Aula 13 - TCP: Controle de Congestionamento

Diego Passos

Universidade Federal Fluminense

Redes de Computadores I

Material adaptado a partir dos slides originais de J.F Kurose and K.W. Ross.

Revisão da Última Aula...

- **TCP: Timeout**.
  - Estimado **dinamicamente**.
  - Amostras a cada ACK recebido.
  - **Média Movente Exponencialmente Ponderada**.
  - Estima-se também o desvio das amostras.
  - Timeout: RTT estimado mais 4X o desvio.
- **TCP: Confiabilidade**.
  - Baseado em pipeline.
  - ACKs cumulativos.
  - Temporizador único.
- **TCP infere perdas por:**
  - Estouro do temporizador.
  - ACKs duplicados.
    - **Fast Retransmit**.
- **TCP: Retransmissões.**
  - Apenas do segmento "perdido".
  - Mais antigo ainda pendente.
- **TCP: ACKs.**
  - Podem ser atrasados.
  - Sempre indicam próximo # de sequência esperado.
- **Controle de fluxo:**
  - **Transmissor não sobrecarregará receptor**.
  - Receptor anuncia quantidade de *buffer* disponível.
  - Janela do transmissor não excede este valor.
- **Gerenciamento de conexão:**
  - ***3-way handshake***: abertura de conexão.
    - Parâmetros, *e.g.*, # de seq.
  - Fechamento de conexão: bit FIN.

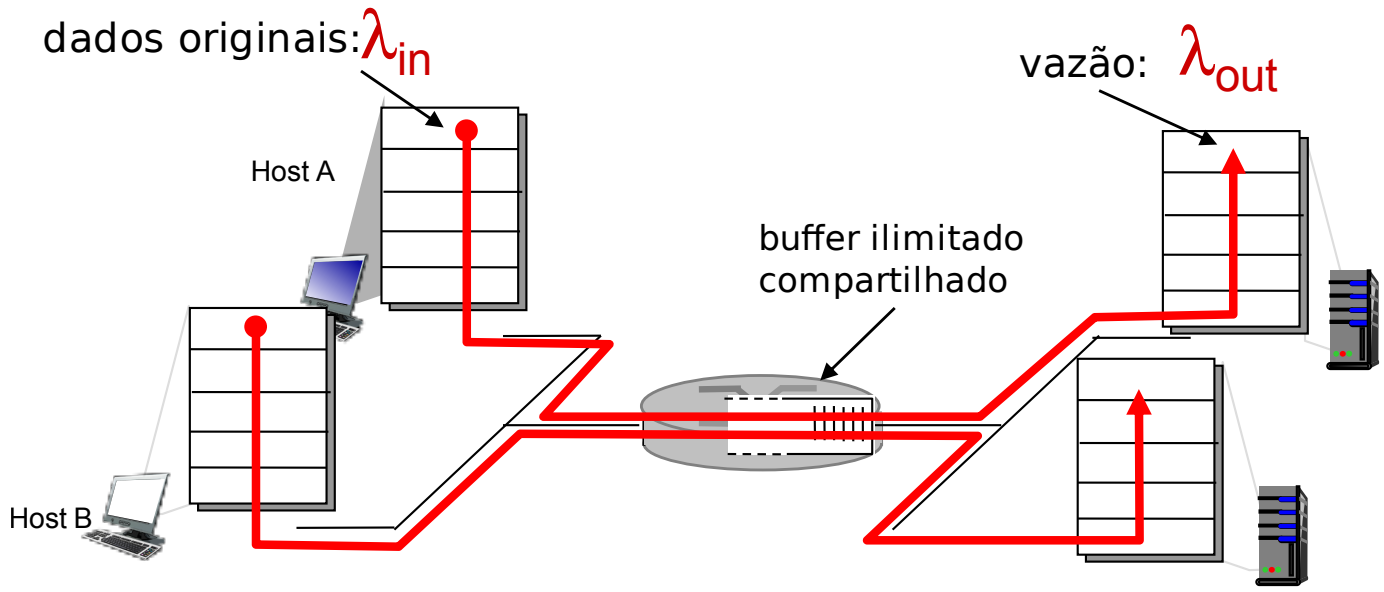
Controle de Congestionamento em Redes

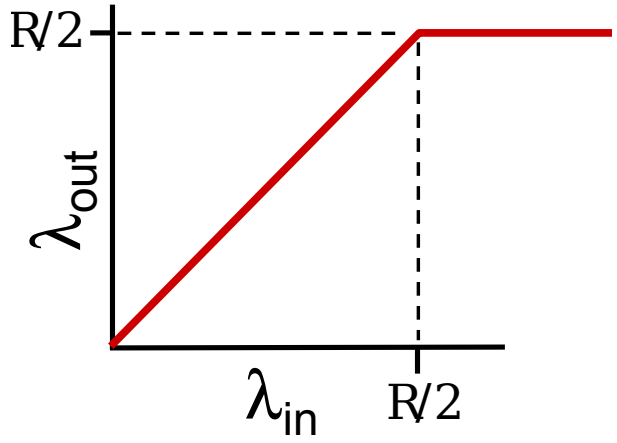
Princípios de Controle de Congestionamento

- **Congestionamento:**
  - Informalmente: “fontes demais gerando tráfego demais para a **rede**”.
  - **Diferente do controle de fluxo**.
  - Manifestações:
    - Pacotes perdidos (*overflow* de *buffers* nos roteadores).
    - Longos atrasos (enfileiramento nos *buffers* dos roteadores).
  - Um dos 10 problemas mais importantes em redes!

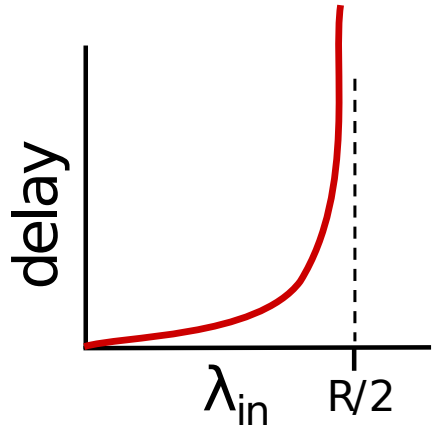
Causas/Custos do Congestionamento: Cenário 1

- Dois transmissores, dois receptores.
- Um roteador, *buffers* **infinitos**.
- Capacidade do enlace de saída: R.
- Sem retransmissões.





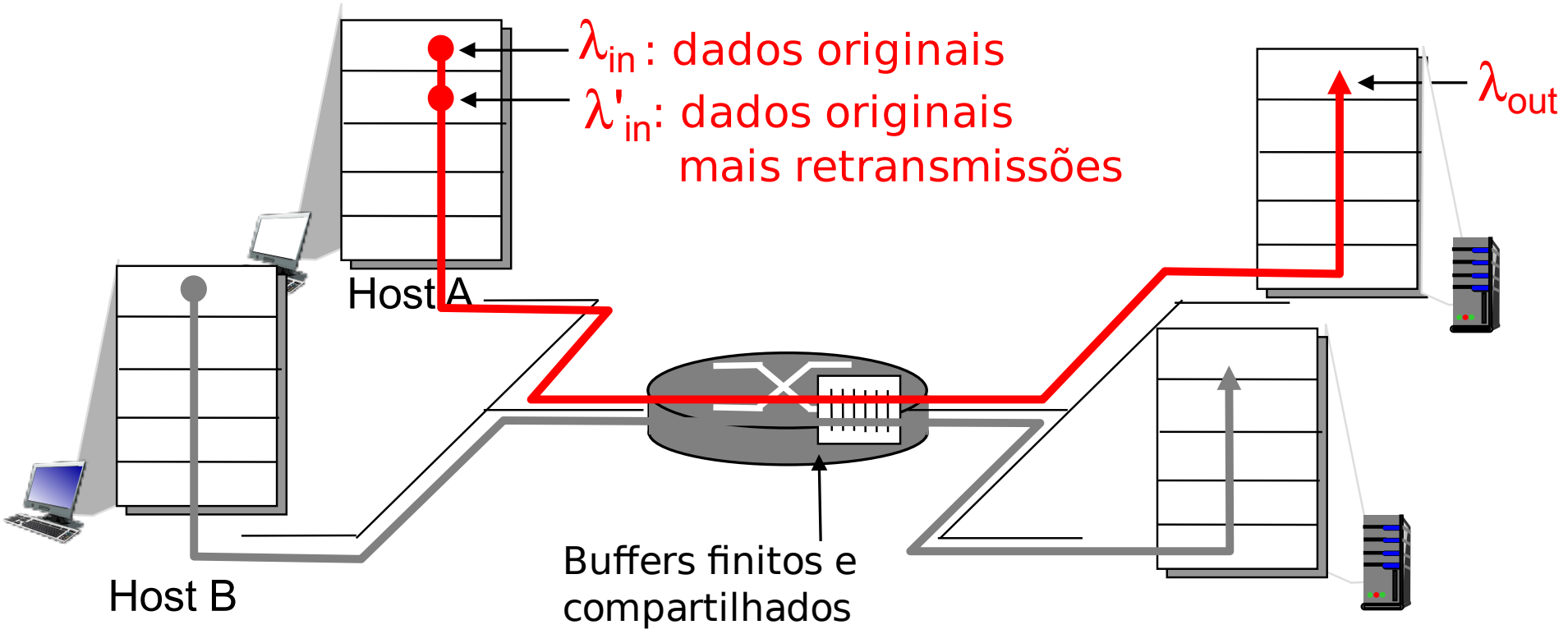
Vazão máxima por conexão: R/2.



