# CRIPTOGRAFIA

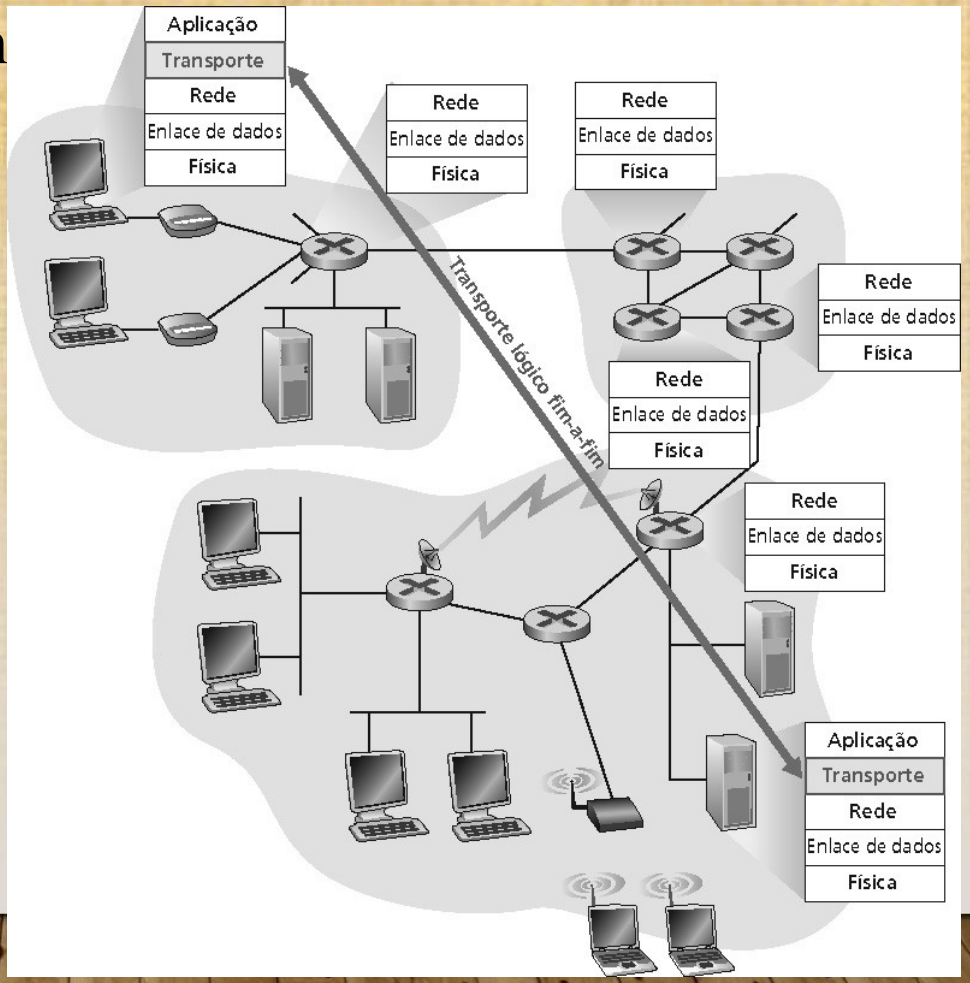
- Revisão de Camadas de Protocolos
  - Camada de transporte;
  - Diferenças entre TCP e UDP;
  - Portas de comunicação (sockets);
  - Análise de protocolos (sniffer).

# CAMADA DE TRANSPORTE

- O termo "qualidade de serviço" descreve a finalidade da camada 4, a camada de transporte.
- As suas responsabilidades principais são:
  - multiplexação/demultiplexação;
  - transferência de dados confiável;
  - controle de fluxo;
  - controle de congestionamento.

# PROTOCOLOS E SERVIÇOS DA CAMADA DE TRANSPORTE

- Fornecem comunicação lógica entre processos de aplicação em diferentes hosts
- Os protocolos de transporte são executados nos sistemas finais
  - Lado emissor: quebra as mensagens da aplicação em segmentos e envia para a camada de rede
  - Lado receptor: remonta os segmentos em mensagens e passa para a camada de aplicação
- Há mais de um protocolo de transporte disponível para as aplicações
  - Internet: TCP e UDP



# CAMADA DE TRANSPORTE

- Características do TCP:
  - Orientado para conexão;
  - Confiável;
  - Divide as mensagens enviadas em segmentos;
  - Reenvia tudo o que não foi recebido;
  - Reagrupa as mensagens a partir de segmentos recebidos.

# CAMADA DE TRANSPORTE

- Características do UDP:
  - Sem conexão;
  - Não confiável;
  - Melhor esforço;
  - Sem ordem de entrega;
  - Não reagrupa as mensagens de entrada;
  - Não usa confirmações;
  - Não fornece controle de fluxo.

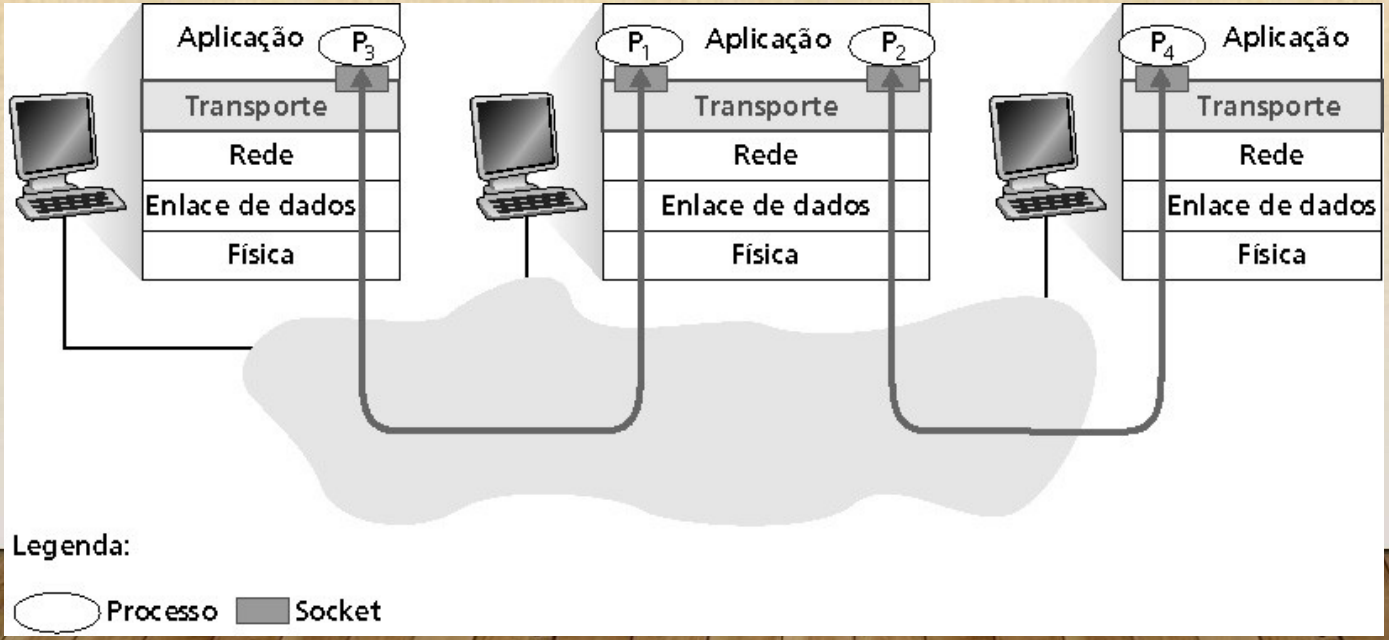
# MULTIPLEXAÇÃO
# DEMULTIPLEXAÇÃO

**Demultiplexação no hospedeiro receptor:**

entrega os segmentos recebidos ao socket correto

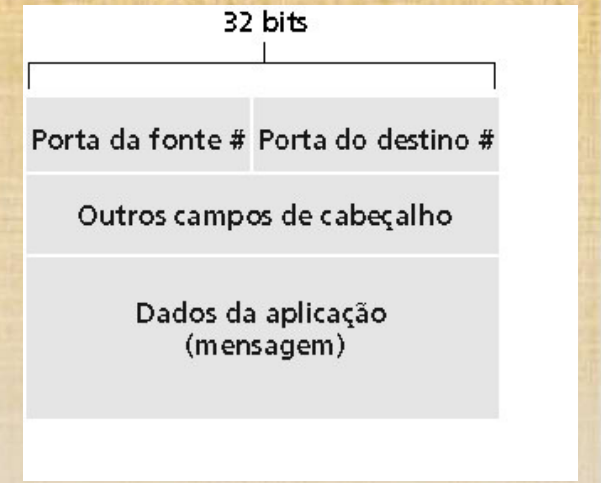
**Multiplexação no hospedeiro emissor:**

coleta dados de múltiplos sockets, envelopa os dados com cabeçalho (usado depois para demultiplexação)



