SEGURANÇA E ACELERAÇÃO DE INTERNET: UTILIZAÇÃO DE PROXY SERVERS PARA MANUTENÇÃO DE WEB CACHES

Elvis Pontes
Sérgio Hirata
Solli Honório

RESUMO

A necessidade impreterível do mercado corporativo, do meio acadêmico e de usuários domésticos em acessar e utilizar diversas aplicações da Internet implica em um aumento gradual da quantidade de dados a serem transmitidos, portanto, um aumento do tráfego em uma rede já saturada. Entretanto, este tráfego é composto, em sua maioria, por diversas cópias da mesma informação, acessados por usuários distintos, mas em muitos casos próximos entre si. No diagrama tradicional, a cada requisição de usuário, uma nova conexão é feita, mesmo que o recurso já tenha sido solicitado minutos atrás. O objetivo final dos servidores *proxy-cache* é diminuir a latência para o uso de aplicações na Internet, tornando possível o uso ou de dados previamente consultados, por um ou diversos usuários. O uso de servidores *proxy-cache* permite que requisições já feitas anteriormente não necessitem de novas conexões com a Internet, otimizando o uso de banda e tornando mais rápido o acesso à rede. Pelo simples fato de não ser necessária a busca de objetos (imagens e arquivos) sempre que solicitados, já que estes foram previamente armazenados nos servidores *proxy-cache*, o consumo de recursos e o tráfego na Internet tendem a diminuir, futuros acessos terão o tempo de busca otimizados.

Palavras-chaves: Aceleração de Internet, *Proxy, Proxy* reverso, *Web Cache*.

1. INTRODUÇÃO

A popularidade da *Web*[1] está causando sérios problemas de desempenho nos acessos à Internet, de forma que a redução da latência tem se tornado uma questão importante. O número de usuários que se conectam cresce de forma muito acentuada, sendo que mais de 2/3 do tráfego Internet atual é gerado pela *Web*.

A latência para se recuperar um documento *Web* depende de vários fatores, tais como: largura de banda, tempo de propagação, velocidade dos computadores cliente e servidor, etc.

No esforço de minimizar as conseqüências deste crescimento, alguns métodos podem ser adotados. Pode-se fazer uma atualização dos recursos: um servidor mais rápido, *switch*[2], aumento da banda. Entretanto, isto além de não ser economicamente viável, pode não resolver o problema, uma vez que são numerosos os fatores que envolvem uma única transação *Web*. Alternativas como *cache* de WWW[3], espelhamento de arquivos e outros têm sido adotados para resolver os gargalos existentes na rede.

A iniciativa de se implantar um sistema de *cache* WWW que armazene localmente objetos (páginas HTML[4], imagens e arquivos) da Internet, pode melhorar a qualidade do serviço fornecido aos usuários. Servidores *proxy* ajudam a diminuir significantemente o tempo médio de acesso a páginas e a transferência de arquivos, porque muitos deles são requisitados mais de uma vez e, exceto na primeira, as requisições são atendidas localmente.

As vantagens do uso de servidores *proxy-cache* são evidentes, visto que existem muitos recursos na Internet que são utilizados por diversos usuários. No esquema tradicional, a cada requisição de usuário, uma nova conexão é feita, mesmo que o recurso já tenha sido solicitado minutos atrás. O uso de servidores *proxy-cache* permite que requisições já feitas anteriormente não necessitem de

---

[1] *Web*: "a Web" ou "WWW" para encurtar - ("teia do tamanho do mundo", traduzindo literalmente) é uma rede de computadores na Internet que fornece informação em forma de hipertexto.

[2] *Switch*: Um *switch*, que em gíria aportuguesada foi traduzido para comutador, é um dispositivo utilizado em redes de computadores para reencaminhar tramas (dados) entre os diversos nós.

[3] WWW: Web

[4] HTML: A sigla HTML deriva da expressão inglesa *HyperText Markup Language*. Trata-se de uma linguagem de marcação utilizada para produzir páginas na Internet.

novas conexões com a Internet, otimizando o uso de banda e tornando mais rápido o acesso à rede.

Numa tentativa de resolver o problema de implementação de um serviço desse tipo, diversas formas de *proxy* foram criadas: *proxy* transparente, *proxy* reverso, *web proxy*,

O *Proxy*/*Cache* colabora em três dos maiores desafios dos clientes:

- Aceleração de aplicações e conteúdo *Web*: O *Proxy*/*cache* reduz a latência, utilização de banda e sobrecarga nos servidores, aumentando a distribuição do conteúdo *Web* e aplicações baseadas em *web* como sistemas de CRM[5] (*Customer Relationship Management*) e ERP[6] (*Enterprise Resource Planning*).

