PMR3412 - Redes Industriais - 2021

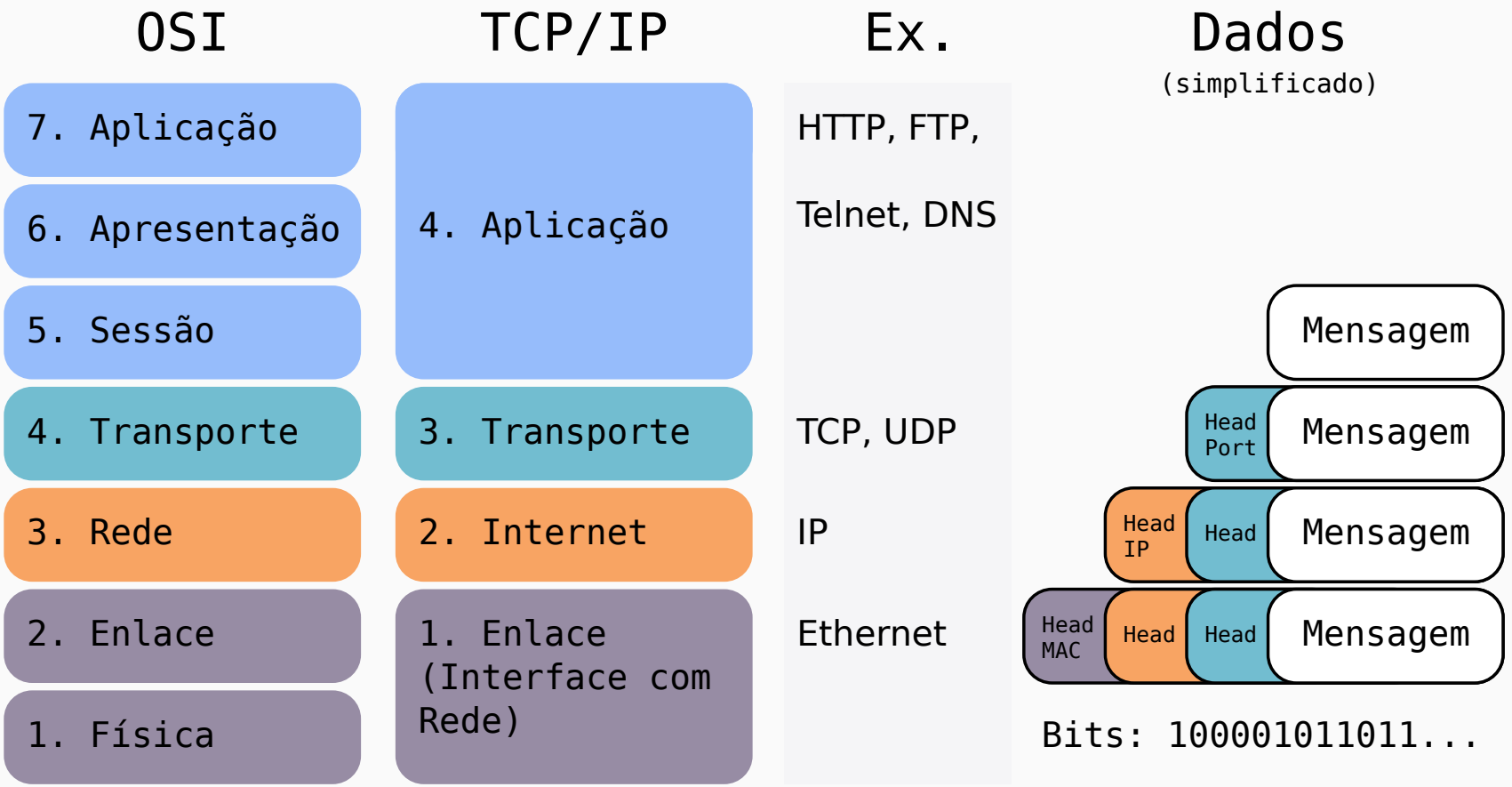
Aula 04 - TCP e UDP (e portas e sockets)

Prof. Dr. André Kubagawa Sato
Prof. Dr. Marcos de Sales Guerra Tsuzuki

9 de Setembro de 2021

PMR-EPUSP

Revisão - Relembrando o Modelo TCP/IP



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 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 TCP/IP se identifica através de uma ou mais portas
- A porta é representado por 16-bits e pode ser de dois tipos:

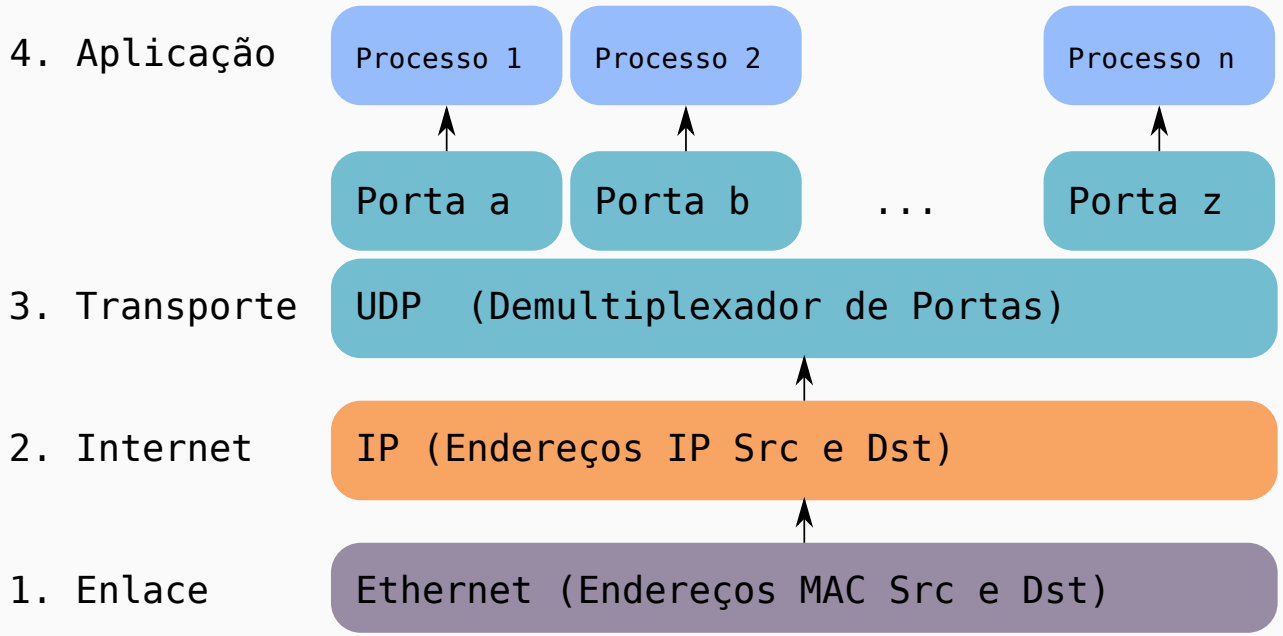
1. Portas Efêmeras (1024 até 65535): geralmente utilizada pelos clientes ao iniciar a comunicação.
2. Portas Conhecidas, ou Well-known Ports (1 até 1023): possibilita encontrar servidores sem configuração prévia. Exemplos:

