Redes de Computadores

Camada de Aplicação

Prof. Renê Pomilio de Oliveira

Slides baseados nas aulas da Profa. Dra. Kalinka Castelo Branco (ICMC/USP)
Prof. Dr. Anderson Chaves Carniel (UTFPR)

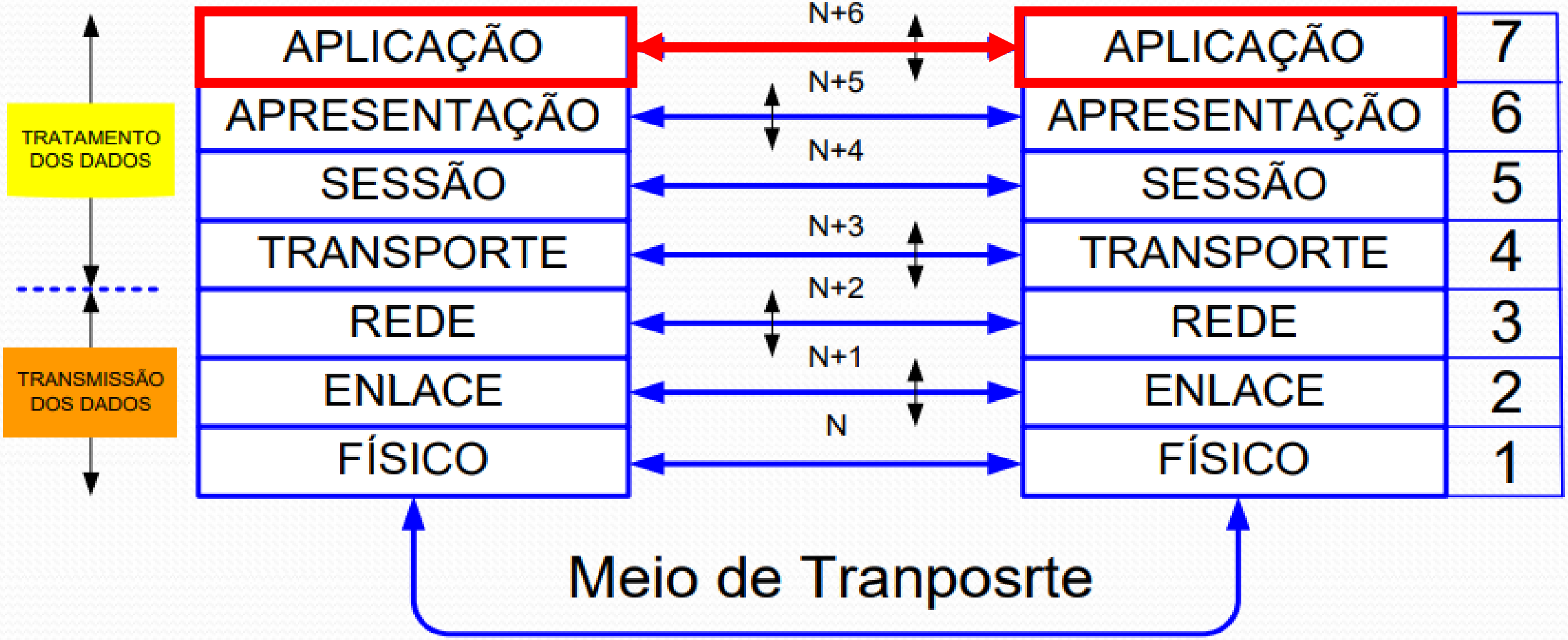
Câmpus Dois Vizinhos

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS DOIS VIZINHOS

MINISTÉRIO DA EDUCAÇÃO

BRASIL
GOVERNO FEDERAL

Entidades da Camada

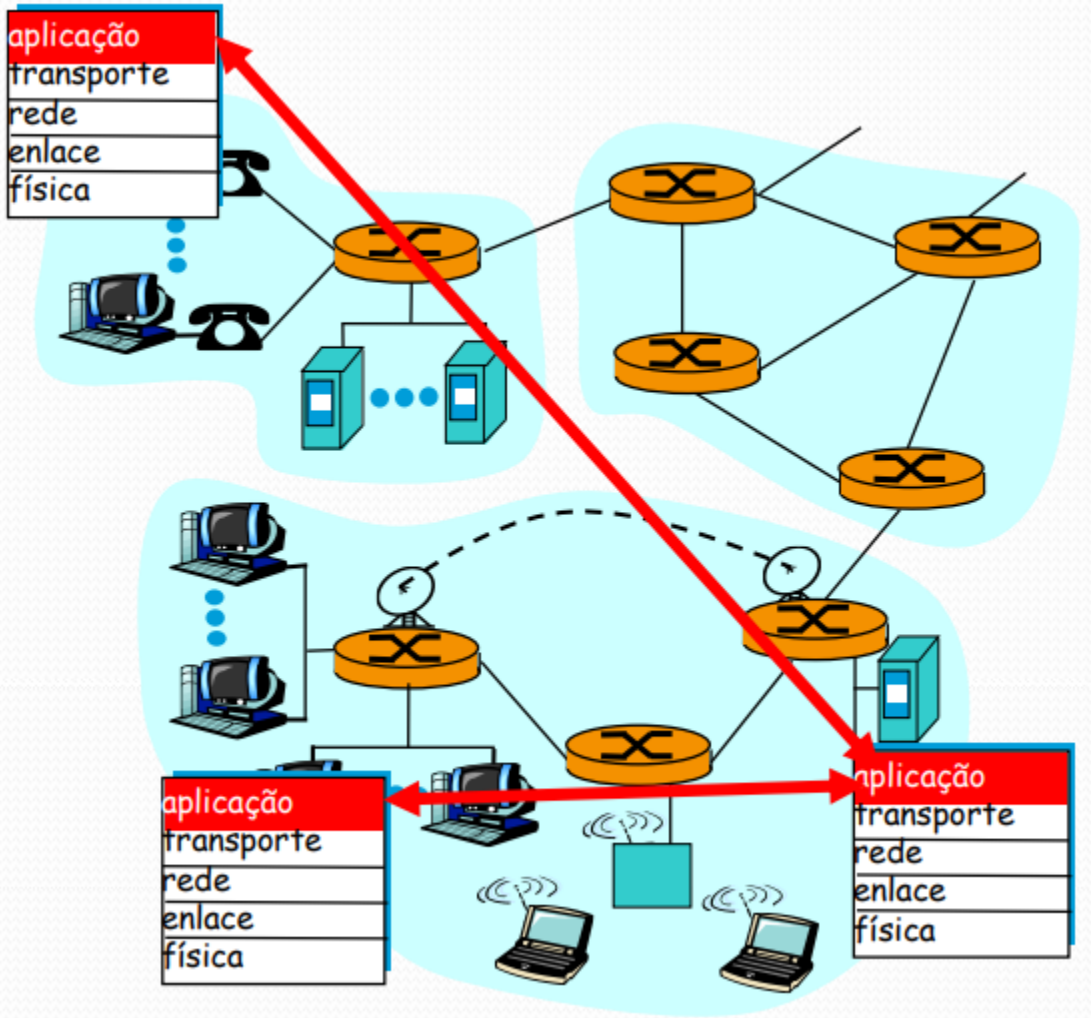


Camada de Aplicação – aplicações

- E-mail
- Web
- Instant messaging
- Login remoto
- Compartilhamento de arquivos P2P
- Jogos de rede multi-usuários
- Vídeo-clipes armazenados
- Voz sobre IP
- Vídeo conferência em tempo real
- Computação paralela em larga escala

Criando uma aplicação de rede

- Programas que:
  - Executam em diferentes sistemas finais
  - Comunicam-se através da rede
  - p.ex., Web: servidor Web se comunica com o navegador
- Programas não relacionados ao núcleo da rede
  - Dispositivos do núcleo da rede não executam aplicações de usuários
  - Aplicações nos sistemas finais permite rápido desenvolvimento e disseminação