- Segurança na Internet: pode-se implementar processos de segurança, incluindo proxy, caching, controle de acesso, filtragem de conteúdo, anti-virus web, scanning SSL (Secure Sockets Layer), bloqueio de Instant message (msn messenger, AOL Instant Messenger) e P2P (peer to peer conection), antispam e relatórios

- Distribuição de vídeo: O *Proxy*/*cache* melhora a qualidade dos treinamentos *online*, vídeos executivos e serviços de vídeo sob demanda.

Por isso, é de suma importância que as organizações repensem o modo como seu acesso à *Web* está sendo utilizado e como ele poderá a vir a influenciar no desempenho de sua rede.

Este artigo trata desses tipos de solução, abordando protocolos ICP (*Internet Cache Protocol*), HTCP (*Hypertext Control Protocol*), WCCP (*Web Cache Control Protocol*), CARP (*Cache Array Routinng Protocol*), que são protocolos proprietários ou baseados em RFCs (*Request for Coments*) que, em implementação conjunta a servidores *proxy-cache*, constituem em uma solução muito interessante para *proxy* transparente, como será exibido no decorrer do artigo.

[5] CRM: *software* integrado para gerenciamento da relação com o cliente
[6] ERP: *software* integrado para Planejamento de Recursos Empresariais

2. *PROXY-CACHE*

O servidor *proxy* permite aos clientes existentes na rede a acessarem indiretamente outros serviços de rede. Um cliente conecta-se no servidor *proxy* e solicita uma conexão, arquivo e outros recursos disponíveis em outro servidor da rede. O *proxy* entrega o objeto buscando em um servidor específico ou servindo de um *cache*.

2.1. Hierarquia de *Cache*

A conexão dos servidores de *cache* de forma hierárquica permite um melhor aproveitamento dos recursos de *hardware* e de rede, pois desse modo pode-se ativer compartilhamento entre os mesmos.

A hierarquia baseia-se na definição de servidores pais e filhos (ou vizinhos). Os servidores pais são os que ficam mais próximos do objeto original solicitado. Quando um servidor filho solicita um objeto ao pai, este, caso possua o objeto, simplesmente o encaminha ao filho solicitante. Caso contrário, o servidor pai tenta recuperar o objeto em sua origem, depois o grava em seu disco e em seguida o encaminha ao filho solicitante.

No caso de servidores vizinhos, quando um irmão solicita a outro um objeto, podem ocorrer duas situações: o vizinho possui o objeto em questão e o encaminha ao irmão solicitante; ou o vizinho não tem o objeto, então a resposta simplesmente será a negativa a solicitação.

Outra hipótese de configuração de hierarquia é a existência de vários irmãos e pais para um servidor. Neste caso opta-se pela solicitação ao servidor mais próximo.

3. PRINCIPAIS TIPOS DE *PROXY*

a. Proxy Web

A aplicação mais comum é o *web proxy cache*. Isto determina o cacheamento de paginas *web* e arquivos disponíveis em servidores *web* remotos, permitindo o acesso mais rápido e confiável de clientes da rede local.

Quando recebe uma requisição para um recurso *web*, um *proxy cache* procura o objeto está armazenado localmente, se encontrar, o objeto é entregue imediatamente; caso contrário, o objeto é buscado na internet entregue ao usuário e grava uma cópia localmente. O *cache* normalmente utiliza um algoritmo de expiração para remover os documentos do cache, baseados no tempo eu está armazenado, tamanho e histórico de acesso. Alguns dos algoritmos utilizados são:

- *Least Recently Used* (LRU) — menos utilizado recentemente — Mantém o *cache* dos objetos referenciados recentemente, remove do *cache* o objeto referenciado há mais tempo. Problema: não leva em conta o tamanho do objeto – vários objetos pequenos podem substituir um objeto grande
- *Least Frequently Used* (LFU) — menos frequentemente utilizado. Remove o objeto menos popular, porém não considera tempo do último acesso. Considera o número de acessos. Problema: o *cache* pode ficar com objetos muito acessados e velhos
- *First in First out* (FIFO) - Objetos são removidos na mesma ordem que entram. Não é muito usado em servidores *cache.*
- *Size* - Baseado no tamanho do objeto; Substitui primeiro objetos com grande tamanho. Problema: o *cache* pode ficar preenchido com objetos antigos.
- *Greed Dual Size* – GDS. Atribui valores baseados no custo de um hit para os objetos armazenados no *cache*. O custo pode ser latência ou pacotes transmitidos pela rede. São mantidos em *cache* objetos menores, referenciados mais vezes.

b. *Proxy* transparente

Muitas empresas e escolas utilizam o *proxy* para reforçar as políticas de uso da rede ou incorporar serviços de segurança e cacheamento. A simples introdução de um servidor *proxy* pode não resolver o problema, visto que os usuários podem esquivar-se de utilizar o servidor *proxy* retornando a configuração original do *browser*, exceto no caso do *proxy* também efetuar o serviço de NAT (*Net Address Translation*).

