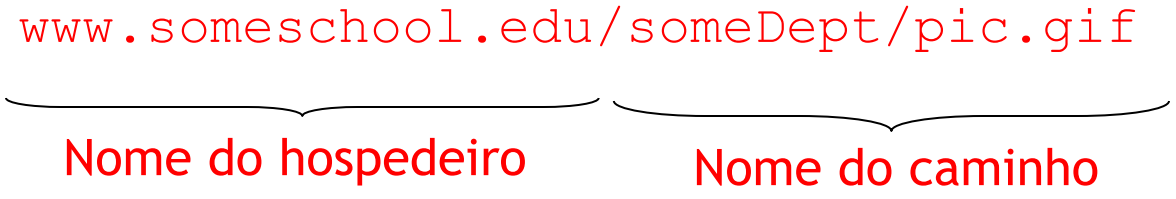
Web: Jargões

- Quase todas as páginas WWW consistem de:
  - página base HTML, e
  - vários objetos referenciados.
  - endereçados por uma URL
- URL tem duas partes: nome de hospedeiro, e nome de caminho:

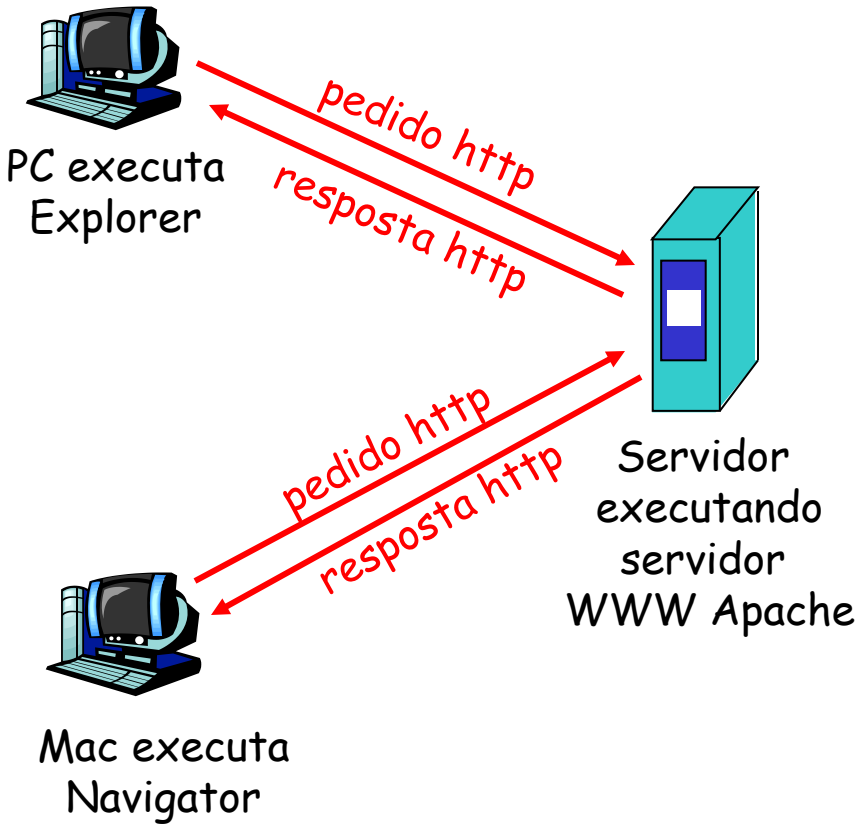
- Agente usuário para WWW é o browser:
  - Internet Explorer
  - Firefox
  - Chrome
- Servidor para WWW é o "servidor Web":
  - Apache (domínio público)
  - MS Internet Information Server (IIS)



WWW: o protocolo HTTP

HTTP: *hypertext transfer protocol*

- protocolo da camada de aplicação para WWW
- modelo cliente/servidor
  - *cliente:* *browser* que pede, recebe, "visualiza" objetos WWW
  - *servidor:* servidor Web envia objetos em resposta a pedidos
- http1.0: RFC 1945
- http1.1: RFC 2068



Mais sobre o protocolo HTTP

Usa serviço de transporte TCP:

- cliente inicia conexão TCP (cria *socket*) com servidor, porta 80
- servidor aceita conexão TCP do cliente
- mensagens HTTP (mensagens do protocolo da camada de apl) trocadas entre *browser* (cliente HTTP) e servidor Web (servidor HTTP)
- encerra conexão TCP

HTTP é "sem estado"

- servidor não mantém informação sobre pedidos anteriores do cliente

Nota

Protocolos que mantêm "estado" são complexos!

- história passada (estado) tem que ser guardada
- Caso caia servidor/cliente, suas visões do "estado" podem ser inconsistentes, devem ser reconciliadas

Exemplo de Operação

Usuário entra com a URL:
www.algumaEscola.edu.br/algumDepto/index.html

(contém referência a 10 imagens jpeg)

1a. cliente http inicia conexão TCP ao servidor http (processo) em www.algumaEscola.edu.br Porta 80 é a default para o servidor http .

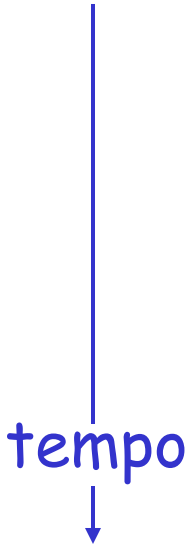
1b. servidor http no host www.algumaEscola esperando pela conexão TCP na porta 80. "aceita" conexão, notificando o cliente

2. cliente http envia mensagem de pedido (contendo a URL) para o socket da conexão TCP

3. servidor http recebe mensagem de pedido, forma mensagem de resposta contendo o objeto solicitado (algumDepto/index.index), envia mensagem para o socket

tempo

Exemplo (cont.)



5. cliente http recebe mensagem de resposta contendo o arquivo html, apresenta o conteúdo html. Analisando o arquivo html encontra 10 objetos jpeg referenciados

4. servidor http fecha conexão TCP.

