

Transmissão de Dados - 1/2019

# Trabalho Final - Servidor Proxy

---

Francisco Matheus Pereira de Oliveira

14/0039937

Rebeca Helen Silva de Sousa

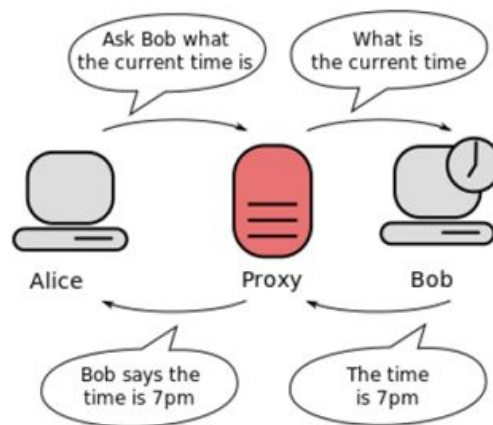
13/0145068

23 de Junho de 2019


## Introdução Teórica

### **Proxy Server Web**

O serviço de web proxy surgiu a partir da necessidade de conectar uma rede local a internet através de um equipamento que compartilha a conexão com as demais máquinas. Assim, o proxy web atua como intermediário entre um dispositivo e os serviços de internet. No momento em que o endereço de um site é digitado no navegador, a solicitação é enviada ao proxy, que então realiza esta solicitação ao servidor no qual o site é hospedado, devolvendo para o cliente o resultado.



O proxy web permite controlar os serviços acessados na internet através do protocolo HTTP (posteriormente explicado), sendo responsável gestão de acesso a sites e outras aplicações baseadas no protocolo. O proxy é amplamente utilizado por empresas dos mais variados portes e segmentos, contribuindo para a manutenção da estratégia de segurança do negócio.




O serviço de proxy web permite que a empresa gerencie os acessos de colaboradores, tendo por base URLs, horários, grupos de usuários, além de outros níveis de controle. Com a implantação do serviço é possível ter acesso a relatórios detalhados de navegação, incluindo consumo de recurso, tempo de navegação, sites mais acessados, além de outras funções que auxiliam na manutenção da integridade do recurso de internet e da rede corporativa.

A aplicação do serviço de proxy também pode auxiliar na redução do tempo de resposta das requisições, através da utilização da função de cache. O armazenamento dos objetos requisitados reduz o tráfego de informações, melhorando o desempenho do acesso à internet, uma vez que desafia a transferência de dados.

Portanto, a utilização destes serviços nas redes empresariais é cada vez mais necessária, devido a infinidade de informações acessíveis através da internet e dos problemas de segurança e produtividade decorrentes a este fato.

## **TCP**

O TCP (Transmission Control Protocol - Protocolo de Controle de Transmissão) é um dos principais protocolos da camada de transporte do modelo TCP/IP. Ele permite gerenciar os dados vindo da (ou com destino à) camada inferior do modelo (ou seja, o protocolo IP). Quando os dados são fornecidos ao protocolo IP, este encapsula-os em datagramas IP ('datagrama' é um nome genérico para uma mensagem enviada sem conexão e sem confirmação, como os pacotes IP), fixando o campo do protocolo em 6 (para saber que o protocolo ascendente é o TCP). O TCP é um protocolo orientado para a conexão, isto é, ele permite que duas máquinas se comuniquem entre elas, além de controlar o estado da transmissão.



As características do protocolo TCP são entregar ordenadamente os datagramas provenientes do protocolo IP, verificar a onda de dados para evitar uma saturação da rede, formatar os dados em segmentos de comprimento variável para 'entregá-los' ao protocolo IP, permitir o multiplex dos dados, ou seja, fazer circular, simultaneamente, as informações de fontes distintas na mesma linha e permitir o início e o fim de uma comunicação de maneira correta.


Graças ao protocolo TCP, os aplicativos podem se comunicar com segurança (pelo sistema de avisos de recepção do protocolo TCP), independentemente das camadas inferiores. Isto significa que os roteadores (que trabalham na camada da Internet) têm, como papel fundamental, o encaminhamento dos dados em forma de datagramas, sem se preocuparem com o controle dos dados, pois este é realizado pelo protocolo TCP.

