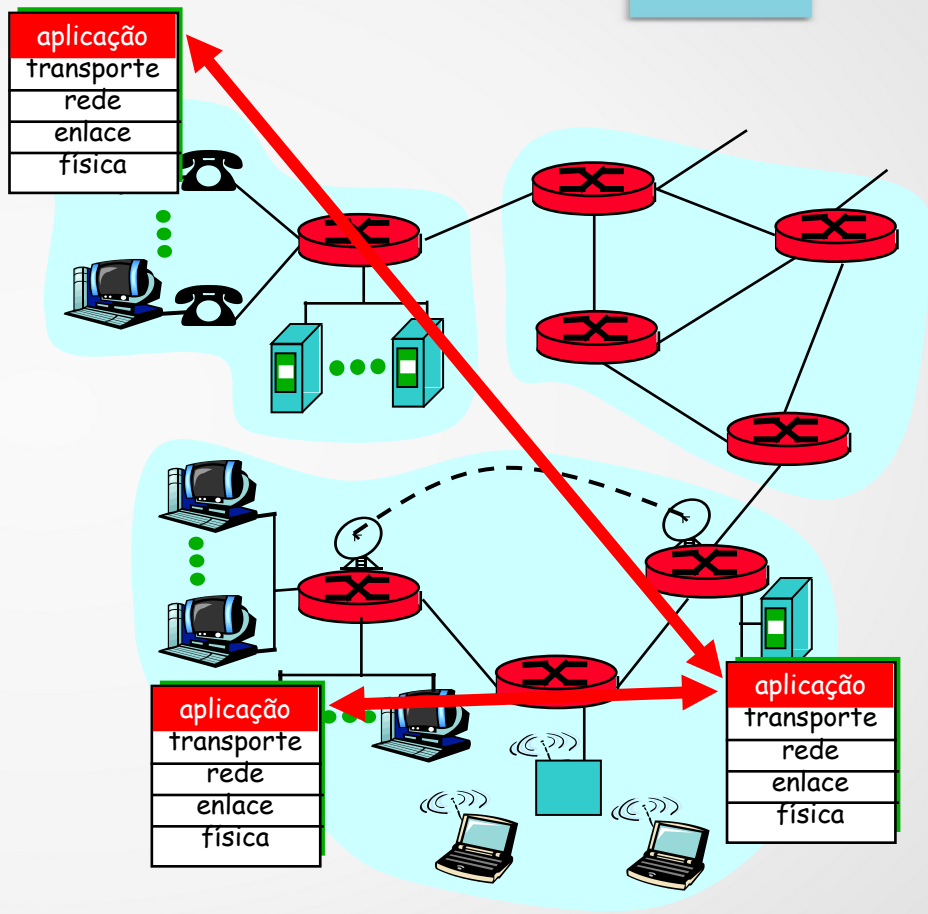
Camada de Aplicação



Camada de Aplicação

Objetivos:

- Entender o aspecto conceitual e de implementação de protocolos de aplicação para redes
  - paradigma cliente-servidor
  - paradigma peer-to-peer (P2P)
  - modelos de serviço da camada de transporte

Aplicações e Protocolo de Aplicação

Aplicação e protocolo de aplicação são elementos distintos!!!

Aplicações são processos distribuídos que se comunicam

- Executadas em sistemas finais como softwares
- trocam mensagens, mesmo entre hosts diferentes
- Ex: cliente e-mail, ftp, browser

Protocolo de aplicação

- É utilizado pela aplicação e define como as mensagens são trocadas, sintaxe das mensagens, semântica dos campos, regras e ações tomadas em casa situação, etc.
- usam serviços de comunicação das camadas inferiores

Aplicações implementam o protocolo da camada de aplicação.

