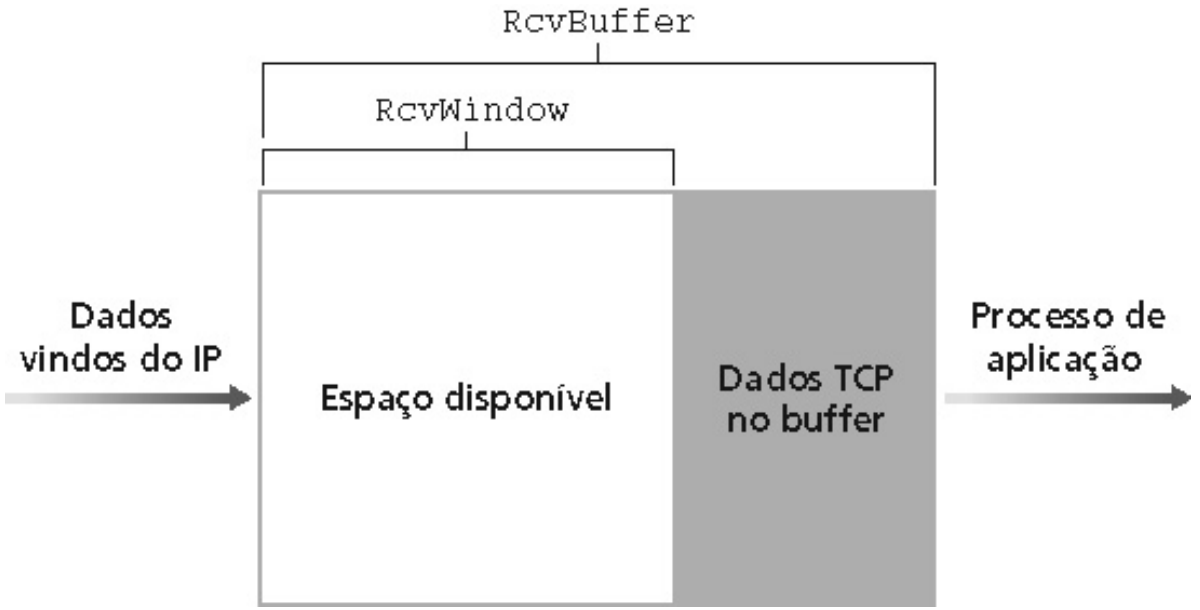
TCP - controle de fluxo

- ✓ Elimina a possibilidade do remetente saturar o destinatário
- ✓ Apresentação supõe que segmentos fora de ordem são descartados
- ✓ Baseado em janelas
  - ✓ Remetente mantém uma variável chamada janela de recepção (*RcvWindow*)

TCP - controle de fluxo

Janela e *buffer* de recepção (fonte: Kurose)



TCP - controle de fluxo

- Destinatário mantém as variáveis
  - *LastByteRead*
  - *LastByteRcvd*

- $LastByteRcvd - LastByteRead \leq RcvBuffer$
  - Para não saturar o *buffer*

- $RcvWindow = RcvBuffer - [LastByteRcvd - LastByteRead]$

TCP - controle de fluxo

- Remetente mantém as variáveis
  - *LastByteSent*
  - *LastByteAcked*
- Remetente precisa ter uma idéia do *buffer* do destinatário
- Destinatário envia quanto há de espaço no *buffer* colocando *RcvWindow* no campo janela de recepção de cada segmento
- *LastByteSent – LastByteAcked* ≤ *RcvWindow*
  - Para não saturar o *buffer* do destinatário

