Capítulo 3: Camada de Transporte

Metas do capítulo:

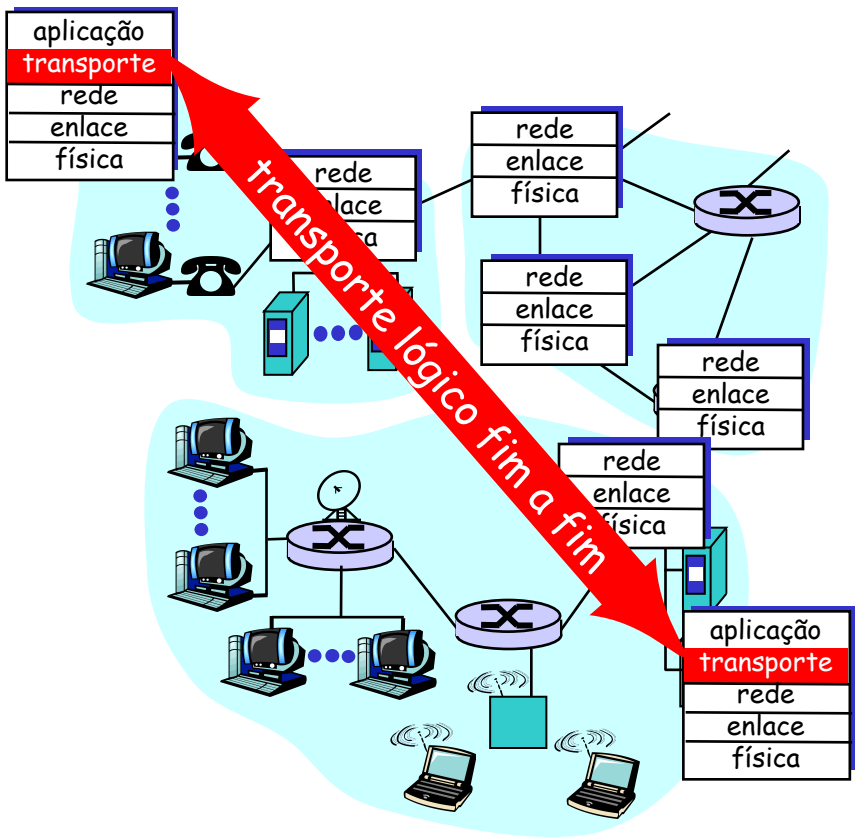
- entender os princípios atrás dos serviços da camada de transporte:
  - multiplexação/ demultiplexação
  - transferência confiável de dados
  - controle de fluxo
  - controle de congestionamento
- aprender sobre os protocolos da camada de transporte da Internet:
  - UDP: transporte não orientado a conexões
  - TCP: transporte orientado a conexões
  - Controle de congestionamento do TCP

Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

Serviços e protocolos de transporte

- fornecem *comunicação lógica* entre processos de aplicação executando em diferentes hospedeiros
- os protocolos de transporte são executados nos sistemas finais:
  - lado transmissor: quebra as mensagens da aplicação em segmentos, repassa-os para a camada de rede
  - lado receptor: remonta as mensagens a partir dos segmentos, repassa-as para a camada de aplicação
- existe mais de um protocolo de transporte disponível para as aplicações
  - Internet: TCP e UDP



Camadas de Transporte x rede

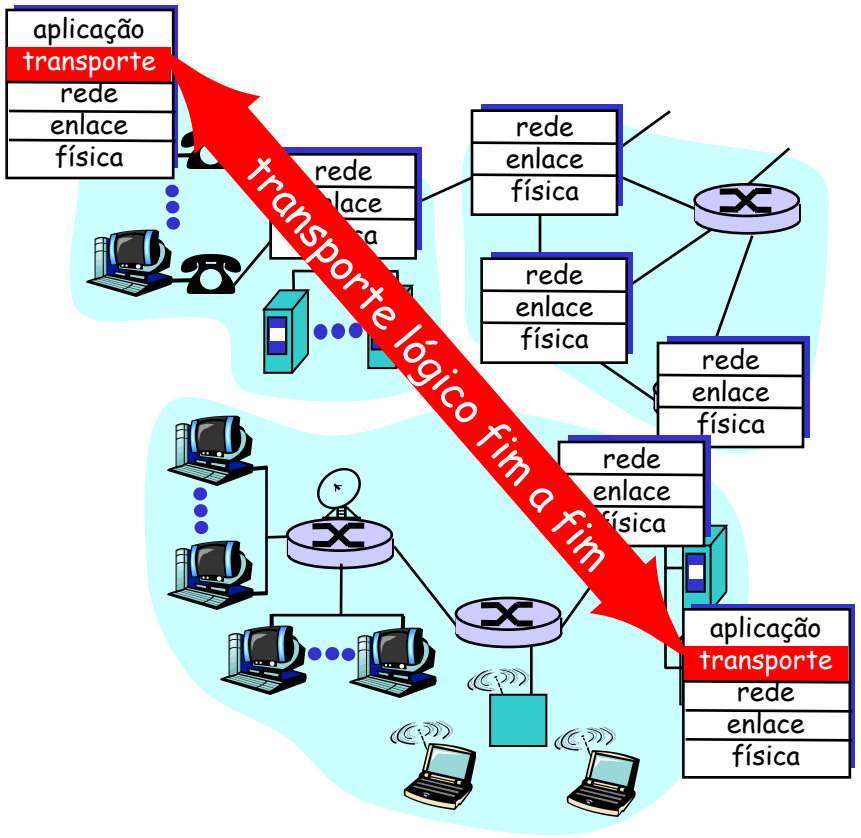
- *camada de rede:* comunicação lógica entre hospedeiros
- *camada de transporte:* comunicação lógica entre os processos
  - depende de e estende serviços da camada de rede

Analogia acadêmica:

- *Escola seria comunicação entre hospedeiros*
- *Salas de aula seria comunicação entre processos*

Protocolos da camada de transporte Internet

- entrega confiável, ordenada (TCP)
  - controle de congestionamento
  - controle de fluxo
  - estabelecimento de conexão ("setup")
- entrega não confiável, não ordenada: UDP
  - extensão sem "gorduras" do "melhor esforço" do IP
- serviços não disponíveis:
  - garantias de atraso máximo
  - garantias de largura de banda mínima



Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

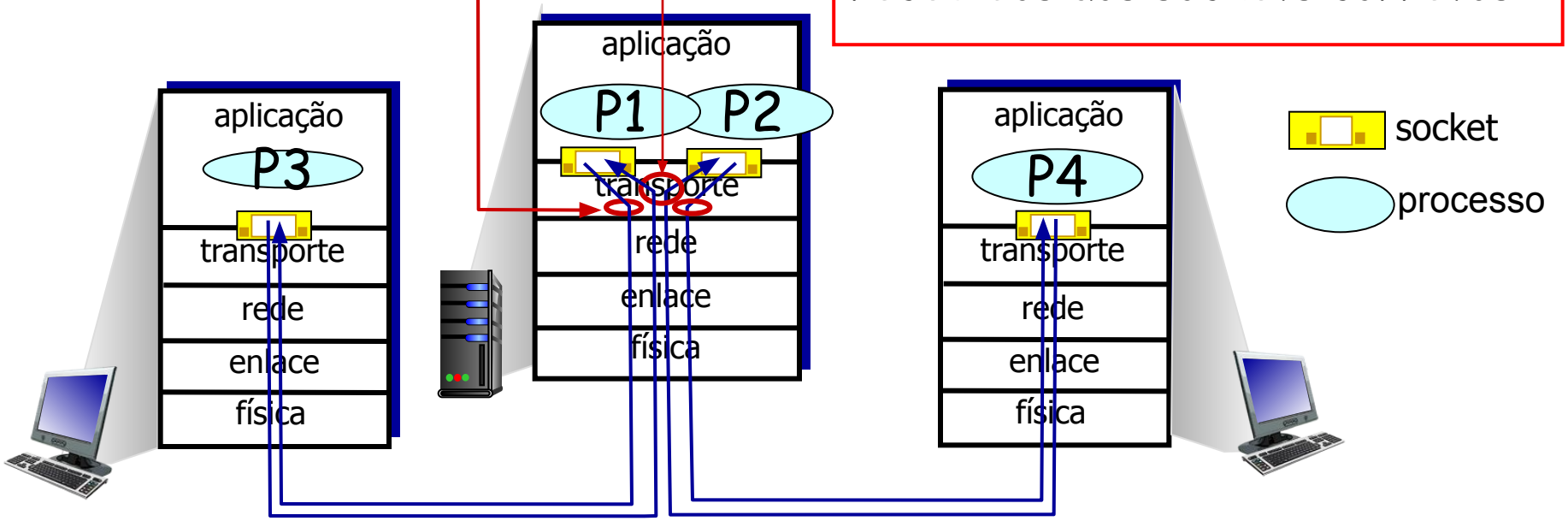
Multiplexação/demultiplexação

Multiplexação no transm.:

reúne dados de muitos sockets, adiciona o cabeçalho de transporte (usado posteriormente para a demultiplexação)