# MULTIPLEXAÇÃO
# DEMULTIPLEXAÇÃO

- Computador recebe datagramas IP
  - Cada datagrama possui endereço IP de origem e IP de destino
  - Cada datagrama carrega 1segmento da camada de transporte
  - Cada segmento possui números de porta de origem e destino (lembre-se: números de porta bem conhecidos para aplicações específicas)
- O hospedeiro usa endereços IP e números de porta para direcionar o segmento ao socket apropriado



# CAMPOS DO SEGMENTO TCP

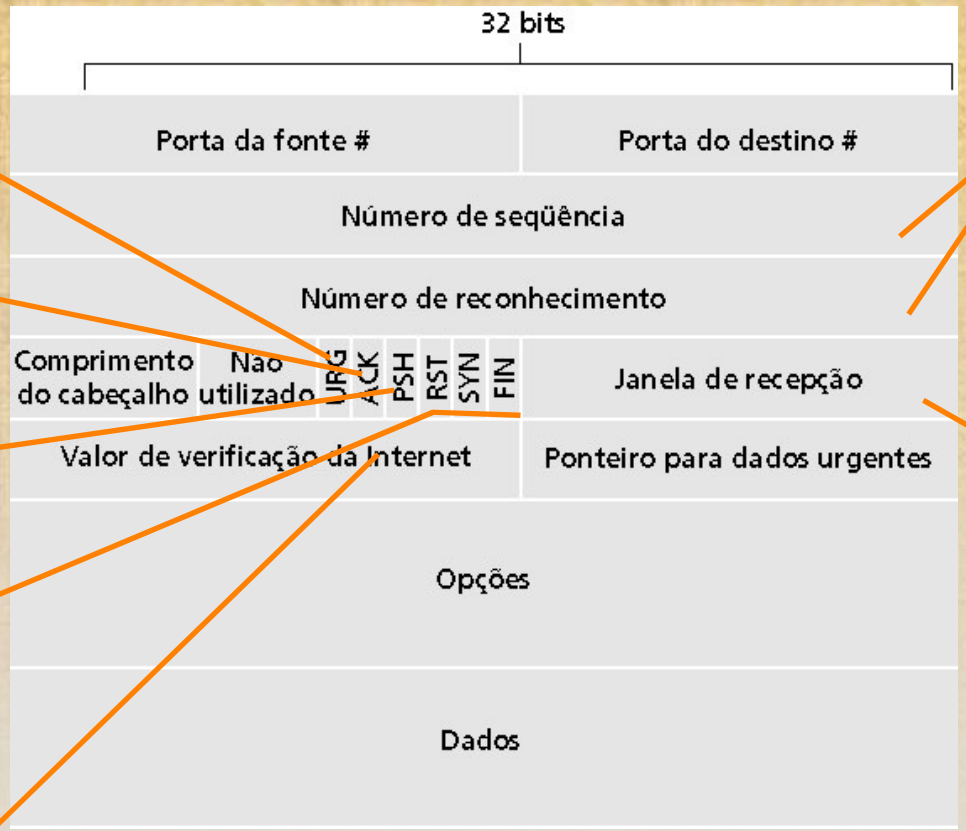
URG: dados urgentes (pouco usados)

ACK: campo de ACK é válido

PSH: produz envio de dados (pouco usado)

RST, SYN, FIN: estabelece conexão (comandos de criação e término)

Internet checksum (como no UDP)

