Web-0

Introdução à Web: HTTP e HTTPs e Navegadores

Tiago Cruz de França

tcruz.franca@gmail.com

Agenda

- **World Wide Web**
- **HTTP**
  - HTTPs
- **Navegadores**

WWW (World Wide Web)

- **Comumente chamamos apenas de Web**
  - Criado em 1989
  - Disponibilizado em 1991
  - Tim Berners-Lee
    - **Principal responsável (idealizador) naquela época**
    - **Pesquisador do CERN (Conselho Europeu de Pesquisas Nucleares)**
      - Suiça
  - Objetivo original
    - **Catálogo de documentos para facilitar a busca dos mesmo pelos pesquisadores do CERN**

Mais sobre a História da Web

- **Documentos iniciais eram apenas texto**
  - Links eram usados para outros documentos quando necessário
    - **Formando uma teia (Web)**

- **A Web cresceu e hoje tem abrangência mundial (World Wide)**
  - Por isso o nome
    - **Seu uso não mais apenas acadêmico ou governamental**

Web: Termos Usados

- **O que chamamos de documentos antes na verdade são chamados de páginas Web**
  - A linguagem básica para criação de páginas é o HTML
    - **O HTML* (HyperText Markup Language) foi criado pelo Tim Berners-Lee**
    - **O HTML é uma linguagem (texto) de marcação que é processada pelo navegador***
    - **O navegador (browser) é o cliente básico necessário para apresentar a página ao usuário**
      - Ex: Firefox, Chrome, Safari, Internet Explorer, Opera...

Mais sobre a Web

- **Atualmente existe site sobre todo tipo de assunto**
  - Páginas pessoais, páginas de banco, institucionais, de cliente de e-mail...
    - **São tantos, que surgiram os serviços de busca**
- **Segurança na Web**
  - Para realizar operações com dados sensíveis (sigilosos) como operações bancárias existem tecnologias de transações seguras
    - **Criptografia, mecanismos de autenticação, etc.**

Navegadores (*Browsers*)

- **Veja uma lista de navegadores em:**
  - http://pt.wikipedia.org/wiki/Anexo:Lista_de_navegadores

- **Atividade:**
  - Instalar e testar pelo menos 3 navegadores

Atividades

- **Acesse uma página com HTTP e HTTPs**
  - Qual a diferença? Veja a barra com a URL
  - Tente acessar um site que use HTTPs, mas que force seu browser a exibir uma mensagem de segurança.
  - Busque no seu navegador:
    - **Onde configurar o proxy**
    - **Onde estão a lista de certificados digitais***
    - **Onde configura cookies e dados em cache**
  - Acesse uma página e veja se seu navegador possui recurso para apresentar o conteúdo (HTML e CSS) da página
  - Pare o cursor sobre um link e veja o resultado
  - Use um “encurtardor” de URL* (busque por um)

HTTP e HTTPs e

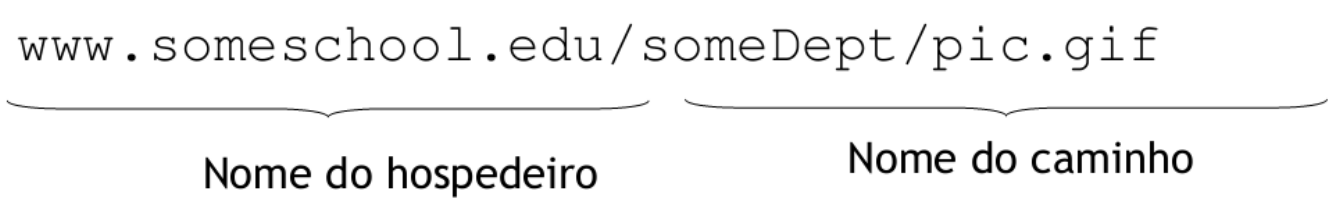
Princípios de Protocolos de Comunicação

Web e HTTP

- **Alguns jargões:**
  - Página Web consiste de objetos
  - Objeto pode ser:
    - **Um arquivo HTML, uma imagem JPEG, um Java applet, arquivo de áudio, etc.**
  - A página Web consiste de arquivo-HTML base que inclui vários objetos referenciados

Web e HTTP

- **Alguns jargões (cont.):**
  - Cada objeto é endereçado por uma URL (*universal resource locator*)
    - **Exemplo de URL:**



Visão geral do HTTP

- **HTTP: hypertext transfer protocol**
  - Protocolo da camada de aplicação da Web
  - Modelo cliente/servidor
    - **Cliente:browser que solicita, recebe e apresenta objetos da Web**
    - **Servidor:envia objetos em resposta a pedidos**

Visão geral do HTTP

- **HTTP 1.0: RFC 1945**
- **HTTP 1.1: RFC 2616, jun/1999**
  - (obsoleta RFC 2068), atualizada pela RFC 2817 (uso de transporte seguro TLS com HTTP)

Objetivo Original do HTTP

- **Capacidade de recuperar de um servidor documentos simples “somente-texto”**
- **Protocolo leve e rápido**

Visão geral do HTTP

- **Utiliza TCP:**
  - Cliente inicia **conexão TCP** (cria *socket*) para o servidor na **porta 80**
  - Servidor aceita uma conexão TCP do cliente
  - Mensagens HTTP (mensagens do protocolo de camada de aplicação) são trocadas entre o browser (cliente HTTP) e o servidor Web (servidor HTTP)
  - A conexão TCP é fechada

Visão geral do HTTP

- **HTTP é "*stateless*"**
  - O servidor não mantém informação (sessão) sobre os pedidos passados dos clientes
- **Protocolos que mantêm informações de "estado" são complexos!**
  - Histórico do passado (estado) deve ser mantido
  - Se o servidor/cliente quebra, suas visões de "estado" podem ser inconsistentes, devendo ser reconciliadas

Conexões HTTP

- **HTTP não persistente**
  - No máximo, um objeto é enviado sobre uma conexão TCP
  - O HTTP/1.0 utiliza HTTP não persistente
- **HTTP persistente**
  - Conexão é mantida e permite que múltiplos objetos possam ser enviados
  - TCP entre o cliente e o servidor
  - O HTTP/1.1 utiliza conexões persistentes em seu modo padrão

