Redes de Computadores I

Camada de Transporte

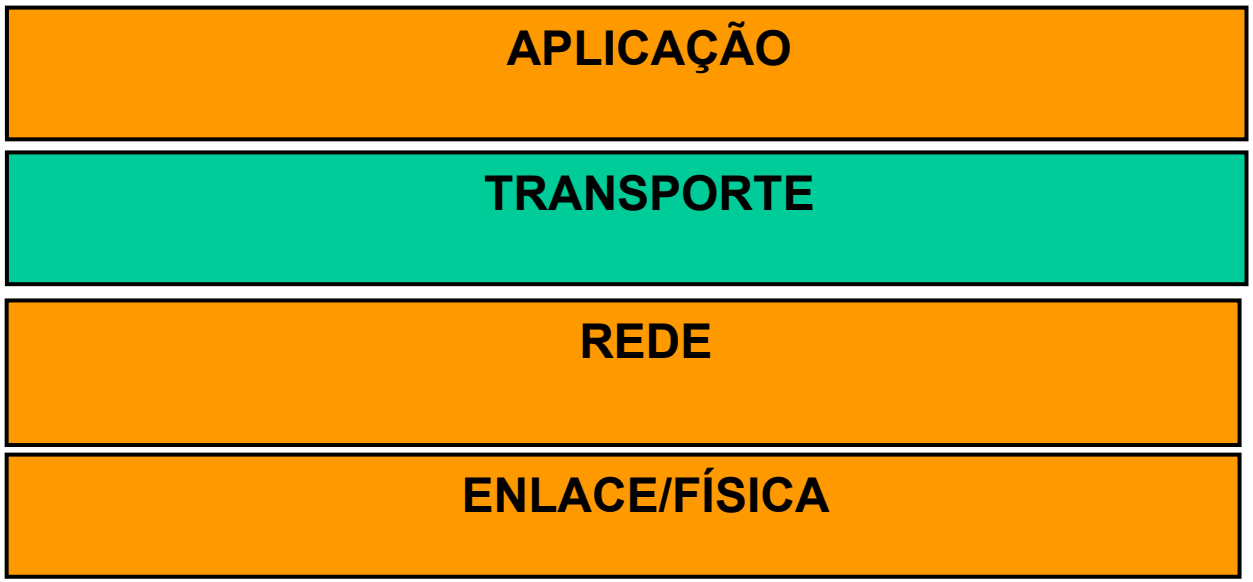
Prof. Ricardo Couto A. da Rocha
rcarocha@ufg.br
UFG – Regional de Catalão

Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Camada de Transporte

Arquitetura TCP/IP



Características

- Responsável pelo transporte dos dados do serviço de transporte
- Comunicação fim-a-fim entre aplicações
- Protocolos orientado a conexão:
  - Controle do fluxo de dados fim-a-fim
  - Controle de erro ➔ detecção e correção de erro fim-a-fim
  - Controle de seqüência (sequenciação)
  - Divisão de mensagens em segmentos

Características

- Mecanismos de identificação de processos
- O nível de transporte oferece:
  - serviços sem conexão (datagramas)
  - serviços orientados à conexão (circuitos virtuais)

Características

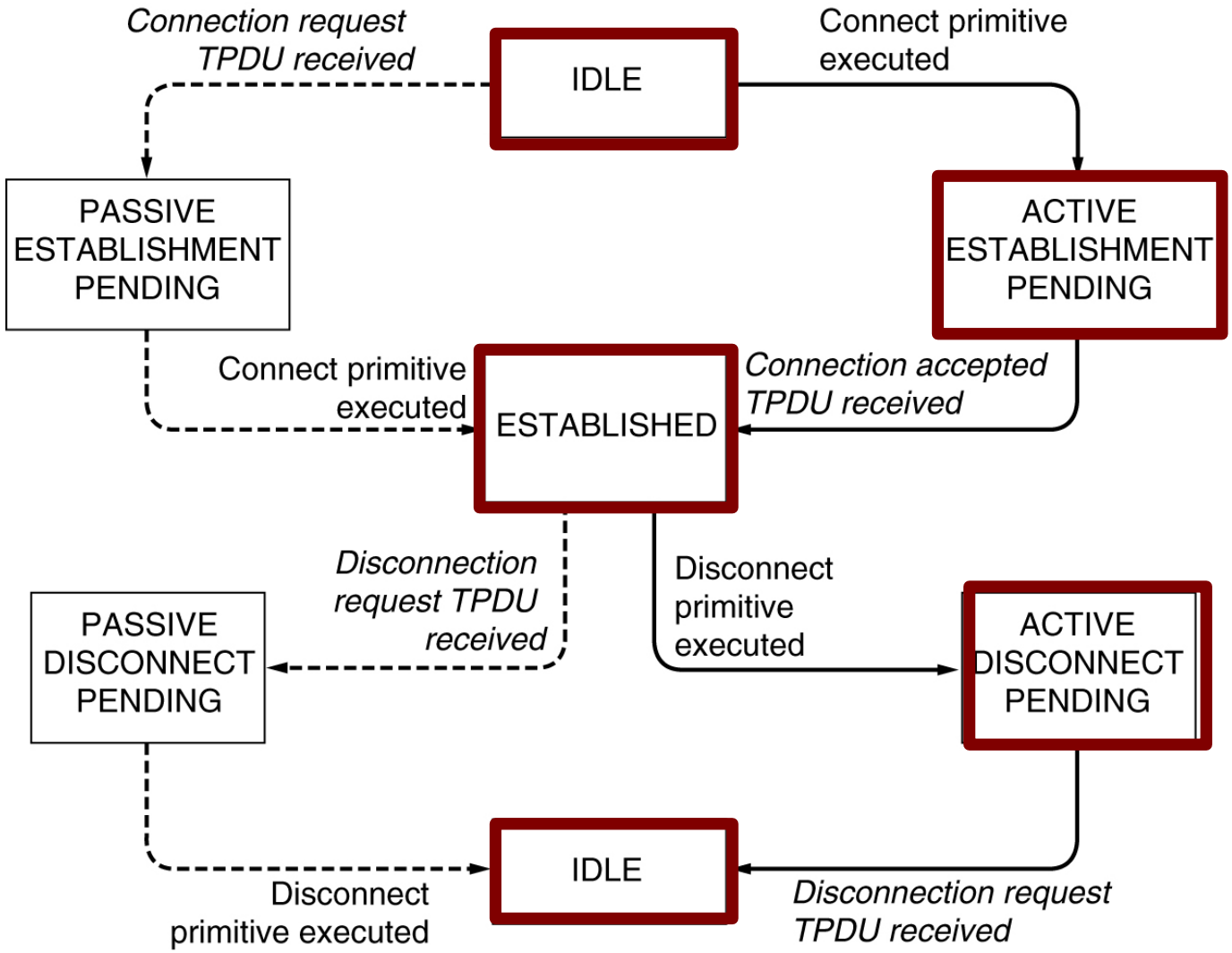
- serviços sem conexão (datagramas)
  - Exemplo: UDP
  - mapeia os pedidos de transmissão para serviços oferecidos pela camada de rede
  - Apresenta baixo overhead
  - Não implementa nenhum mecanismo de confiabilidade
    - ~~Detecção de erro~~
    - ~~Controle de fluxo~~
    - ~~Sequenciação~~

Características

- Serviços orientados à conexão
  - Exemplo: TCP
  - Etapas:
    - Estabelecimento da conexão
    - Transferência de dados
    - Encerramento da conexão

Características

- Ciclo de vida simplificado do TCP



Características - Etapas

- Etapas dos Serviços orientados à conexão
  - Estabelecimento da conexão
    - **Three-Way Handshake**
    - Define o tamanho da janela de transmissão
    - Define número de seqüência
  - Transferência de dados
    - Detecção e correção de erros
    - Sequenciação e ordenação dos pacotes
    - Controle de fluxo → sincronização da velocidade
  - Encerramento de conexão
    - Protocolo que provê garantias que os dois lados estão prontos para terminar a conversa (ninguém ainda tem dados a transmitir)
    - Deve considerar encerramento por falha na conexão

Características

- Protocolos de Transporte
  - Mesmas técnicas do nível de transporte agora aplicadas fim-a-fim
  - Controle de fluxo
    - Stop-and-Wait (bit alternado)
    - Sliding window (Janela deslizante)
  - Controle de erro
    - ARQ Automatic Repeat Request (Stop and Wait = bit alternado na camada de transporte)
    - janela n com retransmissão integral - Go-Back-N
    - janela n com retransmissão seletiva - Selective Repeat

Roteiro

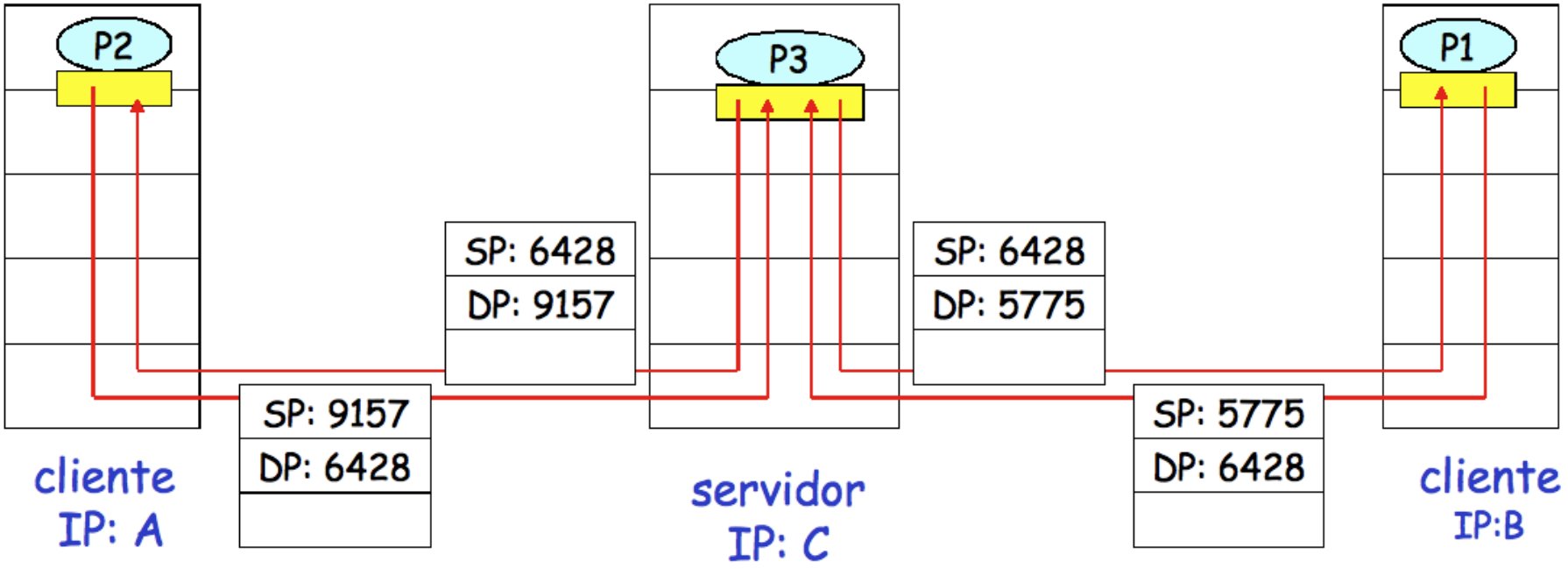
- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Multiplexação

- **Multiplexação** → Mecanismo com o qual um meio de comunicação é compartilhado entre várias conversações concorrentes/paralelas
- Na camada de transporte
  - Define como é possível uma ou mais aplicações se comunicarem simultaneamente com vários processos em outras estações
  - Sessão de comunicação na camada de transporte: conexão, em protocolos confiável
  - Especifica o identificador único de uma conexão

Demux não orientada a conexão

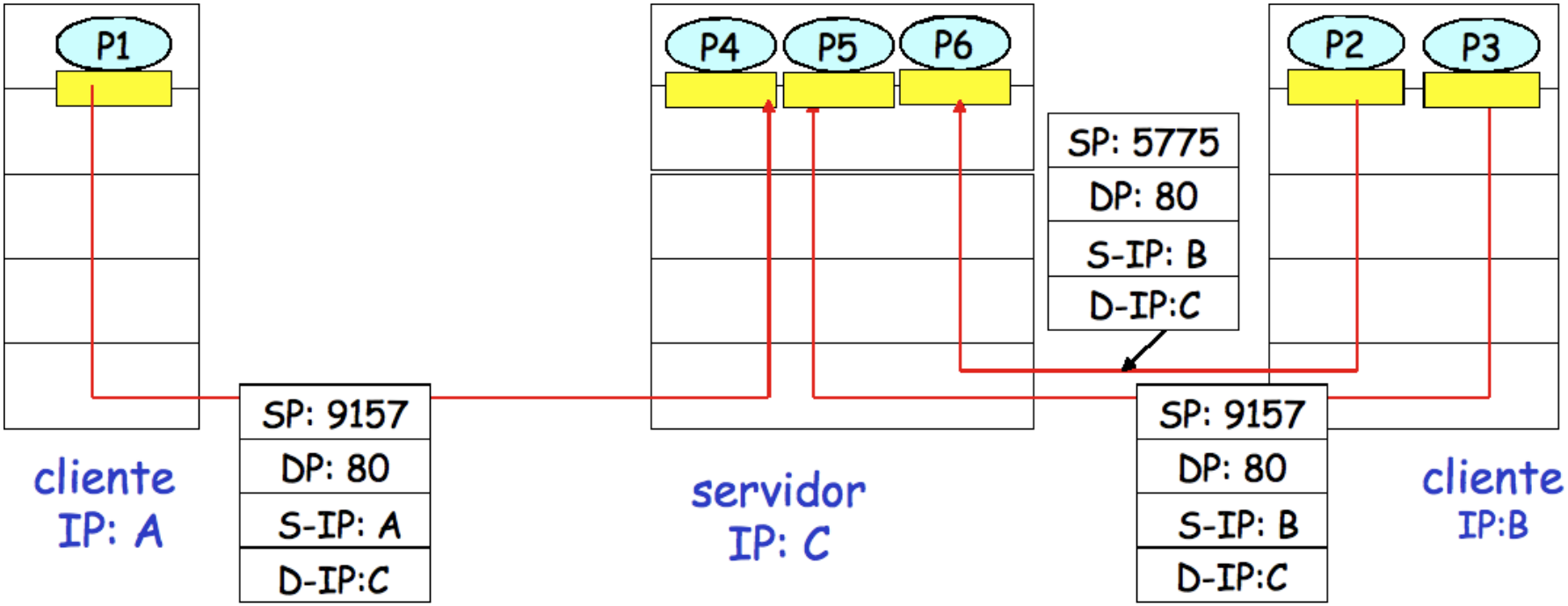
```
DatagramSocket serverSocket = new DatagramSocket(6428);
```



Demux orientada a conexão

- Conexão (socket TCP) é identificado por quatro valores

  (IP_origem, porta_origem, IP_destino, porta_destino)



Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Transmissão no UDP

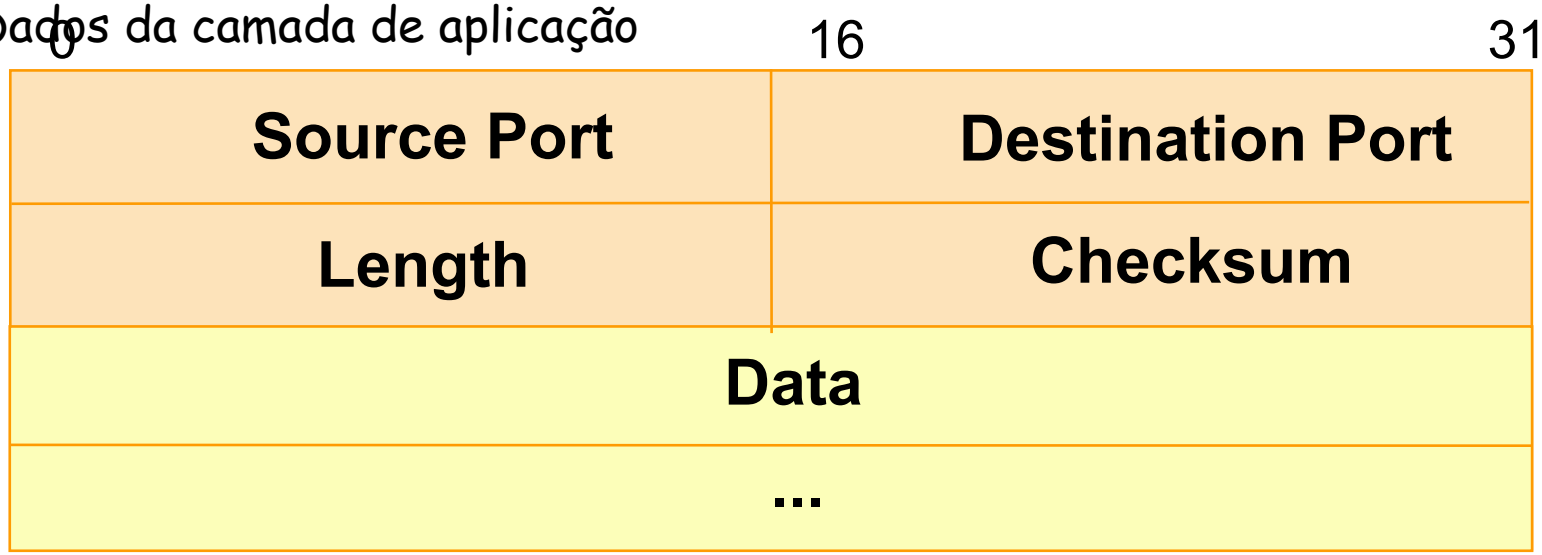
- UDP implementa serviço "best effort", e segmentos podem ser
  - Perdidos
  - Entregues fora de ordem para a aplicação
- Características
  - Não há estabelecimento de conexão
  - Aplicação pode controlar quais dados são enviados e quando
  - Não há estados da conexão
  - Sobrecarga pequena no cabeçalho do pacote

Protocolos

- **UDP (User Datagram Protocol):**
  - Serviço não orientado a conexão
  - Baixo overhead
  - Não confiável
    - Sem detecção de erro
    - Sem controle de seqüência
- **TCP (Transmission Control Protocol):**
  - Serviço orientado a conexão
  - Confiável
  - Detecção e correção de erros
  - Controle de fluxo fim-a-fim
  - ...

