Redes de Computadores I

Aula 13 – Camada de Transporte (Transferência Confiável)

Prof. Carlos Giovanni N. de Carvalho

cgnc@ctu.uespi.br

Agenda

- Princípios de Transferência Confiável

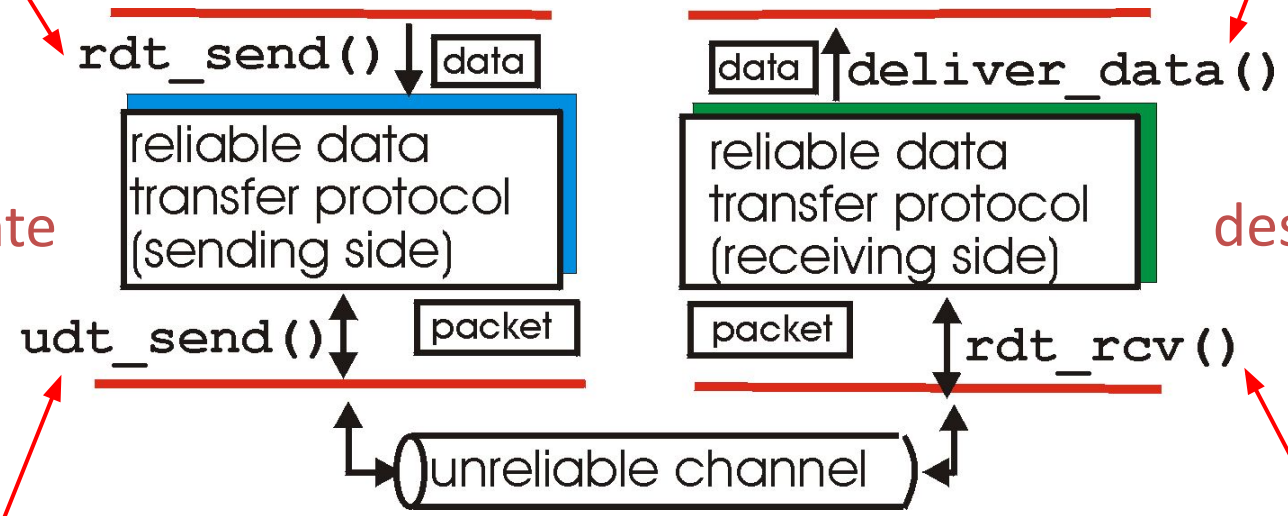
Transferência Confiável

- Princípios
  - Importante nas camadas de aplicação, transporte e enlace
  - Características do canal confiável determinarão complexidade do protocolo de transferência confiável – Reliable Data Transfer (RDT)

Transferência Confiável

rdt_send(): chamado de cima, (p. e., pela apl.). Dados passados para remeter à camada superior do destinatário

deliver_data(): chamado pela **rdt** para remeter dados para cima



lado remetente

lado destinatário

udt_send(): chamado pela rdt, para transferir pacote por canal não confiável ao destinatário

rdt_rcv(): chamado quando pacote chega no lado destinatário do canal

Transferência Confiável

- RDT 1.0
  - Transferência confiável por canal confiável
    - Canal subjacente perfeitamente confiável
      - Sem erros de bit
      - Sem perda de pacotes

Transferência Confiável

- RDT 2.0
  - Canal com Erros de Bit
    - Canal subjacente pode inverter bits no pacote
      - Soma de verificação para detectar erros de bit
    - A questão: como recuperar-se dos erros:
      - Reconhecimentos (ACKs): destinatário diz explicitamente ao remetente que o pacote foi recebido OK
      - Reconhecimentos negativas (NAKs): destinatário diz explicitamente ao remetente que o pacote teve erros
      - Remetente retransmite pacote ao receber NAK

Transferência Confiável

- RDT 2.0
  - Canal com Erros de Bit
    - Novos mecanismos além do RDT 1.0:
      - Detecção de erro
      - Feedback do destinatário: msgs de controle (ACK,NAK) destinatário->remetente

Transferência Confiável

- RDT 2.0
  - Falha fatal
    - O que acontece se ACK/NAK for corrompido?
      - Remetente não sabe o que aconteceu no destinatário!
      - Não pode simplesmente retransmitir: possível duplicação
    - Tratando de duplicatas:
      - Remetente retransmite pacote atual se ACK/NAK corrompido
      - Remetente acrescenta número de sequência a cada pacote
      - Destinatário descarta (não sobe) pacote duplicado
    - Pare e espere
      - Remetente envia um pacote,
      - Depois espera resposta do destinatário

Transferência Confiável

- RDT 2.1
  - Remetente trata de ACK/NACK Corrompidos
    - Remetente
      - # seq acrescentado ao pkt
      - Dois #s seq. (0,1) bastarão. Por quê?
      - Deve verificar se ACK/NAK recebido foi corrompido
      - O dobro de estados
        - » Estado de “lembrar” se pacote “atual” tem # seq. 0 ou 1

