https://hao-ai-lab.github.io/dsc204a-w24/

DSC 204A: Scalable Data Systems
Winter 2024

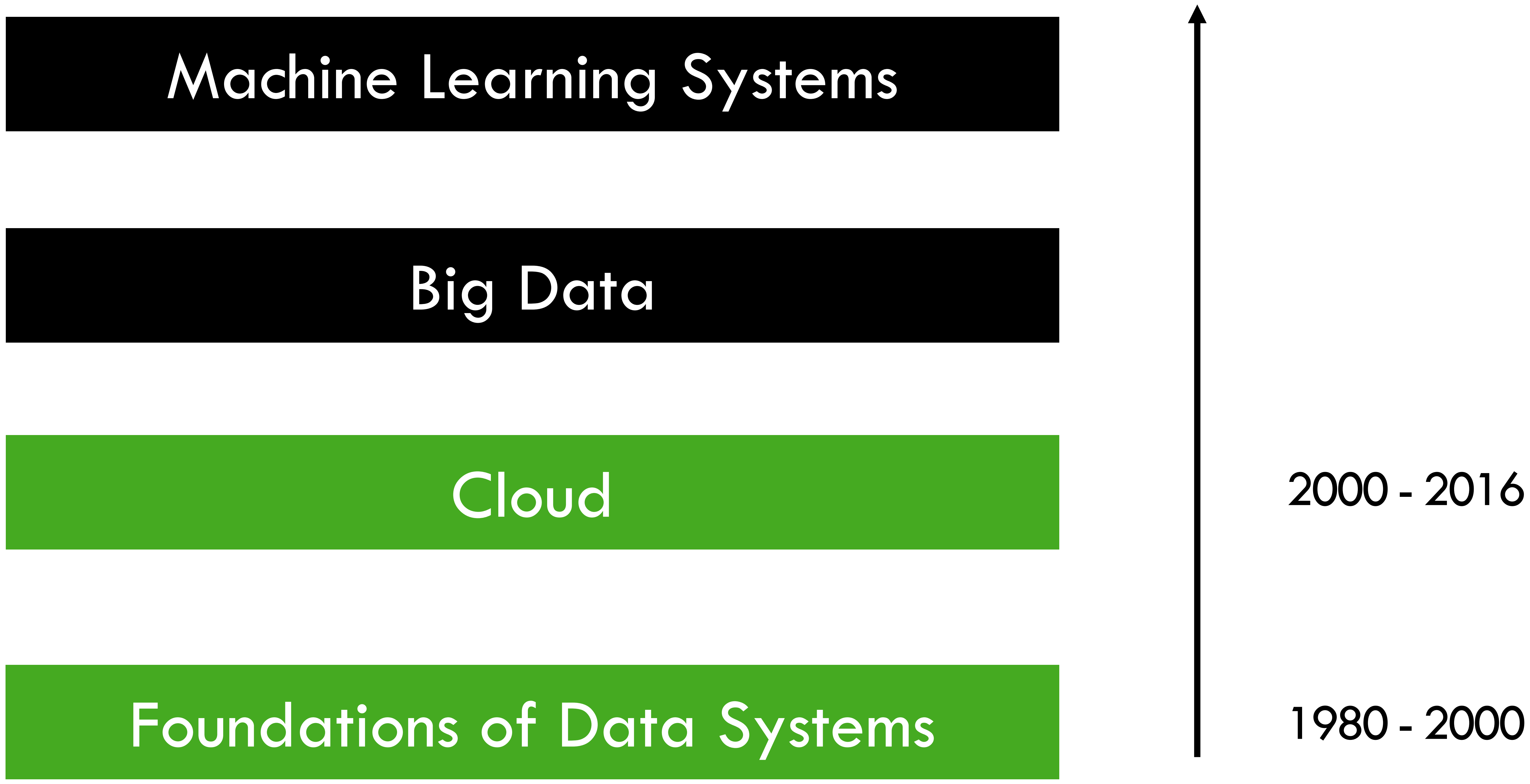
Machine Learning Systems

Big Data

Cloud

Foundations of Data Systems

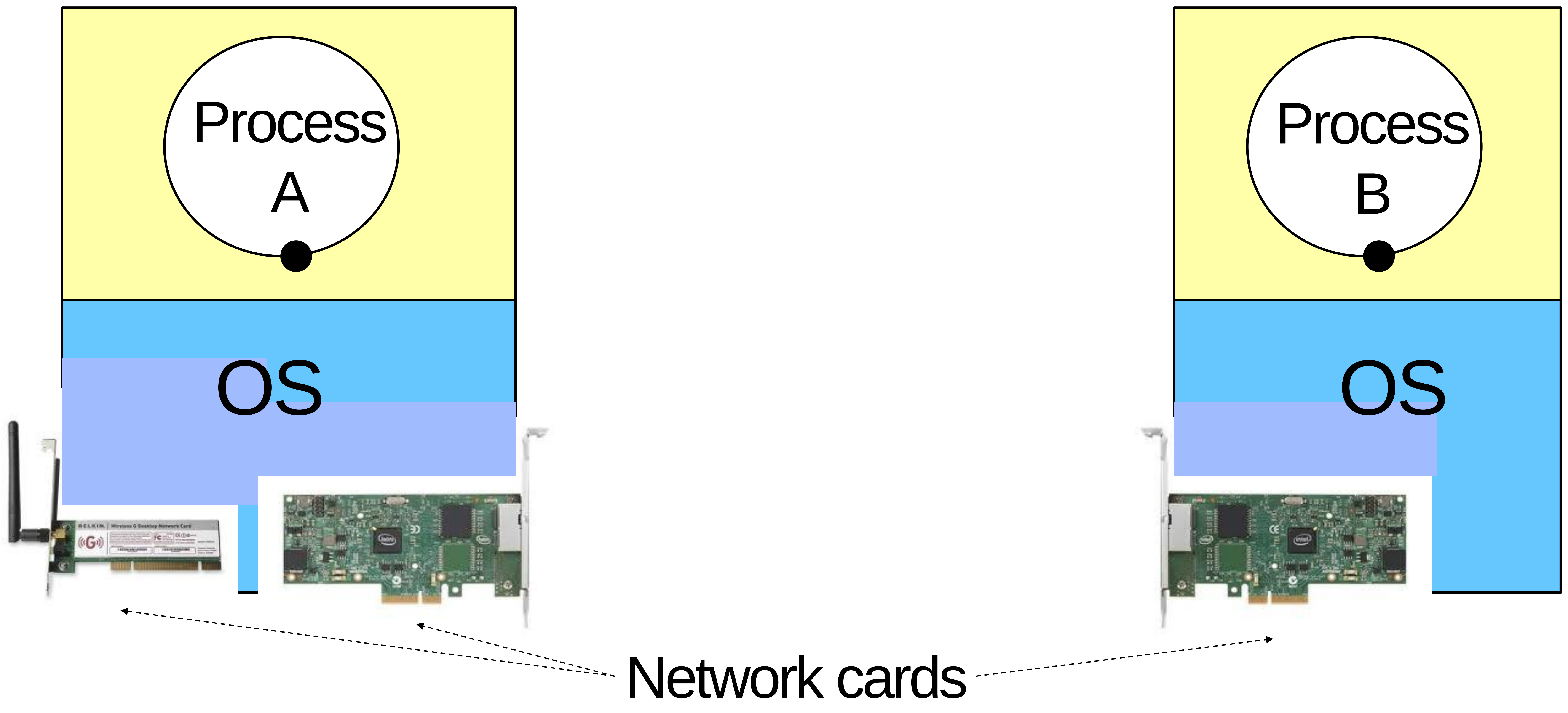
Where We Are


Machine Learning Systems
Big Data
Cloud
2000 - 2016
Foundations of Data Systems
1980 - 2000

