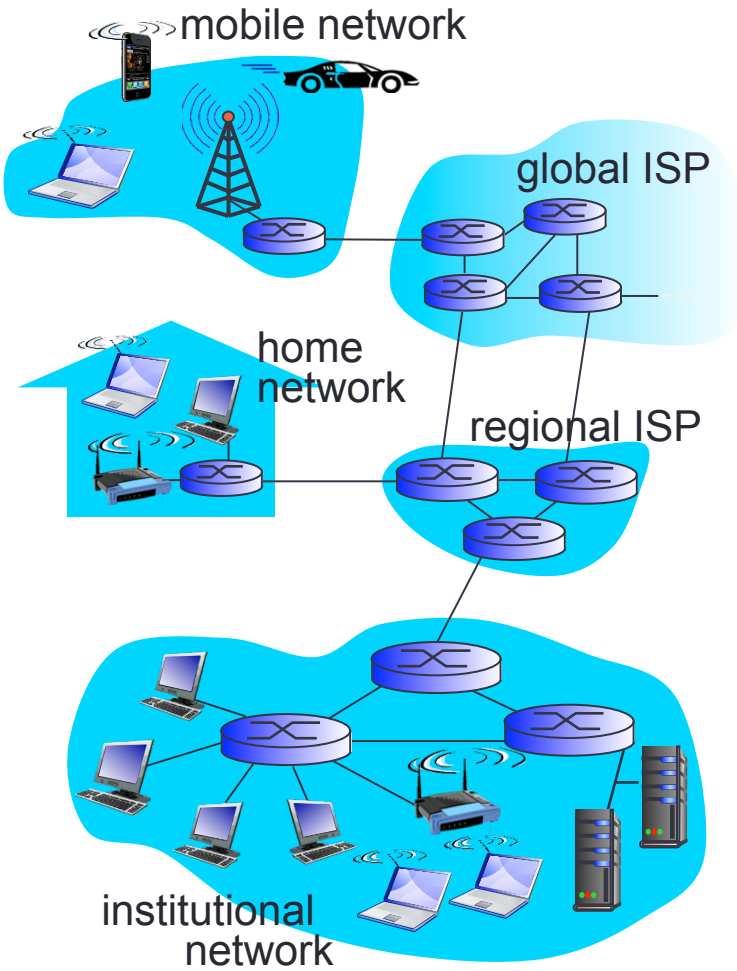
RELATED CONCEPTS IN COMPUTER NETWORKS

Content

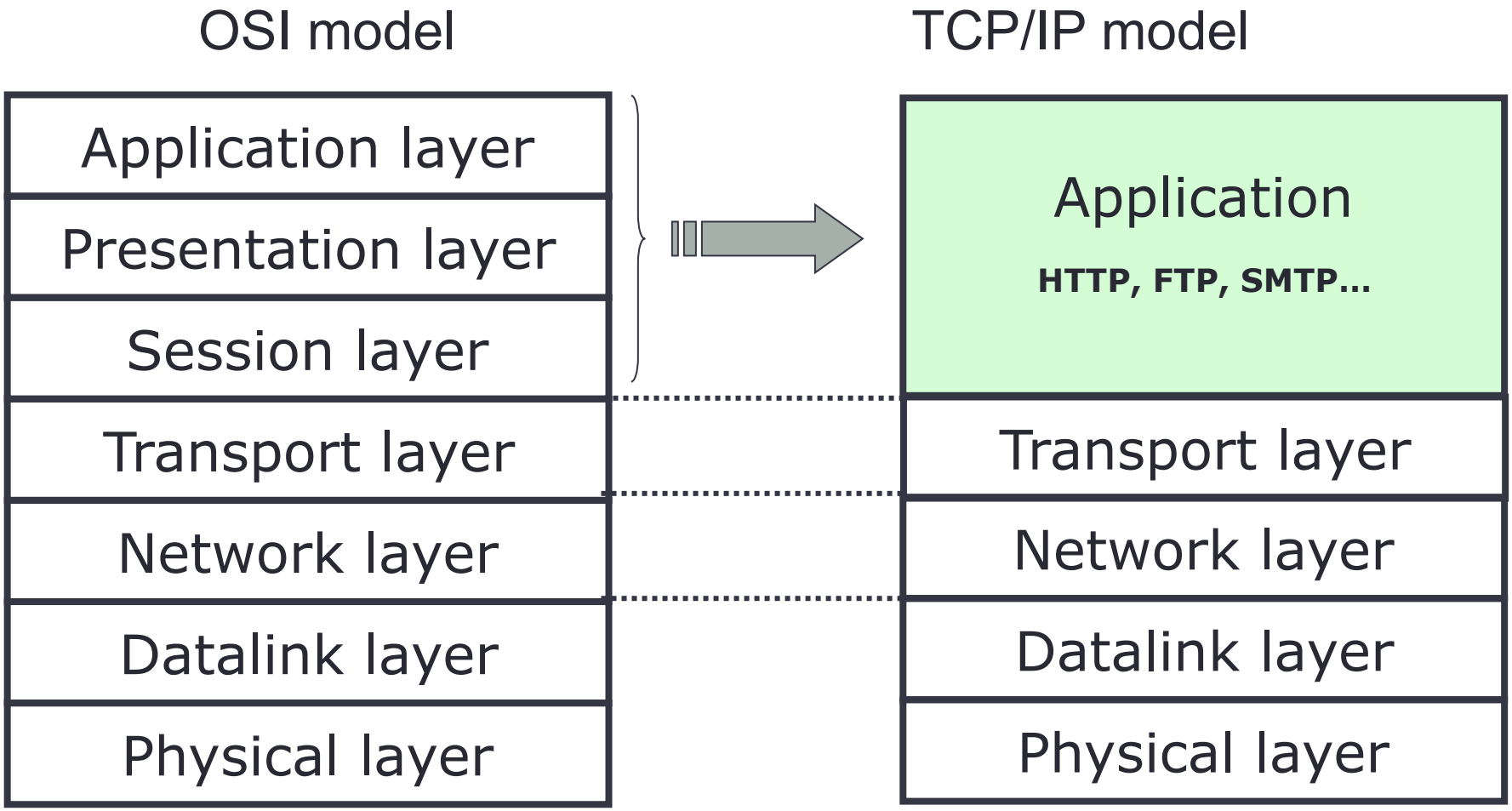
- Computer Networks
- OSI and TCP/IP models
- IP
- Transport layer, TCP, UDP
- Network application models
- Socket concept

Computer Networks

- Inter-connection of nodes by transport medium following some architecture:
  - Using a topology: bus, star, ring, hybrid…
  - Using some communication protocols