Transferência Confiável

- RDT 2.1
  - Remetente trata de ACK/NACK Corrompidos
    - Destinatário
      - Deve verificar se pacote recebido está duplicado
        - » Estado indica se 0 ou 1 é # seq. esperado do pacote
      - Nota: destinatário não sabe se seu último ACK/NAK foi recebido OK no remetente

Transferência Confiável

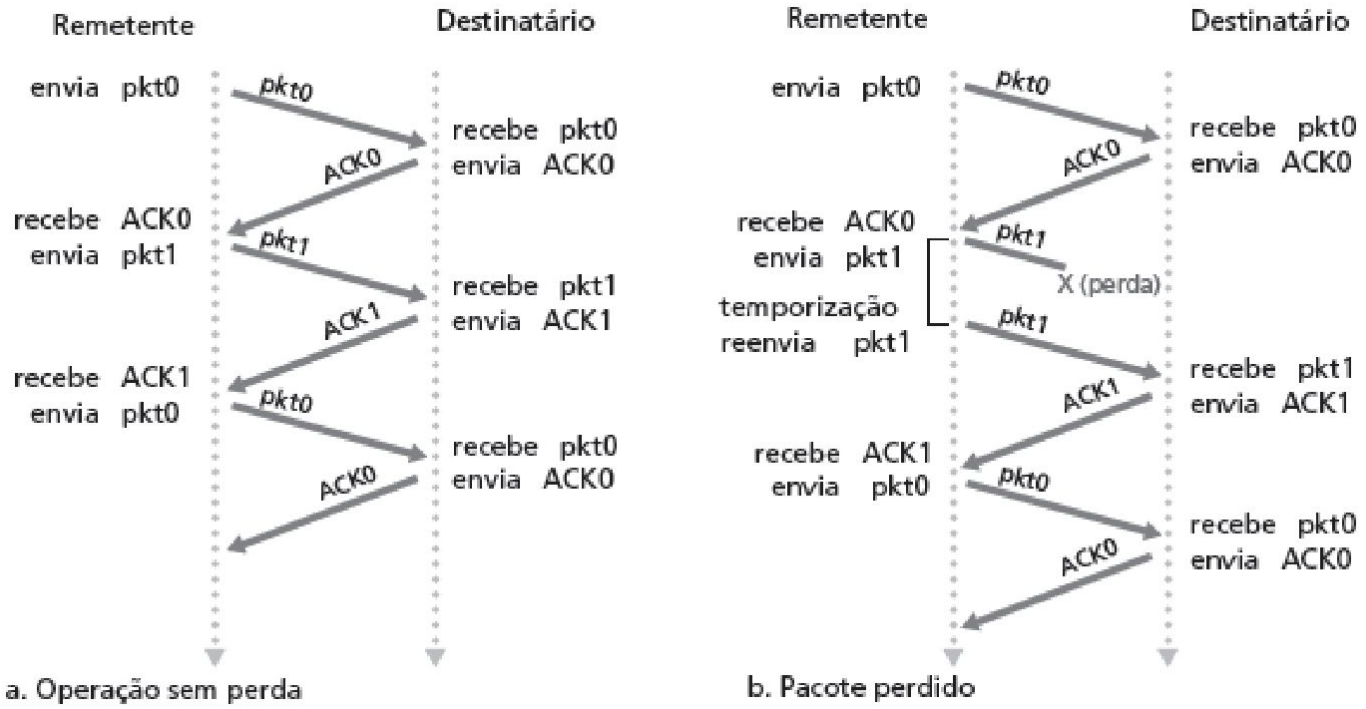
- RDT 2.2
  - Um Protocolo Sem NACK
    - Mesma funcionalidade de RDT 2.1, usando apenas ACKs
    - Em vez de NAK, destinatário envia ACK para último pacote recebido OK
      - Destinatário precisa incluir explicitamente # seq. do pacote sendo reconhecido com ACK
    - ACK duplicado no remetente resulta na mesma ação de NAK: retransmitir pacote atual

Transferência Confiável

- RDT 3.0
  - Canais Com Erros e Perdas
    - Nova suposição: canal subjacente também pode perder pacotes (dados ou ACKs)
      - Soma de verificação, # seq., ACKs, retransmissões serão úteis, mas não suficientes
    - Técnica: remetente espera quantidade “razoável” de tempo por ACK
      - Retransmite se não chegar ACK nesse tempo
      - Se pct (ou ACK) simplesmente atrasado (não perdido):
        - » Retransmissão será duplicada, mas os #s de seq. já cuidam disso
        - » Destinatário deve especificar # seq. do pacote sendo reconhecido com ACK
      - Requer contador regressivo

