SquidPCB

Squid-cache Pornography
Content Blocker

Fernando Lemes da Silva
Nr.USP 2371843
Orientado por Prof. Dr. Roberto Hirata Jr.

19 de fevereiro de 2008

Resumo

Esta monografia tem como objetivo produzir um sistema para filtragem de conteúdo impróprio em um servidor proxy, seja na forma de imagens de pessoas nuas ou na forma de textos pornográficos. O sistema utilizará a URL dos arquivos para agrupar as estatísticas obtidas a partir de cada uma, dado que é uma prática comum organizar os arquivos em pastas de acordo com o seu assunto. Acreditamos que tal abordagem permitirá uma boa classificação, não somente devido a precisão dos filtros de imagem, mas sim pelo uso de imagens possivelmente relacionadas na decisão final de bloqueio.

The objective of this work is to build a content filtering system to block unappropriate content in a proxy server, that could appear in form of nude images or in form of pornographic texts. The system will use the URL of the files to group the statistics retrieved from each one; that is a common practice in organizing files in folders according to its subject. We believe that this approach will permit a good classification, not by the filter's precision, but by the use of the possible relationship between the images in the final blocking decision.

Keywords: Squid, pornography, content filtering, computer vision, skin detection.

Sumário

Agradecimentos

Durante a pesquisa e realização deste trabalho conversei com várias pessoas sobre o tema de meu trabalho de conclusão de curso, e conforme comentários e questões levantadas pude avaliar melhor os problemas que seriam enfrentados durante o desenvolvimento do software proposto.

O principal ponto que levantou dúvidas no início era: "Como fazer para que imagens não fossem incorretamente classificadas como pornografia?", dado que a idéia inicial do projeto era de bloquear conteúdo não acessado, mesmo sem a certeza de seu teor. Por fim acabei convencido, com os argumentos de meu amigo Frederico Zanqueta Poleto, de que não era uma idéia justa bloquear antecipadamente URLs sem ter argumentos convincentes para isto.

Ao professor Dr. Roberto Hirata Jr. devo agradecimentos por me ajudar com a execução deste projeto, principalmente com algumas idéias e pela orientação sobre os procedimentos a serem tomados, ainda que estes não tenham sido seguidos fielmente durante toda a disciplina. Peço desculpas a ele por não ter me dedicado suficientemente a parte de visão computacional, a qual era o foco inicial de nossos trabalhos.

Agradeço também aos amigos da equipe de corrida PlayTeam, Flávio Hernandes e Roberto Cyrillo Junior, pela fotografia utilizada para análise de cores, ao colega advogado Dr. Marco Antônio "Chezmad", pelas noções de Direito, e ao amigo psicólogo Felipe Pierry, pelas observações sobre a influência de material pornográfico sobre crianças.

Por fim gostaria também de agradecer minha namorada Erica Cristina Manso, minha irmã Aline Lemes da Silva e meu amigo Celio Santana, pelo apoio tanto na elaboração deste projeto quanto na conclusão de meu curso de graduação.

Capítulo 1

Introdução

1.1 Objetivos

Esta monografia tem como objetivo mostrar como filtros de conteúdo podem ser utilizados em conjunto com uma estrutura de dados apropriada, para o bloqueio de conteúdo impróprio, seja na forma de imagens de pessoas nuas ou de textos pornográficos, contido nas páginas da internet.

Neste documento, será mostrado como a implementação de um software classificador pode ser utilizada em conjunto com um servidor proxy[1] a fim de bloquear qualquer tipo de conteúdo que possa ser tratado com métodos de classificação computacionais, seja através de algoritmos de visão computacional, para o caso de imagens, ou com busca por palavras ou frases, no caso de textos, de modo que seja possível aferir algo sobre seu conteúdo.

O sistema idealizado neste trabalho não será um servidor proxy, mas sim um software complementar ao programa Squid-cache[2], um popular servidor proxy distribuído sob licença GNU/GPLv2, amplamente testado e utilizado mundialmente. Apesar do projeto ter sido implementado para ser integrado ao software Squid-cache, é possível que outros softwares similares se tornem compatíveis com este projeto simplesmente implementando o protocolo de pesquisa descrito na parte experimental desta monografia.

[1]Servidor proxy: É um software o qual possibilita que uma máquina possa intermediar requisições de acesso de outras máquinas a páginas na internet, eventualmente aplicando algum tipo de restrição.

[2]Mais informações em http://www.squid-cache.org

1.2 Motivação

Devido a popularização do acesso à internet, empresas começaram a explorar este ágil meio de comunicação para vender produtos e serviços de uma maneira nova, com baixo custo e com abrangência mundial. Algumas destas empresas vislumbraram a idéia de vender conteúdo adulto, na forma de textos, fotos, vídeos, etc., devida a facilidade de prover este tipo de material aliada ao anonimato que a internet oferece.

Como a necessidade de identificação neste tipo de website é em muitos casos nula e dada a facilidade com que são espalhadas propagandas sobre os mesmos, é bastante provável que crianças eventualmente sejam atingidas por este tipo de conteúdo sem que haja intenção por parte delas. Desta forma, uma criança pode ser facilmente exposta a conteúdo pornográfico, o qual eventualmente pode influenciar seu comportamento no futuro, principalmente nos casos onde há falta de educação sobre sexo por parte dos adultos.

Outros casos também merecem atenção, como o acesso indevido a este tipo de material dentro de empresas ou locais públicos. Em muitos casos funcionários acessam websites pornográficos dentro de suas empresas, o que pode gerar desde perda de produtividade até demissão por justa causa. Em locais públicos o acesso indiscriminado deste tipo de conteúdo pode levar ao constrangimento de outras pessoas ou eventualmente até vir a ser considerado um crime por atentado ao pudor.

Diante destes problemas surgiu a motivação deste trabalho, que visa bloquear qualquer tipo de conteúdo pornográfico disponível em websites na internet.

1.3 O software

Durante o andamento da disciplina MAC0499, desenvolvi o software chamado **SquidPCB - Squid-cache Pornography Content Blocker** afim de demonstrar a possibilidade de utilização de filtros de conteúdo no acesso a internet.

O software foi idealizado como um programa auxiliar ao Squid-cache, um popular *servidor proxy* desenvolvido como software livre e amplamente utilizado, afim de focar os trabalhos na detecção e identificação do conteúdo pornográfico.

Desenvolvido sob licença de uso GNU/GPLv3, o SquidPCB está disponível para download no site SourceForge.net, e espera-se que ele seja utilizado e contribua para o tratamento dos problemas acima relacionados.

Capítulo 2

Fundamentos

2.1 Revisão bibliográfica

2.1.1 Artigos

Como ponto de partida para o desenvolvimento desta monografia, havia a necessidade de estudar uma maneira de armazenar as informações que este software deveria coletar e como estas informações seriam obtidas. Utilizando conhecimentos básicos de estruturas de dados, foi possível observar que a melhor estrutura que permitiria tanto o armazenamento de objetos relacionados como a recuperação de dados de maneira bastante rápida seria uma estrutura em árvore de múltiplas folhas. Desta forma, a leitura de artigos se concentrou em maneiras de obter a probabilidade de um objeto ser pornografia.

O único artigo lido durante a execução desta monografia foi redigido por três pesquisadores americanos e é entitulado "Finding naked people"[1]. Este artigo fala sobre um método para detecção de imagens de corpos nús utilizando um método de seleção de pixels com cores similares a da pele, realizando análises, nos casos em que a proporção destes pixels é significativa, que identificam possíveis partes do corpo como tronco e membros.

Com uma série de padrões pré-definidos de posicionamento destas partes do corpo, o método descrito conseguiu em seus testes classificar a imagem como contendo um corpo em pouco mais de 50% dos casos. Entretanto, devida a dificuldade de classificar corretamente uma imagem como pornografia ou não, pareceu bastante razoável utilizar a premissa contrária, diminuindo muito a probabilidade da imagem conter pornografia nos casos onde não havia cor de pele suficiente. Nos testes reportados pelo artigo lido, 92% das imagens de teste (não pornográficas) foram eliminadas dos testes mais complexos pela baixa proporção de pixels "cor de pele".

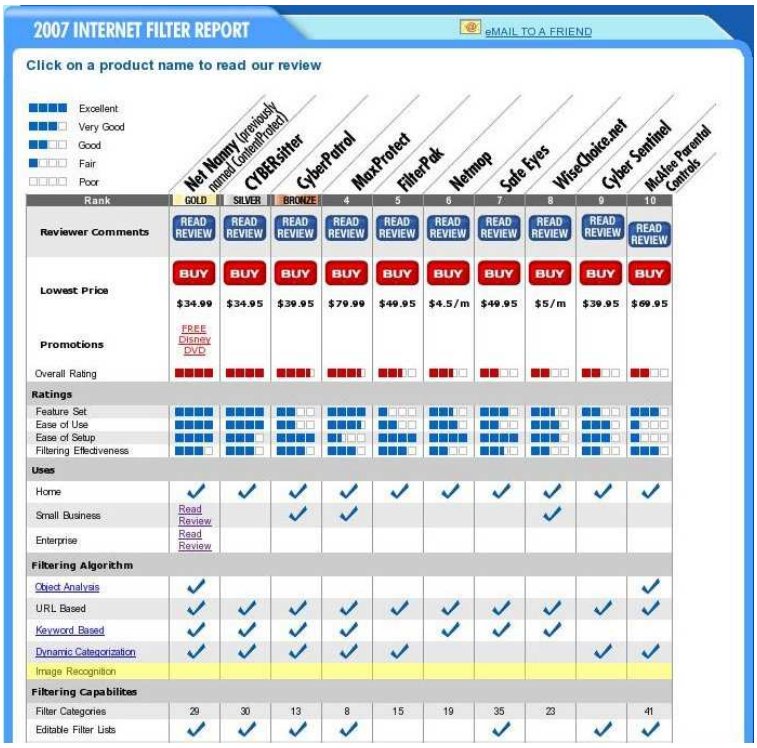
2.1.2 Análise de softwares existentes

Existem diversas maneiras de se bloquear conteúdo pornográfico na internet. Dentre as mais comumente utilizadas estão a utilização de bloqueios de acesso pela ocorrência de palavras pre-determinadas, muitas vezes através de expressões regulares, e a pesquisa em listas negras de sites pornográficos, ambas requerendo a classificação manual ou atualizações constantes.

