UNICARIOCA
Formação de verdade.

Arquitetura de Redes de Computadores

Unidade III – Camada de Transporte

Apresentado por Prof. Fred Sauer

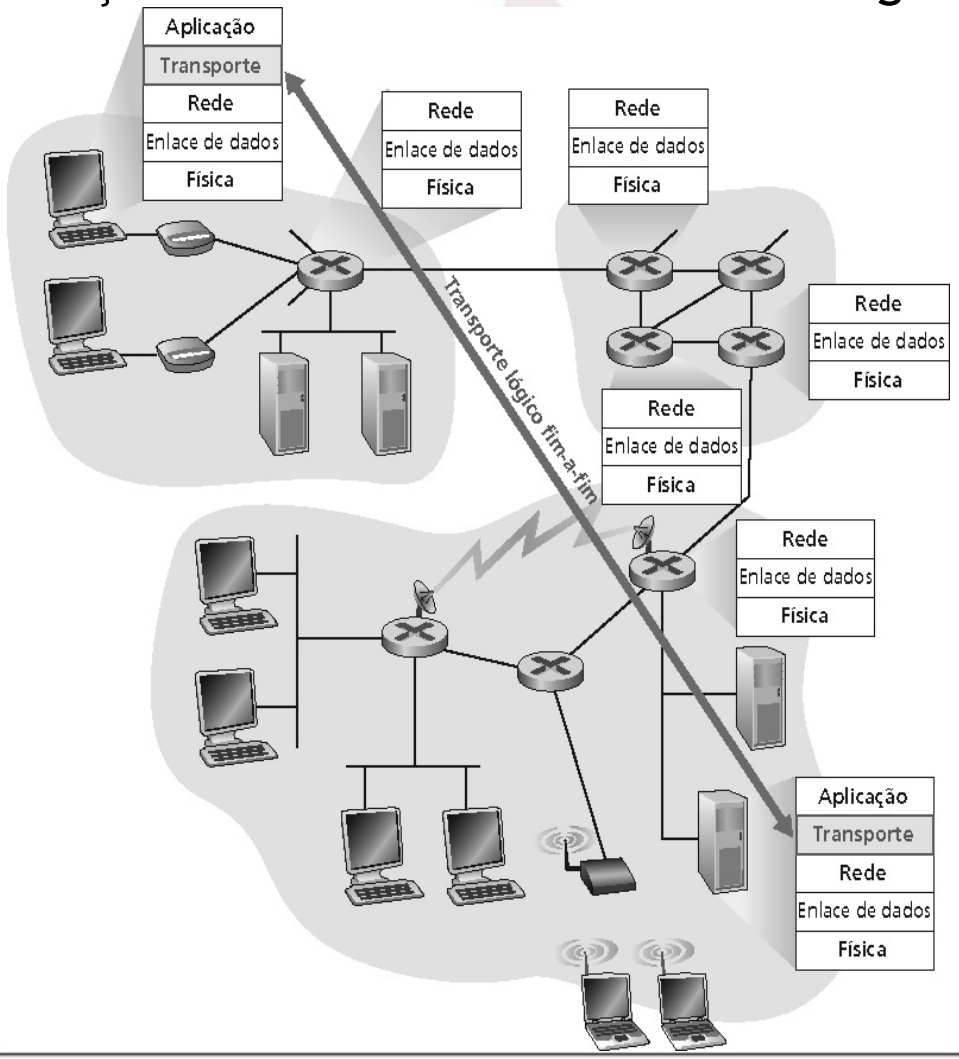
Baseado em Material didático de Prof Sergio Cardoso

Objetivos do Capítulo

- Entender os princípios dos serviços da camada de transporte:
  - Multiplexação/demultiplexação
  - Transferência de dados confiável
  - Controle de fluxo
  - Controle de congestionamento
- Aprender sobre os protocolos de transporte na Internet:
  - UDP: transporte não orientado à conexão
  - TCP: transporte orientado à conexão (confiável)

Camada de Transporte

- Oferecer comunicação confiável fim-a-fim entre origem e destino.



Camada de Transporte

- Funções da camada com o TCP
  - Controle de fluxo fim-a-fim
  - Controle de congestionamento
  - Verificação de erros no segmento via checksum
  - Reordenamento de segmentos recebidos
  - Detecção e recuperação de perdas de segmentos
  - Multiplexação/demultiplexação de várias conexões de transporte em uma conexão de rede

Camada de Transporte

- Provê *comunicação lógica* entre processos de aplicação executados em hospedeiros diferentes
- Protocolos de transporte executam em sistemas terminais (não há camada de transporte em roteadores)

- Tipos de serviços oferecidos na arquitetura TCP/IP:
  - Com conexão: protocolo TCP
  - Sem conexão: protocolo UDP

- Unidade de dados
  - TCP: **segmento**.
  - UDP: **datagrama**.

