1.0 Introdução

Os proxies são principalmente usados para permitir acesso à Web através de um firewall (fig. 1). Um proxy é um servidor HTTP especial que tipicamente roda em uma máquina firewall. O proxy espera por uma requisição de dentro do firewall, a repassa para o servidor remoto do outro lado do firewall, lê a resposta e envia de volta ao cliente.

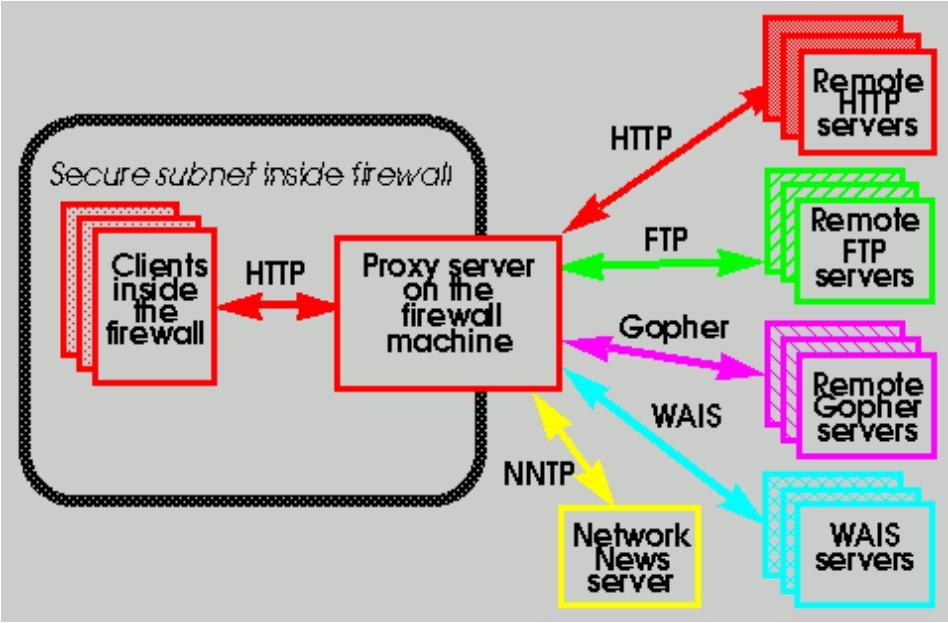

Figura 1: Visão geral de um proxy
O proxy está rodando ou em um servidor firewall ou qualquer outro servidor interno que tenha acesso total a internet - ou em uma máquina dentro do firewall fazendo conexões com o mundo exterior através de SOCKS ou qualquer outro software firewall.

Normalmente, o mesmo proxy é usado por todos os clientes em uma subrede. Isto torna possível para ele fazer caching eficiente de todos os documentos requisitados.

A habilidade que o proxy tem no uso do cache, o torna atrativo para aqueles que não estão dentro do firewall. Configurar um servidor proxy é fácil e os mais populares programas clientes Web já tem suporte a essa ferramenta. Sendo assim, torna-se simples a tarefa de configurar um grupo de trabalho inteiro para usar o serviço de cache do proxy. Isto reduz os custos com tráfego de rede porque muitos documentos que são requisitados são lidos do cache local.

A metodologia atual é baseada em um código de gateway escrito por Tim Berners-Lee como parte do libwww (WWW commom Library). Kevin Altis, Ari Luotonen e Lou Montulli foram os principais contribuidores para a padronização

do proxy. Lou Montulli, autor de Lynx, fez as primeiras mudanças no libwww em colaboração com Kevin Altis. Ari Luotonen mantém o CERN httpd.

1.1 Porque um nível de aplicação proxy?

Um nível de aplicação proxy faz um firewall seguramente permeável para os usuários na organização sem criar um furo na segurança onde hackers poderiam entrar na rede da organização.

Para clientes Web, as modificações necessárias para suportar um nível de aplicação proxy são menores (leva-se apenas 5 minutos para adicionar suporte proxy para o Emacs Web Browser).

Não há necessidade de compilar versões especiais de clientes Web com bibliotecas firewall, o cliente "out-of-the-box" pode ser configurado para ser um cliente proxy. Em outras palavras, quando se usa proxy não necessitamos customizar cada cliente para suportar um tipo ou método especial de firewall: o proxy, em si, é um método padrão para acessar firewalls.

Usuários não têm que ter clientes FTP, Gopher e WAIS separados (muito menos modificados) para acessar um firewall - um simples cliente Web com um servidor proxy trata todos esse casos. O proxy também padroniza a aparência de clientes Gopher e FTP.

