# Arquitetura TCP/IP
# Nível de Transporte (TCP & UDP)

Prof. Helber Silva

# Roteiro

- Introdução
- Protocolo TCP
- Protocolo UDP
- Conclusão

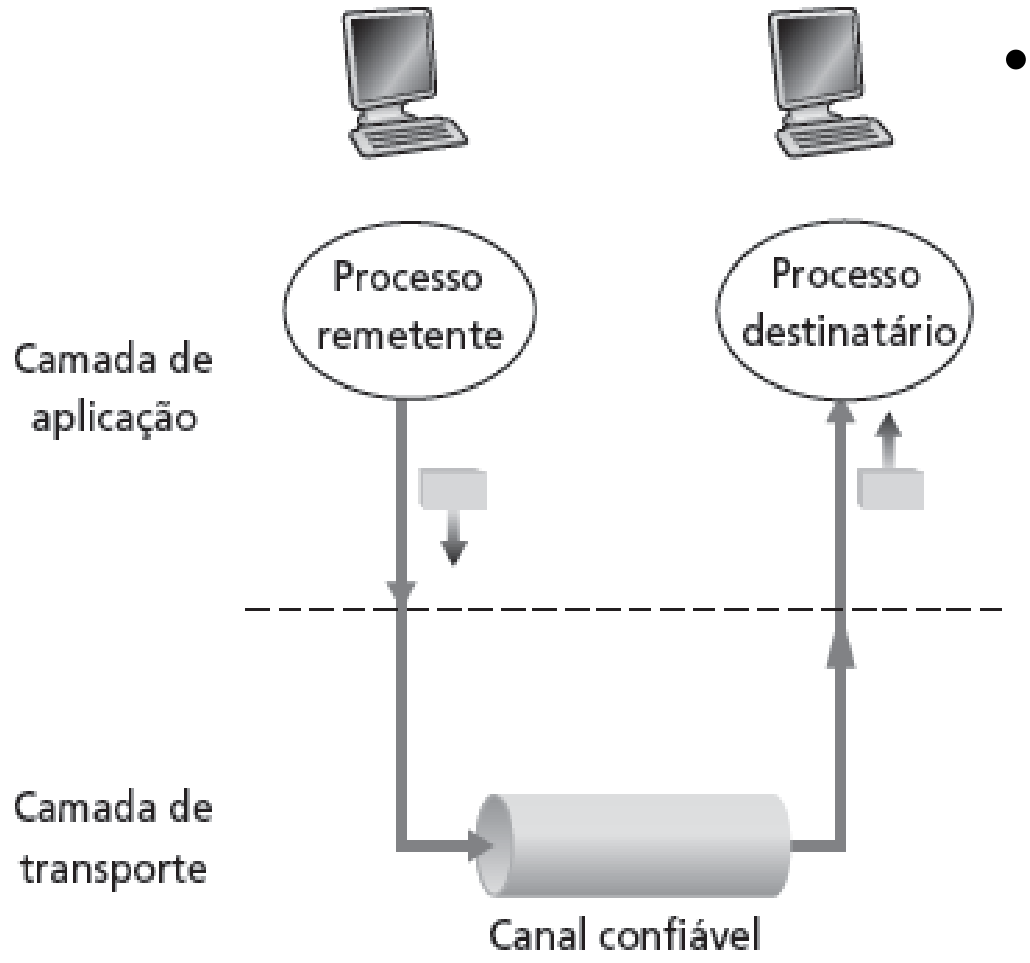
# Introdução

- Na Arquitetura Internet (ou Arquitetura TCP/IP), o Nível de Transporte pode fornecer dois tipos de **serviço de entrega de dados**
  - Orientado à conexão
  - **Não**-orientado à conexão
- Serviço orientado à conexão é prestado pelo protocolo TCP (***T**ransmission **C**ontrol **P**rotocol*), especificado na RFC 793
- Serviço **não**-orientado à conexão é prestado pelo protocolo UDP (***U**ser **D**atagram **P**rotocol*), especificado na RFC 768

# Serviço Orientado à Conexão (TCP)

- Protocolo TCP
  - Garante a **entrega de dados, na sequência correta, sem erros e sem duplicidade** ao Nível de Aplicação
    - O Nível de Rede *não* fornece essas garantias
  - Provê os serviços de **controle de fluxo** e **de congestionamento** fim a fim
  - A PDU do Nível de Transporte relacionada ao TCP é chamada de *segmento*
    - Um segmento encapsula uma mensagem do Nível de Aplicação

# Serviço Orientado à Conexão (TCP)

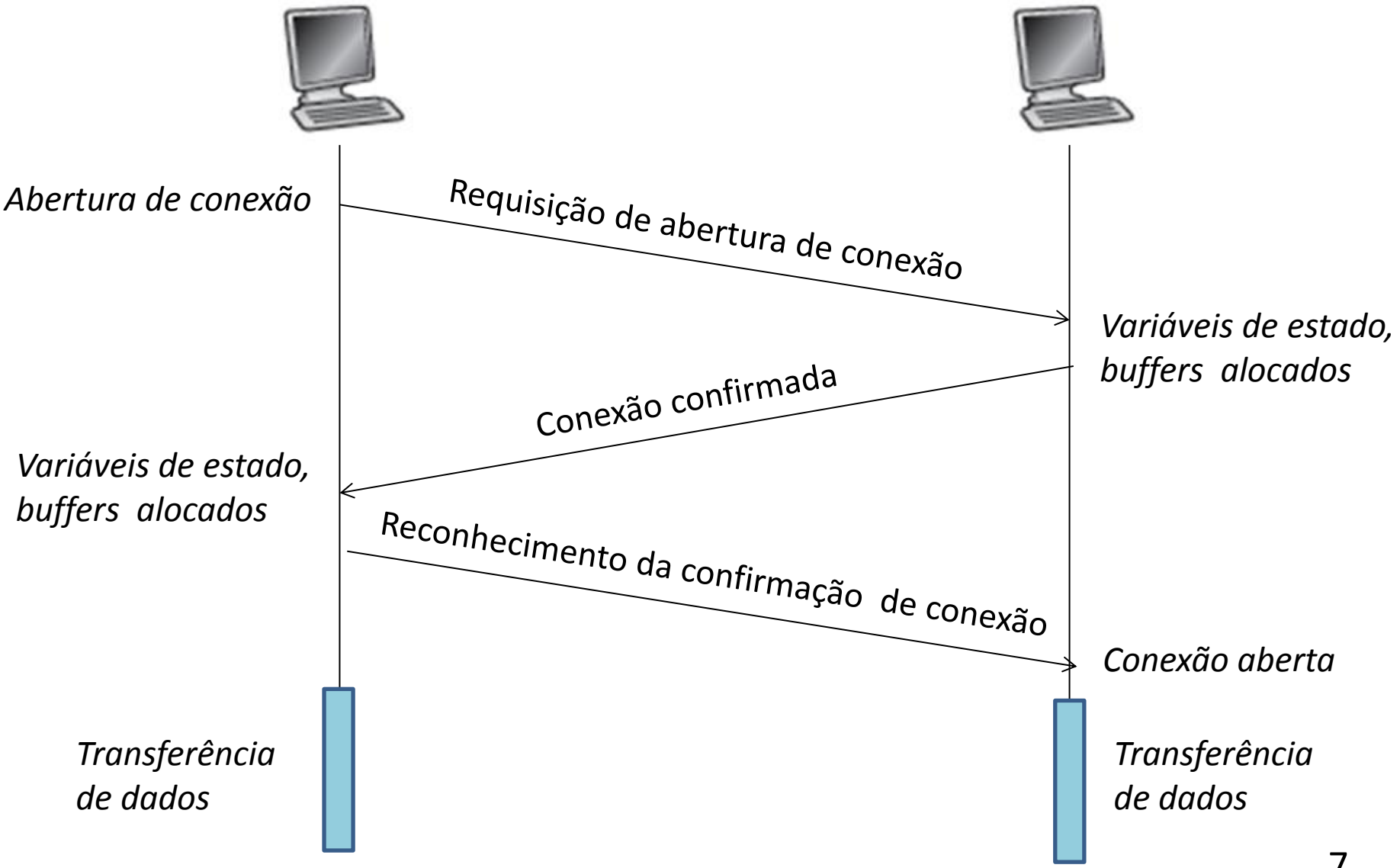



- Uma **conexão TCP** conecta dois processos do Nível de Aplicação que "se apresentam" (e inicializam **variáveis de estado** da conexão TCP) *antes* que esses processos comecem uma sessão de dados entre si

# Abertura de Conexão TCP

- Uma conexão TCP é “aberta” entre dois processos do Nível de Aplicação usando um procedimento conhecido como **aperto de mão de três vias** (em inglês, *three-way handshake*)
  - Remetente e destinatário TCP alocam variáveis de estado e *buffers* (armazenamento) necessários
- Os segmentos usados durante esse procedimento *não contêm dados* do Nível de Aplicação