- Exemplos: http: browser; imap: leitor de e-mail; rtsp: media player; XMPP: whatsapp; DASH: netflix

Paradigma Cliente-Servidor

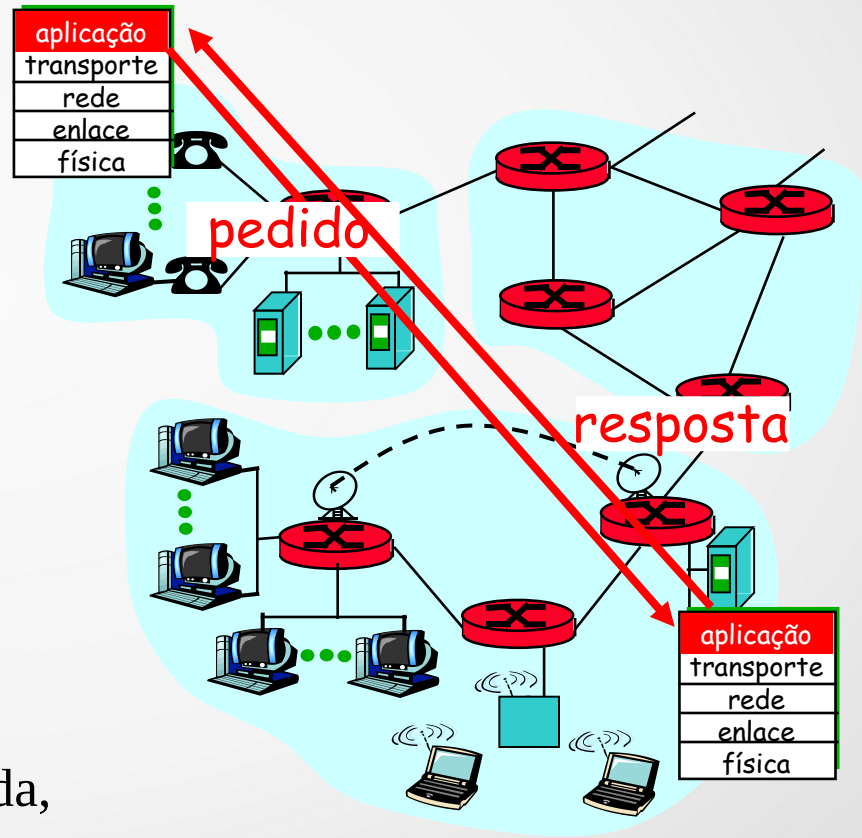
Aplicações de rede têm duas partes básicas: *cliente* e *servidor*

CLIENTE:

- Inicia comunicação com o servidor
- Requisita serviços ao servidor
- Nem sempre está online
- Em geral, usa endereçamento dinâmico
- Ex: web: browser; e-mail: leitor de correio

SERVIDOR:

- Provê os serviços solicitados pelo cliente.
- Em geral, está sempre online
- Em geral, usa ip fixo.
- Ex: web server envia a página web solicitada, servidor de e-mail envia as mensagens, etc.



Paradigma P2P (peer to peer)

- Uma aplicação P2P não precisa de um servidor sempre ligado, havendo uma dependência mínima (se houver).
- Sistemas finais arbitrários se comunicam de modo direto e intermitente, e em geral, são clientes em conexões residenciais (IP dinâmico).
- Possui alta escalabilidade, já que a carga é distribuída por diversos hosts espalhados. Novos nós irão trazer mais demanda de serviço, mas também mais capacidade de serviço.
- Porém, depende de um número considerável de usuários ativos para permitir um bom desempenho.

