

**INSTITUTO FEDERAL**  
Norte de Minas Gerais  
Campus Januária

# Redes de Computadores

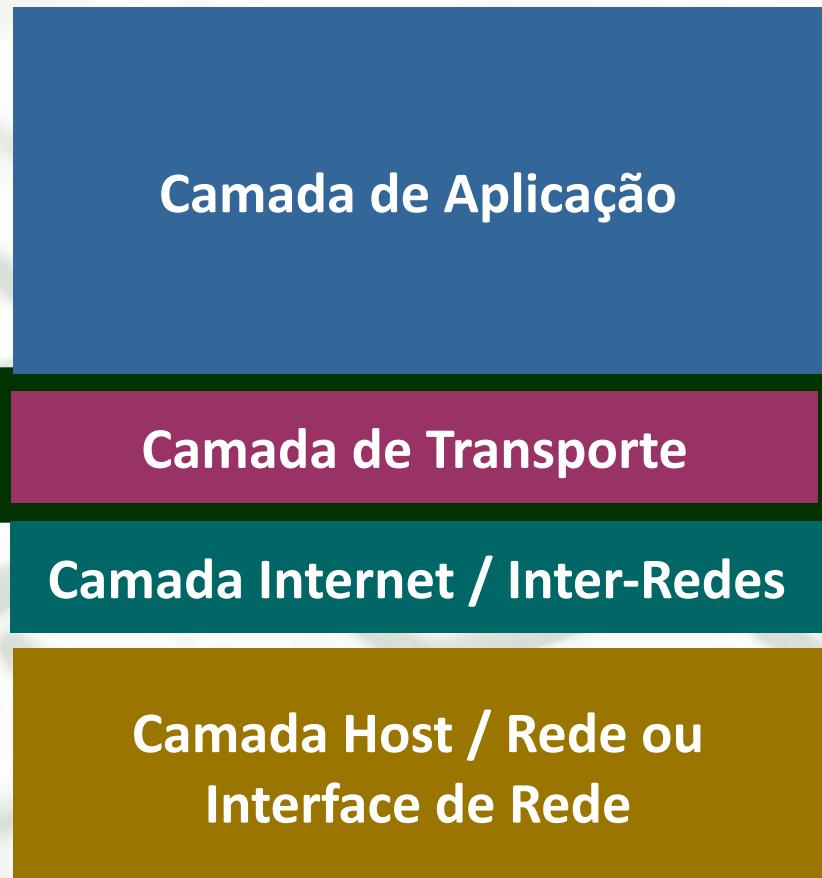
## *- Camada de Transporte -*

# Camada de Transporte

**Modelo OSI**



**Arquitetura TCP / IP**



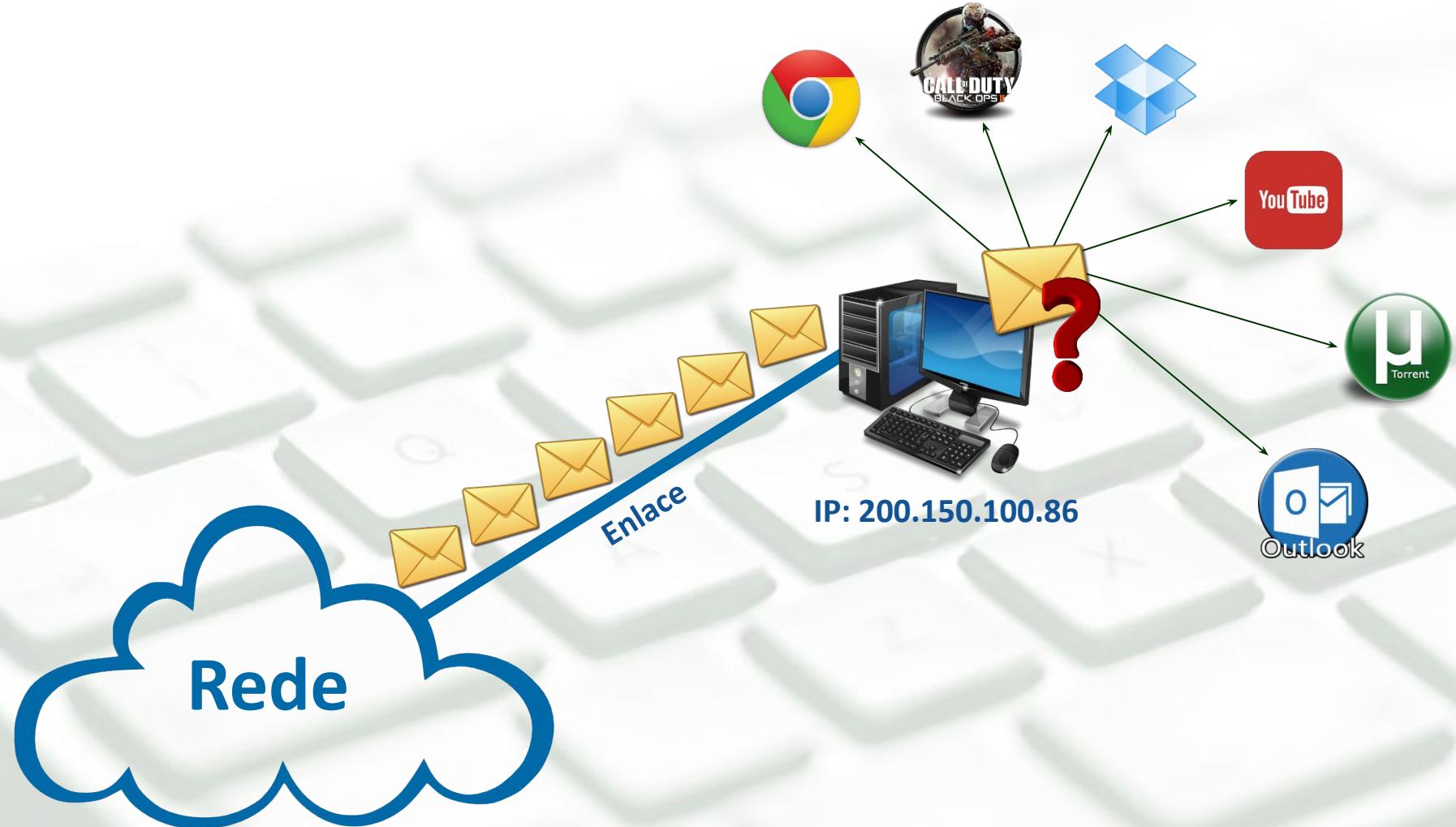
# Revisando...

- A **Camada de Enlace** é responsável por entregar frames entre nós que compartilham o mesmo meio de transmissão.
  - Comunicação *node-to-node*
- A **Camada de Rede** é responsável por prover a comunicação entre *hosts*, mesmo que seja necessário atravessar nós intermediários.
  - Comunicação *host-to-host*

# Camada de Transporte

- A finalidade real de uma rede de computadores é fazer com que **aplicações (processos)** troquem **informações entre si**.
- Porém, um único *host* (computador) pode executar inúmeros processos que utilizam a rede de comunicação simultaneamente.
- A **Camada de Transporte** será responsável pela comunicação inter-processos em execução nos sistemas finais.

# Introdução

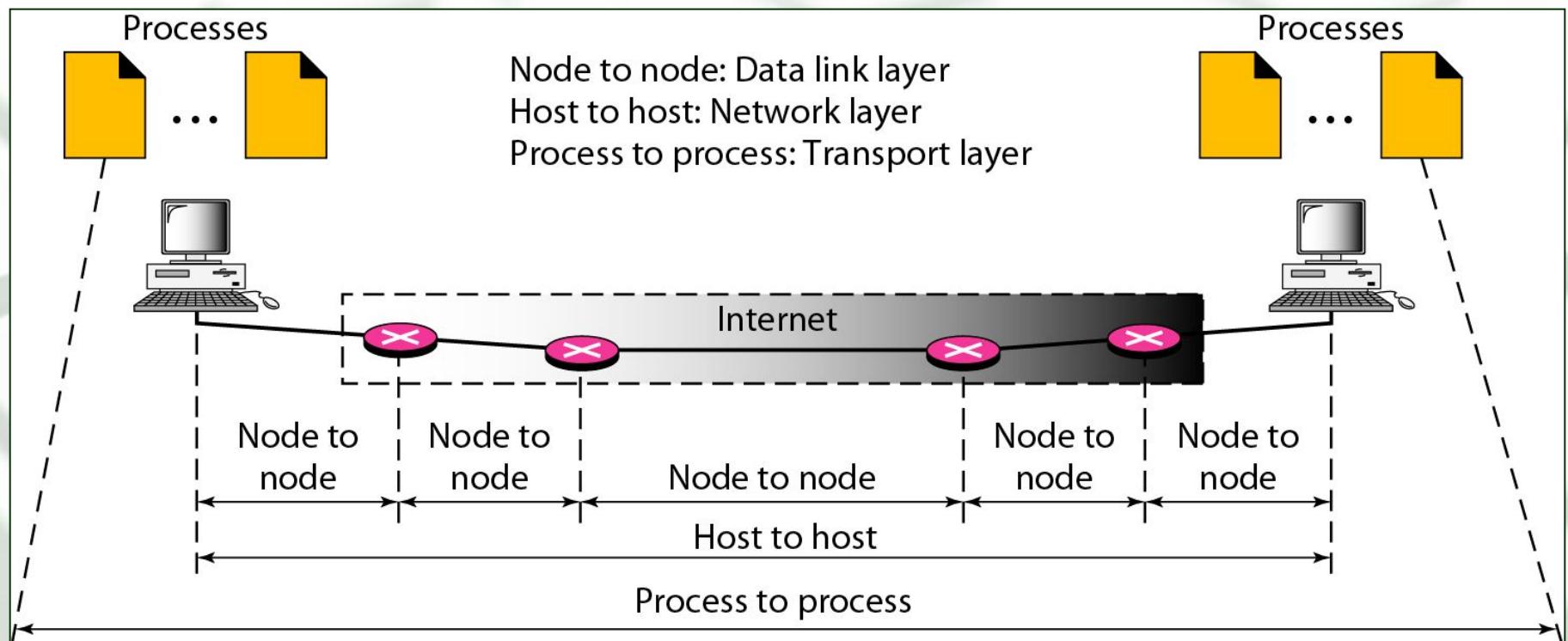


# Camada de Transporte

- A **Camada de Transporte** é responsável por transportar os dados das aplicações através da rede.
  - *Oferece serviços à camada de Aplicação.*
- **Segmentos de dados** são transformados em **pacotes** e repassados para a camada de rede que efetuará o roteamento até o *host* de destino.
- Pacotes recebidos pela rede são repassados para a camada de transporte, que realizará a entrega para a aplicação correspondente.