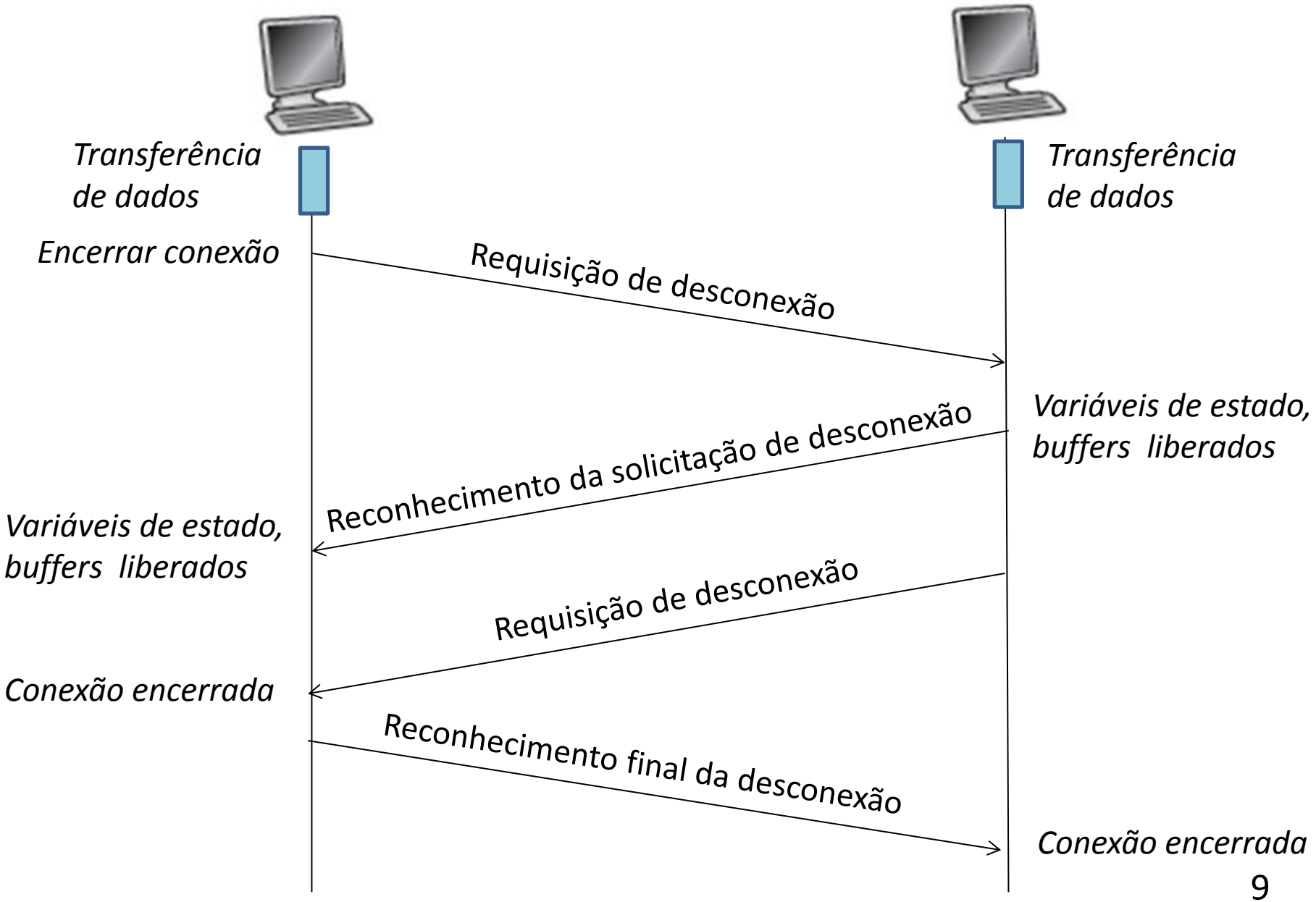
# *Three-Way Handshake*



# Encerramento de Conexão TCP

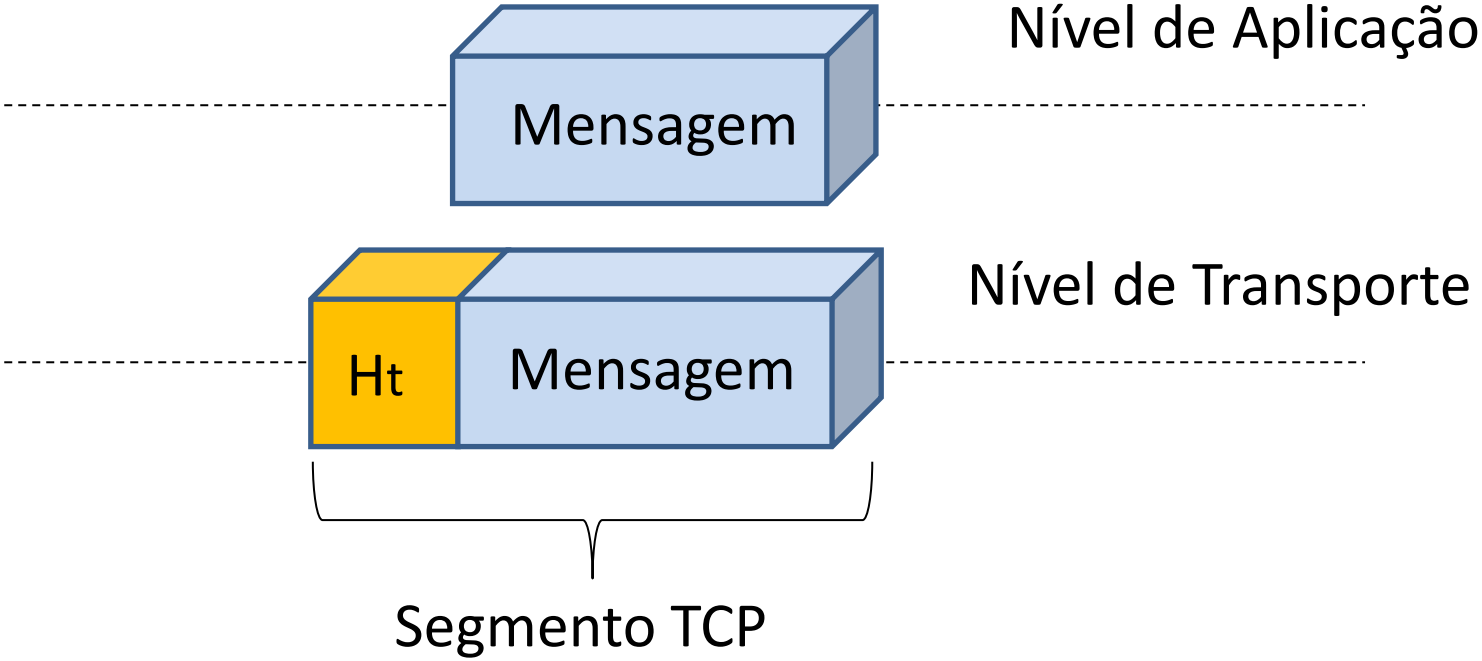
- Após a troca de segmentos que carregam dados (ou seja, mensagens) do Nível de Aplicação, há o encerramento da conexão TCP
  - Remetente e destinatário TCP encerram variáveis de estado e liberam *buffers*
- Esse procedimento é definido por uma sequência de segmentos específicos trocados entre os processos

# Encerramento de Conexão TCP

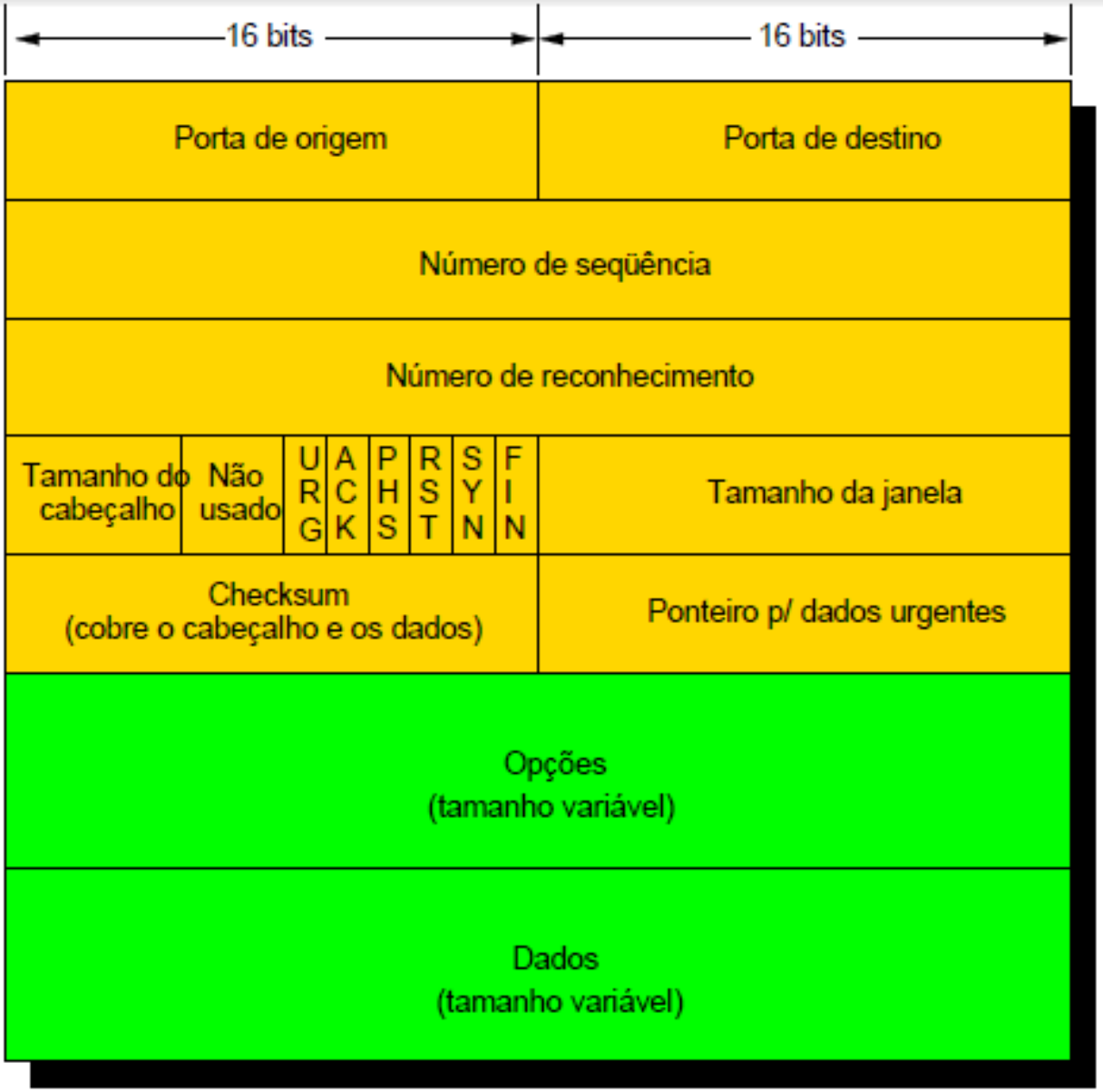


# Segmento TCP

- Encapsula bytes de dados de uma mensagem enviada por um protocolo de Nível de Aplicação
- Inclui ainda os bytes dos cabeçalhos ($H_t$) do protocolo TCP