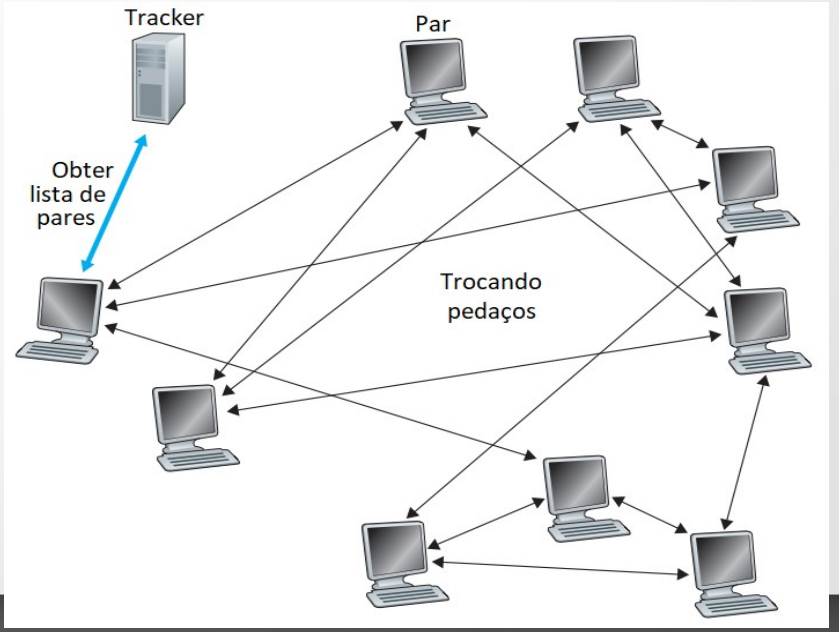
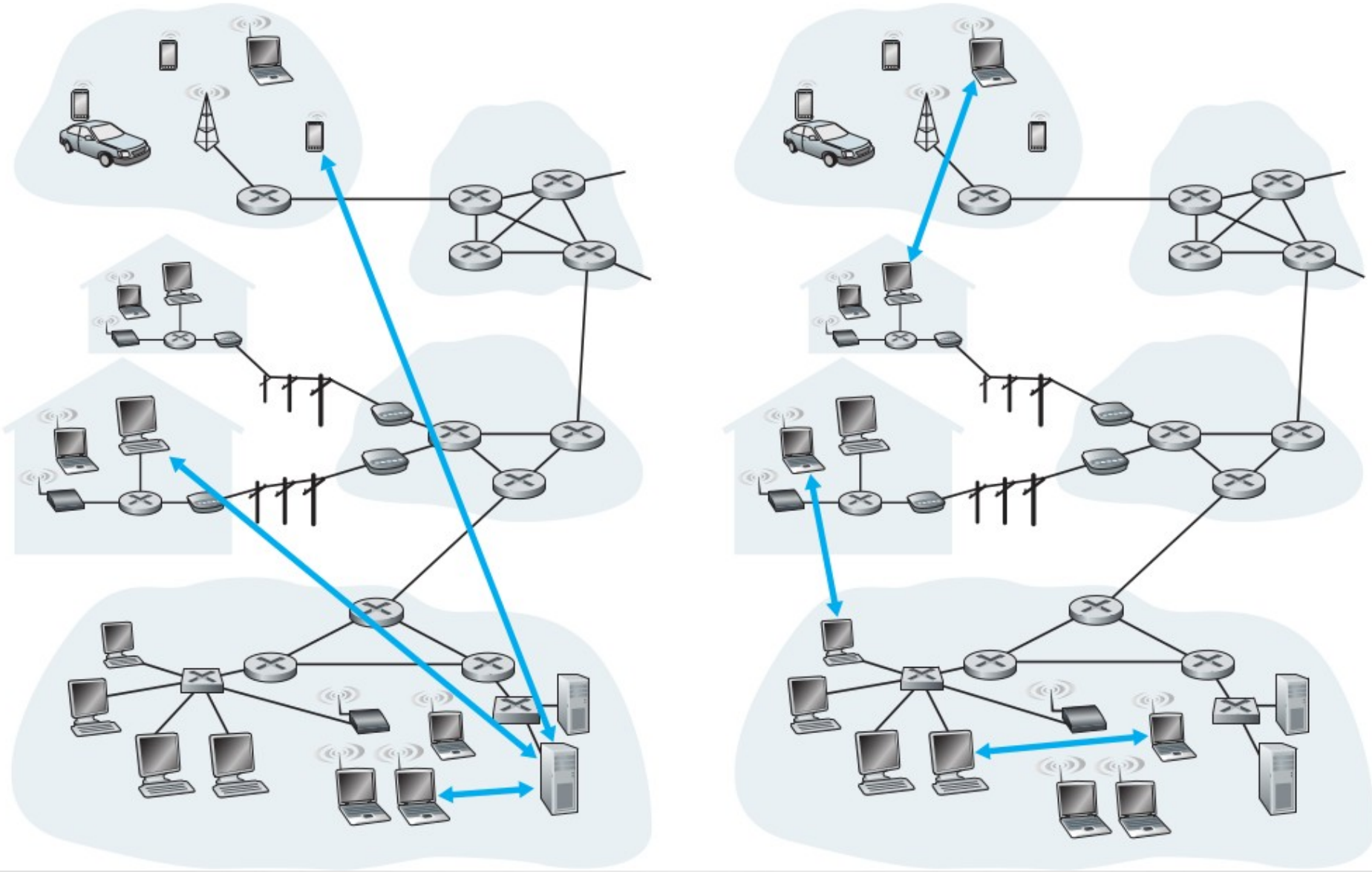