TCP - controle de fluxo

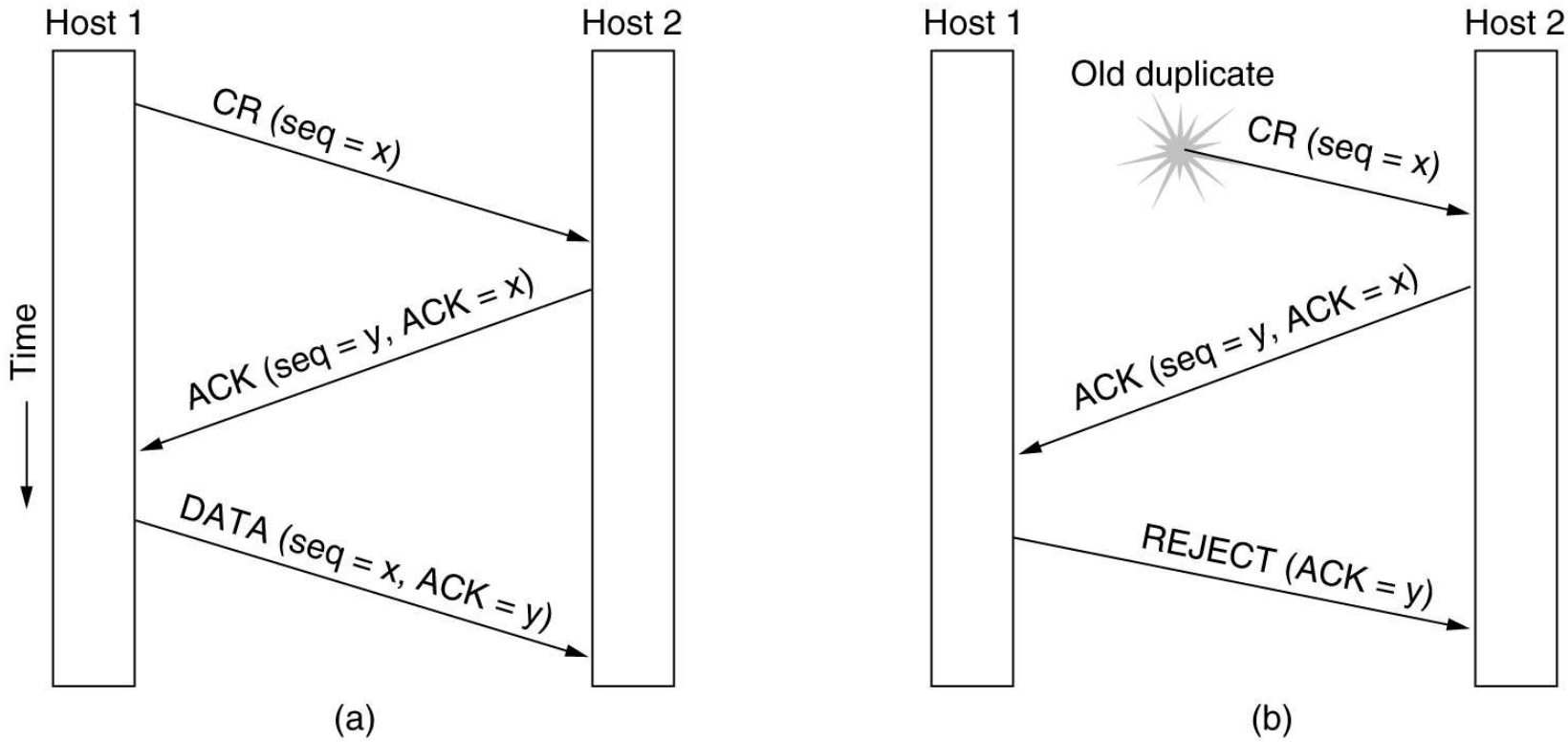
- Problema quando *RcvWindow* = 0 e o destinatário não tem nada a enviar
  - Solução
    - Remetente deve enviar segmento de um octeto de dados quando a janela for zero

TCP - estabelecimento de conexão

- ✓ Inicializa variáveis
  - ✓ Números de sequência
  - ✓ *Buffers*, *RcvWindow*
- ✓ 3-*way handshake* usado para eliminar o problema de duplicatas antigas (atrasadas)
  - ✓ Ex.: pedidos de conexão, ACKs

Estabelecimento de conexão

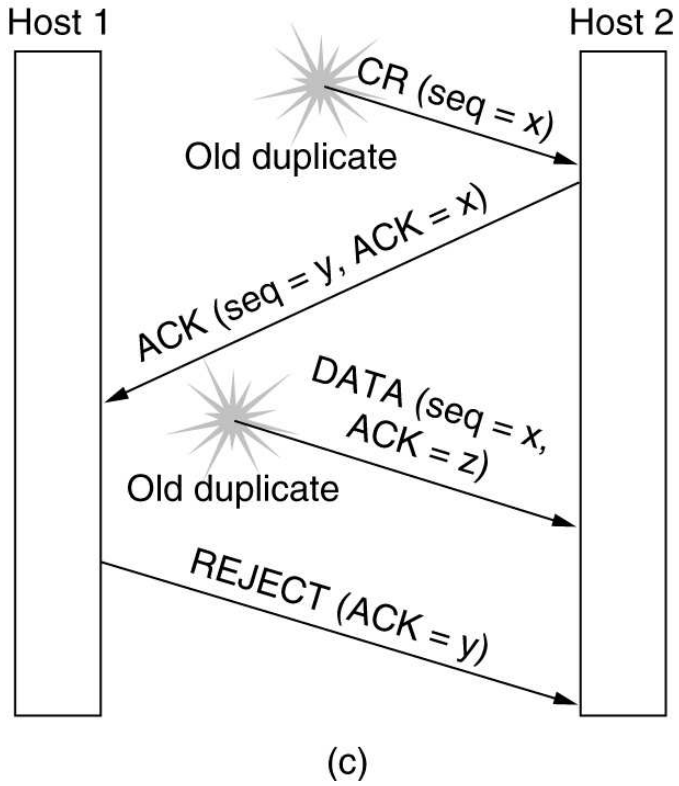
3-*way handshake* genérico (fonte: Tanenbaum)



(a) Normal operation,
(b) Old CONNECTION REQUEST appearing out of nowhere.

Estabelecimento de conexão

3-*way handshake* genérico (fonte: Tanenbaum)



(c) Duplicate CONNECTION REQUEST and duplicate ACK.