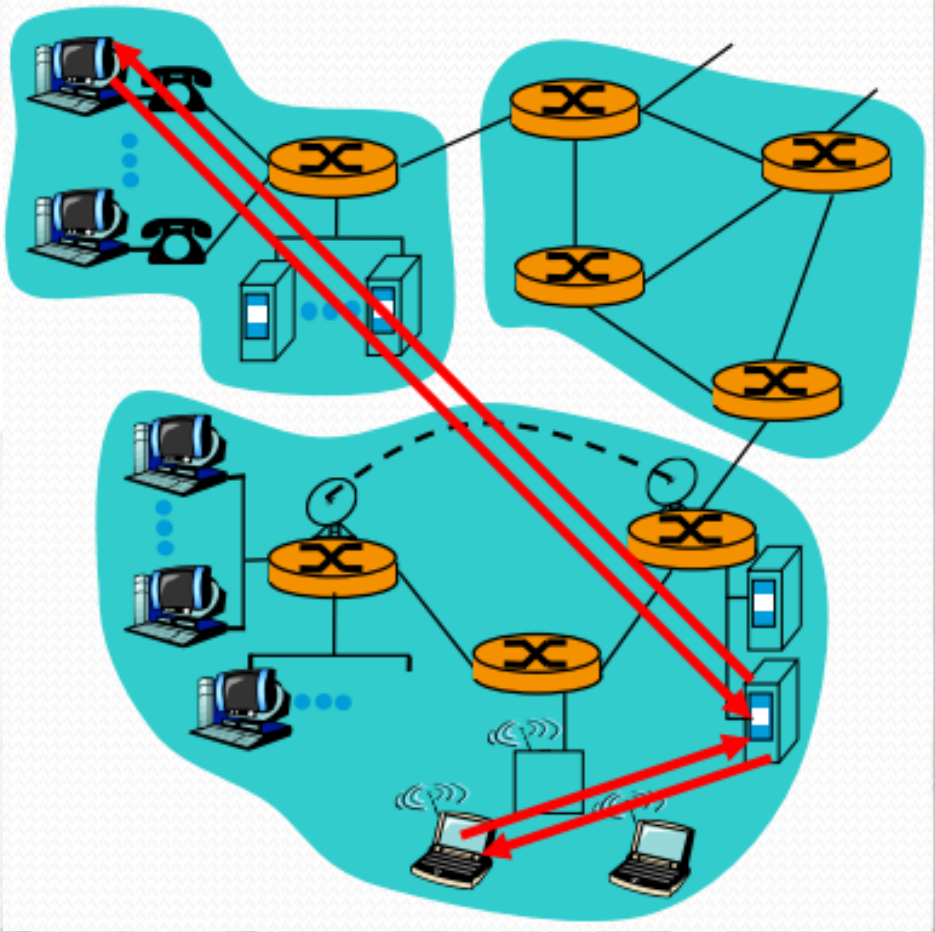


Arquiteturas das aplicações

- Cliente-servidor
- Peer-to-peer (P2P)
- Híbrido de cliente-servidor e P2P

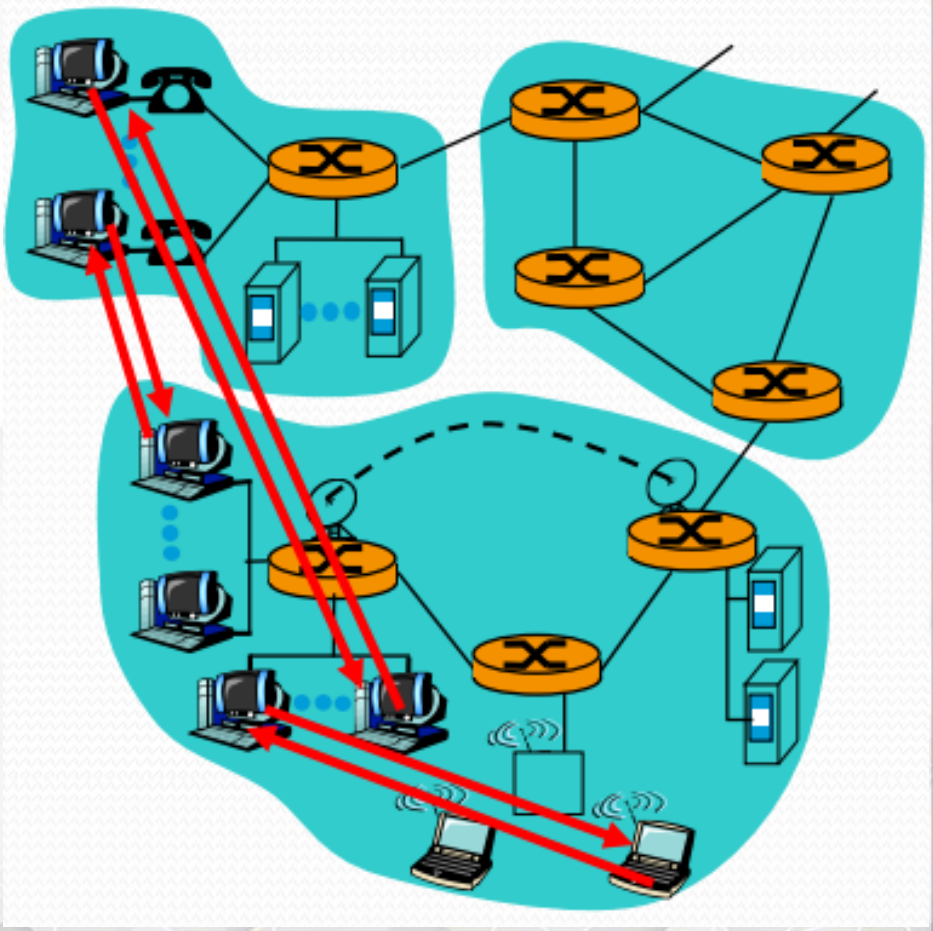
Arquitetura cliente -servidor

- Servidor:
  - ✓ Sempre ligado
  - ✓ Endereço IP permanente
  - ✓ Escalabilidade com *server farms*
- Cliente:
  - ✓ *Comunica-se com o servidor*
  - ✓ *Pode estar conectado intermitentemente*
  - ✓ *Pode ter endereços IP dinâmicos*
  - ✓ *Não se comunica diretamente com outros clientes*



Arquitetura P2P pura

- Não há servidor sempre ligado
  - Sistemas finais arbitrários se comunicam diretamente
  - Pares estão conectados intermitentemente e mudam endereços IP
  - Exemplo: Gnutella, KaZaA, uTorrent e Skype



Altamente escalável Porém, difícil de gerenciar

Híbrido de cliente-servidor e P2P

- Napster