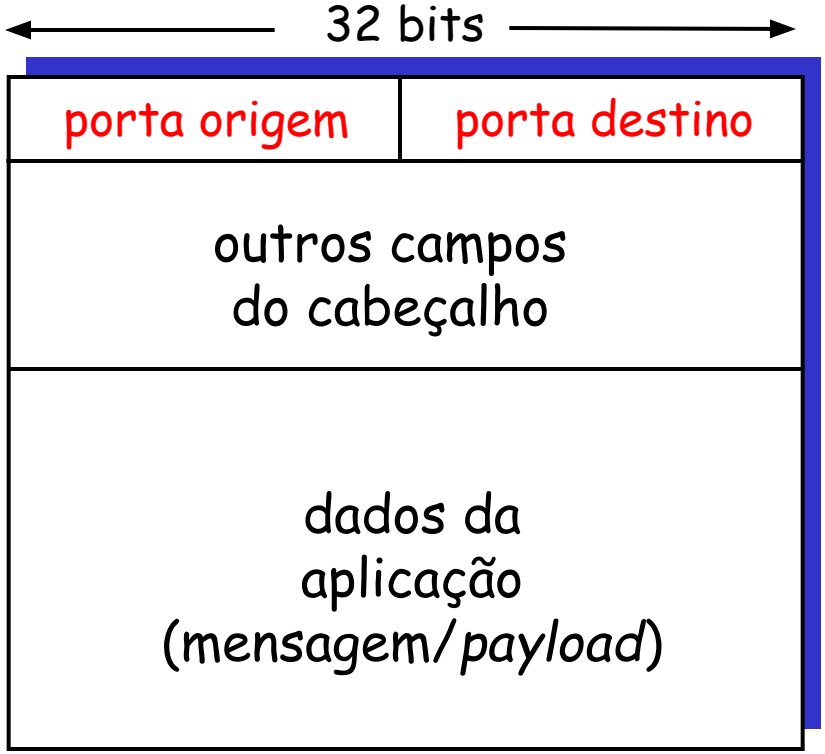
Demultiplexação no receptor:

Usa info do cabeçalho para entregar os segmentos recebidos aos sockets corretos



Como funciona a demultiplexação

- computador recebe os datagramas IP
  - cada datagrama possui os endereços IP da origem e do destino
  - cada datagrama transporta um segmento da camada de transporte
  - cada segmento possui números das portas origem e destino
- O hospedeiro usa os endereços IP e os números das portas para direcionar o segmento ao socket apropriado



formato de segmento TCP/UDP

Demultiplexação não orientada a conexões

- *Lembrete*: socket criado possui número de porta local ao host:

```
DatagramSocket mySocket1 = new
   DatagramSocket(12534);
```

- *Lembrete*: ao criar um datagrama para enviar para um socket UDP, deve especificar:
  - Endereço IP de destino
  - Número da porta de destino

- Quando o hospedeiro recebe o segmento UDP:
  - verifica no. da porta de destino no segmento
  - encaminha o segmento UDP para o socket com aquele no. de porta

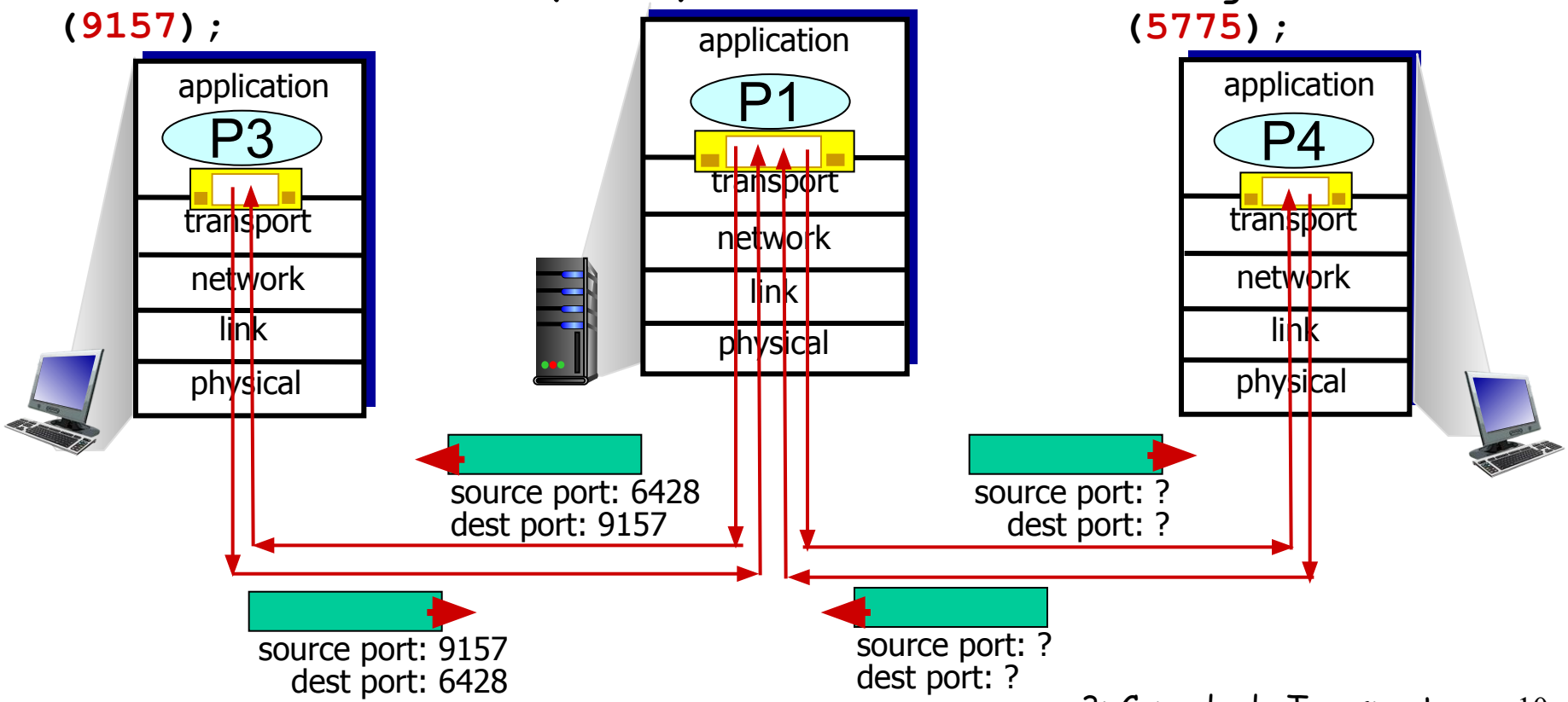
- Datagramas IP com mesmo no. de porta destino, mas diferentes endereços IP origem e/ou números de porta origem podem ser encaminhados para o mesmo socket no destino

Demultiplexação não orientada a conexões: exemplo

```
DatagramSocket
 serverSocket = new
 DatagramSocket
 (6428);
```

```
DatagramSocket
 mySocket2 = new
 DatagramSocket
 (9157);
```