TCP - estabelecimento de conexão

- Três fases
  - SYN
    - Bit SYN = 1
    - Número de sequência inicial aleatório (*client_isn*)
  - SYNACK
    - Recebe SYN
    - Aloca *buffers* e variáveis
    - Bit SYN = 1 e bit ACK = 1 (espera *client_isn* + 1*)*
    - Número de sequência inicial aleatório (*server_isn*)
  - Última fase
    - Recebe SYNACK
    - Aloca *buffers* e variáveis
    - Bit SYN = 0
    - ACK = *server_isn + 1*

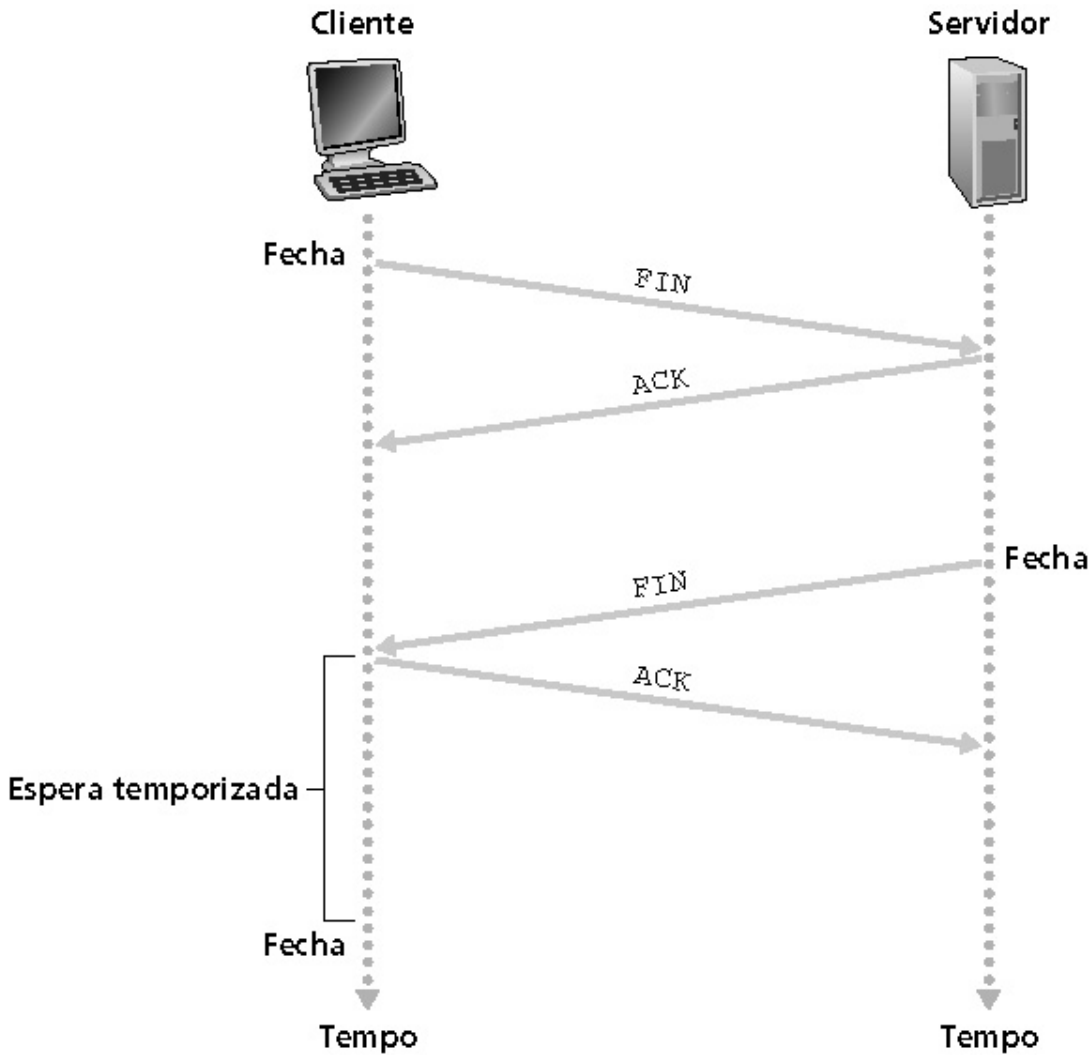
TCP - fechamento de conexão

- ✓ Cliente ou servidor pode encerrar a conexão
- ✓ Libera os *buffers* e variáveis

