

CSI62
Operating Systems and
Systems Programming
Lecture 21

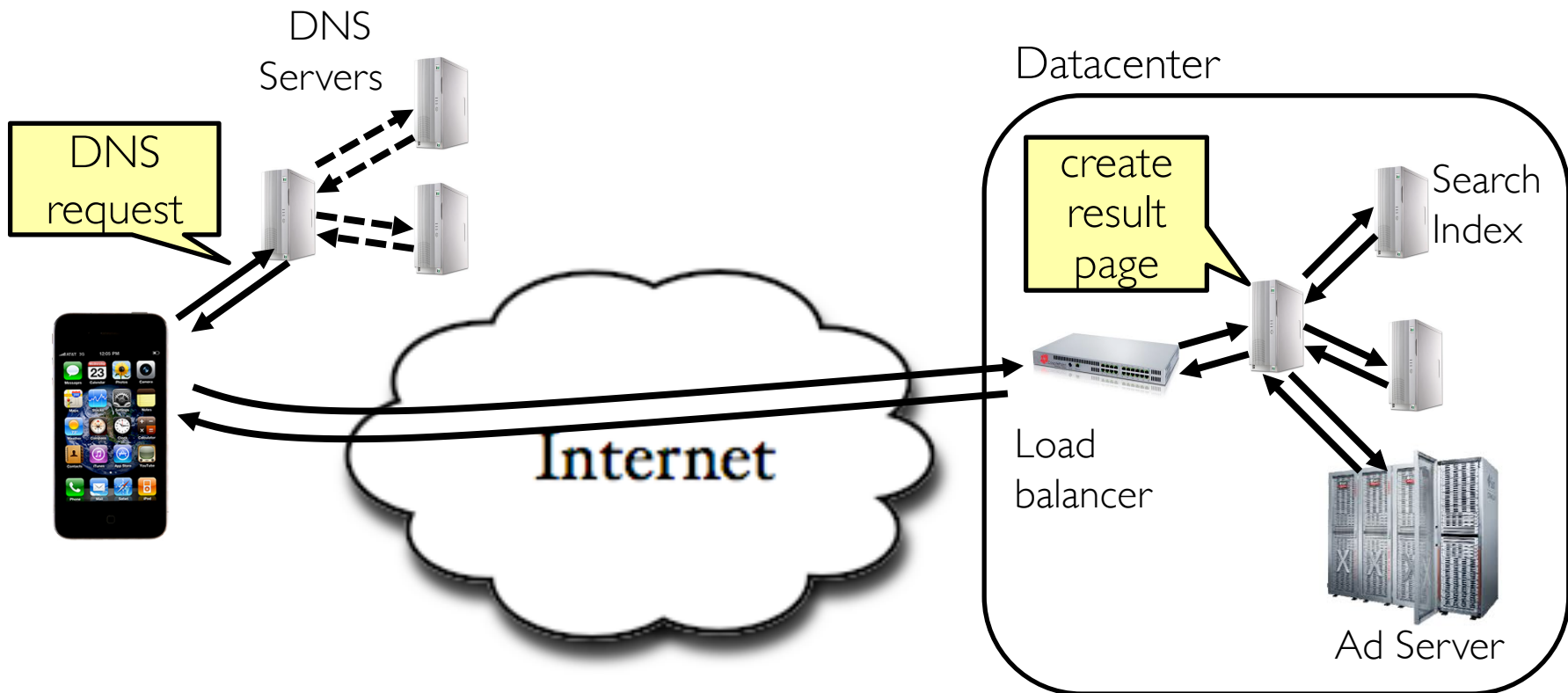
Layering,
E2E Argument

November 14th, 2018

Prof. Ion Stoica

<http://cs162.eecs.Berkeley.edu>

Example: What's in a Search Query?



- Complex interaction of multiple components in multiple administrative domains

Goals for Today

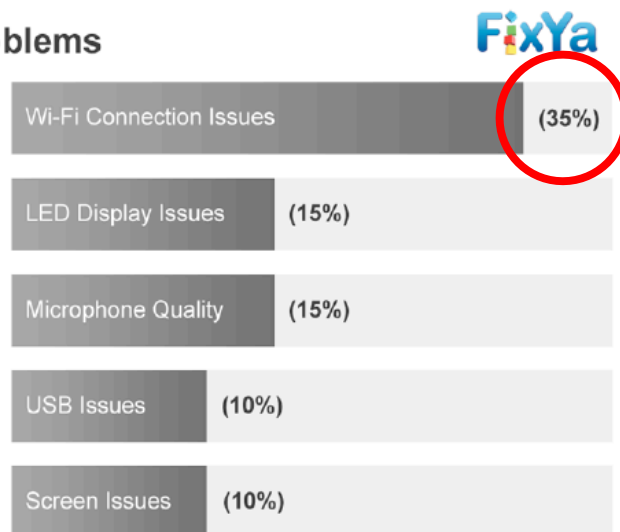
- Layering
- End-to-end arguments

**Some slides generated from Vern Paxson and Scott Shenker
lecture notes**

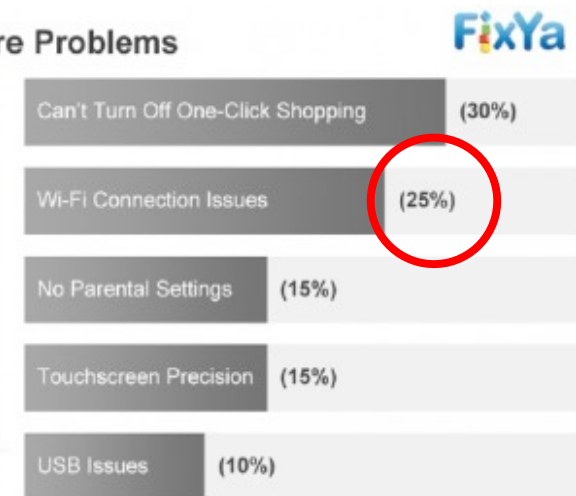
Why is Networking Important?

- Virtually all apps you use communicate over network
 - Many times main functionality is implemented remotely (e.g., Google services, Amazon, Facebook, Twitter, ...)
- Thus, connectivity is key service provided by an OS
 - Many times, connectivity issues → among top complaints

Top 5 iPad 2 Problems



Top 5 Kindle Fire Problems

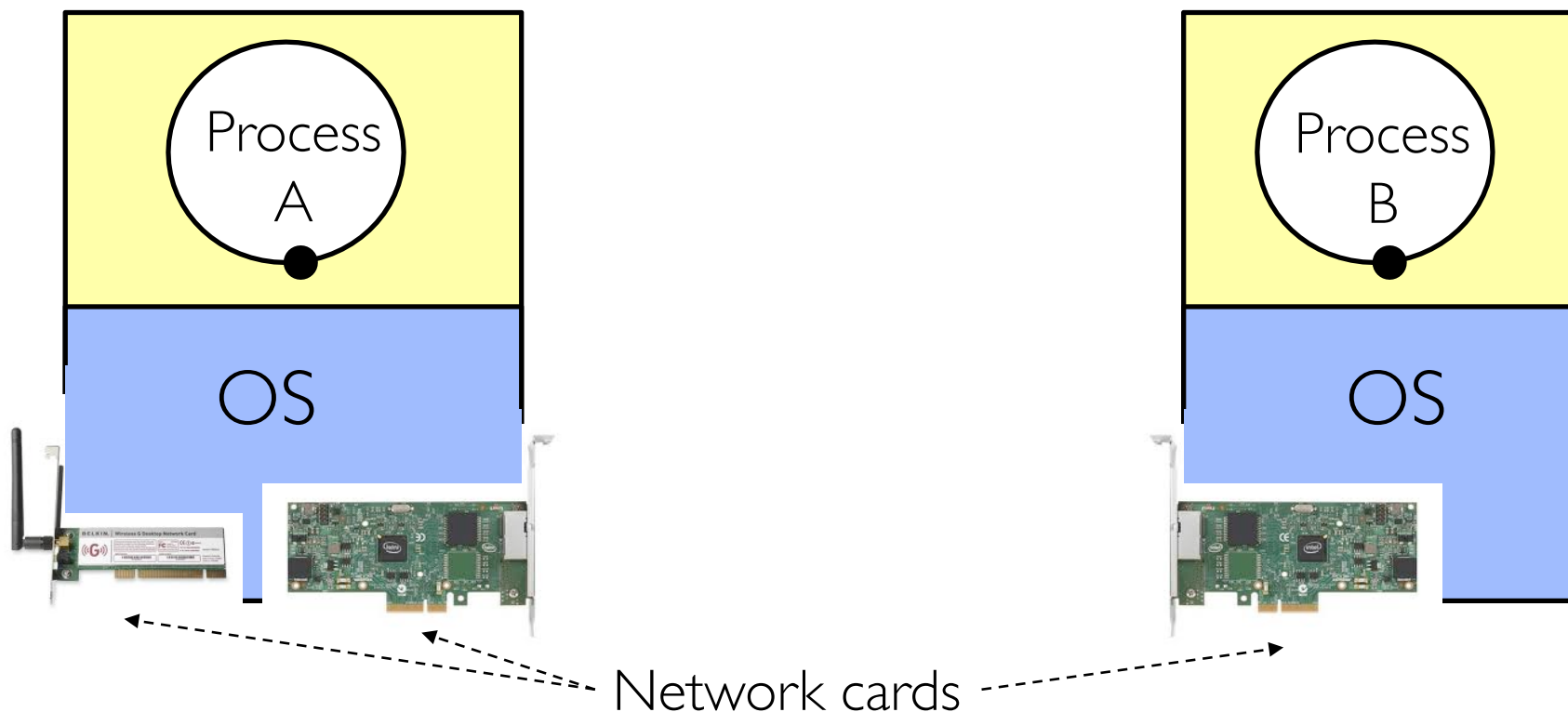


Why is Networking Important?