Os produtos comerciais encontrados são em geral voltados a proteção de crianças, principalmente pela facilidade que uma criança tem de ser abordada na internet por usuários pedófilos. Dentre estes softwares, os mais bem avaliados pelo site "Top 10 Reviews"[1] são:

- Net Nanny (Antigo ContentProtect)
- CYBERsitter
- CyberPatrol

Entretanto, estes softwares se baseiam também na busca de palavras em textos ou URLs e em listas pré-definidas de sites. Também é possível observar que, dentre todos os softwares listados, nenhum possui a capacidade de realizar funções de reconhecimento de imagens.



[1] http://internet-filter-review.toptenreviews.com/

Dentre as soluções freqüentemente adotadas por empresas, a grande maioria se baseia em restrições simples, utilizando listas de sites e expressões regulares para bloqueio das URLs solicitadas. Durante as pesquisas, foi encontrado um software chamado "Dansguardian", utilizado com pouca freqüencia pelas empresas pesquisadas, que realiza análises baseadas no conteúdo de arquivos texto, bloqueando os arquivos que atingissem um limite pré-definido.

Foi cogitada no início dos trabalhos a idéia de desenvolver um módulo ou *"patch"* para este software, devido a receptividade do desenvolvedor em resposta a uma pergunta da seção *FAQ* de seu site[2]:

> ***"8) Does it check images for pornography?***
> *No. To do this it could unencode the image file and scan for high percentages of skin tone. This would take a lot of processing power. It is possible and has been done by other people. I may consider adding this to my program. If anyone wants to contribute and write this bit I would be interested to hear. I'm not sure how it would work for black or asian people who have a much darker skin tone?"*

Apesar de ser entusiasmante a idéia de agregar valor a um software já existente, esta idéia foi abandonada dada a licença de uso imposta pelos desenvolvedores, a qual segue a GPLv2, porém impedindo seu uso do software para fins comerciais.

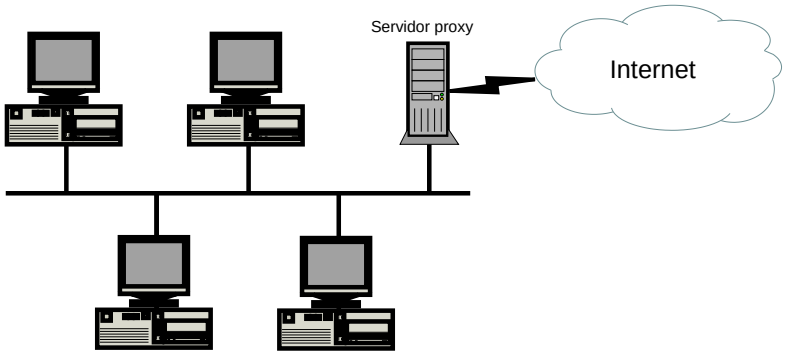
Dada a pesquisa realizada, foi verificado que a utilização de filtros de imagens não é adotada por nenhuma solução encontrada, o que dará a solução proposta por esta monografia o caráter de originalidade sobre as soluções existentes. Porém convém ressaltar que o software a ser desenvolvido tem como objetivo somente filtrar objetos acessados através de um servidor proxy, podendo não ser eficiente em casos de mensageiros instantâneos ou salas de bate-papo.

2.2 Squid-cache

2.2.1 Descrição

O Squid-cache é um servidor proxy com recursos de *cache* de arquivos, distribuído sob a licença GNU/GPL versão 2, e é amplamente utilizado por milhares de empresas no mundo inteiro, devido principalmente a sua confiabilidade e seu código fonte aberto.

[2]http://dansguardian.org/



Este software atua como um intermediário de requisições, permitindo que máquinas clientes acessem servidores através de protocolos como o HTTP, HTTPS e FTP, dentre outros. O principal benefício proporcionado por este tipo de serviço está na utilização de um *cache* dos arquivos acessados por todas as máquinas, de forma que um arquivo acessado por uma máquina pode ser entregue a outra sem que o mesmo seja descarregado da internet duas vezes. Além de evitar o tráfego desnecessário na conexão internet, este recurso pode tornar o acesso a páginas bem mais rápido.

Outro motivo para seu uso está na facilidade de gerenciamento e controle do conteúdo acessado, uma vez que toda requisição de acesso fica centralizada. Utilizando um servidor proxy é possível requisitar a autenticação do usuário, permitindo o acesso somente a determinados sites ou bloqueando o acesso a outros, com regras que podem incluir deste o tipo de requisição (GET, POST, CONNECT, ..) até o horário desta.

Baseado justamente nesta funcionalidade de controle de acesso que um servidor proxy foi escolhido para auxiliar na implementação de um sistema de filtragem de conteúdo pornográfico na internet.

2.2.2 Como funciona

O Squid-cache funciona através de um processo executado em background no sistema operacional, escutando requisições, analisando as regras de acesso e recuperando os arquivos solicitados. As requisições são realizadas através de *sockets*[3] TCP de maneira bastante parecida com as que são feitas diretamente aos servidores web utilizando o protocolo HTTP.

[3]Sockets: São canais bidirecionais de comunicação entre processos, utilizando a política FIFO (First In First Out).

Exemplo:

```
GET http://www.sitequalquer.com.br/pasta1/pasta2/arquivo.ext HTTP/1.1
[Cabeçalhos de requisição]

HTTP/1.1 200 OK
[Cabeçalhos da resposta]

[Arquivo]
        ⋮
```

Com todas as informações sobre uma URL requisitada, o Squid-cache verifica se a requisição deve ser atendida ou não. Utilizando uma lista de regras definidas em seu arquivo de configuração, uma lista de condições é criada tanto para permitir quanto para negar o acesso a uma URL através da opção "http_access". Esta opção, além da ação a ser tomada, recebe uma lista de condições denominadas ACLs (*Access Control List*) que devem todas retornar o valor booleano "verdadeiro" para que a ação seja aplicada. Estas ACLs são criadas utilizando funções pré-definidas que verificam informações sobre a requisição do usuário e permitem que controles sejam criados de maneira bastante simplificada.

Abaixo segue um exemplo de configuração possível, onde as requisições serão autorizadas se o destino da requisição for algum servidor do domínio "usp.br" e se a máquina cliente pertencer a rede 10.0.0.0/8, sendo negadas caso contrário.

```
acl qualquerRede src 0.0.0.0/0.0.0.0
acl redeLocal src 10.0.0.0/255.0.0.0
acl siteUSP dstdomain .usp.br

http_access allow redeLocal siteUSP
http_access deny qualquerRede
```

Neste exemplo é fácil mostrar que uma vez que todas as condições de uma opção "http_access" foram satisfeitas a ação é tomada e nenhuma regra subseqüente é avaliada.

Uma vez que a requisição tenha sido autorizada pelas ACLs, o arquivo referenciado pela URL é recuperado, e de acordo com algumas outras configurações do proxy que definem quais arquivos serão armazenados em *cache*, este é armazenado em memória ou em disco, permitindo que futuras requisições não precisem recuperá-lo novamente.

Desta forma é possível solicitar ao servidor proxy qualquer arquivo acessível

através de uma URL utilizando algum dos protocolos suportados podendo utilizar tanto controles de acesso como o *caching* de arquivos.

2.2.3 Integração

A integração do SquidPCB com o servidor Squid-cache ocorre utilizando a ACL denominada "external", que utiliza um programa externo para avaliar se uma requisição deve ou não ser atendida. Entretanto, para utilizar esta ACL é necessário além de informar o caminho do programa, definir que tipos de dados serão passados ao programa externo.

Como o SquidPCB é um software independente que não tem relação direta com o Squid-cache, um programa adicional chamado SquidExternalACL foi implementado para realizar a integração entre estes dois softwares utilizando *sockets* TCP como meio de comunicação.

Este programa recebe do Squid-cache uma URL separada em três partes: Protocolo utilizado, endereço de destino e o caminho do arquivo a ser acessado. A configuração do Squid-cache para esta ACL externa deve ficar semelhante a:

```
external_acl_type SquidPCB %PROTO %DST %PATH /usr/bin/SquidExternalACL.sh
acl SquidPCBapproved external SquidPCB
```

E as linhas que autorizam os acessos, de acordo com nosso exemplo, devem ficar semelhantes a:

```
http_access allow redeLocal siteUSP SquidPCBapproved
http_access deny qualquerRede
```

Ou:

```
http_access deny !SquidPCBapproved
http_access allow redeLocal siteUSP
http_access deny qualquerRede
```

2.2.4 Vantagens e desvantagens

O sistema idealizado foi previsto para funcionar como uma ACL externa do programa Squid-cache. Foram cogitadas outras opções, principalmente por questões de performance, inclusive a hipótese de criar um *patch* para o Squid-cache manter todo o serviço de classificação de conteúdo.

Entretanto, a grande vantagem de optar por um programa que se integra desta maneira ao Squid-cache é a de permitir que os administradores de redes possam implantar o SquidPCB em seus servidores sem muito trabalho e com facilidade para habilitar ou desabilitar sua utilização.

Dentre as vantagens que podemos apontar:

- **Facilidade de integração:** Da maneira com que é desenvolvido, o SquidPCB pode ser implantado em servidores de produção muito facilmente, e da mesma maneira ele pode ser ativado ou desativado completamente;

- **Independência do Squid-cache:** Independente de como o Squid-cache evolua, as opções para o uso de ACLs externas deverão ser mantidas e o funcionamento do SquidPCB continuará possível com estas versões futuras;

- **Distribuição de sistemas:** Devido a utilização de conexões TCP para a comunicação entre as ACLs externas e o SquidPCB, os softwares não precisam rodar na mesma máquina, sendo possível a utilização de uma máquina com mais capacidade de processamento e memória para o SquidPCB e máquinas com baixo poder de processamento, mas com bastante memória e disco para o proxy. Esta abordagem pode inclusive permitir a utilização do mesmo servidor SquidPCB para vários *"sites"*[4].