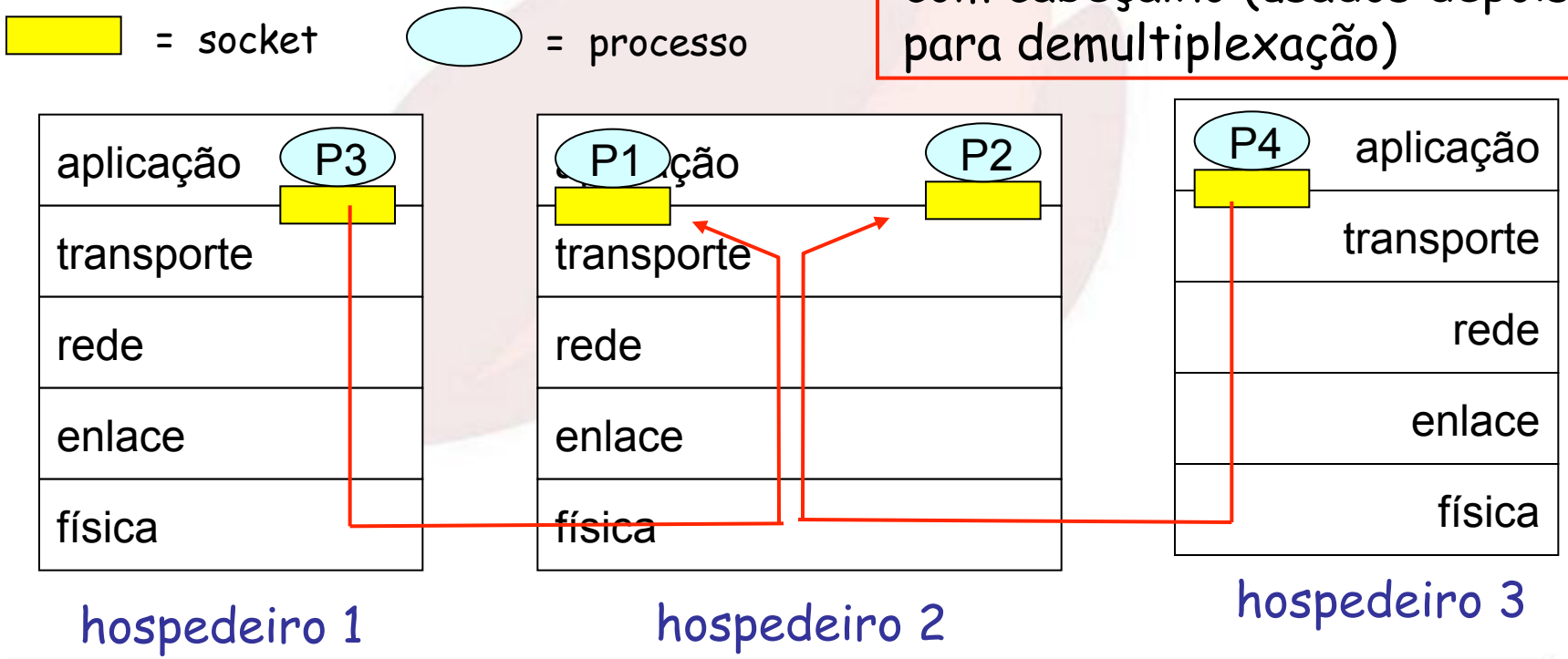
Multiplexação/Demultiplexação

demultiplexação no destinatário:

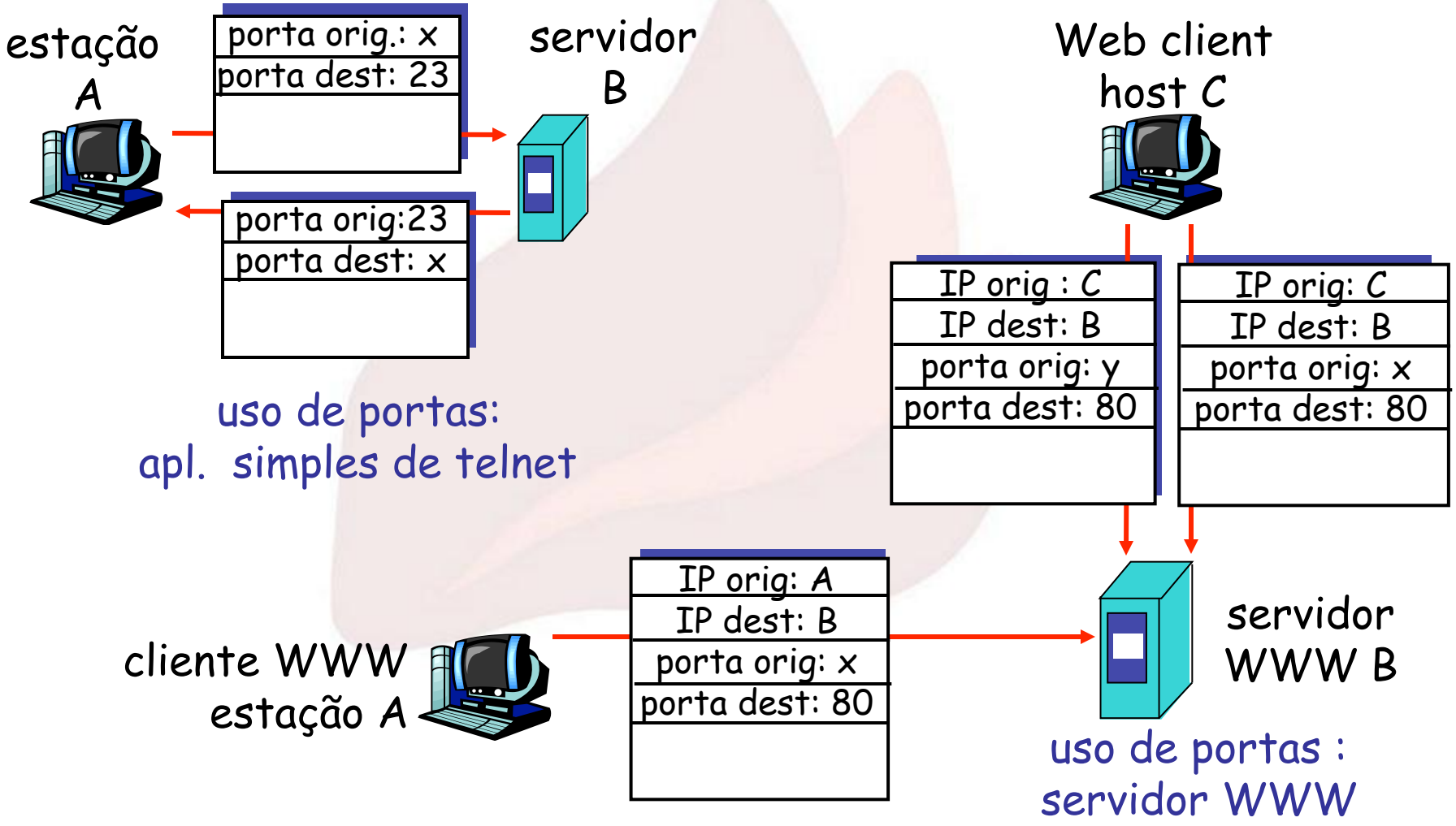
entregando segmentos recebidos ao socket correto

