
UNIVERSIDADE DE
VASSOURAS

UNIVERSIDADE DE VASSOURAS

Curso de Graduação em Engenharia de Software

Aula 10

Redes de Computadores

Prof. André Saraiva
Mestre em Sistemas de Computação
Especialista em Arquitetura e Projeto de Cloud Computing
Analista Sênior Blue Team em Cibersegurança pela Kimoshiro
Tutor EaD pela Universidade Federal Fluminense - UFF

Bibliografia

KUROSE, James F; ROSS, Keith W. **Redes de computadores e a Internet: uma abordagem top-down** . 5. ed. São Paulo (SP): Pearson e Addison Wesley, 2010

UNIVERSIDADE DE VASSOURAS

Tópicos

- Ementa da Disciplina

- Camada de Transporte

TCP: Transmissão Confiável de Dados

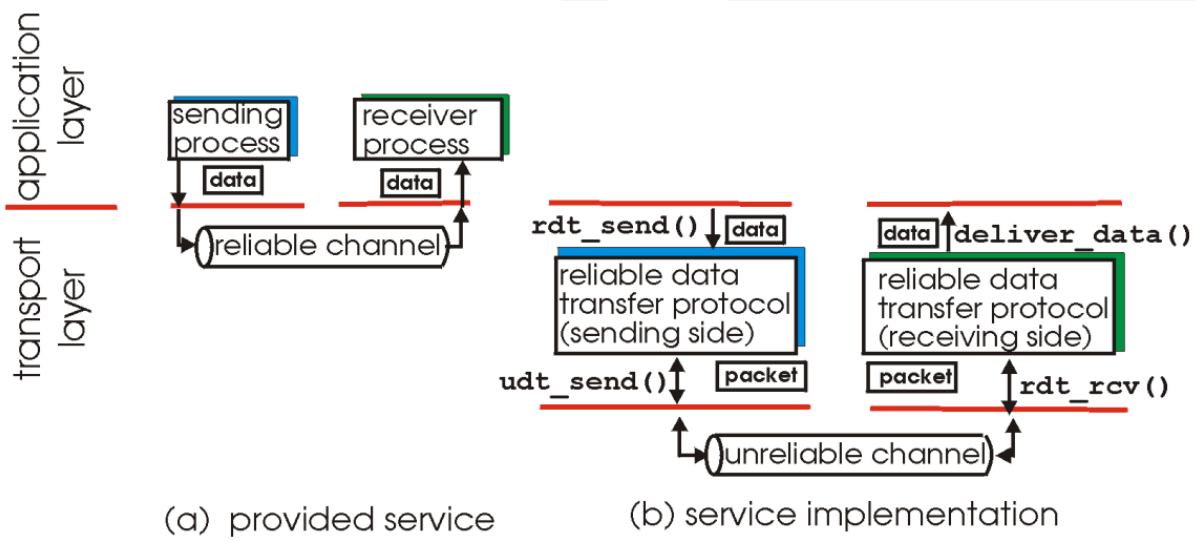
- TCP cria serviço de transmissão confiável sobre serviço não confiável do IP.
  - Utiliza pipeline.
  - ACKs cumulativos.
  - Único temporizador de retransmissão.

- Retransmissões disparadas por:
  - Eventos de *timeout*.
  - ACKs duplicados.

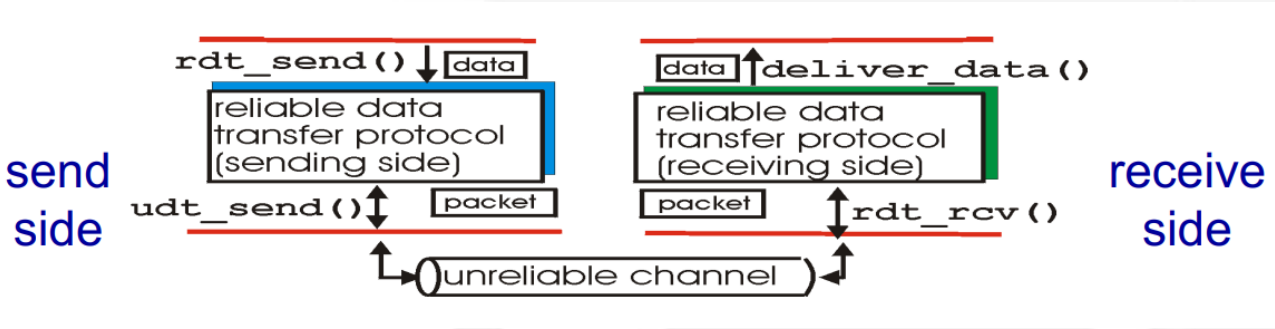
- Vamos considerar inicialmente um transmissor TCP simplificado.
  - Ignorando ACKs duplicados, controle de fluxo e o controle de congestionamento