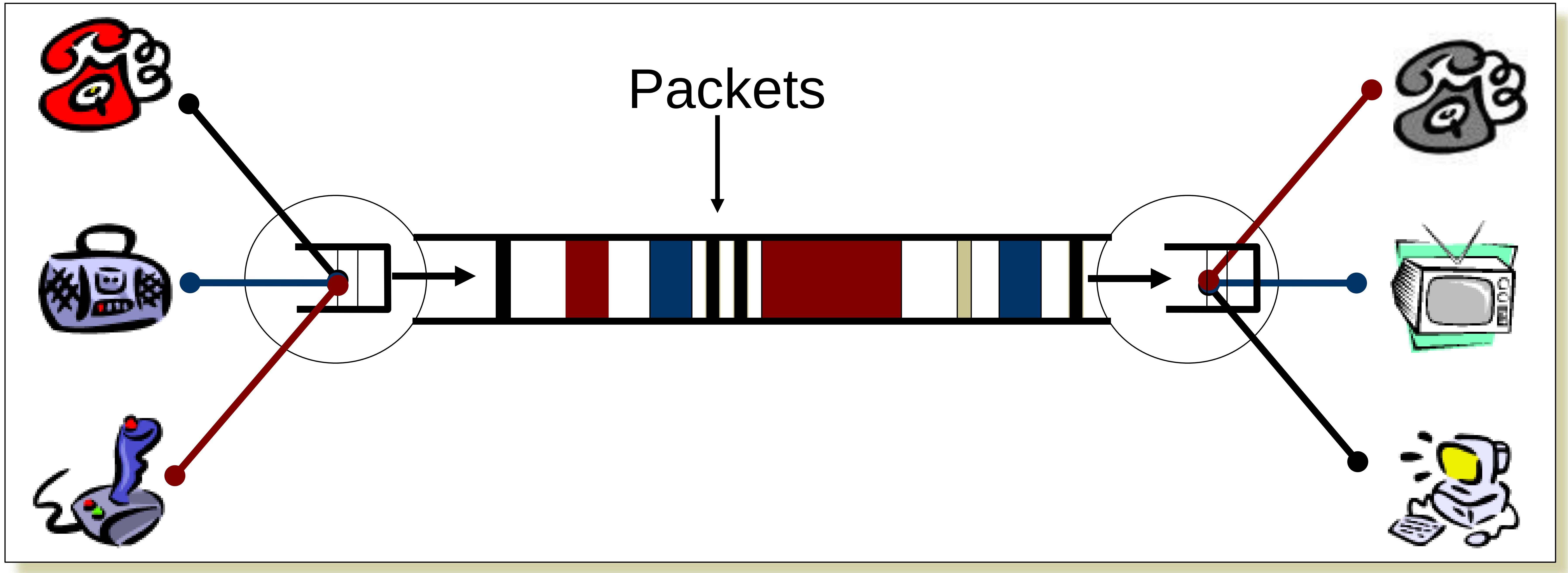
Today’s topic

- Network Basics
- **Layering and protocols**
- Collective communication

Recap: Network Hardware



Recap: Packet Switch



The Problem

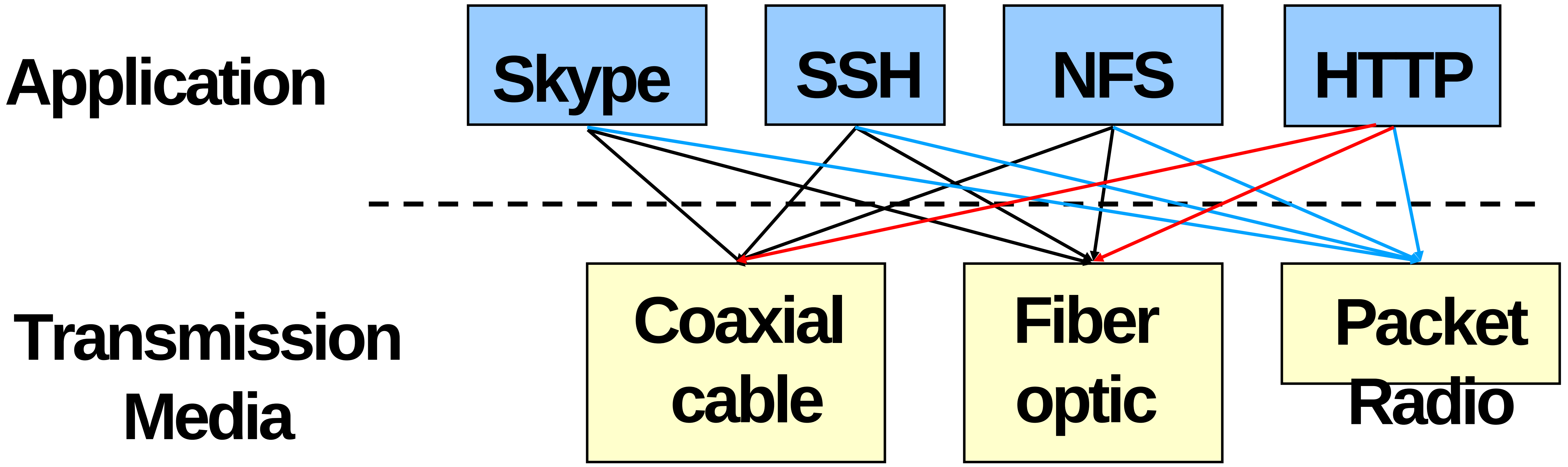
- Many different applications
  - email, web, P2P, etc.

- Many different network styles and technologies
  - Wireless vs. wired vs. optical, etc.

- How do we organize this mess?

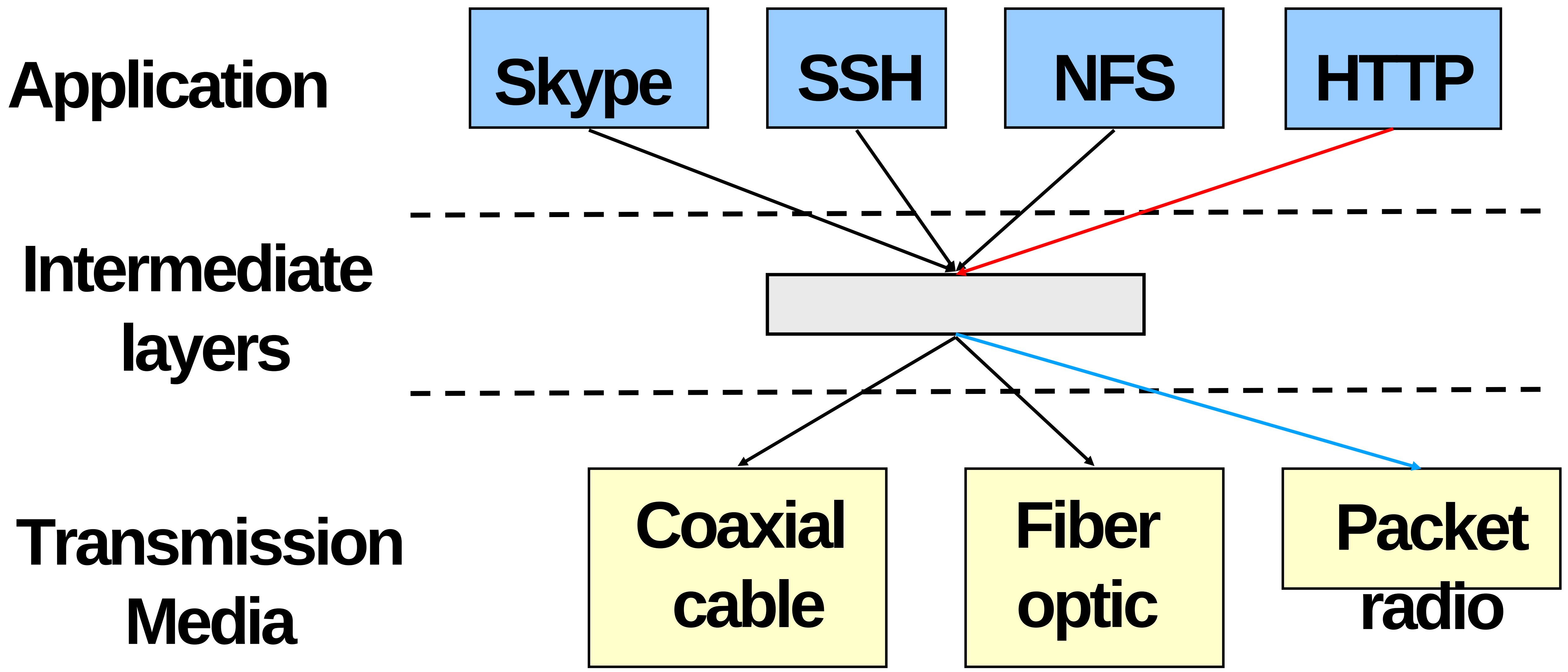
The Problem (cont'd)

- 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"



Goal of This Intermediate Layer

- **Delivery**: deliver packets between to any host in the Internet
  - E.g., deliver a packet from a host in UCSD 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 loses?
- **Flow control**: avoid overflowing the receiver buffer
  - E.g., how do you ensure that a server that can send at 10Gbps doesn’t overwhelm a LTE phone?
- **Congestion control**: avoid overflowing the buffer of a router along the path
  - What happens if we don’t do it?

How to achieve this? Building software for networks

Partition complex system into modules & abstractions (microservices)

- E.g., libraries encapsulating set of functionality
- E.g., OS: system calls -> programming language -> assembly -> processor instructions
- E.g., Cloud microservices: storage, compute, security, networking services.

- Well-defined interfaces hide information
  - Present high-level abstractions
  - **But can impair performance**

