**Teleinformática e Redes 2 – Turma A – Trabalho 1**
**Professor:** João Gondim
**Monitor:**
**Resumo:** Implementação de uma aplicação de Proxy Server.

# 1 Introdução

Em redes de computadores, um *proxy* (em português procurador) é um servidor (um sistema de computador ou uma aplicação) que age como um intermediário para requisições de clientes solicitando recursos de outros servidores. Um cliente conecta-se ao servidor *proxy*, solicitando algum serviço, como um arquivo, conexão, página web ou outros recursos disponíveis de um servidor diferente e o *proxy* avalia a solicitação como um meio de simplificar e controlar sua complexidade. Os servidores *proxy* foram introduzidos para adicionar estrutura e encapsulamento a sistemas distribuídos. Hoje, a maioria dos servidores *proxy* são do tipo *web proxy*, facilitando o acesso ao conteúdo na rede e fornecendo anonimato.

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. Esses servidores têm uma série de usos, como filtrar conteúdo e prover anonimato, entre outros.

Um servidor proxy web de filtro de conteúdo fornece controle administrativo sobre o conteúdo que pode trafegar em ambas as direções pelo proxy. É comumente utilizado em organizações comerciais e não-comerciais (especialmente escolas) para garantir que o uso da Internet está de acordo com a política de uso aceitável. Ele normalmente também produz registros, para dar informações detalhadas sobre as URLs acessadas por usuários específicos ou para monitorar estatísticas de uso da largura de banda.

# 2 Descrição e Implementação

O objetivo deste trabalho é aplicar o conhecimento adquirido em sala sobre o funcionamento de redes de computadores através da construção de um servidor *Proxy Web* com filtro de conteúdo, *cache* de páginas *web* e inspeção de cabeçalhos HTTP. Para tal o servidor deverá receber requisições HTTP processar tanto seu cabeçalho, para verificação do endereço de destino, quanto do campo de dados em busca de termos proibidos pela organização.

Para geração das requisições HTTP um navegador *web* pode ser utilizado, bastando para isso que o servidor e o navegador sejam propriamente configurados. As politicas de restrição de endereços de sites e de termos proibidos devem ser carregados a partir de arquivos para a execução do servidor. Além disso, todas as requisições devem gerar entradas de log mantendo assim a persistência de informações essenciais.

## 2.1 Funcionamento Básico

O servidor *proxy* é um intermediário na comunicação entre o cliente e o servidor a que se deseja conectar. Ele assim recebe todas as mensagens de um determinado *host* ou rede se comportando como um *gateway* tratando tais mensagem. Em seguida ele as encaminha para o endereço de destino, entretanto primeiro ele altera o cabeçalho da mensagem original colocando no campo endereço de

origem o seu próprio endereço. Assim para o destinatário o solicitante da mensagem é na verdade o servidor *proxy*. Quando o servidor recebe a resposta ele pode encaminhar a reposta ao cliente ou então descartá-la, utilizando as políticas definidas anteriormente para tomada desta decisão. Ele ainda pode alterar o conteúdo da resposta com a finalidade de adequá-la as regras impostas. Este funcionamento simples pode ser visto na Figura 1.

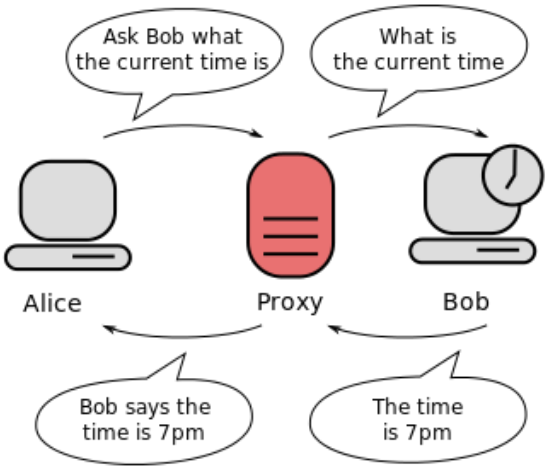


Figura 1: Modelo conceitual *proxy*

A primeira parte do desenvolvimento do servidor consiste em construir os mecanismos necessários ao encaminhamento dos pacotes recebidos de forma correta. O servidor deve ser capaz de tratar múltiplas requisições de diferentes clientes, ou seja vários clientes diferentes estarão conectados ao *proxy* e estes clientes estarão enviando várias mensagens a eles, um exemplo disto pode ser visto na Figura 2.