Atrasos altos à medida que taxa de chegada λ_{in}, se aproxima da capacidade.

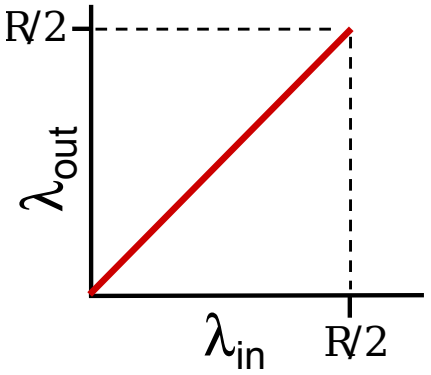
Causas/Custos do Congestionamento: Cenário 2 (I)

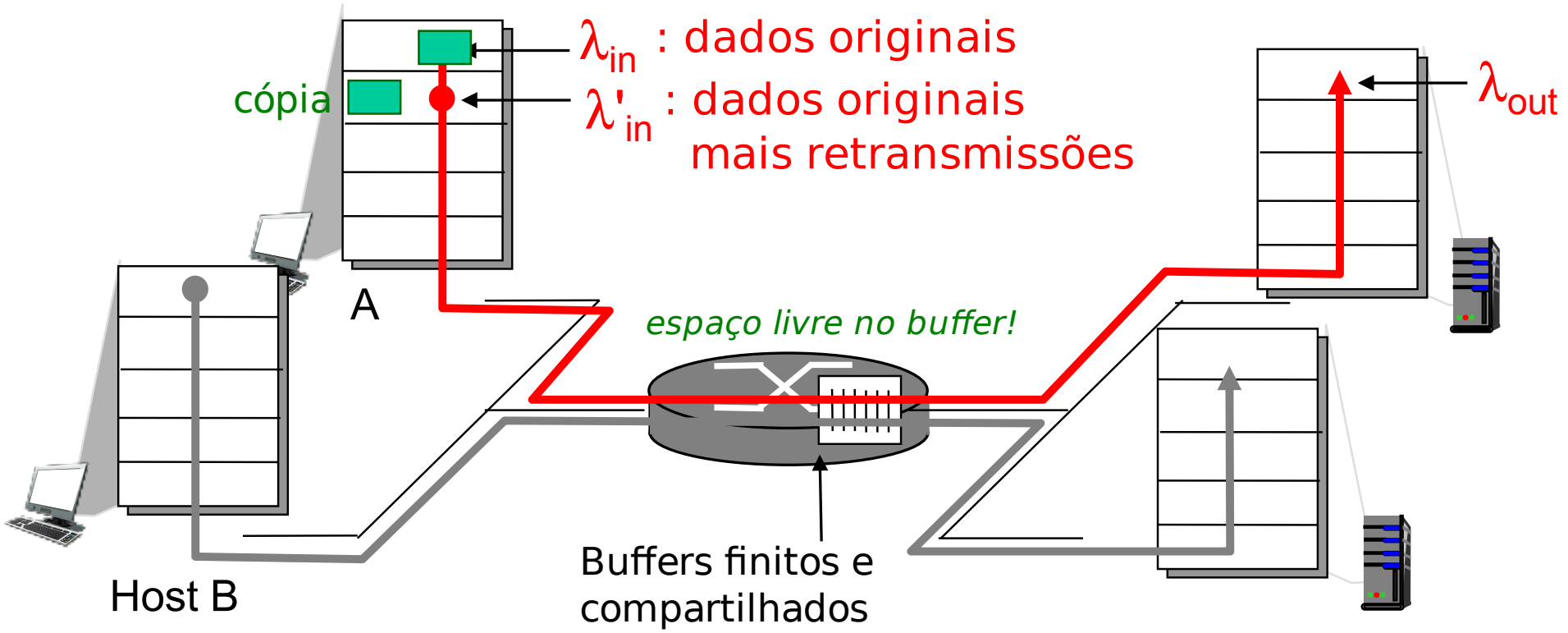
- Um roteador, *buffer* **finito**.
- Transmissor retransmite pacotes após *timeout*.
  - Entrada da camada de aplicação do transmissor = saída da camada de aplicação do receptor: $\lambda_{in} = \lambda_{out}$.
  - Mas a camada de transporte inclui retransmissões: $\lambda'_{in} \geq \lambda_{in}$.



Causas/Custos do Congestionamento: Cenário 2 (II)

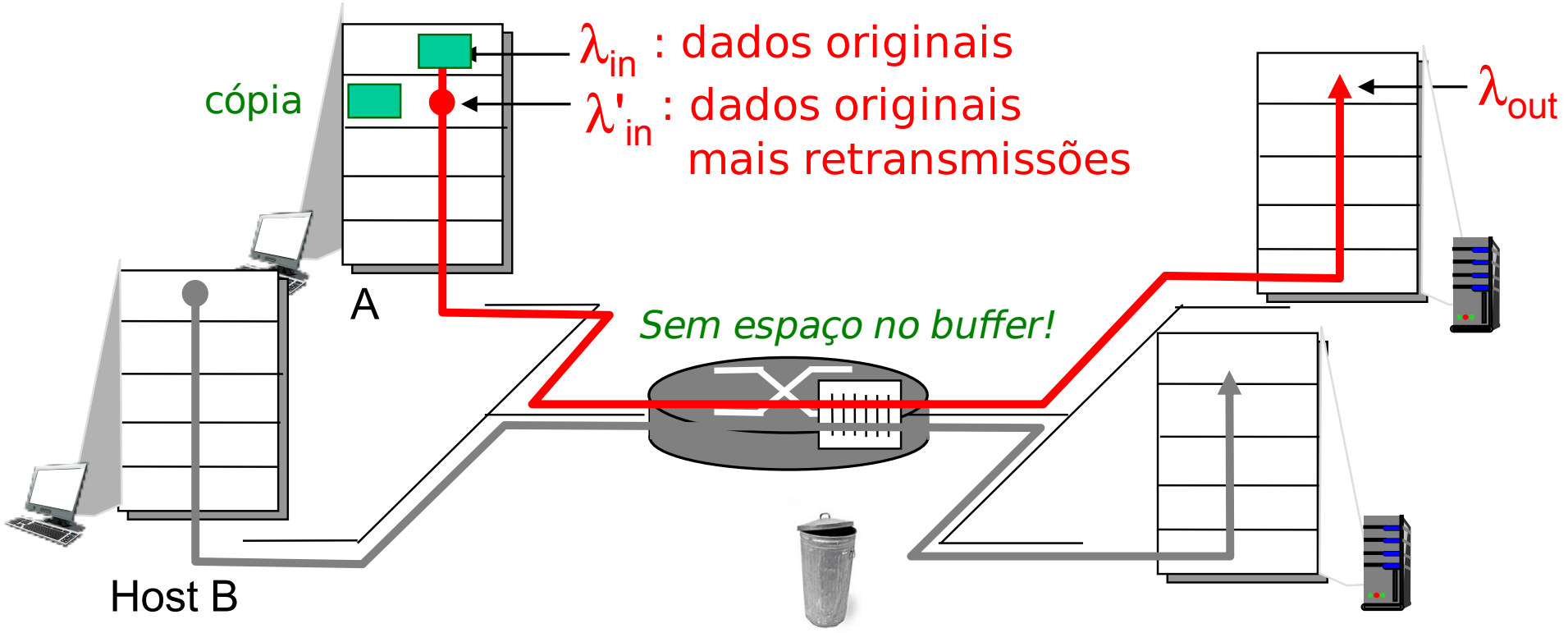
- **Idealização: conhecimento perfeito**.
  - Transmissor só transmite quando **sabe que há espaço disponível no *buffer* do roteador**.





