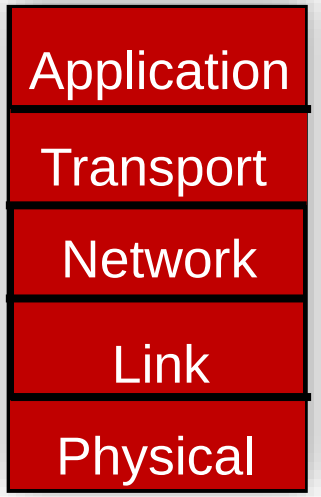


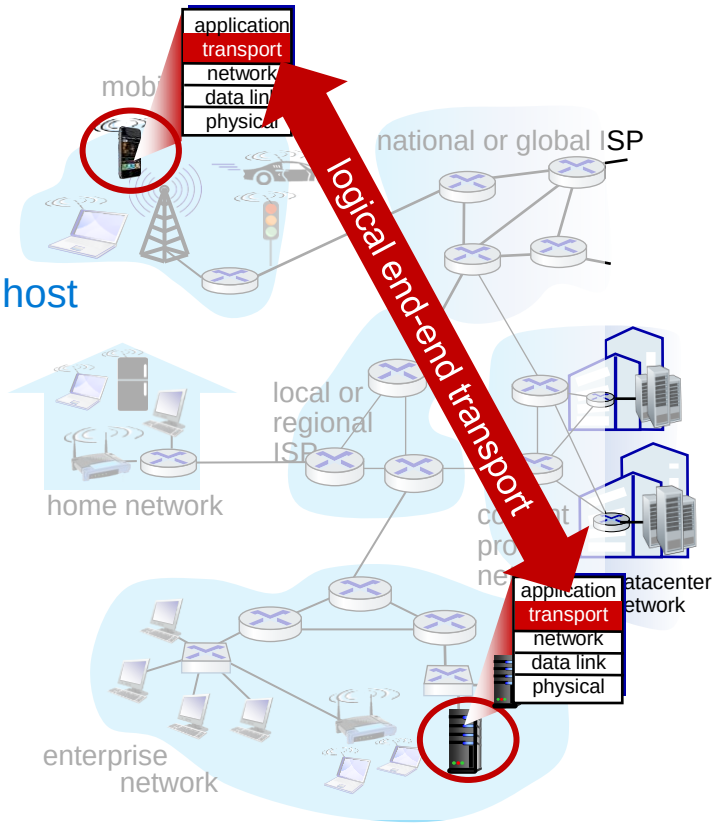
Computer Networks

Transport Layer - Introduction

Amitangshu Pal
Computer Science and Engineering
IIT Kanpur

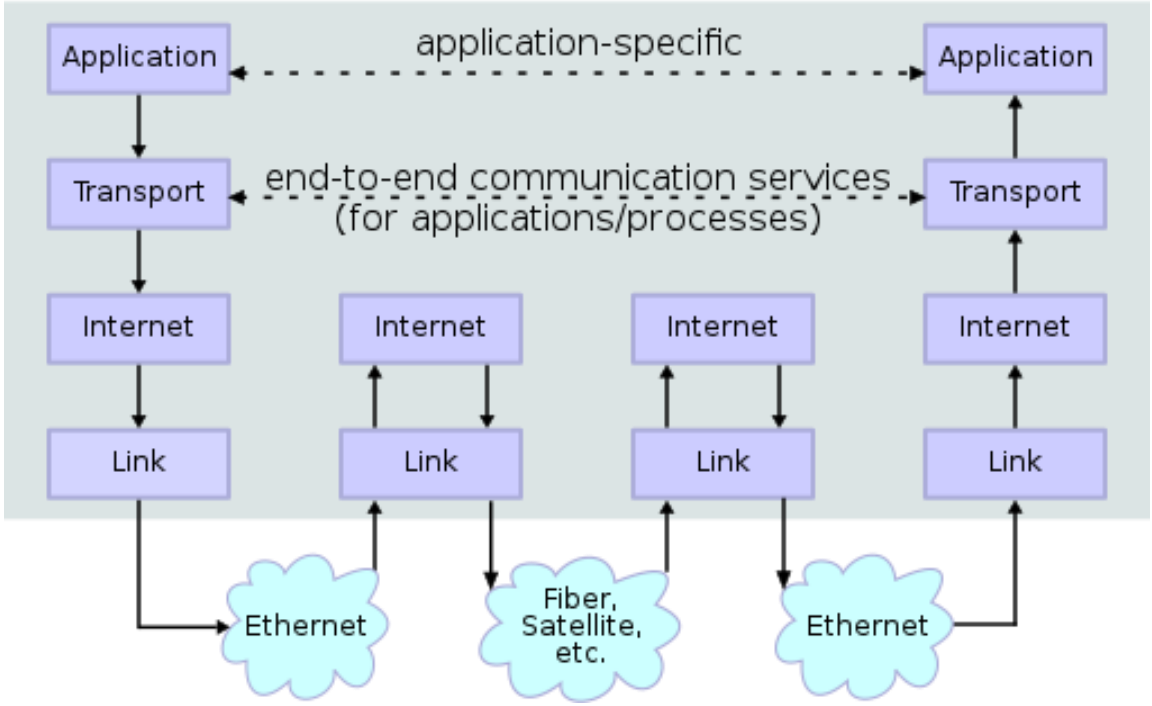
Transport Layer Services and Protocols

- Provide logical communication between application processes running on different hosts
- Transport protocols actions in end systems: ---> means run on host
  - Sender: breaks application messages into segments, passes to network layer