UDP – Formato do Datagrama

- **Source/Destination Port:**
  - Portas usadas pelos processos
- **Length:**
  - Tamanho do datagrama (octetos)
- **Checksum:**
  - Verificação de erro no datagrama (opcional). Normalmente este não é implementado;
- **Data:**
  - Dados da camada de aplicação



Checksum UDP

- Detecção de erros no segmento transmitido
  - Inclusão de campo (checksum) calculado a partir do campo de dados + cabeçalho
  - Transmissor computa o checksum como a soma em complemento de 1 dos inteiros de 16 bits que compõem o segmento
  - Checksum é enviado junto com o segmento
- Destinatário verifica o checksum quando o segmento chega → se o cálculo do checksum não confere com o respectivo campo, então os dados foram corrompidos
  - Verificação não é obrigatória
  - Verificação não garante transmissão livre de erros (cálculo matemático é pensado para diminuir chance de erros aleatórios)

Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Tratamento de Erros

- Duas tarefas podem ser realizadas para tratamento de erros: **detecção** e **correção** de erros;
- **Detecção** → **identifica** a ocorrência de um erro enquanto a **correção**, além de identificar, também tenta **corrigi-lo**;
- O tratamento de erros é implementado a partir de informações de controle (bits extras) que devem ser inseridas juntamente com os dados transmitidos;
- Quanto melhor a estratégia adotada, maior o número de bits de redundância;

Detecção de Erros

- ***Checksum***: O remetente trata os dados (incluindo cabeçalho) como uma seqüência de inteiros binários e computa sua soma;
  - O valor da soma (em complemento de um) de todos os bytes da informação a ser transmitida é armazenada no cabeçalho do pacote;
  - O receptor da mensagem implementa um processo análogo (soma de todos os bytes) e compara com a informação contida no cabeçalho, se igual, informação íntegra; do contrário, existe um erro;
    - **Vantagem**: o custo de processamento exigido para a computação só exige "adição";
    - **Desvantagem**: Não detecta todos os erros comuns.

Detecção de Erros

- Checksum
  - Soma todos os blocos de 16 bits do pacote + carry
  - Valor do campo checksum deve ser desconsiderado (ou igualado a 0x00)
  - Resultado é armazenado em **complemento de um** no campo checksum
- No destino, repete o cálculo
  - Pacote está correto se resultado do cálculo é igual ao do campo checksum recebido
- Erros ainda podem ocorrer e passar desapercebidos (chances limitadas em $1/2^{16}$).

Correção de Erros

Detecção e Correção de pacotes perdidos

- Mecanismos utilizados para identificar se um pacote foi perdido ou não chegou íntegro;
- Destinatário envia **ACKs** (*acknowledgement* ou reconhecimento) positivos ou negativos para confirmar o recebimento do pacote;
  - **Ack positivo**: A estação receptora informa que o pacote foi recebido corretamente;
  - **Ack negativo**: A estação receptora informa que o pacote foi recebido com erro;
- Transmissor usa temporizador (**timeout**) para identificar pacotes perdidos;
  - pacotes totalmente destruídos ou endereço de origem ou destino modificado;
- Se um ACK positivo não for recebido antes do temporizador atingir **timeout** o pacote é retransmitido;

Correção de Erros

Detecção e Correção de pacotes perdidos

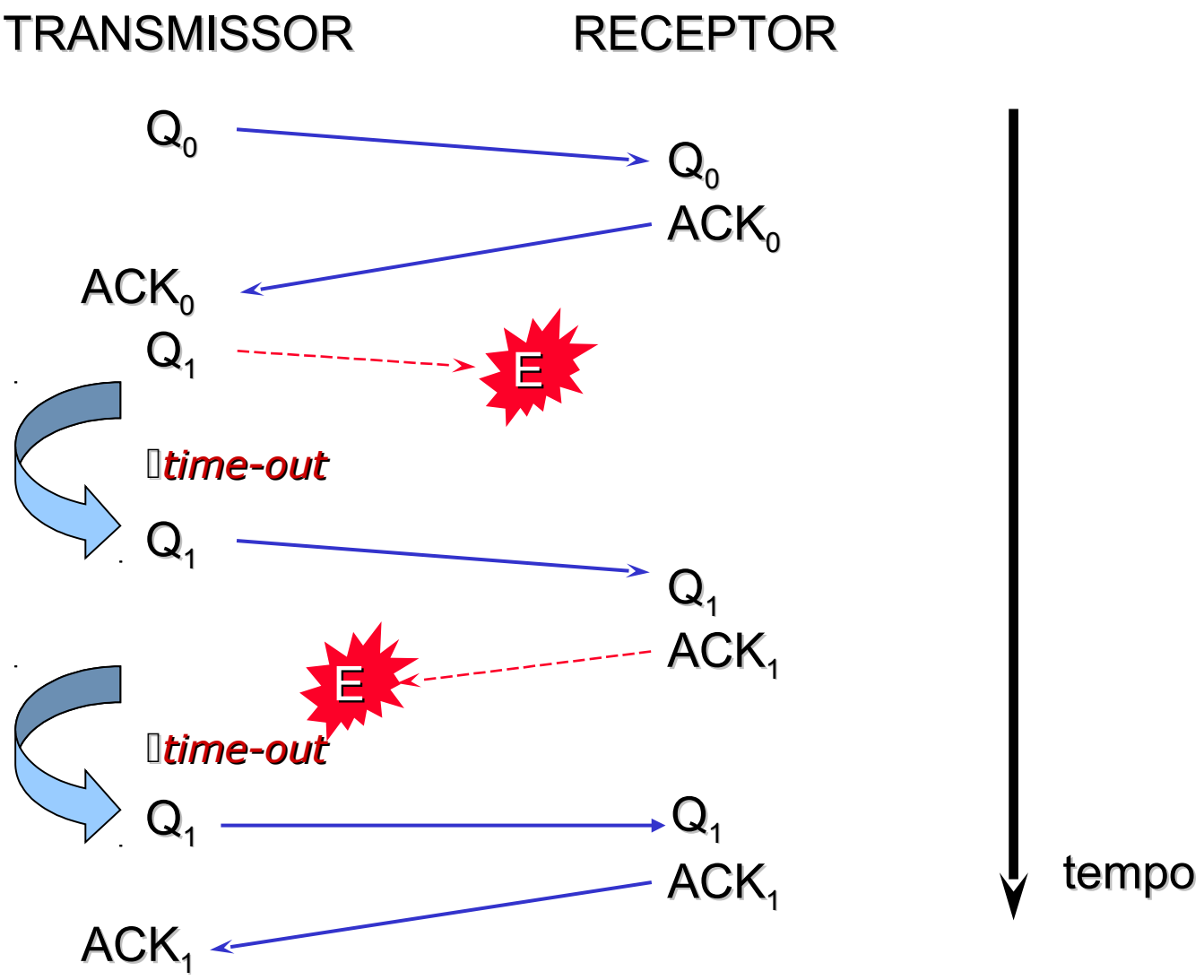
- ACKs positivos + time-outs são suficientes, porém, ACK negativo diminui o retardo;
- Pacotes são numerados para o destinatário distinguir pacotes originais de cópias;
- Os reconhecimentos podem ser enviados de duas formas:
  - Através de um pacote de controle específico para tal tarefa;
  - Ou, pode ser transportado de carona em um campo de controle de um pacote de informação (piggybacking);
- Algoritmos de controle de erros mais utilizados:
  - **bit alternado** (stop and wait);
  - **janela n com retransmissão integral** (Go-Back-N);
  - janela n com **retransmissão seletiva** (selective repeat);

Correção de Erros – Bit Alternado

- **Algoritmo do Bit Alternado**
  - transmissor só envia novo pacote quando recebe ACK do pacote anterior;
  - *time-outs* controlados no transmissor;
  - simples mas ineficiente em termos de utilização do meio. Enquanto o transmissor espera por reconhecimentos o canal de comunicação não é utilizado;
  - pacotes são numerados em 0, 1, 0, 1, 0, ... para distinguir cópias dos originais;

Correção de Erros – Bit Alternado

- Algoritmo do Bit Alternado (seqüência de pacotes)



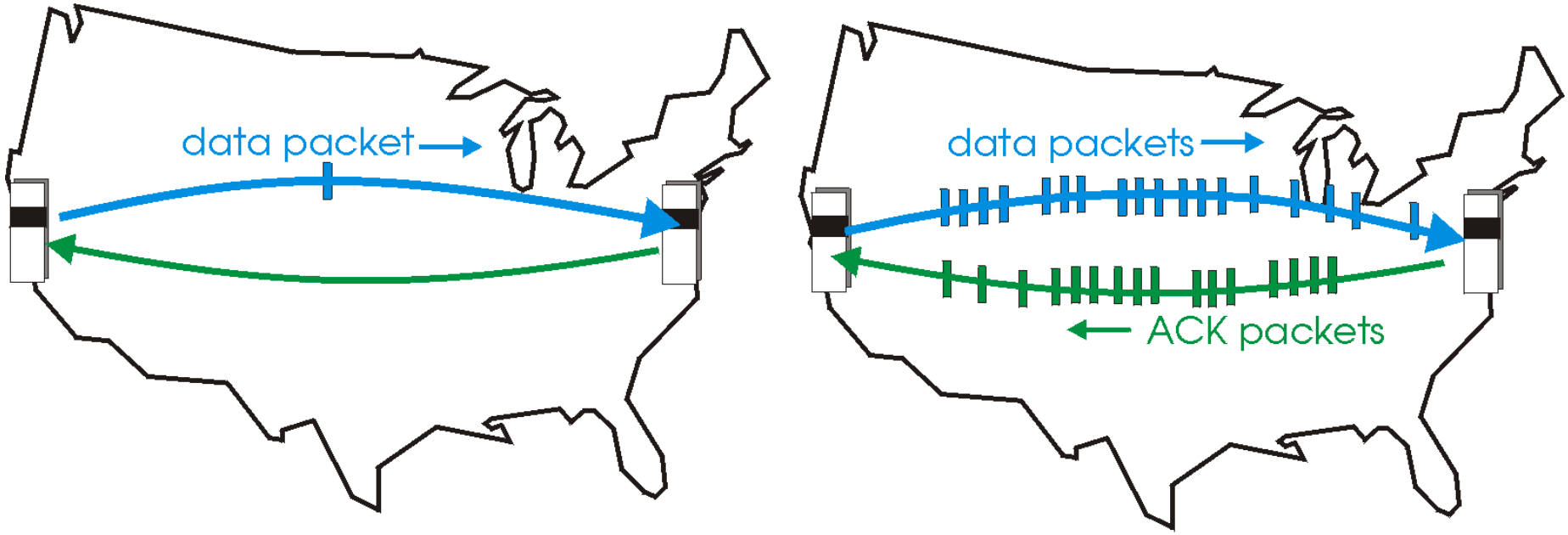
Correção de Erros

- Algoritmos de detecção de erro mais otimizados:
  - Permite enviar **N** pacotes sem ter recebido reconhecimentos dos pacotes anteriormente enviados;
  - A janela de transmissão define o numero de pacotes que podem ser enviados sem receber confirmação;
    - A janela de transmissão é composta de vários pacotes. O tamanho da janela é definido durante o estabelecimento da conexão entre as estações;
  - Detecta erro a partir do reconhecimento positivo e através do temporizador para detecção de timeout;

Go-Back-N

- Algoritmo de detecção de erro mais otimizados: **Go-Back-N**
- Existem duas formas para reparar o erro na transmissão:
  - **retransmissão integral**: todos os pacotes a partir do que não foi recebido são retransmitidos;
  - **retransmissão seletiva**: só o pacote que não foi recebido é retransmitido;
- Para melhorar ainda mais a eficiência de utilização do canal, foi proposto uma forma mais eficiente de enviar o reconhecimento dos pacotes de transporte:
  - O transmissor ao receber o Ack do pacote **i** conclui que ele, e os pacotes antes dele foram recebidos corretamente; (**$1 \leq i \leq n$**)
  - Cada pacote continua tendo o seu próprio timeout. Não existe um timeout único para todos os pacotes de uma janela.

