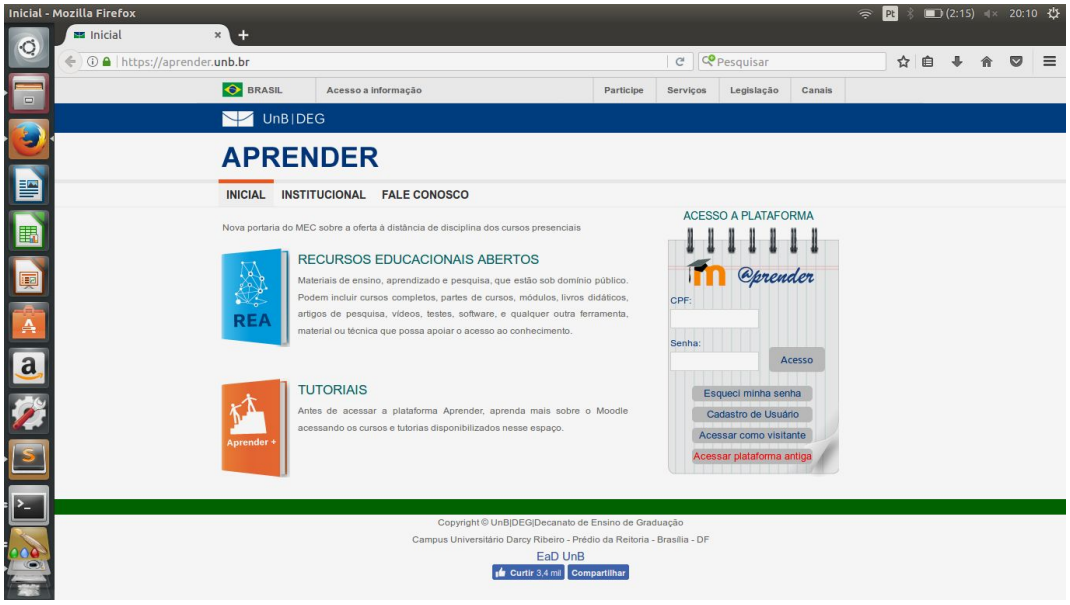

Figura 9 - Resultado do acesso a página www.aprender.unb.br.

```
http://gaia.cs.umass.edu/ HTTP/1.1\r\nHost: gaia.cs.umass.edu\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Fi
refox/54.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gz
ip, deflate\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\n\r\n'



http://gaia.cs.umass.edu/favicon.ico HTTP/1.1\r\nHost: gaia.cs.umass.edu\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/
20100101 Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n'



http://gaia.cs.umass.edu/favicon.ico HTTP/1.1\r\nHost: gaia.cs.umass.edu\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/
20100101 Firefox/54.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-E
ncoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n'
```

Figura 10 - Teste para acessar o site www.gaia.cs.umass.edu.

