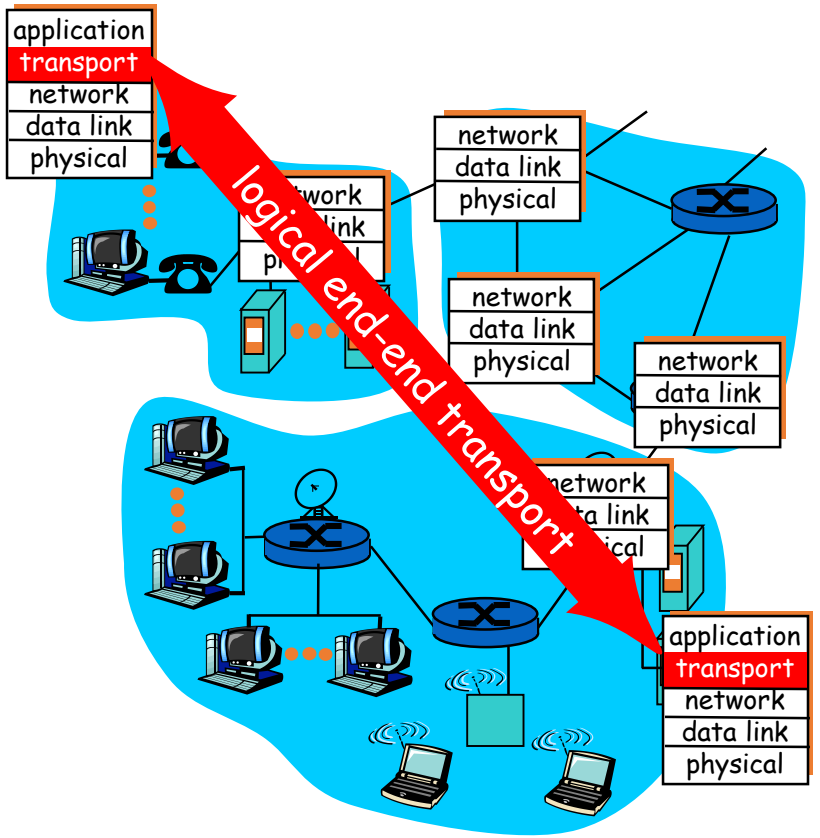
Transport Services and Protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Transport vs. Application and Network Layer

- *application layer:* application processes and message exchange
- *network layer:* logical communication between hosts
- *transport layer:* logical communication support for app processes
  - relies on, enhances, network layer services

Household analogy:

12 kids sending letters to 12 kids

- processes = kids
- app messages = letters in envelopes
- hosts = houses
- transport protocol = Ann and Bill
- network-layer protocol = postal service

End to End Issues

- Transport services built on top of (potentially) unreliable network service
  - packets can be corrupted or lost
  - Packets can be delayed or arrive "out of order"
- Do we detect and/or recover errors for apps?
  - Error Control & Reliable Data Transfer
- Do we provide "in-order" delivery of packets?
  - Connection Management & Reliable Data Transfer
- Potentially different capacity at destination, and potentially different network capacity
  - Flow and Congestion Control

Internet Transport Protocols

TCP service:

- *connection-oriented:* setup required between client, server
- *reliable transport* between sender and receiver
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded

UDP service:

- unreliable data transfer between sender and receiver
- does not provide: connection setup, reliability, flow control, congestion control

Both provide *logical communication* between app processes running on different hosts!

