Aula 9 - Conceitos de Camada de Transporte, UDP

Diego Passos

Universidade Federal Fluminense

Redes de Computadores I

Material adaptado a partir dos slides originais de J.F Kurose and K.W. Ross.

Revisão da Última Aula...

- **Sockets**: API para aplicações de rede.
  - Presente na maioria das linguagens.
  - Abstrações similares.
    - Criação do socket, conexão, envio de dados, recepção, fechamento.
  - Fornece modelos de serviço diferentes: UDP *vs.* TCP.
    - Sem conexão *vs.* orientado a conexão.
    - Sem confiabilidade *vs.* entrega confiável de dados.
    - ...
  - **Bind()**: associa socket a uma porta.
    - **Não pode haver dois ou mais sockets associados à mesma porta de um mesmo protocolo de transporte!**

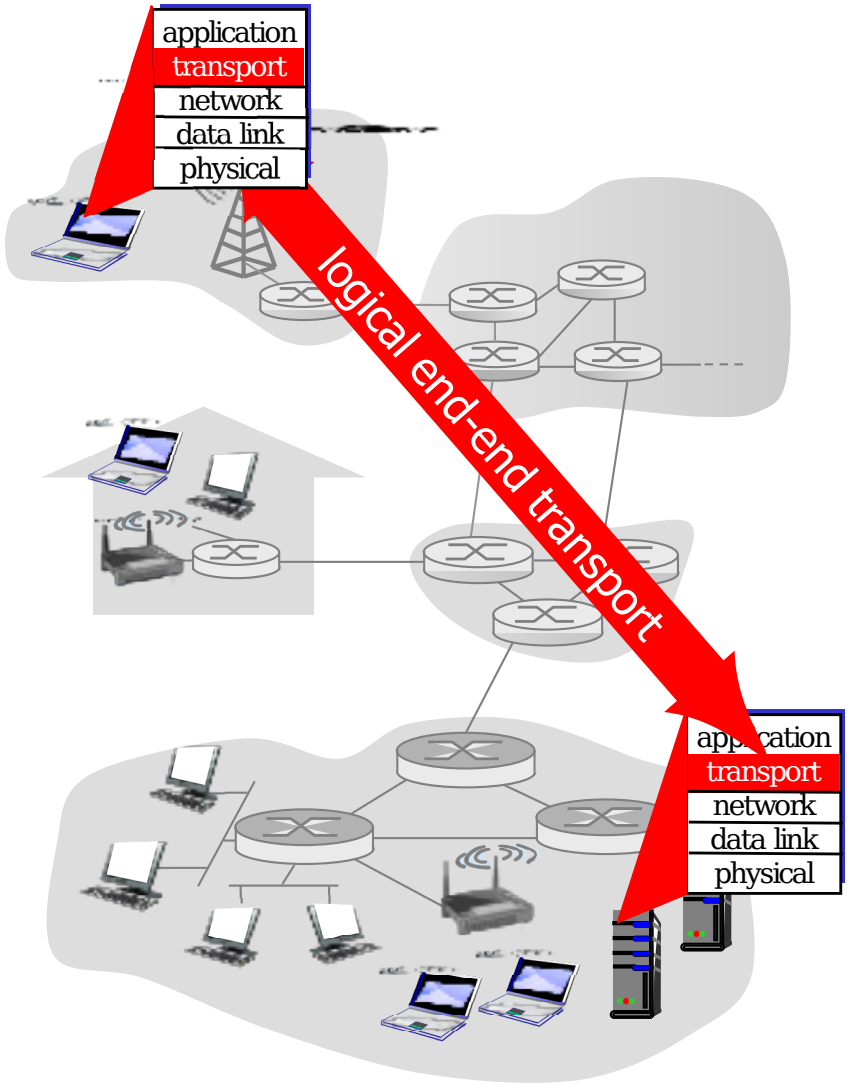
Capítulo 3: Camada de Transporte

- **Nossos objetivos:**
  - Compreender os princípios dos serviços da camada de transporte.
    - Multiplexação, demultiplexação.
    - Transmissão confiável de dados.
    - Controle de fluxo.
    - Controle de congestionamento.
  - Aprender sobre os protocolos de camada de transporte da Internet.
    - UDP: serviço de transporte sem conexão.
    - TCP: serviço orientado a conexão, confiável.
    - Controle de congestionamento do TCP.

Serviços da Camada de Transporte

Serviços e Protocolos de Transporte

- Provê **comunicação lógica** entre **processos** da aplicação rodando em *hosts* diferentes.
- Protocolos de transporte **são executados nos sistemas finais**.
  - Lado transmissor: quebra mensagens da aplicação em **segmentos**, passa segmentos para camada de rede.
  - Lado receptor: remonta segmentos para formar mensagens originais, passa mensagens para a camada de aplicação.
- Mais de um protocolo disponível para as aplicações.
  - Na Internet: TCP e UDP.