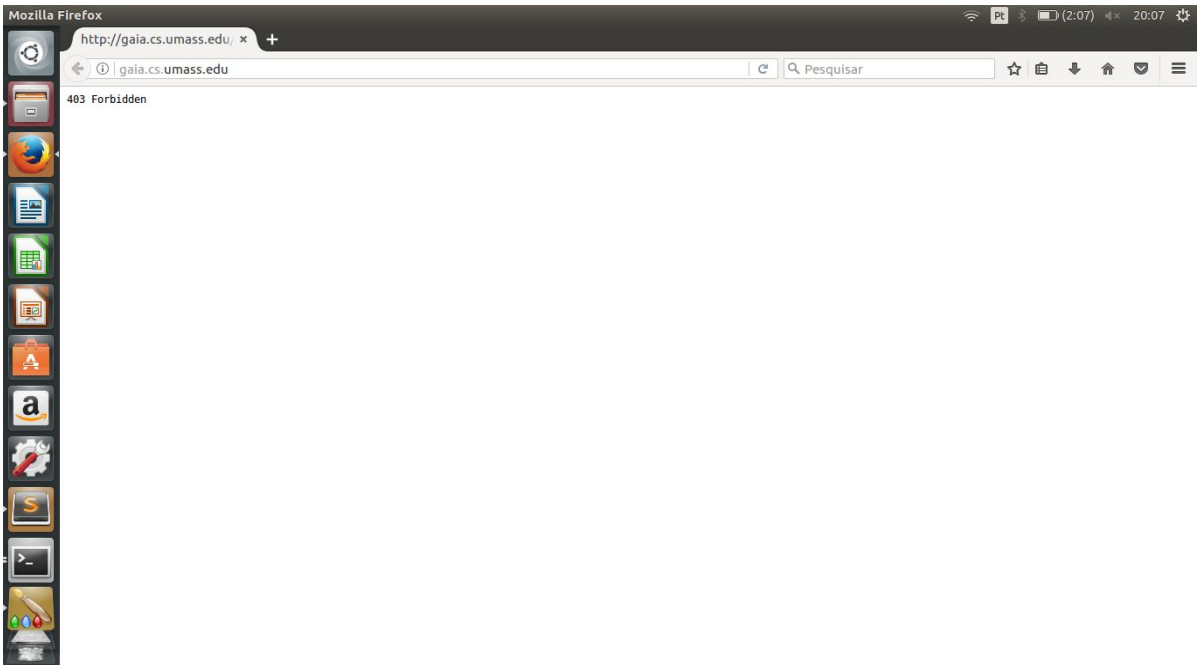

Figura 11 - Resultado do acesso à página www.gaia.cs.umass.edu.

```
26/06/2017 20:00:29: Acesso Realizado - Termo na White List. URL: http://www.falstad.com/ HTTP/1.1\r\nHost: www.falstad.com\r\nUser-Agent:
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado - Termo na White List. URL: http://www.falstad.com/loadedstring_small.gif HTTP/1.1\r\nHost:
www.falstad.com\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0\r\nAccept: */*\r\nAccept-Language:
en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado - Termo na White List. URL: http://www.falstad.com/maze_small.gif HTTP/1.1\r\nHost: www.falstad.com\r
\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r
\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado - Termo na White List. URL: http://www.falstad.com/ripple_small.gif HTTP/1.1\r\nHost: www.falstad.com\r
\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r
\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado - Termo na White List. URL: http://www.falstad.com/membrane_small.gif HTTP/1.1\r\nHost: www.falstad.com\r
\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r
\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado. URL: http://img.groundspeak.com/stats/img.aspx?txt=geocaching.com&uid=ec0bad6f-f419-4f0a-af3d-
fdb4ee805a9e&bg=1 HTTP/1.1\r\nHost: img.groundspeak.com\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r
\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado. URL: http://img.groundspeak.com/stats/img.aspx?txt=geocaching.com&uid=ec0bad6f-f419-4f0a-af3d-
fdb4ee805a9e&bg=1 HTTP/1.1\r\nHost: img.groundspeak.com\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r
\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado. URL: http://img.groundspeak.com/stats/img.aspx?txt=geocaching.com&uid=ec0bad6f-f419-4f0a-af3d-
fdb4ee805a9e&bg=1 HTTP/1.1\r\nHost: img.groundspeak.com\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r
\nConnection: keep-alive\r\n\r\n'

26/06/2017 20:00:30: Acesso Realizado. URL: http://img.groundspeak.com/stats/img.aspx?txt=geocaching.com&uid=ec0bad6f-f419-4f0a-af3d-
fdb4ee805a9e&bg=1 HTTP/1.1\r\nHost: img.groundspeak.com\r\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0\r\nAccept: */*\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nReferer: http://www.falstad.com/\r
\nConnection: keep-alive\r\n\r\n'
```

Figura 12 - Imagem do Log de acesso aos Sites.