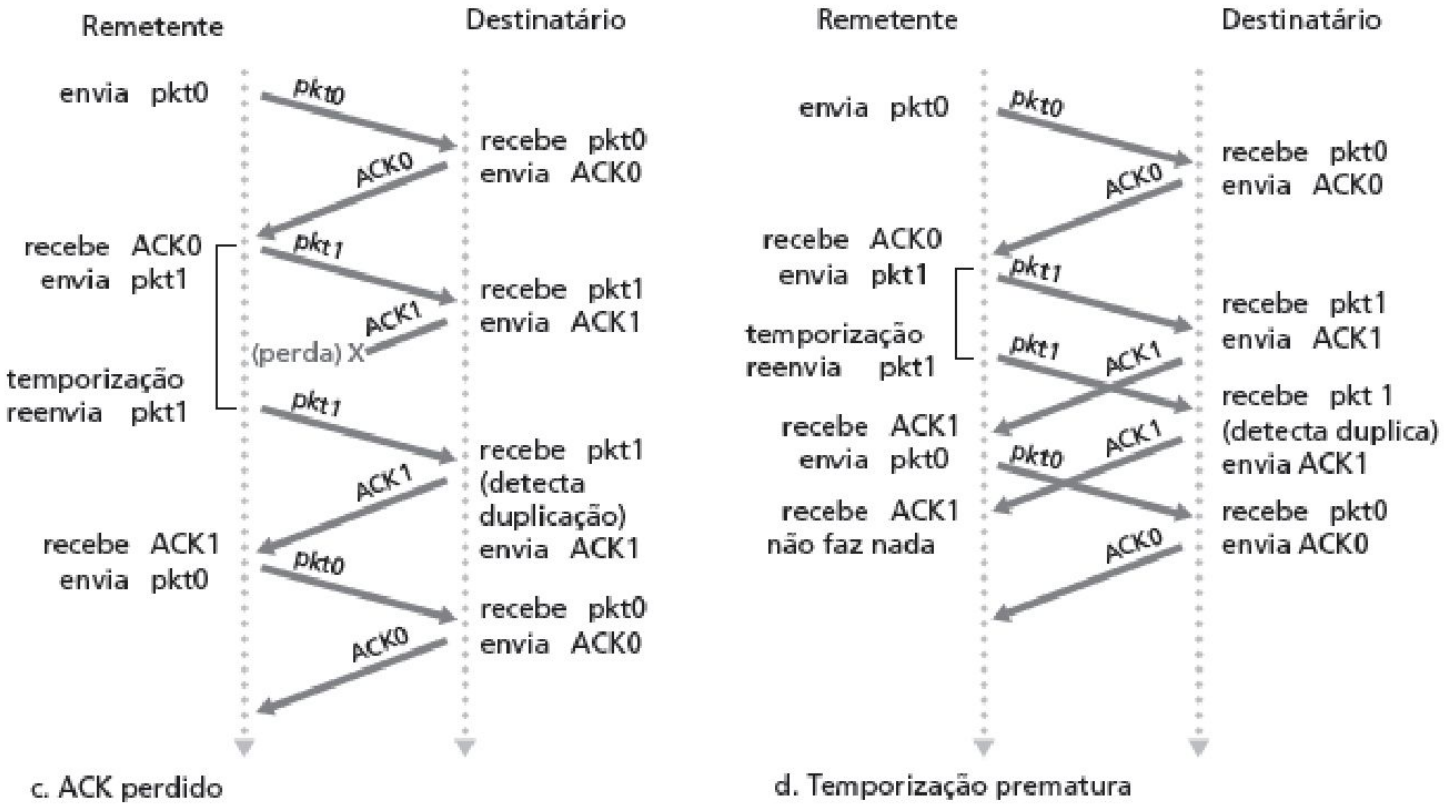
Transferência Confiável

- RDT 3.0



Transferência Confiável

- RDT 3.0



Transferência Confiável

- Desempenho Ruim do RDT 3.0
  - Ex.: enlace 1 Gbps, 15 ms atraso propriedade, pacote 8000 bits:

$$d_{trans} = \frac{L}{R} = \frac{8000\text{bits}}{10^9\,\text{bps}} = 8\,microssegundos$$