## **HTTP**

Como citado anteriormente, o funcionamento do proxy se baseia em conceitos do protocolo HTTP. Dessa forma, a explanação dos conceitos e preceitos deste protocolo são de suma importância.

HTTP é a sigla em língua inglesa de HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto), um protocolo da camada de Aplicação do modelo OSI utilizado para transferência de dados na rede mundial de computadores, a World Wide Web. Também transfere dados de hipermídia (imagens, sons e textos).

Normalmente, este protocolo utiliza o porta 80 e é usado para a comunicação de "sites" (sítios), comunicando na linguagem HTML (Hypertext Markup Language, ou Linguagem de Marcação de Hipertexto). Contudo, para haver comunicação com o



servidor do site é necessário utilizar comandos adequados, que não estão em linguagem HTML.

Para acessar a outro documento a partir de uma palavra presente no documento atual podemos utilizar os chamados links/ (ligações) ou âncoras. Estes documentos encontram-se num "site" (sítio) com um endereço de página da Internet - e para entrarmos neles devemos digitar o respectivo endereço, denominado URI (Universal Resource Identifier ou Identificador Universal de Recurso), que não deve ser confundir com URL (Universal Resource Locator ou Localizador Universal de Recurso), um tipo de URI que pode ser directamente localizado.

Um sistema de comunicação em rede possui diversos protocolos que trabalham em conjunto para o fornecimento de serviços. Para que o protocolo HTTP consiga transferir seus dados pela Web, é necessário que os protocolos TCP e IP (Internet Protocol, Protocolo de Internet) tornem possível a conexão entre clientes e servidores através de sockets TCP/IP.

De acordo com Fielding et al (1999, p. 10), o HTTP utiliza o modelo cliente-servidor, como a maioria dos protocolos de rede, baseando-se no paradigma de requisição e resposta. Um programa requisitante (cliente) estabelece uma conexão com um outro programa receptor (servidor) e envia-lhe uma requisição, contendo a URI, a versão do protocolo, uma mensagem MIME (padrão utilizado para codificar dados em formato de textos ASCII para serem transmitidos pela Internet) contendo os modificadores da requisição, informações sobre o cliente e, possivelmente, o conteúdo no corpo da mensagem.

---

O servidor responde com uma linha de status (status line) incluindo sua versão de protocolo e um código de operação bem sucedida ou um código de erro, seguido pelas informações do servidor, informações da entidade e possível conteúdo no corpo da mensagem. Após o envio da resposta pelo servidor, encerra-se a conexão estabelecida.

---

## Metodologia de Implementação

A linguagem escolhida para a implementação foi o Python 3. Para a implementação das conexões entre o cliente e o proxy foi utilizada a biblioteca *Socket*, que permite definir um socket com conexão do tipo TCP, e escutar com uma porta e IP determinados. Com isso é possível enviar e receber mensagens. Para receber várias requisições foi utilizada a biblioteca *Thread*.

```
# Create a TCP socket
try:
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((IP,Porta))
    server.listen(10)
    print ("Escutando com" ,IP,Porta)
```

O código implementado é constituído de três blocos principais: Main, com configurações iniciais, Conexão do cliente, onde as *threads* de cada cliente são encaminhadas, e Proxy, onde é feita a conexão com o *webserver* requisitado pelo cliente. A parte de conexão com o cliente é feita na função “conn\_cliente” e ela é responsável pela parte de filtragem do servidor proxy.

Primeiramente essa função procura nos dados enviado pelo cliente o url requisitado, depois a filtragem é feita pela função *checksite*, que procura em dois arquivos (*blacklist* e *whitelist*). A função *checksite* retorna um inteiro: 0 para site

contidos na *blacklist*, 1 para sites contidos da *whitelist* e 2 caso não esteja em nenhuma dessas listas.

```
def checksite (url):  
    ret = 2  
    arq = open('blacklist','r')  
    datafile = arq.readlines()  
  
    for line in datafile:  
        if line == url:  
            ret = 0  
    arq.close()  
  
    arq = open('whitelist','r')  
    datafile = arq.readlines()  
  
    for line in datafile:  
        if line == url:  
            ret = 1  
    arq.close()
```

Caso uma página contida do arquivo *blacklist* seja requisitada, o pacote a ser requisitado é descartado e uma página html, como mostra a imagem a seguir, é encaminhada ao remetente informando que o acesso foi negado e o motivo para isso.



