Aula 11 - Protocolos Baseados em Pipeline e Introdução ao TCP

Diego Passos

Universidade Federal Fluminense

Redes de Computadores I

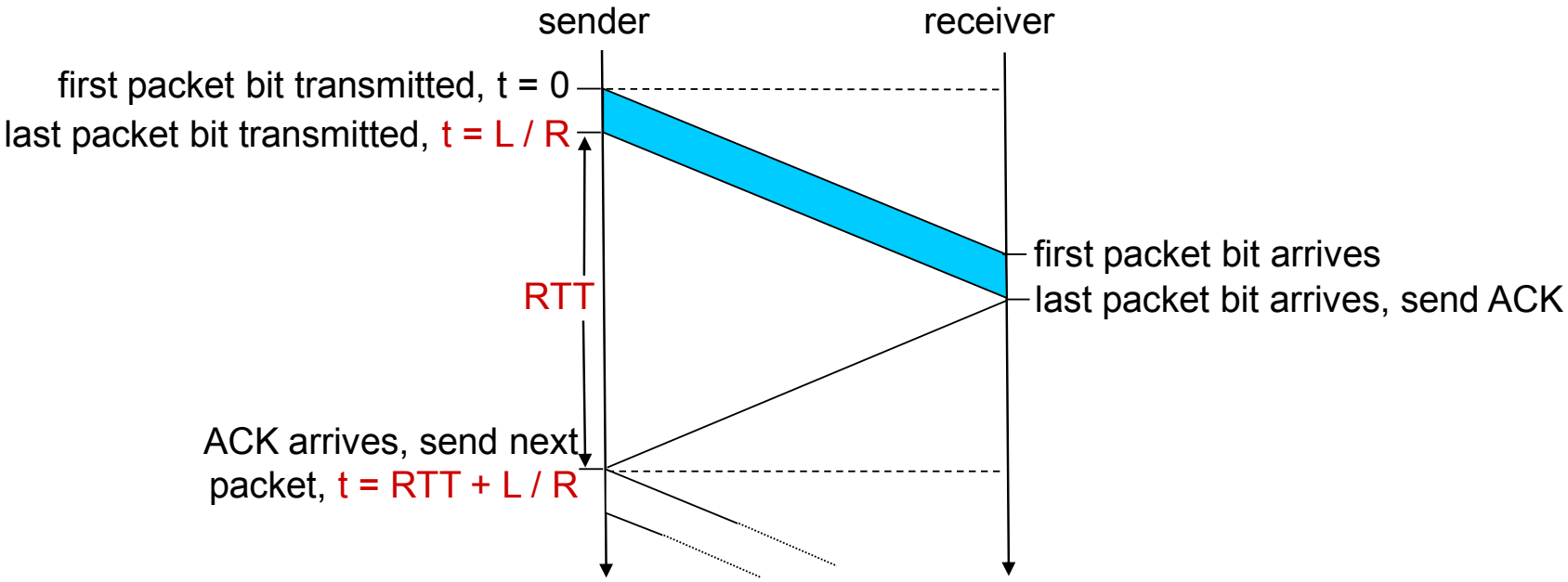
Material adaptado a partir dos slides originais de J.F Kurose and K.W. Ross.

Revisão da Última Aula...

- **Transferência confiável de dados:**
  - Garantir entrega.
  - Garantir integridade.
  - Garantir ordenação.
  - **Embora a rede não garanta.**
- ***Checksum*:**
  - Verificação de integridade.
  - Pacotes errados são descartados.
- **ACK:**
  - Confirmação **positiva do recebimento**.
- **NAK:**
  - Confirmação **negativa do recebimento**.
- **Retransmissões:**
  - Pacotes perdidos/errados são retransmitidos.
- **Números de sequência:**
  - Identifica duplicatas.
  - Duplicatas descartadas.
  - Dá mais flexibilidade às retransmissões.
  - Permite supressão dos NAKs:
    - **ACK duplicado = NAK**.
- **Temporizador:**
  - Necessario se rede descarta pacotes.
  - **Estouro de temporizador interpretado como pacote perdido.**
    - Retransmissão.
- ***Stop-and-wait*:**
  - Novo pacote transmitido apenas após ACK.
  - **Limita desempenho**.