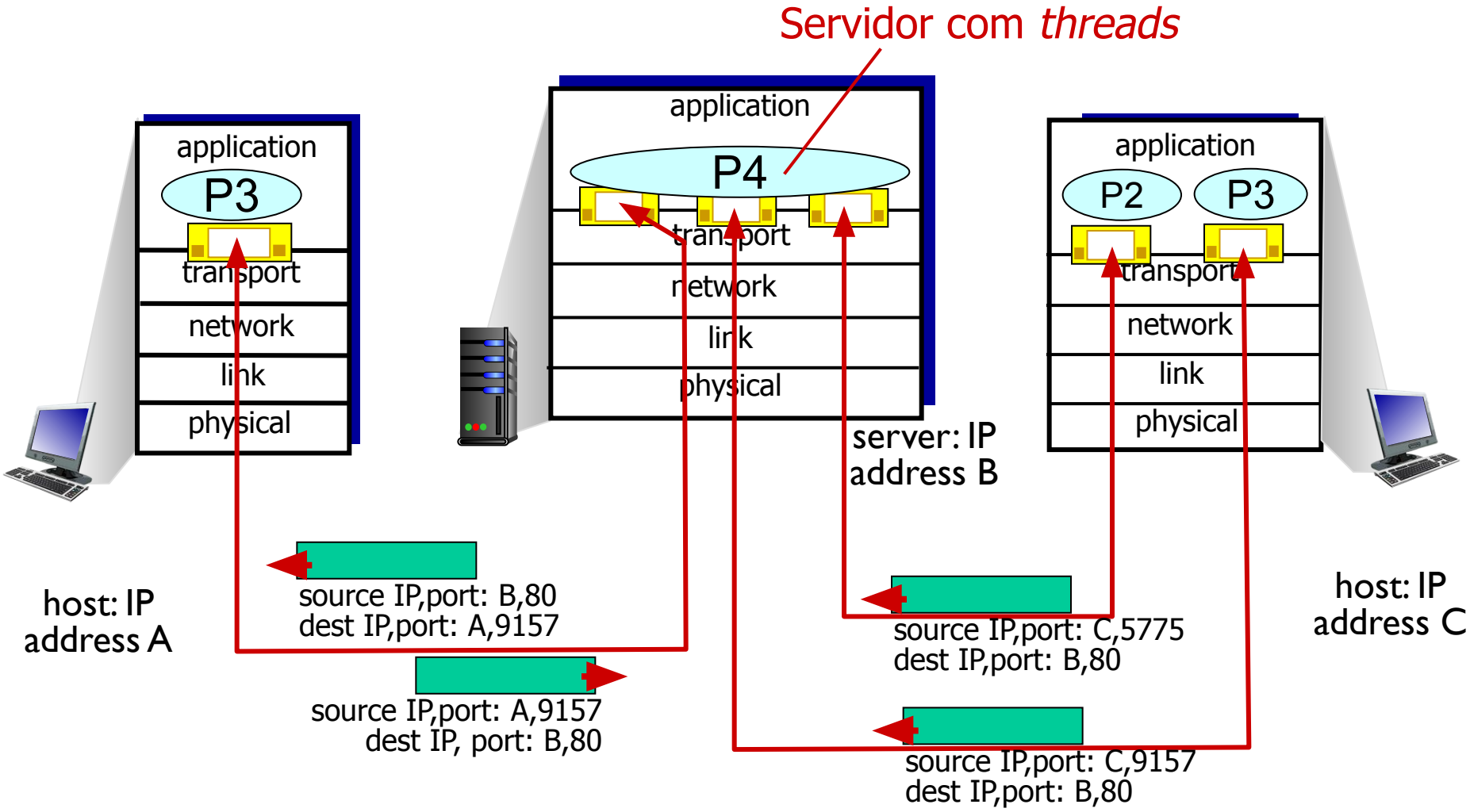
```
DatagramSocket
 mySocket1 = new
 DatagramSocket
 (5775);
```



Demultiplexação Orientada a Conexões

- Socket TCP identificado pela quádrupla:
  - endereço IP origem
  - número da porta origem
  - endereço IP destino
  - número da porta destino
- Demultiplexação: receptor usa todos os quatro valores para direcionar o segmento para o socket apropriado
- Servidor pode dar suporte a muitos sockets TCP simultâneos:
  - cada socket é identificado pela sua própria quádrupla
- Servidores Web têm sockets diferentes para cada conexão de cliente
  - HTTP não persistente terá sockets diferentes para cada pedido

Demultiplexação Orientada a Conexões: Servidor Web com Threads



Conteúdo do Capítulo 3

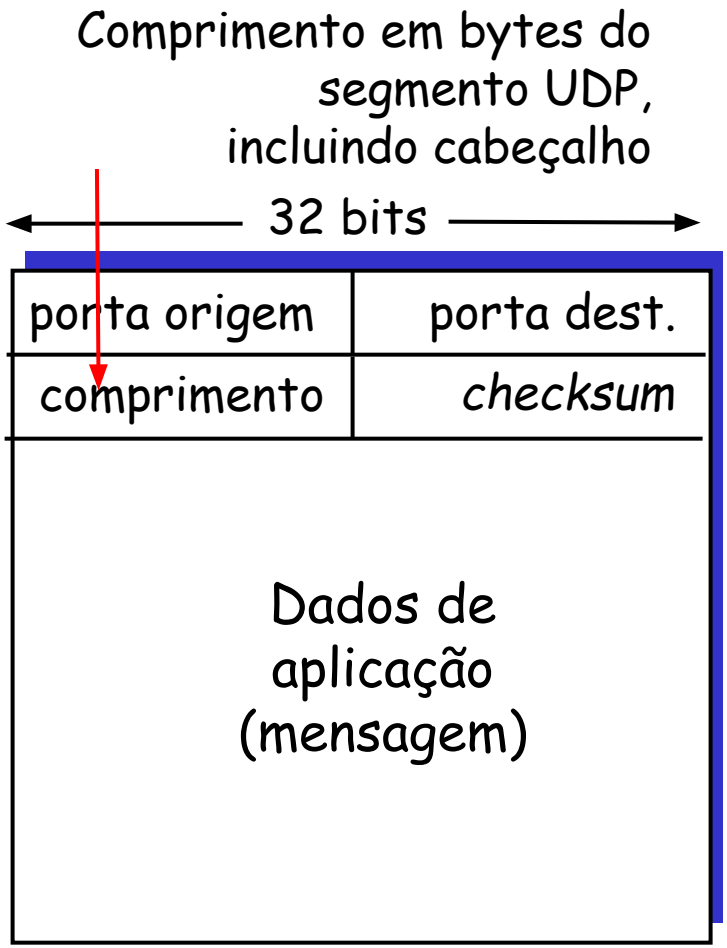
- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

UDP: User Datagram Protocol [RFC 768]

- Protocolo de transporte da Internet mínimo, "sem gorduras",
- Serviço "melhor esforço", segmentos UDP podem ser:
  - perdidos
  - entregues à aplicação fora de ordem
- *sem conexão:*
  - não há saudação inicial entre o remetente e o receptor UDP
  - tratamento independente para cada segmento UDP

- Uso do UDP:
  - aplicações de *streaming* multimídia (tolerante a perdas, sensível a taxas)
  - DNS
  - SNMP
- transferência confiável sobre UDP:
  - adiciona confiabilidade na camada de aplicação
  - recuperação de erros específica da aplicação

UDP: Cabeçalho do segmento



Formato do segmento UDP

Por quê existe um UDP?

- elimina estabelecimento de conexão (que pode causar retardo)
- simples: não mantém "estado" da conexão nem no remetente, nem no receptor
- cabeçalho de segmento reduzido
- Não há controle de congestionamento: UDP pode transmitir tão rápido quanto desejado (e possível)

Soma de Verificação (*checksum*) UDP

Objetivo: detectar "erros" (ex.: bits trocados) no segmento transmitido

Transmissor:

- trata conteúdo do segmento como sequência de inteiros de 16-bits
- checksum: soma (adição usando complemento de 1) do conteúdo do segmento
- transmissor coloca *complemento do valor da soma* no campo *checksum* do UDP

Receptor:

- calcula *checksum* do segmento recebido
- verifica se o *checksum* calculado bate com o valor recebido:
  - NÃO - erro detectado
  - SIM - nenhum erro detectado. *Mas ainda pode ter erros?* Veja depois ....