Nestes casos, deve-se recorrer ao uso de um *proxy* transparente ou *transproxy*, que combina o NAT com o servidor *proxy* sem a necessidade de

configurações adicionais nas estações dos usuários finais. A RFC 3040 define este tipo como *intercepting proxy*.

Dentre as soluções de *proxy* transparente, pode-se destacar o protocolo WCCP (*Web Cache Cordination Protocol*) implementado pelos roteadores da CISCO[7]. As grandes corporações sempre possuem um roteador para servir de *front end*[8] para a Internet e este seria apenas mais um serviço a ser implementado. O WCCP se mostra uma solução interessante, visto que ele permite redundância e balanceamento de carga de servidores *proxy*, e, no caso de falha em todos os servidores, permite a saída direta para a Internet.

c. *Proxy* reverso

Um *proxy* reverso é um servidor *proxy* instalado na rede próxima de um ou mais servidores *web*. Todo o tráfego da internet e com destino de um dos servidores *web* é recebido pelo servidor *proxy*. Existem algumas razões para a instalação de um servidor *proxy*:

- Segurança: O servidor *proxy* é uma camada adicional de defesa;
- Aceleração de criptografia/SSL: Quando os sites seguros da internet são criados, a criptografia SSL, frequentemente não é feita pelo servidor *Web*, mas por um *proxy* reverso que está equipado com um hardware de aceleração de SSL.
- Distribuição de conteúdo: O *proxy* reverso pode distribuir a carga entre diversos servidores ativos na sua área. Neste caso, o *proxy* reverso pode necessitar re-endereçar a URL (*Universal Resource Locator*) em cada página — alterar o endereço conhecido da internet para um endereço interno.
- Servidor / *cache* de conteúdo estático: Um servidor *proxy* pode aliviar a carga dos servidores *web* cacheando o conteúdo estático como figuras, e outros gráficos.
- Compressão: o servidor *proxy* reverso podem otimizar e comprimir o conteúdo para aumentar a velocidade de carga.

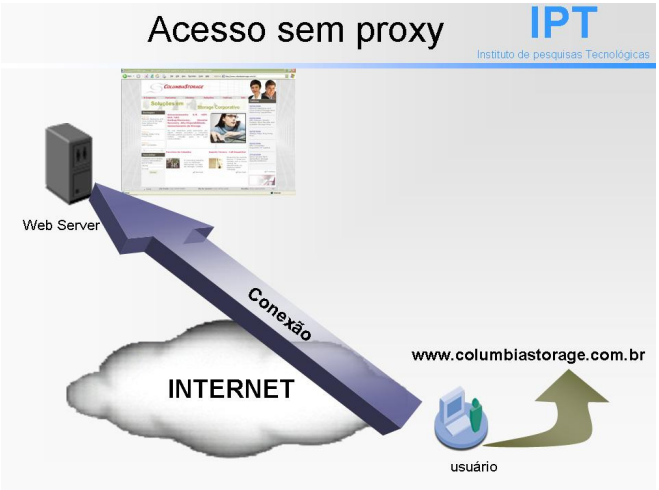
[7] CISCO: fabricante de renome de dispositivos de telecomunicação.
[8] Front end: os termos front end e back end referem-se aos estágios iniciais e finais de um fluxo de processo

4. FUNCIONAMENTO

Todas as vezes que se busca uma pagina na Internet, têm-se que acessar o *web server* do *site* desejado, este servidor muitas vezes está em outro continente, e os objetos precisam navegar milhares de quilômetros até nossa casa. Um outro usuário pode acessar a mesma página, pouco tempo depois e a trilha até o *web server* será novamente trilhada. A idéia de tornar a Internet mais ágil, rápida e eficiente nos leva a imaginar um jeito de compartilhar os objetos trazidos dos *web servers* com o maior número de pessoas possível e não percorrer todo o caminho novamente.

Abaixo se demonstra como funcionaria um acesso à Internet sem o *proxy*/*cache*:

- O *browser* solicita uma conexão ao *web server* do *site*, — e.g., www.ipt.br — pela porta 80 http (*Hyper Text Transfer Protocol*);
- O *browser* estabelece conexão TCP (*Transfer Control Protocol*) com o *web server*;
- O *browser* solicita o conteúdo do *site* (GET / http/1.1);
- Durante o recebimento, o *browser* verifica se novos recursos são necessários para formar a página (text/html)
- Para cada recurso necessário uma nova requisição dentro da mesma conexão é solicitada, otimizando o recebimento da página;
- A página está carregada



Um segundo usuário que pode ser seu vizinho da residência ao lado ou mesmo da baia ao lado, acessando o mesmo conteúdo, precisa percorrer o mesmo caminho.