O proxy permite que os programadores esqueçam as dezenas de milhares de linhas de código necessárias para suportar cada protocolo e se concentrem em coisas mais importantes - é possível ter clientes "peso-leve" que somente compreendam HTTP (nenhum suporte nativo aos protocolos FTP, Gopher, etc) - outros protocolos são manuseados transparentemente pelo proxy. Usando HTTP entre o cliente e o proxy, nenhuma funcionalidade é perdida, pois FTP, Gopher e outros protocolos Web são bem mapeados para o HTTP.

Clientes sem DNS (Domain Name Service) também podem usar a Web. O endereço IP do proxy é a única informação realmente necessária. Organizações usando endereços, por exemplo, classe A (como 10.*.*.*), em suas redes particulares podem ainda acessar a internet contanto que o proxy seja visível tanto para a rede particular como para a Internet.

Proxy permite um alto nível de log das transações de clientes, incluindo endereço IP, data e hora, URL, contagem de bytes e código de sucesso. Qualquer campo (seja de meta-informação ou seja comum) em uma transação HTTP é um candidato para log. Isto não é possível com log no nível IP ou TCP.

Também é possível fazer a filtragem de transações de clientes no nível do protocolo de aplicação. O proxy pode controlar o acesso a serviços por métodos individuais, servidores e domínios, etc.

Outra feature interessante do proxy é a cache. O uso de cache é mais efetivo no servidor proxy do que em cada cliente. Isto salva espaço em disco, desde que somente uma cópia é guardada, como também permite um uso de "cache inteligente", onde os documentos freqüentemente referenciados por muitos clientes são guardados por um periodo mais longo de tempo pelo cache manager.

O uso de cache também torna possível acessar algumas páginas mesmo que servidores estejam fora do ar. Essa facilidade torna o serviço melhor, visto que recursos remotos como um site FTP ocupado que são freqüentemente inacessíveis remotamente podem ser agora acessíveis através do cache local. Pode-se citar uma infinidade de usos que podemos fazer com o cache: fazer uma demonstração de algum lugar com uma baixa velocidade de conexão, ler documentos com a máquina não-conectada (obviamente após colocar todos documentos no cache local), etc.

Em geral, autores de clientes Web não tem razão para usar versões de firewalls em seus códigos. O proxy é mais simples para configurar do que SOCKS e trabalha em todas as plataformas, não somente UNIX.

2.0 Detalhes Técnicos

Quando uma requisição HTTP normal é feita por um cliente, o servidor pega somente o path e a "porção chave" da URL requisitada (Fig. 2); outras partes, como o especificador de protocolo "http:" e o nome do servidor são obviamente claros para o servidor HTTP remoto. O path requisitado especifica um documento ou um script CGI no sistema de arquivos local do servidor; ou ainda algum outro recurso disponível daquele servidor.

Quando um usuário entra:

http://mycompany.com/information/ProxyDetails.html

O browser converte para:

GET /information/ProxyDetails.html

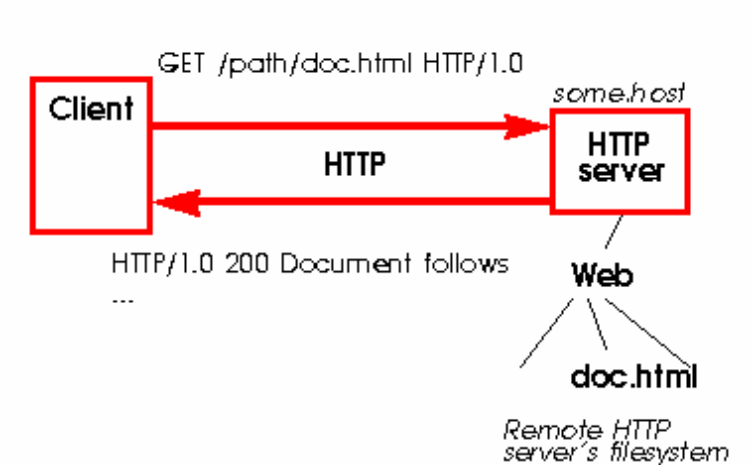

Figura 2: Uma transação HTTP normal
O cliente faz a requisição ao servidor HTTP especificando apenas o recurso relativo àquele servidor (nenhum protocolo ou nome de servidor é colocado na URL).

Quando um cliente envia uma mensagem para um servidor proxy a situação é um pouco diferente. O cliente sempre usa HTTP para transações com o proxy, mesmo quando acessa um recurso oferecido por um servidor remoto usando outro protocolo, como Gopher e FTP.