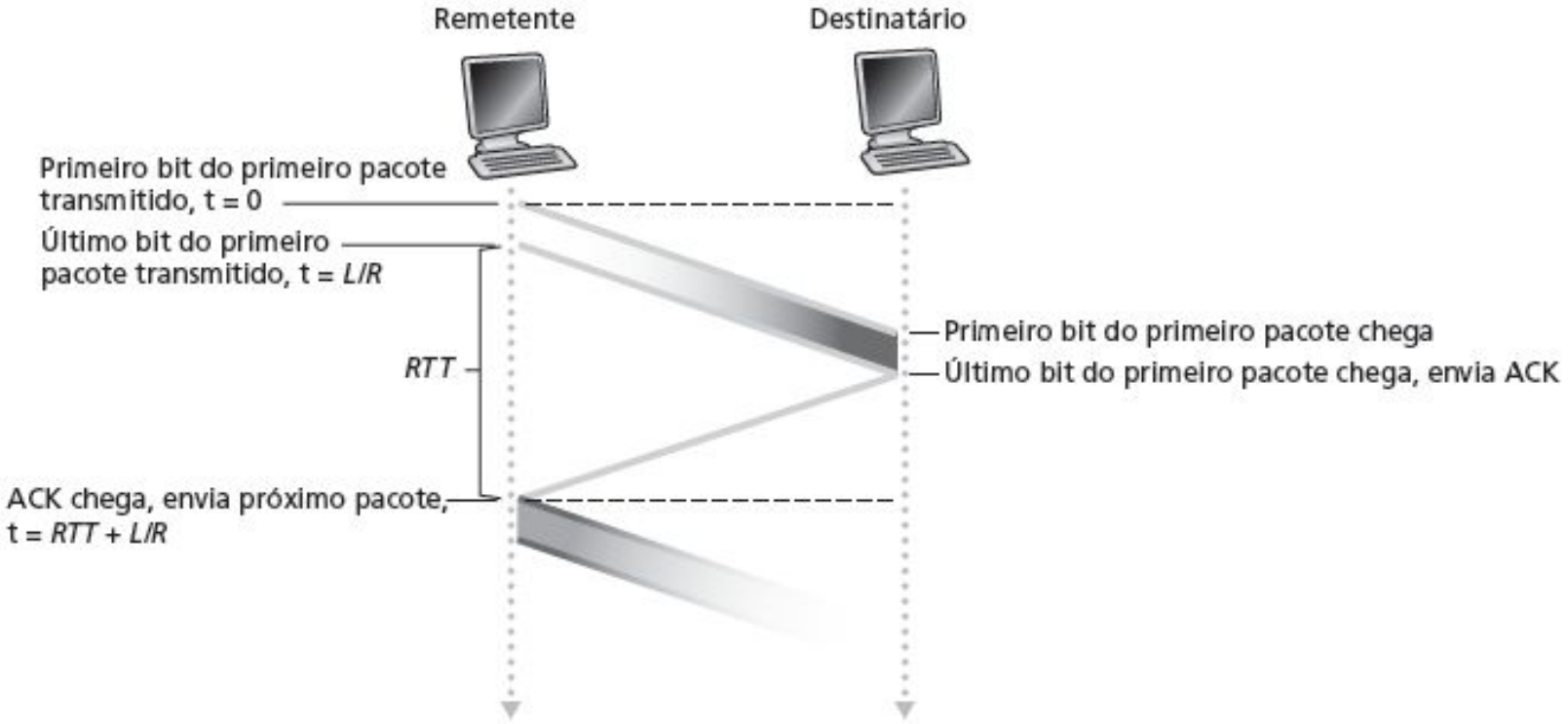
  - U_{remet}: utilização – fração do tempo remet. ocupado enviando

$$U_{remet} = \frac{L / R}{RTT + L / R} = \frac{0{,}008}{30{,}008} = 0{,}00027$$

  - Pct. 1 KB cada 30 ms -> 33 kB/s vazão em enlace de 1 Gbps
    - Protocolo de rede limita uso de recursos físicos!

Transferência Confiável

- Desempenho Ruim do RDT 3.0

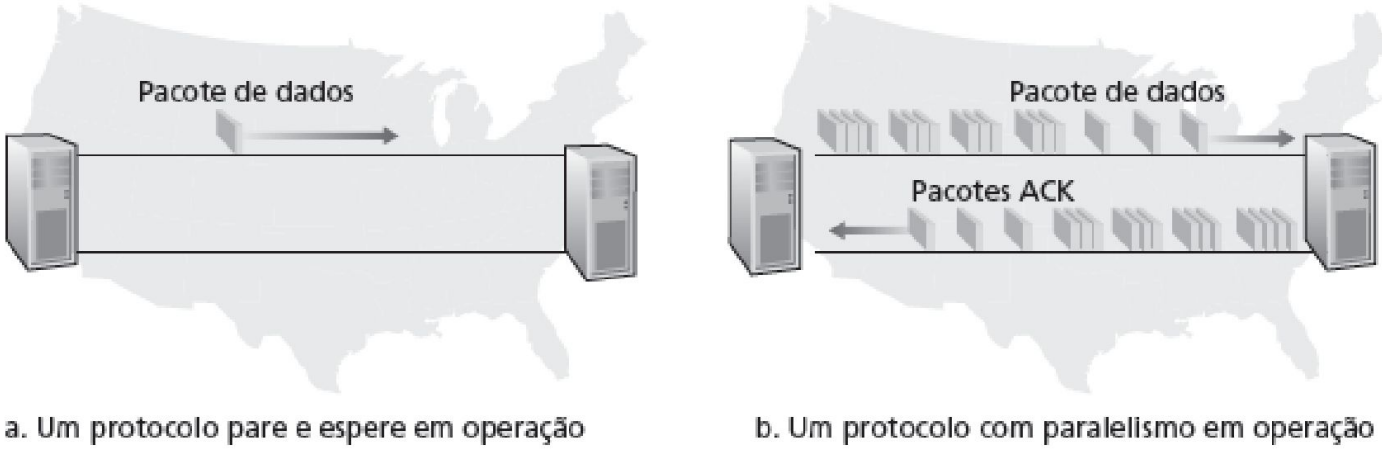


Transferência Confiável

- Protocolos com Paralelismo
  - Remetente permite múltiplos pacotes “no ar”, ainda a serem reconhecidos
    - Intervalo de números de sequência deve ser aumentado
    - Buffering no remetente e/ou destinatário
  - Duas formas genéricas de protocolo com paralelismo: Go-Back-N, repetição seletiva