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

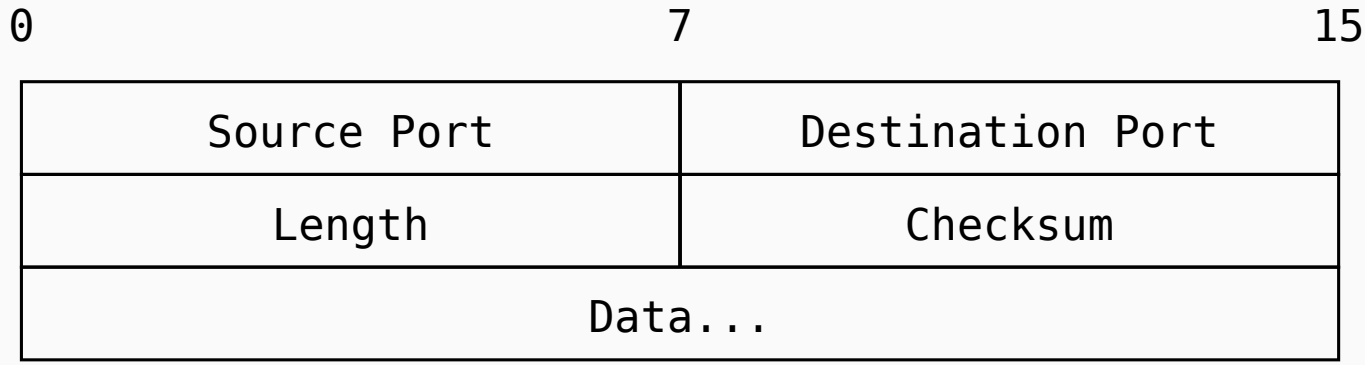
User Datagram Protocol (UDP)

UDP - Introdução

- O User Datagram Protocol (UDP) é não adiciona confiabilidade, controle de fluxo nem recuperação de erro ao IP.
- Assim, o UDP é basicamente uma interface de aplicação ao IP, já que apenas faz a ligação dos do payload do datagrama IP 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).

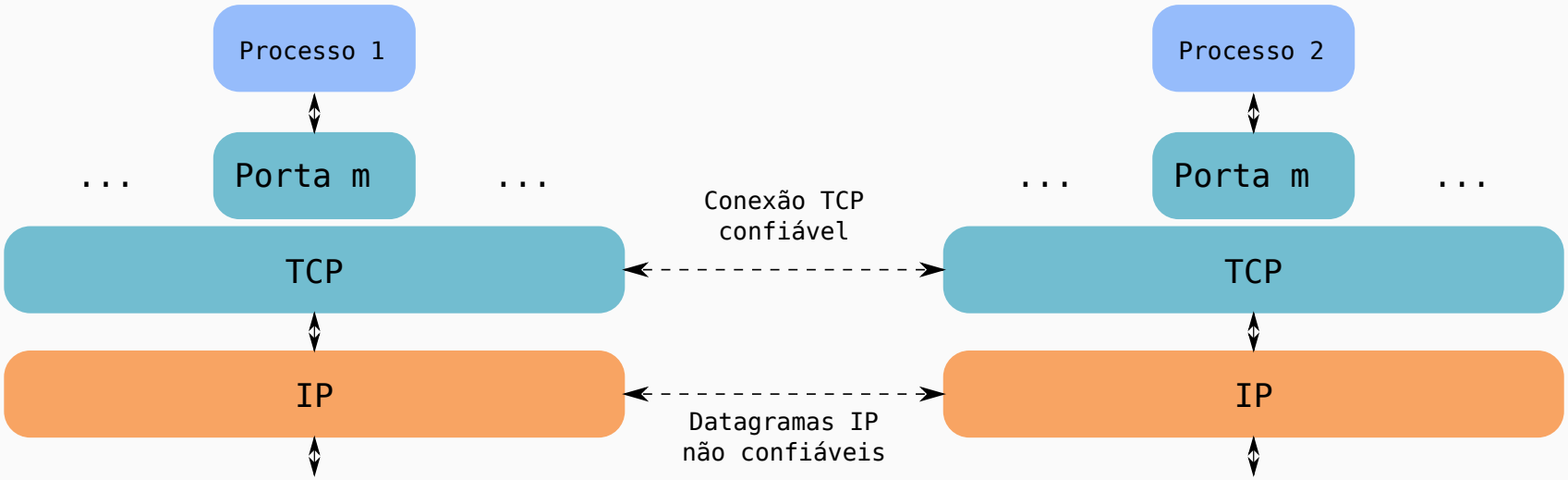
UDP - Aplicações UDP

- 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 origens;
  - 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 UDP, o protocolo Transmission Control Protocol (TCP) fornece um conjunto maior funcionalidades: confiabilidade, controle de fluxo e recuperação de erro.
- Além disso, o TCP é um protocolo orientado a conexão, isto é, requer que dois *host* iniciem uma conexão antes de iniciar uma comunicação.



TCP - Funcionalidades

Entrega de Fluxo de dados

- Para a aplicação, o TCP transfere um fluxo contíguo de bytes. Isso porque o TCP se responsabiliza em criar e gerenciar os 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

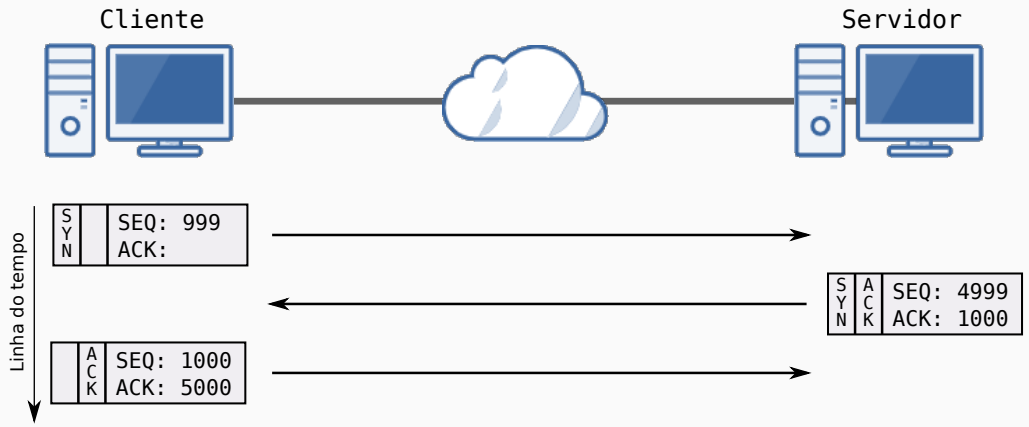
- Para cada fluxo de dados, é necessário manter as informações relativas a essa conexão, que incluem os sockets, números de sequência e tamanho de janela.

Full-Duplex

- Fluxo de dados concomitantes em ambas direções.