Go-Back-N



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

(b) a pipelined protocol in operation

Go-Back-N

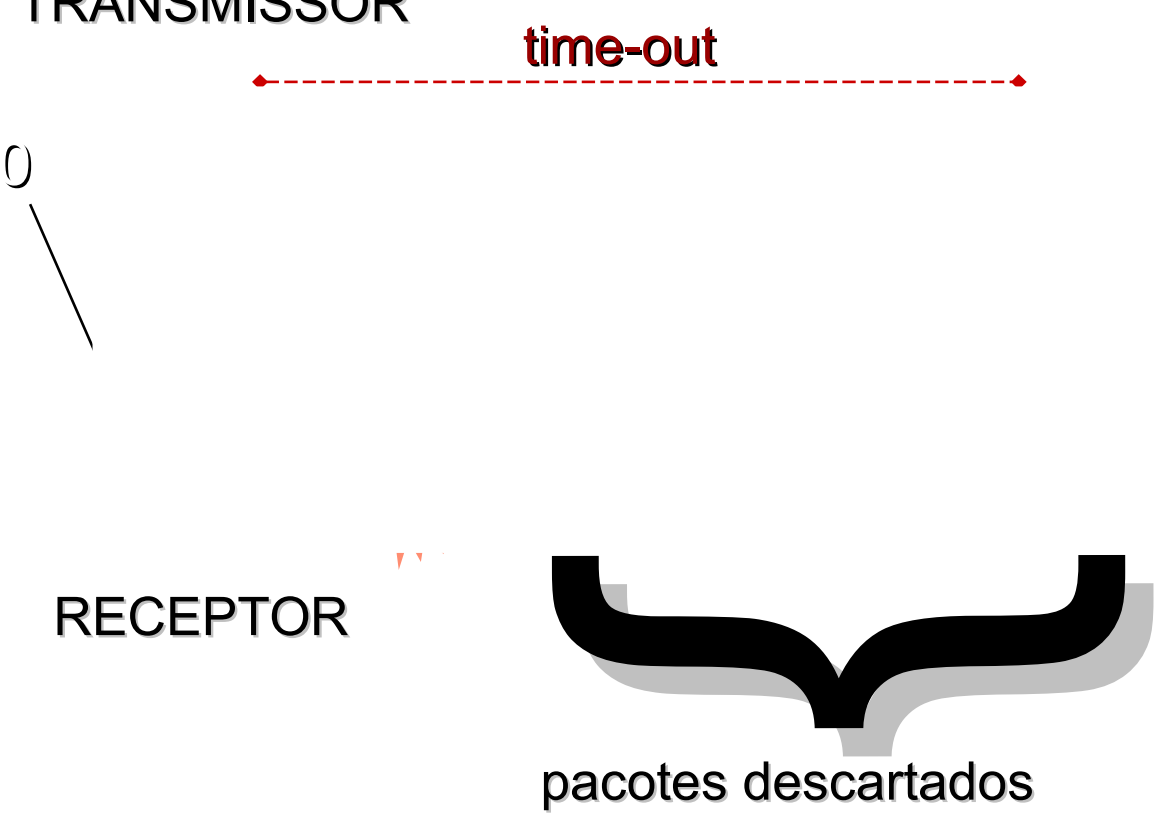
- Exemplo de retransmissão integral

Linha Preta: pacotes enviados

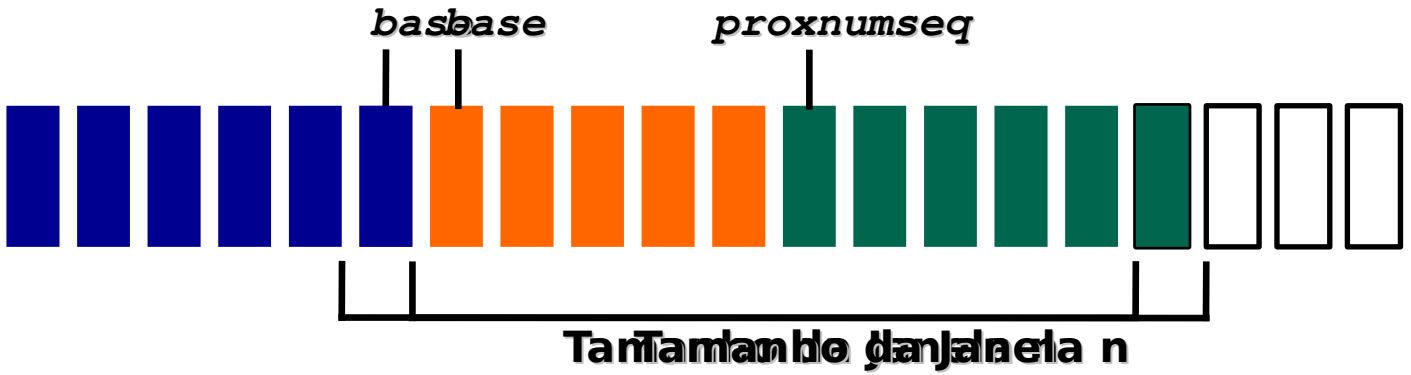
Linha Azul: Reconhecimento

Número vermelho: pacotes descartados

Números verdes: pacotes retransmitidos

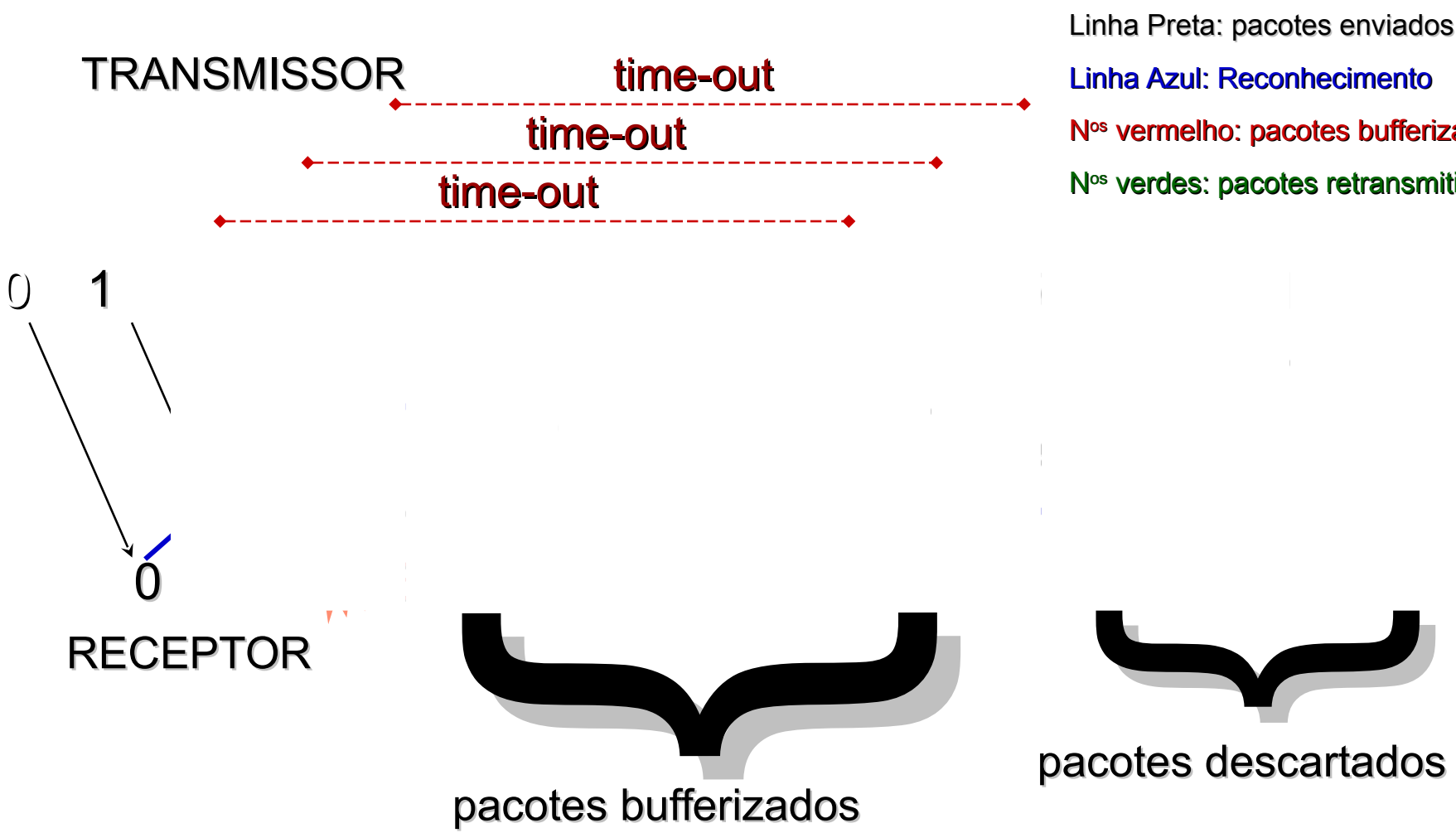


Go-Back-N


base
proxnumseq
Tamanho da Janela n
Já Reconhecido
Enviado, mas ainda não Reconhecido
Autorizado, mas ainda não enviado
Não autorizado

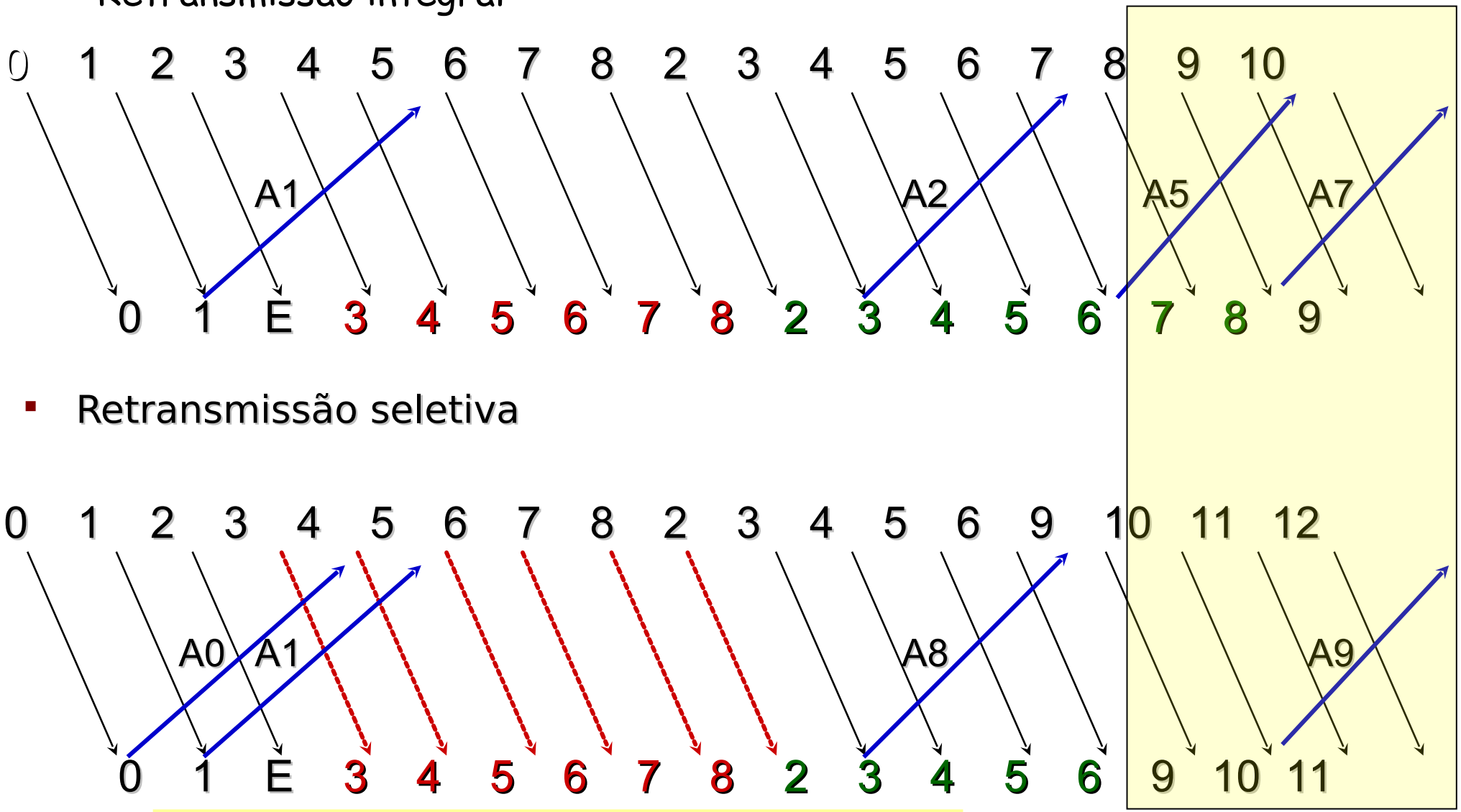
Nível de transporte

- Exemplo de retransmissão seletiva



Comparação entre os mecanismos

- Retransmissão integral



- Retransmissão seletiva

Comparativo é para um cenário particular → instante de envio dos ACKs depende de fatores não discutidos.

Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

TCP

- Características:
  - Baseado em Stream:
    - Seqüência de bytes não estruturada
  - Conexão com Circuito Virtual (Orientado à Conexão):
    - Estabelecimento
    - Transferência de dados
    - Encerramento de conexões
  - Buffers:
    - Buffer de retransmissão com controle automático de envio

TCP

- Segmentos:
  - TCP manipula o stream de dados como uma seqüência de octetos divididos em segmentos
  - Cada segmento é encapsulado dentro de um datagrama IP
- Números de Seqüência:
  - Segmentos no stream de dados são numerados seqüencialmente;

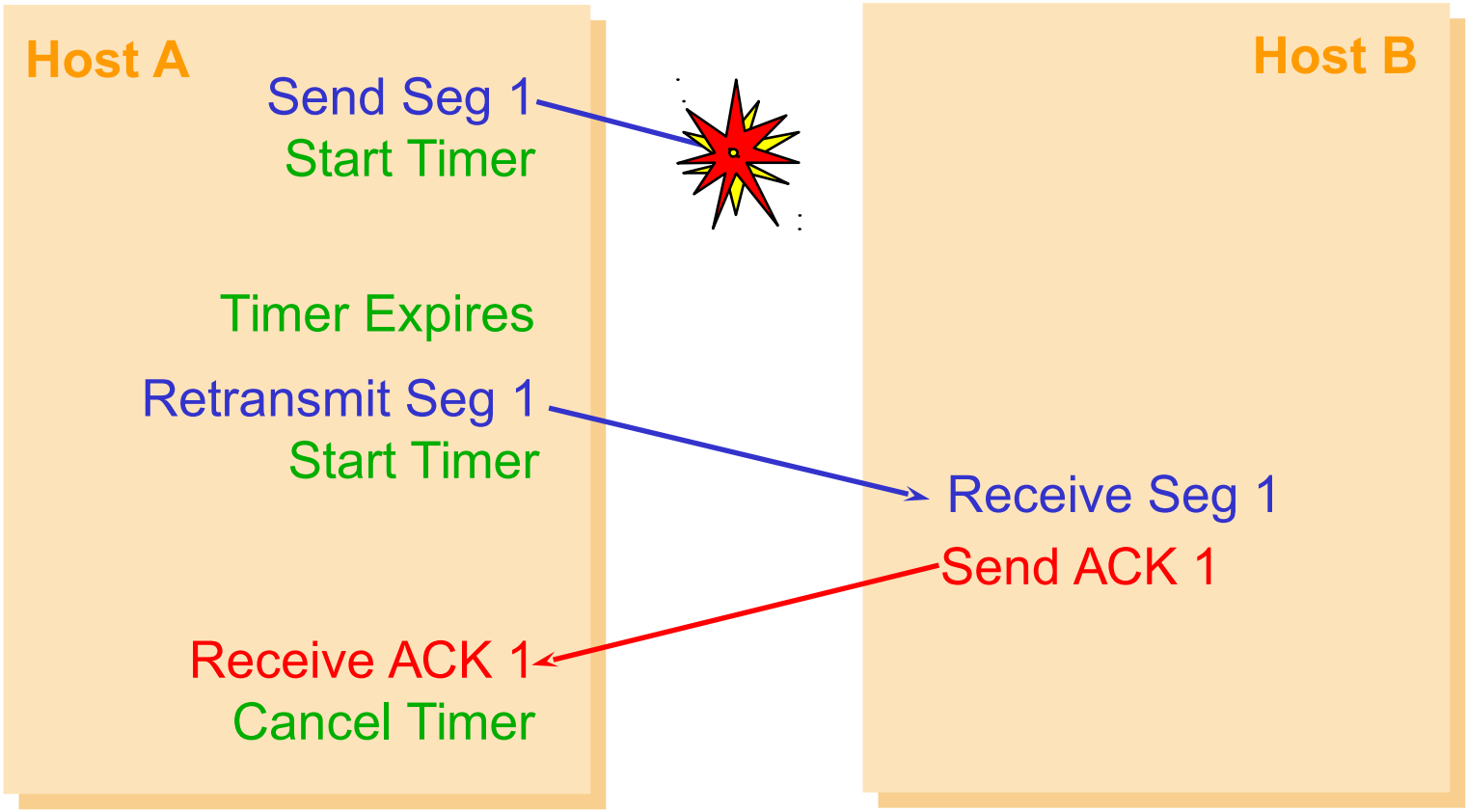
TCP

- Confiabilidade:
  - Implementa confiabilidade através de reconhecimento positivo e retransmissão (caso ocorra erro/falha)
    - Utiliza Piggybacking no reconhecimento
      - Confirmação enviada junto com a parte de dados
    - Retransmissão baseada em timeout

TCP

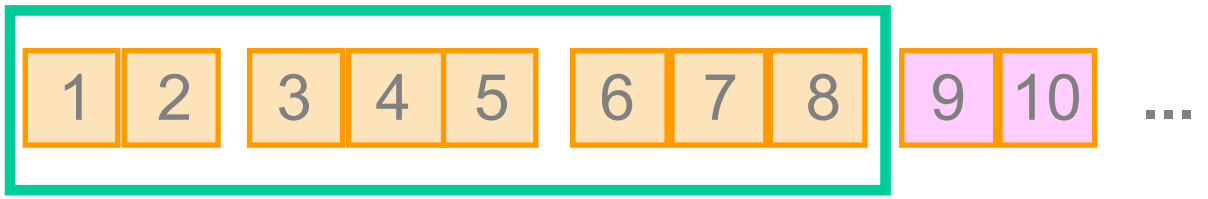
- O TCP ao transmitir os dados de um segmento
  - Guarda uma cópia na fila de retransmissão
  - Aguarda a chegada de um reconhecimento
  - Se timeout estourar, o segmento da fila de retransmissão é enviado

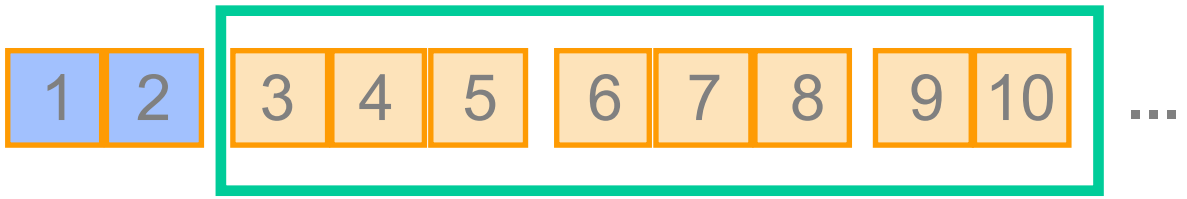
TCP


Host A
Send Seg 1
Start Timer
Timer Expires
Retransmit Seg 1
Start Timer
Receive ACK 1
Cancel Timer
Host B
Receive Seg 1
Send ACK 1

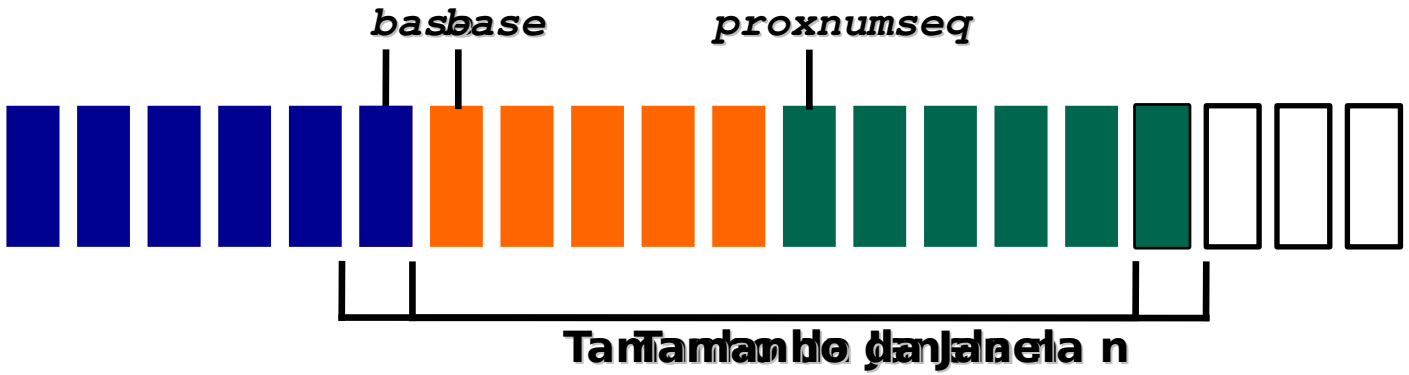
TCP

- Sliding Window (Janela Deslizante):
  - Permite o transmissor enviar múltiplos segmentos antes de receber reconhecimento
  - Eficiência de transmissão e controle de fluxo dinâmico
    - É informado junto com o reconhecimento o números de segmentos que o receptor tem capacidade de receber.