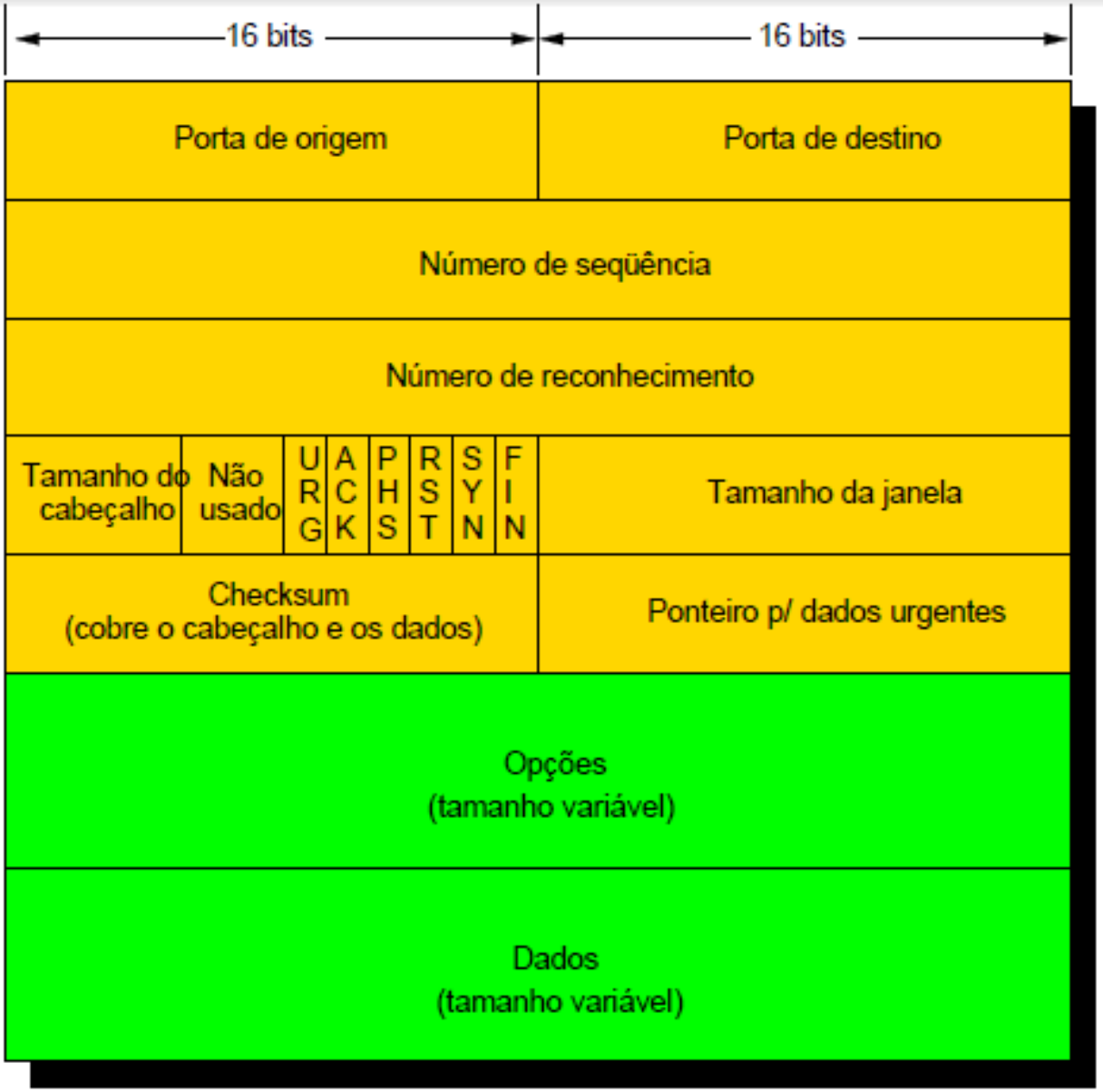
# Formato do Segmento TCP



# Formato do Segmento TCP

- A `Porta de origem` (16 bits) é aquela utilizada pelo hospedeiro transmissor (origem) para o envio do segmento
- A `Porta de destino` (16 bits) é aquela pela qual o hospedeiro receptor (destino) aguarda o segmento
- Os `Números de sequência` e `de reconhecimento` (32 bits) são usados para implementar um serviço de entrega confiável
- O `Tamanho da janela` (16 bits) é um campo utilizado no serviço de controle de fluxo
- O `Tamanho do cabeçalho` (4 bits) é tipicamente de 20 Bytes
  - Em geral, o `cabeçalho` não possui opções

# Formato do Segmento TCP



# Formato do Segmento TCP

- As `Opções` são usadas para mecanismos, como negociação do MSS *(**M**aximum **S**egment **S**ize)* e colocação de marcas de tempo (*timestamps*)
  - MSS representa a máxima quantidade de dados da mensagem do Nível de Aplicação encapsulada no segmento
- Bits `RST`, `SYN` e `FIN`: usados para estabelecer e encerrar conexões TCP
- Bit `ACK`: se `1`, indica que o `Número de reconhecimento` é válido. Se `0`, o segmento não contém uma confirmação e o `Número de reconhecimento` é ignorado

# Formato do Segmento TCP

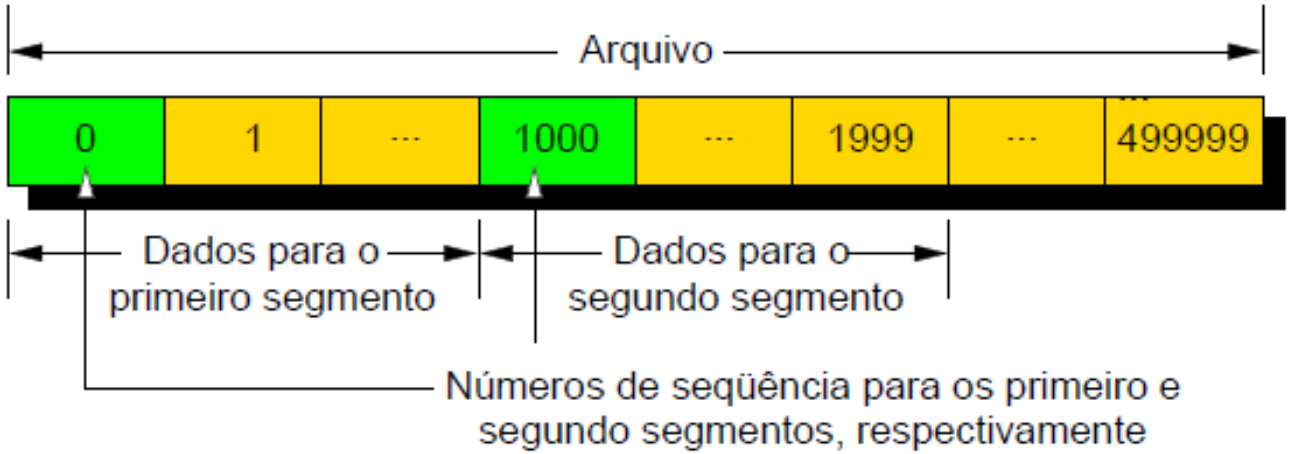
- Bit `PSH`: indica que os dados do segmento devem ser passados para o nível superior imediatamente
- Bit `URG`: indica a presença de dados urgentes, na posição indicada pelo `Ponteiro para dados urgentes`

# Exercício [Kurose, pp. 215, Ex. 3]

- Considere uma conexão TCP entre o hospedeiro A e o hospedeiro B. Suponha que os segmentos TCP que trafegam do hospedeiro A para o hospedeiro B tenham número de porta da fonte $x$ e número de porta do destino $y$. Quais são os números de porta da fonte e do destino para os segmentos que trafegam do hospedeiro B para o hospedeiro A?

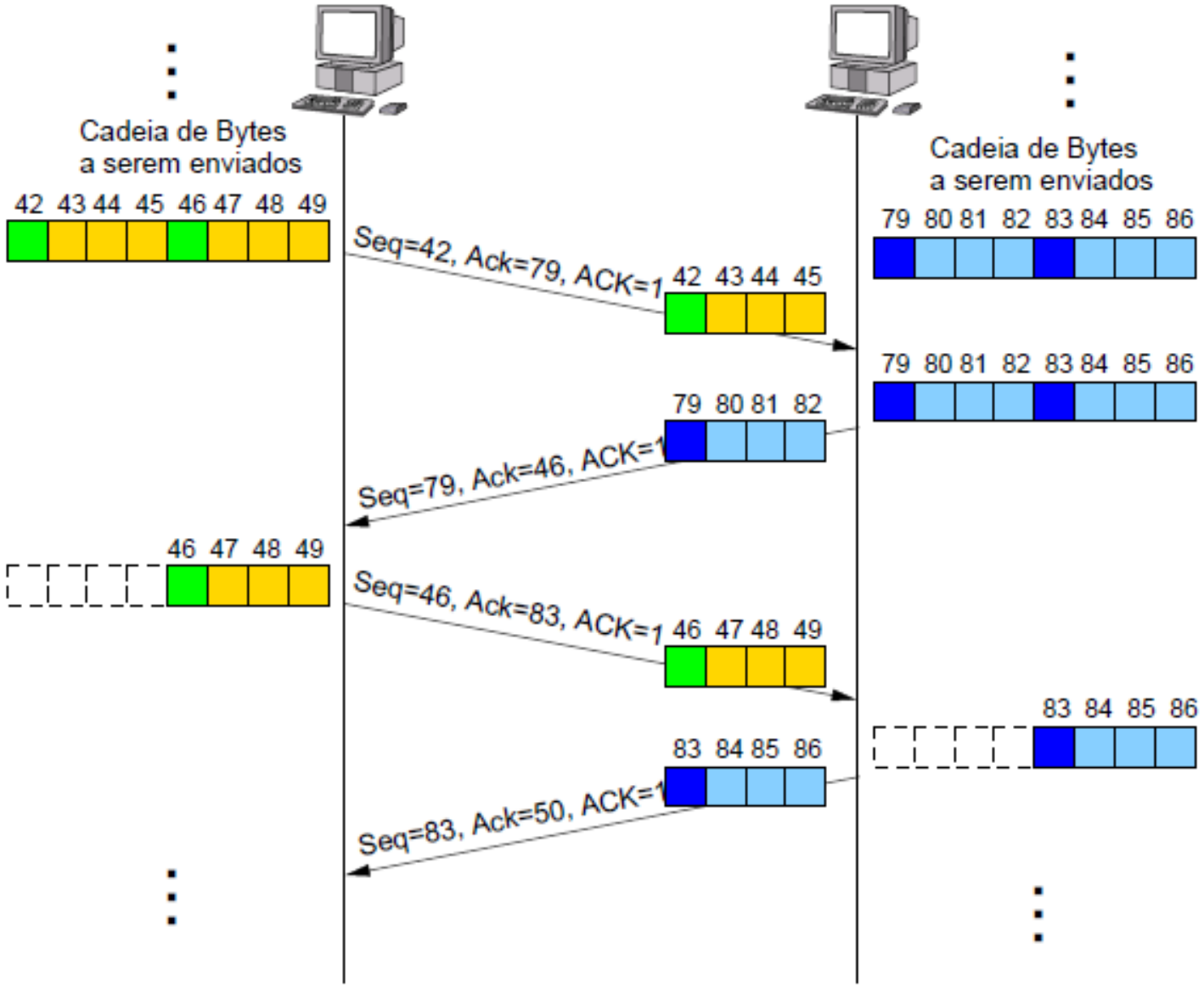
# Números de Sequência e de Reconhecimento