Camada de Transporte *vs.* Camada de Rede

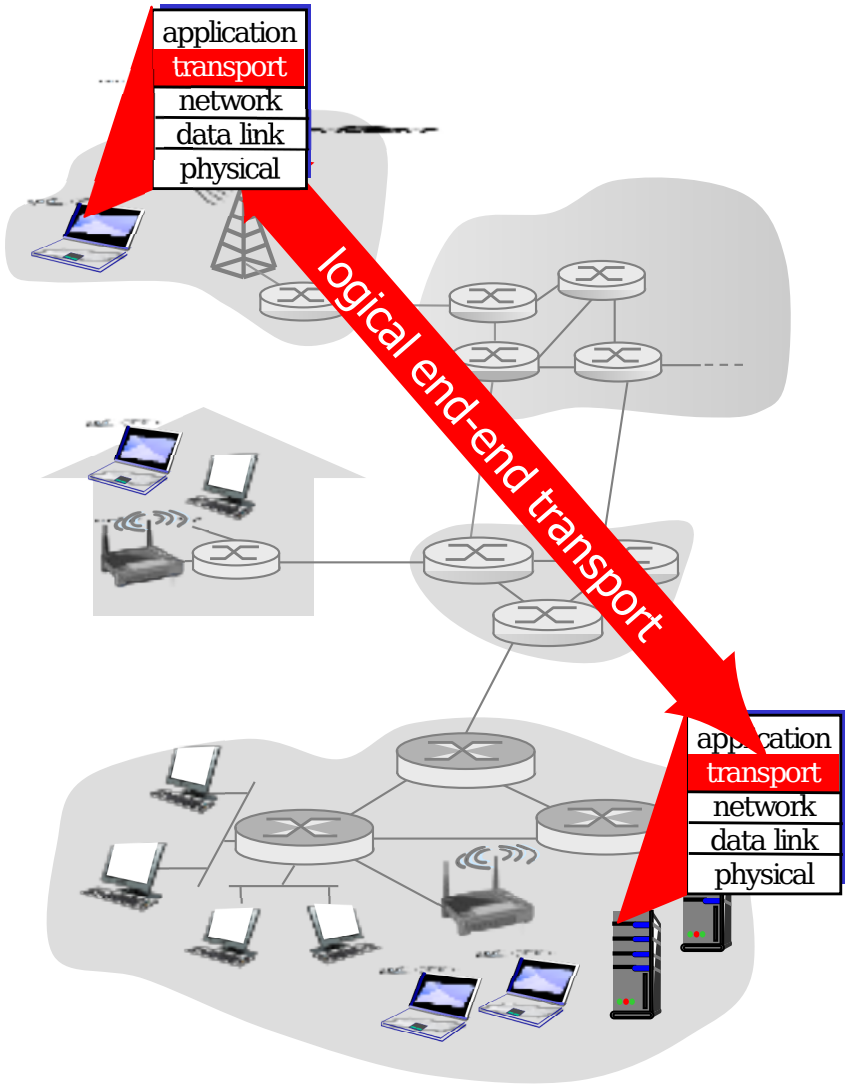
- **Camada de Rede:** comunicação lógica entre *hosts*.
- **Camada de Transporte:** comunicação lógica entre processos.
  - Depende de, e aperfeiçoa, serviços da camada de rede.

Analogia doméstica

- 12 crianças na casa da Ann enviam cartas a 12 crianças na casa do Bill.
  - *hosts* = casas.
  - processos = crianças.
  - mensagens da aplicação = cartas nos envelopes.
  - protocolo de transporte = Ann e Bill que demultiplexam cartas para as crianças.
  - protocolo de camada de rede = correios.

Protocolos de Camada de Transporte da Internet

- Entrega confiável, em ordem (TCP).
  - Controle de congestionamento.
  - Controle de fluxo.
  - *Setup* da conexão.
- Entrega não-confiável, não-ordenada (UDP).
  - Extensão básica do serviço de "melhor esforço" do IP.
- Serviços não disponíveis (nem TCP, nem UDP):
  - Garantias de atraso máximo.
  - Garantias de vazão mínima.