Protocolos Baseados em Pipeline

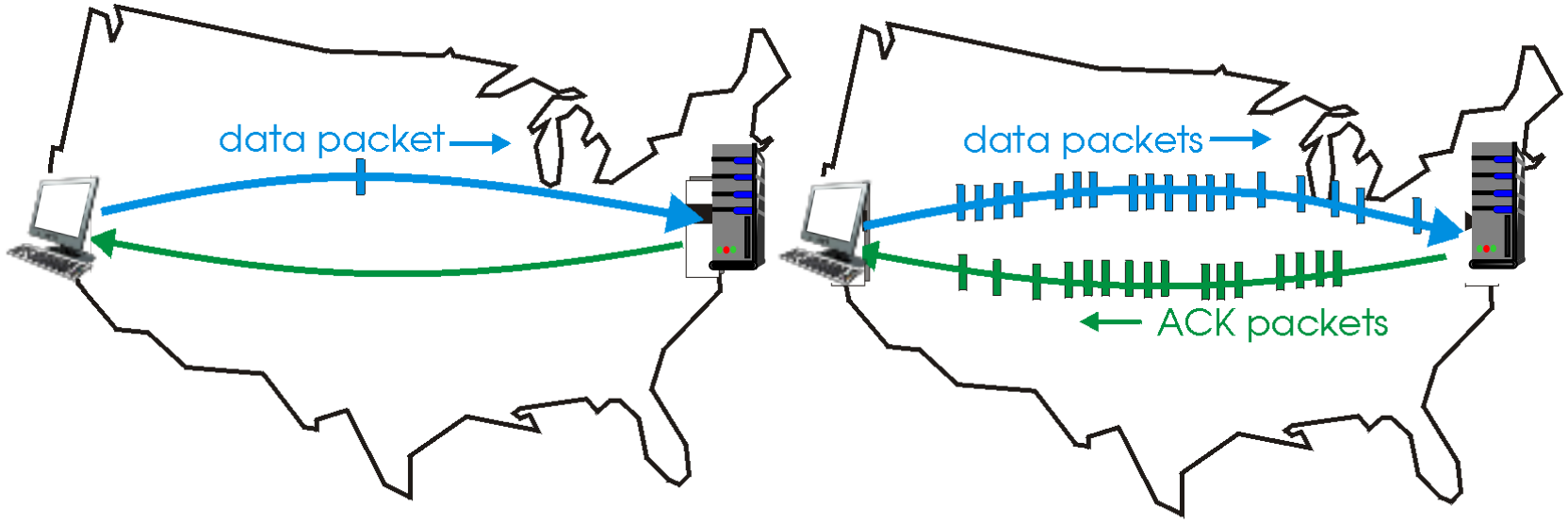
Operação do Tipo *Stop-and-Wait*



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

Protocolos Baseados em Pipeline

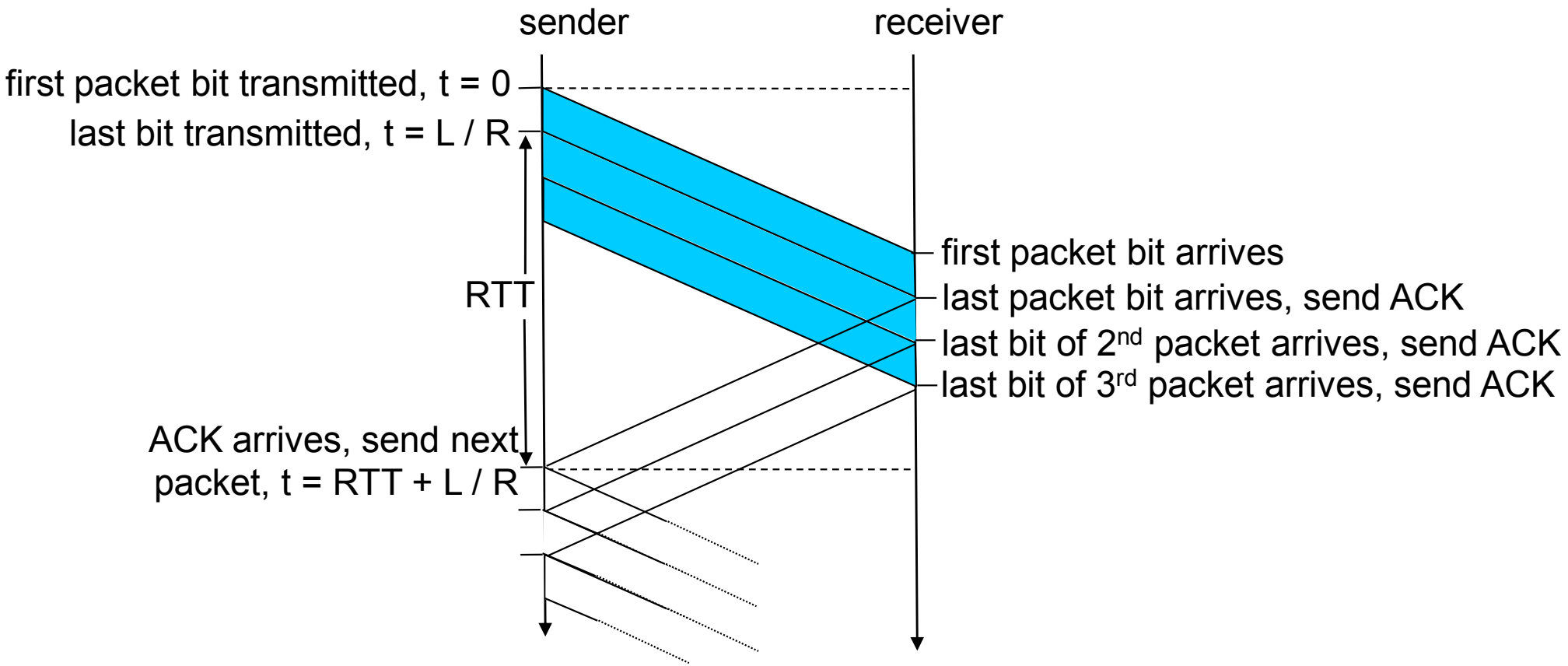
- **Pipeline:** permite que transmissor tenha múltiplos segmentos **em trânsito**.
  - *i.e.*, segmentos enviados, mas com ACK ainda pendente.
  - Faixa dos números de sequência precisa ser aumentada.
  - *Buffers* necessários no transmissor e/ou no receptor.



(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

- Duas formas genéricas de protocolos baseados em pipeline: **go-Back-N** e **repetição seletiva**.

Pipeline: Aumentando a Utilização



$$U_{sender} = \frac{3L/R}{RTT+L/R} = \frac{0{,}0024}{30{,}008} = 0{,}00081$$

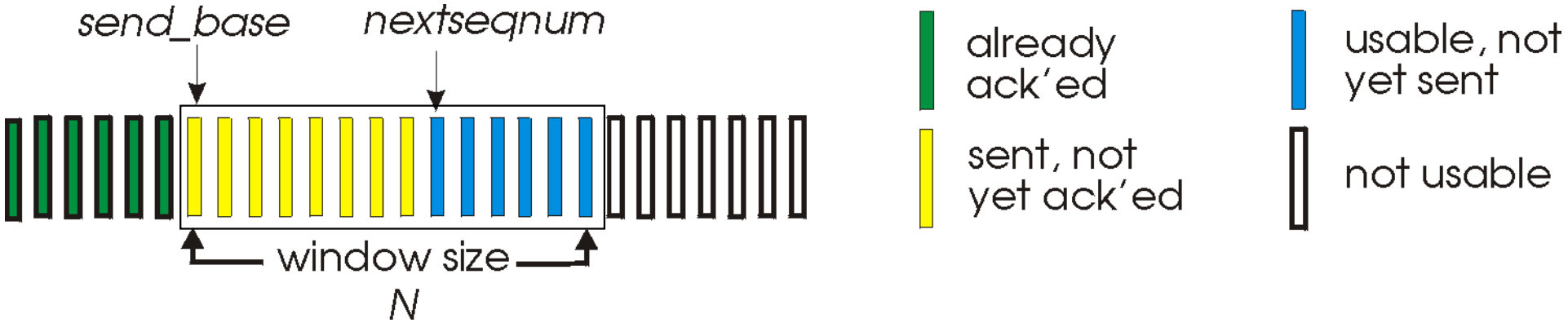
Três vezes mais que no *Stop-and-Wait*.

