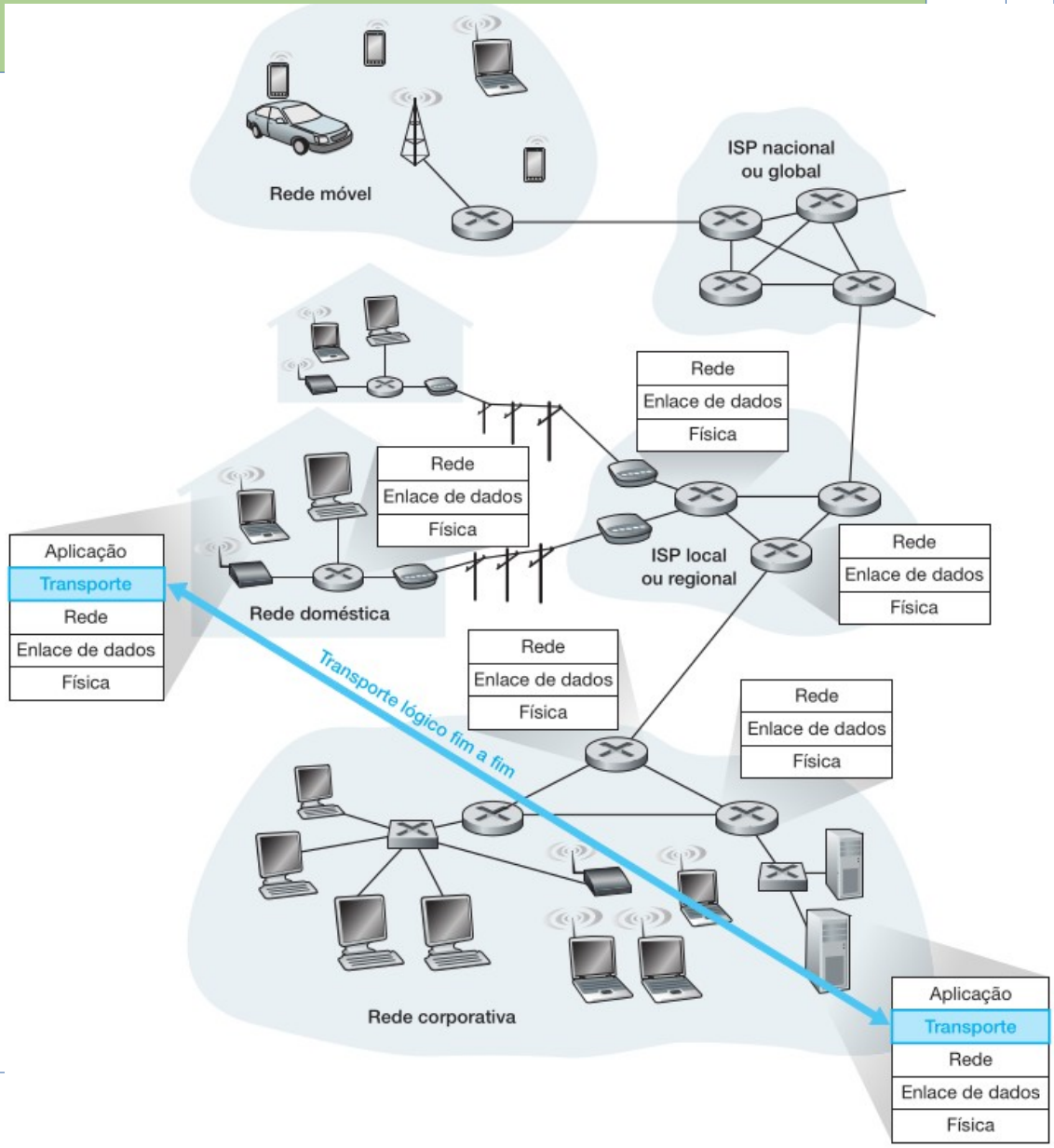
Camada de Transporte



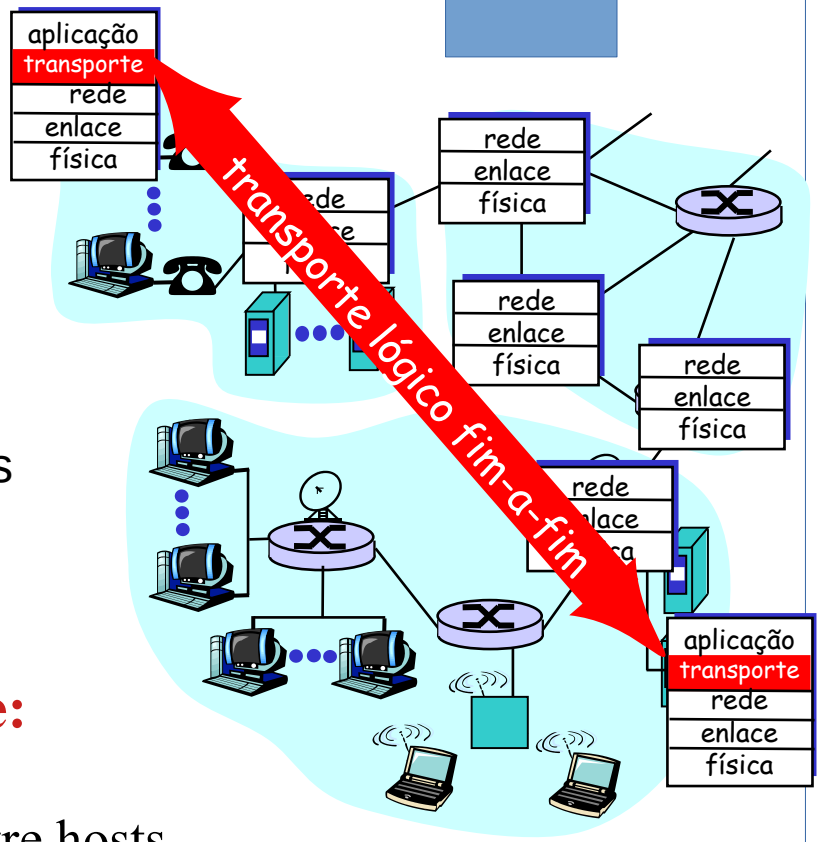
Camada de Transporte

Objetivos da aula:

- Entender os princípios da camada de transporte e seus principais serviços:
  - multiplexação/demultiplexação
  - transporte sem conexão: UDP
  - transporte orientado à conexão: TCP
    - transferência confiável
    - controle de fluxo
    - gerenciamento de conexão
    - controle de congestionamento

Protocolos e Serviços de Transporte

- Serviços de transporte fornecem comunicação lógica entre processos de aplicação em diferentes hosts.
- Esses serviços, que seguem as regras dos protocolos de transporte, são executados nos sistemas finais (hosts) da rede
- A camada de transporte utiliza e aprimora os serviços oferecidos pela camada de rede



Serviço de Transporte x Serviço de rede:

- **camada de rede**: transferência de dados entre hosts
- **camada de transporte:** transferência de dados entre processos