Multiplexação e Demultiplexação

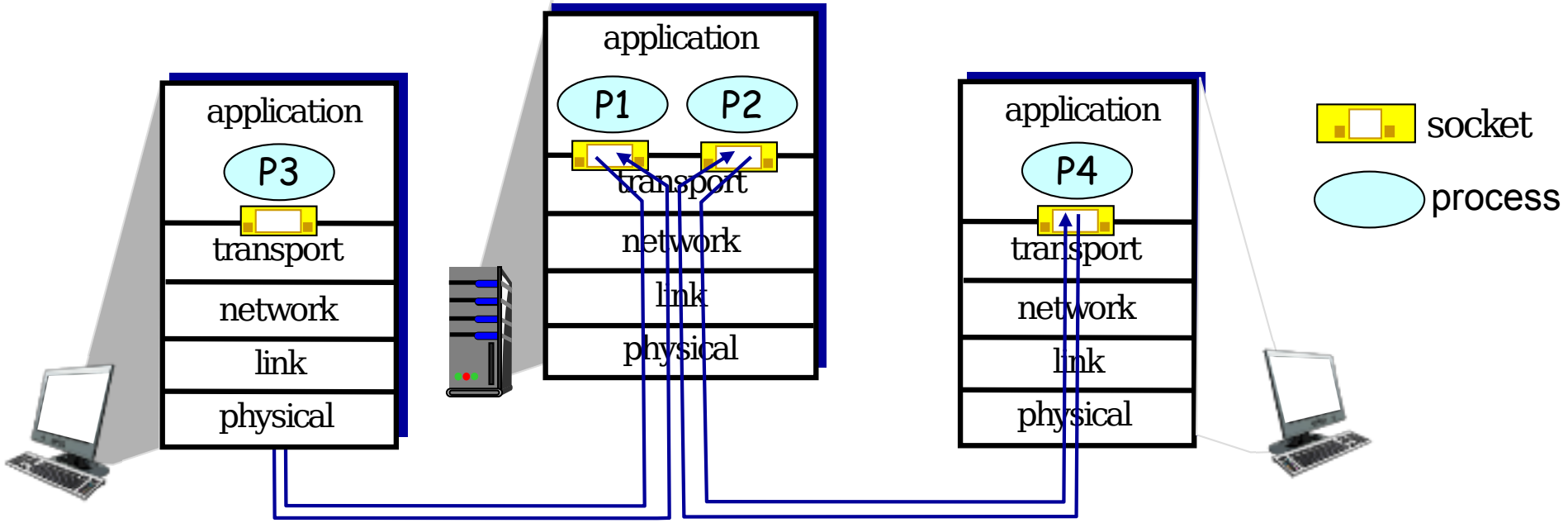
Multiplexação/Demultiplexação

Multiplexação no Transmissor

Lida com dados de múltiplos sockets, adiciona cabeçalho da camada de transporte (usado posteriormente para demultiplexação)

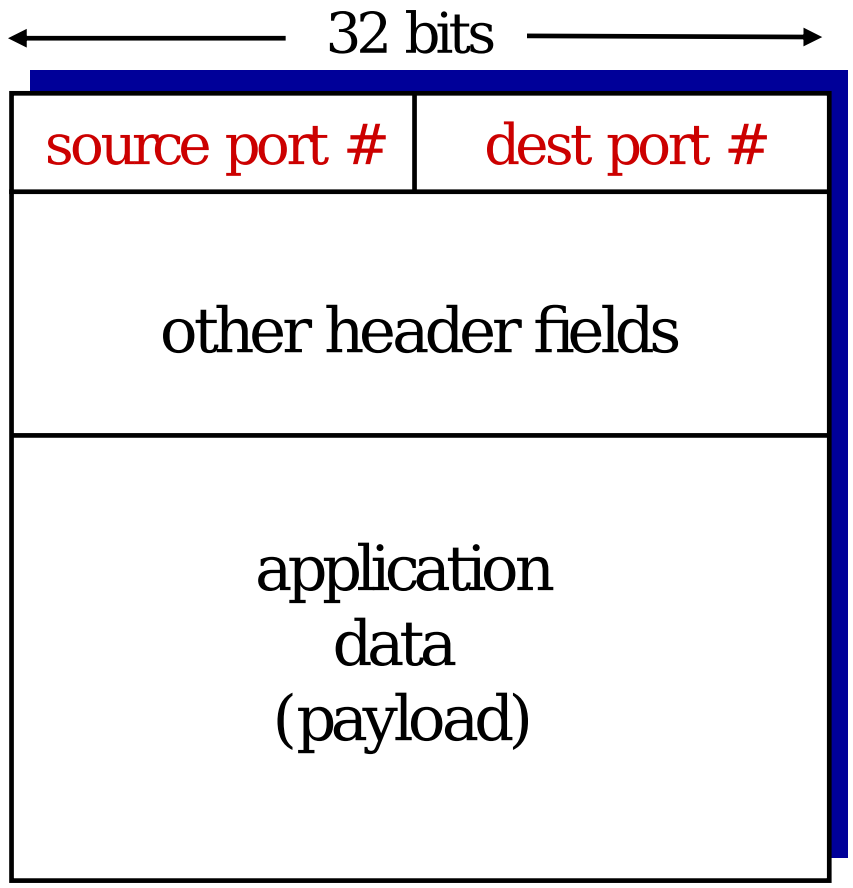
Demultiplexação no Receptor

Usa informação do cabeçalho para entregar segmentos recebidos para o socket correto



Como a Demultiplexação Ocorre

- *Host* recebe datagrama IP.
  - Cada datagrama possui um endereço IP de origem, endereço IP de destino.
  - Cada datagrama carrega um segmento de camada de transporte.
  - Cada segmento possui **números de porta de origem e de destino**.
- *Host* utiliza **tanto os endereços IP quanto os números de porta** para direcionar segmentos aos sockets apropriados.