multiplexação no remetente:

colhendo dados de múltiplos sockets, envelopando dados com cabeçalho (usados depois para demultiplexação)



Exemplos Multi/Demulti



Atribuição de Portas

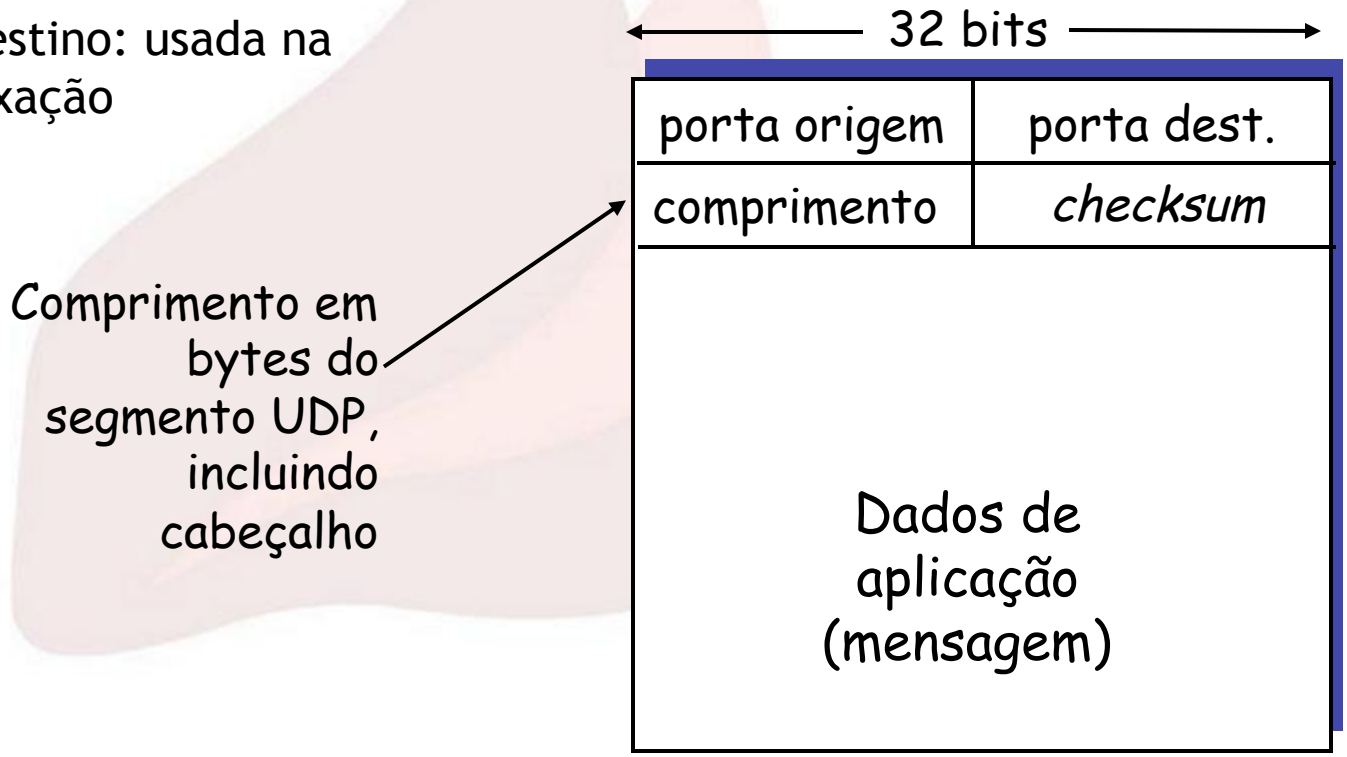
- Atribuídos pelo IANA - (Internet Assigned Numbers Authority) http://www.iana.org/assignments/port-numbers
- Distribuição das portas
  - Portas “bem conhecidas” (well-known ports)
    - São atribuídas pelo IANA e na maioria dos sistemas somente podem ser utilizadas por usuários privilegiados (root) ou por programas executados por usuários privilegiados
    - Variam de 0 a 1023
  - Portas registradas
    - Podem ser executadas por processos ou programas de usuários comuns
    - Variam de 1024 a 49151
  - Portas dinâmicas ou privadas
    - Não podem ser registradas
    - Variam de 49152 a 65535
- Em teoria, acima de 1023, estaria livre para uso dos usuários, porém pode haver conflitos.
  - Para não haver conflitos, a aplicação deve ser enviada para registro no IANA

