Configurando um servidor proxy com o Squid

Configurando um servidor proxy com o Squid

O Squid permite compartilhar a conexão entre vários micros, servindo como um intermediário entre eles e a internet. Usar um proxy é diferente de simplesmente compartilhar a conexão diretamente, via NAT. Ao compartilhar via NAT, os micros da rede acessam a internet diretamente, sem restrições. O servidor apenas repassa as requisições recebidas, como um garoto de recados. O proxy é como um burocrata que não se limita a repassar as requisições: ele analisa todo o tráfego de dados, separando o que pode ou não pode passar e guardando informações para uso posterior.

Compartilhar a conexão via NAT é mais simples do que usar um proxy como o Squid sob vários aspectos. Você compartilha a conexão no servidor, configura os clientes para o utilizarem como gateway e pronto. Ao usar um proxy, além da configuração da rede, é necessário configurar o navegador e cada outro programa que for acessar a Internet (em cada um dos clientes da rede) para usar o proxy. Esta é uma tarefa tediosa e que acaba aumentando bastante seu volume de trabalho, pois toda vez que um micro novo for colocado na rede ou for preciso reinstalar o sistema, será preciso fazer a configuração novamente.

A configuração do proxy muda de navegador para navegador. No Firefox, por exemplo, você a encontra em "Editar > Preferências > Avançado > Rede > Configurações". No IE, a configuração está em "Opções da Internet > Opções > Configurações da Lan > Usar um servidor Proxy":

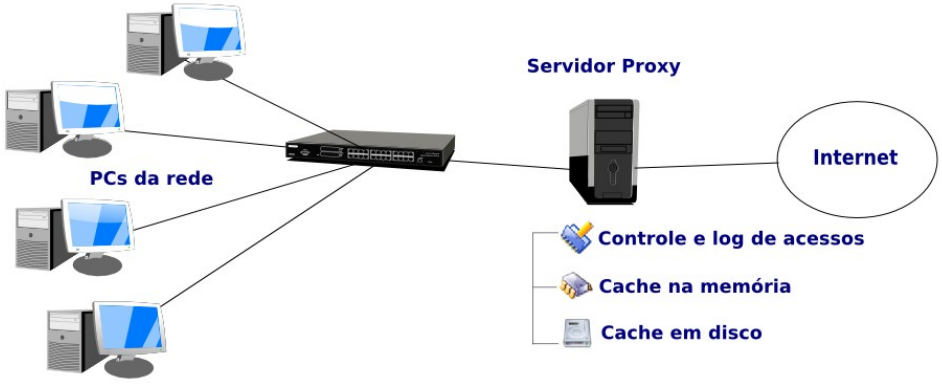
Além do navegador, outros programas podem ser configurados para trabalhar através do proxy: clientes de MSN, VoIP e até mesmo programas P2P. As vantagens de usar um proxy são basicamente três:

1- É possível impor restrições de acesso com base no horário, login, endereço IP da máquina e outras informações, além de bloquear páginas com conteúdo indesejado. É por isso que quase todos os softwares de filtro de conteúdo envolvem o uso de algum tipo de proxy, muitas vezes o próprio Squid (já que, como o software é aberto, você pode incluí-lo dentro de outros aplicativos, desde que respeitando os termos da GPL). Mais adiante estudaremos sobre a configuração do SquidGuard e do DansGuardian.

2- O proxy funciona como um cache de páginas e arquivos, armazenando informações já acessadas. Quando alguém acessa uma página que já foi carregada, o proxy envia os dados

que guardou no cache, sem precisar acessar a mesma página repetidamente. Isso acaba economizando bastante banda, tornando o acesso mais rápido.

Hoje em dia, os sites costumam usar páginas dinâmicas, onde o conteúdo muda a cada visita, mas, mesmo nesses casos, o proxy dá uma boa ajuda, pois embora o html seja diferente a cada visita e realmente precise ser baixado de novo, muitos componentes da página, como ilustrações, banners e animações em flash, podem ser aproveitados do cache, diminuindo o tempo total de carregamento.

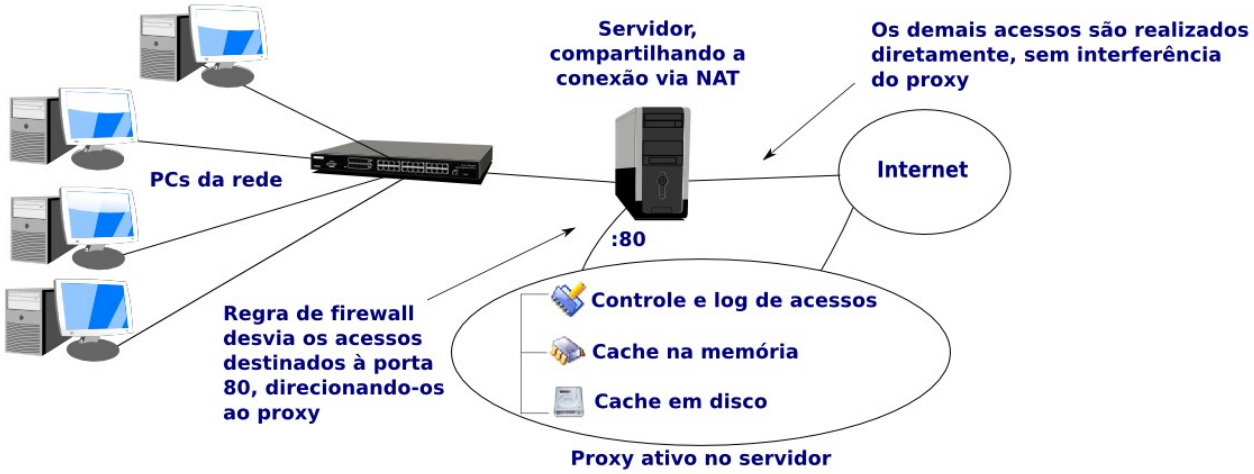


Dependendo da configuração, o proxy pode apenas acelerar o acesso às páginas ou servir como um verdadeiro cache de arquivos, armazenando atualizações do Windows Update, downloads diversos e pacotes instalados através do apt-get, por exemplo. Em vez de ter que baixar o último Service Pack do Windows ou a última atualização do Firefox nos 10 micros da rede, você vai precisar baixar apenas no primeiro, pois os outros 9 vão baixar a partir do cache do Squid.

3- Uma terceira vantagem de usar um proxy é que ele loga todos os acessos realizados através dele. Você pode visualizar os acessos posteriormente usando o Sarg, um gerador de relatórios que transforma as longas listas de acessos dos logs em arquivos html bem organizados.

Mesmo assim, você pode estar achando que as vantagens não vão compensar o trabalho de sair configurando micro por micro, programa por programa para usar o proxy, e que é mais fácil simplesmente compartilhar via NAT. Entretanto, existe a possibilidade de juntar as vantagens das duas formas de compartilhamento, configurando um **proxy transparente** como veremos adiante.

Ao usar um proxy transparente, você tem basicamente uma conexão compartilhada via NAT, com a mesma configuração básica nos clientes. O proxy entra na história como um adicional. Uma regra de firewall envia as requisições recebidas na porta 80 do servidor para o proxy, que se encarrega de responder aos clientes. Toda a navegação passa a ser feita automaticamente através do proxy (incluindo o cache dos arquivos do Windows update, downloads diversos e os pacotes instalados através do apt-get), sem que você precise fazer nenhuma configuração adicional nos clientes.



Instalando o Squid

O Squid é composto de um único pacote, por isso a instalação é simples. Instale o pacote "**squid**" usando o apt-get, yum ou urpmi, como em:

 # apt-get install squid

Toda a configuração do Squid é feita em um único arquivo, o "**/etc/squid/squid.conf**". Caso você esteja usando uma versão antiga do Squid, como a incluída no Debian Woody, por exemplo, o arquivo pode ser o "/etc/squid.conf". Apesar da mudança na localização do arquivo de configuração, as opções descritas aqui vão funcionar sem maiores problemas.

O arquivo original, instalado junto com o pacote, é realmente enorme, contém comentários e exemplos para quase todas as opções disponíveis. Ele pode ser uma leitura interessante se você já tem uma boa familiaridade com o Squid e quer aprender mais sobre cada opção, mas, de início, é melhor começar com um arquivo de configuração mais simples, apenas com as opções mais usadas.

Em geral, cada distribuição inclui uma ferramenta diferente para a configuração do proxy. Uma das mais usadas é o **Webmin**, disponível em várias distribuições. A função dessas ferramentas é disponibilizar as opções através de uma interface gráfica e gerar o arquivo de configuração com base nas opções escolhidas. Em alguns casos, essas ferramentas ajudam bastante, mas, como elas mudam de distribuição para distribuição, acaba sendo mais produtivo aprender a trabalhar direto no arquivo de configuração, que, afinal, não é tão complicado assim. Assim como em outros tópicos do livro, vamos aprender a configurar o Squid "no muque", sem depender de utilitários de configuração.

Comece renomeando o arquivo padrão, de forma a conservá-lo para fins de pesquisa:

mv /etc/squid/squid.conf /etc/squid/squid.conf.orig

Em seguida, crie um novo arquivo "**/etc/squid/squid.conf**", contendo apenas as quatro linhas abaixo:

```
http_port 3128
visible_hostname gdh
```

```
acl all src 0.0.0.0/0.0.0.0
http_access allow all
```

Estas linhas são o suficiente para que o Squid "funcione". Como viu, aquele arquivo de configuração gigante tem mais uma função informativa, citando e explicando as centenas de opções disponíveis. Apenas um punhado das opções são realmente necessárias, pois, ao omití-las, o Squid simplesmente utiliza os valores default. É por isso que acaba sendo mais simples começar com um arquivo vazio e ir inserindo apenas as opções que você conhece e deseja alterar.

As quatro linhas dizem o seguinte:

http_port 3128: A porta onde o servidor Squid vai ficar disponível. A porta 3128 é o default, mas muitos administradores preferem utilizar a porta 8080, que soa mais familiar a muitos usuários.

visible_hostname gdh: O nome do servidor, o mesmo que foi definido na configuração da rede. Ao usar os modelos desse capítulo, não se esqueça de substituir o "gdh" pelo nome correto do seu servidor, como informado pelo comando "hostname".

acl all src 0.0.0.0/0.0.0.0 e **http_access allow all**: Estas duas linhas criam uma acl (uma política de acesso) chamada "all" (todos), incluindo todos os endereços IP possíveis. Ela permite que qualquer um dentro desta lista use o proxy, ou seja, permite que qualquer um use o proxy, sem limitações.

Para testar a configuração, reinicie o servidor Squid com o comando:

/etc/init.d/squid restart

Se estiver no CentOS, Fedora ou Mandriva, pode utilizar o comando "service", que economiza alguns toques no teclado:

 # service squid restart

No Slackware, o comando será "/etc/rc.d/rc.squid restart", seguindo a lógica do sistema em colocar os scripts referentes aos serviços na pasta /etc/rc.d/ e inicializá-los automaticamente durante o boot, desde que marcada a permissão de execução.

Para testar o proxy, configure um navegador (no próprio servidor) para usar o proxy, através do endereço 127.0.0.1 (o localhost), porta 3128. Se não houver nenhum firewall pelo caminho, você conseguirá acessar o proxy também através dos outros micros da rede local, basta configurar os navegadores para usarem o proxy, fornecendo o endereço do servidor na rede local.

Caso necessário, abra a porta 3128 na configuração do firewall, para que o Squid possa receber as conexões. Um exemplo de regra manual do Iptables para abrir a porta do Squid apenas para a rede local (a interface eth0 no exemplo) é:

```
iptables -A INPUT -i eth0 -p tcp --dport 3128 -j ACCEPT
```

Criando uma configuração básica

O problema com o modelo de configuração minimalista que acabamos de ver é que com apenas estas quatro linhas o proxy ficará muito aberto. Se você deixar o servidor proxy ativo no próprio servidor que compartilha a conexão e não houver nenhum firewall ativo, qualquer um na internet

poderia usar o seu proxy, o que naturalmente não é desejado. O proxy deve ficar ativo apenas para a rede local.

Vamos gerar, então, um arquivo mais completo, permitindo que apenas os micros da rede local possam usar o proxy e definindo mais algumas políticas de segurança. Neste segundo exemplo já aproveitei algumas linhas do arquivo original, criando regras que permitem o acesso a apenas algumas portas específicas e não mais a qualquer coisa, como na configuração anterior:

```
http_port 3128
visible_hostname gdh

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 563 # https, snews
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl Safe_ports port 901 # swat
acl Safe_ports port 1025-65535 # portas altas
acl purge method PURGE
acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal

http_access deny all
```

As acl's "SSL_ports" e a "Safe_ports" são as responsáveis por limitar as portas que podem ser usadas através do proxy. Neste exemplo, usei a configuração-modelo indicada na documentação do Squid, que prevê o uso de diversos protocolos conhecidos e também o uso de portas altas, acima da 1024. Ela é tão extensa porque cada porta é especificada em uma linha diferente. Podemos simplificar isso agrupando as portas na mesma linha, o que resulta em um arquivo de configuração muito menor, mas que faz exatamente a mesma coisa:

```
http_port 3128
visible_hostname gdh
```

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 21 80 443 563 70 210 280 488 59 777 901 1025-65535
acl purge method PURGE
acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal

http_access deny all

Veja que em ambos os exemplos adicionei duas novas acl's. A acl "localhost" contém o endereço 127.0.0.1, que você utiliza ao usar o proxy localmente (ao navegar usando o próprio servidor), e a acl "rede local", que inclui os demais micros da rede local. Substitua o "**192.168.1.0/24**" pela faixa de endereços IP e a máscara de sub-rede usada na sua rede local (o 24 equivale à mascara 255.255.255.0).

Depois de criadas as duas políticas de acesso, vão duas linhas no final do arquivo que especificam que os micros que se enquadrarem nelas poderão usar o proxy:

http_access allow localhost
http_access allow redelocal

Lembra-se da acl "all", que contém todo mundo? Vamos usá-la para especificar que os clientes que não se enquadrarem nas duas regras acima (ou seja, clientes não-autorizados, vindos da Internet) não poderão usar o proxy:

http_access deny all

Esta linha deve ir no final do arquivo, depois das outras duas. A ordem é importante, pois o Squid interpreta as regras na ordem em que são colocadas no arquivo. Se você permite que o micro X acesse o proxy, ele acessa, mesmo que uma regra mais abaixo diga que não.

Se você adicionasse algo como:

acl redelocal src 192.168.1.0/24
http_access allow redelocal
http_access deny redelocal

... os micros da rede local continuariam acessando, pois a regra que permite vem antes da que proíbe.

Existem alguns casos de sites que não funcionam bem quando acessados através de proxies, um exemplo comum é o "Conectividade Social", da Caixa. Normalmente nesses casos o problema está

em algum recurso fora do padrão usado pelo sistema do site e não no servidor proxy propriamente dito, mas, de qualquer forma, você pode solucionar o problema de forma muito simples orientando o servidor proxy a repassar as requisições destinadas ao site diretamente.

Para isso, adicionamos uma ACL na configuração, especificando a URL do site e usando a opção "always_direct":

```
acl site dstdomain siteproblematico.com
always_direct allow site
```

Esta regra deve vir antes da regra que libera os acessos provenientes da rede local, como em:

```
acl site dstdomain siteproblematico.com
always_direct allow site

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Depois de configurar o arquivo, não se esqueça de reiniciar o serviço para que a configuração entre em vigor:

 # /etc/init.d/squid restart

Nesse ponto o seu proxy já está completamente funcional. Você pode começar a configurar os navegadores nos PCs da rede local para utilizá-lo e acompanhar o desempenho da rede. Agora que já vimos o arroz com feijão, vamos aos detalhes mais avançados da configuração:

Configurando o cache de páginas e arquivos

Uma das configurações mais importantes com relação ao desempenho do proxy e à otimização do tráfego da rede é a configuração dos caches, onde o Squid guarda as páginas e arquivos já acessados de forma a fornecê-los rapidamente quando solicitados novamente. O Squid trabalha com dois tipos de cache:

1- Cache rápido, feito usando parte da memória RAM do servidor;
2- Cache um pouco mais lento porém maior, feito no HD.

O cache na memória RAM é ideal para armazenar arquivos pequenos, como páginas html e imagens, que serão entregues instantaneamente para os clientes. O cache no HD é usado para armazenar arquivos maiores, como downloads, arquivos do Windows Update e pacotes baixados via apt-get.

O cache na memória RAM é sempre relativamente pequeno, já que o volume de memória RAM no servidor é sempre muito menor do que o espaço em disco. O cache no HD pode ser mais generoso, afinal a idéia é que ele guarde todo tipo de arquivos, principalmente os downloads grandes, que demoram para serem baixados.

A configuração do cache é feita adicionando mais algumas linhas no arquivo de configuração:

1- A configuração da quantidade de memória RAM dedicada ao cache é feita adicionando a opção "cache_mem", que contém a quantidade de memória que será dedicada ao cache. Para reservar 64 MB, por exemplo, a linha ficaria:

cache_mem 64 MB

Como regra geral, você pode reservar 32 ou 64 MB para o cache em um servidor não dedicado, que atende a apenas alguns micros (como o servidor de uma pequena rede local) e até 1/3 da memória RAM total em um servidor proxy dedicado, que atende a um volume maior de usuários (veja mais detalhes a seguir). Em um servidor com 1 GB de RAM, por exemplo, você poderia reservar até 350 MB para o cache na memória.

2- Logo depois vai a opção "maximum_object_size_in_memory", que determina o tamanho máximo dos arquivos que serão guardados no cache feito na memória RAM (o resto vai para o cache feito no HD). O cache na memória é muito mais rápido, mas como a quantidade de RAM é muito limitada, é melhor deixá-la disponível para páginas web, figuras e arquivos pequenos em geral. Para que o cache na memória armazene arquivos de até 64 KB, por exemplo, adicione a linha:

maximum_object_size_in_memory 64 KB

3- Em seguida vem a configuração do cache em disco, que armazenará o grosso dos arquivos. Por default, o máximo são downloads de até 16 MB e o mínimo é zero, o que faz com que mesmo imagens e arquivos pequenos sejam armazenados no cache. Quase sempre é mais rápido ler a partir do cache do que baixar de novo da web, mesmo que o arquivo seja pequeno.

Se você faz download de arquivos grandes com freqüência e deseja que eles fiquem armazenados no cache, aumente o valor da opção "maximum_object_size". Isso é especialmente útil para quem precisa baixar muitos arquivos através do apt-get ou Windows Update em muitos micros da rede. Se você quiser que o cache armazene arquivos de até 512 MB, por exemplo, as linhas ficariam:

maximum_object_size 512 MB
minimum_object_size 0 KB

Você pode definir ainda a percentagem de uso do cache que fará o Squid começar a descartar os arquivos mais antigos. Por padrão, sempre que o cache atingir 95% de uso, serão descartados arquivos antigos até que a percentagem volte para um número abaixo de 90%:

cache_swap_low 90
cache_swap_high 95

4- Depois vem a opção "cache_dir", que é composta por quatro valores. O primeiro, (**/var/spool/squid**) indica a pasta onde o Squid armazena os arquivos do cache, enquanto o segundo (**2048**) indica a quantidade de espaço no HD (em MB) que será usada para o cache. Aumente o valor se você tem muito espaço no HD do servidor e quer que o Squid guarde os downloads por muito tempo.

Continuando, os números **16** e **256** indicam a quantidade de subpastas que serão criadas dentro do diretório. Por padrão, temos 16 pastas com 256 subpastas cada uma. O número "ideal" de pastas e subpastas para um melhor desempenho varia de acordo com o sistema de arquivos usado, mas esta configuração padrão é adequada para a maioria das situações. Combinando as quatro opções, a linha ficaria:

cache_dir ufs /var/spool/squid 2048 16 256

Assim como na maioria das opções do Squid, se a linha "cache_dir" for omitida, é usada a configuração default para a opção, que é usar o diretório "/var/spool/squid" e fazer um cache em disco de 100 MB.

5- Você pode definir ainda o arquivo onde são guardados os logs de acesso do Squid. Por padrão, o Squid guarda o log de acesso no arquivo "/var/log/squid/access.log". Este arquivo é usado pelo Sarg para gerar as páginas com as estatísticas de acesso:

cache_access_log /var/log/squid/access.log

Mais uma configuração que você pode querer alterar é o padrão de atualização do cache. Estas três linhas precisam sempre ser usadas em conjunto, ou seja, você pode alterá-las, mas sempre as três precisam estar presentes no arquivo. Eliminando uma, o Squid ignora as outras duas e usa o default.

Os números indicam o intervalo (em minutos) que o Squid irá aguardar antes de verificar se um item do cache (uma página, por exemplo) foi atualizado, para cada um dos três protocolos. O primeiro número (o 15) indica que o Squid verificará (a cada acesso) se as páginas e arquivos com mais de 15 minutos foram atualizados. Ele faz uma verificação rápida, checando o tamanho do arquivo e, se o arquivo não mudou, ele continua fornecendo aos clientes o arquivo que está no cache, economizando banda da conexão

O terceiro número (o 2280, equivalente a dois dias) indica o tempo máximo, depois do qual o objeto é sempre verificado. Além do http e ftp, o Squid suporta o protocolo gopher, que era muito usado nos primórdios da internet para localizar documentos de texto, mas perdeu a relevância hoje em dia:

refresh_pattern ^ftp: 15 20% 2280
refresh_pattern ^gopher: 15 0% 2280
refresh_pattern . 15 20% 2280

Depois de adicionar todas estas configurações, o nosso arquivo de configuração já ficará bem maior:

```
http_port 3128
visible_hostname gdh

cache_mem 64 MB
maximum_object_size_in_memory 64 KB
maximum_object_size 512 MB
minimum_object_size 0 KB
cache_swap_low 90
cache_swap_high 95
cache_dir ufs /var/spool/squid 2048 16 256
cache_access_log /var/log/squid/access.log
refresh_pattern ^ftp: 15 20% 2280
refresh_pattern ^gopher: 15 0% 2280
refresh_pattern . 15 20% 2280

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 21 80 443 563 70 210 280 488 59 777 901 1025-65535
acl purge method PURGE
acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Aqui já temos uma configuração mais completa, incluindo um conjunto de regras de segurança (para que o proxy seja usado apenas a partir da rede local) e a configuração do cache. Esta é uma configuração adequada para uso em uma rede doméstica ou em um pequeno escritório.