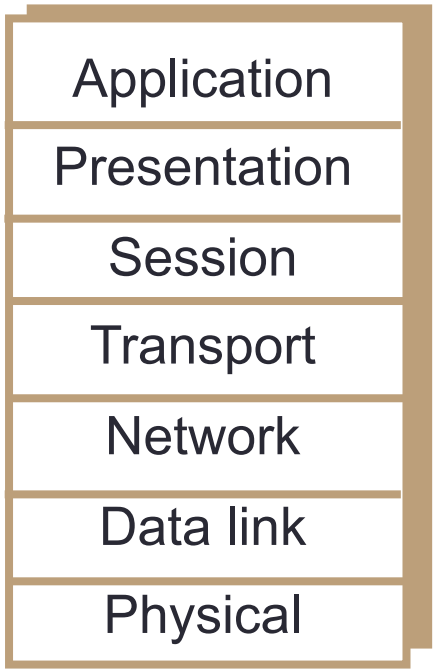


OSI and TCP/IP models



OSI and TCP/IP models

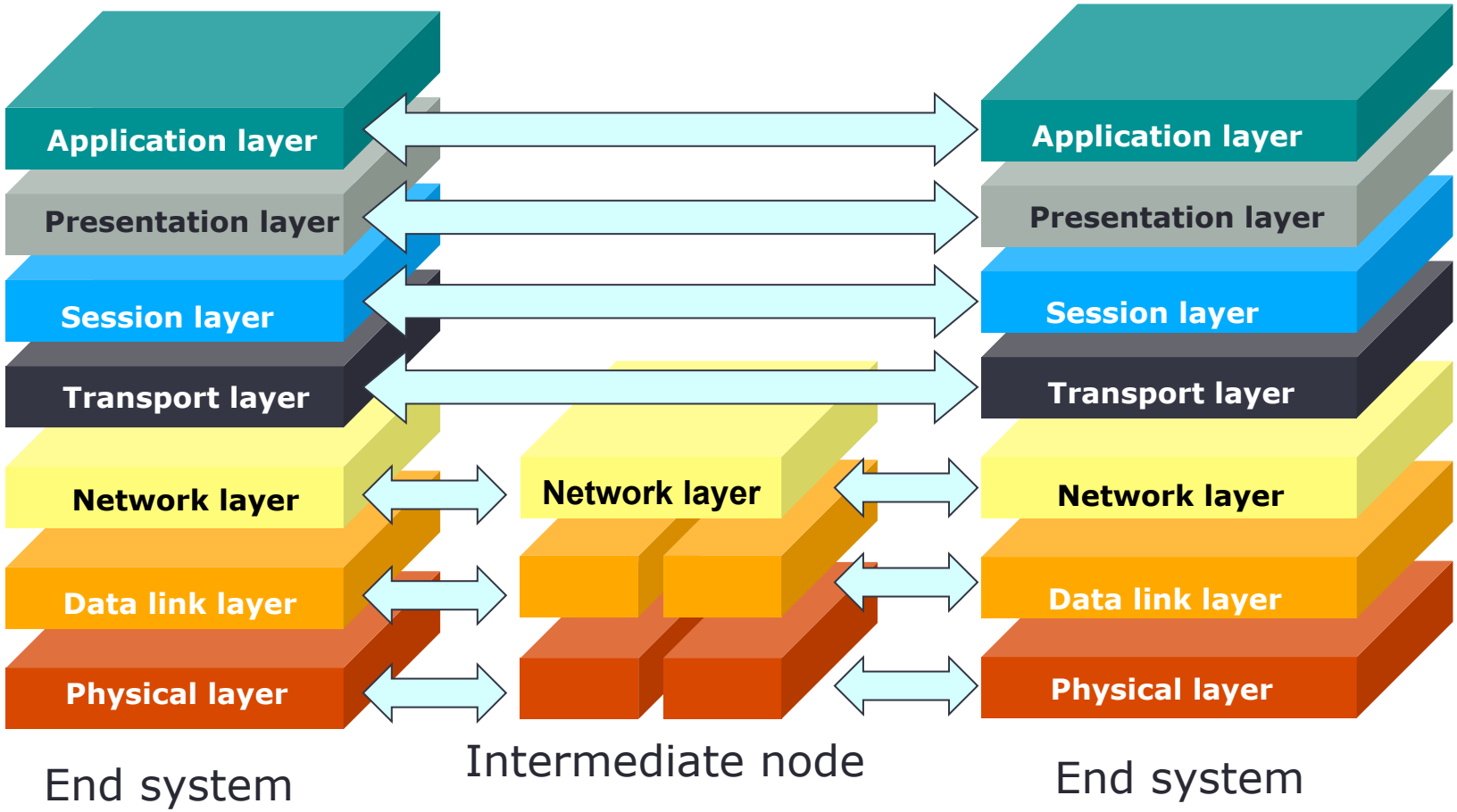
- Application layer: defines communication between different parts of the same application
  - Presentation layer: application data representation, data encryption, compression, conversion…
  - Session layer: manages sessions, synchronization, recovery of data transmission process
- Transport layer: Transmits data between applications
- Network layer: Transmits data between distance network elements: Taking care of routing and forwarding data
- Data link layer: Transmits data between adjacent network elements.
- Physical layer: Transmits bits on the medium. Converting bits to physical form appropriate to the medium.



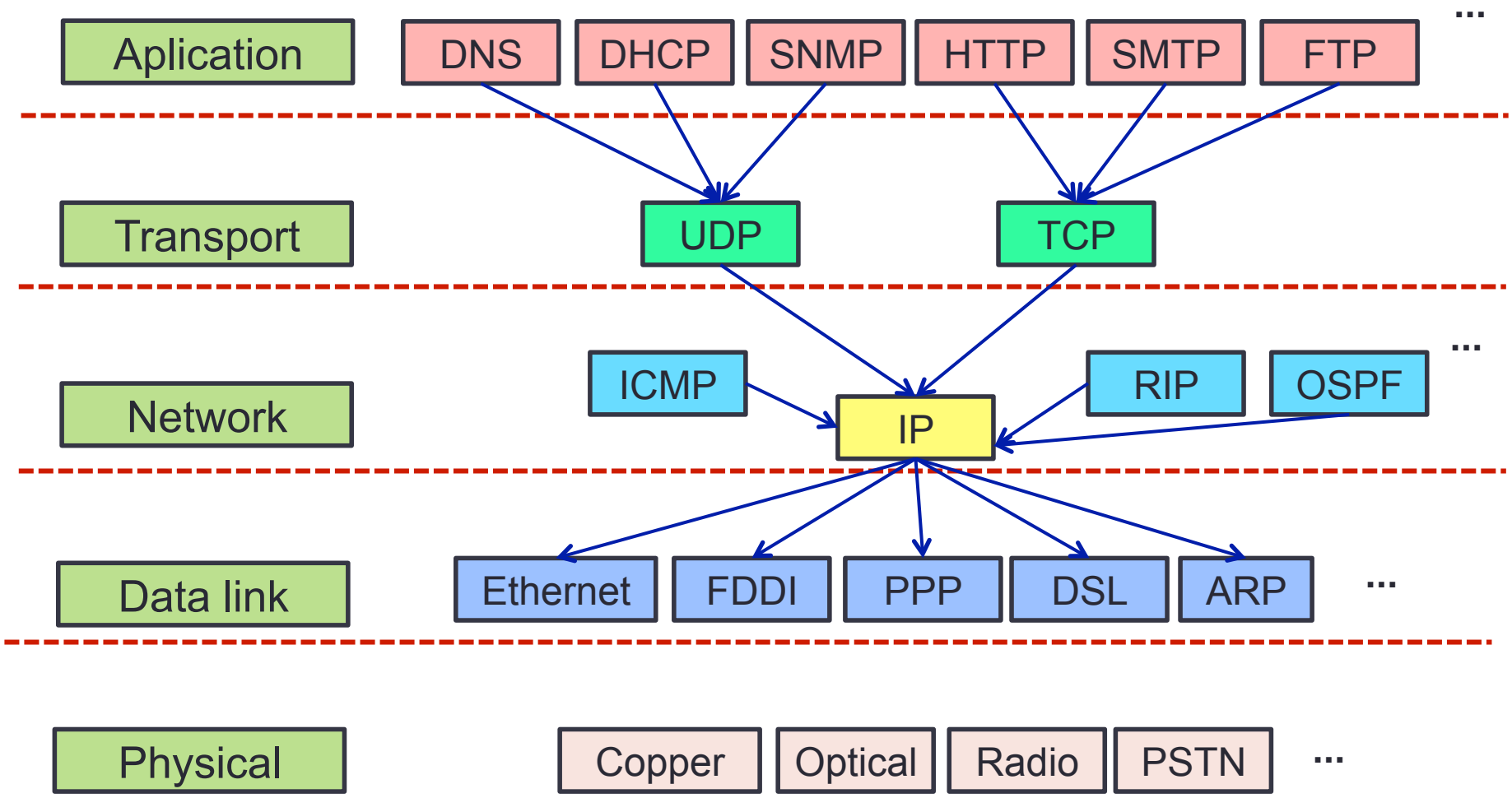
OSI and TCP/IP models

- OSI model: reference model
- TCP model: Internet model
  - Transport layer: TCP/UDP
  - Network layer: IP + routing protocols.