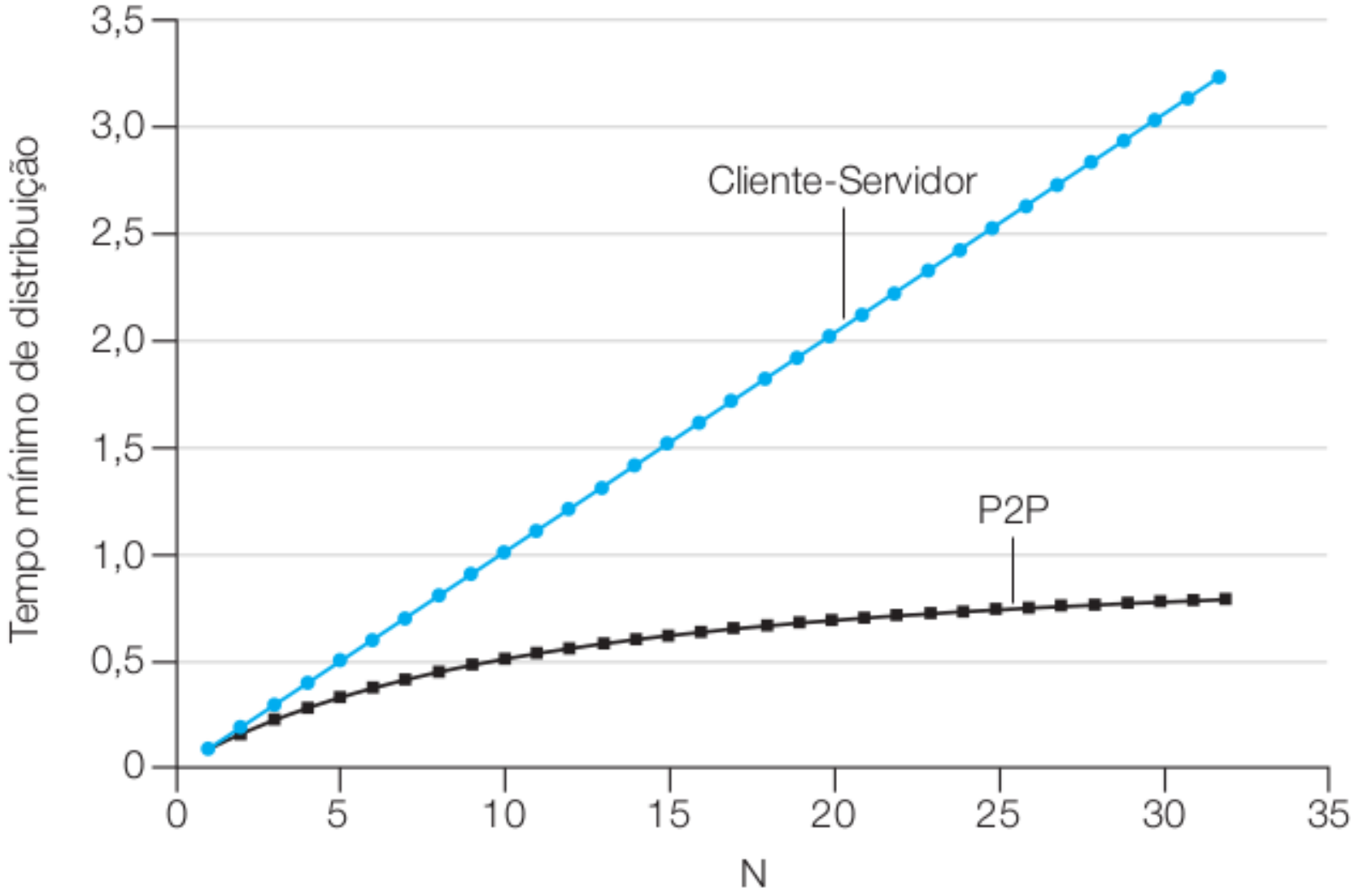