Protocolos Baseados em Pipeline: Visão Geral

- **Go-back-N:**
  - Transmissor pode ter até N segmentos em trânsito no pipeline.
  - Receptor envia apenas **ACKs cumulativos**.
    - Não reconhece pacote se há um "buraco".
  - Transmissor possui um temporizador para o pacote mais antigo em trânsito.
    - De menor número de sequência.
    - Quando o temporizador expira, **todos** os pacotes em trânsito são retransmitidos.
- **Repetição Seletiva:**
  - Transmissor pode ter até N segmentos em trânsito no pipeline.
  - Receptor envia **ACKs seletivos**.
    - *i.e.*, segmentos são reconhecidos individualmente.
  - Transmissor mantém um timer para cada pacote em trânsito.
    - Quanto temporizador expira, **apenas** segmento correspondente é retransmitido.

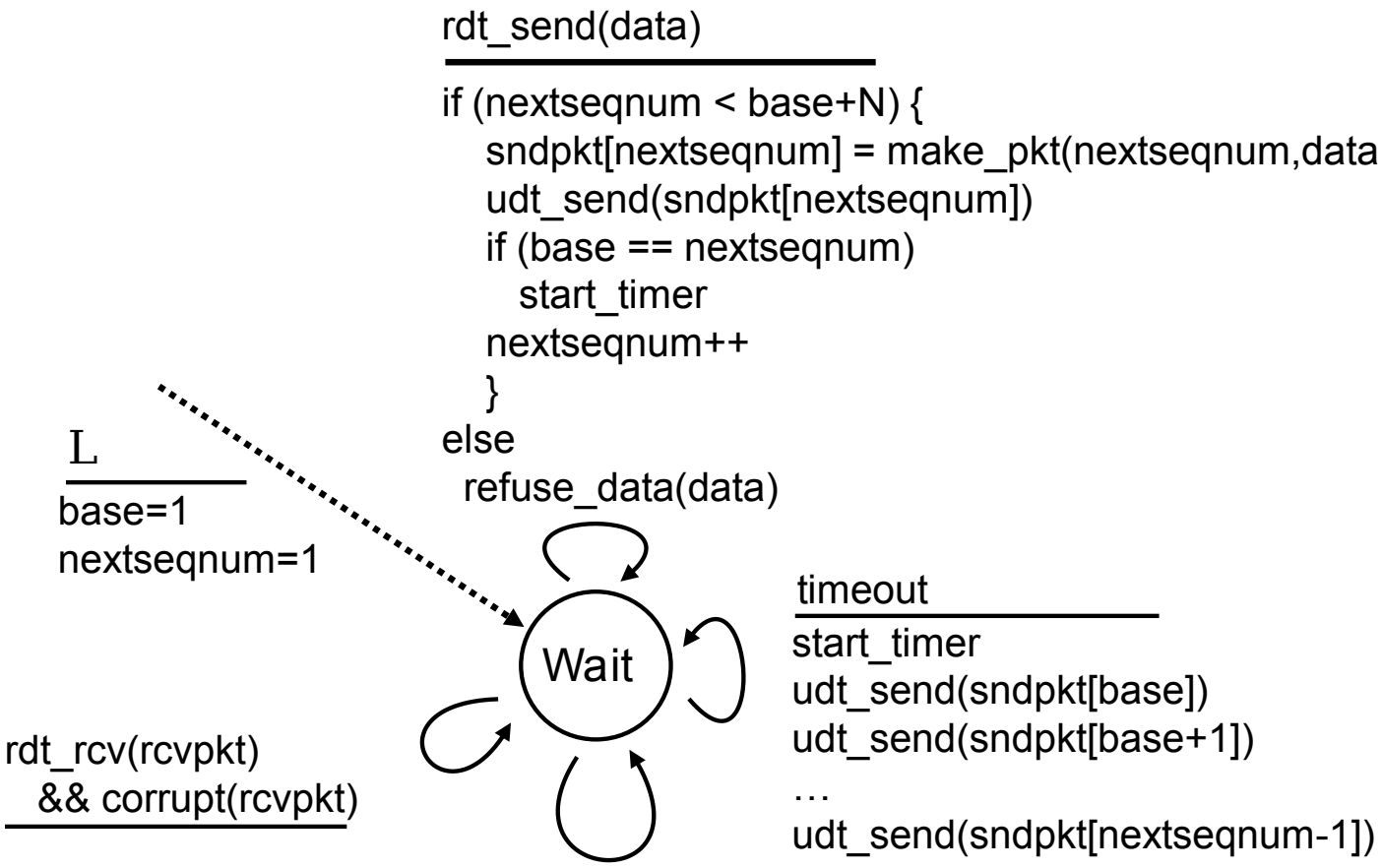
Go-Back-N: Transmissor

- Cabeçalho do segmento contém campo de k bits para o # de sequência.
- “Janela de até N” pacotes em trânsito consecutivos.