Princípio de Transferência Confiável de Dados

- Importante nas camadas de aplicação, transporte e enlace.
- Características do canal não-confiável determinarão complexidade do protocolo de **rdt**.
- **rdt**, do inglês reliable data transfer (transferência de dados confiável).



Princípio de Transferência Confiável de Dados

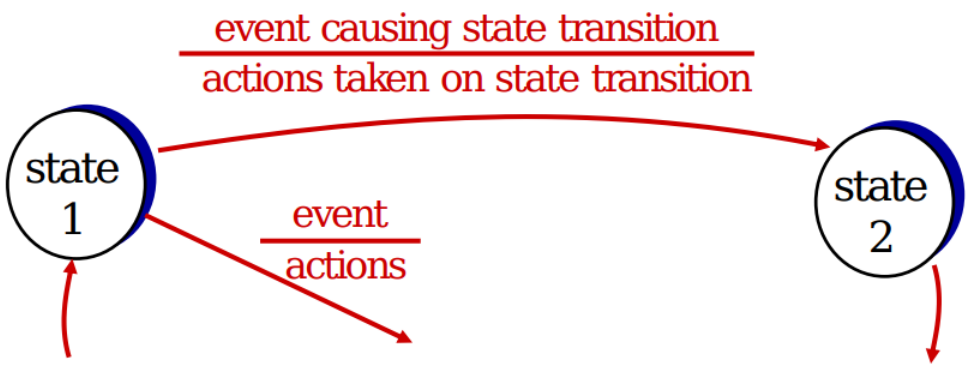


- **rdt_send():** chamada pela aplicação para enviar dados para o transporte.
- **udt_send():** chamado pelo transporte para passa pacote para a rede.
- **rdt_rcv():** chamada quando pacote chega pela rede no lado receptor.
- **deliver_data():** chamado pelo transporte para entregar dados para aplicação.