Conforme a RFC 2612 (http 1.1), o conteúdo do pacote enviado seria o seguinte:

SEM proxy
GET /index.html HTTP/1.1
Host: www.nlanr.net
Accept: */*
Connection: Keep-alive

COM proxy
GET http://www.nlanr.net/index.html HTTP/1.1
Host: www.nlanr.net
Accept: */*
Proxy-connection: Keep-alive

No caso de termos um *proxy*/*cache* instalado o primeiro acesso seria exatamente igual ao exemplo anterior, porém o segundo visitante ao mesmo *site* teria os seguintes passos:

- O *browser* solicita a conexão ao *web server* pela porta 80, porém o *proxy* responde para o *browser* estabelecendo a conexão;
- O *proxy* estabelece uma conexão com o *web server.*;
- O *browser* solicita o conteúdo do *site* (GET/http1.1) para o *proxy;*
- O *proxy* solicita ao *web server* quais objetos formam a página;
- O *proxy* verifica se os objetos existentes no *cache* ainda são válidos. O *proxy* entrega i*media*tamente o conteúdo válido e busca no *web server* apenas os objetos não válidos;
- A página está carregada.

Como vimos na descrição dos passos do segundo acesso, a instalação do *proxy*/*cache* divide a conexão com o *web server* em duas partes; cliente - *proxy* e *proxy* - *web server*. Esta divisão permite ao *proxy*/*cache* verificar quais conteúdos estão armazenados localmente e quais devem ser buscados na Internet. Para definir se os objetos a serem fornecidos ao usuário deverão ser novos ou os armazenados localmente, o *proxy* solicita uma conexão ao *web server* e verifica quais os objetos formam a página neste momento e verifica no índice se todos os objetos estão armazenados e validos. Todos os objetos validos são enviados ao usuário, enquanto os objetos novos são recebidos e entregues posteriormente.

5. PROTOCOLOS *INTERCACHE*

O principal objetivo dos protocolos *Intercache* é fornecer informações, a partir de uma solicitação de um objeto por um cliente ou servidores *cache* vizinhos, do local mais próximo para obtenção dos objetos em questão; se será obtido de um vizinho ou recuperado diretamente de sua origem.

Apesar da transmissão de todos os objetos ser efetuada via o protocolo HTTP, a comunicação entre os servidores *Intercache* é toda realizada via um protocolo menor e muito mais simples.

Os protocolos a serem examinados nesse artigo provêm de propostas de abordagens diferenciadas, como:

- Abordagem hierárquica

- Abordagem por *hashing*

- Adaptativo *web caching* – experimental – em desenvolvimento por WREC (*Web Replication and Caching Work Group*). Esta é baseada em *CacheMesh, WebWave*, Abordagem direta com coordenação central

Os protocolos oriundos destas abordagens são:

- ICP - *Internet Cache Protocol* v2 – RFC 2186, RFC 2187 - 1997

- HTCP – *Hyper Text Caching Protocol* – RFC 2756 - 2000

- CARP - *Cache Array Routing Protocol – expired Internet Draft* - 1998

- WCCP v1 e v2 - *Web Cache Coordination Protocol - expired Internet Draft* - 2001

- *Hierarchical* HTTP *Routing Protocol - expired Internet Draft* - Vinod Valloppillil – 1997

Outros protocolos que seguem estas mesmas abordagens foram criados e estão documentados tanto em RFCs, quanto por Internet *Drafts*, porém não serão discutidos neste artigo.

5.1. Cache Digests

Há de se acrescentar que qualquer que seja a implementação de protocolos no servidor *proxy-cache*, o mesmo pode vincular técnicas como o *Cache Digests*, que prevê a criação de uma tabela em memória, que mantém a relação de todos os objetos armazenados no *cahce* de seus vizinhos. Desta forma pode-se reduzir o atraso da pesquisa pois a consulta aos servidores vizinhos deixa de ser executada.

Todavia, as tabelas de objetos devem estar sempre atualizadas. Para isso é usada a função *hash* para codificar as entradas e diminuir o tamanho das tabelas. Um ponto negativo para manutenção destas tabelas é o fato do servidor necessitar de mais recursos, como processamento e memória para calcular o *hash* de cada objeto.

Esta técnica pode ser implementada em conjunto com o protocolo ICP ou HTCP.

5.2. ICP - Internet Cache Protocol v2 – RFC 2756 2000

A função principal do ICP é determinar se uma determinada página e objetos estão armazenados em um servidor *cache* vizinho, ou seja, ele permite um *cache* consultar outros sobre seu conteúdo e, baseado nos tempos de respostas, decide qual *cache* entregará um dado objeto.

Hierarquicamente, um *cache* solicitado pode ser um pai, um filho ou um irmão. Pais normalmente estão mais próximos das conexões de Internet que os filhos.