6. Passos 1-5 são repetidos para cada um dos 10 objetos jpeg.

Conexões Não-persistentes e Persistentes

- Conexões não persistentes
  - HTTP/1.0 utiliza conexões não persistentes
  - Conexão TCP é fechada após o servidor http enviar o objeto
    - Cada conexão transporta uma requisição e uma resposta
    - No exemplo são necessárias 11 conexões TCP
  - Desvantagens
    - Uma conexão deve ser estabelecida por objeto
      - Necessário alocar buffers no TCP e manter variáveis do TCP tanto no cliente quanto no servidor
      - Atraso na transferência dos objetos
      - Partida lenta do TCP causado pelo controle de congestionamento
      - Conexões podem ser em série ou paralelo
  - Sobrecarrega servidores que atendem vários clientes ao mesmo tempo
  - Conexões paralelas reduzem o atraso

Conexões Não-persistentes e Persistentes

- Conexões Persistentes
  - Default para o HTTP/1.1
  - Servidor deixa a conexão TCP aberta após o envio da resposta
    - Requisições subseqüentes são enviadas pela mesma conexão TCP
  - Conexão é fechada após um certo tempo de inatividade (configurável)
  - Duas versões
    - Sem paralelismo: cliente lança uma nova requisição somente quando a resposta prévia é recebida
    - Com paralelismo: cliente pode lançar pedidos sem respostas prévias (default do HTTP/1.1)

Conexões Não-persistentes e persistentes

- Conexões Persistentes
  - Vantagem
    - Reduz o problema de partida lenta do TCP
      - Após ter enviado o primeiro objeto, não tem de enviar o próximo objeto na taxa inicial lenta
    - Reduz o atraso
  - Desvantagens da sem paralelismo
    - Conexão TCP fica pendurada enquanto resposta não é recebida
  - Vantagens da com paralelismo
    - Conexão fica pendurada por um menor tempo

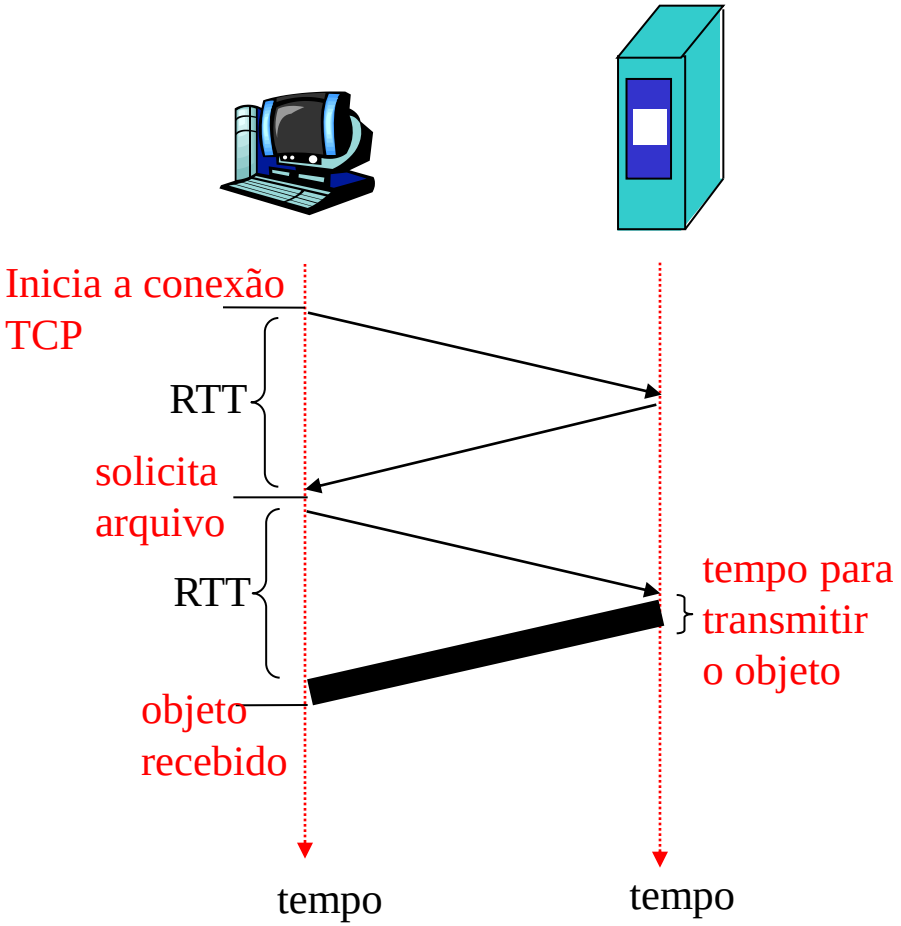
Modelagem do tempo de resposta

Definição de RTT *(Round Trip Time)*: intervalo de tempo entre a ida e a volta de um pacote entre um cliente e um servidor.

Tempo de resposta:

- um RTT para iniciar a conexão TCP
- um RTT para o pedido HTTP e o retorno dos primeiros bytes da resposta HTTP
- tempo de transmissão do objeto (página, imagem, etc.)

total = 2RTT+tempo de transmissão do objeto



Formato de mensagem HTTP

- Dois tipos de mensagem HTTP: *pedido, resposta*
- mensagem de pedido HTTP:
  - ASCII (formato legível por pessoas)
  - Linha de requisição: <método> <URL> <versão do HTTP>

linha do pedido (comandos GET, POST, HEAD)

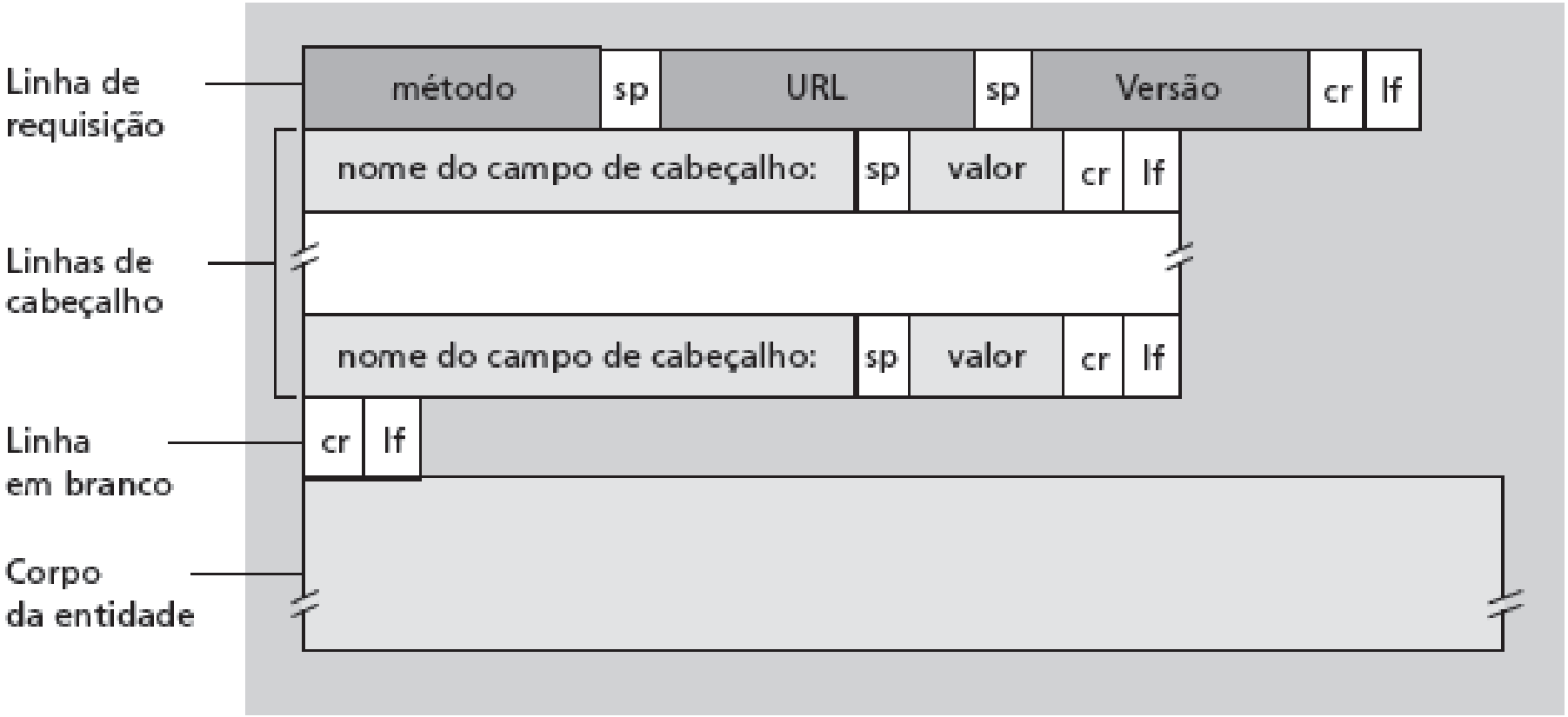
linhas do cabeçalho

```
GET /algumdir/pagina.html HTTP/1.1
Host: www.algumsite.com
Connection: close
User-agent: Mozilla/4.0
Accept-language:br
```