  - Receiver: reassembles segments into messages, passes to application layer
- Two transport protocols available to Internet applications
  - TCP, UDP



Transport vs Network Layer Services

- Network layer: host-to-host delivery
- Transport layer: logical communication between processes
  - relies on, enhances, network layer services

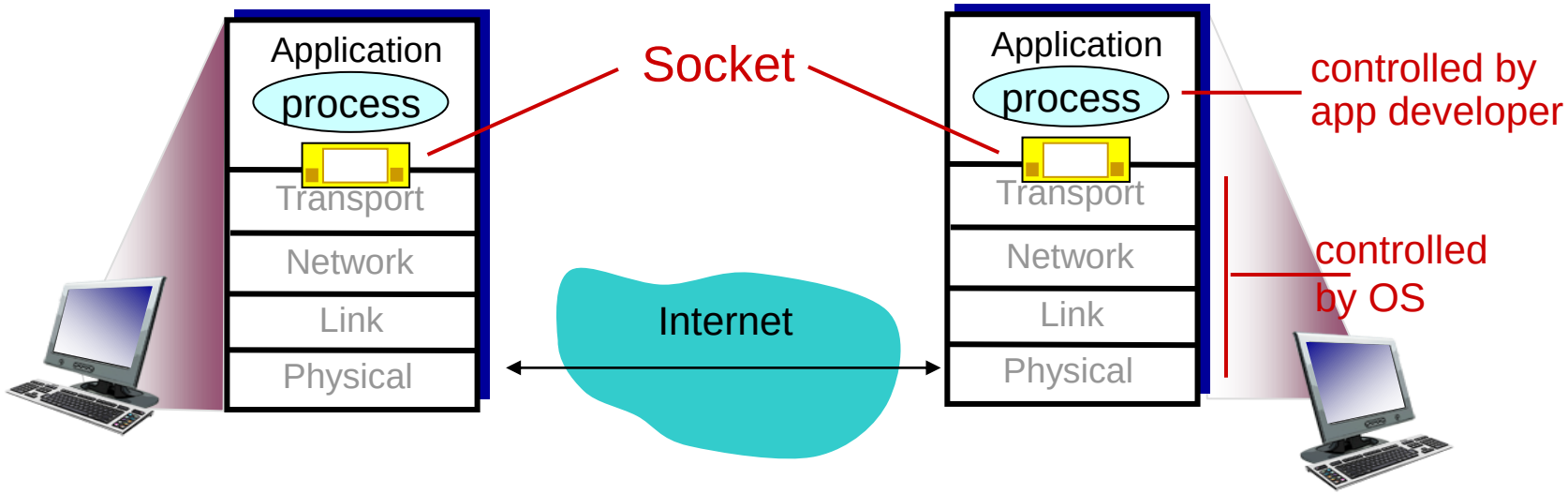


Src: https://commons.wikimedia.org/wiki/File:IP_stack_connections_(corrected).svg