Assim o servidor deve garantir que as mensagens sejam encaminhadas corretamente entre os diversos clientes e processos. Um cliente não pode receber mensagens de outros clientes. Uma estrutura para identificação de requisições deve ser desenvolvida a fim de implementar esta funcionalidade. Um fluxograma do funcionamento deste sistema é ilustrado na Figura 3.

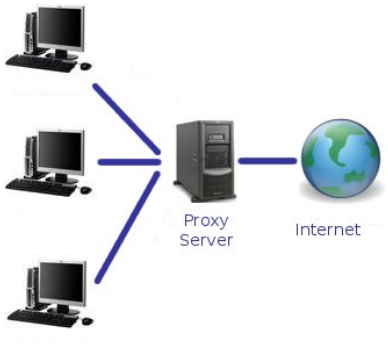


Figura 2: *Proxy* como *gateway* de rede

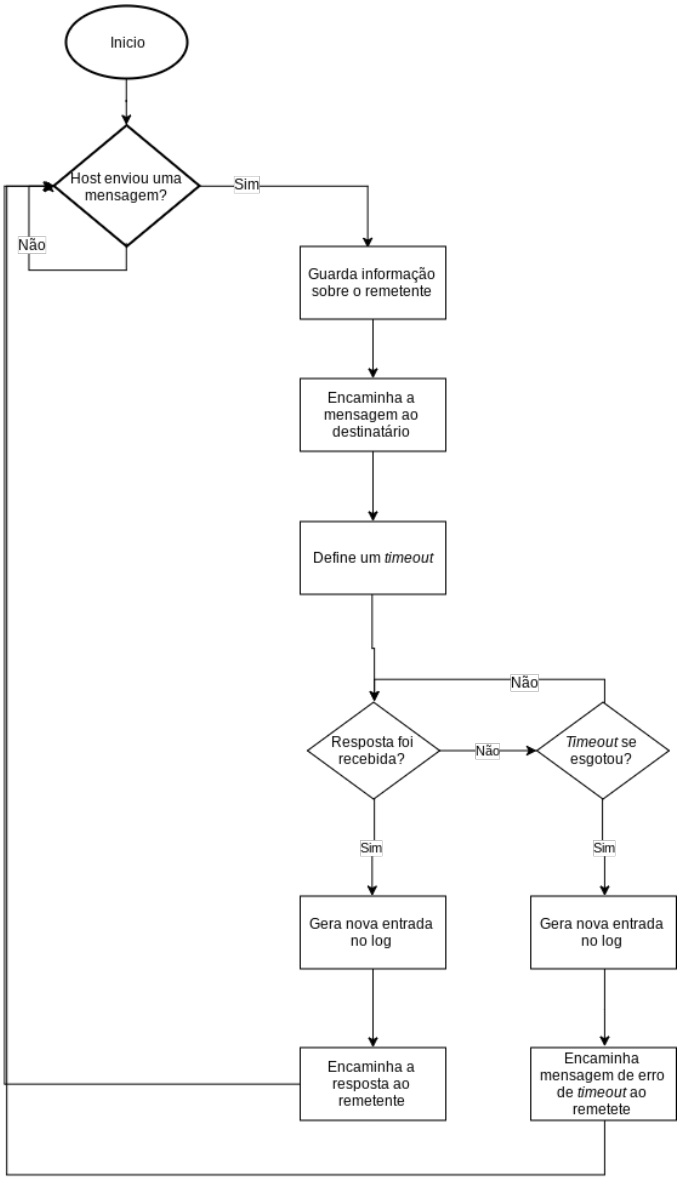


Figura 3: Fluxograma *Proxy* - Encaminhamento

## 2.2 Filtragem de requisições

Para realizar a filtragem das mensagens trocadas entre o *host* e o servidor de destino é necessário a utilização de arquivos para definição de domínios autorizados e restritos e de termos proibidos. Fica a critério do aluno quantos arquivos devem ser definidos e quais os seus nomes, entretanto para fins de explicação chamaremos o arquivo com os domínios autorizados de *whitelist*,o arquivo com domínios restritos de *blacklist* e o arquivo com os termos proibidos de *deny_terms*.

Tais arquivos facilitam a gerência do servidor, permitindo que um site ou palavra possa ser modificada sem a necessidade de se acessar diretamente o código. Novamente a estrutura destes arquivos fica a definição do aluno, um exemplo de como tais arquivos pode ser organizada é mostrada em Figura1 onde cada linha do arquivo é um domínio ou termo.