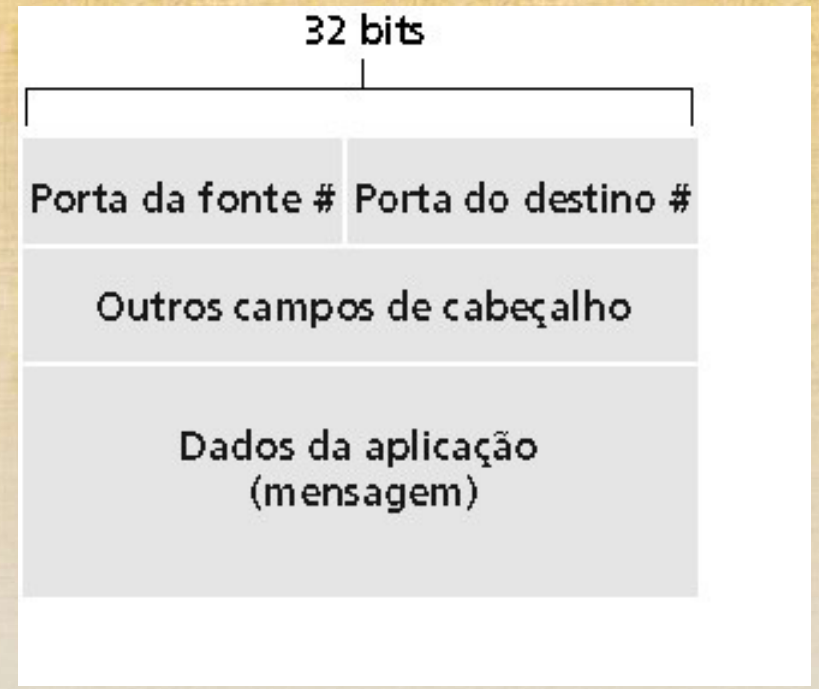



contagem por Bytes de dados

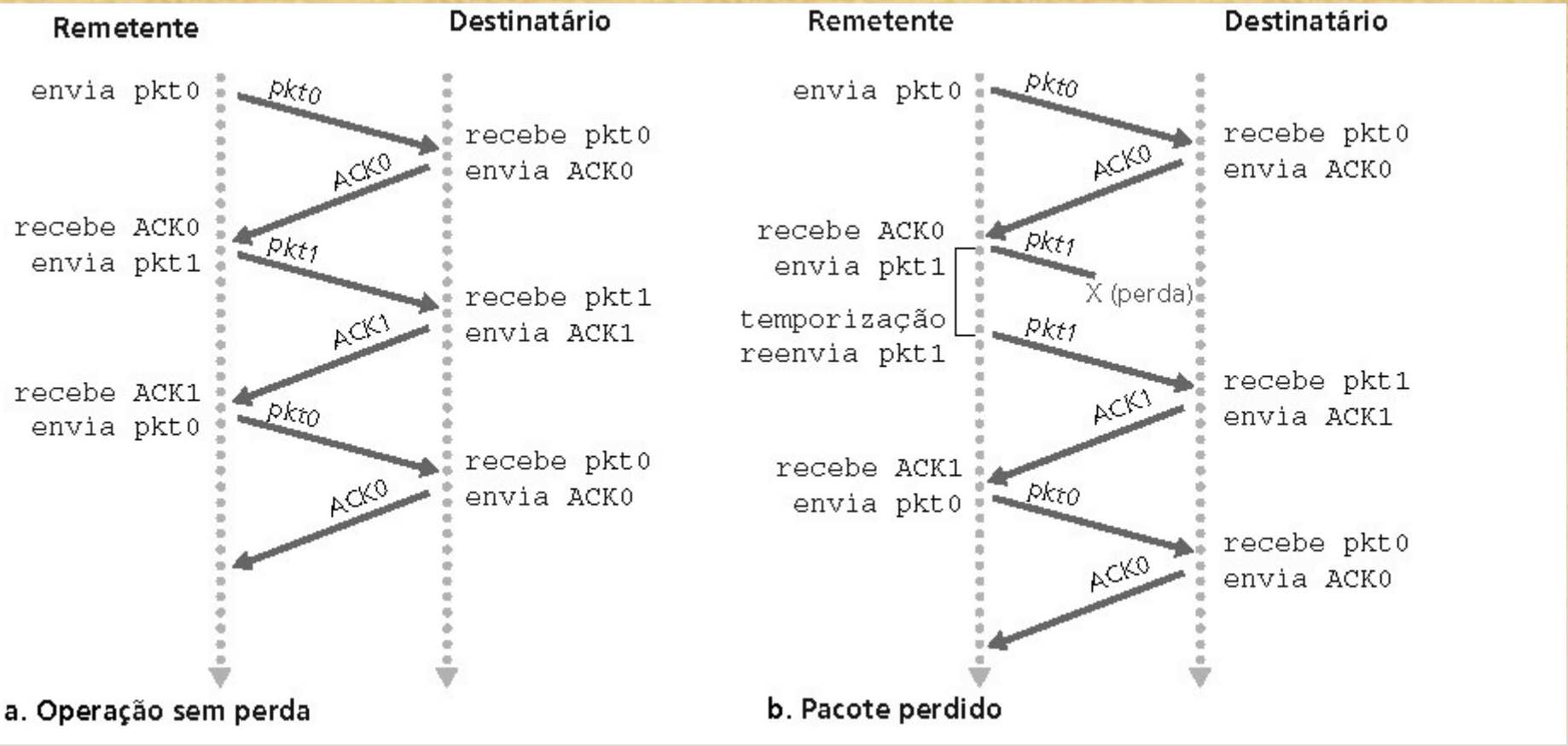
número de Bytes que o receptor está pronto para aceitar

# CAMPOS DO SEGMENTO UDP

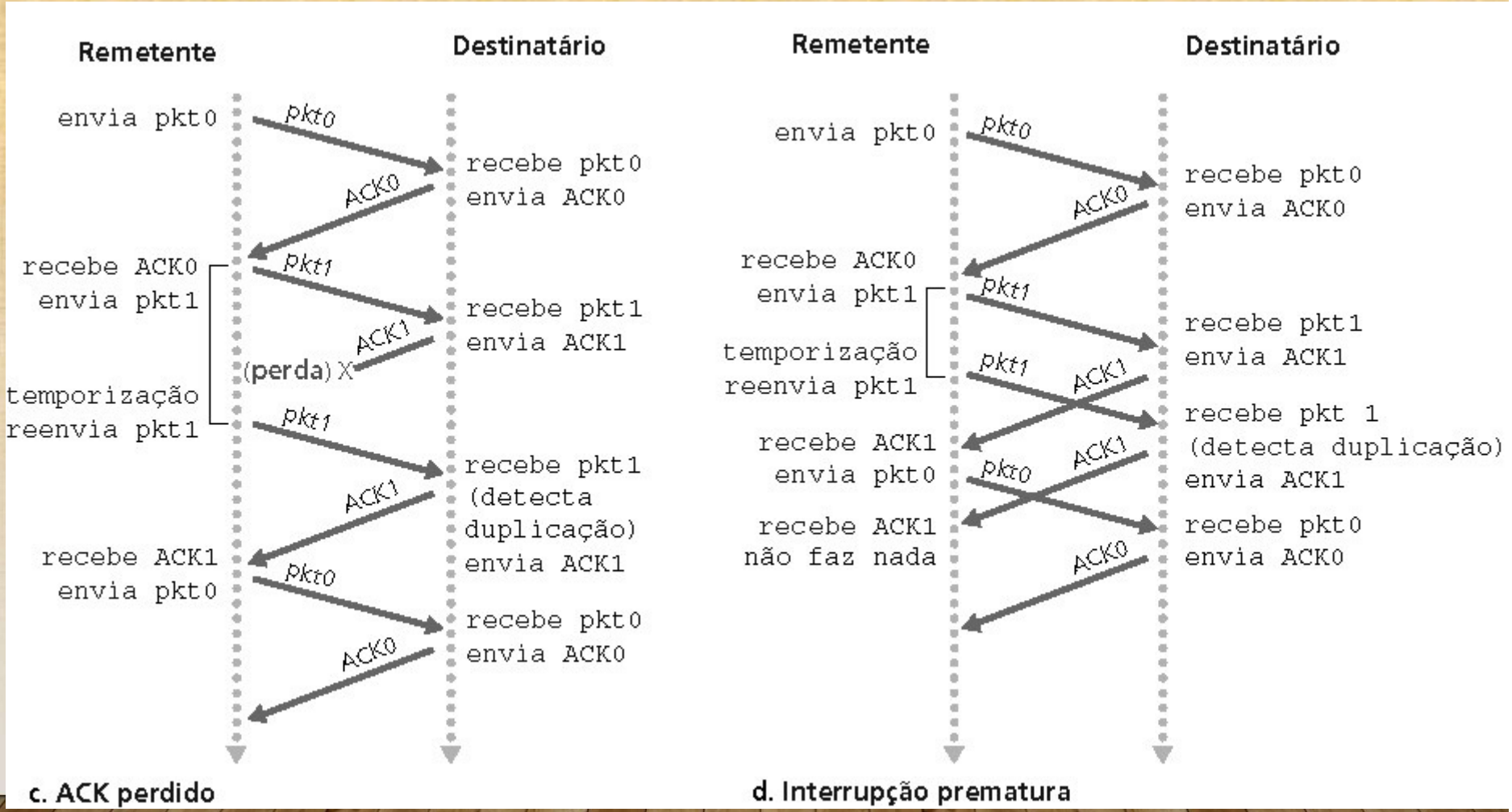
- Muito usado por aplicações de mutimídia contínua (streaming)
  - Tolerantes à perda
  - Sensíveis à taxa
- Transferência confiável sobre UDP: acrescentar confiabilidade na camada de aplicação.