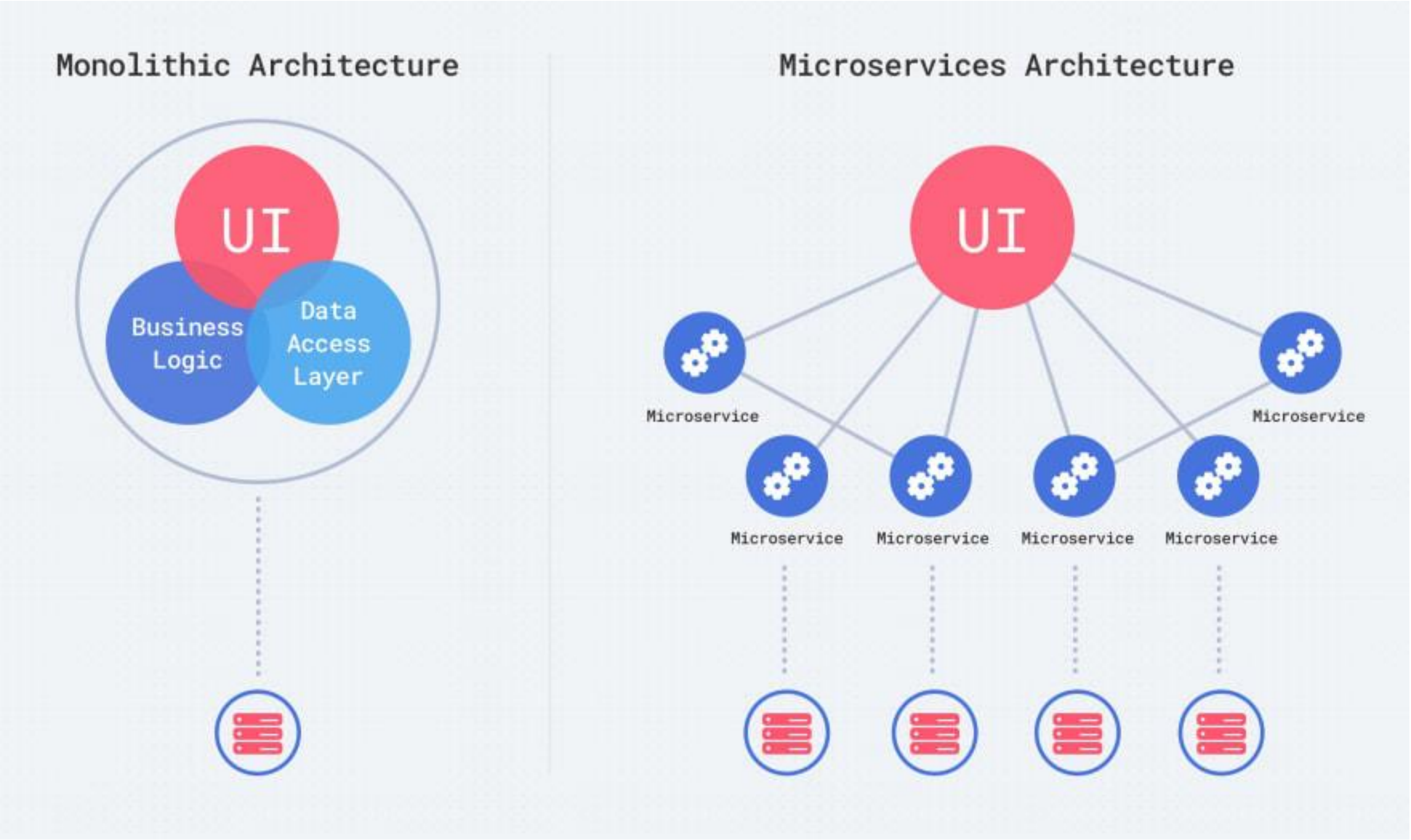
Network's Software: Layers

Like software modularity, but:

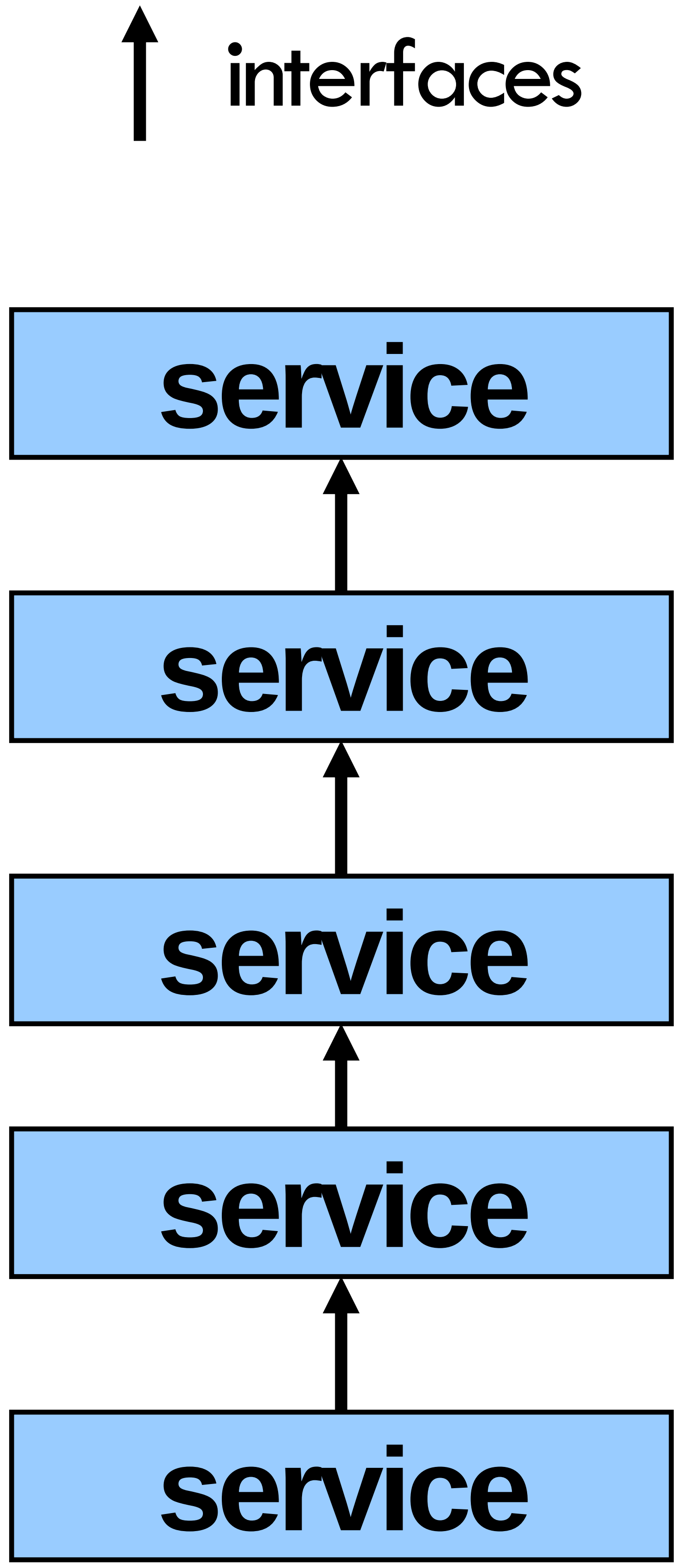
- Implementation distributed across many machines (routers and hosts)

- Must decide:
  - How to break system into modules (layers):
    - **Layering**
  - What functionality does each module implement:
    - **End-to-End Principle:** don't put it in the network if you can do it in the endpoints.

Layers == Microsevices architecture s.t. constraints



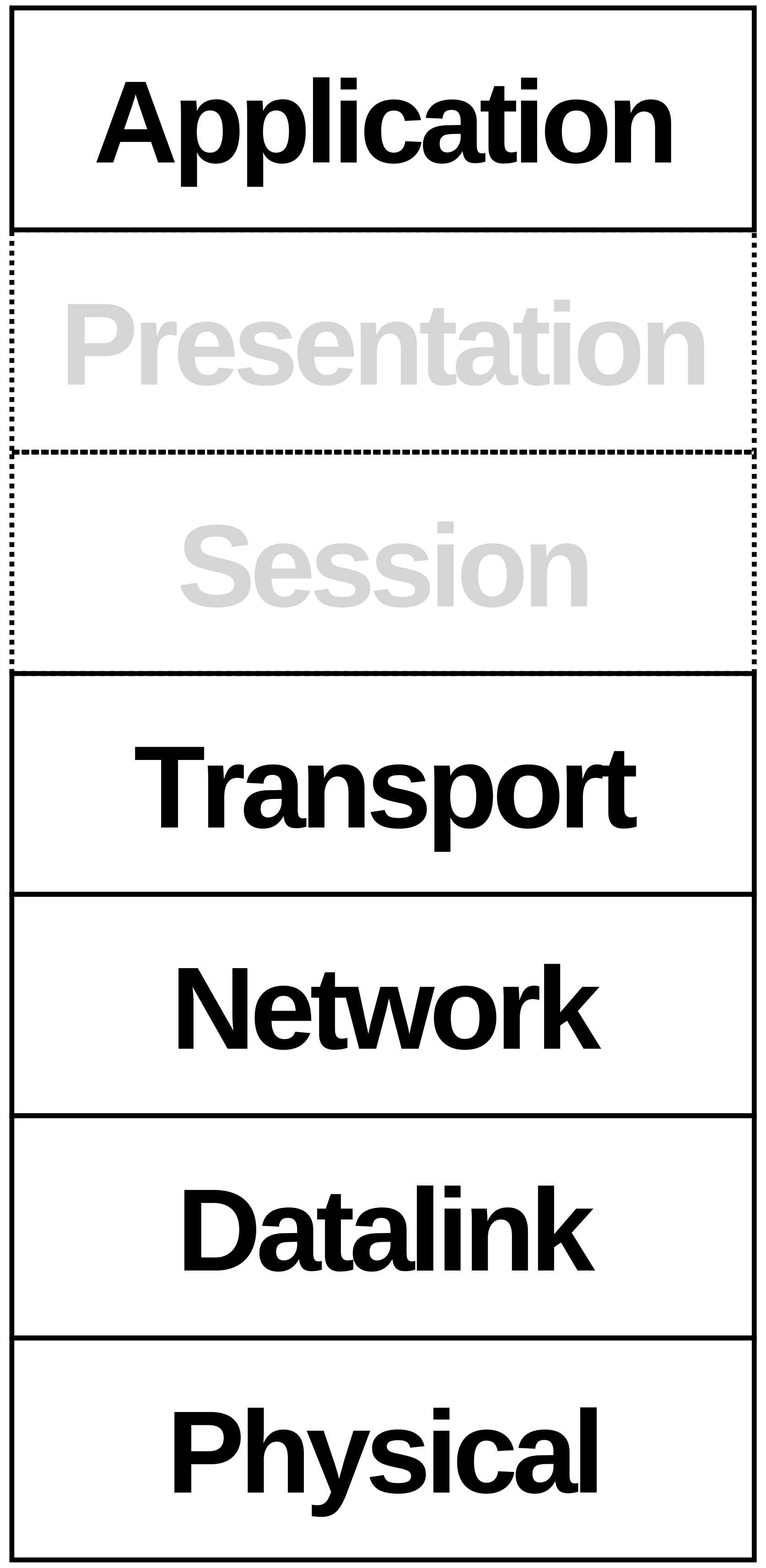
- Each layer solely relies on services from layer below
- Each layer solely exports services to layer above
- Hides implementation details
- Layers can change without disturbing other layers



Properties of Layers (we will see them for each layer)

- **Service**: what a layer does
- **Service interface**: how to access the service
  - Interface for layer above
- **Protocol** (*peer interface*): how peers communicate to achieve 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