Transferência Confiável

- Protocolos com Paralelismo

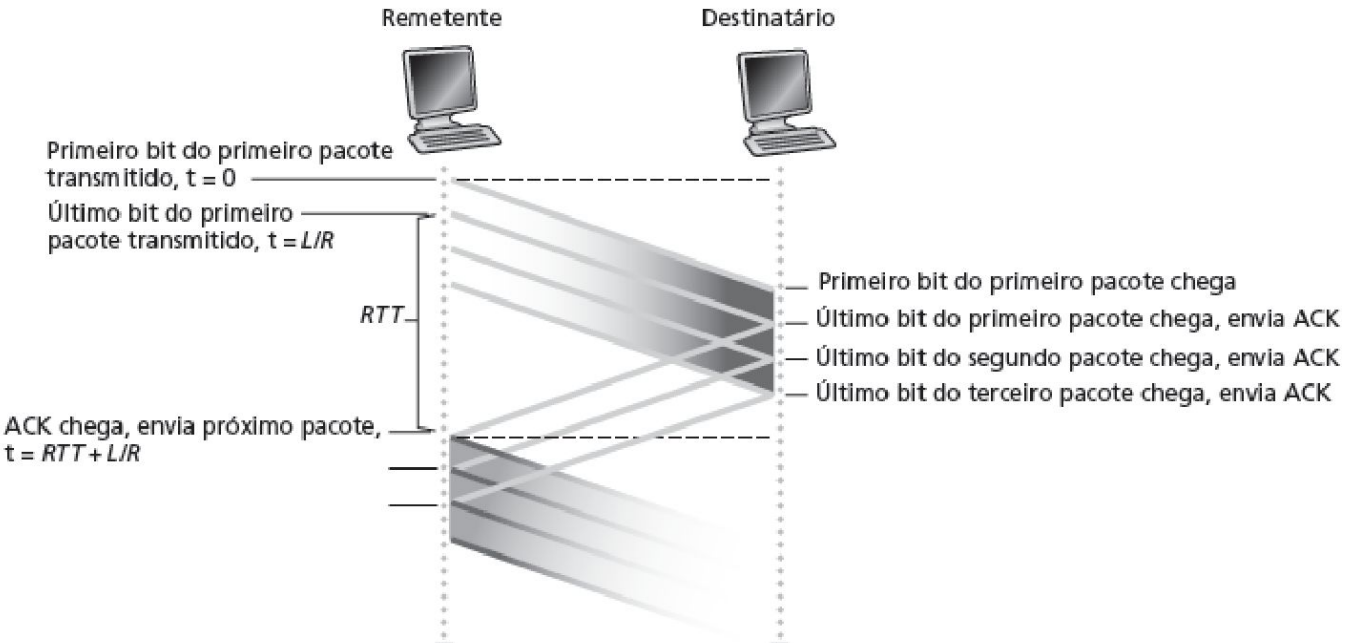


a. Um protocolo pare e espere em operação

b. Um protocolo com paralelismo em operação

Transferência Confiável

- Protocolos com Paralelismo
  - Utilização aumentada



$$U_{remet} = \frac{3 * L / R}{RTT + L / R} = \frac{0{,}02}{30{,}008} = 0{,}0008$$

Transferência Confiável

- Protocolos com Paralelismo
  - Go-back-N: visão geral
    - Remetente: até N pacotes não reconhecidos na pipeline
    - Destinatário: só envia ACKs cumulativos
      - Não envia pct ACK se houver uma lacuna
    - Remetente: tem temporizador para pct sem ACK mais antigo
      - Se o temporizador expirar: retransmite todos os pacotes sem ACK