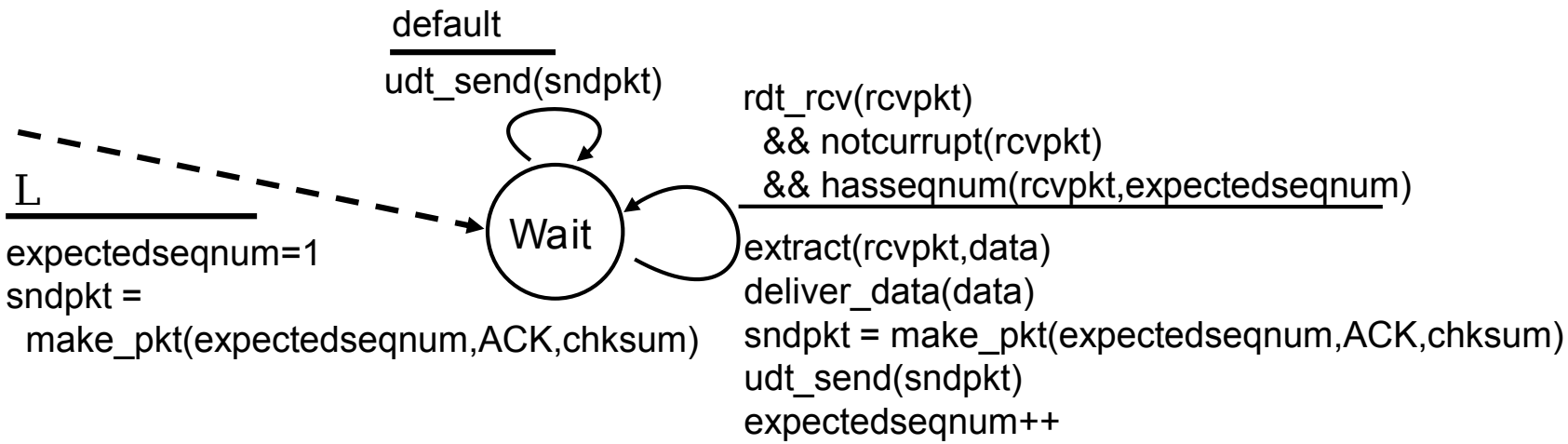


- ACK(n): reconhece todos os pacotes, incluindo o de # de sequência n.
  - **ACK cumulativo**.
  - ACKs repetidos podem ser recebidos (vide receptor).
- Temporizador para o segmento em trânsito mais antigo.
  - Quando expira, todos os pacotes em trânsito são retransmitidos.

Go-Back-N: Máquina de Estados do Transmissor

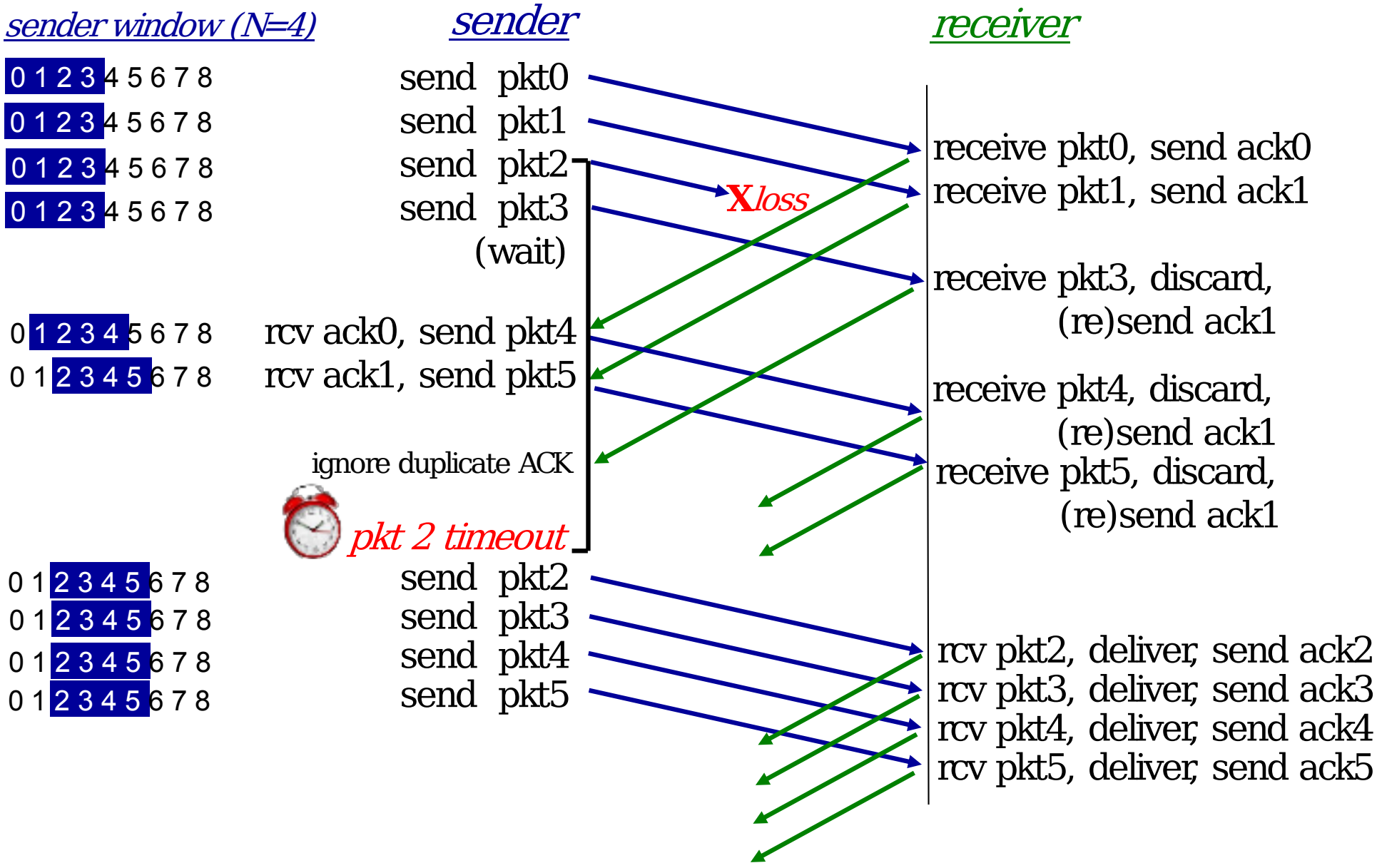


Go-Back-N: Máquina de Estados do Receptor



- Apenas ACK: sempre envia ACK para segmentos corretos reconhecendo recebimento do **maior # de sequência em ordem**.
  - Pode gerar ACKs duplicados.
  - Precisa se lembrar apenas do próximo número de sequência esperado.
- Pacote fora de ordem:
  - Descartado (não é armazenado em *buffer*): **sem *buffer* de recepção**.
  - Mesmo assim, receptor gera ACK para maior # de sequência já recebido em ordem.