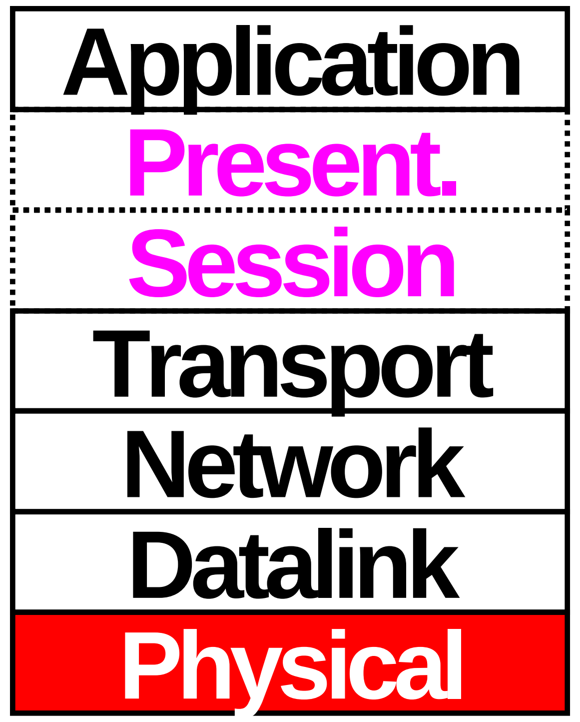
OSI Layering Model



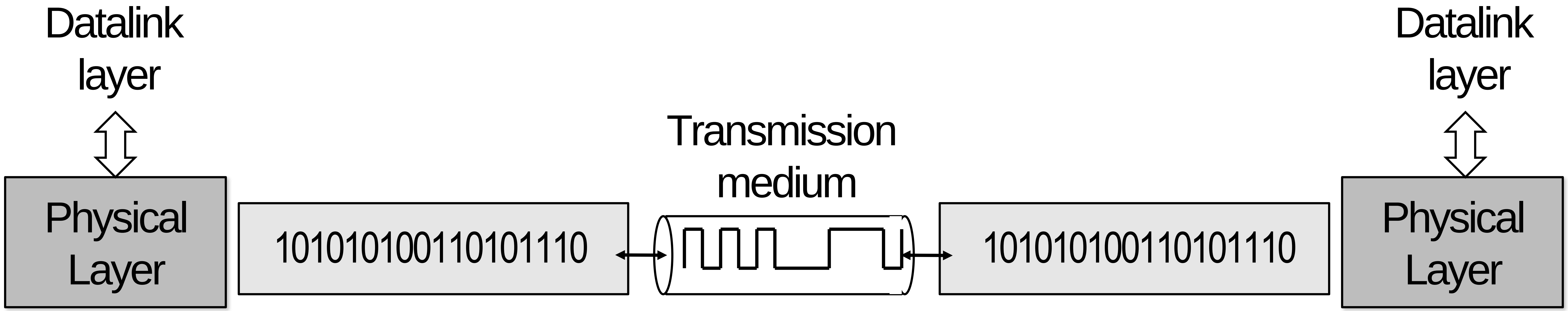
- Open Systems Interconnection (OSI) model
  - Developed by International Organization for Standardization (ISO) in 1984
  - **Seven** layers
- Internet Protocol (IP)
  - Only **five** layers
  - The functionalities of the missing layers (i.e., Presentation and Session) are provided by the Application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

Physical Layer (1)

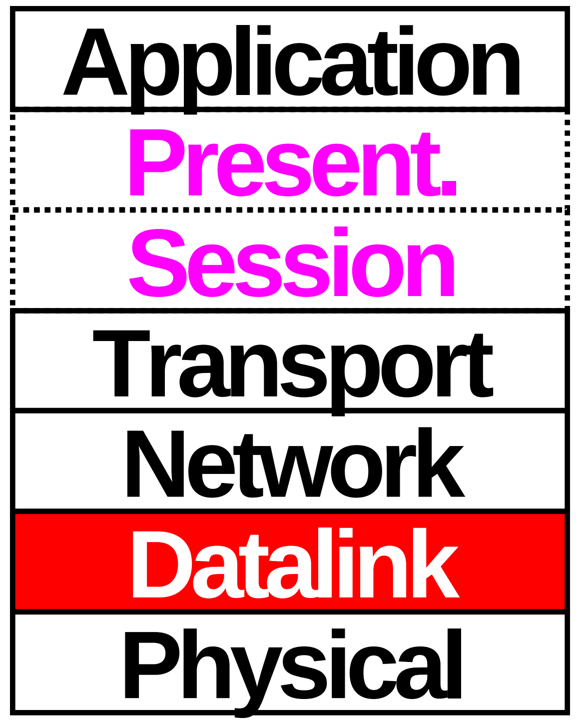




- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send and receive bits
- **Protocol**: coding scheme used to represent a bit, voltage levels, duration of a bit
- Examples: coaxial cable, optical fiber links; transmitters, receivers

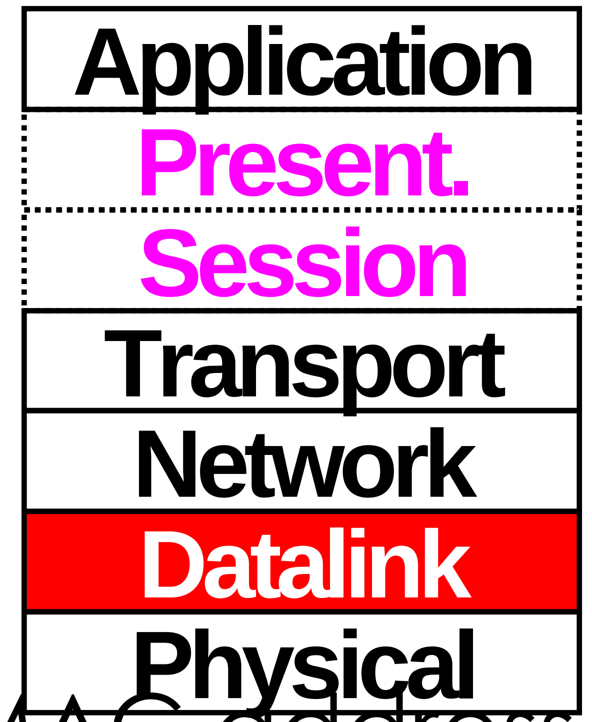


Datalink Layer (2)

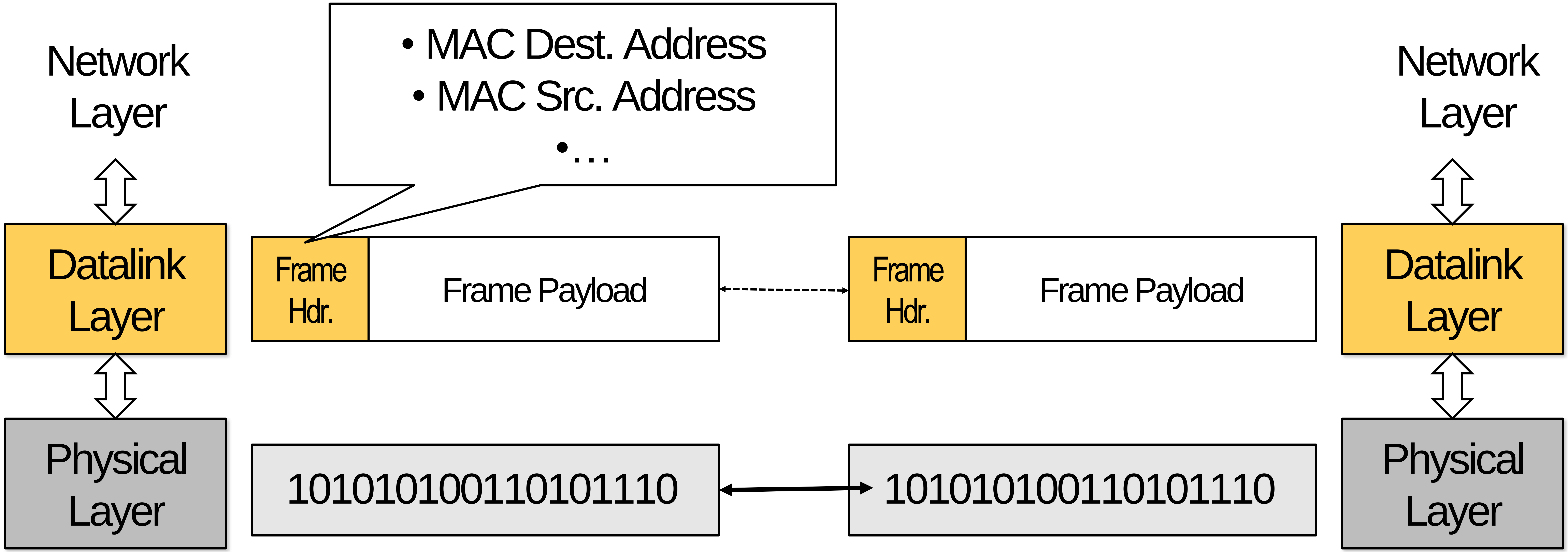


- **Service**:
  - Enable end hosts to exchange frames (atomic messages) on the same physical line or wireless link
  - Possible other services:
    - Arbitrate access to common physical media
    - May provide reliable transmission, flow control
- **Interface**: send *frames* to other end *hosts*; receive *frames* addressed to end host
- **Protocols**: addressing, Media Access Control (MAC) (e.g., CSMA/CD - *Carrier Sense Multiple Access / Collision Detection*)

Datalink Layer (2)