  - Transferência de arquivos P2P
  - Busca de arquivos centralizada:
  - Pares registram conteúdo no servidor central
  - Pares consultam o mesmo servidor central para localizar conteúdo
- Instant messaging
  - Conversa entre usuários P2P
  - Localização e detecção de presença centralizadas:
  - Usuários registram o seu endereço IP junto ao servidor central quando ficam online
  - Usuários consultam o servidor central para encontrar endereços IP dos contatos

Processos em comunicação

- Processo: programa que executa num hospedeiro
  - processos no mesmo hospedeiro se comunicam usando comunicação entre processos definida pelo sistema operacional (SO)
  - processos em hospedeiros distintos se comunicam trocando mensagens através da rede

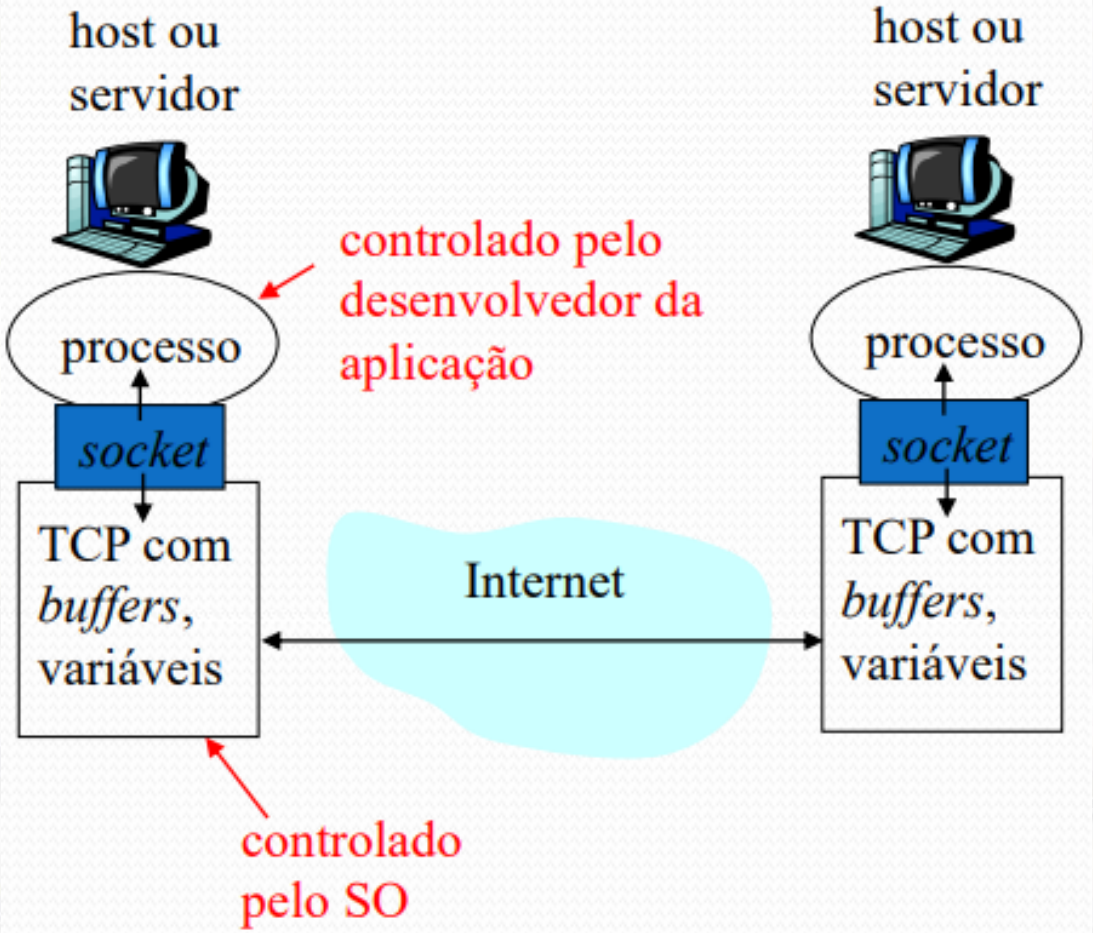
➢ Processo cliente: processo que inicia a comunicação