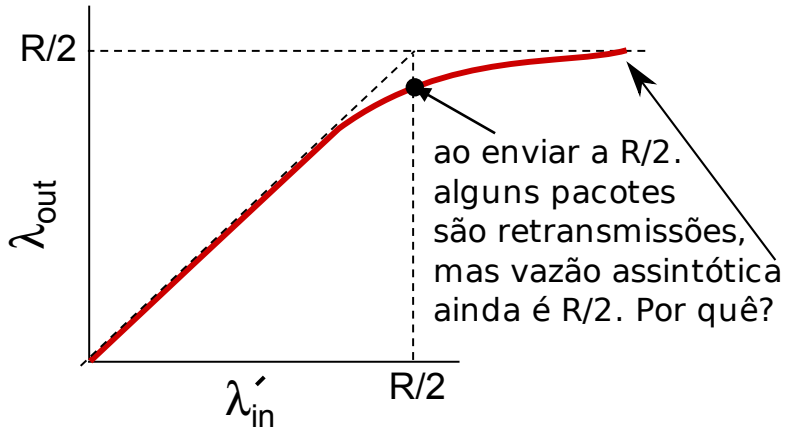
Causas/Custos do Congestionamento: Cenário 2 (III)

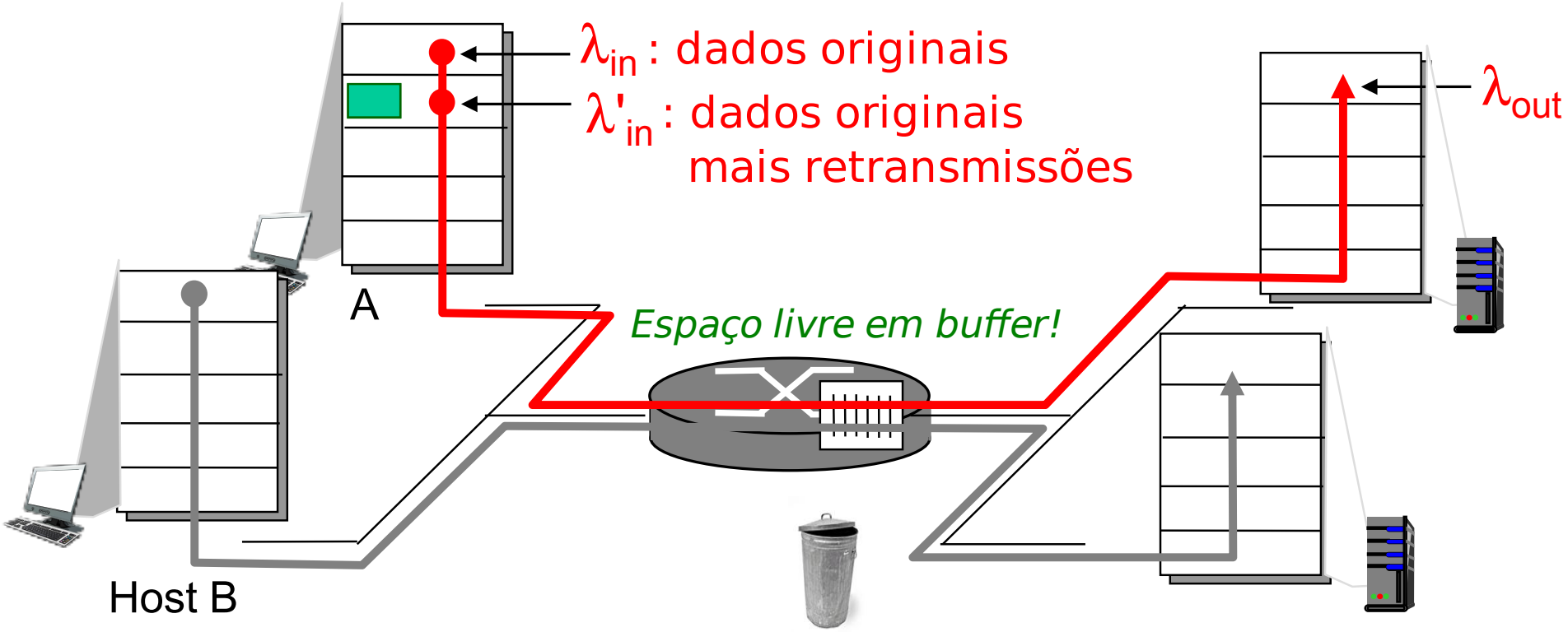
- **Idealização: perdas conhecidas**.
  - Pacotes podem ser perdidos, descartados devido a *buffers* cheios.
  - Transmissor só retransmite quando **sabe que pacote foi perdido**.



Causas/Custos do Congestionamento: Cenário 2 (IV)

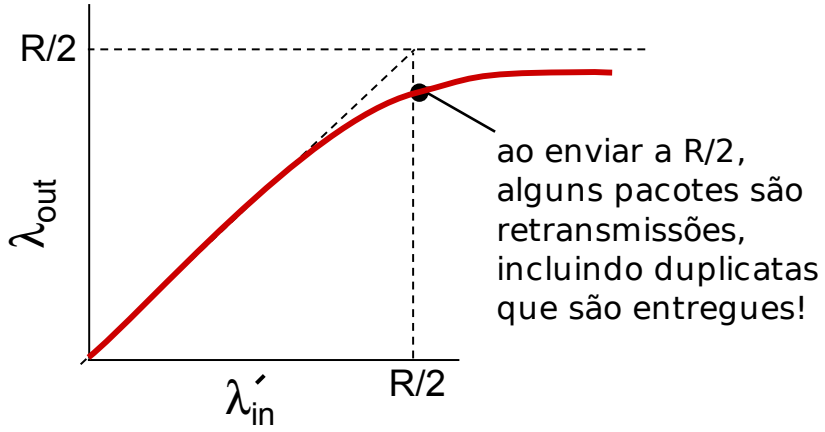
- **Idealização: perdas conhecidas**.
  - Pacotes podem ser perdidos, descartados devido a *buffers* cheios.
  - Transmissor só retransmite quando **sabe que pacote foi perdido**.

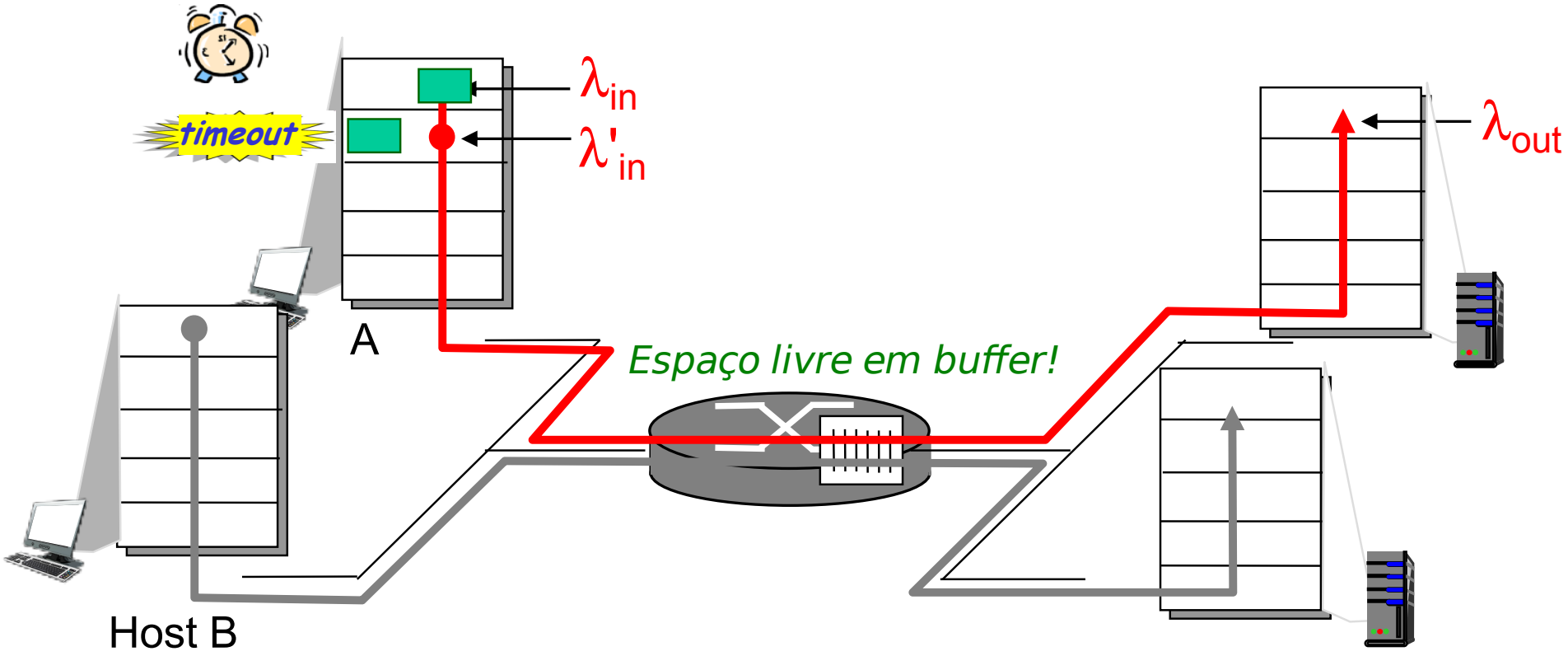




Causas/Custos do Congestionamento: Cenário 2 (V)

