Redes de Computadores I

# Camada de Transporte

Prof. Arliones Hoeller

arliones.hoeller@ifsc.edu.br

abril de 2014

INSTITUTO FEDERAL
SANTA CATARINA
Campus São José

# Camada de transporte

Objetivos do capítulo:

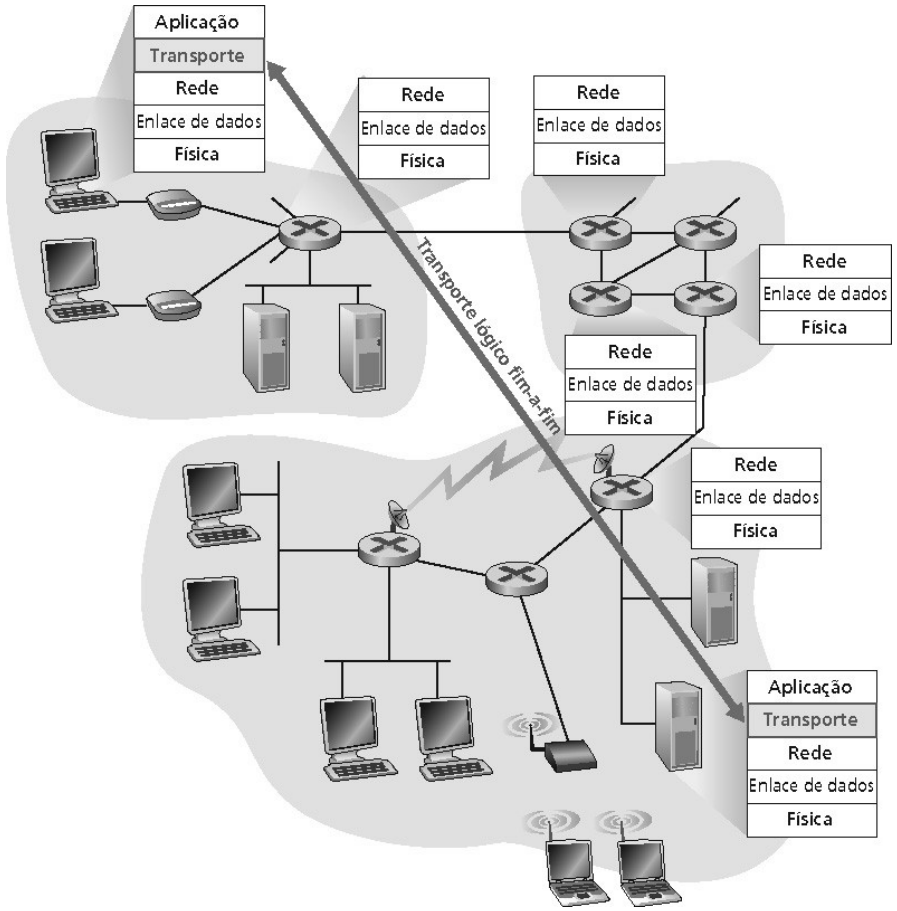
- Entender os princípios por trás dos serviços da camada de transporte:
  - Multiplexação/demultiplexação
  - Transferência de dados confiável
  - Controle de fluxo
  - Controle de congestionamento
- Aprender sobre os protocolos de transporte na Internet:
  - UDP: transporte não orientado à conexão
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

# Camada de transporte

- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP

# Protocolos e serviços de transporte

- Fornecem comunicação lógica entre processos de aplicação em diferentes hospedeiros
- Os protocolos de transporte são executados nos sistemas finais
  - Lado emissor: quebra as mensagens da aplicação em segmentos e envia para a camada de rede
  - Lado receptor: remonta os segmentos em mensagens e passa para a camada de aplicação
- Há mais de um protocolo de transporte disponível para as aplicações
  - Internet: TCP e UDP



# Camada de transporte vs. camada de rede

- Camada de rede: comunicação lógica entre os hospedeiros
- Camada de transporte: comunicação lógica entre os processos
  - Depende dos serviços da camada de rede