OSI and TCP/IP models



Internet protocols mapping on TCP/IP



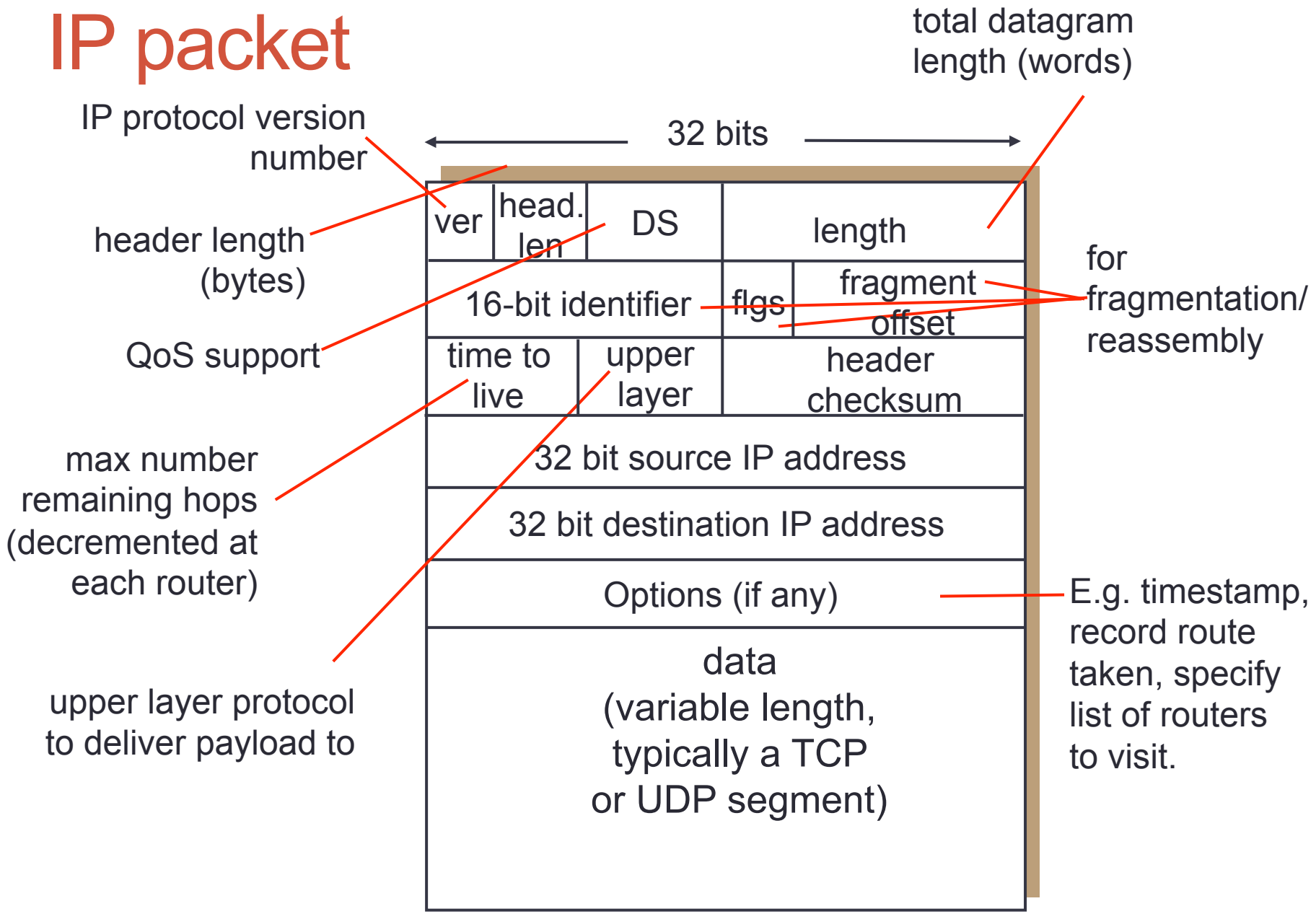
OSI and TCP/IP models

- Layering Makes it Easier
- Application programmer
  - Doesn’t need to send IP packets
  - Doesn’t need to send Ethernet frames
  - Doesn’t need to know how TCP implements reliability
- Only need a way to pass the data down
  - Socket is the API to access transport layer functions

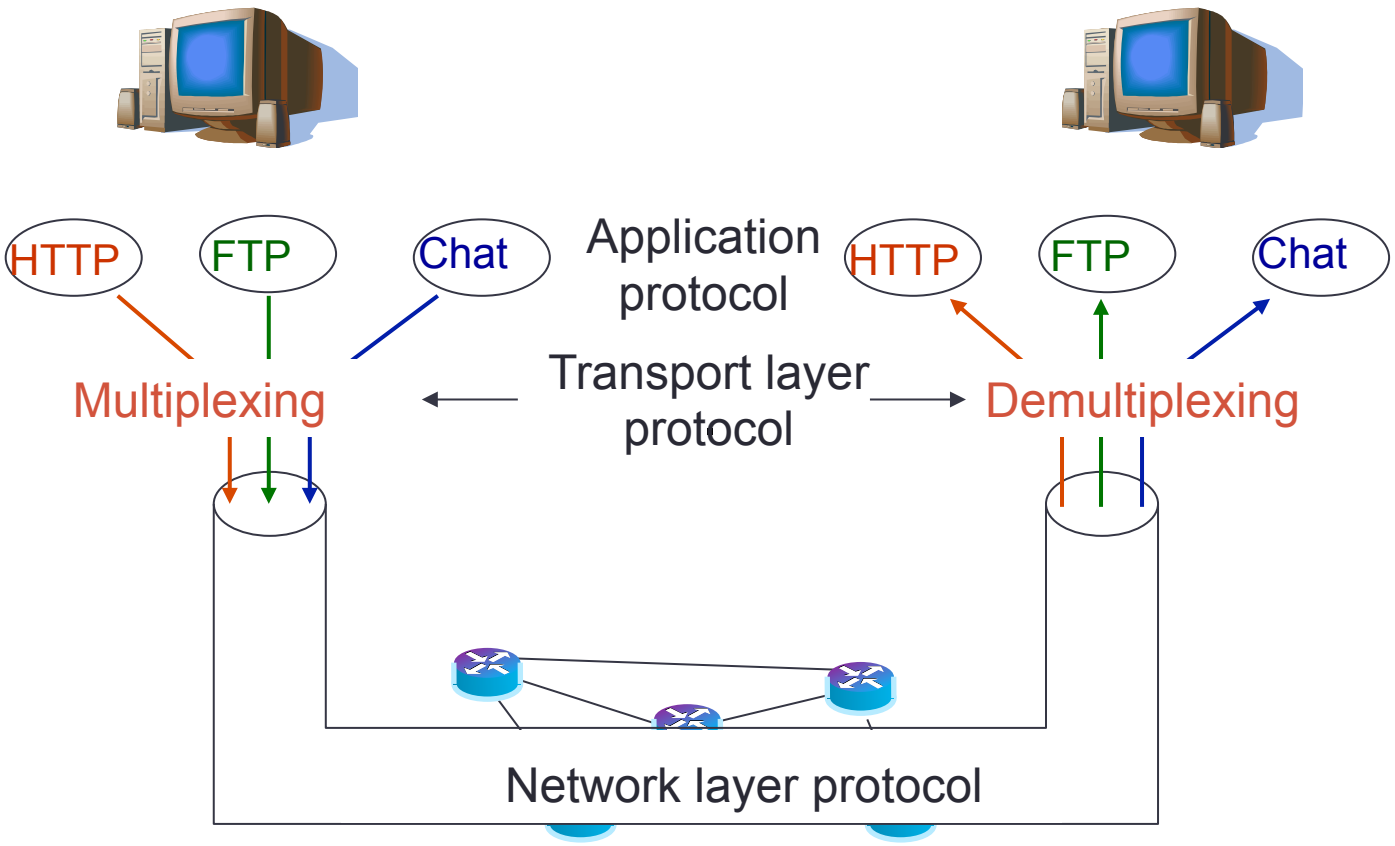
IP

- IP: Internet Protocol
  - Forward data packet between distance network nodes (routers or hosts)
  - Using routing table built by routing protocols such as OSPF, RIP …
- IP address
  - Is assigned to each network interface
  - IP v4: 32 bits
    - 133.113.215.10
  - IP v6: 128 bits
    - 2001:200:0:8803::53
- A host may have a domain name
  - Conversion IP <-> domain name: DNS
  - Ex: soict.hust.edu.vn <--> 202.191.56.65