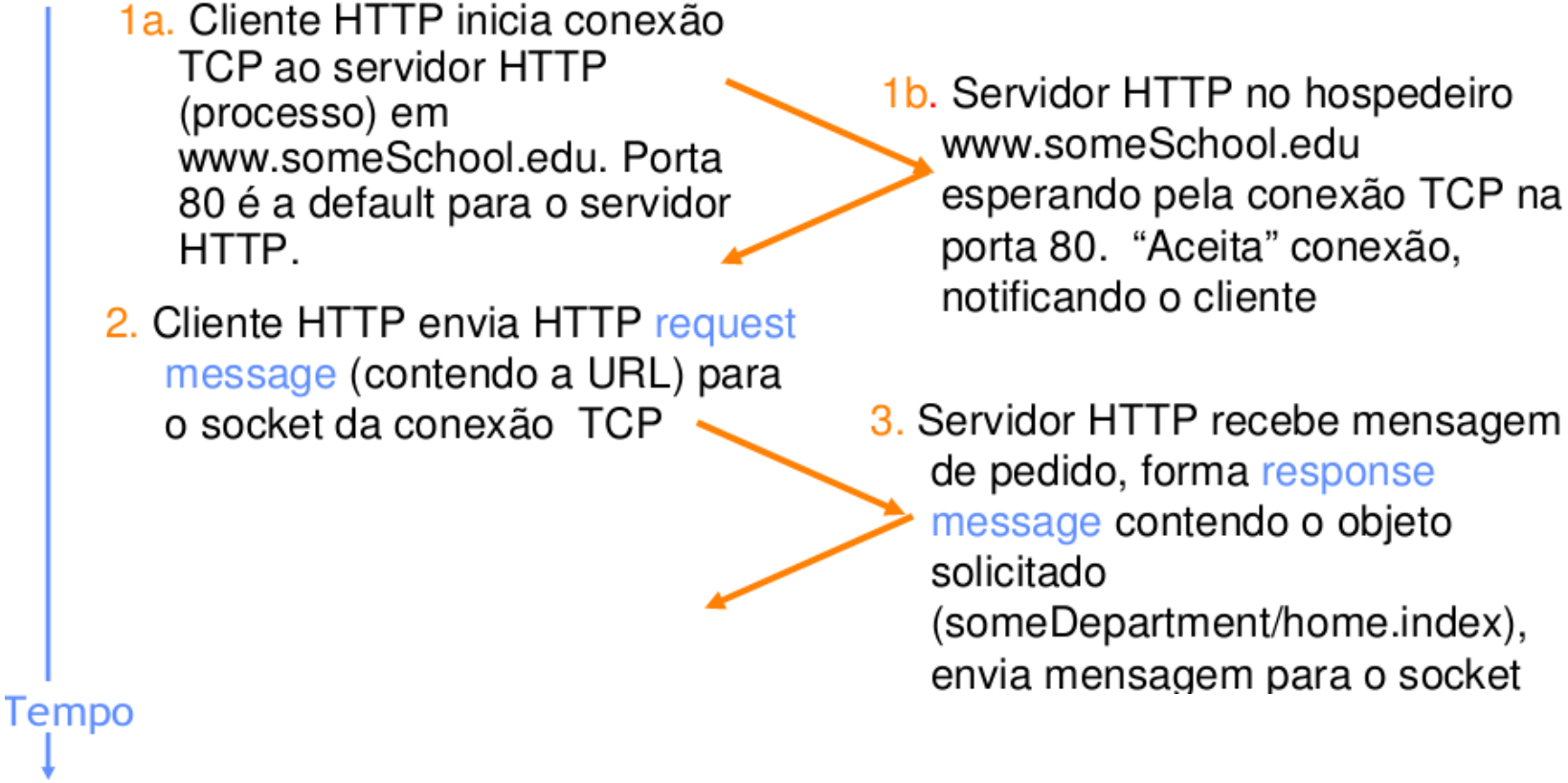
HTTP não persistente

- **Usuário entra com a URL:**

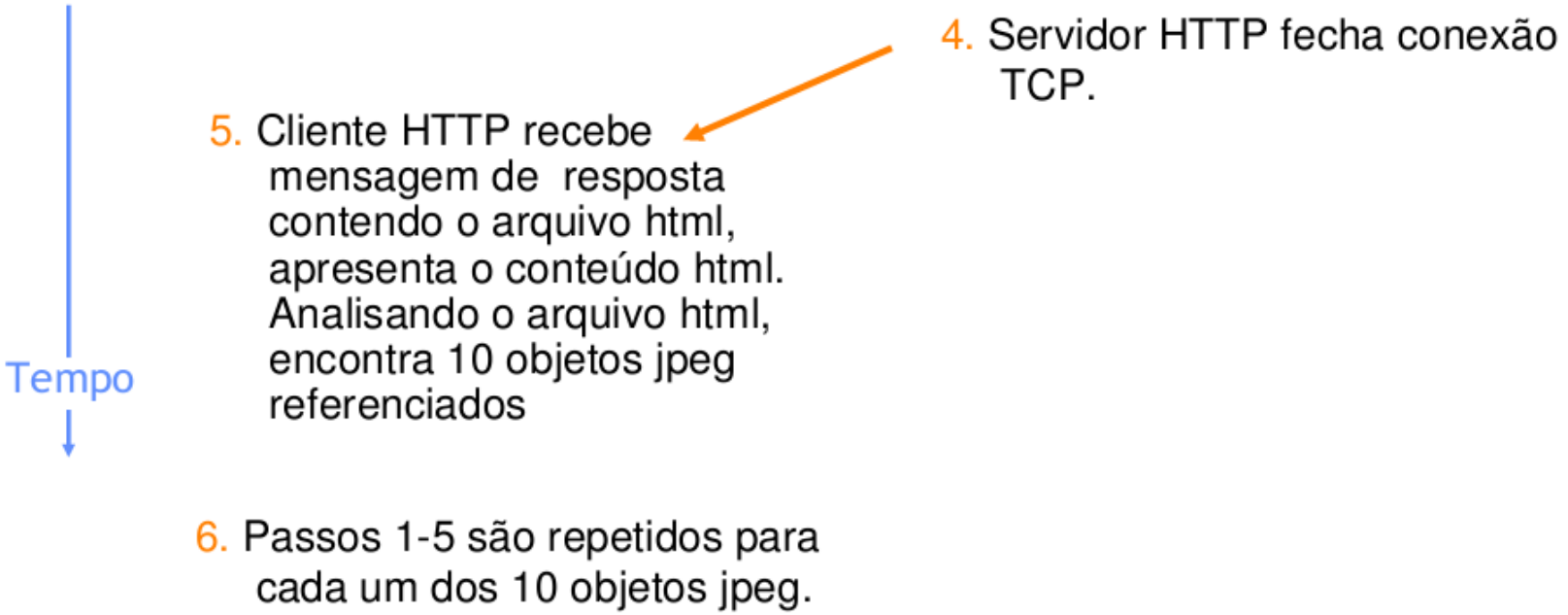
  www.someSchool.edu/someDepartment/home.index