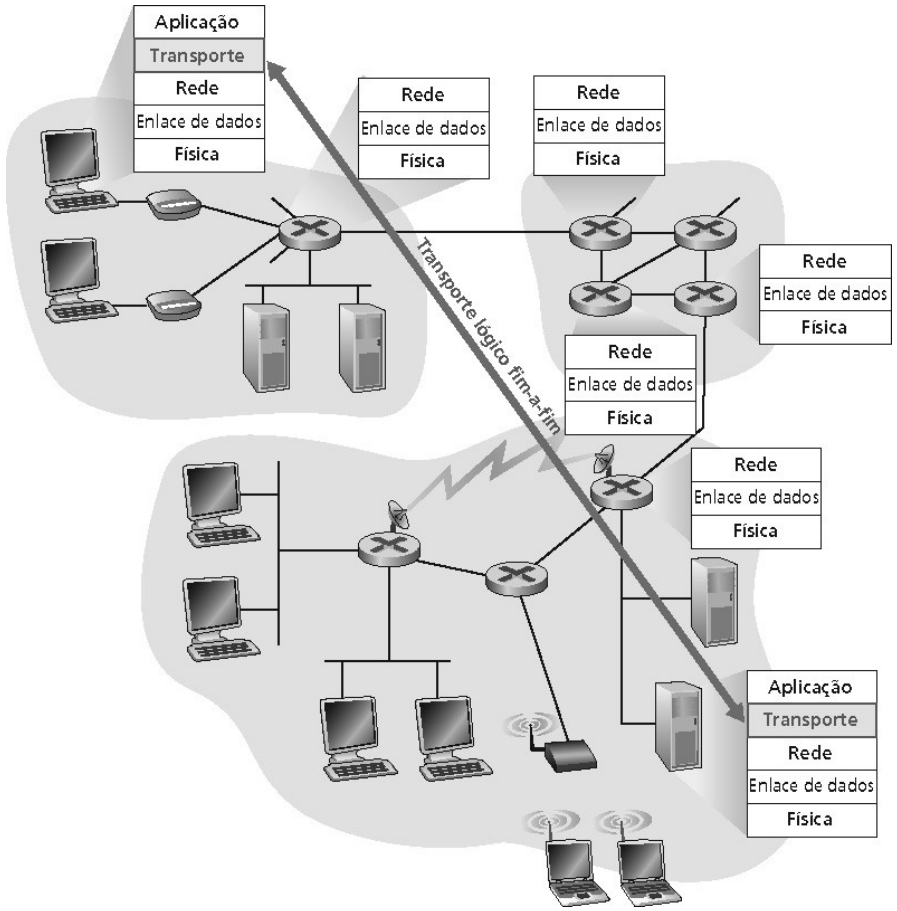
**Analogia com uma casa familiar:**

12 crianças enviam cartas para 12 crianças

- Processos = crianças
- Mensagens da aplicação = cartas nos envelopes
- Hospedeiros = casas
- Protocolo de transporte = Anna e Bill
- Protocolo da camada de rede = serviço postal

# Protocolos da camada de transporte da Internet

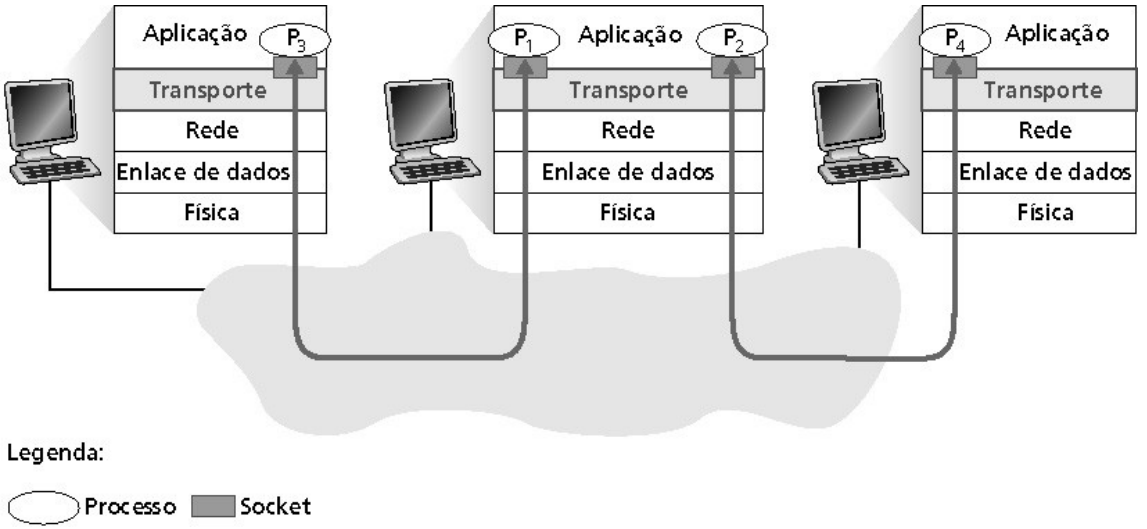
- Confiável, garante ordem de entrega (TCP)
- Controle de congestionamento
  - Controle de fluxo
  - Orientado à conexão
- Não confiável, sem ordem de entrega: UDP
  - Extensão do “melhor esforço” do IP
- Serviços não disponíveis:
  - Garantia a atrasos
  - Garantia de banda