Já em relação as desvantagens:

- **Mais lento:** Devido a necessidade de comunicação com outro processo, utilizando um canal de comunicação independente de sistema operacional, esta forma de integração certamente é mais lenta, sendo penalizada ainda mais em casos onde somente há somente um processador disponível e ambos os processos estão sendo executados na mesma máquina (devido a espera de I/O e trocas de contexto pelos processos);

- **Maior complexidade:** Nesta abordagem existe de fato um aumento da complexidade na implantação de um filtro Squid-cache + SquidPCB, nos casos onde o administrador da rede não possui ainda o Squid-cache, sendo necessária sua instalação para posterior integração com o SquidPCB.

[4] "Site": Um "site", no jargão dos administradores de rede, refere-se a um local onde uma rede com estações e/ou servidores estão localizados fisicamente.

2.3 Imagens

Estabelecido o objetivo inicial deste trabalho como sendo a filtragem de conteúdo pornográfico utilizando um servidor proxy, será descrito nesta seção seu diferencial que é justamente a análise de imagens utilizando métodos de visão computacional.

A princípio a tarefa de detecção da presença de pessoas nuas em uma foto pode parecer simples utilizando algum analizador de imagens, porém muitas imagens podem gerar falsos positivos, como fotos de rostos ou mesmo objetos que tenham cores similares a da pele.

A presença de cores tom de pele é um bom indício de que a imagem pode ser de um corpo nú, porém somente baseado nesta informação não podemos afirmar nada sobre seu conteúdo. No artigo "finding naked people"[1], os autores descrevem esta técnica como uma análise inicial para descartar imagens com baixa proporção de pixels "cor de pele", realizando uma procura por padrões geométricos de posicionamento do tronco e membros para as imagens restantes e só declarando como contendo pessoas nuas as imagens nas quais algum padrão é encontrado.

Neste projeto somente a proporção de pixels "cor de pele" de cada imagem será levada em consideração, sem realizar nenhum teste de complexidade computacional mais elevada, procurando utilizar as informações sobre os demais arquivos possivelmente relacionados para aferir a possibilidade da imagem ser de uma pessoa nua ou não.

É importante notar que apesar desta parte do projeto ter sido concebido de forma bastante simples, é possível que novos filtros ou classificadores possam ser implementados de maneira bastante simples a fim de aumentar a confiabilidade do sistema. Mais informações a respeito podem ser encontradas na seção 4.8.

2.3.1 ImageJ

Dado que parte deste trabalho requeria a análise de várias imagens em busca por padrões, surgiu a necessidade de utilizar algum software para realizar estes procedimentos. Desta forma, utilizando uma ferramenta citada durante a disciplina de "Introdução a Visão Computacional e Processamento de Imagens", optou-se por utilizar o software ImageJ[2] em conjunto com seu *plugin* 3D Color Inspector[3].

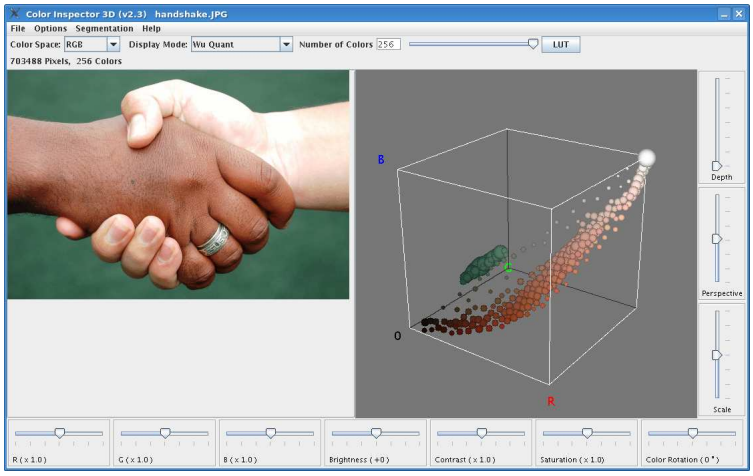
Tanto o programa principal quanto seu *plugin* possuem seu código fonte aberto e são de domínio público e possuiam as funcionalidades necessárias para os testes realizados. Através destes programas foi possível analisar várias imagens comuns e com nús, percebendo como as cores podem ser

visualizadas e qual o padrão seguido por imagens que tinham pixels com cor de pele.

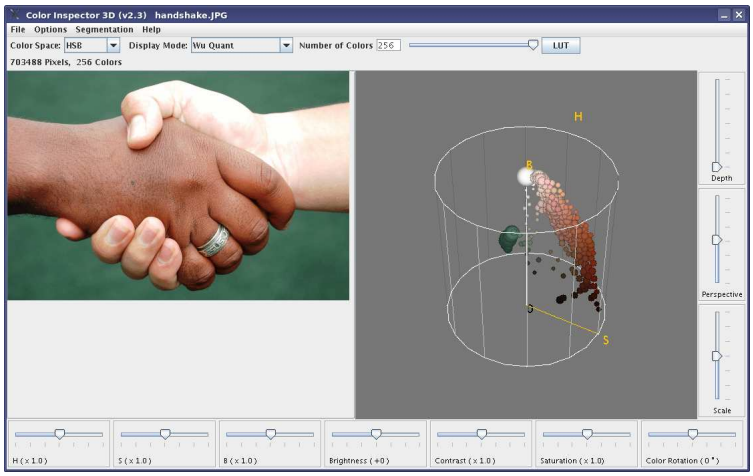
2.3.2 Espaço de cor

Uma das principais tarefas para a segmentação de tons de pele foi a escolha de uma regra para determinar se um pixel seria classificado como pele ou não.

Inicialmente foram realizadas algumas análises com o software ImageJ usando o espaço de cor RGB. Foi possível através dele observar que os pixels identificados visualmente como cor de pele se encontravam próximo da diagonal principal do cubo de visualização RGB, tendendo mais para a cor vermelha.



Porém, seguindo as orientações do Prof. Hirata, testes com outros espaços de cor foram realizados, e percebeu-se que utilizando a visualização HSB (Hue/Saturation/Brightness) os pixels com cor de pele ficam bem destacados, de forma que é bastante simples separar parte do cilindro de visualização como sendo cor de pele.



Desta forma, o espaço de cor HSB foi o escolhido para auxiliar a classificação de um pixel como sendo cor de pele ou não.

2.3.3 Artigo relacionado

Posteriormente a escolha do método que seria utilizado para classificar um pixel como cor de pele, foi encontrado um artigo bastante interessante que descrevia muito bem o problema de segmentação de cor de pele que foi tratado nesta monografia. O artigo tem o título "A Survey on Pixel-Based Skin Color Detection Techniques"[4] e apresenta várias soluções possíveis e suas taxas de acerto.

Um dos métodos descritos por este artigo chamou a atenção pela sua simplicidade, que com sete comparações utilizando o espaço de cor RGB, determinava se o pixel era cor de pele ou não. Porém, como houve um trabalho experimental em determinar um método usando a representação HSB, esta parte da monografia foi mantida, porém não impedindo que em versões futuras do software implementado o método seja revisto.

2.3.4 Experimento

Foram realizados alguns experimentos utilizando a transformação de RGB para HSV, a qual para a análise realizada produz o mesmo resultado que o HSB, e verificou-se empiricamente que seu funcionamento é bastante satisfatório para as necessidades desde projeto.

Para realizar a conversão do formato RGB para o HSB no software implementado foram utilizadas as fórmulas abaixo retiradas da biblioteca online Wikipedia[5]:

[5]HSB no Wikipedia: http://en.wikipedia.org/wiki/HSV_color_space

Variáveis de entrada:

$r, g, b \in [0, 1]$
$min = min\{r, g, b\}$
$max = max\{r, g, b\}$

$$h = \begin{cases} 0^\circ & se \quad max = min \\ 60^\circ \times \frac{g-b}{max-min} & se \quad max = r \quad e \quad g \geq b \\ 60^\circ \times \frac{g-b}{max-min} + 360^\circ & se \quad max = r \quad e \quad g < b \\ 60^\circ \times \frac{b-r}{max-min} + 120^\circ & se \quad max = g \\ 60^\circ \times \frac{r-g}{max-min} + 240^\circ & se \quad max = b \end{cases}$$

$$s = \begin{cases} 0 & se \quad max = 0 \\ \frac{max-min}{max} & se \quad max > 0 \end{cases}$$

$b = max$

Utilizando os intervalos abaixo, obtidos através de análises de várias imagens com o software ImageJ, isolou-se de maneira bastante razoável os pixel com cores correspondentes ao tom de pele:

$$h \in [3, 30] \quad s \in [0.05, 0.90] \quad v \in [0.10, 0.98]$$

Alguns resultados obtido com estes filtros podem ser visualizados abaixo, onde todos os pontos classificados como cor de pele foram alterados para branco, enquanto que os demais foram alterados para preto.

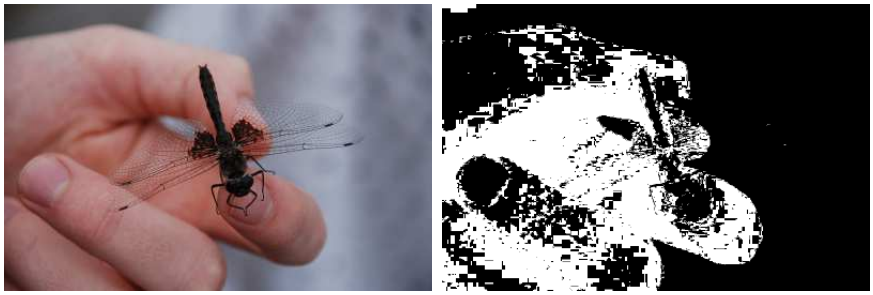


Foto por Marcus Obal sob licença GNU Free Documentation License
Imagem filtrada contendo 31,17% de pixels com cor de pele.