A acl "Safe_ports", alimentada com um conjunto de portas (80, 21 e outras) é usada para restringir as portas de saída do servidor proxy, evitando, por exemplo, que ele seja usado para enviar e-mails (porta 25). Isso evita um conjunto de abusos comuns e é uma configuração importante em qualquer servidor que precise ser configurado de forma minimamente segura. Naturalmente, você pode adicionar outras portas à lista, conforme necessário.

Em uma rede maior, você provavelmente iria querer adicionar algumas limitações de acesso, limitando o acesso a algumas páginas, criando um sistema de autenticação ou limitando o uso com base no horário, entre outras possibilidades. Vamos a elas.

Adicionando restrições de acesso

Em um ambiente de trabalho, a idéia é que os funcionários usem a internet para comunicação, pesquisa e outras funções relacionadas ao que estão fazendo. Muitas empresas permitem que sejam acessados os e-mails pessoais e coisas do gênero, mas sempre até um certo limite. Seu chefe não vai gostar se começarem a passar a maior parte do tempo no Orkut, por exemplo. Chegamos então às regras para restrição de acesso, que são uma necessidade em muitas redes.

Bloqueando por domínios ou palavras

O Squid permite bloquear sites indesejados de forma bastante simples usando o parâmetro "dstdomain". Ele permite que você crie acl's contendo endereços de sites que devem ser bloqueados (ou permitidos). Isso é feito em duas etapas. Primeiro você cria a acl, especificando os endereços e, em seguida, usa o parâmetro "http_access" para bloquear ou liberar o acesso a eles. Veja um exemplo:

```
acl bloqueados dstdomain orkut.com playboy.abril.com.br
http_access deny bloqueados
```

Aqui eu criei uma acl chamada "**bloqueados**", que contém os endereços "orkut.com" e "playboy.abril.com.br" e em seguida usei o parâmetro "http_access deny" para bloquear o acesso a eles. Você pode incluir diversas acls diferentes dentro da configuração do Squid, desde que use um nome diferente para cada uma. De certa forma, elas são similares às variáveis, que usamos ao programar em qualquer linguagem.

Ao aplicar a regra, o Squid faz a resolução do domínio e passa a bloquear todas sub-páginas que estiverem hospedadas dentro dele. Existe uma ressalva: muitos sites podem ser acessados tanto com o "www" quanto sem. Se os dois estiverem hospedados em servidores diferentes, o Squid considerará que tratam-se de dois sites diferentes, de forma que ao bloquear apenas o "www.orkut.com" os usuários ainda conseguirão acessar o site através do "orkut.com" e vice-versa. Nesses casos, para bloquear ambos, é preciso incluir as duas possibilidades dentro da regra, como em:

```
acl bloqueados dstdomain orkut.com www.orkut.com playboy.abril.com.br
http_access deny bloqueados
```

Você pode incluir quantos domínios quiser dentro da acl, basta separá-los por espaço e deixar tudo na mesma linha. Se a regra começar a ficar muito grande, você tem a opção de transferir as entradas para um arquivo. Nesse caso, crie um arquivo de texto simples, com todos os domínios desejados (um por linha), como em:

```
orkut.com
www.orkut.com
playboy.abril.com.br
www.myspace.com
```

... e use a regra abaixo na configuração do Squid para que ele seja processado e os domínios sejam incluídos na acl. No exemplo, estou usando o arquivo "/etc/squid/bloqueados":

acl bloqueados url_regex -i "/etc/squid/bloqueados"
http_access deny bloqueados

Naturalmente, não seria viável tentar bloquear manualmente todos os sites pornográficos, chats, comunidades online, e todos os outros tipos de sites que não são úteis em um ambiente de trabalho. A idéia seria logar os acessos (com a ajuda do Sarg, que veremos mais adiante) e bloquear os sites mais acessados, conforme tomar conhecimento deles. É sempre uma corrida de gato e rato, mas, em se tratando de pessoas adultas, não há nada que uma boa conversa com o chefe não possa resolver. ;)

De qualquer forma, em alguns ambientes pode ser mais fácil bloquear inicialmente o acesso a todos os sites e ir abrindo o acesso a apenas alguns sites específicos, conforme a necessidade. Neste caso, invertemos a lógica da regra. Criamos um arquivo com sites permitidos, adicionamos a regra que permite o acesso a eles e em seguida bloqueamos o acesso a todos os demais, como neste exemplo:

acl permitidos url_regex -i "/etc/squid/permitidos"
http_access allow permitidos
http_access deny all

Nas versões recentes do Squid, ao bloquear um domínio é automaticamente bloqueado também o endereço IP do servidor correspondente. Isso evita que os usuários da rede consigam burlar o proxy, acessando os sites diretamente pelo IP. De qualquer forma, você pode criar diretamente regras que bloqueiem determinados endereços IP, o que é útil em casos de servidores sem domínio registrado, ou que respondam por vários domínios. Nesse caso, a regra ficaria:

acl ips-bloqueados dst 200.234.21.23 200.212.15.45
http_access deny ips-bloqueados

Você pode descobrir rapidamente o endereço IP de um determinado domínio usando o comando "**host**", como em:

$ host google.com

google.com A 72.14.207.99
google.com A 64.233.187.99
google.com A 64.233.167.99

Depois de adicionar as novas regras, nosso arquivo de configuração ficaria assim:

http_port 3128
visible_hostname gdh
error_directory /usr/share/squid/errors/Portuguese/

cache_mem 64 MB
maximum_object_size_in_memory 64 KB
maximum_object_size 512 MB
minimum_object_size 0 KB
cache_swap_low 90
cache_swap_high 95
cache_dir ufs /var/spool/squid 2048 16 256
cache_access_log /var/log/squid/access.log

```
refresh_pattern ^ftp: 15 20% 2280
refresh_pattern ^gopher: 15 0% 2280
refresh_pattern . 15 20% 2280

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 21 80 443 563 70 210 280 488 59 777 901 1025-65535
acl purge method PURGE
acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

acl bloqueados url_regex -i "/etc/squid/bloqueados"
http_access deny bloqueados

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Veja que coloquei as duas regras antes do "http_access allow redelocal", que abre tudo para a rede local. Como o Squid processa as regras seqüencialmente, as páginas que forem bloqueadas pela acl "bloqueados" não chegam a passar pela regra que autoriza os acessos provenientes da rede local.

Uma segunda possibilidade é usar o parâmetro "dstdom_regex", que permite bloquear sites de uma forma mais geral, com base em palavras incluídas na URL de acesso. Você pode bloquear todas as páginas cujo endereço inclua a palavra "sexo" ou "orkut", por exemplo. Note que, ao usar esta regra, o Squid verifica a existência das palavras apenas na URL do site e não no conteúdo da página. Para criar filtros baseados no conteúdo, você pode utilizar o DansGuardian, que veremos mais adiante.

Crie mais um arquivo de texto, contendo as palavras que devem ser bloqueadas, uma por linha, como em:

```
orkut
xxx
sexo
teens
warez
```

... e adicione a regra abaixo, contendo a localização do arquivo:

```
acl palavrasproibidas dstdom_regex "/etc/squid/palavrasproibidas"
http_access deny palavrasproibidas
```

O uso desta regra é um pouco mais problemático, pois bloqueará todas páginas que contenham qualquer uma das palavras listadas na URL. Esta opção sempre levará a alguns falsos positivos e por isso deve ser usada com mais cuidado.

Uma vantagem é que ela permite bloquear facilmente páginas dinâmicas, onde a palavra é passada como parâmetro da URL. Um exemplo é o Orkut, onde, depois da transferência para o Google, os domínios principais passaram a encaminhar para URLs dinâmicas dentro do domínio do Google, como em:

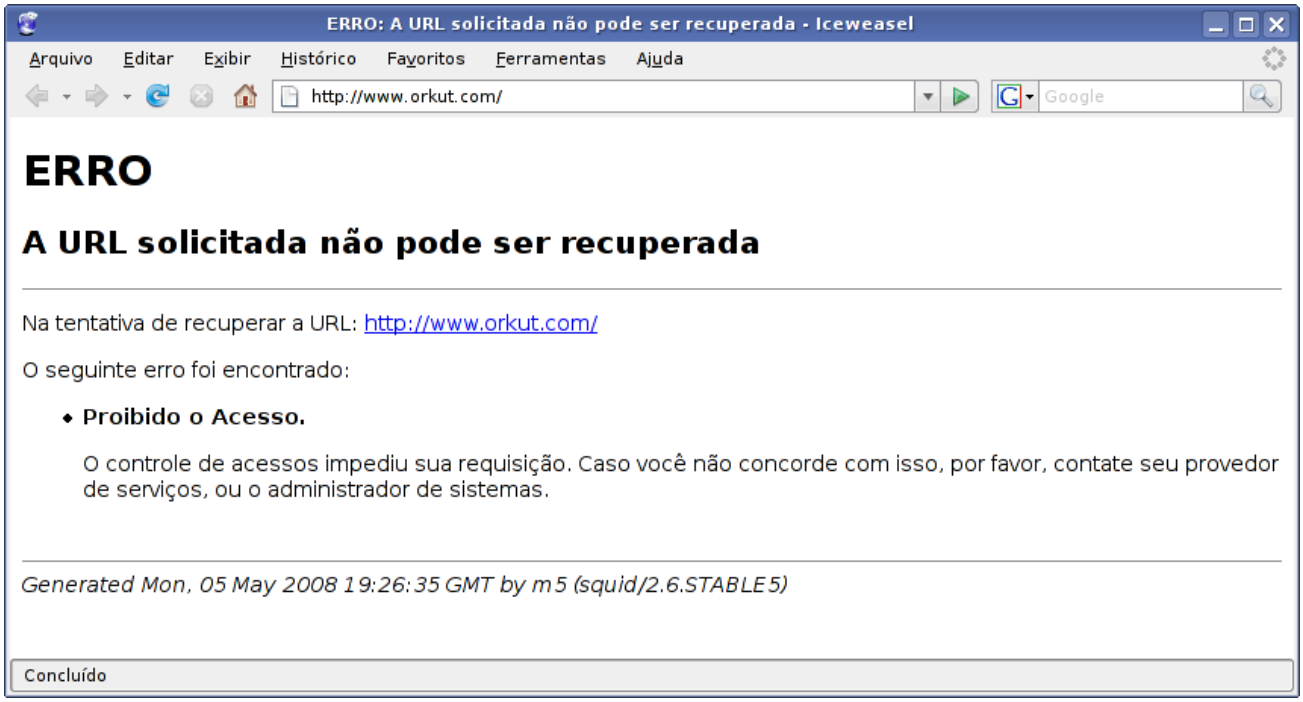
```
https://www.google.com/accounts/ServiceLogin?service=orkut&continue=http%3A%2F
%2Fwww.orkut.com
%2FRedirLogin.aspx%3Fmsg%3D0%26page%3Dhttp%253A%252F%252F
www.orkut.com%252FHome.aspx&hl=pt-BR&rm=false&passive=true
```

Você não poderia simplesmente bloquear o domínio "google.com" usando uma regra url_regex, mas poderia muito bem usar o dstdom_regex para bloquear a palavra "orkut" e assim bloquear o acesso ao site sem bloquear o acesso a outros serviços do Google.

Não existe problema em combinar o bloqueio de domínios e de palavras dentro da URL, você pode lançar mão de uma combinação das duas coisas, de acordo com a situação. Para isso, basta usar as duas regras simultaneamente, como em:

```
acl bloqueados url_regex -i "/etc/squid/bloqueados"
http_access deny bloqueados

acl palavrasproibidas dstdom_regex "/etc/squid/palavrasproibidas"
http_access deny palavrasproibidas

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Incluídas as regras, os clientes passam a ver uma mensagem de erro ao tentar acessar páginas que se enquadrem nos bloqueios:



Por padrão, as mensagens de erro aparecerem em inglês. No nosso caso elas estão aparecendo em português devido à linha "error_directory /usr/share/squid/errors/Portuguese/" que incluí no modelo de configuração anterior.

Você pode personalizar as páginas de erro editando os arquivos dentro da pasta "/usr/share/squid/errors/Portuguese" ou "/usr/share/squid/errors/English" (de acordo com a língua definida na configuração). A pasta contêm várias páginas html, uma para cada tipo de erro indicado.

Bloqueando por horário

As regras a seguir fazem com que o proxy aceite ou recuse conexões feitas dentro de determinados horários. Você pode definir regras para períodos específicos e combiná-las para bloquear todos os horários em que você não quer que o proxy seja usado ou vice-versa. Para que o proxy bloqueie acessos feitos entre meia-noite e 6:00 da manhã e no horário de almoço, por exemplo, você usaria as regras:

acl madrugada time 00:00-06:00
http_access deny madrugada

acl almoco time 12:00-14:00
http_access deny almoco

Estas regras iriam, novamente, antes da regra "http_access allow redelocal" no arquivo de configuração.

Agora, imagine que você quer fazer diferente. Ao invés de bloquear o acesso na hora de almoço, você quer deixar o proxy aberto, para que aqueles que queiram acessar o Orkut ou acessar os e-mails possam fazer isso fora do horário de trabalho. Neste caso, você usaria uma regra como:

acl almoco time 12:00-14:00
http_access allow almoco

Esta regra entraria no arquivo de configuração antes das regras "http_access deny bloqueados" e outras restrições. Assim, os acessos que forem aceitos pela regra do almoço não passarão pelas regras que fazem o bloqueio, como em:

```
acl almoco time 12:00-14:00
http_access allow almoco

acl bloqueados url_regex -i "/etc/squid/bloqueados"
http_access deny bloqueados
acl extban url_regex -i "/etc/squid/extban"
http_access deny extban

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Você pode também combinar o bloqueio de palavras ou domínio com as regras de bloqueio por horário, permitindo que os usuários acessem um determinado site apenas no horário de almoço, por exemplo. Neste caso, a regra seria:

acl almoco time 12:00-14:00
acl orkut dstdomain orkut.com www.orkut.com
http_access allow orkut almoco
http_access deny orkut

Fazendo isso, o Squid entende que os endereços incluídos dentro da acl "orkut" devem ser permitidos, mas apenas dentro do horário especificado na acl "almoco". Em seguida é incluída mais uma regra, que bloqueia o acesso ao site em outros horários.

Gerenciando o uso da banda

O Squid oferece uma forma simples de limitar o uso da banda disponível e definir o quanto cada usuário pode usar (mantendo parte do link livre para os demais), utilizando um recurso chamado "delay pools". Imagine, por exemplo, que você tem um link de 1 megabit para uma rede com 20 usuários. Se cada um puder ficar baixando o que quiser, é provável que a rede fique saturada em determinados horários, deixando a navegação lenta para todo mundo.

Você pode evitar isso limitando a banda que cada usuário pode usar e a banda total, que todos os usuários somados poderão usar simultaneamente. É recomendável, neste caso, que o servidor proxy (que combina todos os acessos via http) consuma um pouco menos que o total de banda disponível, de forma a sempre deixar um pouco reservado para outros protocolos.

Um link de 1 megabit (1024 kbits) corresponde a 131.072 bytes por segundo. Nas regras do Squid, sempre usamos bytes, por isso lembre-se de fazer a conversão, dividindo o valor em kbits por 8 e multiplicando por 1024 para ter o valor em bytes.

Podemos, por exemplo, limitar a banda total usada pelo Squid a 114.688 bytes por segundo, deixando 128 kbits do link livres para outros protocolos e limitar cada usuário a no máximo 16.384 bytes por segundo, que correspondem a 128 kbits. Nem todos os usuários vão ficar baixando arquivos a todo momento, por isso o valor ideal reservado a cada usuário vai variar muito de acordo com a rede. Você pode acompanhar o uso do link e ir ajustando o valor conforme a utilização.

Neste caso, a parte final do arquivo de configuração ficaria:

```
acl redelocal src 192.168.1.0/24
delay_pools 1
delay_class 1 2
delay_parameters 1 114688/114688 16384/16348
delay_access 1 allow redelocal
http_access allow localhost
http_access allow redelocal
http_access deny all
```

A acl "redelocal" está agora condicionada a três novas regras, que aplicam o uso do limite de banda. O acesso continua sendo permitido, mas agora dentro das condições especificadas na linha "delay_parameters 1 114688/114688 16384/16384", onde vão (respectivamente) os valores com a banda total disponível para o Squid e a banda disponível para cada usuário.

Veja que nessa regra limitamos a banda apenas para a acl "redelocal" e não para o "localhost". Isso significa que você continua conseguindo fazer downloads na velocidade máxima permitida pelo link ao acessar a partir do próprio servidor; a regra se aplica apenas às estações.

É possível também criar regras de exceção para endereços IP específicos, que poderão fazer downloads sem passar pelo filtro. Nesse caso, criamos uma acl contendo o endereço IP da estação que deve ter o acesso liberado usando o parâmetro "src" e a colocamos antes da regra que limita a velocidade, como em:

```
acl chefe src 192.168.1.2
http_access allow chefe
```

```
acl redelocal src 192.168.1.0/24
delay_pools 1
delay_class 1 2
delay_parameters 1 114688/114688 16384/16348
delay_access 1 allow redelocal
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Esta regra de exceção pode der usada também em conjunto com as demais regras de restrição de acesso que vimos anteriormente. Basta que a acl com o IP liberado seja colocada antes das acls com as restrições de acesso, como em:

```
acl chefe src 192.168.1.2
http_access allow chefe

acl bloqueados url_regex -i "/etc/squid/bloqueados"
http_access deny bloqueados
acl palavrasproibidas dstdom_regex "/etc/squid/palavrasproibidas"
http_access deny palavrasproibidas

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Concluindo, mais um tipo de bloqueio útil em muitas situações é com relação a formatos de arquivos. Você pode querer bloquear o download de arquivos .exe ou .sh para dificultar a instalação de programas nas estações, ou bloquear arquivos .avi ou .wmv para economizar banda da rede, por exemplo.

Neste caso, você pode usar a regra a seguir, especificando as extensões de arquivo desejadas. Ela utiliza o parâmetro "url_regex" (o mesmo que utilizamos para criar o bloqueio de domínios), dessa vez procurando pelas extensões de arquivos especificadas:

```
acl extban url_regex -i \.avi \.exe \.mp3 \.torrent
http_access deny extban
```

Esta regra aceita também o uso de um arquivo externo, de forma que se a lista começar a ficar muito grande, você pode migrá-la para dentro de um arquivo de texto e especificar sua localização dentro da acl, como em:

```
acl extban url_regex -i "/etc/squid/extban"
http_access deny extban
```

Dentro do arquivo, você inclui as extensões desejadas (sem esquecer da barra invertida antes do ponto), uma por linha, como em:

```
\.avi
\.exe
\.mp3
\.torrent
```

Uma observação é que bloquear arquivos .torrent usando essa regra não impede que os usuários da rede utilizem o bittorrent, mas apenas que baixem os arquivos .torrent para iniciarem o download (o que acaba sendo o suficiente para fazer os usuários menos técnicos desistirem de baixar o arquivo). Para realmente bloquear o download de arquivos via bittorrent, é necessário bloquear diretamente os endereços dos trackers.

Continuando, sempre que fizer alterações na configuração, você pode aplicar as mudanças usando o comando:

 # /etc/init.d/squid reload

Evite usar o "/etc/init.d/squid restart" em ambientes de produção, pois ele força um reinício completo do Squid, onde o proxy precisa finalizar todas as conexões abertas, finalizar todos os processos e desativar o cache, para só então ler a configuração e carregar todos os componentes novamente. Isso faz com que o proxy fique vários segundos (ou até minutos, de acordo com o número de clientes conectados a ele) sem responder conexões, fazendo com que o acesso fique fora do ar:

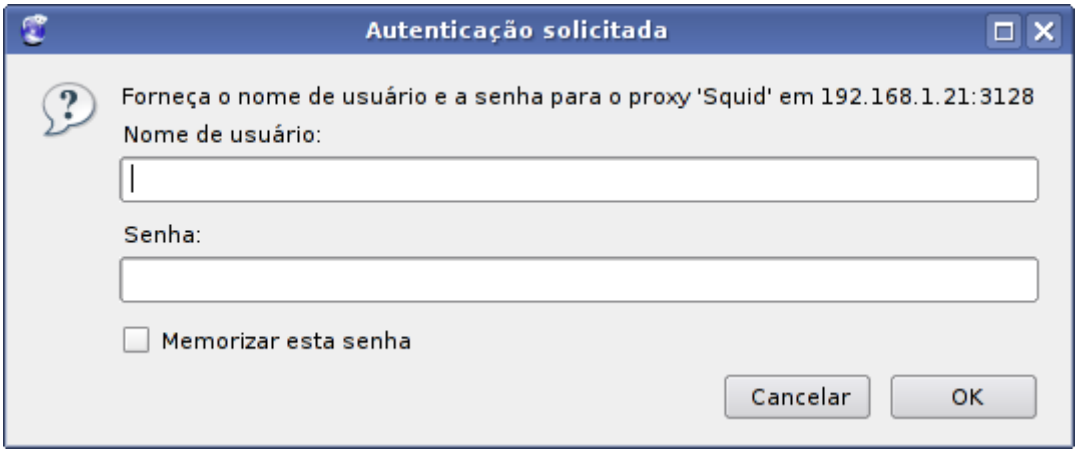
 # /etc/init.d/squid restart

Restarting Squid HTTP proxy: squid Waiting.....done.

O parâmetro "reload" permite que o Squid continue respondendo aos clientes e aplique a nova configuração apenas às novas requisições. Ele é suportado por diversos outros serviços onde o "restart" causa interrupção no serviço, como no caso do Apache.

Proxy com autenticação

Você pode adicionar uma camada extra de segurança exigindo autenticação no proxy. Este recurso pode ser usado para controlar quem tem acesso à internet e auditar os acessos em caso de necessidade. Quase todos os navegadores oferecem a opção de salvar a senha, de modo que o usuário precisa digitá-la apenas uma vez a cada sessão:



A forma mais simples de implementar autenticação no Squid é usando o módulo "ncsa_auth", que faz parte do pacote principal. Ele utiliza um sistema simples, baseado em um arquivo de senhas, onde você pode cadastrar e bloquear os usuários rapidamente.

Para criar o arquivo de senhas, precisamos do script "**htpasswd**". Nas distribuições derivadas do Debian ele faz parte do pacote **apache2-utils**, que você pode instalar via apt-get:

apt-get install apache2-utils

No CentOS e no Fedora ele faz parte do pacote principal do apache (o pacote "httpd"), que pode ser instalado através do yum.

Em seguida, crie o arquivo que será usado para armazenar as senhas, usando o comando "touch" (que simplesmente cria um arquivo de texto vazio):

touch /etc/squid/squid_passwd

O próximo passo é cadastrar os logins usando o htpasswd, especificando o arquivo que acabou de criar e o login que será cadastrado, como em:

htpasswd /etc/squid/squid_passwd gdh

Depois de terminar de cadastrar os usuários, adicione as linhas que ativam a autenticação no squid.conf:

```
auth_param basic realm Squid
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd
acl autenticados proxy_auth REQUIRED
http_access allow autenticados
```

O "auth_param basic realm Squid" indica o nome do servidor, da forma como ele aparecerá na janela de autenticação dos clientes; esta é na verdade uma opção meramente estética. O

"/usr/lib/squid/ncsa_auth" é a localização da biblioteca responsável pela autenticação. Eventualmente, ela pode estar em uma pasta diferente dentro da distribuição que estiver usando; nesse caso, use o comando "**locate**" ou a busca do sistema para encontrar o arquivo e altere a linha indicando a localização correta. Finalmente, o "/etc/squid/squid_passwd" indica a localização do arquivo de senhas que criamos no passo anterior.

Estas quatro linhas criam uma acl chamada "autenticados" (poderia ser outro nome), que contém os usuários que se autenticarem usando um login válido. Estas linhas devem ser colocadas antes de qualquer outra regra que libere o acesso, já que, se o acesso é aceito por uma regra anterior, ele não passa pela regra que exige autenticação.

Entretanto, se você usar uma configuração similar a essa:

```
auth_param basic realm Squid
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd
acl autenticados proxy_auth REQUIRED
http_access allow autenticados

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

... vai notar que a regra de autenticação essencialmente desativa a regra que bloqueia o acesso de usuários fora da rede local. Todos os usuários tem acesso ao prompt de autenticação e todos que se autenticam ganham acesso, mesmo que estejam utilizando endereços fora da faixa usada na rede. Para evitar isso, é necessário restringir o acesso de usuários fora da rede local antes da regra de autenticação. Veja um exemplo:

```
# Bloqueia acessos de fora da rede local antes de passar pela autenticação:
acl redelocal src 192.168.1.0/24
http_access deny !redelocal

# Outras regras de restrição vão aqui, de forma que o acesso seja negado
# antes mesmo de passar pela autenticação:
acl bloqueados url_regex -i "/etc/squid/bloqueados"
http_access deny bloqueados

# Autentica o usuário:
auth_param basic realm Squid
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd
acl autenticados proxy_auth REQUIRED
http_access allow autenticados

# Libera o acesso da rede local e do localhost para os autenticados,
# bloqueia os demais:
http_access allow localhost
http_access allow redelocal
http_access deny all
```

Veja que agora usamos a regra "http_access deny !redelocal" no início da cadeia. A exclamação inverte a lógica da regra, fazendo com que ela bloqueie todos os endereços que não fizerem parte da acl "redelocal".

Ao implementar a autenticação, você passa a poder criar regras de acesso com base nos logins dos usuários e não mais apenas com base nos endereços IP. Imagine, por exemplo, que você queira que apenas dois usuários da rede tenham acesso irrestrito ao proxy. Os demais (mesmo depois de autenticados), poderão acessar apenas no horário do almoço, e quem não tiver login e senha válidos não acessa em horário nenhum. Neste caso, você poderia usar esta configuração:

```
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd
acl autenticados proxy_auth REQUIRED

acl permitidos proxy_auth gdh tux
acl almoco time 12:00-13:00

http_access allow permitidos
http_access allow autenticados almoco
```

Aqui temos os usuários que passaram pela autenticação divididos em duas regras. A acl "autenticados" inclui todos os usuários, enquanto a acl "permitidos" contém apenas os usuários gdh e tux.

Graças à regra "http_access allow permitidos", os dois podem acessar em qualquer horário, enquanto os demais caem na regra "http_access allow autenticados almoco", que cruza o conteúdo das acls "autenticados" e "almoço", permitindo que eles acessem, mas apenas das 12:00 às 13:00.

Além do módulo ncsa_auth que, como vimos, permite usar um arquivo de senhas separado, válido apenas para o proxy, o Squid suporta também um conjunto de módulos que permitem fazer com que o Squid se autentique em um servidor externo, integrando o proxy a um sistema de autenticação já existente.

O mais simples é o módulo **smb_auth**, que permite que o Squid autentique os usuários em um servidor Samba, configurado como PDC. Com isso, os usuários passam a utilizar o mesmo login e senha que utilizam para fazer logon. Para usar o smb_auth, você usaria a configuração a seguir, especificando o domínio (na opção -W) e o endereço do servidor PDC (na opção -U):

```
auth_param basic realm Squid
authenticate_ip_ttl 5 minutes
auth_param basic program /usr/lib/squid/smb_auth -W dominio -U 192.168.1.254
acl autenticados proxy_auth REQUIRED
http_access allow autenticados
```

Com o módulo smb_auth, o Squid simplesmente repassa o login e senha fornecido pelo usuário ao servidor PDC e autoriza o acesso caso o PDC retorne uma resposta positiva. Não é necessário que o servidor com o Squid faça parte do domínio, nem que exista uma cópia do Samba rodando localmente (são necessários apenas os pacotes "samba-client" e "samba-common", que correspondem ao cliente Samba), por isso a configuração é bastante simples. Naturalmente, para utilizar esta configuração você deve ter um servidor PDC na sua rede, como aprenderemos a configurar em detalhes no capítulo sobre o Samba.

Outros módulos que podem ser usados são o "squid_ldap_auth", que permite que o servidor autentique os usuários em um servidor LDAP e o "ntlm_auth", que permite integrar o servidor Squid ao Active Directory.

Configurando um proxy transparente

Uma garantia de que os usuários realmente vão usar o proxy e, ao mesmo tempo, uma grande economia de trabalho e dor de cabeça para você é o recurso de proxy transparente. Ele permite configurar o Squid e o firewall de forma que o servidor proxy fique escutando todas as conexões na porta 80. Mesmo que alguém tente desabilitar o proxy manualmente nas configurações do navegador, ele continuará sendo usado. Outra vantagem é que este recurso permite usar o proxy sem precisar configurar manualmente o endereço em cada estação. Basta usar o endereço IP do servidor rodando o proxy como gateway da rede.

Lembre-se de que, para usar o proxy transparente, você já deve estar compartilhando a conexão no servidor via NAT, como vimos anteriormente. O proxy transparente apenas fará com que o proxy intercepte os acessos na porta 80, obrigando tudo a passar pelas suas regras de controle de acesso, log, autenticação e cache, diferente de um proxy tradicional, onde você não ativa o compartilhamento via NAT, fazendo com que o proxy seja a única porta de saída da rede.

Para ativar o proxy transparente, rode o comando abaixo. Ele direciona as requisições recebidas na porta 80 para o Squid:

**iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j **
REDIRECT --to-port 3128

O "eth0" no comando indica a placa da rede local, onde o proxy recebe as requisições dos outros micros da rede e o "3128" indica a porta usada pelo Squid. Adicione o comando junto com os 4 comandos que compartilham a conexão no final do arquivo "/etc/rc.local" ou ao seu script de firewall para que ele seja executado durante o boot.

Finalmente, você precisa ativar o suporte ao modo transparente dentro do arquivo "/etc/squid/squid.conf" e reiniciar o serviço.

Se você está usando uma versão recente, do **Squid 2.6** em diante, a configuração é mais simples. Basta substituir a linha "**http_port 3128**" no início do arquivo por:

http_port 3128 transparent

Ou seja, na verdade você precisa apenas adicionar o "transparent" para que o Squid passe a entender as requisições redirecionadas pela regra do firewall.

No caso das versões mais antigas, anteriores à 2.6 (como a usada no Debian Sarge e no Ubuntu 5.10), é necessário adicionar as quatro linhas abaixo, no final do arquivo "/etc/squid/squid.conf" (nesse caso, sem alterar a linha "http_port 3128"):

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Em qualquer um dos dois casos, você precisa reiniciar o serviço para que a alteração entre em vigor:

 # /etc/init.d/squid restart

Em caso de dúvida sobre qual versão do Squid está instalada, use o comando "squid -v", que além de reportar a versão, informa todas as opções que foram usadas durante a compilação:

 # squid -v

> Squid Cache: **Version 2.6.STABLE2**
>
> configure options: '--prefix=/usr' '--exec_prefix=/usr' '--bindir=/usr/sbin' '--sbindir=/usr/sbin' '--libexecdir=/usr/lib/squid' '--sysconfdir=/etc/squid' '--localstatedir=/var/spool/squid' '--datadir=/usr/share/squid' '--enable-async-io' '--with-pthreads' '--enable-storeio=ufs,aufs,diskd,null' '--enable-linux-netfilter' '--enable-linux-proxy' '--enable-arp-acl' '--enable-epoll' '--enable-removal-policies=lru,heap' '--enable-snmp' '--enable-delay-pools' '--enable-htcp' '--enable-cache-digests' '--enable-underscores' '--enable-referer-log' '--enable-useragent-log' '--enable-auth=basic,digest,ntlm' '--enable-carp' '--with-large-files' 'i386-debian-linux' 'build_alias=i386-debian-linux' 'host_alias=i386-debian-linux' 'target_alias=i386-debian-linux'

Em resumo, ao usar o proxy transparente você vai ter a conexão compartilhada via NAT no servidor e configurará os clientes para acessar através dela, colocando o servidor como gateway da rede. Ao ativar o proxy, a configuração dos clientes continua igual, a única diferença é que agora (graças à nova regra do Iptables) todo o tráfego da porta 80 passará, obrigatoriamente, pelo servidor Squid. Isso permite que você se beneficie do log dos acessos e do cache feito pelo proxy, sem ter que se sujeitar às desvantagens de usar um proxy, como ter que configurar manualmente cada estação.

Uma observação importante é que esta configuração de proxy transparente **não** funciona em conjunto com o sistema de autenticação incluso no Squid. Ao usar o proxy transparente a autenticação deixa de funcionar, fazendo com que você precise escolher entre as duas coisas.

Outra limitação importante do uso do proxy transparente é que ele atende apenas ao tráfego da porta 80. Como a conexão é compartilhada via NAT, todo o tráfego de outros protocolos (incluindo páginas em HTTPS, que são acessadas através da porta 443) é encaminhado diretamente, sem passar pelo proxy. Ou seja, embora seja uma forma simples de implementar um sistema de cache e algumas restrições de acesso, o uso do proxy transparente está longe de ser uma solução ideal.

Em situações onde você realmente precisa ter controle sobre o tráfego da rede, a única opção acaba sendo utilizar um proxy "normal", sem NAT. Uma solução para reduzir seu trabalho de administração nesse caso é implantar um sistema de configuração automática de proxy nos clientes.

Configuração automática de proxy nos clientes

Usar um proxy transparente é a forma mais simples de fazer com que todas as estações da rede utilizem o proxy, sem precisar configurar cada uma das máquinas manualmente. Entretanto, usar um proxy transparente está longe de ser uma solução perfeita, pois o proxy só atende a requisições na porta 80 (ou seja, não funciona para FTP, HTTPS e outros protocolos) e usar o proxy transparente automaticamente impede que seja usada autenticação dos usuários.

Se você acha as limitações de usar um proxy transparente pesadas demais, mas também não quer arcar com o trabalho de configurar cada estação para usar o proxy manualmente, existe a opção de usar um script **PAC** (Proxy Auto-Configuration), um arquivo que é disponibilizado na rede local através de um servidor web.

Os clientes são então configurados para buscarem a configuração de proxy no script. Esta não é exatamente uma configuração automática (você ainda tem o trabalho de configurar os clientes para utilizarem o script), mas é um bom ponto de partida. A principal vantagem em relação à configuração manual é que ao usar o script você pode alterar a configuração de proxy em todas as estações simplesmente modificando o script, em vez de precisar reconfigurar cada estação manualmente.

Para começar, você precisa instalar um servidor web em algum servidor da rede, que será usado para disponibilizar o arquivo. Para manter as coisas simples, você pode utilizar o próprio servidor que está disponibilizando o proxy. Basta instalar o Apache usando o gerenciador de pacotes, como em:

 # apt-get install apache2

ou:

 # yum install httpd

Em seguida, crie o arquivo "**/var/www/wpad.dat**" (no servidor), com o seguinte conteúdo:

```
function FindProxyForURL(url, host)
{
return "PROXY 192.168.1.1:3128";
}
```

No exemplo, o "192.168.1.1:3128" corresponde ao endereço do servidor proxy e a porta utilizada pelo Squid.

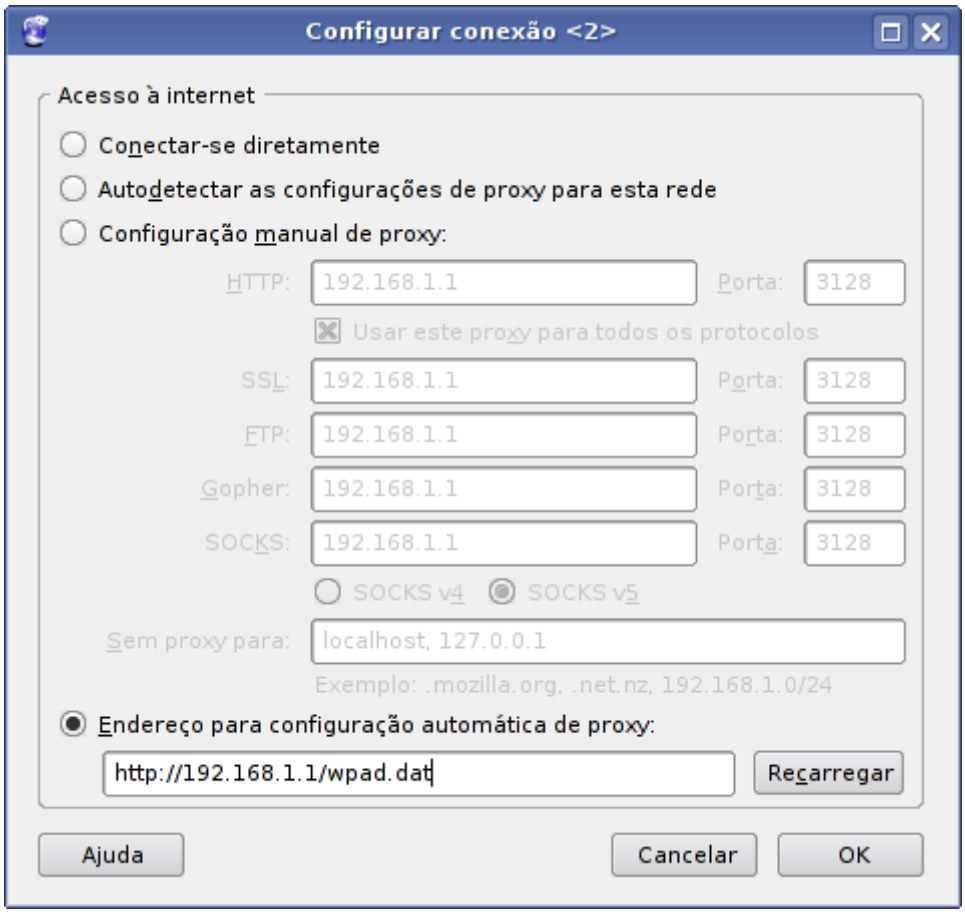
O diretório "/var/www" é o diretório raiz do servidor web, de forma que ao colocar o arquivo wpad.dat dentro dele, ele passará a ser disponibilizado através do endereço "http://ip-do-servidor/wpad.dat", como em "http://192.168.1.1/wpad.dat". O arquivo contém um pequeno javascript que será processado pelos clientes antes de cada requisição orientando-os a utilizarem o proxy.