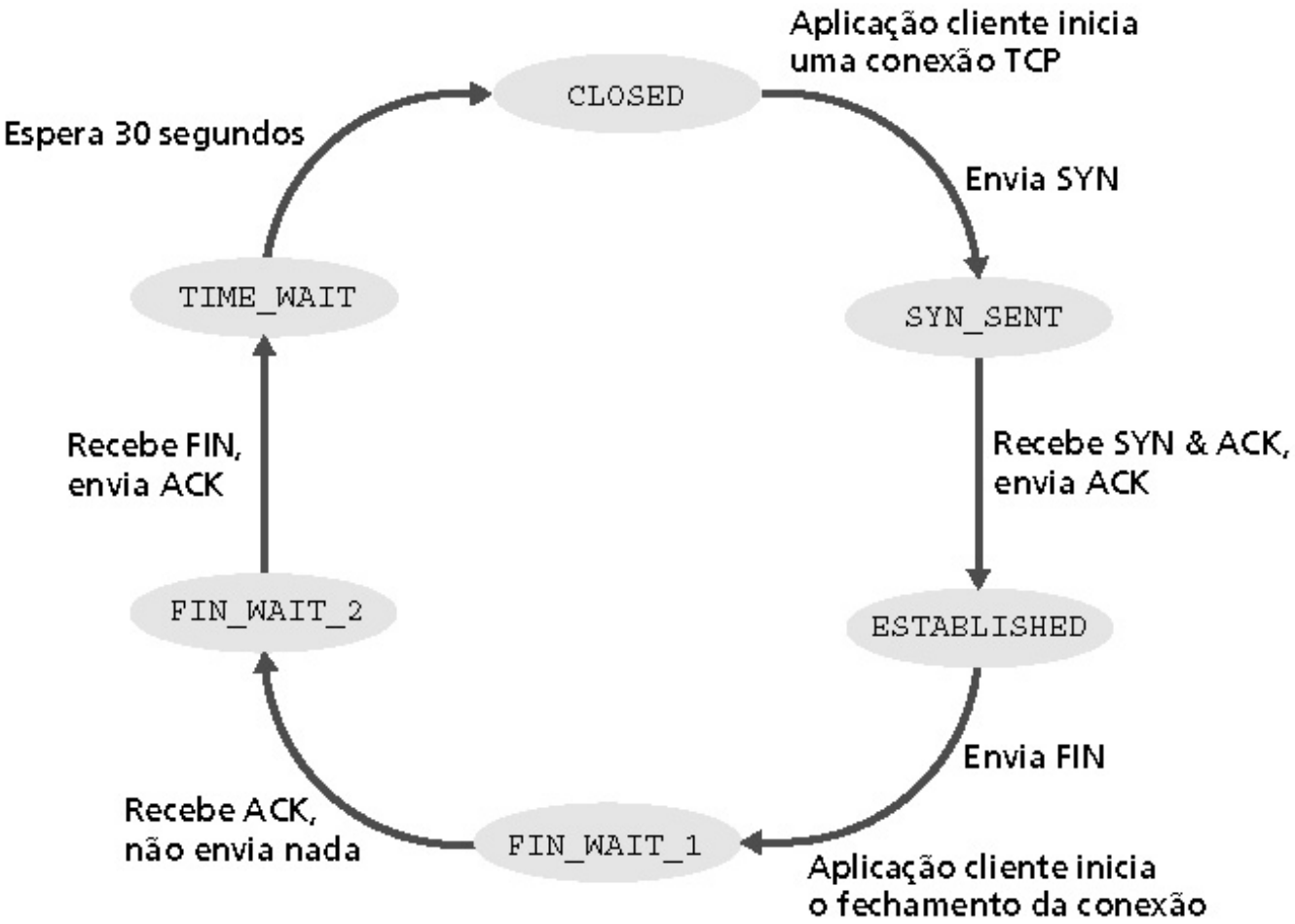
TCP - fechamento de conexão

- FIN
  - Bit FIN = 1
- ACK
  - Responde com ACK a FIN recebido
  - Bit ACK = 1
- ACK e FIN do respondedor podem ser "enviados" juntos
- Espera temporizada
  - Tempo para poder reenviar o ACK
    - ACKs do iniciador podem ser perdidos
    - Respondedor espera pelo ACK durante um certo tempo e se o ACK não chega reenvia o FIN