Transferência Confiável

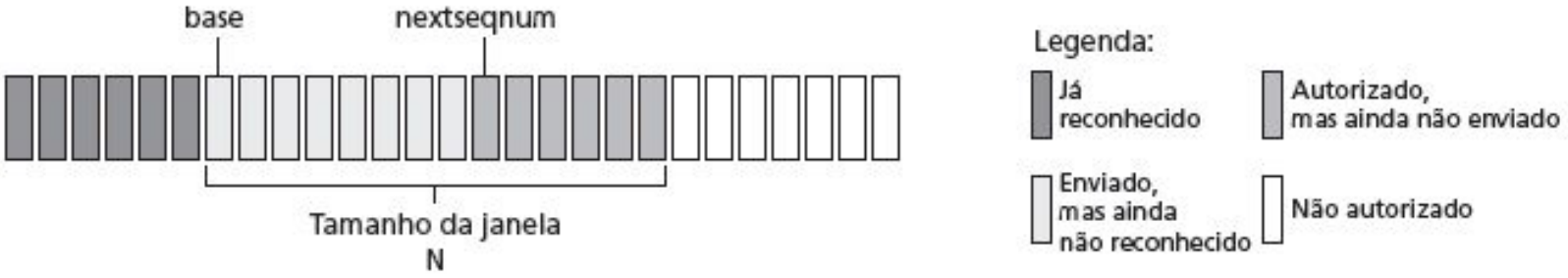
- Protocolos com Paralelismo
  - Repetição seletiva: visão geral
    - Remetente: até pacotes não reconhecidos na pipeline
    - Destinatário: reconhece (ACK) pacotes individuais
    - Remetente: mantém temporizador para cada pct sem ACK
      - Se o temporizador expirar: retransmite apenas o pacote sem ACK

Transferência Confiável

- Protocolos com Paralelismo
  - Go-Back-N
    - Remetente:
      - # seq. de k bits no cabeçalho do pacote
      - “janela” de até N pcts consecutivos sem ACK permitidos
    - ACK(n): ACK de todos pcts até inclusive # seq. n – “ACK cumulativo”
      - Pode receber ACKs duplicados (ver destinatário)
    - Temporizador para cada pacote no ar
    - timeout(n): retransmite pct n e todos pcts com # seq. mais alto na janela

