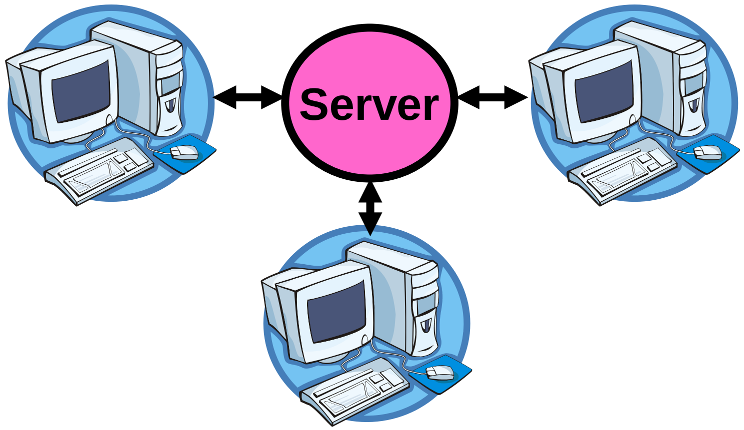
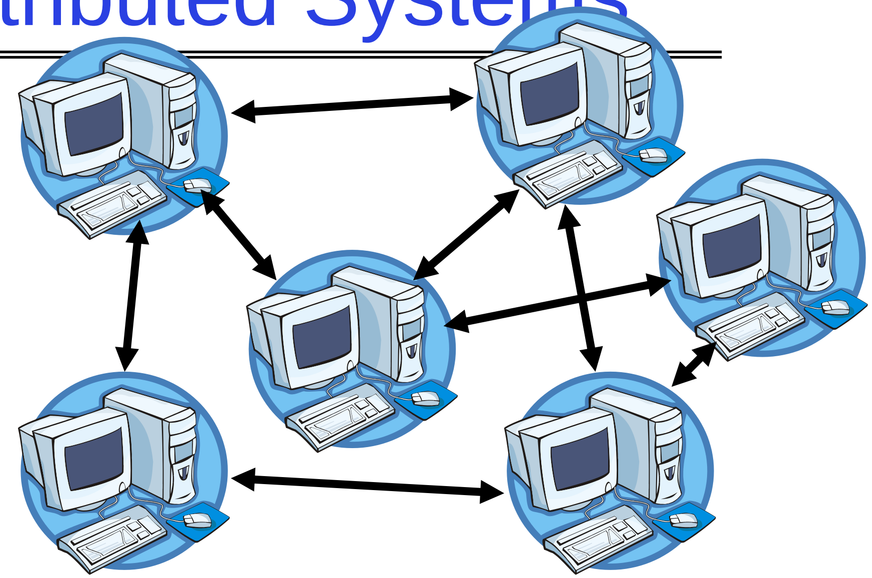
CSE150
Operating Systems
Lecture 22

Distributed Systems and Networking

Centralized vs Distributed Systems



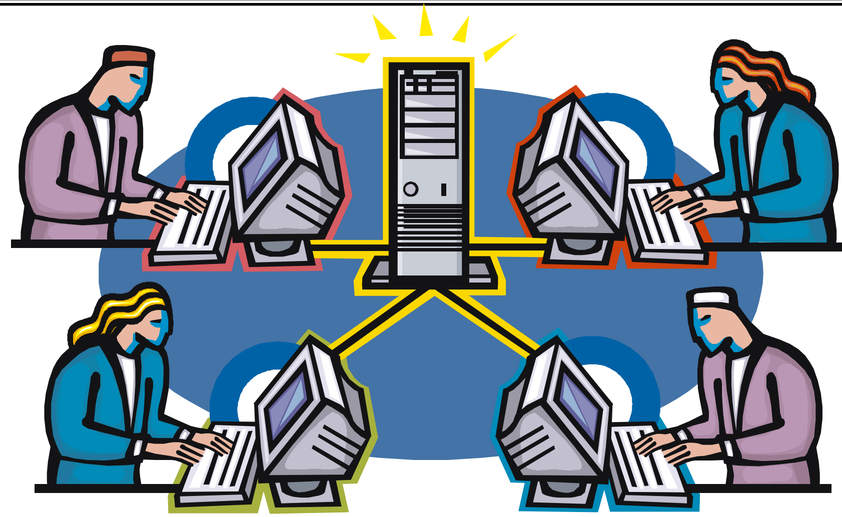
Client/Server Model



Peer-to-Peer Model

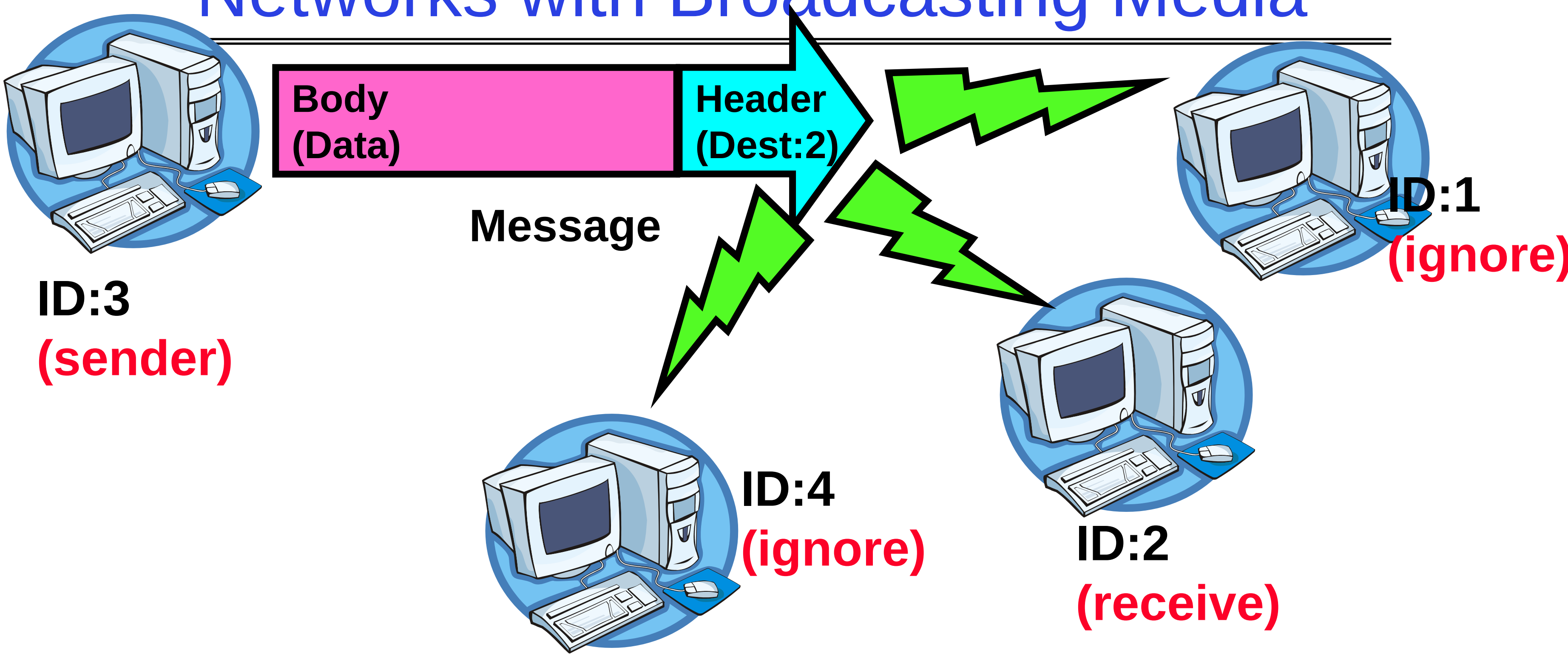
- Centralized System: System in which major functions are performed by a single physical computer
  - Originally, everything on single computer
  - Later: client/server model
- Distributed System: physically separate computers working together on some task
  - Early model: multiple servers working together
    - » Probably in the same room or building
    - » Often called a "cluster"
  - Later models: peer-to-peer/wide-spread collaboration