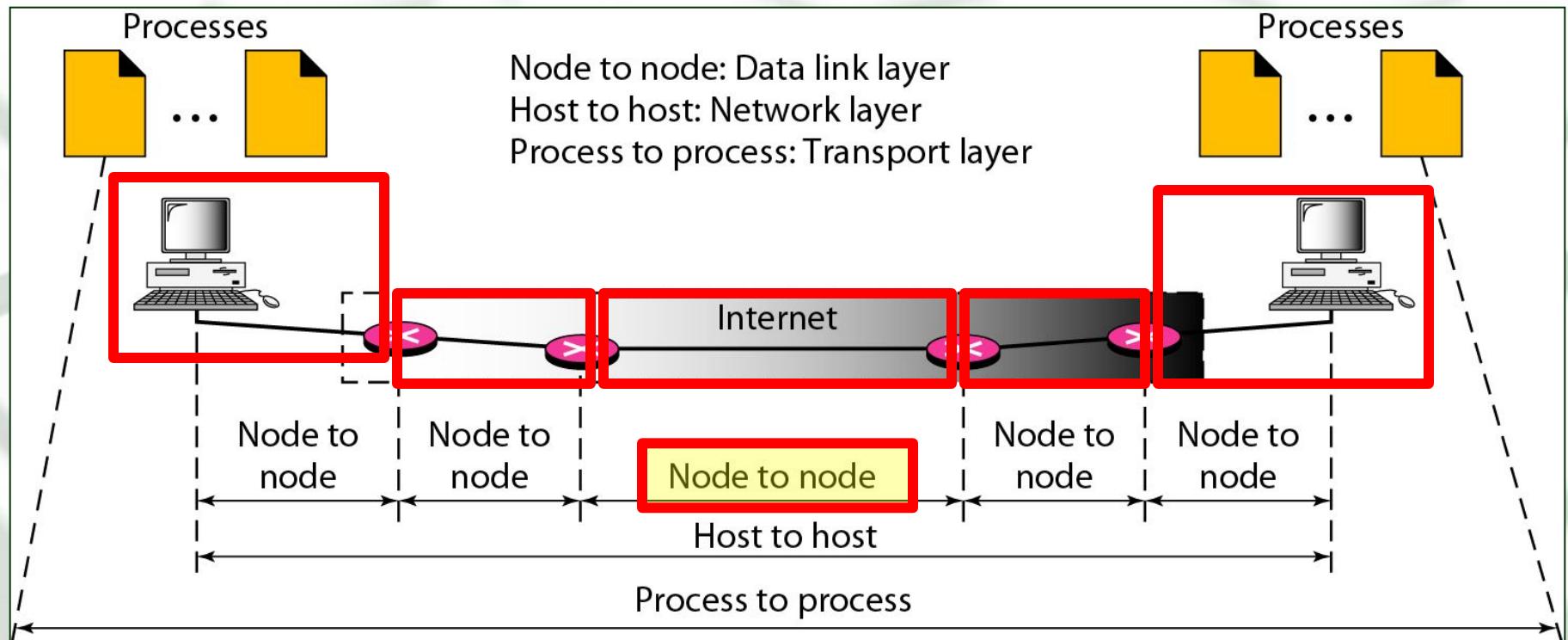
# Comunicação Inter-Processos

## ■ Comunicação *process-to-process*.



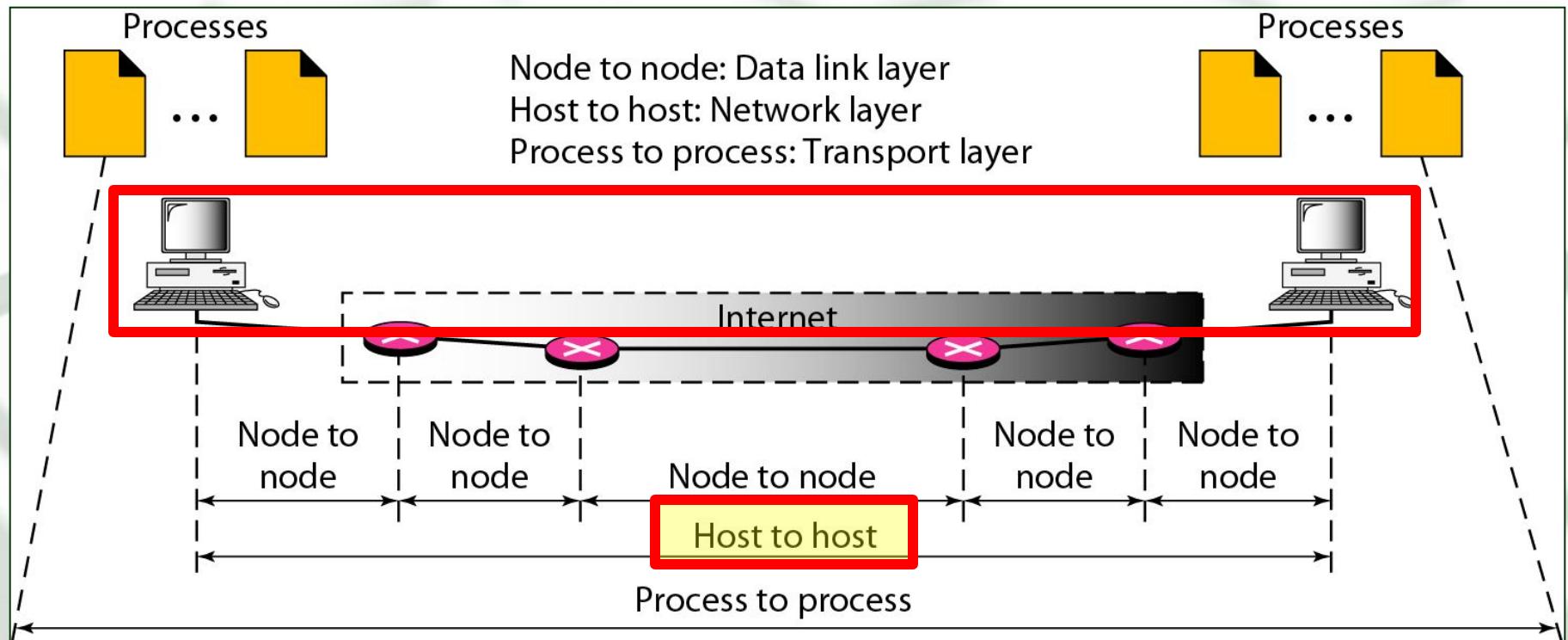
# Comunicação Inter-Processos

## ■ Comunicação *process-to-process*.



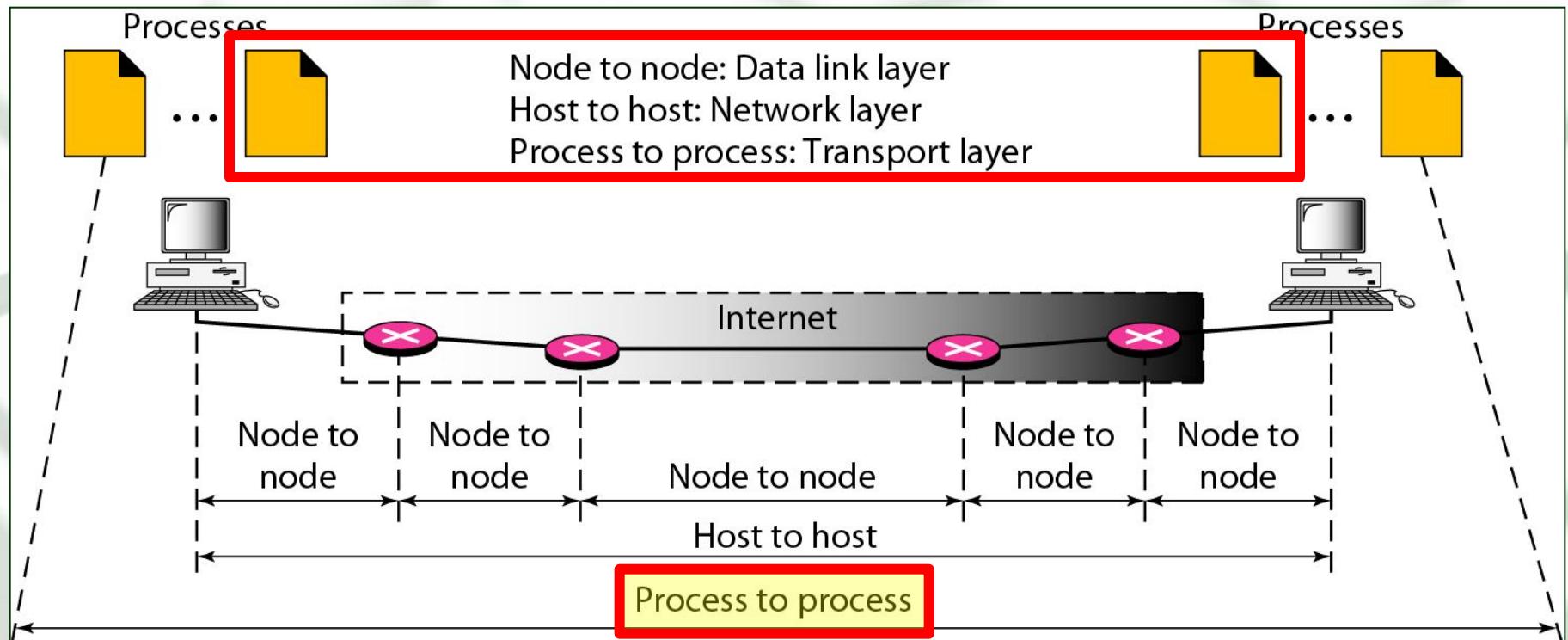
# Comunicação Inter-Processos

## ■ Comunicação *process-to-process*.



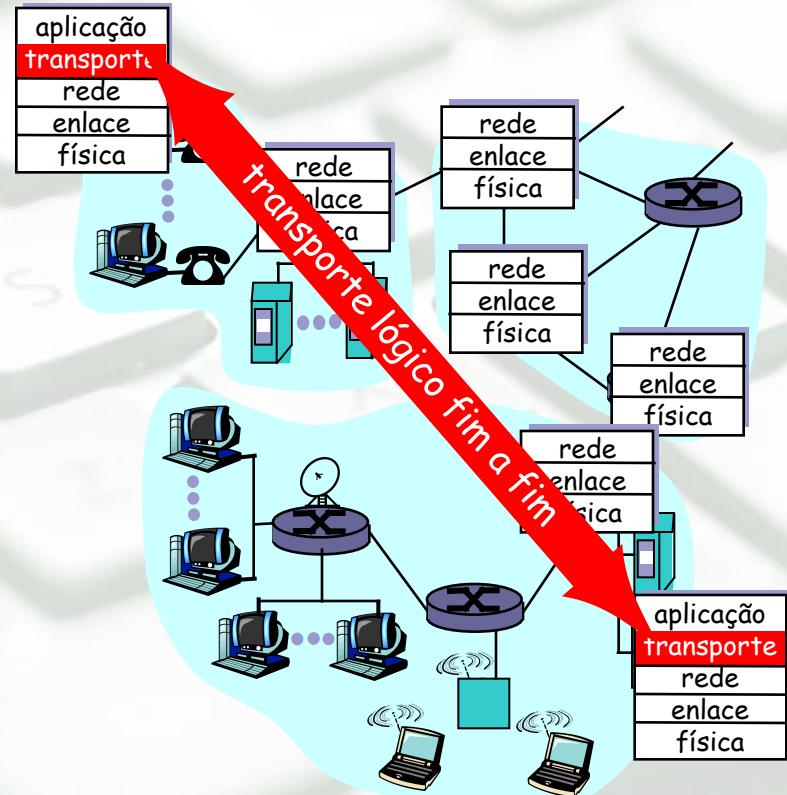
# Comunicação Inter-Processos

## ■ Comunicação *process-to-process*.

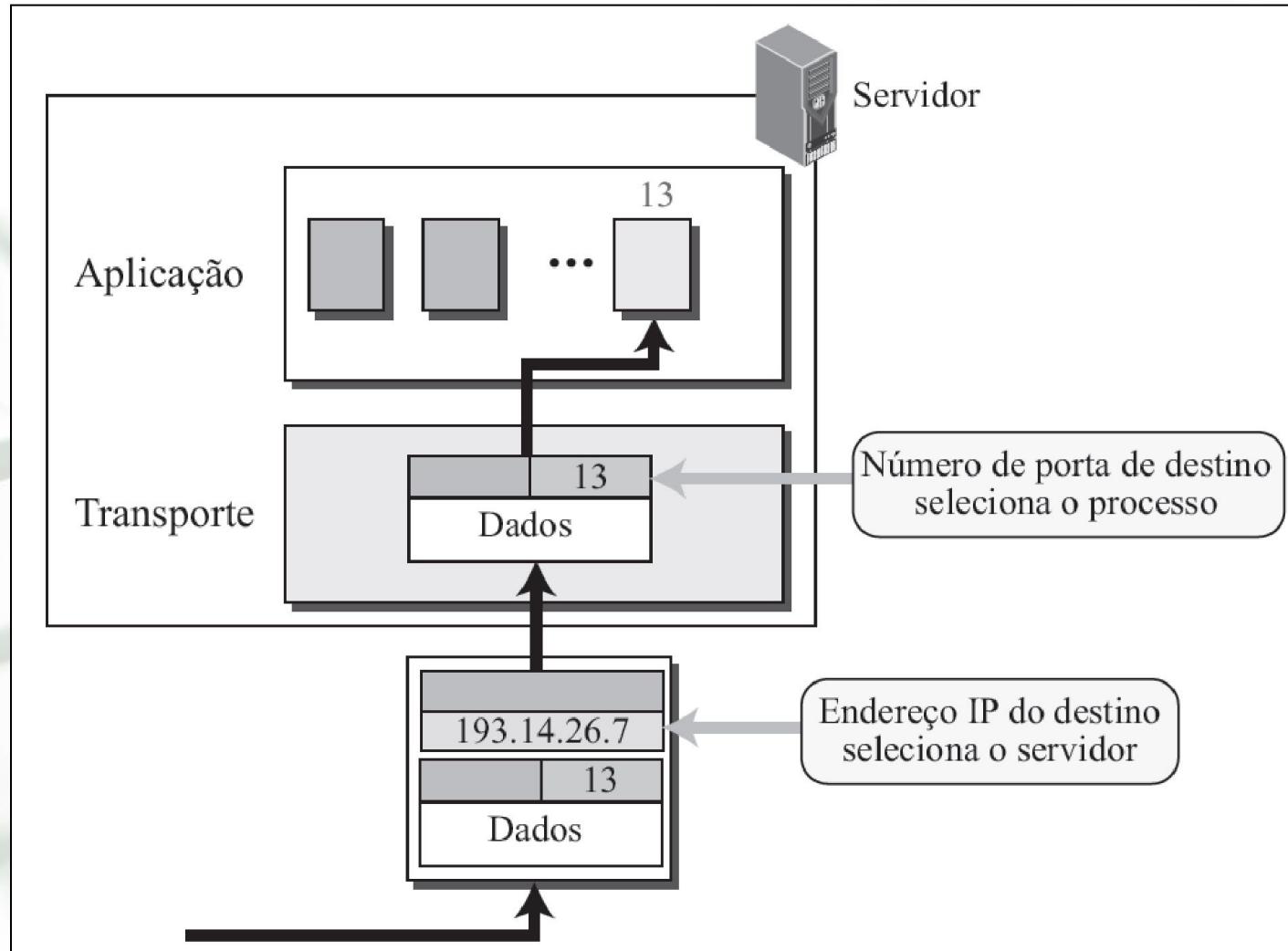


# Endereçamento

- Para qualquer tipo/nível de comunicação, um esquema de endereçamento é necessário!
- Em nível de Enlace...
  - MAC Address
- Em nível de Rede...
  - Endereço IP
- Em nível de Transporte...
  - Porta de comunicação.



# Endereçamento



# Portas de Comunicação

- Na arquitetura TCP/IP, uma porta de comunicação é um número inteiro de 16 bits.
  - **0 <-> 65.535**
- Uma porta mapeia um determinado processo que está utilizando a rede de comunicação.

*Problema... Para um cliente recuperar e-mails de um servidor, ele precisa saber antecipadamente qual a porta que este servidor está “escutando”...*

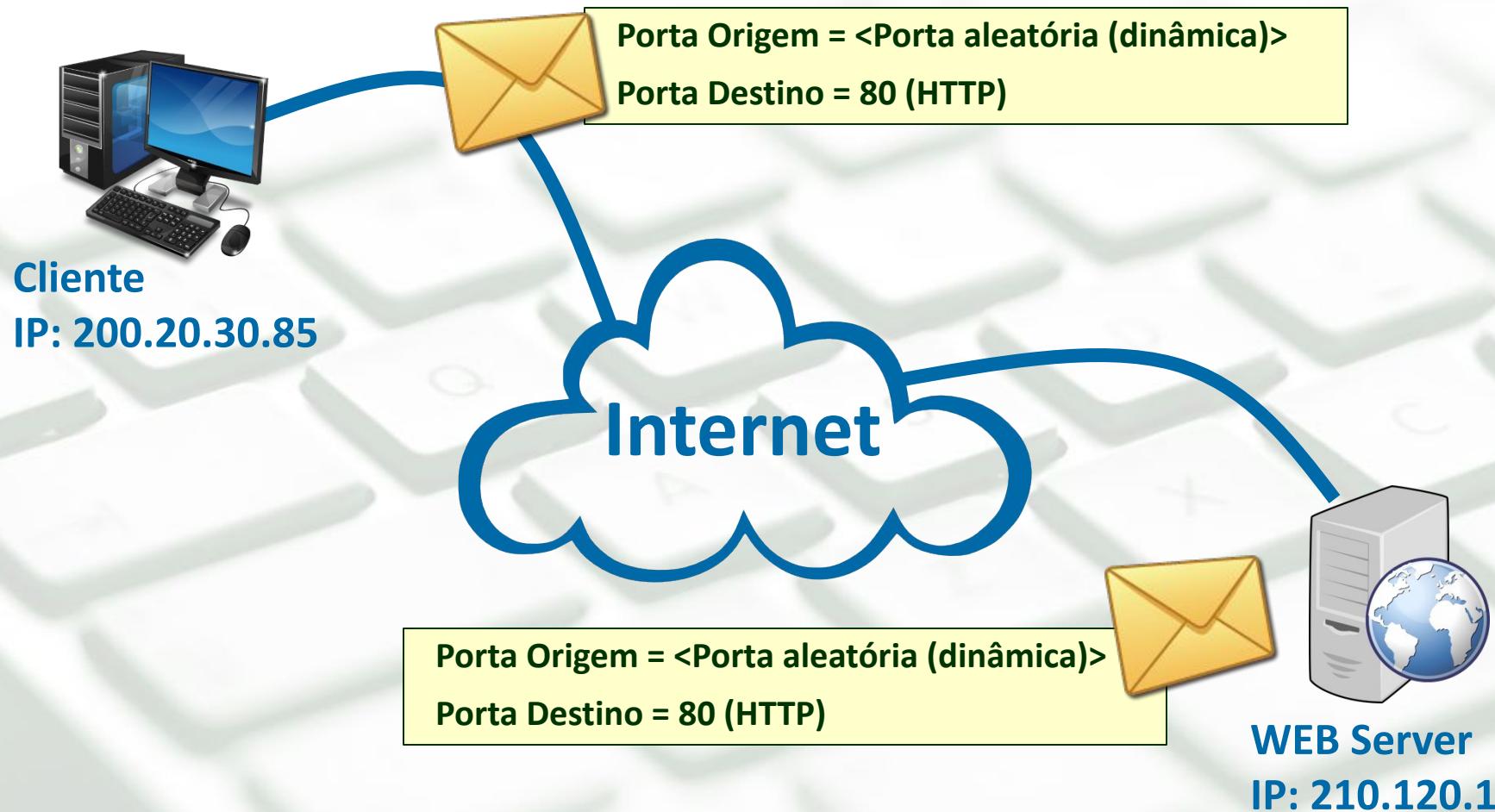
# Faixas IANA

Nome	Faixa	Descrição
Portas Conhecidas	0 – 1023	Atribuídas e controladas pela IANA.
Portas Registradas	1024 – 49151	Necessita registro junto à IANA.
Portas Dinâmicas	49152 – 65535	Portas para uso geral.

# Portas Conhecidas

- Servidores de aplicações padrões e seus protocolos de rede já possuem portas bem definidas, designadas por meio de RFCs (normativas).
  - 21 → *FTP*
  - 22 → *SSH*
  - 25 → *SMTP*
  - 53 → *DNS*
  - 67 → *DHCP*
  - 80 → *HTTP*
  - 110 → *POP3*

# Paradigma de Comunicação



# Socket / Soquete



Cliente

IP: 200.20.30.85

200.20.30.85:58569

A combinação **IP:PORTA** identifica exclusivamente um processo em execução.



Chamamos essa combinação de **SOCKET** de rede.