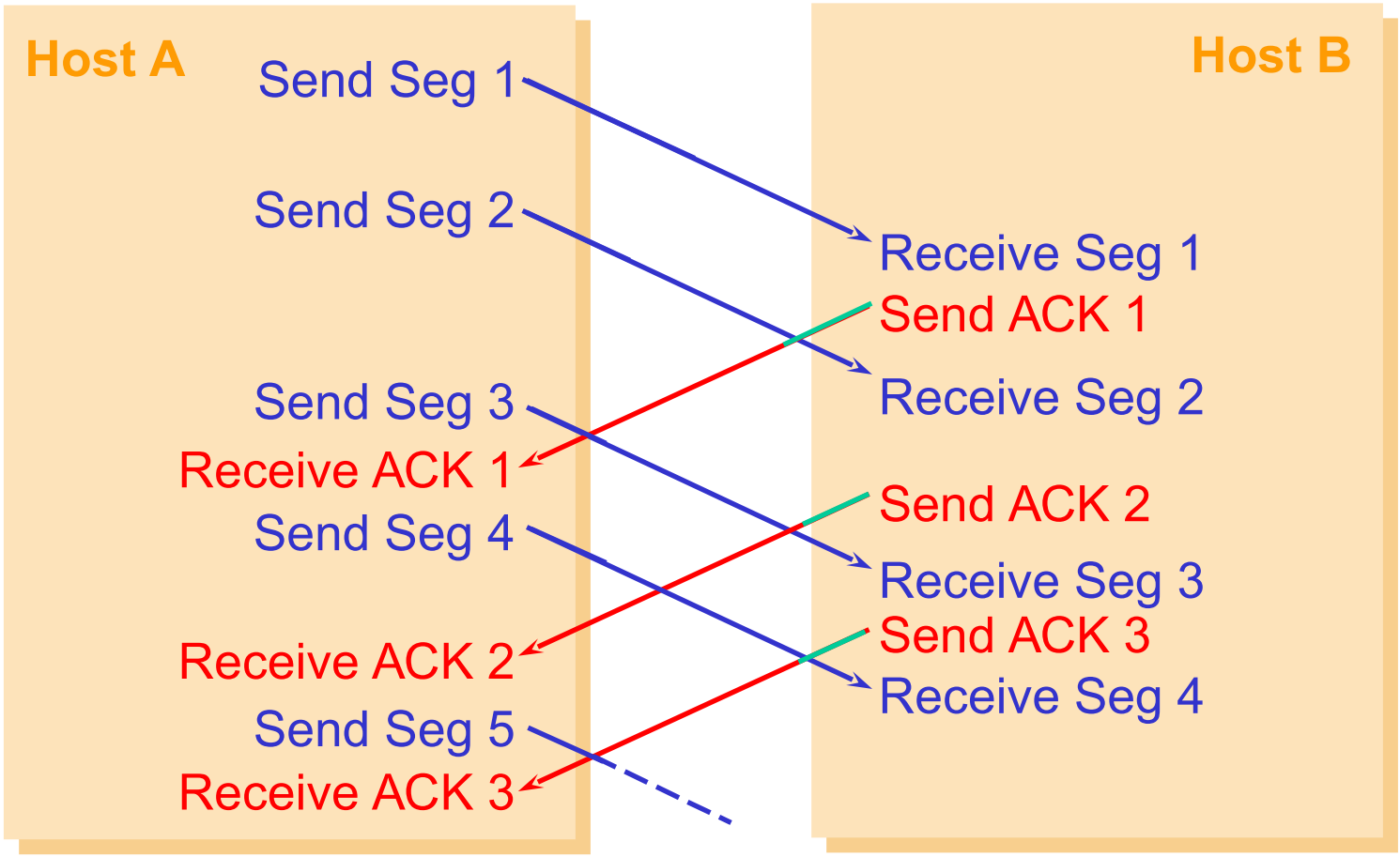


Janela Deslizante

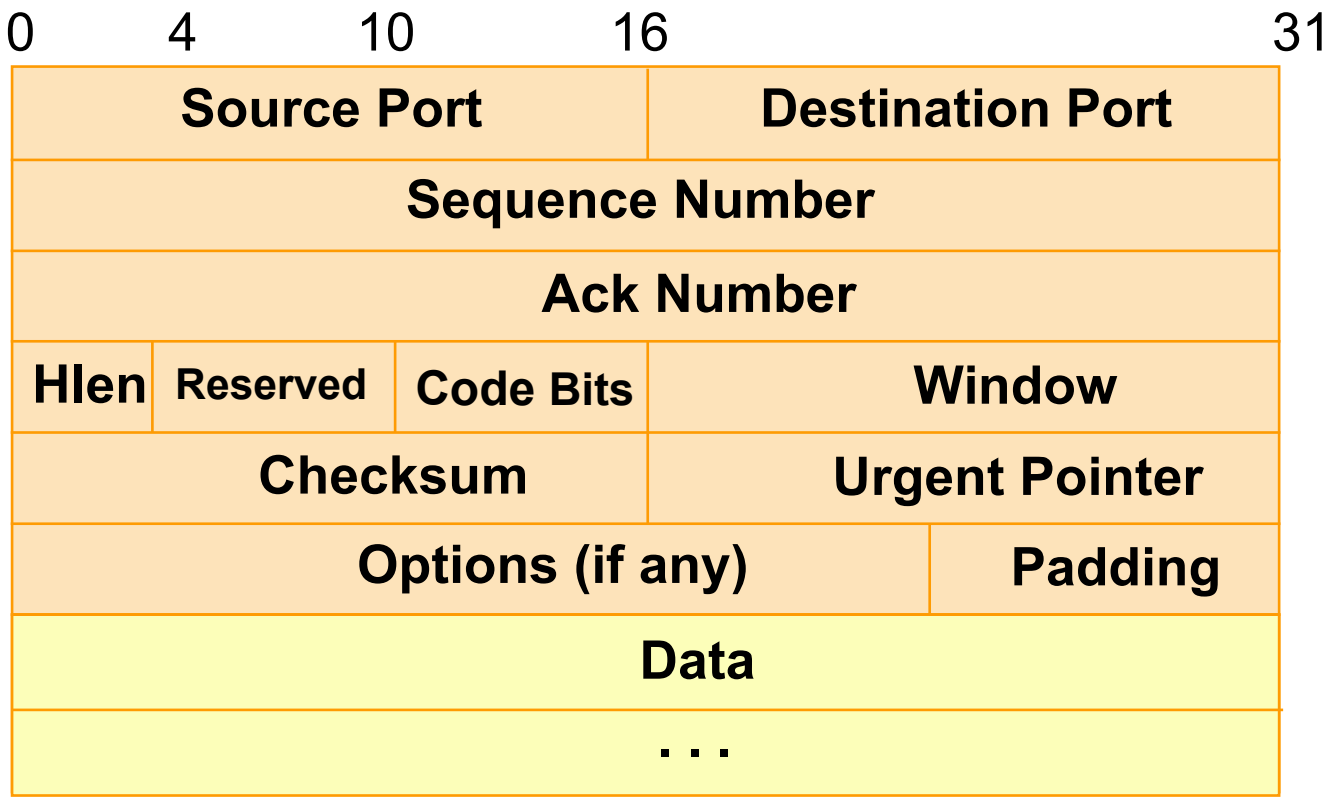


TCP

- Exemplo: com Janela de transmissão = 3



Pacote TCP


0
4
10
16
31
Source Port
Destination Port
Sequence Number
Ack Number
Hlen
Reserved
Code Bits
Window
Checksum
Urgent Pointer
Options (if any)
Padding
Data
. . .

TCP

- Campos do Segmento:
  - **Source/Destination Port:**
    - Identificam os processos origem e destino da conexão
  - **Sequence Number:**
    - Posição/Número do segmento dentro do stream de dados
  - **Ack Number:**
    - Reconhecimento (ACK) que o destino espera receber
  - **Hlen**
    - Tamanho do cabeçalho (unidade: 4 octetos)

TCP

- Campos do Segmento:
  - **Code Bits:**
    - **URG**: Campo Urgent Pointer válido
    - **ACK**: Confirmação do pedido de conexão
    - **PSH**: Segmento requer um push
    - **RST**: Reseta a conexão
    - **SYN**: Estabelecimento de conexão
    - **FIN**: Origem finalizou seu stream de bytes. Encerramento da Conexão.

TCP

- Campos do Segmento:
  - **Window:**
    - Especifica o tamanho da janela
  - **Checksum:**
    - Verificação de erro no segmento
  - **Urgent Pointer:**
    - Posição dos dados urgente no segmento
  - **Options:**
    - Tamanho de Segmento Máximo (MSS)

TCP

- Portas, Conexões e Endpoints:
  - TCP utiliza o conceito de conexão para identificar os dois pontos envolvidos na comunicação
  - A conexão é identificada por um par de Endpoints
- Um Endpoint é um par de inteiros da forma:

Host , Port

(128.9.0.32,1184) (128.10.2.3,25)

Maximum Segment Size (MSS)

- O TCP tenta evitar a fragmentação implementada pelo protocolo IP na camada de rede através do MSS;
- O MSS é definido pelo TCP durante o estabelecimento da conexão (three-way-handshake);
- O MSS representa o tamanho máximo da parte de dados de um segmento TCP, tendo o propósito de evitar a fragmentação na camada de rede;

Max. Transmission Unit (MTU)

- A Maximum Transmission Unit (MTU) é o tamanho máximo de um pacote IP (incluindo o seu cabeçalho) que evita a implementação de fragmentação
  - Fragmentação: quebra do pacote em pacotes menores por não caber no tamanho máximo da camada de enlace.
- O pacote IP a ser transmitido sofrerá fragmentação caso seja maior que o MTU.
- O MTU do protocolo IP é diferente em cada tecnologia. Por exemplo, vide tabela ao lado.

MTU	Protocol	RFC
576	Default	879
1500	PPP Default	1134
1500	Ethernet	895
1006	SLIP	1055
1492	PPPOE	2516

Calculando o tamanho do MSS

- O MSS é calculado a partir do MTU
- A RFC879 define como o MSS é calculado pelo protocolo TCP:
  - O MSS do TCP é o tamanho máximo de um datagrama IP definido pela MTU menos 40 (considerando IPv4 e sem campos opcionais no cabeçalho);
  - `MSS = MTU - sizeof(TCPHDR) - sizeof(IPHDR)`
  - `MSS = MTU-20-20`
- Por exemplo, para calcular o MSS em uma rede Ethernet:
    - Passo 1: Definir a MTU em uma rede Ethernet = 1500 bytes
    - Passo 2: Calcular o MSS do protocolo TCP em uma rede Ethernet
      - `MSS = MTU - sizeof(TCPHDR) - sizeof(IPHDR)`
      - `MSS = 1500 - 20 -20 =` **`1460 bytes`**
  - Resumindo:
    - **`TCP Maximum Segment Size = IP Maximum Datagram Size - 40`**

Para IPv6, cabeçalho (fixo) é 40 bytes. Campo **options** pode especificar cabeçalho maior tanto em IPv4 como IPv6. Tamanhos usados nos cálculos deve ser ajustado de acordo.

Calculando o tamanho do MSS

- O MSS é calculado a partir do MTU
- A RFC879 define como o MSS é calculado pelo protocolo TCP:
  - O MSS do TCP é o tamanho máximo de um datagrama IP definido pela MTU menos 40;
  - `MSS = MTU - sizeof(TCPHDR) - sizeof(IPHDR)`
  - `MSS = MTU - 20 -20`
- Por exemplo, para calcular o MSS em uma rede Ethernet:
  - Passo 1: Definir a MTU em uma rede Ethernet = 1500 bytes
  - Passo 2: Calcular o MSS do protocolo TCP em uma rede Ethernet
    - `MSS = MTU - sizeof(TCPHDR) - sizeof(IPHDR)`
    - `MSS = 1500 - 20 -20 = 1460 bytes`
- TCP Maximum Segment Size = IP Maximum Datagram Size - 40

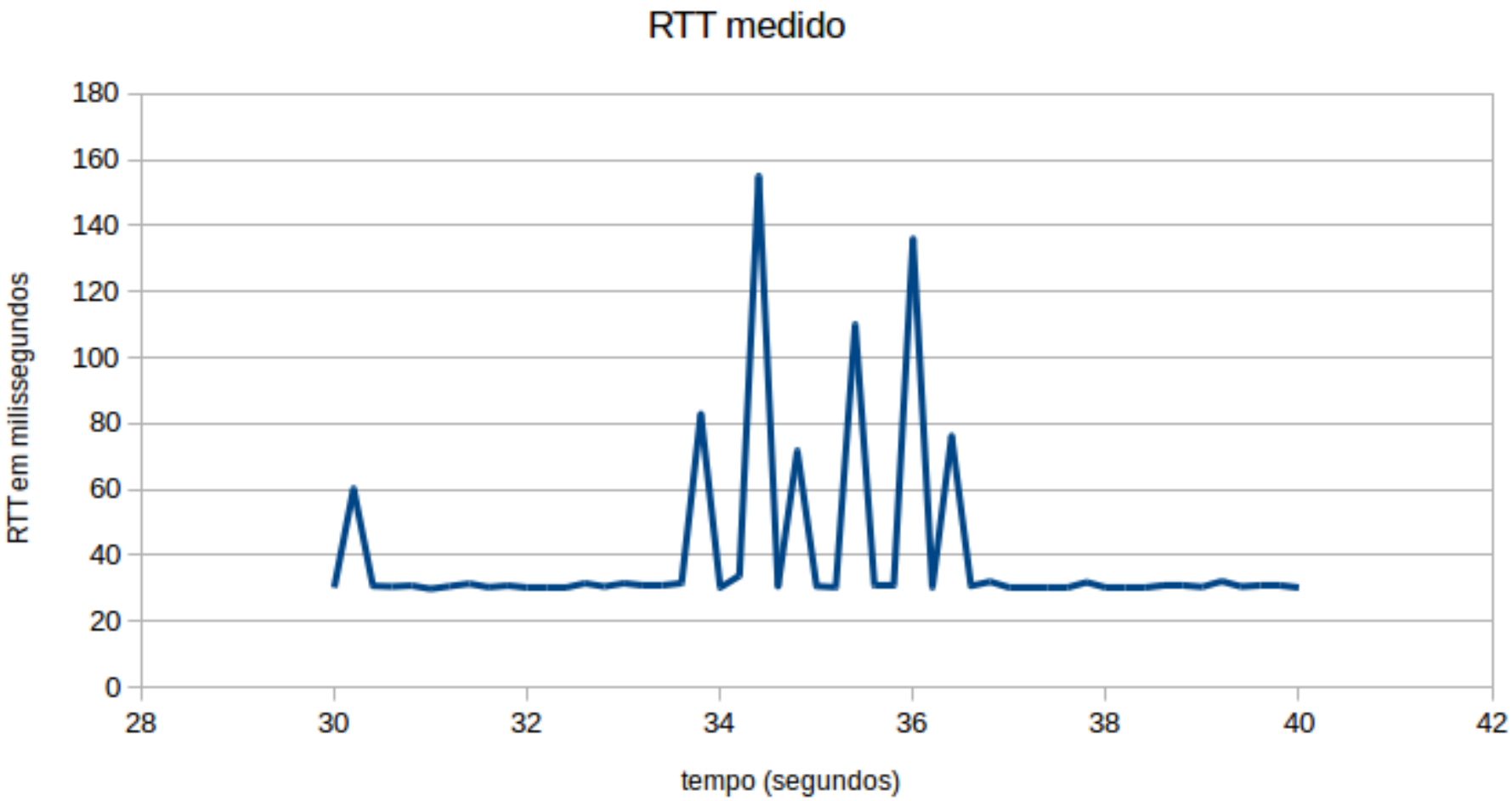
Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Estimativa de Timeout e RTT

- **Timeout** → tempo máximo de espera por recebimentos de ACKs do TCP (para uma dada conexão)
  - Expirado timeout, segmento é considerado perdido (não entregue)
- Boa escolha de um timeout torna o protocolo eficiente
- Fundamentos da escolha do timeout
  - Timeout deve variar **dinamicamente**, em função de congestionamentos
  - Timeout deve ser uma **função de RTT**
    - RTT → Round-trip delay time (tempo de ida e volta de um pacote)
    - RTT é imprevisível!

Medição e Variação de RTT



Medição de RTT por 2 min usando ping (`ping -i 0.2 8.8.8.8`) do IP `189.41.16X.XXX` para IP `8.8.8.8`