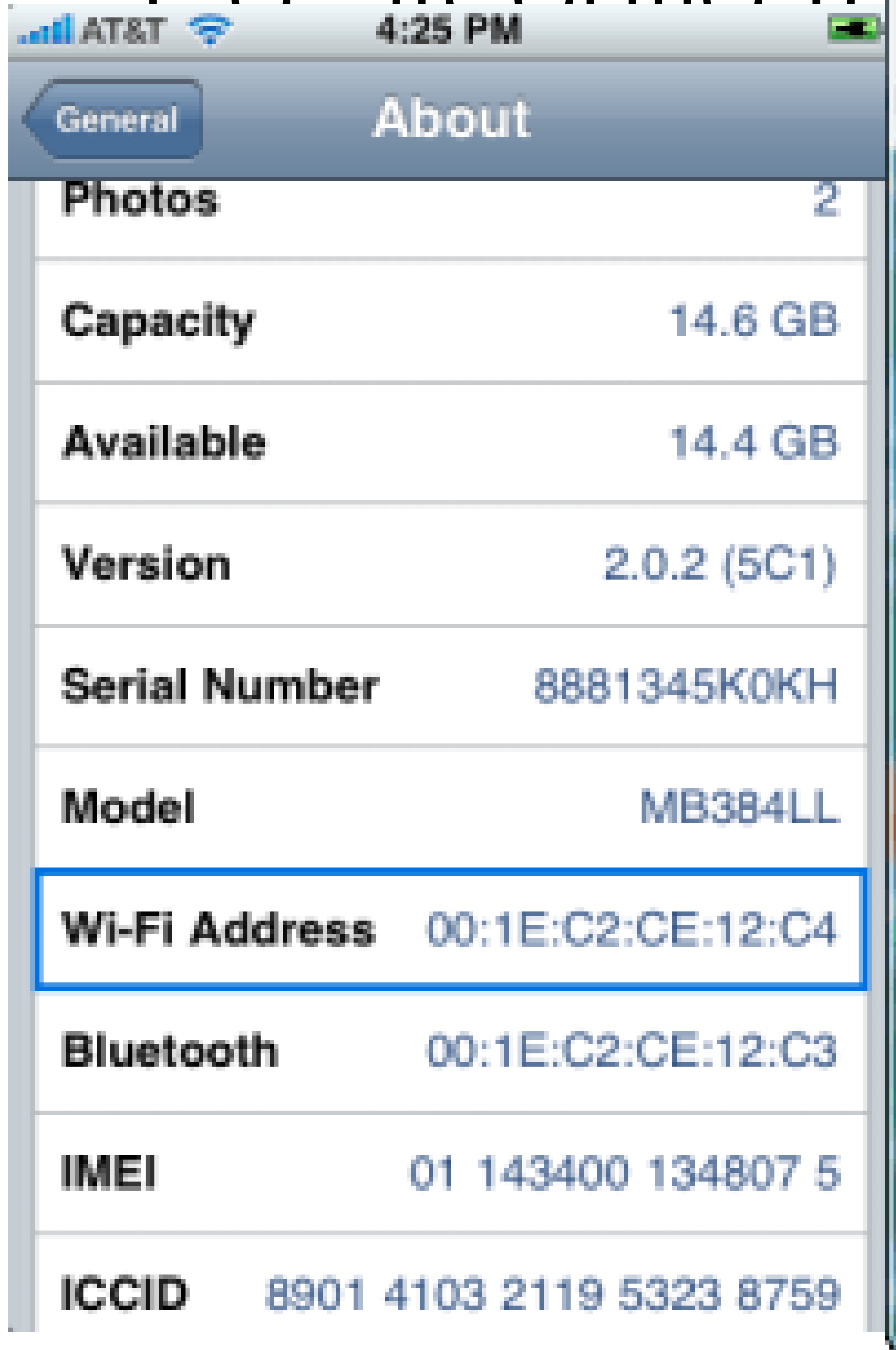
- Each frame has a header which contains a source and a destination MAC address
- MAC (Media Access Control) address
  - Uniquely identifies a network interface
  - 48-bit, assigned by the device manufacturer

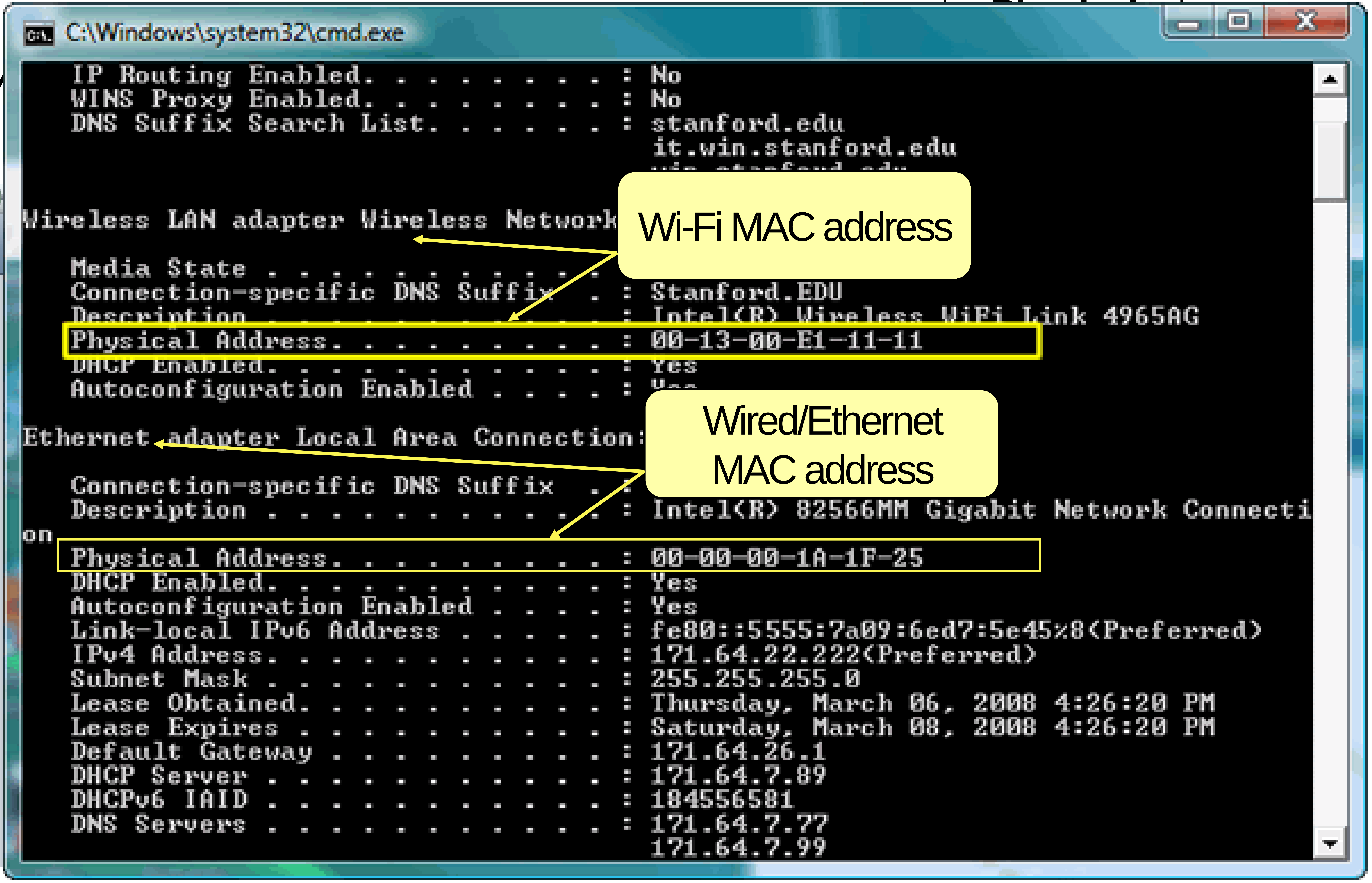


Application
Present.
Session
Transport
Network
Datalink

MAC Address Examples

- Can easily find M
  - E.g., ifconfig (I



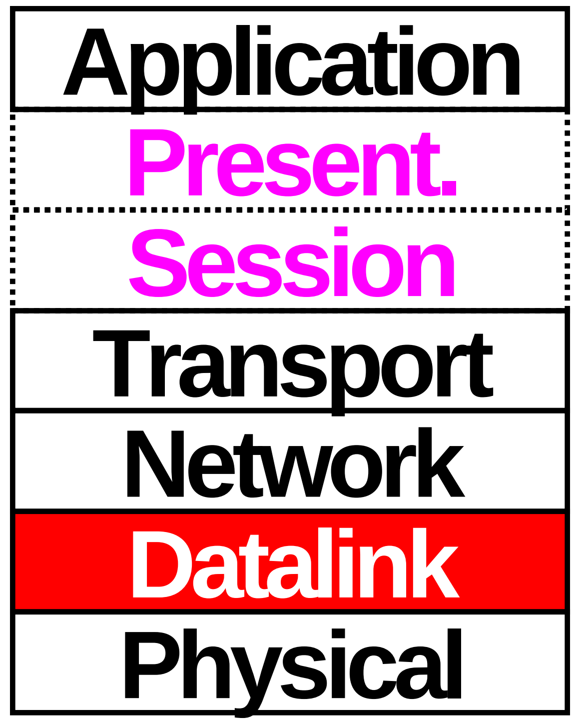


Application
Present.
Session
Transport
Network
Datalink
Physical

Local Area Networks (LANs)