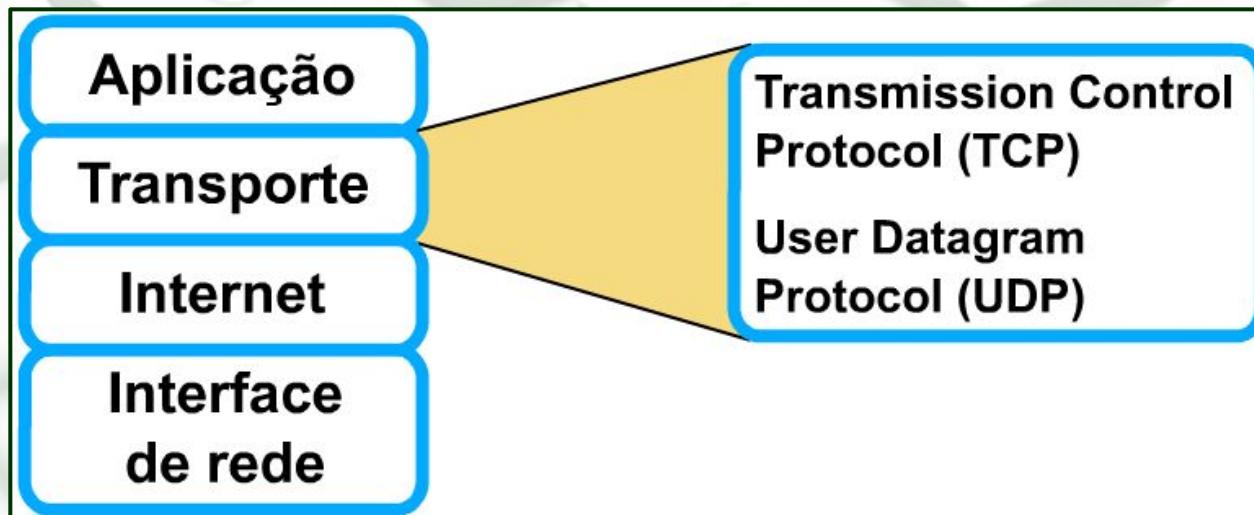
210.120.10.80:80

WEB Server

IP: 210.120.10.80

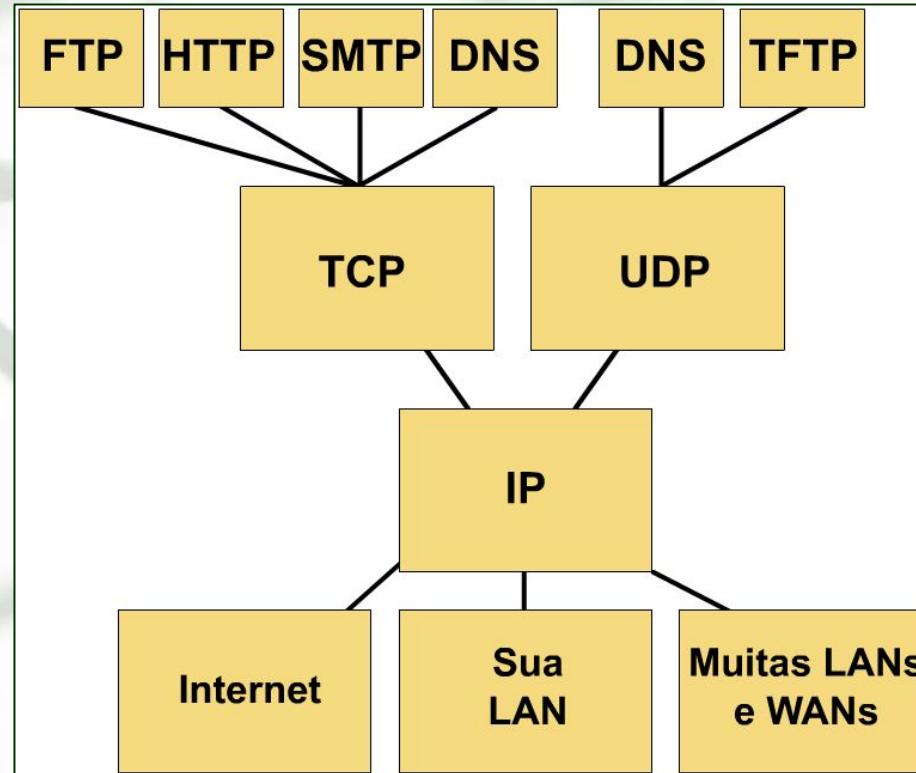
# Protocolos de Transporte

- TCP/IP prevê dois serviços de transporte...
  - ***TCP – Transmission Control Protocol***
    - *Connection-oriented*
  - ***UDP – User Datagram Protocol***
    - *Connectionless*



# Arquitetura TCP/IP

- Aplicações selecionam o **protocolo de transporte** de acordo com o tipo de tráfego e requisitos de qualidade.



# Protocolos de Transporte

Como o protocolo IP **não oferece** nenhuma garantia de entrega de dados em nível de rede, a solução foi implementar mecanismos de confiabilidade na camada superior (**transporte**), através do protocolo **TCP**.

**TCP/IP**

**Serviço Confiável**  
Endpoints da rede

**Modelo *Best-Effort***  
Núcleo da rede

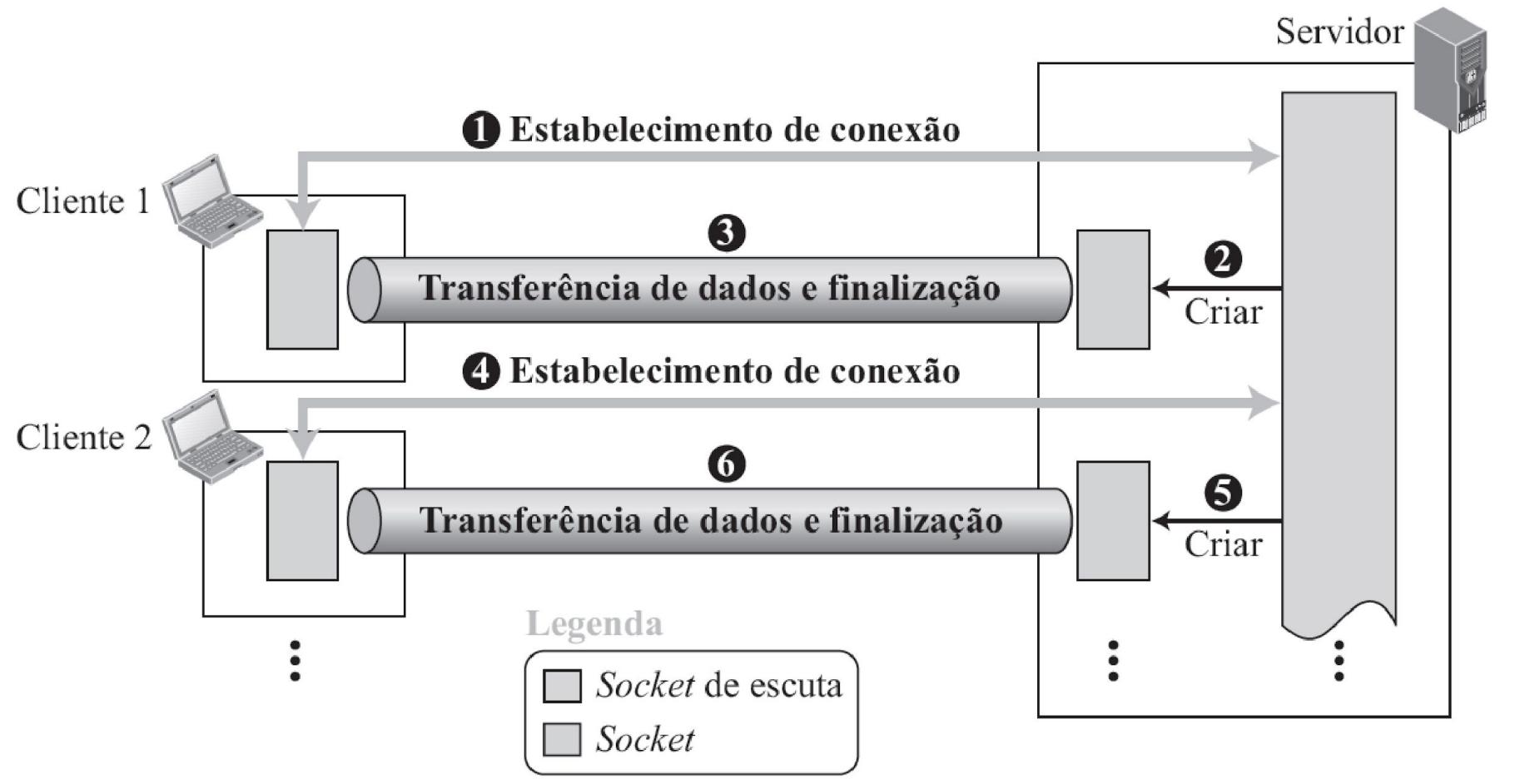
# Protocolos de Transporte

- O TCP (*Transmission Control Protocol*) oferece um serviço de **transporte confiável**.
- Implementa mecanismo de **conexão virtual** entre os processos comunicantes.
  - Estabelecimento de conexão.
  - Transferência de dados.
  - Encerramento de conexão.
- Adota uma abordagem de **fluxo de dados contínuos** (*data stream*).

# Outras Características do TCP

- Define como “*segmento*” a unidade de transmissão.
  - *Fluxo de bytes da aplicação são encapsulados em segmentos.*
- Mecanismo para Controle de Erros e de Seqüência.
  - *Garantia de reconstrução do stream original (sem erros) e na mesma ordem em que a origem enviou.*
- Mecanismo para Controle de Fluxo
  - *Regula a taxa de transmissão do nó transmissor para evitar descartes pelo nó receptor.*
- Comunicação *Full Duplex*
  - *Possibilidade de envio e recepção de dados simultâneos.*

# Sockets em TCP



# Funcionamento do TCP

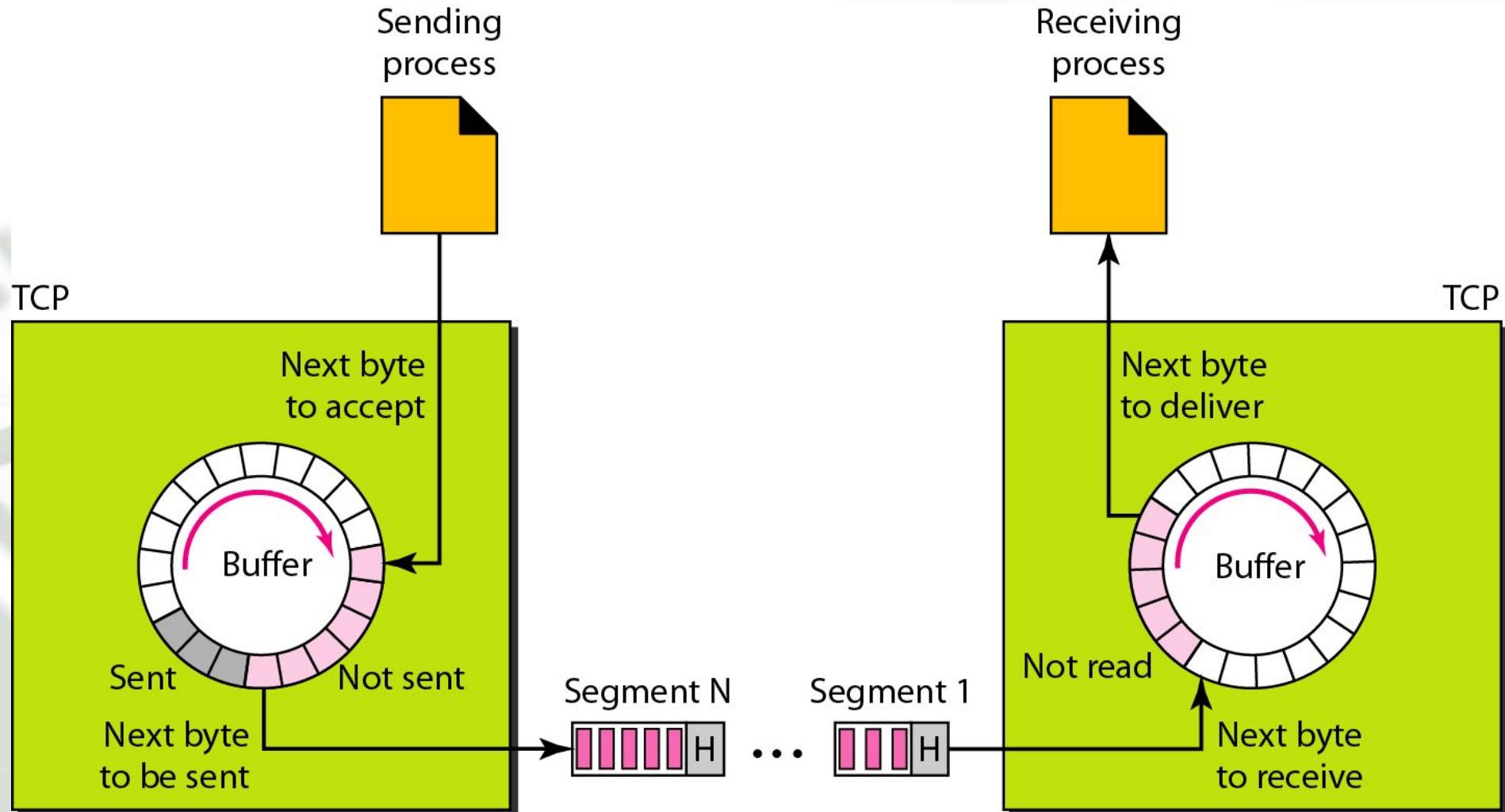
- O TCP, ao contrário do IP, envia dados na forma de um **fluxo de bytes**.
- Essa característica cria um ambiente no qual os dois processos parecem estar conectados por meio de um “**canal imaginário**”.



# Funcionamento do TCP

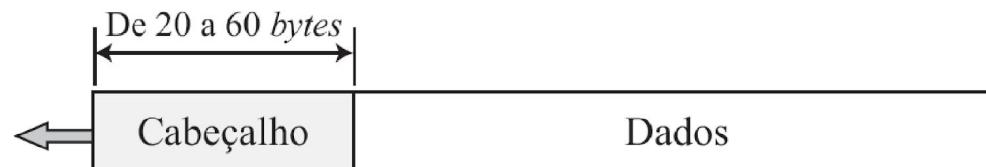
- Como a **taxa de geração de fluxo de dados** nem sempre será a mesma **taxa de leitura e tratamento do fluxo** recebido, é necessário a criação de *buffers*.
- *Buffer* de Transmissão
- *Buffer* de Recepção

# Funcionamento do TCP

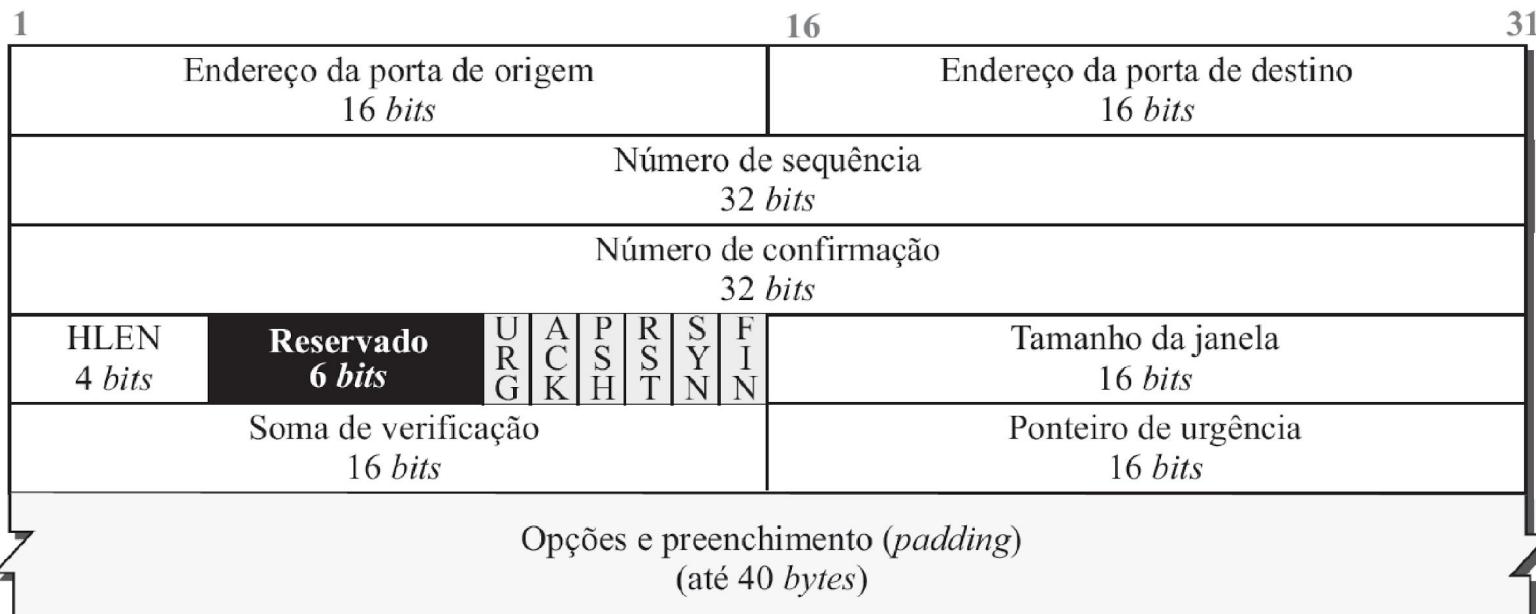


# Segmento TCP

## ■ Formato do Segmento TCP



(a) Segmento



(b) Cabeçalho

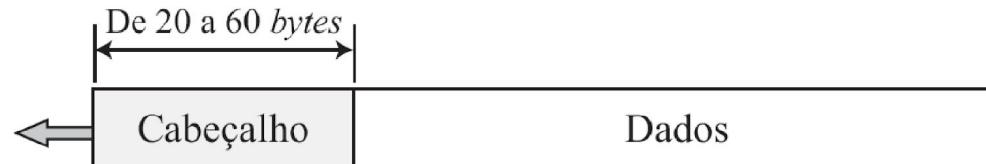
# *Piggybacking*

***Piggybacking*** é uma técnica utilizada para carregar informações de controle juntamente com os dados úteis de uma transmissão.