- Virtually all apps you use communicate over network
 - Many times main functionality is implemented remotely (e.g., Google services, Amazon, Facebook, Twitter, ...)
- Thus, connectivity is key service provided by an OS
 - Many times, connectivity issues → among top complaints
- Some of the hottest opportunities in the OS space:
 - Optimize OS for network elements (e.g., intrusion detection, firewalls)
 - OSes for Software Defined Networks (SDNs)

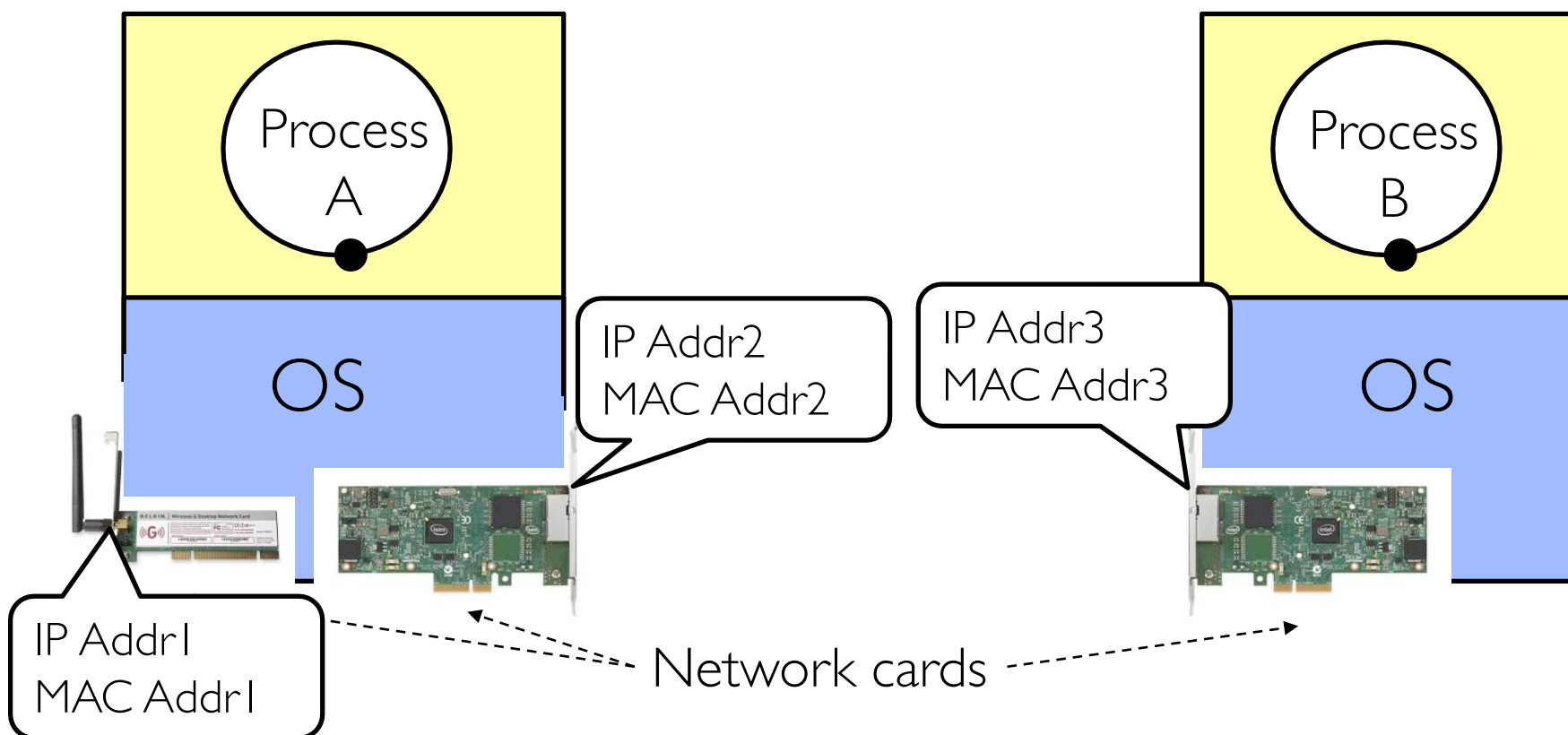
Network Concepts

- **Network (interface) card/controller:** hardware that physically connects a computer to the network
- A computer can have more than one networking cards
 - E.g., one card for wired network, and one for wireless network



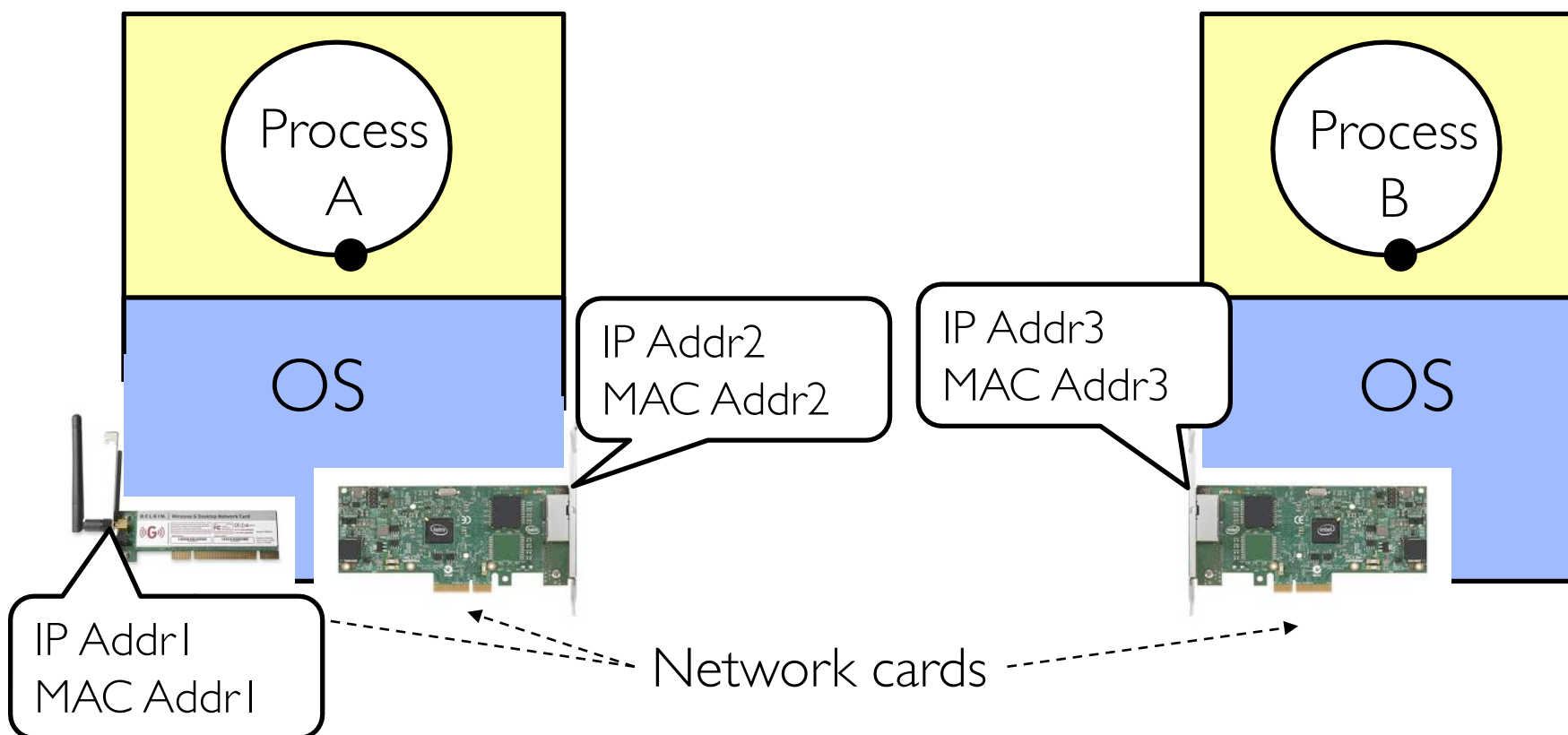
Network Concepts (cont' d)

- Typically, each network card is associated two addresses:
 - Media Access Control (MAC), or physical, address
 - IP, or network, address (can be shared by network cards on same host)



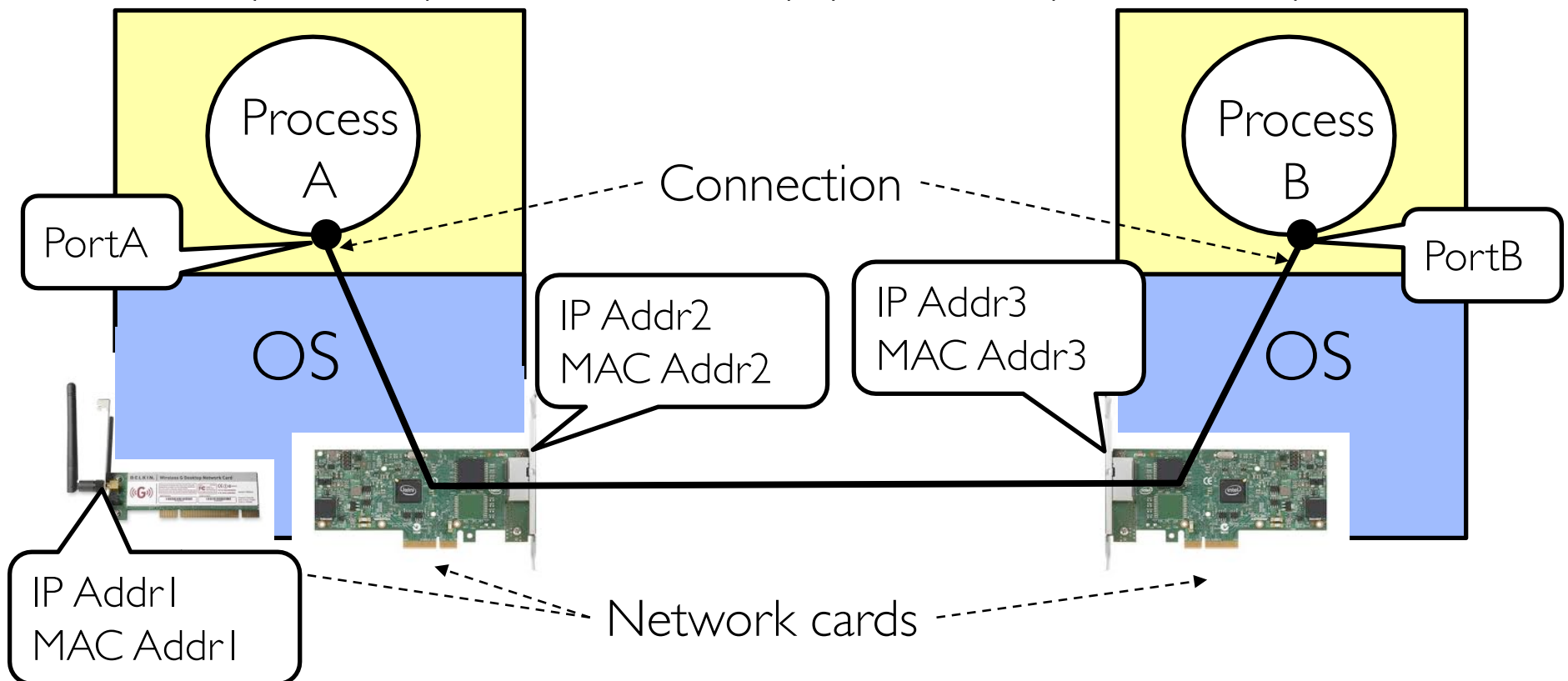
Network Concepts (cont' d)