  A página possui 10 objetos jpeg referenciados

HTTP não persistente



HTTP não persistente

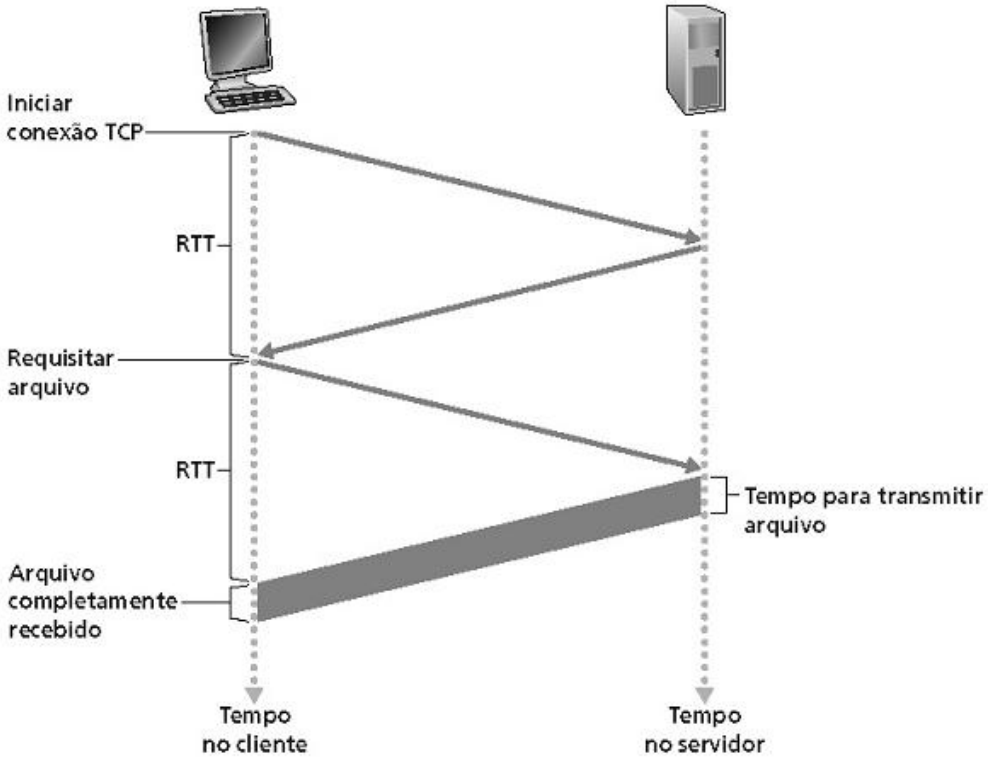


Tempo de resposta

- **Definição de RTT (ROUND TRIP TIME):**
  - Tempo para enviar um pequeno pacote que vai do cliente para o servidor e retorna.

Tempo de resposta

- **Tempo de resposta:**
  - Um RTT para iniciar (abrir) a conenexão TCP
  - Um RTT para requisição HTTP e primeiros bytes da resposta HTTP para retorno
  - Tempo de transmissão de arquivo



Total = 2RTT+ tempo de transmissão do arquivo

HTTP persistente

- **Características do HTTP não-persistente:**
  - Requer 2 RTTs por objeto
  - SO deve manipular e alocar recursos do hospedeiro para cada conexão TCP. Mas os browsers freqüentemente abrem conexões TCP paralelas para buscar objetos referenciados

HTTP persistente

- **HTTP persistente**
  - Servidor deixa a conexão aberta após enviar uma resposta
  - Mensagens HTTP subseqüentes entre o mesmo cliente/servidor são enviadas pela conexão

HTTP persistente

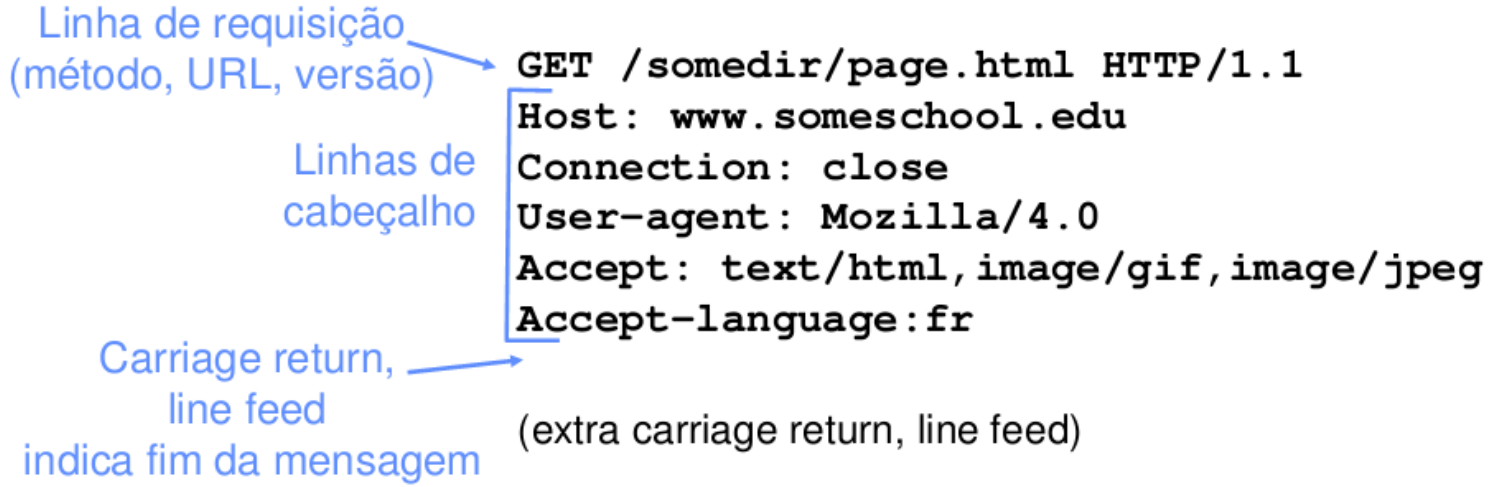
- **Persistente sem *pipelining* (paralelismo):**
  - O cliente emite novas requisições apenas quando a resposta anterior for recebida
  - Um RTT para cada objeto referenciado