➢ Processo servidor: processo que espera para ser contatado

❑ **Nota**: aplicações com arquiteturas P2P possuem processos clientes e processos servidores

Sockets

- Os processos enviam/ recebem mensagens para/dos seus **sockets**
- Um socket é análogo a uma porta
  - Processo transmissor envia a mensagem através da porta
  - O processo transmissor assume a existência da infra-estrutura de transporte no outro lado da porta que faz com que a mensagem chegue ao socket do processo receptor



Os protocolos da camada de aplicação definem

- Tipos de mensagens trocadas, ex. mensagens de pedido e resposta
- Sintaxe dos tipos das mensagens: campos presentes nas mensagens e como são identificados
- Semântica dos campos, i.e., significado da informação nos campos
- Regras para quando os processos enviam e respondem às mensagens

- Protocolos de domínio público:
  - definidos em RFCs
  - ex, HTTP e SMTP
- Protocolos proprietários:
  - Ex., KaZaA, Skype

De que serviço de transporte uma aplicação precisa?

- Perda de dados
  - algumas apls (p.ex. áudio) podem tolerar algumas perdas
  - outras (p.ex., transf. de arquivos, telnet) requerem transferência 100% confiável
- Temporização
  - algumas apls (p.ex., telefonia Internet, jogos interativos) requerem baixo retardo para serem “viáveis”
- Largura de banda:
  - algumas apls (p.ex., multimídia) requerem quantia mínima de banda para serem “viáveis”
  - outras apls (“apls elásticas”) conseguem usar qualquer quantia de banda disponível

Web e HTTP

- Primeiro algum jargão
  - Páginas Web consistem de objetos
  - Objeto pode ser um arquivo HTML, uma imagem JPEG, um arquivo de áudio,…
  - Páginas Web consistem de um arquivo HTML base que inclui vários objetos referenciados
  - Cada objeto é endereçável por uma URL
  - Exemplo de URL:
    - https://www.tecmundo.com.br/produto/147830-eletronicos-olho-black-friday.htm

nome do hospedeiro

nome do caminho

Protocolo HTTP

- *HTTP: hypertext transfer protocol*
  - protocolo da camada de aplicação da Web
  - modelo cliente/servidor
    - cliente: browser que pede, recebe, "visualiza" objetos Web
    - servidor: servidor Web envia objetos em resposta a pedidos
  - HTTP 1.0: RFC 1945
  - HTTP 1.1: RFC 2068

Mais sobre o protocolo HTTP

- Usa serviço de transporte TCP:
  - cliente inicia conexão TCP (cria socket) ao 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

Mais sobre o protocolo HTTP

- 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

Conexões HTTP

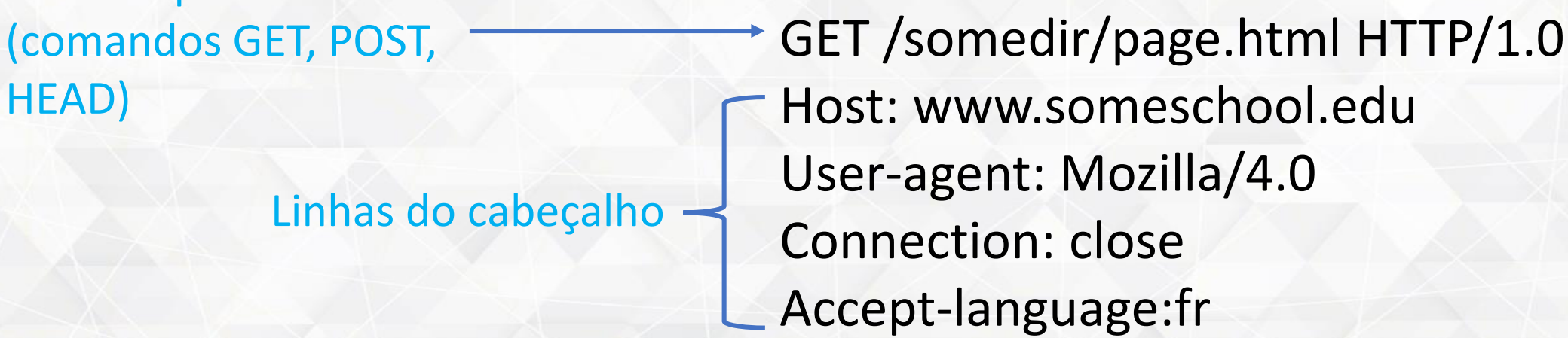
- HTTP não persistente
  - No máximo um objeto é enviado numa conexão TCP
  - HTTP/1.0 usa o HTTP não persistente
- HTTP persistente
  - Múltiplos objetos podem ser enviados sobre uma única conexão TCP entre cliente e servidor
  - HTTP/1.1 usa conexões persistentes no seu modo default