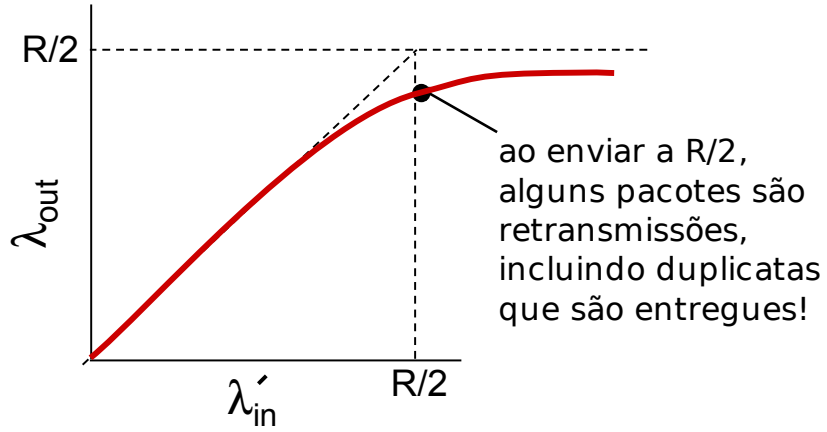
- **Realistico: duplicatas**.
  - Pacotes podem ser perdidos, descartados devido a *buffers* cheios.
  - Temporizador pode expirar prematuramente, enviando **várias** cópias que são entregues.





Causas/Custos do Congestionamento: Cenário 2 (VI)

- **Realistico: duplicatas**.
  - Pacotes podem ser perdidos, descartados devido a *buffers* cheios.
  - Temporizador pode expirar prematuramente, enviando **várias** cópias que são entregues.

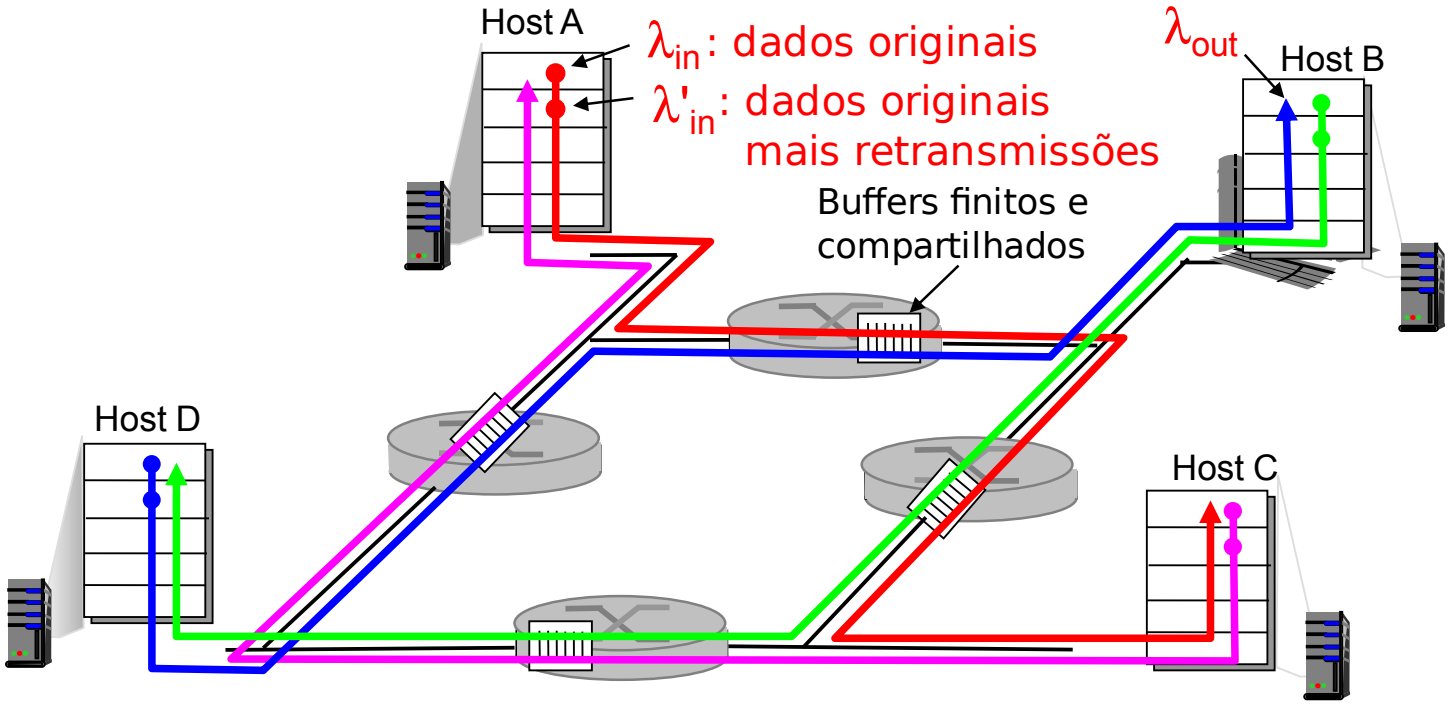


- **"Custos" do congestionamento:**
  - Mais trabalho (retransmissões) para um dado "goodput".
  - Retransmissões desnecessárias: enlace carrega múltiplas cópias do mesmo pacote.
    - Reduz goodput.

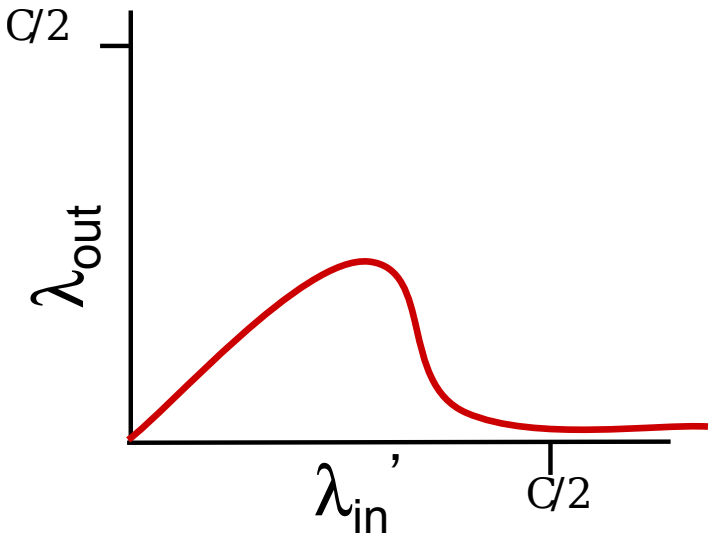
Causas/Custos do Congestionamento: Cenário 3

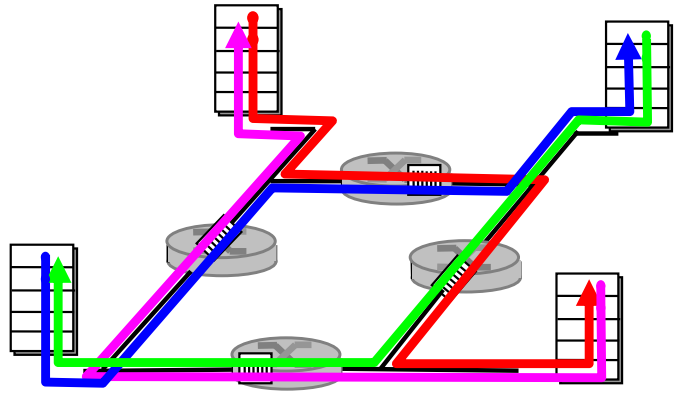
- Quatro transmissores.
- Caminhos de múltiplos saltos.
- Temporizadores/retransmissões.