HTTP persistente

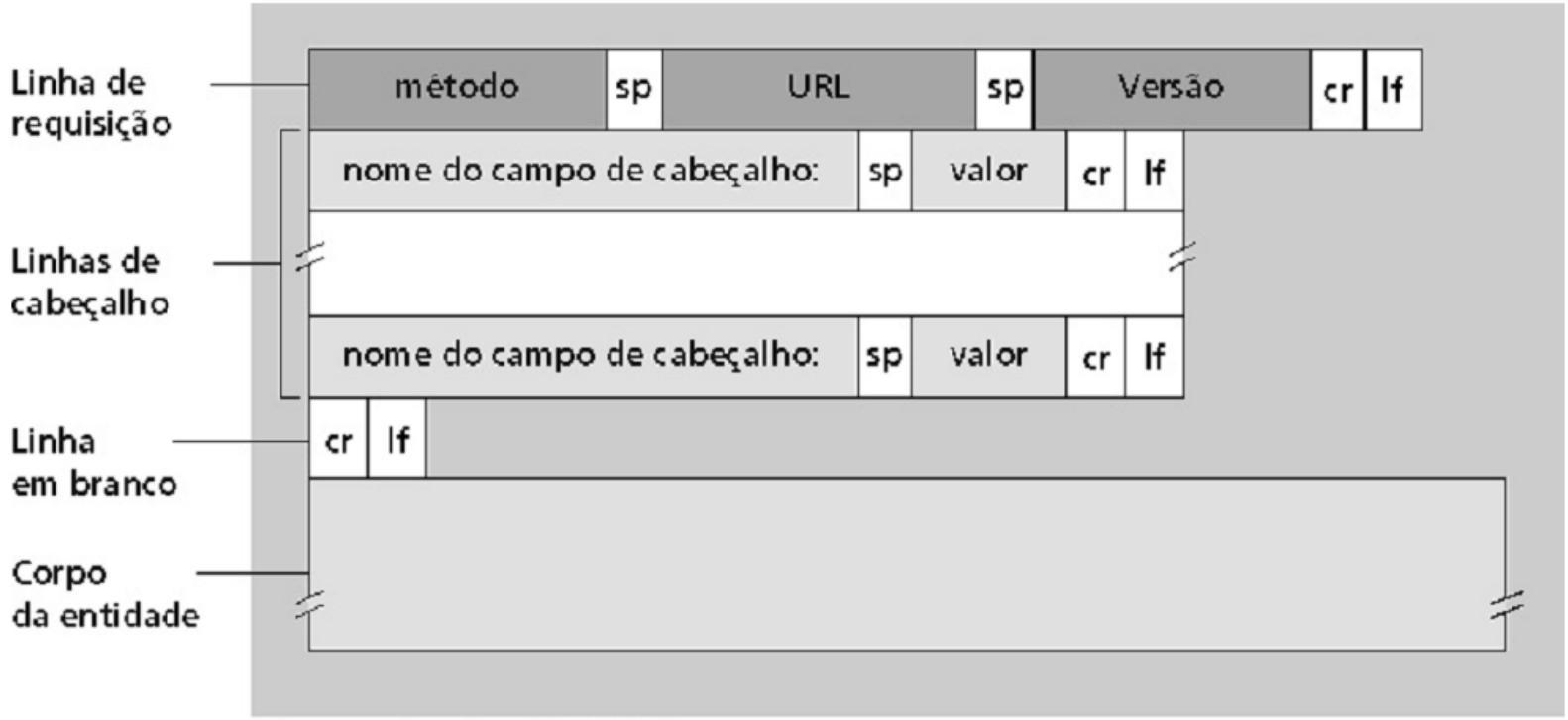
- **Persistente com *pipelining* (paralelismo):**
  - Padrão no HTTP/1.1
  - O cliente envia requisições assim que encontra um objeto referenciado
  - Tão pequeno como um RTT para todos os objetos referenciados

Mensagem HTTP request

- **HTTP *request message*:**
  - ASCII (formato legível para humanos)



Formato da mensagem de requisição HTTP



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

Tipos de Métodos

- **HTTP/1.0**
  - GET
    - **Requisita objeto**
  - POST
    - **Fornece dados de entrada para localizar página**
  - HEAD
    - **Pede para o servidor deixar o objeto requisitado fora da resposta**

Tipos de métodos

- **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**

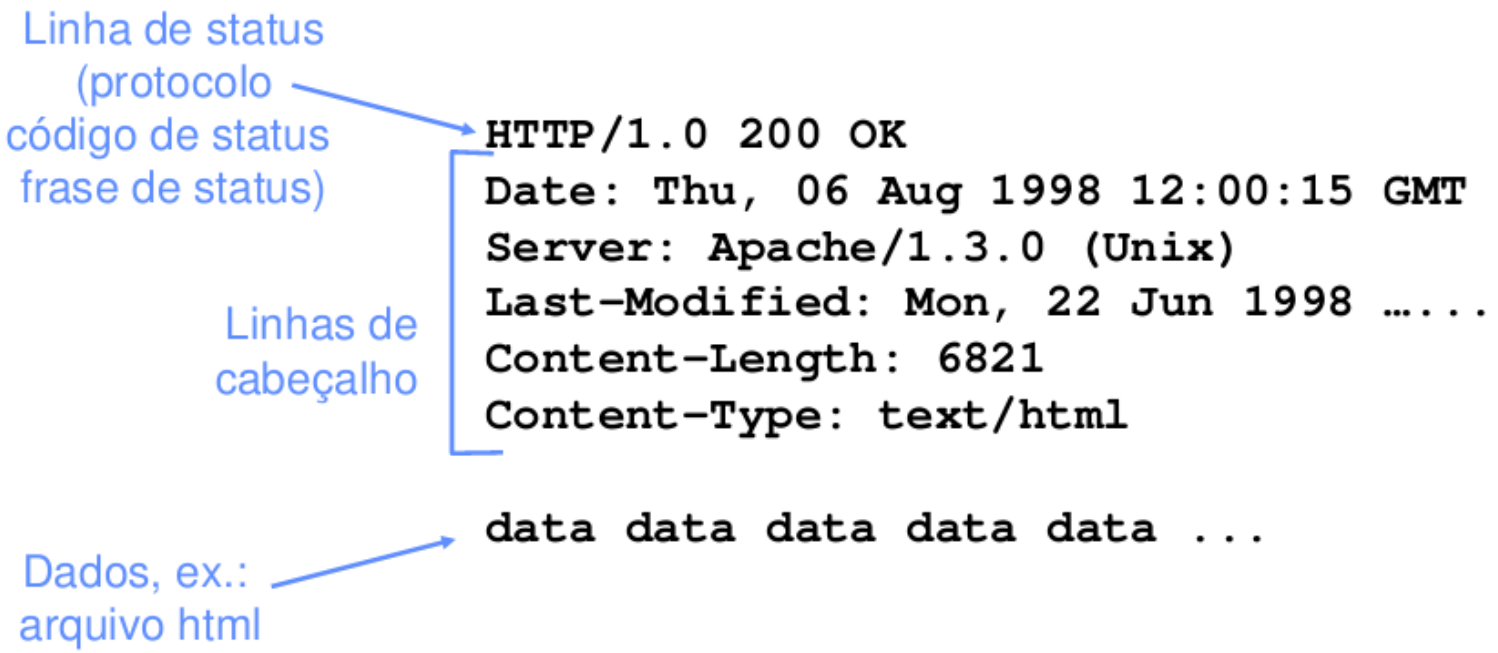
Entrada de formulário

- **Método Post:**
  - Apresenta ao servidor os dados de entrada de formulário no corpo da entidade, o que influencia o objeto a ser retornado