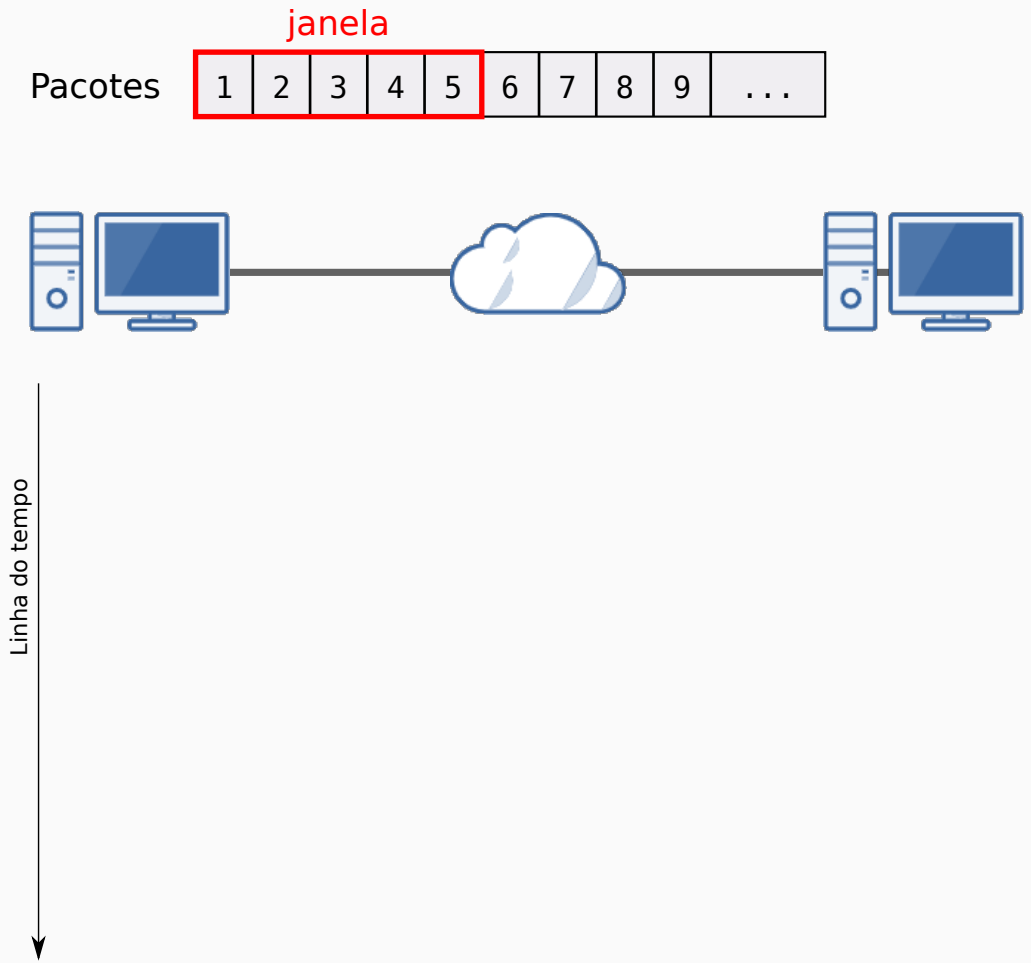
TCP - Estabelecendo uma conexão

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



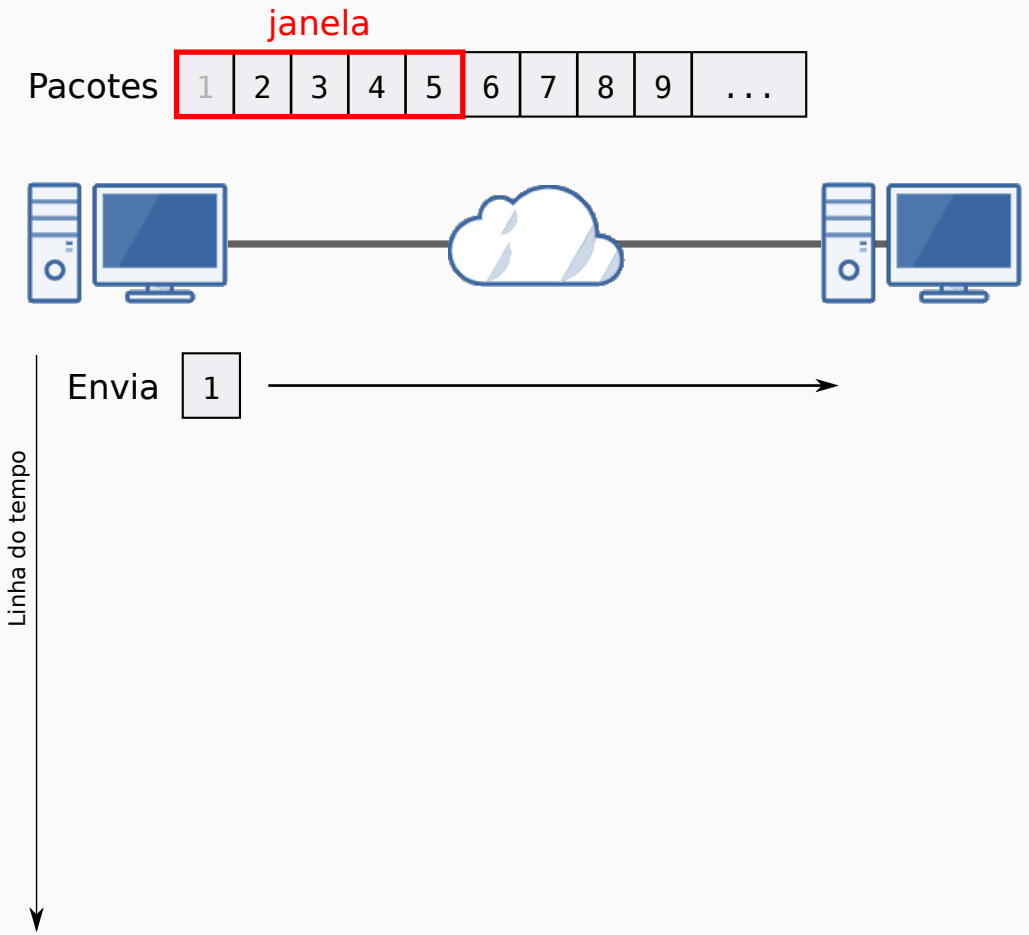
TCP - O Princípio da Janela

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



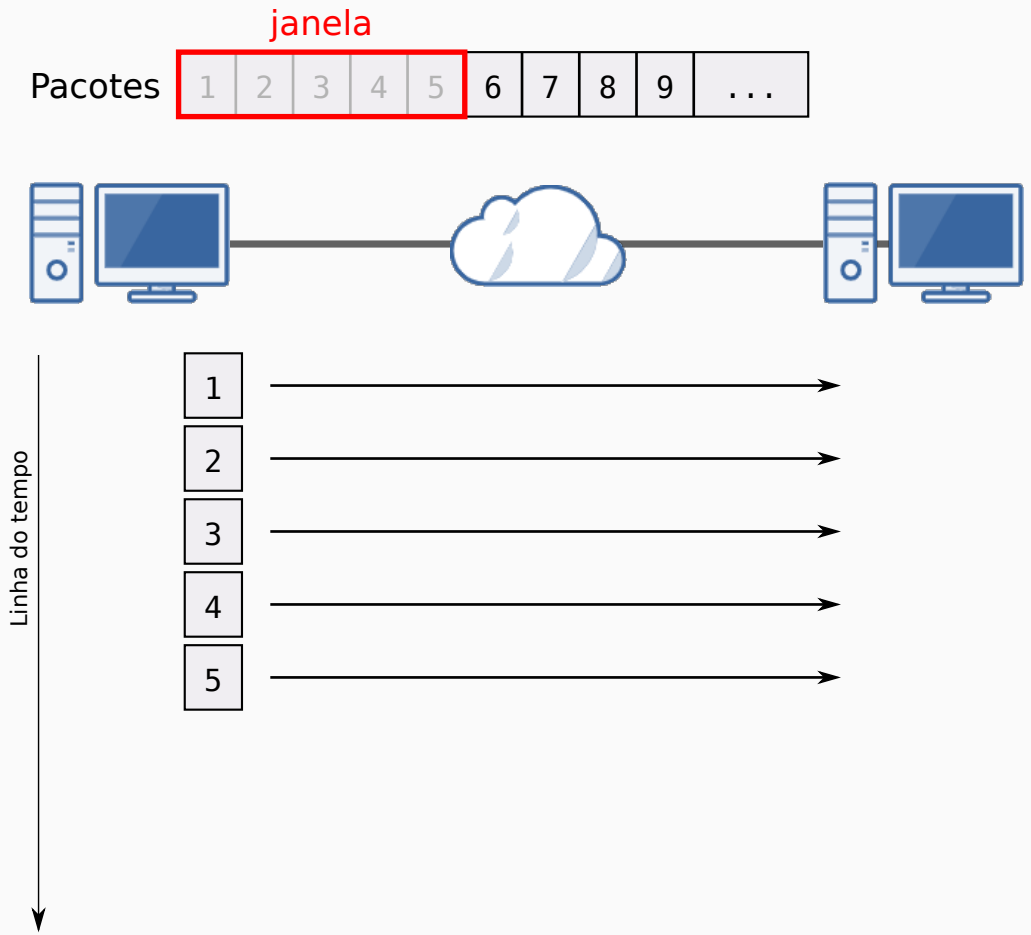