- **Pergunta:** o que acontece quando $\lambda_{in}, \lambda'_{in}$ aumentam?
- **Resposta:** todos os pacotes azuis são descartados, vazão do fluxo azul cai a zero.



Causas/Custos do Congestionamento: Cenário 3 (II)





- **Outro "custo" do congestionamento:**
  - Quando pacote é descartado, qualquer capacidade de transmissão já utilizada é desperdiçada!

Abordagens para Controle de Congestionamento

- Duas linhas gerais:

Fim-a-fim

- Sem *feedback* explícito da rede.
- Congestionamento **inferido** a partir de atrasos, perdas observados pelos sistemas finais.
- Abordagem usada pelo TCP.

Asistido pela rede

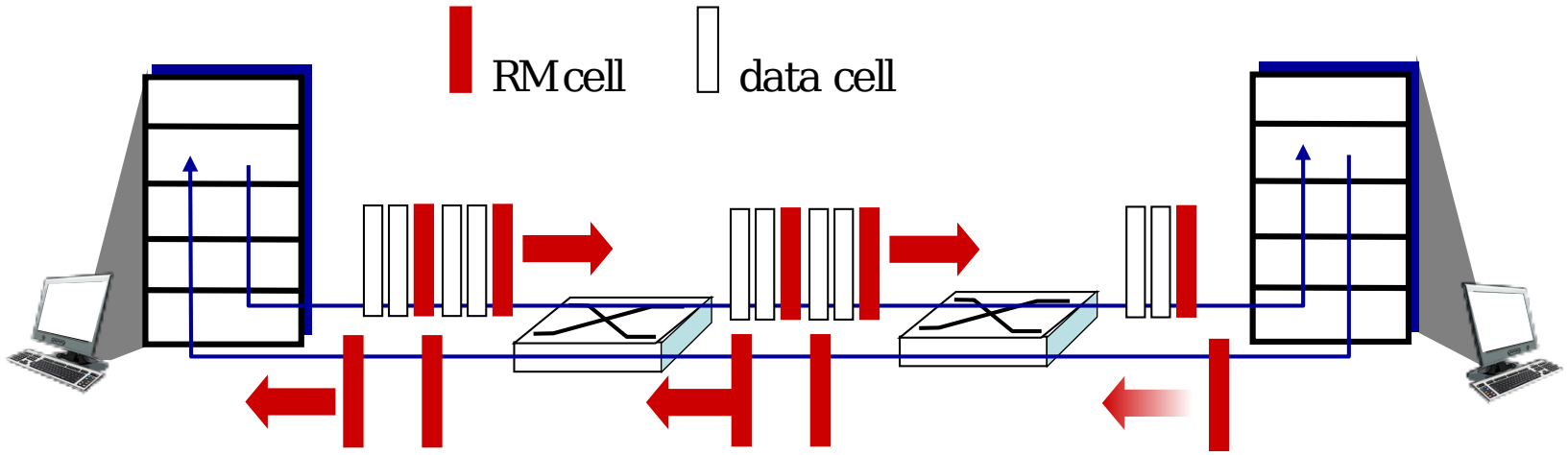
- Roteadores proveem *feedback*.
  - Um único bit indicando congestionamento (SNA, DECbit, ECN do TCP/IP, ATM).
  - Informação explícita da taxa a ser utilizada.

Estudo de Caso: Controle de Congestionamento do ABR no ATM (I)

- **ABR: *Available Bit Rate*:**
  - "Serviço elástico".
  - Se o caminho está "desocupado":
    - Transmissor deve usar capacidade disponível.
  - Se caminho está congestionado:
    - Transmissor reduz taxa para o mínimo garantido.

- **Células RM (*Resource Management*)**:
  - Enviadas pelo transmissor entrelaçadas com as de dados.
  - Dois bits na célula RM são marcados pelos comutadores ("assistido pela rede"):
    - **Bit NI:** não aumente a taxa (congestionamento moderado).
    - **Bit CI:** indicador de congestionamento.
  - Células RM são devolvidas ao transmissor pelo receptor, mantendo-se os bits intactos.

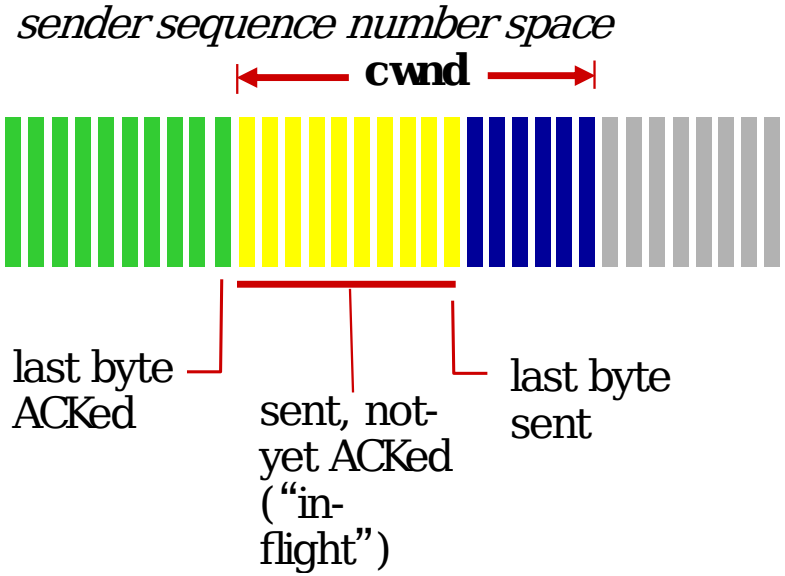
Estudo de Caso: Controle de Congestionamento do ABR no ATM (II)



- Campo ER (*Explicit Rate*) de dois bytes na célula RM.
  - Comutador congestionado pode reduzir valor do ER.
  - Taxa de transmissão do transmissor será taxa suportada no gargalo.
- Bit EFCI nas células de dados: colocado em 1 por comutadores congestionados.
  - Se célula de dados precedendo RM tem bit EFCI igual a 1, receptor marca o bit CI na célula RM enviada de volta.

Controle de Congestionamento do TCP

Controle de Congestionamento do TCP



- Taxa de transmissão do TCP:
  - Aproximadamente, envia cwnd bytes e espera RTT pelos ACKs, e então envia mais dados.

$$taxa = \frac{cwnd}{RTT} \; B/s$$

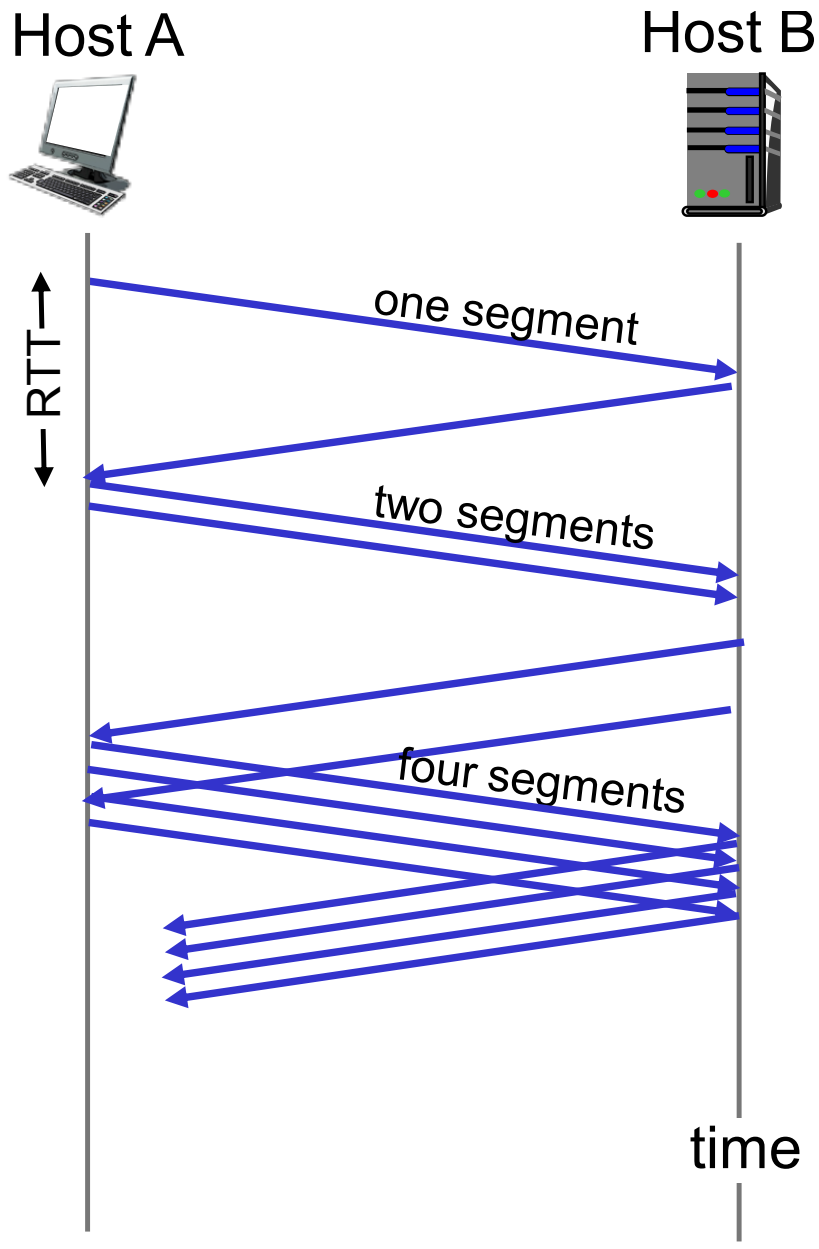
- Transmissor limita taxa de transmissão:

$$UltimoByteEnviado - UltimoByteConfirmado \leq cwnd$$

- **cwnd** é dinâmica, função do congestionamento percebido da rede.

Slow Start

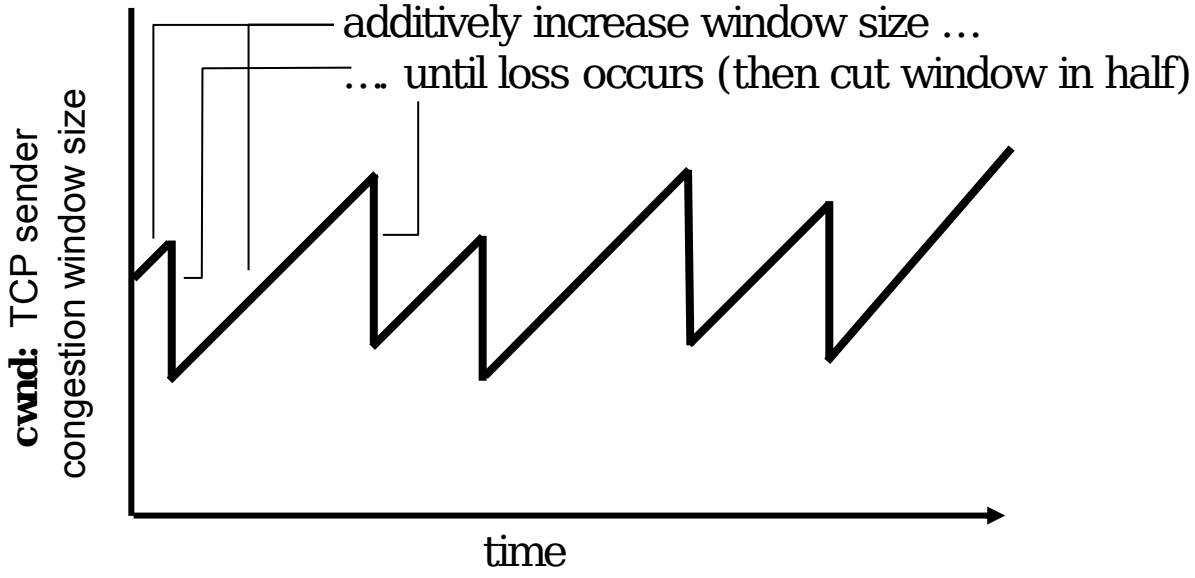
- Quando a conexão começa, aumente a taxa exponencialmente até que **cwnd ≥ ssthresh**.
  - Inicialmente, **cwnd = 1 MSS**.
  - Dobra a cada RTT.
  - Equivalente a aumentar em 1 MSS a cada ACK recebido.
- **Resumo:** taxa inicial é baixa, **mas aumenta rapidamente**.



Controle de Congestionamento do TCP: *Congestion Avoidance*

- Ocorre quando **cwnd ≥ sstresh**.
- **Abordagem:** transmissor aumenta taxa de transmissão (tamanho da janela), prospectando capacidade utilizável até que perda ocorra.
  - **Incremento aditivo:** aumenta a **cwnd** em 1 MSS a cada RTT até que perda seja detectada.
  - **Decremento multiplicativo:** corta **cwnd** pela metade após evento de perda.
- Fase **mais conservadora que o *slow start***.

AIMD saw tooth behavior: probing for bandwidth

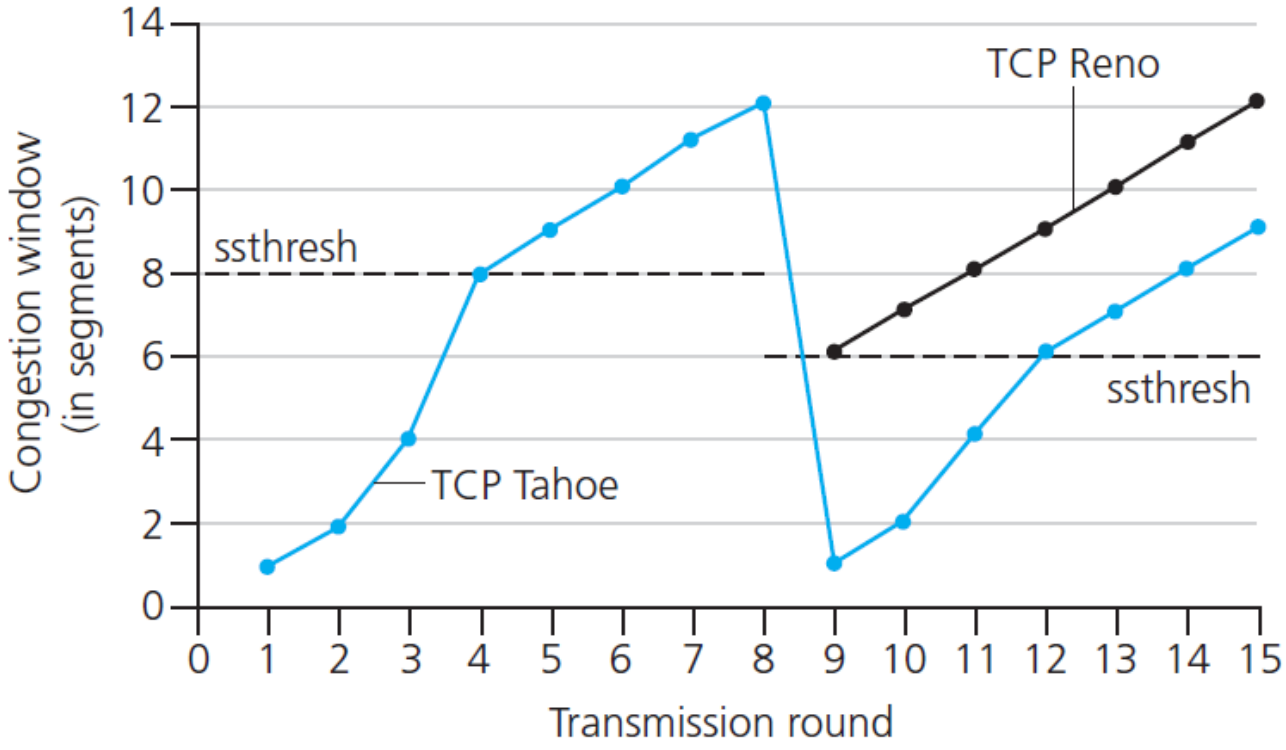


TCP: Detectando e Reagindo a Perdas

- Perda indicada por estouro de temporizador:
  - **cwnd** volta para 1 MSS.
  - Janela cresce exponencialmente (*slow start*) até limiar, depois cresce linearmente (AIMD).
- Perda indicada por 3 ACKs duplicados (**TCP Reno**):
  - ACKs duplicados indicam que a rede ainda é capaz de entregar alguns segmentos.
  - **cwnd** é cortada pela metade e depois cresce linearmente.
  - Mecanismo de ***Fast Recovery***.
- **TCP Tahoe** sempre volta janela a 1 MSS (tanto em estouro de temporizador, quanto para 3 ACKs duplicados).
  - *i.e.*, não existe *fast recovery*.