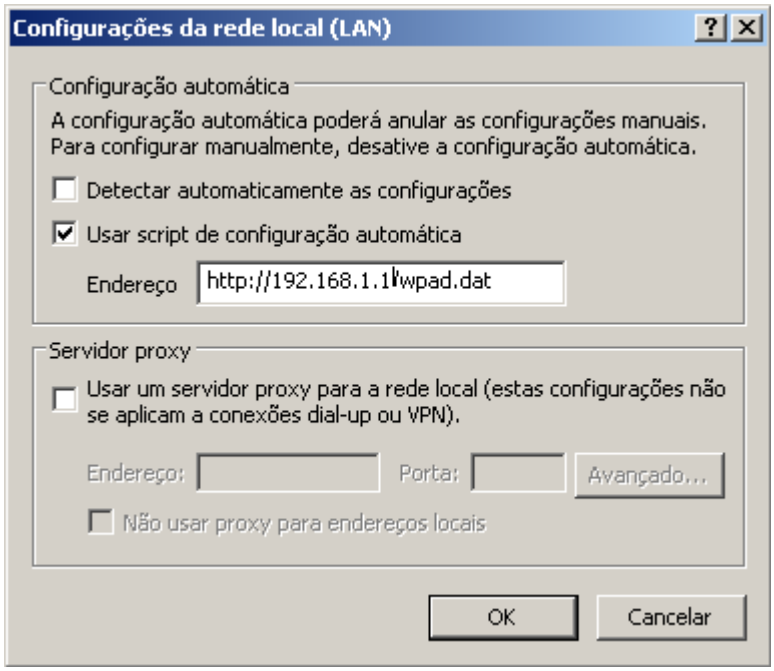
O arquivo wpad.dat pode incluir diversos outros parâmetros. Aqui temos uma versão um pouco mais incrementada do arquivo, que inclui exceções para o site da empresa (ou outro site qualquer, que você defina) e para a faixa de endereços da rede local; endereços que devem ser acessados diretamente, sem passar pelo proxy:

```
function FindProxyForURL(url, host) {
if (shExpMatch(url,"*.gdhpress.com.br/*")) {return "DIRECT";}
if (isInNet(host, "192.168.1.0", "255.255.0.0")) {return "DIRECT";}
return "PROXY 192.168.1.101:3128";
}
```

Você pode ver uma lista com outros parâmetros que podem ser usados no: http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html

Depois de disponibilizar o arquivo, falta apenas configurar os clientes para obterem a configuração de proxy através dele. No Firefox, o endereço vai no "Editar > Preferências > Avançado > Rede > Configurações > Endereço para configuração automática de proxy", enquanto no IE vai no "Ferramentas > Opções da Internet > Conexões > Configurações da LAN > Usar script de configuração automática":





Depois de feita a configuração, você pode checar o uso do proxy pelos clientes monitorando o arquivo "/var/log/squid/access.log" do servidor. Você verá várias entradas referentes aos acessos feitos pelos clientes.

Essa é a configuração mais elementar, onde os clientes são manualmente configurados para utilizarem o arquivo. O degrau seguinte é o **WPAD** (Web Proxy Auto-Discovery protocol), que permite automatizar a configuração dos clientes, permitindo que eles localizem o arquivo automaticamente.

Para usar o WPAD, precisaremos configurar também o servidor DHCP e o servidor DNS da rede para orientarem os clientes a utilizarem o arquivo. A alteração do servidor DHCP consiste em adicionar duas linhas no arquivo "**/etc/dhcp3/dhcpd.conf**", a primeira contendo a diretiva "code 252 = text" e a segunda contendo a localização do arquivo wpad.dat:

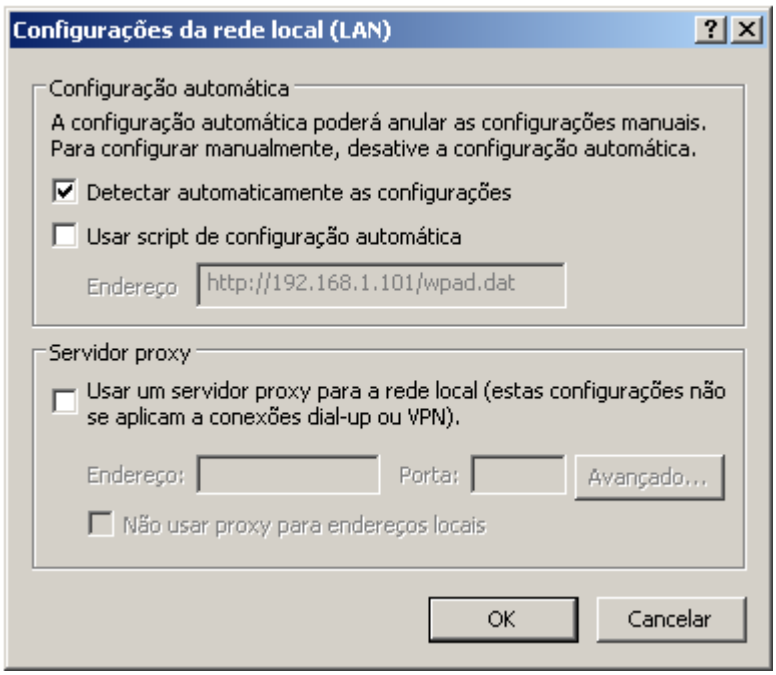
```
option wpad-url code 252 = text;
option wpad-url "http://192.168.1.1/wpad.dat\n";
```

A primeira linha é incluída na seção global da configuração, enquanto a segunda é incluída na seção correspondente à subnet, como em:

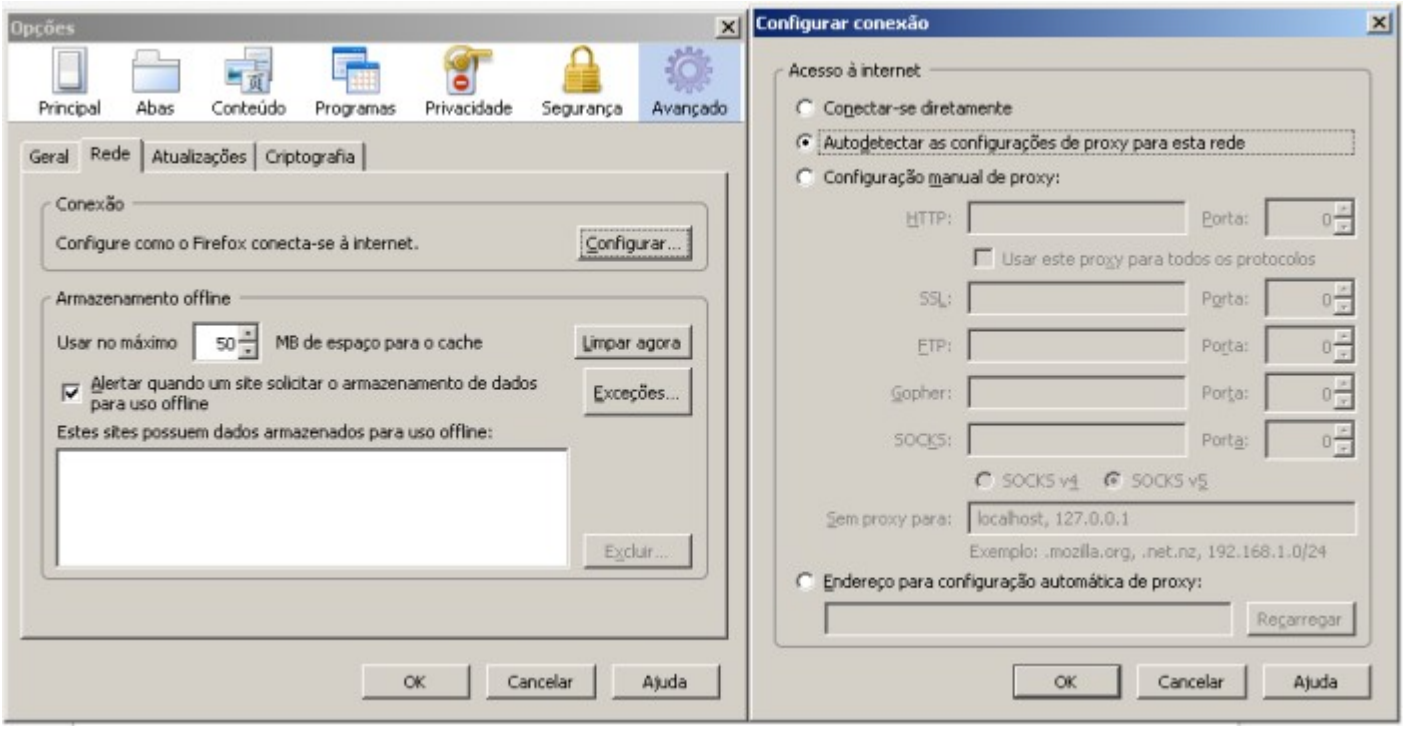
```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;
authoritative;
option wpad-url code 252 = text;

subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.100 192.168.1.199;
option routers 192.168.1.1;
option domain-name-servers 208.67.222.222;
option broadcast-address 192.168.1.255;
option wpad-url "http://192.168.1.1/wpad.dat\n";
}
```

O "/n" na última linha insere uma quebra de linha. Ele é um workaround para um bug do IE 6.0, que não lê a configuração se o \n não estiver presente. Depois de alterar o arquivo, não esqueça de reiniciar o serviço para que a alteração entre em vigor.

Depois de configurar o DHCP, você pode configurar os clientes com a opção "Detectar automaticamente as configurações" na configuração do proxy, em vez de especificar a localização do arquivo manualmente:



Atualmente, apenas o Internet Explorer é compatível com a configuração automática via proxy. Você pode selecionar a opção "Autodetectar as configurações de proxy para esta rede" no Firefox, mas você perceberá que os clientes continuarão tentando acessar a web diretamente, sem lerem o arquivo wpad.dat e sem utilizarem o proxy:



Isso nos leva à configuração do DNS, que complementa a configuração, atendendo a máquinas com o Firefox, Opera e outros navegadores.

A configuração do DNS é um pouco mais complexa, pois é necessário configurar também um domínio para a rede local e ajustar (novamente) também a configuração do servidor DHCP para que os clientes utilizem o domínio.

Mais adiante teremos um capítulo dedicado à configuração de servidores DNS utilizando o Bind, onde veremos mais detalhes sobre a configuração de domínios. Por enquanto, vou apenas me limitar a uma receita rápida para que você coloque o domínio no ar e possa assim ativar a configuração automática do proxy.

O primeiro passo é instalar o Bind usando o gerenciador de pacotes, como em:

 # apt-get install bind

O arquivo de configuração padrão é o "/etc/bind/named.conf". A configuração do domínio é feita em duas partes. Primeiro adicionamos uma entrada referente ao domínio no arquivo principal, indicando a localização de um arquivo externo (onde vai a configuração) e em seguida adicionamos a configuração propriamente dita neste segundo arquivo. Como estamos configurando um domínio local, você pode especificar qualquer domínio, não é necessário que ele esteja realmente registrado. No exemplo, estou usando o domínio "gdhn.com.br".

Comece adicionando a configuração referente a ele no final do arquivo "**/etc/bind/named.conf**" (no Debian você pode também utilizar o arquivo "/etc/bind/named.conf.local", que é carregado como um include):

```
zone "gdhn.com.br" IN {
type master;
file "/etc/bind/db.gdhn";
};
```

Veja que na terceira linha especificamos o arquivo externo (db.gdhn), onde irá a configuração do domínio. Esse arquivo precisa ser criado na mesma pasta do arquivo principal, ou seja, será o arquivo "**/etc/bind/db.gdhn**". O conteúdo do arquivo será o seguinte (veja mais detalhes sobre o significado das opções no capítulo sobre DNS):

```
@ IN SOA etch.gdhn.com.br. hostmaster.gdhn.com.br. (
2008030645 3H 15M 1W 1D )
NS etch.gdhn.com.br.
gdhn.com.br. A 192.168.1.1
wpad IN A 192.168.1.1
```

O "etch" no exemplo é o nome do servidor, enquanto o "192.168.1.1" é o endereço IP usado por ele. Isso cria o domínio "gdhn.com.br" e o "wpad.gdhn.com.br", ambos apontando para o endereço IP do servidor. Dessa forma, ao tentarem baixar o arquivo "http://wpad.gdhn.com.br/wpad.dat", os clientes baixarão na verdade o "http://192.168.1.1/wpad.dat". Depois de terminar, não esqueça de reiniciar o Bind para que a alteração entre em vigor:

```
# /etc/init.d/bind restart
```

Naturalmente, para que o domínio seja utilizado, é necessário também configurar os clientes para utilizarem o servidor DNS interno que acabamos de configurar. Isso é feito especificando o endereço IP do servidor como único servidor DNS na opção "domain-name-servers" da configuração do DHCP e adicionando duas novas opções na seção global do arquivo:

```
ddns-domainname "gdhn.com.br.";
option domain-name "gdhn.com.br.";
```

Este é um exemplo de arquivo "**/etc/dhcp3/dhcpd.conf**", depois das alterações:

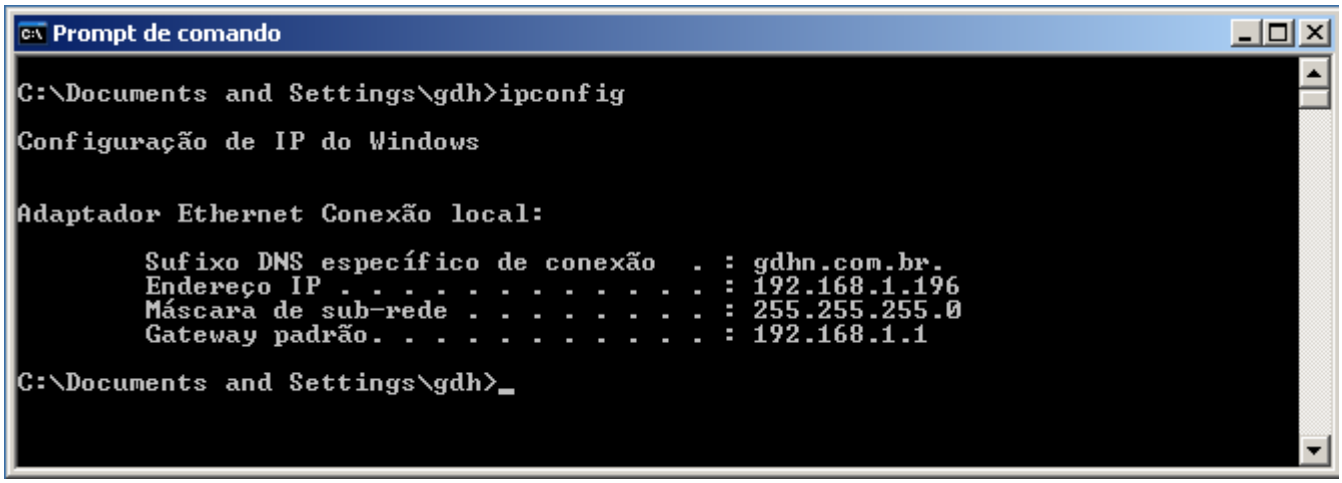
```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;
authoritative;
option wpad-url code 252 = text;
ddns-domainname "gdhn.com.br.";
option domain-name "gdhn.com.br.";

subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.100 192.168.1.199;
option routers 192.168.1.1;
option domain-name-servers 192.168.1.1;
option broadcast-address 192.168.1.255;
option wpad-url "http://192.168.1.1/wpad.dat\n";
}
```

Com essa configuração, os clientes Linux receberão a seguinte configuração de DNS (salva no arquivo "/etc/resolv.conf") do servidor DHCP:

```
search gdhn.com.br.
nameserver 192.168.1.1
```

Veja que é incluída a linha "search gdhn.com.br", que especifica que eles devem fazer pesquisas dentro do domínio e que o endereço do servidor é usado como DNS primário.

Naturalmente, a mesma configuração é fornecida aos clientes Windows configurados via DHCP. Veja um exemplo:



Com isso, você pode configurar o Firefox para localizar o proxy automaticamente e, graças à configuração do DNS, ele será capaz de encontrar o arquivo wpad.dat e utilizar a configuração definida por você.

Uma pequena exceção fica por conta de versões antigas do Firefox para Linux, que possuem um bug na variável usada para localizar o arquivo. Em vez de procurarem o arquivo wpad.dat no host "wpad" dentro do domínio da rede (que leva ao nosso servidor), elas tentam sempre baixar o arquivo a partir da URL "http://wpad/wpad.dat", sem respeitar a configuração do DNS.

Ao usar clientes Linux rodando alguma das versões afetadas pelo bug, você verá uma série de entradas como essa no log do Squid:

192.168.1.183 TCP_MISS/503 1479 GET **http://wpad/wpad.dat** - DIRECT/wpad text/html

A solução nesses casos é editar o arquivo "**/etc/hosts**" nos clientes afetados, adicionando uma entrada relacionando o host "wpad" com o endereço do seu servidor, como em:

192.168.1.1 wpad

Com isso, as requisições passam a ser destinadas ao endereço correto, solucionando o problema.

Mais detalhes sobre a configuração dos caches

A configuração dos caches é o parâmetro da configuração que afeta mais diretamente o desempenho do servidor proxy e por isso deve ser sempre definida com cuidado em qualquer servidor que irá atender a um grande volume de usuários. Embora a configuração pareça simples, ela na verdade esconde diversos detalhes pouco intuitivos.

Como comentei a pouco, em um servidor de rede local que atende um pequeno volume de clientes, você pode reservar apenas 32 ou 64 MB de memória RAM para o cache do Squid (de forma que ele não consuma toda a memória do servidor, prejudicando seu desempenho em outras tarefas) e, em um servidor dedicado para uma rede de maior porte você pode reservar até 1/3 da memória total do servidor. Você deve ter se perguntado por que reservar apenas 1/3 da memória, se a função do servidor será rodar apenas o proxy. Porque não reservar logo toda a memória para o Squid?

A resposta é que além da RAM reservada ao cache em memória, o Squid precisa de memória RAM para diversas outras tarefas, incluindo o gerenciamento das conexões e armazenamento da metadata dos arquivos armazenados no cache em disco, sem falar da memória RAM consumida pelo sistema operacional para fazer cache de disco e outras atividades. Reservando muita memória para o cache, o sistema é obrigado a utilizar memória swap, o que acaba reduzindo o desempenho em vez de aumentar.