# DADOS CONFIÁVEIS NO TCP (PARE E ESPERE)



# DADOS CONFIÁVEIS NO TCP (PARE E ESPERE)



# DADOS CONFIÁVEIS NO TCP (PARE E ESPERE)

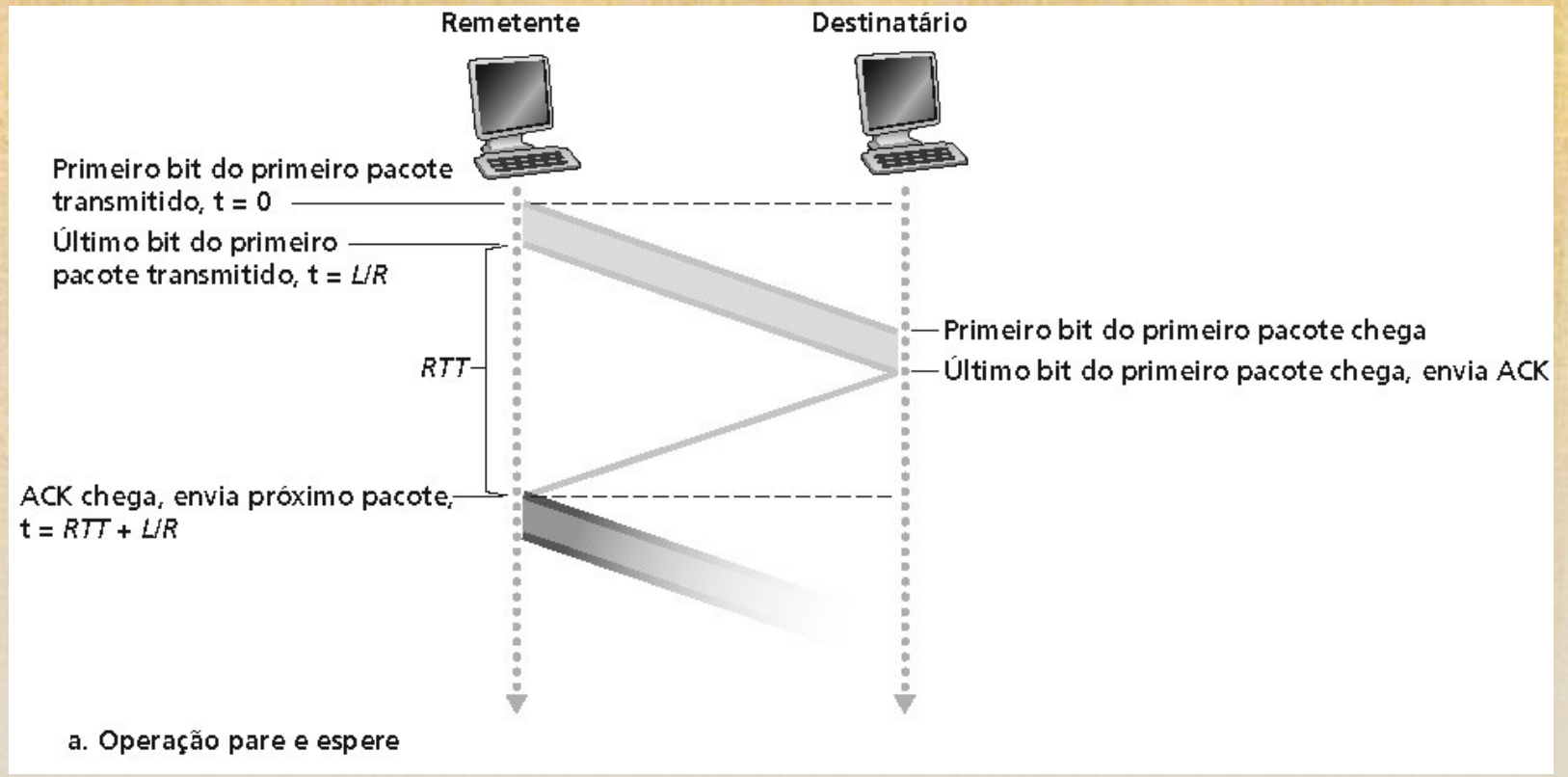
- Método (pare e espera) funciona, mas o desempenho é sofrível
- Exemplo: enlace de 1 Gbps, 15 ms de atraso de propagação, pacotes de 1 KB:
-

$$T_{tx} = \frac{L \text{ (tamanho do pacote em bits)}}{R \text{ (taxa de transmissão, bps)}} = \frac{8 \text{ kb/pkt}}{1 \text{Gb/s}} = 8 \text{ us}$$

$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

- $U_{sender}$: utilização – fração de tempo do transmissor ocupado
- Um pacote de 1 KB cada 30 ms -> 267 kB/s de vazão sobre um canal de 1 Gbps
- O protocolo de rede limita o uso dos recursos físicos!

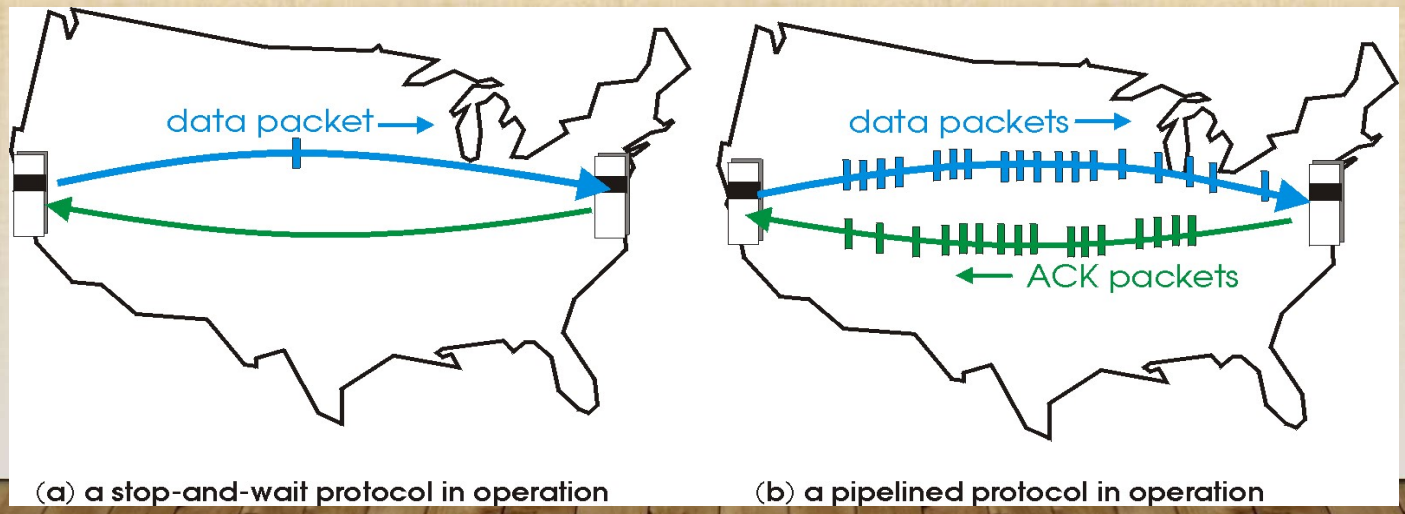
# DADOS CONFIÁVEIS NO TCP (PARE E ESPERE)