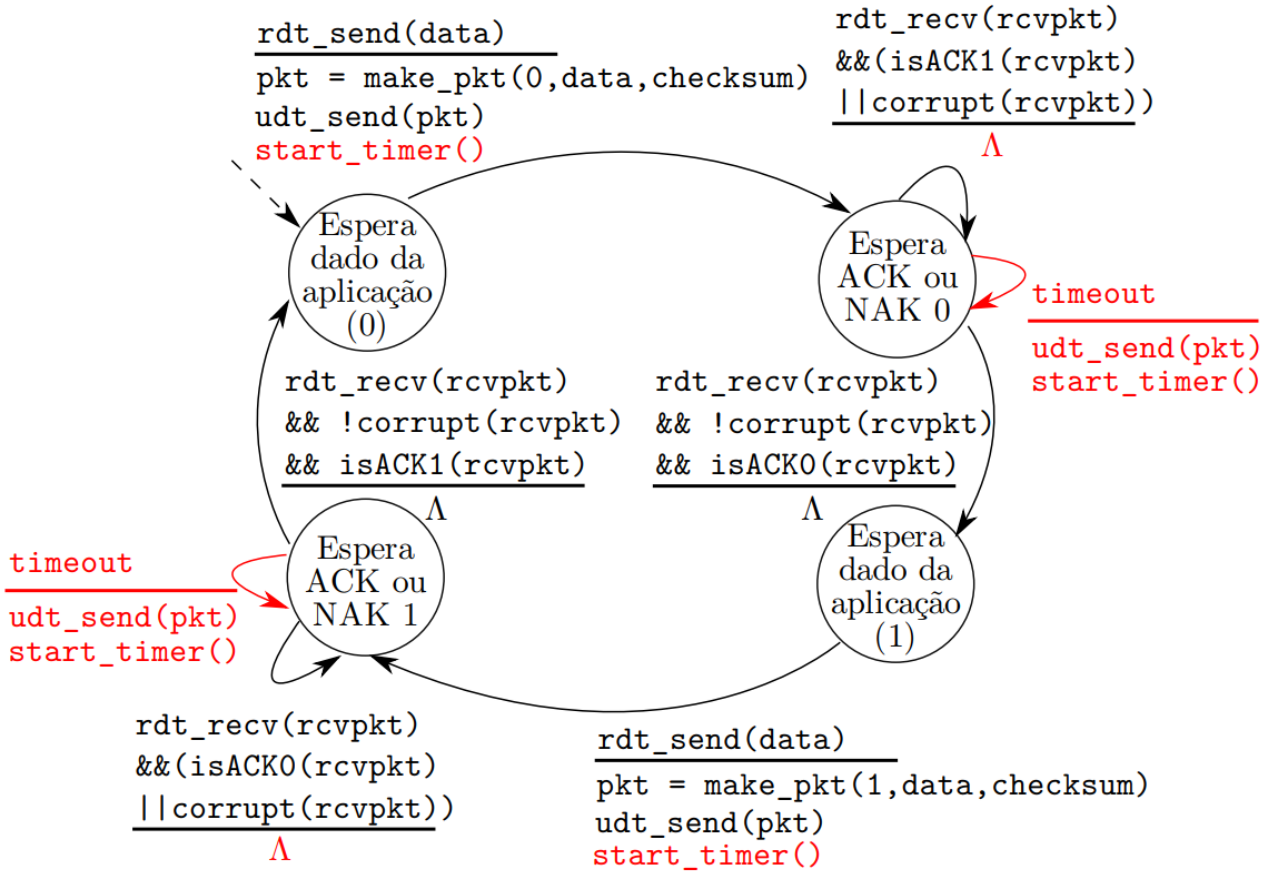
Transferência Confiável de Dados

- Nós iremos incrementalmente desenvolver os lados transmissor e receptor de um protocolo rdt.
- Consideraremos apenas transmissão unidirecional de dados.
- Mas informação de controle trafegará nos dois sentidos!
- **Usar máquinas de estado para especificar transmissor, receptor.**

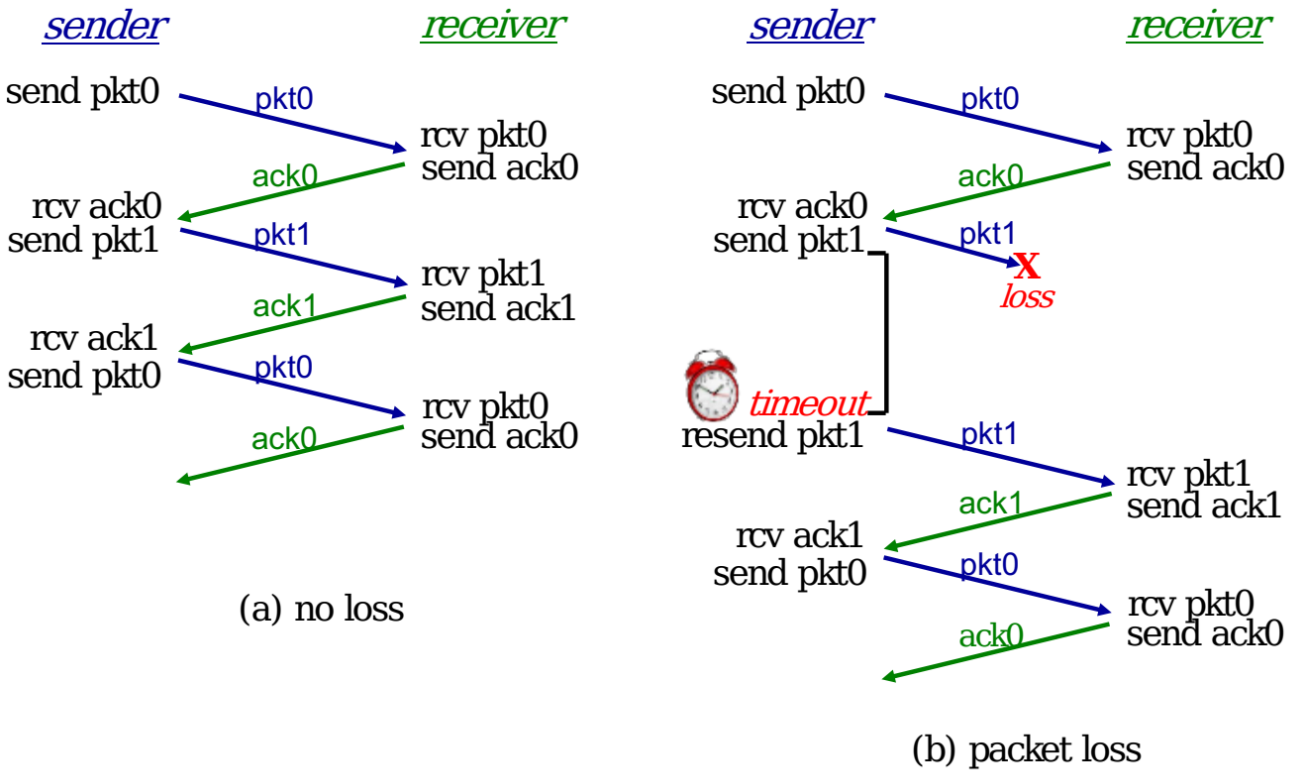
Estado: o "estado" muda para o próximo estado exclusivamente por um próximo evento.