# Camada de transporte

- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP

# Multiplexação/demultiplexação




Demultiplexação no hospedeiro receptor:

Entrega os segmentos recebidos ao socket correto

Multiplexação no hospedeiro emissor:

Coleta dados de múltiplos sockets, envelopa os dados com cabeçalho (usado depois para demultiplexar)

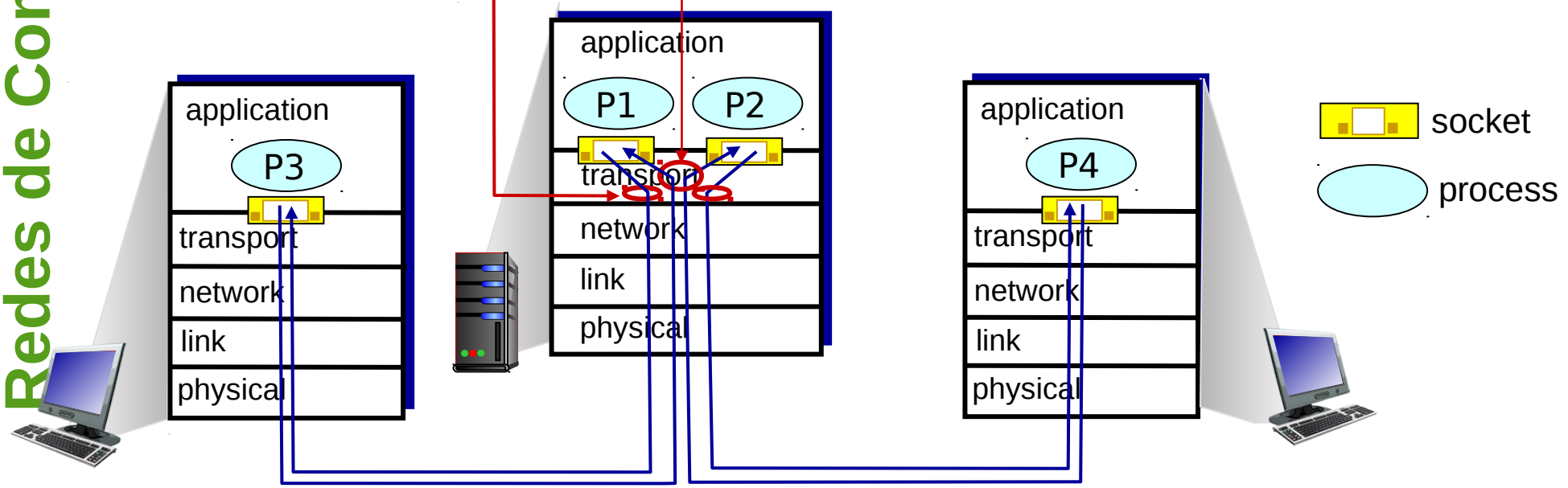
# Multiplexing/demultiplexing

*multiplexing at sender:*

handle data from multiple sockets, add transport header (later used for demultiplexing)

