PMR3412 - Redes Industriais - 2023

Aula 04 - TCP e UDP (+ portas e sockets)

Prof. Dr. Newton Maruyama

31 de Agosto de 2023

PMR-EPUSP

Notas de autoria (Disclaimer) - Equipe envolvida

Os slides que serão utilizados nesse ano são baseados no curso desenvolvido para os anos 2020, 2021 e 2022. Participaram da concepção do curso e desenvolvimento do material os seguintes professores:

- Prof. Dr. André Kubagawa Sato
- Prof. Dr. Marcos de Sales Guerra Tsuzuki
- Prof. Dr. Edson Kenji Ueda
- Prof. Dr. Agesinaldo Matos Silva Junior
- Prof. Dr. André César Martins Cavalheiro

Conteúdo

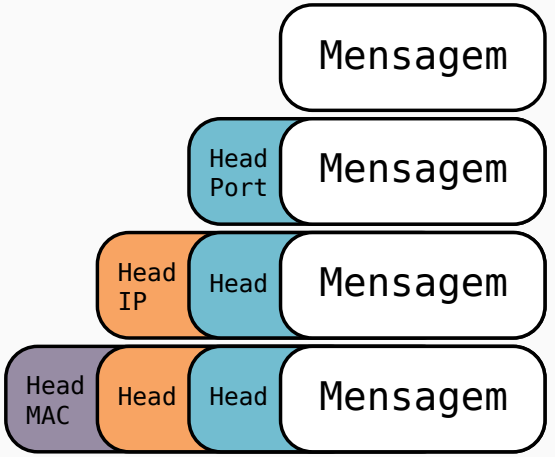
Revisão

Revisão - Relembrando o Modelo TCP/IP

OSI	TCP/IP	Ex.
7. Aplicação	4. Aplicação	HTTP, FTP, Telnet, DNS
6. Apresentação		
5. Sessão		
4. Transporte	3. Transporte	TCP, UDP
3. Rede	2. Internet	IP
2. Enlace	1. Enlace (Interface com Rede)	Ethernet
1. Física		

Dados

(simplificado)



Bits: 100001011011...

Uma Rápida Digressão mas não tão rápida

Digressão - Processos

- No contexto de sistemas operacionais o código do programa já foi denominado de *Job* e *Task*.
- Atualmente utiliza-se o termo *Processo* e *Thread*.
- Processos possuem um ciclo de vida como ilustrado abaixo:

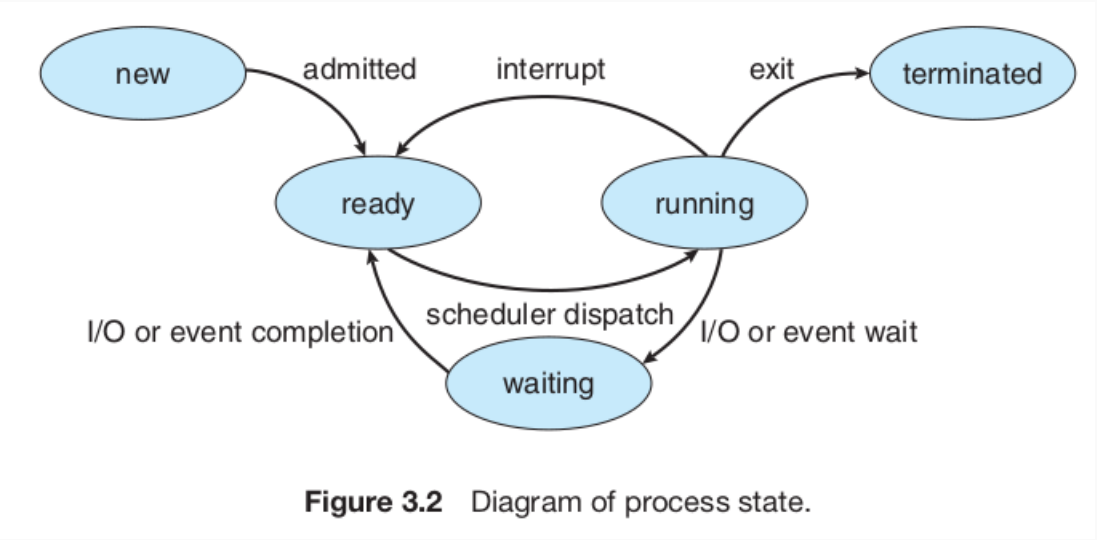


Figure 3.2 Diagram of process state.

- Um programa é interrompido porque pode necessitar de dados de um dispositivo de I/O, ou deve esperar por um evento (como comunicação com outro processo) ou ainda porque sua porção de execução se esgotou (*Time Slice*).
- O SO coordena a execução de múltiplos processos através de um módulo denominado Escalonador.

Digressão - Chaveamentos de Processos

- Qdo o escalonador decide retirar o processo em execução todas as informações (variáveis, pilhas, conteúdo dos registradores da CPU, ponteiro de instruções, etc.) são armazenadas numa estrutura de dados denominada PCB (*Process Control Block*).
- A figura abaixo ilustra as ações de um escalonador chaveando a execução de dois processos: P_0 e P_1.

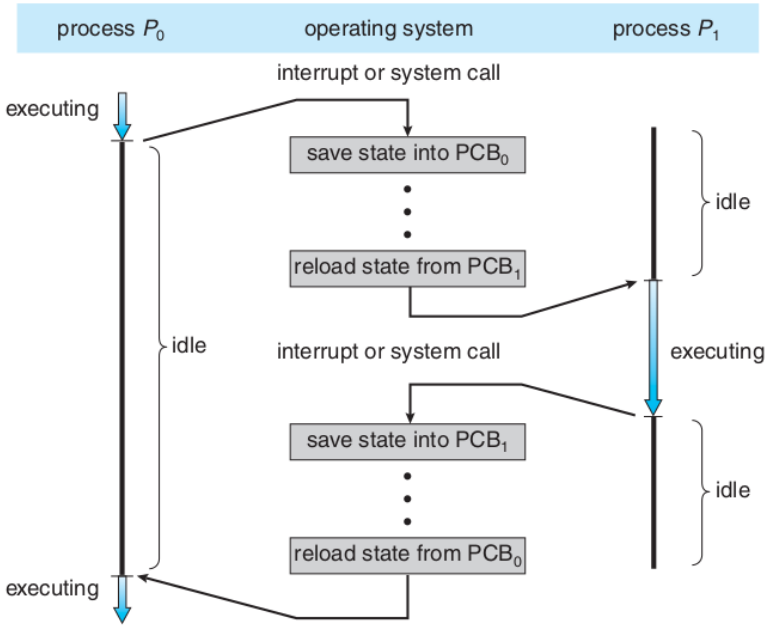


Figure 3.4 Diagram showing CPU switch from process to process.

Digressão - Threads

- Uma *thread* é muitas vezes definida como um processo leve (*Lightweight Process*).
- As *threads* são criadas dentro de um processo e compartilham códigos e dados. Permitem maior eficiência de execução. Ex: Operações simultâneas em editores de texto, browsers, etc.
- Anteriormente a computação era baseada em processos independentes.
- Qdo o conceito de *thread* foi introduzido os processos antigos foram denominados: *single-threaded process*.
- A figura abaixo ilustra os conceitos de processo *single-threaded* e *multi threaded*.

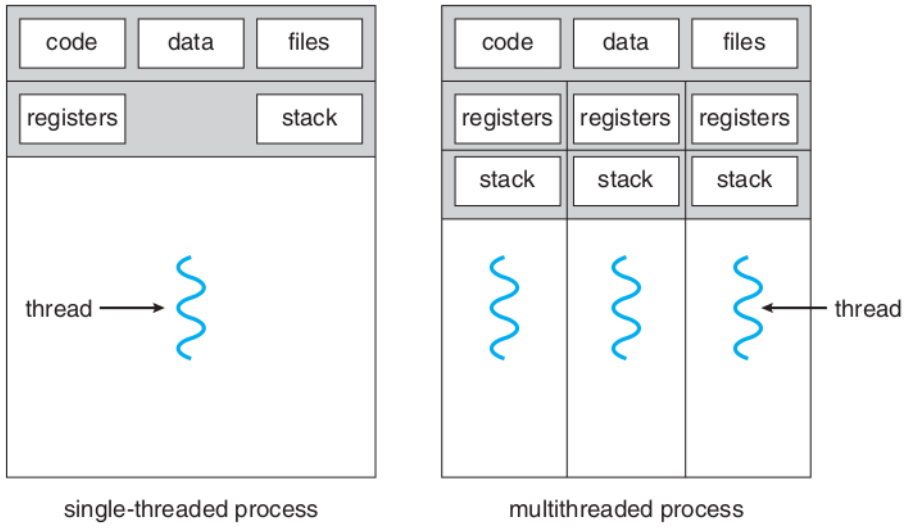
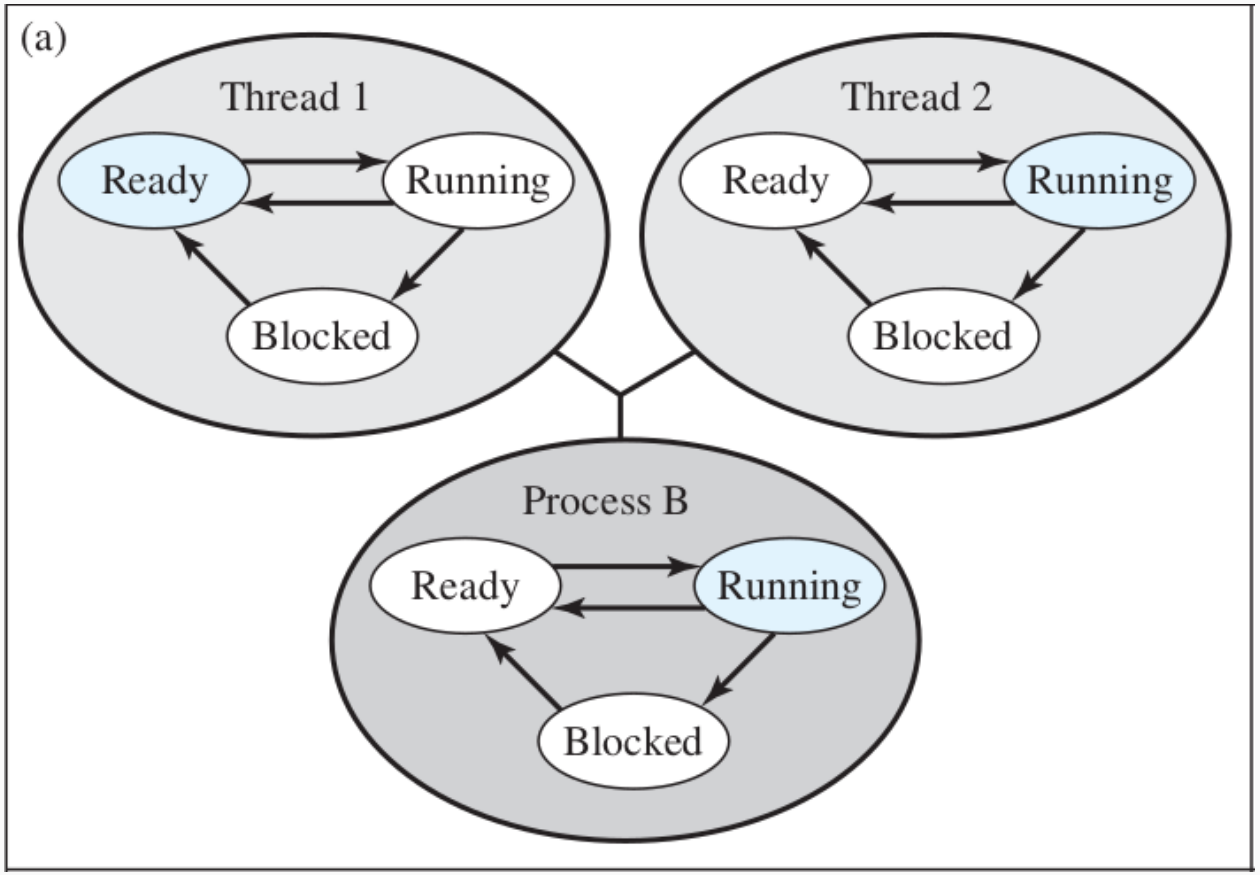