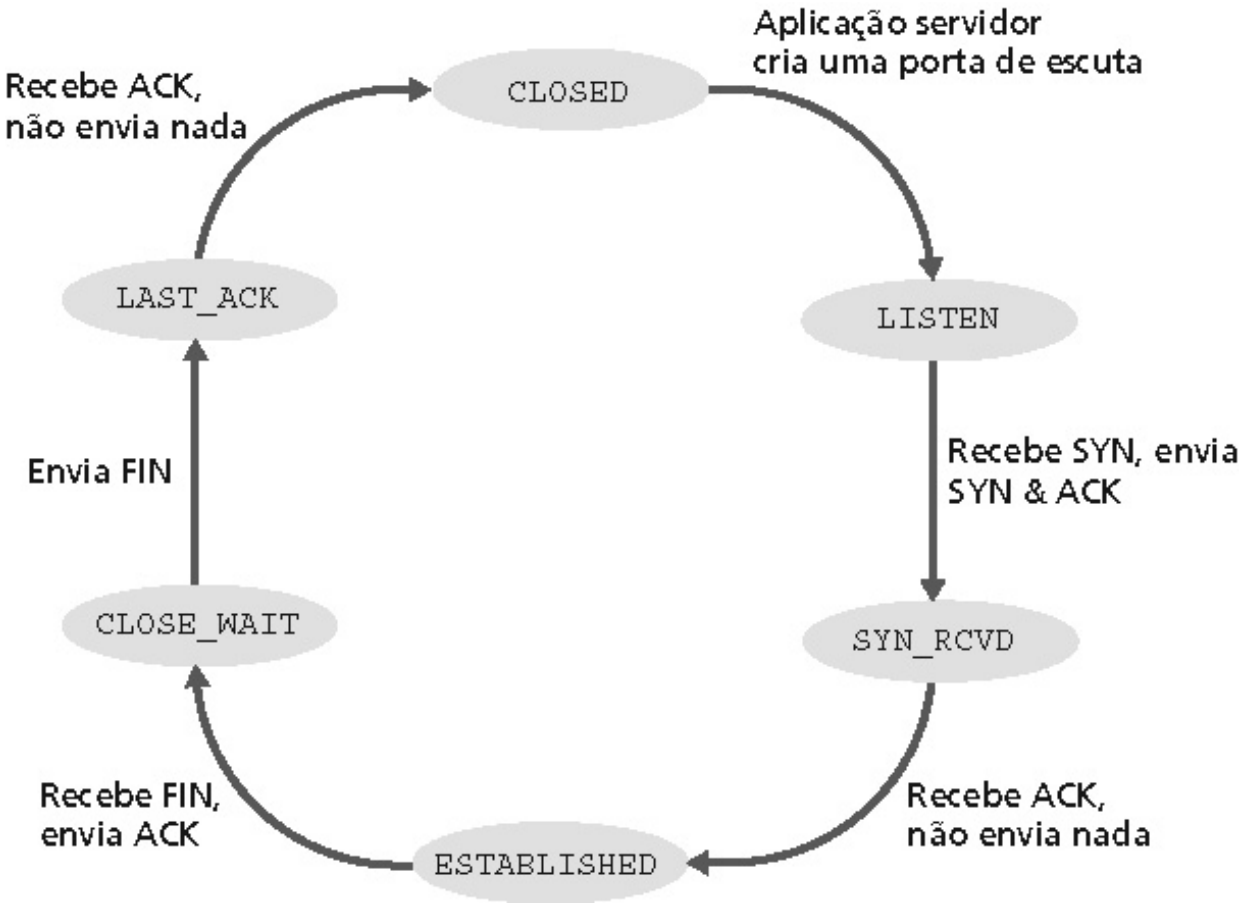
Exemplo de fechamento de conexão (fonte: Kurose)



Sequência típica de estados do TCP cliente



Sequência típica de estados do TCP servidor



Controle de congestionamento

- Sintomas
  - Perdas de pacotes
  - Atrasos grandes
- Classificação
  - Fim-a-fim
    - Congestionamento é inferido a partir das perdas e dos atrasos observados nos sistemas finais
    - Ex.: TCP
  - Assistido pela rede
    - Roteadores fornecem realimentação explícita a respeito do congestionamento na rede
    - Ex.: ATM

TCP - controle de congestionamento

- ✓ Remetente limita a transmissão
- ✓ Baseado em janelas
  - ✓ Remetente e destinatário mantêm uma variável chamada janela de congestionamento (*CongWin*)
- ✓ *LastByteSent* – *LastByteAcked* ≤ min{*CongWin*, *RcvWindow*}
  - ✓ Para não congestionar a rede e não saturar o *buffer* do destinatário
- ✓ Apresentação supõe que a limitação da janela de recepção é desprezível e que o *flavor* é o Reno

TCP - controle de congestionamento

- ✓ Permite o envio de *CongWin* segmentos de tamanho máximo *MSS* antes de receber um ACK
- ✓ Com uma janela de tamanho *CongWin* e um *round trip time RTT*, a vazão é dada por

CongWin x MSS / RTT

TCP - controle de congestionamento

- ✓ Eventos de perda
  - ✓ Estouro de temporização
  - ✓ Recebimento de três ACKs duplicados
- ✓ Três componentes
  - ✓ Partida lenta (*Slow Start*)
  - ✓ Aumento aditivo, diminuição multiplicativa (*Additive Increase, Multiplicative Decrease* - AIMD)
  - ✓ Reação a eventos de estouro de temporização

TCP - controle de congestionamento

- Algoritmo partida lenta
  - Inicialmente *CongWin* = 1 (RFC 3390)
  - Janela *CongWin* aumenta de um segmento a cada ACK recebido (dobra a cada RTT)

$$CongWin = CongWin + 1$$

  - Até um limite *Threshold* ou uma perda
    - *Threshold* $\rightarrow$ AIMD
    - 3 ACKs duplicados $\rightarrow$ AIMD
    - Estouro de temporizador $\rightarrow$ início da partida lenta