Em uma pequena rede, raramente o desempenho do HD será um gargalo, já que o proxy precisará atender a um pequeno volume de requisições. Entretanto, em uma grande rede, com mais do que 100 ou 200 clientes o desempenho do proxy é freqüentemente gargalado pelo volume de leituras que o HD é capaz de realizar.

Uma dica nesse caso é utilizar HDs SATA (ou SCSI) com suporte a NCQ; neles a controladora pode realizar leituras fora de ordem, levando em conta a posição dos arquivos nos discos magnéticos. Isso faz com que eles sejam capazes de realizar um volume de leituras de pequenos arquivos muito maior do que HDs IDE e HDs SATA sem NCQ (em algumas situações específicas, até duas vezes mais), o que melhora assustadoramente o desempenho em servidores proxy. Verifique também a possibilidade de adicionar mais memória RAM ao servidor, já que com um cache maior na memória RAM, menos arquivos precisarão ser lidos a partir do HD.

Outra dica é que você pode reduzir bastante o volume de operações de acesso a disco fazendo com que o cache despreze arquivos muito pequenos (menores do que 2 KB, por exemplo) através da opção "minimum_object_size", como em:

maximum_object_size 512 MB
minimum_object_size 2 KB

Isso faz com que pequenas imagens e outros tipos de arquivos muito pequenos, usados em profusão em muitas páginas web sejam simplesmente baixados novamente a cada acesso, em vez de precisarem ser lidos no cache do HD. Como os arquivos são muito pequenos, o aumento no uso do link não deve ser considerável.

Continuando, se você precisar alterar a localização da pasta do cache na linha "cache_dir" (para colocá-lo em uma pasta em uma partição separada, por exemplo), você vai logo perceber que o Squid deixará de inicializar, acusando um erro de permissão no diretório do cache, como nesse exemplo:

```
Restarting Squid HTTP proxy: squid* Creating squid spool directory structure
2008/06/31 16:35:46| Creating Swap Directories
FATAL: Failed to make swap directory /mnt/sda2/squid/00: (13) Permission denied
Squid Cache (Version 2.6.STABLE5): Terminated abnormally.
CPU Usage: 0.000 seconds = 0.000 user + 0.000 sys
Maximum Resident Size: 0 KB
Page faults with physical i/o: 0
```

Para solucionar o problema, pare o Squid, ajuste as permissões da nova pasta, de forma que a posse seja transferida para o usuário "proxy" e o grupo "proxy" (usados pelo Squid) e, para concluir, execute o comando "squid -z", que faz com que ele reformate o diretório do cache, criando a estrutura apropriada:

```
# /etc/init.d/squid stop
# chown -R proxy.proxy /mnt/sda2/squid
# squid -z
# /etc/init.d/squid start
```

Em versões antigas do Squid você ficava limitado a um único diretório de cache, de forma que a única forma de adicionar uma nova partição era realmente movendo o cache para ela. Nas versões atuais (desde o Squid 2.0) existe a opção de simplesmente adicionar novas linhas "cache_dir" na configuração, sem apagar as antigas. Isso permite que você simplesmente adicione novas pastas e partições ao cache, mantendo as antigas, como em:

```
cache_dir ufs /var/spool/squid 2048 16 256
cache_dir ufs /mnt/sda2/squid 5120 16 256
cache_dir ufs /mnt/sda3/squid 10240 16 256
```

Não esqueça de que, ao adicionar uma nova pasta, você deve parar o proxy, ajustar as permissões de acesso e rodar o comando "squid -z" para que o Squid crie as estruturas necessárias, como no exemplo anterior.

Note que, mesmo ao usar uma partição separada só para o cache, você não deve reservar mais do que 80% do espaço total da partição para ele, pois o Squid precisa de um pouco de espaço extra para manter um arquivo com o status do cache e para operações de organização em geral, sem falar que é importante manter uma pequena percentagem de espaço livre na partição para reduzir a fragmentação. Se você tiver uma partição de 10 GB e usar na configuração "cache_dir ufs /var/spool/squid 10480 16 256" (reservando exatamente 10 GB para o cache), o Squid vai travar depois de algum tempo, por falta de espaço em disco. O correto no caso seria usar "cache_dir ufs /var/spool/squid 8192 16 256", reservando 8 GB em vez de 10.

Uma observação final é que o volume de espaço em disco reservado ao cache tem efeito sobre o volume de memória RAM consumido pelo Squid (em adição ao cache na memória), pois o Squid precisa manter carregadas informações sobre os arquivos armazenados no cache para localizá-los de forma eficiente.

De uma forma geral, para cada gigabyte de espaço em disco reservado para o cache, o Squid consome cerca de 10 MB a mais de memória RAM (o valor real varia de acordo com o tamanho dos arquivos armazenados no cache), de forma que um cache de 20 GB, por exemplo, aumenta o volume de memória usado pelo Squid em aproximadamente 200 MB.

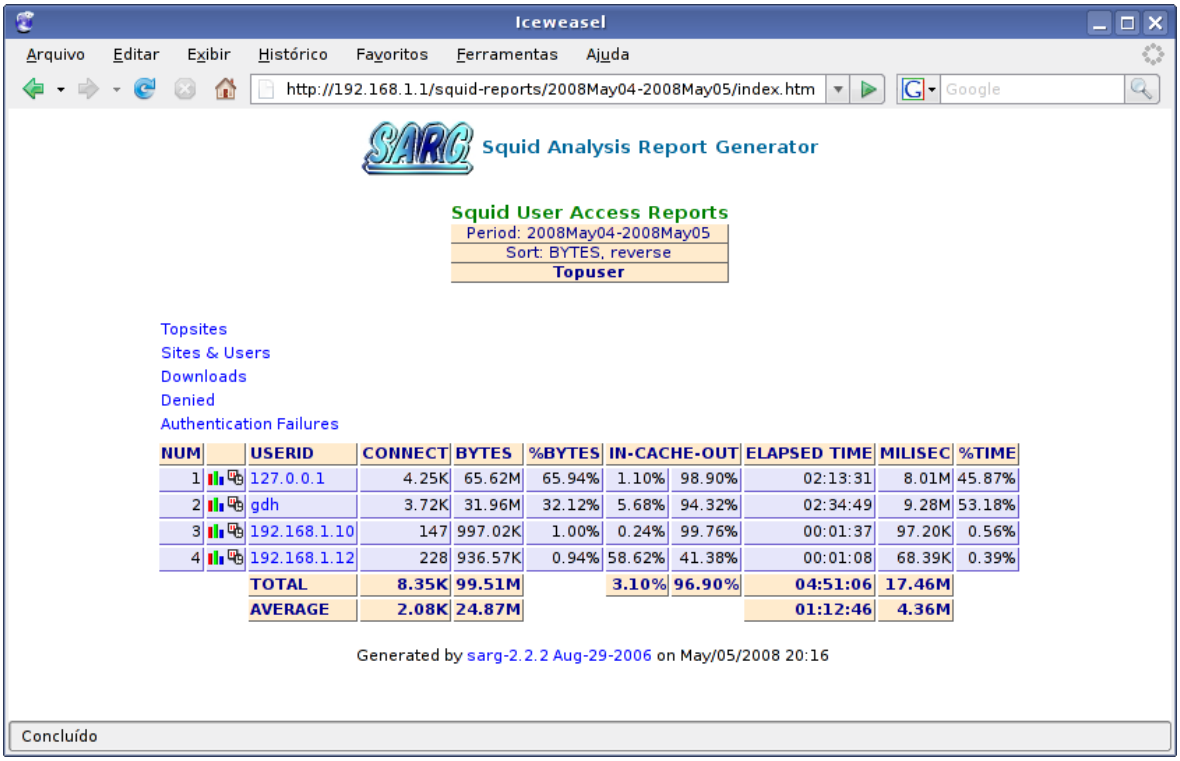
Devido a isso, não é recomendável usar um cache em disco muito grande, a menos que o servidor realmente possua muita memória disponível e precise atender a um volume muito grande de clientes. Para um servidor de rede local, um cache de 5 ou 10 GB já é mais do que suficiente.

Usando o Sarg para monitorar o acesso

O Sarg é um interpretador de logs para o Squid, assim como o Webalizer é para o Apache. Sempre que executado, ele cria um conjunto de páginas, divididas por dia, com uma lista de todas as páginas que foram acessadas e a partir de que máquina da rede veio cada acesso. Caso você tenha configurado o Squid para exigir autenticação, ele organiza os acessos com base nos logins dos usuários. Caso contrário, ele mostra os endereços IP das máquinas.

A partir daí, você pode acompanhar as páginas que estão sendo acessadas (mesmo que não exista nenhum filtro de conteúdo) e tomar as medidas cabíveis em casos de abuso. Todos sabemos que os filtros de conteúdo nunca são completamente eficazes, eles sempre bloqueiam algumas páginas úteis e deixam passar muitas páginas impróprias. Se você tiver algum tempo para ir acompanhando os logs, a inspeção manual acaba sendo o método mais eficiente. Você pode ir fazendo um trabalho incremental, ir bloqueando uma a uma as páginas onde os usuários perdem muito tempo, ou fazer algum trabalho educativo, explicando que os acessos estão sendo monitorados e estabelecendo algum tipo de punição para quem abusar.

Aqui está um exemplo do relatório gerado pelo Sarg. Por padrão, ele gera um conjunto de páginas html dentro da pasta "**/var/www/squid-reports/**" (ou "/var/www/html/squid/", nas distribuições derivadas do Red Hat), que você pode visualizar através de qualquer navegador:



Os acessos são organizados por usuário (caso esteja sendo usada autenticação) ou por IP, mostrando as páginas acessadas por cada um, quantidade de dados transmitidos, tempo gasto em cada acesso, tentativas de acesso bloqueadas pelos filtros de conteúdo e outras informações.

Os logs são inicialmente organizados por período, sendo que os relatórios antigos são mantidos quando o relatório é atualizado (com o tempo o relatório acaba armazenando um volume muito grande de informações). Dentro do relatório de cada período, você tem a lista dos endereços IP e/ou

dos usuários autenticados que utilizaram o proxy e, dentro do relatório referente a cada um, você pode acompanhar o log das páginas acessadas e outras informações, de forma bastante detalhada.

O Sarg é incluído na maioria das distribuições atuais, em alguns casos instalado por padrão junto com o Squid. No Debian e derivados ele pode ser instalado via apt-get:

 # apt-get install sarg

No Mandriva, ele é instalado usando o urpmi e assim por diante:

urpmi sarg

Depois de instalado, chame o comando "**sarg**" (como root) para que os relatórios sejam gerados a partir do log do Squid. O Sarg não é um daemon que fica residente, você precisa apenas chamá-lo quando quiser atualizar o relatório:

 # sarg

Para automatizar esta tarefa, você pode usar o cron para que ele seja executado automaticamente todos os dias ou uma vez por hora, por exemplo. No Debian (e na maioria das outras distribuições) é criado automaticamente um script dentro da pasta "/etc/cron.daily/", que faz com que ele seja executado todos os dias às 6:25 da manhã.

Dentro da mesma pasta, você encontrará um script que executa o logrotate, o serviço do sistema responsável por rotacionar os logs, evitando que eles cresçam até ocupar todo o espaço disponível no HD do servidor. Todos os dias, o logrotate renomeia e compacta os arquivos de log, incluindo o log do Squid, fazendo com que o log do dia anterior receba a extensão ".1" e os logs seguintes as extensões ".2.gz", ".3.gz" e assim por diante. Com isso, a pasta "/var/log/squid" conterá uma seqüência de arquivos (access.log, access.log.1, access.log.2.gz, access.log.3.gz e assim por diante).

No caso do Debian, o logrotate é corretamente configurado para executar o Sarg antes de rotacionar os logs, de forma que ele não deixe de contabilizar os acessos. Caso tenha problemas em outras distribuições, experimente renomear o arquivo "/etc/cron.daily/sarg" para "/etc/cron.daily/ksarg", de forma que ele seja executado imediatamente antes do "/etc/cron.daily/logrotate".

Você pode alterar a pasta onde são salvos os relatórios, limitar o acesso às estatísticas e alterar várias opções cosméticas no arquivo de configuração do Sarg, que é o "**/etc/squid/sarg.conf**" (ou o "/etc/sarg/sarg.conf", de acordo com a distribuição usada). O arquivo é auto-explicativo, nele você pode alterar os diretórios-padrão, alterar o layout da página, cores e títulos, etc. Outro recurso interessante é o envio de uma cópia do relatório por e-mail sempre que o Sarg for executado.

As duas linhas mais importantes dentro do arquivo são a "output_dir /var/www/squid-reports", onde é especificada a pasta onde serão armazenados os relatórios (que, naturalmente, precisa existir no sistema) e a "access_log /var/log/squid/access.log", onde é indicada a localização do arquivo de log do Squid. Ao configurar o Squid para salvar os logs em outro arquivo, é necessário especificá-lo dentro da configuração do Sarg. Se você quer que o relatório seja gerado em Português, substitua a linha "language English" (logo no início do arquivo) para "language Portuguese".

Inicialmente, o relatório poderá ser visualizado apenas a partir do próprio servidor, o que pode ser imprático se você o acessa remotamente. Se você manteve o default, que é salvar os relatórios dentro do diretório "/var/www", pode disponibilizar o relatório para a rede simplesmente instalando o Apache, como em:

 # apt-get install apache2

A partir daí, você pode acessar os relatórios através do http://ip-do-servidor/squid-reports", como em "http://192.168.1.1/squid-reports". Para evitar que outros usuários da rede fiquem bisbilhotando nos relatórios, você pode proteger a pasta com senha, usando um arquivo .htaccess, como veremos em detalhes no capítulo sobre o Apache.

Monitorando com o ntop

Um dos problemas em utilizar um proxy transparente é que o log do Squid e, consequentemente, o relatório do Sarg, mostra apenas o tráfego HTTP, deixando de fora todos os demais protocolos.

Uma forma simples de monitorar o tráfego global da sua rede interna é instalar o ntop no gateway da rede. Ele monitora todo o tráfego, mostrando o volume de banda consumido por cada máquina da rede e, dentro de cada relatório, informações detalhadas sobre os protocolos utilizados. Através dele, é fácil identificar máquinas da rede que estão consumindo uma grande quantidade de banda devido à utilização de programas P2P como o Kazaa ou o Bittorrent, por exemplo.

Para instalar o ntop em distribuições derivadas do Debian, basta instalar o pacote via apt-get:

 # apt-get install ntop

Caso o ntop não esteja disponível através do gerenciador de pacotes da distribuição em uso, você pode baixá-lo no http://www.ntop.org/, onde estão disponíveis pacotes para várias distribuições.

Com o pacote instalado, execute o comando "ntop" como root para definir a senha de administração:

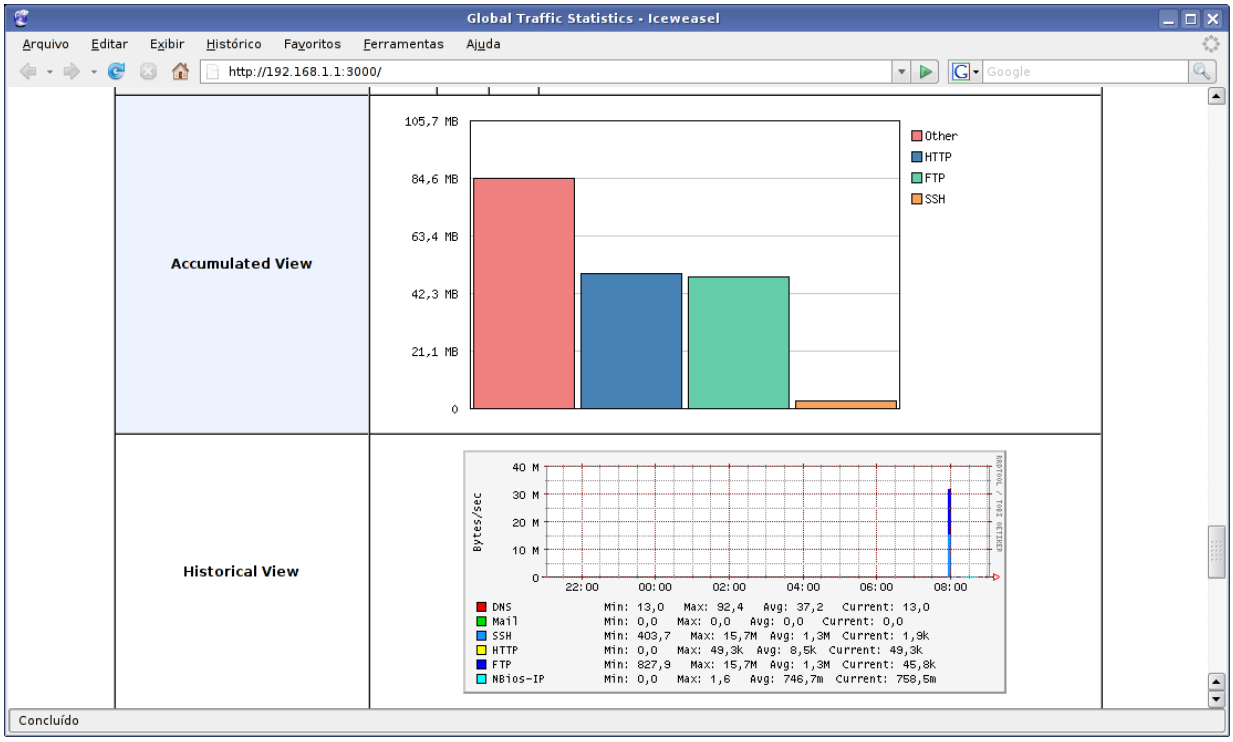
 # ntop

```
ntop startup - waiting for user response!
Please enter the password for the admin user: ********
Please enter the password again: ********
Ter 06 Mai 2008 07:36:34 BRT Admin user password has been set
```

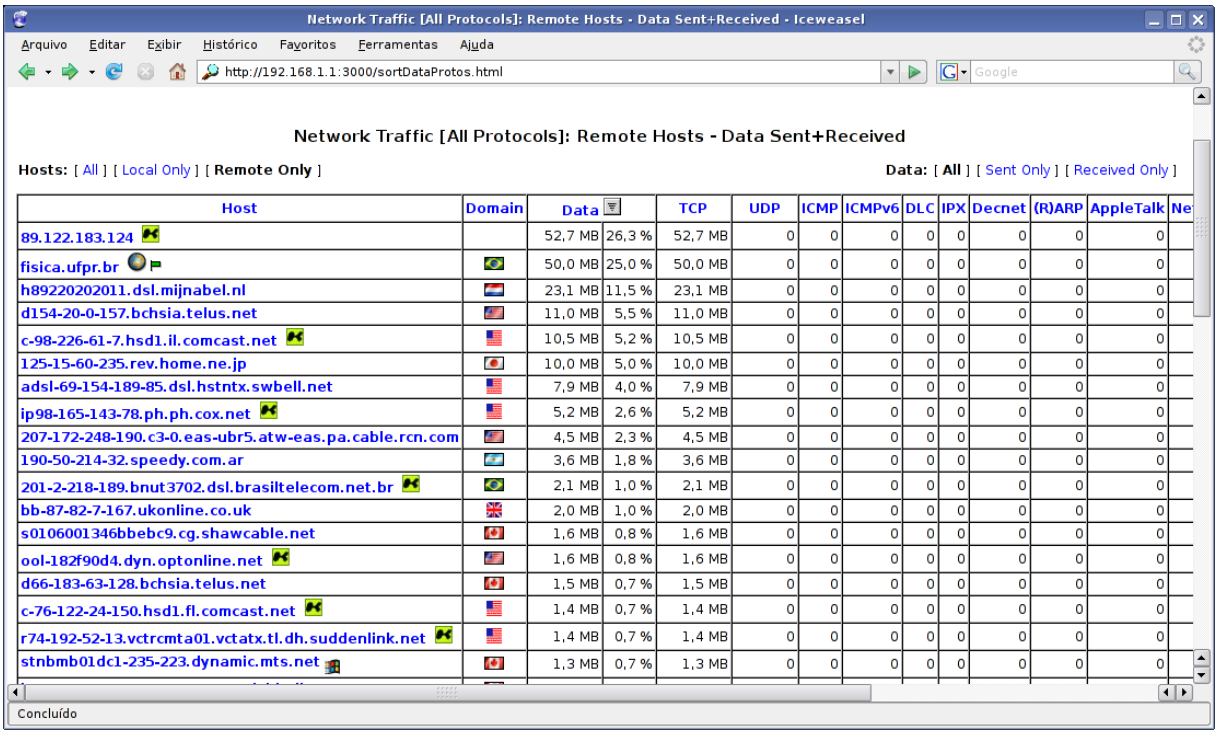
Com a senha definida, pressione "Ctrl+C" para abortar o comando e inicialize o serviço para que o ntop passe a operar em background:

 # /etc/init.d/ntop start

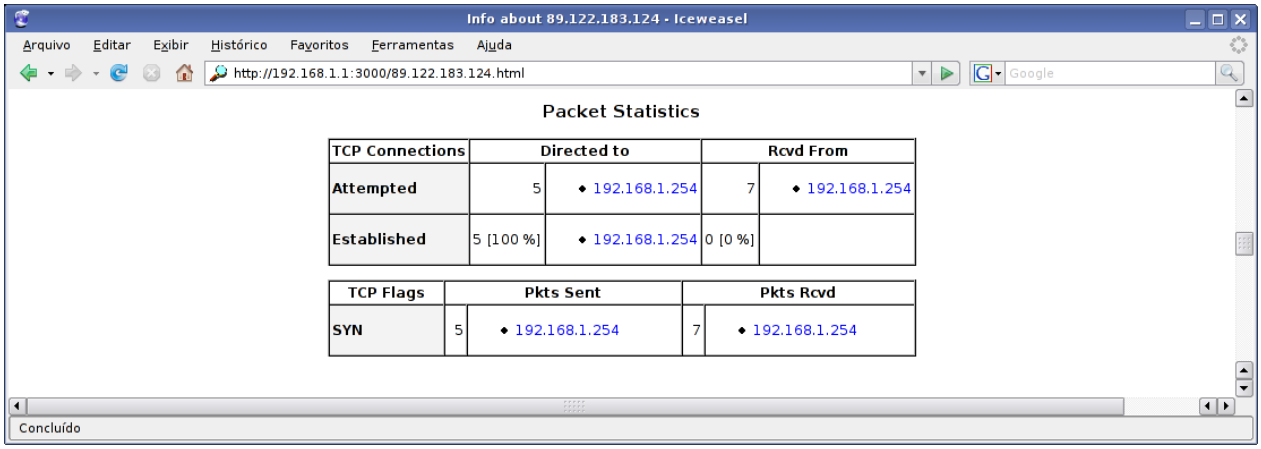
A partir daí, você pode acessar os relatórios, atualizados em tempo real a partir do "http://ip_do_servidor:3000", como em "http://192.168.1.1:3000". Caso o suporte a SSL tenha sido ativado durante a compilação do pacote (ele é um componente opcional, que vem desativado por padrão em muitas distribuições), você poderá acessar a interface também através do "https://ip_do_servidor:3001":



O relatório do ntop pode parecer simples à primeira vista, mas ele esconde um volume surpreendente de detalhes sobre as conexões. Acessando a seção "All Protocols > Traffic" (no menu superior), por exemplo, você tem acesso a um relatório dos hosts de internet que foram acessados através da conexão, organizados de acordo com o volume de dados transferidos:



Os hosts recebem um ícone de classificação de acordo com o tipo de tráfego predominante. Uma bandeira verde indica um site que hospeda arquivos legítimos, enquanto um "K" indica um servidor do Kazaa ou um tracker Bittorrent. Clicando sobre os hosts, você tem acesso a um relatório detalhado com o tipo de tráfego, horários de maior acesso e, o mais importante, uma lista dos endereços IP da rede que acessaram o servidor, o que permite localizar estações rodando programas P2P ou outros aplicativos que consomem muito tráfego da rede:



Diferente do relatório do Sarg, que loga apenas o tráfego que passa pelo Squid, o ntop realmente monitora todo o tráfego de dados que passa pelo servidor, independentemente do protocolo usado. Desde que todo o tráfego de internet realmente passe pelo servidor (ou seja, que ele seja o único gateway disponível), você pode ter certeza de que tudo será logado. Da próxima vez que alguém reclamar que a rede está lenta, bastará olhar o relatório para descobrir o motivo.

Uma dica é que no Debian Etch existe um pequeno bug no script de instalação do pacote que faz com que não seja criado o diretório "/var/lib/ntop/rrd" (onde o ntop armazena sua base de dados) e as permissões do diretório "/var/lib/ntop/" sejam definidas incorretamente, impedindo a operação normal do serviço. Você pode solucionar o problema usando os comandos abaixo:

 # mkdir /var/lib/ntop/rrd
 # chown -R ntop.ntop /var/lib/ntop/

Usando o SquidGuard para bloquear páginas impróprias

Bloquear domínios e endereços IP individuais funciona bem para bloquear páginas específicas, mas não funciona para bloquear páginas pornográficas, por exemplo, simplesmente porque existem muitas delas e você iria morrer louco se tentasse bloquear todas manualmente.

Existem grupos destinados a manter listas com URLs de páginas pornográficas, páginas de cassinos e jogos e páginas ilícitas em geral, que são atualizadas freqüentemente. Por serem construídas através da combinação dos esforços de muitas pessoas, auxiliadas por ferramentas semi-automáticas de indexação e classificação de conteúdo, estas listas permitem bloquear a maior parte das páginas ilícitas sem muito esforço. Apenas a lista mantida pelo Shalla Security possui mais de um milhão e meio de URLs cadastradas, que formam um arquivo compactado de 9 MB.

A lista mais usada é provavelmente a **MESD blacklists**, que é a indicada pela equipe do SquidGuard, por ser completamente livre e utilizável para qualquer fim. Ela tem pouco mais de 1 milhão de links e pode ser baixada no: http://squidguard.mesd.k12.or.us/blacklists.tgz.

Outra lista muito usada é a **Shalla's Blacklists**, disponível no: http://www.shallalist.de/. A lista é livre para uso pessoal ou não-comercial e é mais completa que a lista do MESD, com mais de 1.500.000 de URLs. O uso comercial é permitido, desde que você preencha um contrato de uso, sem custo.

Outra opção é a lista do **URLBlacklist.com**. Ela é uma lista comercial, que conta com mais de 2 milhões de links e é atualizada regularmente, contando inclusive com um script de atualização automática. A assinatura custa de US$ 6 a US$ 55 mensais, de acordo com o tipo de uso.

Estas listas nada mais são do que longas listas de links, com um por linha. Elas até podem ser usadas diretamente no Squid, através da opção url_regex (a mesma que usamos para criar uma lista de sites bloqueados), mas, por serem arquivos muito grandes, o desempenho seria ruim, já que o Squid processa cada linha dos arquivos a cada acesso, o que consome muito processamento.

Entra em cena então o SquidGuard, que permite usar longas listas de URLs, com milhões de links sem uma grande perda de desempenho. Ele permite integrar as listas que vimos a pouco sem comprometer o desempenho do seu servidor proxy. As listas se encarregarão de bloquear a maior parte das páginas impróprias e você poderá fazer ajustes manuais conforme necessário. A página do projeto é a: http://www.squidguard.org/.

Nas distribuições derivadas do Debian, você pode instalá-lo rapidamente via apt-get:

 # apt-get install squidguard

A configuração é feita em três fases. O primeiro passo é baixar os arquivos das listas desejadas e descompactá-los no diretório "/var/lib/squidguard/db". Em seguida, é necessário configurar o arquivo "/etc/squid/squidGuard.conf", especificando os arquivos de listas que serão usados e o comportamento do SquidGuard ao bloquear os acessos e, finalmente, editar o "/etc/squid/squid.conf", adicionando a linha que ativa o uso do SquidGuard.

Vamos começar baixando as listas. Vou usar como exemplo as listas do Shalla e do MESD, mas você pode usar os mesmos passos para utilizar outras listas que desejar.

Comece baixando as listas em um diretório qualquer, como em:

$ wget -c http://squidguard.mesd.k12.or.us/blacklists.tgz
$ wget -c http://www.shallalist.de/Downloads/shallalist.tar.gz

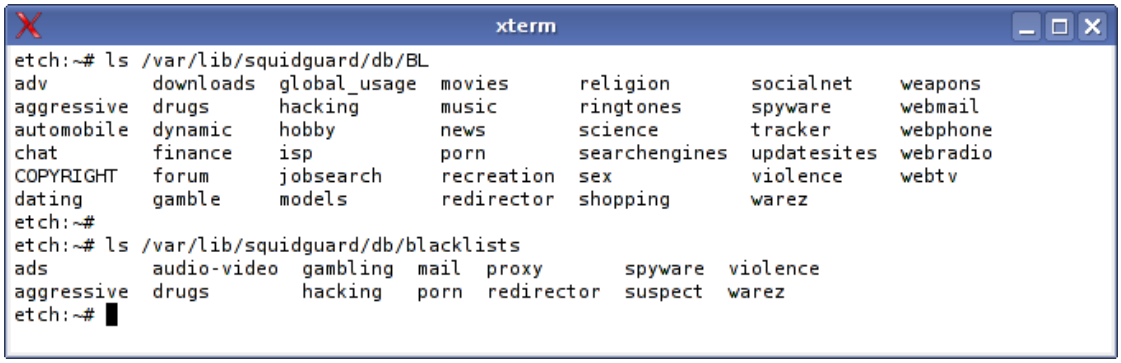
Copie os dois arquivos para o diretório "**/var/lib/squidguard/db**" e descompacte-os, como em:

 # cp blacklists.tgz shallalist.tar.gz /var/lib/squidguard/db/
 # cd /var/lib/squidguard/db/
 # tar -zxvf blacklists.tgz
 # tar -zxvf shallalist.tar.gz

Aproveite para remover os dois arquivos, já que não precisaremos mais deles:

 # rm -f blacklists.tgz shallalist.tar.gz

Com isso, você terá as pastas "BL" (as listas do Shalla) e "blacklists" (listas do MESD) dentro do diretório, cada uma contendo um conjunto de subpastas, como neste screenshot:



Como pode ver, as listas são divididas por assunto. A lista do MESD é concentrada em temas ilegais, enquanto a lista do Shalla inclui listas relacionadas a temas diversos, que você pode bloquear ou não de acordo com a situação. Dentro de cada pasta, você encontra dois arquivos, "domains" e "urls", o primeiro contendo domínios que são bloqueados completamente e o segundo contendo URLs isoladas.

Alguns dos temas cobertos pelas listas são:

adv - Sites que hospedam banners e anúncios em geral, exibidos em outras páginas (bloqueando estes domínios, você bloqueia boa parte dos banners de anúncios). Na lista do MESD a categoria se chama "ads".
audio-video - Páginas (não necessariamente ilegais) que hospedam vídeos e músicas, como o youtube.
chat - Páginas com salas de bate-papo ou contendo clientes web para acesso ao MSN ou outras redes.
drugs - sites que vendem remédios e páginas com conteúdo relacionado ao uso de drogas ou apologia às drogas.
finance - Páginas com informações financeiras, incluindo bancos, empresas de seguros e de crédito. Esta pasta inclui várias subpastas, com as páginas divididas em categorias.
gamble - Cassinos e outras páginas relacionadas a jogos de aposta.
porn - Este dispensa comentários. É a categoria com o maior número de entradas e a

primeira que você precisará bloquear.
sex - Similar à "porn", contém páginas de conteúdo adulto.
proxy - Esta é outra lista que você sempre deve incluir no bloqueio. Ela reúne URLs e endereços IP de proxys externos, que podem ser usados pelos usuários para furar o bloqueio, como o http://proxy.org.
socialnet - Inclui redes sociais, como o Orkut e o Myspace.
tracker - Endereços de trackers com torrents. É interessante bloqueá-los para dificultar o download de arquivos .torrent através da rede, o que tem potencial para consumir muita banda.
warez - Páginas que hospedam programas piratas.
webradio, **webtv** - Rádios e TVs online. É interessante bloqueá-las junto com a categoria "audio-video", caso você queira reduzir o uso de banda da rede.

Com isso, você tem as duas listas à disposição e pode escolher qual delas utilizar, ou mesmo combinar seções de ambas para incrementar o filtro. O próximo passo é configurar o arquivo "**/etc/squid/squidGuard.conf**", especificando as listas a utilizar. Um exemplo básico de arquivo de configuração, usando apenas duas das seções da lista do MESD, seria:

```
# /etc/squid/squidGuard.conf

dbhome /var/lib/squidguard/db/blacklists
logdir /var/log/squid

dest porn {
domainlist porn/domains
urllist porn/urls
}

dest proxy {
domainlist proxy/domains
urllist proxy/urls
}

acl {
default {
pass !porn !proxy all
redirect http://www.gdhpress.com.br
}
}
```

As duas primeiras linhas indicam o diretório contendo as blacklists e o diretório onde serão armazenados os logs. No exemplo, estou usando as listas do MESD, daí o "/var/lib/squidguard/db/blacklists" e estou orientando o SquidGuard a salvar o log no mesmo diretório utilizado pelo Squid, gerando o arquivo "/var/log/squid/squidGuard.log".

Em seguida, temos duas ACLs, batizadas de "porn" e proxy", cada uma incluindo os dois arquivos da categoria correspondente dentro das listas. Para que fossem adicionadas mais seções, bastaria adicionar uma nova ACL para cada uma.

No final, a opção "pass" indica como as duas ACLs serão usadas. No exemplo, usei a linha "pass ! porn !proxy all", que indica que os acessos a páginas citadas nas listas devem ser bloqueados, mas o acesso a outras páginas é aceito.

Concluindo, usei a linha "redirect http://www.gdhpress.com.br", que faz com que todos os acessos bloqueados sejam redirecionados de forma transparente à URL especificada. Dessa forma, o usuário tentando acessar páginas impróprias é sutilmente direcionado a uma página com conteúdo mais saudável :). Você pode substituí-la pelo site da empresa, ou mesmo pela localização de uma página de aviso.

Temos aqui um segundo exemplo de configuração, bem mais incrementado, que usa um número bem maior de ACLs, combinando listas do MESD e do Shalla:

```
# /etc/squid/squidGuard.conf
dbhome /var/lib/squidguard/db
logdir /var/log/squid

dest ads {
domainlist blacklists/ads/domains
urllist blacklists/ads/urls
}
dest aggressive {
domainlist blacklists/aggressive/domains
urllist blacklists/aggressive/urls
}
dest audio-video {
domainlist blacklists/audio-video/domains
urllist blacklists/audio-video/urls
}
dest drugs {
domainlist blacklists/drugs/domains
urllist blacklists/drugs/urls
}
dest gambling {
domainlist blacklists/gambling/domains
urllist blacklists/gambling/urls
}
dest porn {
domainlist blacklists/porn/domains
urllist blacklists/porn/urls
}
dest proxy {
domainlist blacklists/proxy/domains
urllist blacklists/proxy/urls
}
dest redirector {
domainlist blacklists/redirector/domains
urllist blacklists/redirector/urls
}
dest spyware {
domainlist blacklists/spyware/domains
urllist blacklists/spyware/urls
}
dest violence {
domainlist blacklists/violence/domains
urllist blacklists/violence/urls
```

```
}
dest warez{
domainlist blacklists/warez/domains
urllist blacklists/warez/urls
}
dest porn2{
domainlist BL/porn/domains
urllist BL/porn/urls
}
dest socialnet{
domainlist BL/socialnet/domains
urllist BL/socialnet/urls
}
dest tracker{
domainlist BL/tracker/domains
urllist BL/tracker/urls
}

acl {
default {
pass !ads !aggressive !audio-video !drugs !gambling !porn !proxy
!redirector !spyware !violence !warez !porn2 !socialnet !tracker all
redirect http://www.gdhpress.com.br
}
}
```

Veja que nesse segundo exemplo, usei a linha "dbhome /var/lib/squidguard/db". Isso permite que você combine seções das duas listas, indicando o caminho até cada uma, a partir do diretório principal. Esta mesma idéia pode ser usada para combinar outras listas a que você tenha acesso. Basta colocar todas as listas dentro do diretório "/var/lib/squidguard/db" e incluir as ACLs correspondentes dentro da configuração.

Antes que possam ser efetivamente utilizadas, as listas precisam ser convertidas para o formato Berkeley DB, que permite um acesso muito mais rápido do que seria possível ao manipular diretamente os arquivos em texto. Para isso, use, depois de configurar o arquivo "/etc/squid/squidGuard.conf", o comando:

 # squidGuard -C all
(este comando deve ser executado novamente sempre que você incluir novas listas na configuração)

Embora não seja necessário em muitas configurações, é recomendável usar também o comando abaixo para ajustar as permissões de acesso aos arquivos, garantindo que o Squid tenha acesso a eles. O "proxy:proxy" indica o usuário e o grupo utilizados pelo Squid, que podem eventualmente ser diferentes, de acordo com a distribuição usada:

 # chown -R proxy:proxy /var/lib/squidguard/db/*

Os dois comandos a seguir complementam a configuração, fazendo com que todos os arquivos dentro da pasta sejam configurados com permissões 644 e as pastas com 755, que é a configuração correta. Isso previne o aparecimento de erros diversos relacionados a permissões incorretas para os arquivos:

 # find /var/lib/squidguard/db -type f | xargs chmod 644
 # find /var/lib/squidguard/db -type d | xargs chmod 755

Depois de gerar a configuração do SquidGuard, o próximo passo é alterar a configuração do Squid, para que ele seja utilizado. Para isso, edite o arquivo "**/etc/squid/squid.conf**", adicionando a linha:

```
redirect_program /usr/bin/squidGuard
```

Ela deve ser colocada depois das ACLs restritivas (destinadas a bloquear acessos, como no caso das ACLs para bloquear o acesso a uma lista de sites personalizados, ou em determinados horários), mas entretanto antes das regras finais, que permitem o acesso. Um exemplo de arquivo squid.conf completo seria:

```
# /etc/squid/squid.conf
http_port 3128 transparent
visible_hostname gdh

cache_mem 128 MB
maximum_object_size_in_memory 128 KB
maximum_object_size 512 MB
cache_dir ufs /var/spool/squid 4096 16 256
cache_access_log /var/log/squid/access.log

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 21 80 443 563 70 210 280 488 59 777 901 1025-65535
acl purge method PURGE
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

redirect_program /usr/bin/squidGuard

acl redelocal src 192.168.1.0/24
http_access allow localhost
http_access allow redelocal
http_access deny all
```

A posição da regra que ativa o SquidGuard é importante, pois se ela for colocada depois da regra "http_access allow redelocal" (ou similar), as requisições serão liberadas antes de passarem pelo SquidGuard, fazendo com que ele nunca seja usado.

Opcionalmente, você pode incluir também as duas linhas abaixo, logo após a linha que ativa o SquidGuard:

```
redirect_children 8
redirector_bypass on
```

A opção "redirect_children" ajusta o número de processos do SquidGuard que o servidor Squid manterá abertos. Aumentar o número ajuda a melhorar o desempenho do proxy em grandes redes, onde o servidor recebe um volume muito grande de requisições.

A opção "redirector_bypass on" faz com que o Squid continue funcionando mesmo que o SquidGuard trave ou deixe de funcionar por qualquer motivo. Usá-la tem seus prós e contras, já que pode ser preferível que o acesso pare completamente, até que você consiga solucionar o problema, do que permitir que o Squid funcione com os bloqueios desativados. Pense no caso de uma escola primária, por exemplo.