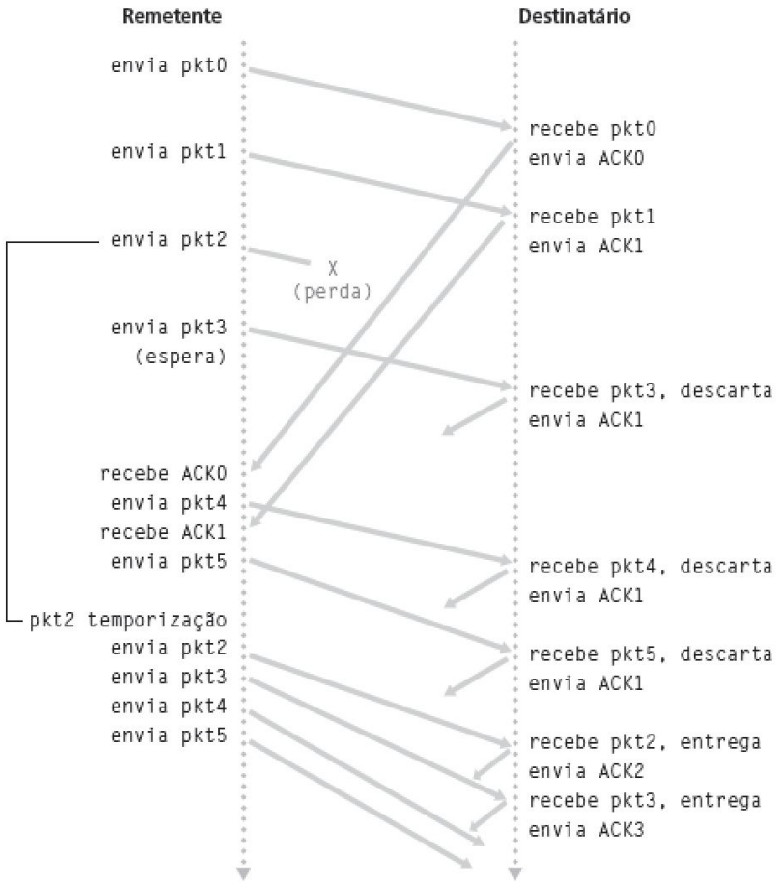
Transferência Confiável

- Protocolos com Paralelismo
  - Go-Back-N



Transferência Confiável

- Protocolos com Paralelismo
  - Go-Back-N

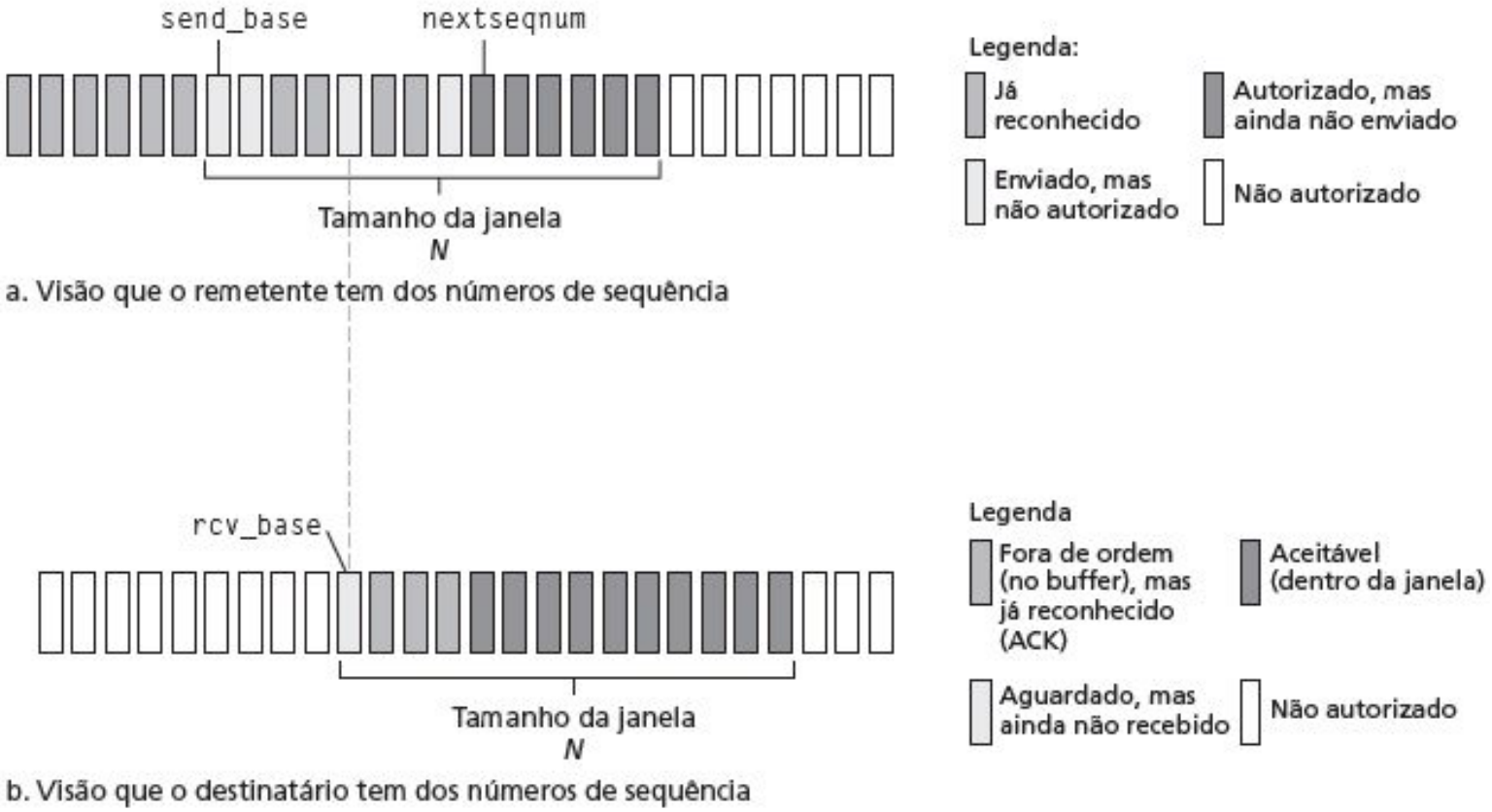


Transferência Confiável

- Protocolos com Paralelismo
  - Repetição Seletiva
    - Destinatário reconhece individualmente todos os pacotes recebidos de modo correto
      - Mantém pcts em buffer, se for preciso, para eventual remessa em ordem para a camada superior
    - Remetente só reenvia pcts para os quais o ACK não foi recebido
      - Temporizador no remetente para cada pct sem ACK
    - Janela do remetente
      - N # seq. consecutivos
      - Novamente limita #s seq. de pcts enviados, sem ACK