Serviços de Transporte do TCP/IP

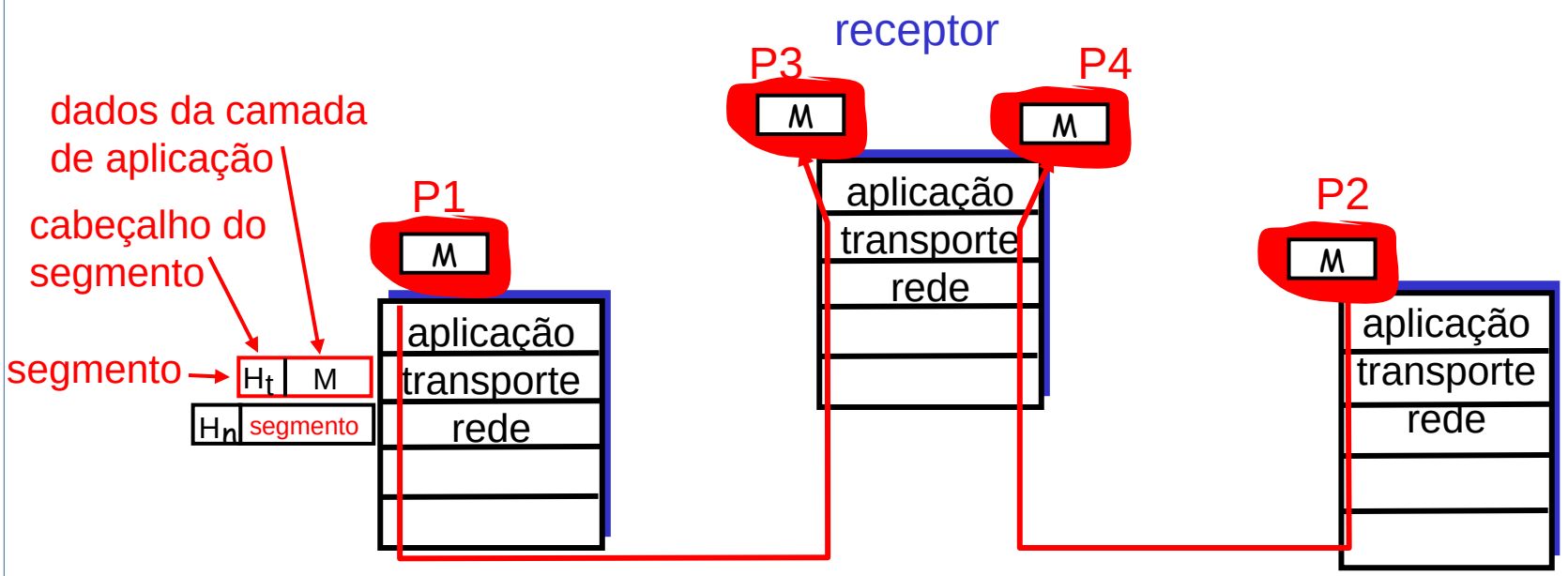
- **TCP** - confiável, sequencial e unicast
  - orientado à conexão
  - controle de congestionamento
  - controle de fluxo
- **UDP** - não confiável (“melhor esforço”), não sequencial, entrega unicast ou multicast.
- **Serviços não disponíveis na Internet:**
  - entrega em tempo real
  - garantia mínima de banda
  - multicast confiável

Multiplexação de Aplicações

Segmento é a unidade de dados trocada entre entidades da camada de transporte

- TPDU: transport protocol data unit (unidade de dados do protocolo de transporte)

Demultiplexação: entrega de segmentos recebidos aos processos de aplicação corretos

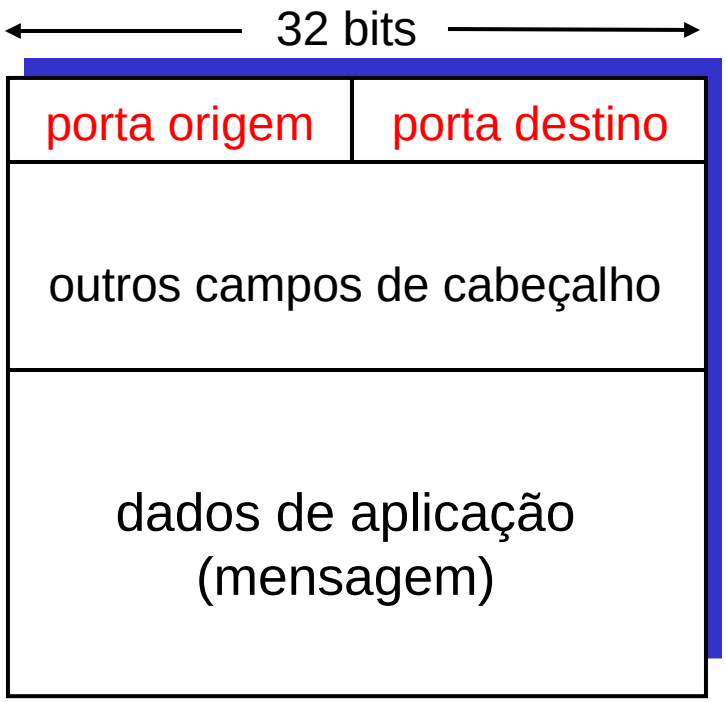


Multiplexação de Aplicações

Multiplexação:
reunir dados de múltiplos processos de aplicação, juntar cabeçalhos com informações para demultiplexação

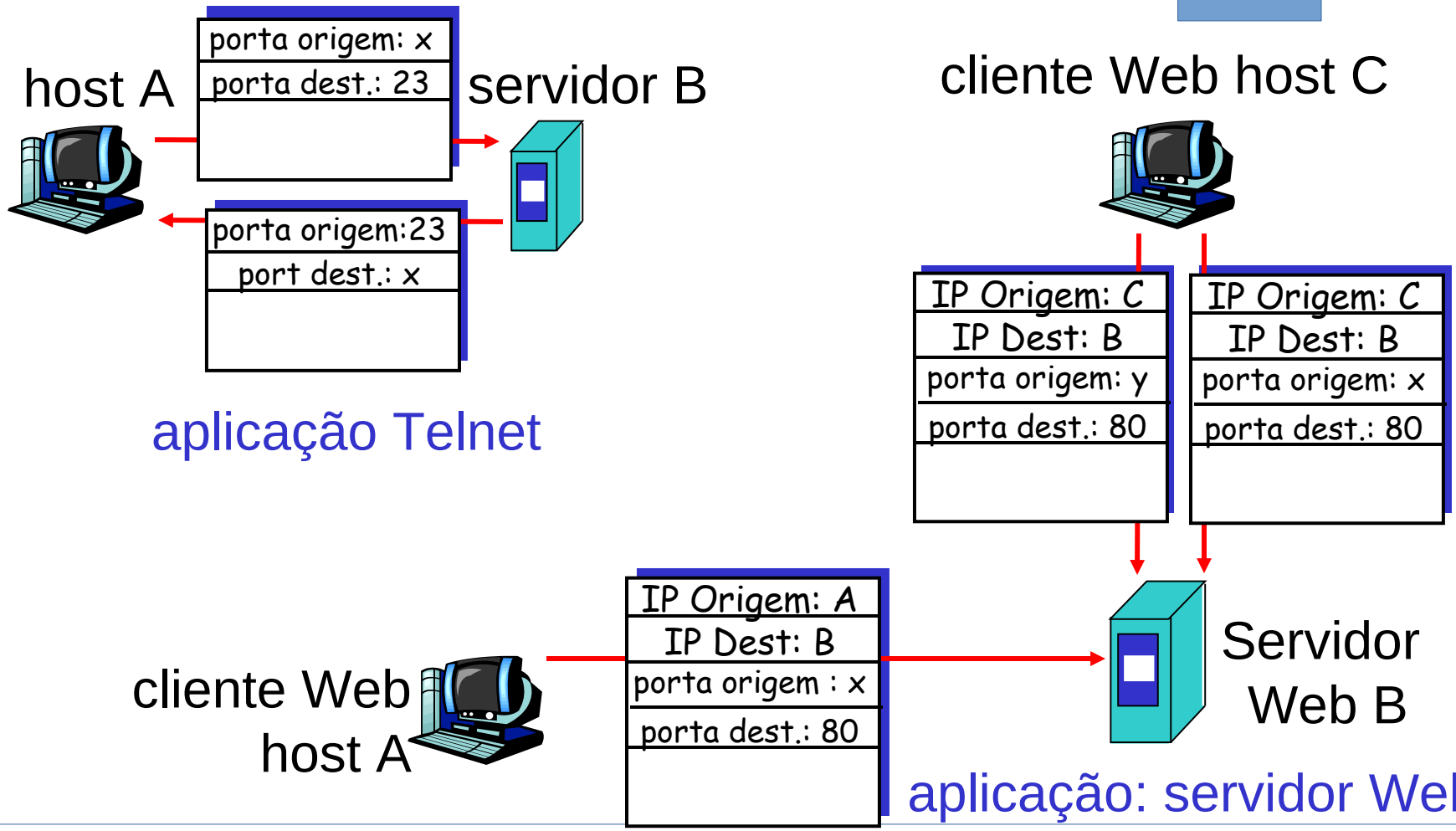
multiplexação/demultiplexação:

- **UDP**: demultiplexação é baseada apenas em número da porta de destino
- **TCP**: demultiplexação depende de 4 valores: IP de origem, IP de destino, porta de origem e porta de destino
- números de porta origem e destino inseridas no cabeçalho de cada segmento



formato do segmento TCP/UDP

Multiplexação: exemplos



UDP: User Datagram Protocol [RFC 768]

- protocolo de transporte da Internet “minimalista”
- serviço baseado em “melhor esforço”
- segmentos UDP podem ser entregues fora de ordem ou serem perdidos.
- *sem conexão:*
  - não há apresentação entre transmissor e receptor.
  - cada segmento UDP é tratado de forma independente dos outros

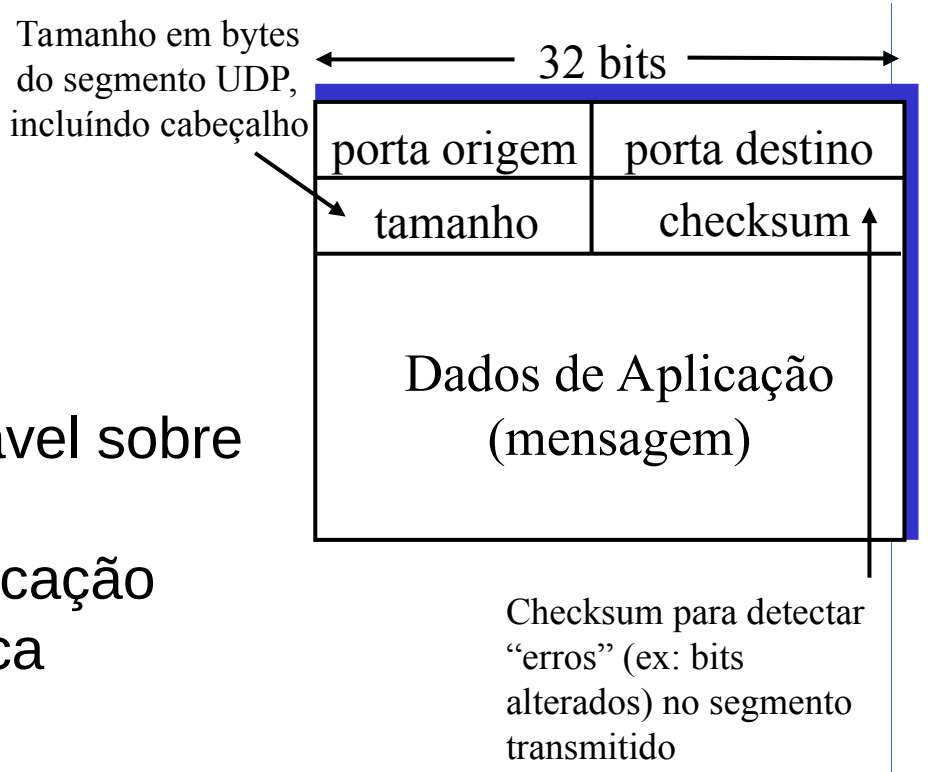
Porque existe o UDP?

- não há conexão prévia (conexão gera atraso)
- simples: não há estado de conexão
- cabeçalho de segmento é reduzido (em relação ao TCP)
- não há controle de fluxo ou congestionamento: UDP envia segmentos tão rápido quanto desejado (e possível, pois a rede pode limitar)

Mais sobre UDP

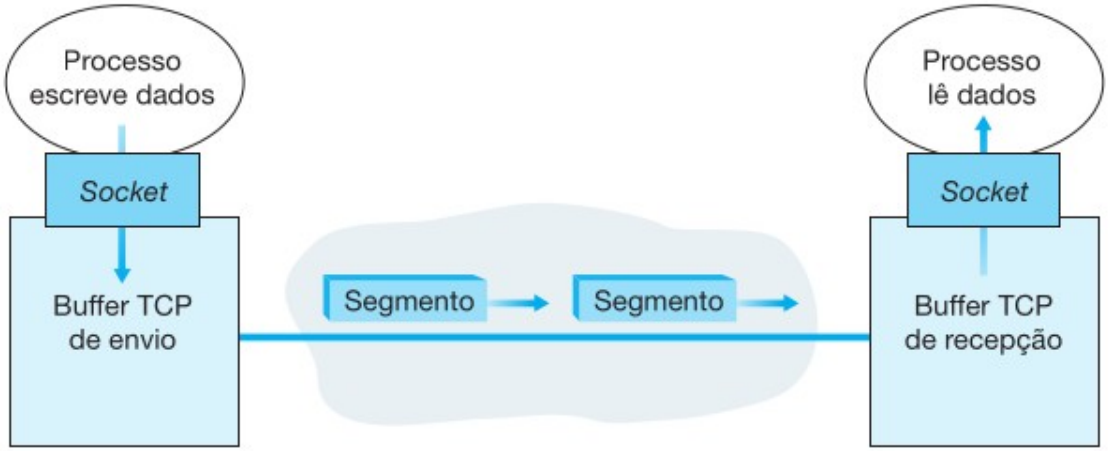
- muito usado em aplicações de multimídia contínua (streaming)
  - tolerantes à perda
  - sensíveis à taxa
- E também por aplicações que se recuperam facilmente de erros:
  - Ex: DNS,DHCP,SNMP.
- Para realizar transferência confiável sobre UDP é necessário acrescentar confiabilidade na camada de aplicação
  - recuperação de erro específica de cada aplicação
  - é o que ocorre no HTTP/3 com QUIC