Thu May 26 14:34:34 BRT 2016

RTT no Enlace e no Transporte

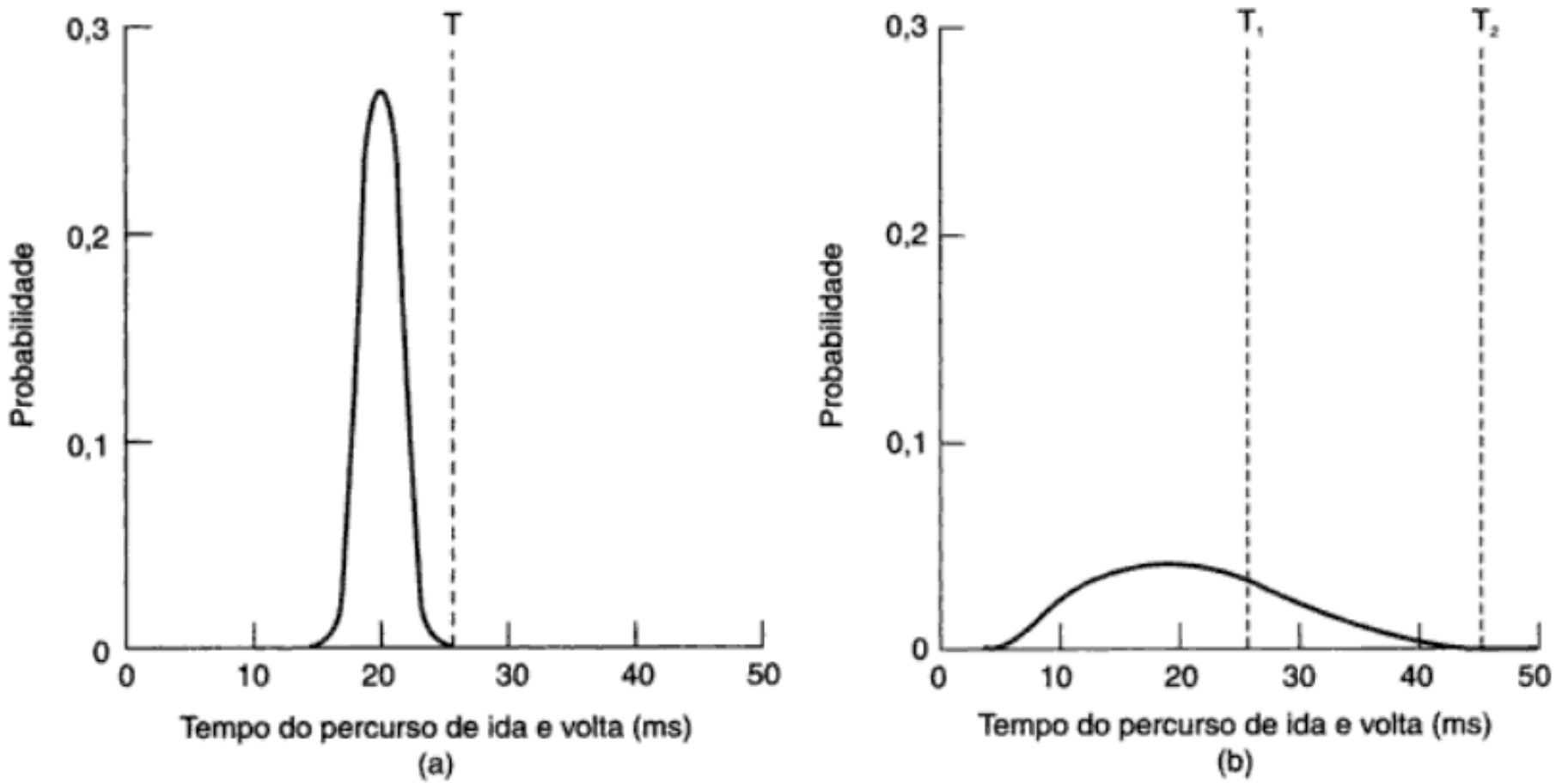
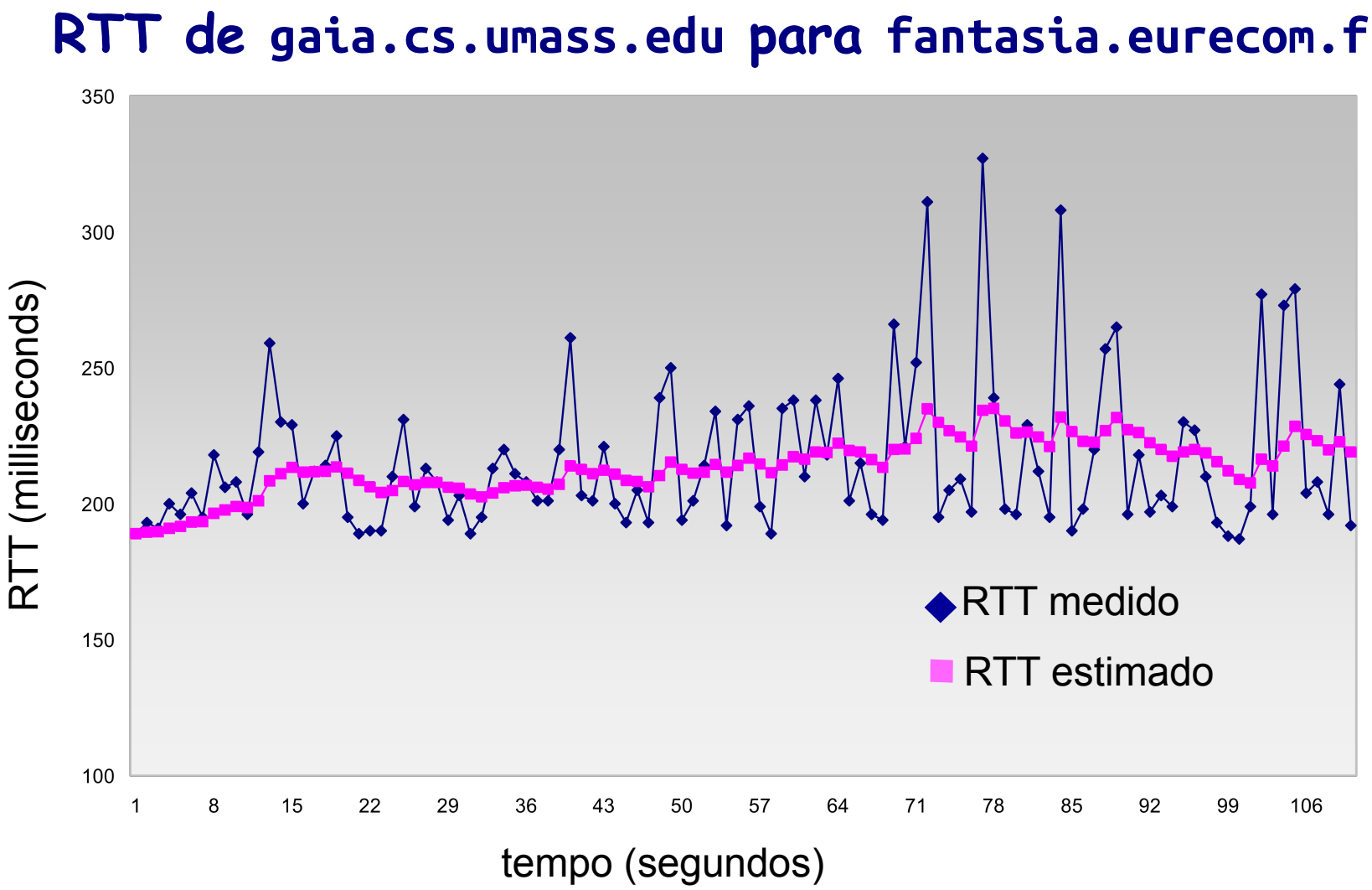


Figura 6.38 (a) Densidade de probabilidades de tempos de chegada de confirmações na camada de enlace de dados. (b) Densidade de probabilidades de tempos de chegada de confirmações para o TCP

Fonte: Tanenbaum, Andrew S. **Redes de computadores**. Pearson Education, 2003.

RTT: estimado vs. medido



Fonte: material do professor do livro Kurose/Ross

RTT

- Diretivas/Preocupações do método de escolha do RTT
  - Estimar um RTT que seja uma aproximação do próximo RTT
  - Usar um RTT estimado que **reaja rapidamente** a alterações do RTT, **sem ser exageradamente afetado por picos** momentâneos RTTs
- RTT deve ser calculado dinamicamente, a partir de estimativa (Algoritmo de Karn)
  - **RTTEstimado**: média ponderada dos RTTs medidos
  - **RTTEstimado = (1-α)*RTTEstimado+α*RTTMedido**
- Valor recomendado: **α=0,125** (RFC 2988)

Karn, P. and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", SIGCOMM 87.

Timeout

- Escolha do **timeout**
  - Deve levar em conta a variância do RTT

- `D` armazena o desvio de RTT
  - `D` é calculado sempre que um *ACK* é recebido
  - `M` é valor observado de `RTT`

$$D=aD+(1-a)|RTT-M|$$

$$Timeout = RTT + 4*D$$

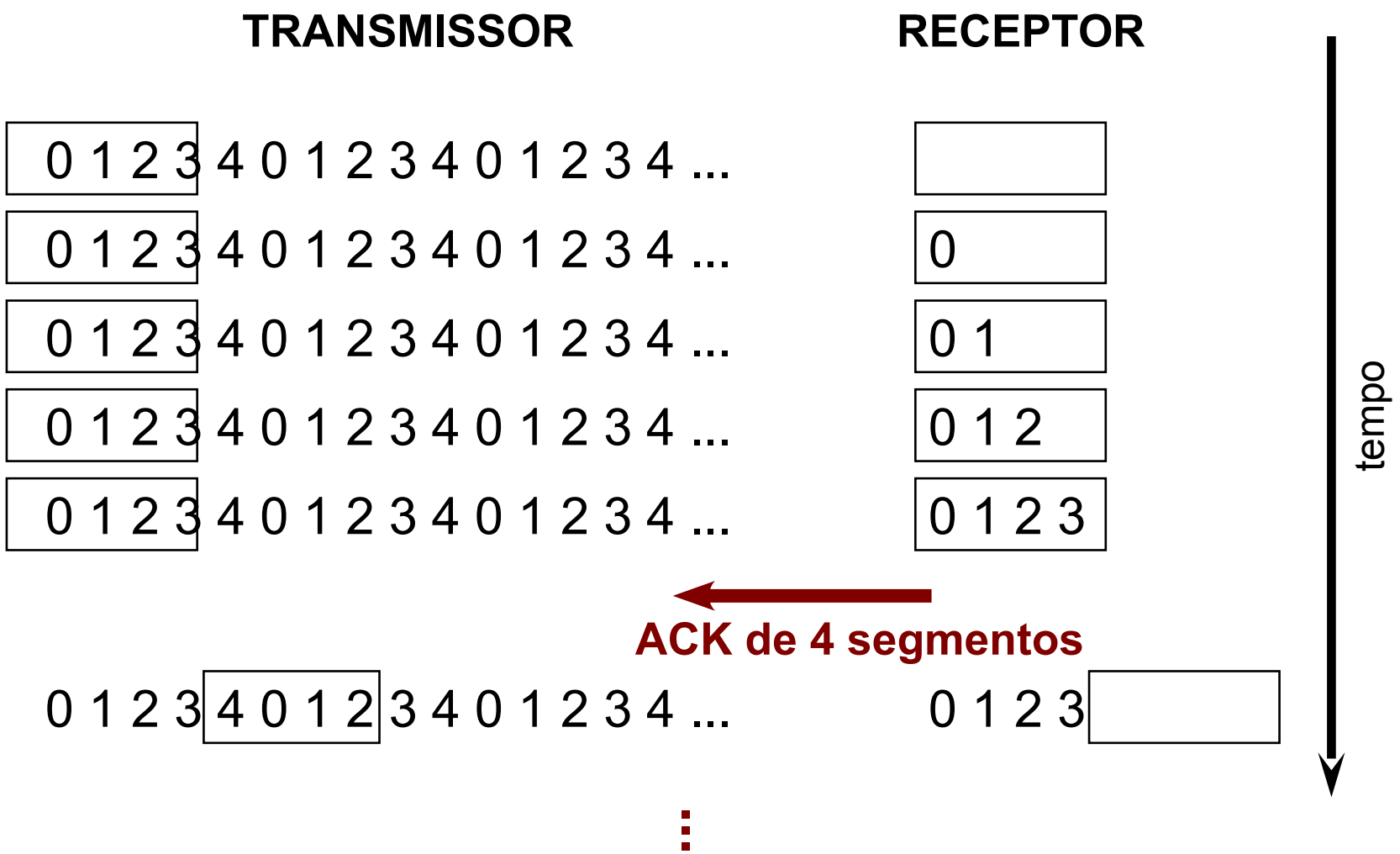
Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Controle de Fluxo Fim-a-Fim

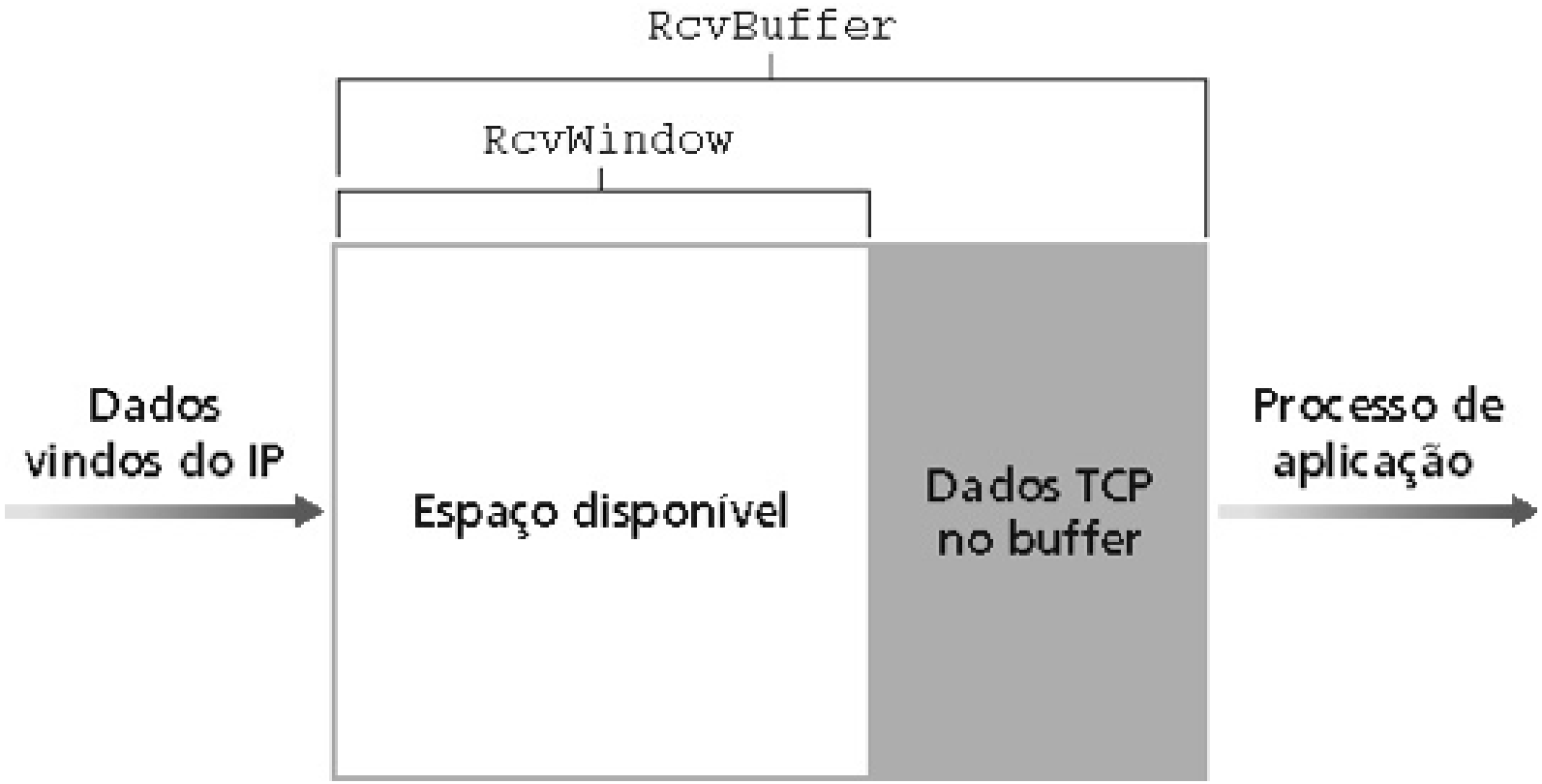
- Regula o fluxo de mensagens (pacotes) entre transmissor e receptor;
- Resolve o problema da diferença entre velocidade de transmissão e recepção;
- Não permite que uma estação transmissora mais rápida sobrecarregue uma estação receptora;
- Técnicas:
  - Stop-and-Wait == Bit Alternado
  - Sliding Window (Janela deslizante)

Controle de Fluxo

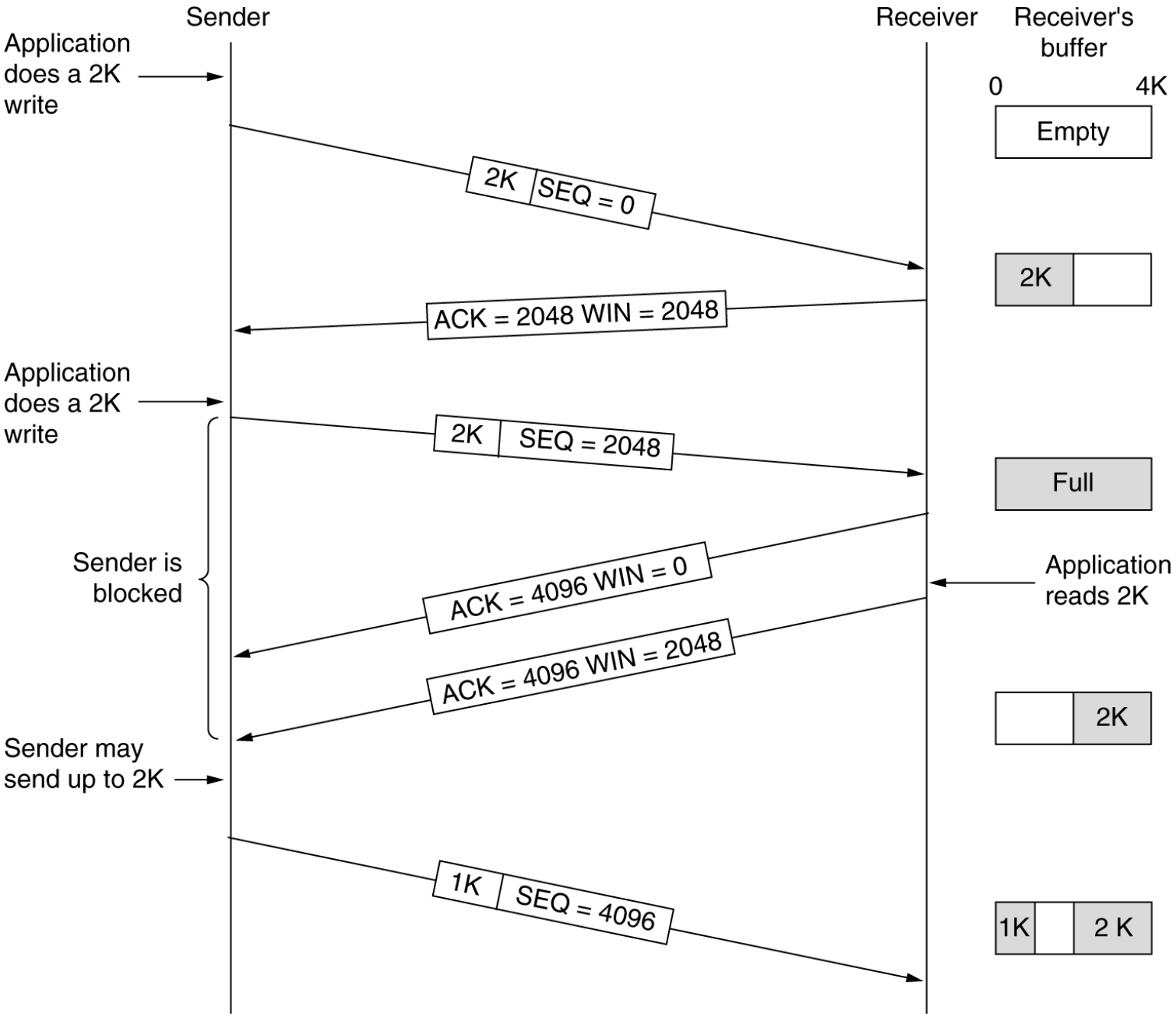


Buffers de Recepção

- **`RcvBuffer`**: tamanho do buffer de recepção
- **`RcvWindow`**: espaço livre no buffer de recepção