Basicamente a pesquisa é feita através de consulta a vizinhos, que responde terem (*HIT*) ou não (*MISS*) um determinado objeto. Após receber todas as respostas, o servidor que originou a pesquisa decidirá a qual servidor pesquisado o objeto será recuperado, dependendo do número de *hops* da resposta de todos os servidores. A resposta que tiver o menor número de

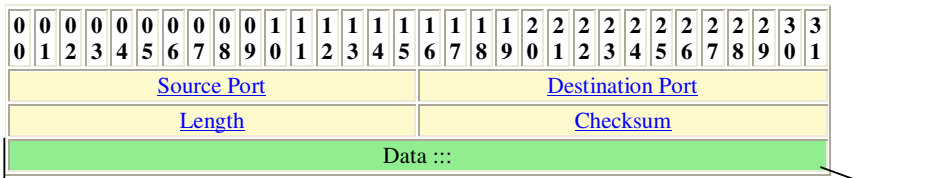
O problema é que o ICP aumenta a latência e o tráfego da rede por causa das respostas *MISS* dos servidores vizinhos.

O formato da mensagem ICP é composto por um cabeçalho de 20bytes, seguido por uma carga útil (*payloado*) de tamanho variável.

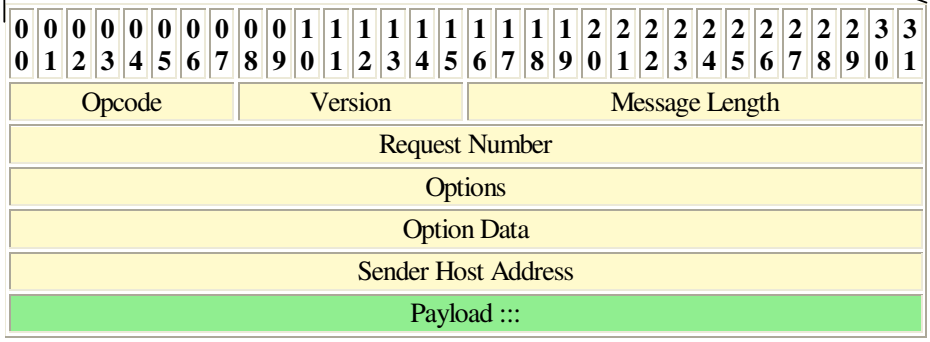
A mensagem ICP NÃO DEVE exceder 16.384 bytes de comprimento.

Tipo: protocolo de camada de aplicação. Porta: 3130 (UDP).

Protocolo UDP



Protocolo ICP



OPCODE

Value	Name
0	ICP_OP_INVALID
1	ICP_OP_QUERY.
2	ICP_OP_HIT.
3	ICP_OP_MISS.
4	ICP_OP_ERR.
5 - 9	
10	ICP_OP_SECHO.
11	ICP_OP_DECHO.
12 - 20	
21	ICP_OP_MISS_NOFETCH.
22	ICP_OP_DENIED.
23	ICP_OP_HIT_OBJ.

Version. 8 bits – O número da versão do protocolo ICP.

Message Length. 16 bits – O comprimento total da mensagem ICP em bytes.

Request Number. 32 bits.- Um identificador opaque. Quando responde a uma pesquisa, este valor deve ser copiado na mensagem de resposta.

Options. 32 bits. – Flag que permite extensões de versão do protocolo em modos limitados.

Option Data. 32 bits. – Recursos opcionais. O seguinte recurso ICP é usado nesse campo: ICP_FLAG_SRC_RTT usa os últimos 16-bits do Option Data para retornar as medidas RTT.

Sender Host Address. 32 bits. – O endereço IPv4 do host que envia a mensagem. Este campo não deveria ser confiável pelo que é fornecido por getpeer-name(), accept(), and recvfrom(). Há certa ambiguidade no propósito original deste campo. Na prática não é usado.

Payload. Comprimento varíavel conforme o código. Frequentemente contém uma URL.

5.3. Hyper Text Caching Protocol - RFC 2756 - 2000

Uma alternativa melhorada para o ICP, é um protocolo para descobrir caches HTTP e dados cacheados, gerenciar conjuntos de caches HTTP, e monitorar atividade cache. Permite usar cabeçalhos de resposta e solicitação cheios (todo cabeçalho HTTP) para gerenciamento de cache.

Expande o domínio do gerenciamento de cache para incluir monitoramento de adições e deleções de servidores cache, solicitações de “deleções” imediatas e o envio de sugestões sobre objetos web como localizações terceirizadas de objetos cacheáveis, ou a medida do não “cacheamento”, ou ainda a não disponibilidade de objetos.

Implementação por UDP é mandatória e opcional por TCP.