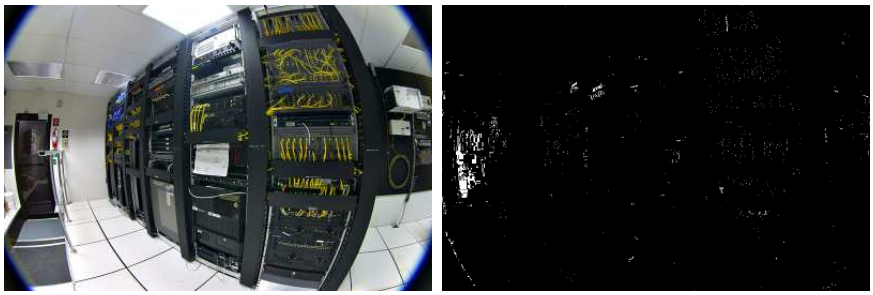


Foto por Gregory Maxwell sob licença GNU Free Documentation License
Imagem filtrada contendo 0,86% de pixels com cor de pele.

É possível observar claramente que mesmo em imagens onde não há visualmente pixels cor de pele, o algoritmo marca alguns pixels como cor de pele.

Uma solução possível para diminuir ainda mais a marcação de pixels como cor de pele seria verificar o tamanho de cada região conexa marcada e descartá-la caso o seu tamanho seja irrelevante perto do tamanho total da imagem. Desta forma somente pedaços significativos de cor de pele seriam considerados, eliminando muitos ruídos.

Capítulo 3

Proposta

Dado o problema a ser tratado, a maneira mais eficiente que foi encontrada para realizar o bloqueio de conteúdo impróprio aos usuários da internet é através de métodos de classificação de conteúdo, usando algoritmos de análise de textos e visão computacional, para realizar uma classificação automática baseada em parâmetros definidos pelo administrador do sistema.

Para este tipo de solução não envolver uma complexidade ainda maior, optou-se por criar um sistema em Java que irá analisar os arquivos acessados e manter uma base de dados de URLs, o qual será integrado ao servidor proxy Squid-cache, adicionando uma nova regra ACL (vide seção 2.2.2) que irá verificar se o objeto solicitado é ou não pornografia.

O sistema desenvolvido será baseado em vários processos, sendo que o processo principal irá manter a base de dados de URLs e os processos secundários atuarão em conjunto com o Squid-cache, recebendo cada URL solicitada e verificando a classificação da mesma. *Sockets* TCP serão utilizados para comunicação entre processos a fim de possibilitar a distribuição de tarefas entre máquinas, assim como permitir a conexão de outros sistemas a este.

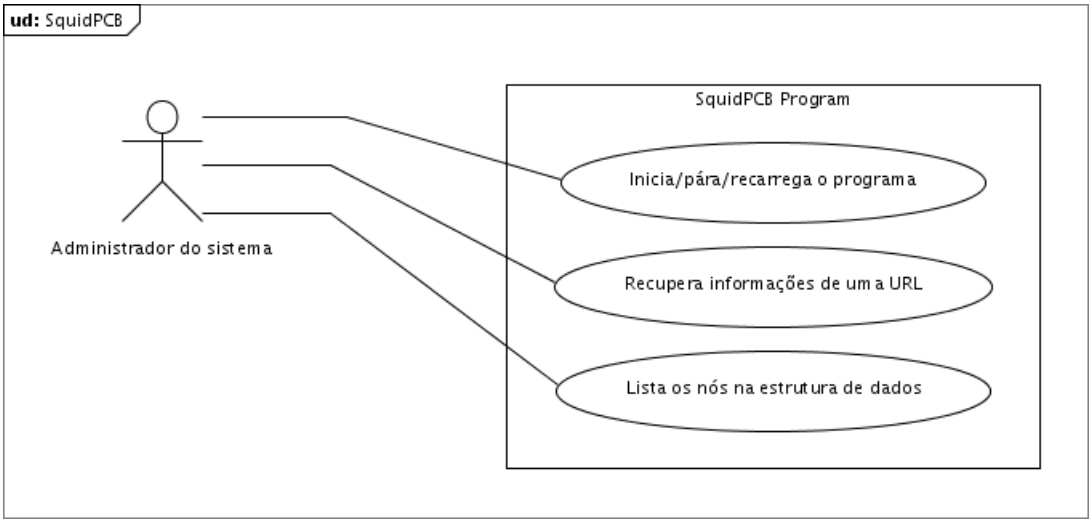
Para cada arquivo acessado será feita uma análise, quando possível, para adicionar novas informações ao sistema. Esta análise deverá ser feita pelo processo principal, isolando a funcionalidade de análise em uma única máquina. Porém é previsto que em planos futuros o sistema poderá ser flexibilizado de forma que os clientes também possam fazer análises e enviar informações ao servidor, aumentando a eficiência em alguns casos.

3.1 Motivação

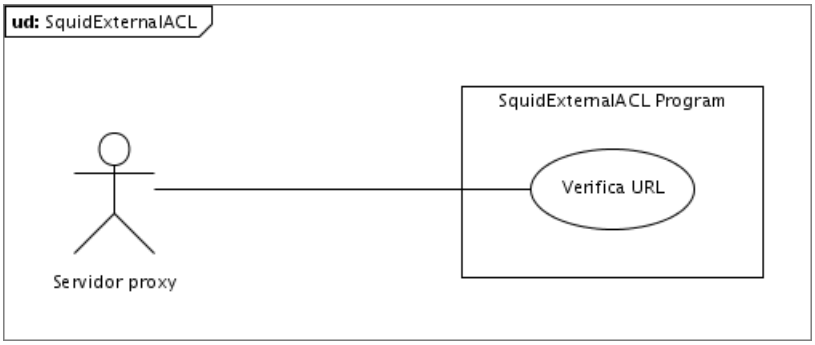
Esta proposta foi escolhida basicamente pelas vantagens enumeradas na seção 2.2.4, que permitem uma integração rápida com um servidor proxy Squid-cache já implantado, além de proporcionar a distribuição dos serviços envolvidos.

A linguagem escolhida para o desenvolvimento deste sistema foi a Java, devido a maior familiaridade do autor com a mesma, além da facilidade de desenvolvimento por não ter que utilizar bibliotecas externas para tarefas como comunicação por *sockets*, ou manipulação de imagens. Também foi decisiva a modularização proporcionada pela orientação a objetos, e a portabilidade oferecida pela linguagem.

Os diagramas de *"use cases"* levantados para este sistema se encontram abaixo e são bastante simples devida a pouca iteração que ocorre entre usuários ou programas com o sistema planejado.



Processo principal



Processo secundário

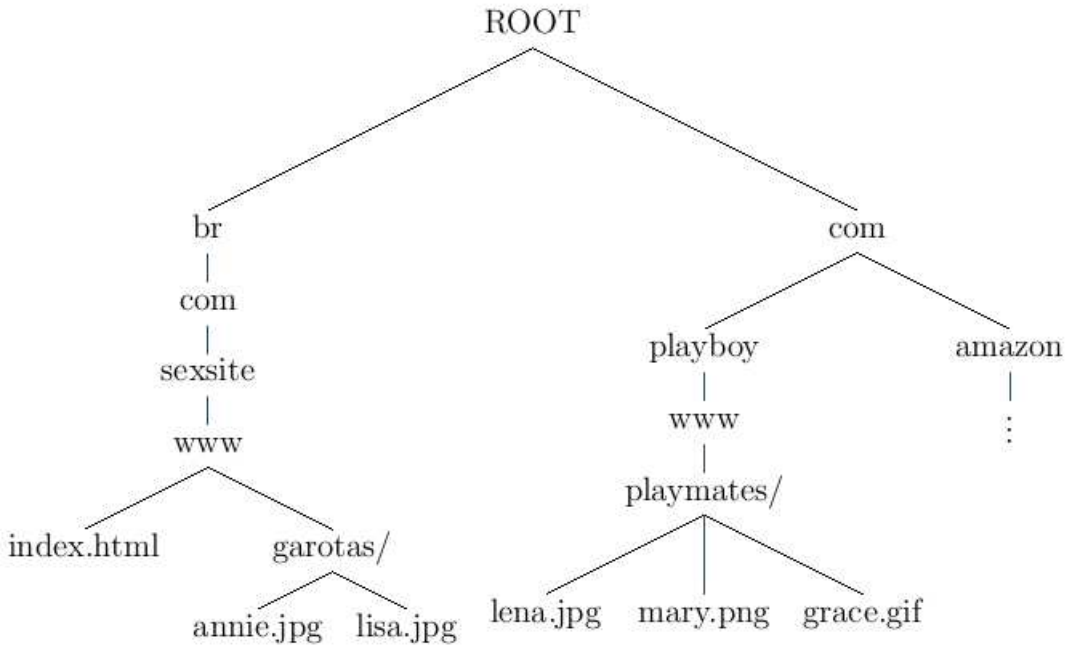
3.2 Estrutura de dados

O armazenamento das informações sobre as URLs será feito em uma estrutura de dados baseada em árvores, de forma que as informações sobre objetos considerados próximos sejam armazenados em nós de mesmo nível e relacionados através dos nós em seu caminho até a raiz. A principal vantagem desta estrutura de dados é a rapidez no acesso a uma informação armazenada, dado que possivelmente o sistema receberá dezenas de requisições por segundo.

A estrutura de árvore imaginada foi concebida tomando como inspiração as árvores geradas por cadeias de Markov de tamanho variável. De maneira análoga, onde de acordo com os eventos passados pode-se afirmar algo sobre a probabilidade do próximo evento, pode-se também de acordo com o caminho de uma URL afirmar que determinados arquivos estão relacionados, simplesmente pelo fato do ser humano tender a organizar os arquivos em pastas, de acordo com sua afinidade.

Como uma lista de URLs com suas probabilidades só iria permitir que o sistema evitasse rodar os algoritmos de classificação duas vezes para o mesmo arquivo, percebeu-se a necessidade de se utilizar algum método que dado o caminho na árvore, pudesse auxiliar no processo de determinação sobre a existência de pornografia no objeto em análise.

Cada nó da arvore idealizada representa um pedaço da URL solicitada por um servidor proxy, de forma que tem-se uma estrutura parecida com a da imagem abaixo:



3.3 Métodos de análise

3.3.1 Textos

Para a análise de arquivos texto, será empregado um método de busca no arquivo por palavras ou frases contidas em um dicionário, utilizando uma estrutura de autômato determinístico. Para cada palavra será atribuído um peso, de forma que caso o texto alcance determinado peso ele será considerado como um texto pornográfico.

