Redes de Computadores

Aulas Práticas

Implementação de um Proxy HTTP

O que é um proxy ?

- Genericamente é um processo que actua funcionalmente como intermediário entre dois sistemas
  - No modelo cliente servidor intermedeia a interacção entre o cliente (executando num computador) e o servidor (executando noutro computador)
  - O proxy pode executar por sua vez num terceiro computador

Porque se usam *proxies* ?

- Pode não ser possível que os dois sistemas intermediados comuniquem diretamente
- Permite controlar o tráfego e informação (conteúdos) trocados, num ponto central de controlo para monitoração e análise de tráfego
- Permite melhorar o desempenho, se tiver funcionalidade acrescida de um sistema de "*caching*"
  - *Proxy c/ caching*
- Permite filtrar seletivamete a informação que o atravessa ou implementar formas de controlo de acesso em relação ao tráfego que o atravessa complementarmente ao controlo de origem ou destino
  - Ex., *application-proxies* como funcionalidade de sistemas *firewall*, complementando filtragem ao nível aplicação à filtragem ao nível transporte e ao nível rede

Proxies vs. Protocolos

- No caso mais simples e geral, cada proxy intermedeia especificamente um dado protocolo
  - Ex., HTTP, FTP, ... ou outro protocolo do nível aplicação na pilha TCP/IP
    - Embora se possa pensar em proxies que actuam a outros níveis (ex., proxy genérico para TCP, etc.) ou proxies com suporte "multi-protocolo"
- Cliente (browser) envia os pedidos e recebe as respostas do proxy HTTP
  - O proxy "sabe" interpretar os pedidos, redirige os pedidos para os servidores HTTP finais, obtém as respostas que sabe interpretar e devolve essas respostas ao cliente
  - É "visto" pelo cliente como se fosse um servidor HTTP a quem se enviam pedidos (URLs) em geral
  - É visto por um servidor final como se fosse um cliente HTTP
- O *proxy* HTTP (como o que se pretende realizar e testar) implementará a intermediação do protocolo HTTP (com servidores HTTP/1.0)

Utilização de Proxies em HTTP

HOST Cliente (Browser)

HOST PROXY

Intermediação

HOST Servidor HTTP (WEB)

Proxy

Controlo
Caching
Filtragem

Ex., Firefix, IE, Safari, Chrome, Opera, Android browser...

Ex., Apache, IIS, GWS, ngineX, Resin, lighthttpd, SJSWS, ...

HOST Cliente
(só envia pedidos via um dado PROXY)

HOST Servidor
(só aceita pedidos que venham de um dado PROXY)

Utilização de Proxies em HTTP

HOST Cliente (Browser)

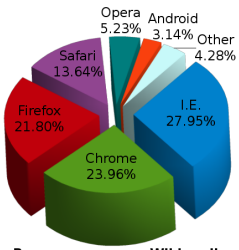
HOST PROXY

Intermediação

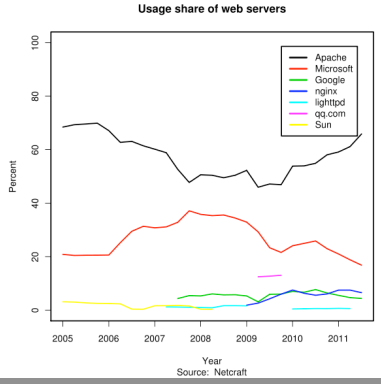
HOST Servidor HTTP (WEB)