formato do segmento UDP

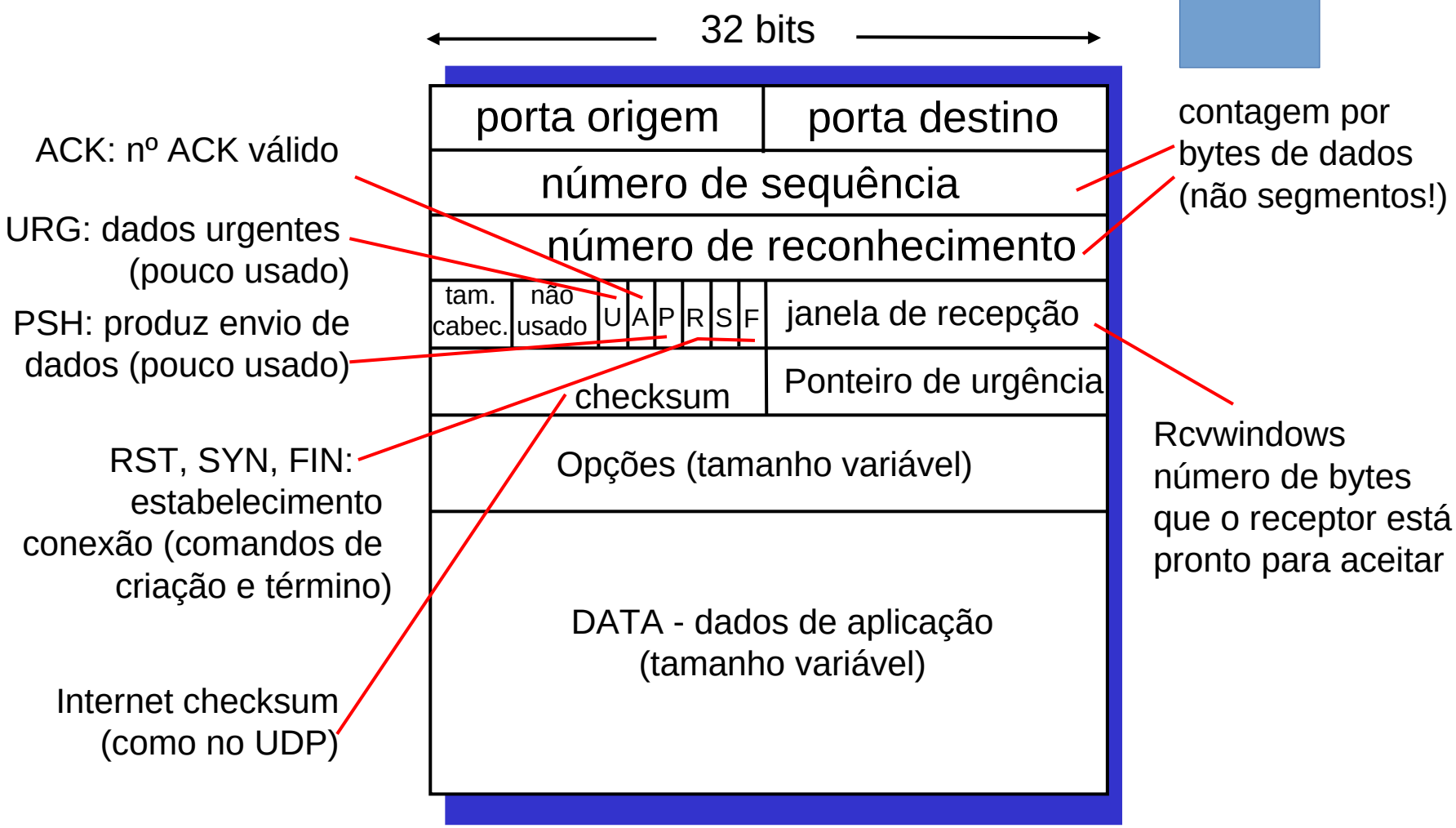


TCP: Overview RFCs: 793, 1122, 2018, 5681, 7323

- Orientado a conexão (handshaking)
  - Troca de msgs de controle antes da troca de dados
- Ponto a ponto:
  - um transmissor e um receptor
- Entrega confiável e sequencial:
  - Quantidade de bytes para definir sequência
- Serviço full-duplex:
  - transmissão bidirecional na mesma conexão
- Paralelismo:
  - Controle de congestionamento e fluxo definem tamanho da janela
- Uso de buffers de envio e recepção
  - Usa maximum segment size - MSS
- controle de fluxo:
  - Emissor não sobrecarrega o receptor
- controle de congestionamento:
  - Evita sobrecarregar a rede como um todo



Estrutura do Segmento TCP



Campos do Segmento TCP

- **Nº de sequência e nº de reconhecimento** - 32 bits cada para sequencializar e confirmar as entregas.
- **Porta origem/destino -** 16 bits cada para indicar as portas utilizadas
- **Janela de recepção** - 16 bits para controle de fluxo, indica o espaço que o transmissor possui livre para aceitar novos segmentos.
- **Comprimento de cabeçalho** - 4 bits - define o tamanho do cabeçalho (que é variável) em palavras de 32 bits. Tipico são 20 bytes, mas pode ser maior
- **Opções** – Geralmente vazio, é opcional e de tamanho variável.
- **Data** – contém os dados.
- **FLAGS** - contém 8 bits que podem representar:
  - SYN, FIN e RST– Início, finalização e restabelecimento da conexão
  - ACK – confirmação de recebimento
  - PSH – dados devem ser enviados imediatamente para a aplicação
  - URG – indica que há dados urgentes

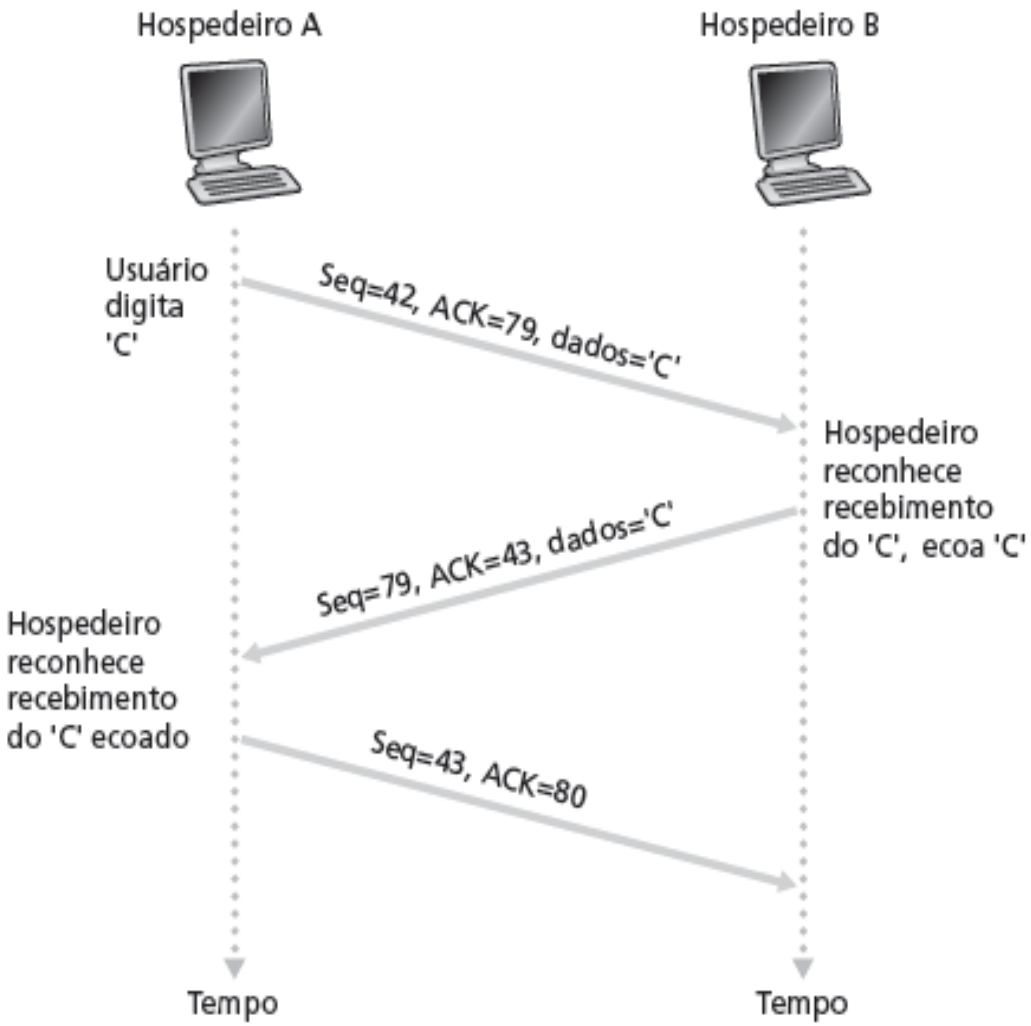
Números de Sequência e ACKs do TCP

Números de sequência: nº do 1º byte no segmentos de dados

ACKs (Acknowledgement):

- Pacote de confirmação
- nº do próximo byte esperado do outro lado
- ACK cumulativo
- A forma como o receptor trata segmentos fora de ordem fica a critério do desenvolvedor.