- **MAC address:** 48-bit unique identifier assigned by card vendor
- **IP Address:** 32-bit (or 128-bit for IPv6) address assigned by network administrator or dynamically when computer connects to network



Network Concepts (cont' d)

- **Connection:** communication channel between two processes
- Each endpoint is identified by a **port number**
 - **Port number:** 16-bit identifier assigned by app or OS
 - Globally, an endpoint is identified by (IP address, port number)



Main Network Functionalities

- **Delivery:** deliver packets between any two hosts in the Internet
 - E.g., how do you deliver a packet from a host in Berkeley to a host in Tokyo?
- **Reliability:** tolerate packet losses
 - E.g., how do you ensure all bits of a file are delivered in the presence of packet losses?
- **Flow control:** avoid overflowing the receiver buffer
 - Recall our bounded buffer example: stop sender from overflowing receiver's buffer
 - » E.g., how do you ensure that a server that can send at 10Gbps doesn't overwhelm a mobile phone?
- **Congestion control:** avoid overflowing the buffer of a router along the path
 - What happens if this happens?

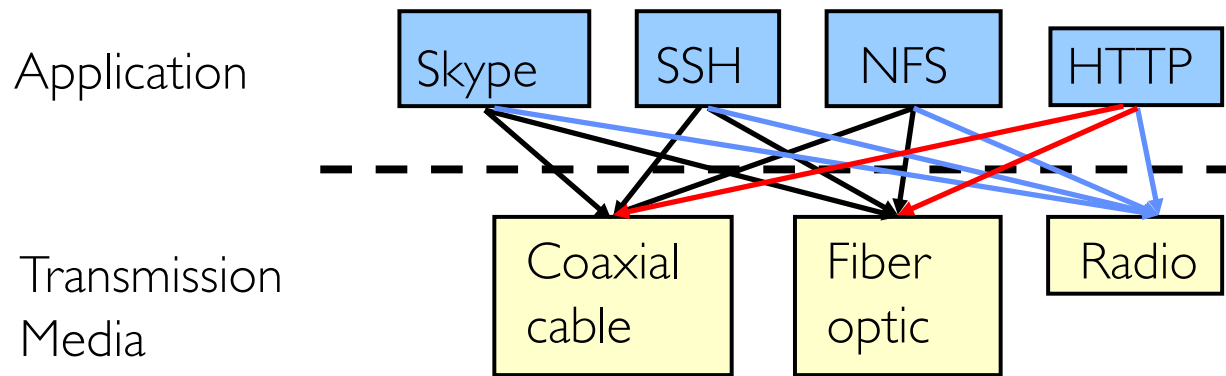
Protocol Standardization

- Ensure communicating hosts speak the same protocol
 - Standardization to enable multiple implementations
 - Or, the same folks have to write all the software
- Standardization: Internet Engineering Task Force
 - Based on working groups that focus on specific issues
 - Produces “Request For Comments” (RFCs)
 - » Promoted to standards via rough consensus and running code
 - IETF Web site is <http://www.ietf.org>
 - RFCs archived at <http://www.rfc-editor.org>
- De facto standards: same folks writing the code
 - P2P file sharing, Skype, <your protocol here> ...

Layering: The Problem

- Many different applications
 - email, web, P2P, etc.
- Many different network styles and technologies
 - Circuit-switched vs packet-switched, etc.
 - Wireless vs. wired vs optical, etc.
- How do we organize this mess?

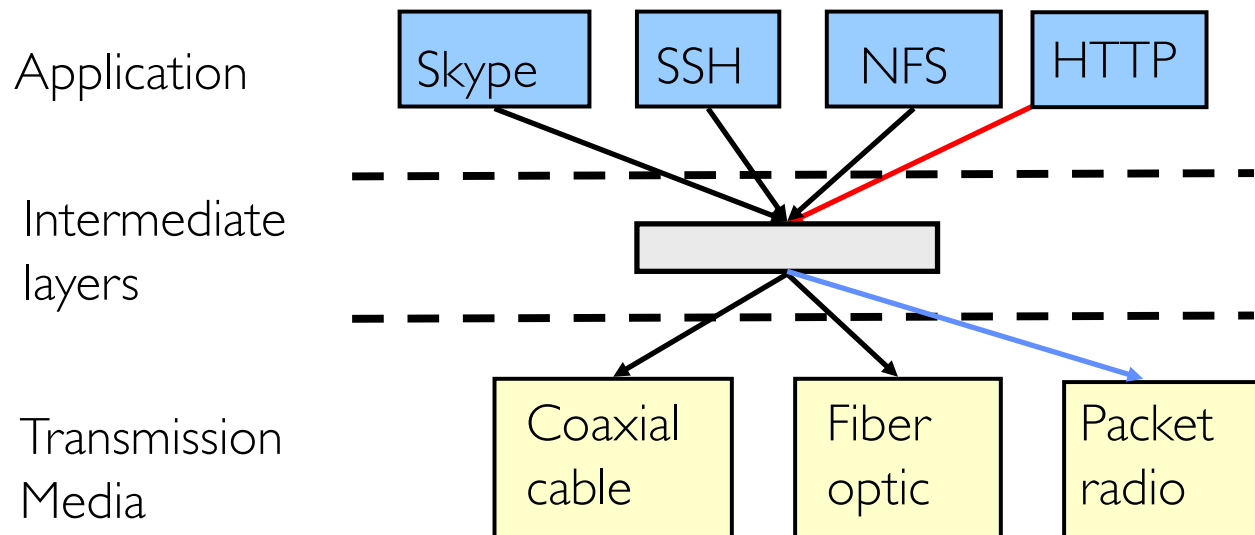
The Problem (cont' d)



- 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
 - Variation on “add another level of indirection”



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 ...
 - ... but also the **basic computational model**
- Well-defined interfaces hide information
 - Isolate **assumptions**
 - Present high-level **abstractions**
 - But can impair performance

Network System Modularity

Like software modularity, but:

- Implementation distributed across many machines (routers and hosts)
- Must decide:
 - How to break system into modules
 - » **Layering**
 - What functionality does each module implement
 - » **End-to-End Principle**
 - Where state is stored
 - » **Fate-sharing**
- We will address these choices in turn

Layering: A Modular Approach

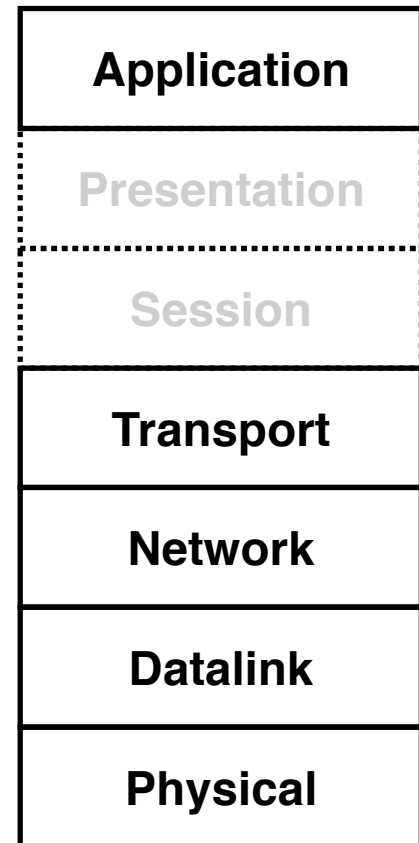
- Partition the system
 - Each layer **solely** relies on services from layer below
 - Each layer **solely** exports services to layer above
- Interface between layers defines interaction
 - Hides implementation details
 - Layers can change without disturbing other layers

Properties of Layers (OSI Model)

- **Service:** *what* a layer does
- **Service interface:** *how* to *access* the service
 - Interface for layer above
- **Protocol** (*peer interface*): *how* peers communicate to implement the service
 - Set of rules and formats that specify the communication between network elements
 - Does **not** specify the implementation on a single machine, but how the layer is implemented **between** machines