- Usados no serviço de entrega confiável
- Número de sequência é aplicado sobre a *cadeia de bytes* transmitidos (e *não* sobre a série de segmentos)




- Número de reconhecimento contém o número de sequência do *próximo byte* que o hospedeiro (*host*) está esperando receber

# Exemplo de Uso de Números de Sequência



# Exercício [Kurose, pp. 216, Ex. 14d]

- Verdadeiro ou Falso:

d. Imagine que o hospedeiro A esteja enviando ao hospedeiro B um arquivo grande por uma conexão TCP. Se o número de sequência para um segmento dessa conexão for *m*, então o número de sequência para o segmento subsequente será necessariamente *m+1*

# *Piggyback Acknowledgement*

- Pôde-se observar, no exemplo anterior, que o reconhecimento para os dados contidos em um segmento pode ser levado em *outro* segmento contendo dados
  - O reconhecimento “pegou uma carona” no outro segmento de dados
- Essa “carona” chama-se *piggyback acknowledgement*

# Exercício [Kurose, pp. 216, Ex. 14a]

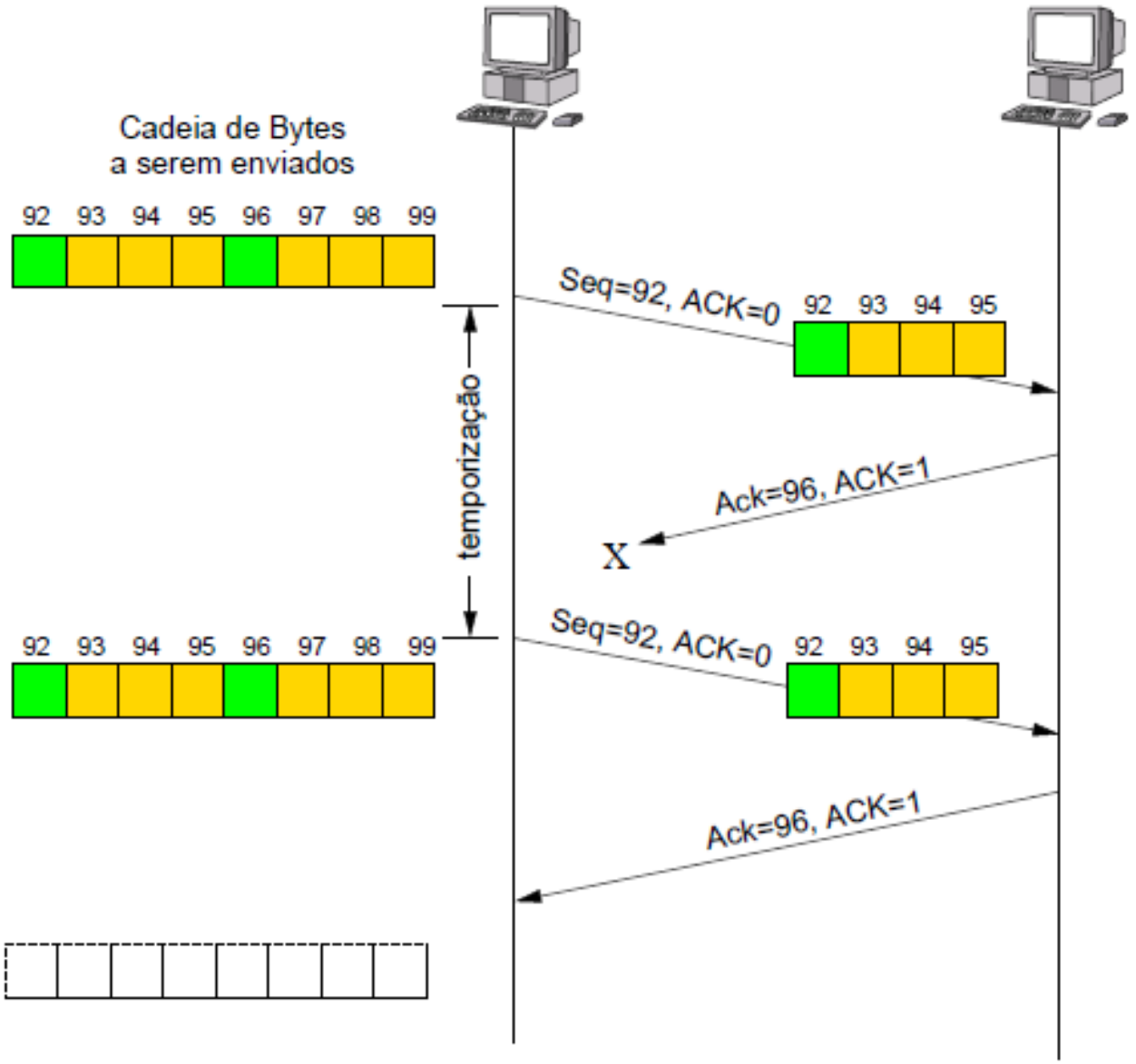
- Verdadeiro ou Falso:

a. O hospedeiro A está enviando ao hospedeiro um arquivo grande por uma conexão TCP. Suponha que o hospedeiro B não tenha dados para enviar para o hospedeiro A. O hospedeiro B não enviará reconhecimentos para o hospedeiro A porque ele não pode dar carona aos reconhecimentos dos dados.

# Temporizador de Retransmissão

- TCP no remetente associa um **temporizador** para cada segmento transmitido
- Quando a temporização expira (devido a congestionamento nos enlaces da rede, por exemplo), e o remetente TCP não obtém reconhecimento do destinatário, o segmento é retransmitido
- Esse temporizador é chamado de **temporizador de retransmissão**
  - Tipicamente, medido em *ms* (milissegundos)
  - 1 ms = 0,001s (1 milésimo de segundo)

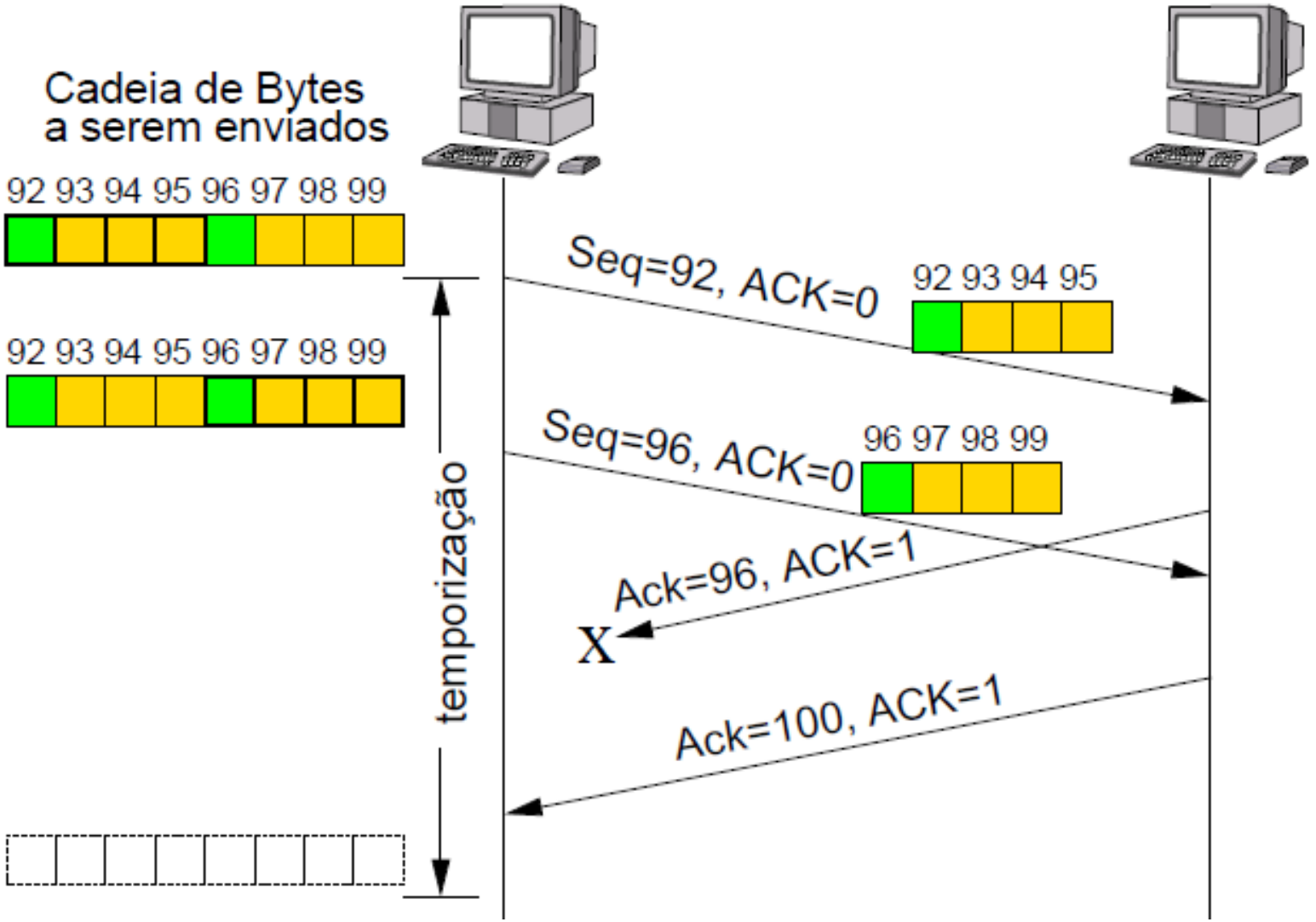
# Uso do Temporizador de Retransmissão



# Estudo de Caso

- Suponha que dois segmentos são transmitidos, e que haja perda apenas do reconhecimento para o primeiro segmento
- **Caso 1**: o reconhecimento para o segundo segmento chega **antes** do temporizador associado ao primeiro expirar
  - O primeiro segmento **não** será retransmitido (reconhecimento cumulativo)