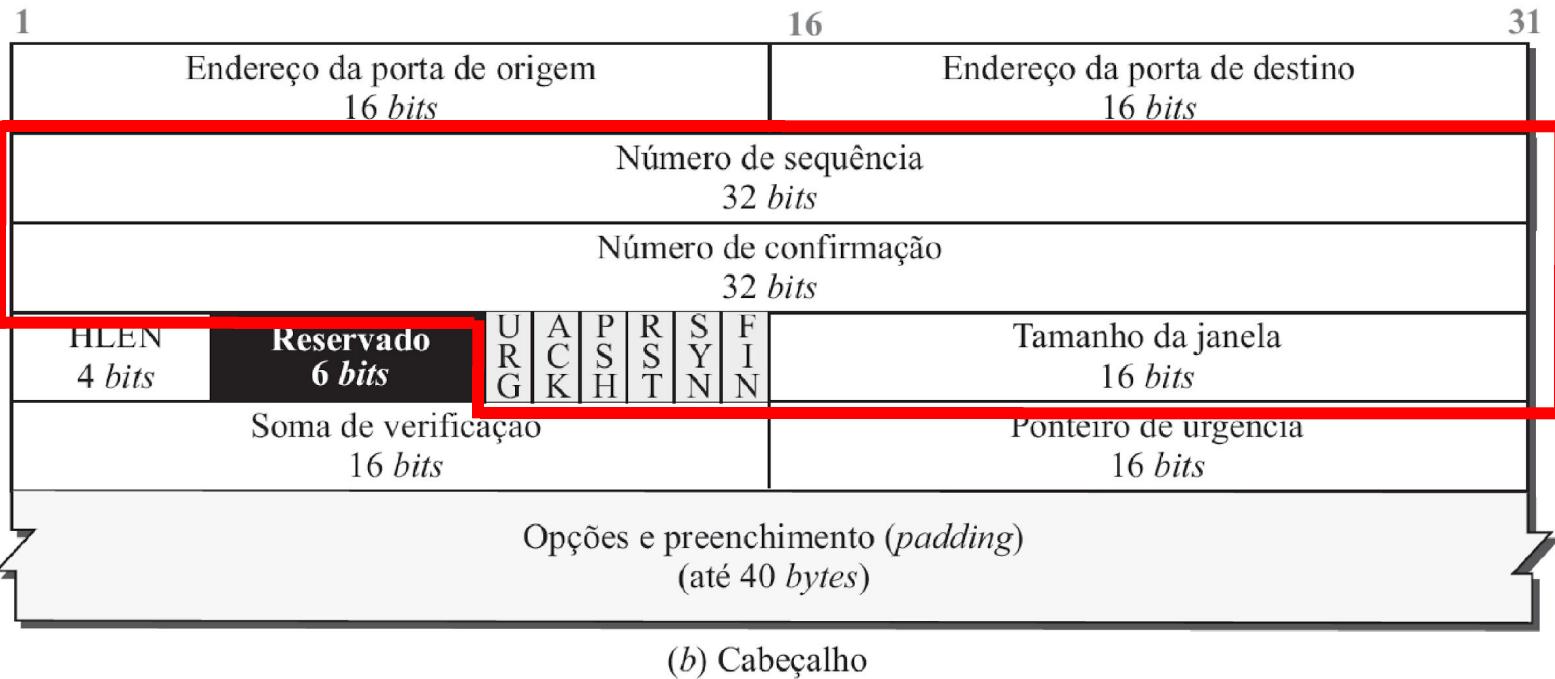
- O protocolo TCP aproveita campos do próprio cabeçalho para enviar informações de controle...
  - Estabelecimento e encerramento de conexão.
  - Tamanho da janela de recepção.
  - Controle de fluxo.
  - Confirmação de recebimentos.

# Segmento TCP

## ■ Formato do Segmento TCP



(a) Segmento

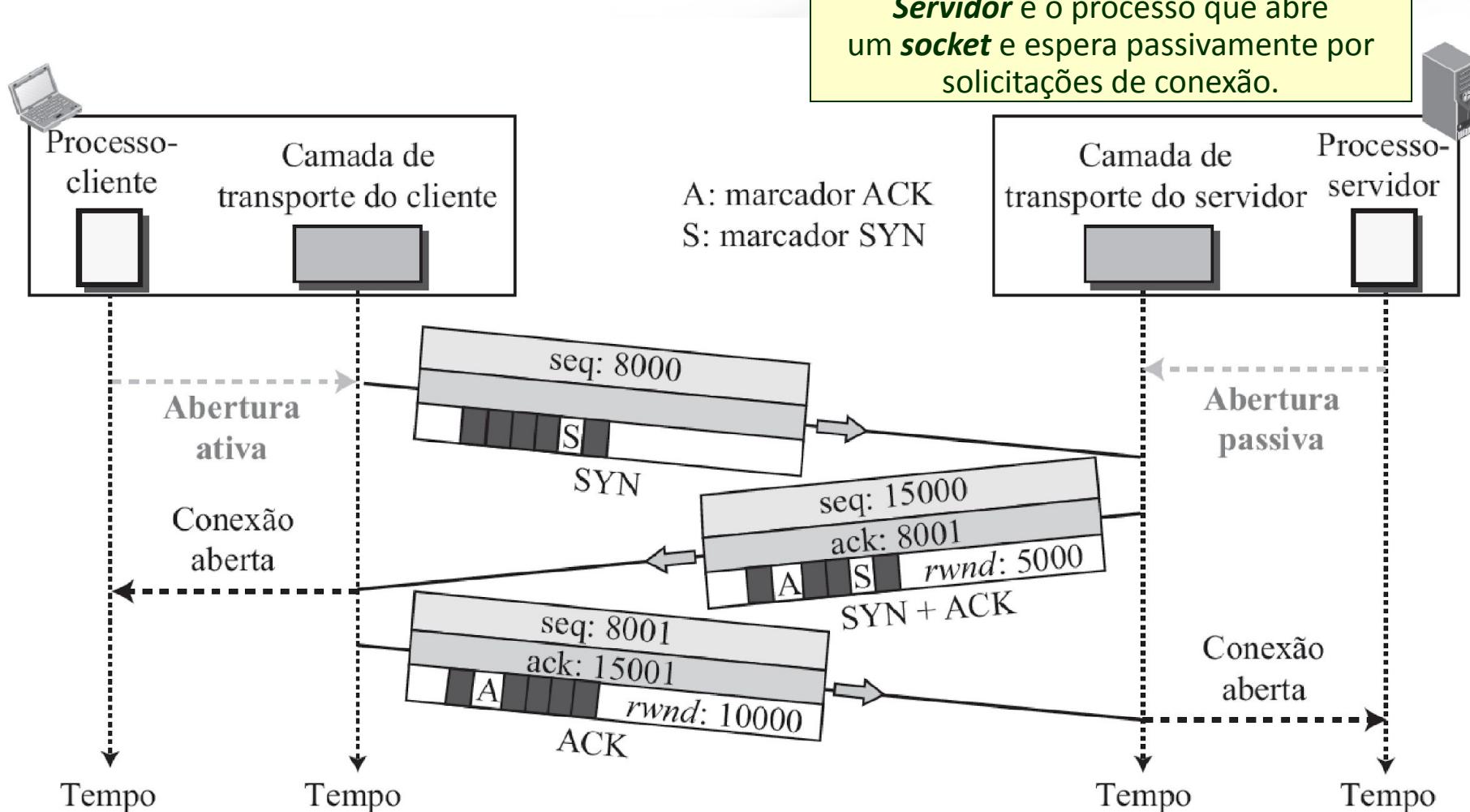


(b) Cabeçalho

# Estabelecimento de Conexão

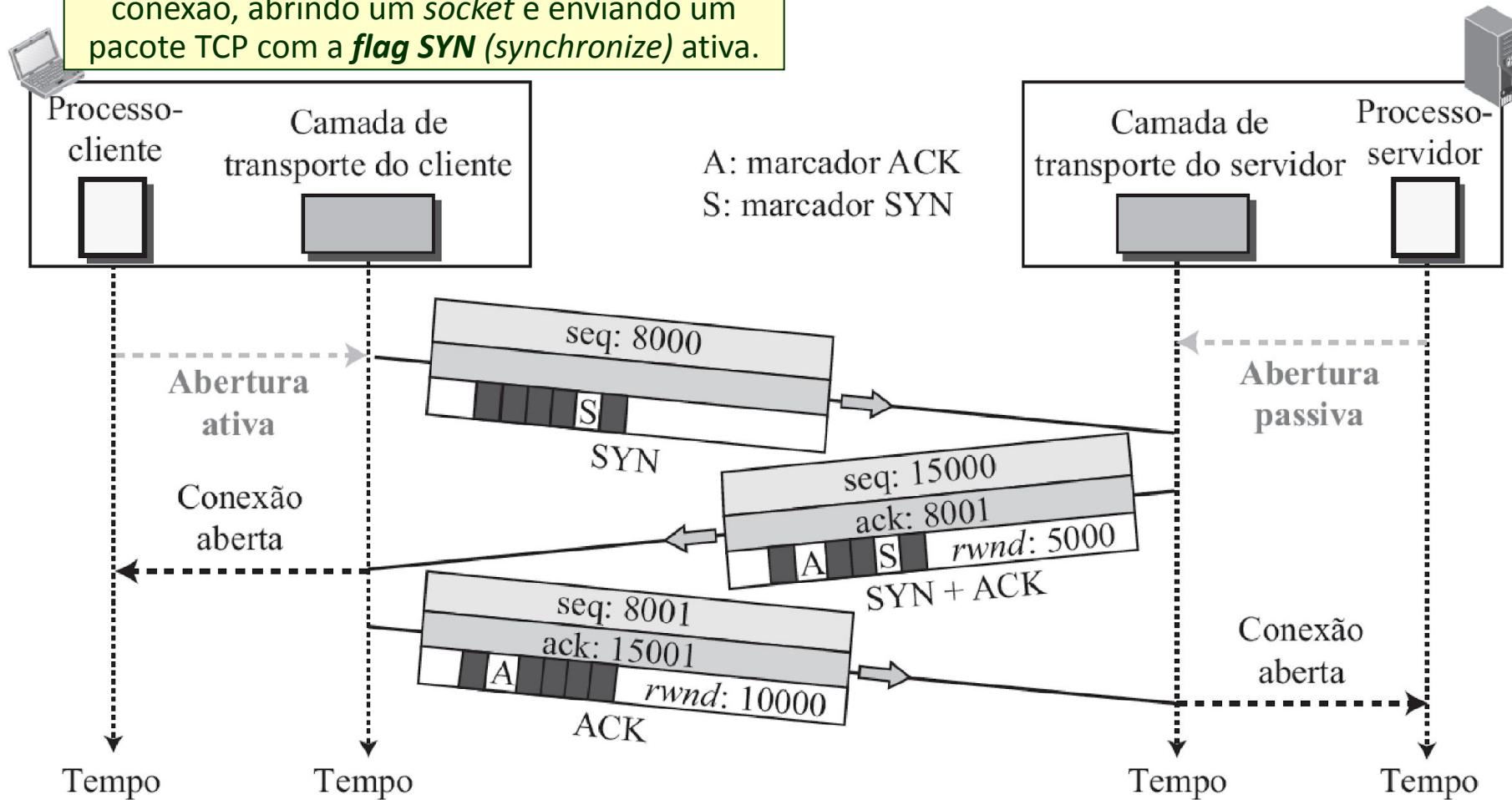
- O TCP é um **protocolo orientado à conexão**.
- A etapa de conexão é fundamental para determinar...
  - Se as partes estão disponíveis.
  - O tamanho dos buffers de transmissão e recepção.
  - A identificação dos bytes transmitidos (para evitar a falta ou a duplicação de dados recebidos).
- Afinal... O TCP é a garantia de uma comunicação confiável.

# Three-Way Handshake

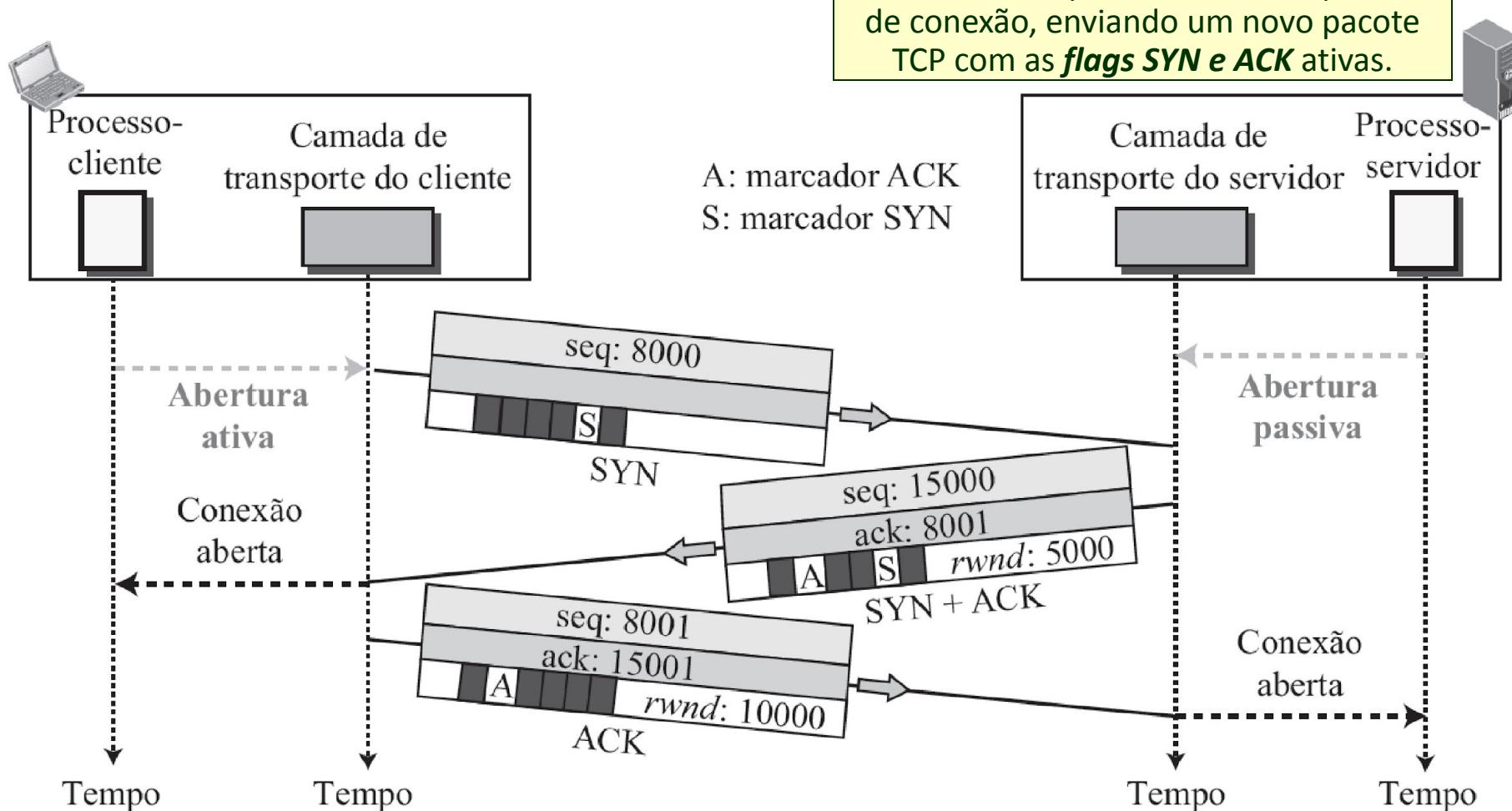


# Three-Way Handshake

**Cliente** é o processo que inicia a solicitação de conexão, abrindo um *socket* e enviando um pacote TCP com a *flag SYN* (*synchronize*) ativa.