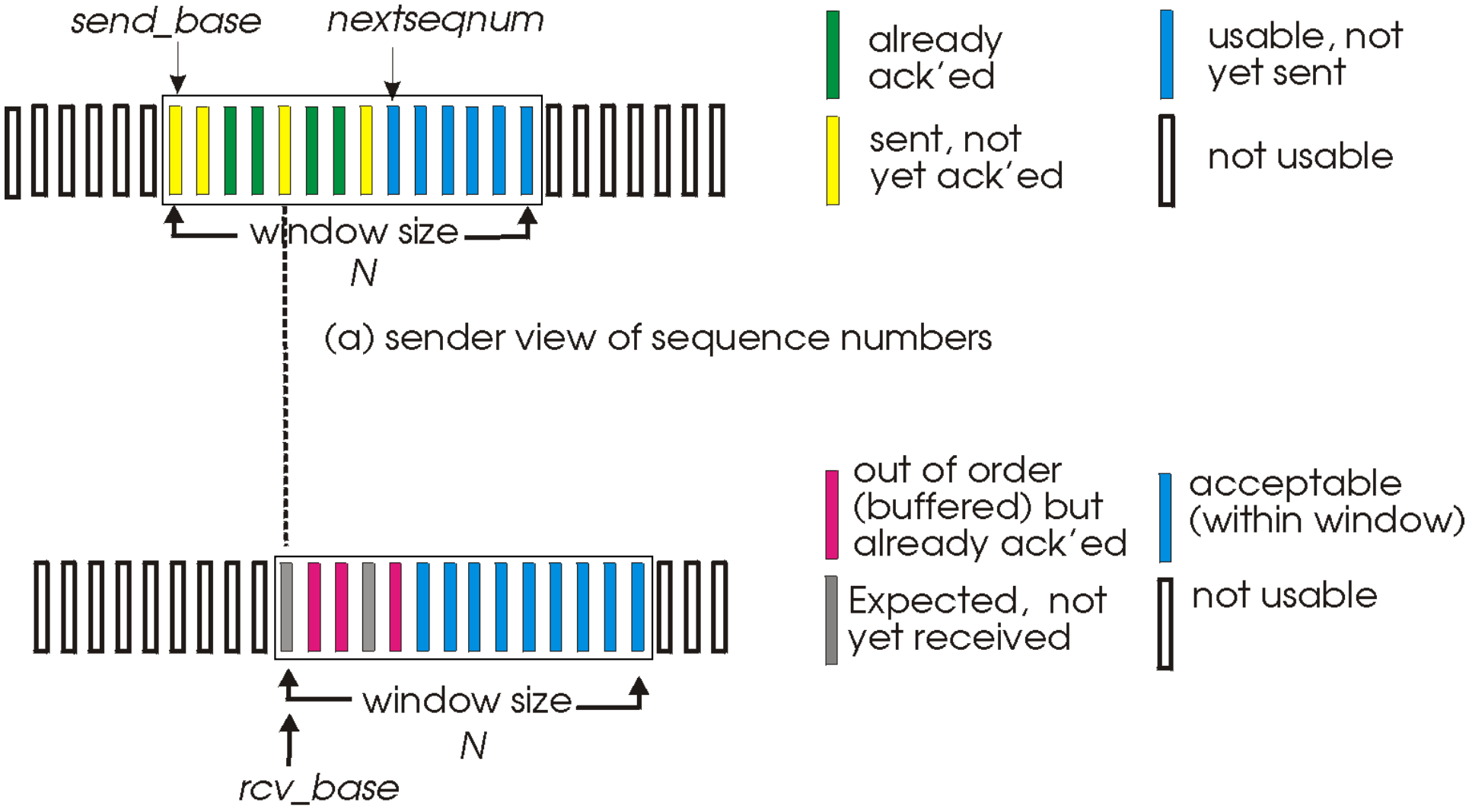
Go-Back-N em Ação



Repetição Seletiva

- Receptor reconhece segmentos recebidos corretamente de forma **individual**.
  - Segmentos recebidos fora de ordem são colocados em *buffer* para posterior entrega à aplicação.
- Transmissor reenvia apenas segmentos para os quais o ACK ainda não foi recebido.
  - Um temporizador para cada segmento em trânsito.
- Janela do transmissor:
  - N números de sequência consecutivos.
  - Limita número de segmentos em trânsito.