OSI Layering Model

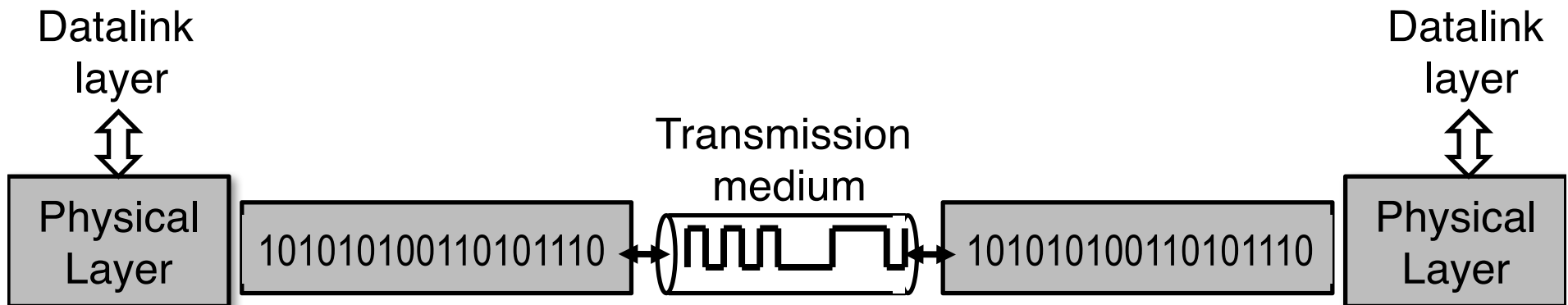
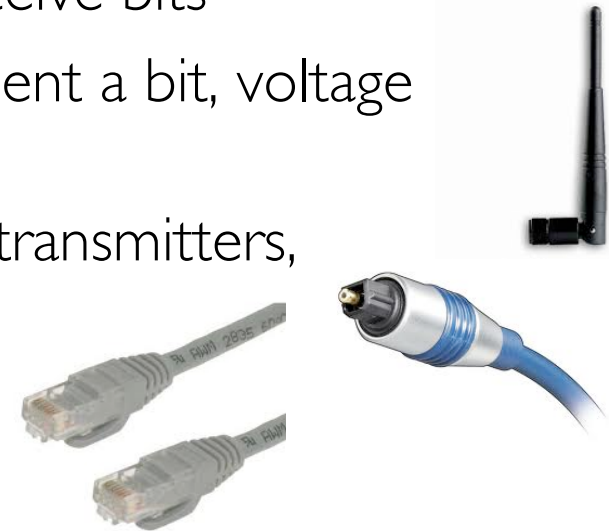
- Open Systems Interconnection (OSI) model
 - Developed by International Organization for Standardization (ISO) in 1984
 - **Seven** layers
- Internet Protocol (IP)
 - Only **five** layers
 - The functionalities of the missing layers (i.e., Presentation and Session) are provided by the Application layer



Physical Layer (1)

Application
Present. Session
Transport
Network
Datalink
Physical

- **Service:** move information between two systems connected by a physical link
- **Interface:** specifies how to send and receive bits
- **Protocol:** coding scheme used to represent a bit, voltage levels, duration of a bit
- Examples: coaxial cable, optical fiber links; transmitters, receivers



Datalink Layer (2)

Application
Present. Session
Transport
Network
Datalink
Physical

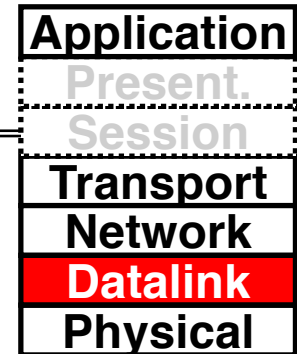
- **Service:**

- Enable end hosts to exchange frames (atomic messages) on the same physical line or wireless link
- Possible other services:
 - » *Arbitrate access* to common physical media
 - » May provide *reliable transmission, flow control*

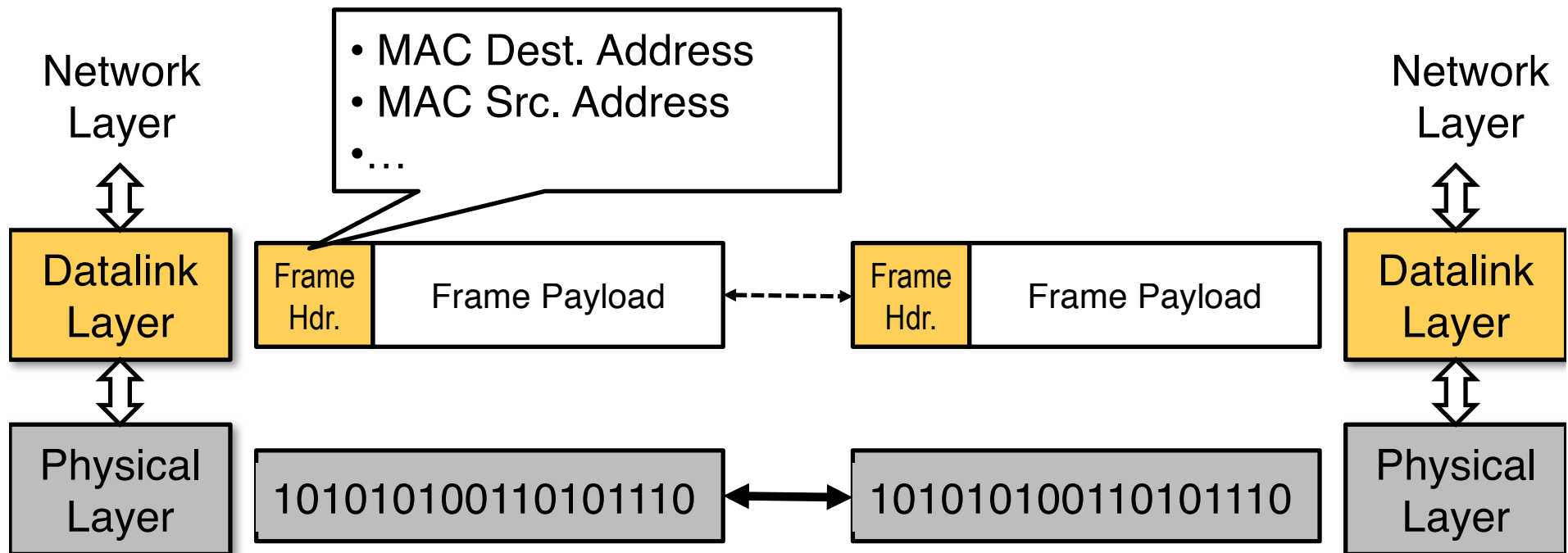
- **Interface:** send *frames* to other end *hosts*; receive *frames* addressed to end host

- **Protocols:** addressing, Media Access Control (MAC) (e.g., CSMA/CD - *Carrier Sense Multiple Access / Collision Detection*)

Datalink Layer (2)



- Each frame has a header which contains a source and a destination MAC address
- MAC (Media Access Control) address
 - Uniquely identifies a network interface
 - 48-bit, assigned by the device manufacturer



MAC Address Examples

Application
Present Session
Transport Network
Datalink

- Can easily find MAC addr. on your machine/device:
 - E.g., ifconfig (Linux, Mac OS X), ipconfig (Windows)

The image shows two screenshots side-by-side. On the left is an iPhone 'About' page with the 'Wi-Fi Address' highlighted as 00:1E:C2:CE:12:C4. On the right is a Windows command prompt window showing network configuration details for two adapters. The 'Wireless LAN adapter Wireless Network' section has its 'Physical Address' highlighted as 00-13-00-E1-11-11, with a yellow callout box pointing to it labeled 'Wi-Fi MAC address'. The 'Ethernet adapter Local Area Connection' section has its 'Physical Address' highlighted as 00-00-00-1A-1F-25, with a yellow callout box pointing to it labeled 'Wired/Ethernet MAC address'.

Local Area Networks (LANs)

Application
Present. Session
Transport
Network
Datalink
Physical

- LAN: group of hosts/devices that
 - are in the same geographical proximity (e.g., same building, room)
 - use same physical communication technology
- Examples:
 - all laptops connected wirelessly at a Starbucks café
 - all devices and computers at home
 - all hosts connected to wired Ethernet in an office

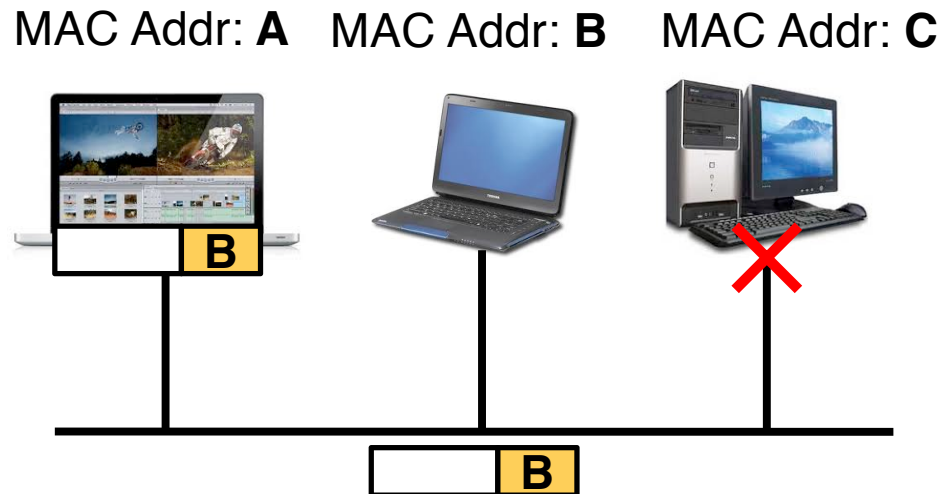


Ethernet cable and port

LANs

Application
Present. Session
Transport
Network
Datalink
Physical

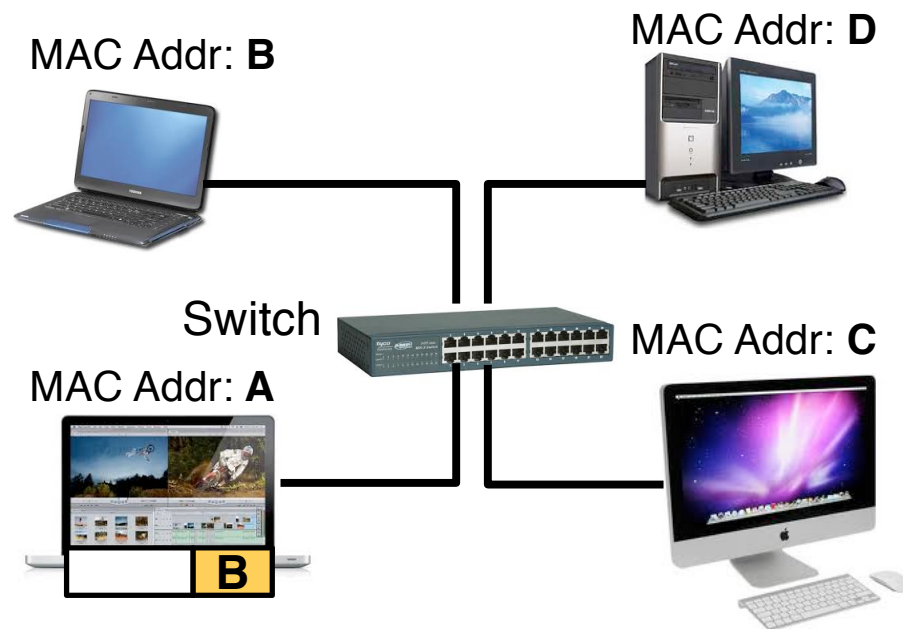
- All hosts in a LAN can share same physical communication media
 - Also called, broadcast channel
- Each frame is delivered to every host
- If a host is not the intended recipient, it drops the frame



Switches

Application
Present. Session
Transport Network
Datalink
Physical

- Hosts in same LAN can be also connected by switches
- A switch forwards frames only to intended recipients
 - Far more efficient than broadcast channel



Media Access Control (MAC) Protocols

Application
Present. Session
Transport
Network
Datalink
Physical

- Problem:
 - How do hosts access a broadcast media?
 - How do they avoid collisions?