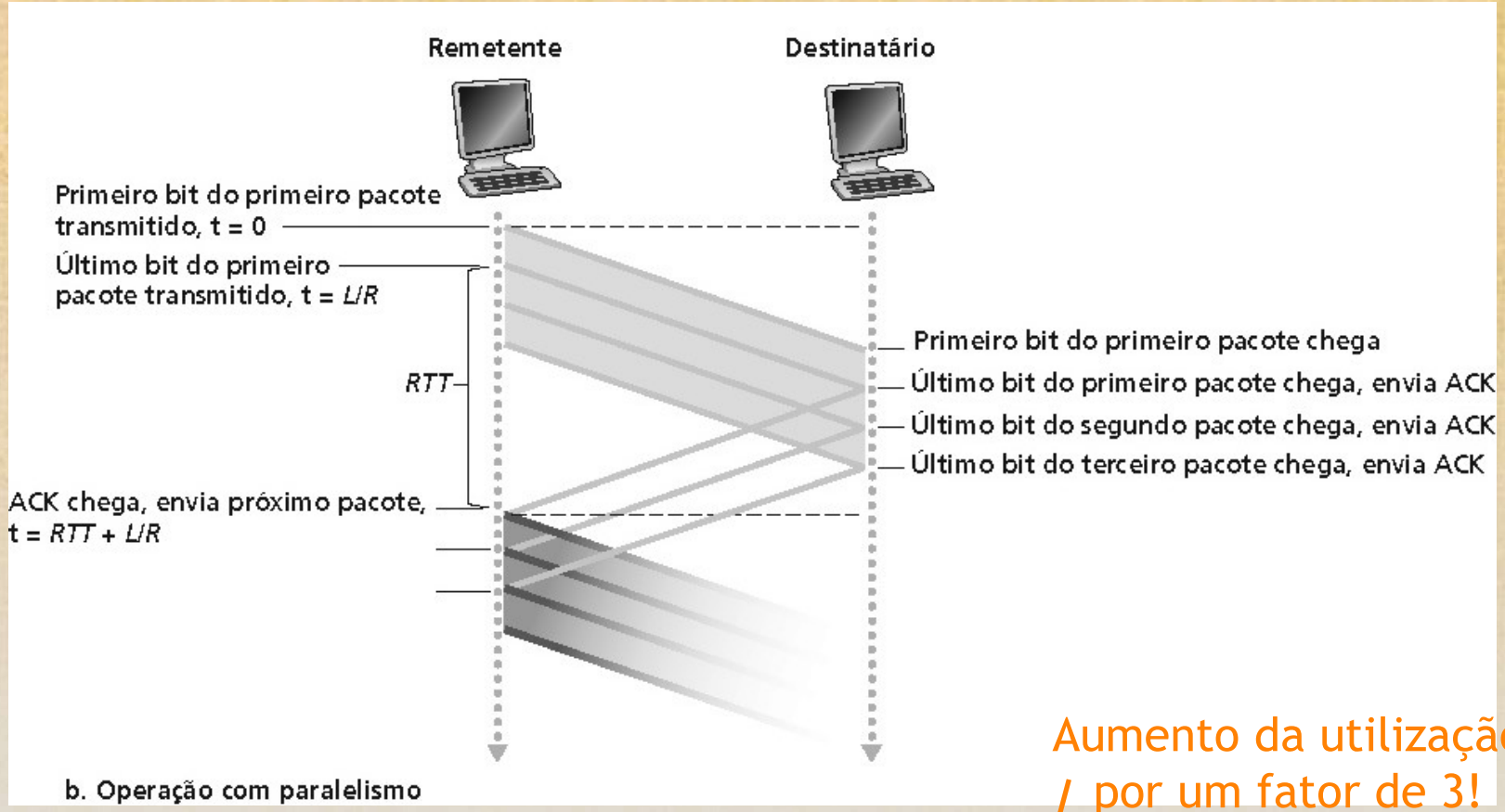

a. Operação pare e espere

# DADOS CONFIÁVEIS NO TCP (PARALELISMO)

- Paralelismo: transmissor envia vários pacotes ao mesmo tempo, todos esperando para serem reconhecidos
- Faixa de números de seqüência deve ser aumentada
- Armazenamento no transmissor e/ou no receptor



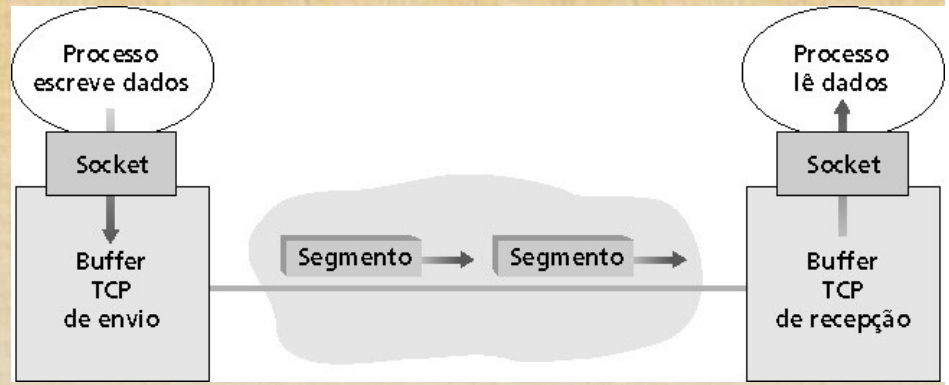
# DADOS CONFIÁVEIS NO TCP (PARALELISMO)




Aumento da utilização por um fator de 3!

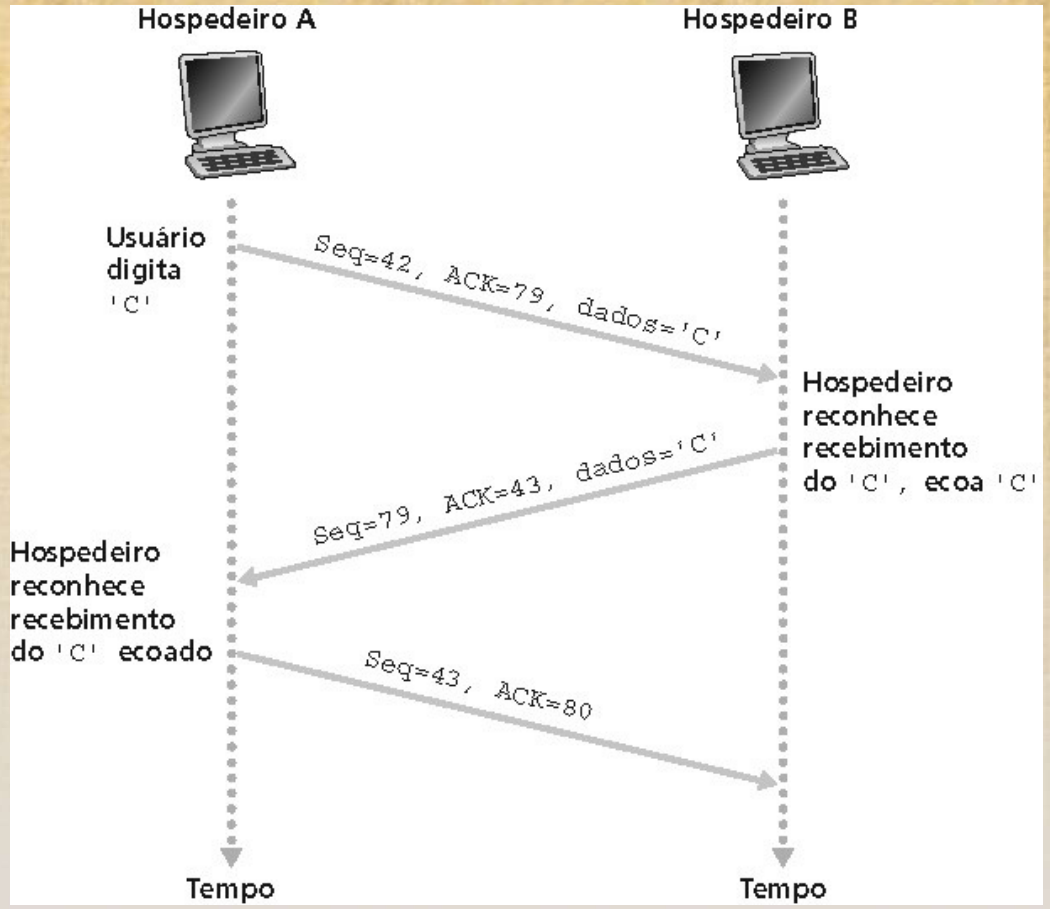
# TCP (OVERVIEW)

- **Ponto-a-ponto**:
  - Um transmissor, um receptor
  - **Confiável, seqüencial**
  - **Pipelined:**
  - Controle de congestão e de fluxo definem tamanho da janela
  - **Buffers de transmissão e de recepção**
  - Dados full-duplex:
  - Transmissão bidirecional na mesma conexão
  - MSS: maximum segment size
  - Orientado à conexão:
  - Apresentação (troca de mensagens de controle) inicia o estado do transmissor e do receptor antes da troca de dados
  - Controle de fluxo:
  - Transmissor não esgota a capacidade do receptor