TCP - controle de congestionamento

- ✓ Algoritmo AIMD
  - ✓ Janela *CongWin* cresce 1/*CongWin* a cada ACK recebido

$$CongWin = CongWin + a/CongWin,\ a = 1$$

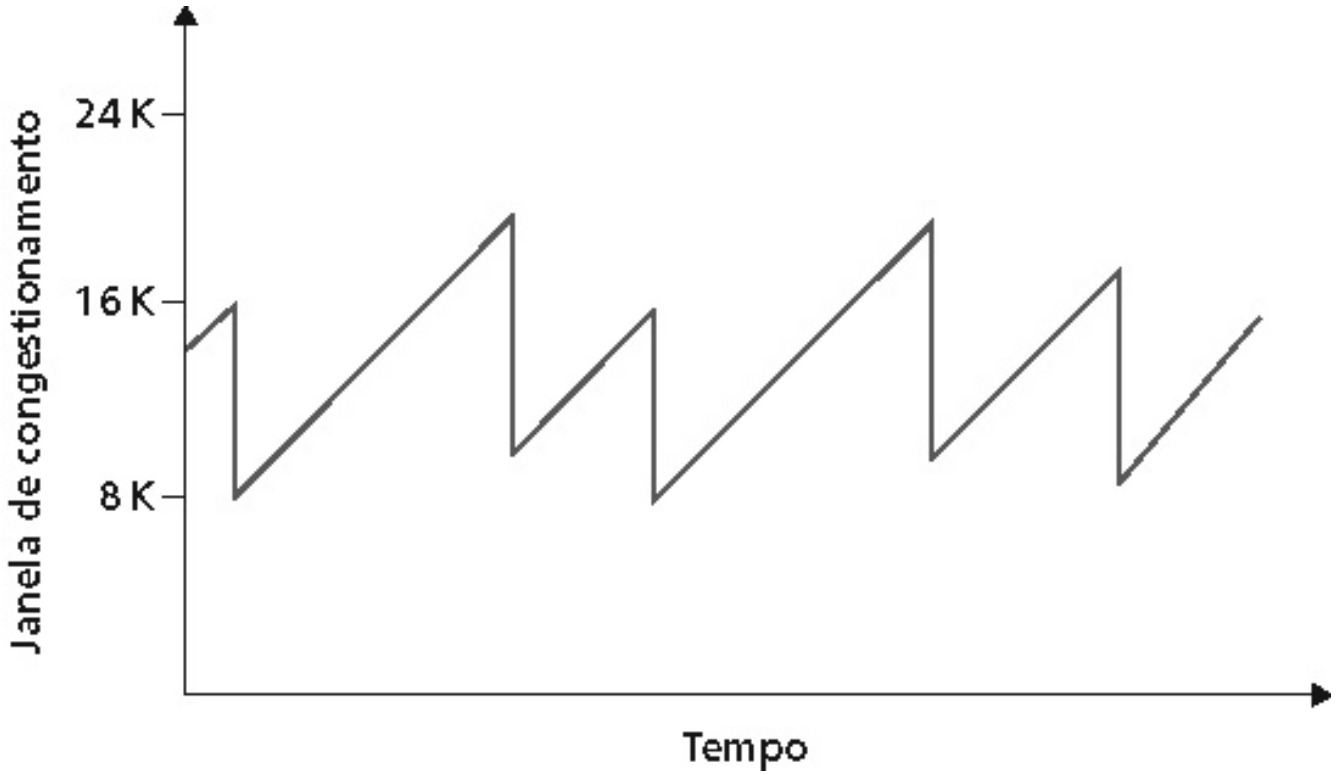
  - ✓ Perda
    - ✓ *Threshold* = 0,5 * *CongWin*
    - ✓ Janela *CongWin* diminui de 0,5 ∗ *CongWin* a cada segmento perdido por ACKs duplicados

$$CongWin = CongWin - b * CongWin,\ b = 0,5$$

    - ✓ *CongWin* = 1 se perda por temporização

TCP - controle de congestionamento

Exemplo de controle de congestionamento com AIMD (fonte: Kurose)



TCP - controle de congestionamento

- Algoritmo AIMD
  - Para *MSS* = 1500, *RTT* = 100 ms e vazão = 10 Gbps, *CongWin* deve ser 83333
  - Isso gera uma taxa de perdas que não pode passar de $2 * 10^{-10} \rightarrow$ não é possível atualmente
  - Solução: uso de outros protocolos de transporte, como o HSTCP

Exemplo de controle de congestionamento do TCP (fonte: Kurose)

Evento	Estado	Ação do transmissor TCP	Comentário
ACK recebido para dado previamente não-confirmado	partida lenta (SS)	CongWin = CongWin + MSS, If (CongWin > Threshold) ajusta estado para "prevenção de congestionamento"	Resulta em dobrar o CongWin a cada RTT
ACK recebido para dado previamente não-confirmado	prevenção de congestiona-mento (CA)	CongWin = CongWin + MSS * (MSS/CongWin)	Aumento aditivo, resulta no aumento do CongWin em 1 MSS a cada RTT
Evento de perda detectado por três ACKs duplicados	SS or CA	Threshold = CongWin/2, CongWin = Threshold, Ajusta estado para "prevenção de congestionamento"	Recuperação rápida, implementando redução multiplicativa o CongWin não cairá abaixo de 1 MSS
Estouro de temporizador	SS or CA	Threshold = CongWin/2, CongWin = 1 MSS, Ajustar estado para "partida lenta"	Entra em partida lenta
ACK duplicado	SS or CA	Incrementa o contador de ACK duplicado para o segmento que está sendo confirmado	CongWin e Threshold não mudam