Entrada de formulário

- **Método URL:**
  - Utiliza o método GET
  - A entrada é enviada no campo de URL da linha de requisição:
    - **www.somesite.com/animalsearch?monkeys&banana**

Mensagem HTTP response



HTTP – Códigos de Retorno

- **Códigos possuem 3 dígitos**
- **Divididos em categorias:**
  - 1XX – informativo
  - 2XX – Sucesso
  - 3XX – Redireção
  - 4XX – Erro do cliente
  - 5XX – Erro do servidor

Códigos de *status* das respostas

- **Localizado na primeira linha da mensagem de resposta servidor cliente.**
- **Alguns exemplos de códigos:**
  - **200 OK**
    - **Requisição bem-sucedida, objeto requisitado a seguir nesta mensagem**
  - **301 *Moved permanently***
    - **Objeto requisitado foi movido, nova localização especificada a seguir nesta mensagem (Location:)**

Códigos de *status* das respostas

- **Alguns exemplos de códigos (cont.):**
  - **400 *Bad request***
    - **Mensagem de requisição não compreendida pelo servidor**
  - **404 *Not Found***
    - **Documento requisitado não encontrado neste servidor**
  - **505 HTTP *version* not supported**

Exercício Prático

- **Simulando um browser realizando uma requisição HTTP a um servidor Web**
  - Para isso, utilize o TELNET e faça uma requisição na porta 80
    - **Ex:**
      - telnet www.google.com.br 80
      - GET http://www.google.com.br HTTP/1.1
      - [ENTER]
      - [ENTER]

Cookies

Estado usuário-servidor: cookies

- **A maioria dos grandes Web sites utilizam cookies para monitorar os seus usuários (RFC 2109)**

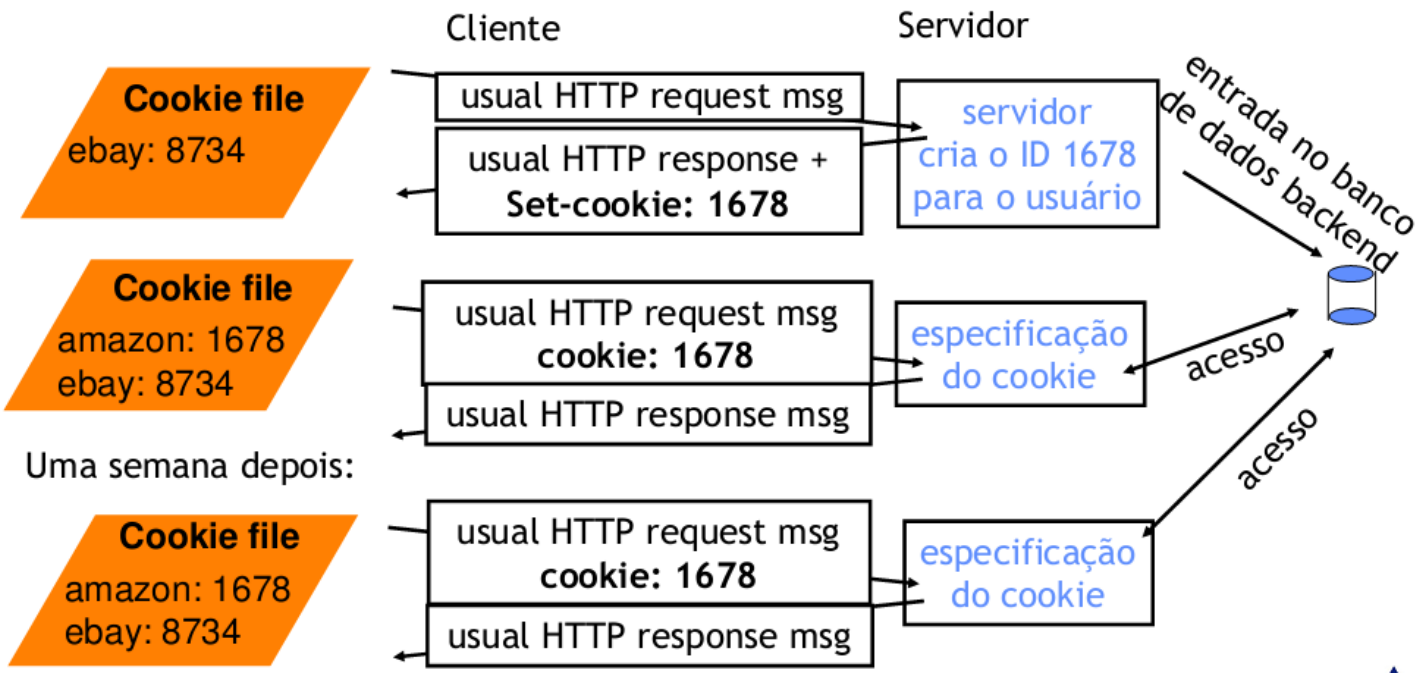
Estado usuário-servidor: cookies

- **Quatro componentes:**
  1. Linha de cabeçalho do cookie na mensagem HTTP response
  2. Linha de cabeçalho de cookie na mensagem HTTP request
  3. Arquivo de cookie mantido no hospedeiro do usuário e manipulado pelo browser do usuário
  4. Banco de dados backend no Web site

Estado usuário-servidor: cookies

- **Exemplo:**
  - Susan acessa a Internet sempre do mesmo PC
  - Ela visita um site específico de e-commerce pela primeira vez
  - Quando a requisição HTTP inicial chega ao site, este cria um ID único e uma entrada no banco de dados *backend* para este ID

Cookies: mantendo “estado”



Cookies

- **O que os cookies podem trazer:**
  - Autorização
  - Cartões de compra
  - Recomendações
  - Estado de sessão do usuário (Web e-mail)

Cookies

- **Cookies e privacidade (por outro lado):**
  - Cookies permitem que sites saibam muito sobre você
  - Você pode fornecer nome e e-mail para os sites
  - Mecanismos de busca usam redirecionamento e cookies para saberem mais sobre você
  - Companhias de propaganda obtêm informações por meio dos sites