Protocolo UDP

- Protocolo UDP (*User Datagram Protocol*)
  - Serviços
    - Transferência de dados não-confiável
    - Multiplexação/Demultiplexação
    - Detecção de erros por checksum
    - Não orientado a conexão
    - Sem controle de congestionamento
    - Sem controle de fluxo
    - Não suporta segmentação de mensagens (nem remontagem)

Formato da mensagem UDP

- A mensagem UDP contém
  - porta origem: usada pelo destino em uma resposta
  - porta de destino: usada na demultiplexação



Protocolo UDP

- Normalmente usado para *streaming* de aplicações de multimídia
  - Tráfego tolerante a perdas, porém sensível à vazão, delay e jitter
- Outros usos do UDP
  - DNS (para segmentos de até 511 Bytes)
  - SNMP (Gerência de rede)
- A aplicação assume as responsabilidades pela garantia da confiabilidade.
- Por que existe UDP?
  - Não há estabelecimento de conexão que possa provocar atraso
  - Simples: não procedimento de conexão nem no transmissor, nem no receptor
  - Cabeçalho de segmento reduzido (8 bytes x 20 bytes do TCP)
  - Não há controle de congestionamento: UDP pode enviar segmentos tão rápido quanto desejado (e possível)

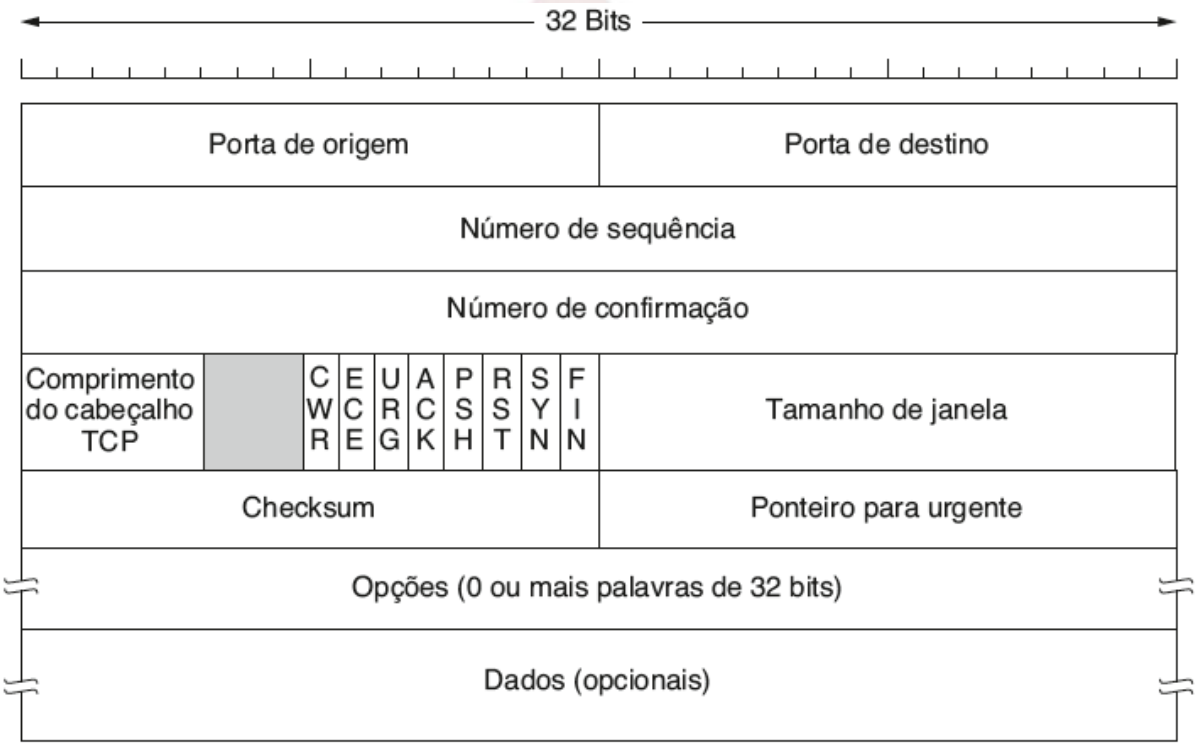
Protocolo TCP

- Características
  - Orientado a *stream* (fluxo)
    - Os dados gerados/recebidos pela aplicação são vistos como uma sequência de bytes, em oposição a uma sequência de pacotes.
  - *Stream* não-estruturada
    - TCP não conhece a estrutura de dados usada pela aplicação
    - não estruturado em mensagens
  - Fornece um serviço confiável para a aplicação
    - Entrega os dados em sequência, sem duplicações ou erros.
    - Possui mecanismos de controle de erro e de fluxo.