Networking Definitions



- Network: physical connection that allows two computers to communicate
- Packet: unit of transfer, sequence of bits carried over the network
  - Network carries packets from one CPU to another
  - Destination gets interrupt when packet arrives
- Protocol: agreement between two parties as to how information is to be transmitted

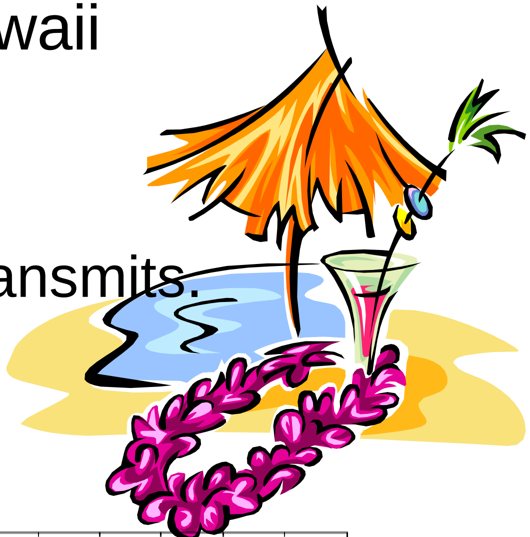
Networks with Broadcasting Media

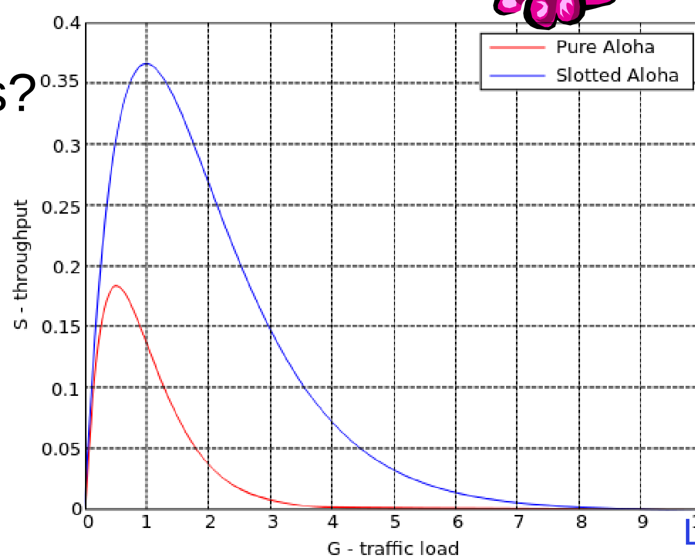


- Delivery: When you broadcast a packet, how does a receiver know who it is for? (packet goes to everyone!)
  - Put header on front of packet: [Destination | Packet]
  - Everyone gets packet, discards if not the target
  - In Ethernet, this check is done in hardware
    - » No OS interrupt if not for particular destination

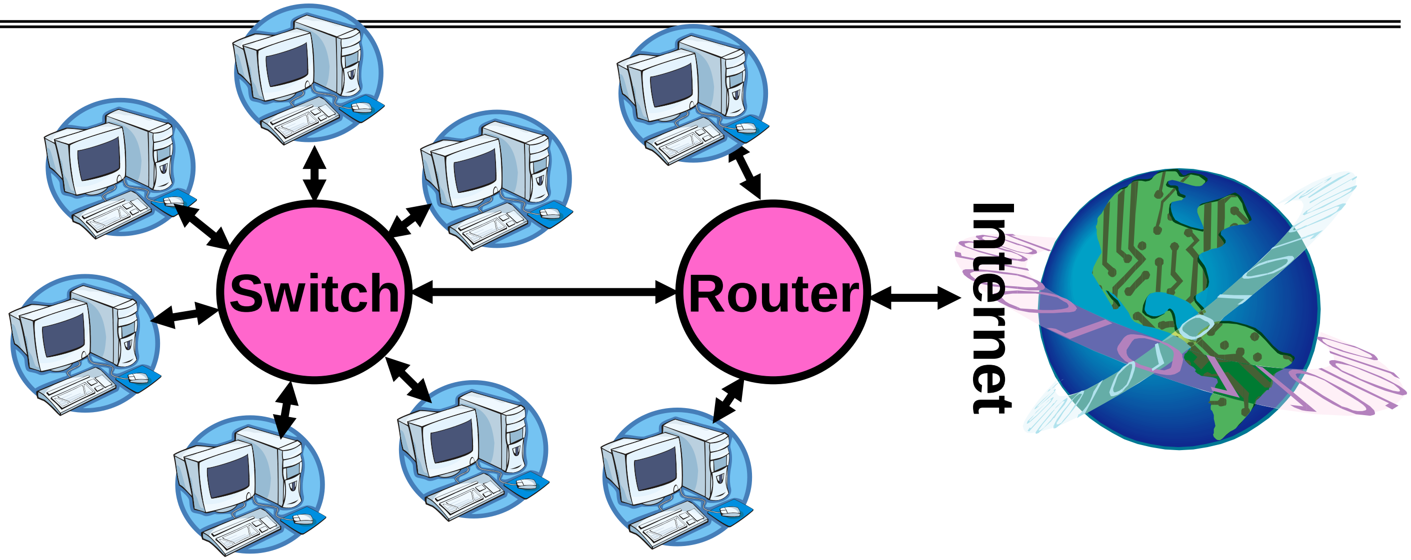
Network Arbitration

- Multiple Access: Act of negotiating use of shared medium
  - What if two senders try to broadcast at same time?
    - » Collision
- Aloha network (70's): packet radio within Hawaii
  - Blind broadcast, with checksum at end of packet. If received correctly (not garbled), send back an acknowledgement.
  - If ACK not received correctly, discard or re-transmits.
  - If two senders try to send at same time, both get garbled, both simply re-send later.
    - » Backoff a random duration.
  - Problem:
    - » Stability: what if load increases?
    - » Inefficiency