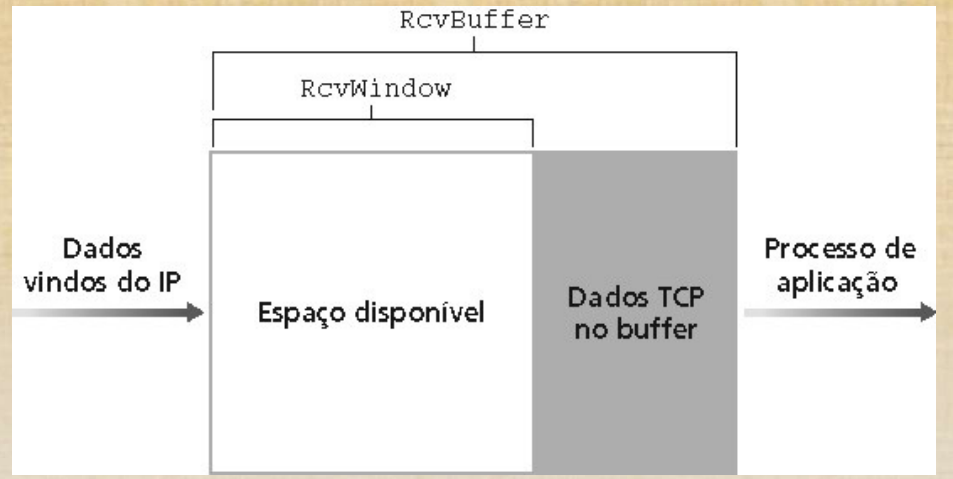
# NÚMERO DE SEQUÊNCIAS E ACK NO TCP

- Números de seqüência:
  - Número do primeiro byte nos segmentos de dados
- ACKs:
- Número do próximo byte esperado do outro lado
- ACK cumulativo
- P.: Como o receptor trata segmentos fora de ordem?
  - A especificação do TCP não define, fica a critério do implementador



# TCP (CONTROLE DE FLUXO)

- lado receptor da conexão TCP possui um buffer de recepção:




- Processos de aplicação podem ser lentos para ler o buffer

Controle de fluxo

Transmissor não deve esgotar os buffers de recepção enviando dados rápido demais

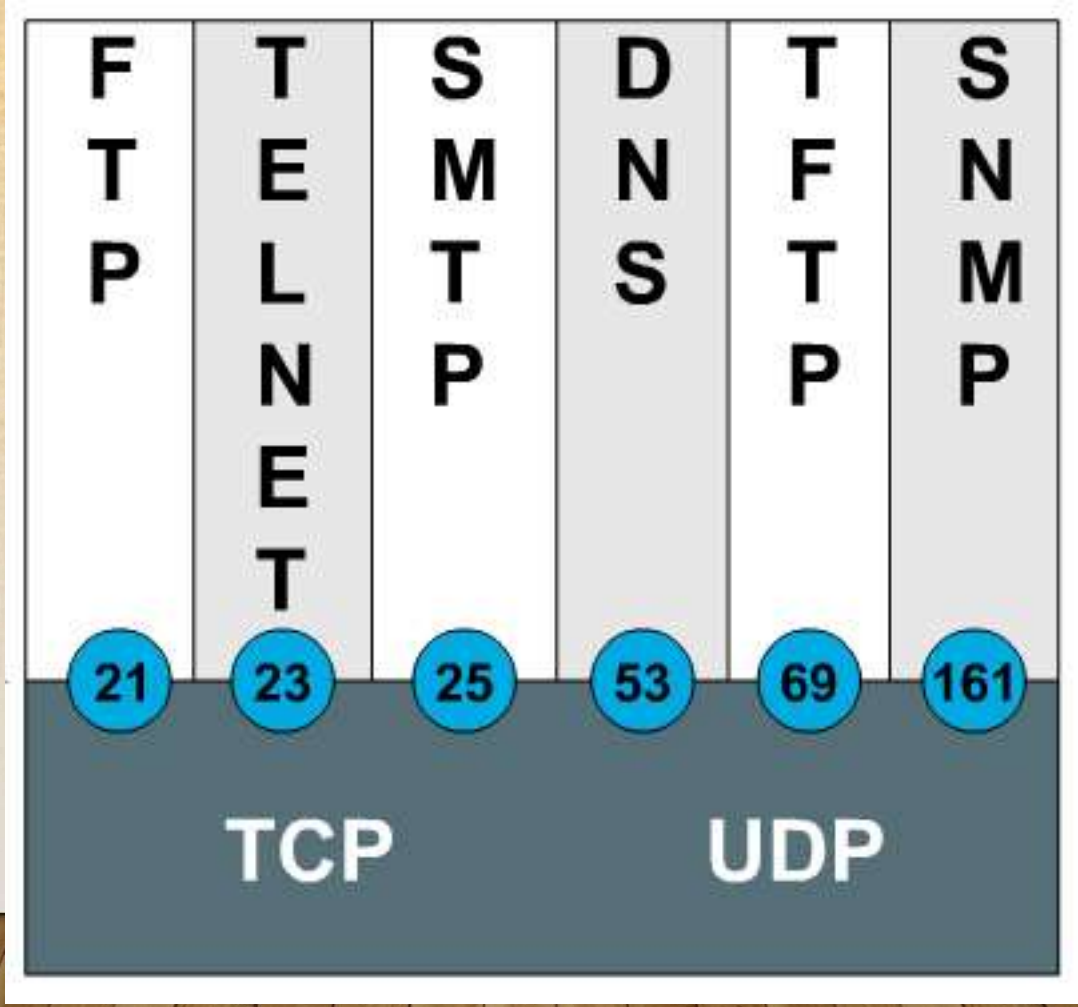
- Serviço de **speed-matching**: encontra a taxa de envio adequada à taxa de vazão da aplicação receptora