Protocolo TCP

- Características do TCP
  - Orientado à conexão
    - Antes do início da transferência de dados é necessário que ambas as partes estejam sincronizadas, com reserva de recursos
      - Abertura da conexão, transferência dos dados e fechamento da conexão
      - Um único "arquivo" por vez → transmissão sequencial
  - Conexões *full-duplex*
    - Duas sequências de bytes (*streams*) independentes fluindo em direções opostas, com nenhuma interação aparente
  - Buffers de transmissão e retransmissão

Cabeçalho do segmento TCP

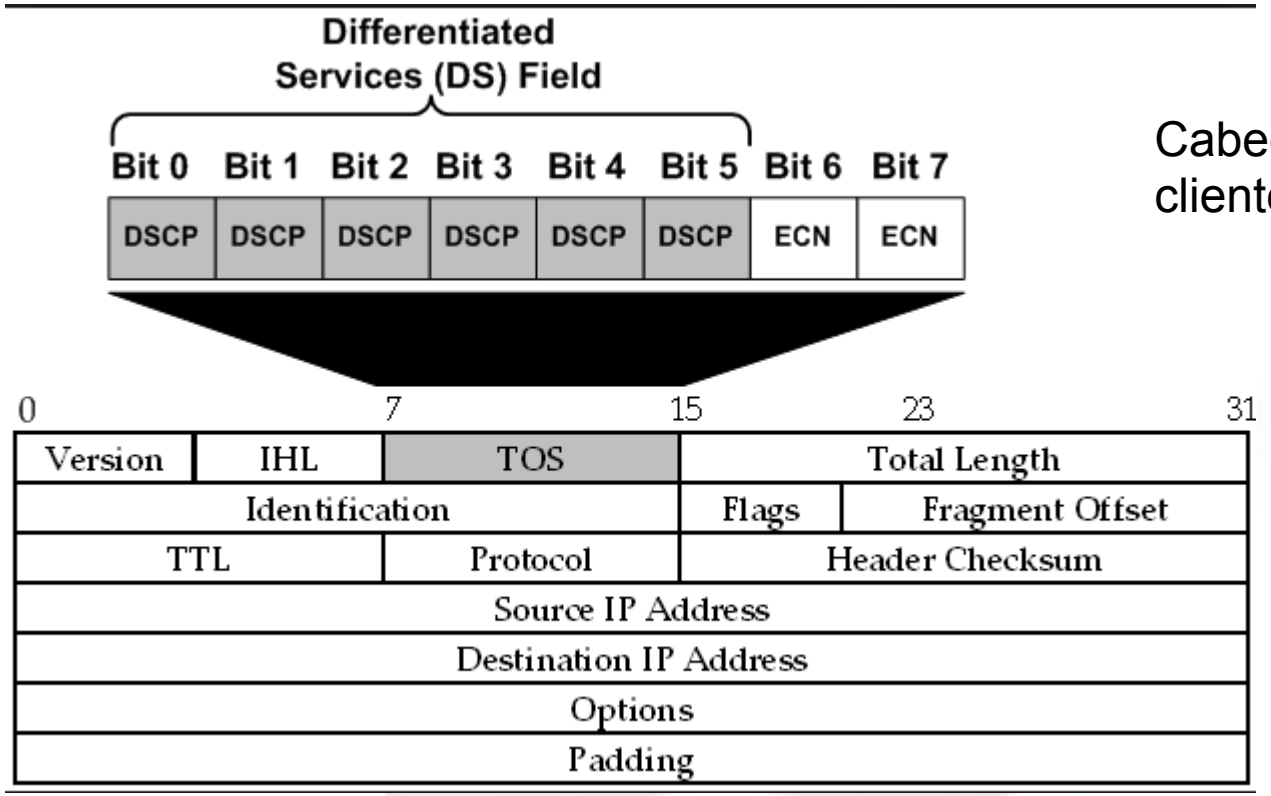


Cabeçalho TCP.

Campos do TCP

- Source port (16 bits) - Porta de origem;
- Destination port (16 bits) – Porta de destino;
- Sequence number (32 bits) - tem um papel duplo: Se o flag SYN está marcado (1), então o conteúdo deste campo é o ISN (Inicial Sequence Number). Se não estiver, significa o número de ordem do primeiro byte do segmento na transmissão
- Acknowledgment number (32 bits) – Se o flag ACK estiver marcado, o conteúdo deste campo é o número de sequencia do segmento que o receptor está esperando, confirmando assim o recebimento de todos os segmentos anteriores.
- Data offset (4 bits) – Define o tamanho do cabeçalho em número de linhas de 4 bytes.
- CWR (1 bit) – Congestion Window Reduced (CWR) – flag que indica o recebimento anterior de um segmento com o ECE marcado e a consequente redução da janela de transmissão.
- ECE (1 bit) – ECN-Echo tem significado duplo:
  - Se o flag SYN estiver marcado, significa que o emissor é ECN capable.
  - Se não estiver, e o cabeçalho IP tiver o Congestion Experienced flag marcado (ECN=11), significa que a rede está congestionada.
- URG (1 bit) – Indica que o campo Urgent Pointer deve ser processado
- ACK (1 bit) – Indica que o campo ACK Number deve ser processado
- PSH (1 bit) – Indica que os dados enviados devem ser imediatamente entregues à aplicação.
- RST (1 bit) – “Reseta” a conexão por falta de feedback
- SYN (1 bit) – Synchronize sequence numbers. Apenas o primeiro pacote de cada peer pode ter este flag marcado, já que ele indica o número aleatório de sequencia a partir do qual os segmentos serão numerados.
- FIN (1 bit) – Finalização da conexão, por não haver mais dados a enviar.
- Window size (16 bits) – Número esperado de bytes a receber na janela de transmissão.
- Checksum (16 bits) – Mecanismo de verificação de erros de todo segmento.
- Urgent pointer (16 bits) – Posição em bytes no segmento onde há dados prioritários a processar.