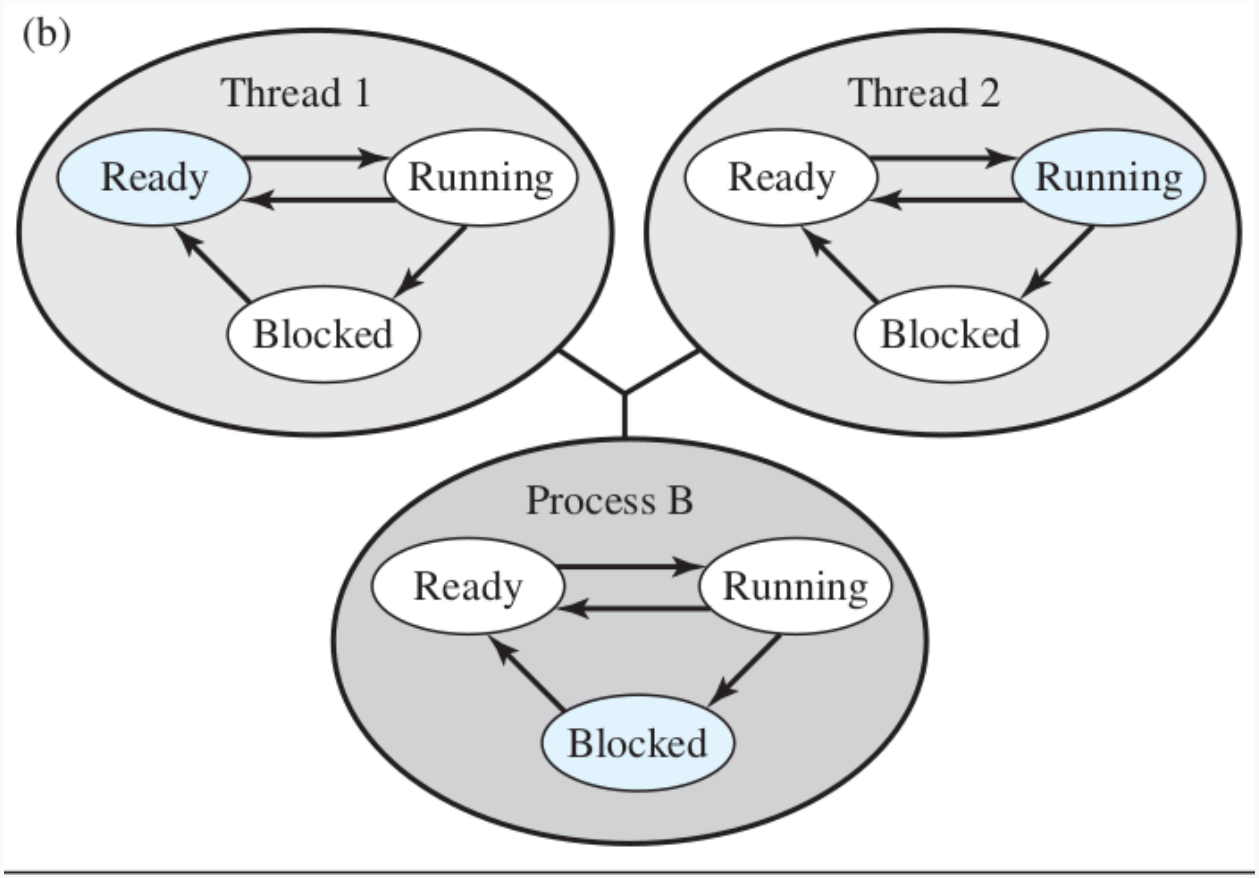
Figure 4.1 Single-threaded and multithreaded processes.

Digressão - threads states × process states



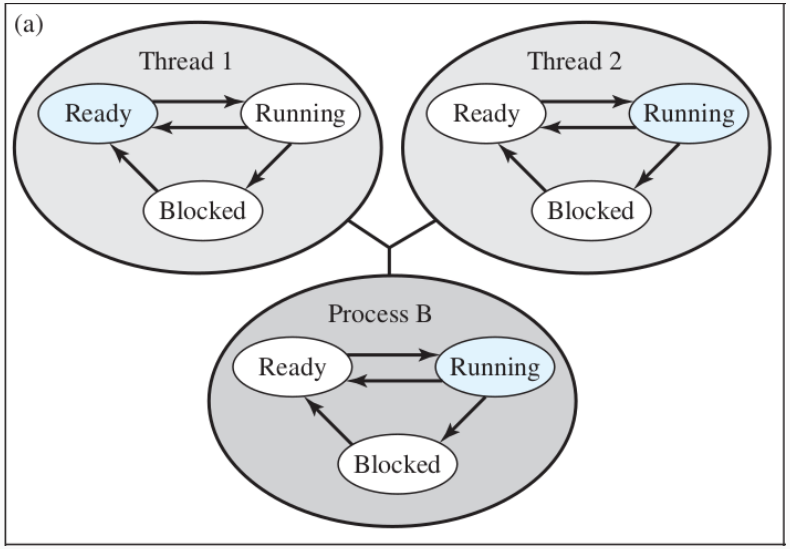
Process P_B is executing in its thread t_2. The application executing in t_2 makes a system call that blocks B. For example, an IO call is made. This causes control to transfer to the kernel. The kernel invokes the IO action, places P_B in the blocked state and switches to another process.

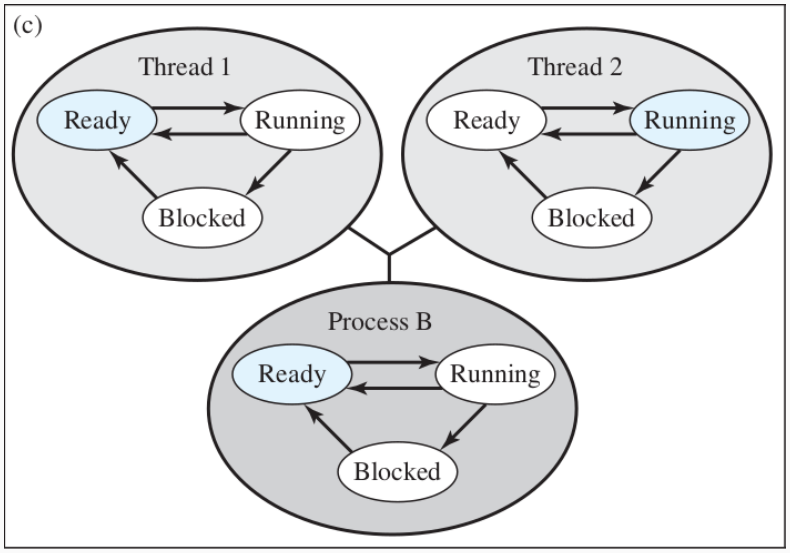
Digressão - threads states × process states



Meanwhile, according to the data structure maintained by the threads library, t_2 of P_B is still in the running state. It is important to note that t_2 is not actually running in the sense of being executed, but it is perceived as being in the running state by the threads library.

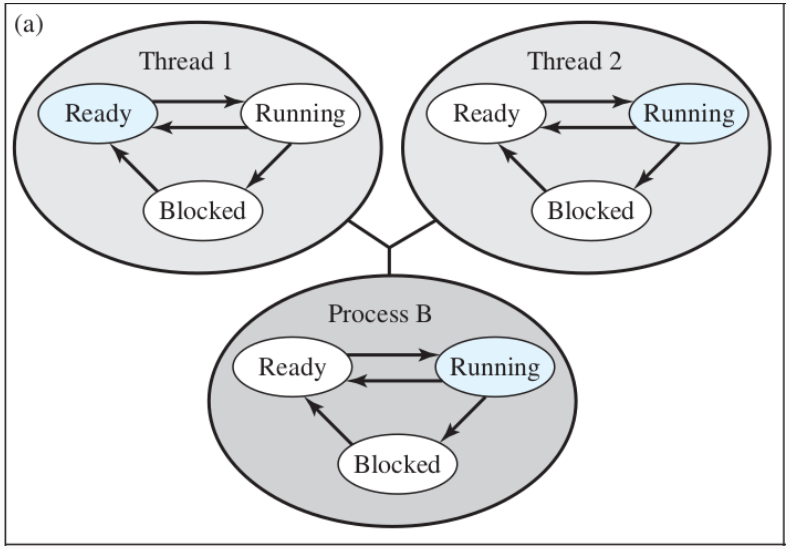
Digressão - threads states × process states

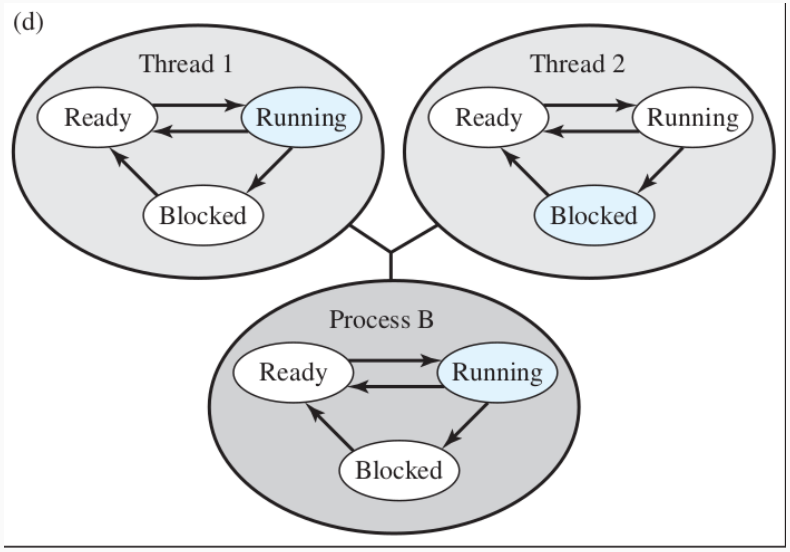




Process P_B is executing in its thread t_2. A clock interrupt passes control to the kernel and the kernel determines that the currently running process P_B has exhausted its time slice. The kernel places P_B in the ready state and switches to another process. Meanwhile, t_2 of P_B is still in the running state.

Digressão - threads states × process states





Process P_B is executing in its thread t_2. t_2 has reached a point where it needs some action performed by thread 1 of process B. t_2 enters a blocked state and thread 1 transitions from ready to running. The process itself remains in the running state.