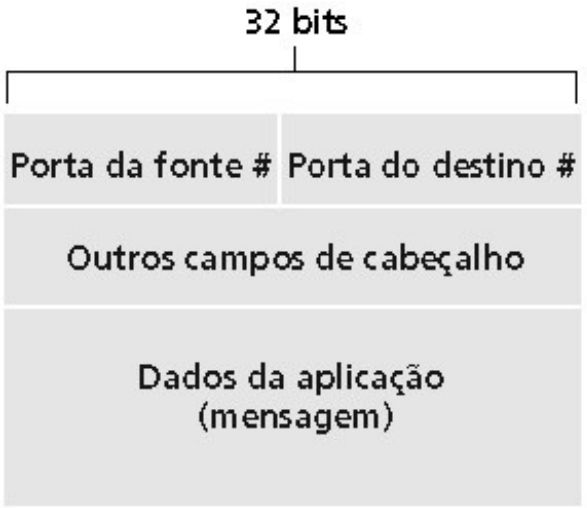
*demultiplexing at receiver:*

use header info to deliver received segments to correct socket



# Como funciona a demultiplexação

- Computador recebe datagramas IP
  - Cada datagrama possui endereço IP de origem e IP de destino
  - Cada datagrama carrega 1 segmento da camada de transporte
  - Cada segmento possui números de porta de origem e destino (lembre-se: números de porta bem conhecidos para aplicações específicas)
- O hospedeiro usa endereços IP e números de porta para direcionar o segmento ao socket apropriado

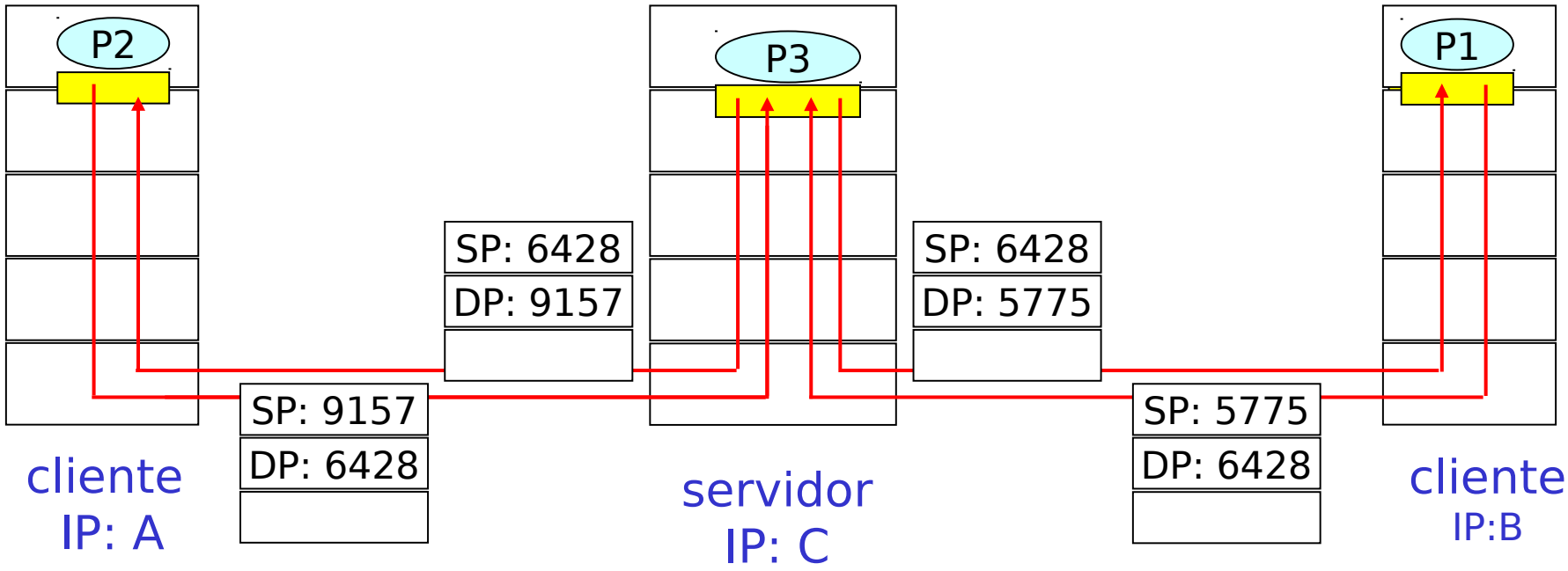


# Demultiplexação não orientada à conexão

- Socket UDP identificado por 2 valores:
  (endereço IP de destino, número da porta de destino)
- Quando o hospedeiro recebe o segmento UDP:
  - Verifica o número da porta de destino no segmento
  - Direciona o segmento UDP para o socket com este número de porta
- Datagramas com IP de origem diferentes e/ou portas de origem diferentes são direcionados para o mesmo socket