Carriage return, line feed indicam fim de mensagem

```
(carriage return, line feed)
```

Mensagem de Pedido http: formato Geral



Obs.: cr = carriage return; lf = line feed

Formato da mensagem de pedido http

- HTTP/ 1.0
  - Método Get
    - Solicita objeto especificado (URL)
  - Método Post
    - Usado pelo cliente http quando o usuário preenche um formulário
    - É uma mensagem de solicitação do objeto
      - Conteúdo específico da página web depende do que o usuário escreveu no formulário
  - Método Head
    - Similar ao método Get
      - Mas servidor envia a resposta sem o objeto solicitado
    - Usado para depuração

Formato da mensagem de pedido http

- HTTP/1.1
  - GET, POST, HEAD
- PUT
  - Envia o arquivo no corpo da entidade para o caminho especificado no campo de URL
- DELETE
  - Apaga o arquivo especificado no campo de URL
- Outros
  - TRACE: Ecoa o pedido, de maneira que o cliente possa saber o que os servidores intermediários estão mudando em seu pedido.
  - OPTIONS: Recupera os métodos HTTP que o servidor aceita.
  - CONNECT: Serve para uso com um proxy que possa se tornar um túnel SSL

Formato da mensagem http

- Método GET
  - Linha de comando indica o arquivo a ser transferido ao Browser
  - Host indica hospedeiro onde o objeto reside;
  - Connection: close indica para servidor usar conexão não persistente
  - User-agent: indica o tipo de browser usado
  - Accept-language: br indica que usuário prefere uma versão em português do objeto

linhas do cabeçalho

```
GET /algumdir/pagina.html HTTP/1.1
Host: www.algumsite.com
Connection: close
User-agent: Mozilla/4.0
Accept-language:br
```

(carriage return, line feed)

formatos HTTP: Resposta

linha de status (protocolo código de status frase de status)

linhas de cabeçalho

dados, e.x., arquivo html

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

Objeto solicitado ...
```

Exemplo de tipo MIME	Descrição
text/plain	Arquivo no formato texto (ASCII)
text/html	Arquivo no formato HTML, utilizado como padrão para documentos Web
Image/gif	Imagem com o formato GIF
Image/jpeg	Imagem com o formato JPEG
application/zip	Arquivo compactado

Formato da mensagem http: Resposta

- Códigos de status (Descritos na RFC 2616)
  - 1xx: Informational (Informação) – utilizada para enviar informações para o cliente de que sua requisição foi recebida e está sendo processada;
  - 2xx: Success (Sucesso) – indica que a requisição do cliente foi bem sucedida;
    - 200 OK: Pedido com sucesso, objeto pedido está na mensagem
  - 3xx: Redirection (Redirecionamento) – informa a ação adicional que deve ser tomada para completar a requisição;
    - 301 Moved Permanently: Objeto pedido movido, nova localização é especificado na mensagem (Location:)
  - 4xx: Client Error (Erro no cliente) – avisa que o cliente fez uma requisição que não pode ser atendida;
    - 400 Bad Request: Mensagem de pedido não é entendida pelo servidor
    - 404 Not Found: Documento pedido não foi encontrado no servidor
  - 5xx: Server Error (Erro no servidor) – ocorreu um erro no servidor ao cumprir uma requisição válida.
    - 505 HTTP Version Not Supported

Formato da mensagem http

- Método Post
  - Usado quando é necessário enviar dados ao servidor
    - para serem processados geralmente por um programa script identificado no Request-URI.
    - informações submetidas são incluídas no corpo da mensagem e formatadas como uma query string, além de conter cabeçalhos adicionais especificando seu tamanho (Content-Lenght) e seu formato (Content-Type).

```
POST /index.html HTTP/1.0
Accept: text/html
If-modified-since: Sat, 29 Oct 1999 19:43:31 GMT
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Nome=NamePessoa&Idade=99&Curso=Computacao
```

HTTP Cliente: faça você mesmo!

1. Telnet para um servidor Web:

```
telnet www.inf.ufsc.br 80
```

Abre conexão TCP para a porta 80 (porta default do servidor http) em www. inf.ufsc.br. Qualquer coisa digitada é enviada para a porta 80 em www.inf.ufsc.br

2. Digite um pedido GET http:

```
GET /~willrich/ HTTP/1.1
Host: www.inf.ufsc.br
```

Digitando isto (tecle carriage return duas vezes), você envia este pedido HTTP GET mínimo (mas completo) ao servidor http

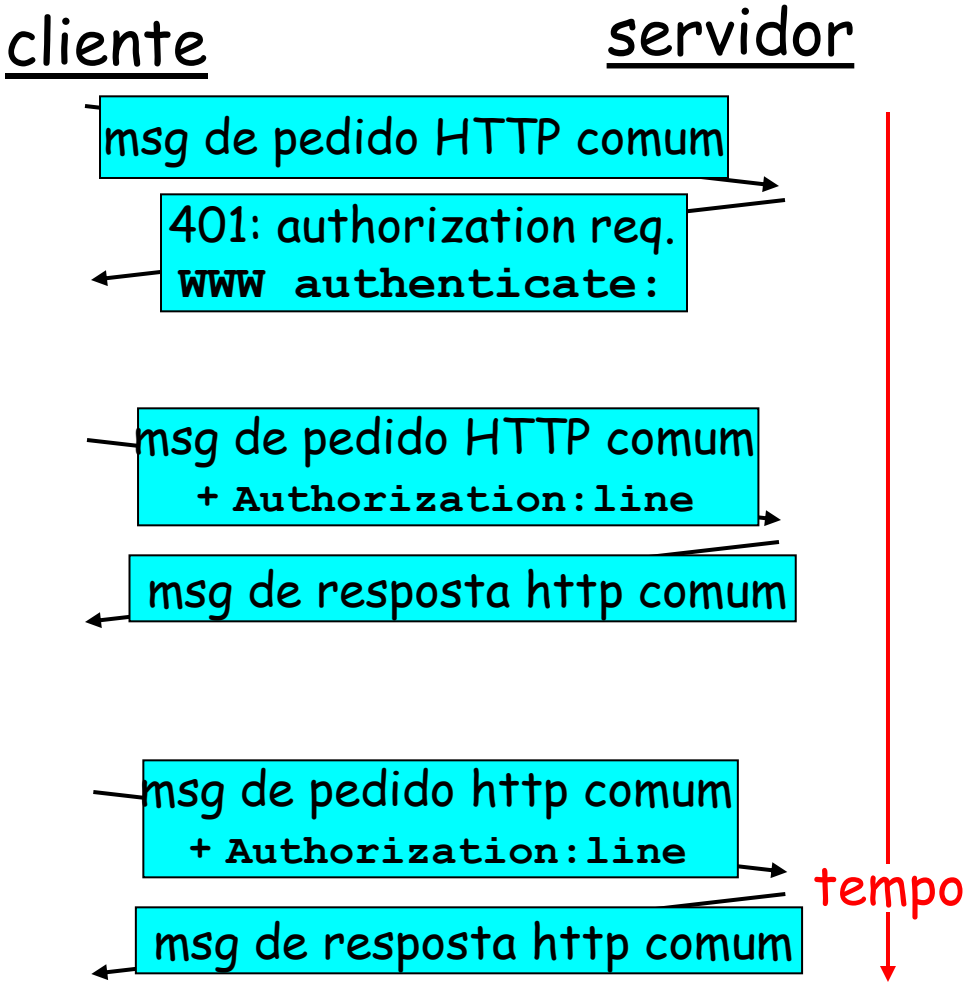
3. Examine a mensagem de resposta enviada pelo servidor http!

Interação usuário-servidor: autenticação

Meta da autenticação: controle de acesso aos documentos do servidor

- sem estado: cliente deve apresentar autorização com cada pedido
- autorização: tipicamente nome, senha
  - `authorization:` linha de cabeçalho no pedido
  - se não for apresentada autorização, servidor nega acesso, e coloca no cabeçalho da resposta

    `WWW authenticate:`

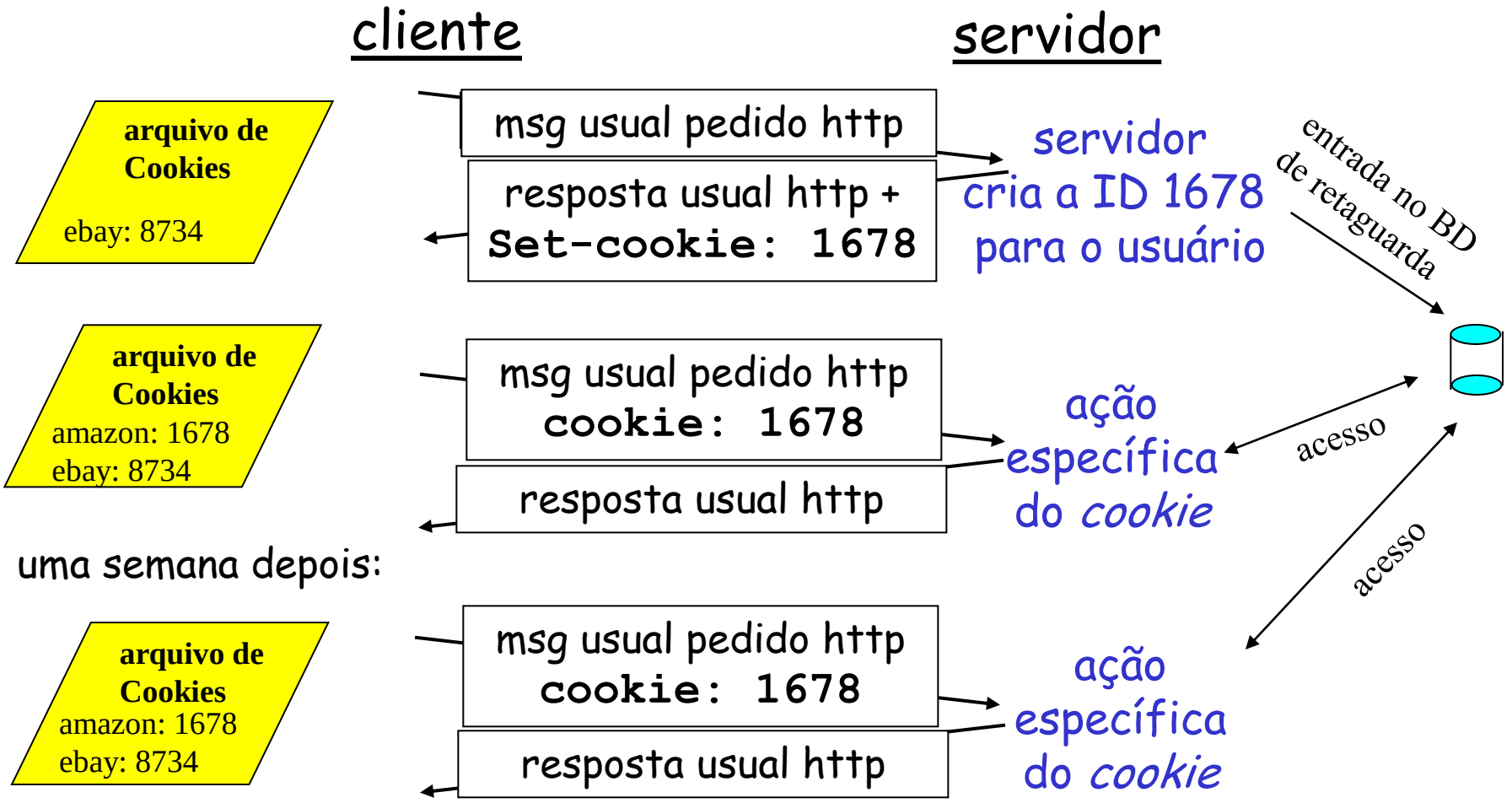


Browser guarda nome e senha para evitar que sejam pedidos ao usuário a cada acesso.

Interação usuário-servidor: cookies

- Cookies
  - Mecanismo que os sites utilizam para obter informações dos usuários
  - Acessando um site usando cookies pela primeira vez