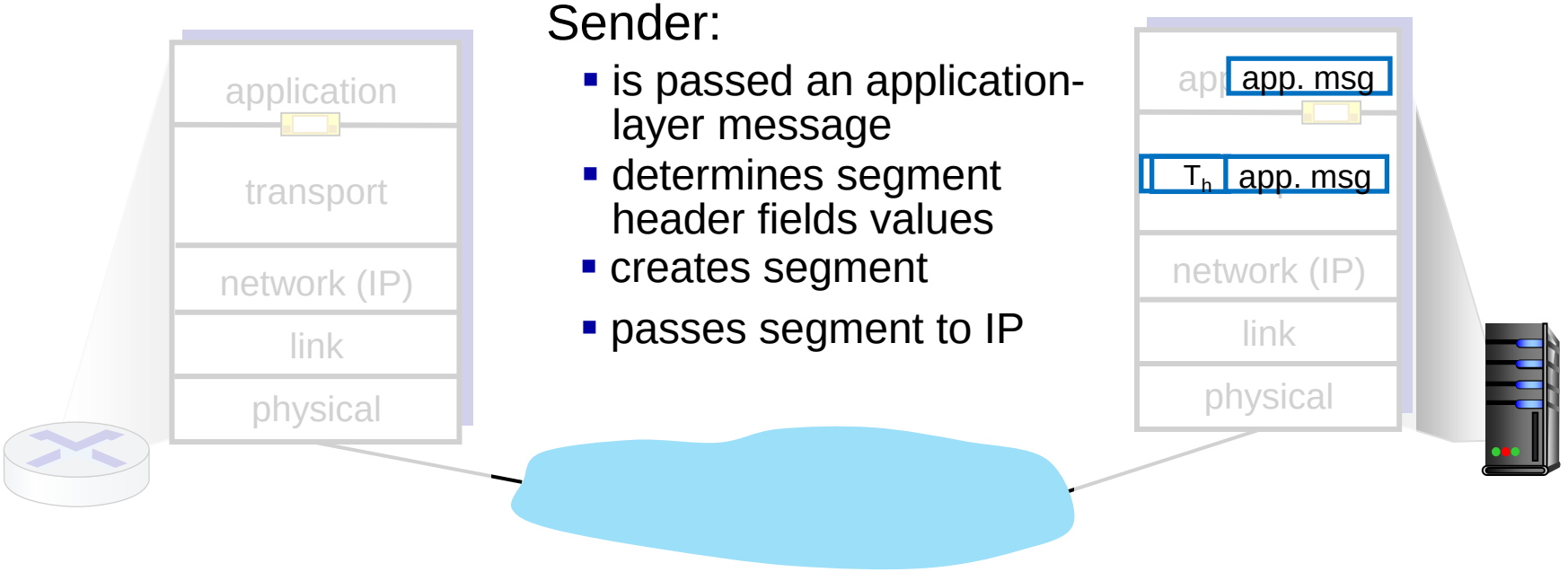
Sockets

Socket: door between application process and end-end-transport protocol

- An application process can send/receive messages to/from another application process via a socket