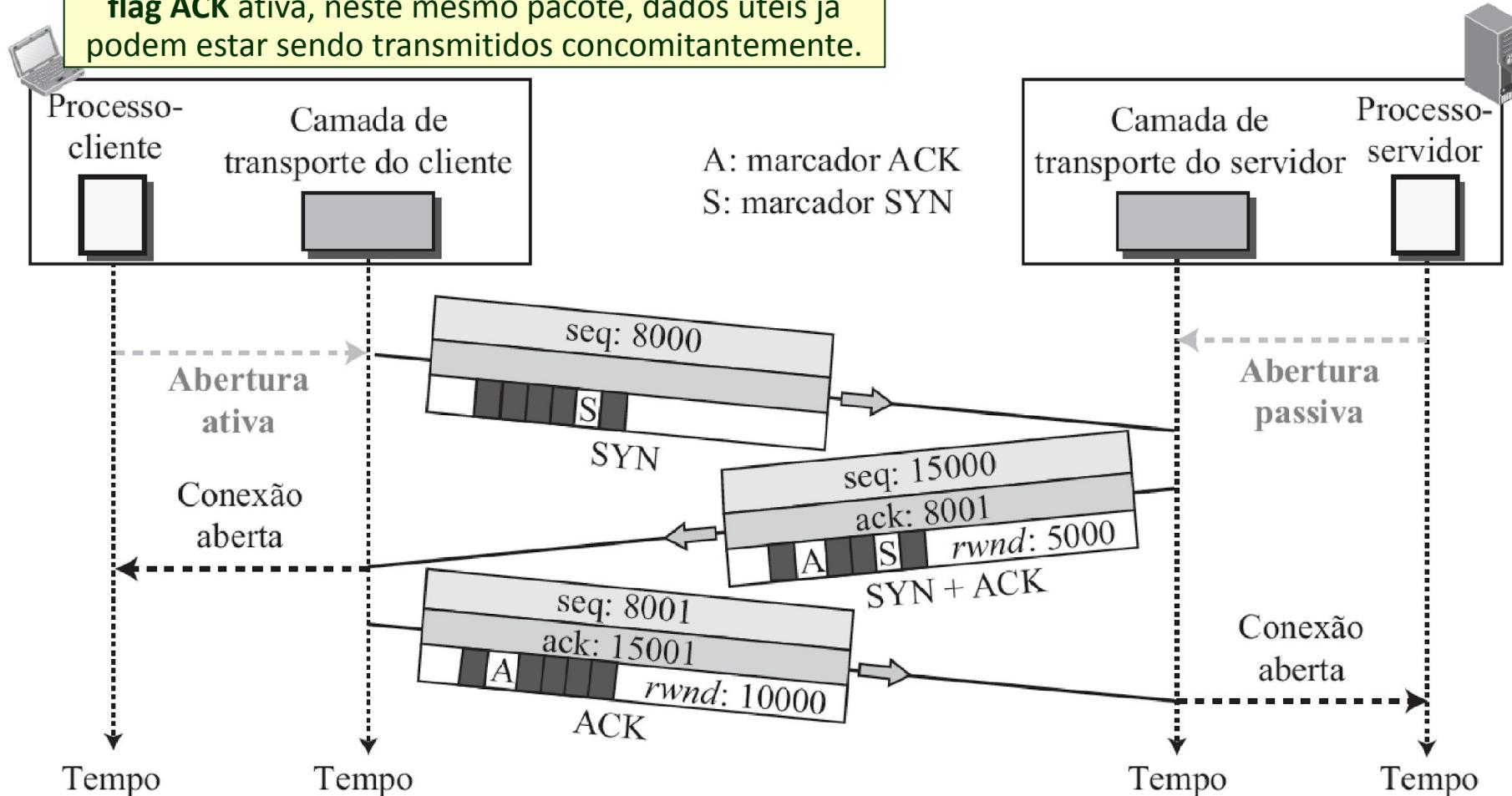


# Three-Way Handshake



# Three-Way Handshake

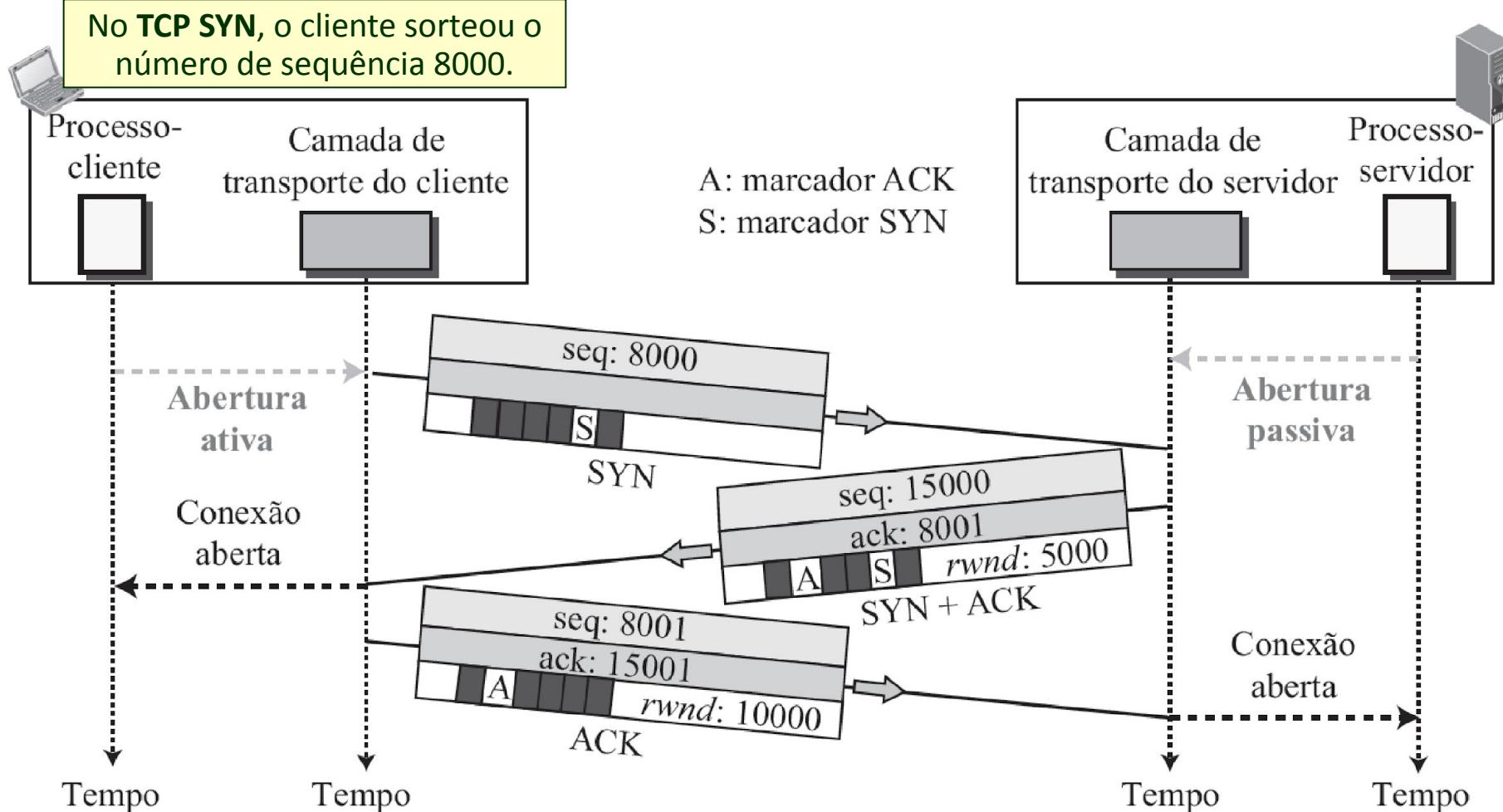
O estabelecimento da conexão é concluído após o processo cliente enviar um novo pacote TCP com a flag **ACK** ativa, neste mesmo pacote, dados úteis já podem estar sendo transmitidos concomitantemente.



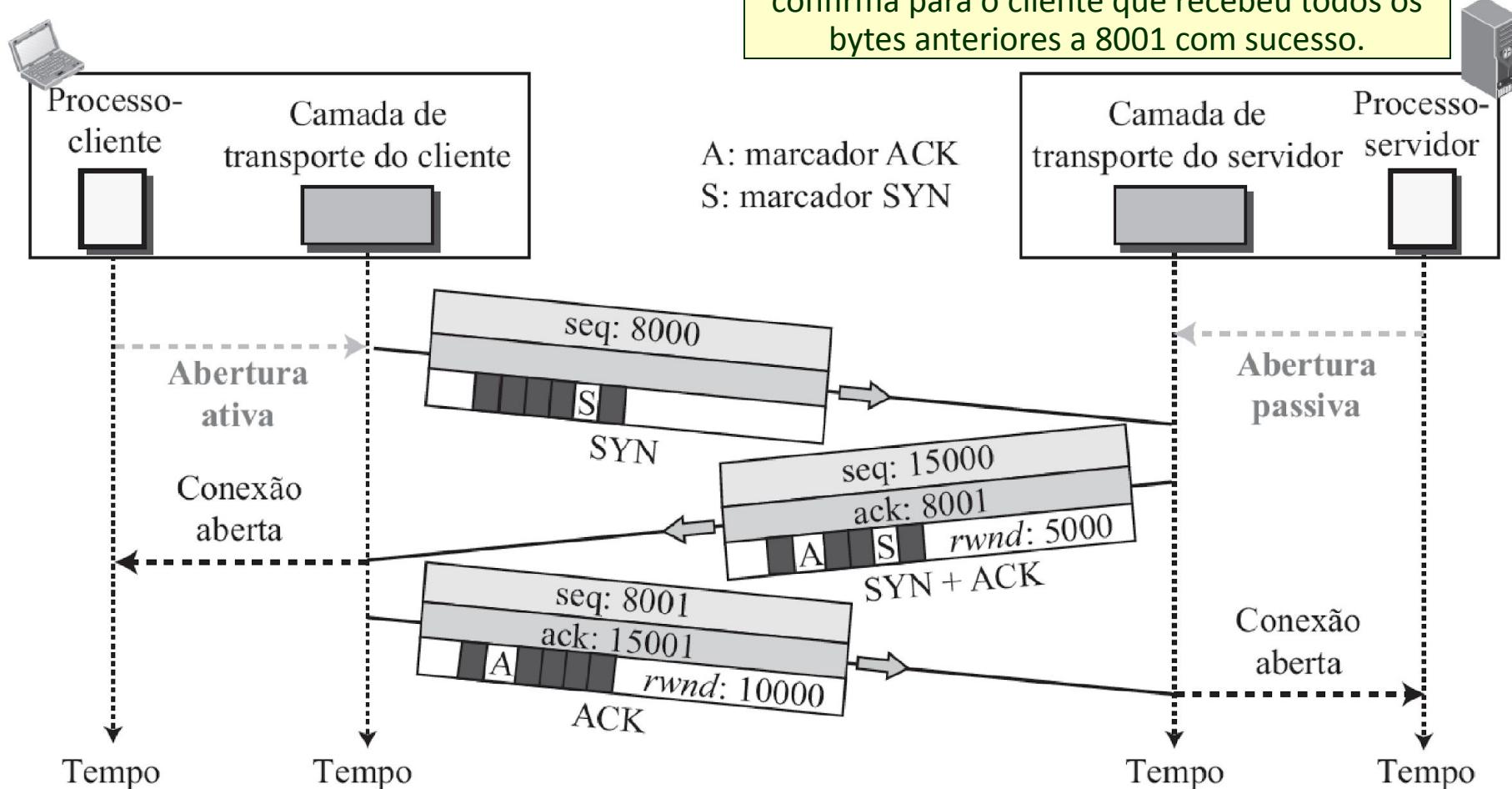
# Sequenciamento

- Para organizar e controlar a recepção do fluxo de dados, o TCP atribui uma numeração de controle para cada byte transmitido, e funciona assim...
  - Durante o estabelecimento da conexão, um **Número de Sequência** ( $0 \leftrightarrow 2^{32}-1$ ) é sorteado.
  - Este **Número de Sequência** indica que o primeiro byte da comunicação possui esse identificador, e os bytes seguintes serão identificados pelos valores subsequentes ao valor sorteado.
  - Após verificar a integridade do conteúdo, o receptor enviará um pacote solicitando o próximo byte da transmissão, ou seja... **Número de Sequência + 1**.