Abaixo será exibido o formato do protocolo:

<table>
<tr><th>00</th><th>01</th><th>02</th><th>03</th><th>04</th><th>05</th><th>06</th><th>07</th><th>08</th><th>09</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td colspan="16">Message length
-</td></tr>
<tr><td colspan="8">Major version</td><td colspan="8">Minor version</td></tr>
<tr><td colspan="16">Data length</td></tr>
<tr><td colspan="4">Opcode</td><td colspan="4">Response</td><td colspan="6">reserved</td><td>F1</td><td>RR</td></tr>
<tr><td colspan="16">Trans ID
-</td></tr>
<tr><td colspan="16">Op data :::</td></tr>
<tr><td colspan="16">Auth length</td></tr>
<tr><td colspan="16">Sig time
-</td></tr>
<tr><td colspan="16">Sig expire
-</td></tr>
<tr><td colspan="16">Key name :::</td></tr>
<tr><td colspan="16">Signature :::</td></tr>
</table>

5.4. Web Cache Coordination (control, Communication) Protocol

Conteúdo da tecnologia de roteamento primeiramente introduzida em 1997, provê um mecanismo para redirecionamento de fluxo de tráfego [para caches] em tempo

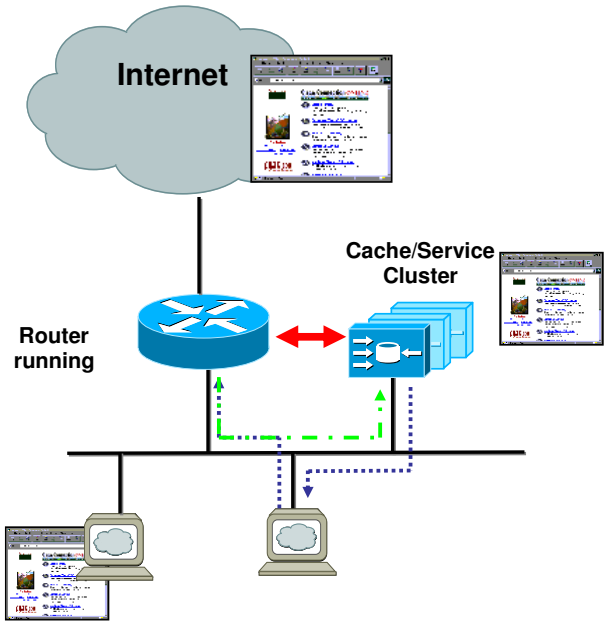
real e possui mecanismos de balanceamento de carga embutido, além de escalonamento, tolerância a falhas, e certeza de serviços (*fail safe*).

Foi desenvolvido em duas versões:

V1 - usado em associação com um ou mais *Web-cache*, para que o roteador possa distribuir o tráfego para os *caches*. Também inclui um mecanismo onde um ou mais *caches* ditam quanto do tráfego HTTP o roteador deverá distribuir entre um conjunto de *caches*.

V2 - especifica interações entre um ou mais roteadores e um ou mais *web-caches*. Assim pode-se estabelecer e manter um redirecionamento transparente de tipos de fluxo de tráfego selecionados através de um grupo de roteadores. O tráfego selecionado é redirecionado para um grupo de *web-caches*, otimizando o uso dos recursos e tempos de resposta.

A Figura abaixo demonstra o funcionamento do WCCP, que permite ao roteador interceptar a solicitação do cliente e encaminha-la para o *proxy-server* mais próximo.



Os pacotes podem ser dos seguintes tipos:

HERE_I_AM

3 bytes	
Type	
Protocol Version	
Hash revision	
Hash Information (1)	
Hash Information (7)	
U	Reserved
Received Id.	

I_SEE_YOU,

3 bytes
Type
Protocol Version
Change number
Received Id.
Number of WCs
Web-Cache List Entry(0)
Web-Cache List Entry (n) v

ASSIGN_BUCKETS.

3 bytes			
Type			
Received ID			
Number of Web Caches			
Web Cache 0 IP address			
Web Cache n IP address			
Bucket 0	Bucket 1	Bucket 2	bucket 3
Bucket 252	Bucket 253	Bucket 254	bucket 255

5.5. Cache Array Routing Protocol - expired Internet Draft - 1998

Algoritmo que distribui URLs entre vários servidores de *cache*, identificando qual o vizinho que deve ser consultado, usa uma função *hash* baseada em um mecanismo de seleção de *proxy*.

Projetado para ser eficiente e escalável em termos de balanceamento de carga com alto número de *hit ratios* e mínima latência. Diferentemente do ICP, não usa mensagens de consulta, diminuindo o tráfego da rede local.

Desvantagem: requer mais processamento

5.6. CRISP Web Cache - Inter Cache Communication Protocol

É uma pesquisa que está em alinhamento com pesquisas sobre comunicação *intercache* usando *multicast – Adaptative web Caching*, LSAM (*hotspots*).

Esta estrutura de *Web proxy servers* autônomos compartilha os objetos com um serviço de mapeamento e que podem duplicar todo o mapeamento de objetos, ou apenas parte dele, para balancear custo de acesso (*overhead*).