TCP é full duplex, então há numerações independentes nc fluxo de A para B, e de B para A

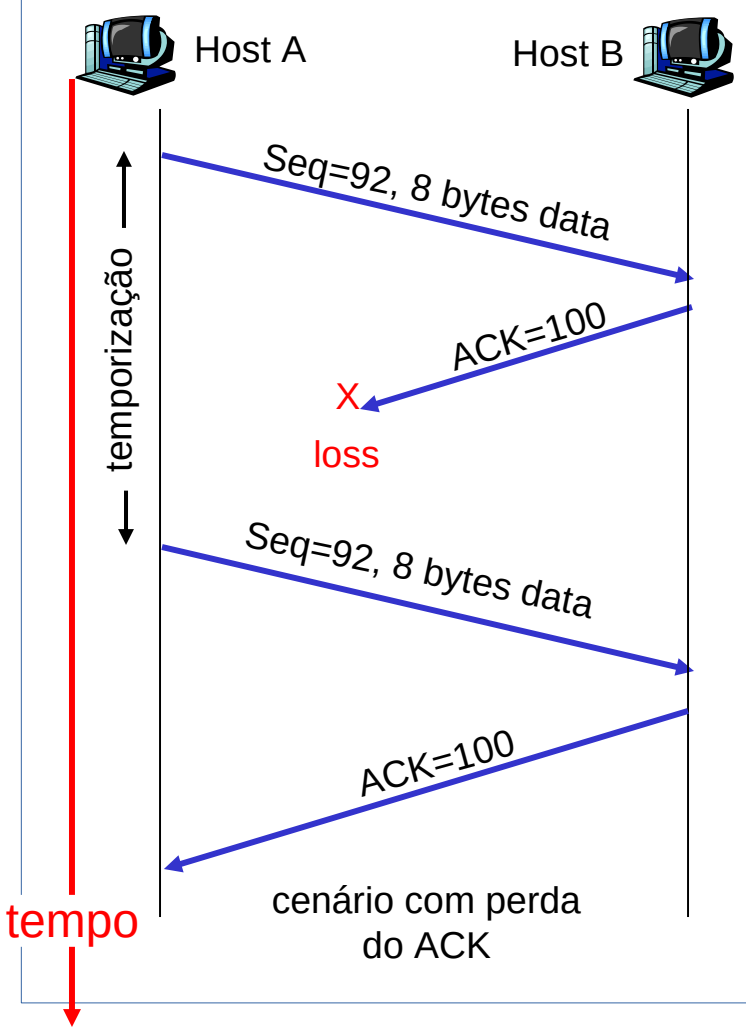


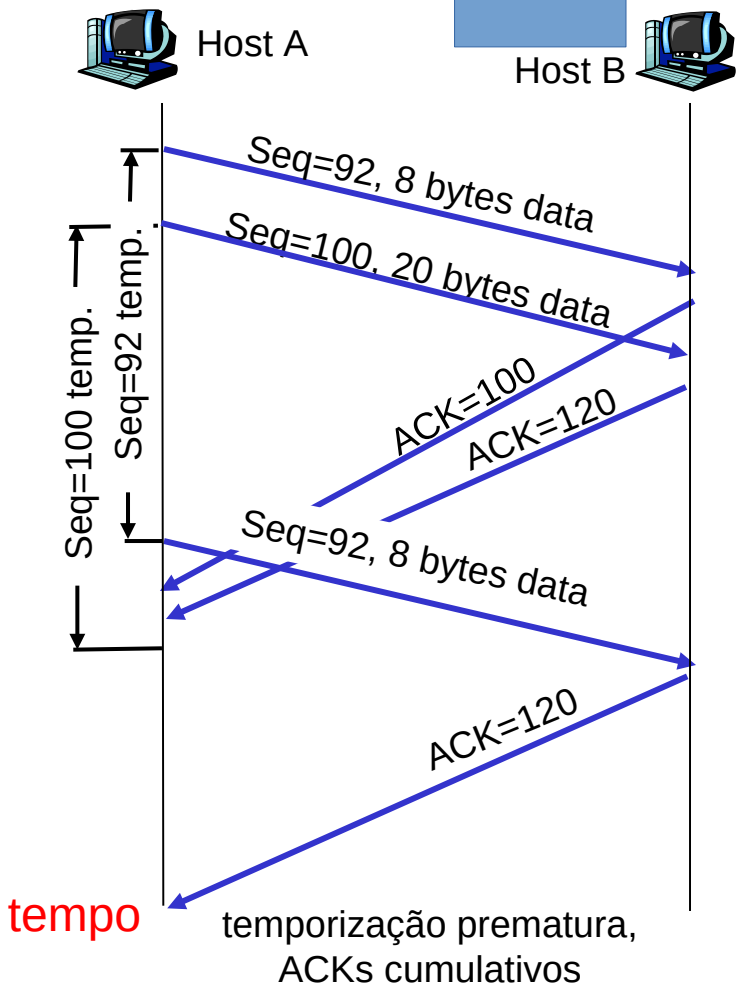
cenário telnet simples

Geração de ACK [RFC 1122, RFC 2581]

Evento	Ação do TCP Receptor
segmento chega em ordem, não há lacunas, segmentos anteriores já aceitos	ACK retardado. Espera até 500ms pelo próximo segmento. Se não chegar, envia ACK
segmento chega em ordem, não há lacunas, um ACK atrasado pendente	imediatamente envia um ACK cumulativo
segmento chega fora de ordem número de sequência chegou maior: gap detectado	envia ACK duplicado, indicando número de sequência do próximo byte esperado
chegada de segmento que parcial ou completamente preenche o gap	reconhece imediatamente se o segmento começa na borda inferior do gap

TCP: cenários de retransmissão





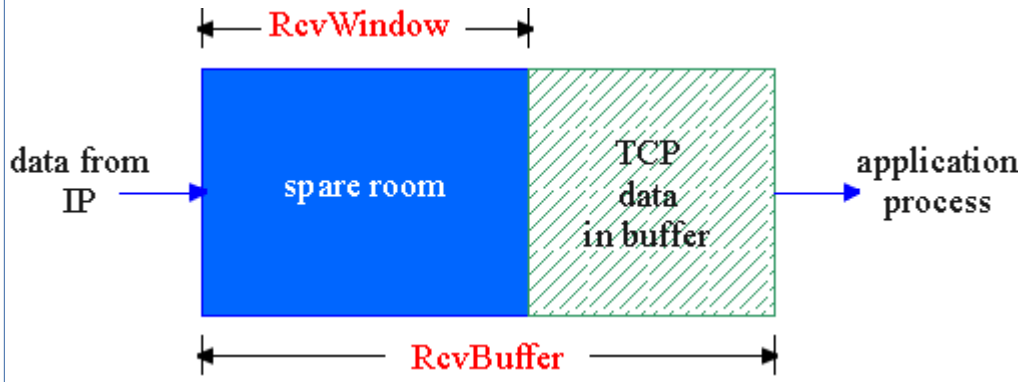
TCP: Controle de Fluxo

controle de fluxo

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

RcvBuffer = tamanho do Buffer de recepção do TCP

RcvWindow = total de espaço livre no buffer



armazenamento de recepção

receptor: informa ao transmissor valor da área livre no buffer (varia constantemente)

- **campo RcvWindow** no segmento TCP

transmissor: mantém a quantidade de dados transmitidos, mas ainda não reconhecidos, menor que o último **RcvWindow** recebido

TCP Estabelecimento de Conexão

TCP transmissor estabelece conexão com o receptor antes de trocar segmentos de dados

- inicializar variáveis: números de sequência, buffers, controle de fluxo (ex. **`RcvWindow`**)
- *cliente:* inicia a conexão

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

- *servidor:* chamado pelo cliente

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

Three way handshake - 3 Passos:

1: cliente envia TCP SYN ao servidor
  - especifica nº de sequência inicial

2: servidor recebe o SYN, e responde com segmento SYNACK
  - reconhece o SYN recebido
  - aloca buffers
  - especifica o nº de sequencia inicial do servidor

3: o sistema final cliente reconhece o SYNACK recebido com ACK

TCP Término de Conexão

Fechando uma conexão, passos:

cliente fecha o socket: `clientSocket.close();`

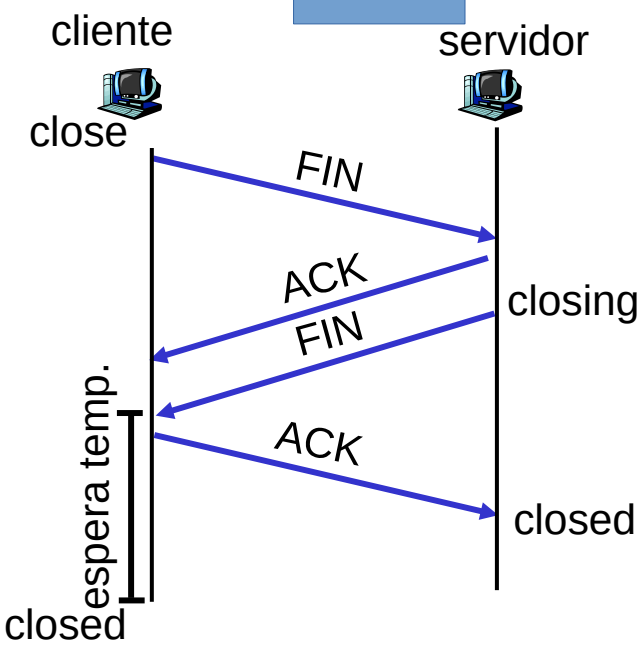
1: cliente envia o segmento TCP FIN ao servidor