Repetição Seletiva: Janelas do Transmissor e do Receptor



Repetição Seletiva: Eventos

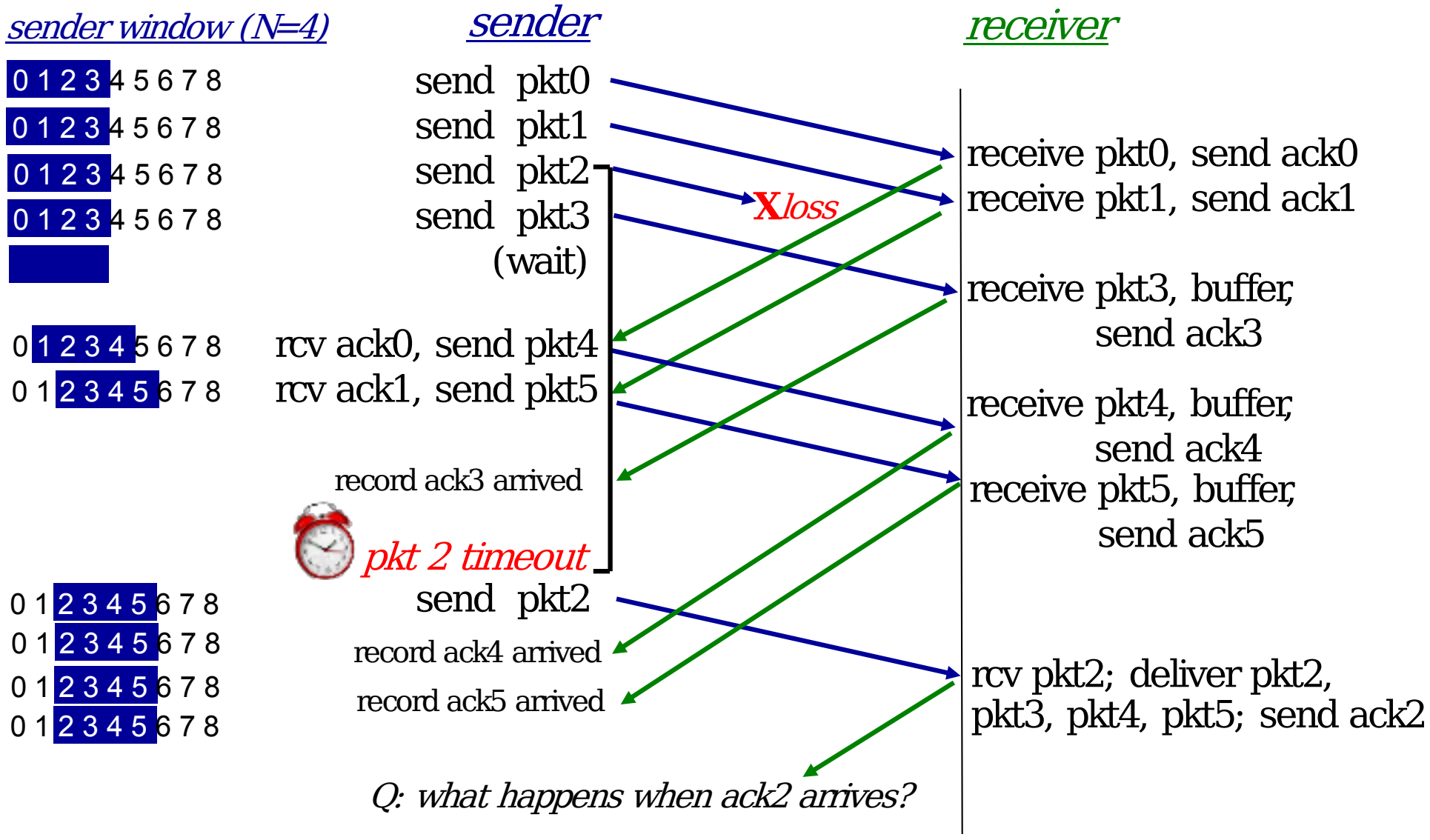
Transmissor

- **Dado da aplicação:**
  - Se há # de sequência disponível na janela, transmita segmento.
- **timeout(n):**
  - Retransmita pacote n, reinicie temporizador.
- **ACK(n):**
  - Marque pacote n como recebido.
  - Se n é o menor # de sequência na janela, avance base da janela para o próximo # de sequência não reconhecido/disponível.

Receptor

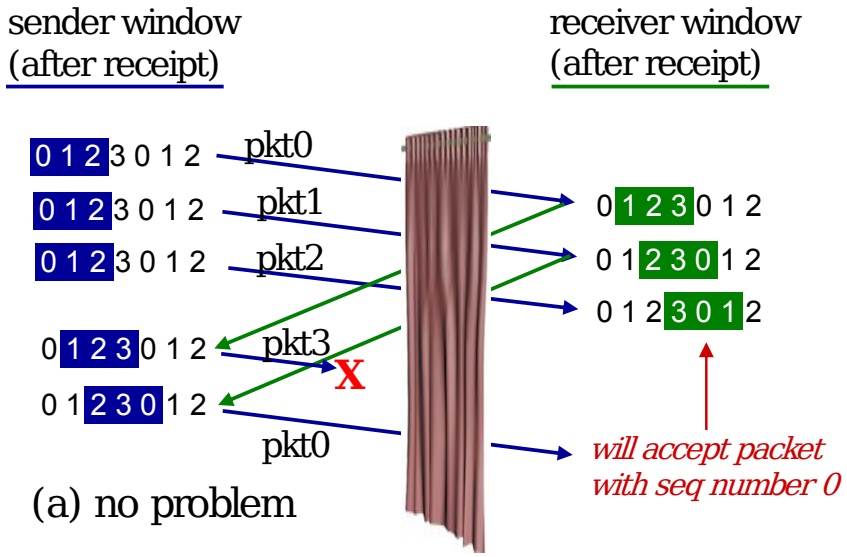
- Segmento n recebido ($rcvbase \leq n < rcvbase + N$).
  - Transmita ACK(n).
  - Se fora de ordem: armazene em *buffer*.
  - Se em ordem, entregue todos os dados contíguos, avance janela para próxima lacuna.
- Segmento n recebido ($rcvbase - N \leq n < rcvbase$).
  - ACK(n).
- Outros:
  - Ignore.

Repetição Seletiva em Ação

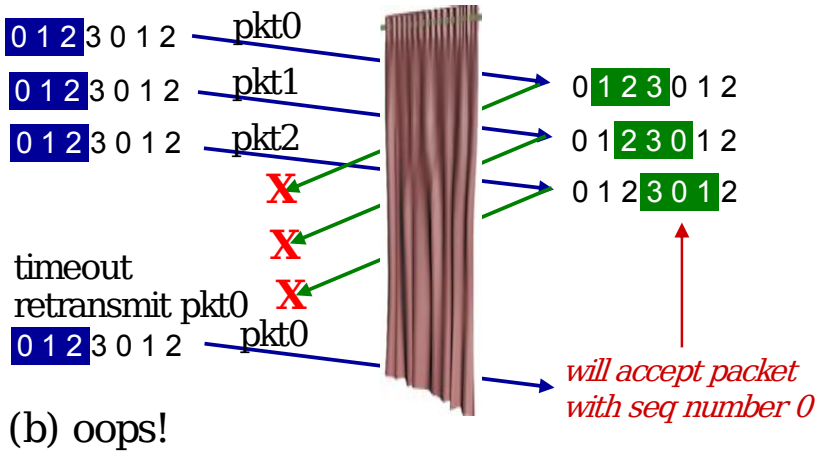


Repetição Seletiva: Dilema

- Exemplo:
  - # de sequência disponíveis: 0, 1, 2, 3.
  - Tamanho da janela: 3.
  - Receptor não vê diferença nos dois cenários!
  - No segundo, dados entregues à aplicação duplicados.
  - **Pergunta:** qual a relação entre o # de sequência e o tamanho da janela para evitar o problema?



receiver can't see sender side.
receiver behavior identical in both cases!
something's (very) wrong!