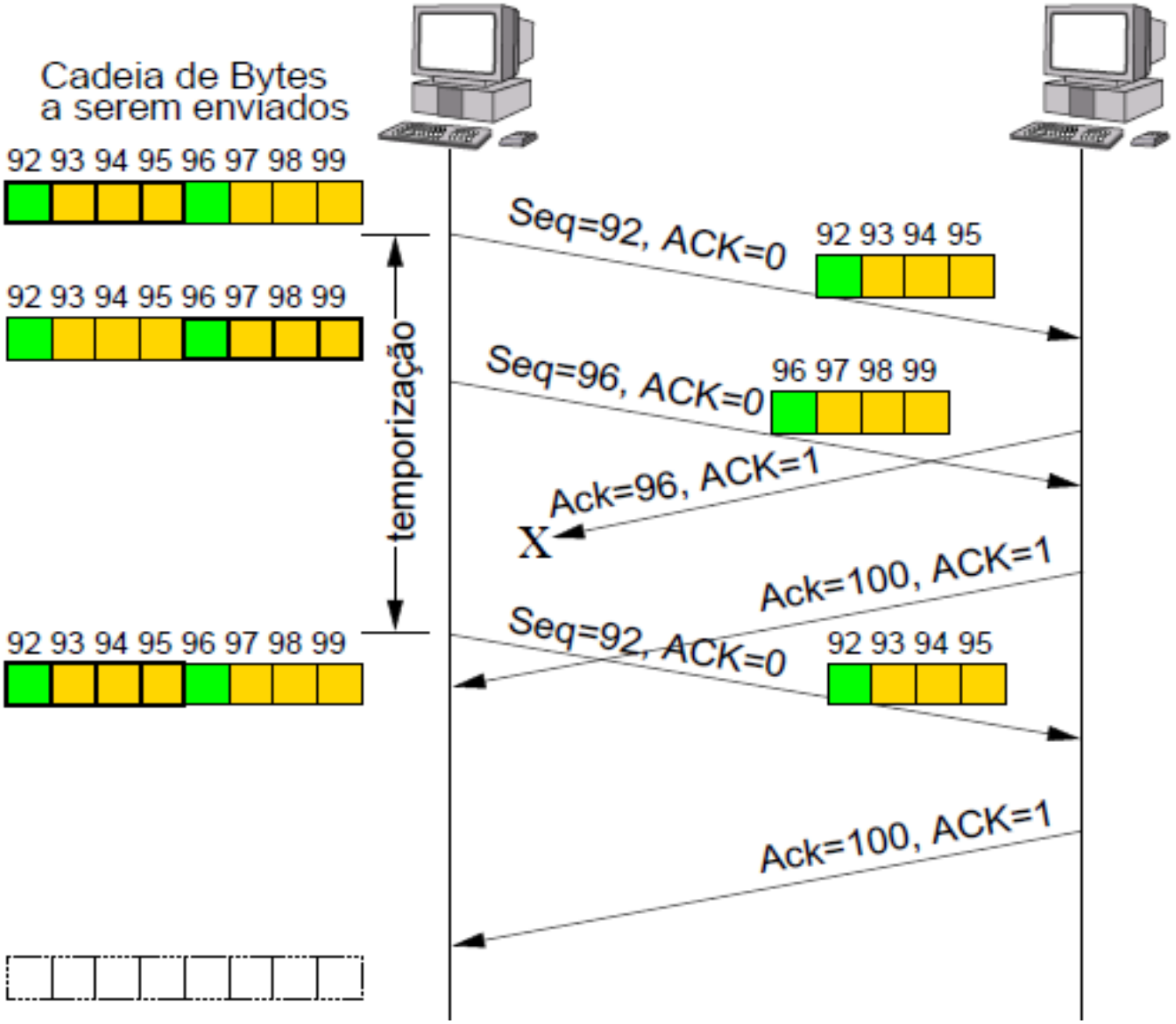
# Caso 1


Cadeia de Bytes
a serem enviados
92 93 94 95 96 97 98 99
92 93 94 95 96 97 98 99
Seq=92, ACK=0
92 93 94 95
Seq=96, ACK=0
96 97 98 99
Ack=96, ACK=1
X
Ack=100, ACK=1
temporização

# Estudo de Caso

- Suponha que dois segmentos são transmitidos, e que haja perda apenas do reconhecimento para o primeiro segmento
- **Caso 2**: o reconhecimento para o segundo segmento chega **depois** do temporizador associado ao primeiro expirar
  - O primeiro segmento (e apenas ele) será retransmitido

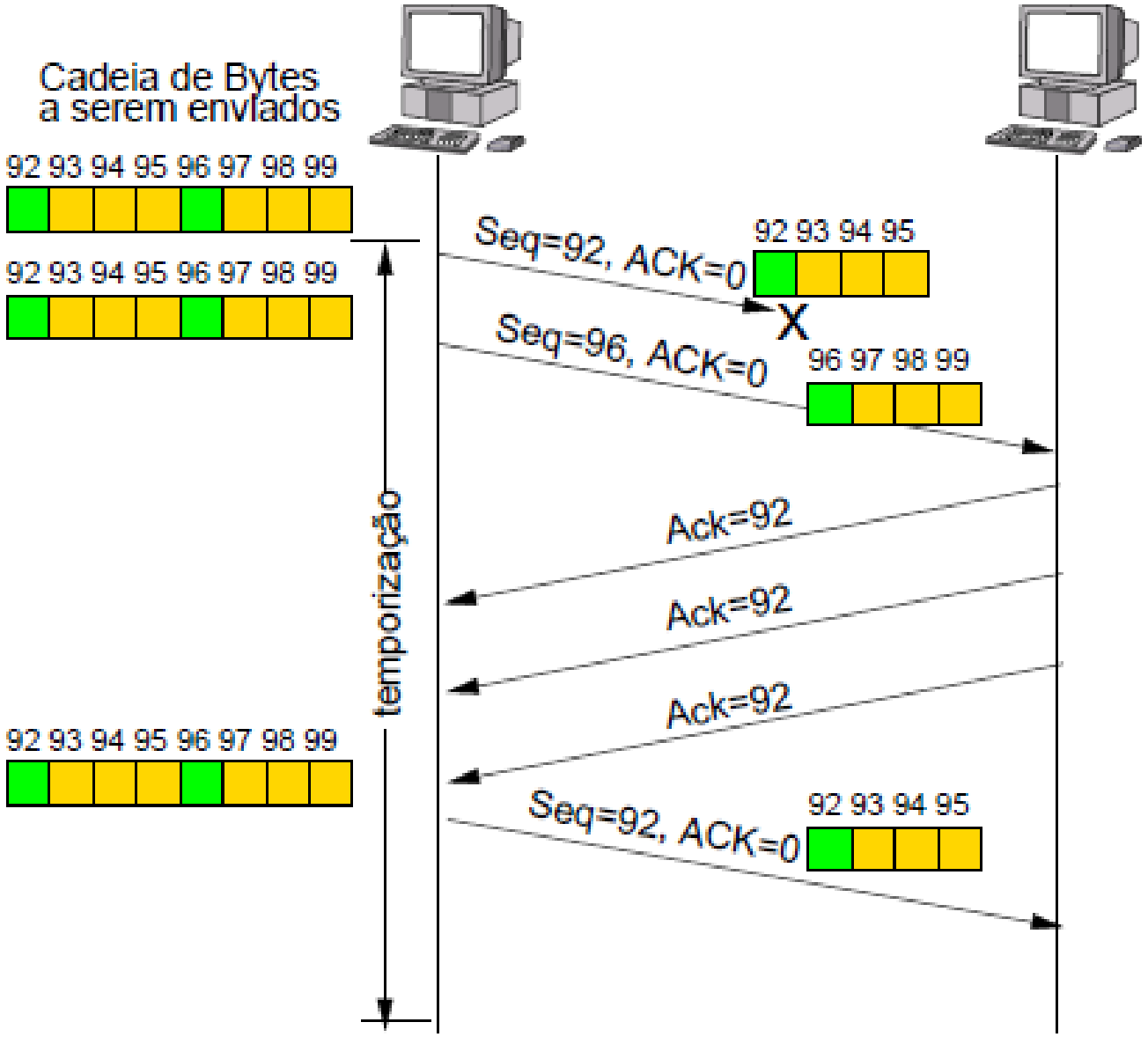
# Caso 2


Cadeia de Bytes
a serem enviados
92 93 94 95 96 97 98 99
92 93 94 95 96 97 98 99
92 93 94 95 96 97 98 99
temporização
Seq=92, ACK=0
92 93 94 95
Seq=96, ACK=0
96 97 98 99
Ack=96, ACK=1
X
Ack=100, ACK=1
Seq=92, ACK=0
92 93 94 95
Ack=100, ACK=1

# Retransmissão acelerada

- Para reduzir o tempo na entrega dos segmentos, o remetente TCP *nem sempre* espera pelo temporizador de retransmissão
- Ao perceber uma lacuna (falha na sequência correta dos segmentos), o destinatário TCP envia **três reconhecimentos (ACKs) duplicados**
- Ao recebê-los, o remetente TCP efetua uma retransmissão **rápida** do segmento
  - Chama-se “rápida” porque a retransmissão ocorre *antes* do temporizador associado ao segmento faltoso expirar

# Exemplo de Retransmissão acelerada



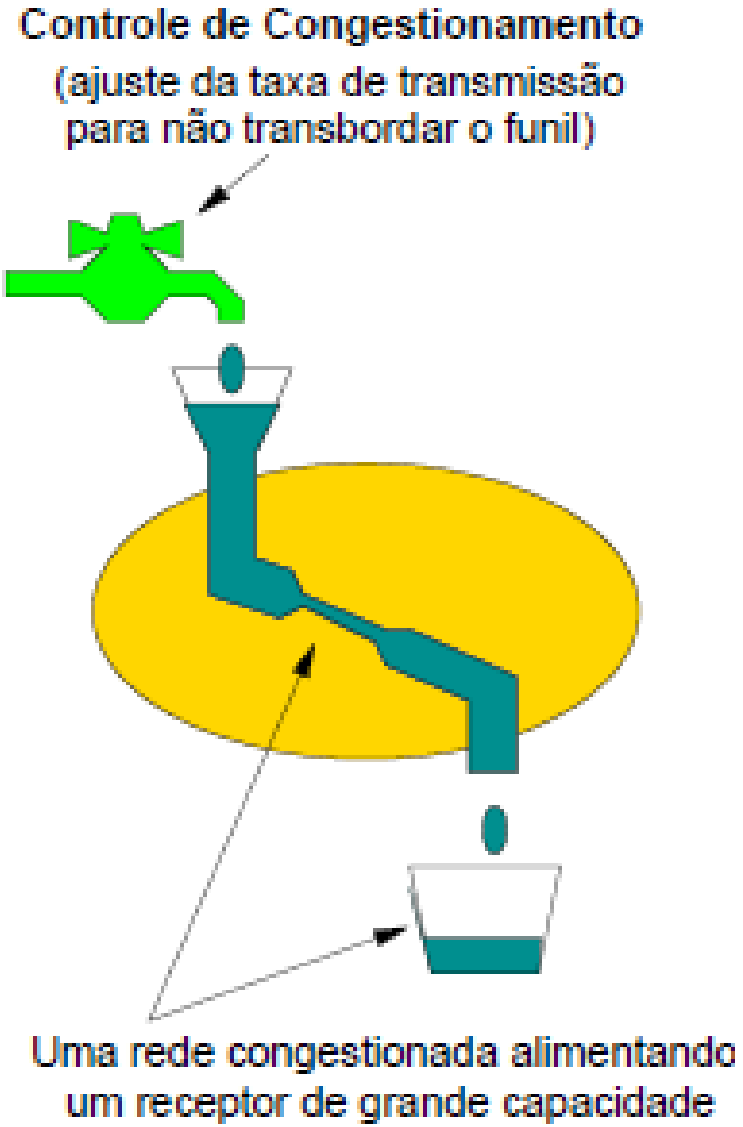
# Controle de Erros

- O TCP usa uma combinação de dois mecanismos para controle de erros
  - **Go Back-N**, onde os segmentos recebidos fora de ordem são imediatamente descartados
  - **Retransmissão Seletiva**, onde os segmentos recebidos fora de ordem são armazenados em um *buffer* intermediário. Quando o segmento faltante é recebido, a sequência é restabelecida