TCP - *flavors*

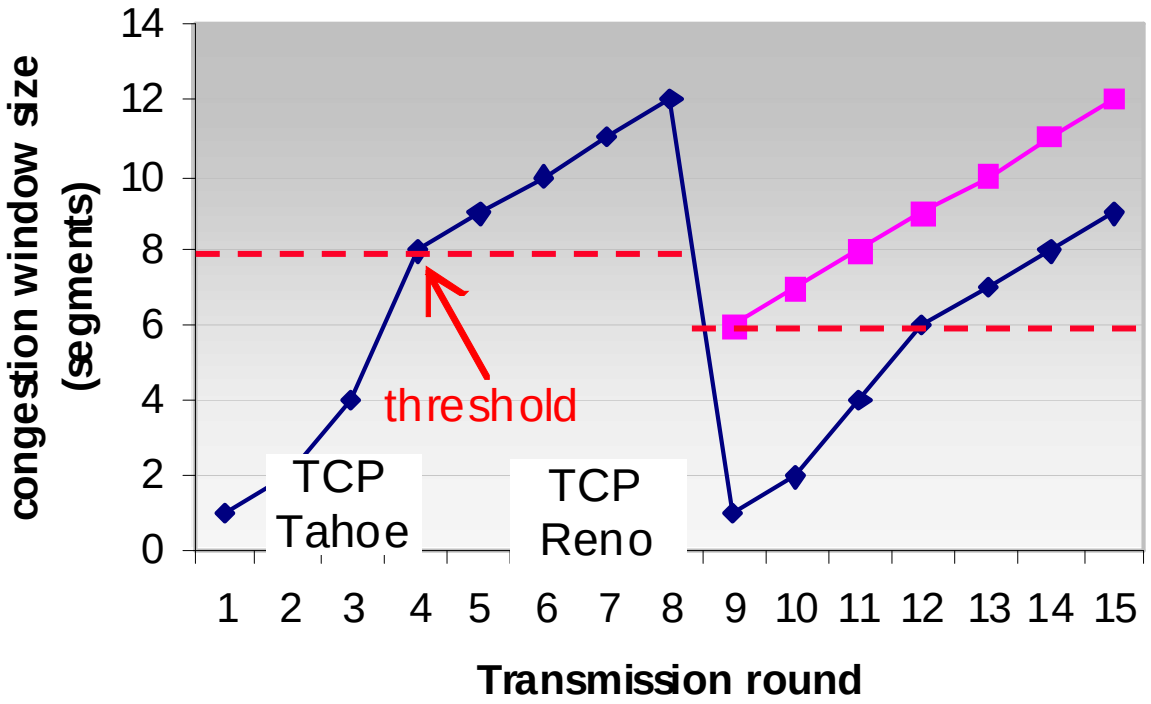
- ✓ Tahoe
  - ✓ Mais antigo
  - ✓ Usa partida lenta e AIMD
  - ✓ Qualquer perda volta à partida lenta
- ✓ Reno
  - ✓ Mais novo
  - ✓ Usa partida lenta e AIMD
  - ✓ Usa retransmissão rápida, recuperação rápida e ACKs atrasados
  - ✓ Apresentado anteriormente
- ✓ Vegas
  - ✓ Usa um outro algoritmo de congestionamento

TCP - *flavors*

- ✓ Sack
  - ✓ Usa retransmissão seletiva
  - ✓ Usa o campo de opções para indicar blocos não contíguos recebidos
- ✓ New Reno

TCP - controle de congestionamento

Exemplo de controle de congestionamento do TCP (fonte: Kurose)