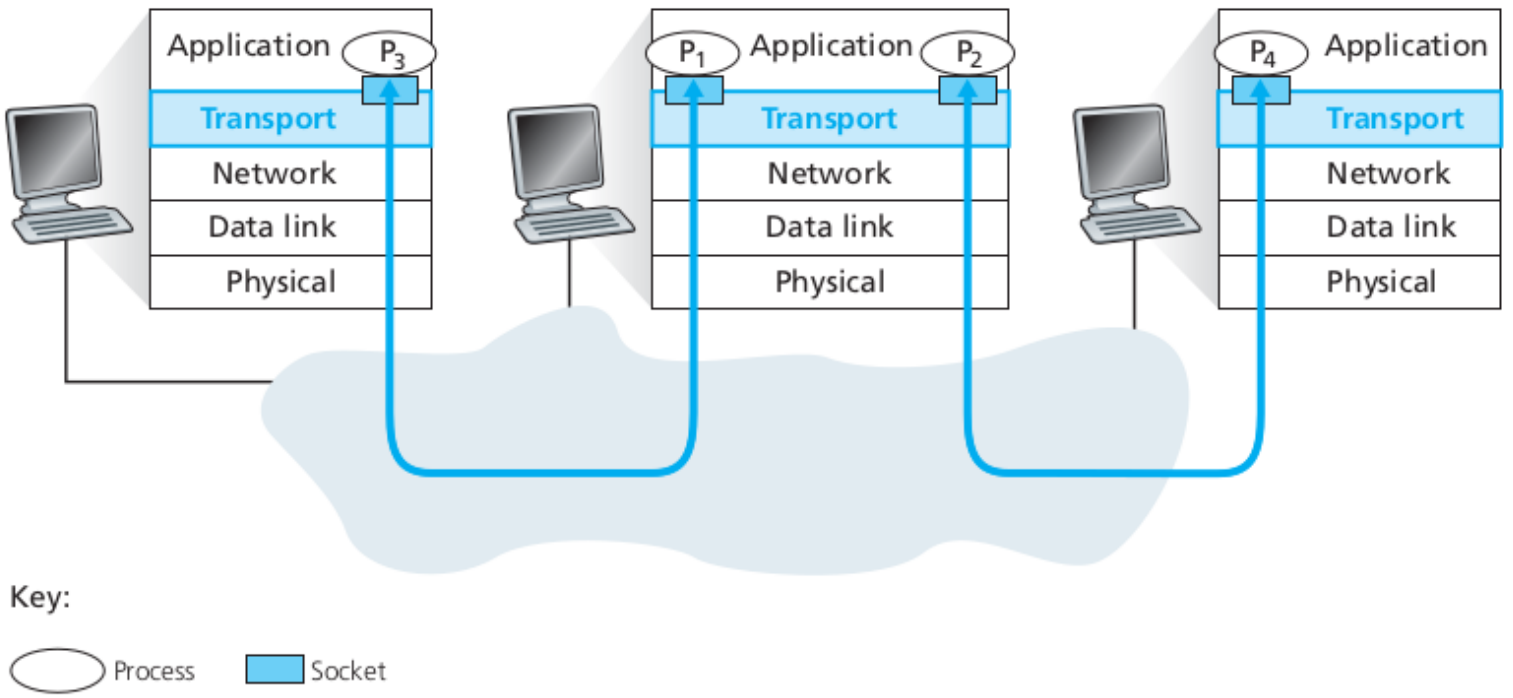


Figure 3.2 ♦ Transport-layer multiplexing and demultiplexing

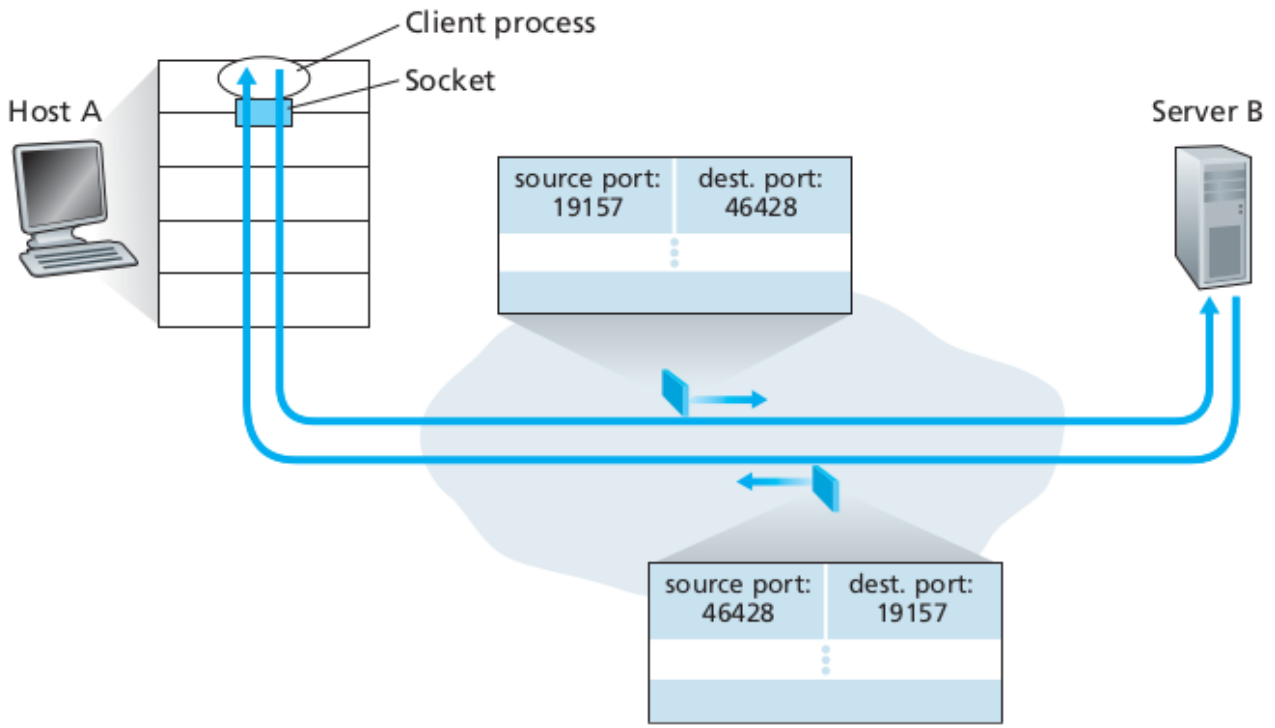


Figure 3.4 ♦ The inversion of source and destination port numbers

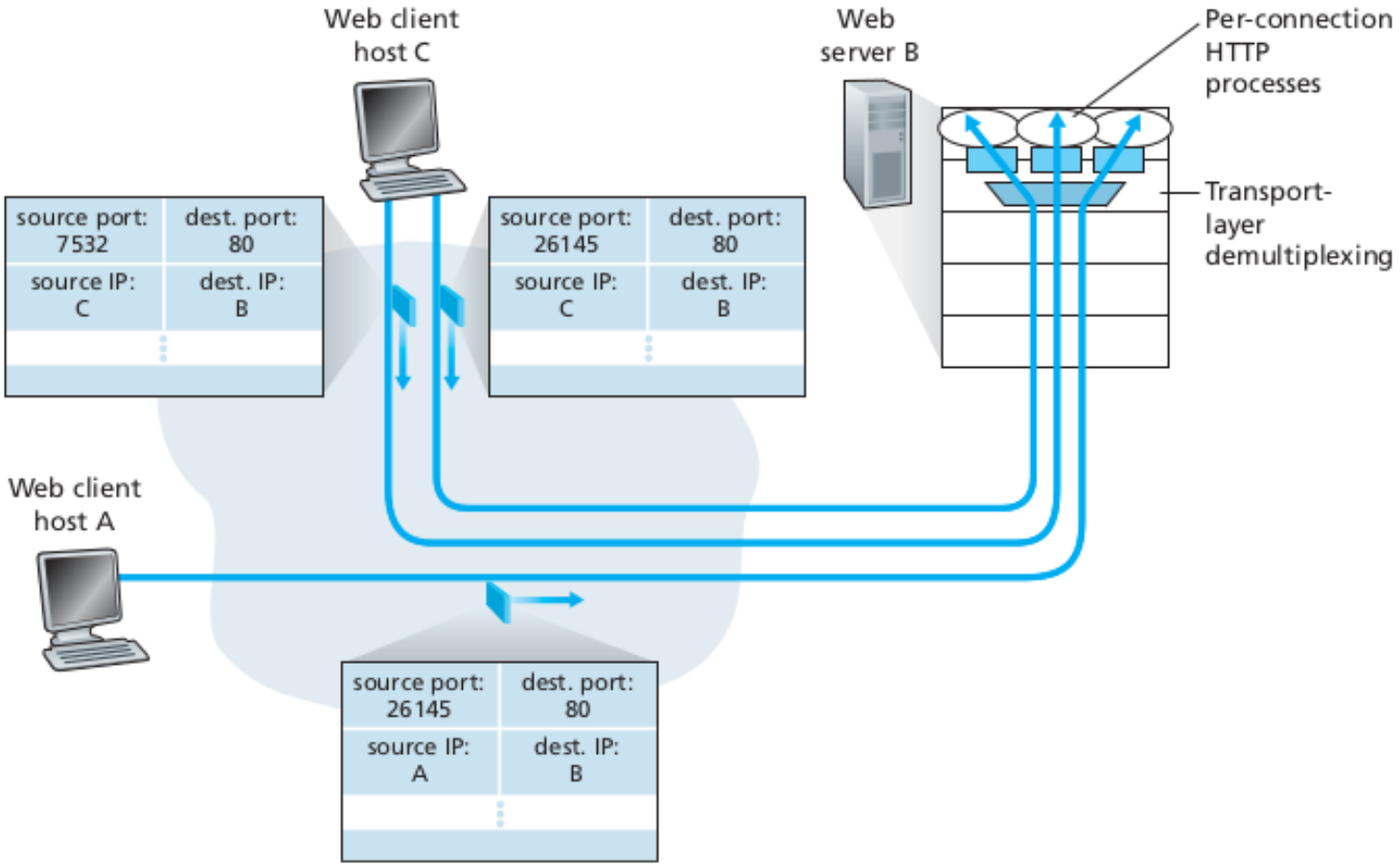


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

Portas

Portas - Introdução

- As Camadas de Enlace (1) e de Internet (2) garantem a entrega de um pacote para um *host* na internet / Internet.
- No entanto, um *host* ou servidor geralmente executa diversos processos (programas). Então é necessário algum mecanismo para determinar para qual processo a mensagem é destinada.
- Além disso, um servidor pode possuir múltiplas conexões com múltiplos *hosts* ao mesmo tempo.
- Por estes motivos, é necessário utilizar portas para identificar as conexões e os programas que estão envolvidos na comunicação.

Portas - Classificação

- Cada processo que deseja se comunicar com outro processo através do protocolo TCP/IP se identifica através de uma ou mais portas
- A porta é representada por um número inteiro de 16-bits e pode ser de dois tipos:

1. Portas Efêmeras (1024 até 65535): geralmente utilizada pelos clientes ao iniciar a comunicação.
2. Portas Conhecidas[1], ou *Well-known Ports* (1 até 1023): possibilita encontrar servidores sem configuração prévia. Exemplos:

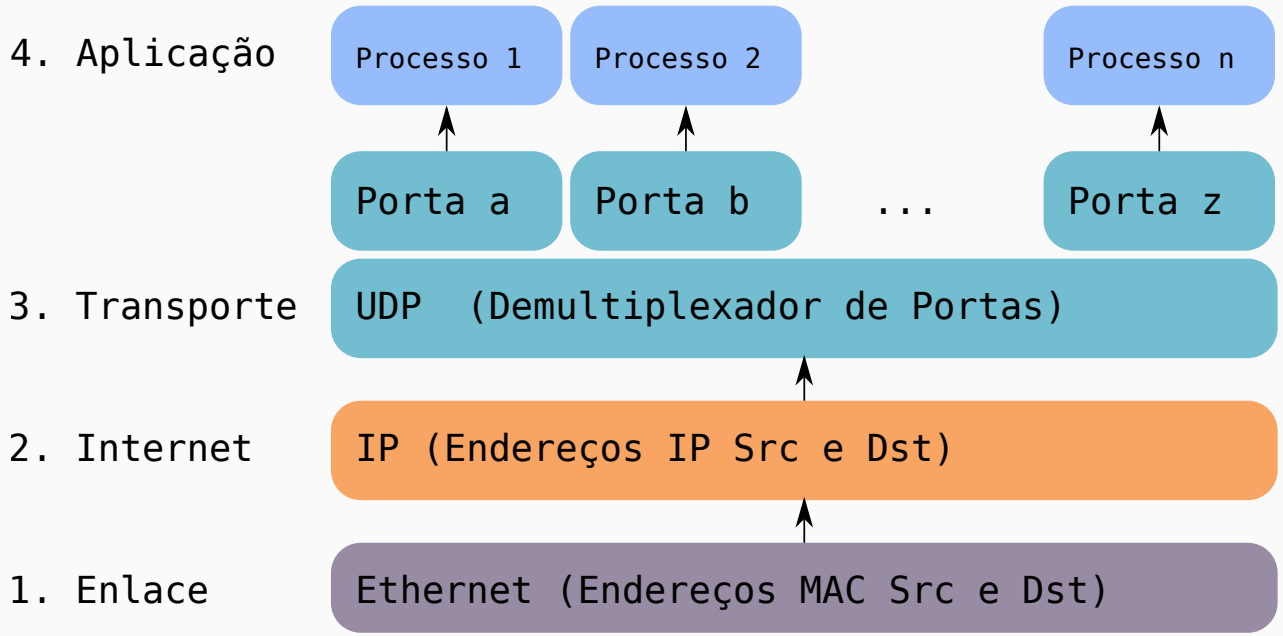
Serviço	Porta	Função
HTTP	80	Tráfico Web
HTTPS	443	Tráfico Web Seguro
FTP	20, 21	Transferência de Arquivo
DNS	53	Resolução de Nome
SMTP	25	Email
POP3	110	Caixa de Correio POP
IMAP	143	Caixa de Correio IMAP
Telnet	23	Acesso Remoto
SSH	22	Acesso Remoto Seguro

[1] definido pela IANA - International Assigned Number Authority.