# Controle de Congestionamento

- Serviço que busca evitar a perda excessiva de segmentos devido a congestionamento na infraestrutura de rede
  - Muitas conexões TCP usando enlaces compartilhados
- Segmentos perdidos por congestionamento na rede devem ser retransmitidos
- Retransmissão de segmentos implica custos (largura de banda nos enlaces, atraso na entrega de dados para as aplicações)

# Controle de Congestionamento



# Controle de Congestionamento TCP

- TCP: implementa serviço de controle de congestionamento *fim a fim*
  - Perda de segmentos (esgotamento de temporizador e três reconhecimentos duplicados) indica congestionamento na rede
- Estratégia baseada na **taxa de envio de dados pelo remetente**: se um remetente TCP percebe “pouco” congestionamento no caminho entre ele e o destinatário, *aumentará* sua taxa de envio; caso contrário, *reduzirá* sua taxa de envio

# Controle de Congestionamento TCP

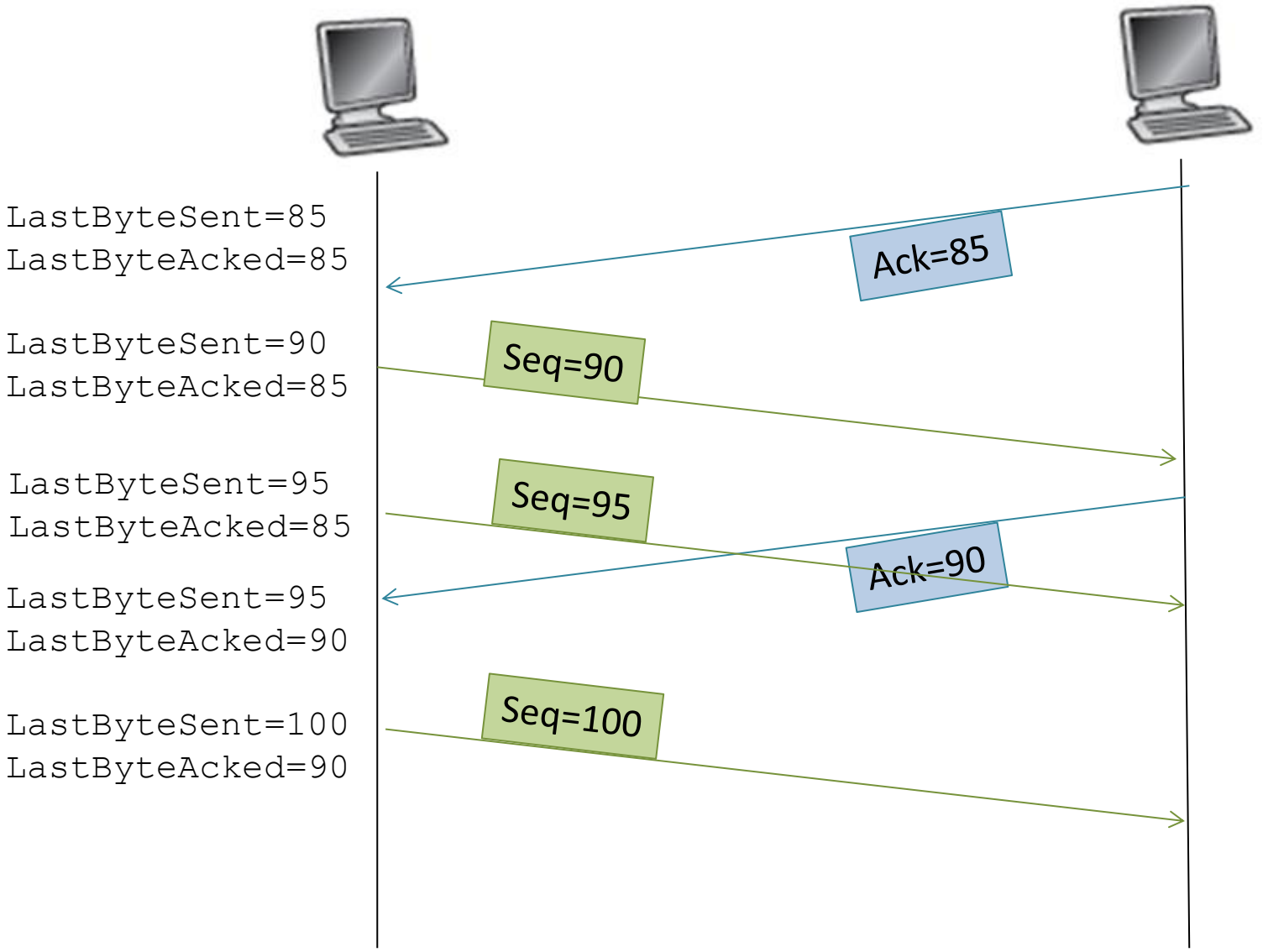
- Limitando taxa de envio de dados no remetente
  - Considera variáveis de estado da conexão TCP:
    - **Janela de Congestionamento** (`CongWin`): limita a taxa de envio de tráfego
    - Último byte enviado (`LastByteSent`)
    - Último byte reconhecido (`LastByteAcked`)
  - Especificamente:

    `LastByteSent - LastByteAcked ≤ CongWin`
  - Essa restrição limita a quantidade de dados não reconhecidos no remetente e, indiretamente, a taxa de envio de dados

# Exemplo: Limitando taxa de envio

- Suponha que:
  - Remetente TCP sempre possui dados a enviar
  - `LastByteSent` = 85
  - `MSS` = 5 bytes
  - `CongWin` = 10 bytes
  - Não há perdas de segmentos na rede

# Exemplo: Limitando taxa de envio



# Controle de Congestionamento TCP

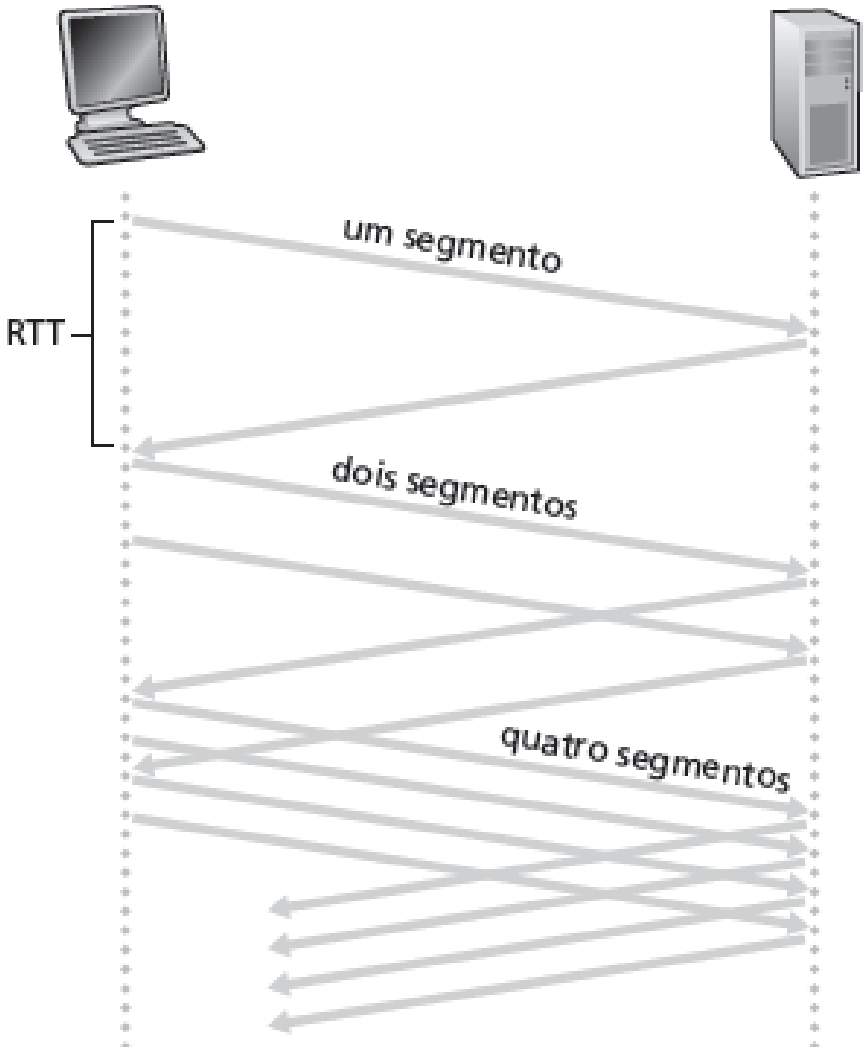
- TCP é autorregulado porque ajusta (aumenta ou reduz) a `CongWin` com base nos segmentos de reconhecimento
- Outra variável de estado, chamada `limiar`, determina a proporção do aumento da `CongWin`

# Controle de Congestionamento TCP

- Algoritmo de controle de congestionamento TCP
  1. Inicialmente, `CongWin` aumenta de forma *exponencial* até atingir o valor de `limiar`
     - Fase chamada **partida lenta** porque valor de `CongWin` é pequeno