IP packet

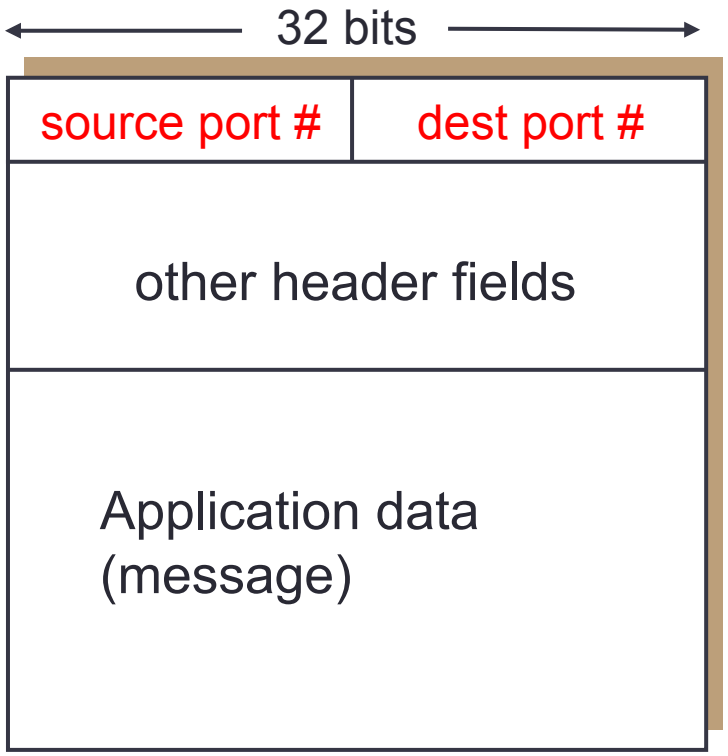

total datagram length (words)
IP protocol version number
32 bits
header length (bytes)
QoS support
max number remaining hops (decremented at each router)
upper layer protocol to deliver payload to
ver
head. len
DS
length
16-bit identifier
flgs
fragment offset
for fragmentation/ reassembly
time to live
upper layer
header checksum
32 bit source IP address
32 bit destination IP address
Options (if any)
E.g. timestamp, record route taken, specify list of routers to visit.
data (variable length, typically a TCP or UDP segment)

Transport layer: Mux/Demux



Transport layer: Mux/Demux

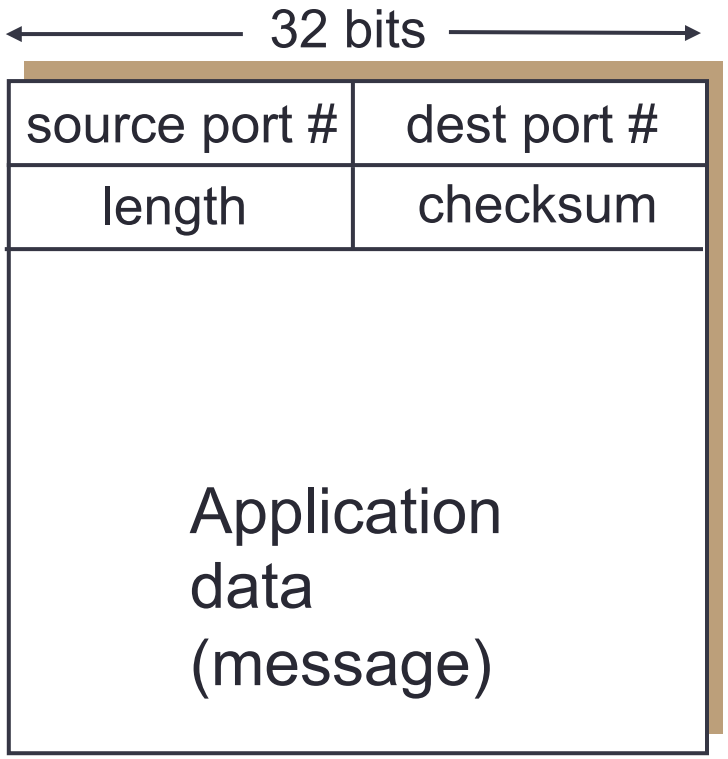
- How data from different applications between two hosts can be delivered to right application?
  - Each application process is assigned a transport port (16 bits)
  - Application sends data to the transport layer through the port.
- Socket:
  - Application access point for application
  - It is a combination of (Address IP, transportnport)



TCP/UDP segment format

Transport layer: UDP

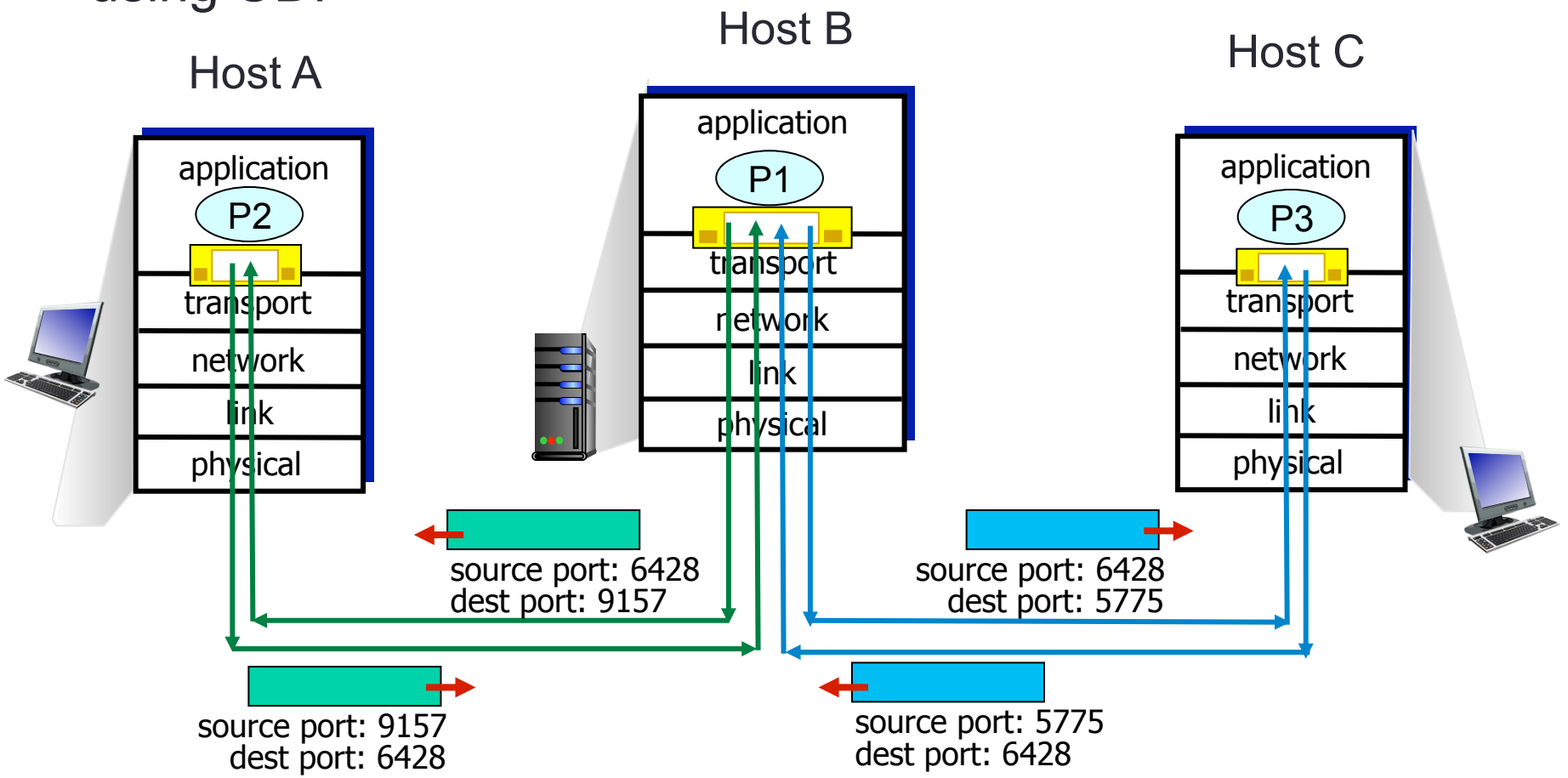
- MUX/DeMUX
- Best effort
- Connectionless
- Send independent datagrams
- Drop error datagrams
- No congestion control



UDP mux/demux

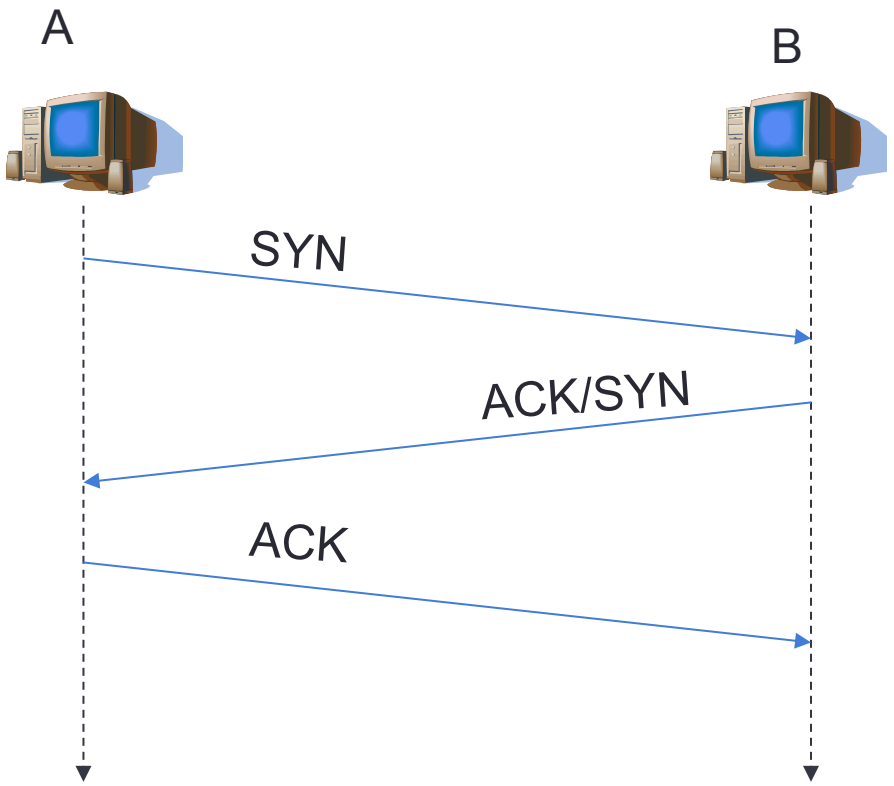
Each process uses a port to communicate with other process.