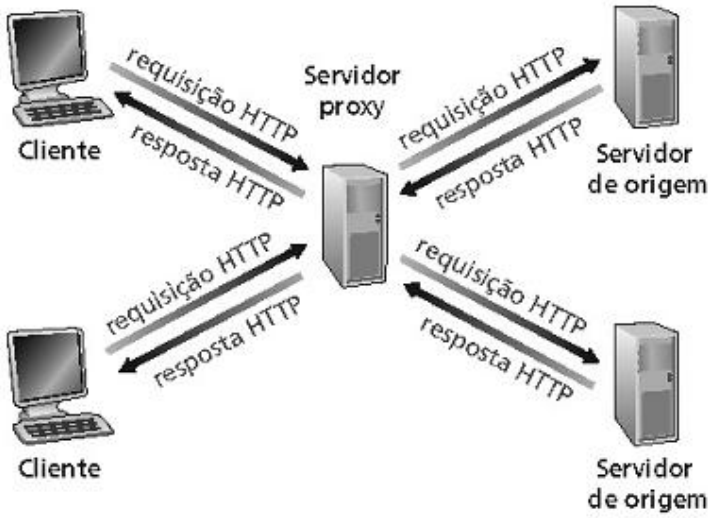
Proxy

Proxy

- **Aplicação intermediária entre cliente HTTP e servidor Web**
- **Permite a diminuição de tráfego no link de internet e pode ser utilizado para filtrar conteúdo**

Proxy server

- **Objetivo: atender o cliente sem envolver o servidor Web originador da informação**



Proxy

- **Usuário especifica o proxy no navegador**
  - Navegador “passa” pelo proxy a cada nova requisição HTTP
  - O proxy consulta seu *cache*
    - **Se o objeto solicitado estiver em cache, será verificado se seu conteúdo está atualizado**
    - **Caso esteja atualizado devolve o objeto em cache ao cliente**
    - **Se o objeto estiver desatualizado ou não estiver no cache, o proxy envia requisição ao servidor**

Proxy Hierárquico

- **É possível que um proxy faça consulta a outro proxy, criando assim uma hierarquia de consulta**
  - Possibilita a distribuição de carga HTTP em uma rede mais complexa (filiais)

Exemplo de Proxy

- **Squid**
  - Suporta HTTP, HTTPs, FTP
  - Analisa cabeçalho e dados da aplicação
    - **Atua no nível 7 do modelo OSI**

Web caches (proxy server)

- **Usuário configura o browser:**
  - Acesso Web é feito por meio de um proxy
- **Cliente envia todos os pedidos HTTP para o Web cache**
  - Se o objeto existe no Web cache: Web cache retorna o objeto
  - Ou o Web cache solicita objeto do servidor original e então envia o objeto ao cliente

Mais sobre Web caching

- **O cache atua tanto no servidor como no cliente**
- **Tipicamente, o cache é instalado pelo ISP (universidade, companhia, ISP residencial)**

Mais sobre Web caching

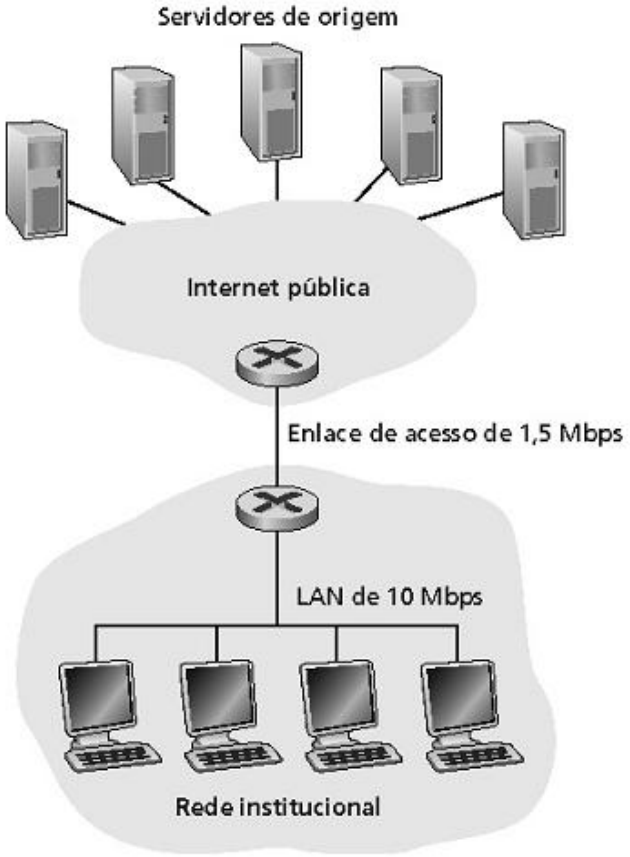
- **Por que Web caching?**
  - Reduz o tempo de resposta para a requisição do cliente
  - Reduz o tráfego num enlace de acesso de uma instituição
  - A densidade de caches na Internet habilita os “fracos” provedores de conteúdo a efetivamente entregarem o conteúdo (mas fazendo P2P file sharing)

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

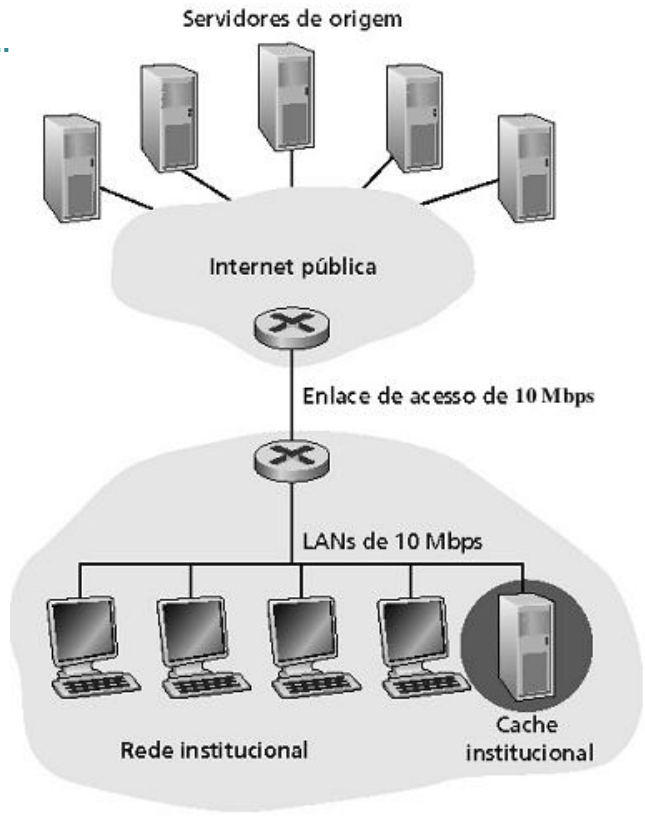
Exemplo de caching

- **Conseqüências:**
  - Utilização da LAN = 15%
  - Utilização do link de acesso = 100%
  - Atraso total = atraso da Internet + atraso de acesso + atraso da LAN = 2 segundos + minutos + milissegundos