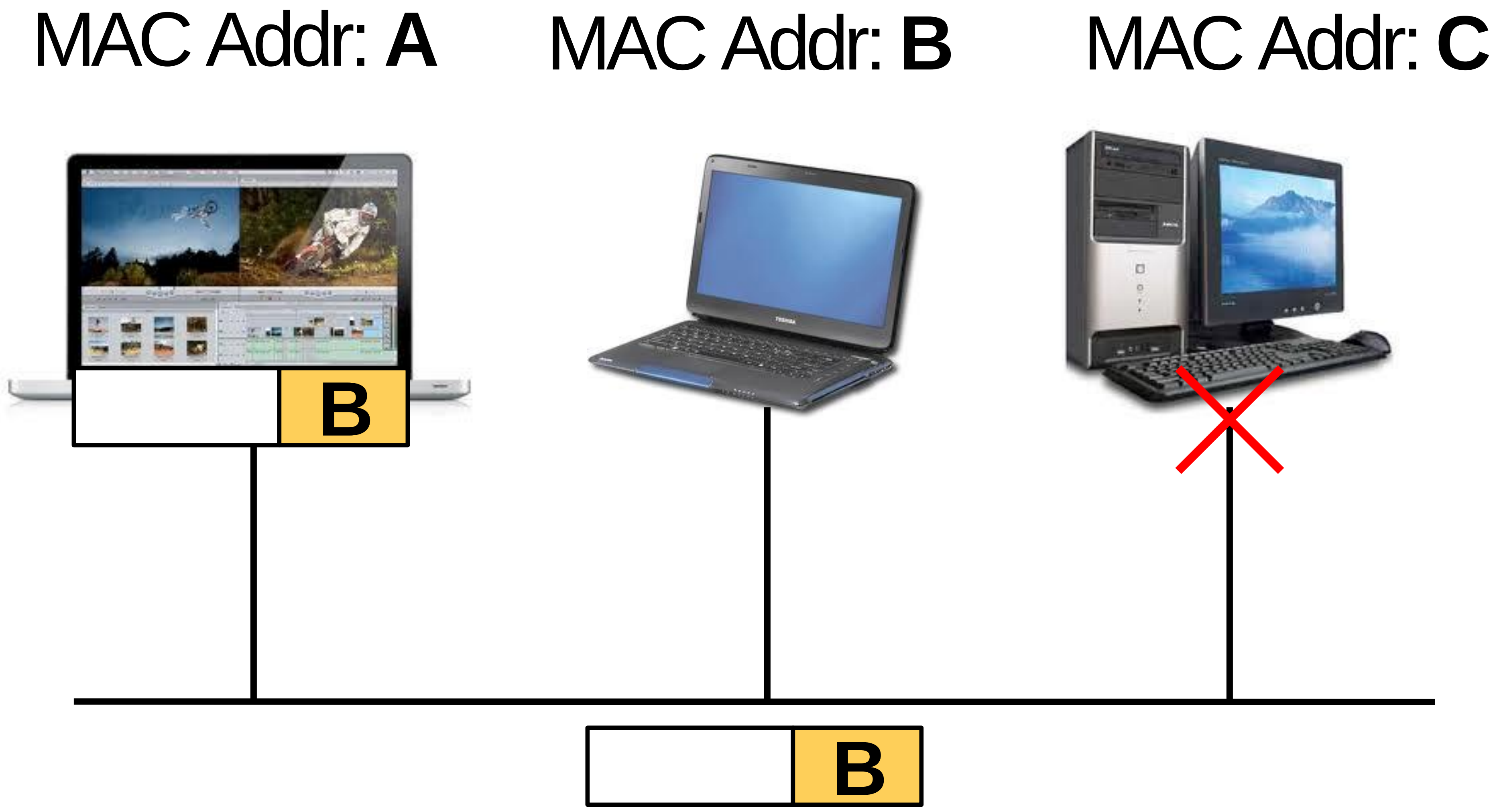
- LAN: group of hosts/devices that
  - are in the same geographical proximity (e.g., same building, room)
  - use same physical communication technology
- Examples:
  - all laptops connected wirelessly at a Starbucks café
  - all devices and computers at home
  - all hosts connected to wired Ethernet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

Ethernet cable and port



LANs

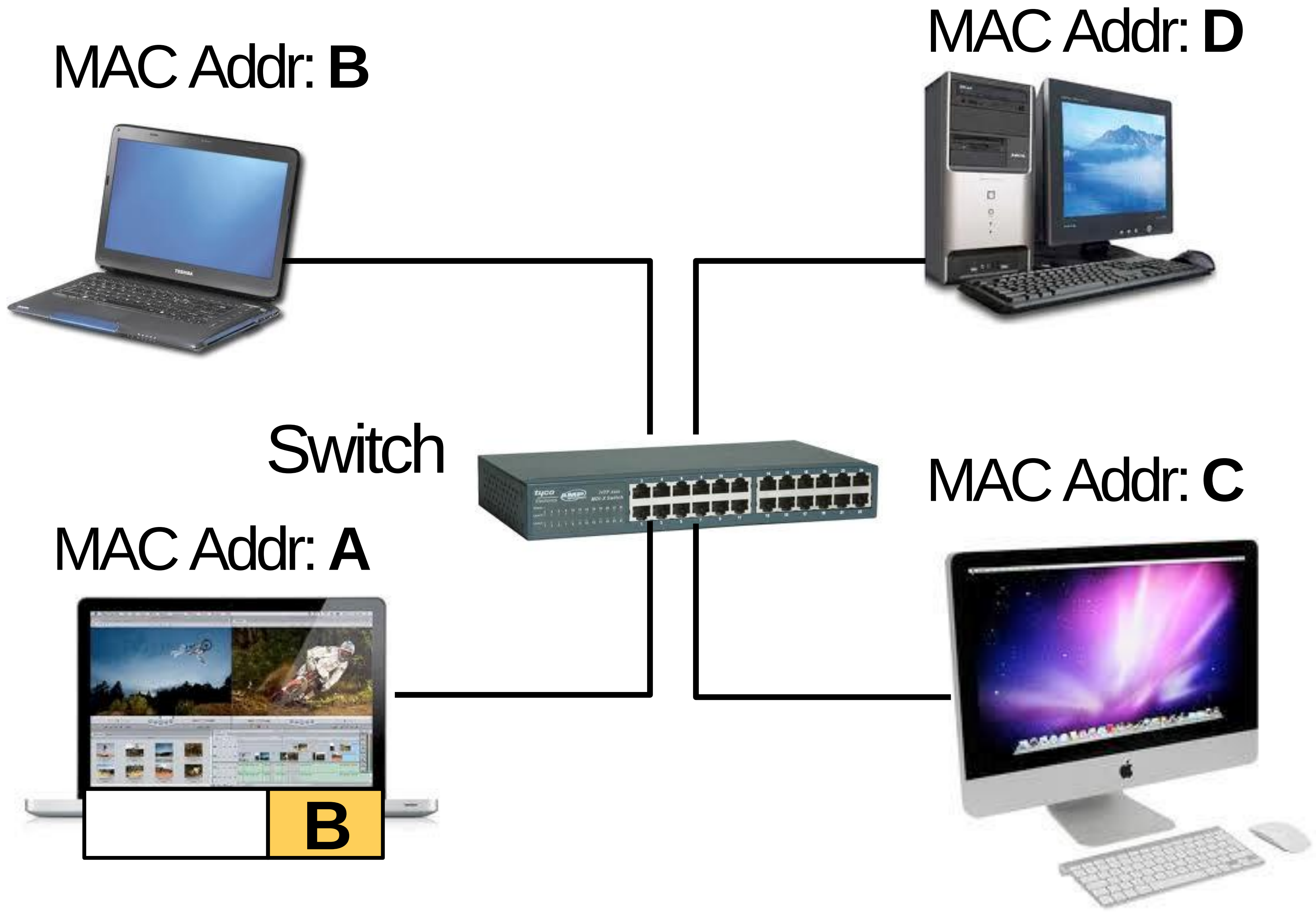
- All hosts in a LAN can share same physical communication media
  - Also called, broadcast channel
- Each frame is delivered to every host
  - "Hubs" forward from one wire to all the others
- If a host is not the intended recipient, it drops the frame

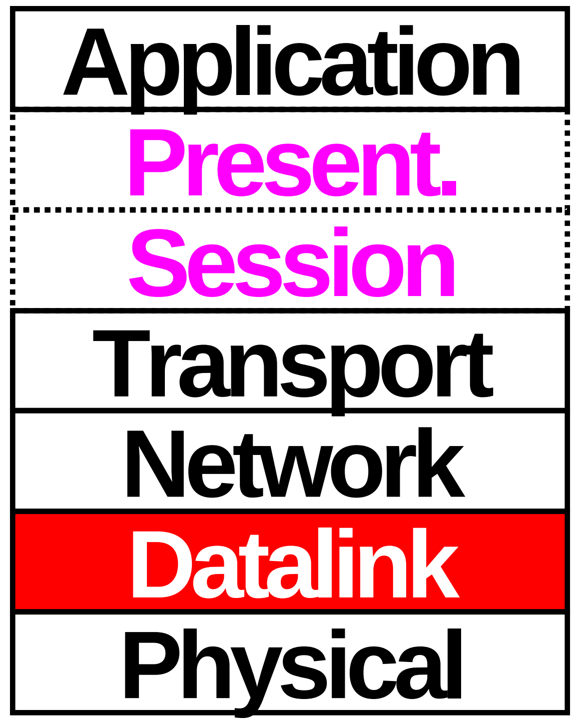


Application
Present.
Session
Transport
Network
Datalink
Physical

Switches

- 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

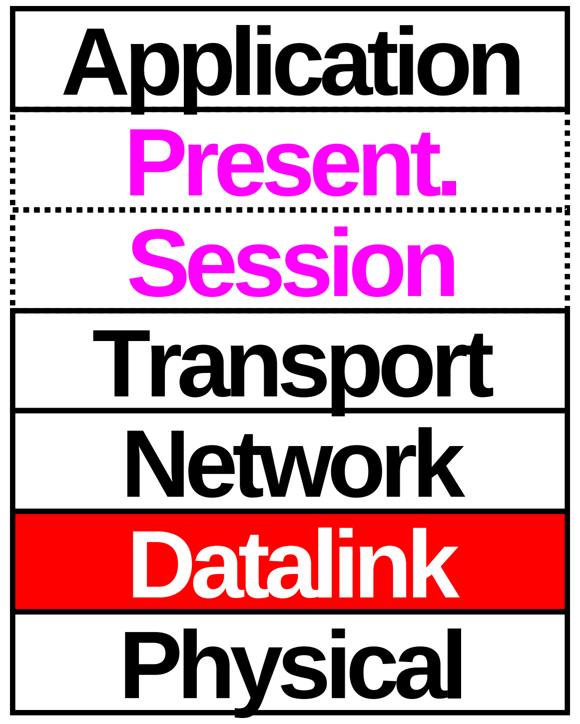
- Problem:
  - How do hosts access a broadcast media?
  - How do they avoid collisions?