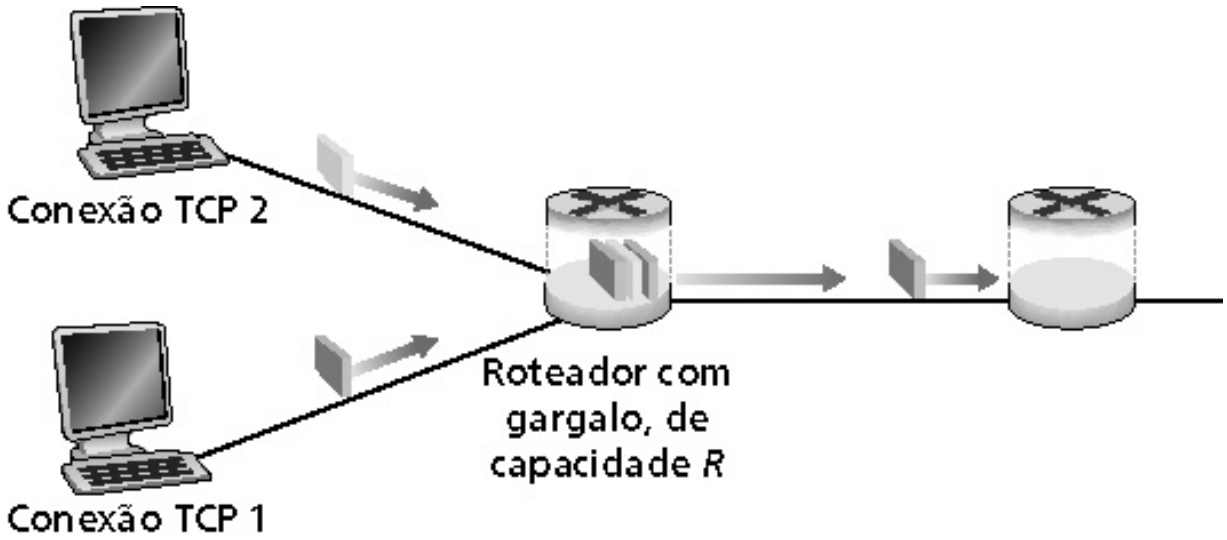


TCP - justiça

- ✓ AIMD é justo
- ✓ Na apresentação supõe-se
  - ✓ Duas conexões TCP
  - ✓ Mesmos MSSs e RTTs
  - ✓ Enlace R

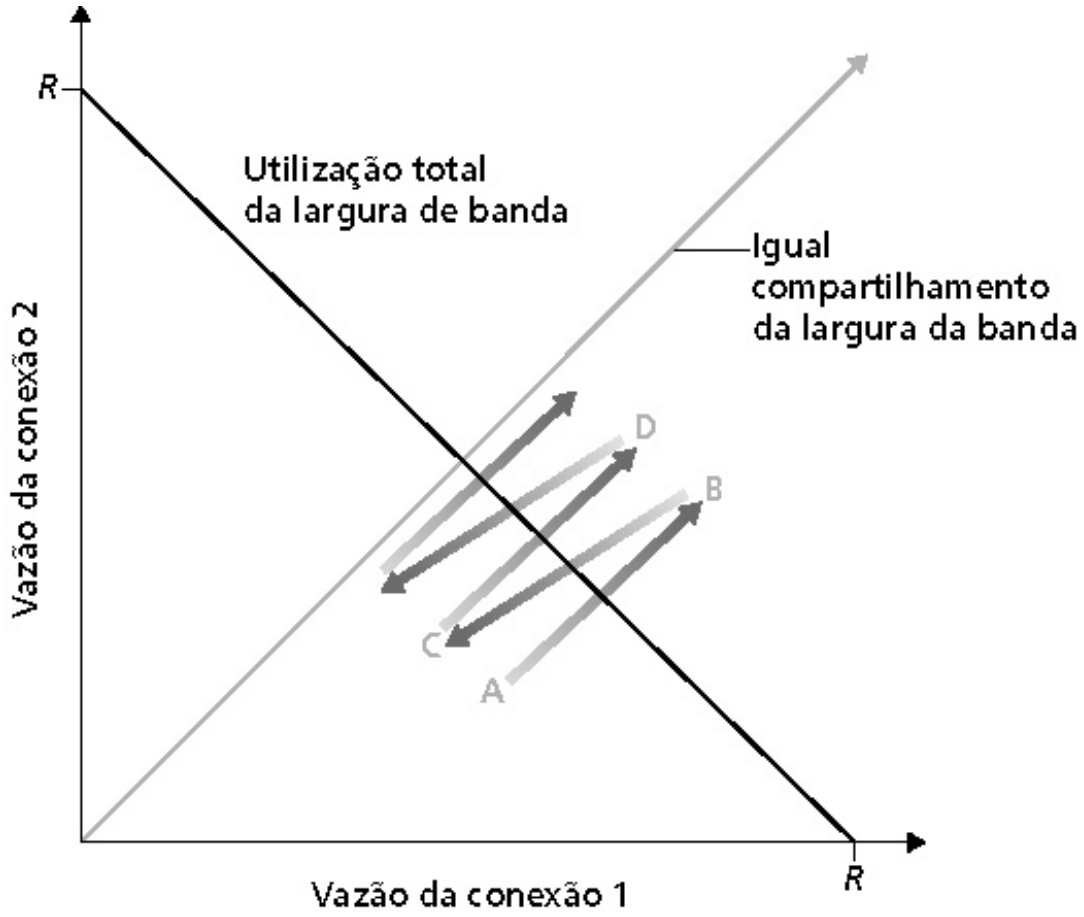
TCP - justiça

Compartilhamento de um enlace congestionado (fonte: Kurose)



TCP - justiça

Vazão das conexões (fonte: Kurose)



Bibliografia

- ✓ Kurose – Capítulos 3 e 7
- ✓ Tanenbaum – Capítulo 6