Flags ECT e CE



Cabeçalho IP (roteadores e cliente/servidor)

```
---- Type of Service ----
  {111. ....}= 07h [007d] Precedence: Network Control
  {...1 ....}= 01h [001d] Delay: Low
  {.... 1...}= 01h [001d] Throughput: High
  {.... .1..}= 01h [001d] Reliability: High
  {.... ..1.}= 01h [001d] ECN Capable Transport(ECT): Yes
  {.... ...0}= 00h [000d] Congestion Experienced(CE): No
```

Controle proativo de congestionamento

O pedido de conexão é feito por quem deseja transmitir dados. Um tamanho máximo de janela é combinado. O ECT é marcado, informando que deseja usar o controle de congestionamento

Caso algum roteador no caminho não suporte o controle, descarta o pacote e envia um ICMP "Parameter Problem"

Se o destino não suportar o controle, faz o mesmo. Se suportar, já responde com o ECT marcado.

ECT 1, CE 0	CWR 0, ECE 0, SYN 1	→	ECT 1, CE 0	CWR 0, ECE 0, SYN 1	→	ECT 1, CE 0	CWR 0, ECE 0, SYN 1	→	ECT 1, CE 0	CWR 0, ECE 0, SYN 1
IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header

ECT 1, CE 0	CWR 0, ECE 0, SYN 1, ACK 1	←	ECT 1, CE 0	CWR 0, ECE 0, SYN 1, ACK 1	←	ECT 1, CE 0	CWR 0, ECE 0, SYN 1, ACK 1	←	ECT 1, CE 0	CWR 0, ECE 0, SYN 1, ACK 1
IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header

ECT 1, CE 0	CWR 0, ECE 0, ACK 1	→	ECT 1, CE 0	CWR 0, ECE 0, ACK 1	→	ECT 1, CE 0	CWR 0, ECE 0, SYN 1	→	ECT 1, CE 0	CWR 0, ECE 0, ACK 1
IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header

Roteador congestiona !
O roteador marca o flag CE.

ECT 1, CE 0	CWR 0, ECE 0	→	ECT 1, CE 0	CWR 0, ECE 0	→	ECT 1, CE 1	CWR 0, ECE 0	→	ECT 1, CE 1	CWR 0, ECE 0
IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header

O destino marcará o flag ECE de todos os segmentos até receber um CWR

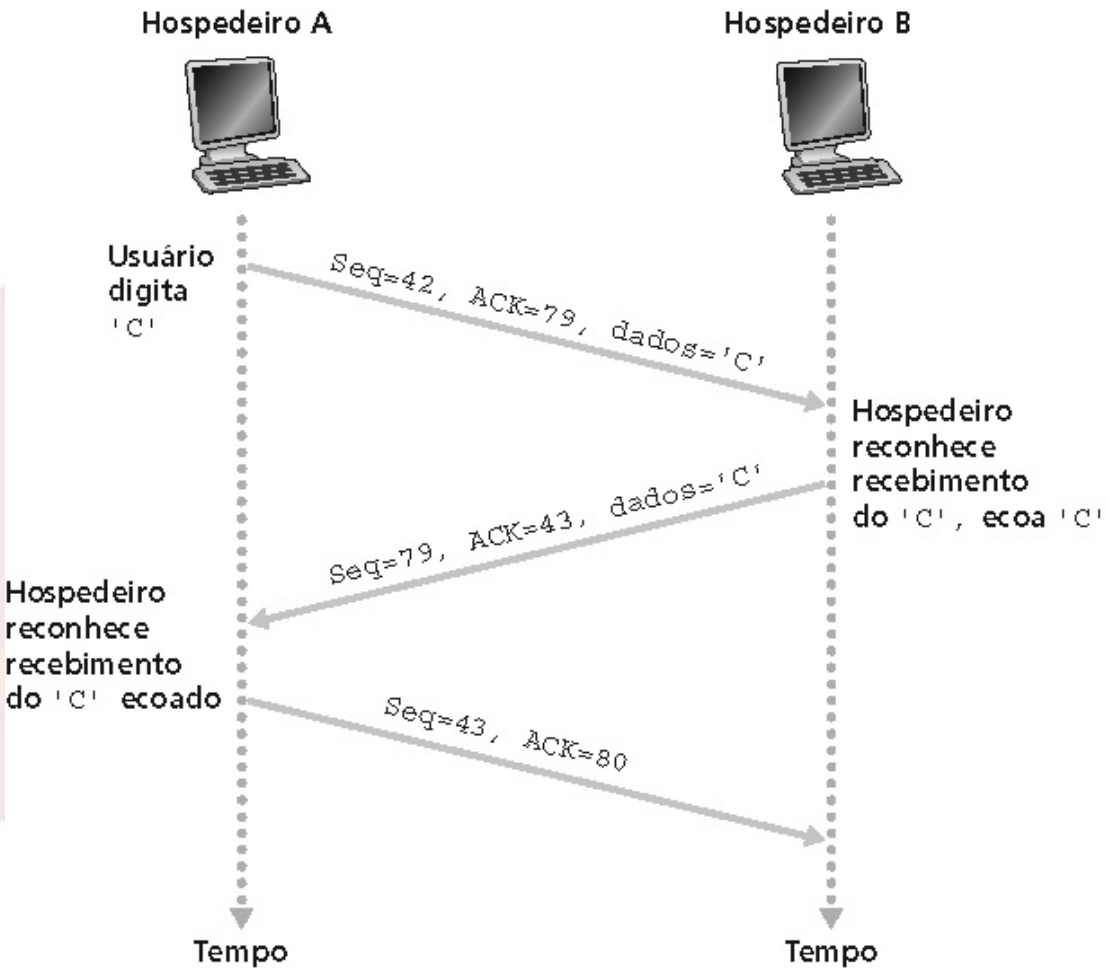
ECT 1, CE 0	CWR 0, ECE 1, ACK 1	←	ECT 1, CE 0	CWR 0, ECE 1, ACK 1	←	ECT 1, CE 0	CWR 0, ECE 1, ACK 1	←	ECT 1, CE 0	CWR 0, ECE 1, ACK 1
IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header