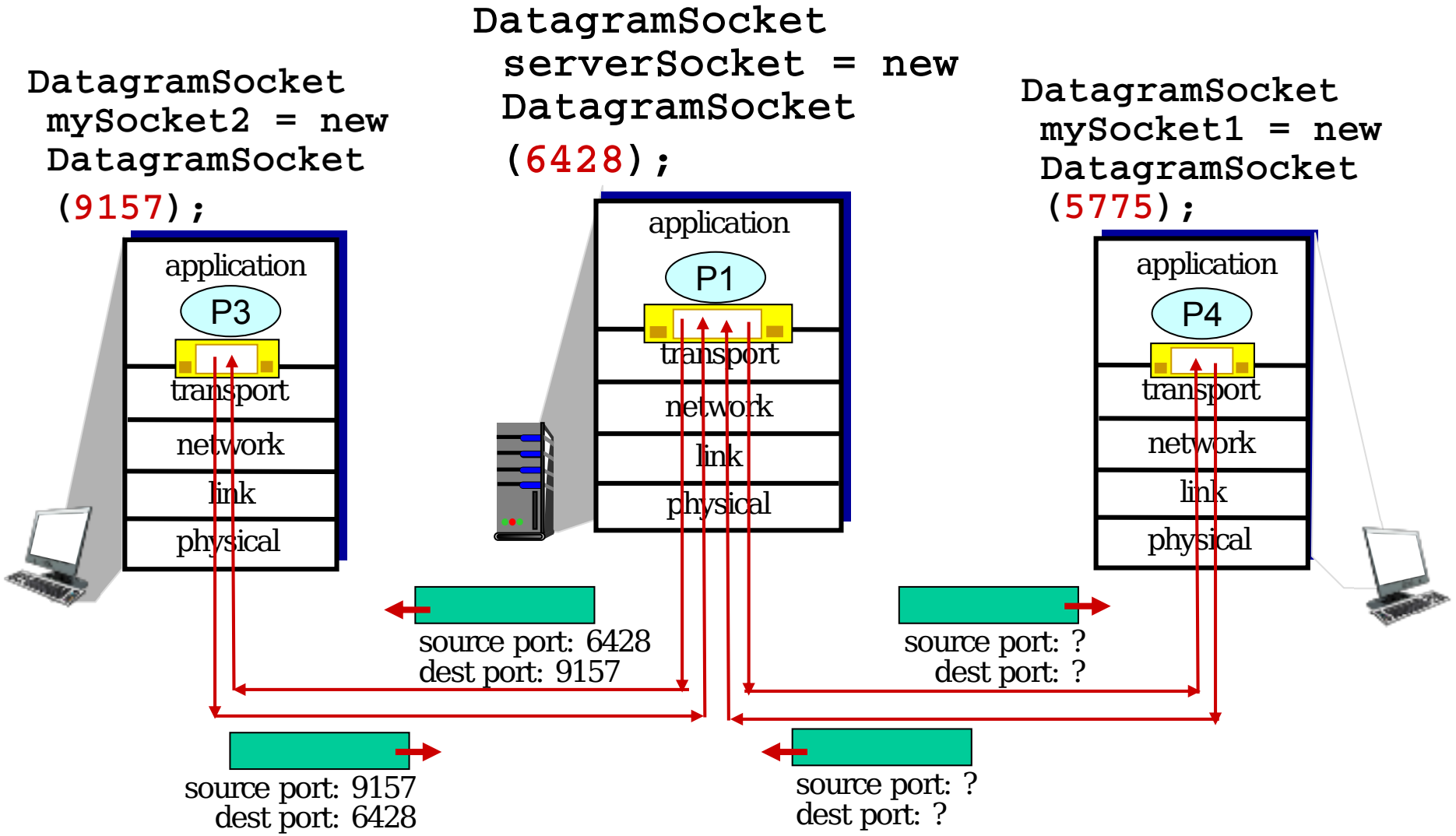
TCP/UDP segment format

Demultiplexação Sem Conexão

- **Lembre-se:** socket criado tem # de porta no *host*.
  `DatagramSocket mySocket1 = new DatagramSocket(12534);`
- **Lembre-se:** quando criamos datagrama para enviar pelo socket UDP, é preciso especificar:
  - Endereço IP de destino.
  - # de porta de destino.