# Controle de Congestionamento TCP

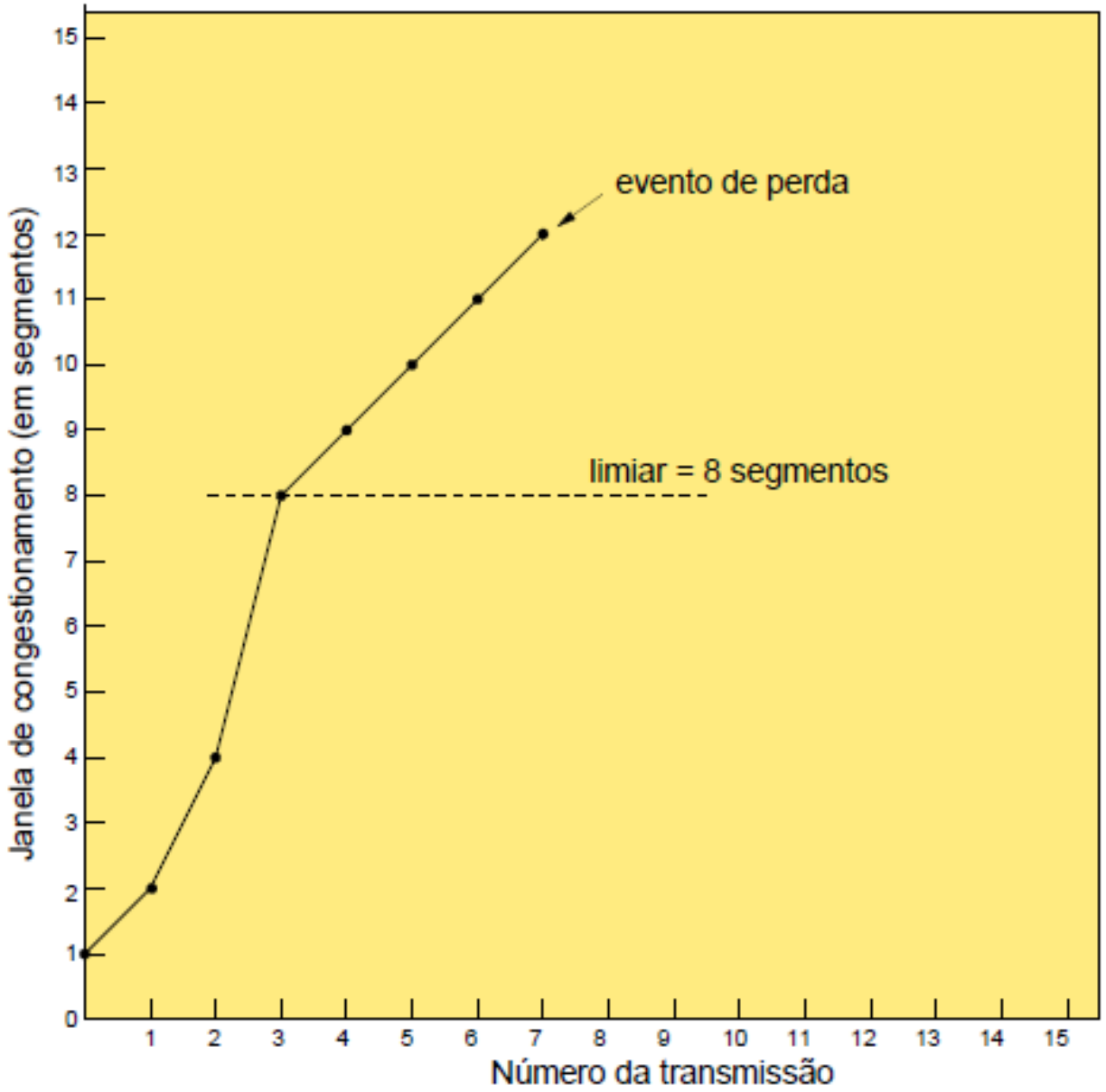
- Partida Lenta TCP



# Controle de Congestionamento TCP

- Algoritmo de controle de congestionamento TCP
  2. Em seguida, `CongWin` aumenta de forma *linear* até ocorrer um **evento de perda**
     - Eventos de perda representam esgotamento de temporização e recebimento de três reconhecimentos duplicados

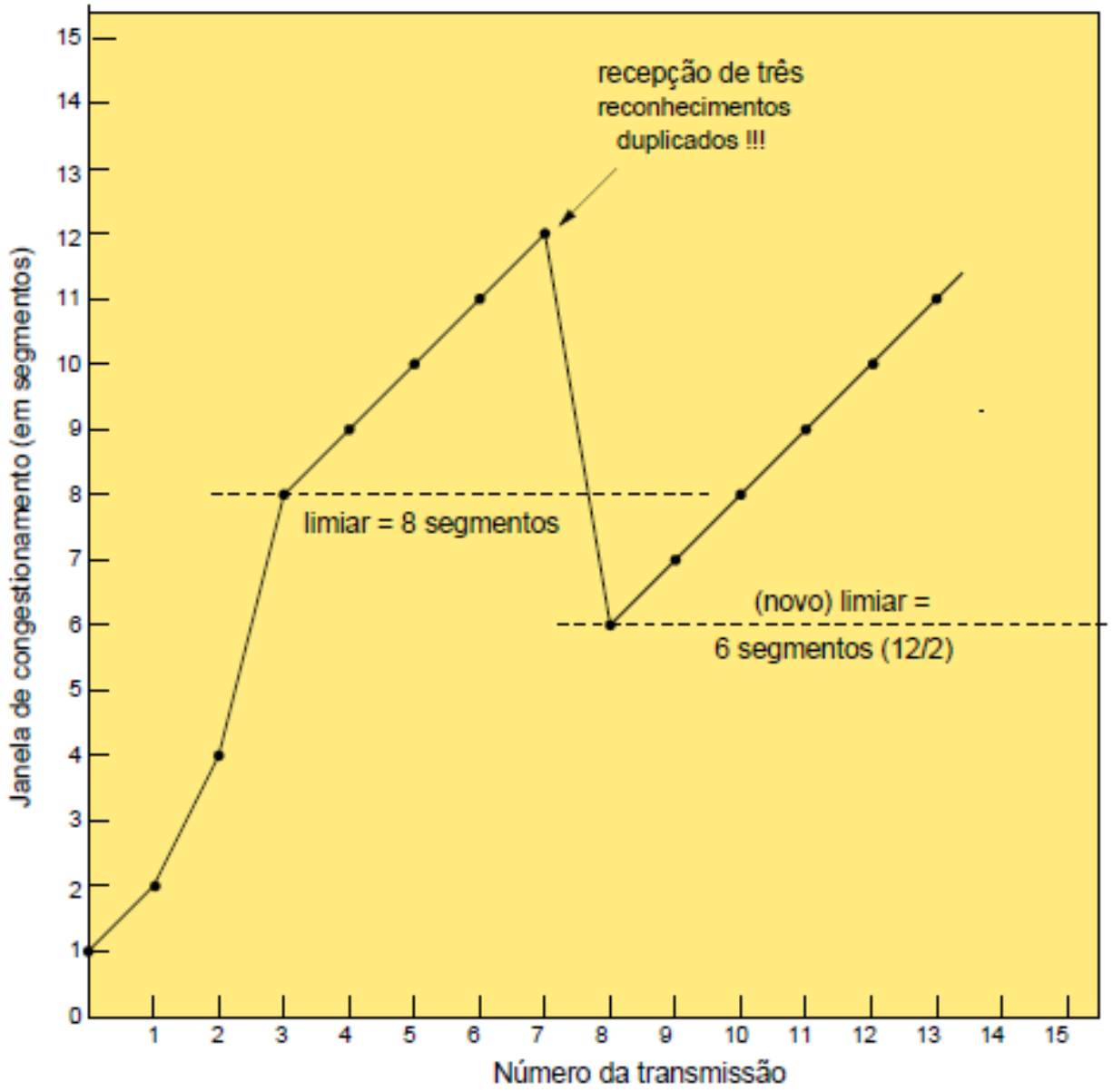
# Controle de Congestionamento TCP



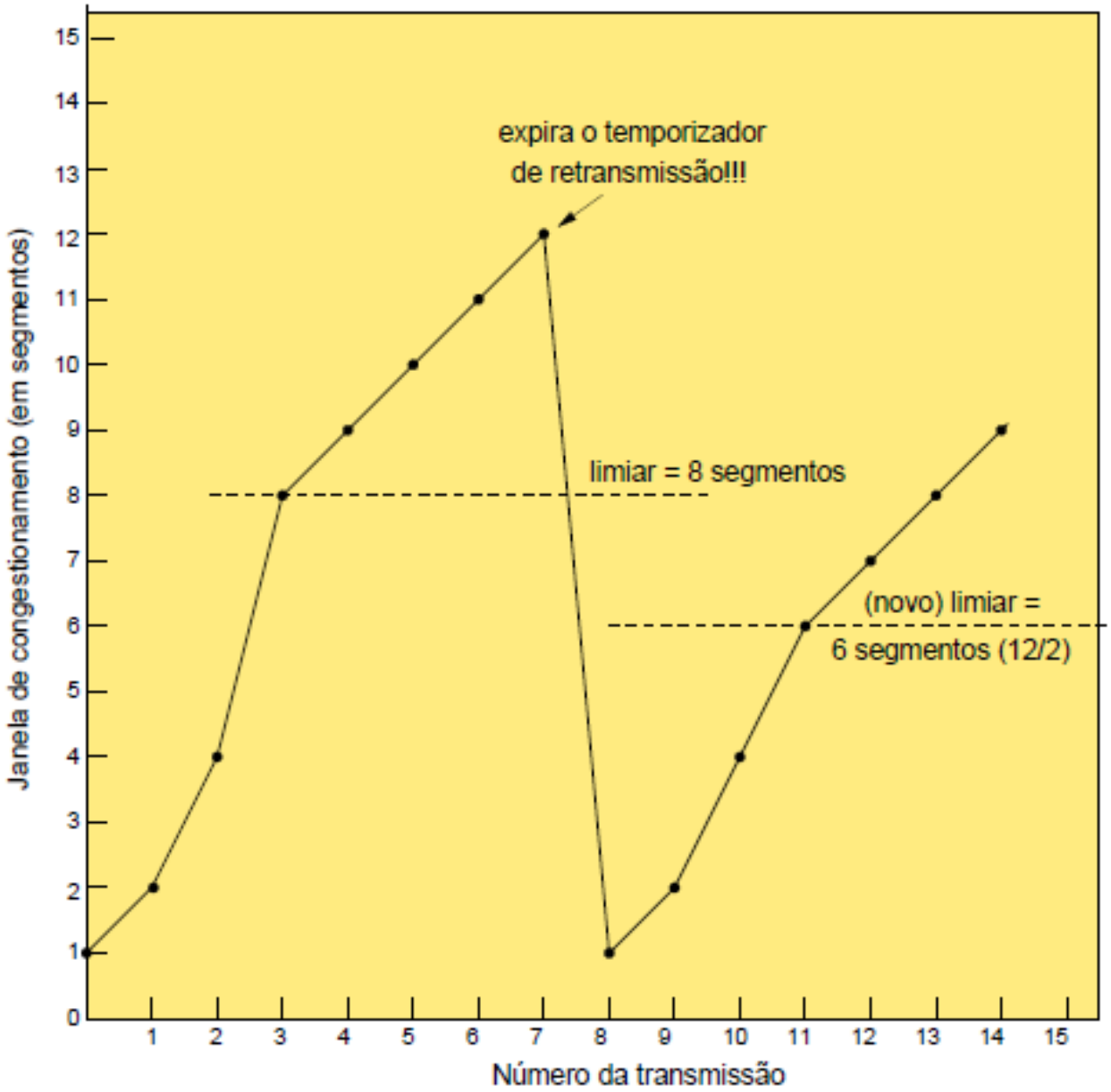
# Controle de Congestionamento TCP

- Algoritmo de controle de congestionamento TCP
  3. Imediatamente após a ocorrência de um evento de perda, o valor do `limiar` é definido na *metade* da `CongWin`, isto é, `limiar = (CongWin/2)`
  4. Comportamento do remetente TCP passará a ser regido pela natureza do evento de perda