# Demultiplexação não orientada à conexão

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

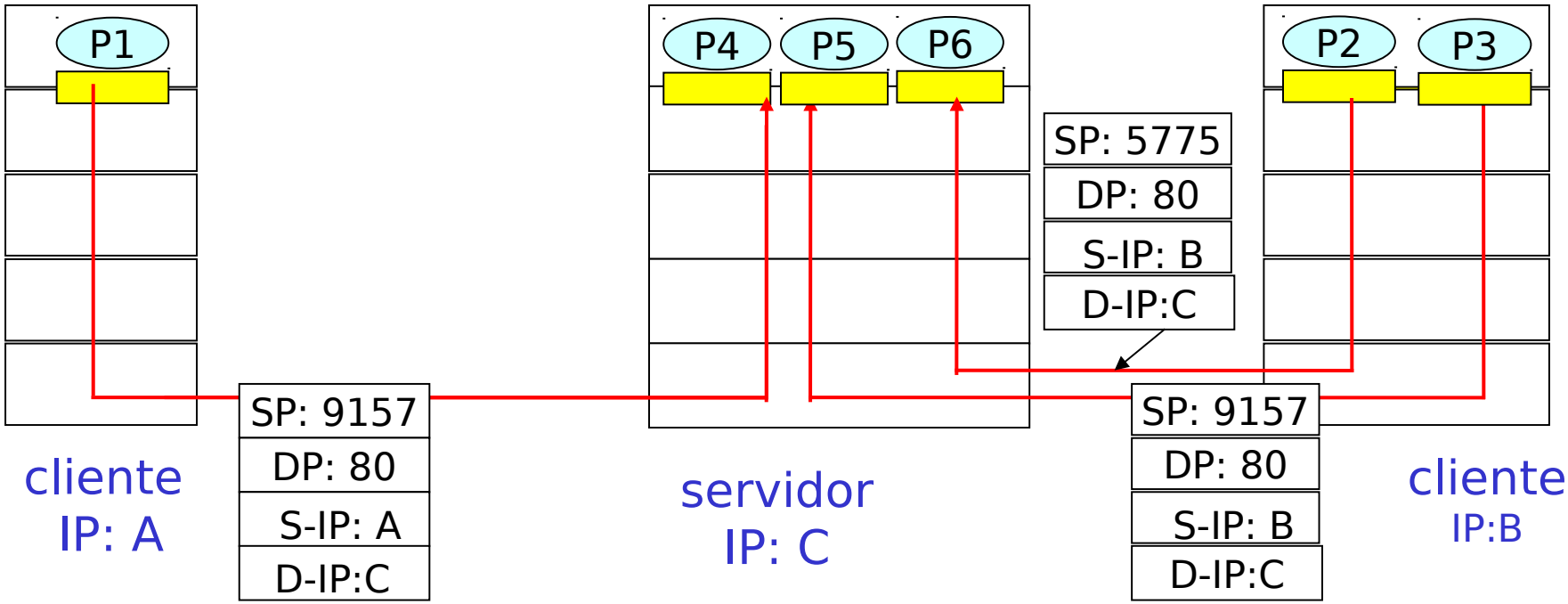



SP fornece o “endereço de retorno” 0

# Demux orientada à conexão

- Socket TCP  identificado por 4 valores:
  - Endereço IP de origem
  - End. porta de origem
  - Endereço IP de destino
  - End. porta de destino
- Hospedeiro receptor usa os quatro valores para direcionar o segmento ao socket apropriado
- Hospedeiro servidor pode suportar vários sockets TCP simultâneos:
  - Cada socket é identificado pelos seus próprios 4 valores
  - Servidores Web possuem sockets diferentes para cada cliente conectado
  - HTTP não persistente terá um socket diferente para cada requisição