HTTP persistente

- Persistente sem paralelismo
  - Cliente emite novas requisições apenas quando a resposta anterior for recebida
- Persistente com paralelismo
  - É usado o padrão HTTP/1.1;
  - Cliente envia a requisição assim que encontra um objeto referenciado

Formato de mensagem HTTP: pedido (get)

- Dois tipos de mensagem HTTP: pedido (require), resposta (response)
- mensagem de pedido HTTP:
  - ASCII (formato legível por pessoas)



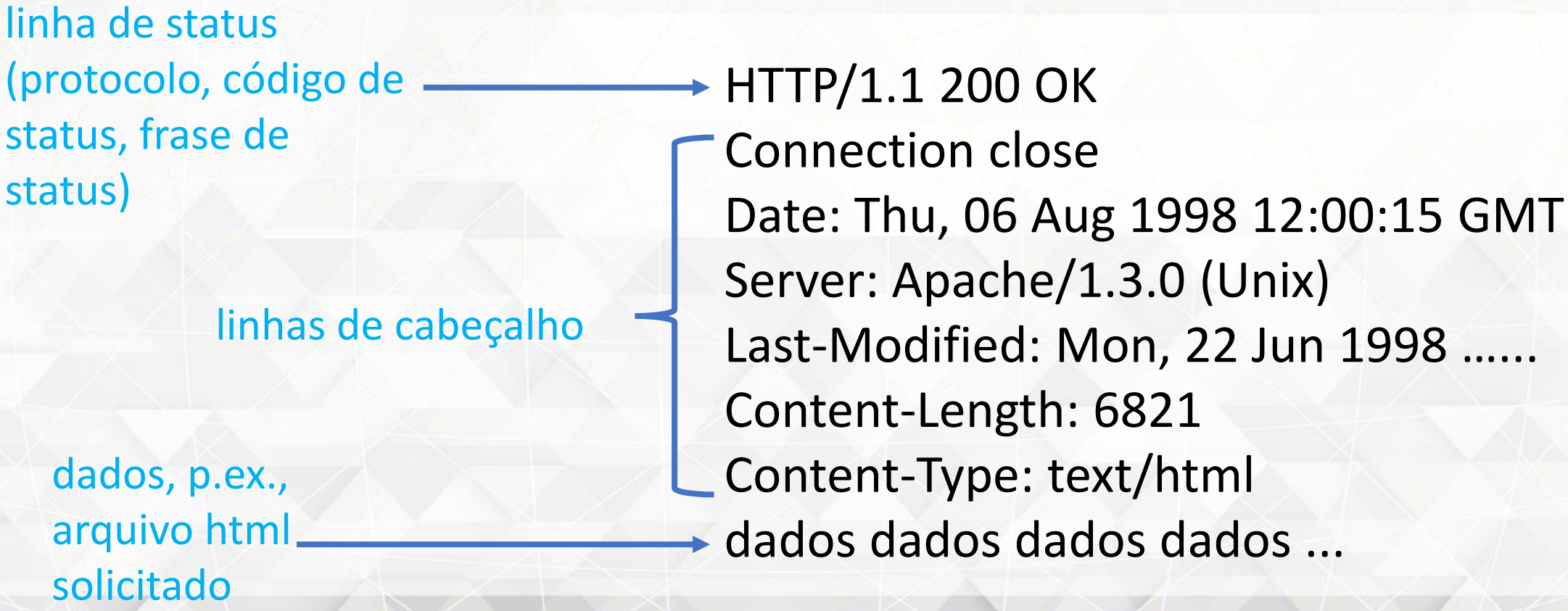
Tipos de métodos

- HTTP/1.0
  - GET
  - POST
  - HEAD
    - Pede para o servidor não enviar o objeto requerido junto com a resposta (usado p/ debugging)
- HTTP/1.1
  - GET, POST, HEAD
  - PUT
    - Upload de arquivo contido no corpo da mensagem para o caminho especificado no campo URL
  - DELETE
    - Exclui arquivo especificado no campo URL

Enviando conteúdo de formulário

- Método POST :
  - Conteúdo é enviado para o servidor no corpo da mensagem
- Método GET:
  - Conteúdo é enviado para o servidor no campo URL:

Formato de mensagem HTTP: resposta



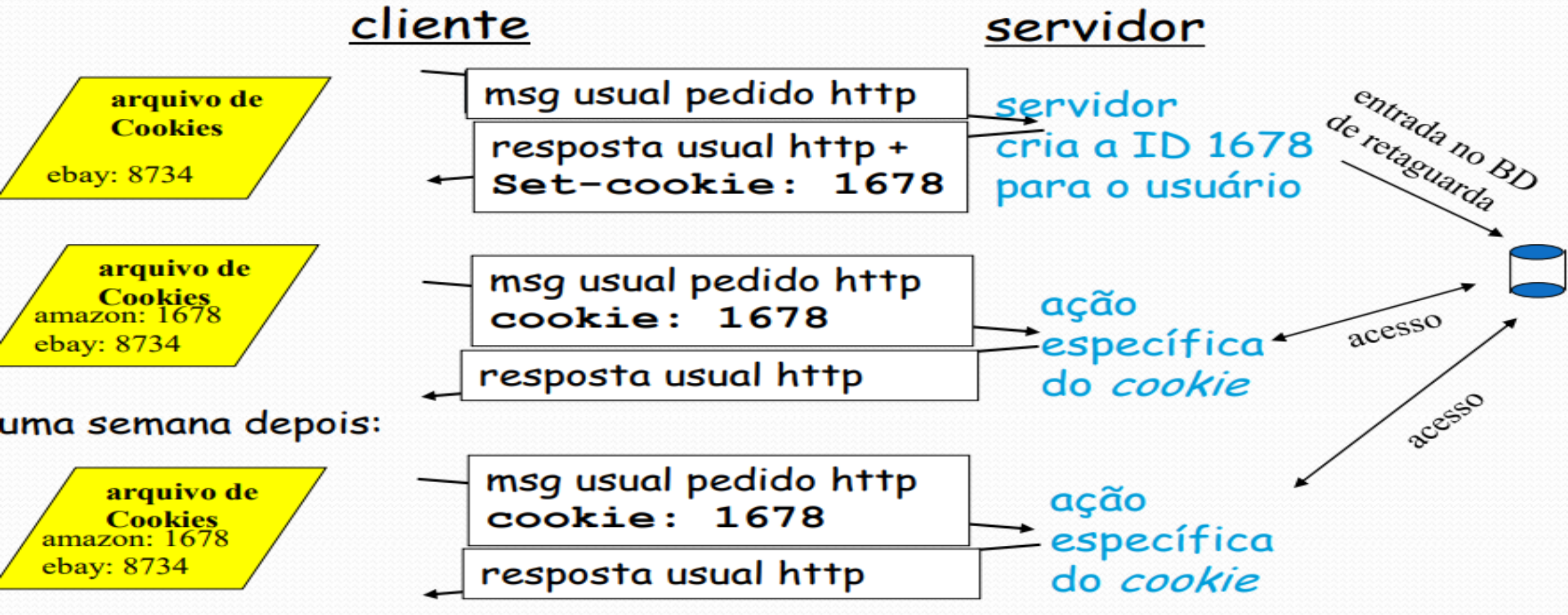
códigos de status da resposta HTTP

- Na primeira linha da mensagem de resposta servidor->cliente. Alguns códigos típicos:
- 200 OK
  - sucesso, objeto pedido segue mais adiante nesta mensagem
- 301 Moved Permanently
  - objeto pedido mudou de lugar, nova localização especificado mais adiante nesta mensagem (Location:)
- 400 Bad Request
  - mensagem de pedido não entendida pelo servidor

códigos de status da resposta HTTP

- 404 Not Found
  - documento pedido não se encontra neste servidor
- 505 HTTP Version Not Supported
  - versão de http do pedido não usada por este servidor

Cookies: manutenção do “estado”



Cookies (continuação)

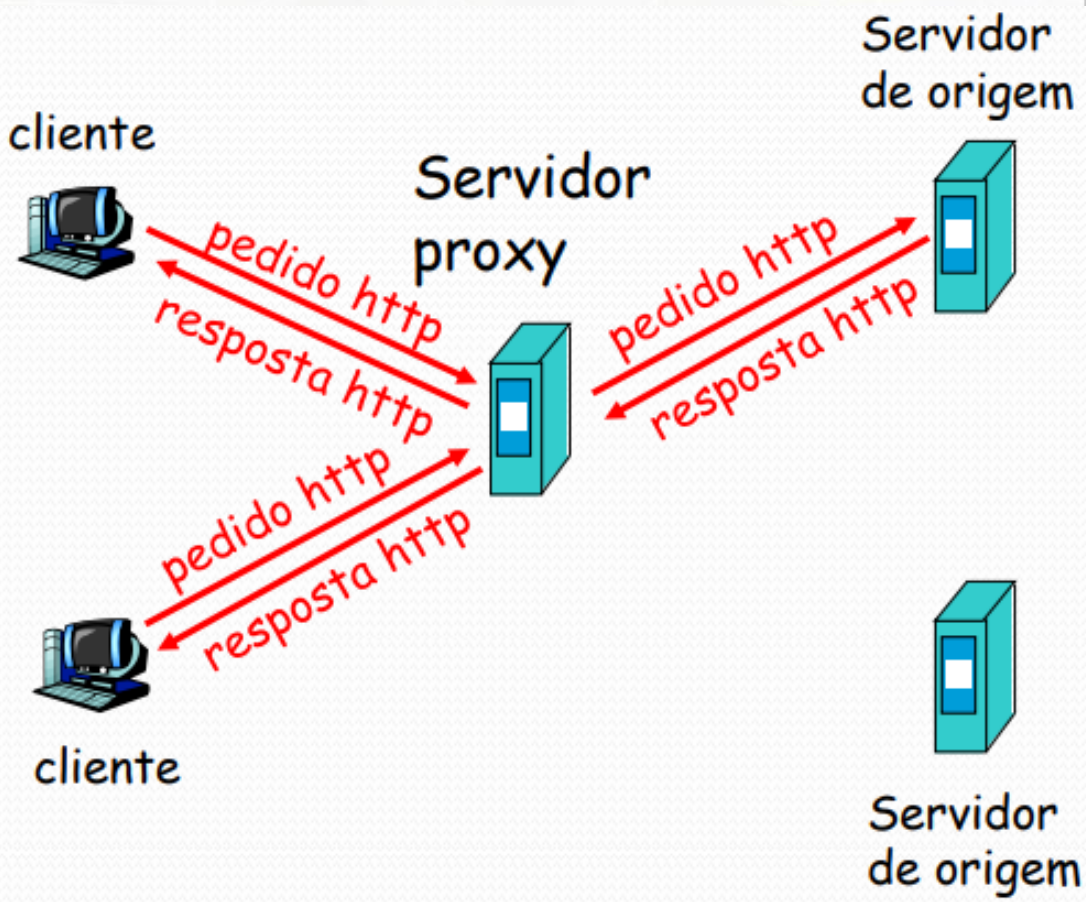
- O que os cookies podem obter:
  - autorização
  - carrinhos de compra
  - sugestões
  - estado da sessão do usuário (Webmail)

❑ Nota: Cookies e privacidade:

1. cookies permitem que os sites os aprendam muito sobre você
2. você pode fornecer nome e e-mail para os sites
3. mecanismos de busca usam redirecionamento e cookies para aprender ainda mais
4. agências de propaganda obtêm perfil a partir dos sites visitados

Cache Web (servidor proxy)

- Meta: atender pedido do cliente sem envolver servidor de origem
  - usuário configura browser: acessos Web via proxy
  - cliente envia todos pedidos HTTP ao proxy
    - se objeto no cache do proxy, este o devolve imediatamente na resposta HTTP
    - senão, solicita objeto do servidor de origem, depois devolve resposta HTTP ao cliente



Mais sobre Caches Web

- Cache atua tanto como cliente quanto como servidor
- Tipicamente o cache é instalado por um ISP (universidade, empresa, ISP residencial)

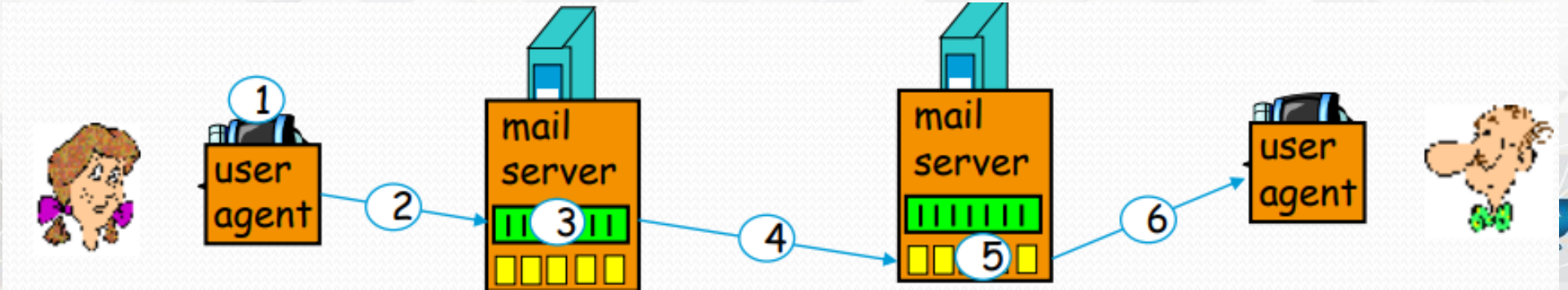
- Para que fazer cache Web?
  - Redução do tempo de resposta para os pedidos do cliente
  - Redução do tráfego no canal de acesso de uma instituição
  - A Internet cheia de caches permitem que provedores de conteúdo “pobres” efetivamente forneçam conteúdo!

Correio Eletrônico: SMTP

- usa TCP para a transferência confiável de msgs do correio do cliente ao servidor, porta 25
- transferência direta: servidor remetente ao servidor receptor
- três fases da transferência
  - handshaking (cumprimento)
  - transferência das mensagens
  - Encerramento

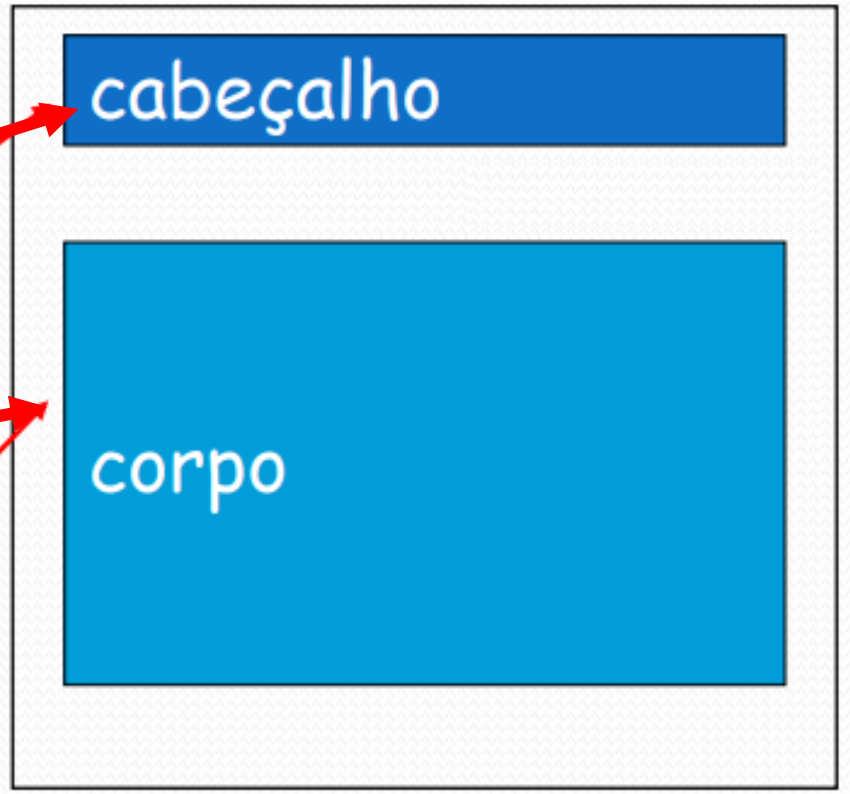
Cenário: Alice envia uma msg para Bob

1) Alice usa o UA para compor uma mensagem “para” bob@someschool.edu
2) O UA de Alice envia a mensagem para o seu servidor de correio; a mensagem é colocada na fila de mensagens
3) O lado cliente do SMTP abre uma conexão TCP com o servidor de correio de Bob
4) O cliente SMTP envia a mensagem de Alice através da conexão TCP
5) O servidor de correio de Bob coloca a mensagem na caixa de entrada de Bob
6) Bob chama o seu UA para ler a mensagem