FIGURA 2.2 (A) ARQUITETURA CLIENTE-SERVIDOR; (B) ARQUITETURA P2P



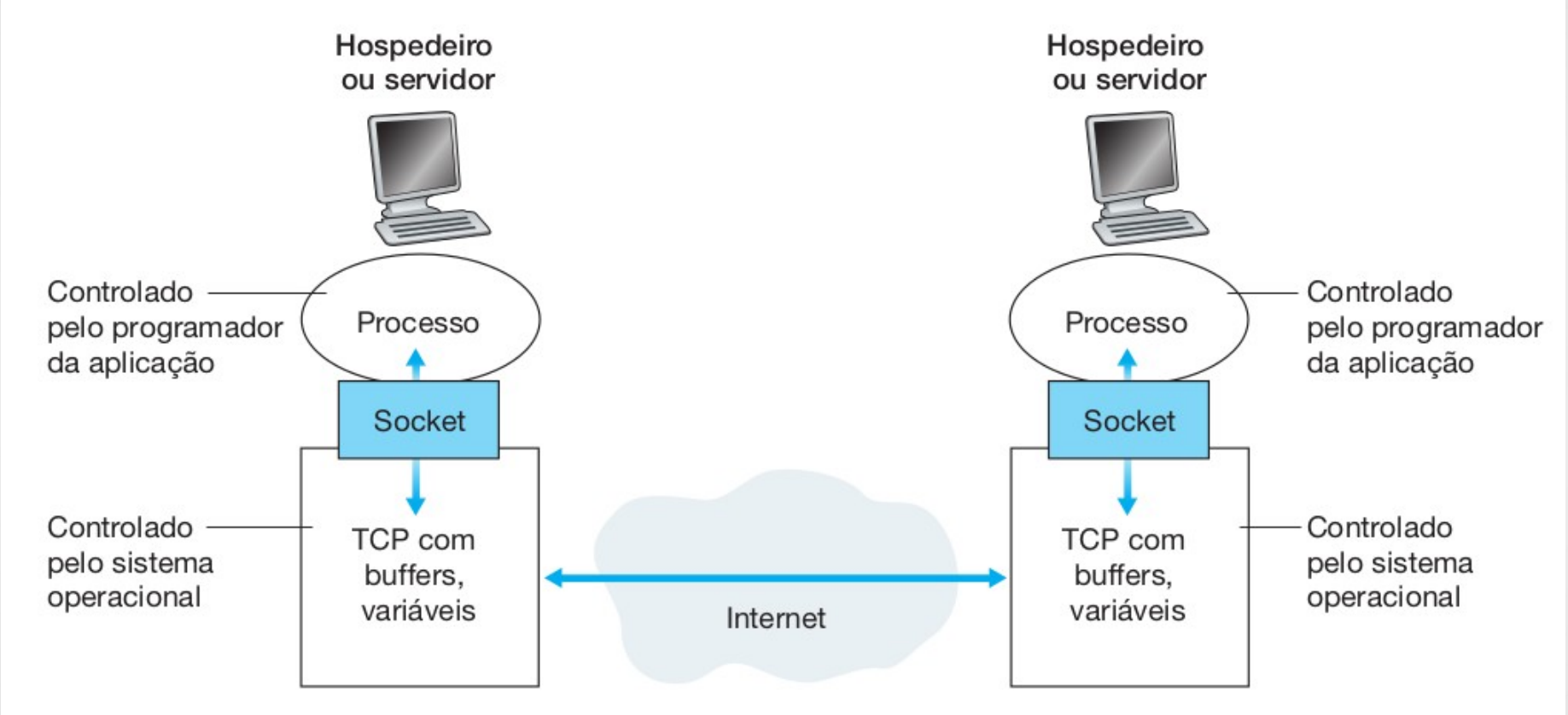
TEMPO DE DISTRIBUIÇÃO PARA ARQUITETURAS P2P E CLIENTE-SERVIDOR



Sockets

- Processos enviam/recebem mensagens de/para um socket
- socket: é a interface entre a aplicação e a rede
  - dois processos se comunicam enviando dados para o socket e lendo dados de dentro do socket
  - o programador (em linguagem de alto nível) só tem a opção de escolher o serviço e alguns poucos parâmetros
  - complexidade das camadas inferiores é abstraída
- Socket realiza a identificação do processo envolvido na comunicação utilizando: Endereço IP e Número da Porta
  - Endereço IP: identifica a rede e o host individualmente
  - Porta: endereça o processo de aplicação