2: servidor recebe FIN, responde com ACK. Fecha a conexão, envia FIN. (ACK e FIN pode estar no mesmo segmento)

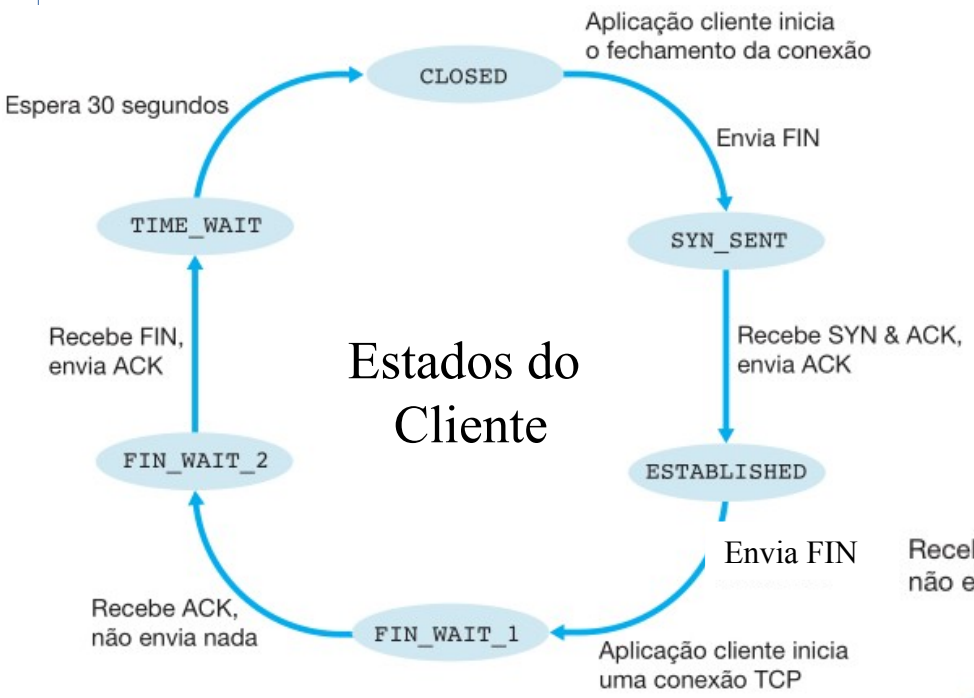
3: cliente recebe FIN, responde com ACK.

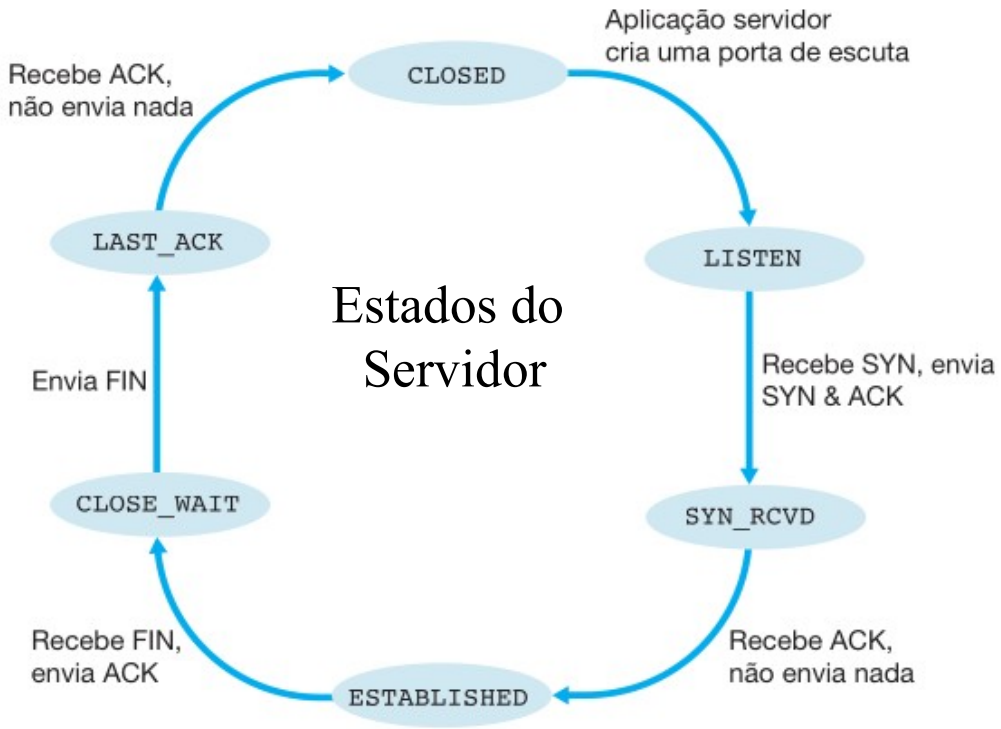
Entra em “espera temporizada” - vai responder com ACK a FINs recebidos

4: servidor, recebe ACK. Conexão fechada.



TCP Controle de Conexão





Princípios de Controle de Congestionamento

Congestionamento:

- informalmente: “muitas fontes enviando dados acima da capacidade da *rede* tratá-los”
- diferente de controle de fluxo!
- sintomas:
  - perda de pacotes (saturação de buffer nos roteadores)
  - atrasos grandes (filas nos buffers dos roteadores)
- um dos problemas mais importantes na Internet!

Abordagens do controle de congestionamento

Existem duas abordagens gerais para o problema de controle de congestionamento:

Controle de congestionamento fim-a-fim:

- não usa realimentação explícita da rede
- congestionamento é inferido a partir das perdas e dos atrasos observados nos sistemas finais
- abordagem usada pelo TCP

Controle de congestionamento assistido pela rede:

- roteadores enviam informações para os sistemas finais
  - bit único indicando o congestionamento (SNA, DECbit, TCP/IP ECN, ATM)
  - taxa explícita do transmissor poderia ser enviada