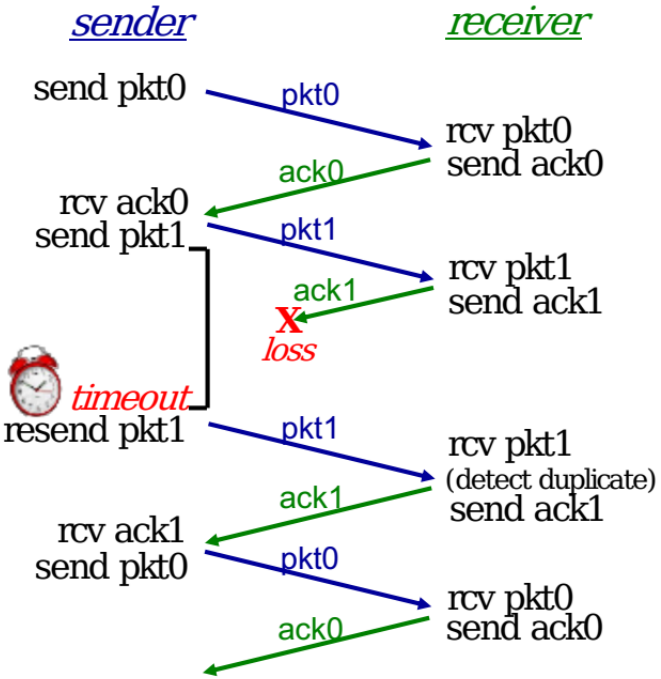
rdt3.0 – Canal com erros de Bit



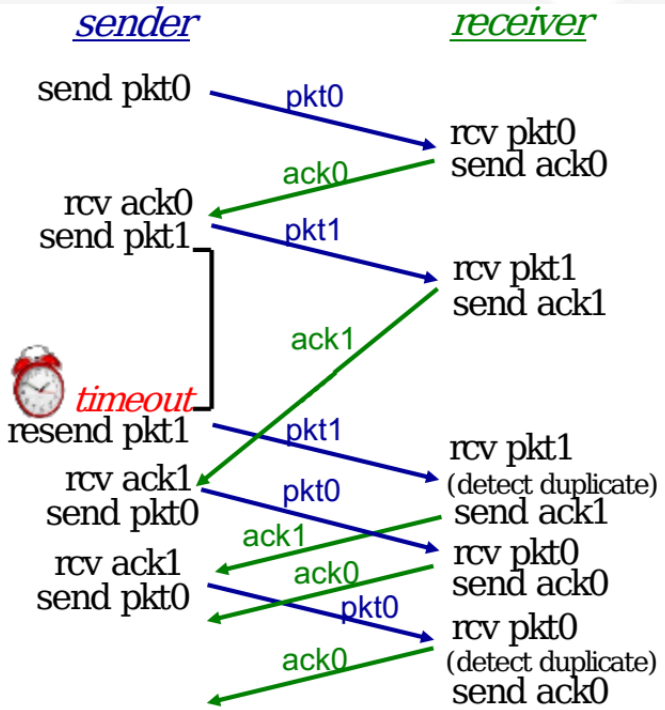
rdt3.0 – Canal com erros de Bit



rdt3.0 – Canal com erros de Bit

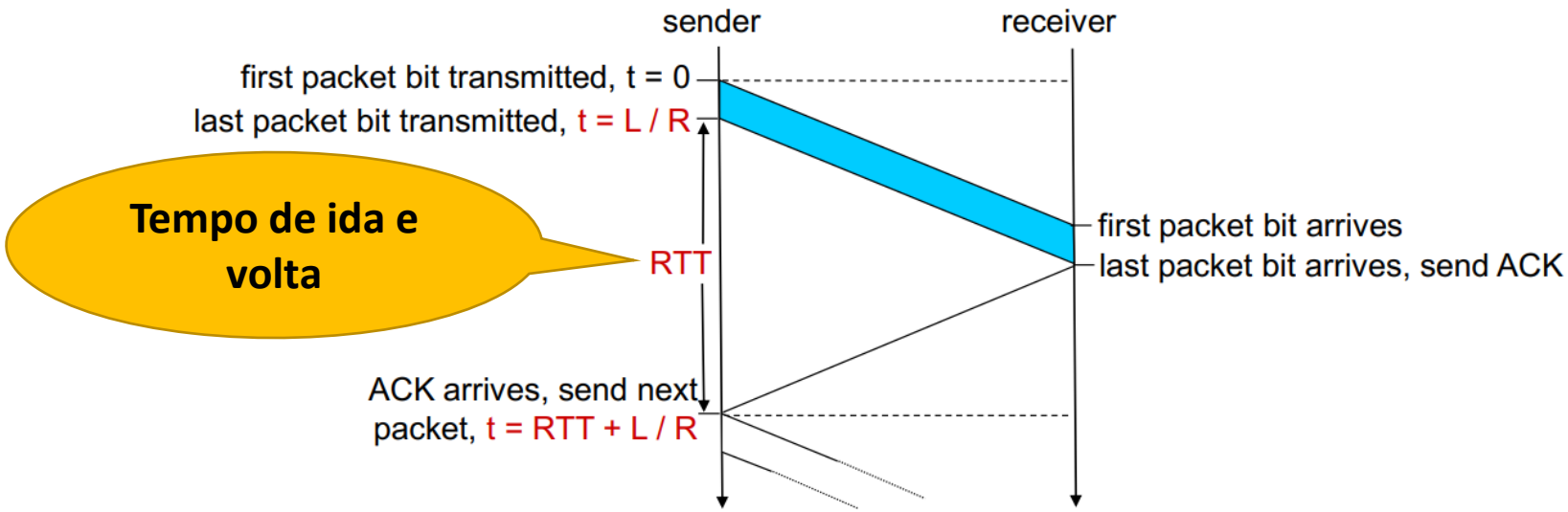


(c) ACK loss