Também haverá a necessidade de "normalizar" este peso de acordo com o tamanho T, em bytes, do texto. Para isto, iremos dividir o peso obtido por um coeficiente k resultante da seguinte fórmula:

$$k = \begin{cases} log_2\left(\frac{T}{4096}\right) & se \quad T \geq 8192 \\ 1 & c.c. \end{cases}$$

Estes valores foram escolhidos através de uma média de tamanho de vários arquivos HTML, utilizando uma amostra de 305.813 arquivos[1], os quais indicaram que arquivos com *"mime-type"* igual a "text/html" possuem o tamanho médio é de 6.939 bytes. Desta forma é razoável pensar que arquivos

[1]Valores obtidos pelo servidor proxy da empresa Koho Comércio de Equipamentos e Serviços de Informática Ltda.

maiores que 8.192 bytes devam sofrer algum tipo de normalização em relação ao seu tamanho.

3.3.2 Imagens

Referente a análise das imagens, a primeira idéia é de atribuir uma probabilidade somente analisando o percentual de pixels considerados cor de pele na imagem acessada. Entretanto, casos especiais - como imagens muito pequenas - devem ser tratadas de outra forma por não ser possível afirmar algo sobre a mesma.

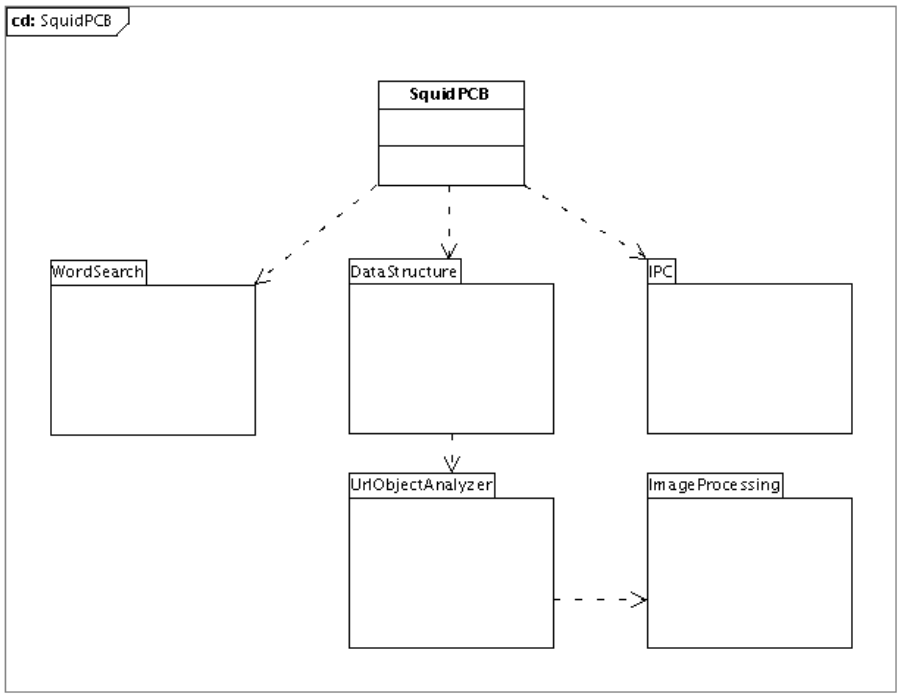
Outras idéias a serem analisadas são a eliminação de partes conexas de pixels cor de pele que não atinjam um percentual mínimo em relação ao tamanho da imagem e a eliminação de pequenos buracos nas regiões conexas com a operação de morfologia de imagens denominada fechamento, composta por uma dilatação e uma erosão necessariamente nesta ordem.

Um exemplo da operação de fechamento pode ser observado abaixo:

Antes *Depois*

3.4 Organização do sistema

As classes que serão utilizadas pelo sistema foram divididas em cinco pacotes, conforme o diagrama de classes abaixo.



A função de cada classe planejada segue abaixo:

- **ConfigurationParser:** Esta classe será do tipo utilitária e tem como papel analisar o arquivo de configuração, receber opções de linha de comando, definir parâmetros aceitos e valores padrões para as opções e repassar a qualquer classe os parâmetros de configuração solicitados;
- **Logging:** Esta classe também será do tipo utilitária, responsável por gerar os logs do funcionamento do programa usando sete níveis de "verbose" pré-definidos;
- **SquidExternalACL:** Programa a ser chamado pelo Squid-cache para análise das URLs. Ele receberá a URL do Squid-cache e se conectará ao programa SquidPCB para solicitar a análise do arquivo referenciado;
- **SquidPCB:** Responsável pela inicialização de todo o sistema de análise das URLs. Esta classe deverá instanciar um objeto DataStructureManager, dois objetos WordSearch e um objeto ConnectionServer para conexão dos programas SquidExternalACL com este programa;
- **DataStructureManager:** Esta classe será responsável por manter a estrutura de dados do programa SquidPCB. Ela conterá a referência para a árvore de dados que será armazenada e cuidará de todas as operações de *swapping* e persistência em disco;

- **UrlNode:** Este objeto representará cada nó da árvore de dados, e poderá representar tanto parte de um nome de máquina, quanto um nome de pasta ou de arquivo;
- **UrlParser:** Este objeto será responsável por receber URLs e separá-las em pedaços que serão utilizados para indexar os nós da árvore de dados;
- **SkinDetector:** Classe responsável por possuir métodos de tratamento de imagem a fim de determinar a proporção de pixels cor de pele ou aplicar filtros em imagens. Quaisquer aperfeiçoamentos de visão computacional ou mudanças de foco no filtro de imagem serão tratadas nesta classe ou adicionadas em classes do mesmo pacote;
- **ConnectionClient:** Realizará a intercomunicação entre o sistema principal e os programas SquidExternalACL que serão invocados pelo Squid-cache para analisar cada URL;
- **ConnectionServer:** Esta classe tem como função realizar a comunicação entre o programa principal (SquidPCB) e o verificador de URLs (SquidExternalACL), rodando a partir do programa principal e escutando os pedidos de conexão TCP na porta especificada no arquivo de configuração;
- **ConnectionThread:** Esta classe será formada por objetos que extendem a classe Thread e serão criados pelo processo ConnectionServer para tratar cada conexão de cliente recebida;
- **UserAuthenticator:** Esta classe terá somente o papel de autenticar usuários e senhas, e armazenar as permissões para cada um destes. O armazenamento destas informações será feito em um arquivo de texto;
- **UrlObject:** Responsável por recuperar o arquivo a partir de uma URL, fazendo o download da mesma a partir de um servidor proxy especificado;
- **UrlObjectAnalyzer:** Classe principal para a análise dos arquivos quanto a probabilidade de serem pornografia. Ela recebe uma URL e realiza a análise apropriada de acordo com cada arquivo.
- **AutomataState:** Classe que representa os nós de um autômato gerenciado pelo objeto WordSearch;

- **ScoreBoard:** Classe para armazenamento de um placar de ocorrências e pontos de palavras encontradas pelo objeto WordSearch;
- **WordSearch:** Classe utilitária para busca de palavras em textos. Ela é responsável por manter um automato determinístico eficiente de modo a somar pontos em um objeto ScoreBoard para cada palavra pré-definida encontrada no texto.

3.5 Integração com o Squid-cache

A integração deste sistema com o software Squid-cache será feita utilizando o recurso de "ACL externa" provida por este programa, de forma que não seja necessária a aplicação de nenhum *patch* ao programa Squid-cache, e a integração possa ser feita da maneira mais simplificada possível.

O sistema SquidPCB será composto por um processo servidor, que irá armazenar todas as informações e realizar o processamento das imagens, e por um ou mais processos clientes que irão ser chamados pelo Squid-cache, proporcionando de fato a integração do conjunto.

Capítulo 4

Parte Experimental

4.1 Planejamento

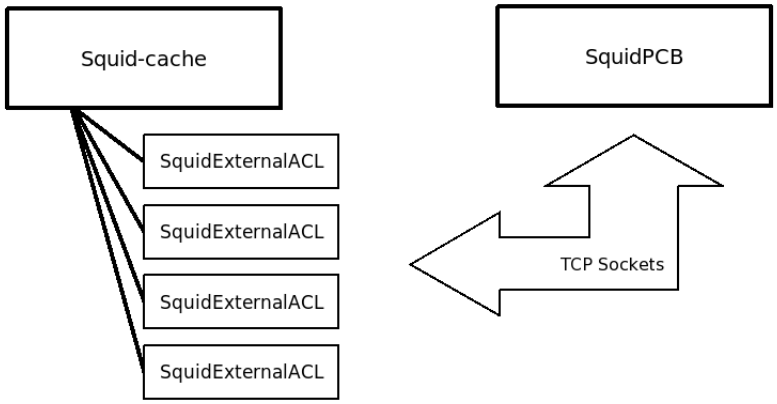
O planejamento inicial da parte experimental do SquidPCB foi baseado no diagrama classes e na descrição das funcionalidades de cada classe. Entretanto, o software sofreu várias modificações em sua estrutura conforme a progressão do desenvolvimento.

Acredita-se que este problema ocorreu devido ao pouco tempo gasto com este planejamento e a falta de um colega de trabalho para discutir alternativas sobre as melhores maneiras de executar as várias partes do projeto. Certamente o dispêndio de mais horas com planejamento refletiria em um desenvolvimento mais ágil, diminuindo bastante o tempo gasto com a refatoração do código.

4.2 Detalhes da integração

A integração do Squid-cache com o SquidPCB ocorre através de um recurso nativo do programa Squid-cache chamado ACL externa.

O Squid-cache possui uma opção de informar um programa externo, o qual receberá um ou mais parâmetros de uma requisição solicitada por um cliente. Este programa externo - que em neste projeto será chamado de SquidExternalACL - receberá em sua entrada padrão alguns valores que o permitem saber qual foi a URL solicitada, para então acessar o processo servidor do SquidPCB para solicitar informações sobre esta URL.