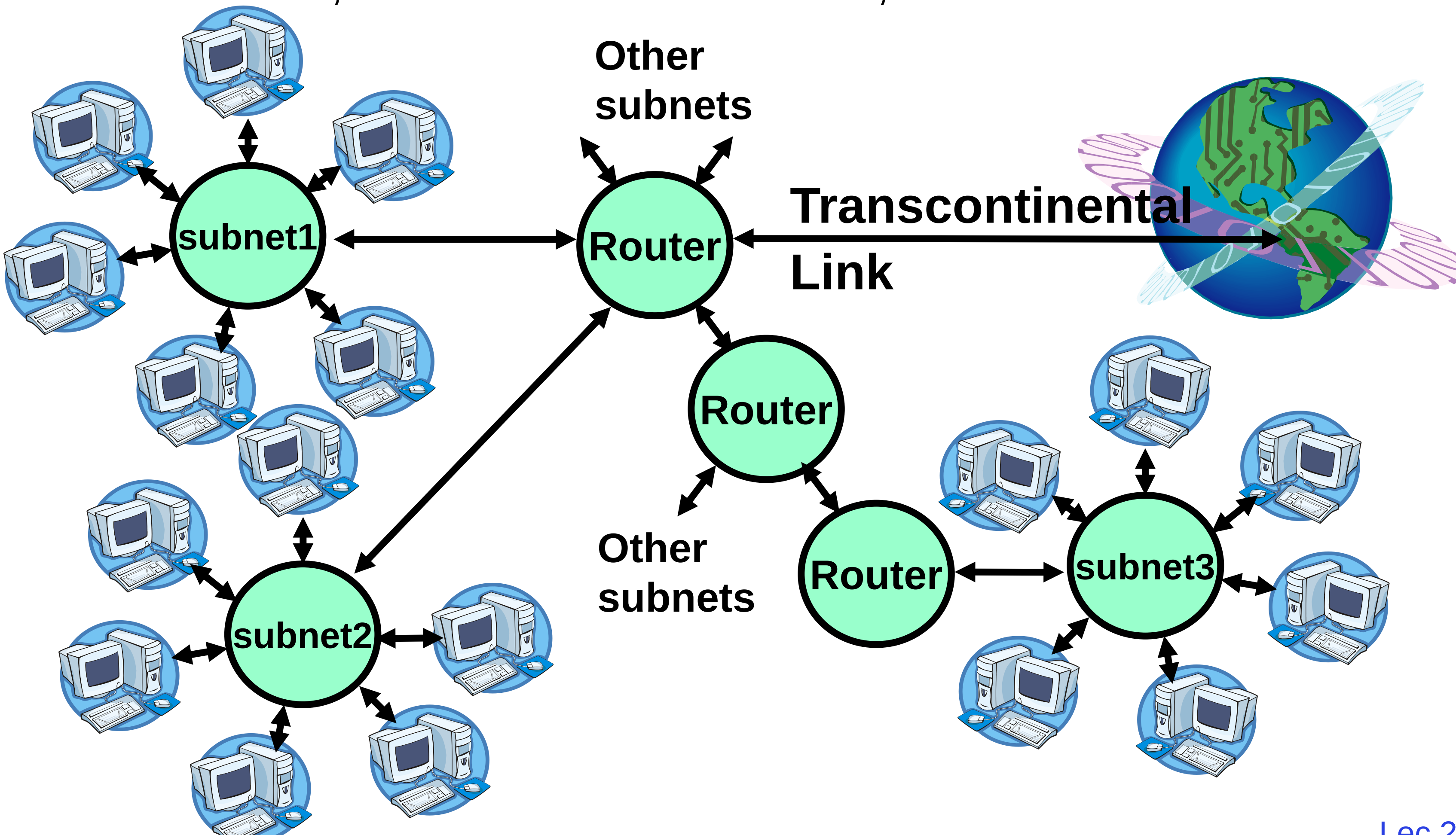
Networks with Point-to-Point Media



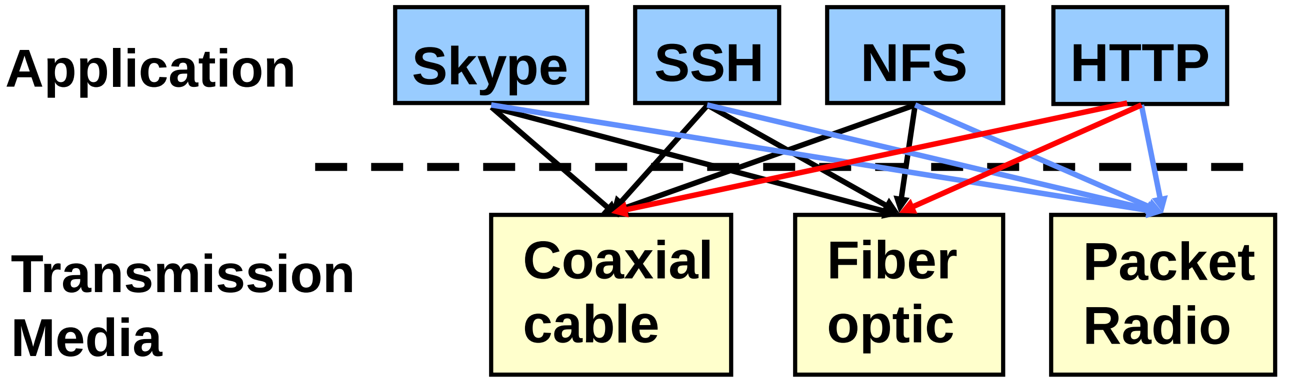
- Why have a shared bus at all? Why not simplify and only have point-to-point links + routers/switches?
  - Originally wasn't cost-effective
  - Now, easy to make high-speed switches and routers that can forward packets from a sender to a receiver.
- Point-to-point network: a network in which every physical wire is connected to only two computers
- Switch: a bridge that transforms a shared-bus (broadcast) configuration into a point-to-point network.
- Router: a device that acts as a junction between two networks to transfer data packets among them.

Hierarchical Networking: The Internet

- How can we build a network with millions of hosts?
  - Hierarchy! Not every host connected to every other one
  - Use a network of Routers to connect subnets together
    - » Routing is often by prefix: e.g. first router matches first 8 bits of address, next router matches more, etc.



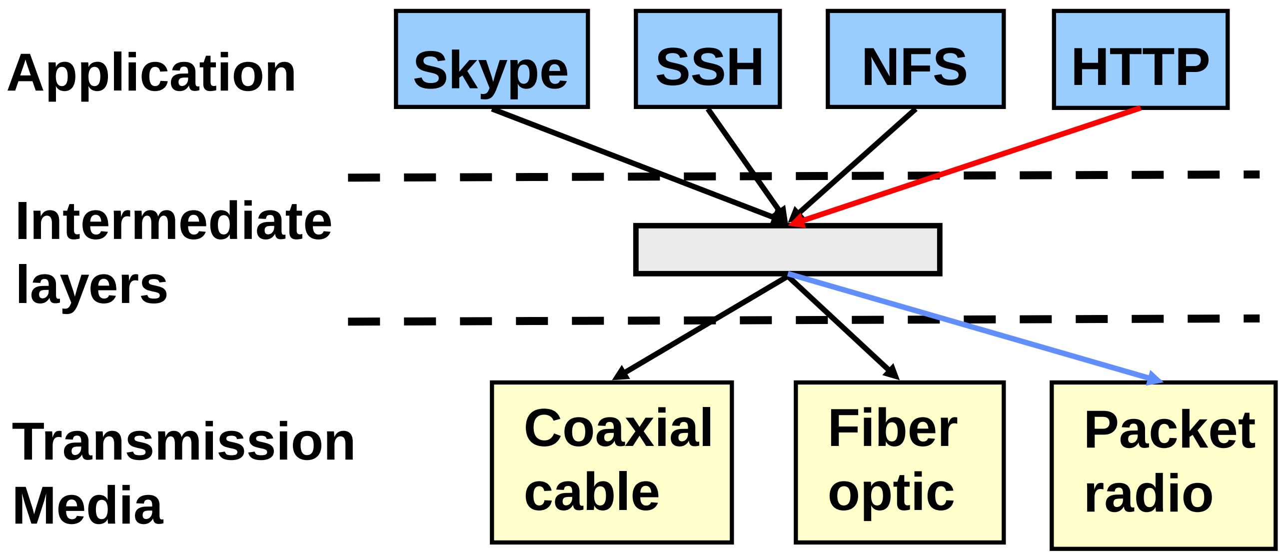
The Problem



- Re-implement every application for every technology?
- No! But how does the Internet design avoid this?