Tabela 1: Exemplo de arquivos de configuração

| *whitelist* | *blacklist* | *deny_terms* |
|---|---|---|
| www.aprender.unb.br | www.facebook.com.br | lol |
| www.unb.br | www.youtube.com.br | gatinhos |
| www.cic.unb.br | www.netflix.com.br | fofinhos |

A filtragem das mensagem deverá ocorrer da seguinte forma:

- Se o site consultado encontra-se cadastrado na *whitelist* os pacotes são encaminhados entre o remetente e o destinatário, gerando uma entrada no log de encaminhamento autorizado
- Se o site consultado encontra-se cadastrado na *blacklist* a requisição é descartada, uma mensagem de acesso não autorizado é enviada ao remetente informando da violação e uma entrada no log de encaminhamento recusado é gerada.
- Se o site consultado encontra-se cadastrado em nenhuma das duas listas o conteúdo da requisição e da resposta devem ser examinado em busca dos termos constantes no arquivo *deny_terms*. Caso o conteúdo de qualquer uma delas apresente algum dos termos proibidos, a requisição/resposta deve ser descartada e uma mensagem no formato HTML deverá ser enviada ao cliente informando que o conteúdo daquela página não é autorizado e em seguida uma entrada no log de encaminhamento recusado deve ser gerada.

O fluxograma deste processo pode ser conferido na Figura 4.