TCP - O Princípio da Janela

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



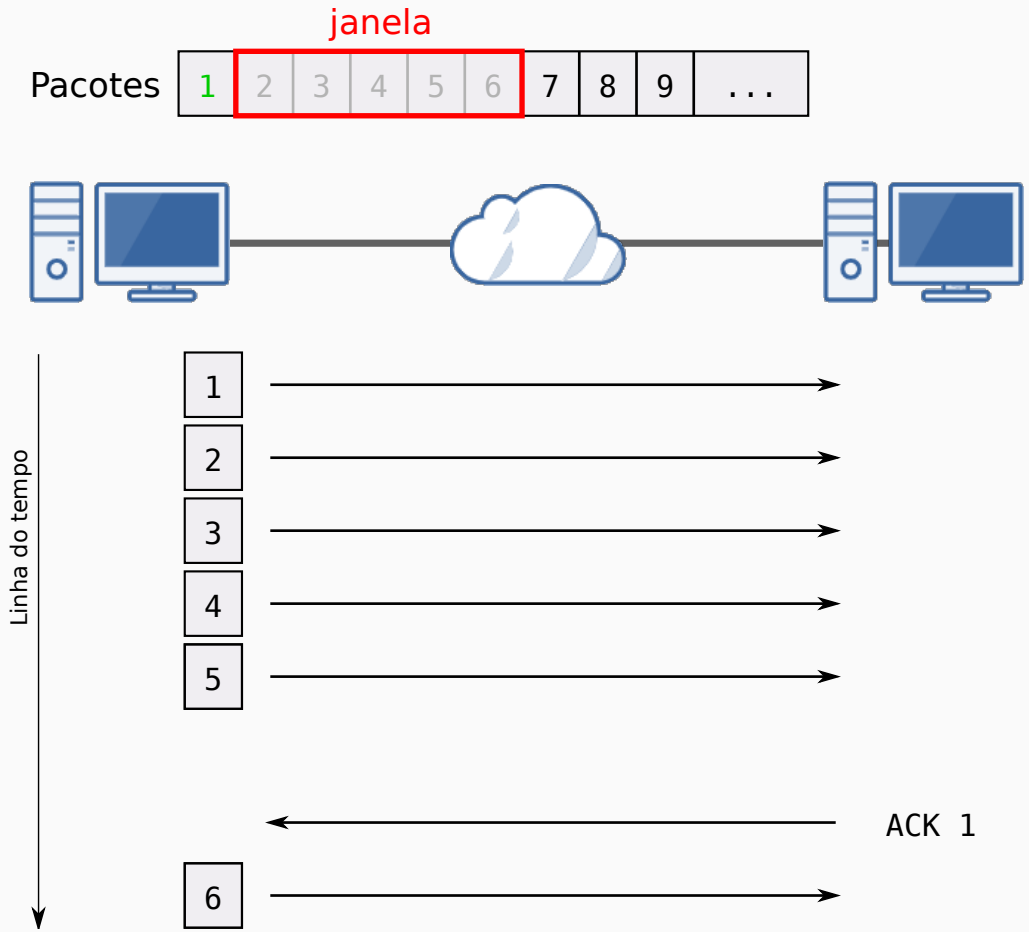
TCP - O Princípio da Janela

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



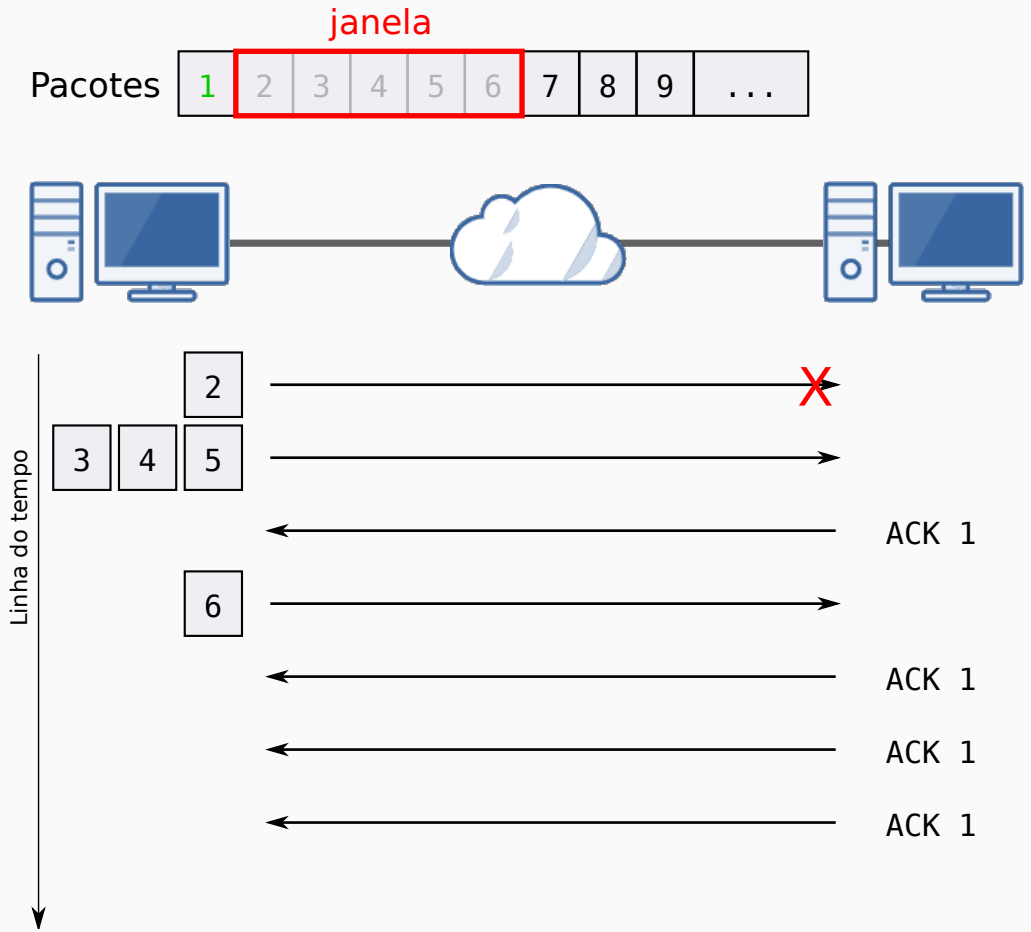
TCP - O Princípio da Janela