- Quando *host* recebe segmento UDP:
  - Verifica o # de porta de destino no segmento.
  - Direciona o segmento UDP para o socket com aquele # de porta.
- Datagramas com **o mesmo # de porta de destino**, mas com IPs e/ou portas de origem diferentes serão direcionados **ao mesmo socket** no destinatário.

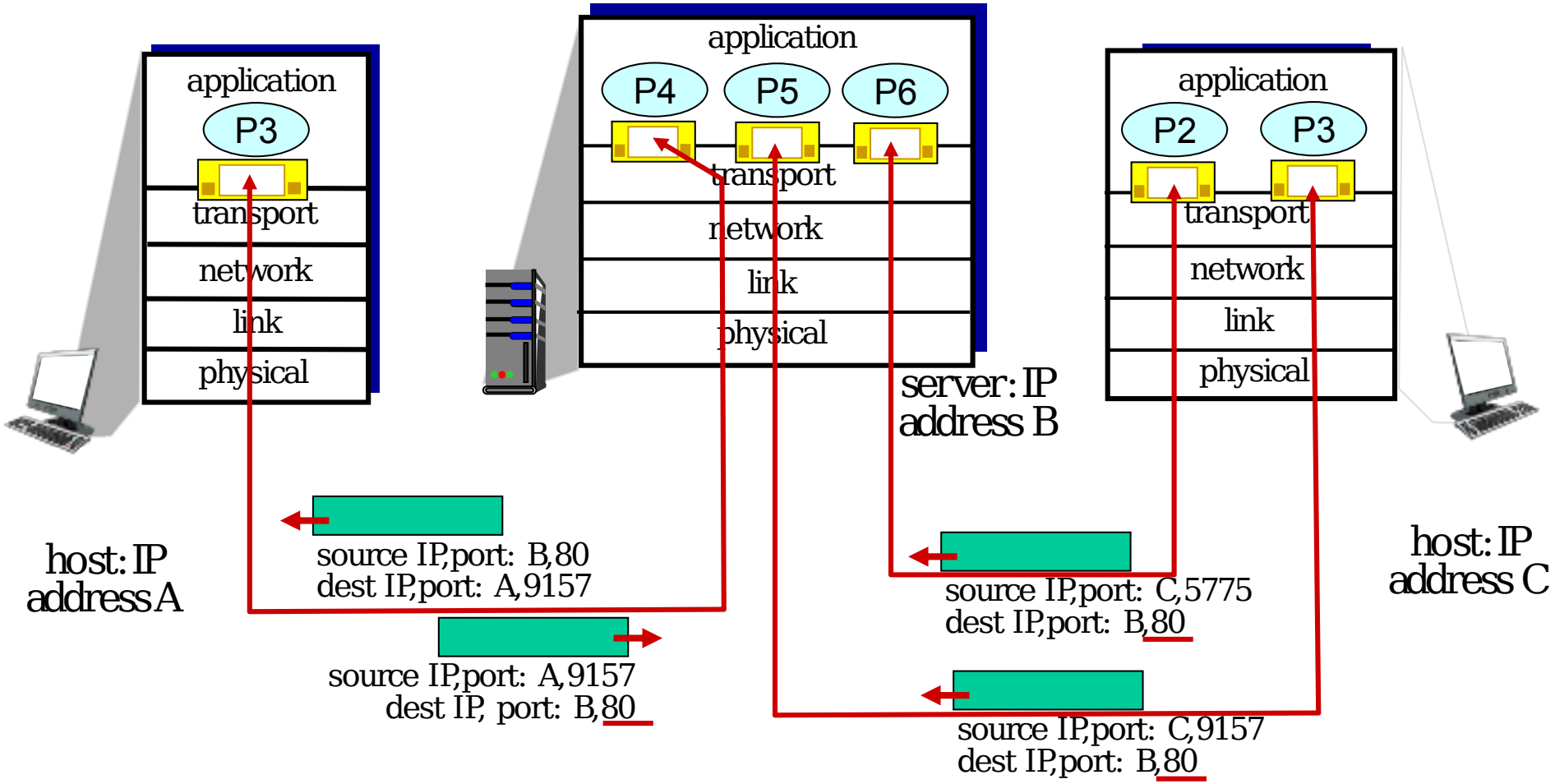
Demultiplexação Sem Conexão: Exemplo



Demultiplexação Orientada a Conexão

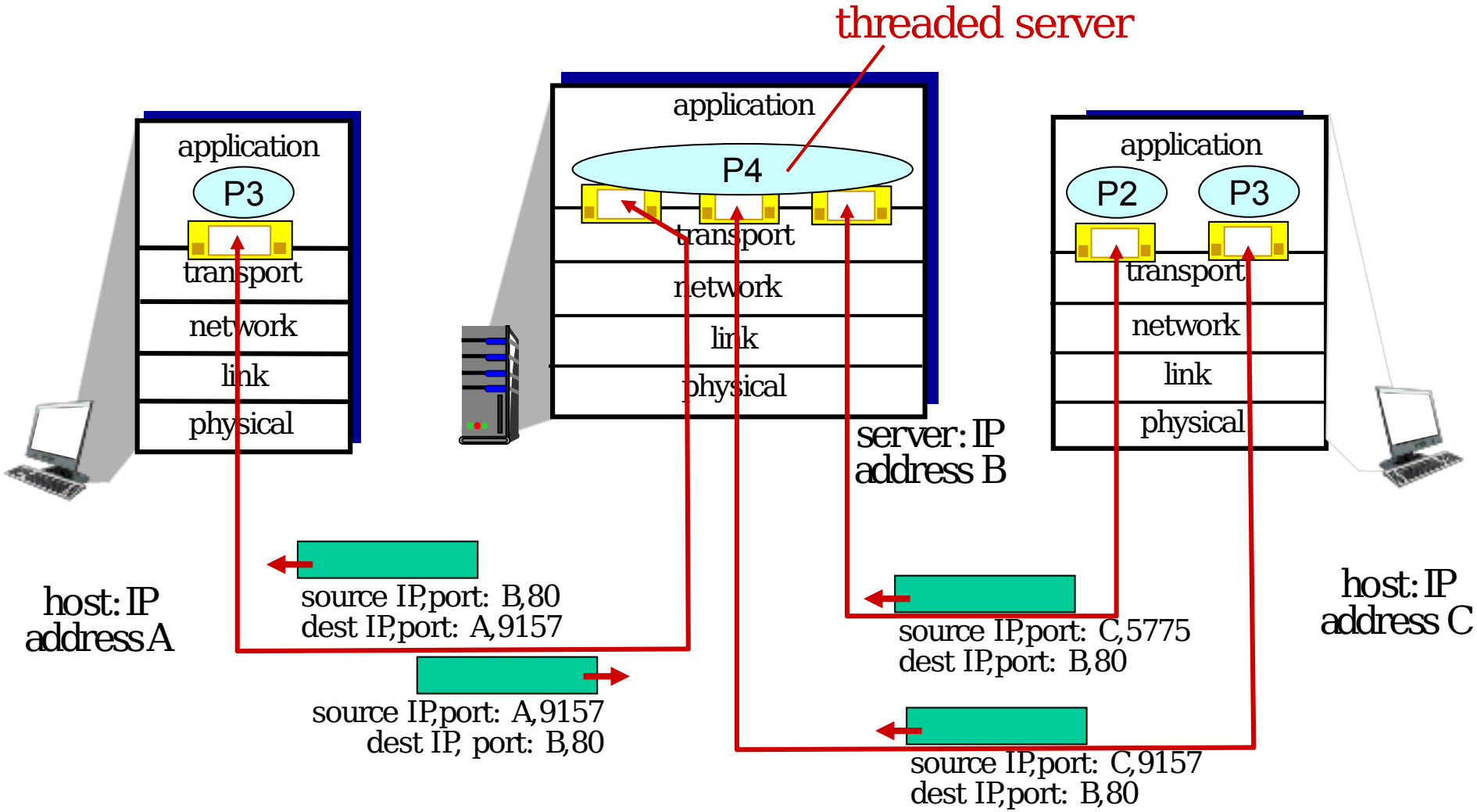
- Socket TCP é identificado por tupla de 4 componentes:
  - **IP de origem**.
  - **IP de destino**.
  - **Porta de origem**.
  - **Porta de destino**.
- Demultiplexação: receptor usa todos os quatro valores para direcionar segmento a socket correto.
- *Host* servidor pode suportar múltiplos sockets TCP simultâneos.
  - Cada socket identificado pela sua própria tupla de quatro valores.
- Servidores web têm sockets diferentes para cada cliente conectado.
  - No HTTP não-persistente, um socket para cada requisição.

Demultiplexação Orientada a Conexão: Exemplo



Três segmentos, todos destinados ao IP de B na porta de destino 80 são demultiplexados para sockets diferentes.