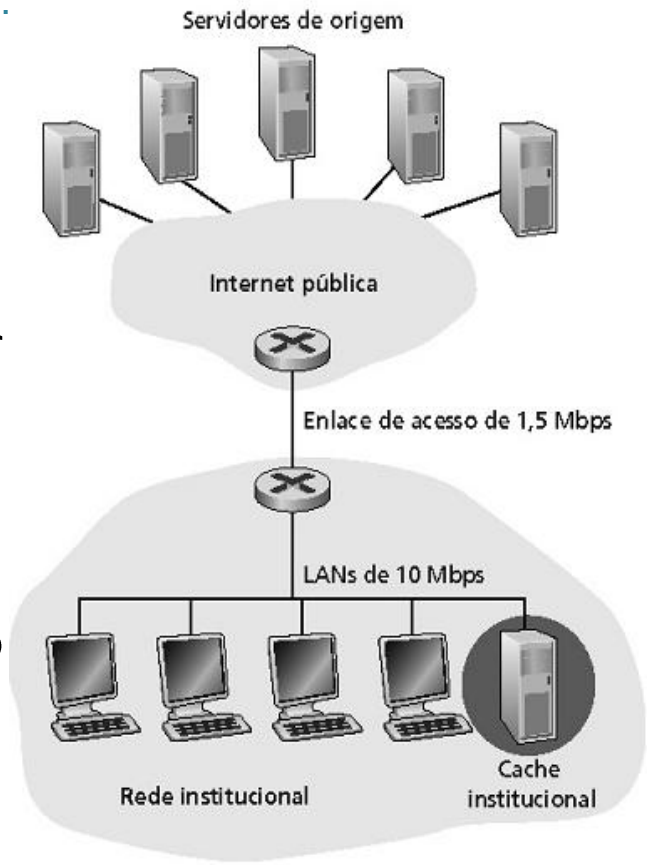
Exemplo de caching

- **Solução possível**
  - Aumentar a taxa do enlace de acesso, para 10 Mbps p.ex.
- **Conseqüências**
  - Utilização da LAN = 15%
  - Utilização do enlace de acesso = 15%
  - Atraso total = atraso da Internet + atraso de acesso + atraso da LAN = 2 segundos + ? ms + ? ms
  - Freqüentemente é um Upgrade caro



Exemplo de caching

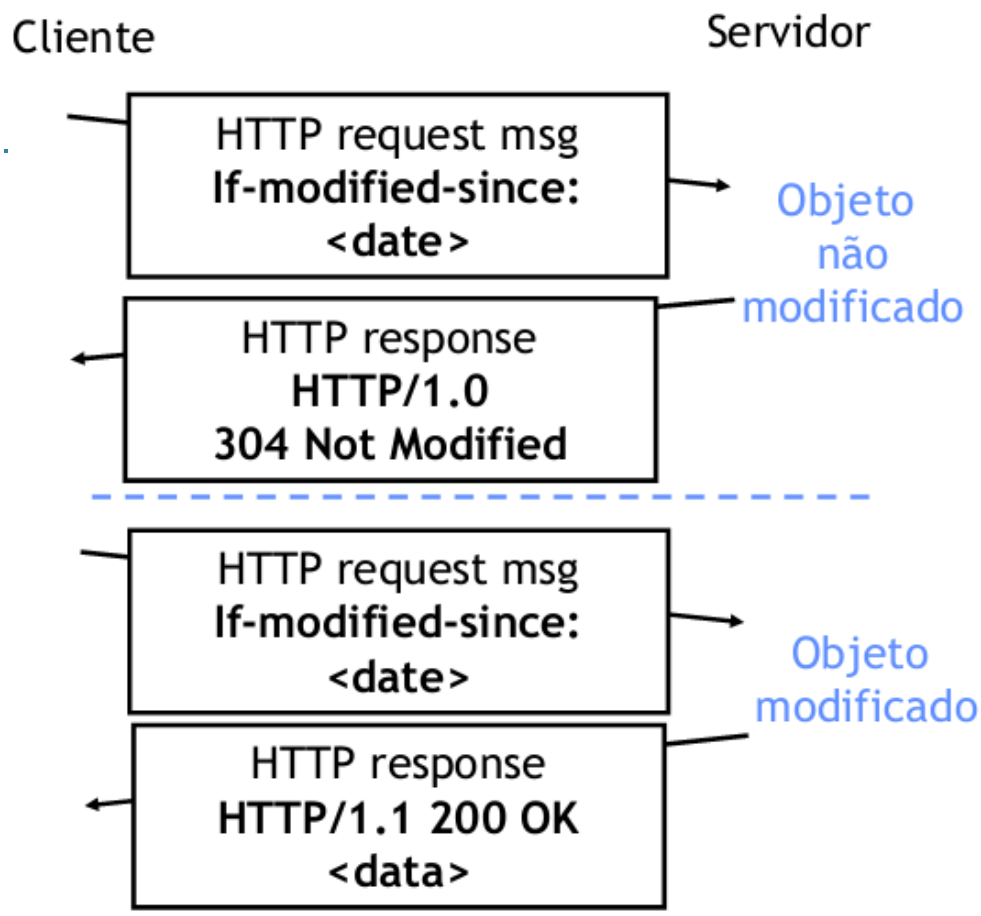
- **Instalação do cache**
  - Suponha taxa de acertos de 40%
- **Conseqüência**
  - 40% das requisições serão satisfeitas quase que imediatamente
  - 60% das requisições serão satisfeitas pelo servidor de origem
  - Utilização do enlace de acesso reduzida para 60%, resultando em atrasos insignificantes (como 10 ms)
  - Média de atraso total = atraso da Internet + atraso de acesso + atraso da LAN = (0.6)*(2.01) segundos + (0,4) *ms < 1,4 segundos



GET condicional

- **Razão:**
  - não enviar objeto se a versão que o cliente já possui está atualizada.
- **Cliente:**
  - especifica data da versão armazenada no pedido HTTP
    - **If-modified-since: <date>**
- **Servidor:**
  - resposta não contém objeto se a cópia é atualizada:
    - **HTTP/1.0 304 Not Modified**