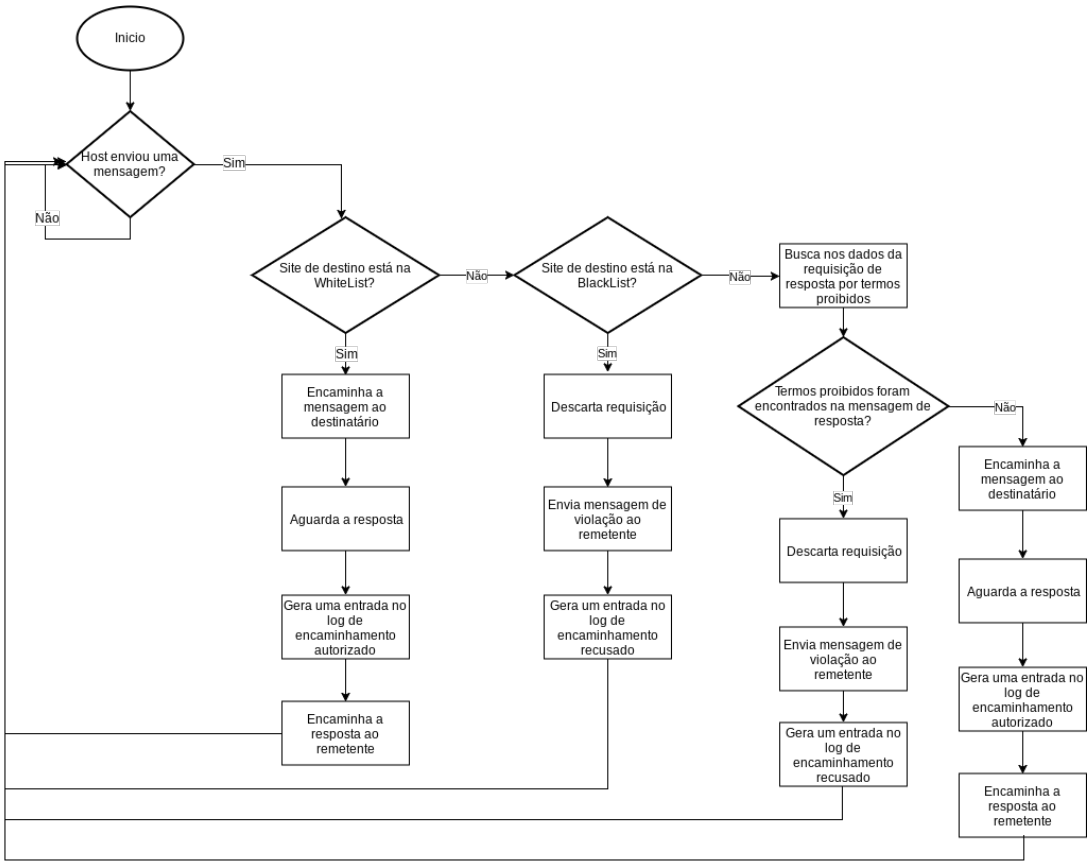


Figura 4: Fluxograma *Proxy* - Filtragem

## 2.3 *Caching*

Outra função muito importante de um *proxy web* é a de *caching* que é o armazenamento de páginas HTTP internamente no servidor de *proxy* para que quando uma nova requisição ocorra ela possa ser respondida mais rapidamente. Uma simplificação de tal serviço pode ser visto na Figura 5.

Um *proxy* de *cache* HTTP permite por exemplo que o cliente requisite um documento na *World Wide Web* e o *proxy* procure pelo documento no seu sistema de arquivos (*cache*). Se encontrado, a requisição é atendida e o documento é retornado imediatamente. Caso contrário, o *proxy* busca o documento no servidor remoto, entrega-o ao cliente e salva uma cópia em seu *cache*. Isto permite uma diminuição na latência, já que o servidor *proxy*, e não o servidor original, é requisitado, proporcionando ainda uma redução do uso da banda.

A implementação dessa funcionalidade pode ser implementada fazendo-se o *hash* das urls solicitadas tornando mais fácil a busca das páginas no sistema de arquivos do servidor *proxy*. O armazenamento das páginas por sua vez não pode ser por tempo indefinido, uma política de gerência das páginas armazenadas também deve ser implementada de forma a não extrapolar a capacidade de armazenamento da máquina.

O cabeçalho de resposta HTTP contem informações sobre o período de validade daquela resposta, estando a página obsoleta se a data e hora assinalada já foi atingida. O Servidor *proxy* deve observar e respeitar este parâmetro não oferecendo esta página como resposta a requisições do cliente, descartando imediatamente o arquivo.

Políticas adicionais de gerência das páginas armazenadas podem ser implementadas com o proposito de melhorar a execução dos serviços oferecidos pelo servidor *proxy*. Politicas de verificação das páginas armazenadas também podem ser implementadas como por exemplo o envio de requisições do tipo *HEAD* para verificação da modificação do conteúdo das páginas.