- Three solutions:
 - Channel partition
 - “Taking turns”
 - Random access

MAC Protocols

Application
Present. Session
Transport Network
Datalink
Physical

- **Channel partitioning protocols:**

- Allocate I/N bandwidth to every host
- Share channel efficiently and fairly at high load
- **Inefficient at low load** (where load = # senders):
 - » I/N bandwidth allocated even if only 1 active node!
- E.g., Frequency Division Multiple Access (FDMA); optical networks

- **“Taking turns” protocols:**

- Pass a token around active hosts
- A host can only send data if it has the token
- More efficient at low loads: single node can use \gg I/N bandwidth
- Overhead in acquiring the token
- **Vulnerable to failures** (e.g., failed node or lost token)
- E.g., Token ring

MAC Protocols

Application
Present. Session
Transport Network
Datalink
Physical

- **Random Access**

- Efficient at low load: single node can fully utilize channel
- High load: collision overhead

- Key ideas of random access:

- **Carrier sense (CS)**

- » *Listen before speaking, and don't interrupt*
- » Checking if someone else is already sending data
- » ... and waiting till the other node is done

- **Collision detection (CD)**

- » *If someone else starts talking at the same time, stop*
- » Realizing when two nodes are transmitting at once
- » ...by detecting that the data on the wire is garbled

- **Randomness**

- » *Don't start talking again right away*
- » Waiting for a random time before trying again

- Examples: CSMA/CD, Ethernet, best known implementation

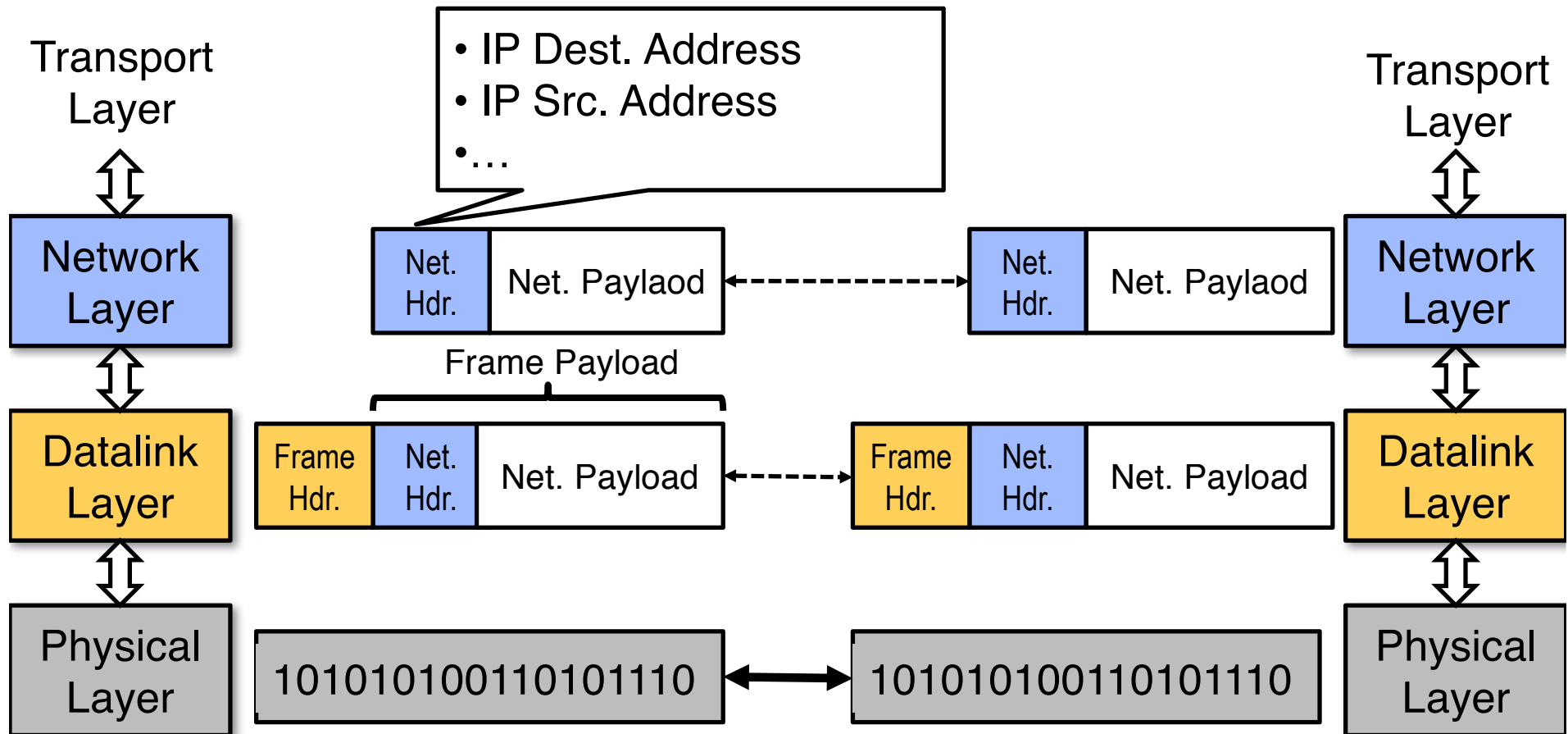
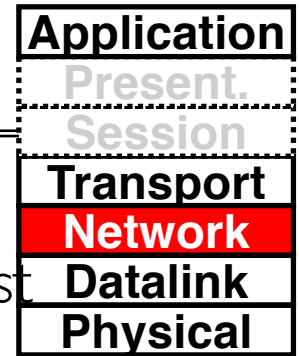
(Inter) Network Layer (3)

Application
Present. Session
Transport
Network
Datalink
Physical

- **Service:**
 - Deliver packets to specified **network addresses** across multiple datalink layer networks
 - Possible other services:
 - » Packet *scheduling/priority*
 - » Buffer management
- **Interface:** send *packets* to specified network address destination; receive packets destined for end host
- **Protocols:** define network addresses (globally unique); construct forwarding tables; packet forwarding

(Inter) Network Layer (3)

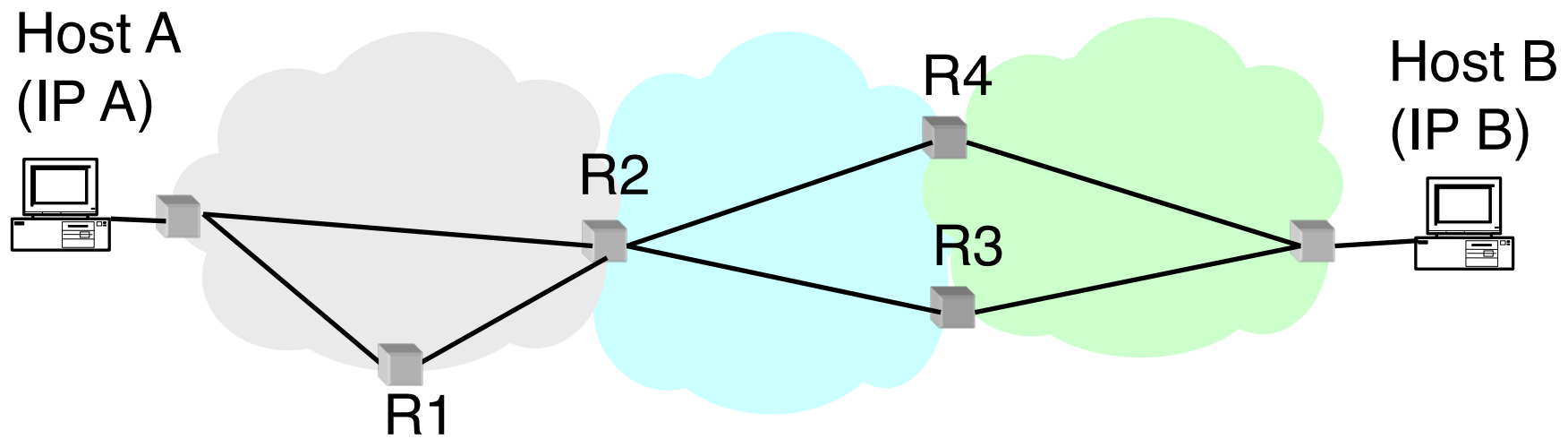
- **IP address:** unique addr. assigned to network device
- Assigned by network administrator or dynamically when host connects to network