Solution: Intermediate Layers

- Introduce intermediate layers that provide set of abstractions for various network functionality & technologies
  - A new app/media implemented only once



Software System Modularity

Partition system into modules & abstractions:

- Well-defined interfaces give flexibility
  - ***Hides*** implementation - thus, it can be freely changed
  - Extend functionality of system by adding new modules
- E.g., libraries encapsulating set of functionality
- E.g., programming language + compiler abstracts away not only how the particular CPU works …

Network System Modularity

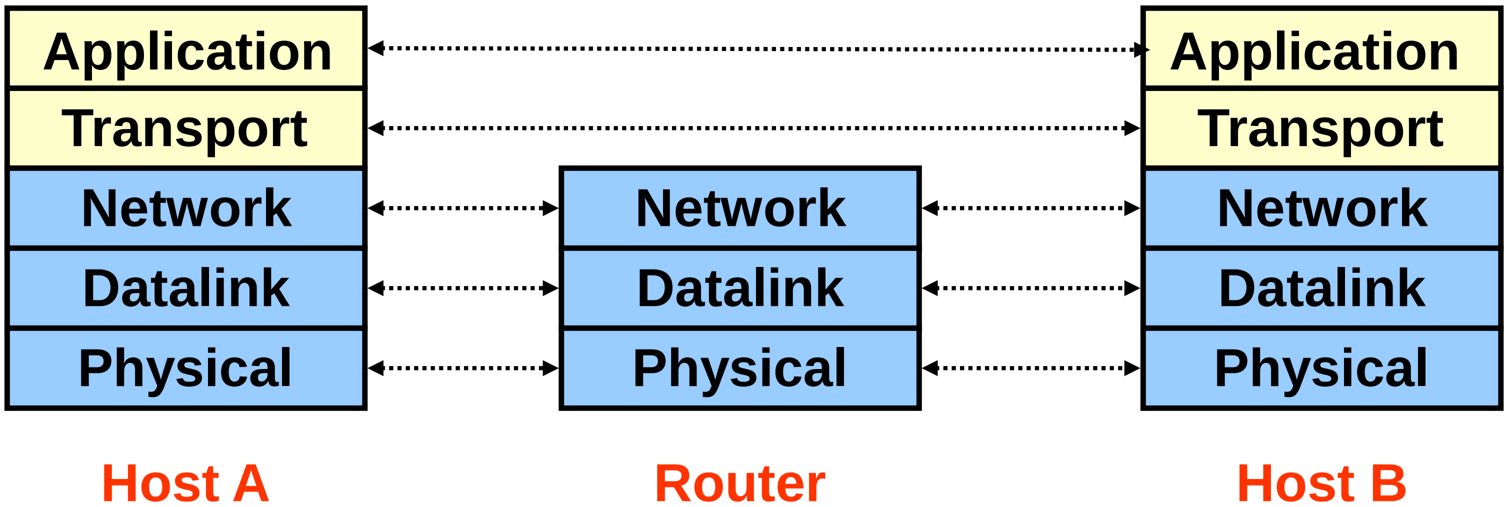
Like software modularity, but:

- Implementation distributed across many machines (routers and hosts)

- Must decide:
  - How to break system into modules:
    - » **Layering**

Internet Architecture: The Five Layers

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts
- Logically, layers interacts with peer's corresponding layer