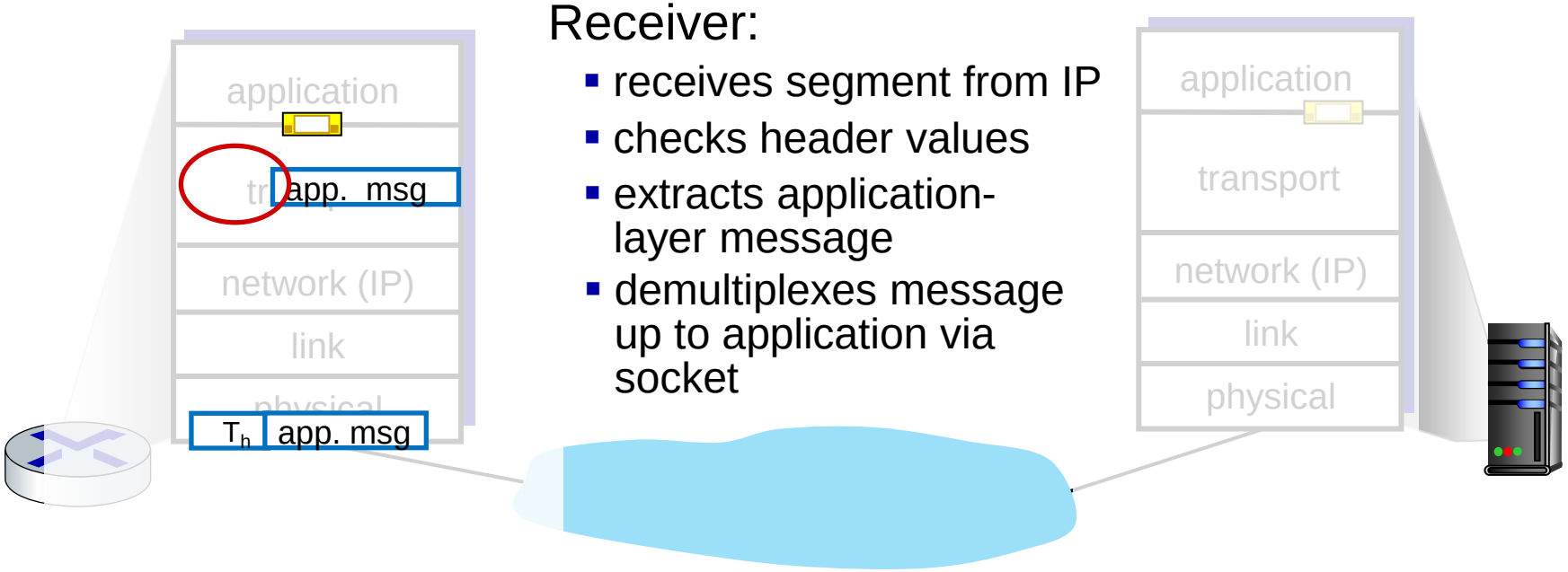
Transport Layer Actions



Sender:

- is passed an application-layer message
- determines segment header fields values
- creates segment
- passes segment to IP

Transport Layer Actions

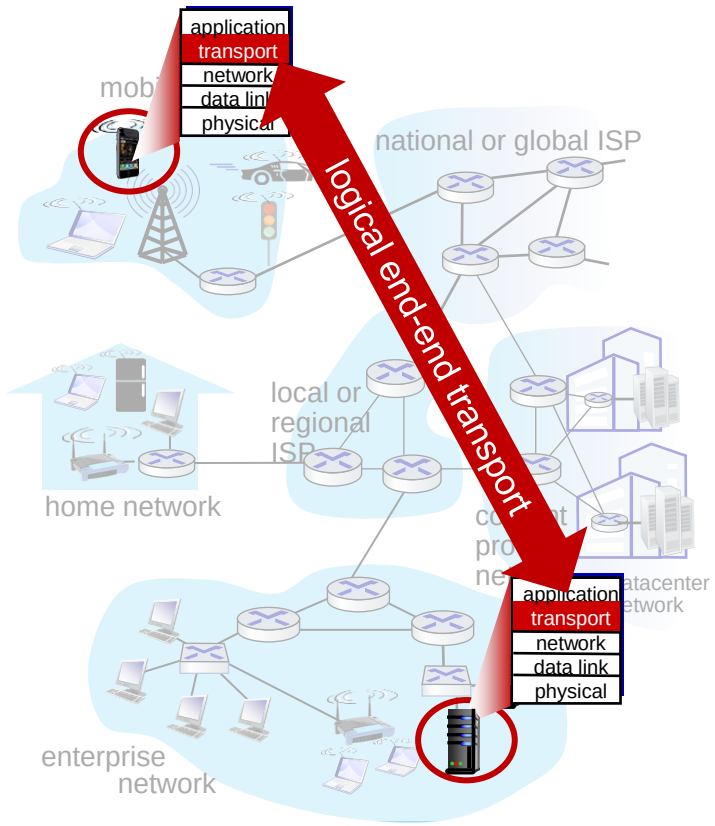


Receiver:

- receives segment from IP
- checks header values
- extracts application-layer message
- demultiplexes message up to application via socket

Two Principal Internet Transport Protocols

- **TCP:** Transmission Control Protocol
  - reliable, in-order delivery
  - congestion control
  - flow control
  - connection setup
- **UDP:** User Datagram Protocol
  - unreliable, unordered delivery
  - no-frills extension of “best-effort” IP