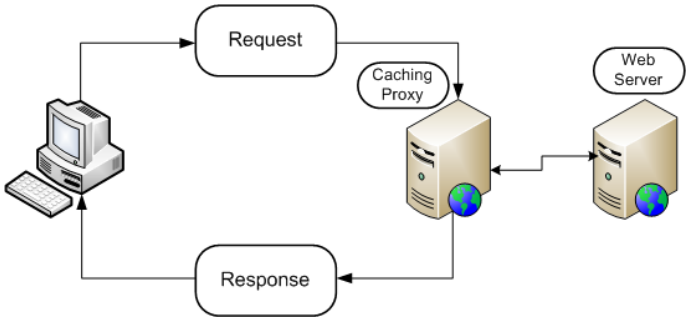


Figura 5: *Proxy Caching*

## 2.4 Inspeção de cabeçalhos HTTP

A inspeção dos cabeçalhos HTTP, tanto de requisições quanto de respostas, é uma funcionalidade menos comum em um *Proxy Web* sendo normalmente utilizada para fins de depuração de aplicações *Web*. Com esta funcionalidade ativada, as requisições/respostas são interceptadas antes de ser enviadas pelo proxy/entregues ao browser e mostradas na tela em uma janela específica com duas subjanelas: uma para requisições outra para respostas (espaço para requisições terá um botão para envio e o da resposta um para recepção). Deve-se prover a capacidade de editar os campos HTTP das requisições antes do envio.

## 2.5 Restrições

1. O servidor só deverá tratar requisições do tipo HTTP. Não serão realizadas requisições do tipo HTTPS durante os testes.
2. Não será aceita a utilização de bibliotecas *proxy* ou de HTTP Parser já desenvolvidas.
3. O uso das demais bibliotecas deverá ser aprovada previamente pelo professor da disciplina.
4. O trabalho deve ser realizado em grupos de no máximo 4 alunos.

# 3 Relatório

Um relatório final do projeto deve ser apresentado. Este relatório deve conter:

- Apresentação teórica sobre o *Proxy Server Web*, TCP e o protocolo HTTP. Exemplos deverão ser apresentados;

- Documento apresentando a arquitetura do sistema desenvolvido;
- Explicação da arquitetura produzida e da relação entre dos principais componentes;
- Documentação de todo o código desenvolvido;
- *Screenshots* e explicação do funcionamento das funcionalidades implementadas;

# 4 Avaliação

A avaliação consiste em 3 componentes:

- Código do projeto.
  - O trabalho deve ser desenvolvido **OBRIGATORIAMENTE** utilizando-se de uma ferramenta de versionamento *online*(Ex.:GitHub, BitBucket, etc). O professor da disciplina deverá ser adicionado ao repositório do trabalho quando o mesmo estiver concluído.
  - O código e o relatório deverão ser enviados também pelo grupo até a data fixada na tarefa;
- Apresentação e demonstração do funcionamento
  - A apresentação do trabalho será agendada pelo grupoi da disciplina;
  - Não serão aceitos trabalho enviados fora do prazo;
  - Trabalhos não apresentados também não serão considerados.
  - Será feita uma verificação dos *commits* para validação da divisão de trabalho entre os alunos que executaram o projeto.
- Nota do Relatório.

# 5 Observações

As seguintes observações deverão ser consideradas:

- Para efeito de testes do servidor *proxy* o mesmo será executado configurado com um IP de loopback(127.0.0.1);
- Um navegador será configurado para utilizar este endereço como *proxy* enviando assim todas suas requisições a este servidor;
- O trabalho deve rodar **OBRIGATORIAMENTE** na plataforma GNU/Linux;
- O programa pode ser feito em linguagem C ou C++ usando a SOCKETS API;
- No relatório do trabalho deve conter as instruções para compilação/execução do código e a relação de todas as bibliotecas utilizadas;
- Códigos copiados serão considerados "cola"e todos os alunos envolvidos ganharão nota zero;
- Dúvidas sobre o trabalho deverão ser tiradas **utilizando grupo da disciplina**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.

# 6 Prazo para entrega: 23:59h do dia 29/11/2017