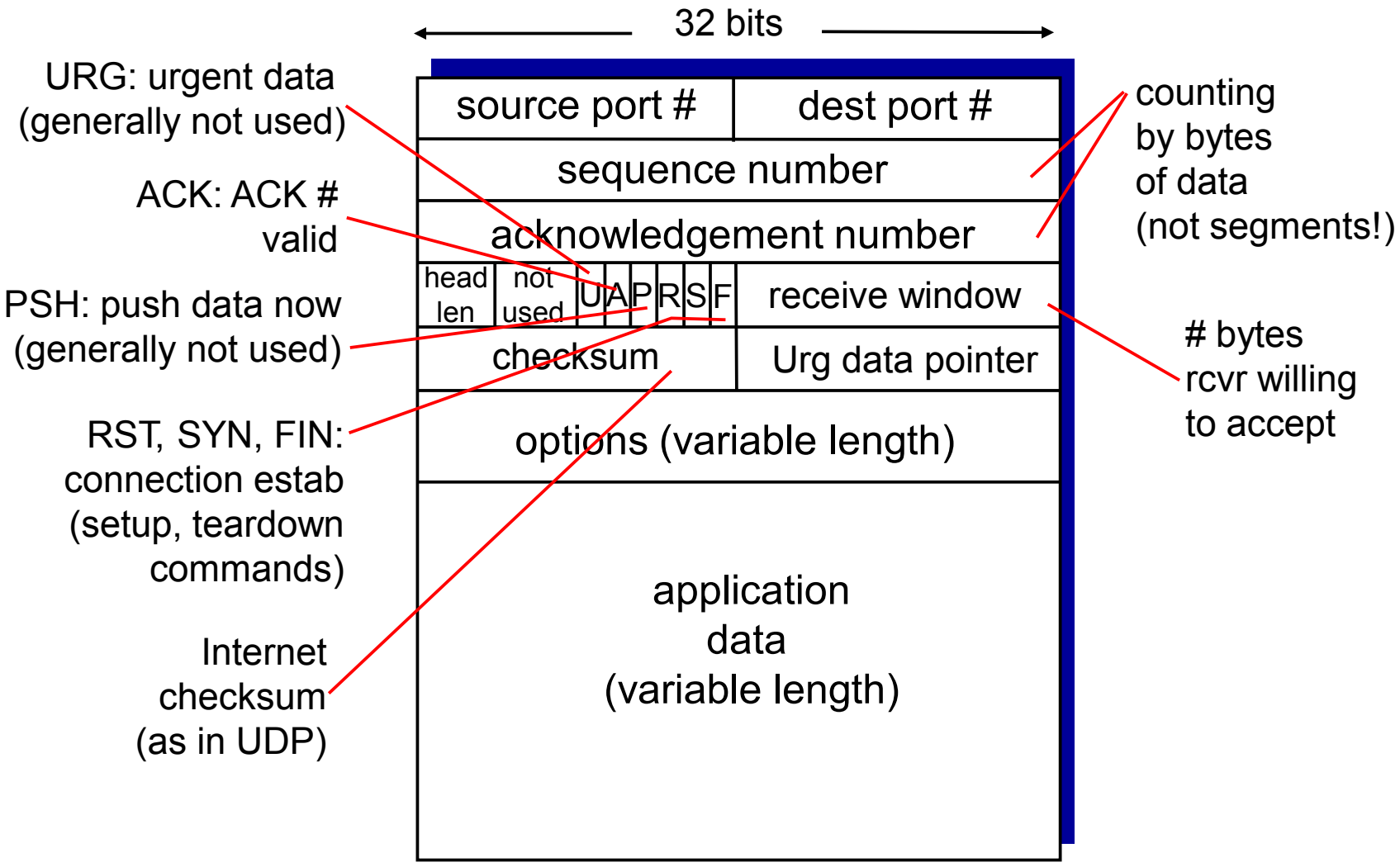


Introdução ao TCP

TCP: Visão Geral [RFCs: 793, 1122, 1323, 2018, 2581]

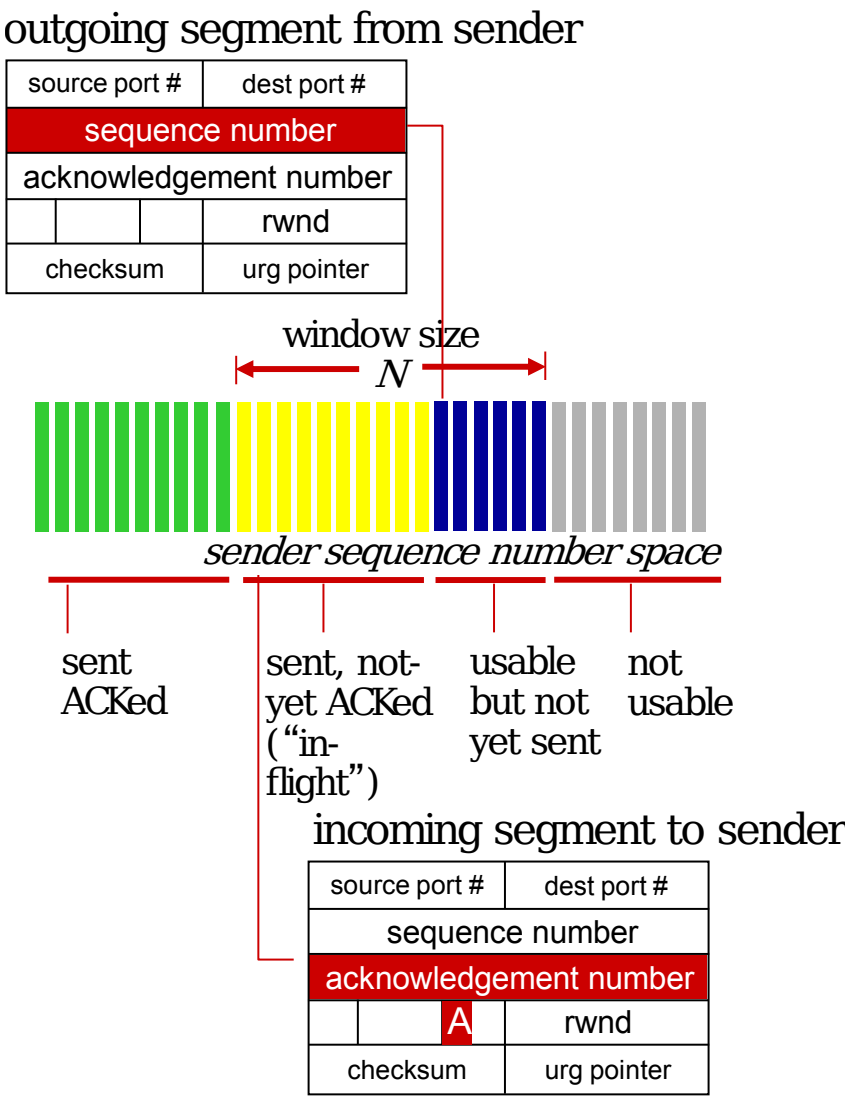
- **Ponto-a-ponto:**
  - Um transmissor, um receptor.
- **Fluxo de bytes confiável, ordenado:**
  - Sem "fronteiras entre mensagens".
- **Baseado em Pipeline:**
  - Controle de fluxo e controle de congestionamento configuram tamanho da janela.
- **Comunicação *full-duplex*:**
  - Dados podem fluir nas duas direções em uma mesma conexão.
  - MSS: *Maximum Segment Size*.
- **Orientado a conexão:**
  - Um *handshake* (troca de mensagens de controle) inicia os estados no transmissor, receptor **antes da troca de dados**.
- **Controle de fluxo:**
  - Transmissor não afogará o receptor.