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

Application
Present.
Session
Transport
Network
Datalink
Physical

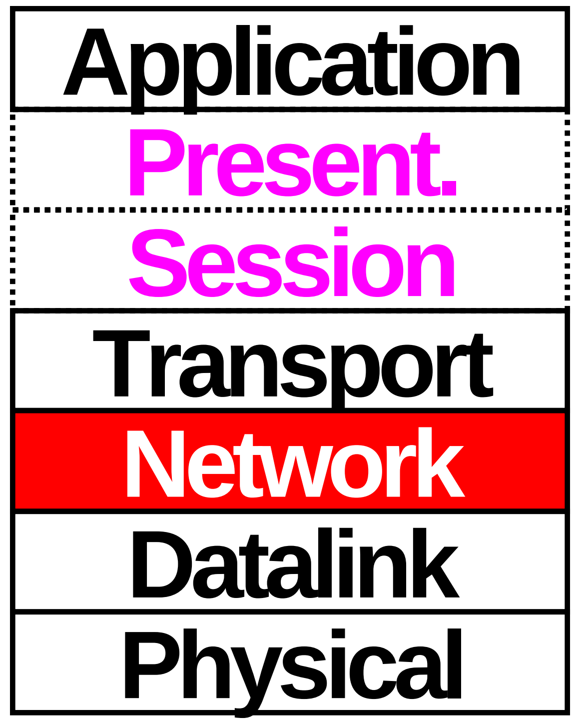
MAC Protocols

- **Channel partitioning protocols:**
  - Allocate 1/N bandwidth to every host
  - Share channel efficiently and fairly at high load
  - Inefficient at low load (where load = # senders):
    - 1/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 >> 1/N banwidth
  - Overhead in acquiring the token
  - Vulnerable to failures (e.g., failed node or lost token)
  - E.g., Token ring



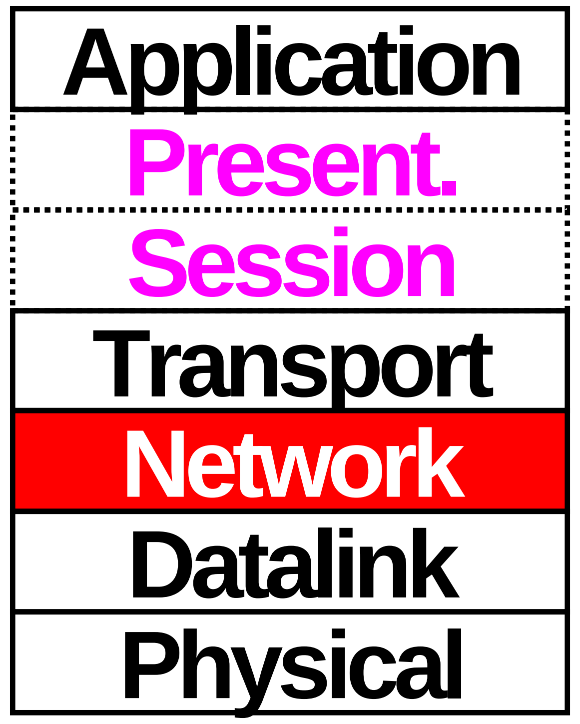
MAC Protocols

- **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



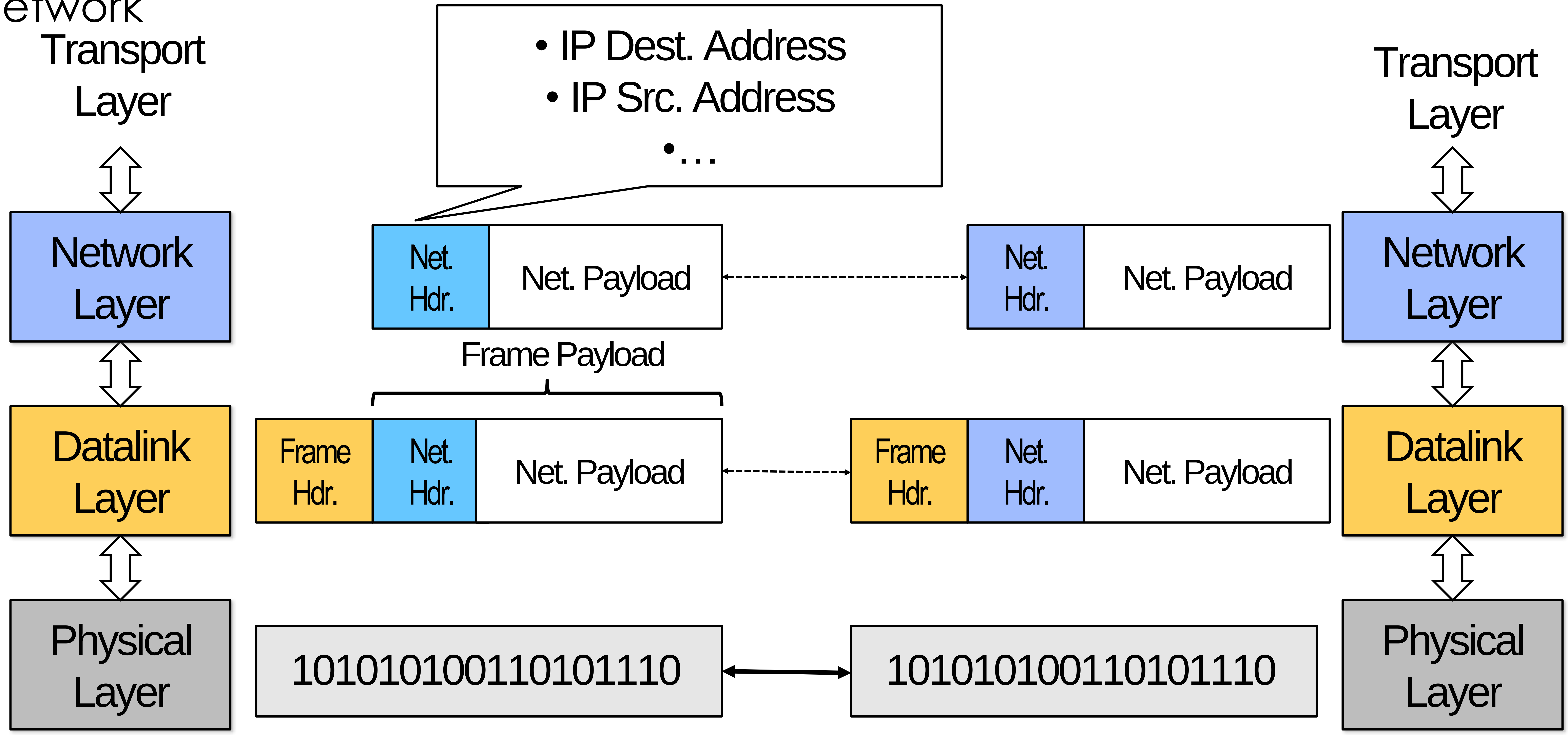
(Inter) Network Layer (3)

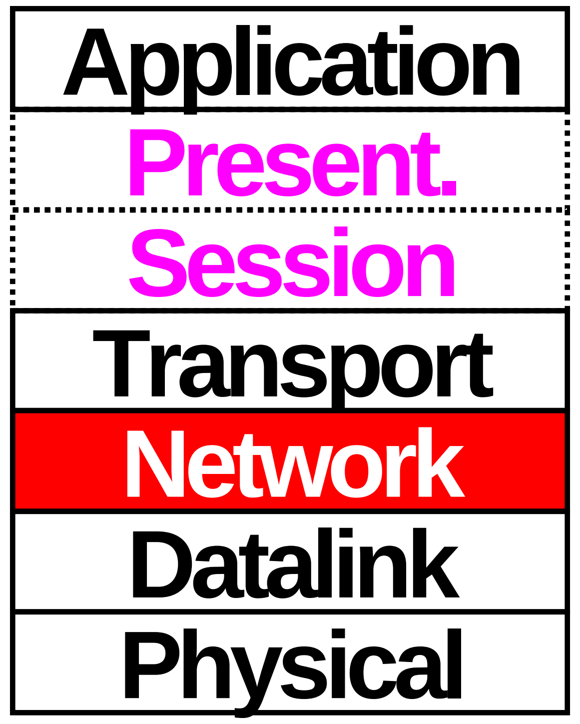
- **Service**:
  - Deliver packets to specified **network (IP) addresses** across multiple datalink layer networks
  - Possible other services:
    - Packet *scheduling/priority*
    - Buffer management
- **Interface**: send *packets* to specified network address destination; receive packets destined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



(Inter) Network Layer (3)