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



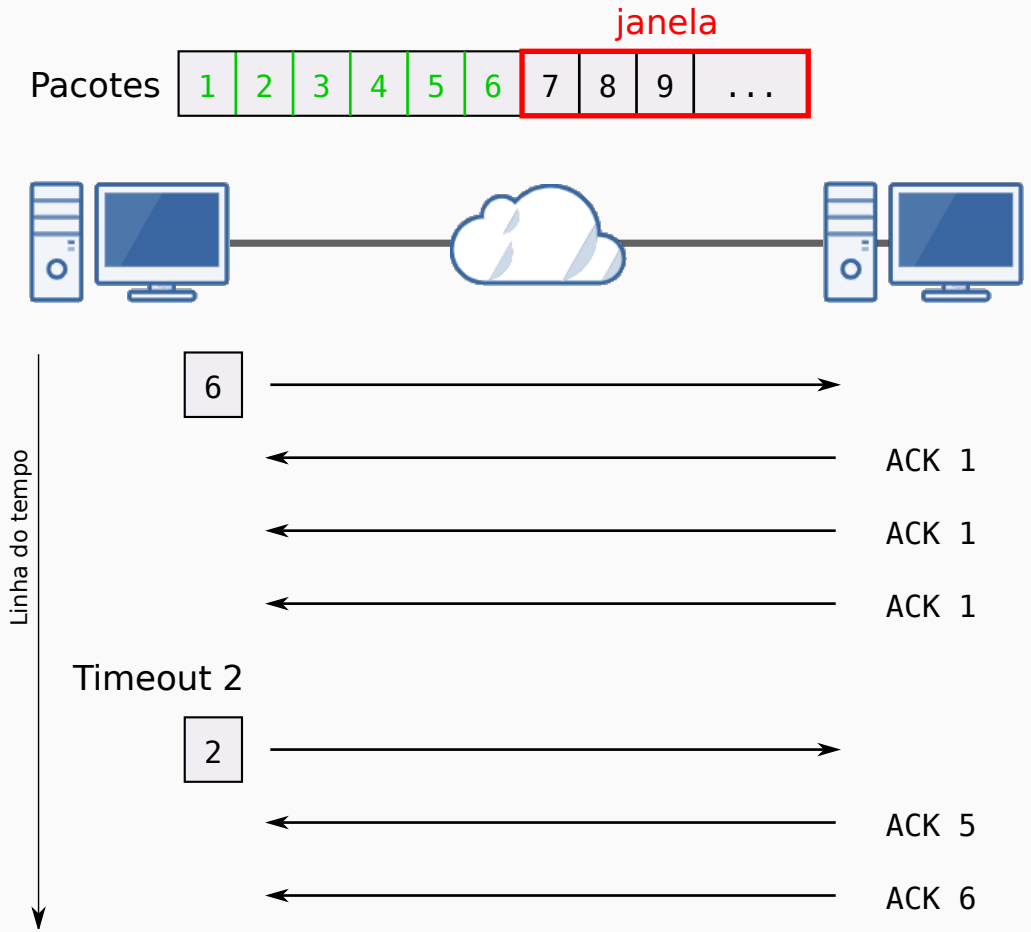
TCP - O Princípio da Janela (Pacote 2 Perdido)

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



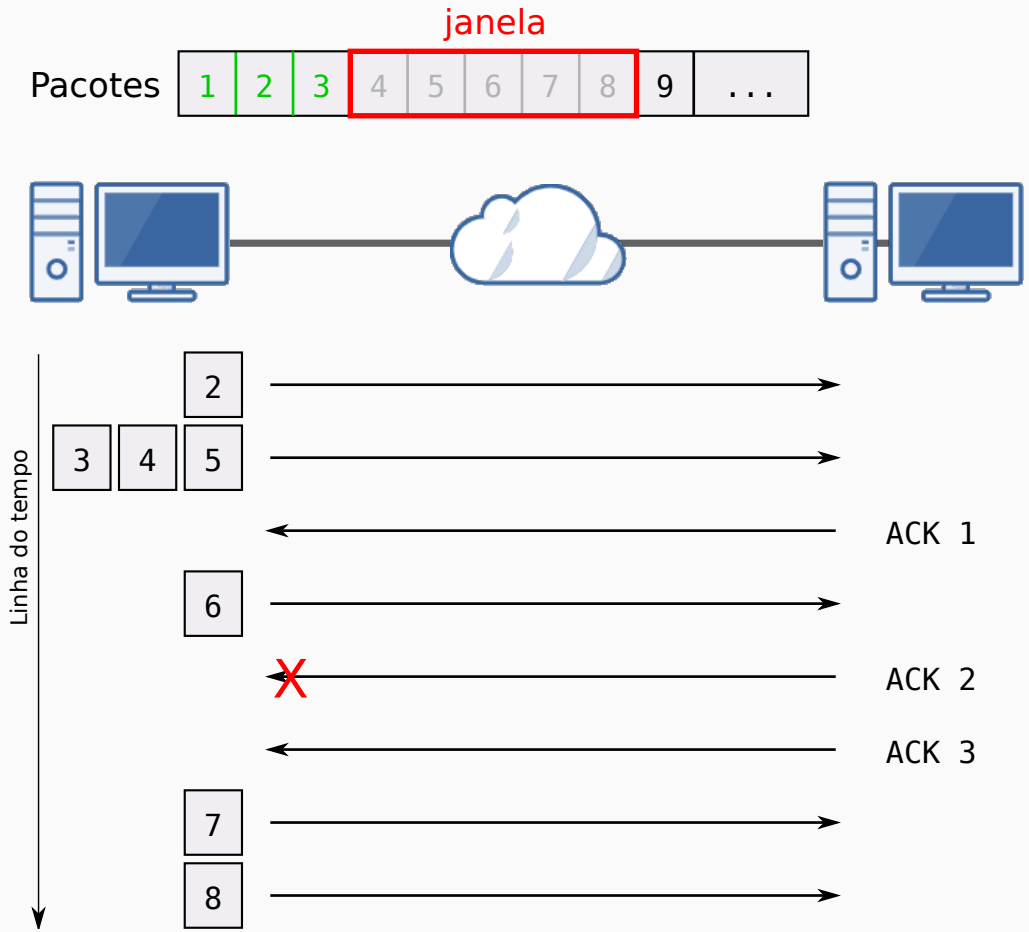
TCP - O Princípio da Janela (Pacote 2 Perdido)