4.2.1 O protocolo

O acesso ao processo servidor é feito utilizando um protocolo bastante simples onde as seguintes regras devem ser satisfeitas:

- Ao receber a conexão, o servidor obrigatoriamente deve anunciar na primeira linha o nome "SquidPCB" seguido de um espaço em branco e da versão do programa precedida pela letra "v". A seguir ele deverá aguardar a digitação de comandos, sendo estes um por linha;
- A autenticação do cliente deverá ser feita através de dois comandos: "USER" e "PASS". Ambos deverão ser sucedidos por um ou mais espaços e do nome de usuário e senha, respectivamente;
- O comando "INFO", com uma URL completa como argumento obrigatório, deverá ser implementado pelo servidor, e este deverá responder em uma única linha os dados sobre as probabilidades encontradas para a URL informada;
- O único dado obrigatório que deve ser respondido pelo comando INFO é a probabilidade do objeto, na forma "OBJ=*probabilidade*". Outros valores opcionais são: GRP (probabilidade média dos arquivos no mesmo nível), MAX (probabilidade máxima encontrada no caminho até o nó) e AVG (probabilidade média encontrada no caminho até o nó), que deverão aparecer sucedidos de um sinal de "=" e da probabilidade. Os valores de probabilidade deverão sempre aparecer na forma de um número real compreendido entre 0 e 1;
- O comando "LIST" poderá ser implementado pelo servidor, de forma que o servidor irá retornar em múltiplas linhas uma listagem dos nós

a partir de um determinado ponto da árvore. Este comando tem como propósito a possibilidade de realizar análises da árvore, verificando eventualmente o porque de algum nó estar sendo bloqueado incorretamente;

- Em caso de sucesso na execução destes comandos o servidor retornará o número "200" seguido de uma mensagem de texto opcional, ou "400" seguido de uma mensagem de erro opcional. Os valores descritos poderão ser incrementados, desde que não haja mudança na centena, visando indicar o retorno de forma mais precisa a fim de que o programa cliente possa informar o usuário com mais precisão sobre o motivo da falha;

- O comando "QUIT" deverá ser interpretado pelo servidor, que ao recebê-lo deverá responder com o código "200" seguido de uma mensagem de texto opcional e desconectar a conexão TCP;

- O servidor deverá aceitar os comandos "SAVE" para salvar uma cópia da estrutura de dados em disco, "RELOAD" para recarregar as configurações do sistema, e "SHUTDOWN" para derrubar todas as conexões, salvar todos os dados necessários e finalizar o processo principal.

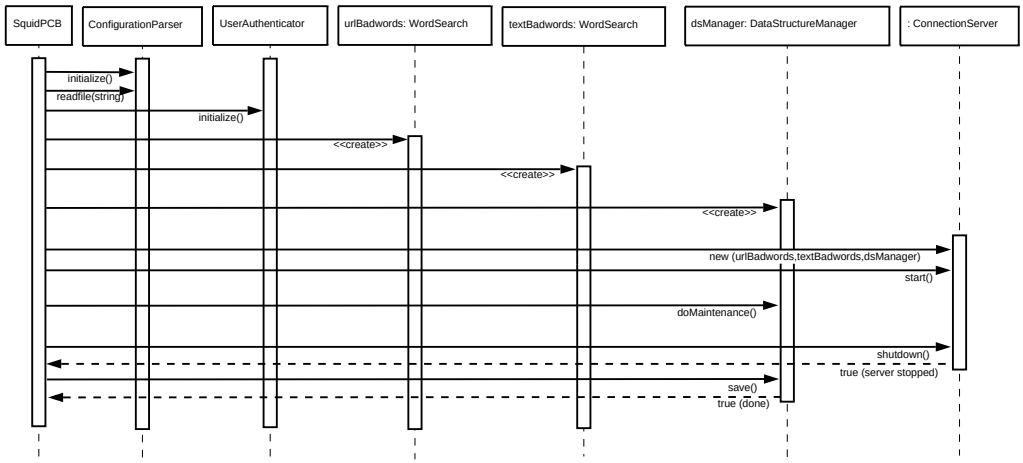
Com os dados obtidos do servidor, o SquidExternalACL verificará se o conteúdo deve ser considerado pornográfico ou não, retornando em sua saída padrão os caracteres "ERR" caso afirmativo ou "OK" caso contrário. É importante notar que o Squid-cache pode invocar um ou mais processos filhos SquidExternalACL para tratar de várias requisições ao mesmo tempo.

4.3 Diagramas de seqüência

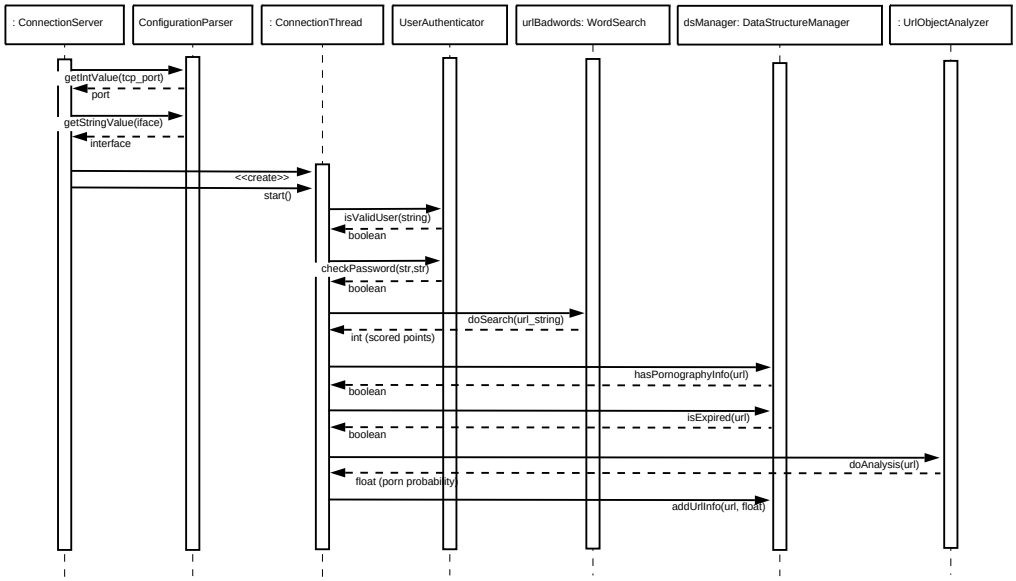
Abaixo é descrito o funcionamento de cada programa com maiores detalhes que permitirão compreender como o programa é organizado de tal forma que ele possa ser futuramente modificado sem grandes dificuldades.

4.3.1 SquidPCB

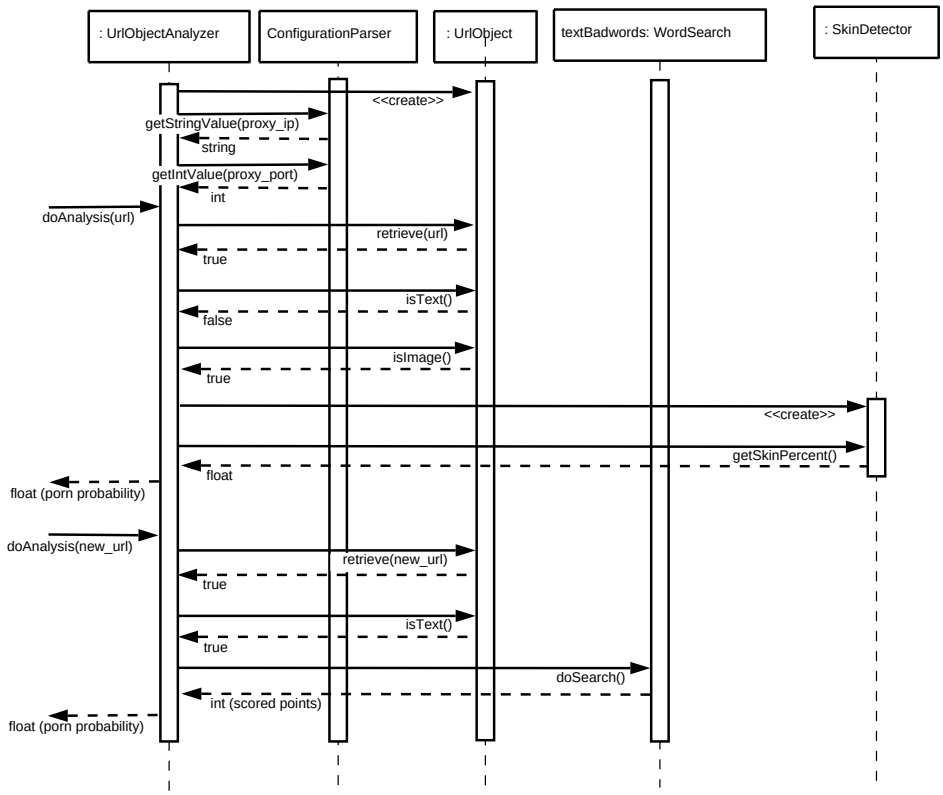
O programa SquidPCB basicamente cuida de toda a inicialização do servidor e da manutenção das *threads* principais, sendo estas as de recebimento de conexões e manutenção da estrutura de dados.



O funcionamento da classe ConnectionServer segue abaixo, com o caso onde a URL solicitada pelo cliente que se conecta não existe na estrutura de dados, requerendo que o processo de análise de arquivo seja executado.