# Demux orientada à conexão



# Camada de transporte

- 3.1 Serviços da camada de transporte
- 3.2 Multiplexação e demultiplexação
- 3.3 Transporte não orientado à conexão: UDP
- 3.4 Princípios de transferência confiável de dados
- 3.5 Transporte orientado à conexão: TCP
  - Estrutura do segmento
  - Transferência confiável de dados
  - Controle de fluxo
  - Gerenciamento de conexão
- 3.6 Princípios de controle de congestionamento
- 3.7 Controle de congestionamento do TCP

# UDP: User Datagram Protocol [RFC 768]

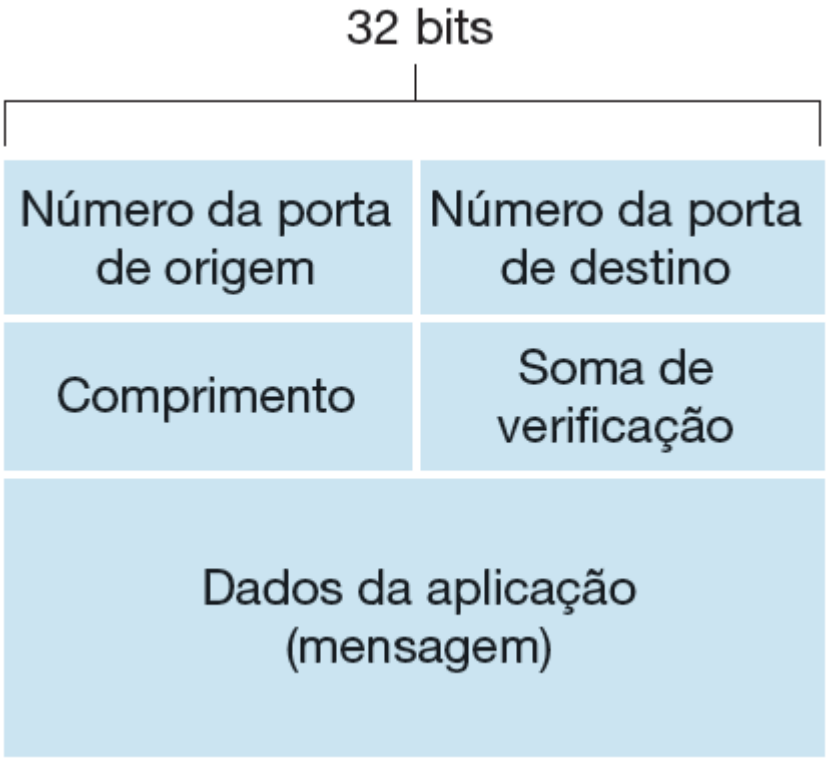
- Protocolo de transporte da Internet “sem gorduras” “sem frescuras”
- Serviço “best effort” , segmentos UDP podem ser:
  - Perdidos
  - Entregues fora de ordem para a aplicação
- Sem conexão:
  - Não há apresentação entre o UDP transmissor e o receptor
  - Cada segmento  UDP é tratado de forma independente dos outros

Por que existe um UDP?

- Não há estabelecimento de conexão (que possa redundar em atrasos)
- Simples: não há estado de conexão nem no transmissor, nem no receptor
- Cabeçalho de segmento reduzido
- Não há controle de congestionamento: UDP pode enviar segmentos tão rápido quanto desejado (e possível)

# Mais sobre UDP

- Muito usado por aplicações de mutimídia contínua (streaming)
  - Tolerantes à perda
  - Sensíveis à taxa
- Outros usos do UDP (por quê?):
  - DNS
  - SNMP
- Transferência confiável sobre UDP: acrescentar confiabilidade na camada de aplicação
  - Recuperação de erro específica de cada aplicação



# UDP checksum

Objetivo: detectar “erros” (ex.: bits trocados) no segmento transmitido

Transmissor:

- Trata o conteúdo do segmento como seqüência de inteiros de 16 bits
- Checksum: soma (complemento de 1 da soma) do conteúdo do segmento
- Transmissor coloca o valor do checksum no campo de checksum do UDP

Receptor:

- Computa o checksum do segmento recebido
- Verifica se o checksum calculado é igual ao valor do campo checksum:
  - NÃO - erro detectado
  - SIM - não há erros. **Mas, talvez haja erros apesar disso***?* Mas depois...