Exemplo do *Checksum* Internet

- Exemplo: adição de dois inteiros de 16-bits

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
transbordo	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
soma	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
soma de verificação	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

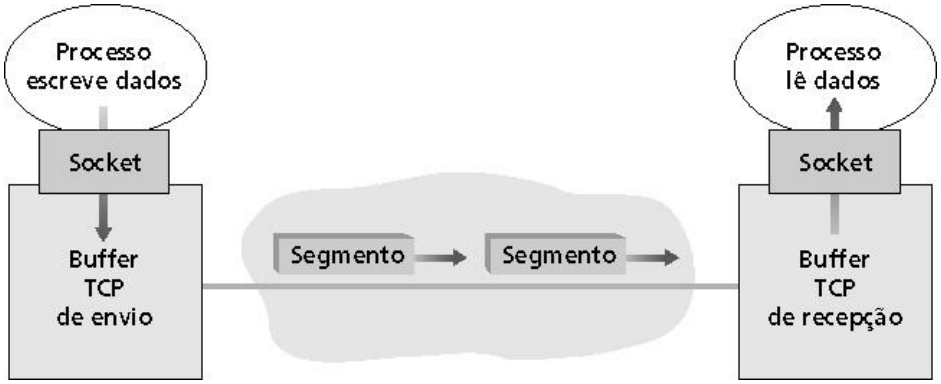
Note que: ao adicionar números, o transbordo (vai um) do bit mais significativo deve ser adicionado ao resultado

Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
  - estrutura do segmento
  - transferência confiável de dados
  - controle de fluxo
  - gerenciamento da conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

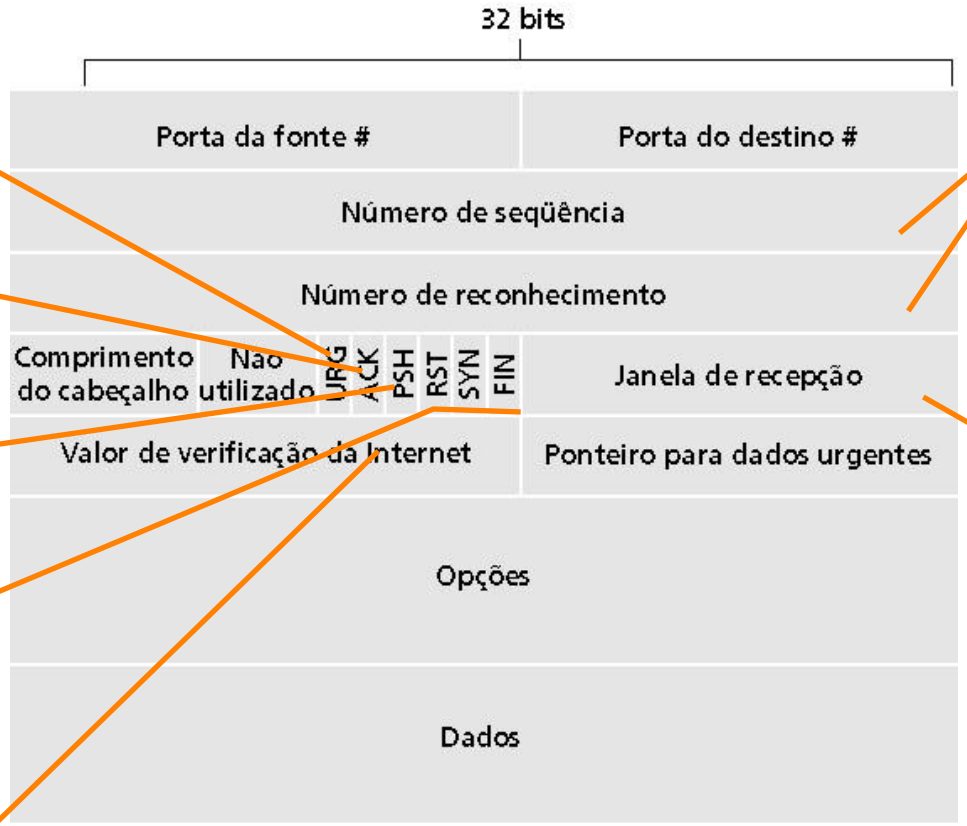
TCP: Visão geral RFCs: 793, 1122, 1323, 2018, 2581

- ponto a ponto:
  - um transmissor, um receptor
- fluxo de bytes, ordenados, confiável:
  - não estruturado em msgs
- com paralelismo (*pipelined*):
  - tam. da janela ajustado por controle de fluxo e congestionamento do TCP



- transmissão full duplex:
  - fluxo de dados bi-direcional na mesma conexão
  - MSS: tamanho máximo de segmento
- orientado a conexão:
  - *handshaking* (troca de msgs de controle) inicia estado do transmissor e do receptor antes da troca de dados
- fluxo controlado:
  - receptor não será afogado pelo transmissor

Estrutura do segmento TCP



URG: dados urgentes (pouco usado)

ACK: campo de ACK é válido

PSH: produz envio de dados (pouco usado)

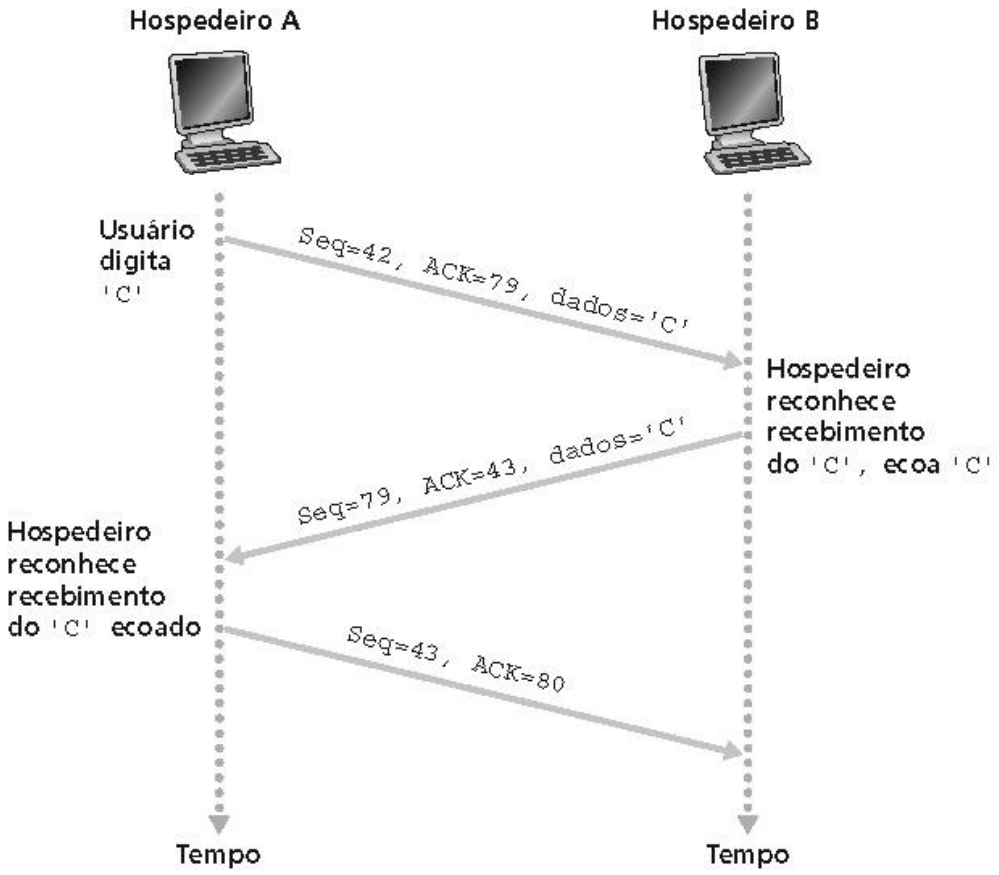
RST, SYN, FIN: estabelec. de conexão (comandos de criação e término)

Internet checksum (como no UDP)

contagem por bytes de dados (não segmentos!)

número de bytes receptor está pronto para aceitar

TCP: nos. de seq. e ACKs



cenário telnet simples

Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
  - estrutura do segmento
  - transferência confiável de dados
  - controle de fluxo
  - gerenciamento da conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

Transferência de dados confiável do TCP

- O TCP cria um serviço rdt sobre o serviço não confiável do IP
  - Segmentos transmitidos em "paralelo" (*pipelined*)
  - Acks cumulativos
  - O TCP usa um único temporizador para retransmissões
- As retransmissões são disparadas por:
  - estouros de temporização
  - acks duplicados
- Considere inicialmente um transmissor TCP simplificado:
  - ignore acks duplicados
  - ignore controles de fluxo e de congestionamento