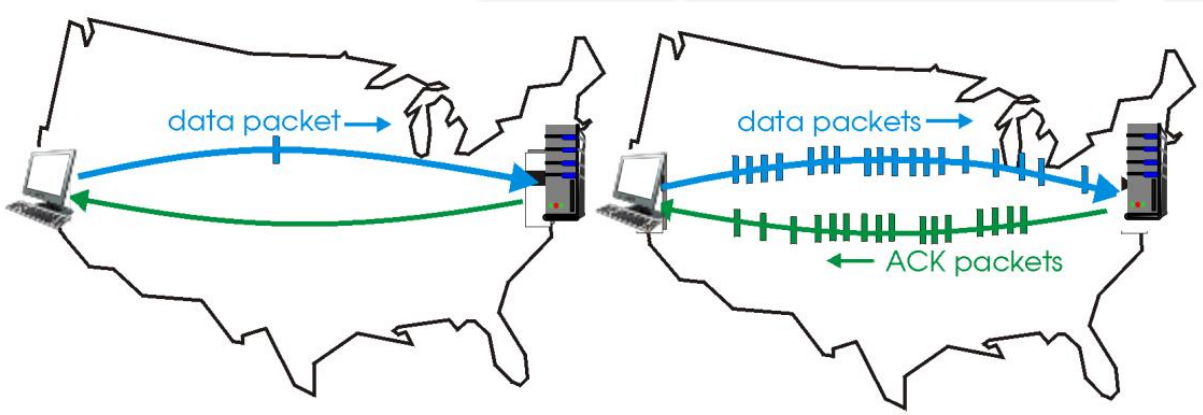
(d) premature timeout/ delayed ACK

Operação do Tipo Stop-and-Wait



Protocolos baseados em Pipeline

- **Pipeline:** permite que transmissor tenha múltiplos segmentos **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