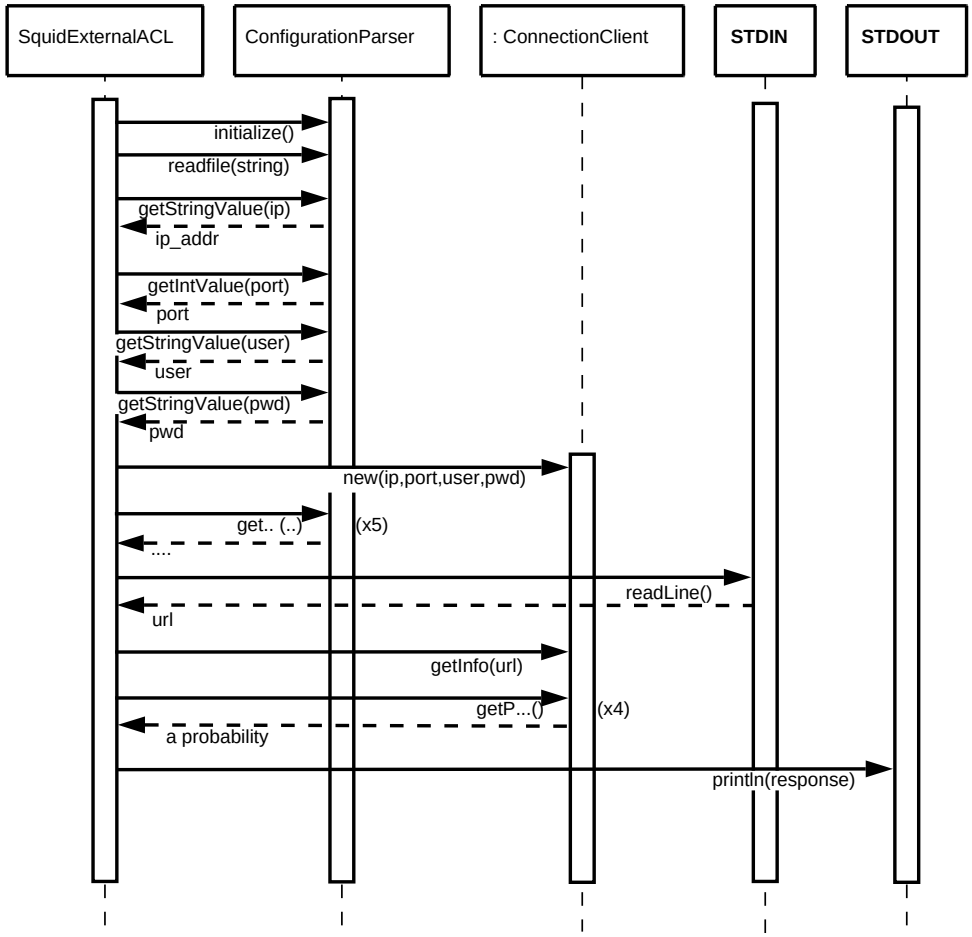


O mecanismo de análise dos arquivos referenciados pelas URLs está bem definido e isolado de outros componentes. Um objeto da class UrlObjectAnalyzer recebe a URL a ser analizada e, de acordo com o tipo de arquivo encontrado, ela chama um método de análise de uma classe diferente.



4.3.2 SquidExternalACL

O programa auxiliar SquidExternalACL, ao ser iniciado, se conecta com o servidor e realiza um processo de autenticação, o qual permite que o serviço seja oferecido para qualquer máquina em uma rede de forma minimamente segura. Após conectado, este programa passa a receber as URLs passadas pelo programa Squid-cache e a realizar consultas no servidor, classificando como impróprio ou não o conteúdo de cada arquivo referenciado.



4.4 A estrutura de dados

Inicialmente, a estrutura de dados foi planejada para ser formada por objetos que estendiam uma classe abstrata chamada UrlNode. Entretanto, após alguns problemas nos testes iniciais, principalmente com URLs incompletas, a idéia foi abandonada e a classe abstrata foi tornada uma classe comum, não havendo nenhuma distinção entre os nós da árvore. É possível que esta idéia ainda seja interessante dependendo dos recursos que poderão ser incorporados ao programa, principalmente visando um menor consumo de memória. Porém, para este trabalho, a estrutura de dados simplificada foi suficiente para o bom funcionamento do software em ambiente de testes.

O problema mais crítico encontrado durante os testes da estrutura de dados foi o alto consumo de memória, o qual foi creditado aos objetos HashMap e Vector utilizados em cada nó interno. Desta forma, com o servidor em

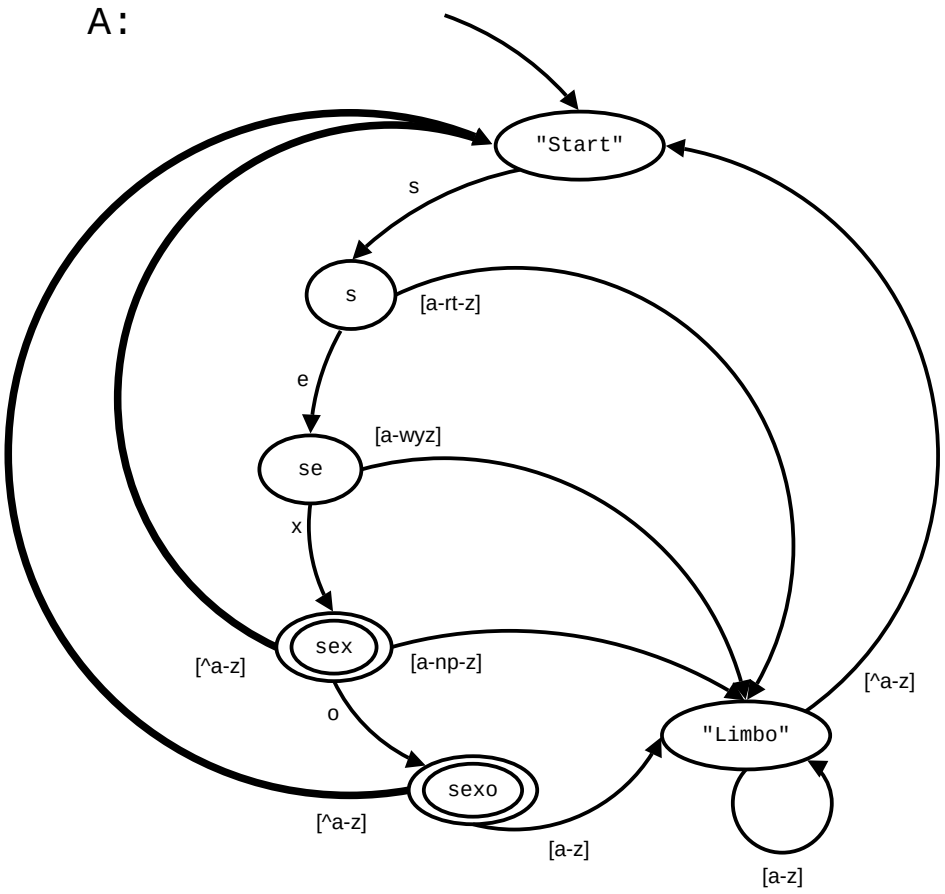
funcionamento por alguns dias, o uso da memória real da máquina poderia se chegar ao fim, obrigando o sistema operacional a usar sua memória virtual até esta também se esgotar.

Para tratar este problema sem ter de perder informações sobre os nós já analisados, optou-se por utilizar um método de *"swapping"* para disco, semelhante ao utilizado pelos sistemas operacionais com memória virtual, adotando a política conhecida como LRU (Least Recently Used) para enviar um certo ramo de uma árvore para o disco.

Apesar desta solução envolver acesso a disco, que é uma operação normalmente lenta, os testes mostraram uma boa eficiência deste método, o qual certamente oferecerá um desempenho satisfatório em um ambiente de produção.

4.5 Reconhecedor de palavras

Para realizar a análise de textos, um algoritmo baseado em um autômato determinístico foi utilizado para "ler" os textos procurando por palavras ou frases, em uma quantidade possivelmente grande. Este algoritmo foi escolhido justamente por possibilitar a leitura de arquivos de forma rápida e independente da quantidade de palavras cadastradas, atendendo perfeitamente os requisitos do projeto.



O autômato A considerado acima reconhece as palavras “sex” e “sexo”, sendo que nas transições para o estado inicial (“Start”) o algoritmo soma a um placar os pontos definidos para cada palavra reconhecida. Este exemplo considera somente o alfabeto das letras minúsculas de **a** a **z**, porém o algoritmo implementado reconhece qualquer caractere ASCII, tanto na definição de palavras quanto na entrada a ser avaliada.

É bastante claro que a velocidade deste método se deve ao fato de o algoritmo ser linear ao tamanho do texto de entrada. Em testes realizados com cerca de 1.000 palavras-chave em arquivos de até 20 megabytes, foram obtidos tempos de resposta sempre inferiores a 5 segundos.

4.6 Analisador de imagens

Durante a discussão sobre os métodos de visão computacional a serem empregados no analisador de imagens, discutiu-se algumas possibilidades, inclusive

a de implementar um algoritmo para detecção de imagens de corpos nús semelhante ao publicado no artigo "Finding naked people" [1].

O artigo descreve métodos utilizando filtros de cores para tons de pele e análise geométrica dos tons de pele filtrados, buscando por padrões de posicionamento pré-definidos. O gasto computacional descrito pareceu ser bastante pesado considerando o poder de processamento existente nas máquinas atuais e a quantidade de imagens que seriam submetidas ao servidor proxy em um curto espaço de tempo.

Os valores de precisão obtidos pelos testes dos pesquisadores apontaram somente 52,2% de acertos nas imagens contendo pessoas nuas pois, apesar de algumas imagens passarem pelo filtro de cor de pele, as mesmas não eram reconhecidas pelo filtro geométrico pelo fato do filtro não conseguir associar uma espinha dorsal e membros nas configurações de corpos pré-definidas.

Entretanto, o algoritmo descartou com uma precisão considerável (96,6%) as imagens nas quais não havia presença de corpos nus. Estes valores associados a idéia de agrupamento por similaridade seriam bastante razoáveis para construir um filtro com boa precisão.

Durante a implementação do sistema, devido a problemas com o prazo e por acreditar que a computação envolvida tornaria o sistema muito lento, o trabalho foi iniciado no analisador de imagens somente quantificando a proporção de pixels com cor de pele em uma imagem. Dado que 92,6% das imagens de teste do artigo foram descartadas pelo filtro de cor de pele, o qual descartava imagens com menos de 30% de pixels cor de pele, foi tomado como um teste aceitável a utilização dos demais arquivos considerados próximos na classificação de um arquivo, o que seria suficiente para evitar o bloqueio incorreto de algumas imagens.