Eventos do transmissor TCP

Dados recebidos da aplicação:

- Cria segmento com no. de sequência (nseq)
- nseq é o número de sequência do primeiro byte de dados do segmento
- Liga o temporizador se já não estiver ligado (temporização do segmento mais antigo ainda não reconhecido)
- Valor do temporizador: calculado anteriormente

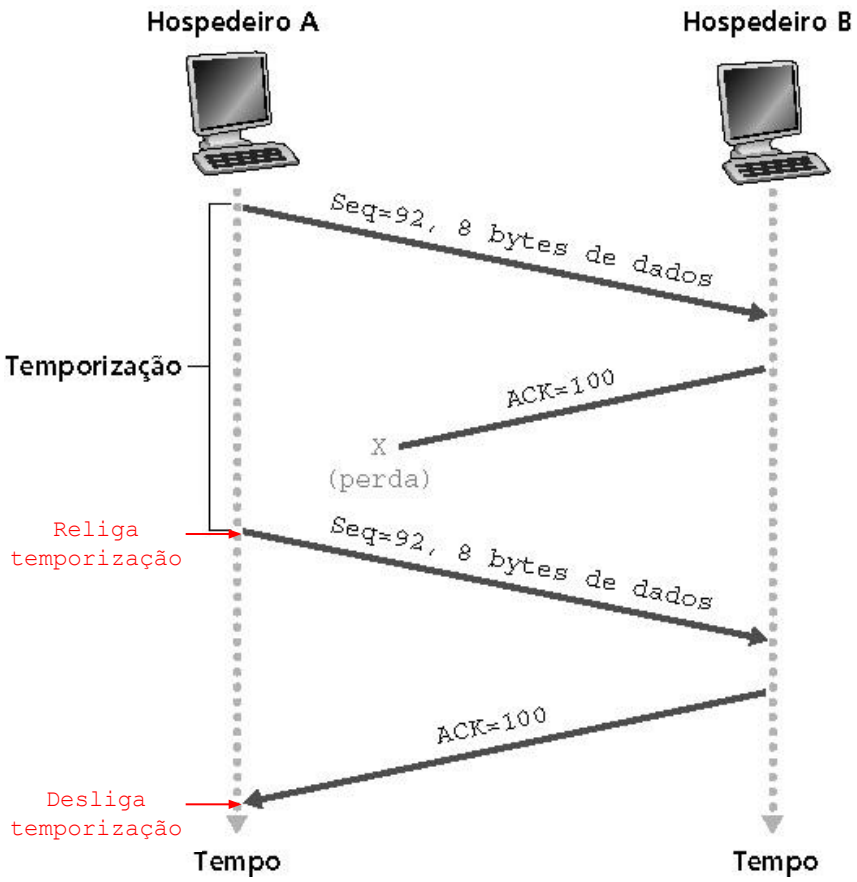
Estouro do temporizador:

- Retransmite o segmento que causou o estouro do temporizador
- Reinicia o temporizador

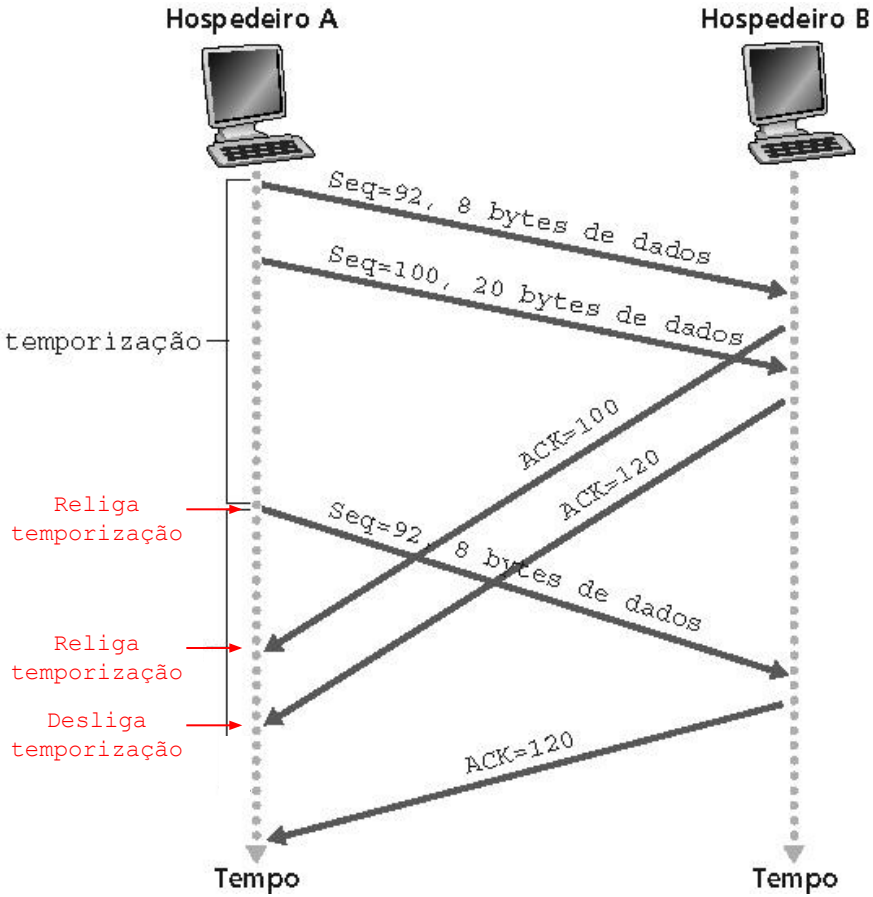
Recepção de Ack:

- Se reconhecer segmentos ainda não reconhecidos
  - atualizar informação sobre o que foi reconhecido
  - religa o temporizador se ainda houver segmentos pendentes (não reconhecidos)

TCP: cenários de retransmissão



Cenário com perda do ACK



Temporização prematura, ACKs cumulativos

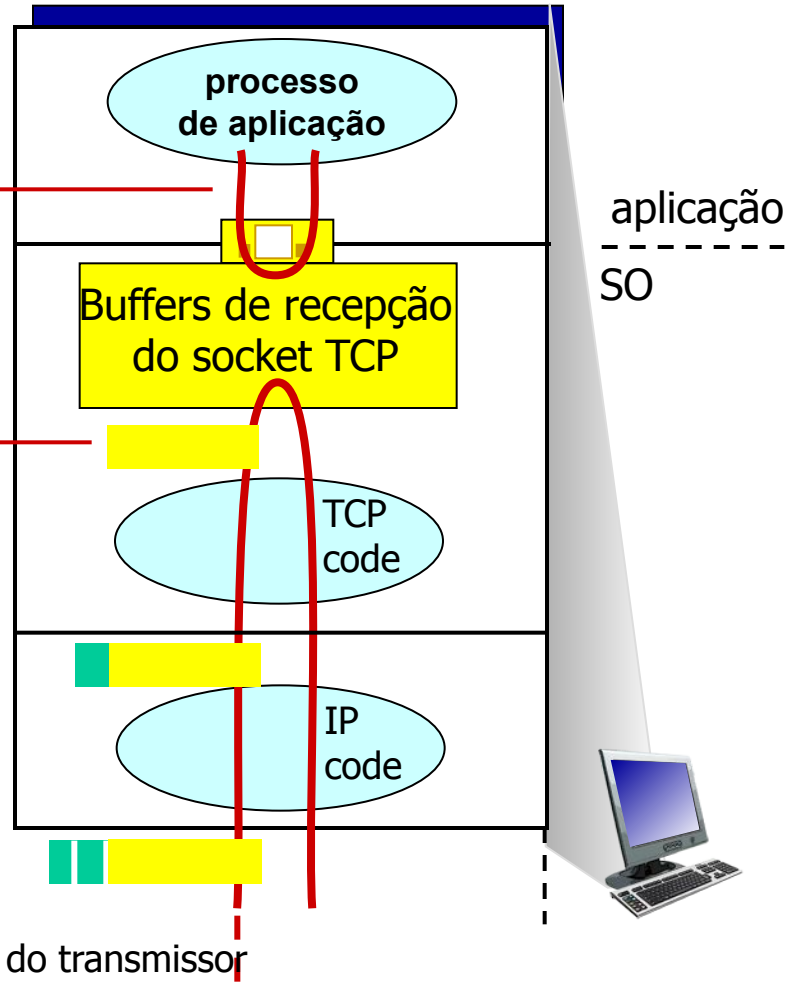
Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
  - estrutura do segmento
  - transferência confiável de dados
  - controle de fluxo
  - gerenciamento da conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

Controle de Fluxo do TCP

a aplicação pode remover dados dos buffers do socket TCP ....