Proxy

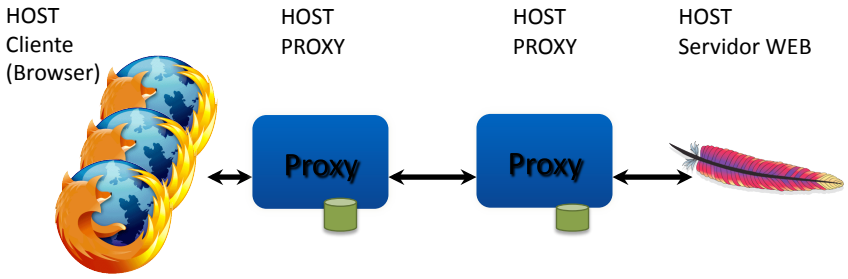
Controlo
Caching
Filtragem



Browser usage on Wikimedia March 2012



Utilização de Proxies em HTTP



Desenvolvimento de um proxy HTTP para o protocolo HTTP/1.0

- Pretende-se implementar um proxy capaz de suportar o protocolo HTTP/1.0
  - Conexões não persistentes
- A ideia é partir da implementação que já anteriormente foi feita para suportar o CHAT
  (ChatHttpServer)
- Código auxiliar fornecido para reutilização:
  - HTTPUtilities
  - Url (processamento/*parsing* de URLs e suas partes constituintes)
  - Ver tb o exemplo de parsing de URLs ou pedidos usando URLs

HTTP

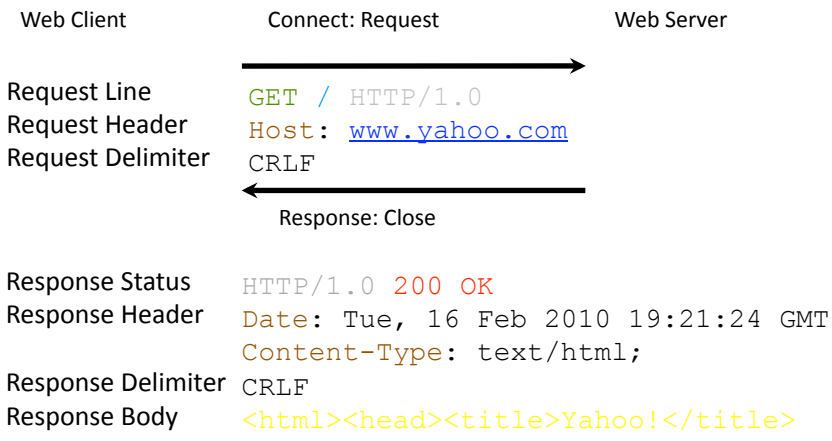


- Mar/1989, Tim Berns Lee, "Information Management: A Proposal", CERN
  - http://www.w3.org/History/1989/proposal.html
- Out/1990 - "WorldWideWeb" *coined*
  - http://info.cern.ch/NextBrowser.html
- Out/1994 – Fundado o W3C
- Mai/1996 - RFC 1945 (HTTP 1.0)
- Jun/1999 - RFC 2616 (HTTP 1.1)

HTTP 1.0

```
HTTP/1.0 200 OK
Date: Tue, 16 Feb 2010 19:21:24 GMT
Content-Type: text/html;
CRLF
<html><head><title>Yahoo!</title>
```

HTTP 1.0



HTTP 1.1 vs. 1.0

- Métodos adicionais (PUT, DELETE, TRACE, CONNECT + GET, HEAD, POST)
- Cabeçalhos adicionais (Headers)
- Codificação de conteúdos transferidos (*chunk encoding / MIME types*)
- Conexões persistentes (*content-length* muito importante)
- *Pipelining* de pedidos

HTTP 1.1 (exemplo de um pedido de um browser)

GET / HTTP/1.1
Host: localhost:80
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_5_8) AppleWebKit/534.50.2 (KHTML, like Gecko) Version/5.0.6 Safari/533.22.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
CRLF

HTTP 1.1 (exemplo de pedido de um browser via um proxy)