Formato de uma mensagem

- SMTP: protocolo para trocar msgs de correio
- RFC 822: padrão para formato de mensagem de texto:
  - linhas de cabeçalho, p.ex.,
    - To:
    - From:
    - Subject:
  - Corpo
    - a “mensagem”, somente de caracteres ASCII



DNS: Domain Name System

- Pessoas: muitos identificadores:
  - CPF, nome, no. da Identidade
- hospedeiros, roteadores Internet :
  - endereço IP (32 bit) - usado p/ endereçar datagramas
  - “nome”, ex., jambo.ic.uff.br - usado por gente

- Domain Name System:
  - base de dados distribuída implementada na hierarquia de muitos servidores de nomes
  - protocolo de camada de aplicação permite que hospedeiros, roteadores, servidores de nomes se comuniquem para resolver nomes (tradução endereço/nome)
  - complexidade na borda da rede

DNS

- Roda sobre UDP e usa a porta 53
- Especificado nas RFCs 1034 e 1035 e atualizado em outras RFCs.

- Outros serviços:
  - apelidos para hospedeiros (aliasing)
  - apelido para o servidor de mails
  - distribuição da carga

Servidores de nomes DNS

- Por que não centralizar o DNS?
  - ponto único de falha
  - volume de tráfego
  - base de dados centralizada e distante
  - manutenção (da BD)
  - Não é escalável!

- Nenhum servidor mantém todos os mapeamento nome-para-endereço IP
- servidor de nomes local:
  - cada provedor, empresa tem servidor de nomes local (default)
  - pedido DNS de hospedeiro vai primeiro ao servidor de nomes local
- servidor de nomes oficial:
  - p/ hospedeiro: guarda nome, endereço IP dele
  - pode realizar tradução nome/endereço para este nome

Programação com sockets

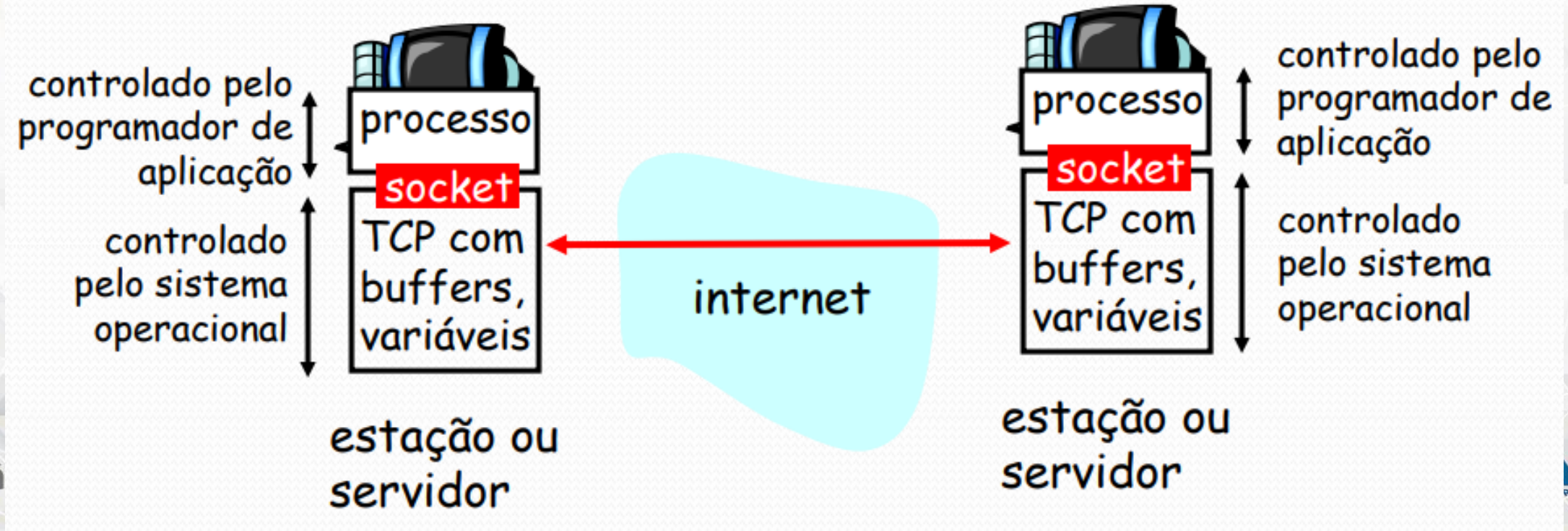
- Meta: aprender a construir aplicações cliente/servidor que se comunicam usando sockets
- API Sockets
  - apareceu no BSD4.1 UNIX em 1981
  - são explicitamente criados, usados e liberados por aplicações
  - paradigma cliente/servidor
  - dois tipos de serviço de transporte via API Sockets
    - datagrama não confiável
    - fluxo de bytes, confiável

Programação com sockets

- socket: uma interface (uma “porta”), local ao hospedeiro, criada por e pertencente à aplicação, e controlado pelo SO, através da qual um processo de aplicação pode tanto enviar como receber mensagens para/de outro processo de aplicação (remoto ou local)

Programação com sockets usando TCP

- Socket: uma porta entre o processo de aplicação e um protocolo de transporte fim-a-fim (UDP ou TCP)
- Serviço TCP: transferência confiável de bytes de um processo para outro



Programação com sockets usando TCP

- Cliente deve contactar servidor:
  - processo servidor deve antes estar em execução
  - servidor deve antes ter criado socket (porta) que aguarda contato do cliente
- Cliente contacta servidor para:
  - criar socket TCP local ao cliente
  - especificar endereço IP, número de porta do processo servidor
  - Quando cliente cria socket: TCP do cliente estabelece conexão com TCP do servidor
  - Quando contatado pelo cliente, o TCP do servidor cria socket novo para que o processo servidor possa se comunicar com o cliente

ponto de vista da aplicação:

TCP provê transferência confiável, ordenada de bytes (“tubo”) entre cliente e servidor