# PORTAS DE COMUNICAÇÃO (SOCKETS)

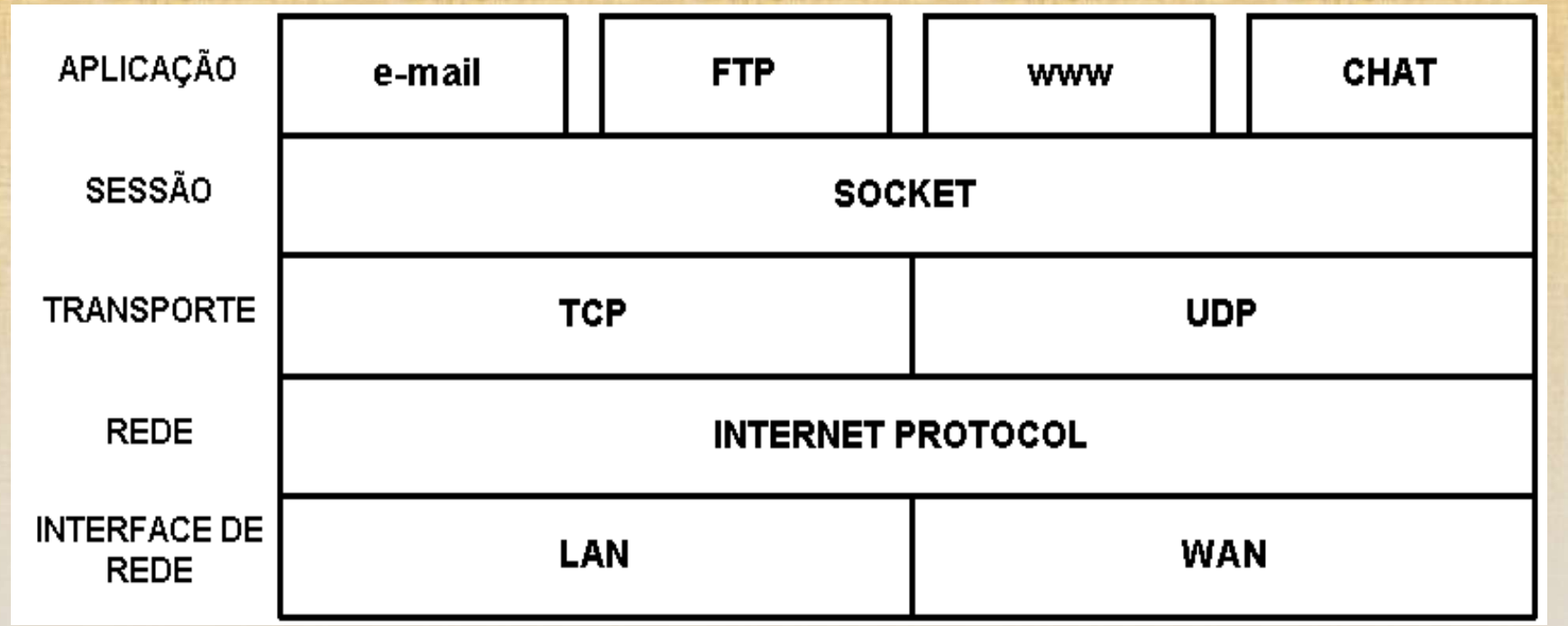
Camada de Aplicação

Número de portas

Camada de Transporte

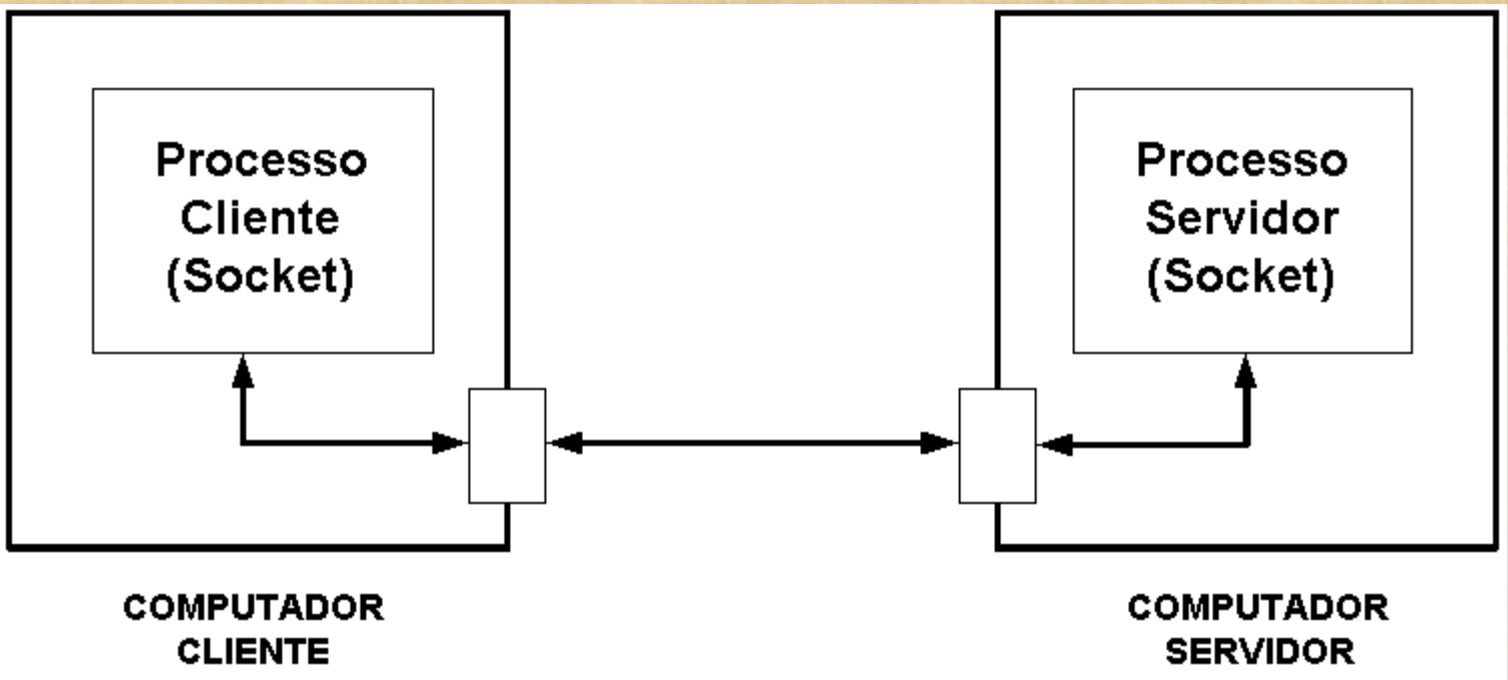


# PORTAS DE COMUNICAÇÃO (SOCKETS)

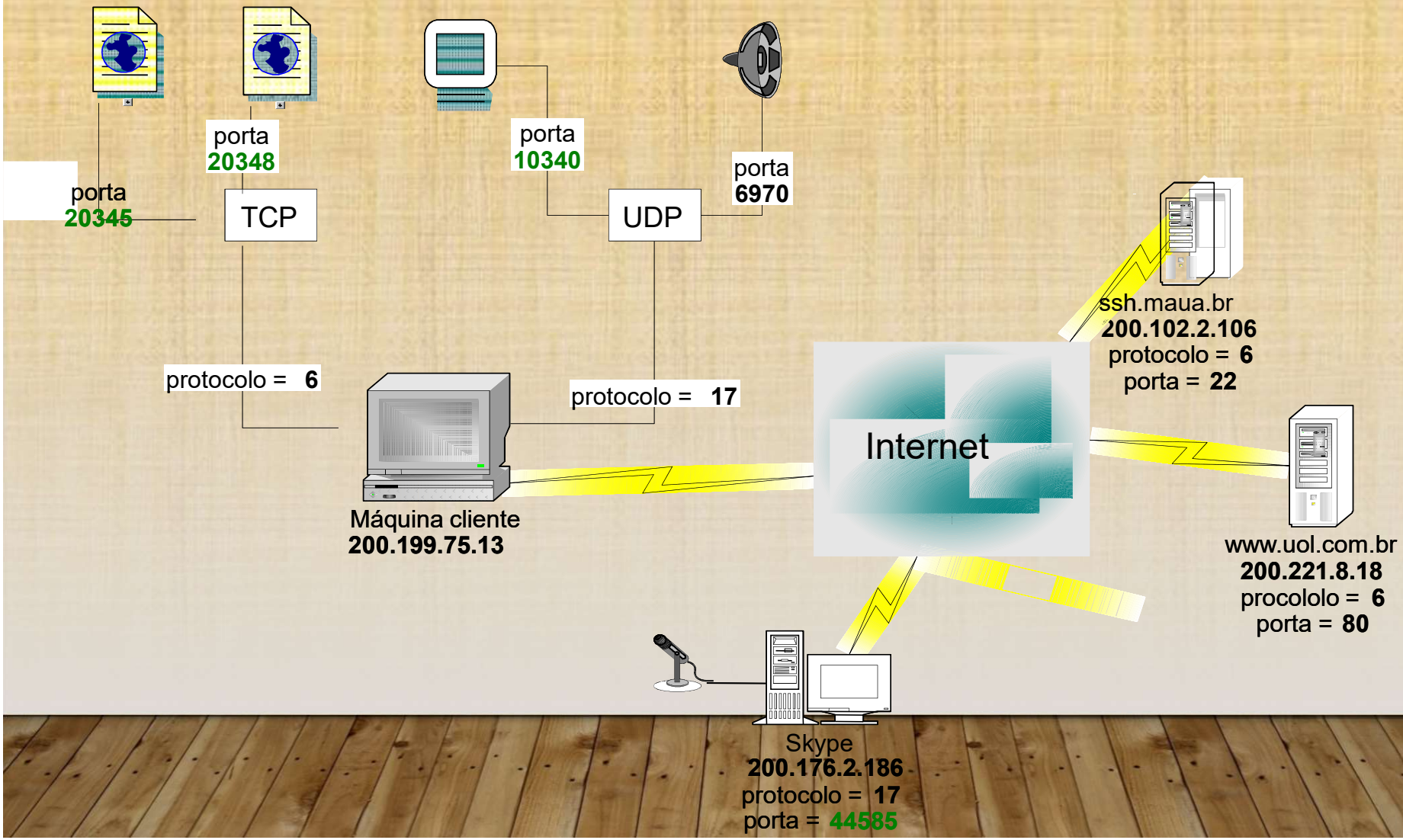


# PORTAS DE COMUNICAÇÃO (SOCKETS)

- Conexão entre os sockets (arquitetura cliente/servidor)