User Datagram Protocol (UDP)

UDP - Introdução

- O *User Datagram Protocol* (UDP) não provê confiabilidade, controle de fluxo e recuperação de erro ao protocolo IP.
- Assim, o UDP é basicamente uma interface de aplicação ao protocolo IP, já que apenas faz a ligação dos datagramas às aplicações correspondentes.
- Por este motivo, o UDP é mais rápido do que o TCP, o que é sua grande vantagem.



UDP - Formato do Datagrama

- Src e Dst Port: portas de origem e destino.
- Length: tamanho em bytes do datagrama completo.
- Checksum: feito em cima do cabeçalho, dados e um pseudo-cabeçalho de IP (que contém o end. IP do remetente/destinatário, o protocolo e o tamanho do datagrama UDP).

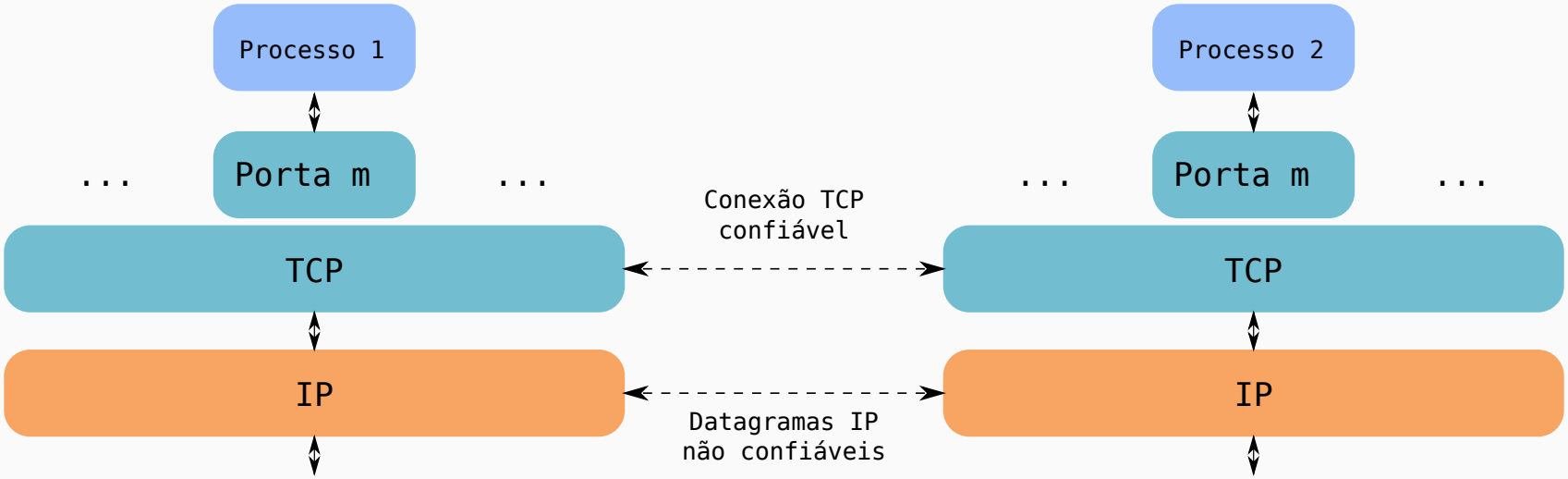
UDP - Aplicações UDP

- Uma interface de programação de aplicações UDP deve fornecer funcionalidades para:
  - criar novas portas de recepção;
  - operação para recebimento que retorna, além dos bytes recebidos, indicações da porta e endereço de origem;
  - operação de envio, que recebe como parâmetros os dados, as portas e endereços de origem e destino.
- Algumas importantes aplicações UDP incluem: Trivial File Transfer Protocol (TFTP): , Domain Name System (DNS), Simple Network Management Protocol (SNMP), Lightweight Directory Access Protocol (LDAP).

TCP

TCP - Introdução

- Ao contrário do protocolo UDP, o protocolo Transmission Control Protocol (TCP) provê um conjunto maior funcionalidades: confiabilidade, controle de fluxo e recuperação de erro.
- Além disso, o TCP é um protocolo orientado a conexão (*Connection Oriented*), isto é, requer que dois *hosts* iniciem uma conexão antes de iniciar uma comunicação.



TCP - Funcionalidades

Entrega de Fluxo de dados

- Dentro da perspectiva da aplicação, o TCP transfere um fluxo contíguo de bytes. O TCP se responsabiliza em criar e gerenciar blocos de bytes denominados *Segmentos*.

Confiabilidade

- Um número de sequência é atribuído a cada byte enviado e a camada TCP espera uma confirmação (ACK) do destinatário.

Controle de Fluxo

- O destinatário, através do envio de ACKs, indica a quantidade de bytes que pode receber. Assim, evita o *overflow* dos seus *buffers* internos.

Multiplexação

- Através de portas, assim como o UDP.

Conexões Lógicas

- Para cada fluxo de dados, é necessário manter as informações relativas a essa conexão, que incluem os *sockets* [a], números de sequência e tamanho de janela.

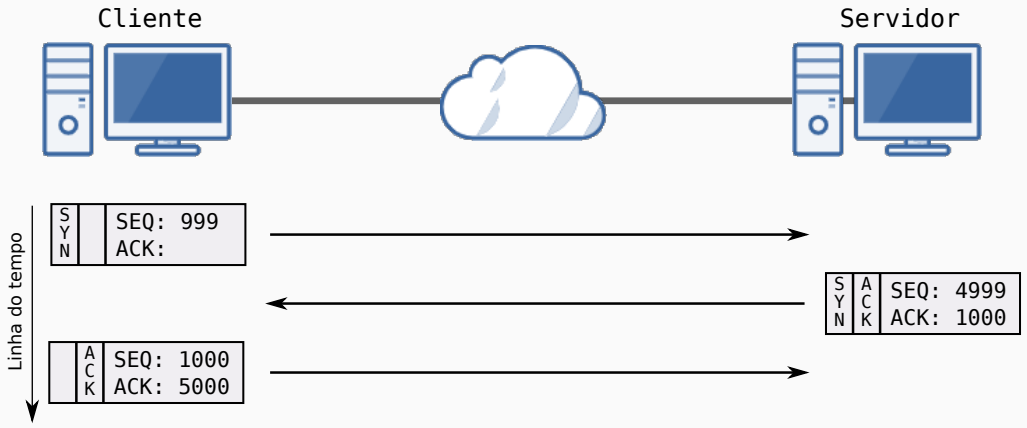
Full-Duplex

- Fluxo de dados concomitantes em ambas direções.

[a] API que implementa comunicação entre processos (Inter-processes communication) utilizando o protocolo TCP/IP. Os processos podem estar na mesma máquina ou distribuídos pela rede.

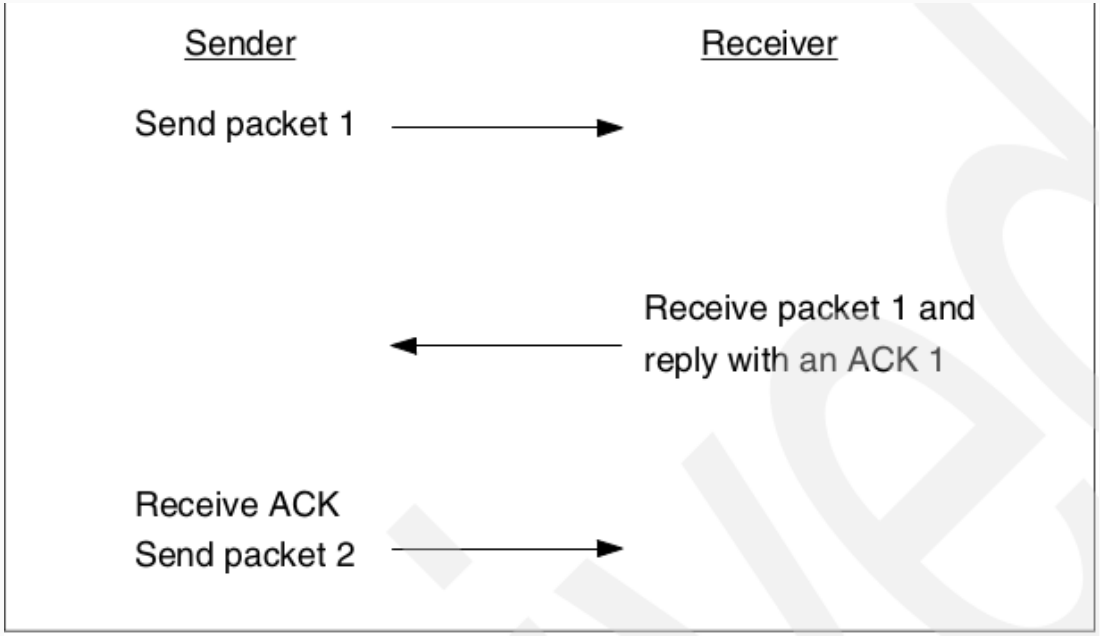
TCP - Estabelecendo uma conexão

- Antes de iniciar uma comunicação, uma conexão deve ser estabelecida entre os processos.
- Handshake de Três Vias:



TCP - O Princípio da Janela

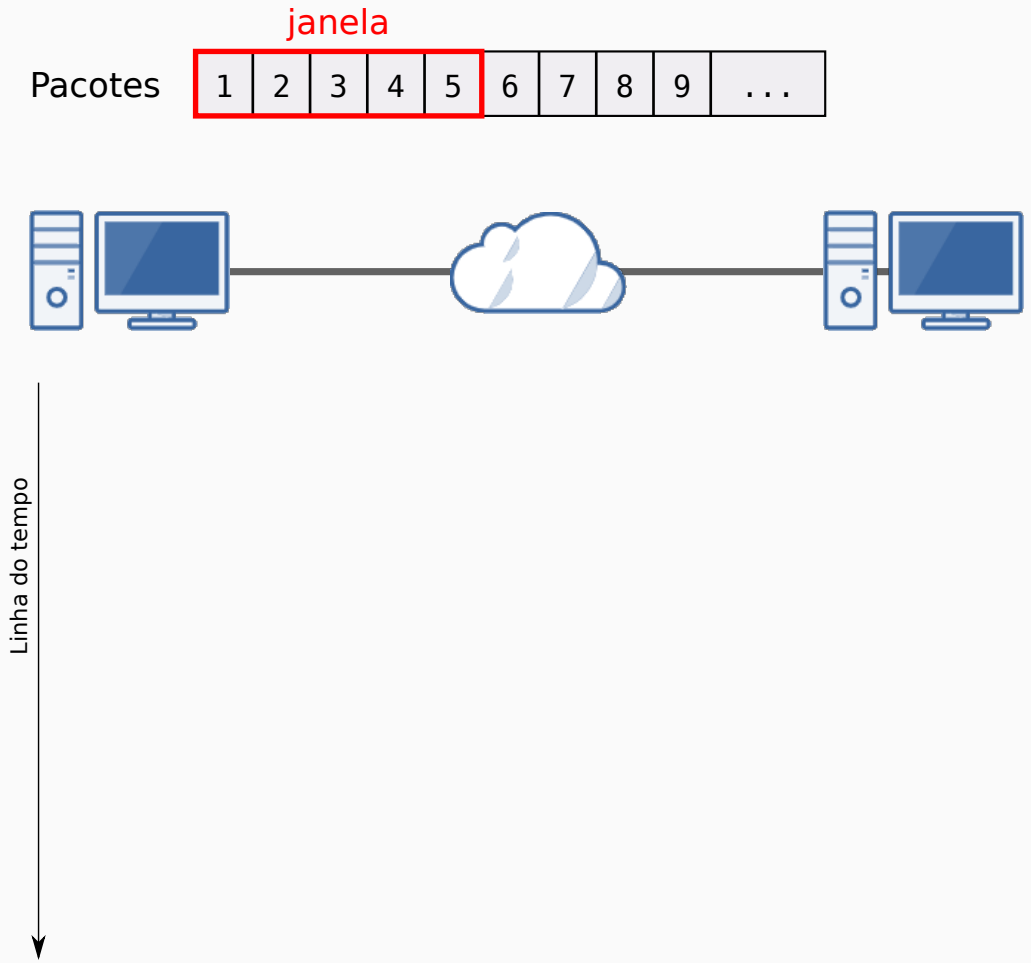
- Em primeiro lugar, apresenta-se aqui o "Princípio da Janela" que contém as idéias principais do controle de fluxo.
- Posteriormente apresenta-se o princípio da janela adaptado ao protocolo TCP.
- Inicialmente, vamos analisar um possível algoritmo de transmissão aqui denominado "Naive Scheme":



- Esse algoritmo pode funcionar ?