Processos de aplicação/transporte e sockets



Protocolos de aplicação definem:

- **Tipo de mensagens trocadas.** Ex: request, response, get, post, etc.
- **Sintaxe da mensagem:** quais os campos existentes
- **Semântica da mensagem:** significado dos dados nos campos
- **Regras** de quando e como os processos enviam/recebem mensagens
- **Protocolos Abertos:**
  - São definidos em RFC's e todos tem acesso a sua especificação para poderem implementar, permitindo interoperabilidade
  - Ex: HTTP, SMTP
- **Protocolos proprietários:**
  - Criados por empresas, podem ser fechados e não há permissão para integrar em outras aplicações.
  - Ex: Skype, Zoom.

Serviços de Transporte - Requisitos

Transferência confiável: identificar se os dados chegaram ao destino

- algumas aplicações (ex:áudio) toleram alguma perda, mas outras (ex: transferência de arquivos) exigem transferência 100% confiável.

Temporização: os dados devem chegar em um intervalo de tempo

- algumas aplicações (ex:telefonia Internet e jogos interativos) exigem baixos atrasos para operar

Vazão (largura de banda): garantir que exista uma velocidade mínima

- algumas aplicações (ex:multimídia) exigem banda mínima para operar, enquanto outras são “elásticas”, ou seja, melhoram quando a banda disponível aumenta, mas funcionam com menos.

Segurança: preservar confidencialidade, integridade e autenticidade

- é possível oferecer alguns serviços de segurança (ex: através de codificação dos dados), para garantir a segurança.

Requisitos de Transporte de Aplicações Comuns

Aplicação	Perda de dados	Vazão	Sensibilidade ao tempo
Transferência/*download* de arquivo	Sem perda	Elástica	Não
E-mail	Sem perda	Elástica	Não
Documentos Web	Sem perda	Elástica (alguns k*bits/s*)	Não
Telefonia via Internet/videoconferência	Tolerante à perda	Áudio: alguns k*bits/s* – 1M*bit/s* Vídeo: 10 k*bits/s* – 5 M*bits/s*	Sim: décimos de segundo
Streaming de áudio/vídeo armazenado	Tolerante à perda	Igual acima	Sim: alguns segundos
Jogos interativos	Tolerante à perda	Poucos k*bits/s* – 10 k*bits/s*	Sim: décimos de segundo
Mensagens de *smartphone*	Sem perda	Elástica	Sim e não

Serviços de Transporte da Internet

serviço TCP:

- *orientado à conexão:* conexão requerida entre cliente e servidor
- *transporte confiável:* recupera dados perdidos na transmissão
- *controle de fluxo:* impede que o transmissor inunde o receptor
- *controle de congestionamento :* impede o excesso de tráfego
- *não oferece:* garantias de temporização, vazão ou segurança (há o TLS na camada de aplicação para segurança)

serviço UDP:

- protocolo mais simples e minimalista
- transferência de dados não confiável entre os processos transmissor e receptor
- não oferece: estabelecimento de conexão, confiabilidade, controle de fluxo e de congestionamento, garantia de temporização e de vazão.

Obs: Confiabilidade não é o mesmo que segurança!!!!

Aplicações e Protocolos de Transporte da Internet

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte
Correio eletrônico	SMTP (RFC 5321)	TCP
Acesso a terminal remoto	Telnet (RFC 854)	TCP
Web	HTTP 1.1 (RFC 7230)	TCP
Transferência de arquivo	FTP (RFC 959)	TCP
Streaming de multimídia	HTTP (p. ex., YouTube), DASH	TCP
Telefonia por Internet	SIP (RFC 3261), RTP (RFC 3550) ou proprietária (p. ex., Skype)	UDP ou TCP