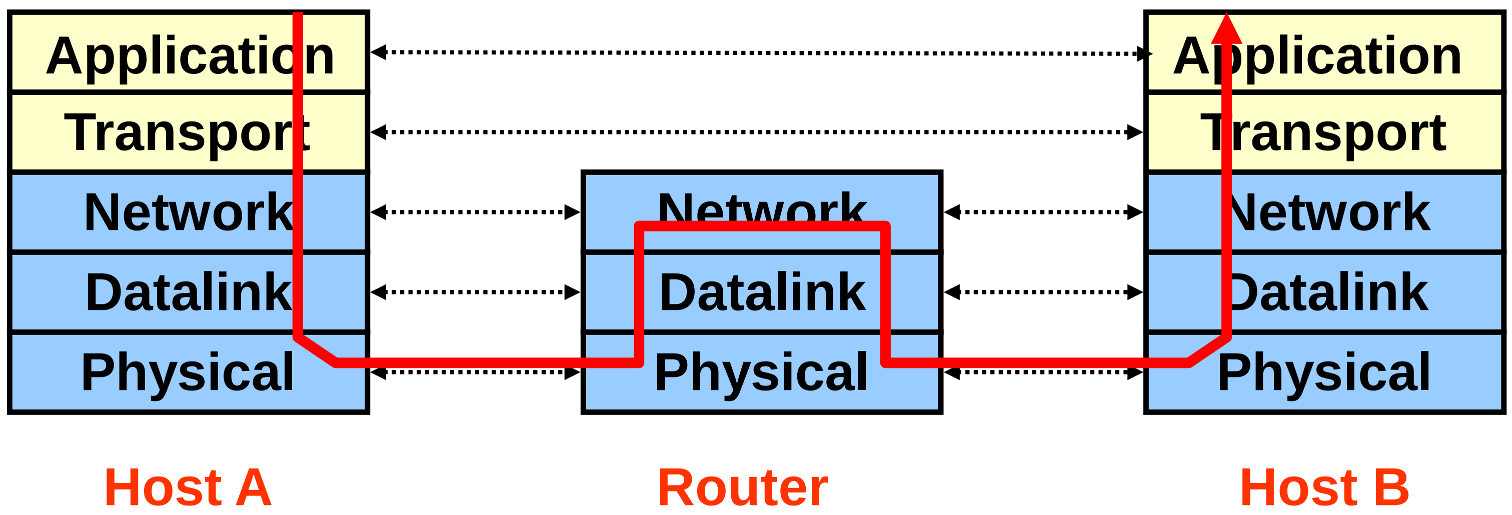


Layering: A Modular Approach

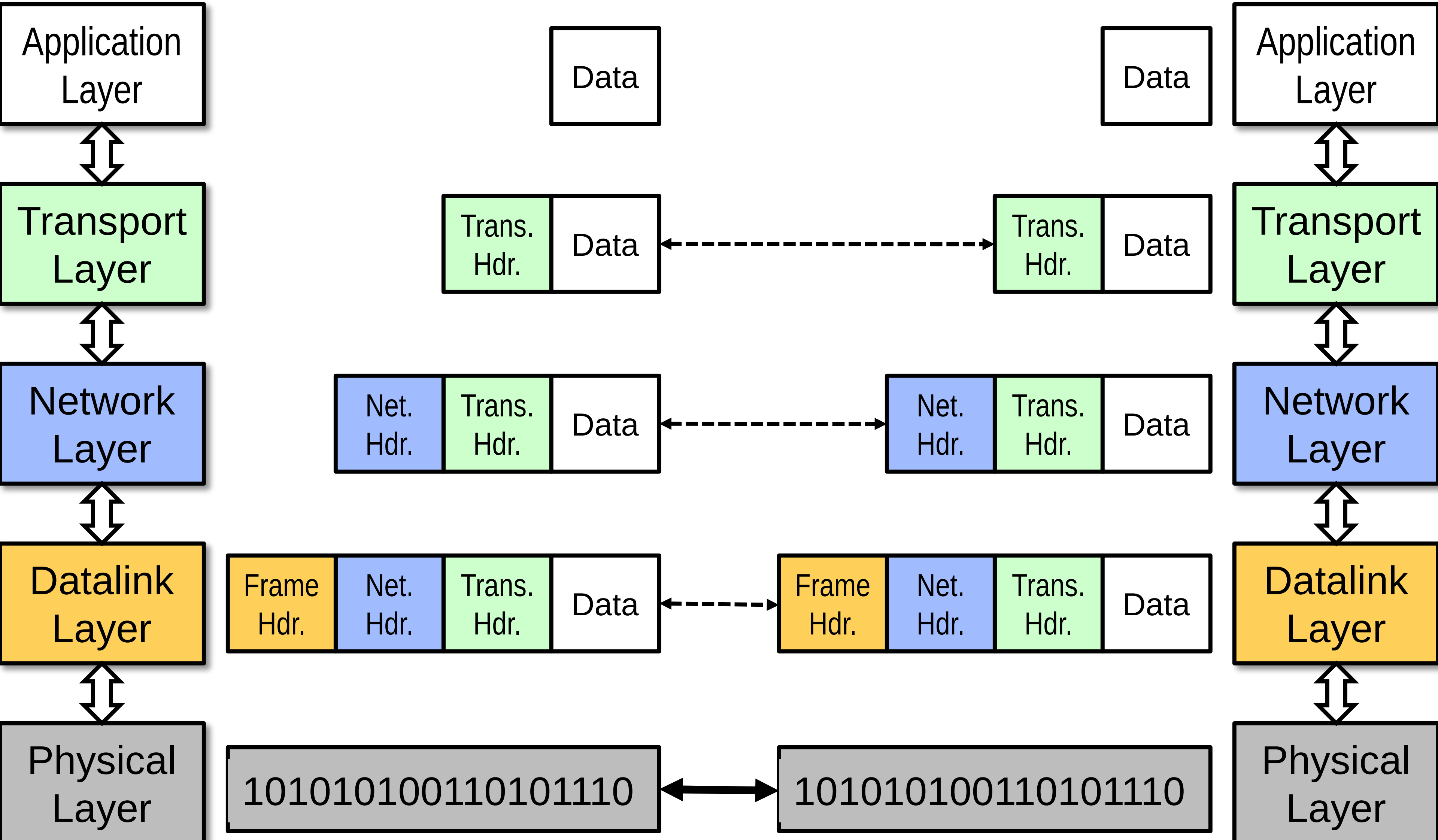
- Partition the system
  - Each layer solely relies on services from layer below
  - Each layer solely exports services to layer above
- Interface between layers defines interaction
  - Hides implementation details
  - Layers can change without disturbing other layers

Physical Communication

- Communication goes down to physical network
- Then from network peer to peer
- Then up to relevant layer



Layering: Packets in Envelopes



Application Layer (7 - not 5!)

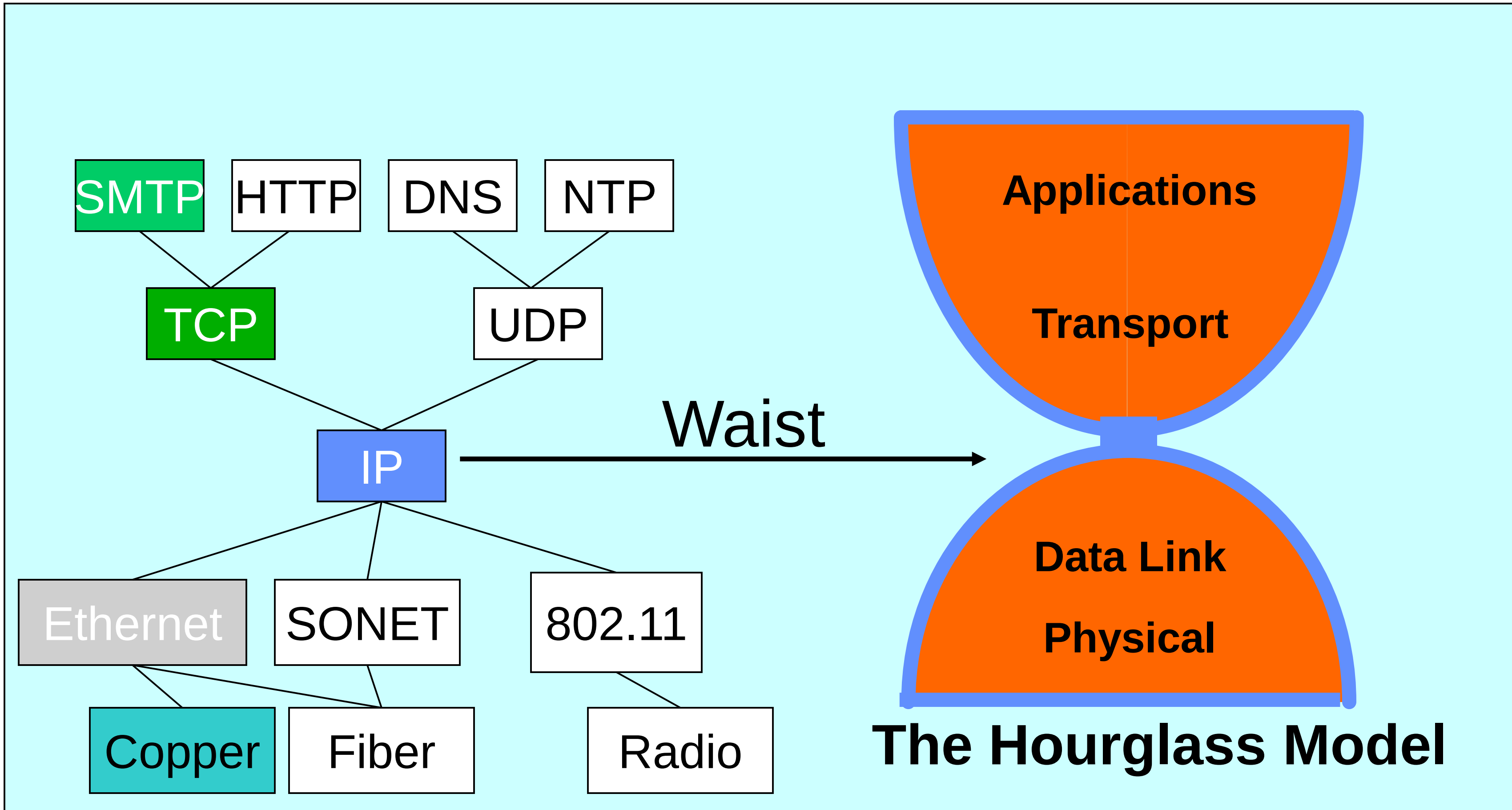
Application
Present.
Session
Transport
Network
Datalink
Physical

- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application

- Examples: Skype, SMTP (email), HTTP (Web), Halo, BitTorrent …

- What happened to layers 5 & 6?
  - "Session" and "Presentation" layers
  - Part of ***OSI*** architecture, but not Internet architecture
  - Their functionality is provided by application layer

The Internet *Hourglass*



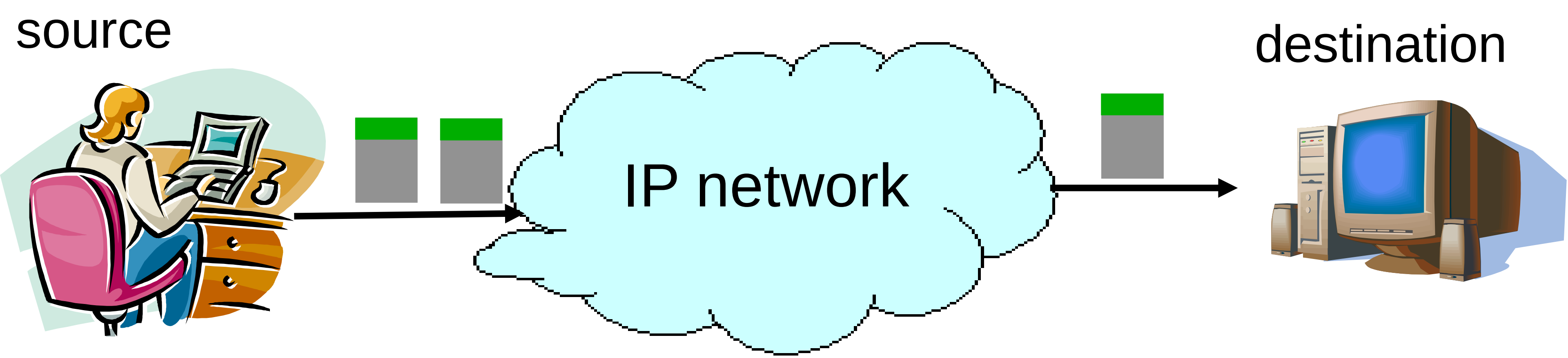
There is just one network-layer protocol, **IP**

The "narrow waist" facilitates interoperability

Internet Protocol (IP)

Application
Present.
Session
Transport
Network
Datalink
Physical

- Internet Protocol: Internet's network layer
- Service it provides: "Best-Effort" Packet Delivery
  - Tries it's "best" to deliver one packet to its destination
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



Implications of Hourglass

Single Internet-layer module (**IP**)**:**

- Allows arbitrary networks to interoperate
  - Any network technology that supports IP can exchange packets
- Allows applications to function on all networks
  - Applications that can run on IP can use any network
- Supports simultaneous innovations above and below IP

Transport Layer (4)

Application
Present.
Session
Transport
Network
Datalink
Physical

- **Service**:
  - Provide end-to-end communication between processes
  - Demultiplexing of communication between hosts
  - Possible other services:
    - » Reliability in the presence of errors
    - » Timing properties
    - » Rate adaption (flow-control, congestion control)
- **Interface**: send message to "specific process" at given destination; local process receives messages sent to it
- **Protocol**: port numbers, perhaps implement reliability, flow control, packetization of large messages, framing
- Prime Examples: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol)

Internet Transport Protocols

Application
Present.
Session
Transport
Network
Datalink
Physical

- Datagram service (**UDP**)
  - No-frills extension of "best-effort" IP
  - Multiplexing/Demultiplexing among processes
- Reliable, in-order delivery (**TCP**)
  - Connection set-up & tear-down
  - Discarding corrupted packets (segments)
  - Retransmission of lost packets (segments)
  - Flow control
  - Congestion control
- Services not available
  - Delay and/or bandwidth guarantees
  - Sessions that survive change-of-IP-address

TCP Service

1) Open connection: 3-way handshaking

2) Reliable byte stream transfer from (IPa, TCP_Port1) to (IPb, TCP_Port2)
   - Indication if connection fails: Reset

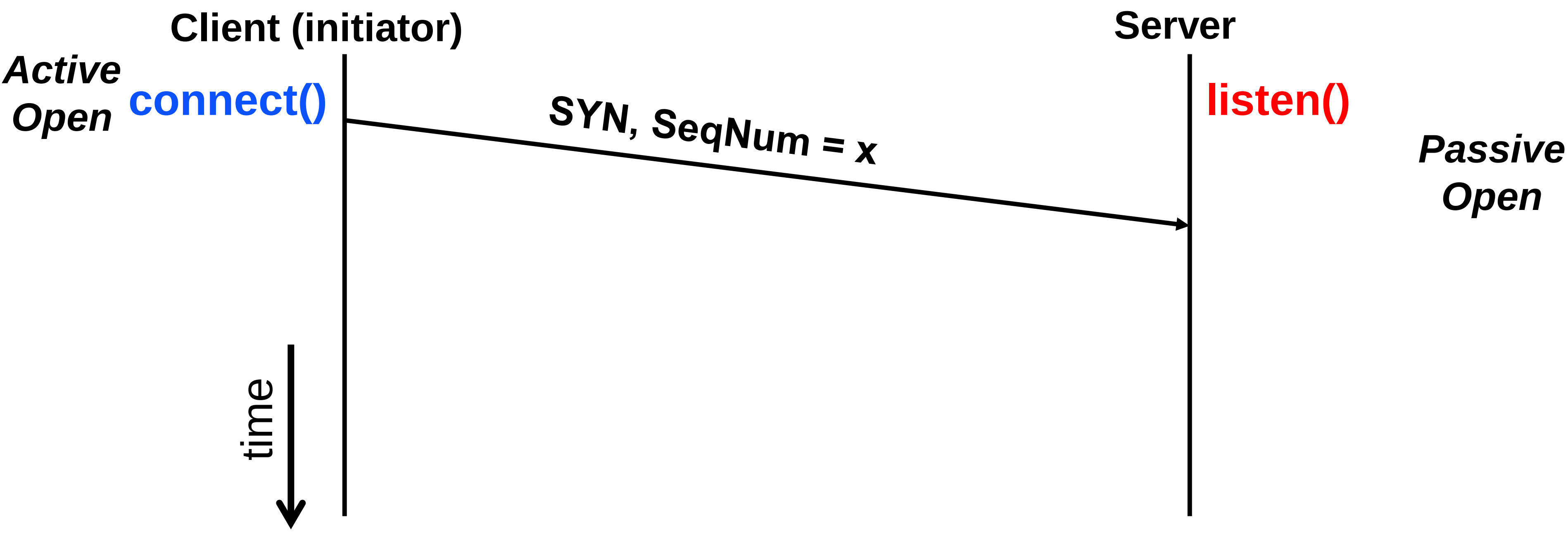
3) Close (tear-down) connection

Open Connection: 3-Way Handshaking

- Goal: agree on a set of parameters, i.e., the start sequence number for each side
  - Starting sequence number: sequence of first byte in stream
  - Starting sequence numbers are random