# Sequenciamento

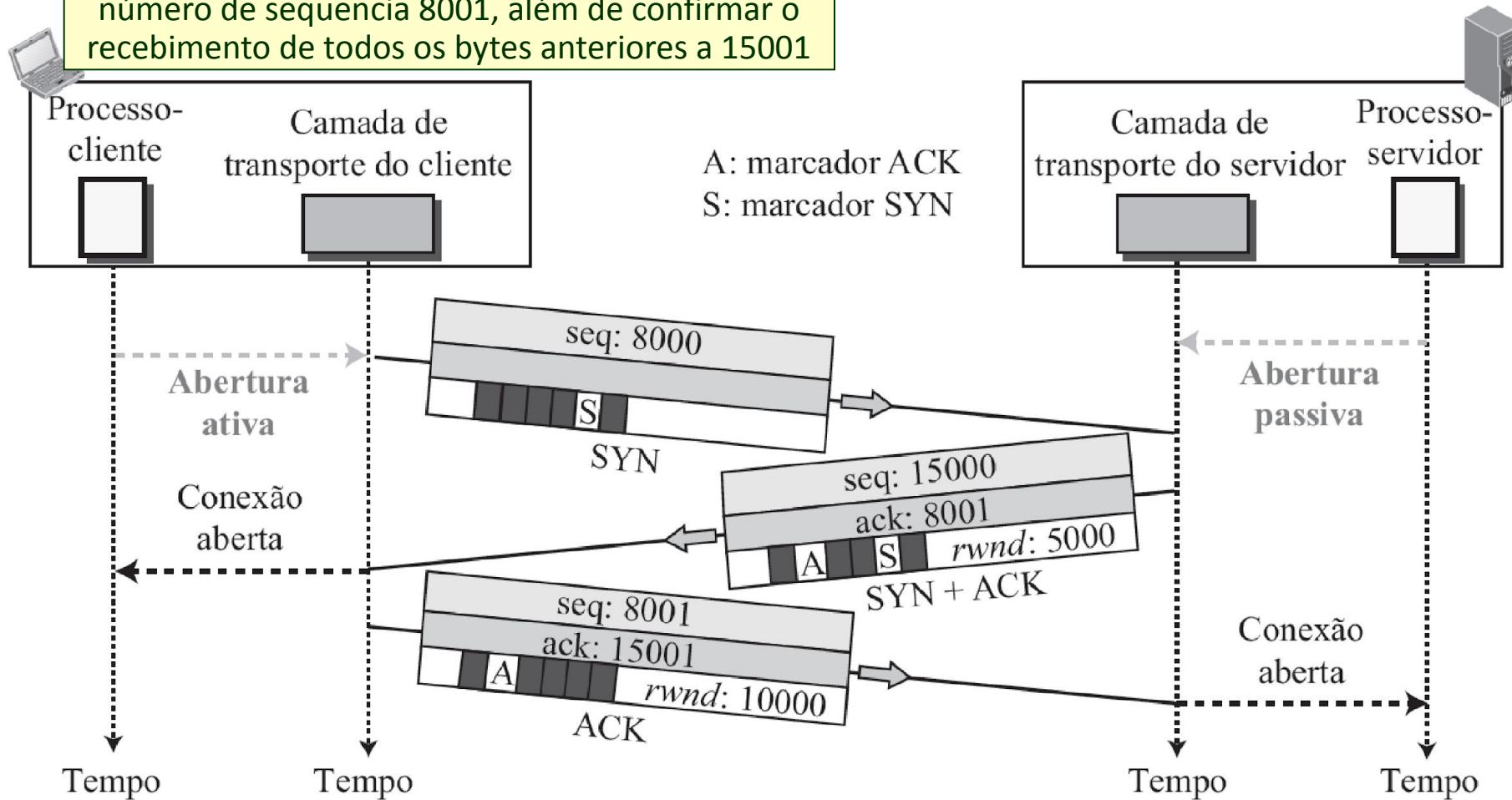


# Sequenciamento

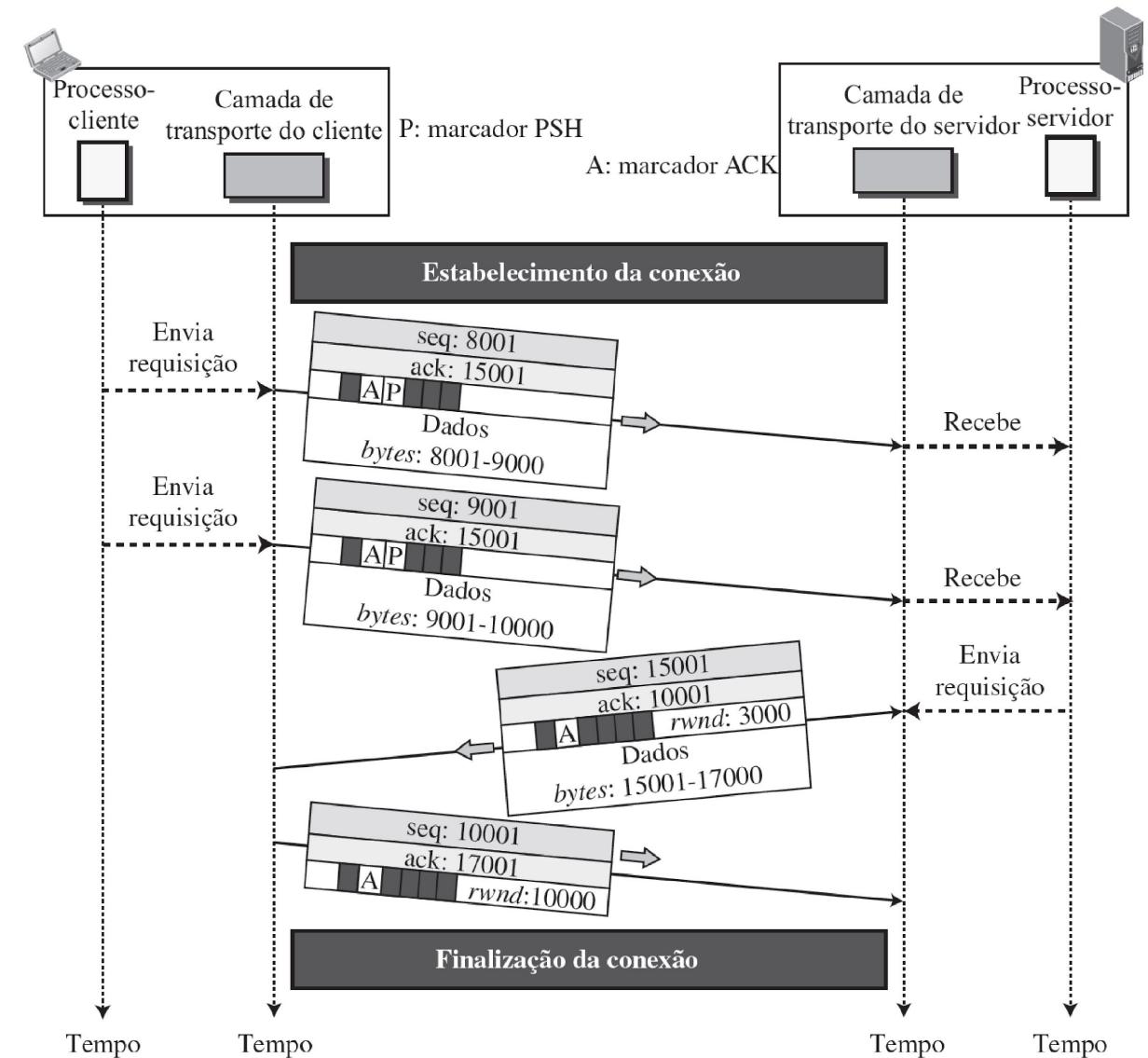


# Sequenciamento

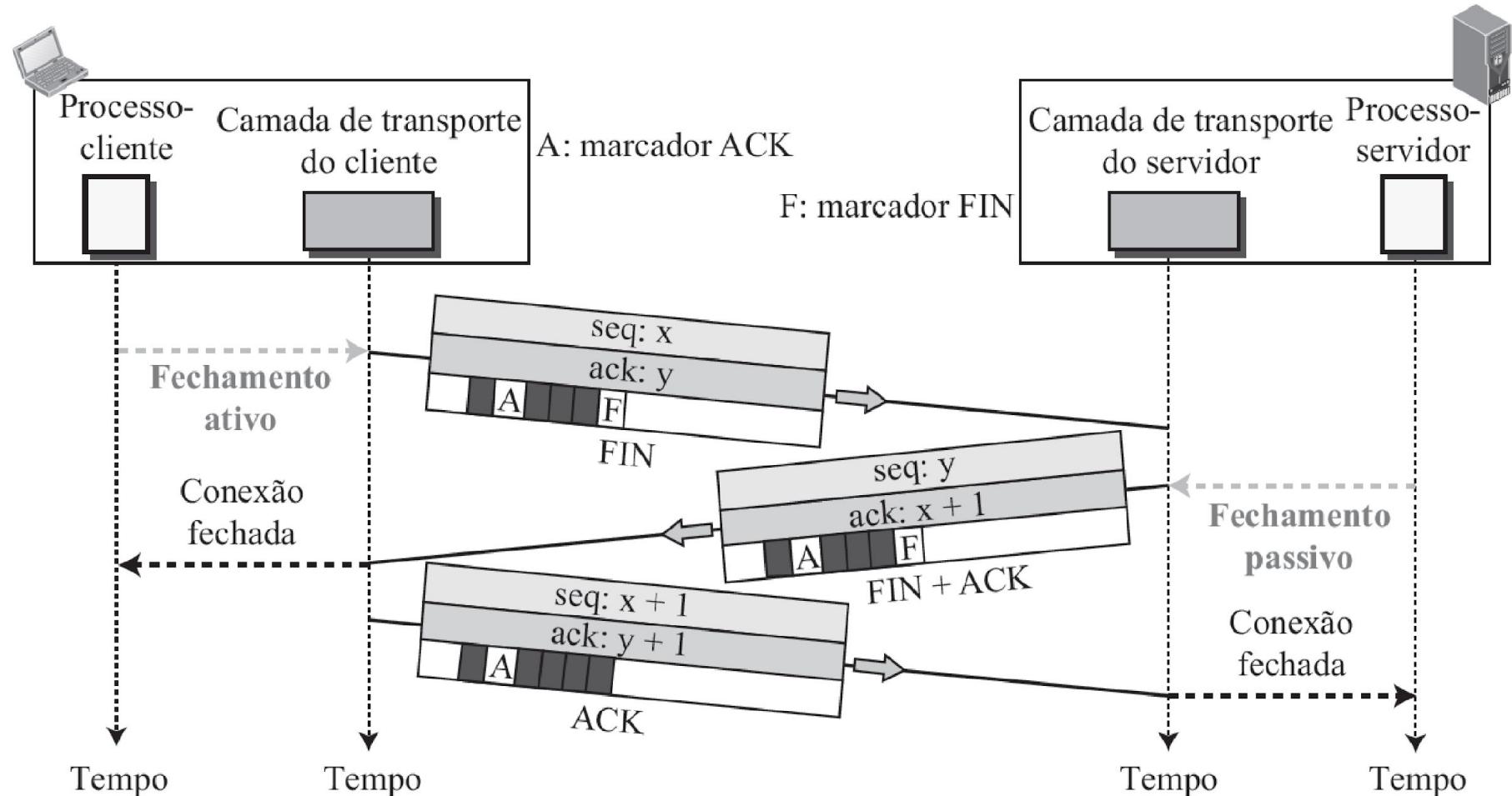
O cliente envia os primeiros dados úteis da comunicação, identificando o primeiro byte pelo número de sequencia 8001, além de confirmar o recebimento de todos os bytes anteriores a 15001