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



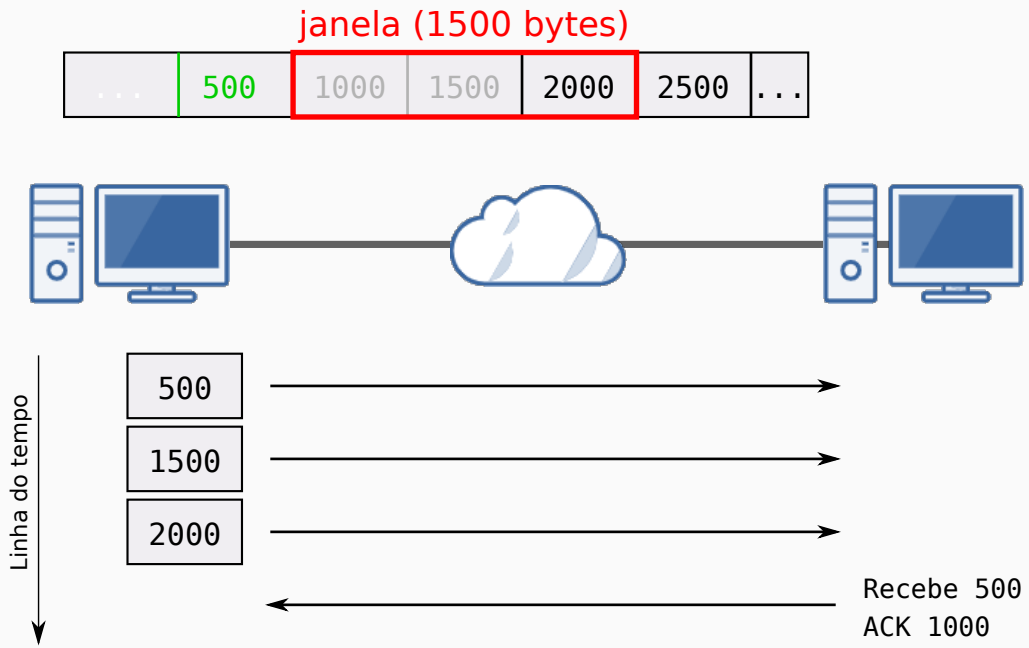
TCP - O Princípio da Janela (ACK 2 Perdido)

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



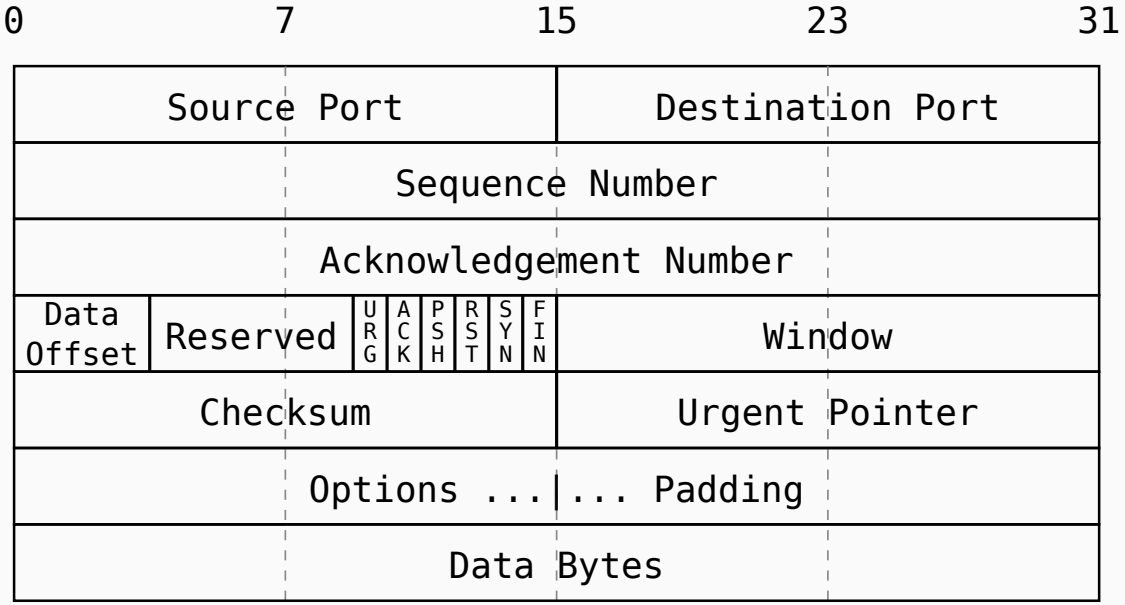
TCP - O Princípio da Janela adaptado ao TCP

- 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;



TCP - Formato do Segmento

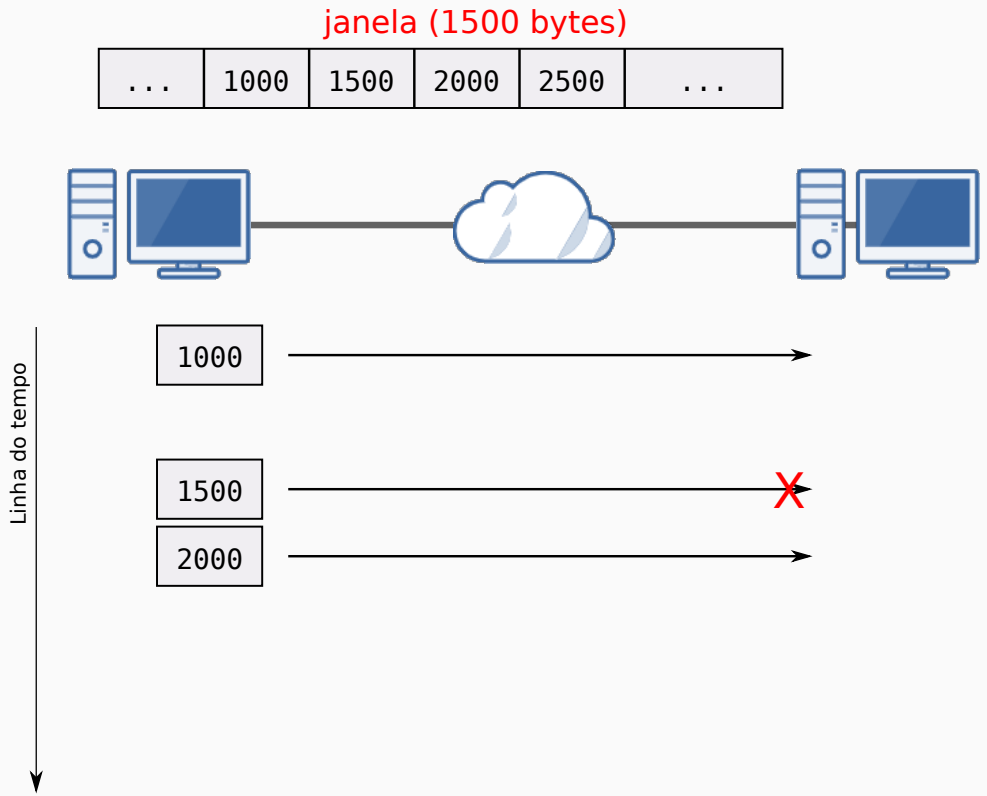
- Confira subseção 4.3.1 do livro texto para mais detalhes.