... mais devagar do que o receptor TCP está entregando (transmissor está enviando)



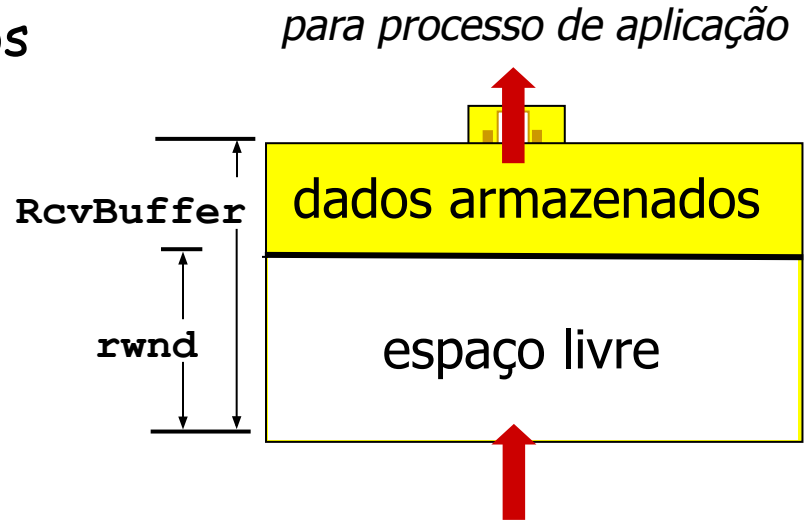
pilha de protocolos no receptor

Controle de fluxo

o receptor controla o transmissor, de modo que este não inunde o buffer do receptor transmitindo muito e rapidamente

Controle de Fluxo do TCP: como funciona

- O receptor "anuncia" o espaço livre do buffer incluindo o valor da **`rwnd`** nos cabeçalhos TCP dos segmentos que saem do receptor para o transmissor
  - Tamanho do **`RcvBuffer`** é configurado através das opções do socket (o valor default é de 4096 bytes)
  - muitos sistemas operacionais ajustam **`RcvBuffer`** automaticamente.
- O transmissor limita a quantidade os dados não reconhecidos ao tamanho do **`rwnd`** recebido.
- Garante que o buffer do receptor não transbordará



armazenamento no lado do receptor

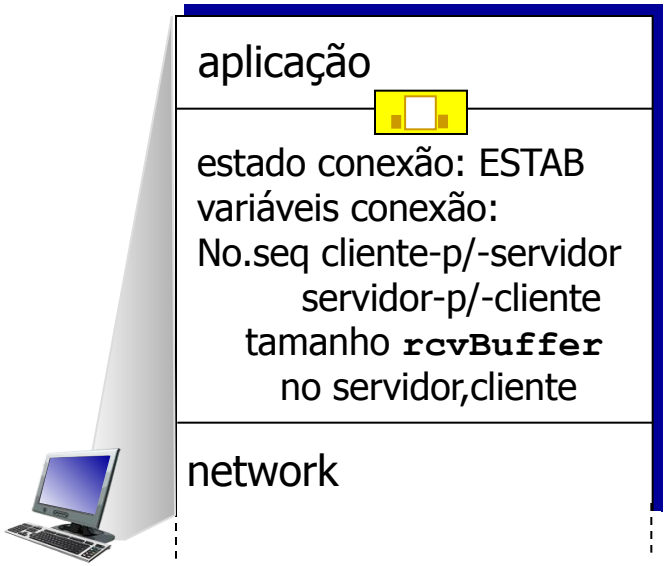
Conteúdo do Capítulo 3

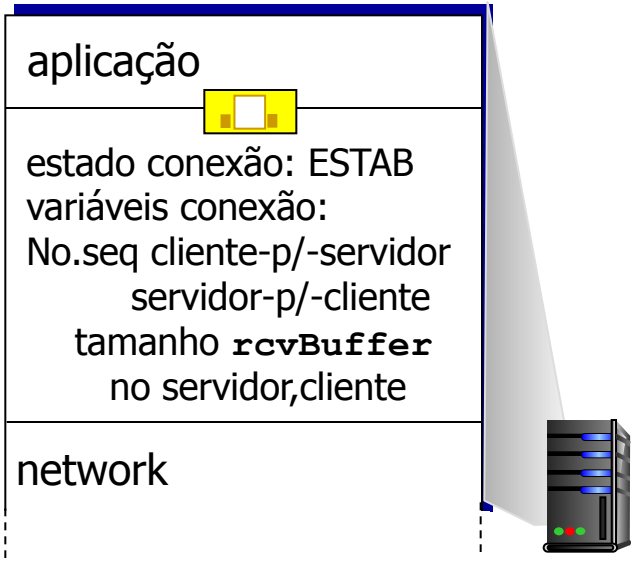
- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
  - estrutura do segmento
  - transferência confiável de dados
  - controle de fluxo
  - gerenciamento da conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

TCP: Gerenciamento de Conexões

antes de trocar dados, transmissor e receptor TCP dialogam:

- concordam em estabelecer uma conexão (cada um sabendo que o outro quer estabelecer a conexão)
- concordam com os parâmetros da conexão.