# PORTAS DE COMUNICAÇÃO (SOCKETS)



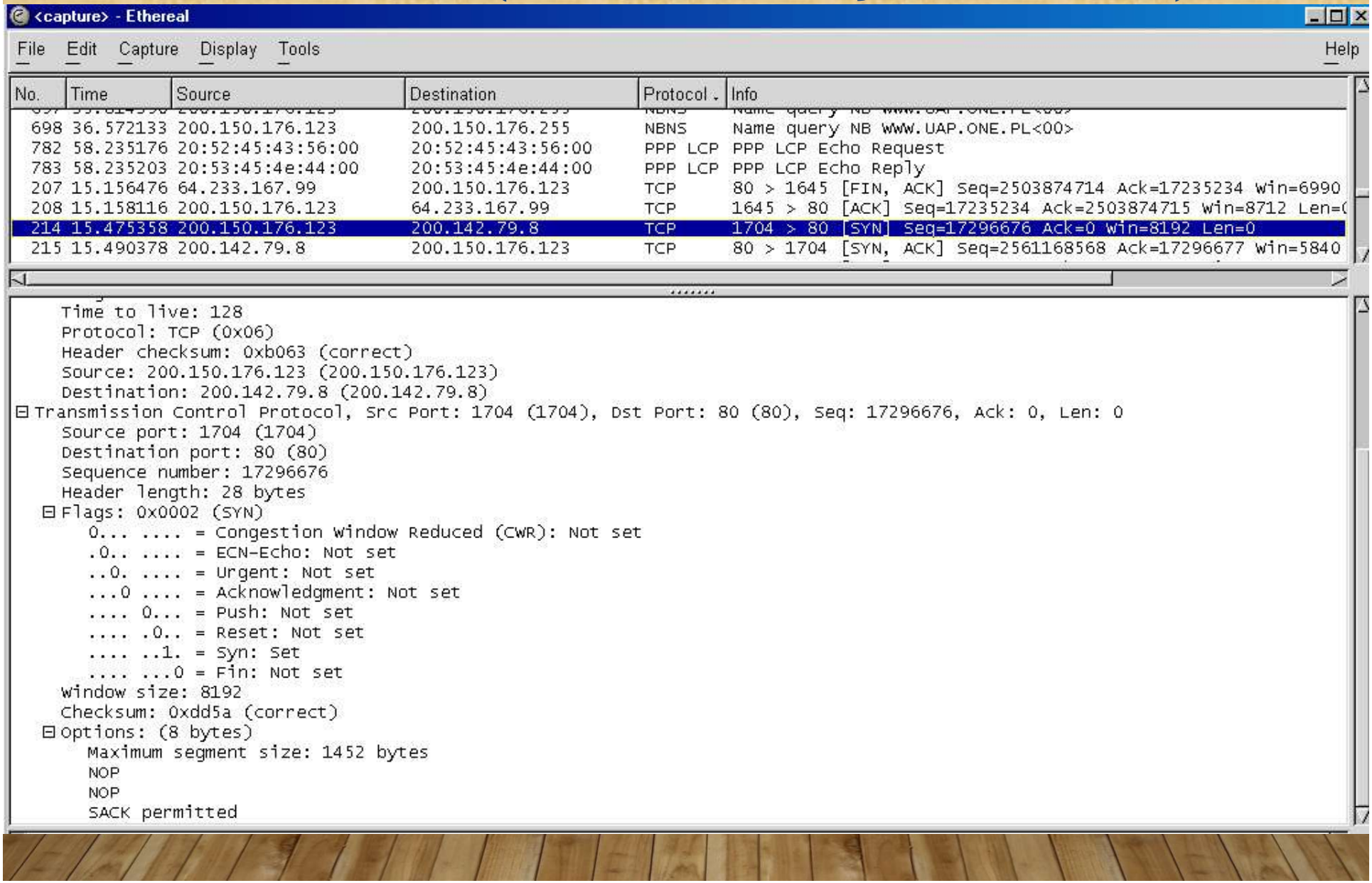
# PORTAS DE COMUNICAÇÃO (SOCKETS)

- Para cada serviço é associado uma porta e um número de identificação de 0 à 65535 (2^16).
- Os números de portas têm os seguintes conjuntos atribuídos:
  - Números de 1 à 1024 – portas padronizadas pelo IANA (Internet Assigned Numbers Authority) usadas pelos serviços mais conhecidos na Internet (ftp, dns, smtp, http, etc);
  - Números acima de 1024 à 65535 - não são regulamentados (porta de comunicação selecionada dinamicamente pelo sistema operacional da máquina cliente para um serviço específico, por exemplo, http).

# PORTAS DE COMUNICAÇÃO (SOCKETS)

- ftp-data 20/tcp - File Transfer (Data)
- ftp 21/tcp - File Transfer (Control)
- ssh 22/tcp - SSH - Remote Login Protocol
- telnet 23/tcp - Telnet
- smtp 25/tcp - Simple Mail Transfer
- domain 53/tcp/udp - Domain Name Server
- www-http 80/tcp - World Wide Web HTTP
- pop3 110/tcp - Post Office Protocol - Version 3
- snmp 161/udp - SNMP
- imap3 220/tcp - Interactive Mail Access Protocol v3
- https 443/tcp - https

# SNIFFER (SINALIZAÇÕES TCP)



# SNIFFER (TCP - SYN=1)

```
⊟ Transmission Control Protocol, Src Port: 1704 (1704), Dst P
    Source port: 1704 (1704)
    Destination port: 80 (80)
    Sequence number: 17296676
    Header length: 28 bytes
  ⊟ Flags: 0x0002 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 8192
    Checksum: 0xdd5a (correct)
  ⊟ Options: (8 bytes)
      Maximum segment size: 1452 bytes
      NOP
      NOP
      SACK permitted
```

# SNIFFER (TCP – SYN=1+ACK=1)

```
⊟ Transmission Control Protocol, Src Port: 80 (80), Dst Port
    Source port: 80 (80)
    Destination port: 1704 (1704)
    Sequence number: 2561168568
    Acknowledgement number: 17296677
    Header length: 24 bytes
  ⊟ Flags: 0x0012 (SYN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x0e20 (correct)
  ⊟ Options: (4 bytes)
      Maximum segment size: 1452 bytes
```

# SNIFFER (TCP – SYN=0+ACK=1)

```
⊟ Transmission Control Protocol, Src Port: 1704 (1704), Dst
    Source port: 1704 (1704)
    Destination port: 80 (80)
    Sequence number: 17296677
    Acknowledgement number: 2561168569
    Header length: 20 bytes
  ⊟ Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 8712
    Checksum: 0x1a9d (correct)
```

# SNIFFER (TCP – FIN=1, ACK=1)

```
⊟ Transmission Control Protocol, Src Port: 1704 (1704), Dst
    Source port: 1704 (1704)
    Destination port: 80 (80)
    Sequence number: 17296832
    Acknowledgement number: 2561297167
    Header length: 20 bytes
  ⊟ Flags: 0x0011 (FIN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...1 = Fin: Set
    Window size: 8712
    Checksum: 0x23a9 (correct)
```

# SNIFFER (TCP – FIN=0, ACK=1)

```
⊟ Transmission Control Protocol, Src Port: 80 (80), Dst Port
    Source port: 80 (80)
    Destination port: 1704 (1704)
    Sequence number: 2561297167
    Acknowledgement number: 17296833
    Header length: 20 bytes
  ⊟ Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x2ee1 (correct)
```