One process can communicate with several others using UDP

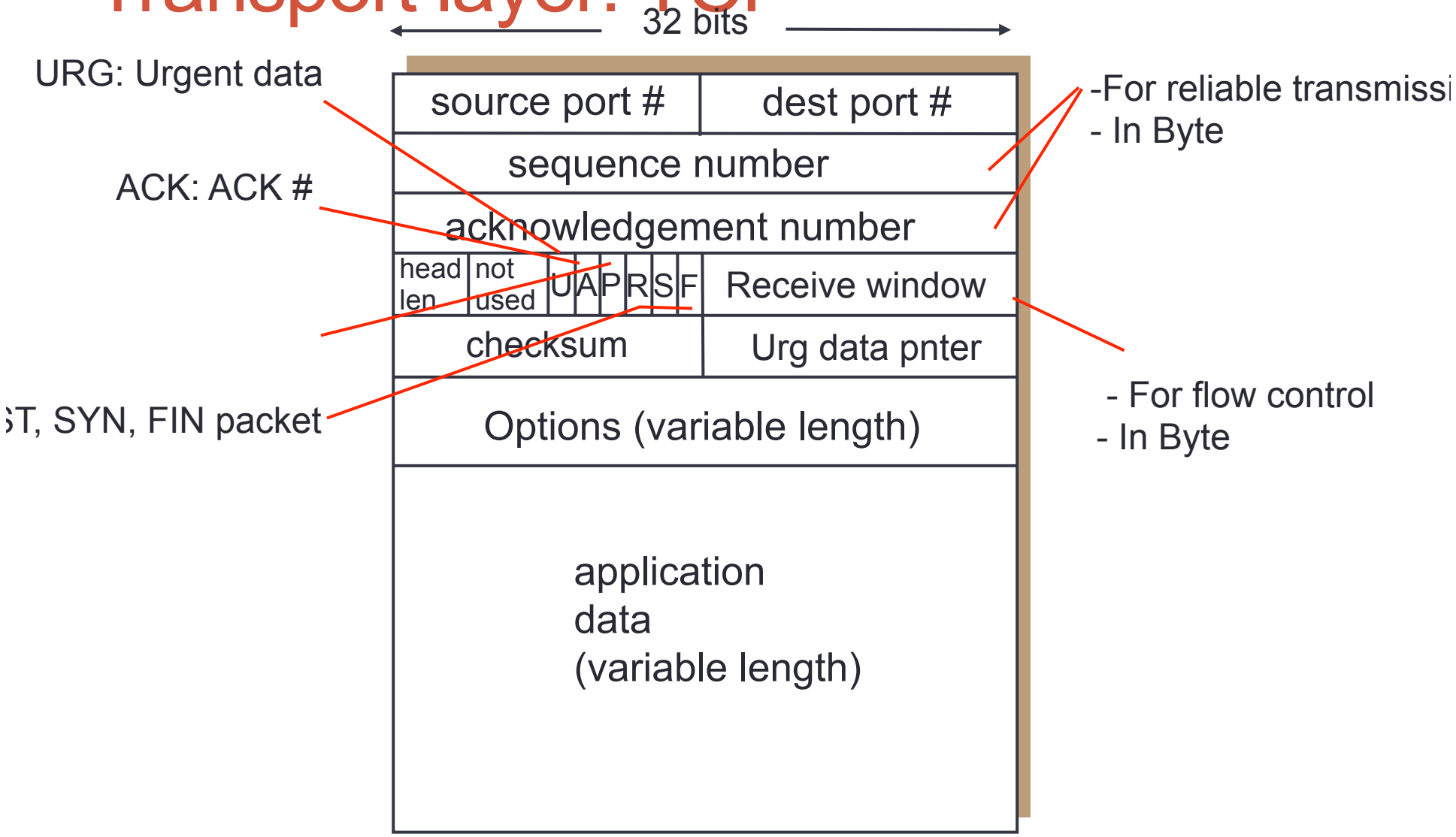


Transport layer: TCP

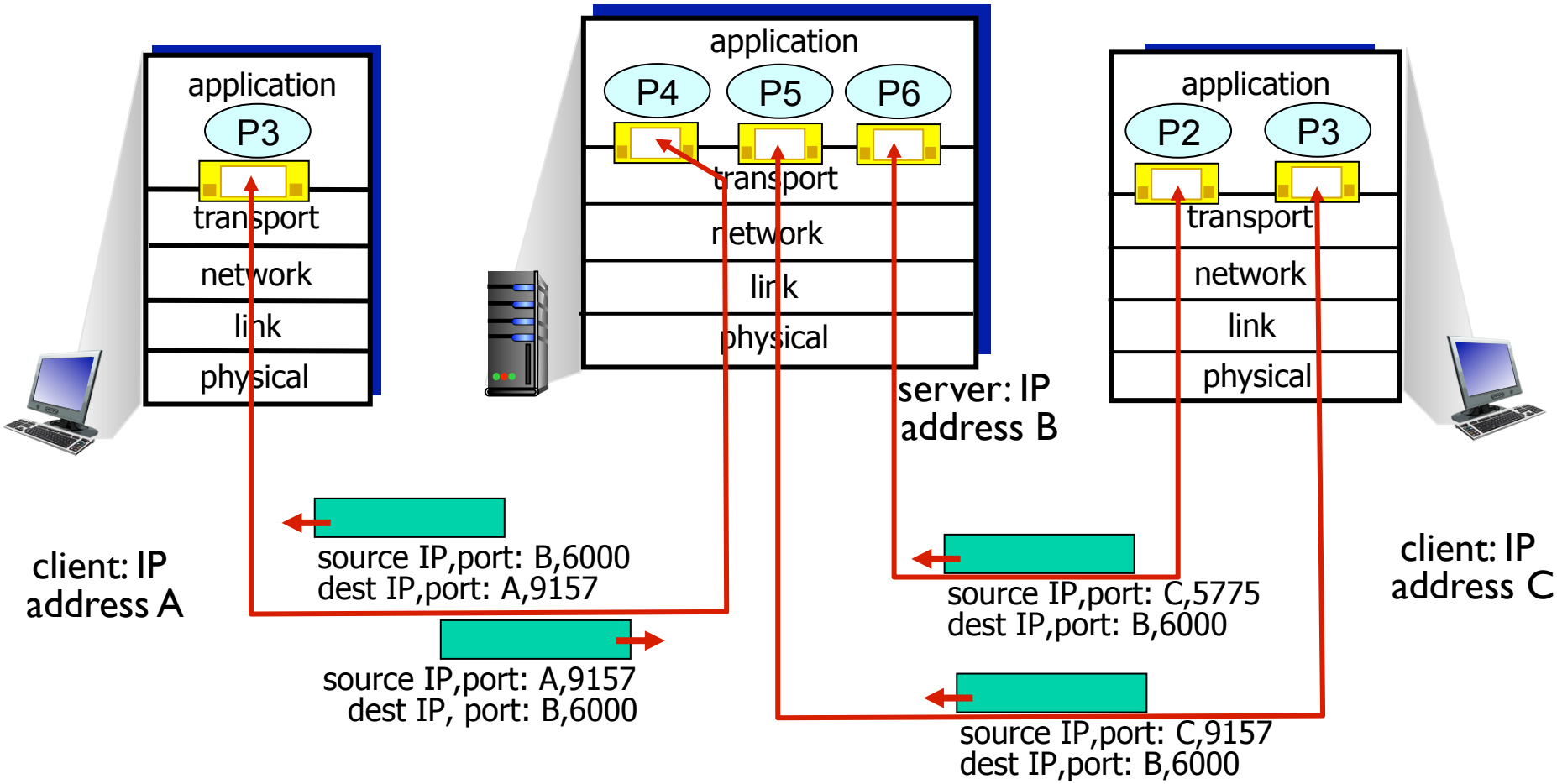
- Connection oriented protocol
  - 3-step connection opening
- Reliable protocol
  - Re-transmission on error
- Flow control
- Congestion control