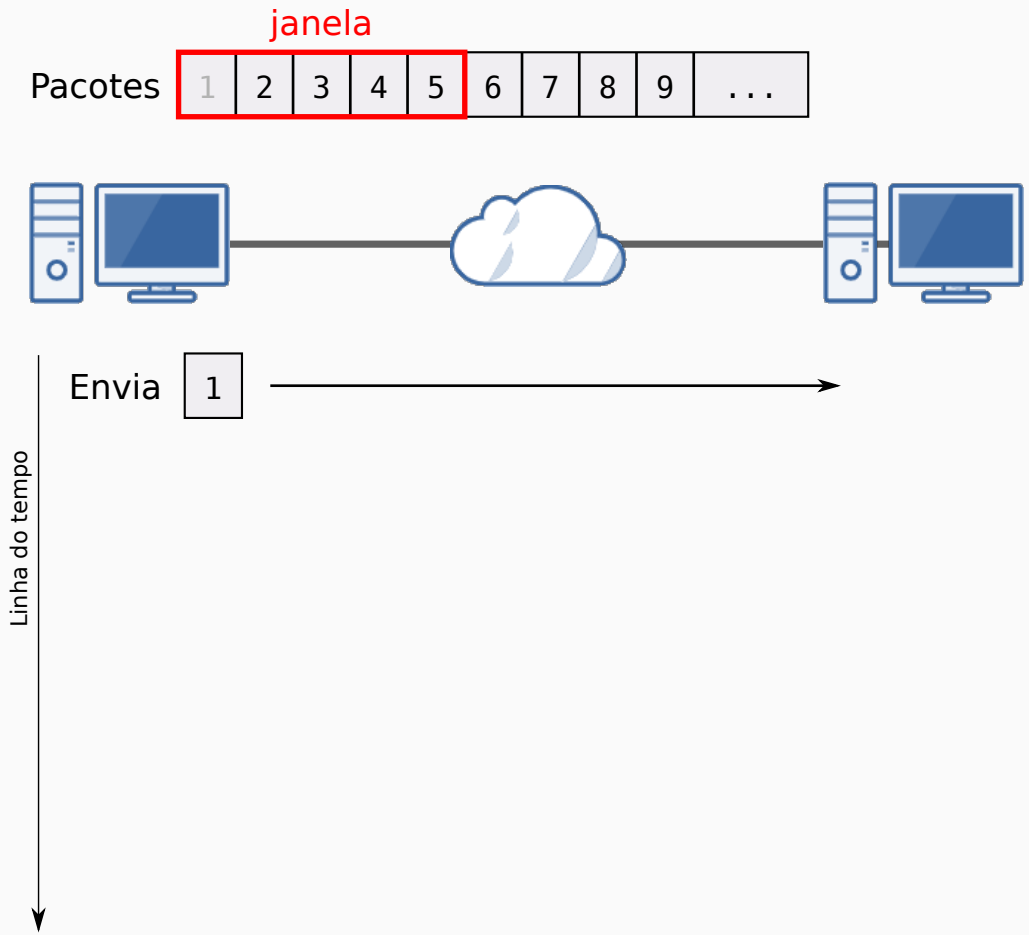
TCP - O Princípio da Janela

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



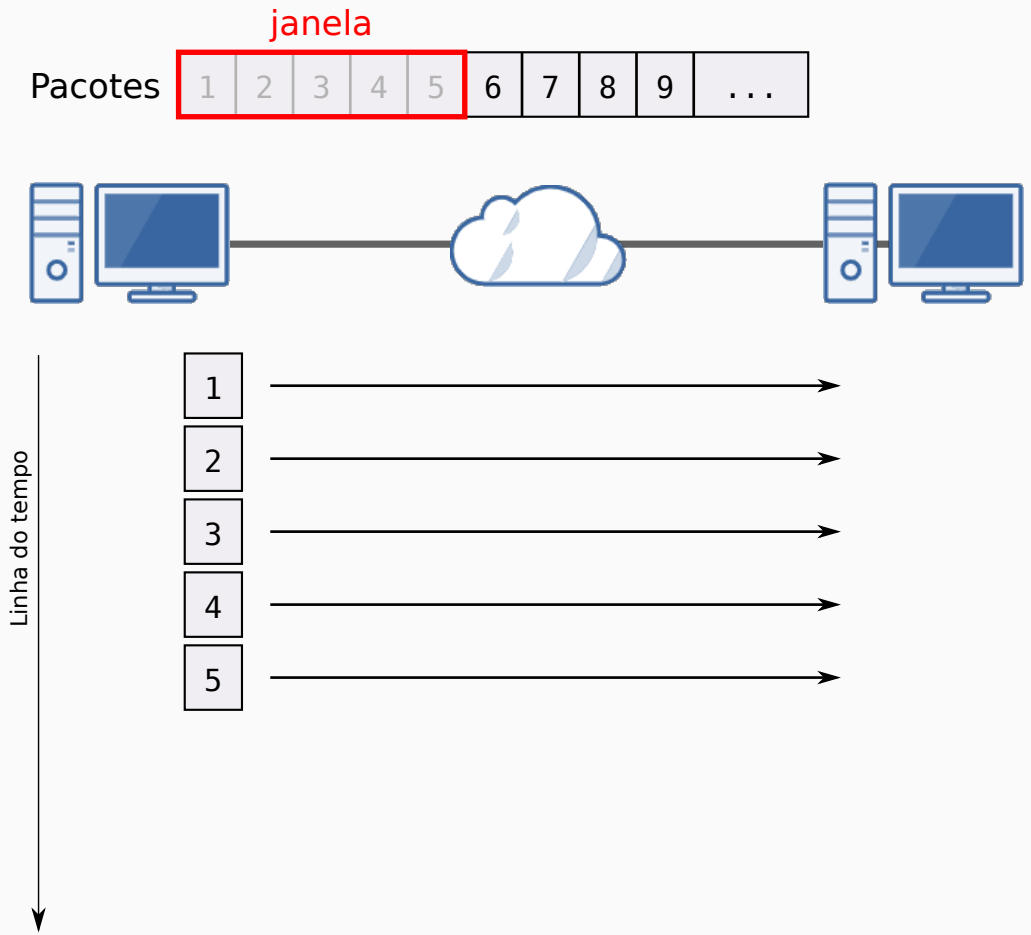
TCP - O Princípio da Janela

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



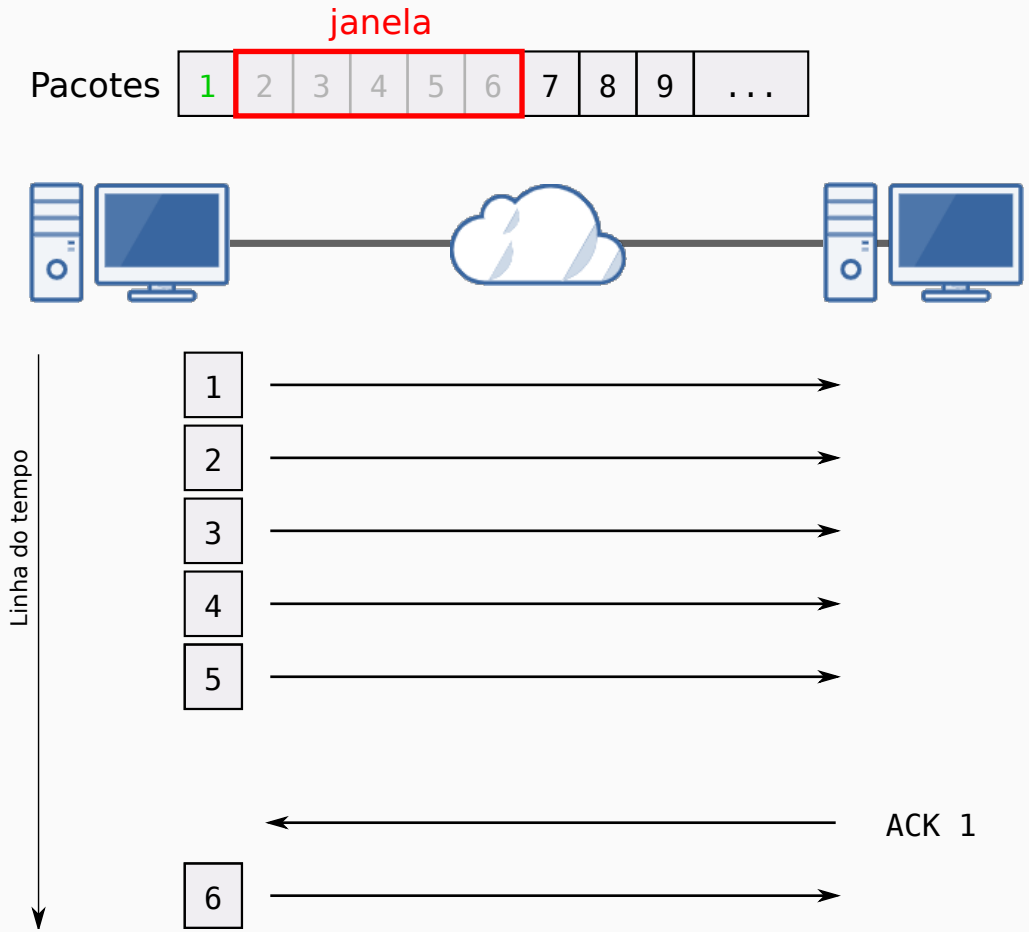
TCP - O Princípio da Janela

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



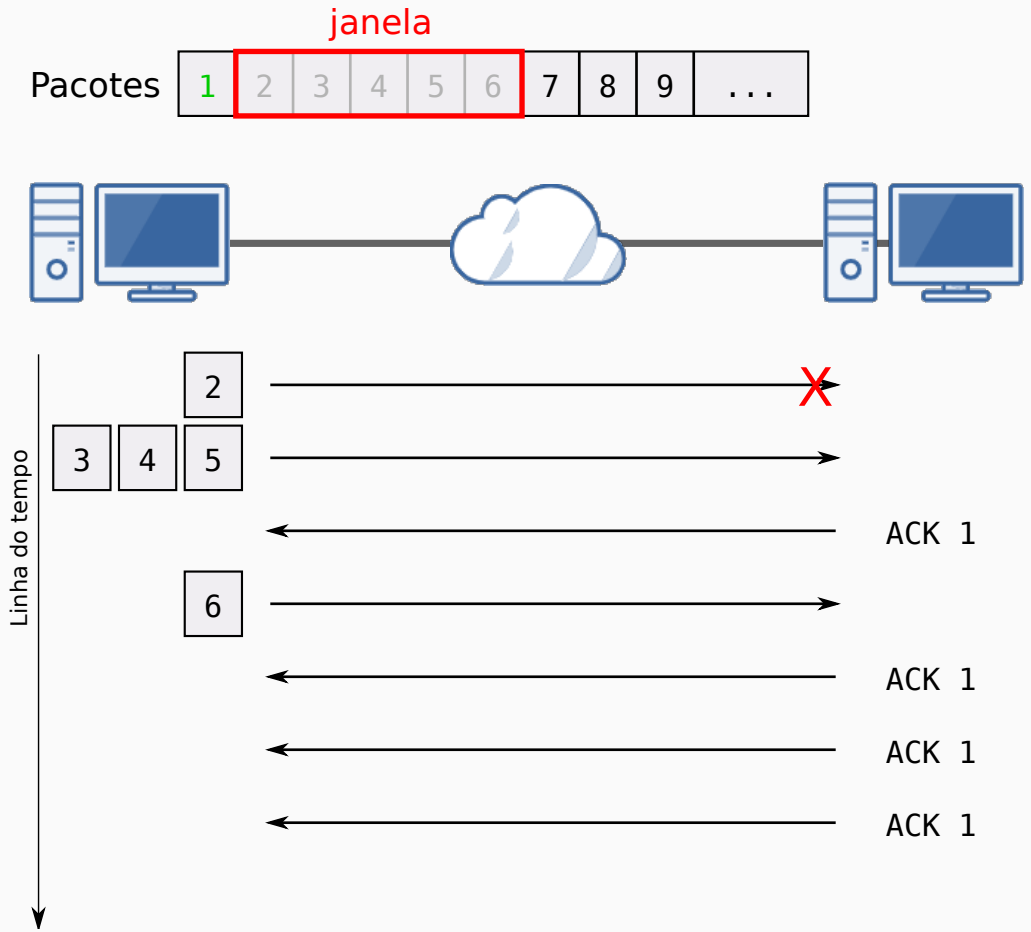
TCP - O Princípio da Janela

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



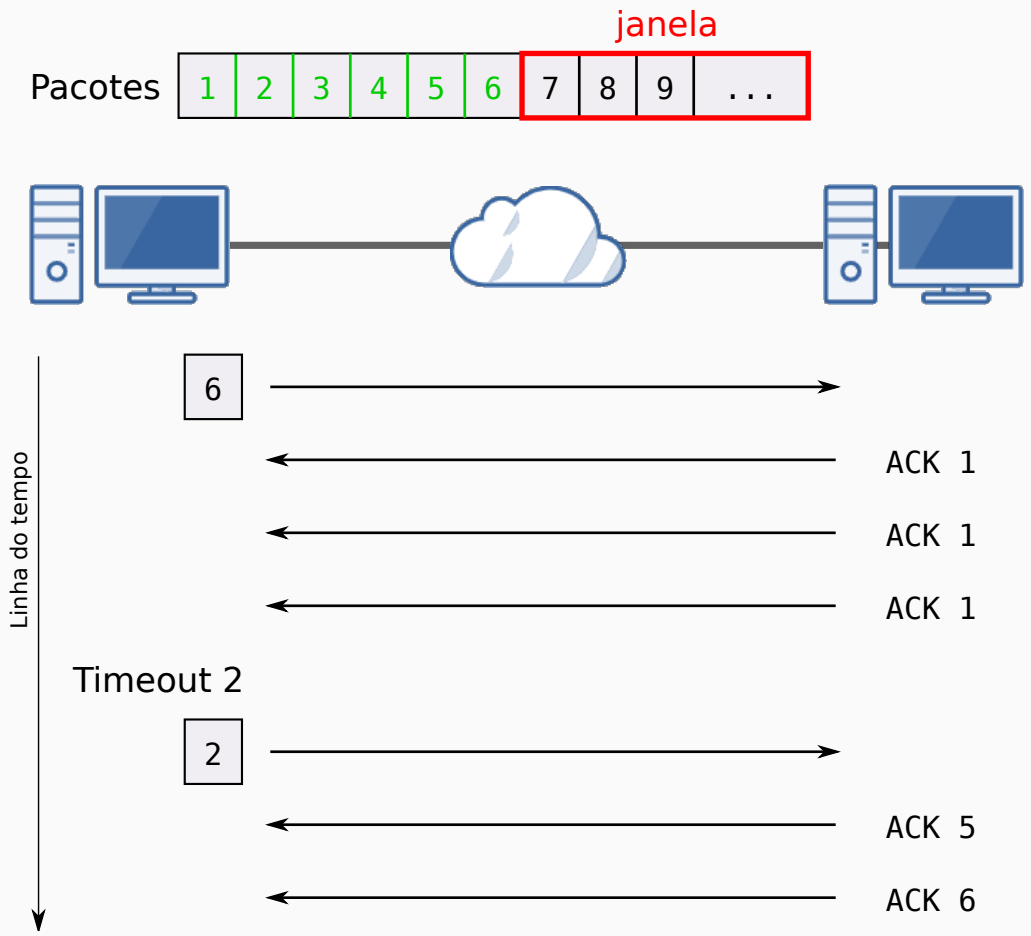
TCP - O Princípio da Janela (Pacote 2 Perdido)

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



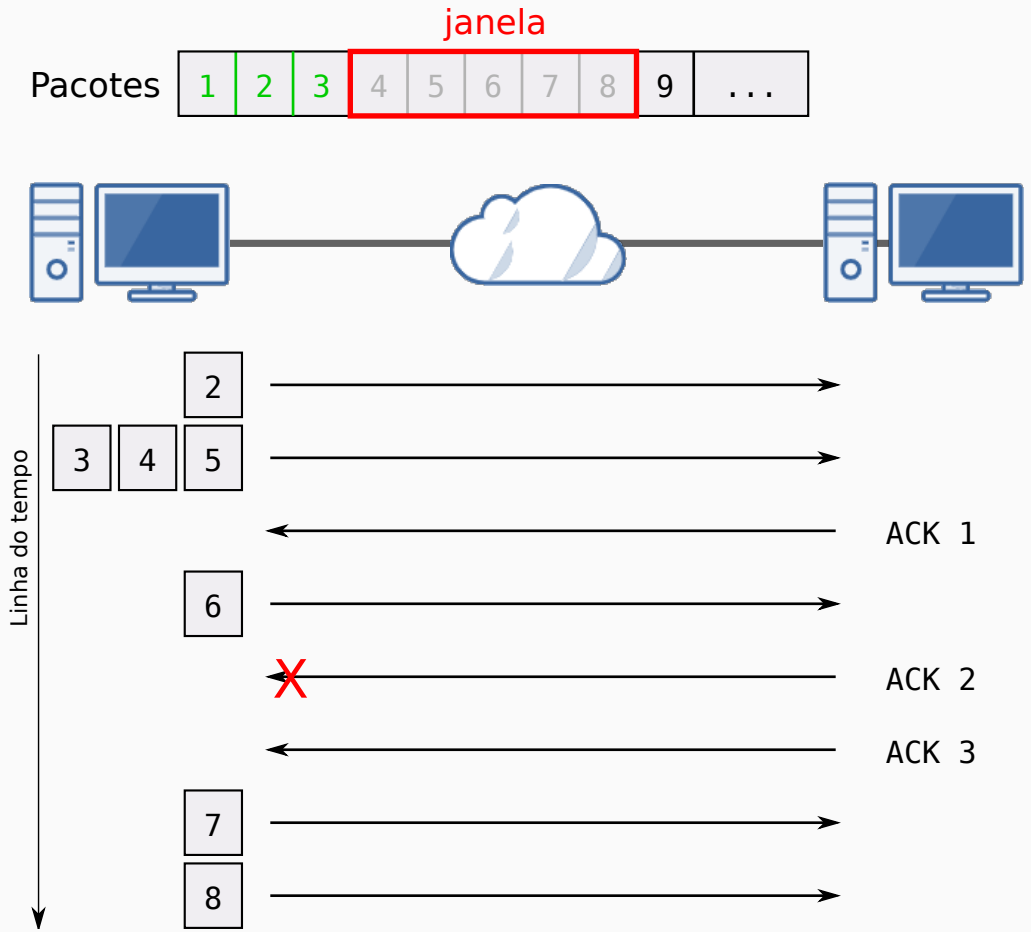
TCP - O Princípio da Janela (Pacote 2 Perdido)

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



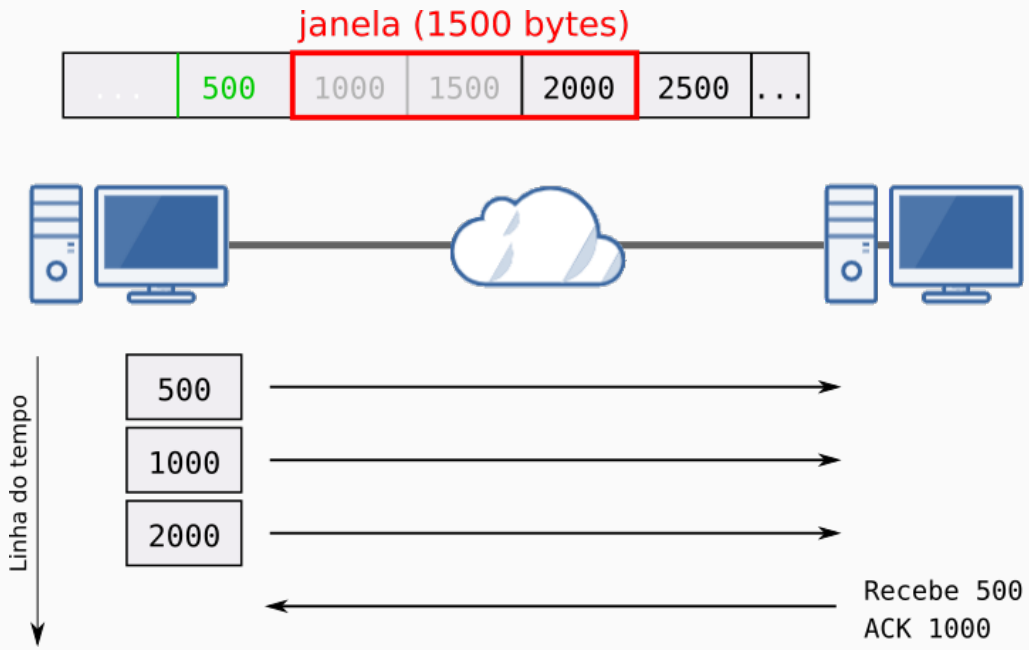
TCP - O Princípio da Janela (ACK 2 Perdido)