Demultiplexação Orientada a Conexão: Exemplo (Threads)

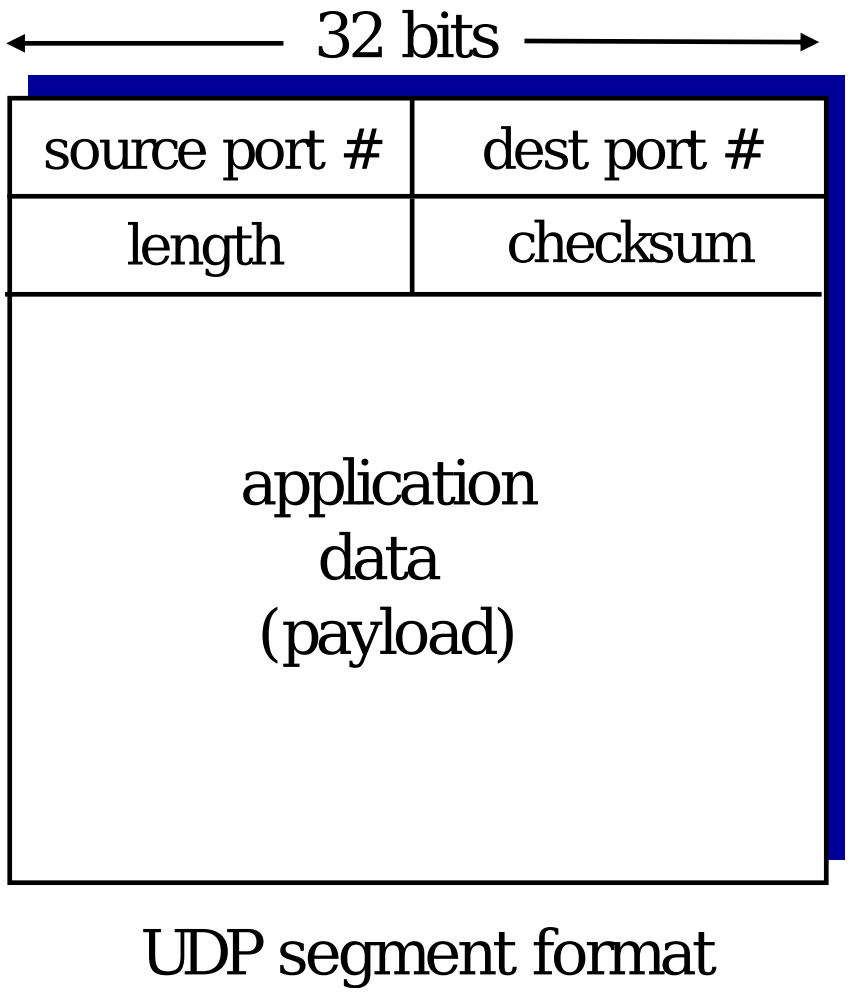


Transporte Sem Conexão: UDP

UDP: *User Datagram Protocol* [RFC 768]

- "Serviço básico", "mínimo" da camada de transporte da Internet.
- Modelo de serviço de "melhor esforço".
- Segmentos UDP podem ser:
  - Perdidos
  - Entregues, porém fora de ordem para a aplicação.
- **Sem conexão:**
  - Não há comunicação inicial entre UDP do transmissor e do receptor.
  - Datagramas são simplesmente enviados.
  - Cada segmento UDP é tratado de forma completamente independente dos demais.
- Usos do UDP:
  - Aplicações de *Streaming* multimídia (tolerantes a perda, sensíveis a taxa).
  - DNS.
  - SNMP.
- Transferência confiável sobre UDP:
  - Possível, mas depende da aplicação.
  - Adição de confiabilidade da própria aplicação.
  - Métodos de recuperação de erros específicos de cada aplicação.

UDP: Cabeçalho de um Segmento



UDP segment format

- **Campo *length***: tamanho do segmento, incluindo cabeçalhos.

Por que existe um UDP?

- Sem estabelecimento de conexão (que adiciona atraso).
- Simples: não armazena estado da comunicação no transmissor ou no receptor.
- Cabeçalho pequeno.
- Sem controle de congestionamento: UDP transmite na mesma taxa que a aplicação gera.