O origem reduz a janela como se tivesse havido perda de pacotes e informa através do CWR

ECT 1, CE 0	CWR 1, ECE 0	→	ECT 1, CE 0	CWR 1, ECE 0	→	ECT 1, CE 0	CWR 1, ECE 0	→	ECT 1, CE 0	CWR 1, ECE 0
IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header		IP Header	TCP Header

Números de sequência

- Números de sequência:
  - Número do primeiro byte nos segmentos de dados
- ACKs:
  - Número do próximo byte esperado do outro lado
  - ACK cumulativo



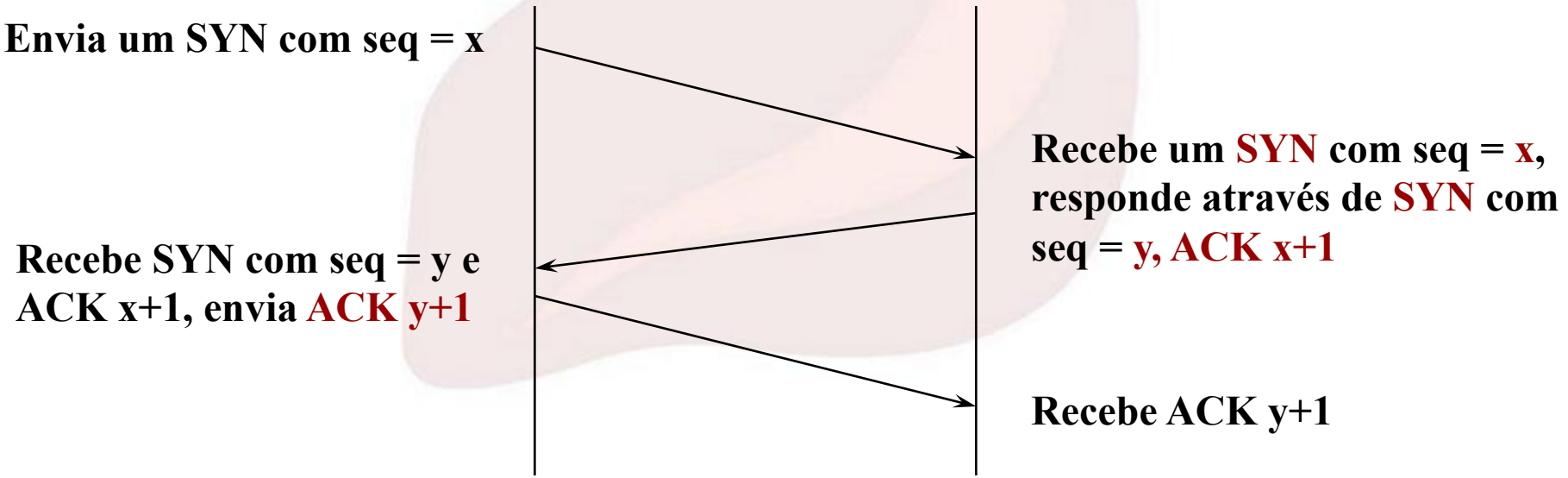
Gerenciamento de conexão

- TCP transmissor estabelece conexão com o receptor antes de trocar segmentos de dados
- Inicializar variáveis:
  - Números de sequência
  - Buffers, controle de fluxo (ex. RcvWindow)