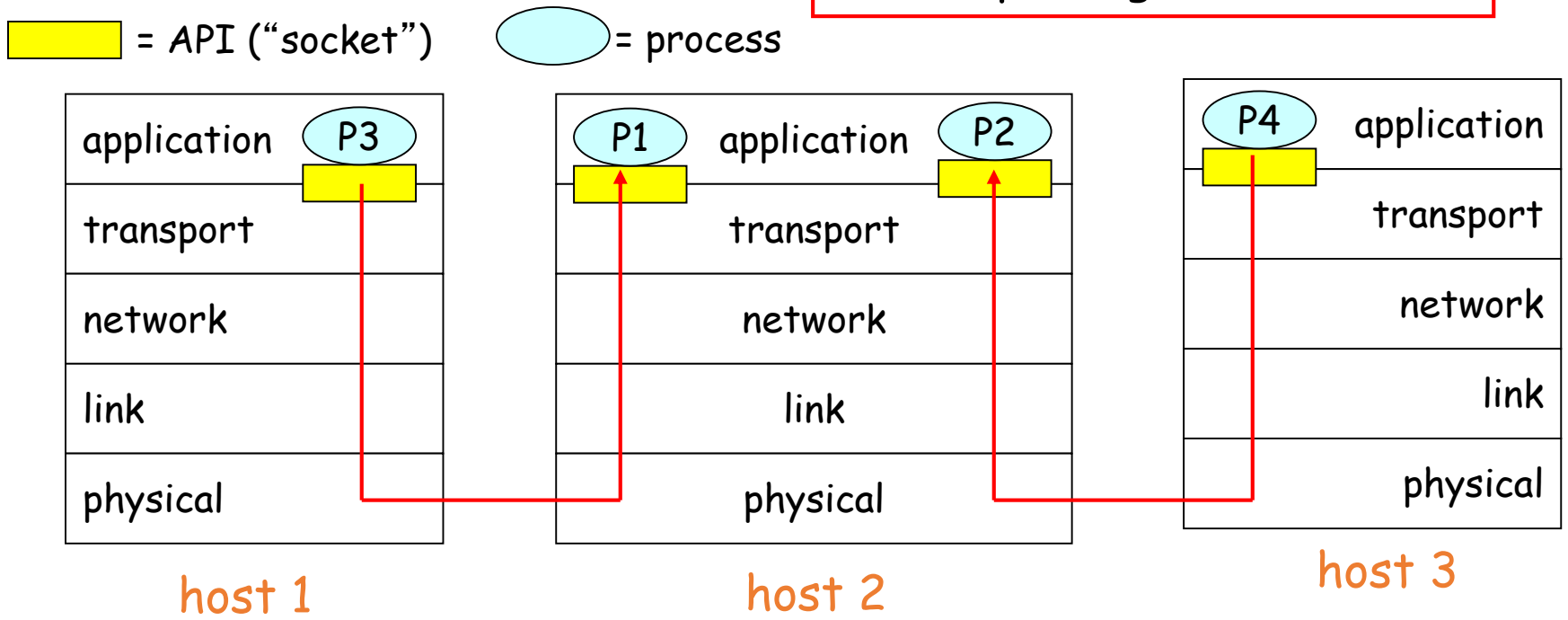
Multiplexing/Demultiplexing

Demultiplexing at rcv host:

delivering received segments to correct application process

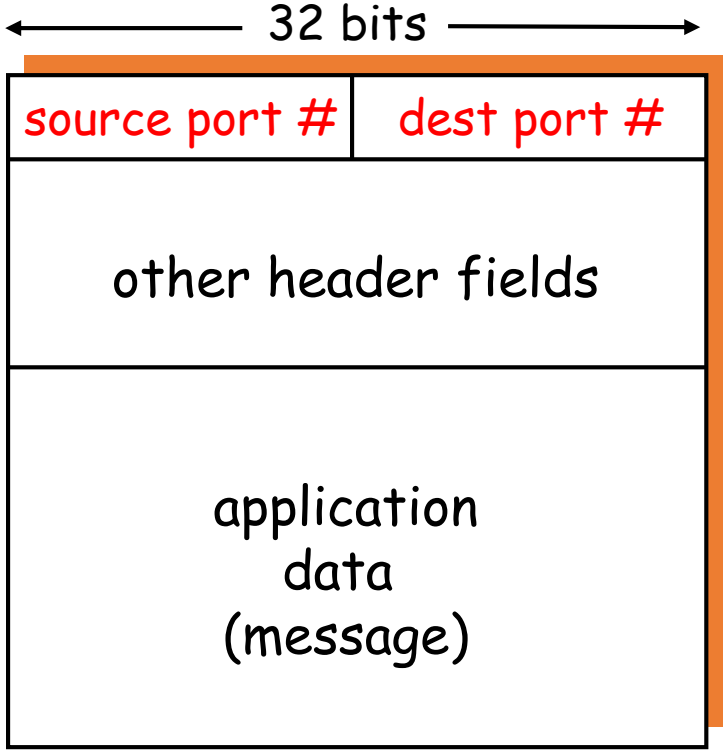
Multiplexing at send host:

gathering data from multiple app processes, enveloping data with header (later used for demultiplexing)



How Demultiplexing Works

- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries 1 transport-layer segment
  - each segment has source, destination port number (recall: well-known port numbers for specific applications)
- host uses IP addresses & port numbers to direct segment to appropriate app process (identified by "socket')



TCP/UDP segment format