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

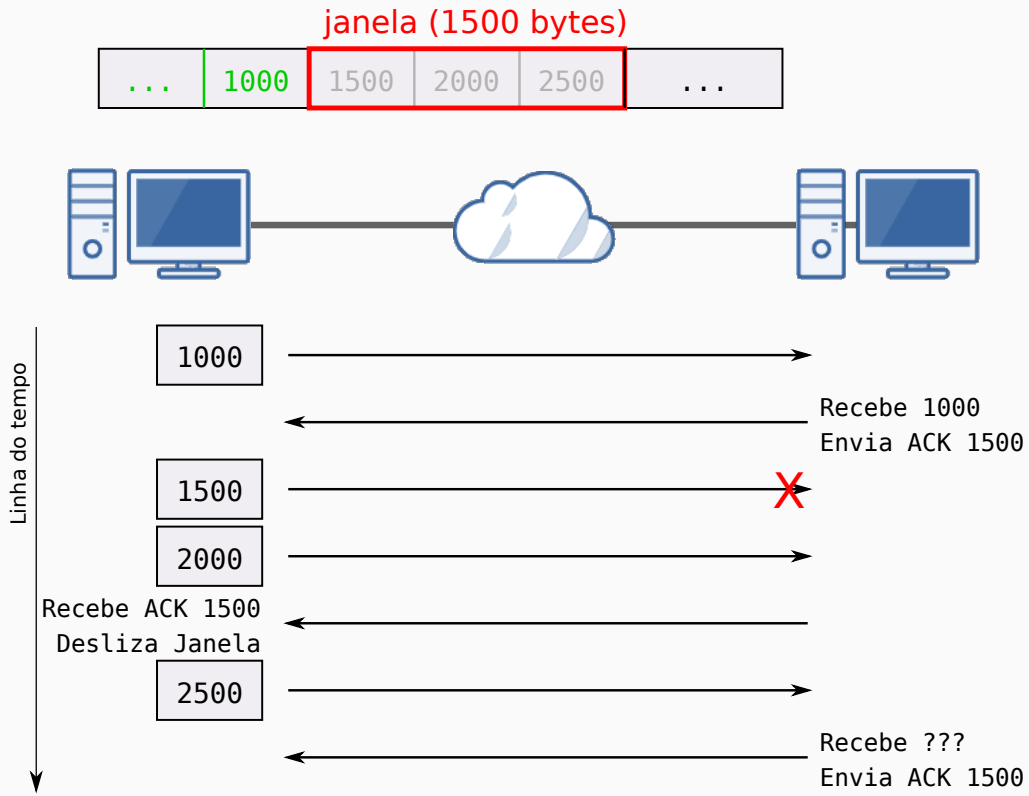
TCP - ACKs e retransmissões

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



TCP - ACKs e retransmissões

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

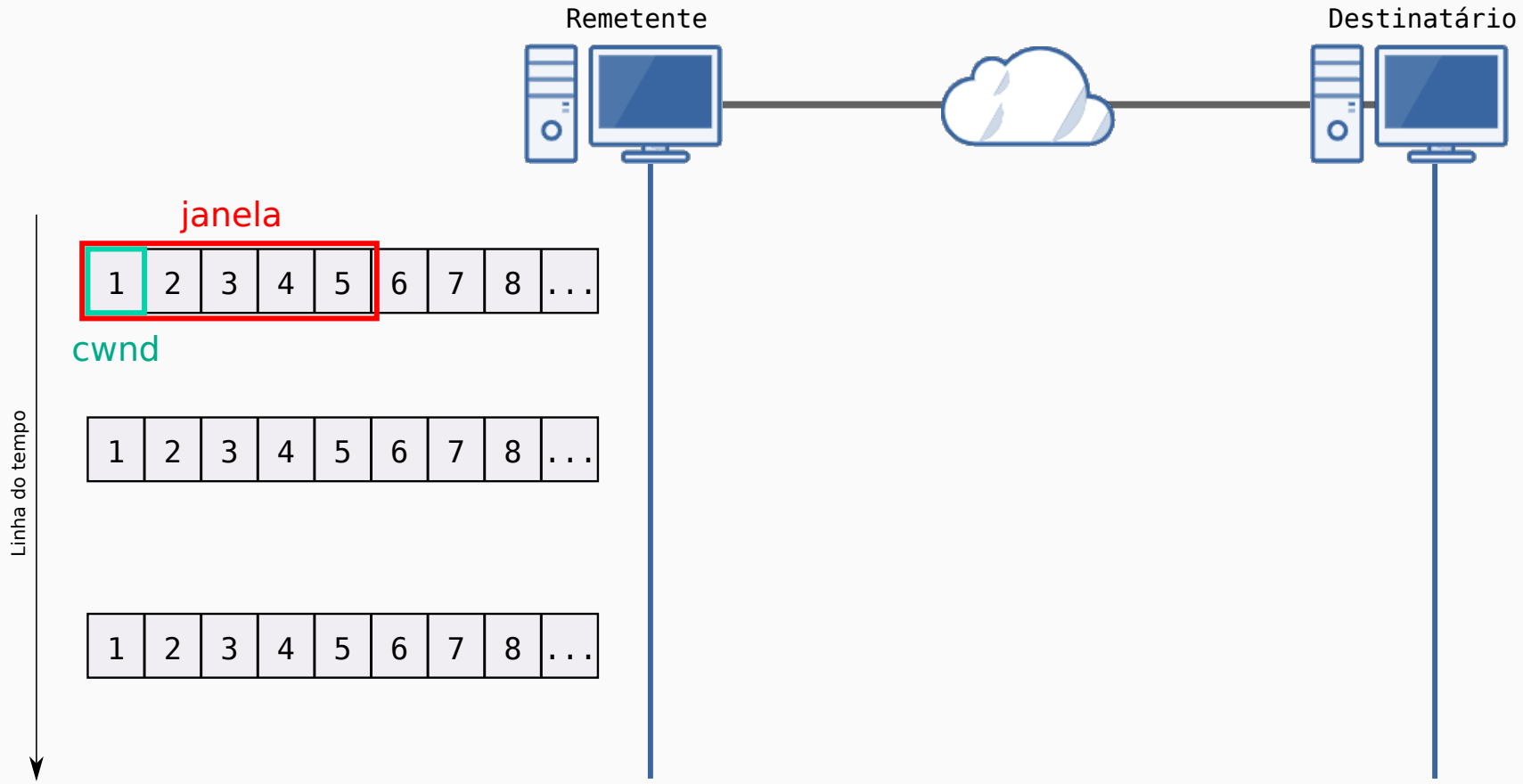


TCP - Controle de Congestionamento

- 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

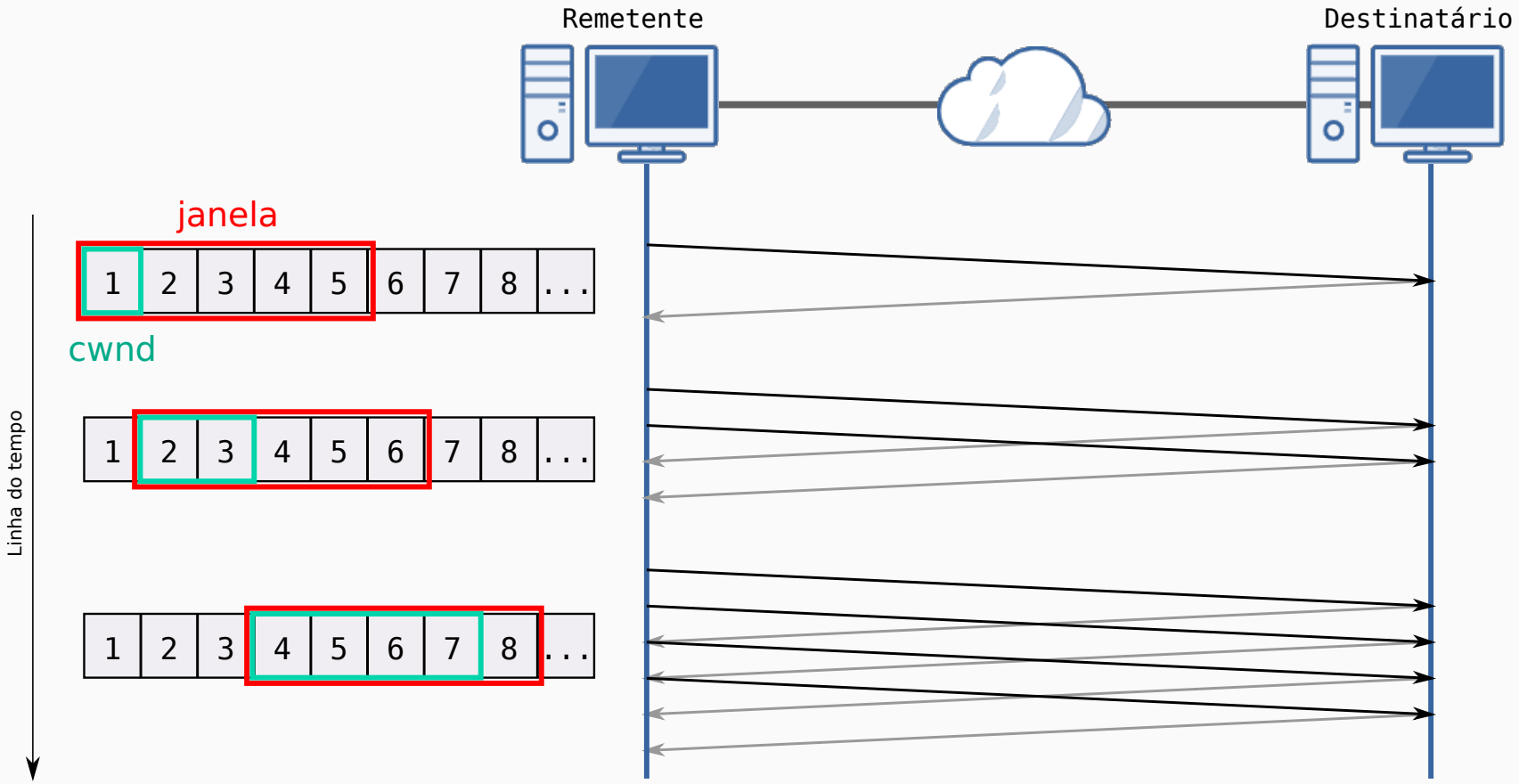