O grupo de trabalho que está à frente da pesquisa é *Web Replication and Caching* (WRC) *Work Group*.

6. PROBLEMAS CONHECIDOS: HTTP Proxy/Caching RFC 3040, RFC 3143

Aqui serão apresentados apenas alguns problemas conhecidos com o cacheamento em servidores *proxy*, confomre a RFC 3040, que trata da classificação e terminologia, e a RFC 3143, que apresenta realmente os conhecidos problemas de cacheamento:

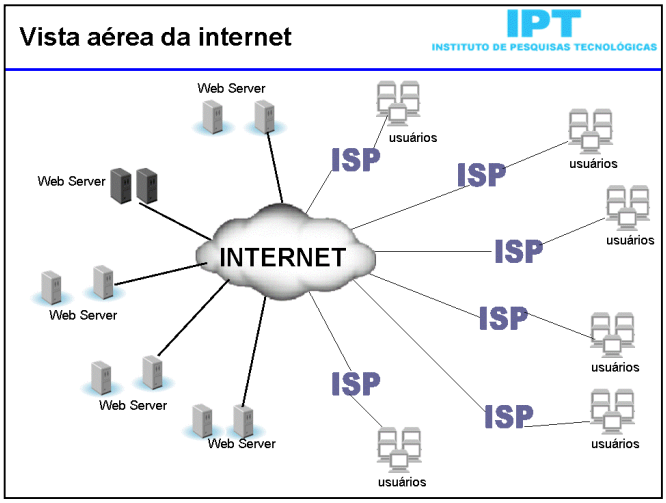
A seleção de pontos em redes heterogêneas *Caching proxy*;

Interceptação dos *proxies* dificulta autenticação baseada em IP;

A performance pode tender a um problema exponencial do tipo O(n^2), onde n é o número de *proxies* participantes. Isto pode fazer com oque o trafico ICP/http seja dominante dentro de uma rede.

7. ONDE ACELERAR

A distribuição dos dados em pontos mais próximos dos usuários finais ajuda a minimizar o tráfego da Internet. Os principais pontos que serão analisar: *Webserver*, ISP, Ambientes corporativos



Os usuários finais não serão analisados apesar de serem em maior número, o custo/benefício de melhorar a infra-estrutura não é interessante. O que não acontece no ambiente corporativo, onde a velocidade de acesso à Internet influi na produtividade de muitos colaboradores, além de alguns eventos internos serem transmitidos pela Internet para os escritórios remotos. O pronunciamento de um executivo da empresa que deve ser visto por todos os colaboradores causa um tráfego muito grande se for transmitido em *unicast*,[9] não importando a banda existente, esta jamais será suficiente.

7.1. Acelerar no *WebServer*

A fonte de todo conteúdo da Internet pode ser auxiliada com a instalação de um *proxy*/*cache*. O *webserver* poderia estar replicado em um *cache* aumentando a quantidade de conexões permitidas, pois um sistema preparado para fazer apenas cacheamento será mais rápido e mais eficiente que um servidor de propósito geral fazendo este papel. A segurança também seria melhorada pelo fato que se pode programar os routers para permitir que os *web servers* comuniquem-se apenas com o *cache*. Uma das características do *cache* é permitir que todas as requisições de

[9] Unicast: endereçamento para um pacote é feito a um único destino

um mesmo número de IP (Internet *Protocol*) utilizem a mesma conexão. Numa instalação com mais de um *cache*, mesmo que uma invasão consiga sobre carregar um *cache*, não afetaria o *site* como um todo.

A implementação de um ambiente de *cache* pode reduzir o número de *Web servers* visto que as requisições de conexão das páginas serão respondidas pelo *cache*, deixando os servidores mais aliviados, permitindo a redução da quantidade de servidores.

Benefícios desta implementação: redução da quantidade de *web server*; aumento da segurança; tráfego criptografado entre os *web servers* e *caches*; maior quantidade de acesso a pagina *web*;

7.2. Acelerar no ISP

Um dos pontos de maior interesse de economia de banda é o ISP.

O *Cache* possui a capacidade de integrar-se com servidores de filtragem de conteúdo e antivírus. No caso do filtro de conteúdo, o *cache* verifica qual o usuário está solicitando o acesso à pagina e verifica no filtro de conteúdo se aquele usuário pode acessar o *site*, se estiver autorizado, o acesso é permitido, caso contrário o usuário recebe uma mensagem personalizada de acesso negado.

A integração com antivírus é efetuada pela requisição de verificação das paginas da Internet pelos servidores de antivírus no primeiro acesso, se não contiver vírus, a pagina é entregue ao usuário, ou imediatamente apagada, programável, no caso de infecção.