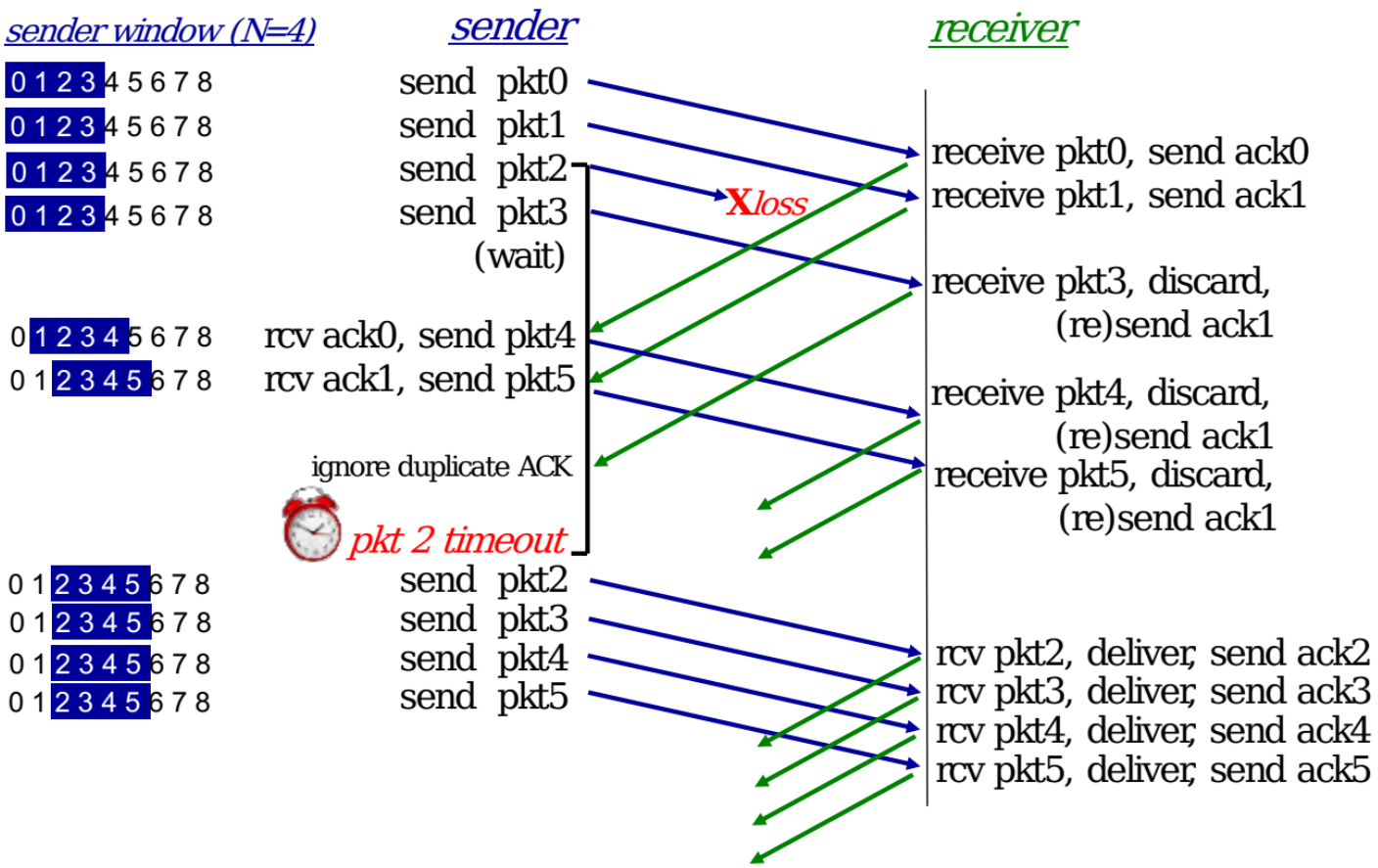
Protocolos baseados em Pipeline

- **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.