Wide Area Network

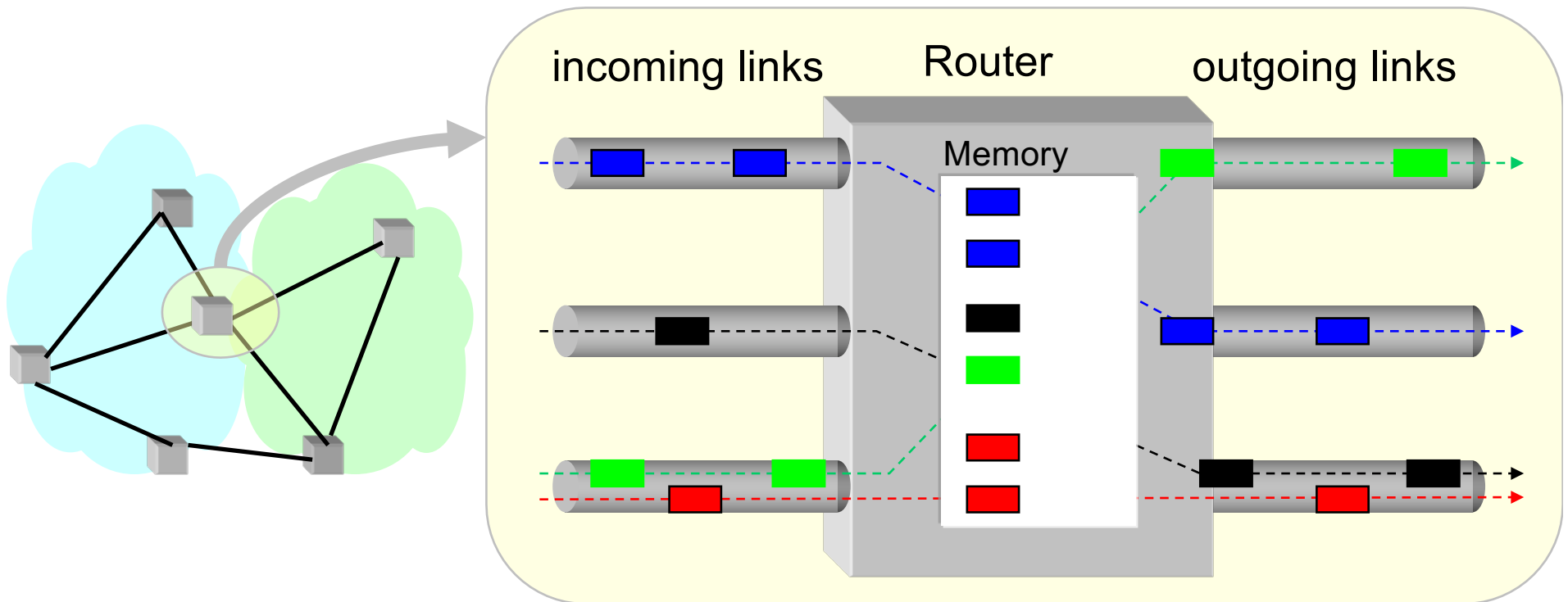
Application
Present. Session
Transport
Network
Datalink
Physical

- **Wide Area Network** (WAN): network that covers a broad area (e.g., city, state, country, entire world)
 - E.g., Internet is a WAN
- WAN connects multiple datalink layer networks (LANs)
- Datalink layer networks are connected by **routers**
 - Different LANs can use different communication technology (e.g., wireless, cellular, optics, wired)



Routers

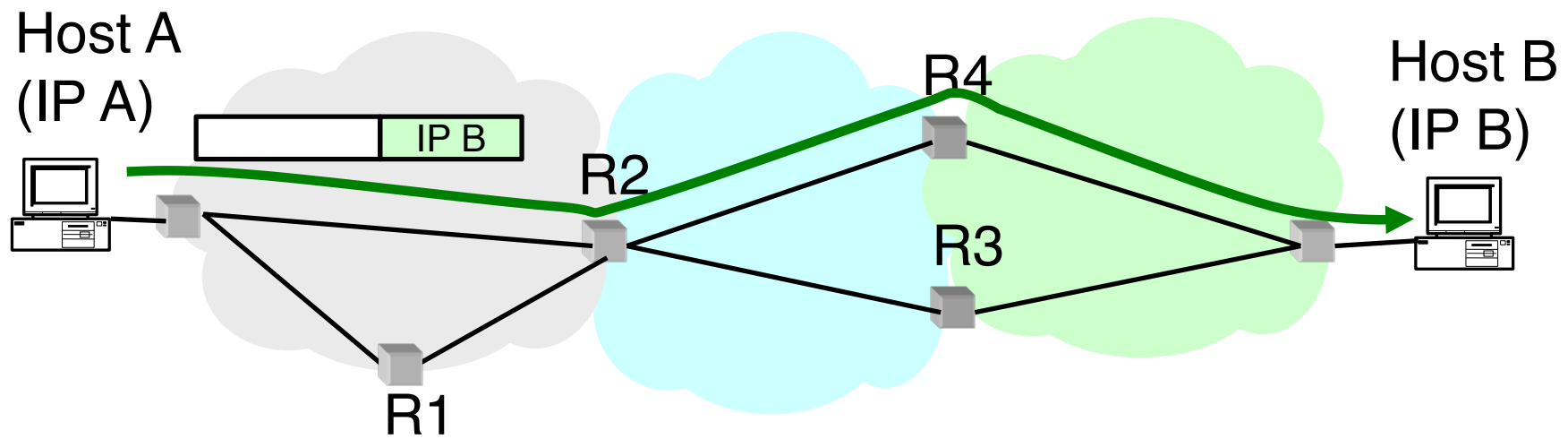
- **Forward** each packet received on an **incoming link** to an **outgoing link** based on packet's destination IP address (towards its destination)
- **Store & forward**: packets are buffered before being forwarded
- **Forwarding table**: mapping between IP address and the output link



Packet Forwarding

Application
Present. Session
Transport
Network
Datalink
Physical

- Upon receiving a packet, a router
 - read the IP destination address of the packet
 - consults its forwarding table → output port
 - forwards packet to corresponding output port



IP Addresses vs. MAC Addresses

Application
Present.
Session
Transport
Network
Datalink
Physical

- Why not use MAC addresses for routing?
 - Doesn't scale
- Analogy
 - MAC address → SSN
 - IP address → (unreadable) home address
- MAC address: uniquely associated to the device for the entire lifetime of the device
- IP address: changes as the device location changes
 - Your notebook IP address at school is different from home



1051 Euclid Ave
Berkeley, CA 94722



10 7th Street NW
Washington, DC 21115

IP Addresses vs. MAC Addresses

Application
Present. Session
Transport
Network
Datalink
Physical

- Why does packet forwarding using IP addr. scale?
- Because IP addresses can be aggregated
 - E.g., all IP addresses at UC Berkeley start with **0xA9E5**, i.e., any address of form 0xA9E5**** belongs to Berkeley
 - Thus, a router in NY needs to keep a **single** entry for **all** hosts at Berkeley
 - If we were using MAC addresses the NY router would need to maintain **an entry for every** Berkeley host!!

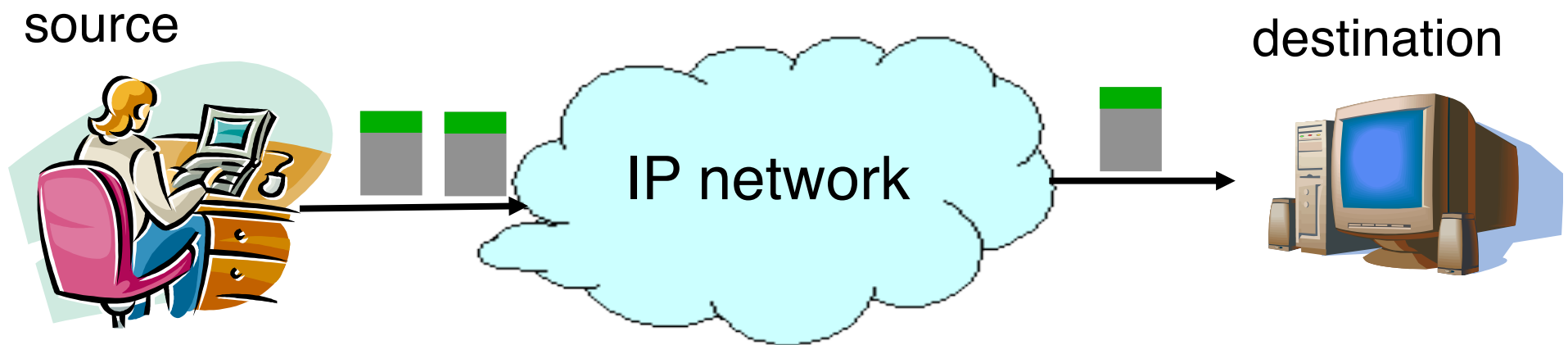


- Analogy:
 - Give this letter to person with SSN: 123-45-6789 vs.
 - Give this letter to “John Smith, 123 First Street, LA, US”

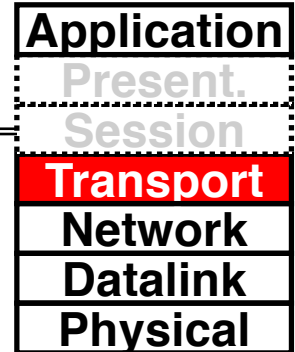
The Internet Protocol (IP)

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 packet to its destination
 - Packets may be lost
 - Packets may be corrupted
 - Packets may be delivered out of order



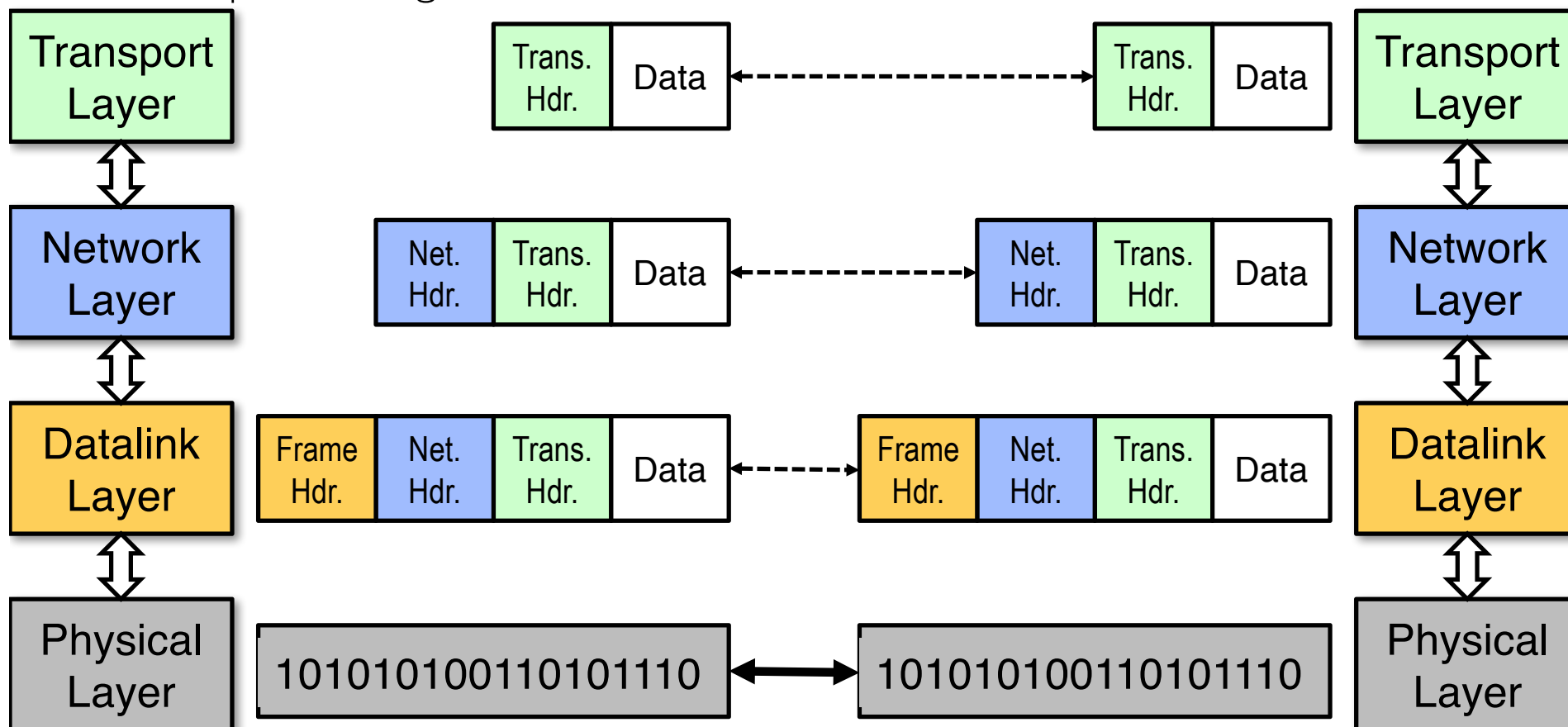
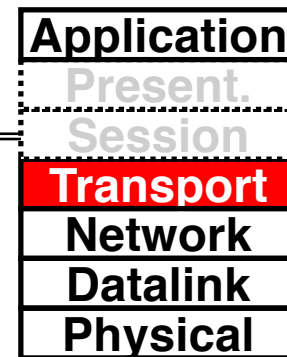
Transport Layer (4)