Entretando, ao invés de especificar somente o pathname e possíveis palavras que complementariam a procura para o proxy (como ocorre em uma requisição normal), todo a URL é especificada (fig.3 e 4). Desta forma o proxy tem todas as informações necessárias para fazer a requisição para o servidor remoto especificado na URL.

Nada melhor que um exemplo para clarear as coisas: se o usuário digitasse a seguinte URL:

http://mycompany.com/information/ProxyDetails.html

O browser, sabendo da existência do proxy, converteria para a seguinte requisição:

GET http://mycompany.com/information/ProxyDetails.html

O browser conecta-se então ao servidor e o proxy providencia a conexão com a Internet. Nesse caso, o proxy converteria a requisição para:

GET /information/ProxyDetails.html

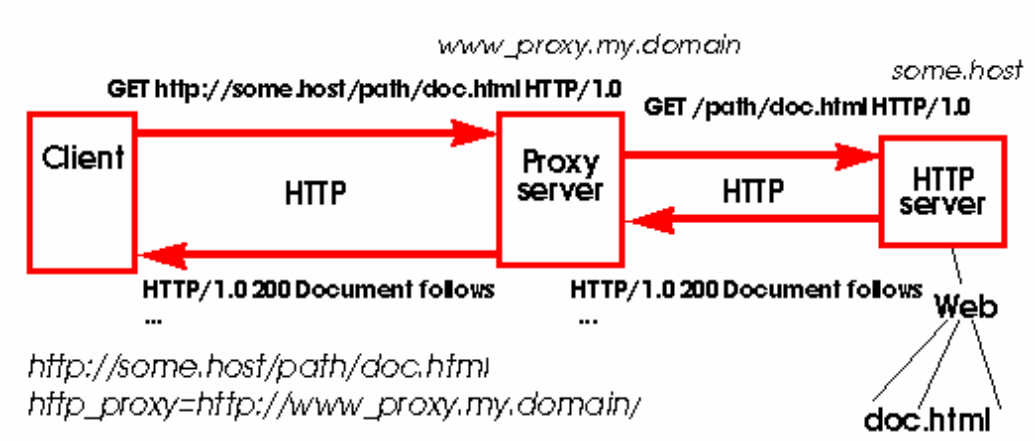

Figura 3: Uma trasação HTTP com proxy.
O cliente faz uma requisição ao proxy usando HTTP mas especificando toda a URL; o proxy se conecta ao servidor remoto e pede o recurso relativo àquele servidor sem especificar protocolo ou o nome do servidor na URL

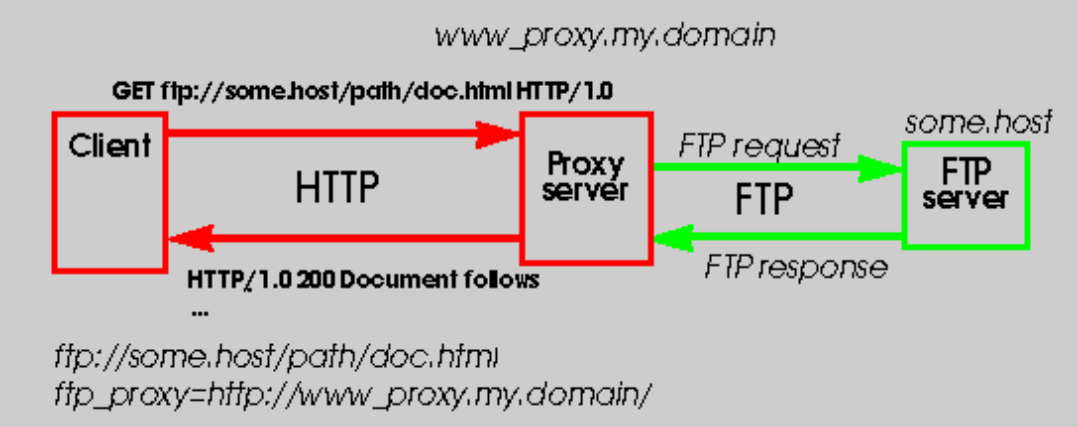

Figura 4: Trasação FTP com proxy.
O cliente faz a requisição ao proxy usando HTTP (embora o recurso seja um FTP). O proxy analisa a URL recebida e percebe que deve abrir uma conexão FTP. O resultado (o arquivo, no caso) é enviado para o cliente usando HTTP

Deste ponto o proxy age como um cliente para conseguir o documento: ele chama o mesmo protocolo libwww que um cliente deveria chamar para que o documento fosse obtido. Entretando, a "apresentação" do documento no proxy é através de HTTP, independente do protocolo que foi usado para consegui-lo. Um comando list, por exemplo, é retornado como um documento HTML.