    - Resposta inclui um cabeçalho Set-cookie: #
    - # é um número de identificação
  - Recebendo a mensagem o cliente inclui o nome do servidor e o número de identificação associado ao usuário
  - Acessando o mesmo site posteriormente
    - Pedido do usuário inclui um cabeçalho Cookie: #
    - Não sabe o nome do usuário, mas pode manter um estado do usuário

Cookies: manutenção do "estado" (cont.)



Interação usuário-servidor: cookies

- Finalidades
  - Quando servidor necessita de identificação, mas não quer perturbar o usuário com um pedido de nome e senha
  - Quando servidor quer lembrar das preferências de um usuário de modo que possa realização uma adaptação do conteúdo
  - Se o usuário estiver fazendo compras em um site, pode ser usado para anotar os itens que o usuário está comprando

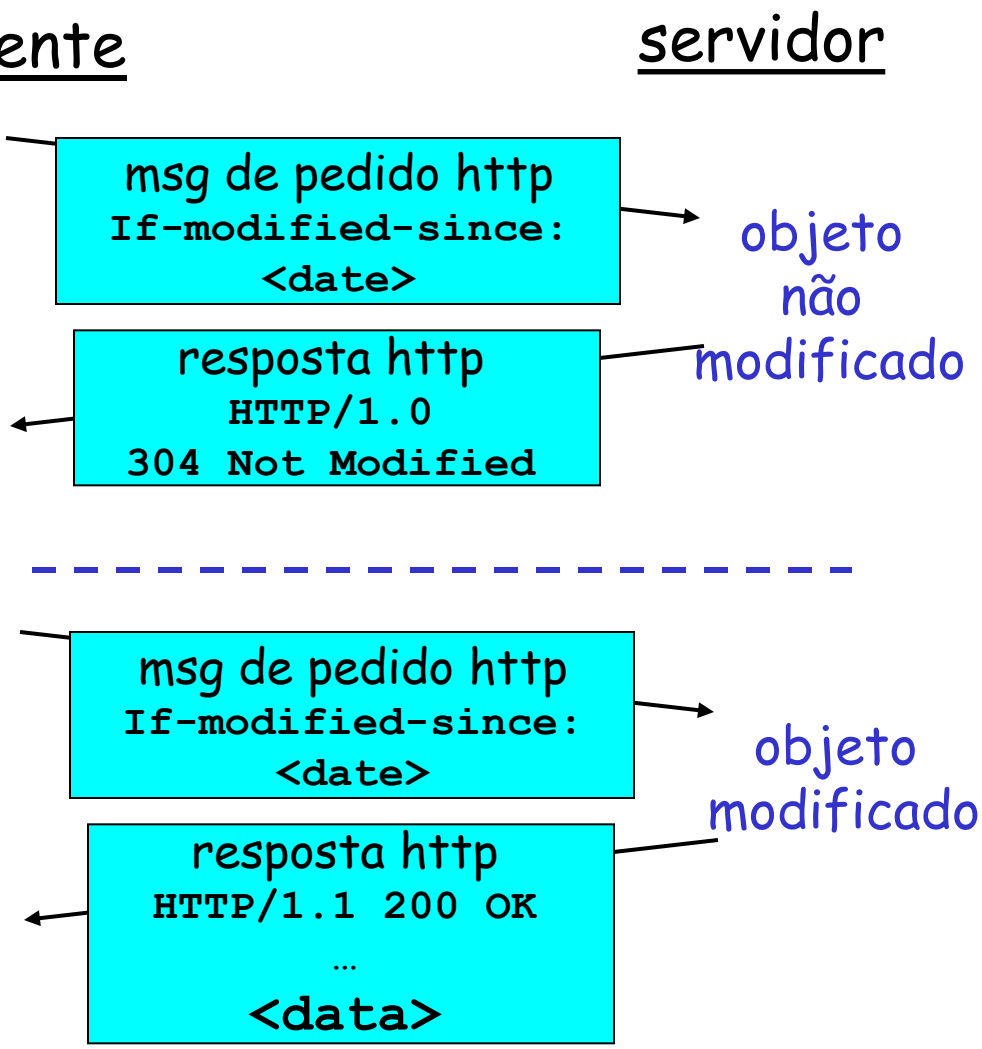
Interação usuário-servidor: GET condicional

- Meta: não enviar objeto se cliente já tem (no cache) versão atual
- cliente: especifica data da cópia no cache no pedido http