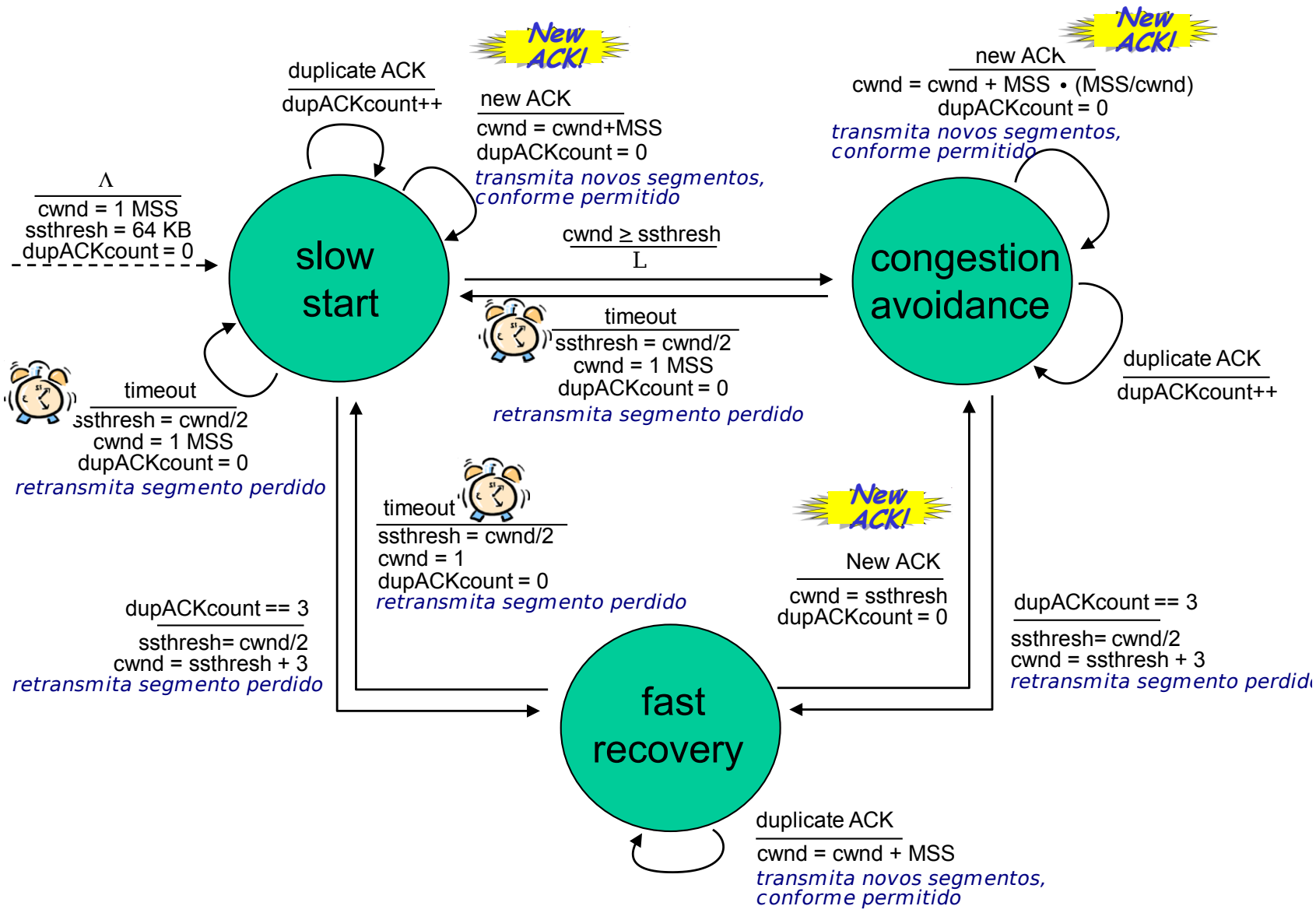
TCP: Alternando entre *Slow Start* e *Congestion Avoidance*

- **Pergunta:** quando o crescimento exponencial deve alternar para linear?
- **Resposta:** quando a **cwnd** chega à metade do seu valor antes do *timeout*.



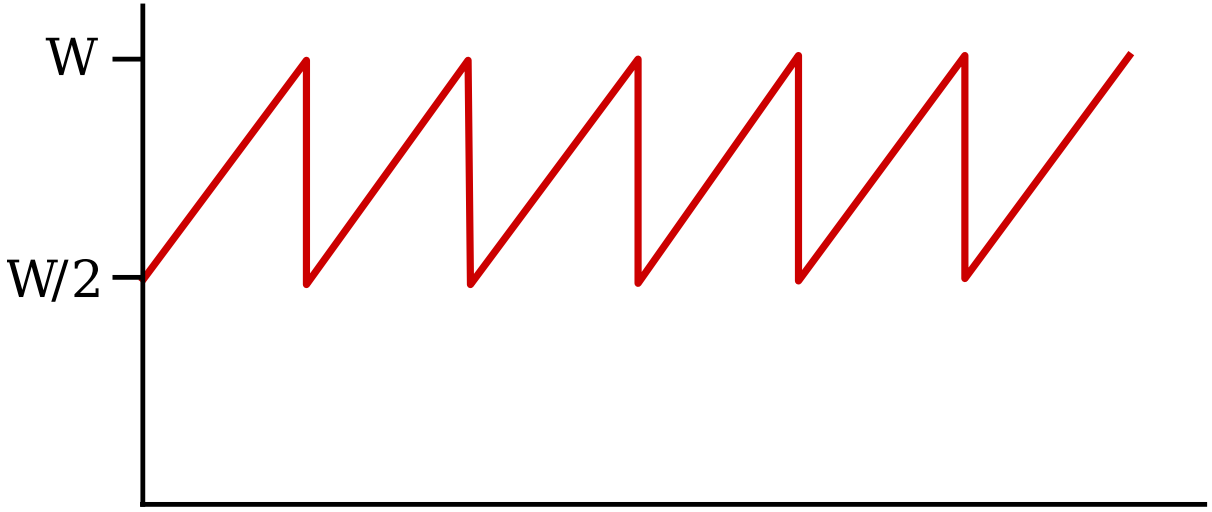
- **Implementação:**
  - Variável **ssthresh**.
  - Em um evento de perda, **ssthresh** recebe metade da **cwnd** imediatamente antes da perda.

Controle de Congestionamento do TCP Reno: Sumário



Vazão do TCP

- Calcular a vazão média como função do tamanho da janela, RTT?
  - Ignorar *slow start*, assumir que sempre há dados a enviar.
- W: tamanho da janela (medida em bytes) quando a perda ocorre.
  - Tamanho médio da janela (bytes em trânsito) é $\frac{3W}{4}$
  - Vazão média é $\frac{3}{4} \times \frac{W}{RTT}$.



Futuro do TCP: Canais “Longos” e de Alta Capacidade

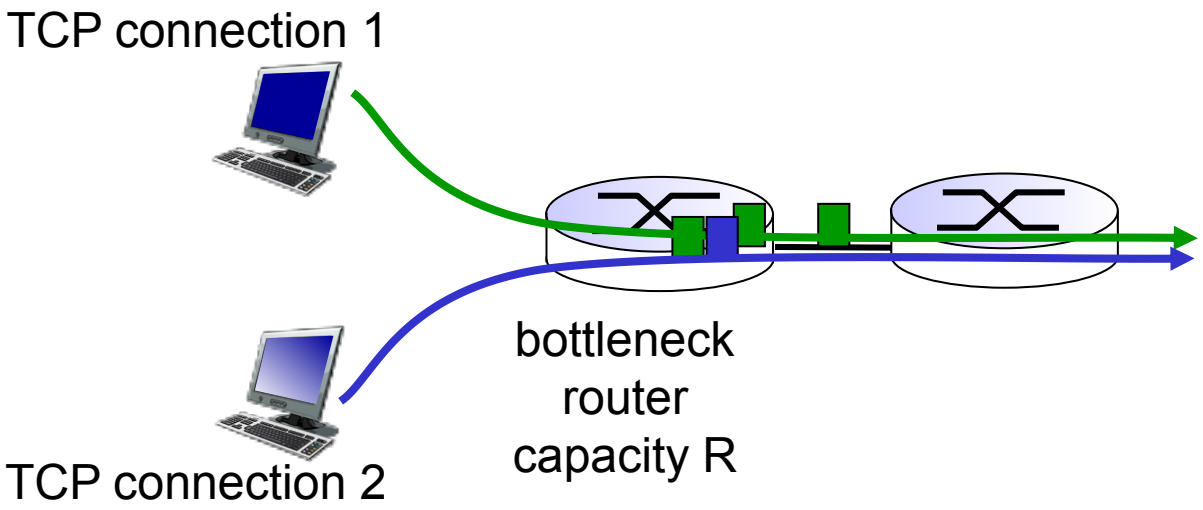
- Exemplo: segmentos de 1500 bytes, RTT de 100 ms, deseja-se vazão de 10 Gb/s.
- Requer W = 83333 segmentos em trânsito.
- Vazão TCP em termos da probabilidade de perda de segmentos L [Mathis 1997]:

$$T_{TCP} = \frac{1{,}22 \cdot MSS}{RTT \cdot \sqrt{L}}$$

- Para alcançar uma vazão de 10 Gb/s é necessária uma perda L = 2×10^{-10}.
  - **Extremamente baixa!**
- Novas versões do TCP para canais/redes de alta velocidade.