Janela de Transmissão

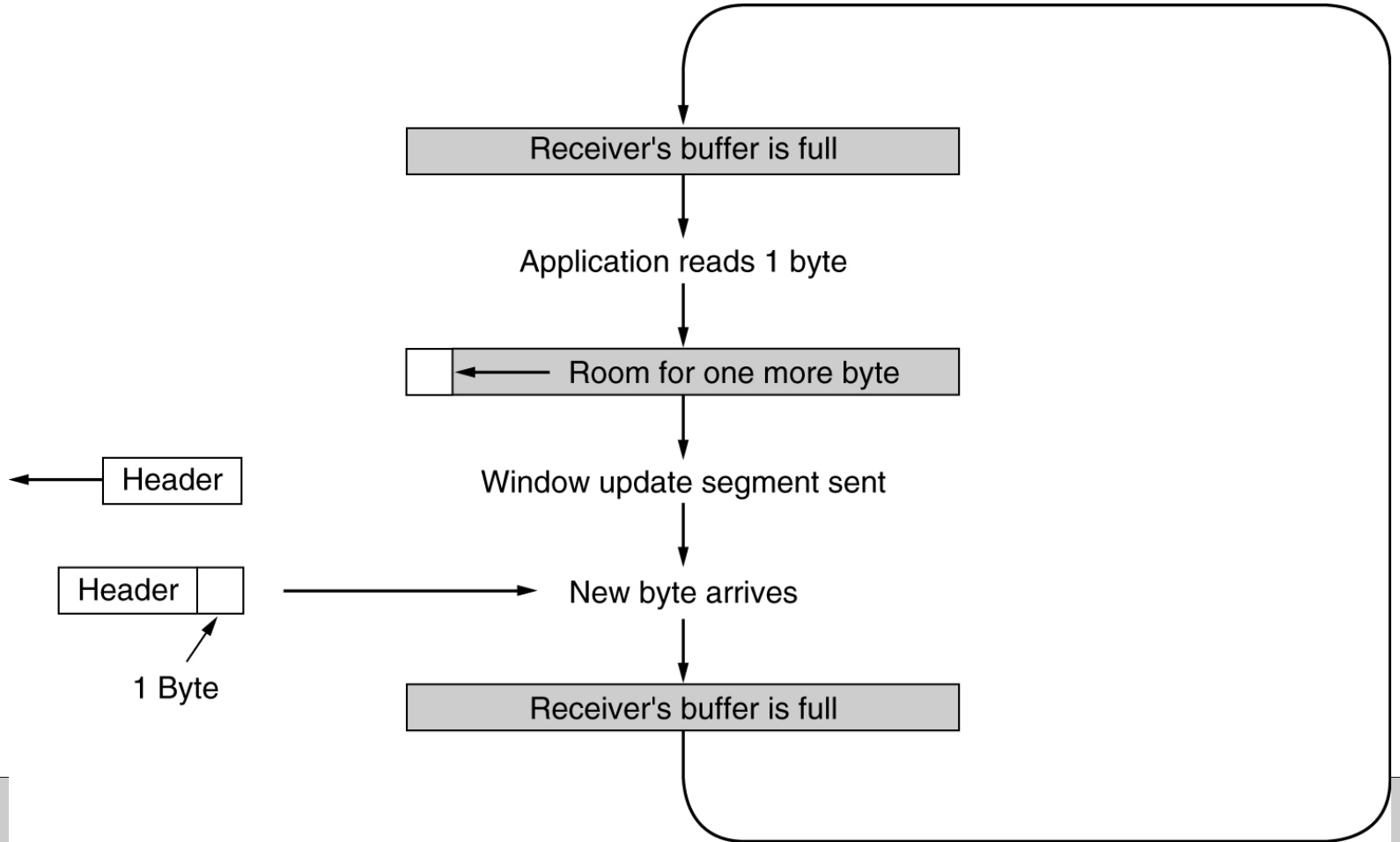


Janela de Transmissão

- Situações em que o desempenho pode ser inadequado
  - Janela sempre vazia: mau uso da largura de banda
    - Exemplo: aplicação de Telnet (terminal remoto)
    - Cada byte tende a ser enviado em único pacote e confirmado isoladamente
    - Melhoria: Algoritmo de Nagle
      - Envia um byte e aguarda confirmação
      - Enquanto isso, buffer de transmissão é preenchido pela aplicação
  - Janela sempre cheia: síndrome da *silly window*

Desempenho

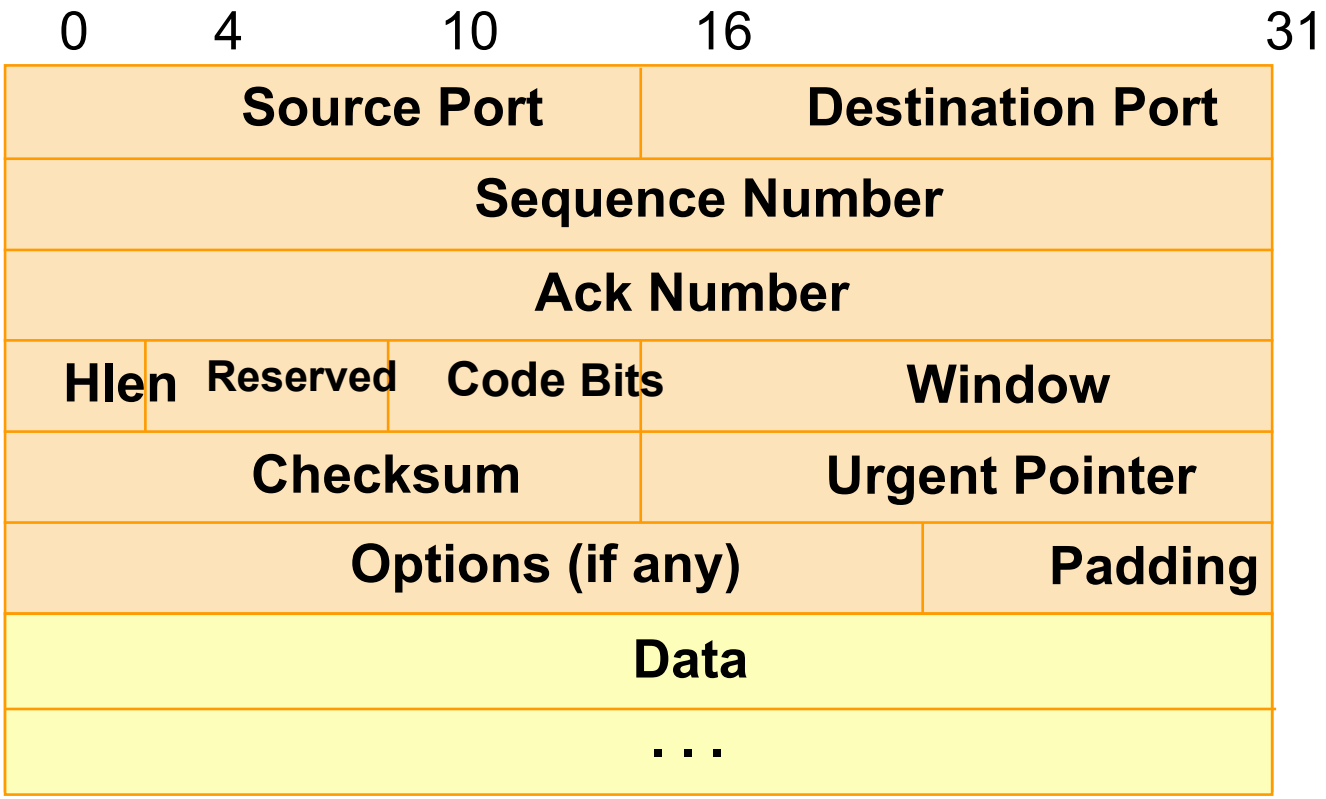
- Síndrome da *"silly window"*
  - Assim que 1 byte é liberado na janela, o receptor avisa e o transmissor envia 1 byte



Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

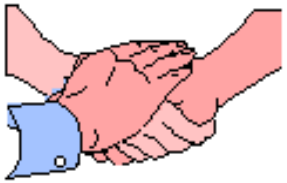
TCP


0
4
10
16
31
Source Port
Destination Port
Sequence Number
Ack Number
Hlen
Reserved
Code Bits
Window
Checksum
Urgent Pointer
Options (if any)
Padding
Data
. . .

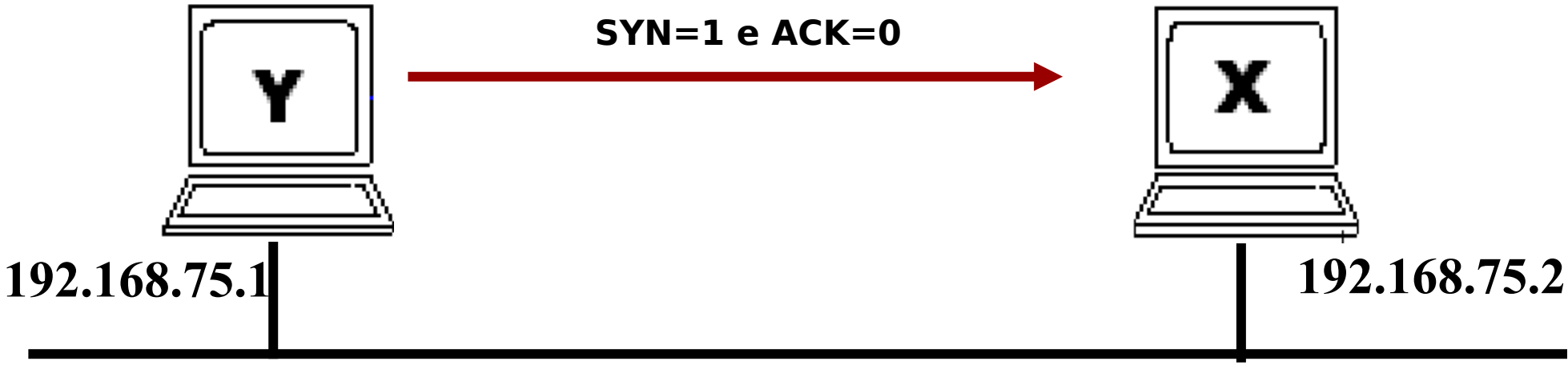
TCP

- Campos do Segmento:
  - **Code Bits:**
    - **URG:** Campo Urgent Pointer válido
    - **ACK:** Confirmação do pedido de conexão
    - **PSH:** Segmento requer um push
    - **RST:** Reseta a conexão
    - **SYN:** Estabelecimento de conexão
    - **FIN:** Origem finalizou seu stream de bytes. Encerramento da Conexão.

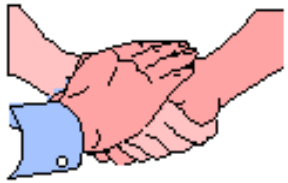
TCP

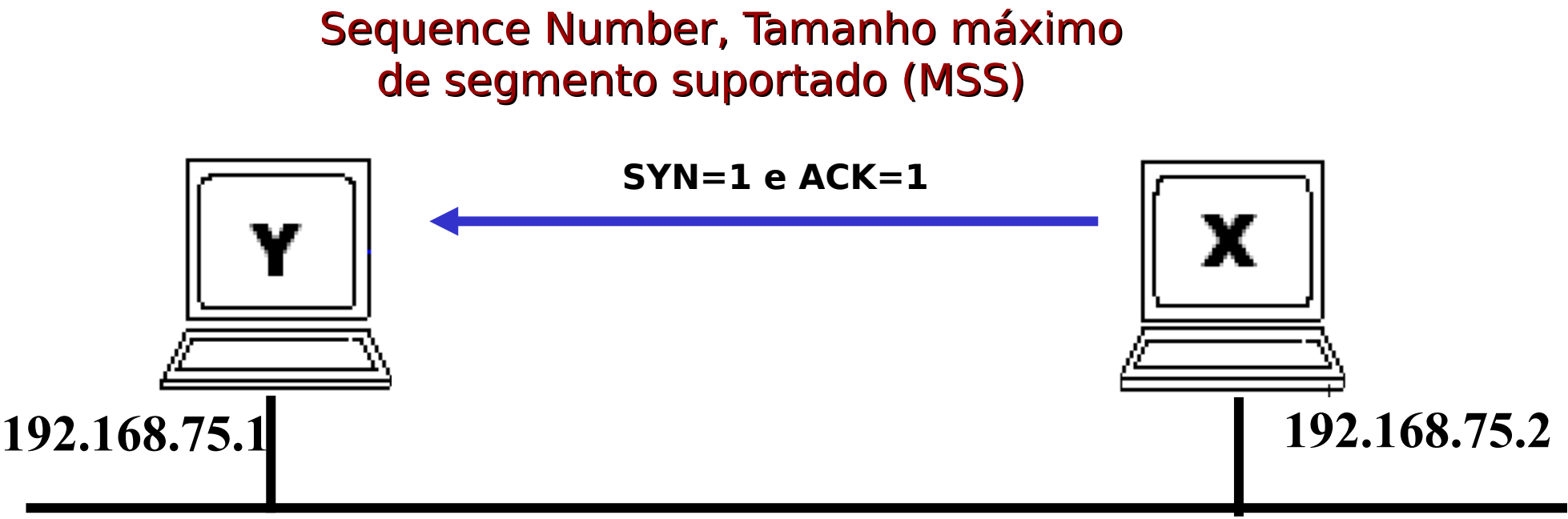
- Three-way Handshake 

Número de seqüência, Tamanho máximo de segmento suportado (MSS), dentre outros.

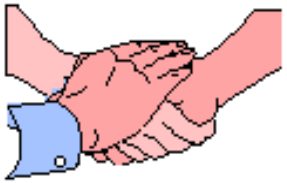


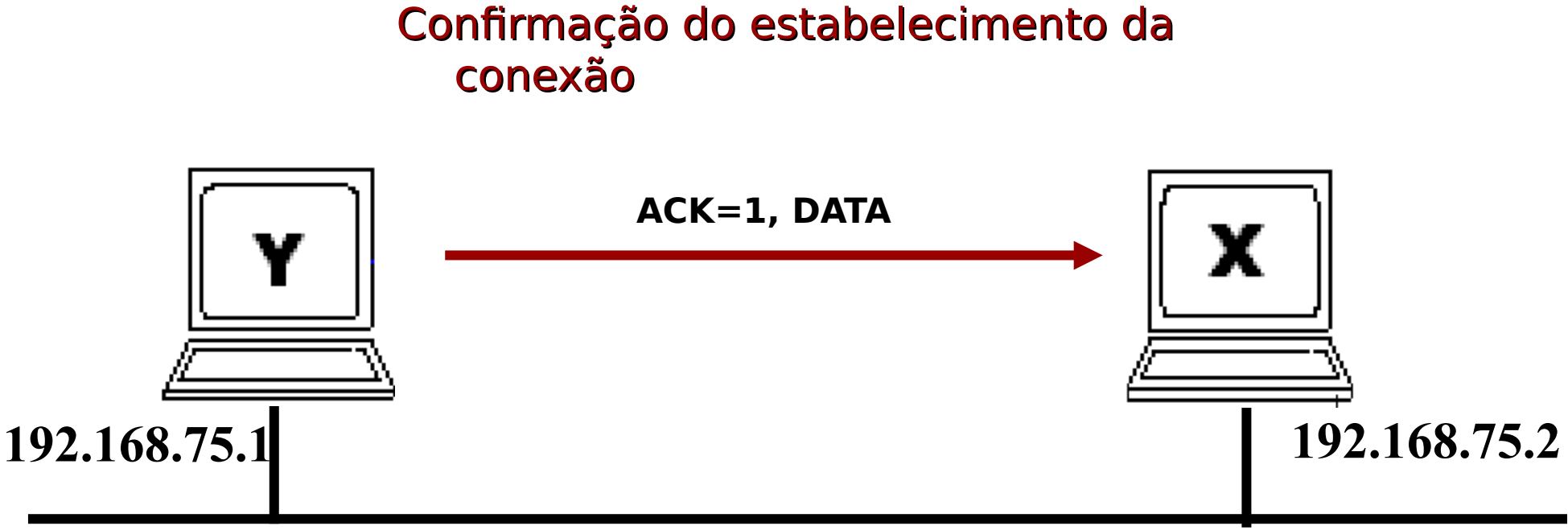
TCP

- Three-way Handshake 

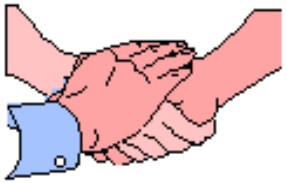


TCP

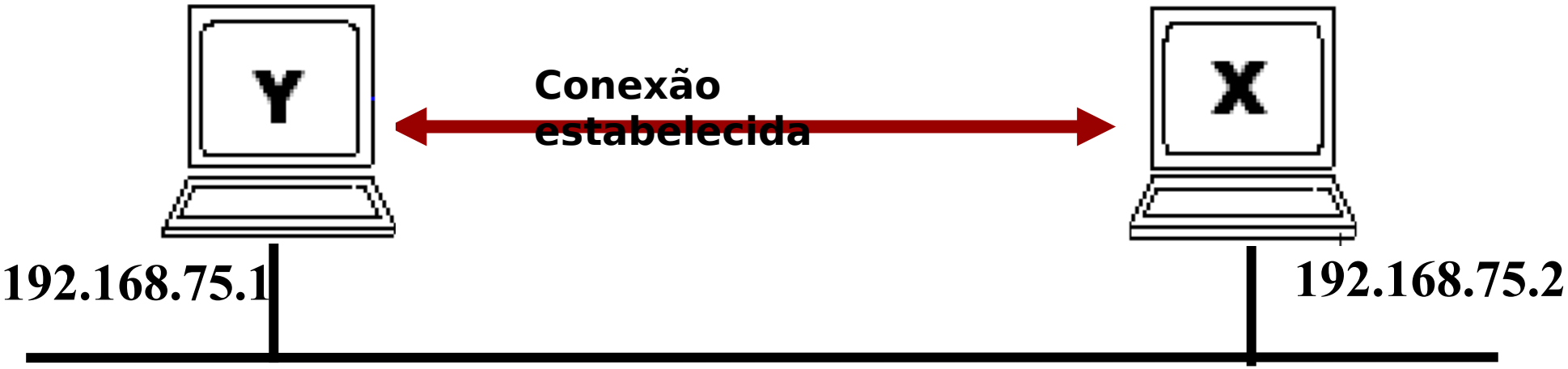
- Three-way Handshake 



TCP

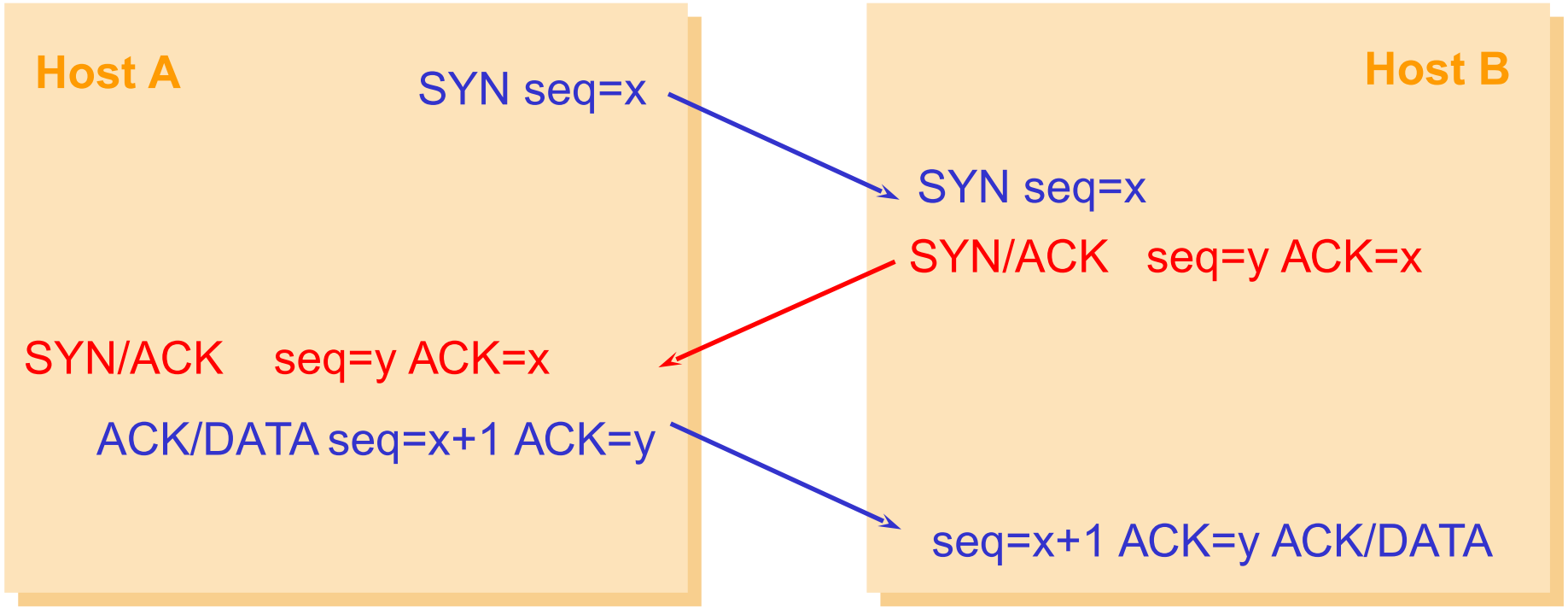
- Three-way Handshake 

Após estabelecer a conexão, todos os segmentos de dados trocados entre os hosts envolvidos têm o bit **ACK = 1**