```
If-modified-since:
  <date>
```

- servidor: resposta não contém objeto se cópia no cache é atual:

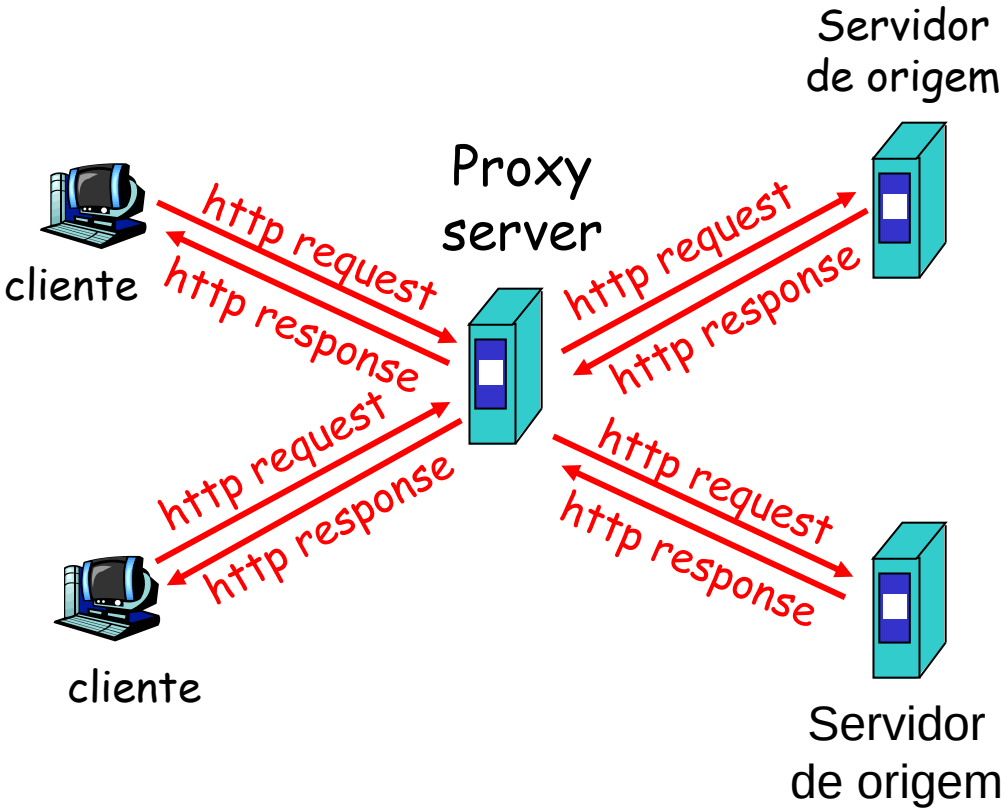
```
HTTP/1.0 304 Not
  Modified
```



Web Caches (proxy server)

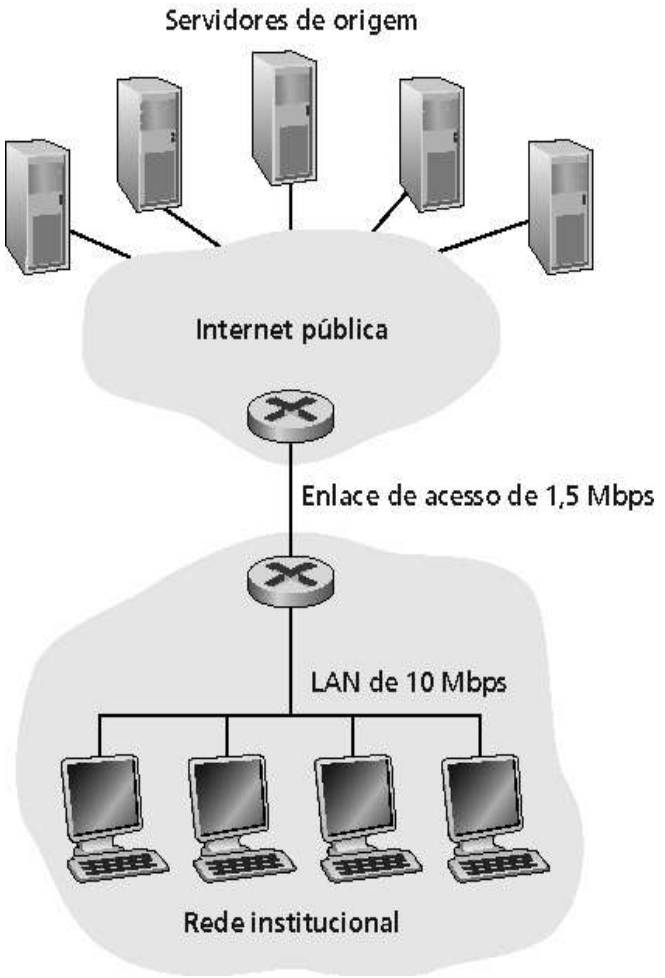
Meta: satisfazer pedido do cliente sem envolver o servidor de origem

- Usuário configura o browser: acessos web via cache web
- Cliente envia todos os pedidos http para a cache web
  - Se o objeto está na cache ele é imediatamente retorna uma resposta http
  - Senão o objeto é pedido para o servidor de origem, então retorna a resposta para o cliente



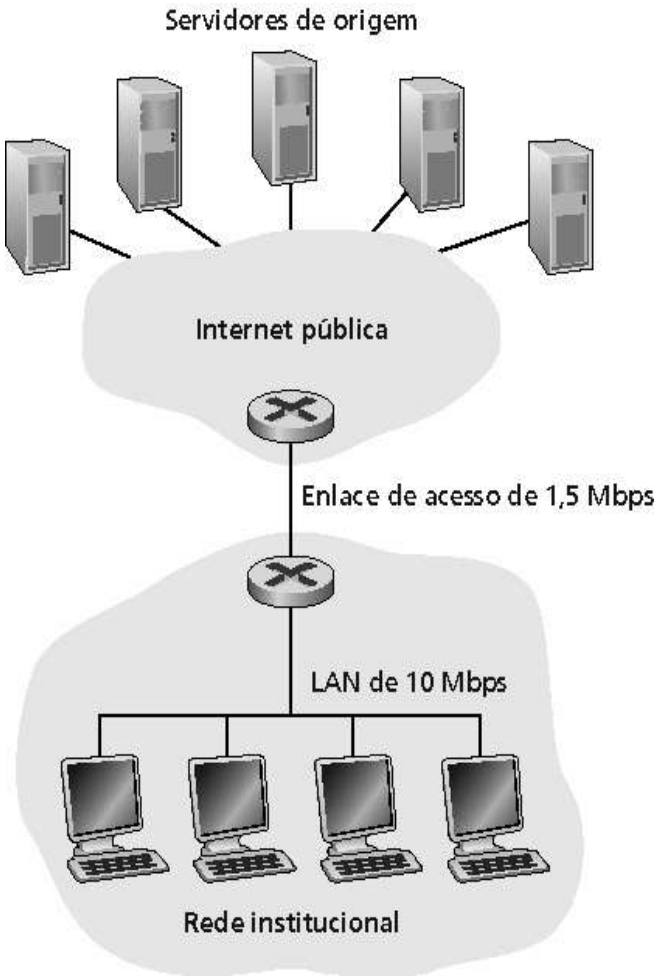
Exemplo de caching

- Suponha:
  - Tamanho médio objeto = 100.000 bits
  - Taxa média de requisições dos browsers da instituição para os servidores de origem = 15/s
  - Atraso do roteador institucional para ir a qualquer servidor de origem e retornar ao roteador = 2 s
- Conseqüências:
  - Utilização da LAN = 15%
  - Utilização do link de acesso = 100%
  - Atraso total = atraso da Internet + atraso de acesso + atraso da LAN



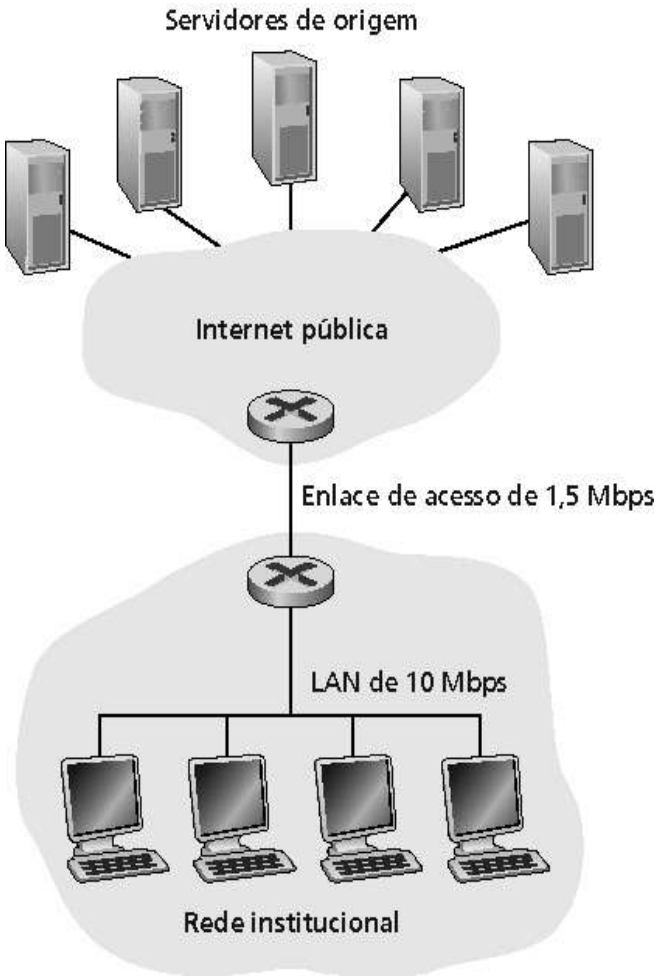
Exemplo de caching

- Solução possível
  - Aumentar a largura de banda do enlace de acesso, como, 10 Mbps
- Conseqüências
  - Utilização da LAN = 15%
  - Utilização do enlace de acesso = 15%
  - Atraso total = atraso da Internet + atraso de acesso + atraso da LAN
    - Tempo de acesso é reduzido pelo aumento da taxa de conexão
  - Freqüentemente é um upgrade caro



Exemplo de caching

- Instalação do cache
  - Suponha que a taxa de acertos seja .4
- Consequência
  - 40% das requisições serão satisfeitas quase que imediatamente
  - 60% das requisições serão satisfeitas pelo servidor de origem
  - Utilização do enlace de acesso reduzida para 60%, resultando em atrasos insignificantes (como 10 ms)
  - Média de atraso total é reduzir com a redução do tráfego de saída



Cache: Implementações

- http://squid-cache.org
  - Squid é um servidor proxi que suporta HTTP, HTTPS, FTP e outros
  - Roda em S.O. tipo Unix e Windows

Protocolo HTTPS

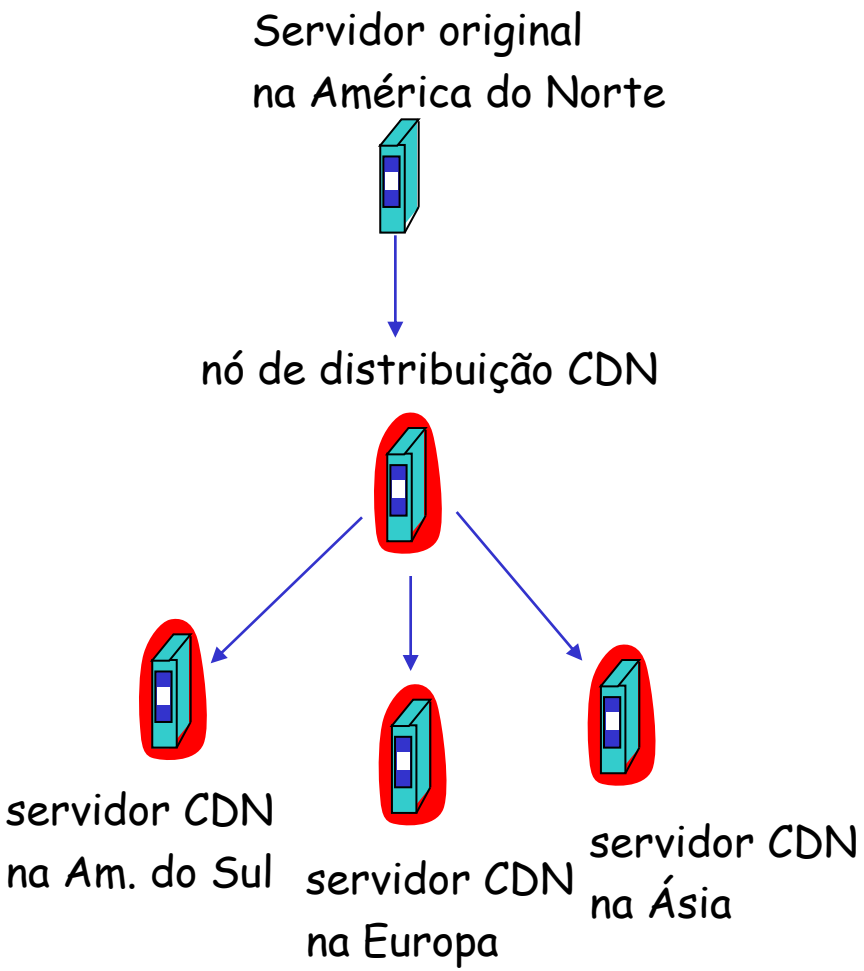