- **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 adaptation (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
- Examples: TCP and UDP

Port Numbers

- Port number: 16-bit number identifying the end-point of a transport connection
 - E.g., 80 identifies the port on which a processing implementing HTTP server can be connected

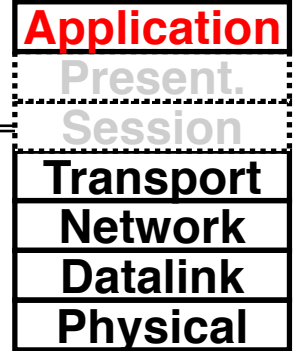


Internet Transport Protocols

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

Application Layer (7 - not 5!)

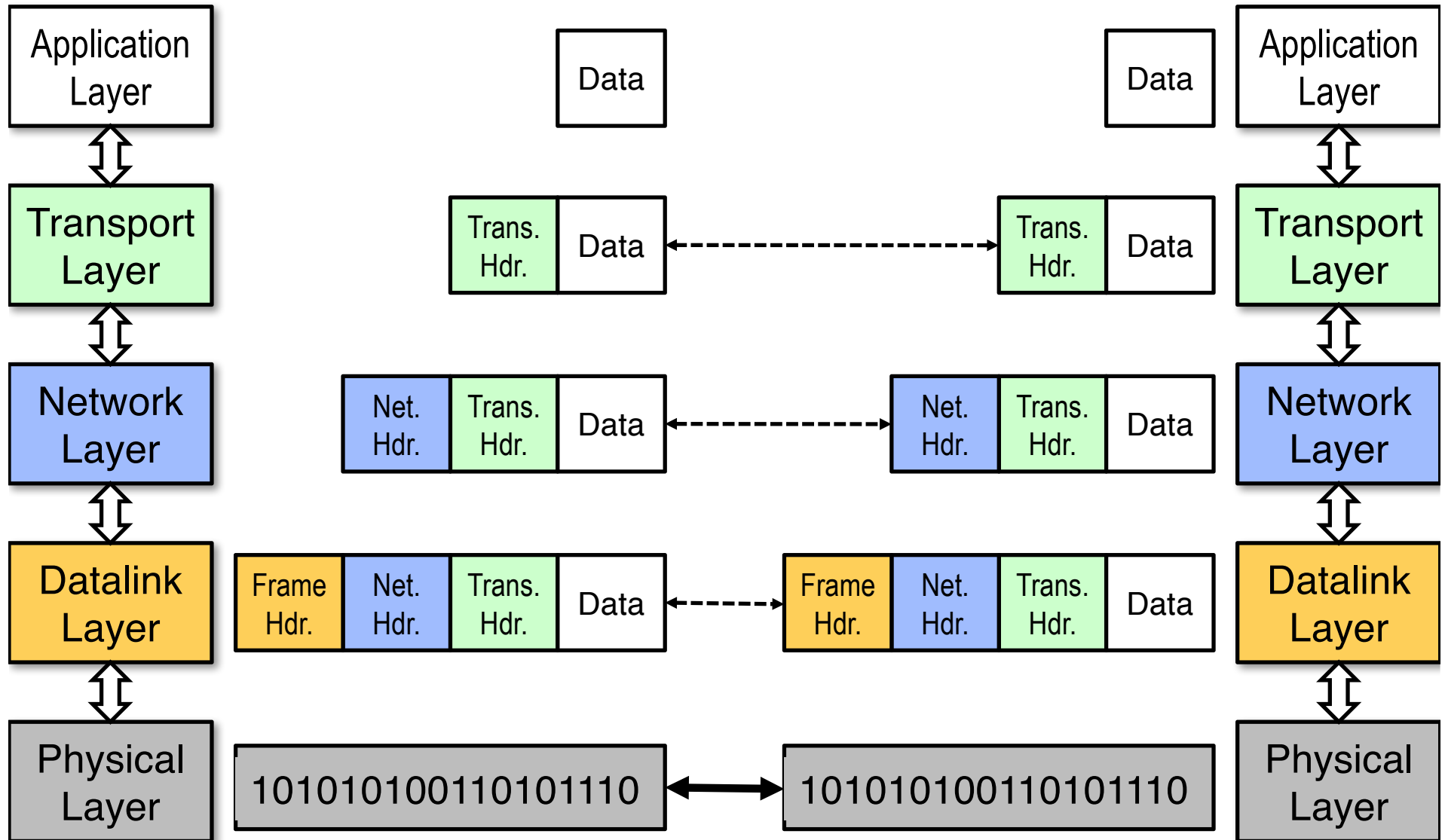


- **Service:** any service provided to the end user
- **Interface:** depends on the application
- **Protocol:** depends on the application

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

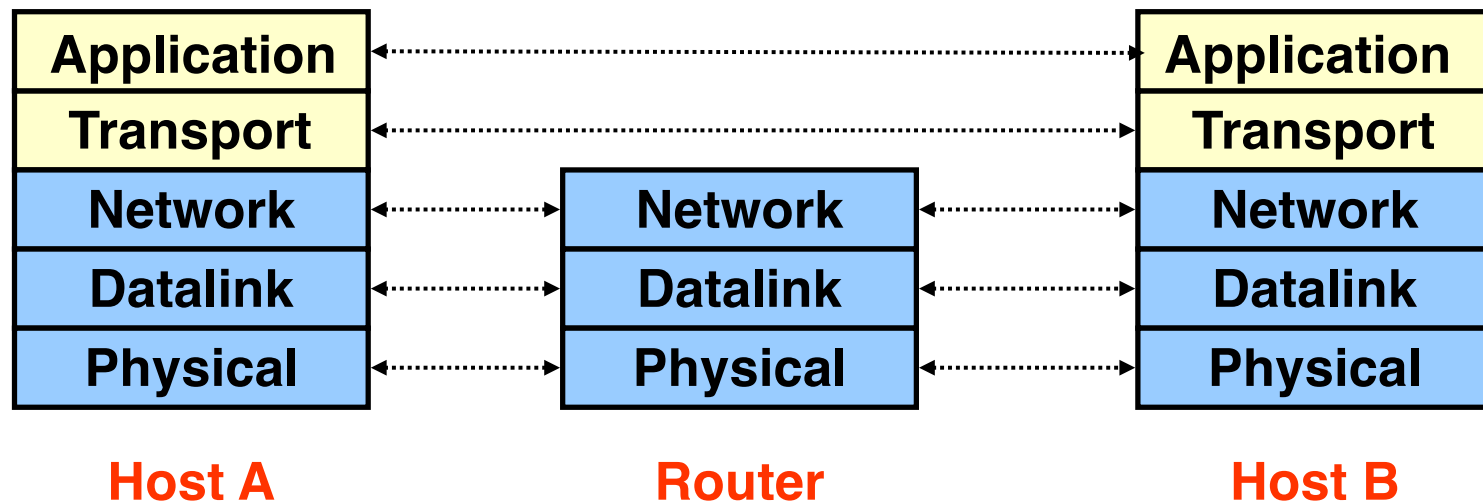
- What happened to layers 5 & 6?
 - “Session” and “Presentation” layers
 - Part of **OSI** architecture, but not Internet architecture
 - Their functionality is provided by application layer

Application Layer (5)