A busca por termos proibidos é feita pela função *finddeny\_terms*, que também procura os termos proibidos em um arquivo. A requisições para sites proibidos e sites

com termos proibidos recebem de volta uma mensagem no formato HTML, mostrada abaixo, avisando da proibição.



Para as requisições são gerados logs que registram o histórico de acessos intermediados pelo servidor proxy.

```
06/22/19 16:55:06 _ Acesso permitido _ Mensagem sem deny_terms detectportal.firefox.com
06/22/19 16:55:07 _ Acesso permitido _ Mensagem sem deny_terms snippets.cdn.mozilla.net
06/22/19 16:55:26 _ Acesso permitido _ Mensagem sem deny_terms detectportal.firefox.com
06/22/19 16:55:27 _ Acesso permitido _ Mensagem sem deny_terms snippets.cdn.mozilla.net
06/22/19 16:55:32 _ Acesso permitido _ URL presente na whitelist: gaia.cs.umass.edu
06/22/19 16:55:39 _ Acesso bloqueado _ URL presente na blacklist: www.play-hookey.com
06/22/19 16:55:47 _ Acesso bloqueado _ URL presente na blacklist: www.play-hookey.com
06/22/19 16:55:56 _ Acesso bloqueado _ Página requisitada contém deny_terms www.watchthatpage.com
06/22/19 16:55:56 _ Acesso bloqueado _ Página requisitada contém deny_terms www.watchthatpage.com
06/22/19 16:55:56 _ Acesso bloqueado _ Página requisitada contém deny_terms www.watchthatpage.com
06/22/19 16:56:01 _ Acesso permitido _ Mensagem sem deny_terms www.watchthatpage.com
```

Na função proxy a requisição é enviada ao cliente caso esteja na *whitelist* ou não tenha termos proibidos. A resposta é enviado ao cliente por meio da função *send* da biblioteca *socket*.



```
def proxy(webserver, port, conn, addr, data, permission):
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((webserver, port))
        s.sendall(data)

        while 1:
            reply = s.recv(buffer_size)
            if (len(reply) > 0):
                founded_reply = finddeny_terms(reply)
                if founded_reply == 1 and permission == 2:
                    permission = 4
                    geralog(webserver, permission, data)
                    replydeny = getdenypage(permission)
                    conn.send(replydeny)
                else:
                    geralog(webserver, permission, data)
                    conn.send(reply)
```

## Código

#Servidor

import socket

import sys

import os


import os.path

import datetime

from \_thread import start\_new\_thread

IP = '127.0.0.1'