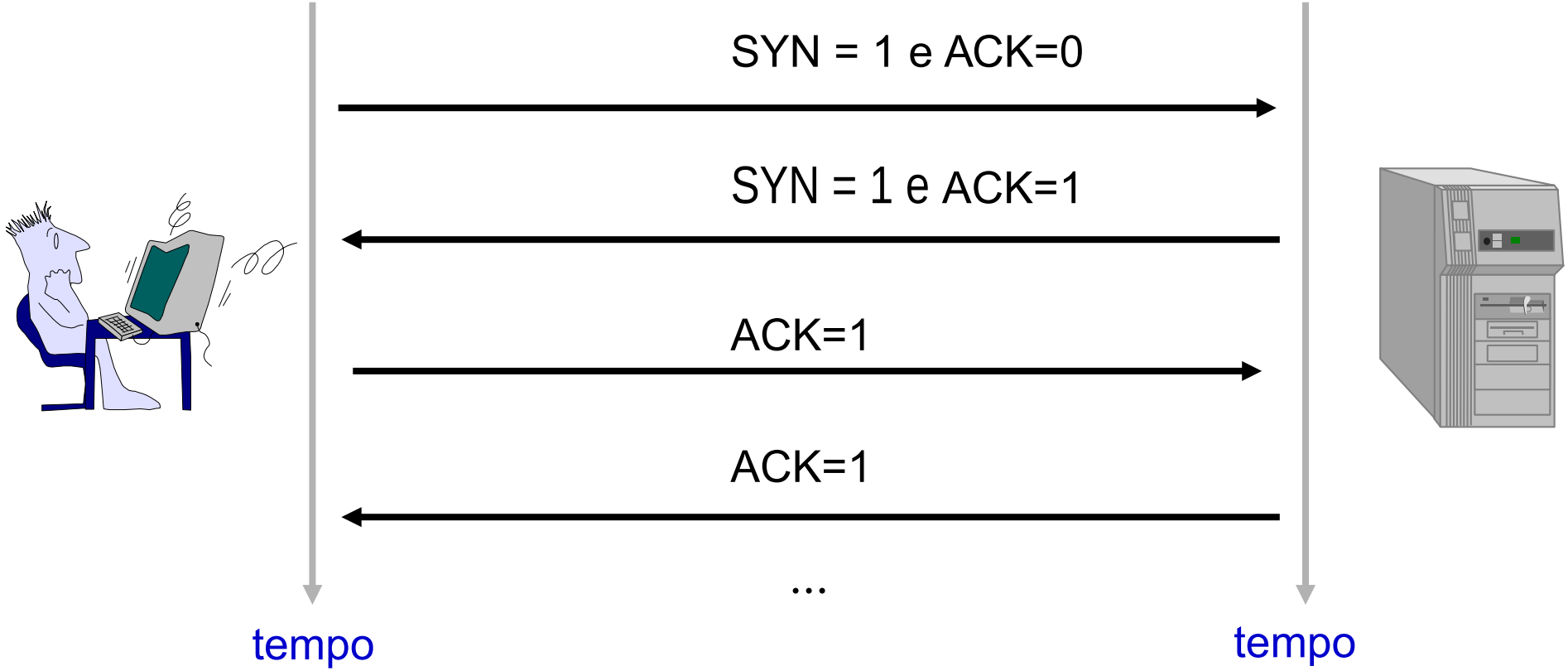
TCP

- Three-way Handshake



TCP

- Flag ACK
  - Uma conexão TCP sempre se inicia com o cliente enviando um pacote com o flag SYN = 1 e ACK= 0.



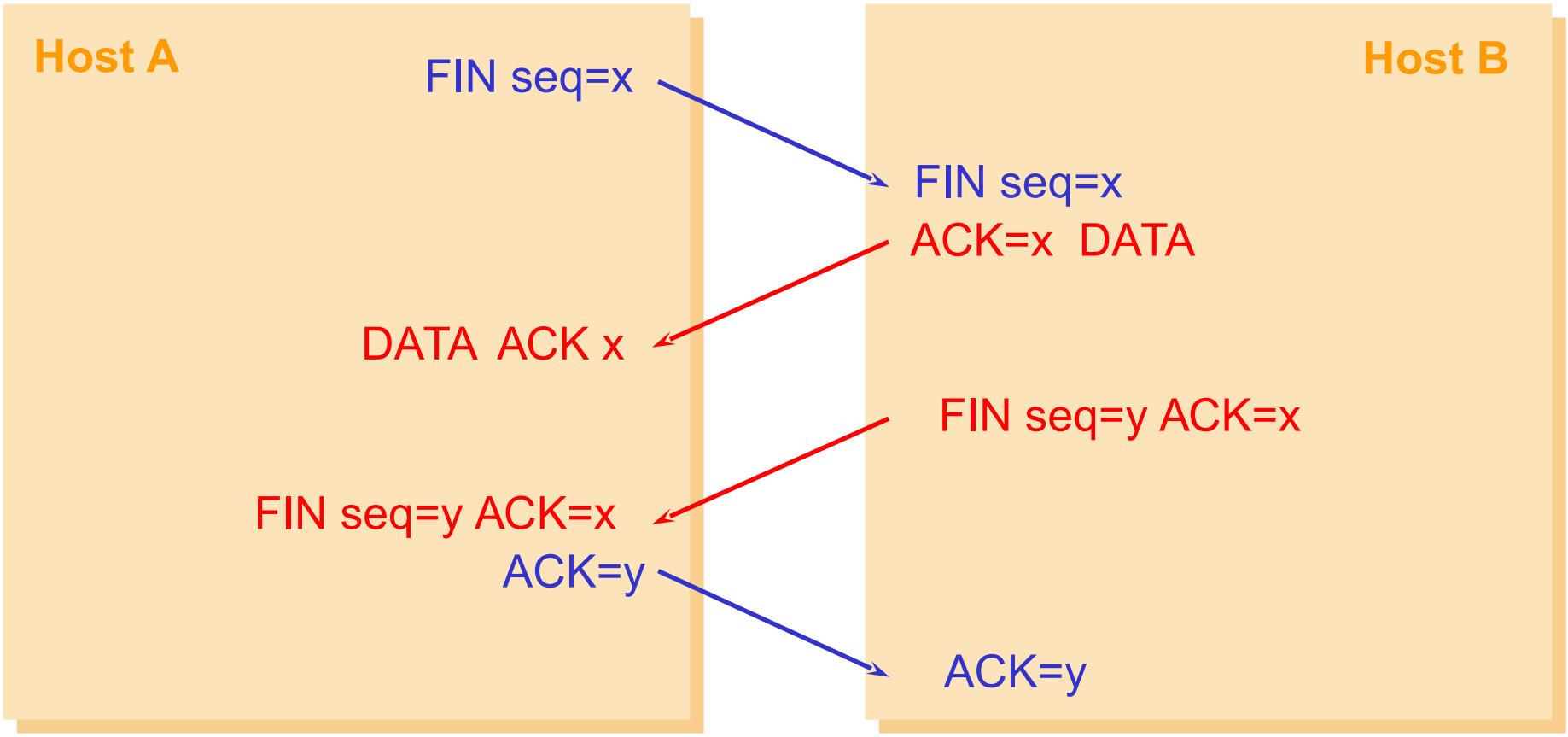
Características

- Encerramento da conexão por falha
  - Uma conexão termina depois de decorrido um certo tempo sem que chegue nenhuma T-PDU → trivial PDU: pacote sem dados efetivos
  - Se um lado desconectar ou falhar, o outro vai notar a falta de atividade e também se desconectar
  - Para evitar que uma conexão seja desfeita, os participantes devem assegurar o envio de T-PDUs periódicas informando que estão "vivos", quando não têm dados para transmitir;
  - Caso muitas T-PDUs se percam durante uma conexão, um dos lados pode fechar a conexão indevidamente;

Encerramento da Conexão

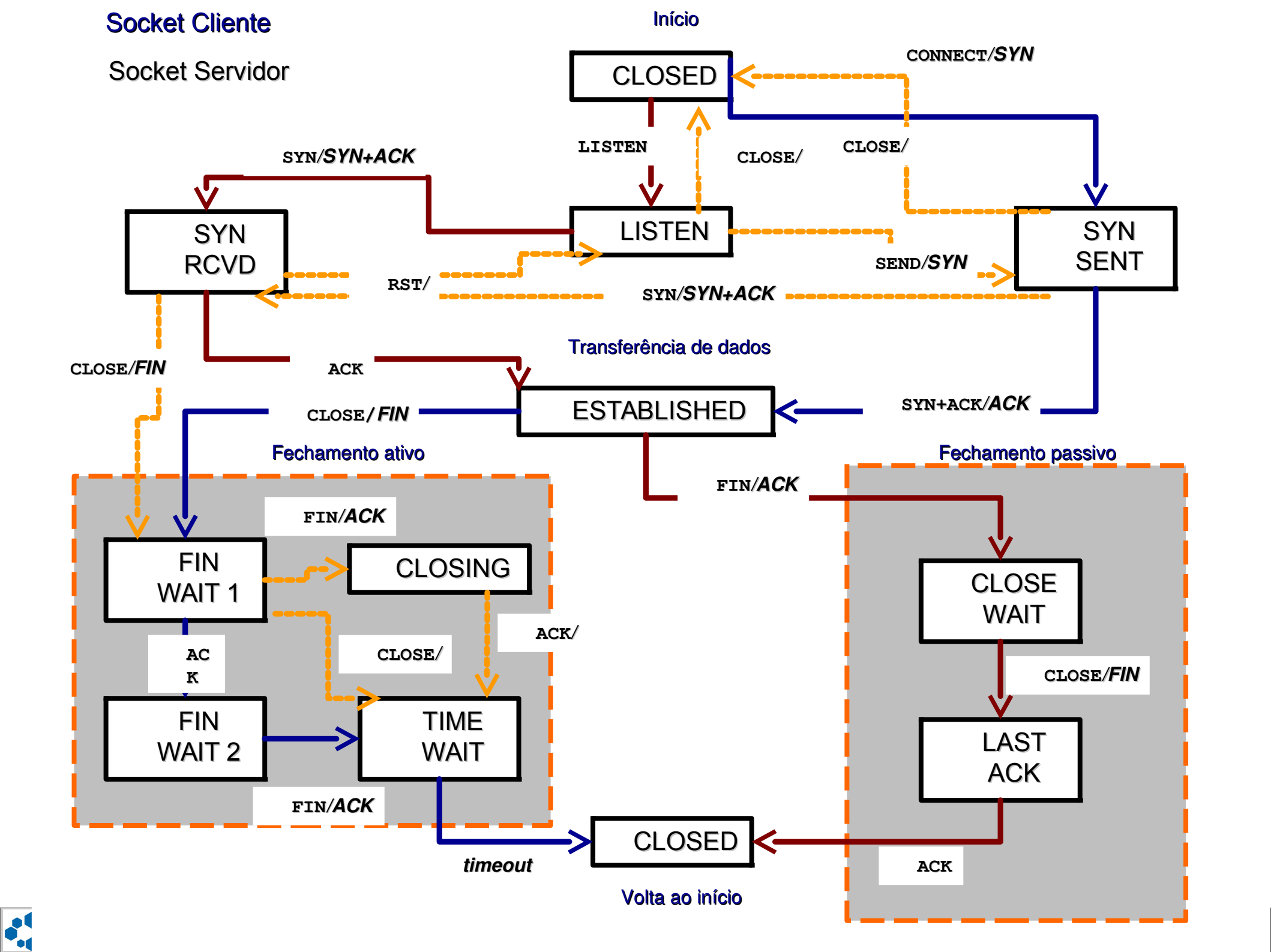
- Encerramento bem comportado da conexão: dois lados concordam e estão a par do encerramento
  - Um problema presente é evitar que dados sejam perdidos depois que um dos lados encerrou a conexão
  - Uma entidade de transporte ao pedir uma desconexão deve aguardar por um tempo antes de fechar a conexão, podendo receber dados durante esse período

Encerramento de Conexão



Estados da Conexão TCP

- **CLOSED**: não há conexão
- **LISTEN**: servidor esperando conexão
- **SYN RECV**: SYN recebido
- **SYN SENT**: aplicação iniciou conexão
- **ESTABLISHED**: transferência de dados
- **FIN WAIT 1**: aplicação finalizou transferência
- **FIN WAIT 2**: outro lado concordou com fechamento
- **TIMED WAIT**: espera por fim dos pacotes
- **CLOSING**: 2 lados tentaram fechar simultaneamente
- **CLOSE WAIT**: outro lado solicitou fim conexão
- **LAST ACK**: espera por fim dos pacotes


Socket Cliente
Socket Servidor
Início
CLOSED
CONNECT/SYN
LISTEN
CLOSE/
CLOSE/
SYN/SYN+ACK
SYN RCVD
LISTEN
SYN SENT
SEND/SYN
RST/
SYN/SYN+ACK
Transferência de dados
CLOSE/FIN
ACK
CLOSE/ FIN
ESTABLISHED
SYN+ACK/ACK
Fechamento ativo
Fechamento passivo
FIN/ACK
FIN WAIT 1
CLOSING
ACK/
AC K
CLOSE/
FIN WAIT 2
TIME WAIT
FIN/ACK
timeout
CLOSED
Volta ao início
CLOSE WAIT
CLOSE/FIN
LAST ACK
ACK

Roteiro

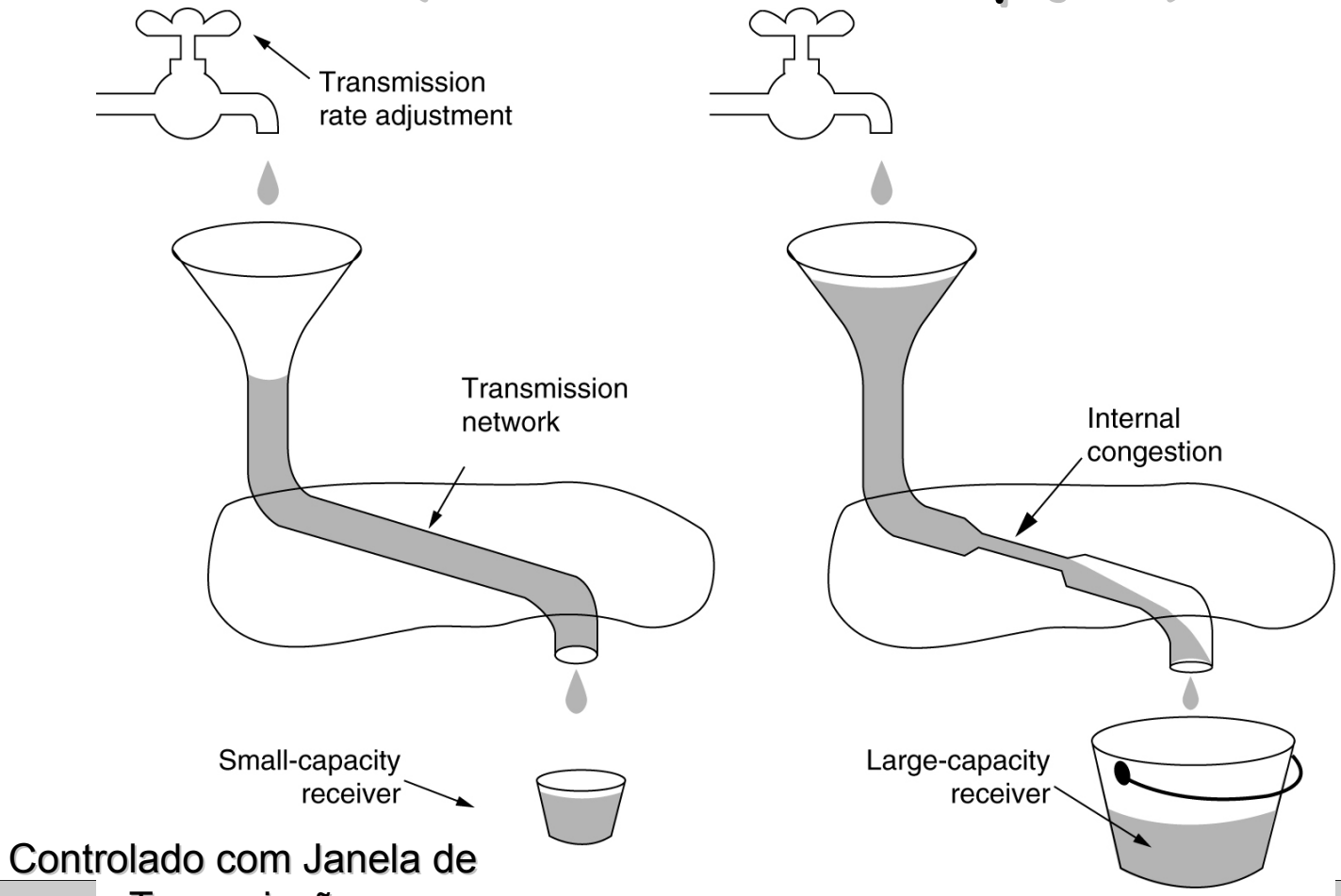
- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