Transport layer: TCP

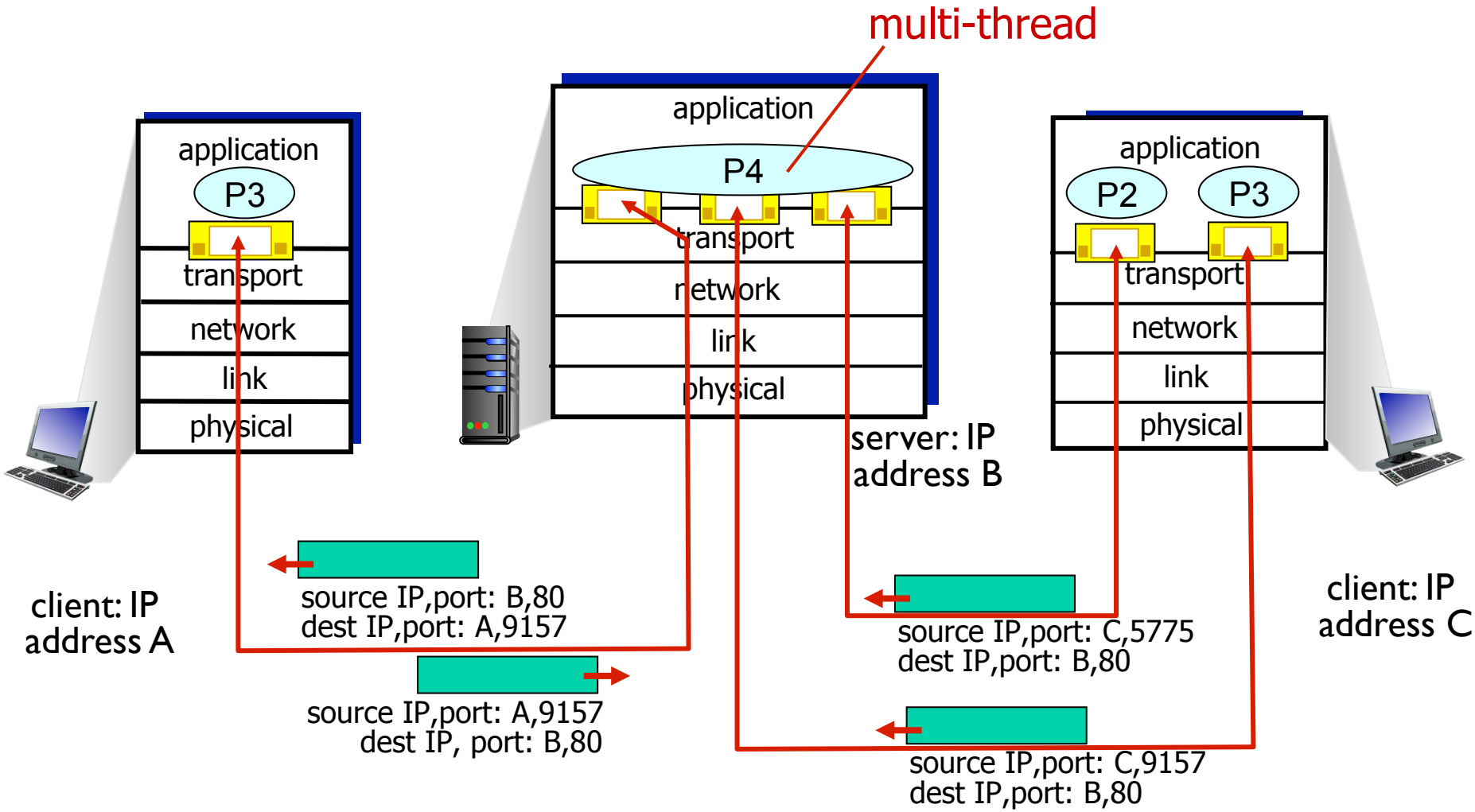


TCP mux/demux

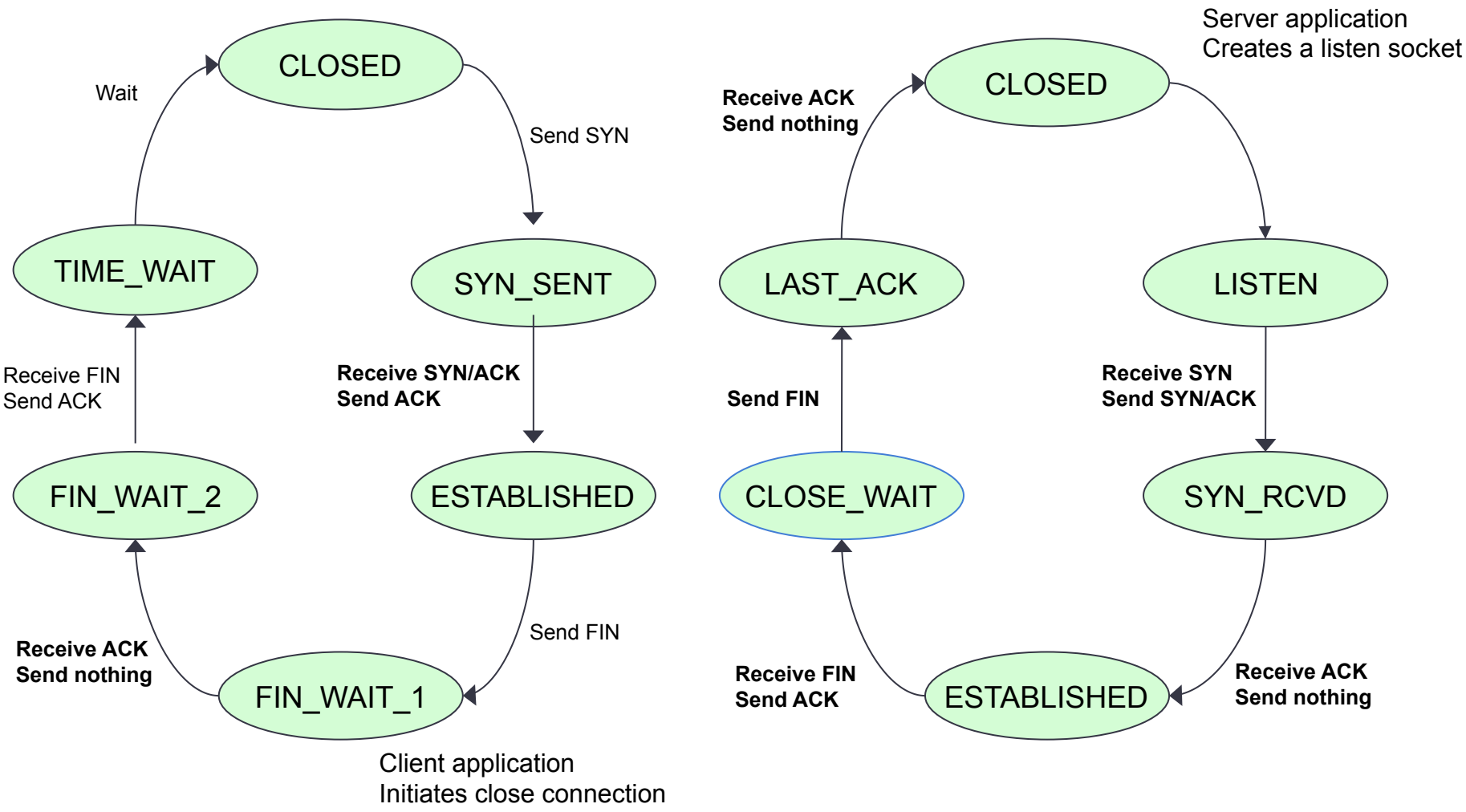


One process communicates with another process by using a channel (connection) TCP