Protocolos baseados em Pipeline



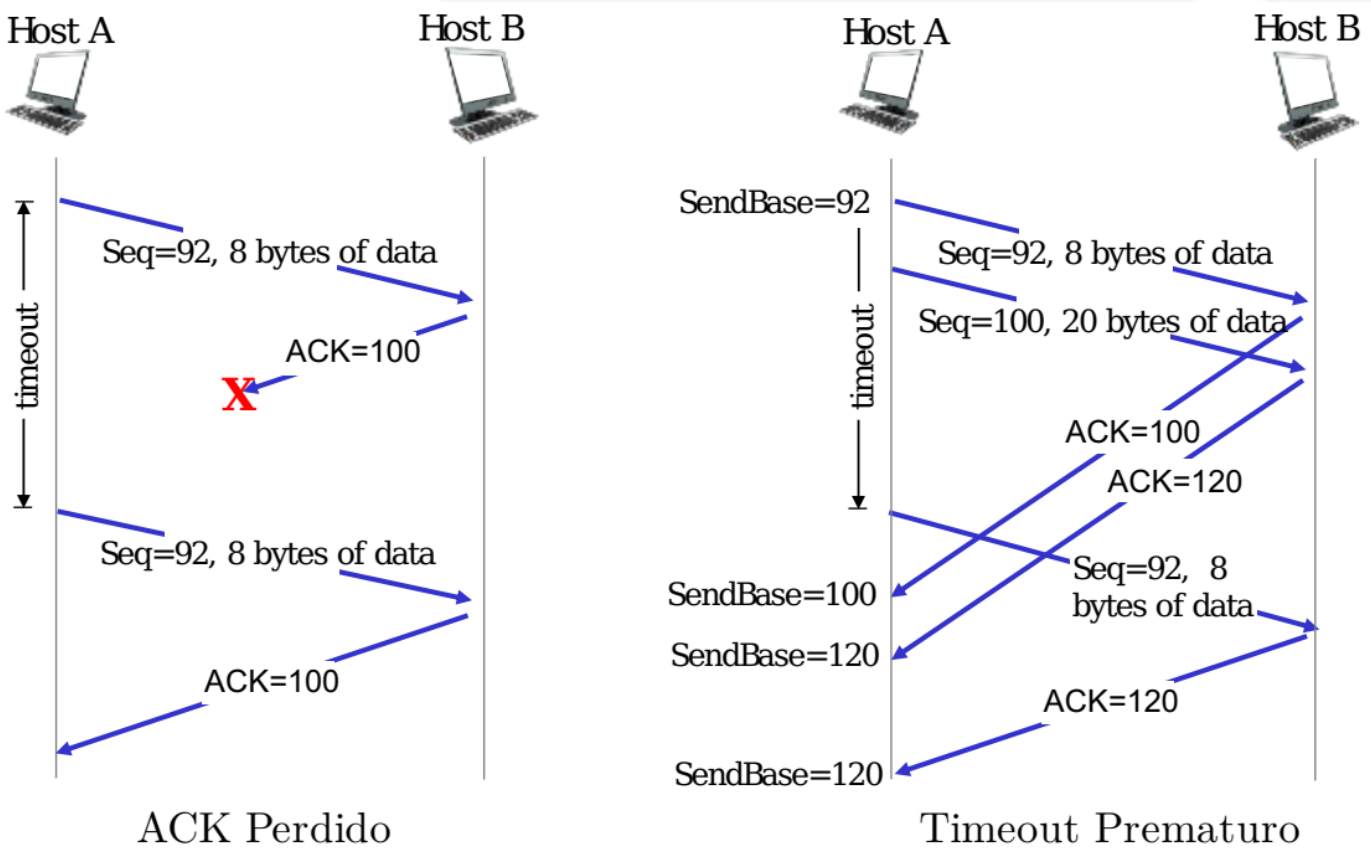
TCP: Eventos no transmissor

- **Dados recebidos da aplicação:**
  - Cria segmento com # de sequência.
  - # de sequência é **índice da posição do primeiro byte no fluxo de dados da aplicação**.
  - Inicia o temporizador, se ele já não estiver rodando.
    - Intervalo de expiração: ***TimeOutInterval***.

- **Estouro de temporizador:**
  - Retransmita segmento que causou o estouro.
  - Reinicie o temporizador.

- **Recepção de ACK:** Se ACK reconhece segmentos ainda pendentes:
  - Atualize os números de sequência já reconhecidos.
  - **Reinicie o temporizador se ainda há segmentos não reconhecidos**.

TCP: Eventos no transmissor - Retransmissão



TCP: Controle de Congestionamento

- **Congestionamento:**
  - Informalmente: “fontes demais gerando tráfego demais para a **rede**”.

- **Diferente do controle de fluxo**.
  - Pacotes perdidos (*overflow* de *buffers* nos roteadores).
  - Longos atrasos (enfileiramento nos *buffers* dos roteadores).

TCP: Controle de Congestionamento