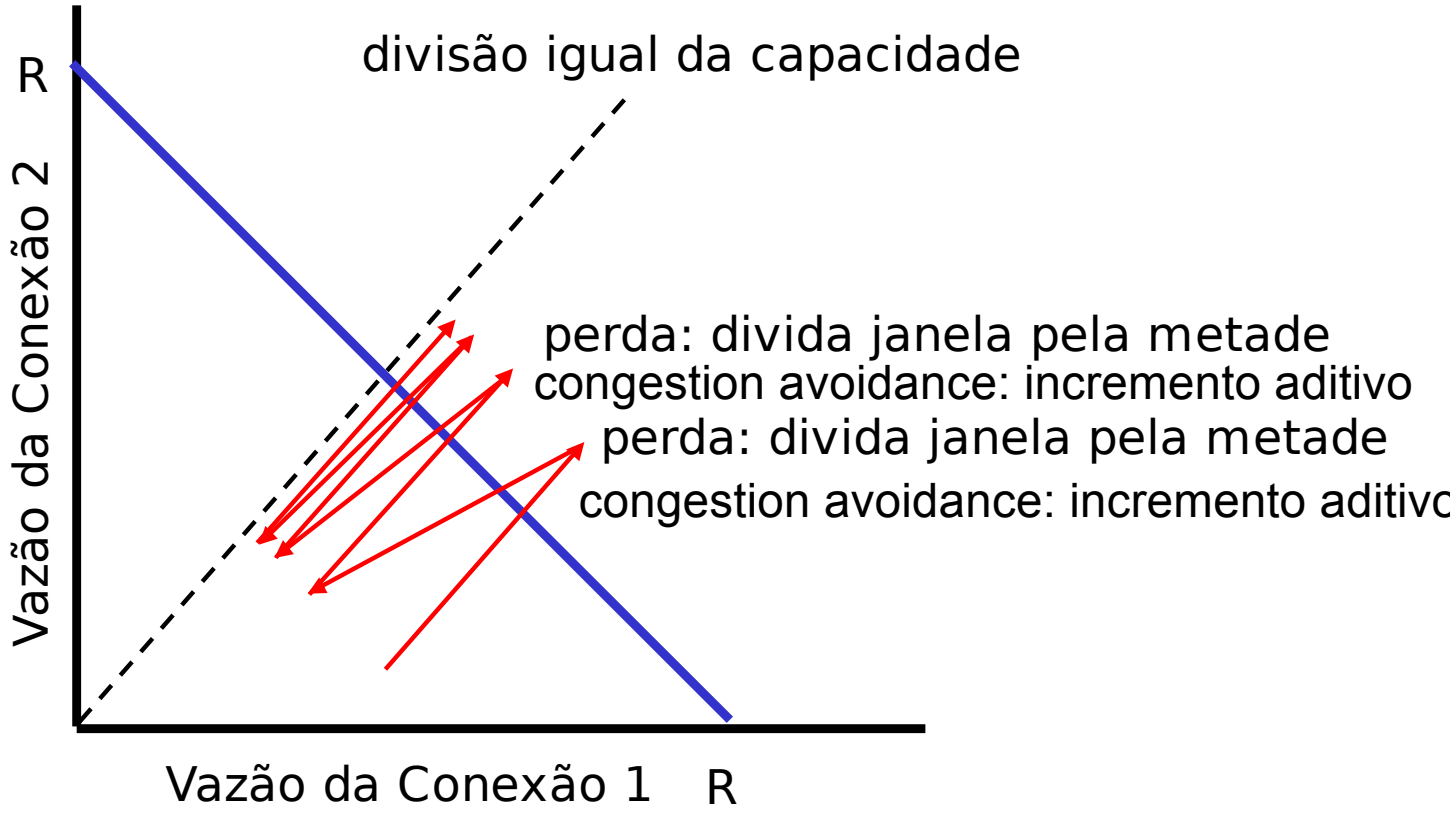
TCP: Justiça

- **Objetivo:** se k conexões TCP dividem o mesmo enlace de gargalo de capacidade R, cada conexão deveria obter uma taxa média de $\frac{R}{K}$.



Por que o TCP é Justo?

- Duas conexões competindo:
  - Crescimento aditivo sempre aumenta janela em 1 MSS.
  - Decremento multiplicativo reduz pela metade.



Justiça: Mais

- **Justiça e UDP:**
  - Aplicações multimidia muitas vezes não usam TCP.
    - Não querem que taxa seja reduzida pelo controle de congestionamento.
  - Ao invés disso, usam UDP.
    - Transmitem a uma taxa constante, toleram perdas.

- **Justiça, conexões TCP paralelas:**
  - Aplicação pode abrir múltiplas conexões simultâneas entre par de *hosts*.
  - *Browsers* fazem isso muitas vezes.
  - *e.g.*, enlace com capacidade R e 9 conexões TCP existentes.
    - Nova aplicação abre 1 conexão TCP, recebe vazão de $\frac{R}{10}$.
    - Nova aplicação abre 11 conexões TCP, recebe vazão de $\frac{11R}{20} > \frac{R}{2}$.

Resumo da Aula (I)...

- **Controle de Congestionamento:**
  - Evitar congestionamento da rede.
  - Limitar taxa de transmissão das fontes.
  - **≠ controle de fluxo.**
- **Custos do congestionamento:**
  - Atrasos altos.
  - Retransmissões.
  - Queda no *goodput*.
  - Desperdício de recursos.
- **Duas abordagens:**
  - Fim-a-fim (inferido pelos *hosts*).
  - Assistido pela rede: explicitamente avisado.

- **Controle de congestionamento do TCP:**
  - **Fim-a-fim.**
  - Dividido em fases: *Slow Start*, *Congestion Avoidance*.
  - **Inferido via perdas**.
- ***Slow Start*:**
  - Taxa começa lenta, mas **aumenta exponencialmente**.
  - Aumento de 1 MSS a cada ACK.
  - Executado até que **cwnd ≥ ssthresh**.
- ***Congestion Avoidance*: AIMD.**
  - Incremento Aditivo: 1 MSS por RTT.
  - Decremento multiplicativo: divide pela metade em caso de perda.

Resumo da Aula (II)...

- **Em caso de perda:**
  - **ssthresh = cwnd/2**.
  - **cwnd = 1 MSS**.
  - Volta-se ao *Slow Start*.
- ***Fast Recovery*:**
  - Otimização do TCP Reno (*vs.* Tahoe).
  - Perda por ACK duplicado.
  - Tenta se manter no CA.
- **Vazão do TCP:**
  - Eficiência menor que 100%.
  - Pior para enlaces com alta capacidade e alto RTT.
- **TCP: justiça.**
  - Objetivo: divisão justa da vazão.
  - Sob certas condições, TCP alcança.
  - UDP, conexões múltiplas podem interferir.

Sumário do Capítulo 3

- Princípios por trás dos serviços da camada de transporte.
  - Multiplexação, demultiplexação.
  - Transferência confiável de dados.
  - Controle de fluxo.
  - Controle de congestionamento.
- Instanciação, implementação na Internet.
  - UDP.
  - TCP.

Próxima Aula...

- Começamos um novo capítulo: camada de rede.
- Último do período.
- Particularmente, na próxima aula:
  - Conceitos básicos da camada de rede.
  - Redes de datagramas *vs.* circuitos virtuais.