TCP: Controle Congestionamento

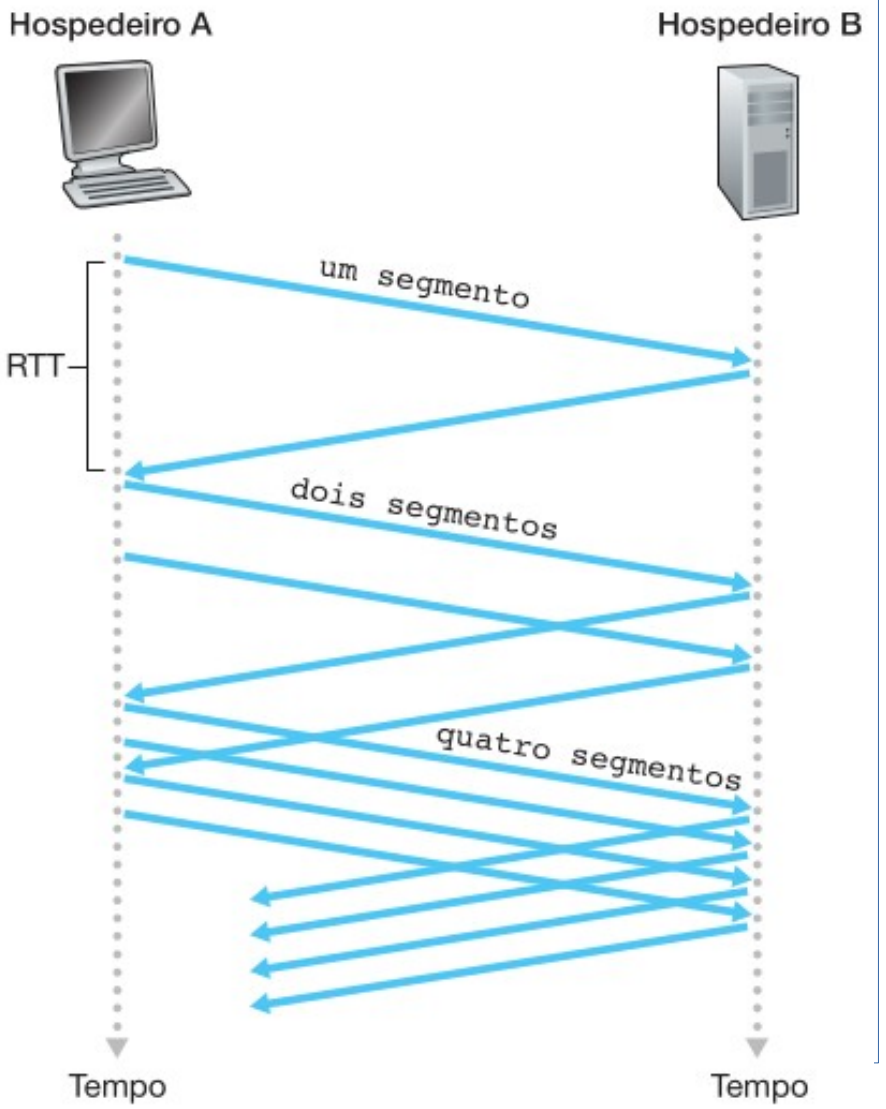
- Utiliza controle fim-a-fim (não há assistência da rede)
- A taxa máxima de transmissão é limitada pelo tamanho da janela de congestionamento (congwin).
- Para isso, há um “teste” para reconhecer a taxa ideal, em que é possível transmitir sem que haja perdas.
  - Aumenta o tamanho da janela até ocorrer perda.
  - Se houver perda, diminui o tamanho da janela e volta a aumentar aos poucos, conforme as confirmações de entrega são recebidas.
- Utiliza algoritmos slow start e congestion avoidance.
- Há variações nas regras para cada implementação TCP (Ex: Tahoe e Reno).

TCP Slowstart

algoritmo Slowstart

inicializar: Congwin = 1
para (cada segmento reconhecido)
    Congwin++
até (evento perda OU
    CongWin > threshold)

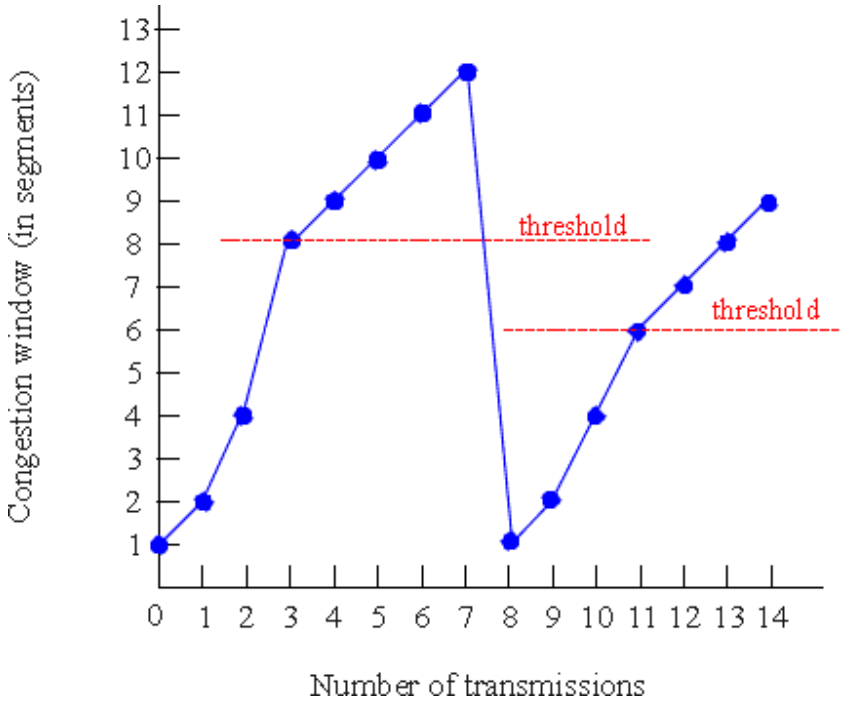
- Congwin → Janela de congestionamento
- Threshold → define o limite entre a fase slow start e a fase congestion avoidance



TCP: Congestion Avoidance

Congestion avoidance

```
/* acabou slowstart        */
/* Congwin > threshold */
Até (evento perda) {
  cada w segmentos reconhecidos:
      Congwin++
  }
threshold = Congwin/2
Congwin = 1
realiza slowstart
```



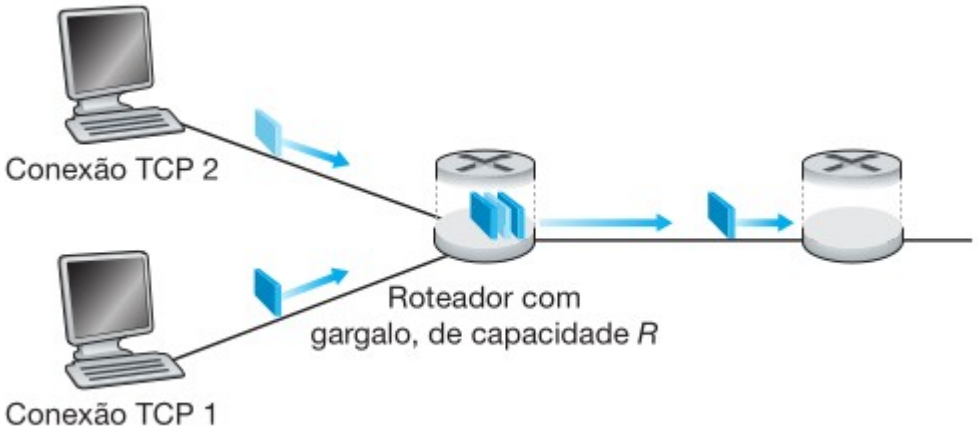
1: TCP Reno pula a fase slowstart após três ACKs duplicados

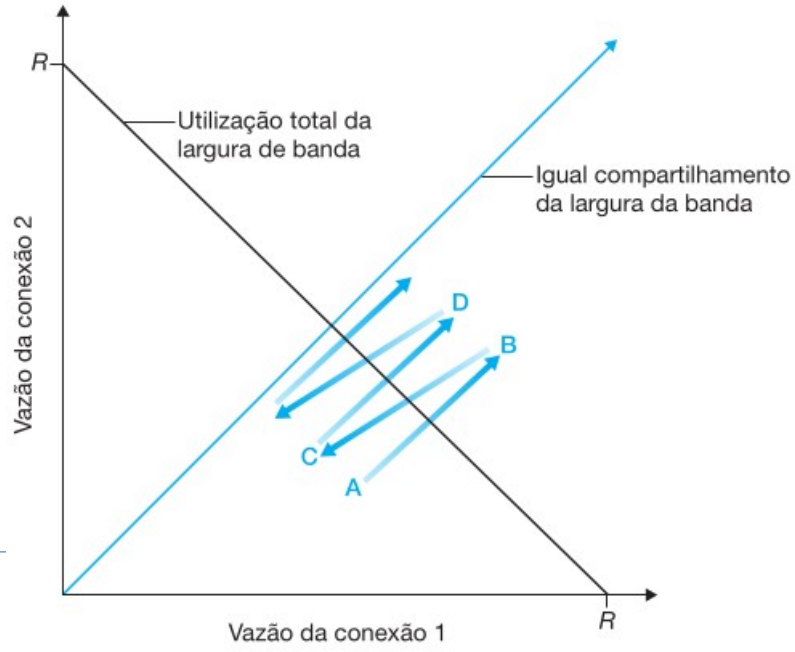
TCP Equidade

Objetivo: se N sessões TCP devem passar pelo mesmo gargalo, cada uma deve obter 1/N da capacidade do enlace, dividindo igualmente.