4.7 Definição de pornografia

Como método de decisão sobre um objeto ser ou não pornografia, foi utilizado um conjunto de limiares para quatro valores retornados pela estrutura de dados. Estes valores correspondem a:

- **Probabilidade do objeto:** Corresponde a probabilidade do arquivo ser considerado pornografia;
- **Probabilidade do grupo:** Corresponde a probabilidade média dos arquivos encontrados no mesmo nível da estrutura de dados;
- **Probabilidade média:** Corresponde a probabilidade média dos objetos encontrados da raiz ao arquivo solicitado;

- **Probabilidade máxima:** Corresponde a probabilidade máxima dentre os objetos encontrados da raiz ao arquivo solicitado.

A decisão de bloqueio sobre um arquivo solicitado é tomada pelo processo SquidExternalACL, que bloqueia o arquivo solicitado caso algum dos limiares pré-estabelecidos seja ultrapassado. Seria bastante desejável que as decisões de bloqueio fossem mais flexiveis, e uma maneira de torná-las flexíveis é descrita na seção "Trabalhos futuros" deste projeto.

4.8 Trabalhos futuros

Existem diversos pontos onde este projeto pode ser melhorado - que vão desde melhorias nos mecanismos de classificação por visão computacional até a controles mais simples ou mais flexíveis do que deve ser bloqueado ou não. A organização deste trabalho foi conduzida de forma a tornar suas partes o mais independentes possível umas das outras, permitindo sua evolução de maneira mais simples.

Dentre os pontos levantados que podem ser melhorados na implementação do projeto é pode-se citar:

- **Inclusão de listas negras e brancas:** Este mecanismo extremamente simples de decisão de bloqueio estava na especificação inicial do projeto, porém com a modificação da estrutura de dados e as mudanças sobre o método de decisão de bloqueio a questão foi deixada de lado. A implementação, apesar de ser razoavelmente simples, deve ser bem projetada a fim de não conflitar com outras eventuais melhorias na estrutura de dados. Para esta melhoria, são necessária alterações somente nas classes UrlNode e DataStructureManager do pacote Datastructure;

- **Adição de filtro de "ruídos" nas imagens de tons de pele[1]:** As imagens com uma proporção de pixels cor de pele significativas poderiam passar por um filtro que descartaria blocos conexos de pixels cor de pele com tamanho muito pequeno perto da imagem toda. A implementação deste filtro deverá ser feita dentro da classe SkinDetector do pacote ImageProcessing e possíveis mudanças podem ocorrer na classe UrlObjectAnalyzer do pacote de mesmo nome;

- **Contagem de componentes conexas de imagens de tons de pele[1]:** Um outro teste bastante simples que pode ser implementado é a

[1]Imagens onde os pixels tem profundidade de cor de 1 bit e somente o pixels que representavam anteriormente pixels cor de pele possuem valor um.

contagem de componentes conexas de pixels cor de pele. Em geral, uma quantidade alta de componentes conexas pode indicar tanto a presença de roupas quando a não caracterização de uma cena de sexo, onde os corpos estão em contato, formando uma área contígua de pixels cor de pele. Da mesma maneira que o filtro anterior, esta implementação deve ser feita no pacote ImageProcessing enquanto que sua chamada deverá ser incluída na classe UrlObjectAnalyzer;

- **Flexibilização das decisões de bloqueio:** A decisão de bloqueio implementada é extremamente simples, e não permite a flexibilidade desejada na tomada de decisões por parte do administrador do filtro. Algo bastante semelhante ao mecanismo de ACLs do Squid-cache poderia ser implementado permitindo criar regras como: Arquivos texto com probabilidade superior a 60% serão bloqueados porém arquivos de imagem serão bloqueados somente com probabilidade superior a 75% e caso as demais imagens no mesmo nível da estrutura de dados possuam probabilidade média superior a 60%. A implementação desta flexibilização poderia ser feita em uma classe independente tendo chamadas realizadas a partir da classe SquidExternalACL, que somente passaria a esta classe a responsabilidade de analizar as regras e os valores obtidos para o objeto;

- **Programação de plugin para navegador:** Uma idéia mais audaciosa é a programação de um plugin para navegadores, de forma que um administrador de rede possa facilmente adicionar uma imagem ou página em uma lista de branca ou negra simplesmente clicando sobre o objeto e selecionando uma ação. Para isto a interface TCP/IP oferecida para consulta de URLs poderia ser bastante útil, uma vez que a mesma pode ser expandida com outros comandos a serem executados na estrutura de dados.

A fim de possibilitar a divulgação deste trabalho e permitir que outros programadores possam colaborar com o mesmo, um projeto no site SourceForge.net foi criado e pode ser acessado através da URL:

http://sourceforge.net/projects/squidpcb

O mantenedor pode ser contactado através do e-mail:

ferlemes(arroba)users.sourceforge.net

Capítulo 5

Conclusão

5.1 Resultados

Os resultados obtidos durante a execução deste projeto se mostraram bastante satisfatórios para a tarefa de filtragem de conteúdo, principalmente do tipo texto. Talvez outra abordagem pudesse ser feita, colocando o SquidPCB não como um programa auxiliar, mas sim um intermediário entre o Squidcache e a internet, permitindo não só o bloqueio mas também a alteração do conteúdo, o que permitiria censurar um texto ou imagem sem dificuldades.

A conclusão desta monografia é de que é perfeitamente possível utilizar um sistema de filtagem de conteúdo pornográfico em um servidor proxy de produção. Certamente com mais trabalhos de aprimoramento no filtro de imagens este programa poderá se tornar uma ótima solução de código aberto para o controle do conteúdo acessado, seja por crianças ou por outros usuários, os quais por algum motivo deverão sofrer este tipo de censura.

5.2 Relacionamento com o curso

O relacionamento desta monografia com o curso foi muito nítido em certos pontos, principalmente nas disciplinas:

1. **MAE0228 - Introdução a probabilidade e processos estocásticos:** Esta matéria foi bastante útil no que diz respeito a concepção da estrutura de dados utilizada para este projeto, dado que a mesma foi utilizada não só por proporcionar um tempo de busca baixo, mas também pela idéia do armazenamento ser inspirada em cadeias de Markov;

2. **MAC0323 - Estruturas de dados:** Esta matéria contribuiu da mesma maneira que MAE0228 na elaboração da estrutura de dados

utilizada neste projeto;

3. **MAC0332 - Engenharia de software:** Esta matéria foi bastante útil no que diz respeito a necessidade de tornar o código legível para possibilitar o trabalho em grupo. Também destaco o fato desta matéria ter sido responsável por apresentar a UML (Unified Modeling Language);

4. **MAC0328 - Algoritmos em grafos:** Esta matéria colaborou bastante com o aprendizado de MAC0414, o qual foi imprescindível para a análise de textos utilizada no projeto;

5. **MAC0414 - Linguagens formais e autômatos:** Com os conceitos aprendidos nesta disciplina, o algoritmo de análise de textos foi criado, proporcionando uma análise rápida e eficiente quanto a probabilidade do texto conter pornografia;

6. **MAC0448 - Programação para redes de computadores:** Esta matéria foi especialmente útil para compreender a programação com sockets em redes TCP/IP, necessária tanto para a interação do SquidPCB com o Squid-cache quanto para com o SquidExternalACL;

7. **MAC0422 - Sistemas operacionais:** Esta matéria abordou vários conceitos aplicados, como *"swapping"*, políticas de gerenciamento de memória, concorrência entre threads (mutex/semáforos), além de informações sobre o gerenciamento de processos nos sistemas operacionais;

8. **MAC0417 - Visão e processamento de imagens:** Esta matéria contribuiu muito com o projeto, principalmente na compreensão dos diversos espaços de cor existentes, além de métodos de segmentação e operadores morfológicos de imagem. Infelizmente nem tudo que foi aprendido nesta disciplina pode ser aproveitado devida a falta de tempo para implementação de algoritmos mais eficientes.

5.3 Softwares utilizados

Para o desenvolvimento deste aplicativo foi utilizado principalmente o ambiente de desenvolvimento de software Fedora Eclipse, baseado no Eclipse 3.2. A utilização deste software foi fundamental para o desenvolvimento deste projeto, uma vez que o mesmo me permitiu diminuir a quantidade de erros, principalmente os de digitação, cometidos durante a fase de desenvolvimento de código.

A utilização do software Eclipse poderia passar despercebida por este trabalho, porém dado que durante todo o meu curso de Bacharelado em Ciências da Computação eu sequer o tinha utilizado por acreditar que um editor mais simples como o *vim* era o suficiente para me auxiliar com o desenvolvimento de meus programas.

Outro software fundamental foi o ImageJ[2], que permitiu através do *plugin* 3D Color Inspector[3] descobrir qual o espaço de cor mais adequado para isolar a cor de de uma maneira bastante simples.

Vários outros softwares, em sua maioria softwares livres, foram utilizados durante a elaboração deste trabalho. São eles:

1. **Kile:** Editor de arquivos no formato LaTeX bastante simples e fácil de usar;

2. **Kivio:** Um editor de diagramas parte do pacote KOffice;

3. **Dia:** Outro editor de diagramas bem simples, mas que foi útil para o desenho dos diagramas de seqüencia;

4. **Poseidon for UML:** Um editor UML de código fechado, porém multiplataforma e bastante útil na elaboração dos diagramas UML;

5. **OpenOffice Writer:** Um popular editor de texto, o qual foi utilizado na preparação de um pôster para a apresentação do projeto.

Referências Bibliográficas

[1] Margaret Fleck, David Forsyth, and Chris Bregler (1996) "Finding Naked People," 1996 European Conference on Computer Vision , Volume II, pp. 592-602. (http://www.cs.hmc.edu/~fleck/naked.html)

[2] Rasband, W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA, http://rsb.info.nih.gov/ij/, 1997-2007.

[3] Kai Uwe Barthel (k.barthel at fhtw-berlin.de), 3D Color Inspector/Color Histogram, Internationale Medieninformatik, Berlin, Germany, http://rsb.info.nih.gov/ij/plugins/color-inspector.html, 2004-2007.

[4] Vladimir Vezhnevetz, Vassili Sazonov, Alla Andreeva, "A Survey on Pixel-Based Skin Color Detection Techniques". Proc. Graphicon-2003, pp. 85-92, Moscow, Russia, September 2003. (http://graphics.cmc.msu.ru)