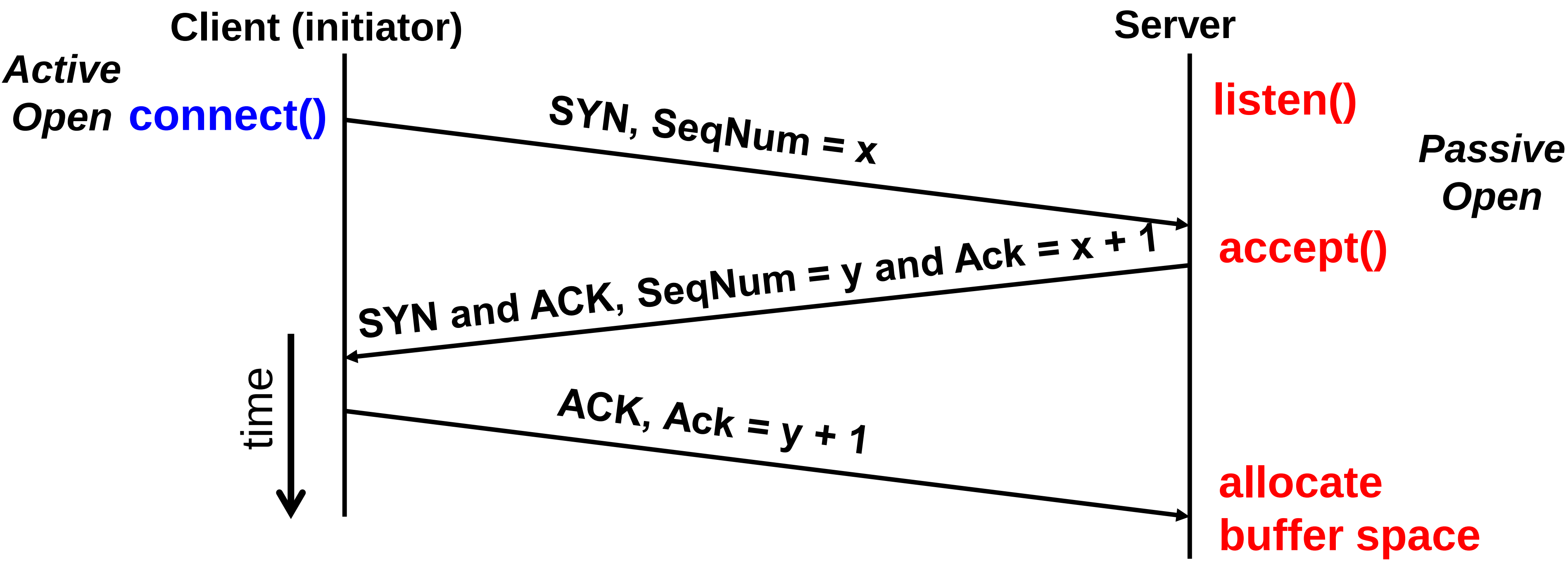
Open Connection: 3-Way Handshaking

- Server waits for new connection calling listen()
- Sender call connect() passing socket which contains server's IP address and port number
  - OS sends a special packet (SYN) containing a proposal for first sequence number, x



Open Connection: 3-Way Handshaking

- If it has enough resources, server calls accept() to accept connection, and sends back a SYN ACK packet containing
  - Client's sequence number incremented by one, (x + 1)
  - A sequence number proposal, y, for first byte server will send



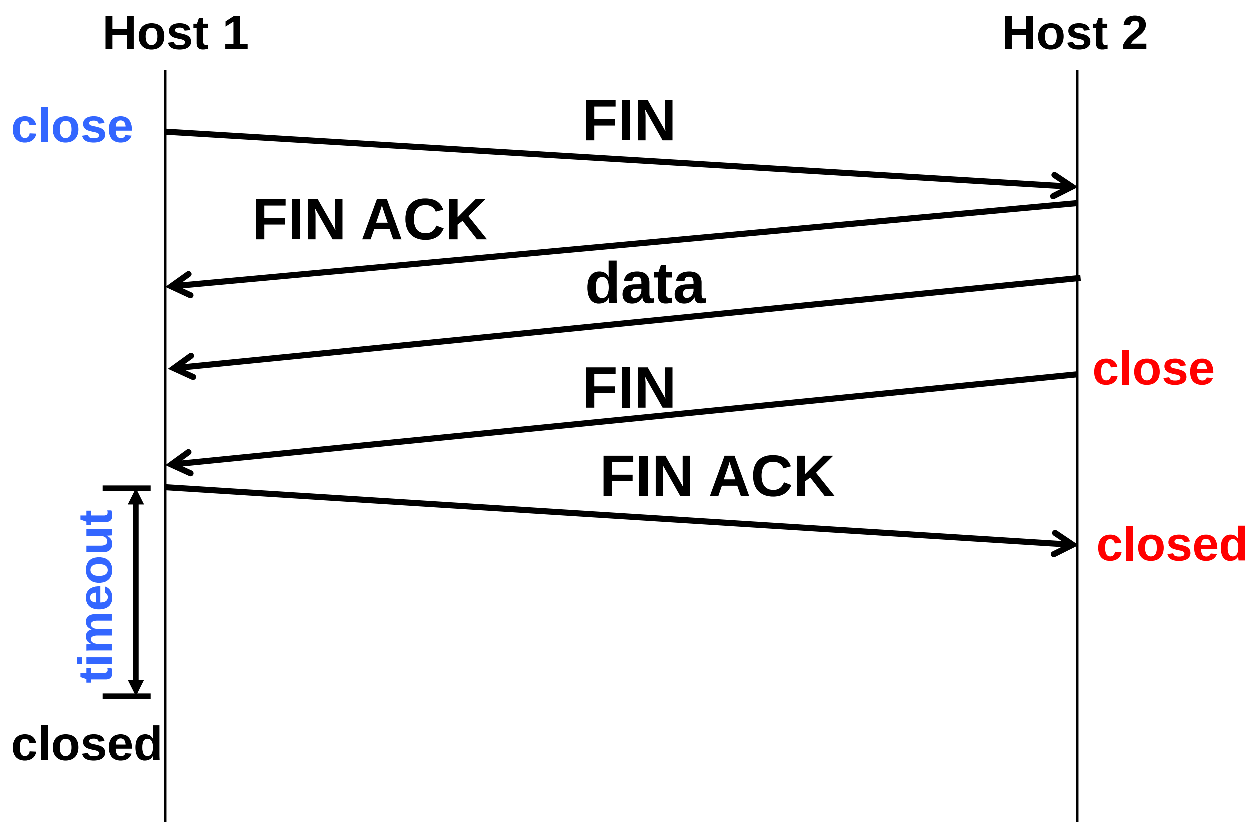
3-Way Handshaking (cont'd)

- Three-way handshake adds 1 RTT delay

- Why?
  - Full-duplex transmissions
  - Congestion control: SYN (40 byte) acts as cheap probe