Benefícios desta implementação: Redução no link entre o ISP e a Internet; aumento da segurança; Possibilidade de autenticar os usuários, reduzindo o acesso dos inadimplentes; Criação de produto restringindo o acesso de certos usuários, Criação de um produto Internet "*vírus free*" a ser comercializado para os usuários ou diferencial de oferta, através da integração do *cache* com *farm*[10] de servidores de antivírus; Eliminação da utilização do site como replicador de *spam*;

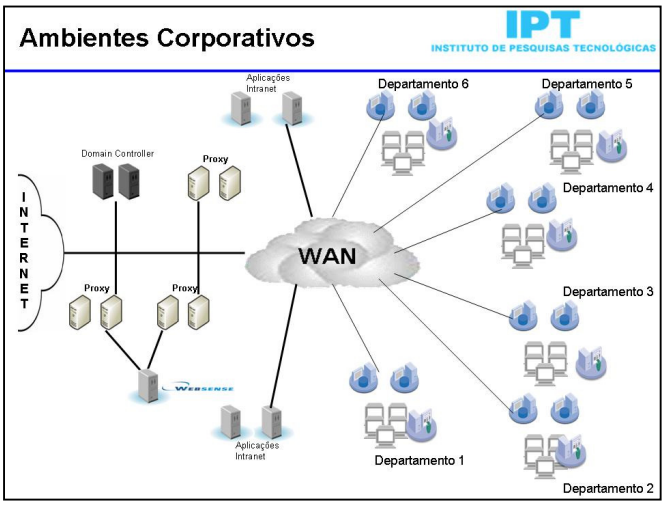
7.3. Acelerar no usuário final

A aceleração no usuário final residencial não é compensadora, devido ao custo de implementação dos *caches*.

[10] Farm:

7.4. Acelerar nos usuários corporativos

Outro ponto são empresas um onde ao empregar-se o conceito de *cache* outros benefícios podem sobressaltar-se: além de minimizar a necessidade de banda para interligar a empresa a Internet, pod-se criar um ambiente controlado de acesso a Internet.



Os ambientes corporativos possuem necessidades especiais e diferentes de simples acesso a Internet. Muitas empresas possuem escritórios em diferentes cidades e são interligados para compartilhamento de arquivos. Neste ambiente se pode criar uma estrutura de caches, onde um (PAI) será responsável por acessar a Internet e passar as páginas para os outros caches (filhos).

Benefícios desta implementação: Filtragem de conteúdo; Tráfego P2P; NetNews;

7.5. Aceleração de aplicativos de negócios

O desafio: acesso rápido dos escritórios remotos às aplicações de missão critica.

Aplicações como ERP e CRM são vitais para os negócios, permitindo que os colaboradores em escritórios ao redor do mundo acessem informações, criticas para os negócios, atualizadas e precisas. O problema é que empresas globais com aplicações baseadas em *Web* normalmente sofrem com grandes *delays* e performance extremamente baixa em seus escritórios remotos devido ao

congestionamento dos *lin*ks da WAN, servidores sobre carregados, e conexões dos escritórios remotos limitadas pelo alto custo.

O *Proxy*/*cache* configurado para servir como *Enterprise Content Delivery Network* para acelerar as aplicações de negócios minimiza o *delay* provocados pelos servidores centralizados e WAN, trazendo aos usuários remotos acesso rápido às aplicações criticas, aumentando o retorno do investimento das aplicações removendo as barreiras para a utilização e aumentando a produtividade, sem aumento de banda ou servidores.

Esta solução tem a flexibilidade para entregar conteúdo http, FTP e *streaming media*, além de oferecer o controle dos dados cacheáveis para maximizar a performance para aplicação de negócios e a economia de banda. Muitas aplicações possuem uma parte significante de conteúdo dinâmico, que será cacheado sem alterar a aplicação, o administrador habilita o gerenciadores para ajustar a maneira que o *proxy*/*cache* trata este tipo de conteúdo.

8. CONCLUSÃO

O uso de servidores *proxy* na Internet, em especial servidores *proxy-cache*, contribui muito para a economia da banda disponível e para tornar o serviço mais rápido para os usuários. As vantagens do uso de servidores *proxy-cache* são evidentes, visto que existem muitos recursos na Internet que são utilizados por diversos usuários. No esquema tradicional, a cada requisição de usuário, uma nova conexão é feita, mesmo que o recurso já tenha sido solicitado alguns segundos antes. O uso de servidores *proxy-cache* permite que requisições já feitas anteriormente não necessitem de novas conexões com a Internet, otimizando o uso de banda e tornando mais rápido o acesso à rede.

Isto posto, conclui-se que a utilização de grupos de servidores *cache*, configurados de forma hierárquica auxilia o atendimento da demanda do volume de dados que trafegam na rede. Além disto, para conseguir confiabilidade e redundância do grupo criado, pode-se criar *cluster* de servidores com compartilhamento de carga e utilização do protocolo *intercache* que prover maior eficiência, ou uma implementação conjunta entre WCCP e HTCP, por exemplo.