Multiplexing and Demultiplexing

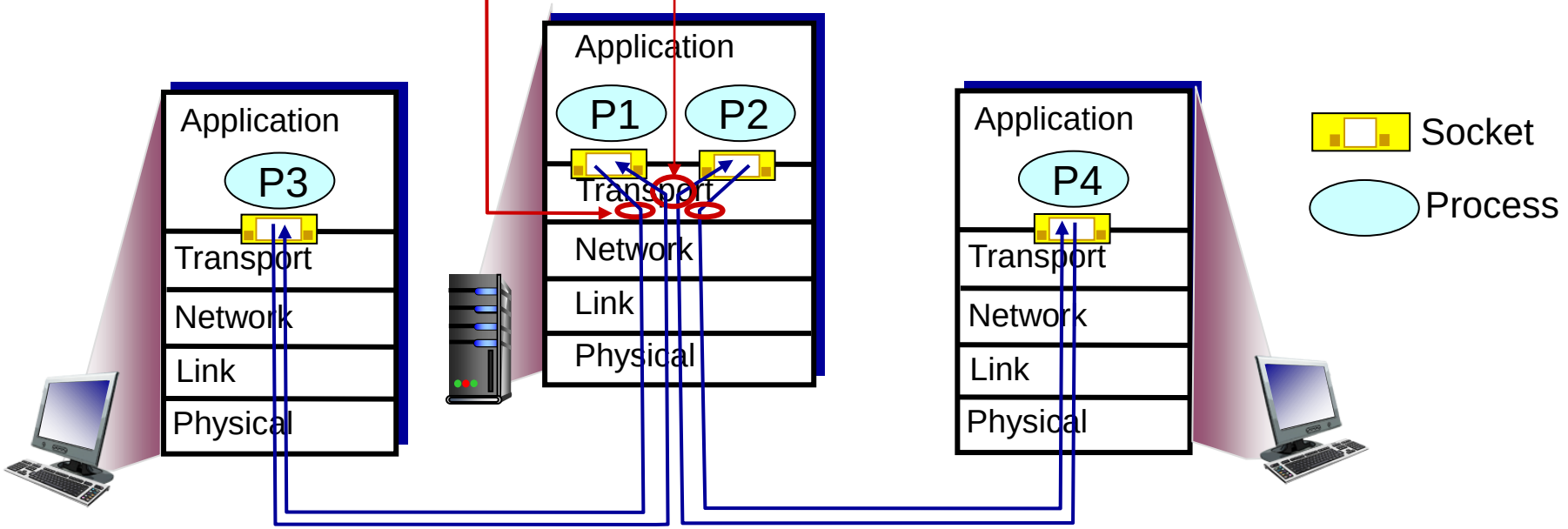
Multiplexing/demultiplexing

Multiplexing at sender:

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

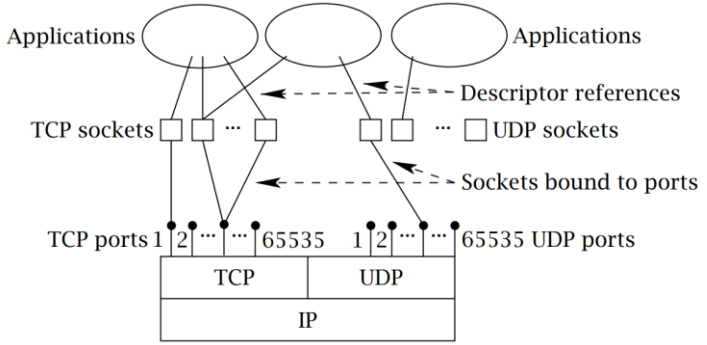
Demultiplexing at receiver:

Use header info to deliver received segments to correct socket

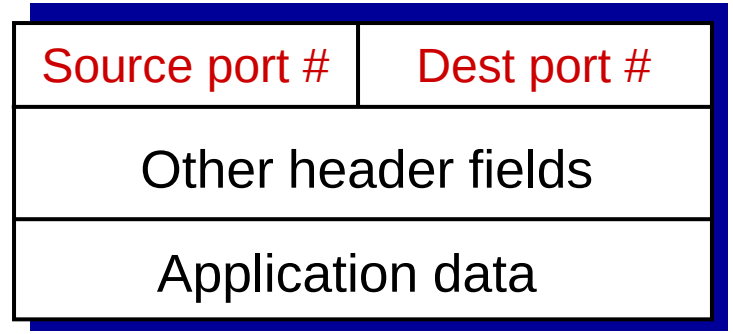


How Demultiplexing Works

- Host receives IP datagrams
  - Each datagram has source IP address, destination IP address
  - Each datagram carries one transport-layer segment
  - Each segment has source, destination port number
- Host uses IP addresses & port numbers to direct segment to appropriate socket



Src: https://linuxwheel.com/chapter-5-fundamentals-of-tcp-ip-transport-and-application/



