

Implementação de uma aplicação de Proxy Sever

Eduardo Scartezini
MATRÍCULA

Departamento de Ciência da Computação
Instituto de Ciências Exatas
Email: <http://www.michaelsell.org/contact.html>

Patrícia Guimarães
13/0015989

Departamento de Ciência da Computação
Instituto de Ciências Exatas

Abstract—Este relatório trata sobre o trabalho final do primeiro semestre de 2017 da matéria de Transmissão de Redes da Universidade de Brasília. Nele, não só explicamos alguns conceitos teóricos relacionados ao trabalho, sendo a maior parte deles parte da teoria vista em sala de aula, como também explicamos melhor nosso servidor *Proxy Web*, que possui filtro de conteúdo e *cache* de páginas *web*.

Keywords—Redes de computador, dados, HTTP, proxy, filtro, *cache*.

I. APRESENTAÇÃO TEÓRICA

A. Proxy Web

Um servidor *proxy* um tipo de servidor, ou seja, um sistema ou uma aplicação, que atua como um intermediário entre os clientes e outros servidores. A relação ocorre da seguinte forma: um cliente conecta-se com um servidor proxy e faz uma nova requisição. O servidor proxy analisa esta requisição, podendo alterá-la, ou mesmo alterar a resposta do servidor.

Um servidor Web proxy possui a mesma função de intermediária, porém, entre o cliente e a página web que ele deseja acessar. Essa ação provê anonimato ao cliente, dado que o servidor mascara as informações do usuário.

B. TCP

O Protocolo de Controle de Transmissão (em inglês, Transmission Control Protocol) um dos protocolos mais importantes da Camada de Transporte, quarta camada segundo o Modelo OSI ou terceira camada segundo o modelo TCP/IP. Ela é responsável por receber dados da Camada de Aplicação, verificar sua integridade, separá-los em pacote e, então, transmiti-los para a camada abaixo, a camada de Rede.

Uma de suas principais características é ser orientado à conexão, ou seja, há uma troca de pacotes entre cliente e servidor, ditos pacotes de controle, antes de serem trocados os pacotes de dados. Dessa forma, são estabelecidos parâmetros para a conexão. Este procedimento ocorre por meio do *handshake* ou *three-way handshake*.

O *handshake* se inicia quando um *Host A*, neste caso, o cliente, envia um segmento de sincronização (*SYN*) ao *Host B*, o servidor, que contém um número inicial de sequência. O servidor, ao receber o *SYN*, envia uma mensagem de aceite da conexão (*SYN ACK*). Nele, ele reconhece o pedido feito e envia o número inicial de sua sequência. Por fim, o cliente confirma o recebimento do aceite enviando um segmento de

acknowledgment (*ACK*). Podemos entender melhor observando a figura abaixo.

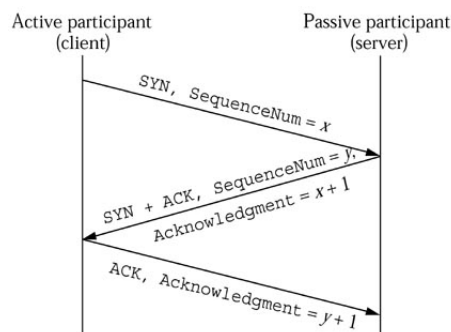


Fig. 1. Como ocorre o *handshake*. Retirada do site: <http://www.wiki.hping.org/122>

Já para encerrar a conexão, um dos *hosts*, neste caso, o cliente, envia um segmento *FIN* solicitando o término da conexão. O *host B*, neste caso, o servidor, envia, então dois segmentos: um *ACK*, indicando que confirma o recebimento do segmento, e, então, também envia um *FIN*. Quando o *host A* enviar um segmento *ACK*, a conexão estará oficialmente terminada, conforme mostra a figura abaixo.

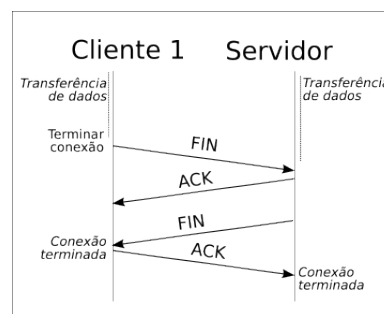


Fig. 2. Finalizando a conexão. Retirada do site: https://upload.wikimedia.org/wikipedia/commons/7/75/TCP_termination.png

C. Protocolo HTTP

O Protocolo de Transferência de Hipertexto (em inglês, HyperText Transfer Protocol) um exemplo de protocolo da Camada de Aplicação, a última camada de ambos os modelos citados acima. É ele quem permite a comunicação de dados entre redes de computadores, principalmente com a World Wide Web, conhecida popularmente pelo acrônimo WWW.

Seu funcionamento ocorre por meio de requisições de clientes, que desejam um determinado recurso, e de respostas de servidores, que contêm solicitações. Em suas requisições, o cliente envia também um cabeçalho, que contém informações que o identificam e, entre outras informações, o método utilizado na requisição. Podemos entender melhor observando o exemplo abaixo.

```
GET / HTTP/1.1
Host: spesa.com.br
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
Gecko/20061201 Firefox/2.0.0.3 (Ubuntu-feisty)
Accept:
text/xml,application/xml,application/xhtml+xml,text/
html;q=0.9,text/plain;q=0.8,image/png
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Fig. 3. Exemplo de requisição feita por um cliente. Retirada do site: <https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>

O servidor, então envia sua resposta, que contém não só o conteúdo HTML da página desejada, mas também um cabeçalho de resposta. Nele, a informação mais importante encontrada é o código de resposta, que vai indicar se a requisição foi concluída com sucesso, se ela não é autorizada, se não foi encontrada, entre outras opções.

```
HTTP/1.x 200 OK
Date: Fri, 04 May 2007 16:05:43 GMT
Server: Apache/2.0.59 (Unix) mod_ssl/2.0.59 OpenSSL/0.9.7a DAV/2
PHP/4.4.4 mod_bwlimited/1.4
Cache-Control: no-cache
Keep-Alive: timeout=3, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1
```

Fig. 4. Exemplo de resposta feita por um servidor. Retirada do site: <https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>

Uma última característica importante desse protocolo que ele dito *stateless*, isto é, ele não retém as informações quando recebe mais de uma requisição, sendo necessário utilizar-se de outras formas para guardar as informações desejadas.

II. ARQUITETURA DO SISTEMA

III. DOCUMENTAÇÃO DO CÓDIGO

IV. FUNCIONALIDADES IMPLEMENTADAS

Em nosso trabalho, nós

A. Funcionamento Básico

B. Filtragem de Requisições

C. Caching

V. CONCLUSÃO

REFERENCES

- [1] H. Kopka and P. W. Daly, *A Guide to L^AT_EX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.