GET http://asc.di.fct.unl.pt/ HTTP/1.1
Host: asc.di.fct.unl.pt
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_5_8) AppleWebKit/534.50.2 (KHTML, like Gecko) Version/5.0.6 Safari/533.22.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie:
__utma=148045474.1196602131.1317253730.1334621895.1334690328.83;
__utmc=148045474;
__utmz=148045474.1329257742.35.2.utmcsr=asc.di.fct.unl.pt|utmccn=(referral)|utmcmd=referral|utmcct=/mst/menu.html;
__utma=95942283.1923607104.1317309834.1334621879.1334690384.18;
__utmc=95942283; __utmz=95942283.1334690384.18.4.utmcsr=di.fct.unl.pt|utmccn=(referral)|utmcmd=referral|utmcct=/;
_saml_idp=aHR0cHM6Ly9pZHAuZmN0LnVubC5wdC9pZHAvc2hpYmJvbGV0aA%3D%3D
Pragma: no-cache
Connection: keep-alive
Proxy-Connection: keep-alive
CRLF

Aspectos de implementação

- **Deve suportar a passagem do porto onde atende os clientes passando como parâmetro**
  - java proxy <port> , ex., java proxy 8080
- **A implementação deve suportar inicialmente o comando GET e não necessita de ser concorrente**
  - Ex., para outros pedidos, poderá retornar "Not Implemented", excepção 501
- Posteriormente, poderá acrescentar suporte para os comandos HEAD, POST e PUT, por exemplo
- Pelas razões apresentadas, deve seguir uma abordagem em que todos os pedidos que sejam feito em HTTP/1.1 sejam aceites, mas sejam convertidos para pedidos HTTP/1.0

Gestão de pedidos, erros e excepções

- Pedido != GET: Not Implemented (501)
- *Unparseable request*: Bad Request (400)
- Parsing usando as ferramentas auxiliares dadas
- Postel's law:
  - ***Be liberal in what you accept, and conservative in what you send***
    convert HTTP 1.1 request to HTTP 1.0
    convert \r to \r\n
    etc...

Conversão HTTP/1.1 para HTTP/1.0

- Para suportar esta conversão:
  - Substituir a versão HTTP 1.1 por HTTP 1.0 nos pedidos
  - Substituir o URL pelo caminho completo do recurso
- Ex.:
  - Browser-Proxy
    - GET http://www.google.com/index.html HTTP/1.1
  - Proxy-Servidor final (www.google.com)
    - GET /index.html HTTP/1.0
    - Host: www.google.com:80

Outros aspectos de implementação

- Para garantir mais compatibilidade, o proxy deve passar sempre todos os cabeçalhos HTTP recebidos do cliente para o servidor final (e vice-versa)
  - de modo a garantir que todas as páginas sejam vistas exactamente da mesma forma como se o cliente estivesse a comunicar com o servidor final
- Mas o proxy vai ter que filtrar (não passando) os cabeçalhos específicos que digam respeito à versão HTTP/1.1
- Ex.:
  - **"Connection: ..."**
  - **"Proxy-Connection: ... "**
  - **"Keep-Alive: ... "**
- A resposta passada ao browser pode ser passada sem qualquer tratamento

Para testar

- Usar os browsers e diferentes servidores para verificar a "generalidade" do proxy implementado
  - Redirigir o proxy (configuração do browser) para ensaiar a implementação
- Mais possibilidades:
  - Telnet to your proxy and issue a request

```
> ./proxy 5000
> telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.localdomain
(127.0.0.1).
Escape character is '^]'.
GET http://www.google.com/ HTTP/1.0

(HTTP response...)
```


Aspectos a ter em conta quando estiver a ensaiar o seu proxy

- *Debugging* no proxy (controlo de pedidos browser-servidor e respostas servidor-browser)
  - Ver exemplos da aula prática
- Alguns "sites" (servidores web) não funcionam quando o proxy não implementa as instruções dos métodos PUT e POST

Documentação complementar

- Ver materiais (aula prática 6)
- Ter em conta a matéria teórica relativa ao protocolo HTTP
- O livro base da cadeira apresenta informação de detalhe sobre o protocolo HTTP no Cap. 3
- RFC 1945: normalização do protocolo HTTP versão 1.0, contendo a sua especificação minuciosamente documentada
- Google, Wikipedia, ...

- É muito importante “fazer”, “testar”, “ensaiar”, “praticar” o funcionamento do protocolo HTTP e proxy a implementar