Depois de tudo terminado, reinicie o Squid para que a configuração entre em vigor:

 # /etc/init.d/squid restart

Se você estiver configurando um servidor de produção, com usuários acessando o proxy enquanto está configurando, use o comando abaixo para ativar a configuração sem derrubar os usuários conectados:

 # squid -k reconfigure

Com tudo pronto, verifique se o SquidGuard está mesmo ativo usando o comando abaixo:

 # tail /var/log/squid/squidGuard.log

Ele mostrará as mensagens de inicialização do SquidGuard. Se tudo estiver correto, as duas últimas linhas serão:

```
2008-06-14 09:16:02 [4521] squidGuard 1.2.0 started (1208175362.060)
2008-06-14 09:16:02 [4521] squidGuard ready for requests (1208175362.105)
```

Se algum erro impedir a inicialização do serviço, ele exibirá a mensagem de erro, permitindo que você localize o problema, como em:

```
2008-06-14 10:45:04 [4408] init domainlist /var/lib/squidguard/db/BL/socialnet/domains
2008-06-14 10:45:04 [4408] /var/lib/squidguard/db/BL/socialnet/domains: Permission denied
```

Nesse caso, temos um problema com as permissões de acesso de uma das listas especificadas na configuração, o que poderia ser resolvido usando os comandos para acertar as permissões que vimos no início do tópico.

Com o SquidGuard ativo, os acessos a páginas impróprias serão drasticamente reduzidos e você conserva a possibilidade de refinar o bloqueio, adicionando novos endereços manualmente. Não esqueça de atualizar os arquivos das blacklists periodicamente, já que elas são atualizadas de forma freqüente.

Com o tempo, é provável que você precise desbloquear algumas páginas manualmente (depois de verificar seu conteúdo), a pedido dos usuários. Nesse caso, você pode criar uma lista branca, autorizando o acesso aos sites manualmente inseridos nela.

Para isso, adicione uma nova ACL no arquivo squidGuard.conf, adicionando as seguintes linhas próximo ao início do arquivo:

```
dest white {
domainlist white/domains
```

```
urllist white/urls
}
```

No final do arquivo, ao especificar o uso das ACLs, inclua o parâmetro "white" (sem a exclamação) antes dos demais, como em:

```
acl {
default {
pass white !porn !proxy all
redirect http://www.gdhpress.com.br
}
}
```

Com isso, o conteúdo da ACL "white" será processado primeiro e o acesso às páginas especificadas no arquivo será liberado. Falta agora criar a pasta e os dois arquivos citados na configuração:

 # mkdir /var/lib/squidguard/db/white
 # touch /var/lib/squidguard/db/white/domains
 # touch /var/lib/squidguard/db/white/urls

Use o arquivo "domains" para incluir domínios que devem ser permitidos por completo, como em "gdhn.com.br" e o arquivo "urls" para incluir páginas ou seções isoladas, como em "gdhn.com.br/tutoriais/", sempre um por linha.

Depois de editar os arquivos, é necessário fazer com que o SquidGuard atualize a conversão das listas e reiniciar o Squid para que as alterações entrem em vigor, como em:

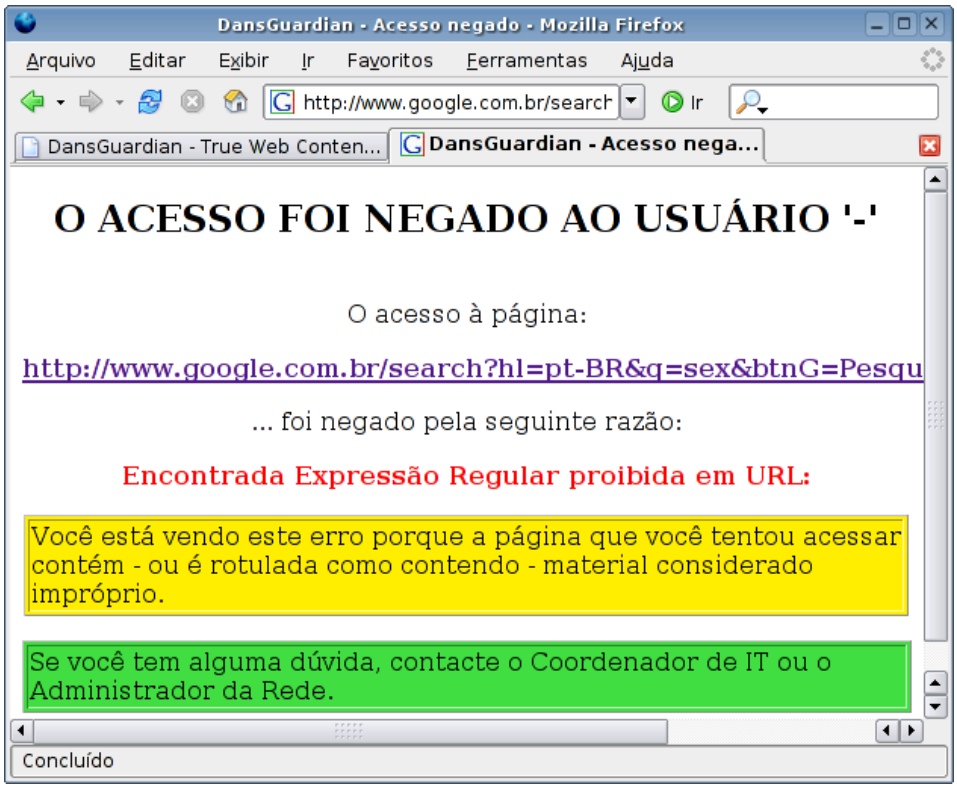
 # squidGuard -C all
 # squid -k reconfigure

Usando o DansGuardian

O DansGuardian é outra opção de filtro de conteúdo desenvolvido para trabalhar em conjunto com o Squid, filtrando conteúdo indesejado. A grande diferença entre ele e o SquidGuard é que o SquidGuard se limita a bloquear páginas contidas nas listas, enquanto o DansGuardian utiliza um filtro adaptativo, que avalia o conteúdo da página e decide se ela é uma página imprópria com base no conteúdo, utilizando um conjunto de regras adaptativas. Ele inclui um conjunto de regras prontas, que contém palavras, frases e tipos de arquivos freqüentemente usados em páginas impróprias, além de uma lista de páginas conhecidas e trabalha cruzando todas essas informações.

Se você quer um filtro de conteúdo que possa implantar rapidamente e manter ativo sem ter muito trabalho, o SquidGuard é a melhor opção, pois ele permite obter bons resultados mesmo com apenas uma configuração rápida. Se, por outro lado, você quer um filtro programável, que você possa adaptar e personalizar de forma a obter melhores resultados, então o DansGuardian pode ser uma boa opção.

No screenshot abaixo, por exemplo, temos uma pesquisa do Google por um termo indecoroso, que foi bloqueada por ter retornado um grande número de páginas impróprias. O domínio "google.com" naturalmente não foi bloqueado no filtro, mas mesmo assim o DansGuardian foi capaz de bloquear a página baseado no conteúdo:



Originalmente, o DansGuardian foi desenvolvido como um filtro de conteúdo para uso em escolas primárias e para quem tem crianças pequenas em casa, mas nada impede que ele seja usado também em outras situações.

Atualmente, o DansGuardian é um produto "semicomercial", que tem o código aberto e é gratuito para uso pessoal ou para qualquer fim não-comercial (pode ser usado em uma escola ou escritório,

por exemplo, desde que implementado internamente), mas é pago para uso comercial (quando você cobra pelo serviço de implantação, ou o fornece como parte de uma solução comercial). Você pode ver mais detalhes sobre a licença de uso no:
http://dansguardian.org/?page=copyright2

Ao instalar, comece verificando se já não existe um pacote disponível na distribuição que está usando. O DansGuardian é um pacote de uso muito comum; por isso, a maioria das distribuições o inclui nos CDs de instalação. No Debian, por exemplo, você pode instalá-lo com um:

 # apt-get install dansguardian

Você pode também encontrar pacotes para várias distribuições, junto com o tradicional pacote com código fonte no http://dansguardian.org/?page=download2.

Depois de instalar o pacote, inicie-o com o comando:

 # /etc/init.d/dansguardian start

ou:
 # dansguardian &

Para que o DansGuardian funcione, é preciso que o Squid esteja instalado e ativo. Ele trabalha sobre o Squid, implementando suas políticas de acesso, mas deixando que o próprio Squid faça o acesso à web, cache e autenticação.

O principal arquivo de configuração é o "**/etc/dansguardian/dansguardian.conf**". Ao editá-lo pela primeira vez, é importante verificar algumas opções:

 # UNCONFIGURED

Esta linha deve ficar comentada, indicando que o arquivo já foi configurado por você.

language = 'portuguese'

Esta opção configura a língua em que as mensagens de acesso bloqueado serão mostradas aos clientes.

loglocation = '/var/log/dansguardian/access.log'

Aqui vai a localização do arquivo de log do dansguardian, onde ficam armazenados os endereços das páginas cujo acesso foi bloqueado. Serve tanto para verificar a eficiência do filtro, quanto para identificar falsos-positivos, ou seja, páginas legítimas que estão sendo bloqueadas por engano. Essas exceções podem ser especificadas individualmente no arquivo "/etc/dansguardian/exceptionsitelist", que funciona como uma white list, contendo uma lista de páginas que sempre são permitidas, mesmo que sejam encontradas palavras proibidas dentro do texto.

filterport = 8080

A porta onde o DansGuardian fica ativo. Ele sempre deve utilizar uma porta diferente do Squid, pois são duas coisas separadas. O padrão é a porta 8080.

proxyip = 127.0.0.1

O endereço IP do servidor proxy que será usado. Por padrão ele vai utilizar uma cópia do Squid ativa na mesma máquina, mas é possível utilizar outro servidor Squid disponível na rede.

proxyport = 3128

A porta TCP onde o servidor Squid especificado na opção acima está ativo. Lembre-se de que, por padrão, o Squid usa a porta 3128.

A filtragem de páginas funciona em dois níveis. Ao receber a requisição do cliente, o DansGuardian verifica se o endereço a ser acessado está em uma das listas de domínios ou IPs proibidos. Caso esteja, o cliente recebe a mensagem de erro e o acesso sequer é feito, economizando banda.

Se não existir nenhum bloqueio relacionado ao domínio, a requisição é enviada ao Squid e o acesso é realizado. Ao receber os arquivos da página, o DansGuardian verifica o conteúdo em busca de expressões e palavras "ruins", freqüentemente encontradas em páginas indesejadas, e também palavras "boas", normalmente encontradas em páginas de bom conteúdo.

Cada palavra ruim soma um certo número de pontos. Por exemplo, a palavra "sexy" soma apenas 5 pontos, enquanto a expressão "sex orgies" soma 80 pontos. Palavras "boas", por outro lado, subtraem pontos, fazendo com que a página tenha uma possibilidade menor de ser bloqueada. A palavra "education" subtrai 20 pontos, enquanto "medical problem" subtrai 50. As listas com palavras boas e ruins, juntamente com o peso positivo ou negativo de cada uma, são armazenadas na pasta "**/etc/dansguardian/phraselist**".

No final, o site recebe uma nota, apelidada pelos desenvolvedores de "naughtynesslimit", ou "índice de sem-vergonhice", resultado da soma de todas as palavras boas e ruins. Você define um índice máximo a ser tolerado no arquivo "**/etc/dansguardian/dansguardianf1.conf"**, na opção:

naughtynesslimit = 160

Quanto mais baixo o número, mais severa é a censura, porém mais páginas boas acabam sendo bloqueadas por engano (falsos positivos). Os valores recomendados pelos desenvolvedores são "60" para crianças pequenas, "100" para pré-adolescentes e "160" para adolescentes. Para um público adulto, onde a principal preocupação seja não bloquear páginas úteis, mesmo que isso faça com que uma ou outra página inadequada passe pelo filtro de vez em quando, você pode arriscar "200" ou mesmo "240".

Como você pode notar dando uma olhada no conteúdo dos arquivos das listas de palavras, o DansGuardian vem configurado com listas em inglês, que deixam passar muitos sites nacionais. Você pode baixar um arquivo com listas em outras línguas, incluindo português no: http://dansguardian.org/downloads/grosvenor/languages.tar.gz

Para instalar, descompacte o arquivo "languages.tar.gz" e copie os arquivos de dentro da pasta "languages", que será criada para a pasta "**/etc/dansguardian/phraselist/**". Falta agora configurar o DansGuardian para utilizar os novos arquivos. Para isso, abra o arquivo "**/etc/dansguardian/weightedphraselist**" e adicione as linhas:

```
.Include</etc/squid/dansguardian/languages/weightedphraselist.pornsites.portuguese>
.Include</etc/squid/dansguardian/languages/weightedphraselist.pornwords.portuguese>
```

Antes de usar estas listas de palavras, verifique o conteúdo dos arquivos. As listas de palavras em português são excessivamente rigorosas, o que faz com que seja bloqueado o acesso a um número muito grande de sites "bons", mesmo ao usar um naughtynesslimit alto. Use-os com cautela.

Aparentemente, os arquivos disponíveis no site foram escritos por um estrangeiro, por isso, não se adaptam bem à nossa realidade. Se decidir corrigir os arquivos, não deixe de enviá-los para os mantenedores, para que sejam incluídos no pacote.

Note que na configuração são especificados todos os arquivos de palavras que são utilizados. Na pasta existem várias categorias diferentes e, em algumas situações, você pode querer desabilitar algumas delas, a fim de flexibilizar o filtro. Você pode adicionar novas palavras ou editar o peso de cada uma, editando diretamente os arquivos.

Concluindo, abra também o arquivo "**/etc/dansguardian/bannedphraselist**" e inclua a linha:

.Include</etc/squid/dansguardian/languages/bannedphraselist.portuguese>

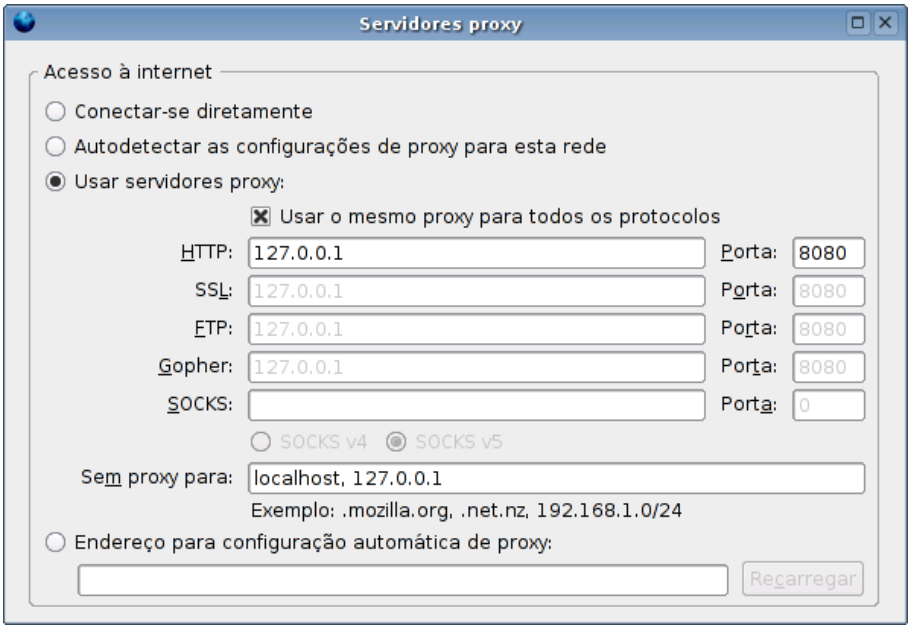
Lembre-se de que é necessário reiniciar o DansGuardian para que qualquer uma das alterações tenha efeito:

 # /etc/init.d/dansguardian restart

O DansGuardian pode ser usado tanto dentro da rede, quanto localmente. Além de ser utilizado em redes de todos os tamanhos, muita gente com crianças em casa se dá ao trabalho de instalá-lo avulso, no micro de casa.

Nas configurações do proxy (nos clientes), coloque o endereço IP do servidor (como, por exemplo, 192.168.1.1) e a porta do DansGuardian, definida no arquivo de configuração. Lembre-se de que, por padrão, ele usa a porta 8080.

No Firefox, a opção de configurar um proxy está em "Editar > Preferências > Proxy". Ou seja, com exceção da porta diferente, a configuração para usar o DansGuardian é a mesma que seria usada em conjunto com um proxy tradicional. Em casos onde ele é usado num micro doméstico, com o objetivo de servir como um simples filtro de conteúdo, você pode até mesmo rodá-lo localmente. Neste caso, use o endereço "127.0.0.1" como proxy:



Atualizando as blacklists

Além do filtro com base em palavras, o DansGuardian utiliza uma lista de sites proibidos, que sequer chegam a ser acessados. Por padrão, o DansGuardian vem com uma lista muito pequena e desatualizada, apenas como exemplo. Para efetivamente usar este recurso, é preciso baixar uma lista mais elaborada.

Você pode baixar uma lista longa e atualizada no http://urlblacklist.com/, o mesmo site que citei no tópico sobre o SquidGuard. As listas do UrlBlacklist são mais adequadas para uso no DansGuardian, pois incluem também listas de termos (que são usadas pelo DansGuardian para complementar o filtro estático baseado em URLs), mas ele possui a desvantagem de ser um serviço não-gratuito, onde você precisa assinar o serviço para ter acesso completo às listas.

O link completo para a versão mais recente é:
http://urlblacklist.com/cgi-bin/commercialdownload.pl?type=download&file=bigblacklist

Para instalar, basta descompactar o arquivo e mover o conteúdo para dentro da pasta "/etc/dansguardian/", substituindo a pasta "/etc/dansguardian/blacklists" existente:

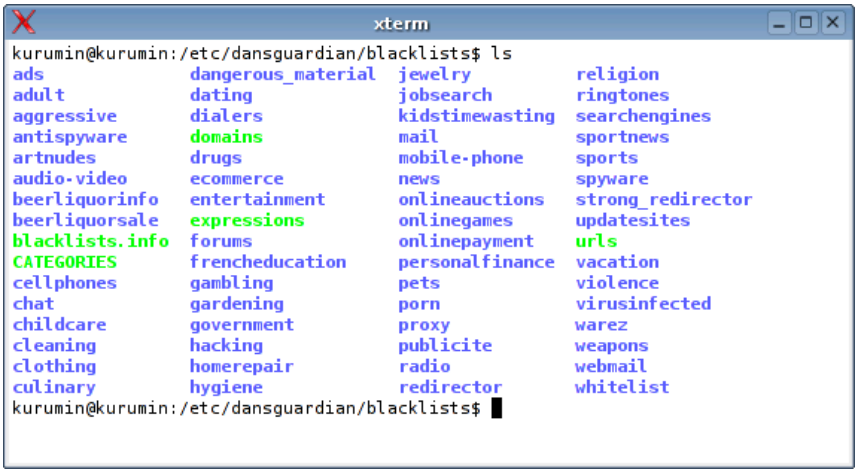
$ tar -zxvf bigblacklist.tar.gz
 # cp -a --reply=yes blacklists/ /etc/dansguardian/

Depois de instalar o arquivão completo, você pode usar o script de atualização, disponível no site, para baixar atualizações de forma automática. Baixe-o em:
http://urlblacklist.com/downloads/UpdateBL

Basta ativar a permissão de execução e executá-lo. Em algumas distribuições é preciso criar a pasta "/var/lib/lrpkg/", onde ele guarda os logs. Sem esta pasta, ele exibe um erro e não conclui a atualização.

 # mkdir -p /var/lib/lrpkg/
 # chmod +x UpdateBL
 # ./UpdateBL

O pacote inclui várias listas diferentes, separadas por assunto. Assim como na lista do Shalla's, as listas incluem muitos assuntos inocentes como, "cellphones", "sports" e "childcare" (saúde infantil). Ele não é uma "blacklist" no sentido estrito da palavra, mas sim um conjunto de listas que incluem também sites sobre conteúdos diversos. A idéia aqui é que você pode bloquear todos os assuntos desejados.