- Remetente inicia um timer para cada pacote;
- Destinatário envia ACK para o último pacote bem sucedido;
- Remetente desliza a janela de acordo com cada ACK;



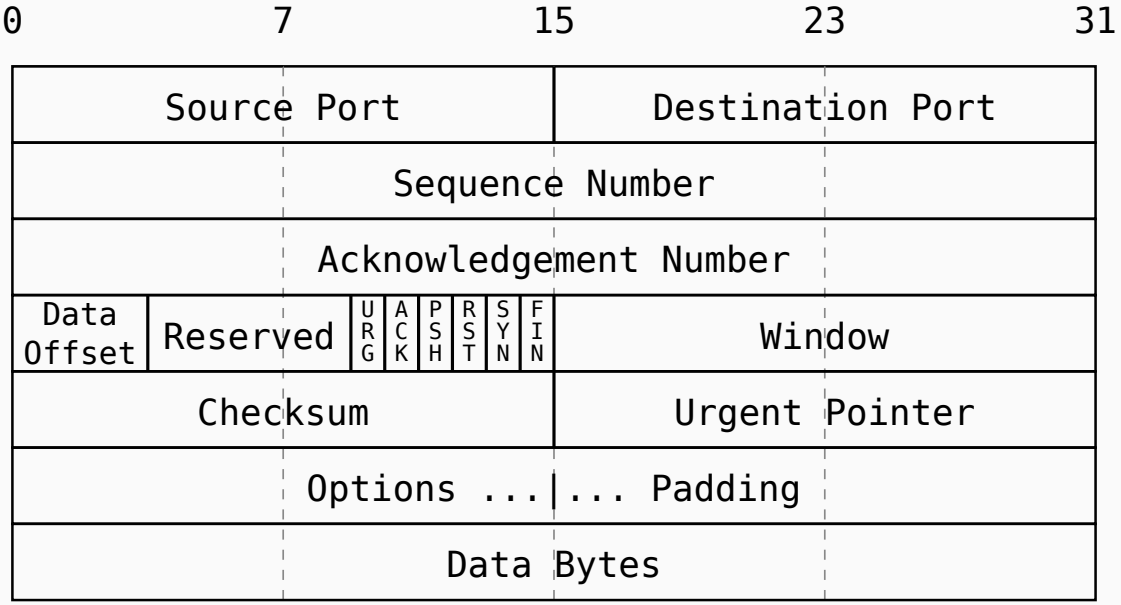
TCP - O Princípio da Janela adaptado ao TCP

- Número de sequência atribuída a cada byte;
- Cada segmento contém o número do primeiro byte;
- Tamanho da janela é definida inicialmente pelo destinatário e é variável (Adaptativo durante o processo);
- ACK envia o número de sequência do próximo bloco esperado.



TCP - Formato do Segmento

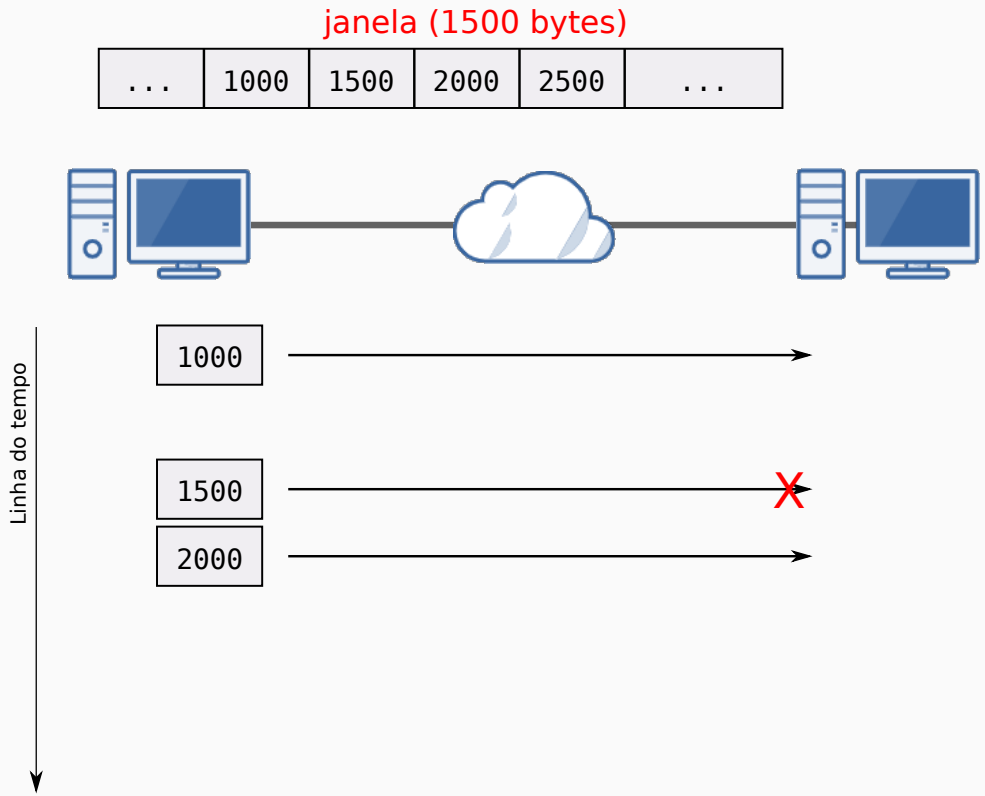
- Confira subseção 4.3.1 do livro texto para mais detalhes.



- Principais opções: Maximum segment size, Window scale, SACK-permitted, SACK e Timestamps.

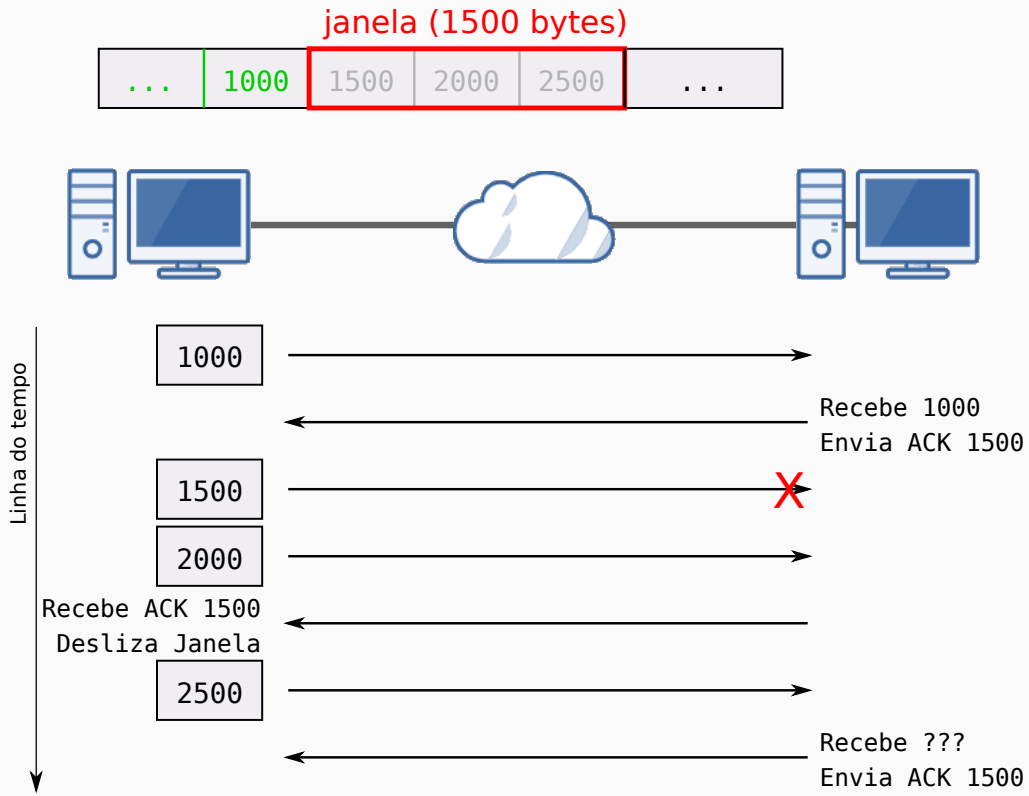
TCP - ACKs e retransmissões

- Timeouts devem ser calculados a partir de uma média ponderada dos tempos de ida e volta a fim de se adaptar a diferentes condições de rede.
- Problema: ACK não informa qual segmento recebido, apenas o último segmento com recebimento bem sucedido.



TCP - ACKs e retransmissões

- Timeouts devem ser calculados a partir de uma média ponderada dos tempos de ida e volta a fim de se adaptar a diferentes condições de rede.
- Problema: ACK não informa qual segmento recebido, apenas o último segmento com recebimento bem sucedido.

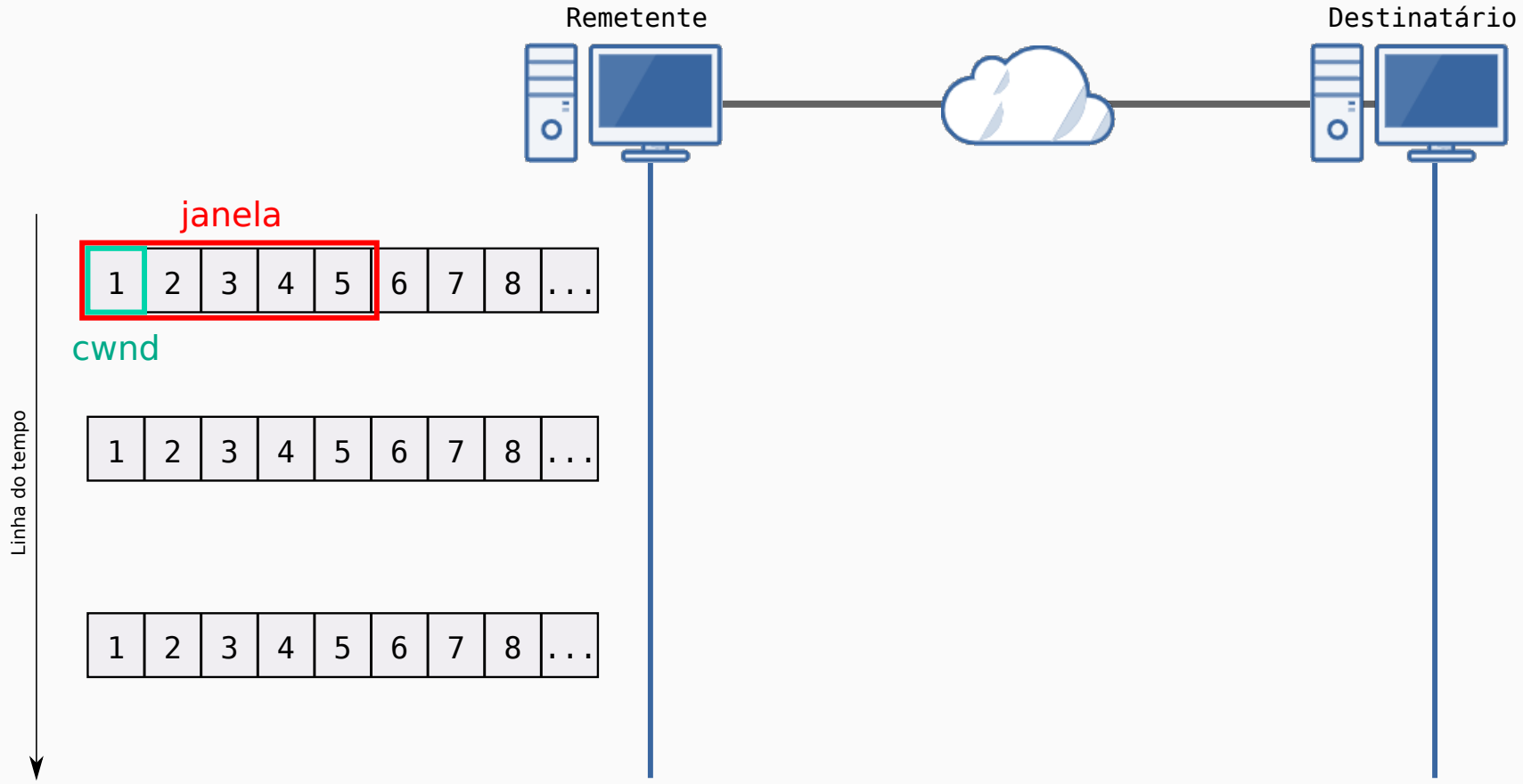


TCP - Controle de Congestionamento

- Controle de congestionamento é a principal diferença o protocolo TCP do UDP.
- Permite que o remetente adapte a taxa de envio para adequar a capacidade da rede e evitar congestionamento
- Implementações modernas de TCP utilizam quatro algoritmos que se complementam:
  - Slow start
  - Congestion avoidance
  - Fast retransmit
  - Fast recovery