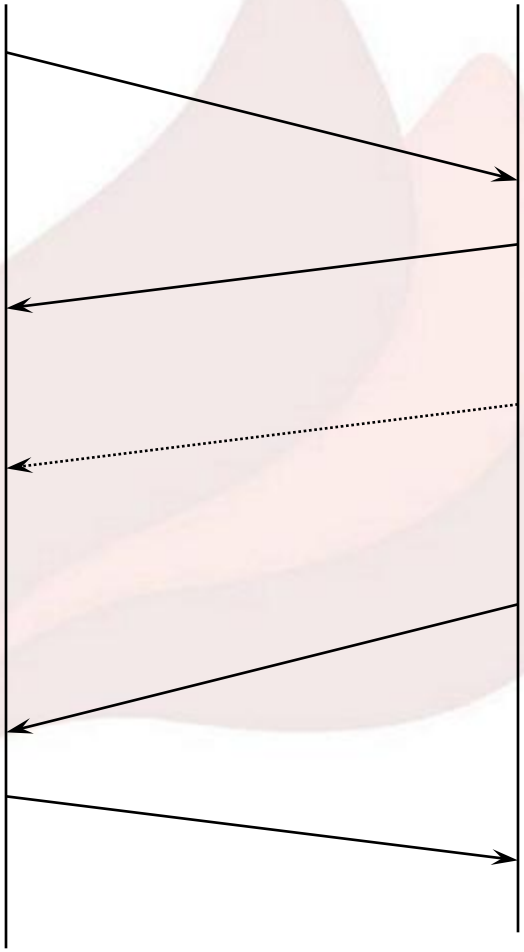
- Three way handshake:
  - Passo 1:
    - Sistema final cliente envia TCP SYN ao servidor
    - Especifica número de sequência inicial
  - Passo 2:
    - Sistema final servidor que recebe o SYN, responde com segmento SYN-ACK
    - Reconhece o SYN recebido
    - Aloca buffers
    - Especifica o número de sequência inicial do servidor
  - Passo 3:
    - O sistema final cliente reconhece o SYNACK

Abertura de conexão TCP

Three-way Handshake



Fechamento de Conexão TCP



Aplicação fecha a conexão
Envia um FIN com seq = x

Recebe um FIN com seq = x,
responde através de ACK x+1
Informa à aplicação

Recebe ACK x+1

(half close)

Aplicação fecha a conexão
Envia um FIN com seq = y,
ACK x+1

Recebe FIN com seq = y e
ACK x+1, envia ACK y+1

Recebe ACK y+1

Controle de fluxo

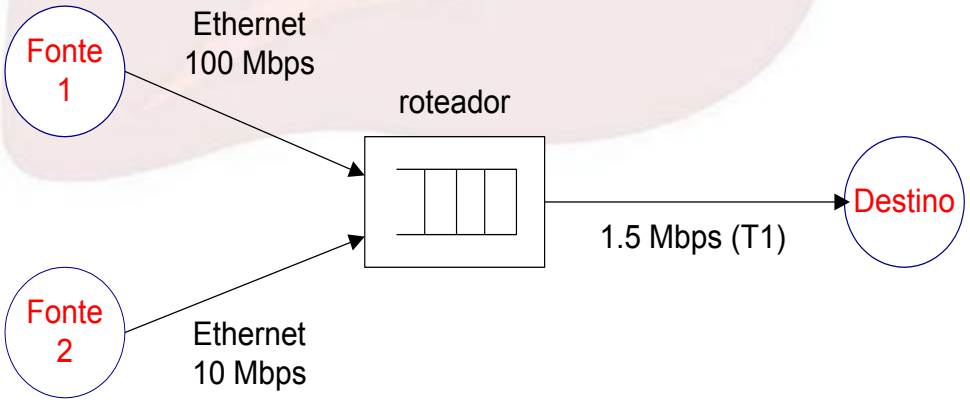
- Evita que o emissor sobrecarregue o receptor
  - Uma função fim-a-fim
  - Baseado no tamanho da janela de recepção
    - Tamanho informado em *bytes*

Controle de Erros

- Entrega confiável de pacotes
  - Controle de erros (confiabilidade)
    - Usa estratégias de retransmissão
  - Reconhecimentos positivos
    - A cada segmento recebido, o receptor envia um reconhecimento (ACK)
  - Reconhecimento acumulativo
    - receptor reconhece o número de sequência do próximo octeto esperado
  - preserva a ordem de transmissão dos pacotes
    - receptor só passa um pacote para a aplicação quando todos os pacotes com número de seqüência inferior já foram entregues

Controle de Congestionamento

- Na comutação de pacotes, os usuários competem por um número limitado de recursos
  - banda passante dos enlaces
  - *buffers* nos roteadores
- Um congestionamento ocorre quando o número de pacotes esperando a transmissão ultrapassa o tamanho dos *buffers* nos roteadores
- Controle de congestionamento
  - Evita que uma grande quantidade de dados sejam injetados na rede, fazendo com que os roteadores e os enlaces fiquem sobrecarregados



Controle de congestionamento

- *Slow Start with Collision Avoidance*
- Na perda de um segmento:
  - É utilizado o *multiplicative decrease*.
    - Redução multiplicativa
  - A janela é reduzida pela metade.
  - desta forma, o tráfego é reduzido exponencialmente nas perdas.
- Na chegada de um ack após o Máx:
  - É utilizado o *additive increase*.
    - Aumento aditivo
  - O tamanho da janela é incrementado de 1 segmento