TCP/UDP segment format

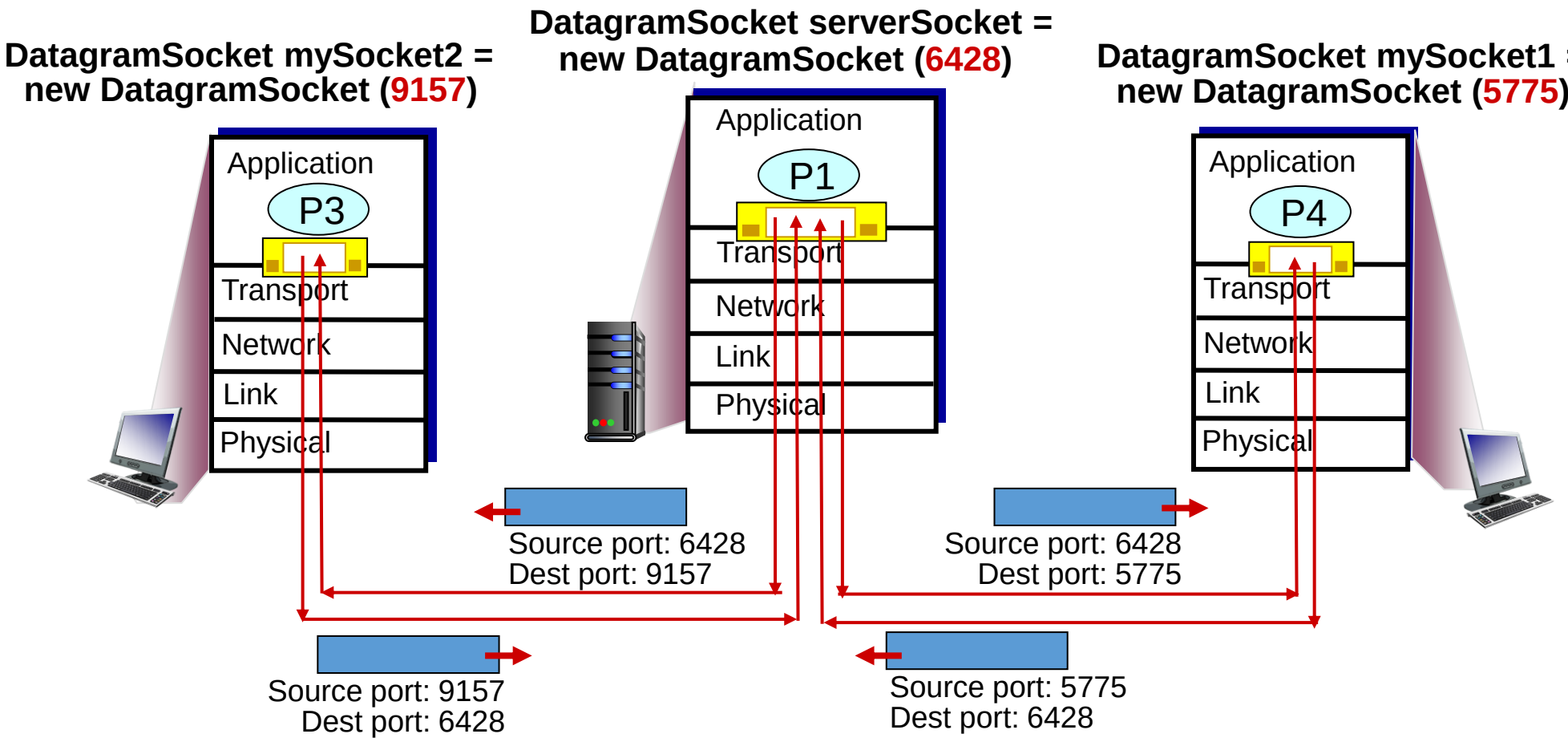
Connectionless Demultiplexing

- When creating datagram to send into UDP socket, must specify
  - Destination IP address
  - Destination port #

- When host receives UDP segment:
  - checks destination port # in segment
  - directs UDP segment to socket with that port #

IP datagrams with same dest. port #, but different source IP addresses and/or source port numbers will be directed to same socket at dest