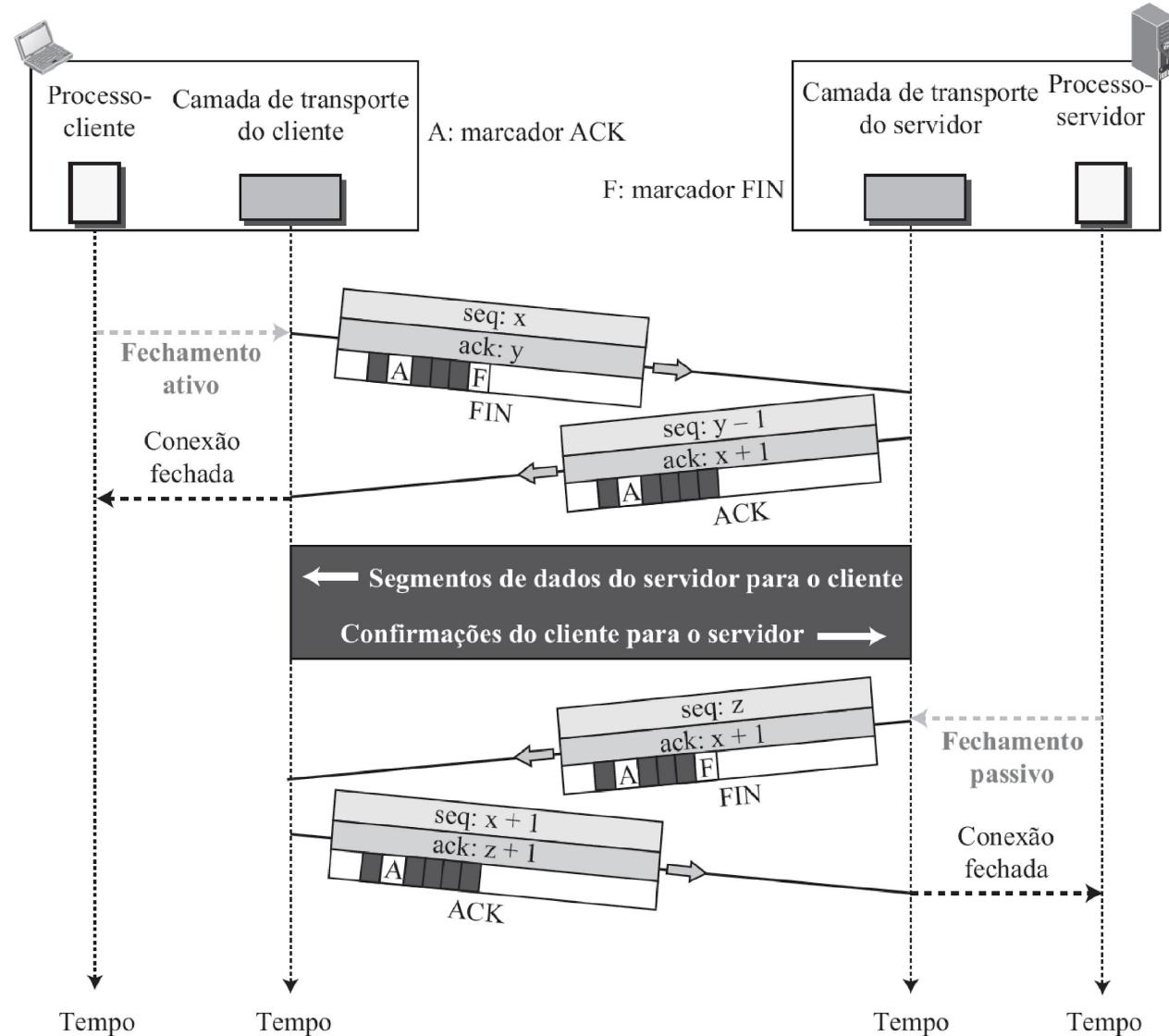
# Transferência de Dados



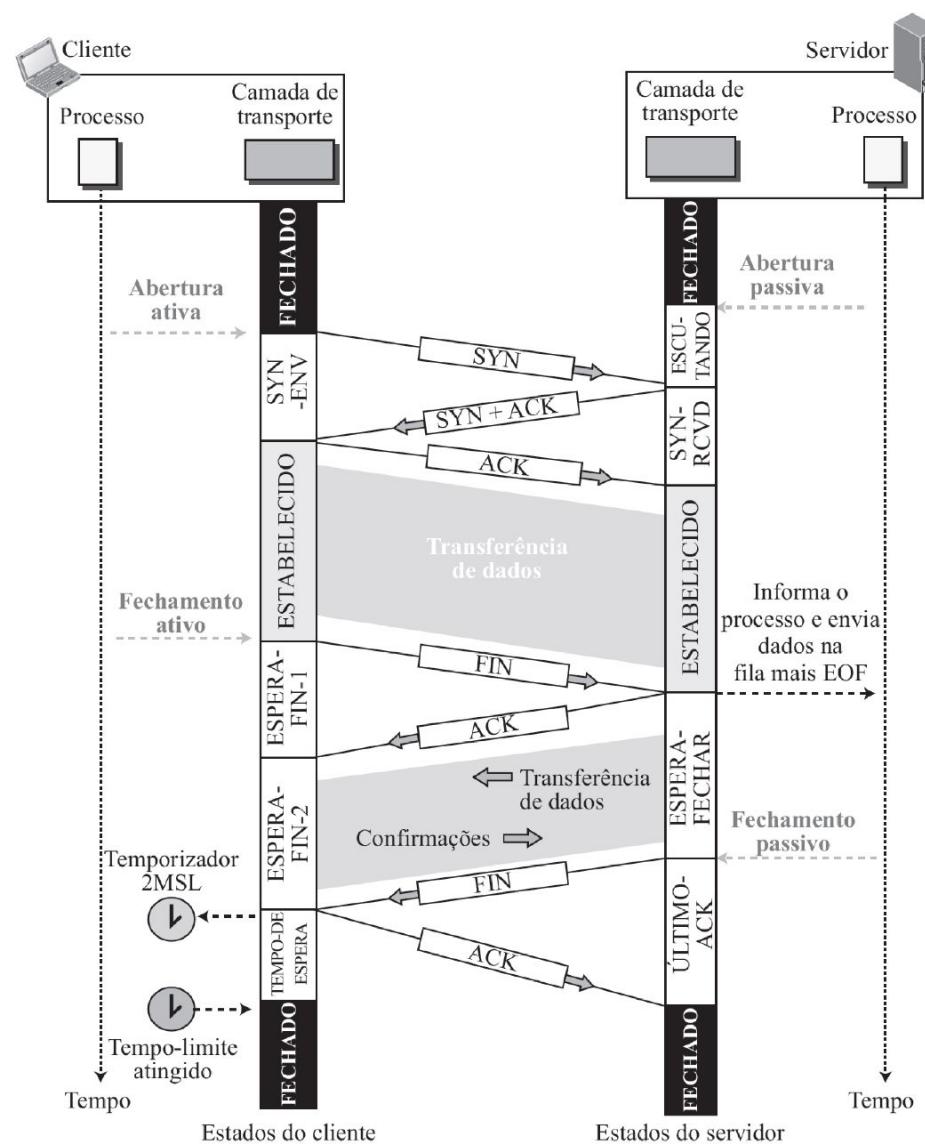
# Encerramento de Conexão



# Semi-Encerramento de Conexão

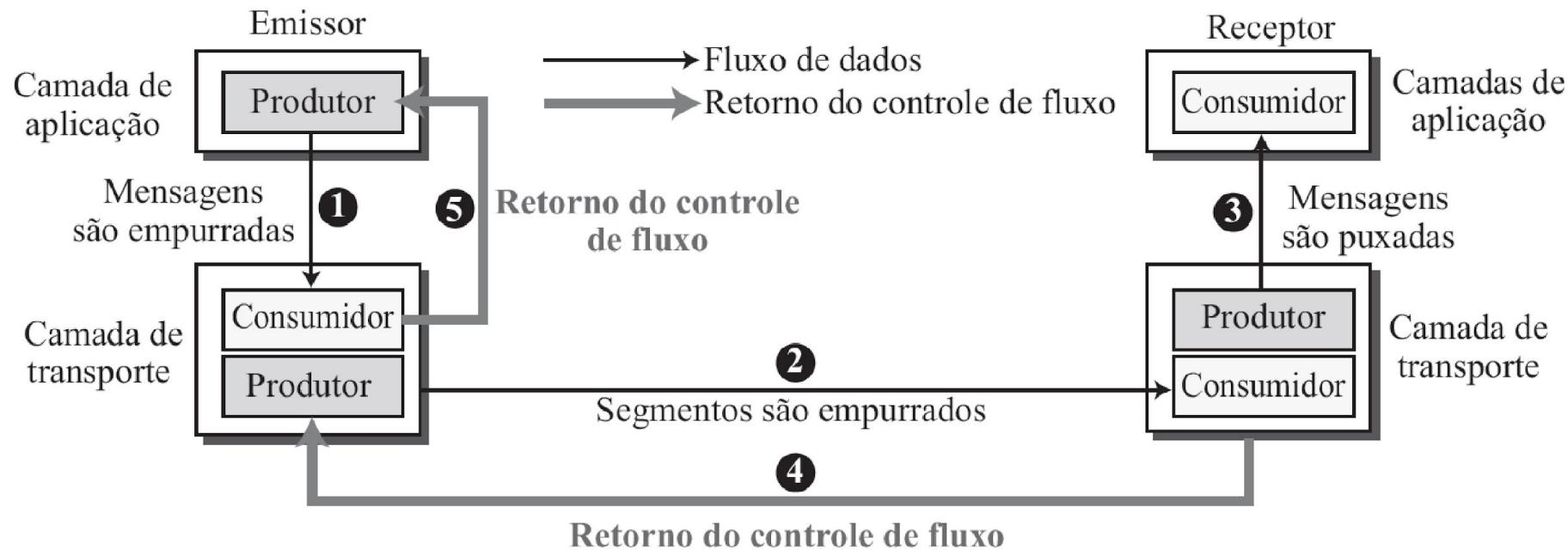


# TCP Timeline



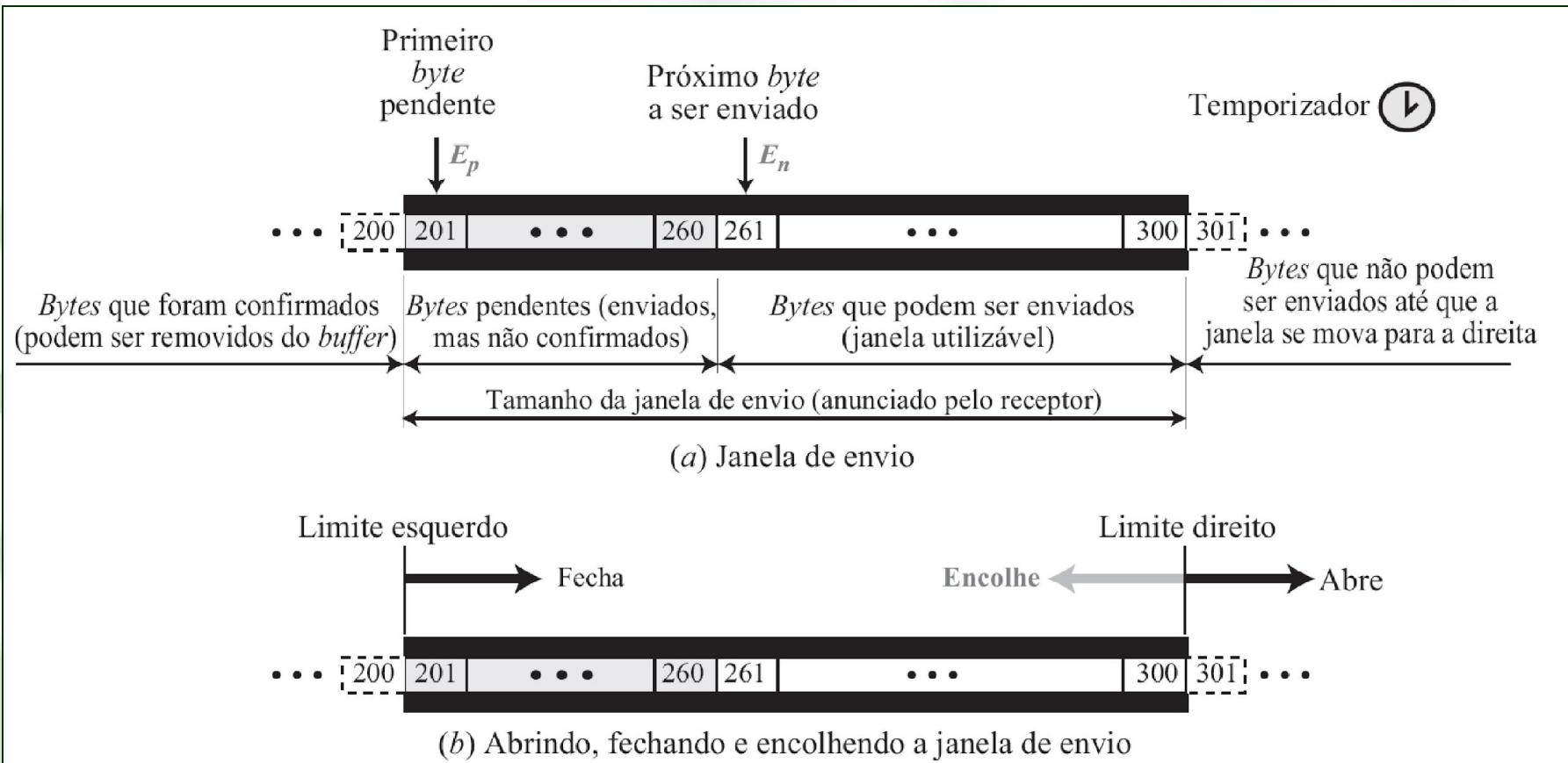
# Controle de Fluxo

## ■ Fluxo de dados padrão da comunicação TCP:



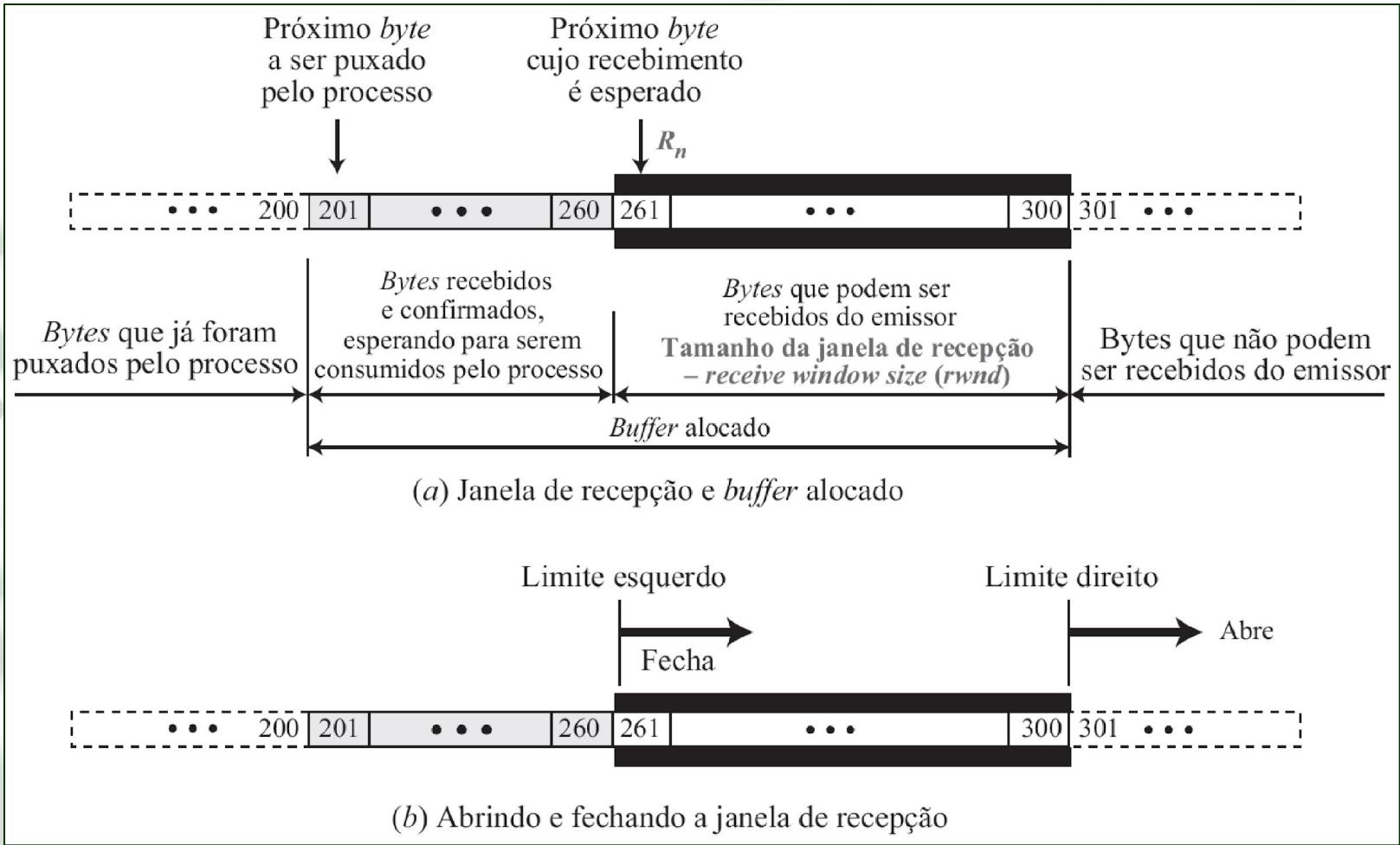
# Controle de Fluxo

## ■ Janela Deslizante (Processo Transmissor)



# Controle de Fluxo

## ■ Janela Deslizante (Processo Receptor)

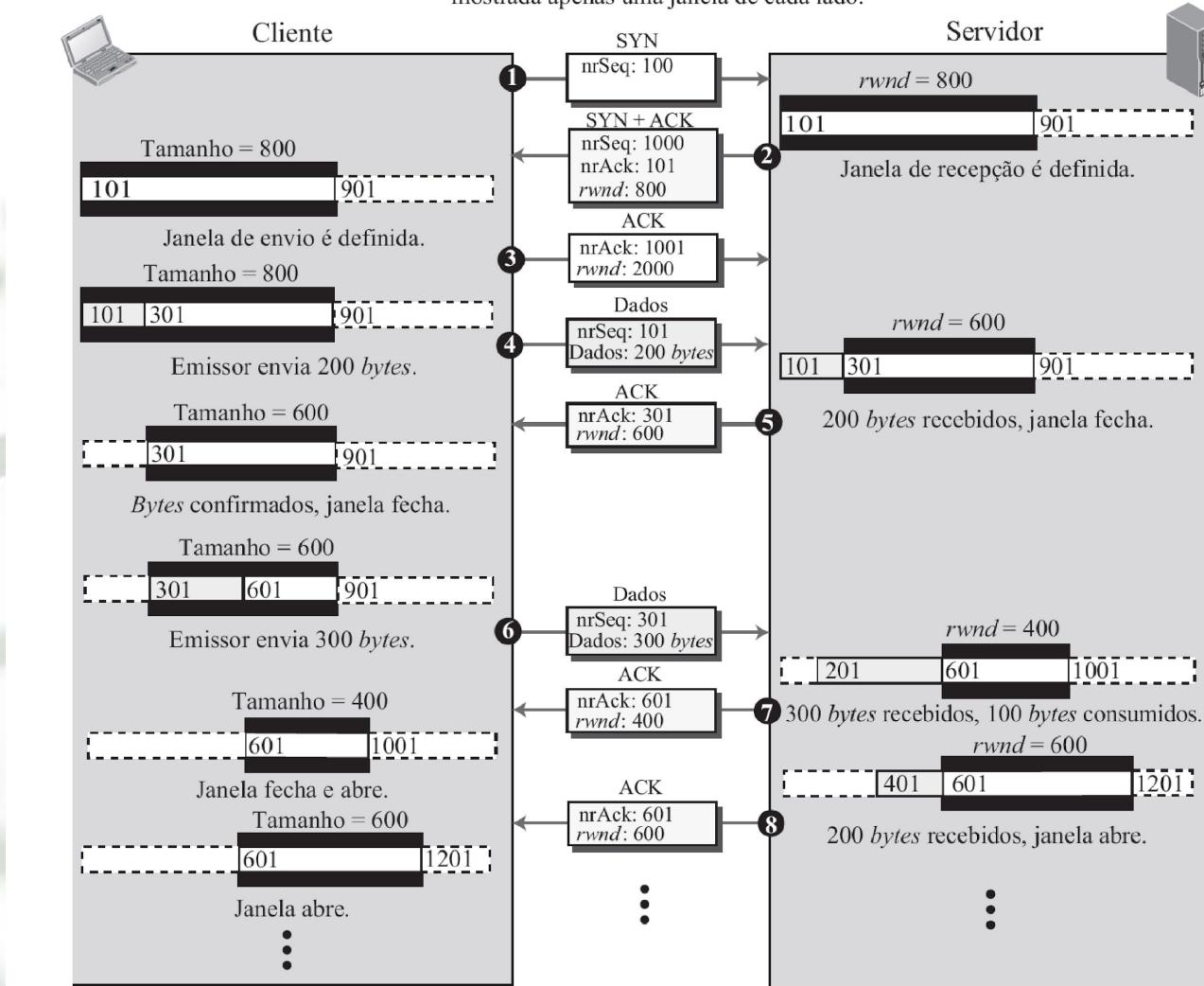


# Controle de Fluxo

- A confirmação do recebimento dos segmentos é feita através do envio de ACKs (*acknowledgements*).
- O protocolo TCP aproveita campos do próprio cabeçalho para enviar essas informações de controle (*piggybacking*).
- Desta forma, os ACKs “pegam carona” em segmentos que transportam dados úteis.

# Janela Deslizante

**Nota:** Consideramos uma comunicação puramente unidirecional do cliente para o servidor. Consequentemente, é mostrada apenas uma janela de cada lado.