Dentro de cada uma das subpastas, você encontra três arquivos: domains (sites completamente bloqueados), expressions (palavras comumente encontradas em sites de conteúdo impróprio) e urls (páginas específicas, dentro de sites permitidos). Para ativar o uso das blacklists, edite os arquivos "**/etc/dansguardian/bannedsitelist**" e "**/etc/dansguardian/bannedurllist**", adicionando (ou descomentando) as linhas referentes às categorias que devem ser ativadas.

Para bloquear páginas de conteúdo adulto (adult), drogas (drugs), páginas pornográficas (porn) e warez, adicione (ou descomente) no arquivo "**/etc/dansguardian/bannedurllist"** as linhas:

.Include</etc/dansguardian/blacklists/adult/urls>
.Include</etc/dansguardian/blacklists/drugs/urls>
.Include</etc/dansguardian/blacklists/porn/urls>
.Include</etc/dansguardian/blacklists/warez/urls>

No arquivo "**/etc/dansguardian/bannedsitelist**" vão as linhas:

.Include</etc/dansguardian/blacklists/adult/domains>
.Include</etc/dansguardian/blacklists/drugs/domains>
.Include</etc/dansguardian/blacklists/porn/domains>
.Include</etc/dansguardian/blacklists/warez/domains>

Você pode usar também os arquivos com expressões proibidas, incluídos no pacote para reforçar a lista adicional, com os termos em português, que já ativamos anteriormente. Para isso, abra novamente o arquivo "**/etc/dansguardian/bannedphraselist**" e adicione as linhas:

.Include</etc/dansguardian/blacklists/adult/expressions>
.Include</etc/dansguardian/blacklists/drugs/expressions>
.Include</etc/dansguardian/blacklists/porn/expressions>
.Include</etc/dansguardian/blacklists/warez/expressions>

Faça o mesmo com outras categorias que quiser adicionar.

Proxy transparente com o DansGuardian

Como vimos até agora, o DansGuardian funciona como uma camada extra, uma espécie de "pedágio", por onde as requisições passam antes de chegarem ao Squid e por onde as respostas passam antes de serem enviadas ao cliente.

Normalmente, os clientes precisam ser configurados manualmente para utilizar o DansGuardian como proxy, acessando-o através da porta 8080. Isso traz de volta o problema de configurar manualmente cada um dos micros e evitar que os usuários removam a configuração para acessar diretamente caso você mantenha o compartilhamento via NAT ativo em adição ao proxy.

Contudo, é possível configurar o DansGuardian para trabalhar como proxy transparente, da mesma forma que fizemos anteriormente com o Squid. Neste caso, o firewall redireciona as requisições recebidas na porta 80 para o DansGuardian e ele as repassa para o Squid, que finalmente faz o acesso. Os clientes precisam apenas ser configurados para acessar a internet usando o servidor onde estão instalados o Squid e DansGuardian como gateway.

Para isso, comece configurando o Squid para trabalhar em modo transparente, adicionando a opção "transparent" na linha "http_port" do squid.conf:

http_port 3128 transparent

Depois vêm as regras de firewall para habilitar o compartilhamento da conexão e direcionar as requisições recebidas na porta 80 para a porta usada pelo DansGuardian. Novamente, é a mesma configuração usada para fazer um proxy transparente no Squid, mudando apenas a porta. Lembre-se de que o "eth0" deve ser substituído pela interface ligada na rede local e o "eth1" pela interface ligada à Internet:

```
modprobe iptable_nat
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT \
--to-port 8080
iptables -A INPUT -m tcp -p tcp -s ! 127.0.0.1 --dport 3128 -j DROP
```

A última regra bloqueia a porta 3128 usada pelo Squid, para impedir que algum espertinho configure o navegador para acessar diretamente através do Squid, sem passar pelo DansGuardian. A única exceção é o endereço 127.0.0.1, ou seja, o próprio servidor. Lembre-se de colocar estes comandos no arquivo "/etc/rc.local" (ou no seu script de compartilhamento da conexão) para não precisar ficar digitando tudo a cada boot.

Obtendo um endereço fixo, usando um DNS dinâmico

Tudo o que vimos ate agora, ou seja, compartilhar a conexão, configurar um servidor proxy e assim por diante soluciona o problema do acesso à web a partir dos micros da rede local, permitindo que uma única conexão via ADSL, cabo, rádio ou outra modalidade de acesso doméstico seja compartilhada entre vários micros.

Sobra então resolver o problema do acesso externo, ou seja, permitir que o servidor da rede seja acessado remotamente, de forma que você possa prestar suporte remoto ou possa configurar uma VPN, interligando duas redes distantes, por exemplo. Uma conexão doméstica oferece três deficiências nesse sentido:

* O endereço IP é dinâmico, mudando periódicamente ou a cada vez que o modem é reiniciado.
* A conexão dificilmente é 100% estável, caindo esporadicamente.
* Algumas portas, como a 21, a 25 e a 80 são fechadas pelas operadoras, de forma a dificultar a criação de servidores domésticos.

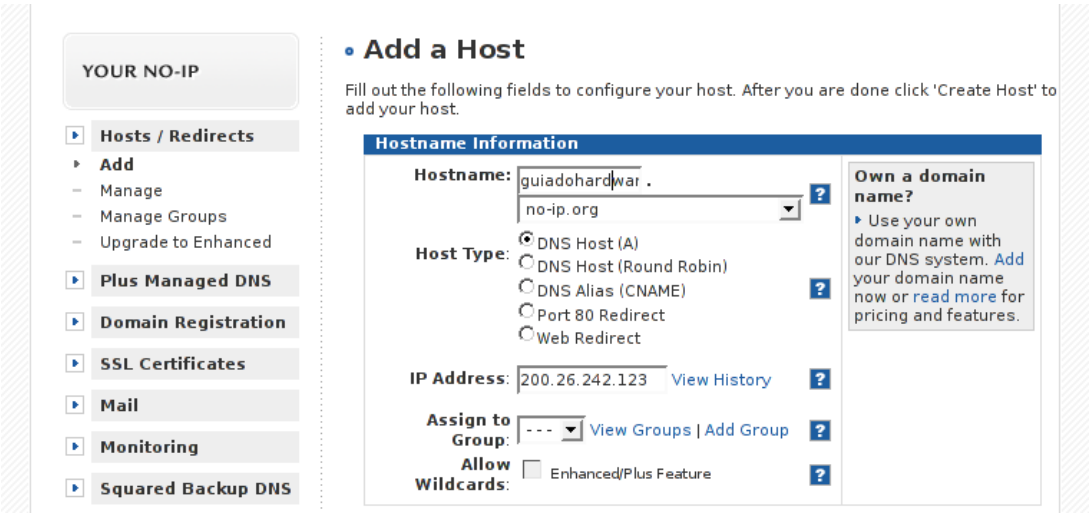
Não podemos fazer nada com relação à estabilidade da conexão, mas podemos alterar as portas utilizadas pelos servidores (o Apache pode ser configurado para utilizar a porta 8080 em vez da 80, por exemplo) e podemos solucionar o problema do IP dinâmico utilizando um serviço de DNS dinâmico (DDNS).

Os serviços de DNS dinâmico trabalham de uma forma bastante simples, onde um cliente instalado no seu servidor (ou em qualquer outra máquina da rede, acessando através da conexão compartilhada por ele) envia informações sobre o endereço IP corrente para os servidores do serviço, o que permite a eles manterem um subdomínio no estilo "meu-nome.no-ip.org" ou "minhaempresa.dyndns.com" apontando para seu endereço IP corrente.

As contas são gratuitas, de forma que você pode tranquilamente criar um domínio virtual para cada rede que você administra, permitindo que, desde que a conexão esteja funcionando, você possa resolver qualquer problema sem levantar da cadeira. Usar um domínio virtual permite também que você crie VPNs, interligando duas ou mais redes com IP dinâmico (como veremos em detalhes no capítulo sobre o OpenVPN) ou até mesmo permitir que os usuários da rede acessem suas máquinas e rodem aplicativos remotamente (como veremos nos capítulos sobre o SSH e acesso remoto).

É possível também usar um DNS dinâmico também para hospedar sites, disponibilizar arquivos via FTP e assim por diante (é necessário apenas utilizar portas alternativas para os servidores). O problema é que a baixa taxa de upload e a relativa instabilidade das conexões domésticas tornarão o acesso dos visitantes muito ruim, sem falar que o uso constante da banda tornará seu acesso muito mais lento. Você pode muito bem configurar um servidor para fins de teste, aplicando as dicas que veremos ao longo do livro, mas para aplicações mais sérias, é necessário ter um servidor dedicado, ou pelo menos um plano de shared hosting.

Os dois serviços de DNS Dinâmico mais usados (ambos gratuitos, com opções de serviços pagos), são o http://www.no-ip.com e o http://www.dyndns.com, que aprenderemos a configurar aqui.



Cadastro de um domínio virtual no no-ip.com

Em ambos os casos, basta fazer um cadastro gratuito para criar sua conta e poder cadastrar os domínios:

http://www.no-ip.com/newUser.php
https://www.dyndns.com/account/create.html

Fica faltando então a parte mais importante que é a instalação do cliente. Para o **No-IP**, você pode utilizar o próprio cliente Linux disponível no:
http://www.no-ip.com/downloads.php?page=linux

Comece descompactando o arquivo. Dentro dele, existe uma pasta chamada "binaries", com o arquivo "noip2-Linux". Este é o executável que faz a atualização do IP. Para usá-lo, copie-o para a pasta "/usr/local/bin", como em:

 # tar -zxvf noip-duc-linux.tar.gz
 # cd noip-2.1.1/binaries/
 # cp -a noip2-Linux /usr/local/bin/

O próximo passo é executar o "noip2-Linux", usando a opção "-C -c" (create config), que cria o arquivo de configuração. Você pode indicar onde o arquivo será criado, basta indicá-lo no comando. Nesta etapa ele pedirá o login de usuário e o domínio registrado no site, como em:

 # noip2-Linux -C -c /etc/noip.conf

```
Auto configuration for Linux client of no-ip.com.
Please enter the login/email string for no-ip.com : meu@email.com
Please enter the password for user 'meu@email.com ' ********
Only one host [meunome.no-ip.org] is registered to this account.
It will be used.
Please enter an update interval:[30]
Do you wish to run something at successful update?[N] (y/N) N
New configuration file '/etc/no-ip.conf' created.
```

Com o arquivo de configuração criado, inicie o noip2-Linux usando o comando abaixo. Inclua o comando em uma dos scripts de inicialização do sistema, como o "**/etc/rc.d/rc.local**", para que ele seja executado durante o boot. Não esqueça de adicionar o "&" no final do comando, ele faz o programa rodar em background. Sem ele, o comando bloqueia o terminal, paralisando a

inicialização do sistema. Note que agora usamos apenas o segundo "c", que indica que ele deve usar o arquivo de configuração anteriormente criado:

 # noip2-Linux -c /etc/noip.conf &

Nas distribuições derivadas do Debian, existe a opção de instalar o pacote disponível via apt-get:

 # apt-get install no-ip

Ao ser instalado, ele cria automaticamente o script "/etc/init.d/no-ip", que se encarrega de ativar o programa durante o boot. Para que ele funcione, fica faltando apenas criar o arquivo de configuração, usando o comando:

 # no-ip -C -c /etc/no-ip.conf

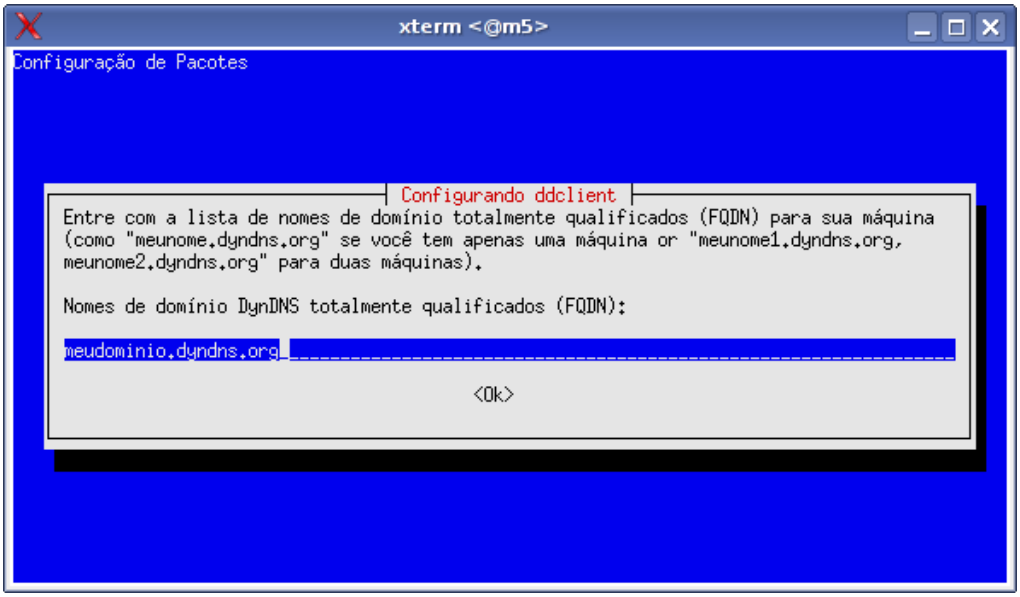
Para que a configuração entre o vigor, reinicie o serviço, usando:

 # /etc/init.d/no-ip restart

Para o **DynDNS**, utilizamos o ddclient, um pequeno aplicativo escrito em perl, que está disponível nas principais distribuições. No caso das derivadas do Debian, a instalação é mais simples, pois ele está disponível diretamente nos repositórios principais, via apt-get:

 # apt-get install ddclient

O script de instalação incluído no pacote faz uma série de perguntas, incluindo o domínio virtual registrado, seu login e senha no serviço, a interface do servidor ligada à internet e o tempo de intervalo entre as atualizações:



No final, ele pergunta se você deseja executar o ddclient como um serviço. Responda que sim para que ele fique ativo continuamente e seja iniciado junto com o sistema.

A configuração é salva no arquivo "**/etc/ddclient.conf**", que você pode editar manualmente caso precise fazer alterações posteriormente. O arquivo inclui basicamente as mesmas opções feitas pelo wizard, com as respostas salvas em texto puro.

No caso do **CentOS** e do **Fedora**, o pacote do ddclient está disponível no repositório do RPMForge. Você encontra instruções de como ativar o repositório no yum no https://rpmrepo.org/RPMforge/Using.

Depois de adicionar o repositório, a instalação é feita da forma tradicional:

 # yum install ddclient

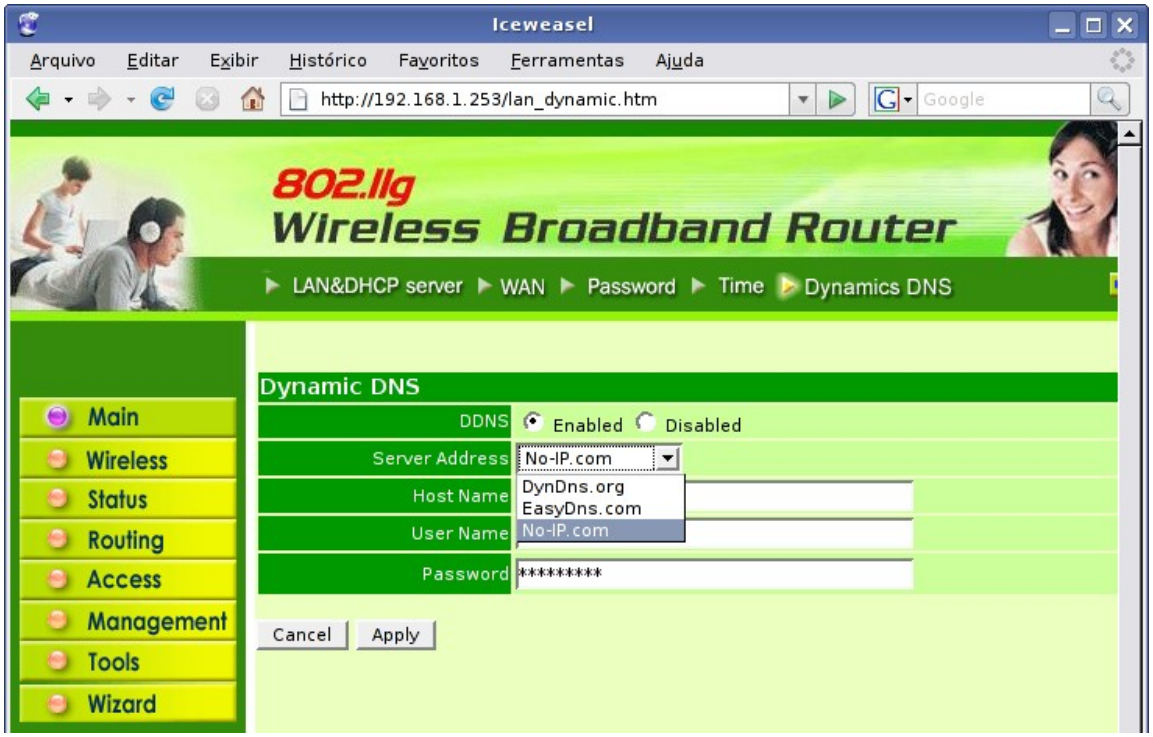
O pacote RPM não inclui o wizard, por isso a configuração precisa ser feita manualmente, editando o arquivo "**/etc/ddclient/ddclient.conf**". Aqui vai um modelo pré-configurado, onde você precisa apenas especificar seu login, senha e o domínio registrado:

daemon=600
cache=/tmp/ddclient.cache
pid=/var/run/ddclient.pid
use=web, web=checkip.dyndns.com/, web-skip='IP Address'
login=**seulogin**
password=**suasenha**
protocol=dyndns2
server=members.dyndns.org
seudominio.dyndns.org

Depois de terminar, reinicie o serviço para que as alterações entrem em vigor:

 # service ddclient restart

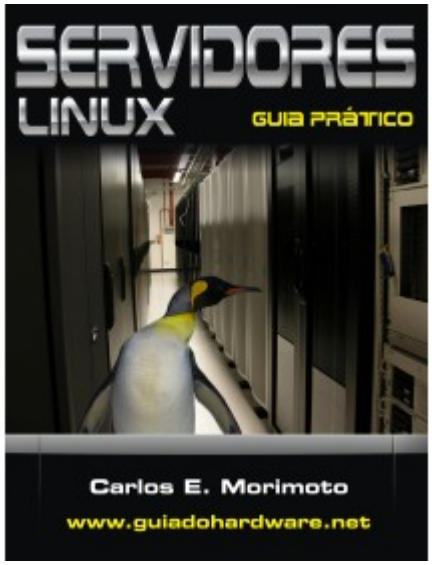
Se você usa um modem ADSL ou um roteador wireless para compartilhar a conexão, procure pela opção de usar um DNS dinâmico (Dynamic DNS) dentro da configuração. A maioria dos modelos atuais incluem clientes para os serviços mais populares:



Com isso, o próprio roteador pode se encarregar da atualização, sem que você precise instalar softwares adicionais no servidor da rede. Outros serviços populares, que você pode testar são o http://www.da.ru, o http://www.dtdns.com/ e o http://dns2go.com.

Fonte: Capitulo 02 do livro Servidores Linux, Guia Prático

Você está lendo um tópico de demonstração do livro Servidores Linux, Guia Prático:



Autor: Carlos E. Morimoto
Páginas previstas: **720**
Formato: 23 x 16 cm
Editora: GDH Press e Sul Editores
Lançamento previsto para: 13 de Agosto de 2008

» **Pré-venda: R$ 63,00** com envio grátis para todo o Brasil