GET condicional



Segurança na Web - HTTPs

Segurança na Web

- **A Web é uma plataforma de aplicações distribuídas que cresce cada vez mais**
  - É relativamente fácil configurar e manipular Servidores Web
  - RISCO:
    - **Muitos usuários não estão cientes dos riscos**

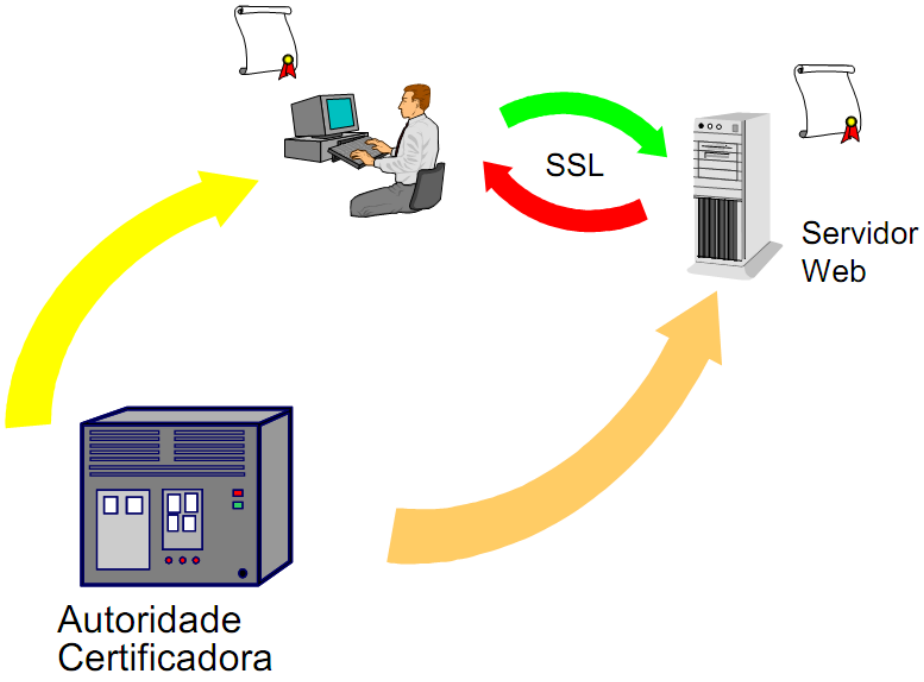
Segurança e HTTP

- **O HTTP fornece esquema mais seguro de autenticação**
  - Elimina a transferência de nome e senha no formato “plain-text”
  - Atualmente as aplicações Web que possuem informação sensível utilizam o HTTPs
    - **HTTP sobre a camada SSL***

SSL – *Secure Sockets Layer*

- **Protocolo desenvolvido pela Netscape**
- **Universalmente aceito na Internet para comunicações que envolvam autenticação e criptografia entre clientes e servidores**
- **Atualmente na versão 3.0**

SSL visão geral



SSL – Visão Geral

Cliente com browser

Loja na Internet com Servidor Web Seguro

1. Cliente conecta com o Lojista
3. O browser usa a chave pública da CA para verificar o certificado do lojista
4. O browser gera uma chave de seção
5. O browser usa a chave pública do Lojista para criptografar a chave de seção e remete junto o seu certificado

2. Lojista envia cópia do seu certificado (e chave pública) para o browser do cliente, indicando que o SSL 2.0 está habilitado
6. O lojista usa a sua chave privada para decodificar a chave de seção e verifica a assinatura digital do cliente

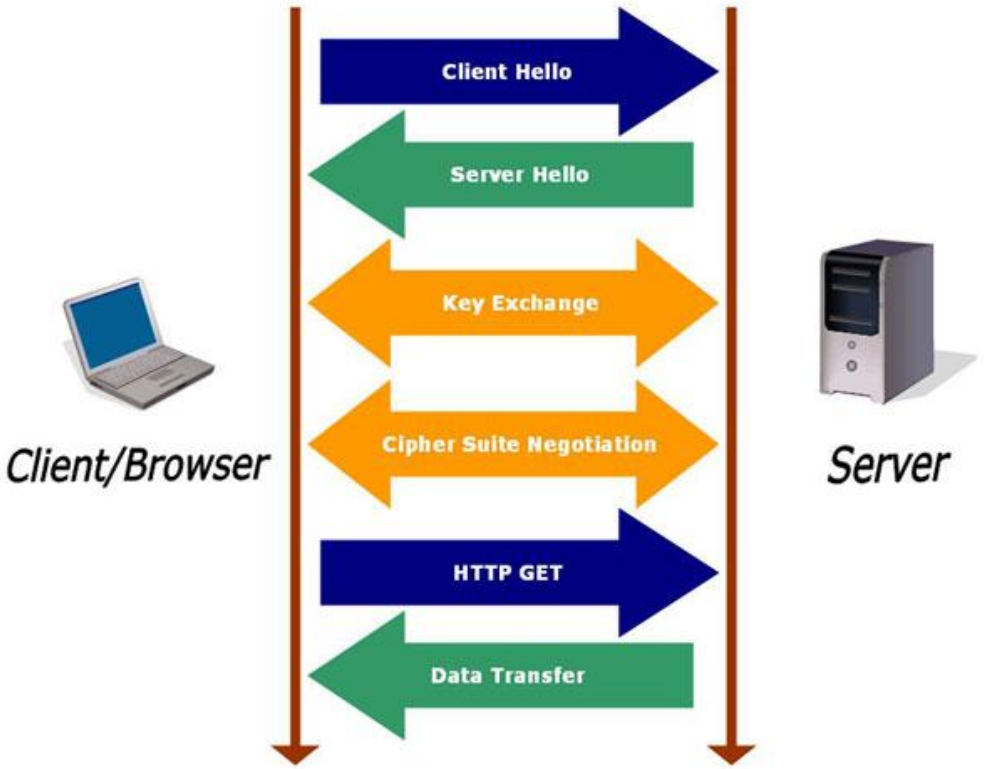
SSL – *Secure Sockets Layer*

- **SSL executa entre o TCP/IP e protocolos de mais alto nível, como o HTTP**
- **Conexão SSL/HTTP**
- **Utiliza a porta TCP/443**
- **Identificada no navegador pelo prefixo HTTPs**

SSL – *Secure Sockets Layer*

- **Permite comunicação segura (criptografada) entre as partes**
- **Permite que servidores se autentiquem para clientes**
- **Permite que clientes se autentiquem para servidores**

HandShake SSL



http://blog.sprocketnetworks.com/?Tag=SSL