Estrutura de um Segmento TCP

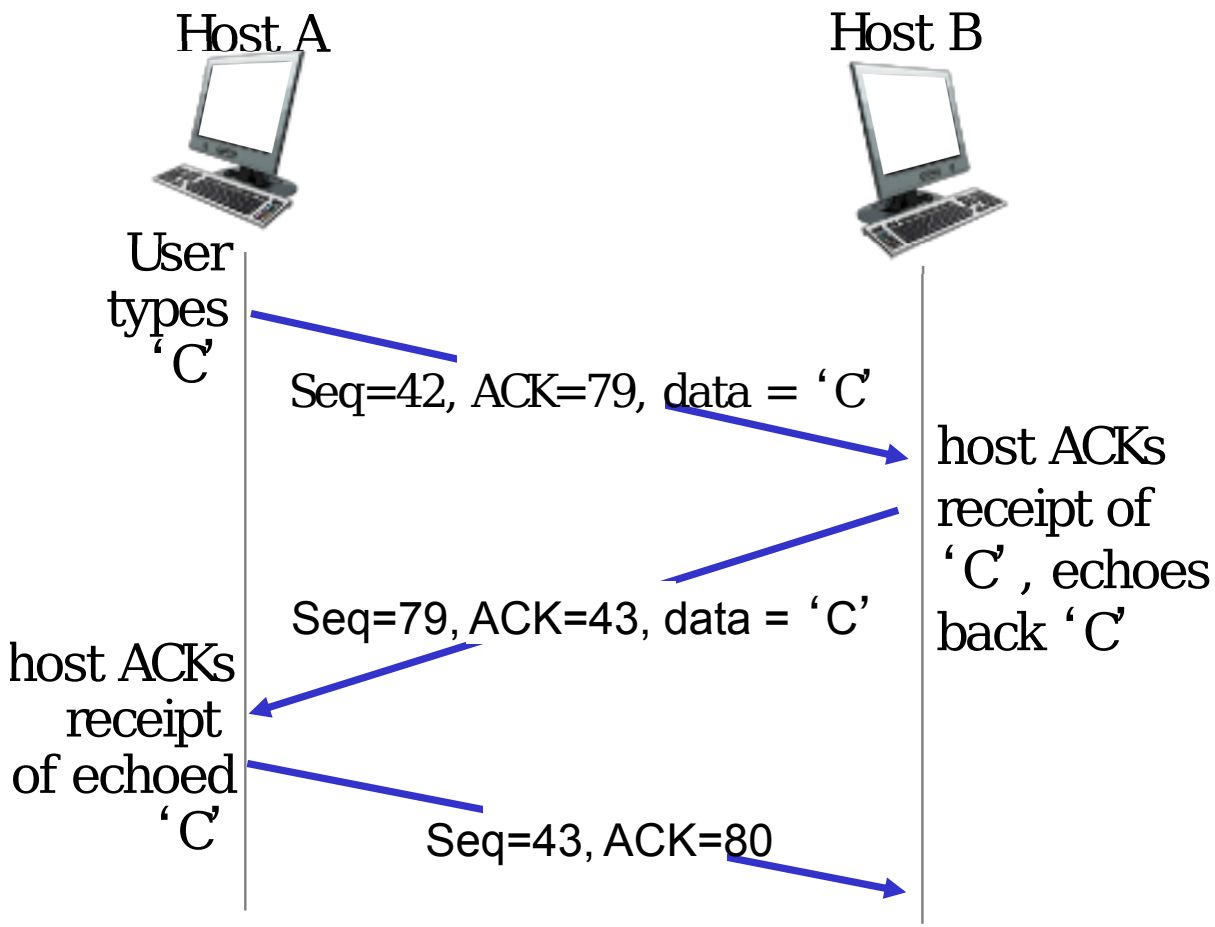


TCP: # de Sequência, ACKs (I)

- **Números de sequência:**
  - "Índice" do primeiro byte do segmento no fluxo de bytes.
- **ACKs:**
  - Número de sequência do próximo byte esperado pelo receptor.
  - ACKs cumulativos.
- **Pergunta:** como o receptor lida com segmentos fora de ordem?
  - Resposta: especificação do TCO não diz – decisão do implementador.



TCP: # de Sequência, ACKs (II)



simple telnet scenario

Resumo da Aula... (I)

- **Pipeline:**
  - Múltiplos segmentos em trânsito.
  - *i.e.*, transmitidos, mas ainda sem ACK.
  - Aumenta utilização.
    - Transmissor passa menos tempo ocioso.
    - Dois exemplos: **Go-Back-N, Repetição Seletiva**.
- **Go-Back-N:**
  - Até N (fixo) segmentos em trânsito.
  - Uso de janela.
  - **ACKs cumulativos**.
  - **Um único temporizador**.
    - Timeout: retransmite todos os segmentos em trânsito.
  - Segmentos fora de ordem descartados.
- **Repetição Seletiva:**
  - Até N (fixo) segmentos em trânsito.
  - Uso de janela.
  - ACKs individuais.
  - Um temporizador por segmento.
    - **Timeout: retransmite apenas segmento correspondente**.
  - Segmentos fora de ordem armazenados em *buffer*.
- **# de seq. e tamanho da janela:**
  - **Janela tem que ser no máximo metade da qtd de # de seq**.

Resumo da Aula... (II)

- **TCP: características.**
  - Ponto-a-ponto (*i.e.*, apenas dois participantes).
  - *Full-duplex*.
  - Pipeline.
  - *Handshake*: abertura de conexão.
  - Controle de fluxo, congestionamento.
- **TCP: segmento**.
  - **Todo segmento de dados é também um ACK**.
- **TCP: # de seq. e ACKs**.
  - **# de seq. conta bytes, não segmentos**.
  - **ACKs cumulativos**.
  - **ACK informa próximo byte esperado**.

Próxima Aula...

- Continuaremos discutindo o TCP:
  - Cálculo do *timeout*.
  - Mecanismos de transferência confiável de dados.
  - Controle de fluxo.
  - Gerenciamento de conexão.
    - *e.g.*, abertura, fechamento.