Connectionless Demultiplexing

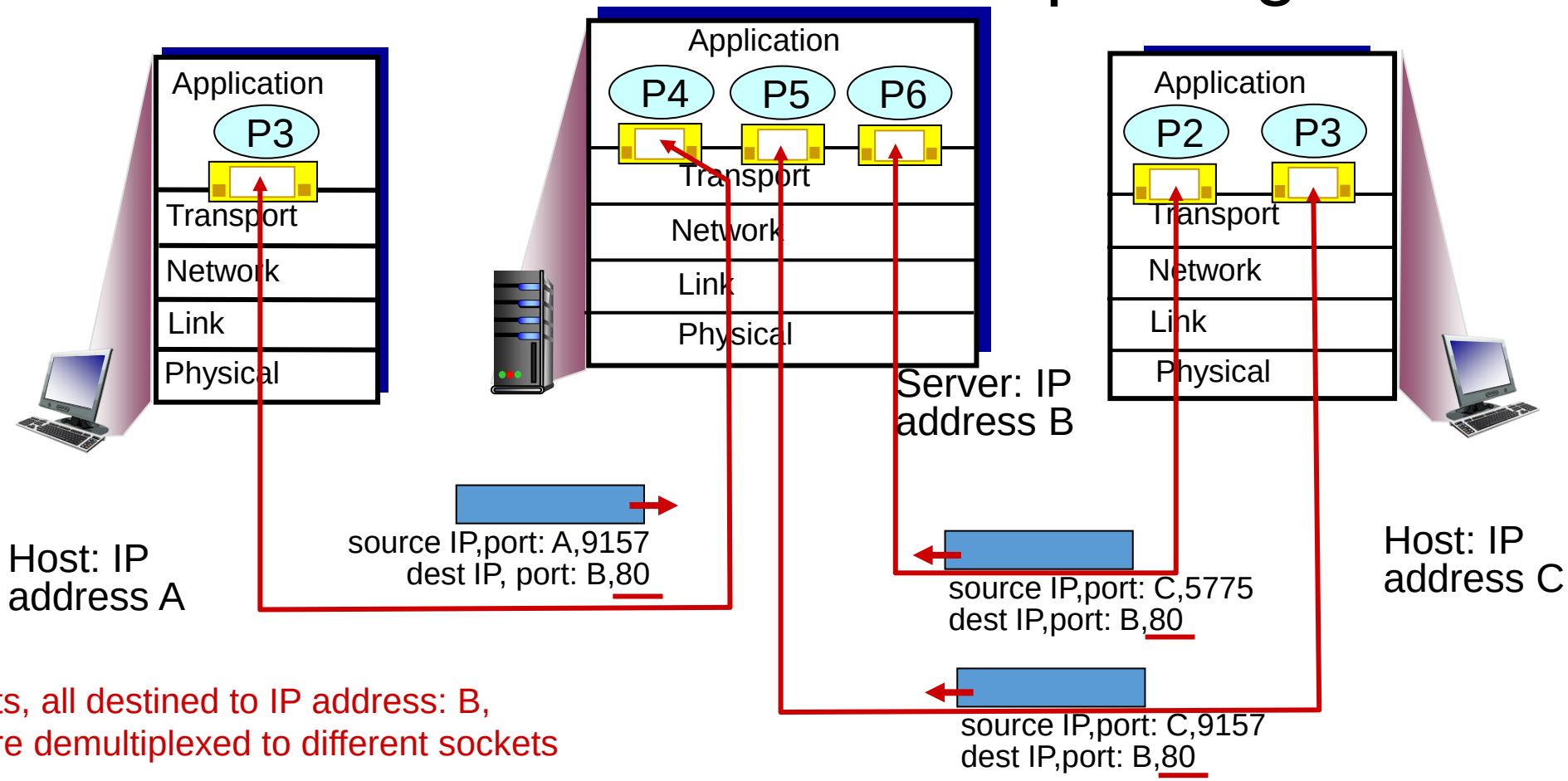


Connection-Oriented Demultiplexing

- TCP socket identified by 4-tuple:
  - Source IP address
  - Source port number
  - Dest IP address
  - Dest port number
- Demux: receiver uses all four values to direct segment to appropriate socket

- Server host may support many simultaneous TCP sockets:
  - Each socket identified by its own 4-tuple
- Web servers have different sockets for each connecting client

Connection-Oriented Demultiplexing



Three segments, all destined to IP address: B, dest port: 80 are demultiplexed to different sockets

Summary

- Transport layer protocols:
  - Multiplexing, demultiplexing

- **UDP:** Demultiplexing using destination IP address and port number
- **TCP:** Demultiplexing using 4-tuple: source and destination IP addresses, and port numbers