Close Connection

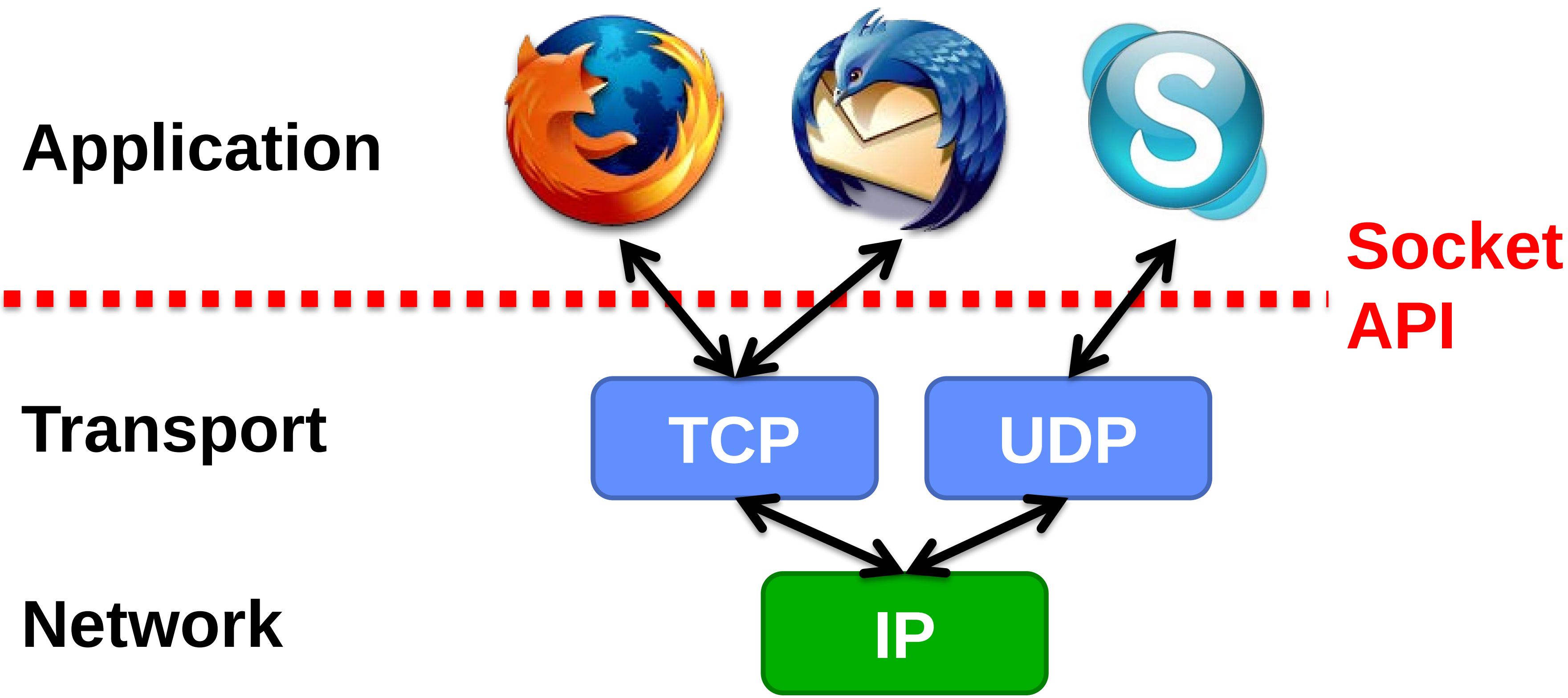
- Goal: both sides agree to close the connection
- 4-way connection tear down



- **Can retransmit FIN ACK if it is lost**
- **Protects against delayed packets from other connection**

Socket API

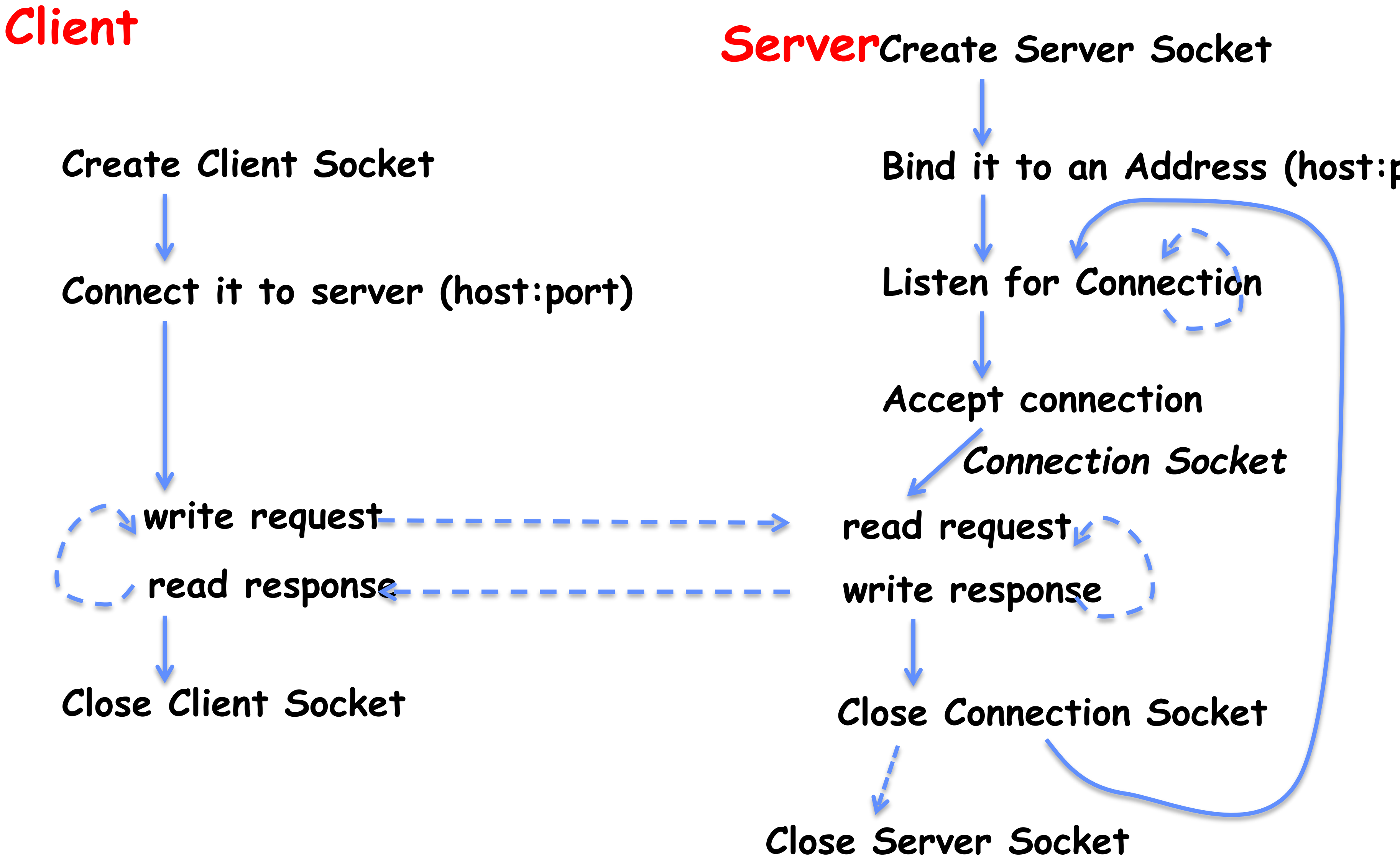
- Base level Network programming interface



BSD Socket API

- Created at UC Berkeley (1980s)
- Most popular network API
- Ported to various OSes, various languages
  - Windows Winsock, BSD, OS X, Linux, Solaris, …
  - Socket modules in Java, Python, Perl, …
- Similar to Unix file I/O API
  - In the form of *file descriptor* (sort of handle).
  - Can share same `read()`/`write()`/`close()` system calls

Sockets in concept



Summary

- Centralized vs Distributed Systems
- Network layering