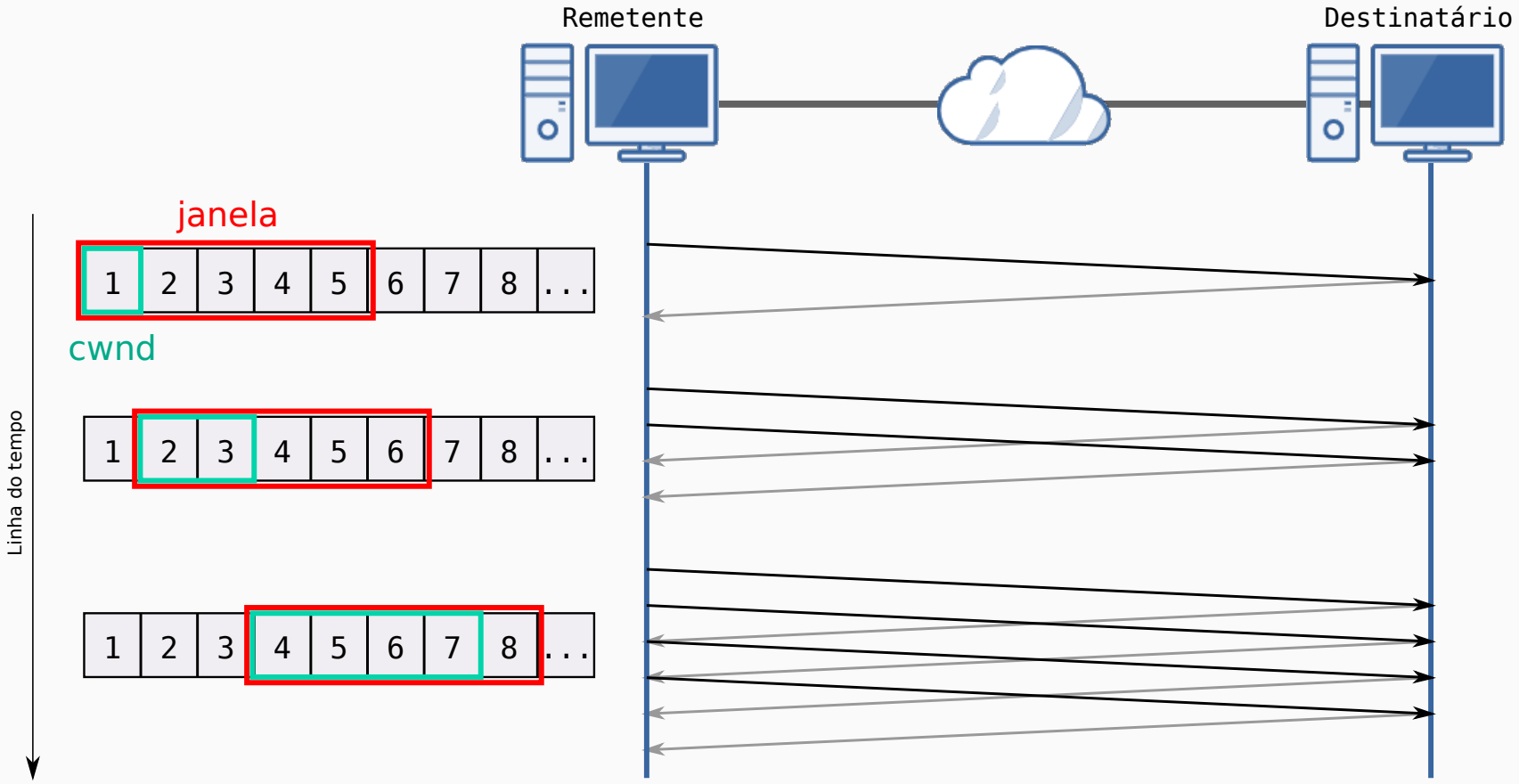
TCP - Controle de Congestionamento: Slow Start

- Determina a taxa de transmissão de pacotes a partir da janela de congestionamento.
- A janela de congestionamento (cwnd) é inicializada com o tamanho de um segmento, e é incrementada cada vez que um ACK é recebido
- É considerado o menor valor entre a janela e o cwnd.



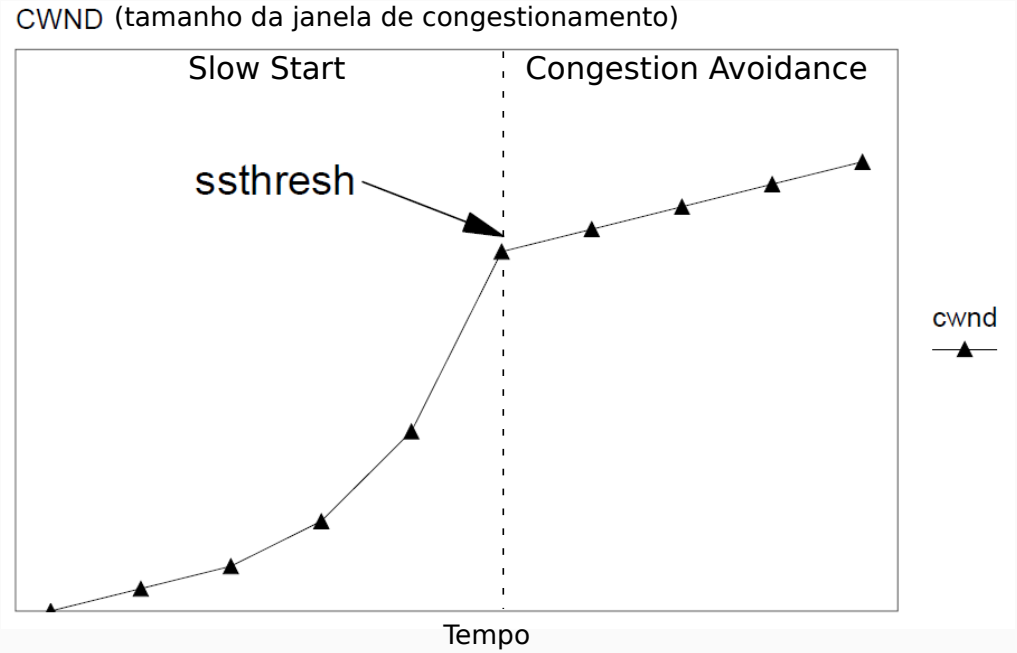
TCP - Controle de Congestionamento: Slow Start

- Determina a taxa de transmissão de pacotes a partir da janela de congestionamento.
- A janela de congestionamento (cwnd) é inicializada com o tamanho de um segmento, e é incrementada cada vez que um ACK é recebido
- É considerado o menor valor entre a janela e o cwnd.



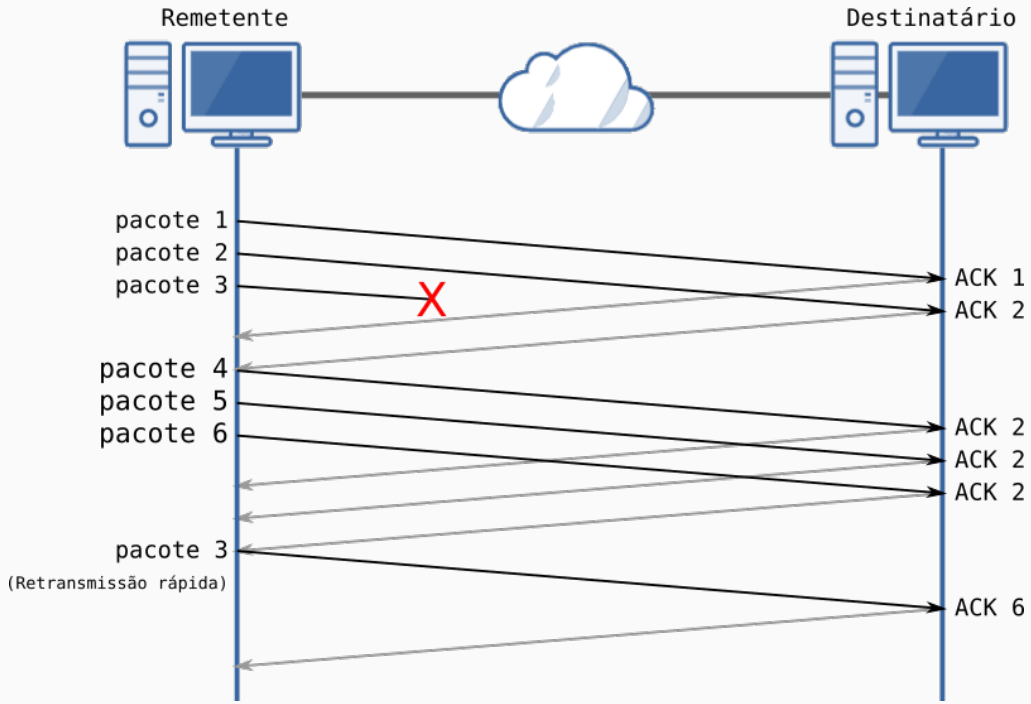
TCP - Controle de Congestionamento: Congestion Avoidance

- Existem dois sinais que indicam congestionamento: timeout e ACKs duplicados.
- Quando um congestionamento é detectado, a variável `ssthresh` recebe metade do tamanho da janela atual.
- Então, é iniciado o algoritmo slow start, que apresenta um crescimento exponencial.
- Quando o cwnd atinge o tamanho indicado por `ssthresh`, o crescimento da janela de congestionamento é incrementada para cada trajeto de ida e volta, ocasionando um crescimento linear.



TCP - Controle de Congestionamento: Fast Retransmit

- O algoritmo Fast Retransmit visa evitar esperar o timeout para retransmitir pacotes; ele se baseia na recepção de ACKs duplicados para detectar segmentos perdidos
- ACKs duplicados podem indicar tanto pacotes perdidos como entregas fora de ordem.
- Assim, para garantir que o segmento foi perdido, o Fast Retransmit faz a recepção somente após receber três ou mais ACKs duplicados.



TCP - Controle de Congestionamento: Fast Recovery

- Uma vez que o algoritmo de fast retransmit envia o segmento perdido, é acionado o congestion avoidance ao invés do slow start.
- Este processo, chamado de Fast Recovery, melhora a taxa de transferência em congestionamentos médios, especialmente para janelas grandes.
- O ACK duplicado indica que a comunicação continua ocorrendo, mesmo com o pacote perdido e, deste modo, seria desnecessário fazer o reinício devagar, diminuindo demasiadamente a taxa de transferência.

Referências

- Para o curso: livro da IBM “TCP/IP Tutorial and technical overview” (disponível em `https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf`.
- Para esta aula: Capítulo 4.

The End!