- HTTPS (HyperText Transfer Protocol Secure)
  - Implementação do protocolo HTTP sobre uma camada SSL ou do TLS.
    - permite que os dados sejam transmitidos através de uma conexão criptografada e que se verifique a autenticidade do servidor e do cliente através de certificados digitais.
  - Porta TCP usada por norma para o protocolo HTTPS é a 443.
    - Browser cria conexão com a porta 443 do servidor quando digitamos URL iniciando com 'https://'.
- Usando quando...
  - Se deseja evitar que a informação transmitida entre o cliente e o servidor seja visualizada por terceiros
    - Compras online, acesso a bancos
  - A existência na barra de tarefas de um cadeado demonstra a certificação de página segura (SSL).

Redes de distribuição de conteúdos (CDNs - *Content distribution networks*)

- Os provedores de conteúdo são os clientes da CDN

Replicação de conteúdo

- A empresa da CDN instala centenas de servidores através da Internet
  - Em ISPs próximos aos usuários
- A CDN replica o conteúdo de seus clientes nos servidores CDN. Quando o provedor atualiza o conteúdo, a CDN atualiza seus servidores



CDN: Seleção do Servidor

- CDN replica o conteúdo em vários servidores
- Desafios
  - Como replicar o conteúdo
  - Onde replicar o conteúdo
  - Como redirecionar os clientes para as réplicas
  - Como encontrar o conteúdo replicado
  - Como escolher entre as réplicas conhecidas

Seleção do Servidor

- Que Servidor?
  - Carga mais baixa → balancear a carga nos servidores
  - Melhor desempenho → melhorar o desempenho do cliente
    - Baseado na geografia? RTT? Vazão? Carga?
  - Qualquer nó vivo → oferece tolerância a faltas
- Como redirecionar o cliente para um servidor particular?
  - Como parte do roteamento → anycast
  - Como parte da aplicação → HTTP redireciona
  - Como parte da resolução do nome → DNS

Direcionamento baseada na aplicação

- HTTP oferece um modo simples para indicar que uma pagina web foi movida
- Servidor recebe um Get do cliente
  - Decide que servidor é o melhor para o cliente particular e objeto
  - Retorna HTTP rediret para o servidor selecionado