TCP - Controle de Congestionamento: Slow Start

- 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.



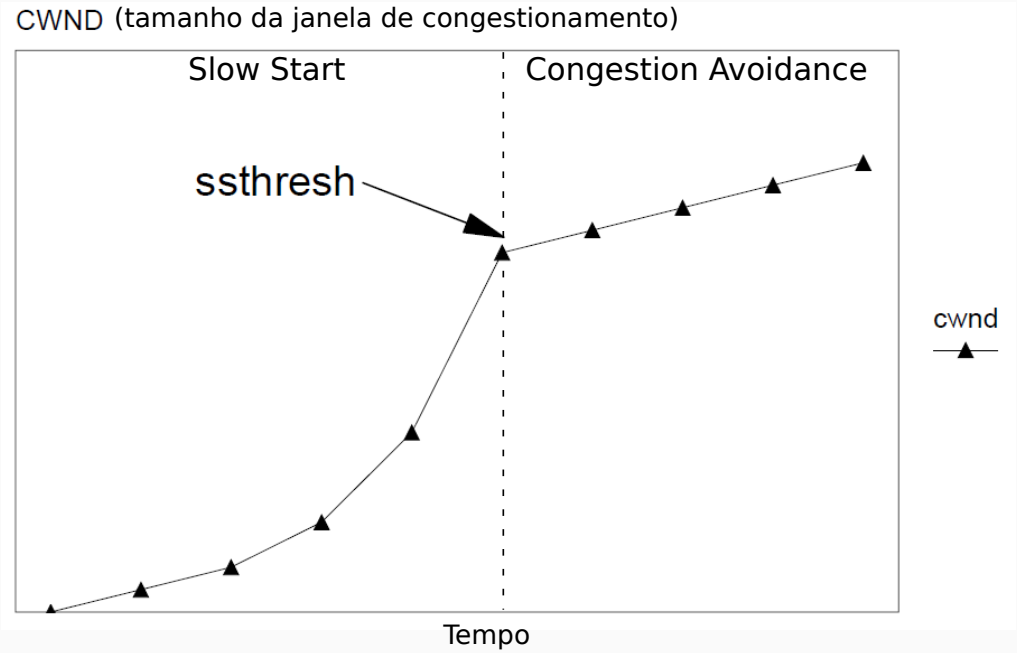
TCP - Controle de Congestionamento: Slow Start

- 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.



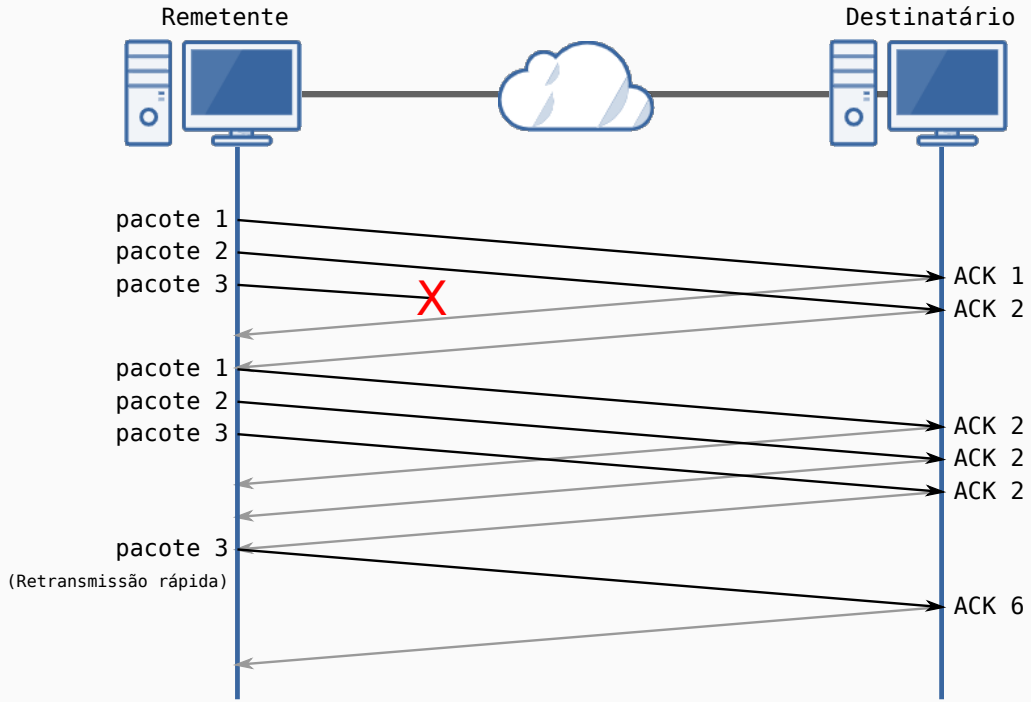
TCP - Controle de Congestionamento: Congestion Avoidance

- 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!