TCP mux/demux



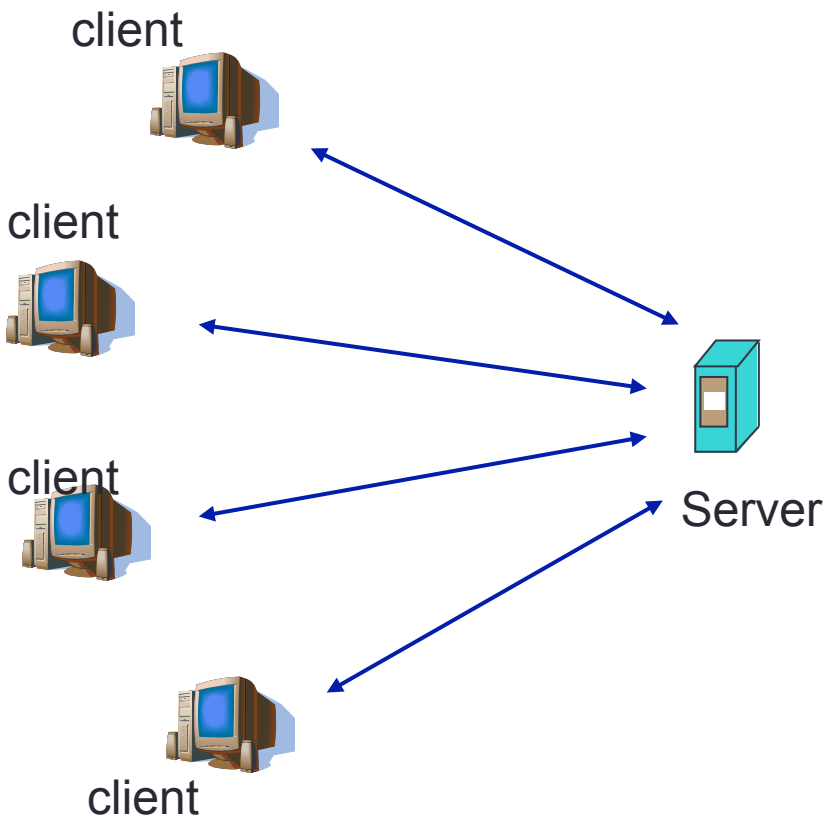
Life cycle of TCP connection



Network application models

- Client/Server
- Peer-to-peer
- Hybrid

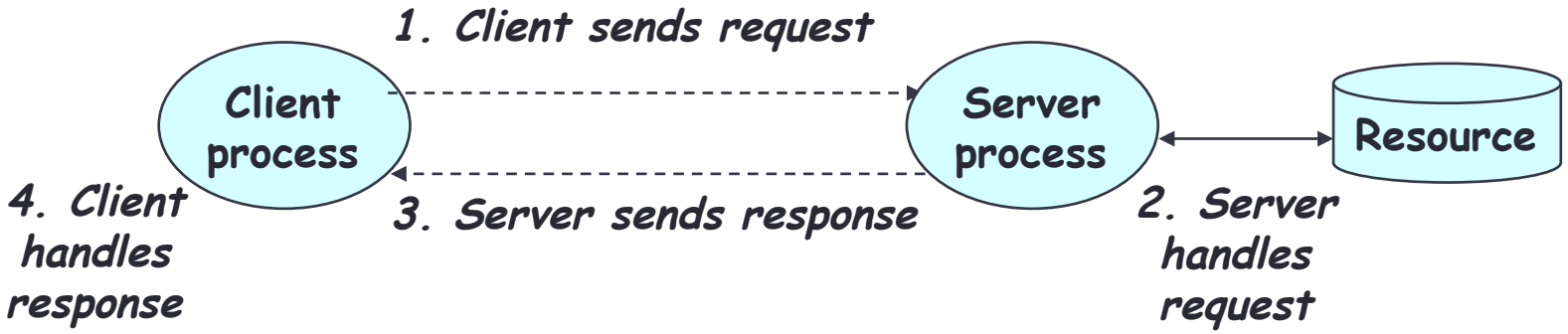
Client server model



- Client
  - Request services from Server
  - Clients do not contact directly each other
- Server
  - “Always” online waitng for requests from Clients
- Ex: Web, Mail, …

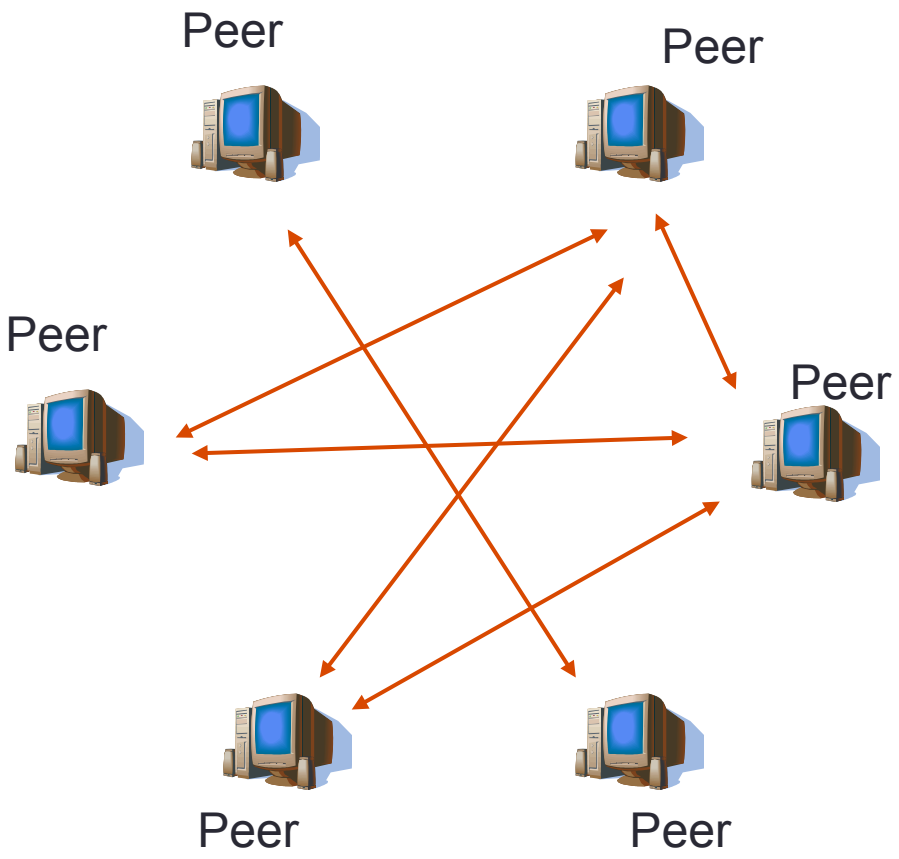
Client/sever model

- Client asks (*request*) – server provides (*response*)
- Typically: single server - multiple clients
- The server does not need to know *anything* about the client
  - even that it exists
- The client should always know *something* about the server
  - at least where it is located



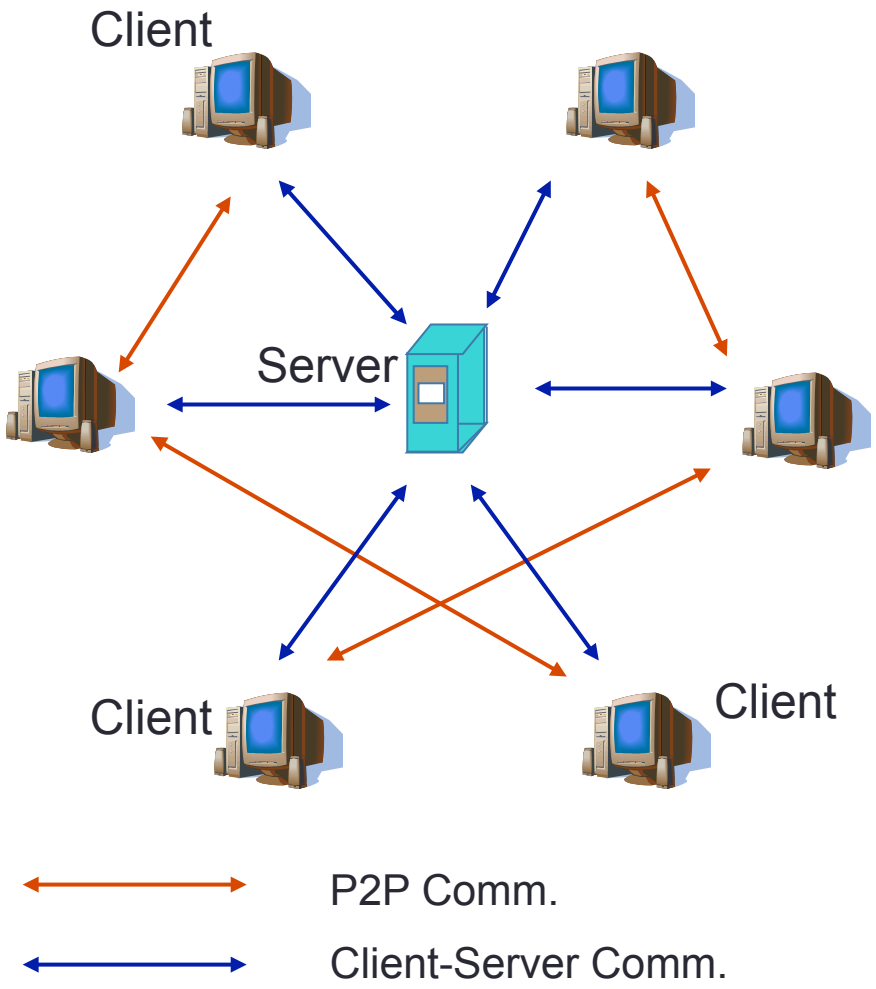
Note: clients and servers are processes running on hosts (can be the same or different hosts).