- Pode fazer decisões específicas de aplicação
- Introduz overhead adicionais → múltiplas conexões, resoluções de nome, etc.

Direcionamento baseada na resolução de nome

- Cliente usa DNS para resolução de nomes
- Servidor de nomes escolhe o endereço de servidor Web selecionado
  - Dependendo do cliente
- Quais informações pode servir para tomar a decisão?
  - DNS round robin (RFC 1794)
  - Métricas [Semi-]estáticas
    - Geografia
    - Métricas de rota

Como Akamai Trabalha

- Clientes buscam o documento html do servidor principal
  - Ex. busca index.html de cnn.com
- URLs para conteúdos replicados são substituídos no html
  - E.x.
    - <img src="http://cnn.com/af/x.gif">

    substituído por
    - <img src="http://a73.g.akamaitech.net/7/23/cnn.com/af/x.gif">
- Cliente é forçado a resolver aXYZ.g.akamaitech.net

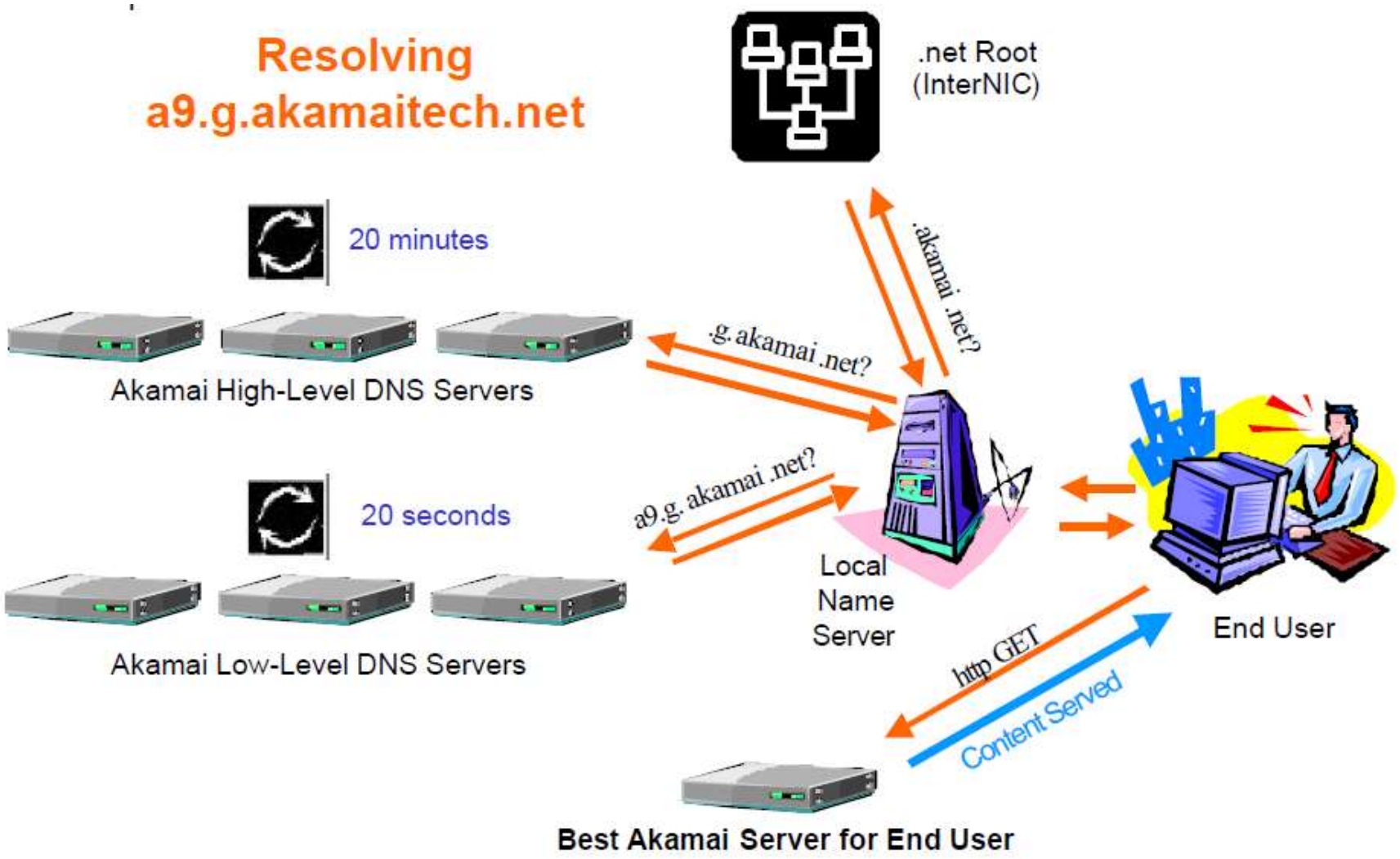
Como Akamai Trabalha

- Que conteúdo é replicado?
  - Akamai apenas replica conteúdos estáticos
- Nome modificado contém o nome do arquivo original
  - Ex.: <img src="http://a73.g.akamaitech.net/7/23/cnn.com/af/x.gif">
- Cliente consulta o servidor Akamai sobre o conteúdo. Servidor:
  - Primeiro checa cache local
  - Se não estiver em cache, pede o arquivo ao servidor primário e cacheia o arquivo

Como Akamai Trabalha

- Resolução de nome:
  - Servidor Raíz resolve o TLD .net
  - Servidor .net resolve o nome de akamaitech.net
  - Servidor de nomes akamaitech.net resolve o nome de g.akamaitech.net
  - Servidor de nome g.akamaitech.net escolhe o servidor da região

Como Akamai Trabalha



a73.g.akamaitech.net/7/23/cnn.com/af/x.gif

Mais sobre CDNs

Roteamento de pedidos

- A CDN cria um "mapa", indicando as distâncias entre os ISPs folhas e os nós CDN
- quando a solicitação chega num servidor DNS oficial :
  - o servidor determina qual é o ISP de onde provém o pedido
  - usa o "mapa" para determinar qual o melhor servidor CDN

Não são apenas páginas Web

- fluxos de áudio/vídeo armazenados
- fluxos de áudio/vídeo de tempo real
  - nós CDN criam uma rede sobreposta na camada de aplicação