UDP: *Checksum*

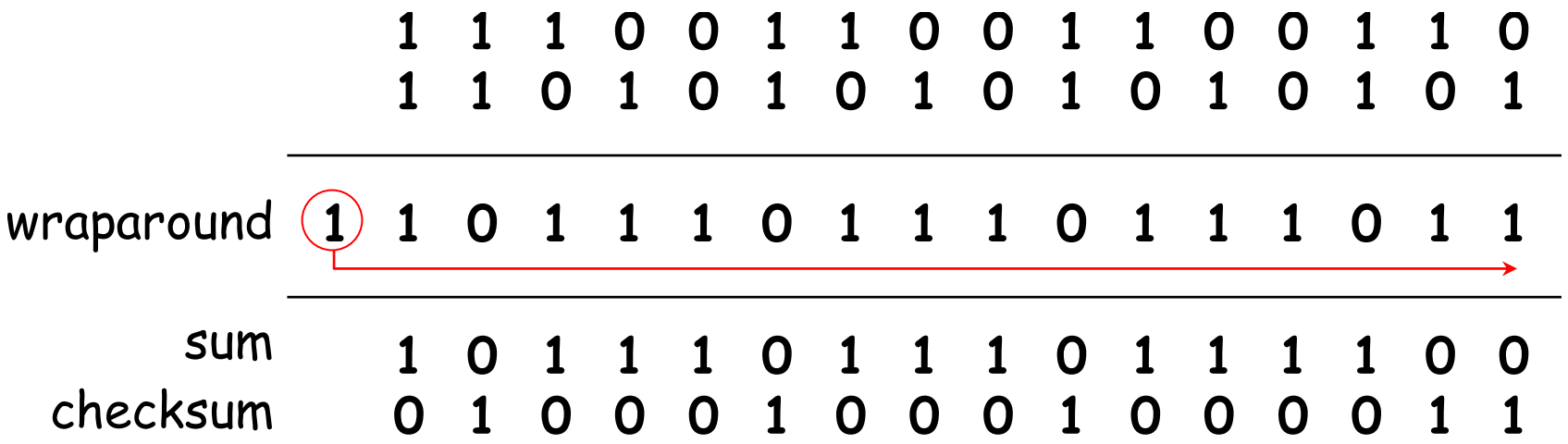
- **Objetivo:** detectar "erros" (*e.g.*, bits com valor trocado) no segmento transmitido.

- **Transmissor:**
  - Trata conteúdo do segmento, incluindo campos de cabeçalho, como uma sequência de inteiros de 16 bits.
  - *Checksum*: soma, em complemento a 1, do conteúdo do segmento.
  - Transmissor coloca valor do *checksum* no campo do cabeçalho UDP.

- **Receptor:**
  - Computa o *checksum* do segmento recebido.
  - *Checksum* computado é igual ao indicado pelo cabeçalho?
    - Não: erro detectado.
    - Sim: nenhum erro detectado.
      - Mas pode haver erros mesmo assim? Mais detalhes em Redes II.

Checksum da Internet: Soma em Complemento a 1

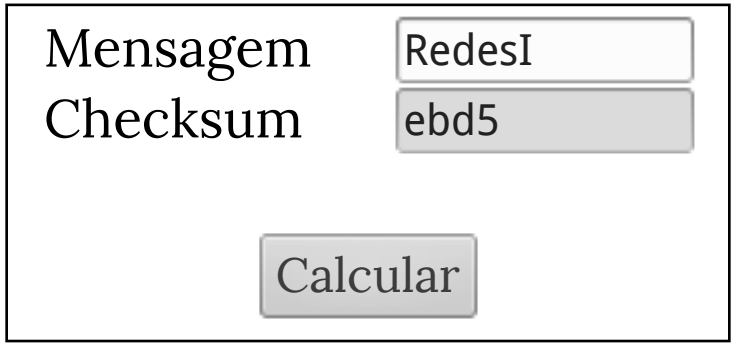
- Soma de dois valores de 16 bits em complemento a 1:



- Note: ao somar dois números, o *vai-um* do bit mais significativo deve ser somado ao resultado.

Internet Checksum: Exemplos

- Experimente o cálculo do *checksum* de algumas mensagens (*strings*):



- Sugestão: calcule o *checksum* de “casa”.
  - Resultado: 0x3d29.
  - Em ASCII: 0x3D → “=”.
  - Em ASCII: 0x29 → “)”.
- **Pergunta:** qual é o *checksum* de “casa)=”?

Resumo da Aula...

- **Camada de transporte:**
  - Comunicação **entre processos**.
  - Executada nas bordas.
  - Transmite **segmentos**.
  - Dois protocolos: TCP, UDP.
- Camada de transporte ≠ camada de rede:
  - **Processos *vs.* *hosts***.
- Modelos de serviço:
  - TCP: confiável, controle de taxa, conexão.
  - UDP: não-confiável, sem controle taxa, sem conexão.
- **Multiplexação:**
  - Segmentos de múltiplos sockets para a camada de rede.
  - Cabelhos auxiliam demultiplexação.
- **Demultiplexação:**
  - IPs, # de porta, identificam socket de destino.
  - UDP: apenas informações do destino.
  - TCP: quatro componentes.
- **UDP:**
  - Serviço básico: datagramas perdidos, fora de ordem.
  - Aplicações que **toleram perda, mas são sensíveis a taxa**.
  - DNS também.
- ***Checksum* do UDP:**
  - Verificação de erros.
  - Soma em complemento a 1.

Próxima Aula...

- Continuamos a discussão sobre a camada de transporte.
- Tópico **muito importante**: princípios de transmissão confiável de dados.
  - Como realizar a entrega confiável de dados sobre uma rede não-confiável?
  - Como lidar com perdas de pacotes?
  - Como lidar com pacotes corrompidos?