Pure Peer to Peer



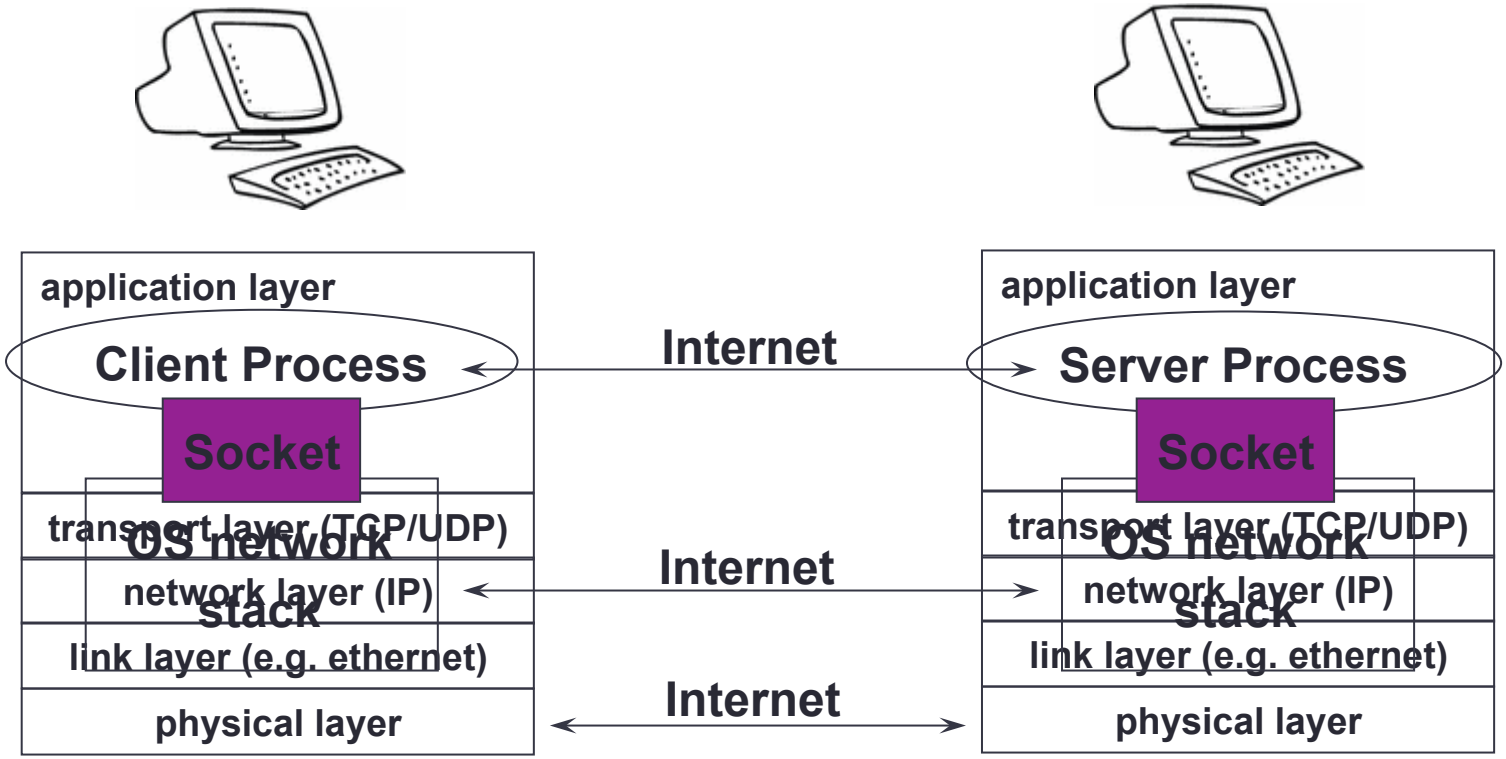
- No central server
- Peers have equal role
- Peers can communicate directly to each other
- Peers do not need to be always online
- E.g. Gnutella, Emule

Hybrid model



- Central server manages user accounts, authentification, stores data for searching process …
- Clients communicate directly after authentication process.
- E.g. Skype
  - Server manages login process.
  - Messages, voices are transmitted directly between servers.

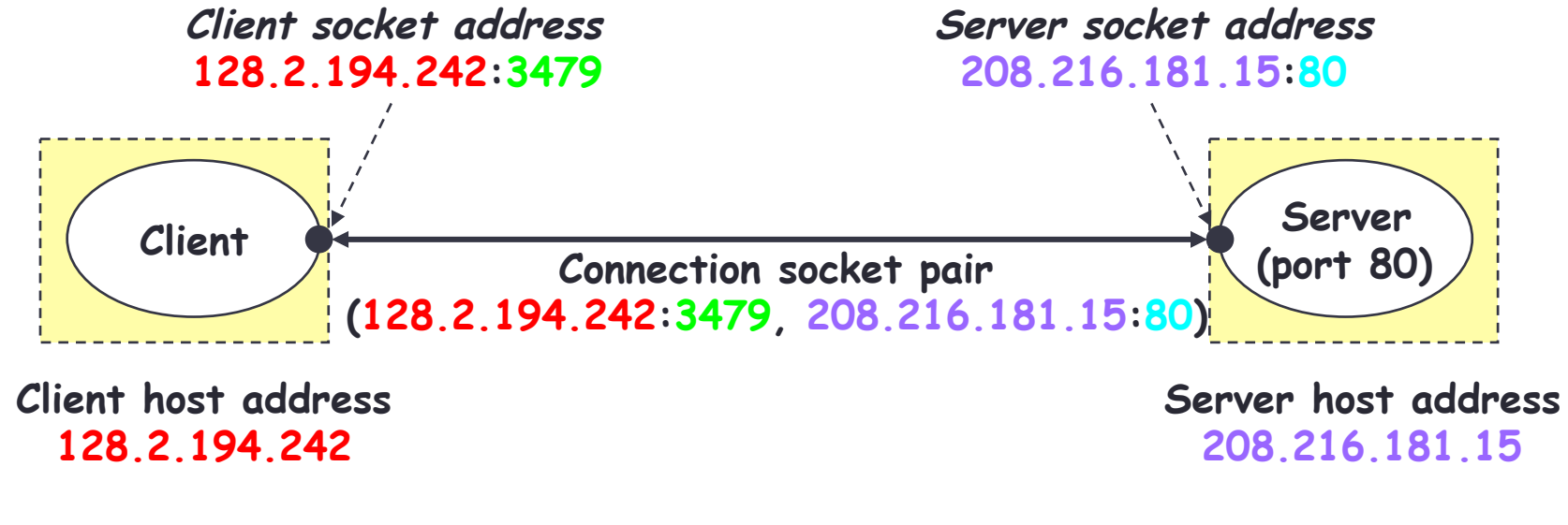
Sockets as means for inter-process communication (IPC)



The interface that the OS provides to its networking subsystem

Internet Connections (TCP/IP)

- Address the machine on the network
  - By IP address
- Address the process/application
  - By the "port"-number
- The pair of *IP-address* + *port* – makes up a "*socket-address*"



Internet Connections (TCP/IP)

- Need to open two sockets of both sides
  - Client socket
  - Server socket
- Client application send/receive data to server through client socket
- Server application send/receive data to client through client socket
- Make two sockets talking to each other.