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

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

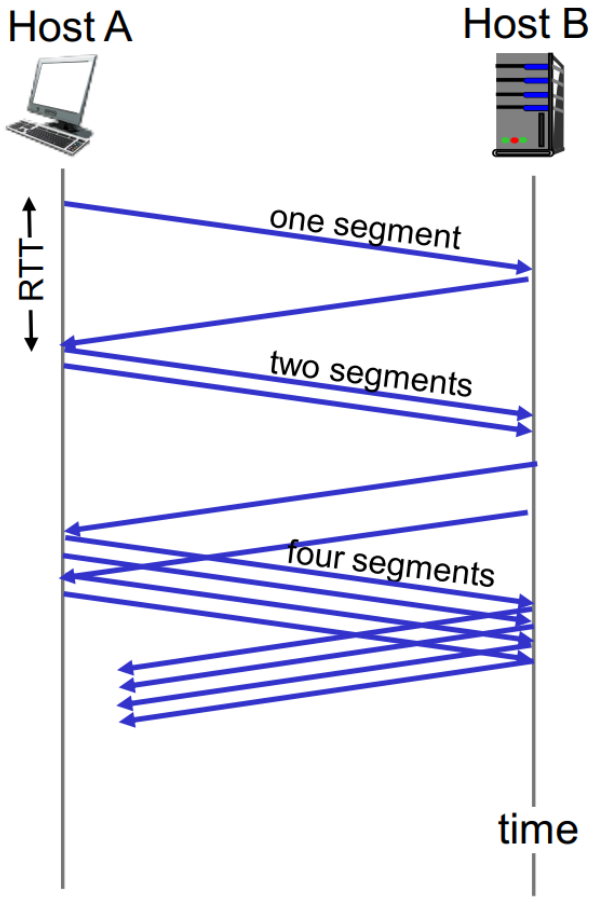
TCP: Abordagem para Congestionamento

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

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

TCP: Abordagem Slow Start

- Quando a conexão começa, aumentando a taxa exponencialmente
- Dobra a cada RTT.
- Equivalente a aumentar em 1 MSS (maximum segment size) a cada ACK recebido.
- **Resumo:** taxa inicial é baixa, **mas aumenta rapidamente**.



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 $\frac{3}{4} \cdot \frac{W}{RTT}$

TCP: Prevenção de Congestionamento

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

Pesquisa para entregar no AVA

- **TCP Reno**
  - **O que é?**
  - **Qual é a premissa do TCP Reno?**
  - **Além do slow start e congestion avoidance, qual outra abordagem para congestionamento é usada?**
  - **Inserir gráficos e mapa de estados do TCP Reno**

Vale pontos extras!!!

Contato

Professor:
André Saraiva, MSc

E-mail:
andre.saraiva@univassouras.edu.br