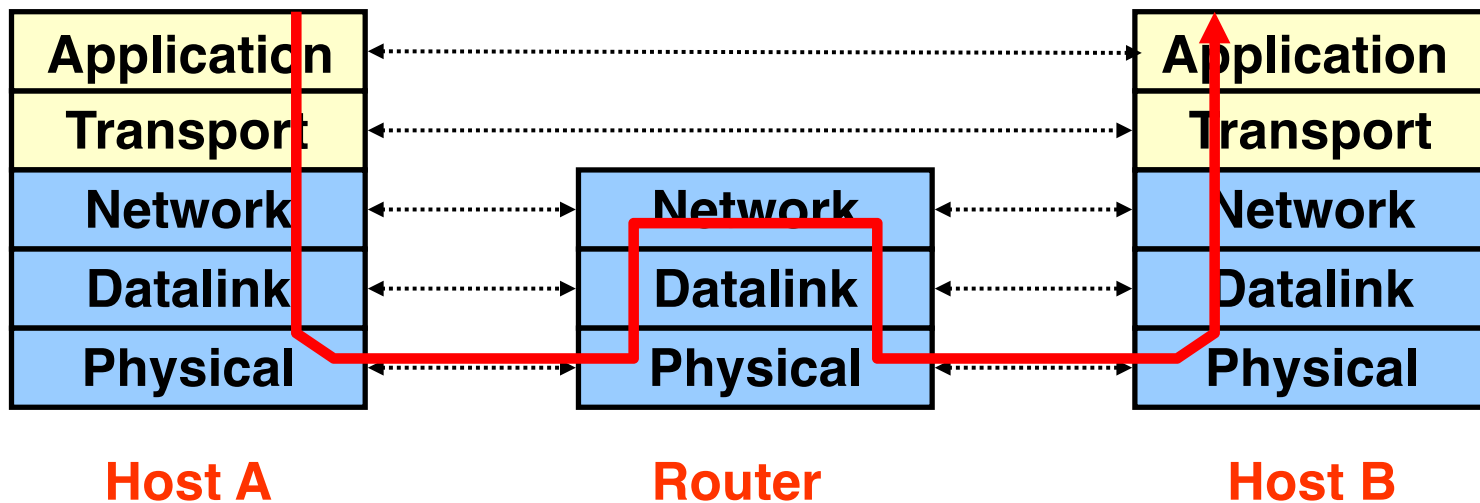
Five Layers Summary

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



Physical Communication

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

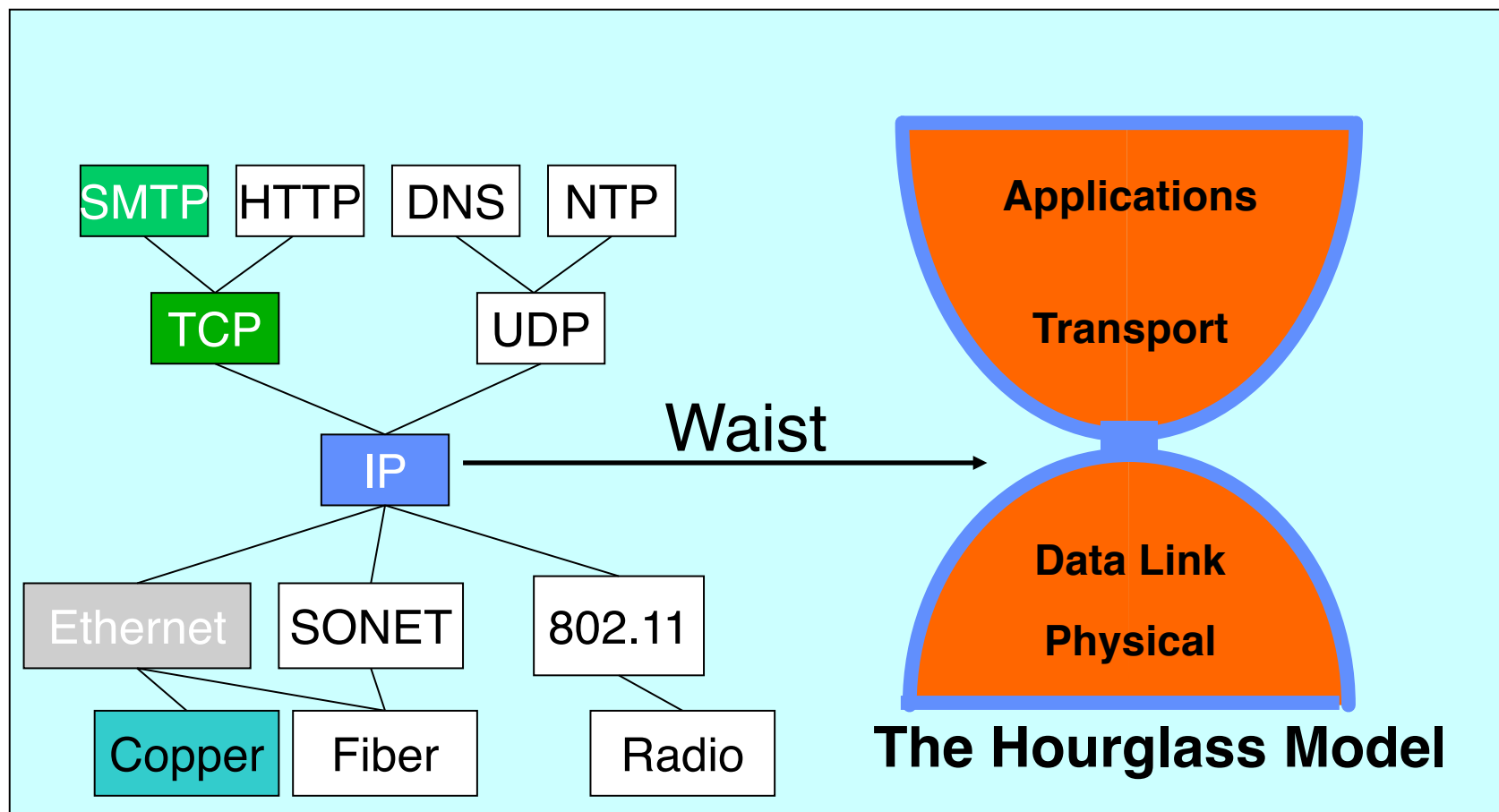


Administrivia

- Midterm 3 coming up on **Wen 11/28 5:00-6:30PM**
 - All topics:
 - » Focus will be on Lectures 18 – 23 and associated readings, and Projects 3
 - » But expect 20-30% questions from materials from Lectures 1-17
 - Closed book
 - 2 pages hand-written notes both sides

BREAK

The Internet *Hourglass*



There is just **one** network-layer protocol, **IP**.
The “narrow waist” facilitates **interoperability**.

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
 - But changing IP itself, i.e., **IPv6**, very involved

Drawbacks of Layering

- Layer N may duplicate layer N-1 functionality
 - E.g., error recovery to retransmit lost data
- Layers may need same information
 - E.g., timestamps, maximum transmission unit size
- Layering can hurt performance
 - E.g., hiding details about what is really going on
- Some layers are not always cleanly separated
 - Inter-layer dependencies for performance reasons
 - Some dependencies in standards (header checksums)
- Headers start to get really big
 - Sometimes header bytes >> actual content

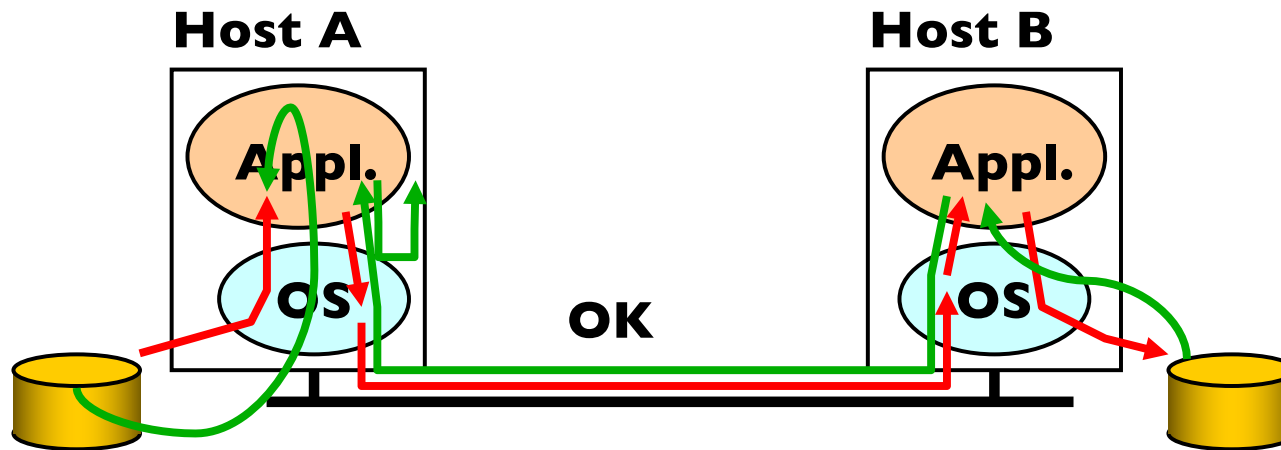
Placing Network Functionality

- Hugely influential paper: “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark (‘84)
- “Sacred Text” of the Internet
 - Endless disputes about what it means
 - Everyone cites it as supporting their position

Basic Observation

- Some types of network functionality can only be correctly implemented **end-to-end**
 - Reliability, security, etc
- Because of this, end hosts:
 - Can satisfy the requirement without network's help
 - Will/**must** do so, since can't **rely** on network's help
- Therefore **don't** go out of your way to implement them in the network

Example: Reliable File Transfer



- Solution 1: make each step reliable, and then **concatenate** them
- Solution 2: end-to-end **check** and try again if necessary

Discussion

- Solution 1 is **incomplete**
 - What happens if memory is corrupted?
 - Receiver has to do the check anyway!
- Solution 2 is **complete**
 - Full functionality can be entirely implemented at application layer with **no** need for reliability from lower layers
- *Is there any need to implement reliability at lower layers?*
 - Well, it could be **more efficient**

End-to-End Principle

Implementing this functionality in the network:

- Doesn't reduce host implementation complexity
- Does increase network complexity
- Probably imposes delay and overhead on all applications, **even if they don't need functionality**
- However, implementing in network **can** enhance performance in some cases
 - E.g., very lossy link

Conservative Interpretation of E2E

- Don't implement a function at the lower levels of the system unless it can be completely implemented at this level
- Unless you can relieve the burden from hosts, don't bother

Moderate Interpretation

- Think twice before implementing functionality in the network
- If hosts can implement functionality correctly, implement it in a lower layer **only** as a performance enhancement
- But do so only if it **does not impose burden** on applications that do not require that functionality
- This is the interpretation we are using

Summary (1/2)

- Layered architecture powerful abstraction for organizing complex networks
- Internet: 5 layers
 - Physical: send bits
 - Datalink: Connect two hosts on same physical media
 - Network: Connect two hosts in a wide area network
 - Transport: Connect two processes on (remote) hosts
 - Applications: Enable applications running on remote hosts to interact
- Unified Internet layering (Application/Transport/Internetwork/Link/Physical) decouples apps from networking technologies

Summary (2/2)

- E2E argument encourages us to keep IP simple
- If higher layer can implement functionality correctly, implement it in a lower layer **only** if
 - it improves the performance significantly for application that need that functionality, and
 - it **does not impose burden** on applications that do not require that functionality