Porta = 8088



```
buffer_size = 8192
```

```
def getdenypage(permission):  
    if permission == 0:  
        arqdenypage = open("blacklistpage", "r")  
    else:  
        arqdenypage = open("deny_terms_page", "r")  
    pagedeny = arqdenypage.read()  
    pagedenybyte = pagedeny.encode('utf-8')  
    arqdenypage.close()  
    return pagedenybyte
```

```
def finddeny_terms (data):  
    #Converte a variavel data em string  
    datastr = data.decode("utf-8")  
    #Abre o arquivo de deny_terms  
    arqdeny = open("deny_terms", "r")  
    deny_term = arqdeny.readlines()  
    #Inicializa a variável de retorno sinalizando que nenhum deny_term foi encontrado  
    founded = 0  
    #Busca todos os termos dentro do arquivo  
    for line in deny_term:  
        #Isola o termo a ser buscado  
        term = line.split('\n')
```



```
        #Procura o termo no pacote de dados recebido/enviado
        achou = datastr.find(term[0])

        if achou == -1:                                #Não encontrou o termo

            founded = 0

        else:                                           #Termo encontrado no pacote

            founded = 1

            arqdeny.close()

            return founded

        #Caso após a consulta de todos os deny_terms nenhum seja encontrado, retorna 0

        arqdeny.close()

        return founded
```

```
def geralog(webserver, permission, data):

    #Gera registro no arquivo de logs

    arqlog = open("logs","a")

    timenow = datetime.datetime.now()

    arqlog.write(timenow.strftime("%X"))

    arqlog.write(' ')

    arqlog.write(timenow.strftime("%X"))

    arqlog.write(' _ ')

    arqlog.write(' Acesso ')

    if permission == 0:

        arqlog.write('bloqueado _ URL presente na blacklist:\t')
```



```
if permission == 1:

    arqlog.write('permitido _ URL presente na whitelist:\t')


if permission == 2:

    arqlog.write('permitido _ Mensagem sem deny_terms\t\t\t')


if permission == 3:

    arqlog.write('bloqueado _ Requisição contém deny_terms\t')


if permission == 4:

    arqlog.write('bloqueado _ Página requisitada contém deny_terms\t')


arqlog.write(webserver)

arqlog.write('\n')

arqlog.close()


def proxy(webserver, port, conn, addr, data, permission):

    try:

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        s.connect((webserver, port))

        s.sendall(data)


    while 1:
```



```
        reply = s.recv(buffer_size)

        if (len(reply)> 0):

            founded_reply = finddeny_terms(reply)

            if founded_reply == 1 and permission == 2:

                permission = 4

                geralog(webserver, permission, data)

                replydeny = getdenypage(permission)

                conn.send(replydeny)

            else:

                geralog(webserver, permission, data)

                conn.send(reply)

        else:

            break

    s.close()

    conn.close()


except socket.error (value, message):

    print("erro proxy")


    s.close()

    conn.close()

    sys.exit(1)
```



```
def checksite (url):  
    ret = 2  
  
    arq = open('blacklist','r')  
    datafile = arq.readlines()  
  
    for line in datafile:  
        if line == url:  
            ret = 0  
  
    arq.close()  
  
    arq = open('whitelist','r')  
    datafile = arq.readlines()  
  
    for line in datafile:  
        if line == url:  
            ret = 1  
  
    arq.close()  
  
    return ret
```



```
def conn_cliente(conn, data, addr):
```

```
    try:
```

```
        data2 = data.decode("utf-8")
```

```
        first_line = data2.split("\n")[0]
```

```
        url = first_line.split(' ')[1]
```

```
        http_position = url.find("://")
```

```
        if (http_position == -1):
```

```
            temp = url
```

```
        else:
```

```
            temp = url[(http_position+3):]
```

```
        port_position = temp.find(":")
```

```
        webserver_position = temp.find("/")
```

```
        if webserver_position == -1:
```

```
            webserver_position = len(temp)
```

```
        webserver = ""
```

```
        port = -1
```

```
        if (port_position == -1 or webserver_position < port_position):
```

```

        port = 80

        webserver = temp[:webserver_position]

    else:

        port = int((temp[(port_position+1):]):webserver_position - port_position -1))

        webserver = temp[:port_position]

    permission = checksite(webserver+'\n')

    if permission == 0:
#Destino contido na blacklist

        geralog(webserver, permission, data)

        replydeny = getdenypage(permission)

        conn.send(replydeny)

        conn.close()

    else:
#Caso não contido na blacklist

        if permission == 2:
#Busca termos proibidos

            founded = finddeny_terms(data)

            if founded == 1:
#Termo proibido encontrado

                permission = 3
#Muda a permissão para 3 (termo proibido encontrado)

                print('Request with deny_terms')

```



```

                                geralog(webserver, permission, data)
#Gera log de requisição bloqueada por conter termos proibidos

                                conn.close()
#Descarta requisição com termo proibido

                                else:

                                proxy(webserver, port, conn, addr, data, permission)
#Encaminha requisição sem termos proibidos

                                else:
#Destino contido na whitelist

                                proxy(webserver, port, conn, addr, data, permission)


except Exception:

    pass


def main():

    # Create a TCP socket

    try:

        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        server.bind((IP,Porta))

        server.listen(10)

        print ("Escutanto com" ,IP,Porta)

```



```
except Exception as erro:
```

```
    print('Deu ruim')
```

```
    print (erro)
```

```
    server.close()
```

```
while 1:
```

```
    try:
```

```
        conn, addr_client = server.accept()
```

```
        data = conn.recv(buffer_size)
```

```
        start_new_thread(conn_cliente, (conn, data,addr_client))
```

```
    except KeyboardInterrupt:
```

```
        server.close()
```

```
        sys.exit(1)
```

```
server.close()
```

```
#Começo2
```

```
main()
```