Garantido com o controle de congestionamento do TCP, que é baseado em AIMD: aumento aditivo, redução multiplicativa

- aumenta a janela de 1 a cada RTT
- Perda reduz a janela por um fator de 2

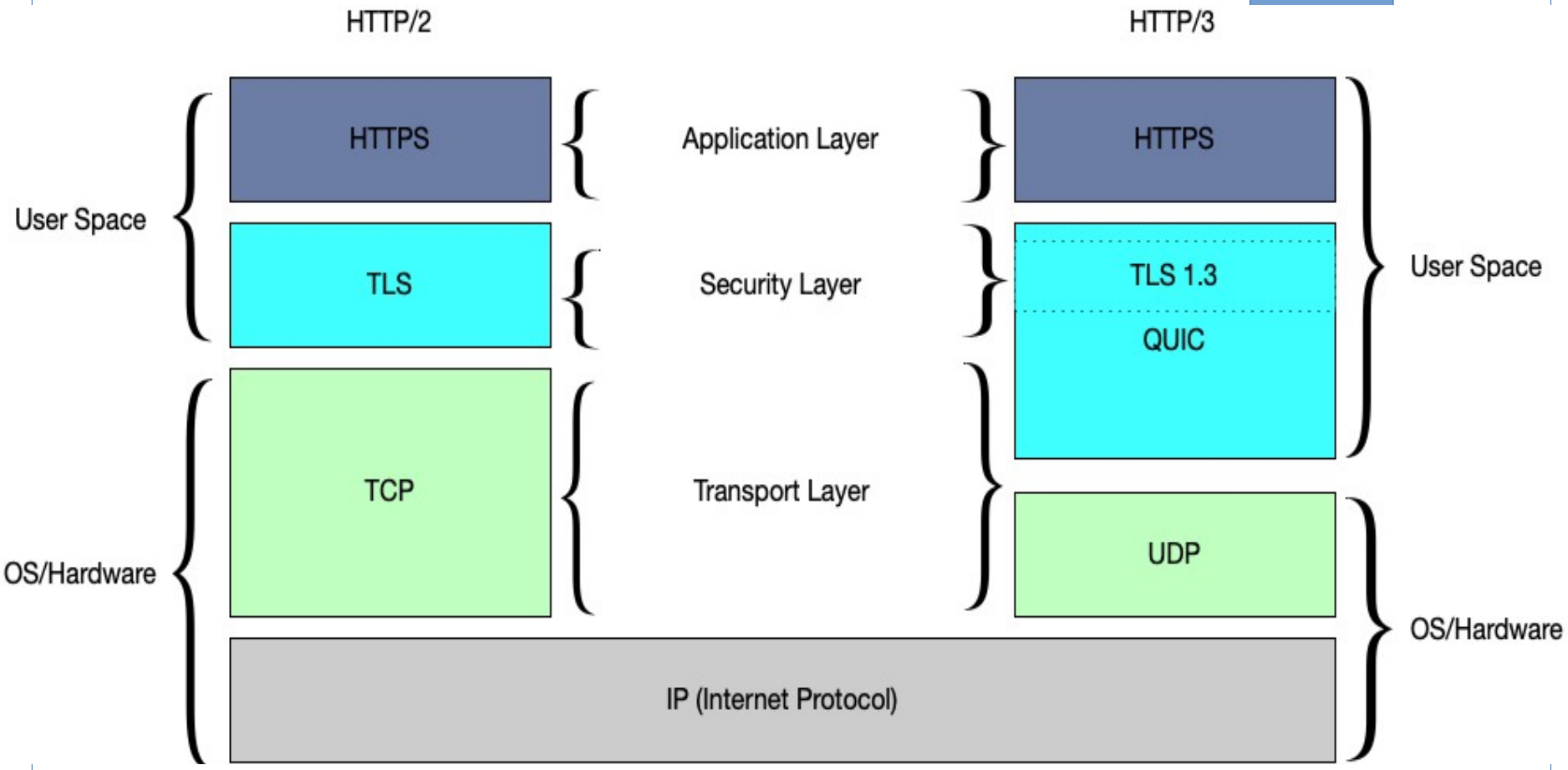




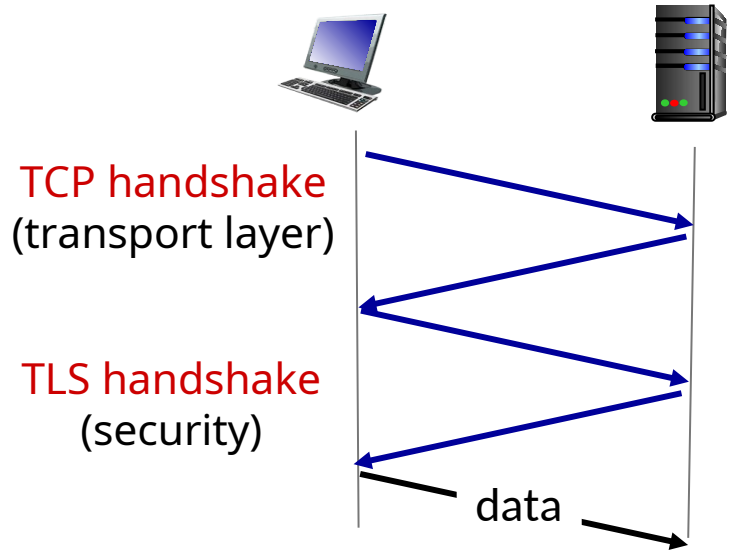
QUIC: Quick UDP Internet Connections RFC9000

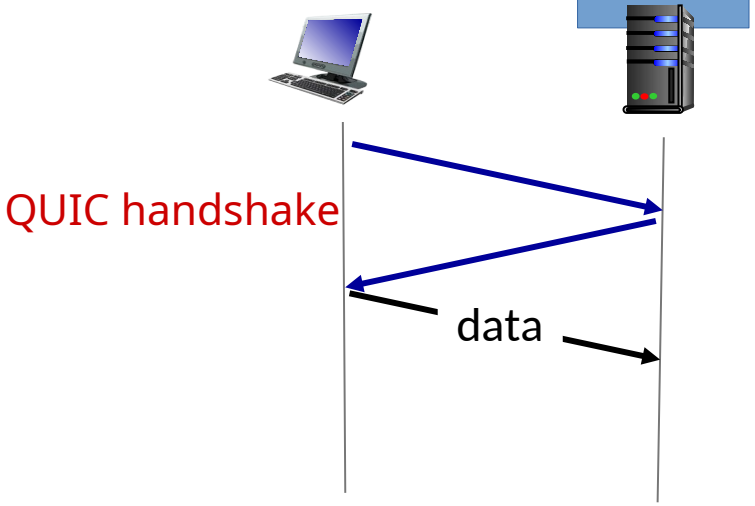
- Protocolo de camada de aplicação, construído sobre o UDP
  - Melhora a performance do HTTP, permite vários “streamings” de objetos dentro da mesma conexão QUIC.
  - Elimina head-of-line blocking (problema na transmissão de um objeto, não bloqueia as demais transmissões paralelas)
  - Já disponível em servidores web e navegadores
- Implementa:
  - Conexão (mais rápida que a do TCP).
  - Controle de erros
  - Controle de congestionamento
  - Segurança: conexão obrigatoriamente são criptografadas

QUIC: Quick UDP Internet Connections



QUIC: menos handshake





TCP (confiabilidade, controle de congestionamento) + TLS (autenticação e cifra)

- 2 serial handshakes

QUIC: autenticação, cifra, confiabilidade, controle de congestionamento

- 1 handshake

Resumo

- princípios por trás dos serviços da camada de transporte:
  - multiplexação/demultiplexação
  - transferência de dados confiável
  - controle de fluxo
  - controle de congestionamento
- Internet:
  - UDP
  - TCP