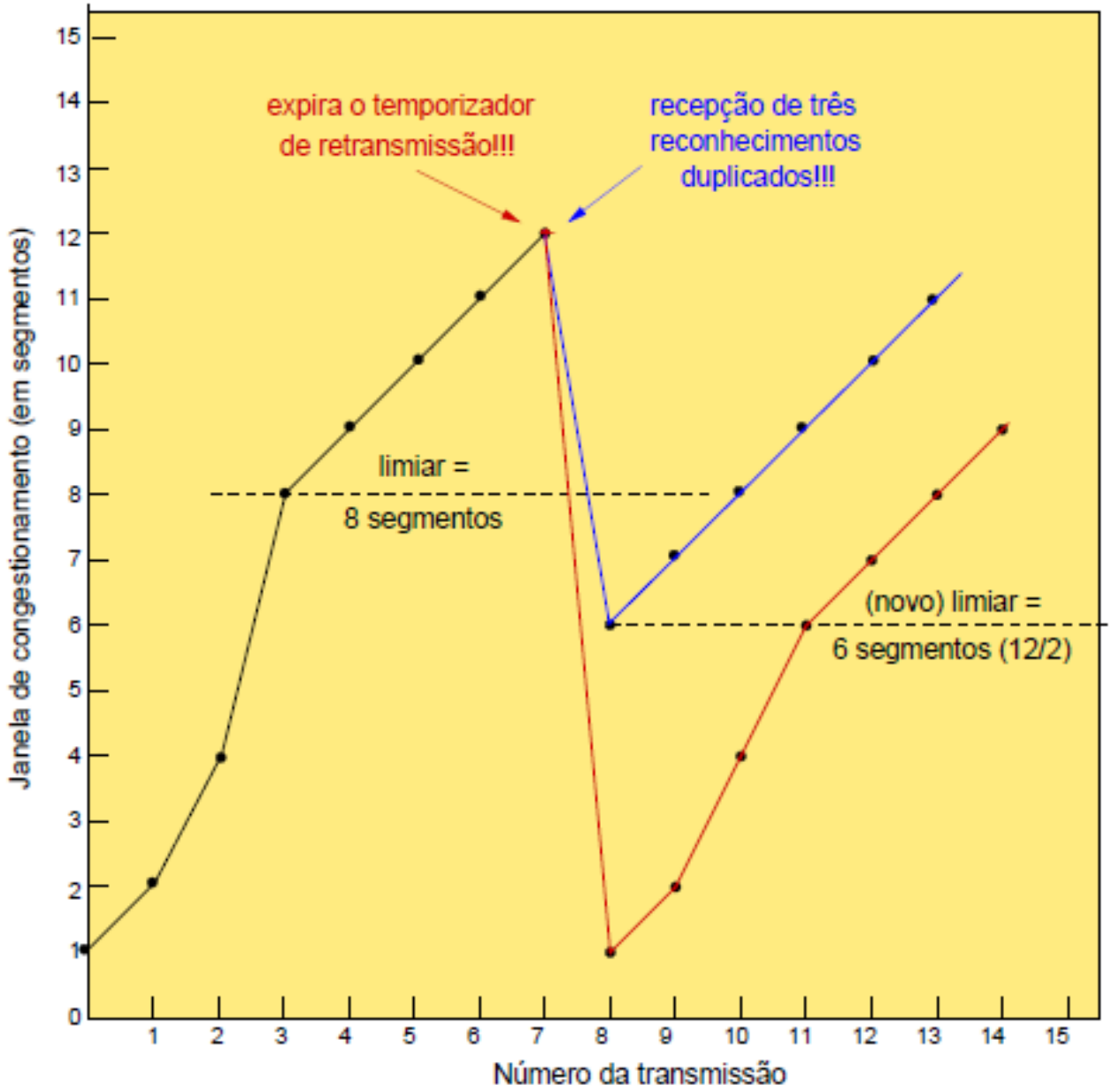
# Recebimento de Três Reconhecimentos Duplicados



# Esgotamento do Temporizador de Retransmissão



# Comparação entre os eventos de perda



# Protocolo UDP

- Definido na RFC 768, o protocolo UDP (*User Datagram Protocol*) é usado por aplicações que não podem (ou não querem) incorporar os custos dos serviços prestados pelo protocolo TCP
  - Protocolo simplificado de Nível de Transporte
- Provê serviço de entrega **não** orientado a conexão
  - Ou seja, não há apresentação antes que processos de Nível de Aplicação iniciem uma comunicação

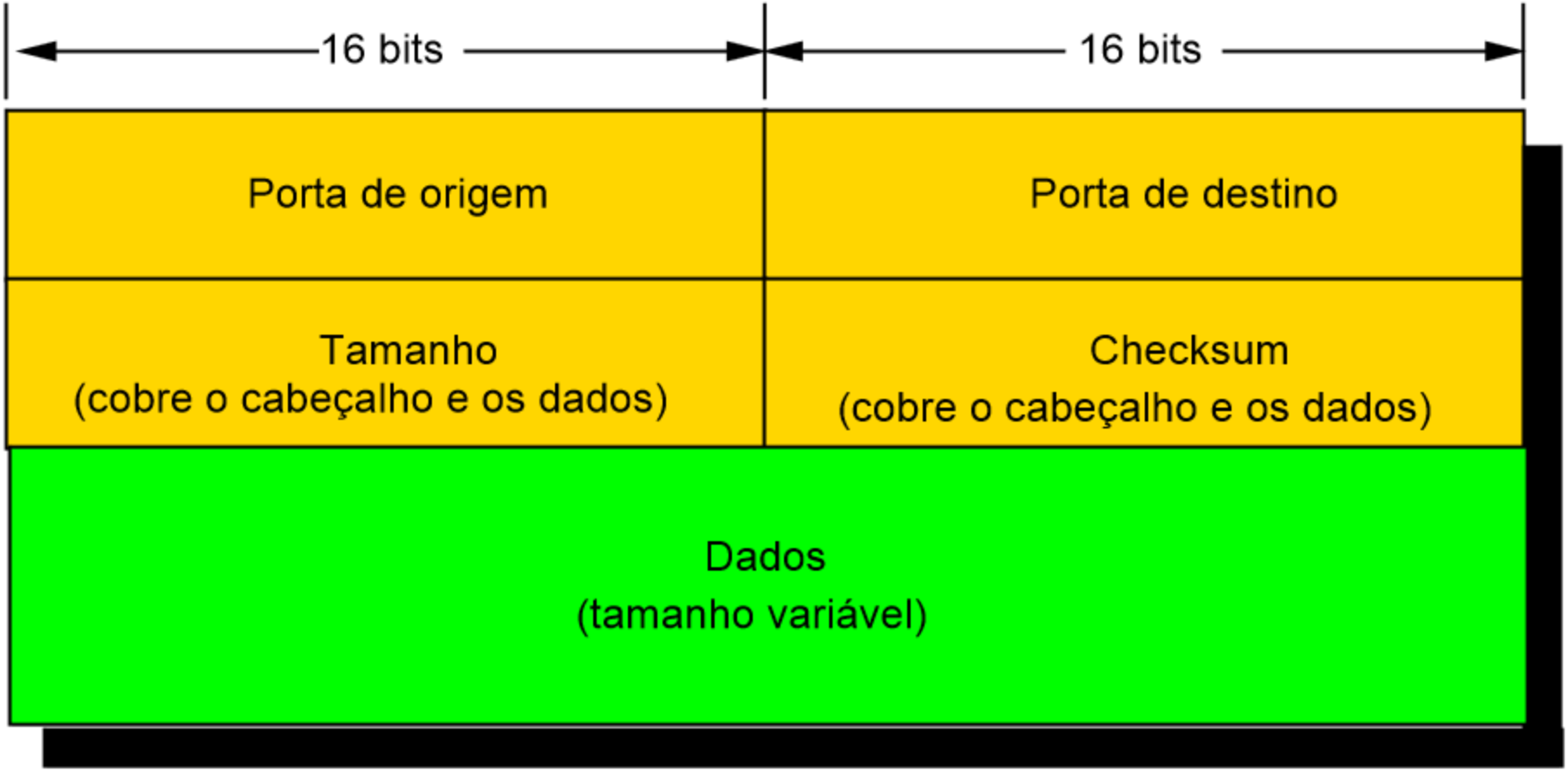
# Serviços do UDP

- Provê serviço **não** confiável de entrega dos dados
  - Não garante a entrega ao destinatário
  - Não fornece serviço de controle de erros
- Não fornece serviços de controle de fluxo e nem de congestionamento
  - Taxa de envio não-regulada no remetente
  - Taxa de recebimento é limitada apenas pela largura de banda nos enlaces
- Mas... Fornece serviço de verificação de erros em segmentos

# Exercício [Kurose, pp. 215, Ex. 3]

- É possível que uma aplicação desfrute de transferência confiável de dados mesmo quando roda sobre UDP? Caso a resposta seja afirmativa, como isso acontece?

# Formato do Segmento UDP



# Relacionamento entre Aplicações e Protocolo de Nível de Transporte

| Aplicação | Protocolo de Nível de Aplicação | Protocolo de Nível de Transporte |
|---|---|---|
| Correio eletrônico | SMTP | TCP |
| World Wide Web (WWW) | HTTP | TCP |
| Transferência de Arquivos | FTP | TCP |
| Tradução de Nomes | DNS | UDP |
| Multimídia | proprietário | UDP ou TCP |

# Conclusão

- Nível de Transporte provê serviços ao Nível de Aplicação
- Protocolo TCP: entrega confiável
- Protocolo UDP: não-orientado a conexão

# Perguntas?

helber.silva@ifrn.edu.br