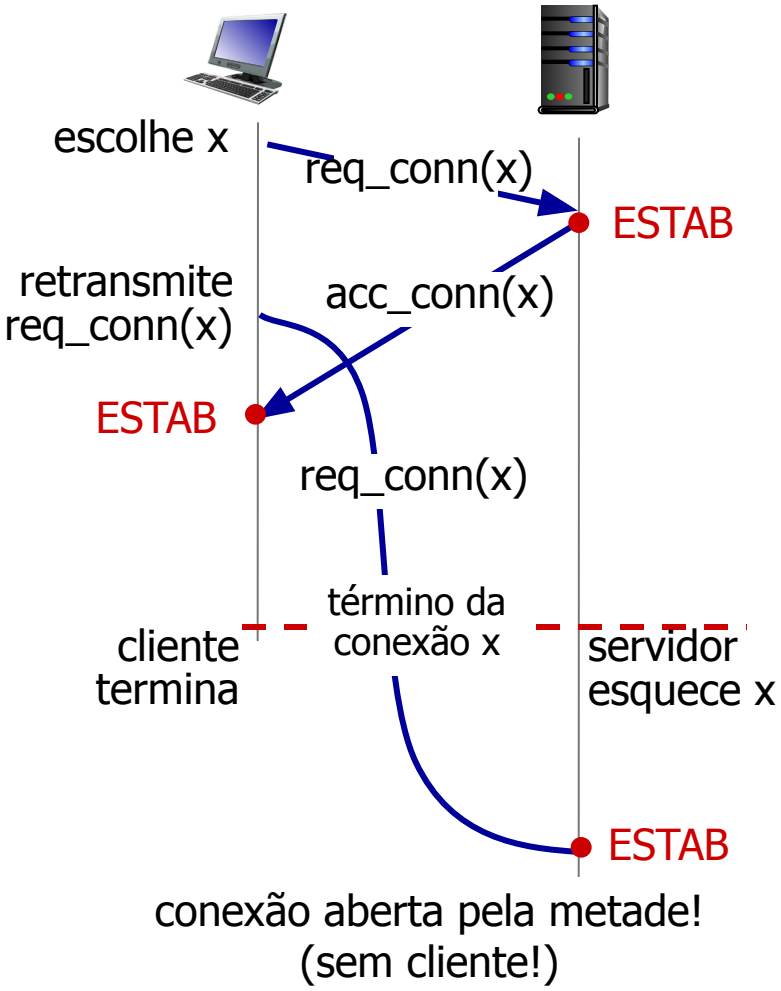


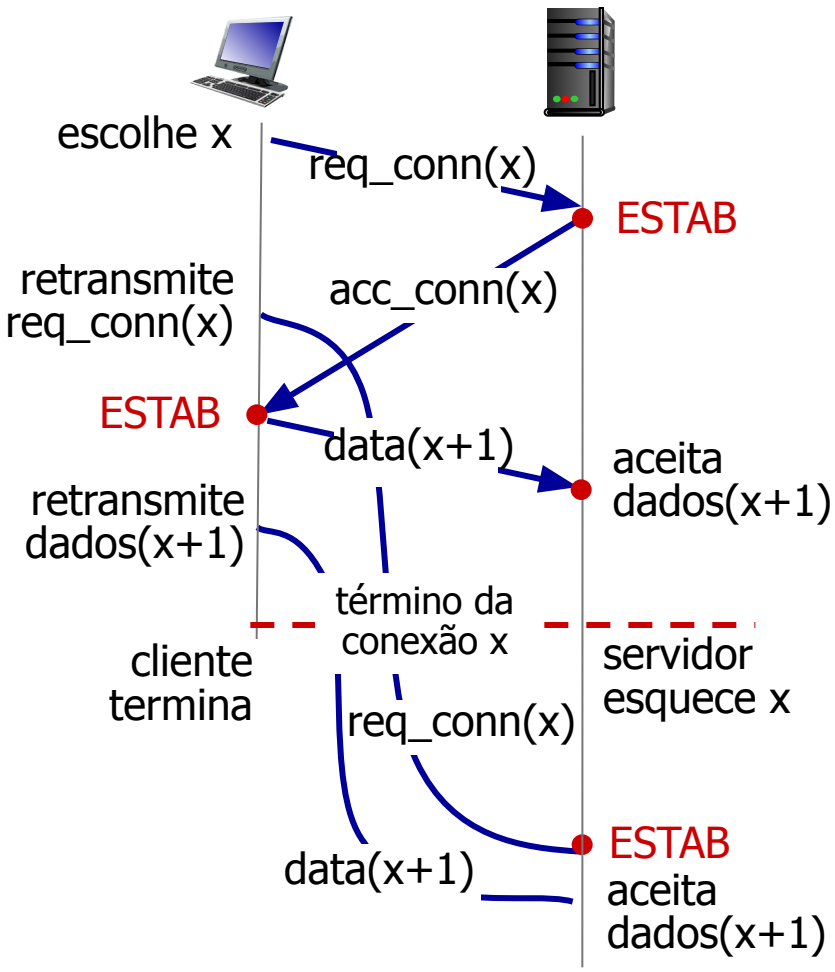

```
Socket clientSocket =
  newSocket("hostname","port
   number");
```

```
Socket connectionSocket =
   welcomeSocket.accept();
```

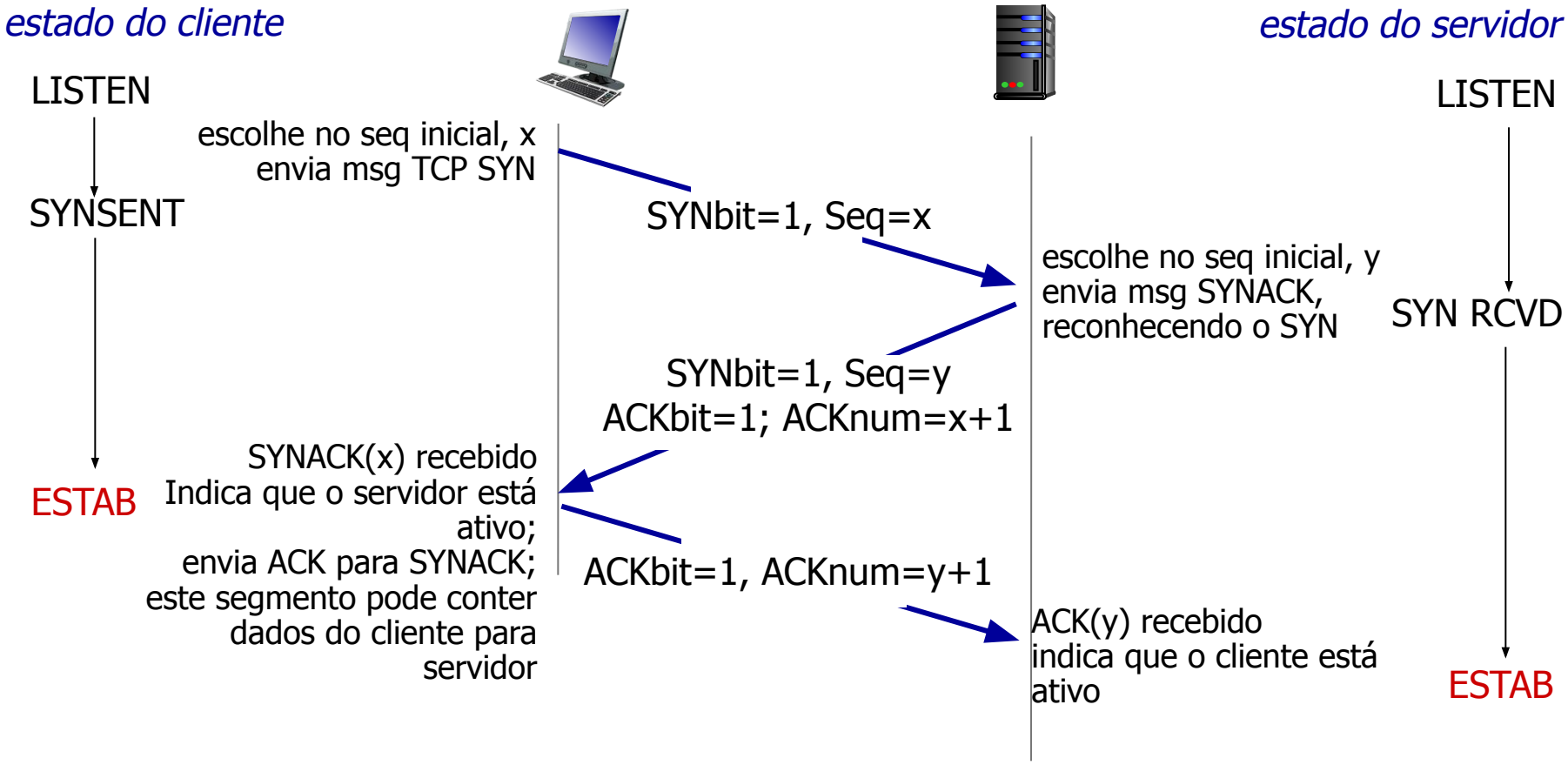
Concordando em estabelecer uma conexão

cenários de falha da apresentação de duas vias:





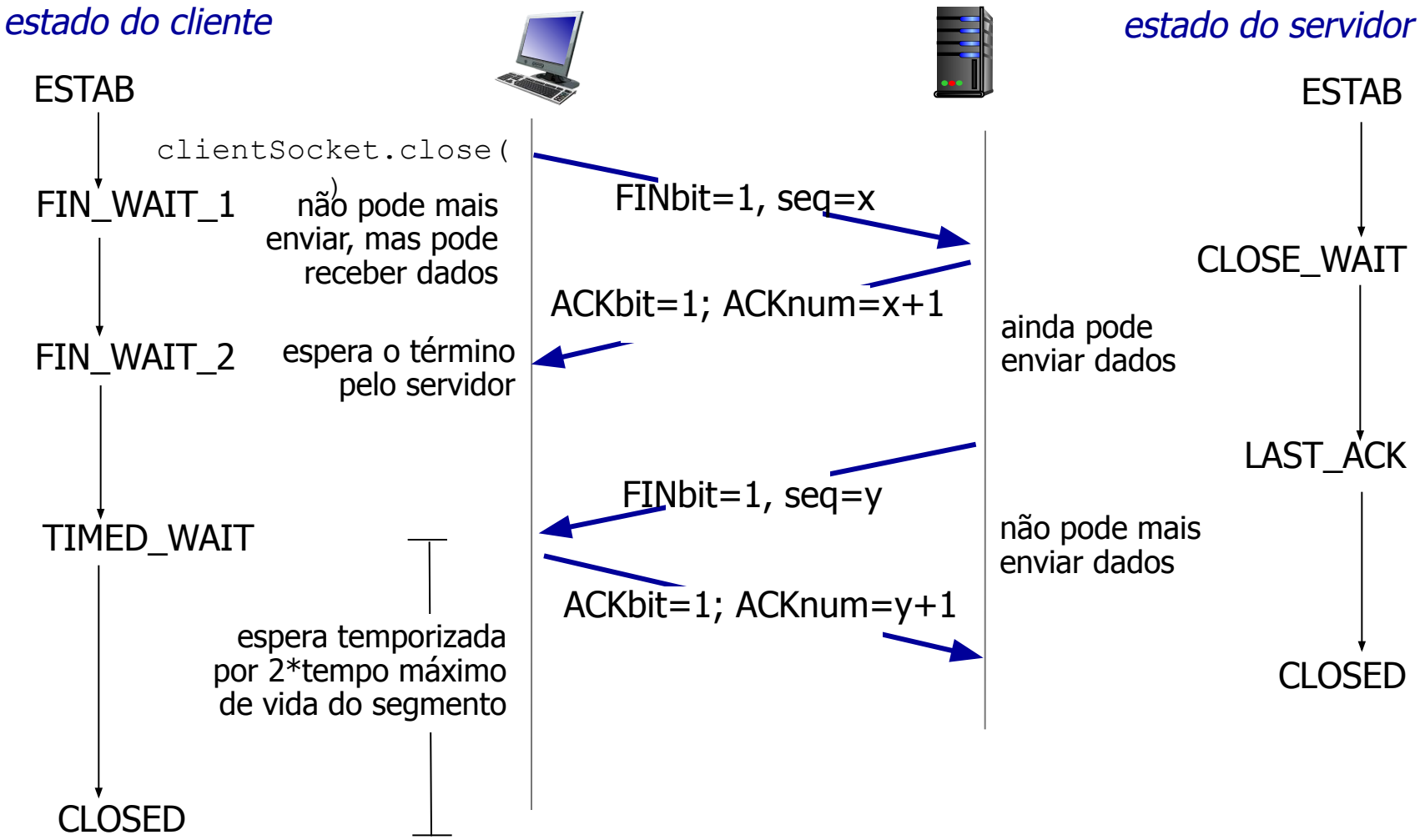
Apresentação de três vias do TCP



TCP: Encerrando uma conexão

- seja o cliente que o servidor fecham cada um o seu lado da conexão
  - enviam segmento TCP com bit FIN = 1
- respondem ao FIN recebido com um ACK
  - ao receber um FIN, ACK pode ser combinado com o próprio FIN
- lida com trocas de FIN simultâneos

TCP: Encerrando uma conexão



Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

Princípios de Controle de Congestionamento

Congestionamento:

- informalmente: "muitas fontes enviando dados acima da capacidade da *rede* de tratá-los"
- diferente de controle de fluxo!
- Sintomas:
  - perda de pacotes (saturação de buffers nos roteadores)
  - longos atrasos (enfileiramento nos buffers dos roteadores)
- um dos 10 problemas mais importantes em redes!

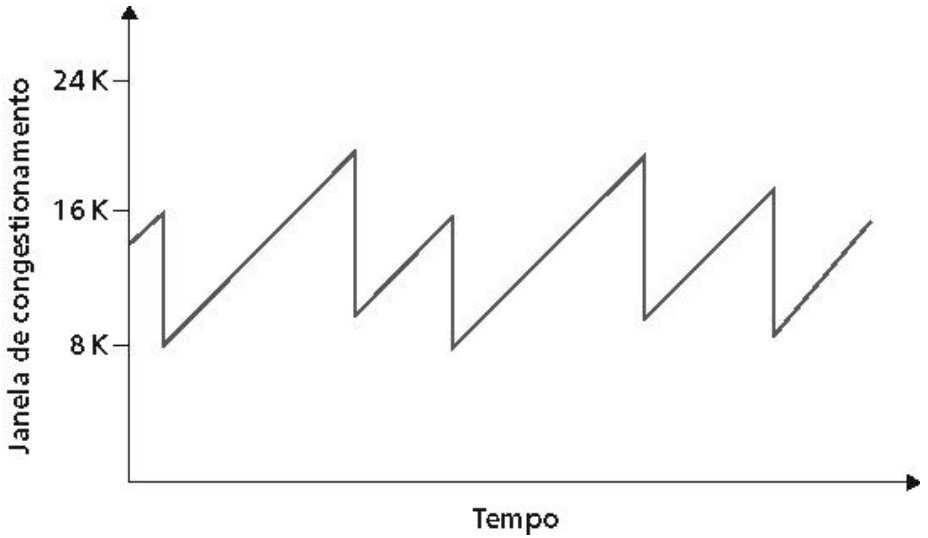
Conteúdo do Capítulo 3

- 3.1 Introdução e serviços de camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado para conexão: UDP
- 3.4 Princípios da transferência confiável de dados
- 3.5 Transporte orientado para conexão: TCP
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento no TCP

Controle de Congestionamento do TCP: aumento aditivo, diminuição multiplicativa

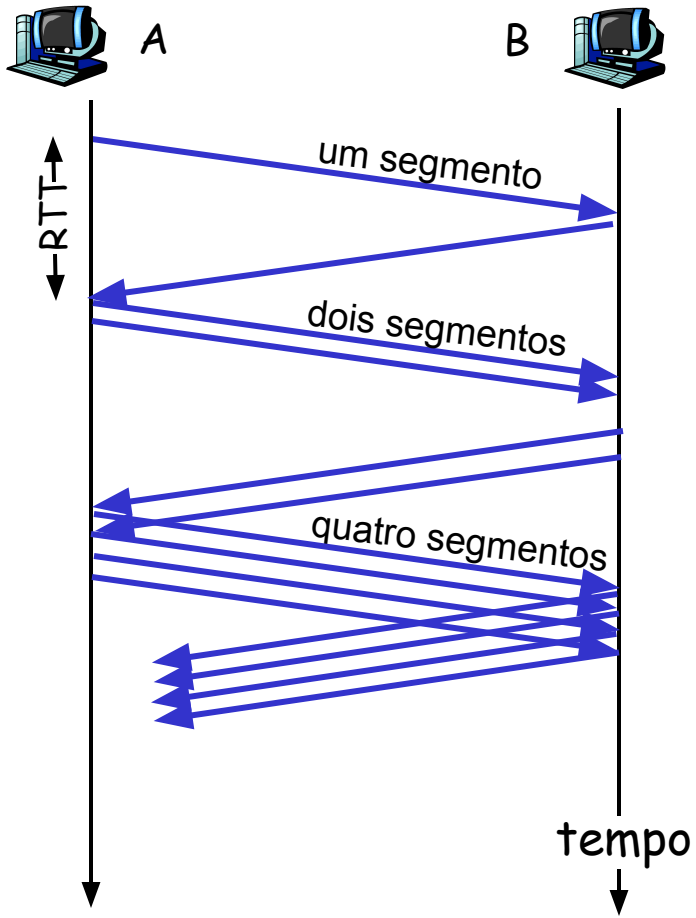
- Abordagem: aumentar a taxa de transmissão (tamanho da janela), testando a largura de banda utilizável, até que ocorra uma perda
  - aumento aditivo: incrementa **cwnd** de 1 MSS a cada RTT até detectar uma perda
  - diminuição multiplicativa: corta **cwnd** pela metade após evento de perda

Comportamento de dente de serra: testando a largura de banda



TCP: Partida lenta

- no início da conexão, aumenta a taxa exponencialmente até o primeiro evento de perda:
  - inicialmente **cwnd** = 1 MSS
  - duplica **cwnd** a cada RTT
  - através do incremento da **cwnd** para cada ACK recebido
- *resumo:* taxa inicial é baixa mas cresce rapidamente de forma exponencial



Notificação Explícita de Congestionamento (ECN)

controle de congestionamento assistido pela rede:

- dois bits no cabeçalho IP (campo ToS) são marcados *pelo roteador de rede* para indicar o congestionamento
- indicação de congestionamento é levada até o receptor
- o receptor (vendo a indicação de congestionamento) seta o bit ECE no segmento de reconhecimento para notificar o transmissor sobre o congestionamento.