Transferência Confiável

- Protocolos com Paralelismo
  - Repetição Seletiva



a. Visão que o remetente tem dos números de sequência

b. Visão que o destinatário tem dos números de sequência

Transferência Confiável

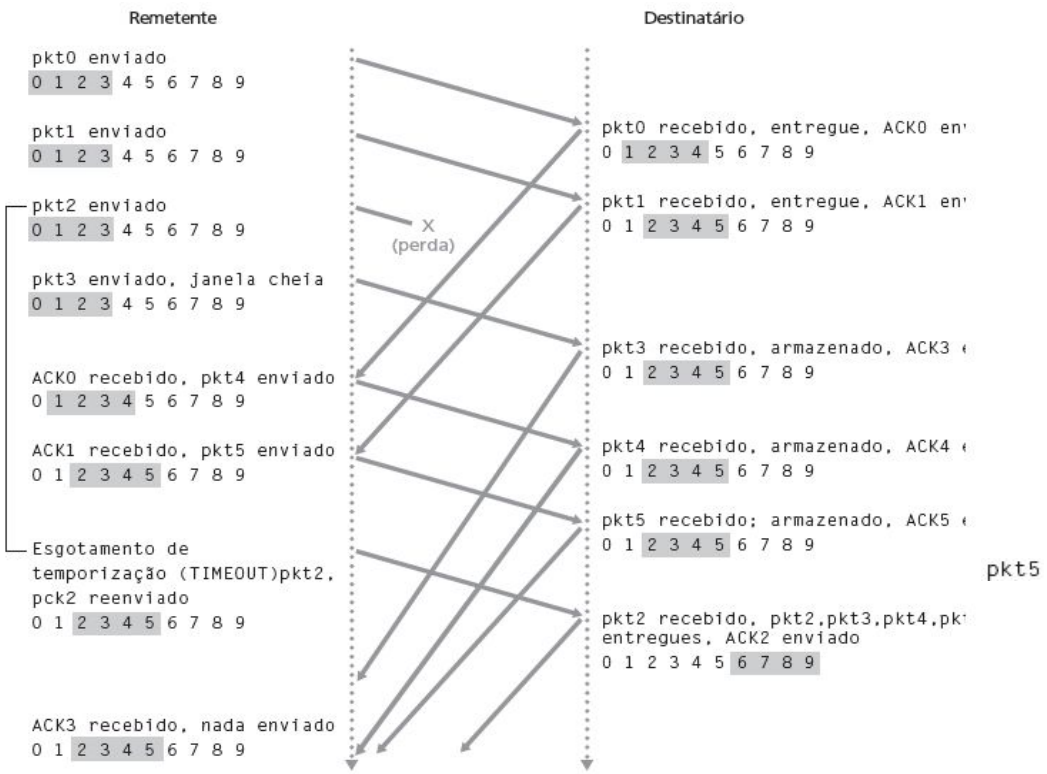
- Protocolos com Paralelismo
  - Repetição Seletiva
    - Remetente
      - Dados de cima:
        - » Se próx. # seq. disponível na janela, envia pct
      - timeout(n):
        - » Reenvia pct n, reinicia temporizador
      - ACK(n) em [sendbase,sendbase+N]:
        - » Marca pct n como recebido
        - » Se n menor pct com ACK, avança base da janela para próximo # seq. sem ACK

Transferência Confiável

- Protocolos com Paralelismo
  - Repetição Seletiva
    - Destinatário
      - pct n em [rcvbase, rcvbase+N-1]
        - » Envia ACK(n)
        - » Fora de ordem: buffer
        - » Em ordem: entrega (também entrega pcts em ordem no buffer), avança janela para próximo pct ainda não recebido
      - pct n em [rcvbase-N,rcvbase-1]
        - » ACK(n)
      - caso contrário:
        - » Ilgnora

Transferência Confiável

- Protocolos com Paralelismo
  - Repetição Seletiva

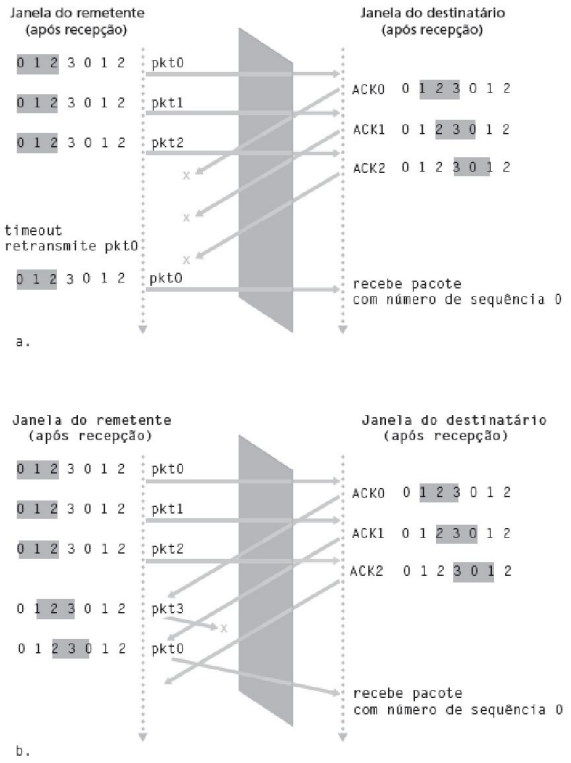


Transferência Confiável

- Protocolos com Paralelismo
  - Dilema da Repetição Seletiva
    - Exemplo:
      - # seq.: 0, 1, 2, 3
      - Tamanho janela = 3
      - Destinatário não vê diferença nos dois cenários!
      - Passa incorretamente dados duplicados como novos em (a)
      - P: Qual o relacionamento entre tamanho do # seq. e tamanho de janela?

Transferência Confiável

- Protocolos com Paralelismo
  - Dilema da Repetição Seletiva



Bibliografia

- KUROSE, James F. ROSS, Keith W. Redes de Computadores e a Internet. 3ª Edição, Pearson, 2007.
- STALLINGS, William. Redes e Sistemas de Comunicação de Dados. 5ª ed., Editora Campus, Rio de Janeiro – RJ, 2005.
- TANENBAUM, Andrew. Redes de Computadores. 4ª ed., Editora Campus, Rio de Janeiro – RJ, 2003.