Controle de Congestionamento

- Congestionamento
  - Rede não dá vazão ao volume de tráfego
  - Largura de banda diminui
  - Perda de mensagens aumenta
- Protocolo reativo
  - Diminuir a taxa de transmissão de pacotes
    - Aumentar a taxa de transmissão não melhora largura da banda
    - Piora o congestionamento

Congestionamento

- Congestionamento X Capacidade do Destinatário (Janela de Recepção)

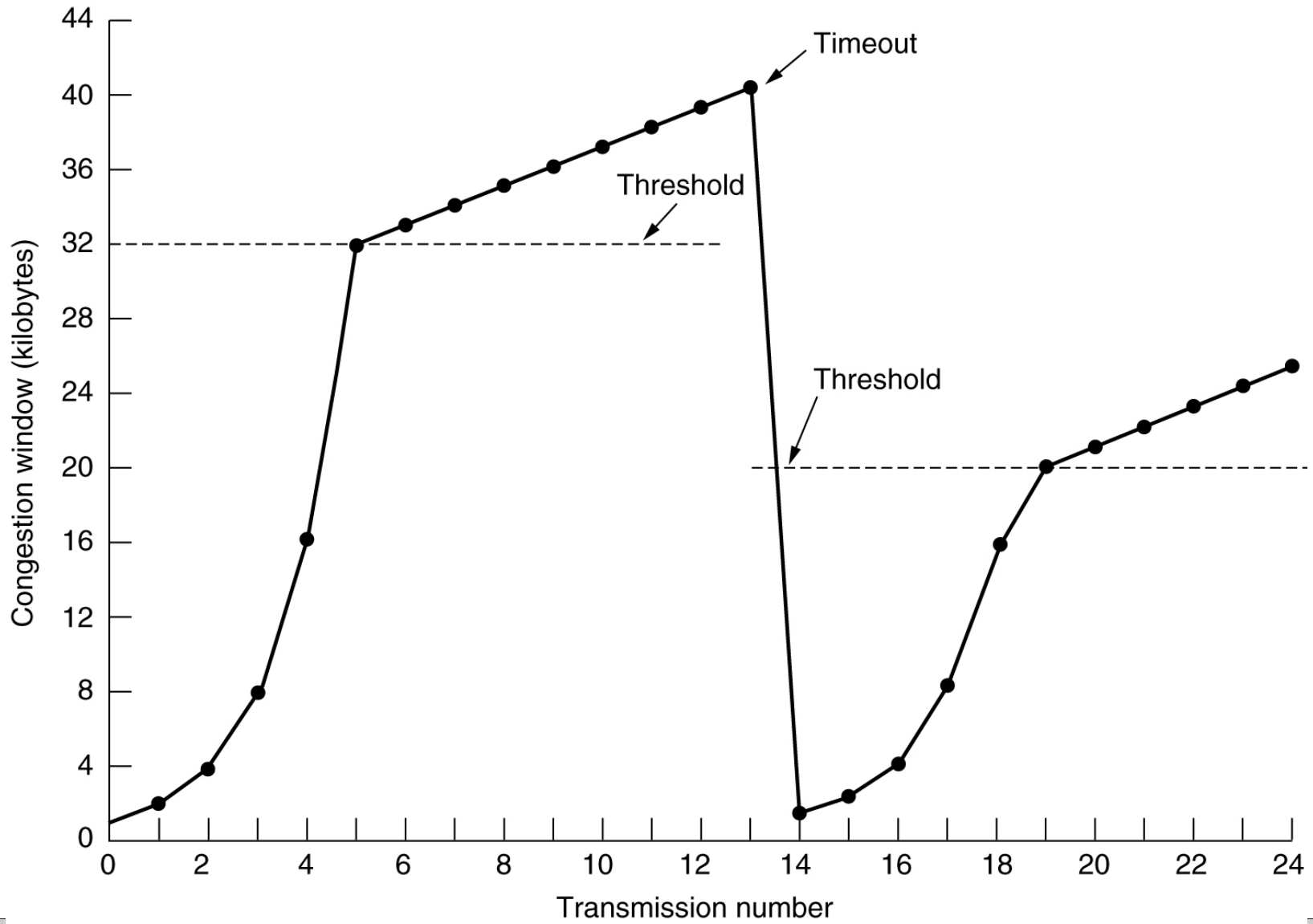


Controlado com Janela de Transmissão

Controle de Congestionamento

- Baseado em uma janela de congestionamento
  - Define quantidade de bytes que podem ser enviados
  - **Slow start**
    - Janela inicia com tamanho pequeno
    - Tamanho aumenta exponencialmente (duplica)
  - Threshold
    - Define um limiar a partir do qual a janela aumenta linearmente
  - Timeout
    - Evento de retorno da janela de congestionamento à posição inicial
    - Após um timeout, threshold assume a metade do threshold anterior.

Controle de Congestionamento



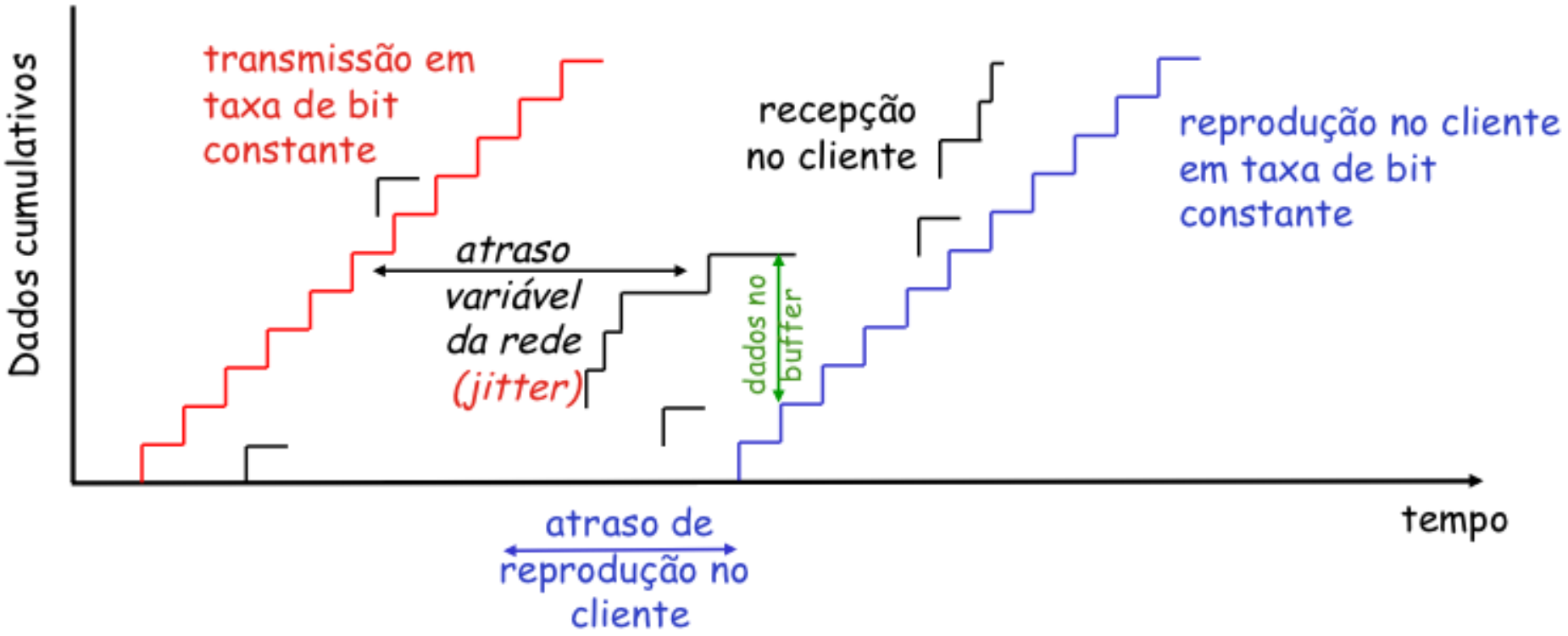
Roteiro

- Visão geral e objetivos
- Multiplexação
- Transmissão não-confiável e Protocolo UDP
- Mecanismos de transmissão confiável
- Protocolo TCP
- Estimativa de RTT e timeout
- Controle de Fluxo no TCP
- Gerenciamento de Conexão
- Controle de Congestionamento
- Requisitos de Aplicações e Camada de Transporte

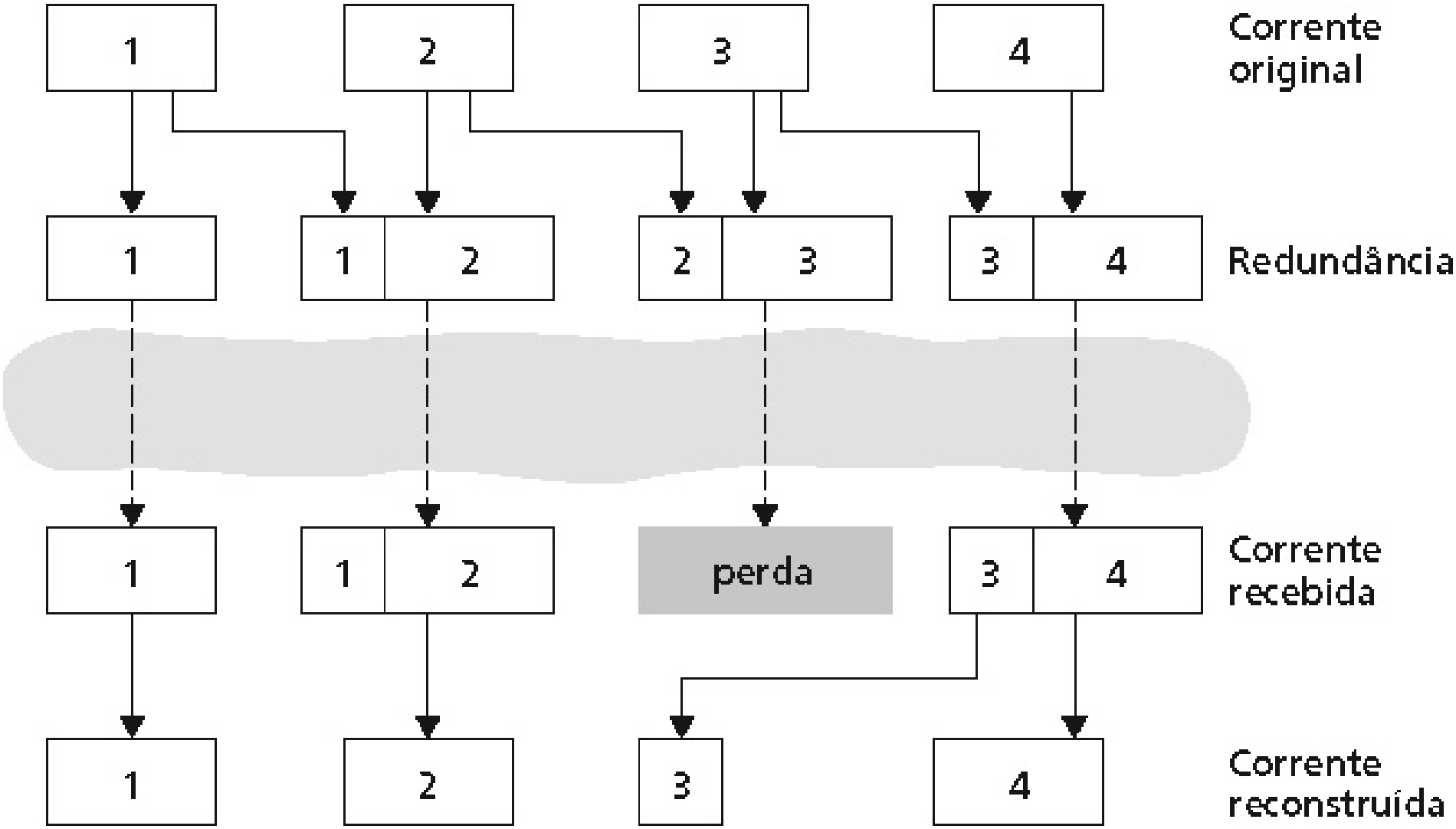
Efeitos Colaterais da adoção de TCP

- Momento de envio efetivo dos pacotes é definido pela camada de transporte e não pela aplicação
  - Velocidade de transmissão limitada pelo controle de congestionamento e fluxo
- Atraso no recebimento dos pacotes pela aplicação
  - Pacotes mantidos no buffer de recepção (segmentos faltantes aguardando retransmissão)
- Overhead/sobrecarga no sistema transmissor e receptor

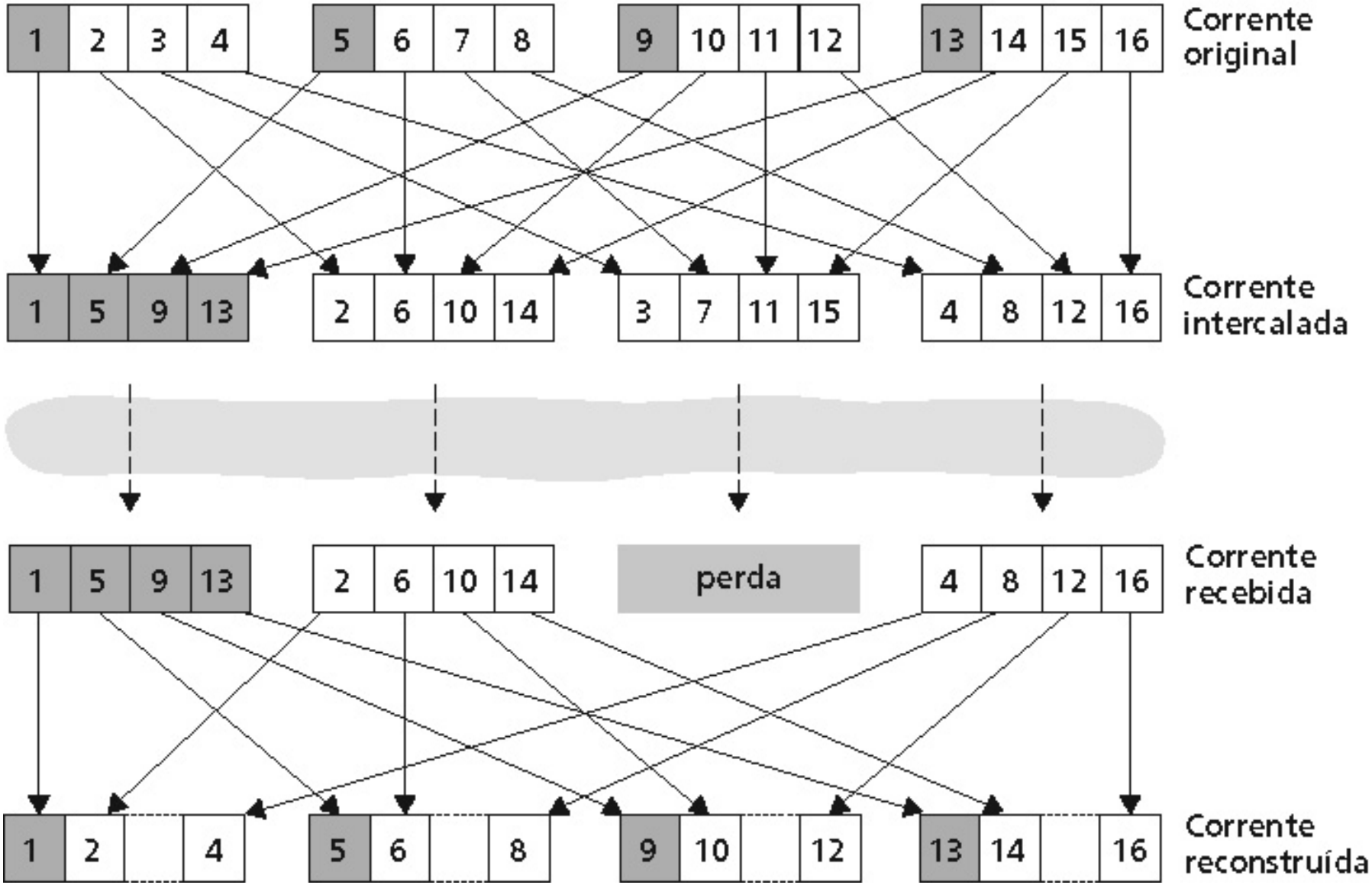
Problemas com Atraso em Multimidia



Exemplo de correção de erros em multimídia: FEC



FEC mais elaborado



Resumo e Conceitos-Chave

Referências

- Iraj Sodagar, *"The MPEG-DASH Standard for Multimedia Streaming Over the Internet,"* **IEEE Multimedia**, vol. 18, no. 4, pp. 62-67, October-December, 2011.