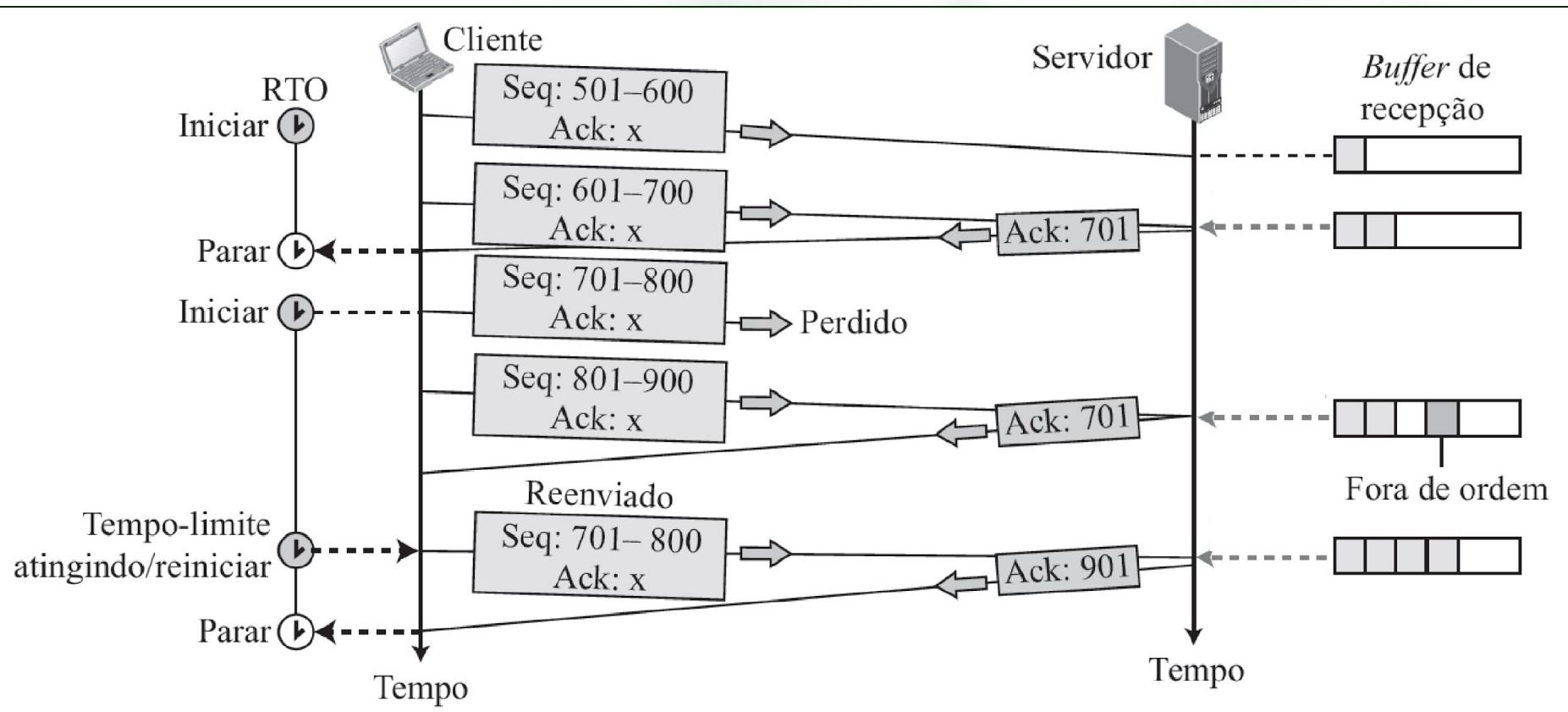


# Tratamento de Erros

- Após o envio de um segmento TCP, um temporizador RTO (*Retransmission TimeOut*) é iniciado.
  - Caso RTO expire antes que seja recebido o ACK correspondente, o segmento é retransmitido pelo nó T(x).
  - RTO é dinamicamente calculado em função do RTT (*Round Trip Time*) da conexão.

# Tratamento de Erros

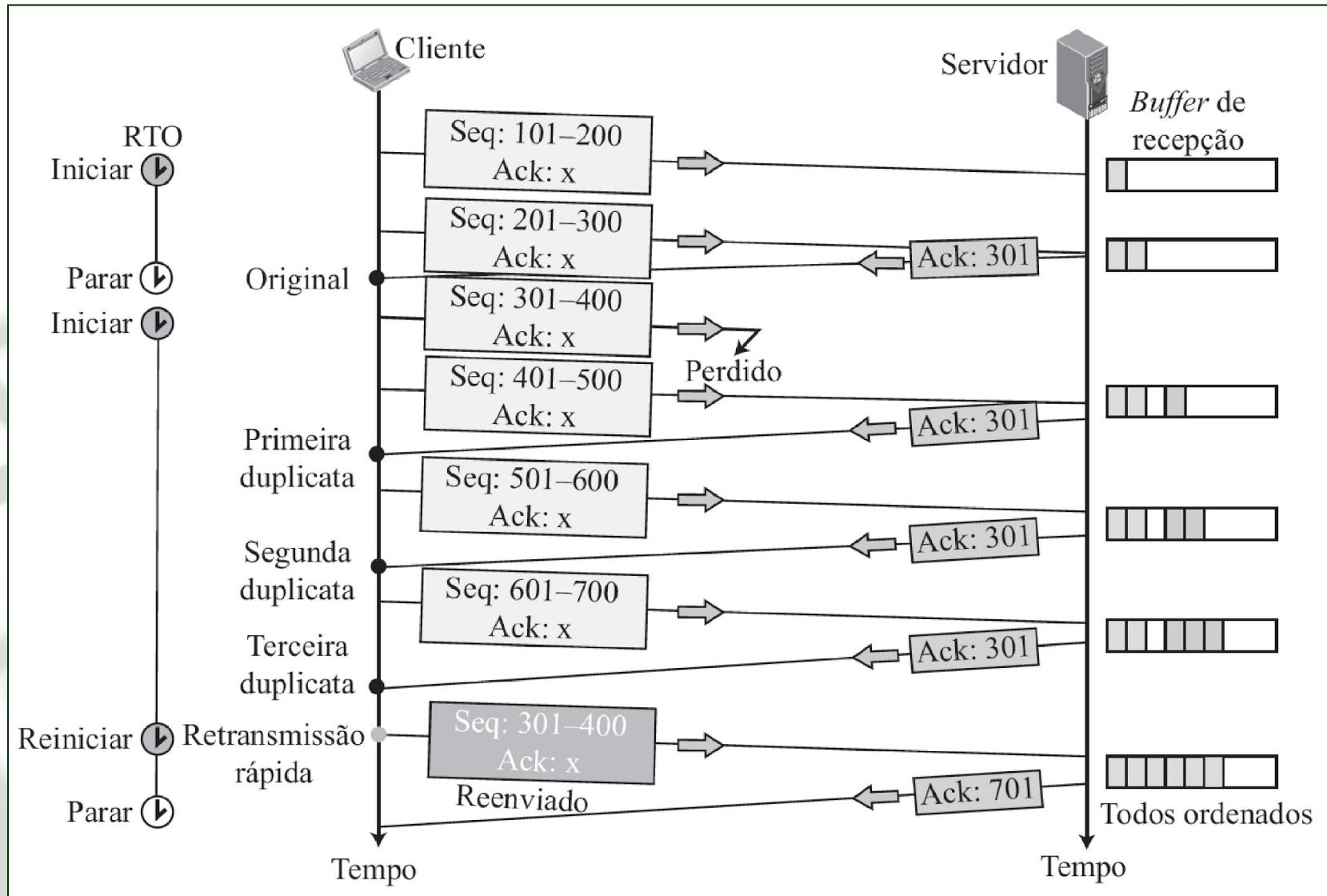
## ■ Fluxo normal...



# Tratamento de Erros

- Na **Retransmissão Rápida**, a retransmissão de um segmento acontece se forem recebidos três ACKs com o mesmo valor.
  - Neste caso, o RTO é ignorado e a retransmissão acontece **imediatamente**.

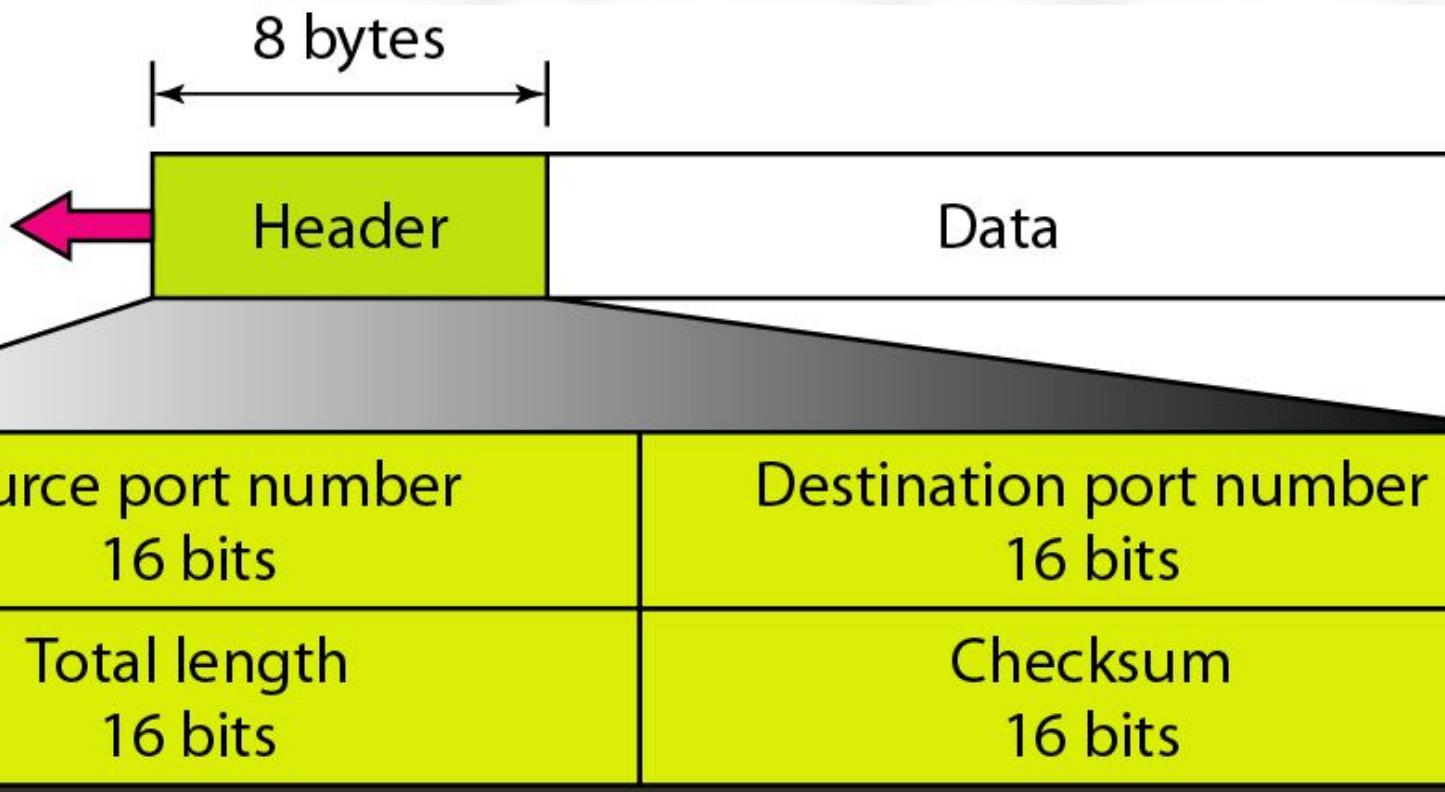
# Tratamento de Erros



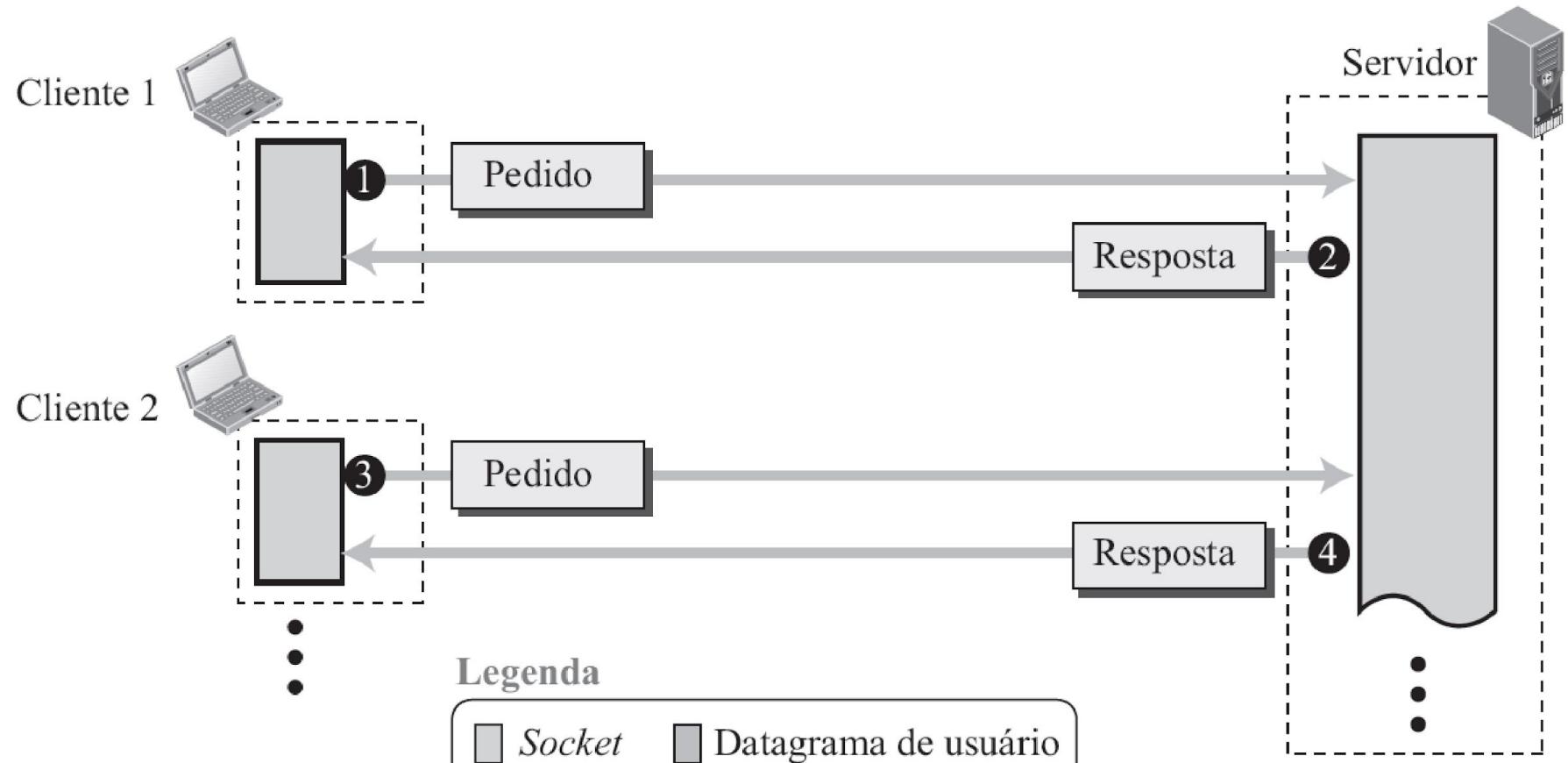
# Protocolo UDP

- O conjunto de protocolos TCP/IP admite um protocolo de transporte sem conexões.
- O UDP oferece um meio para as aplicações enviem datagramas IP encapsulados sem que seja necessário estabelecer uma conexão.
- Serviço de transporte ***best-effort***
  - Melhor esforço.

# Cabeçalho UDP



# Sockets em UDP



# Características do UDP

- Sem conexão
- Não confiável
- Transmite mensagens (chamado de datagramas do usuário)
- Não fornece verificação de software para a entrega da mensagem (não é confiável)
- Não regrupa as mensagens de entrada
- Não usa confirmações
- Não fornece controle de fluxo

# Características do UDP

## Recomendado para aplicações de mídias contínuas (voz, vídeo)

- *Tolerantes de perdas*
- *Sensíveis à taxa de transmissão*

## Outros usos de UDP:

- *DNS (nomes de domínio)*
- *SNMP (gerenciamento)*

## Transferência confiável com UDP:

- *Se necessário, incluir na aplicação...*

### Por quê existe o UDP?

- elimina estabelecimento de conexão (o que pode causar retardo).
- simples: não se mantém “estado” da conexão no remetente/receptor.
- pequeno cabeçalho de segmento.
- sem controle de congestionamento: UDP pode transmitir o mais rápido possível.

# Benefícios do UDP

- UDP ideal para algumas aplicações onde a velocidade é mais crítica do que a confiabilidade.
- Aplicações sensíveis a atrasos na rede, mas poucos sensíveis a perdas de pacotes, como jogos de computadores, áudio e vídeo.
- O UDP suporta *multicasting*. Caso esses recursos sejam necessários, o UDP deverá necessariamente ser utilizado.
- O UDP não perde tempo com criação ou encerramento de conexões. Durante uma comunicação o UDP pode enviar apenas 1 pacotes, enquanto no TCP esse número pode ser superior a 10. Por isso, aplicações que encaixam num modelo de pergunta-resposta também são fortes candidatas a usar UDP.