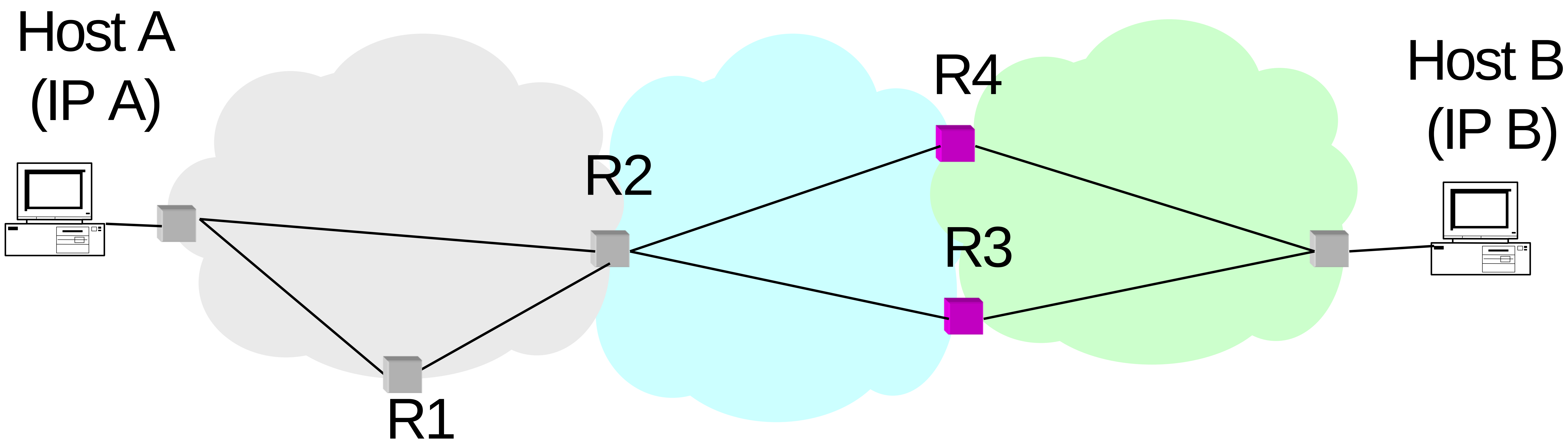
- **IP address**: unique addr. assigned to network device
- Assigned by network administrator or dynamically when host connects to network





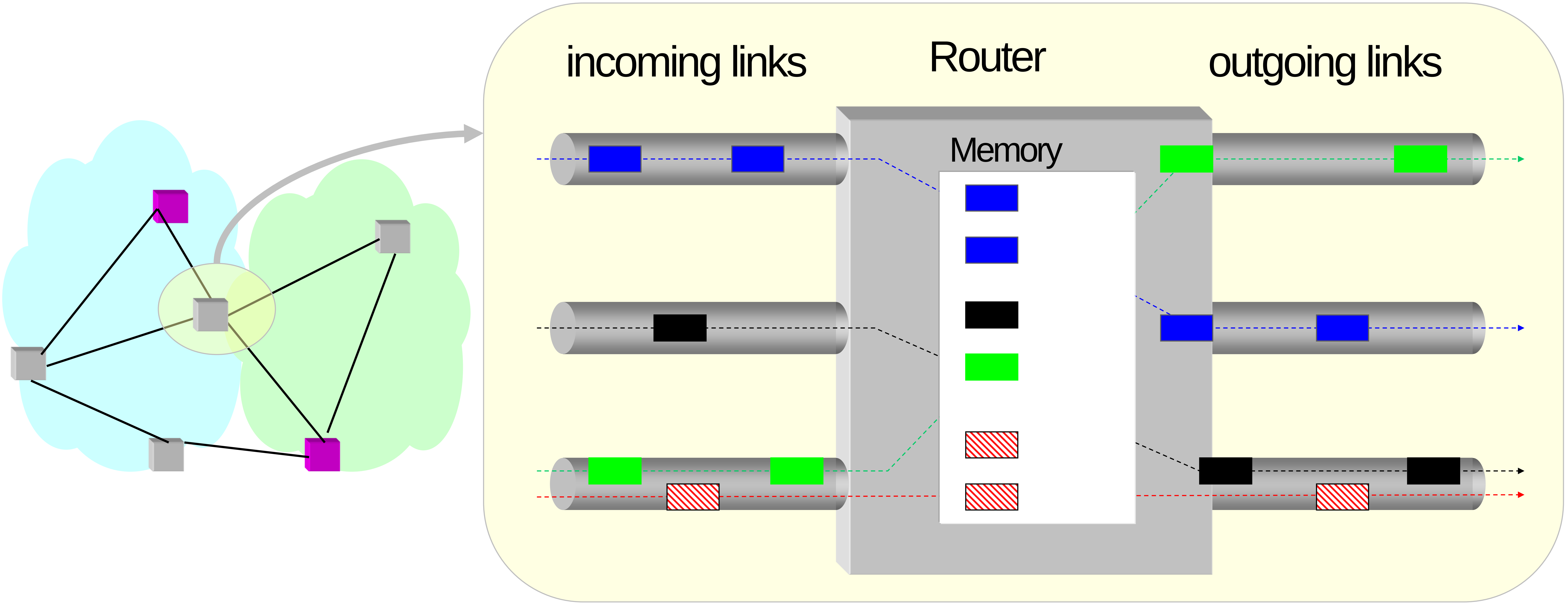
Wide Area Network

- **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 technologies (e.g., wireless, cellular, optics, wired)



Routers

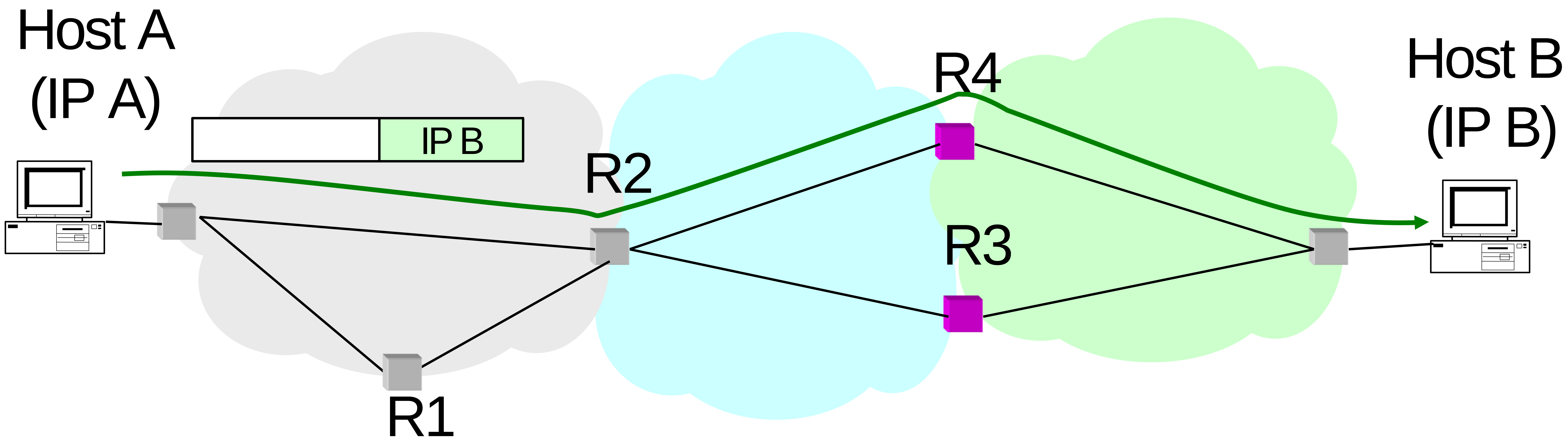
- **Forward** each packet received on an **incoming link** to an **outgoing link** based on packet's destination IP address (towards its destination)
- **Store & forward**: packets are buffered before being forwarded
- **Forwarding table**: mapping between IP address and the output link

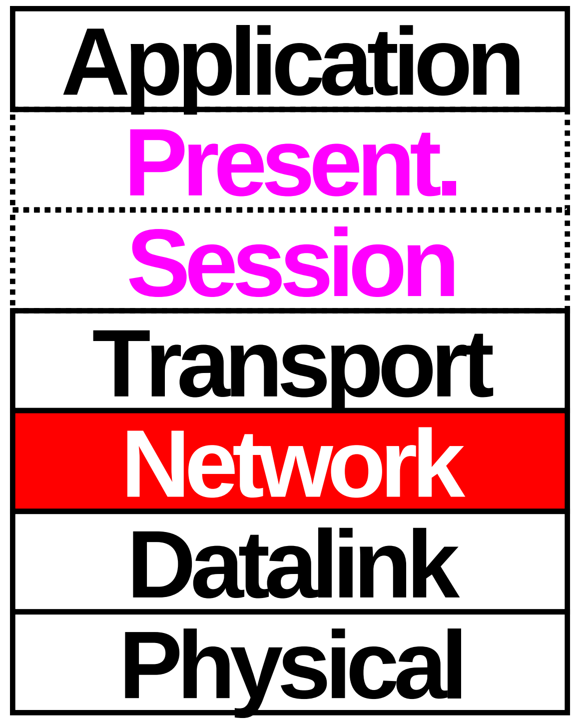


Application
Present.
Session
Transport
Network
Datalink
Physical

Packet Forwarding

- 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





The Internet Protocol (IP)

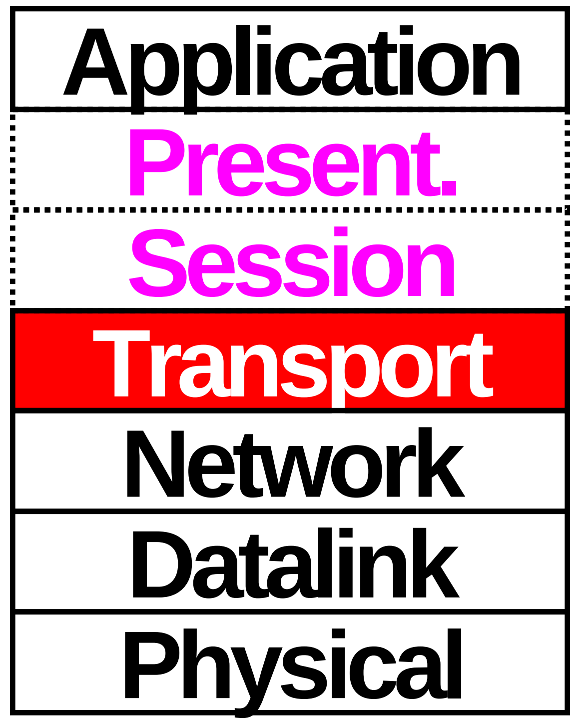
- Internet Protocol: Internet's network layer
- Service it provides: "Best-Effort" Packet Delivery
  - Tries it's "best" to deliver packet to its destination
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order

Application
Present.
Session
Transport
Network
Datalink
Physical

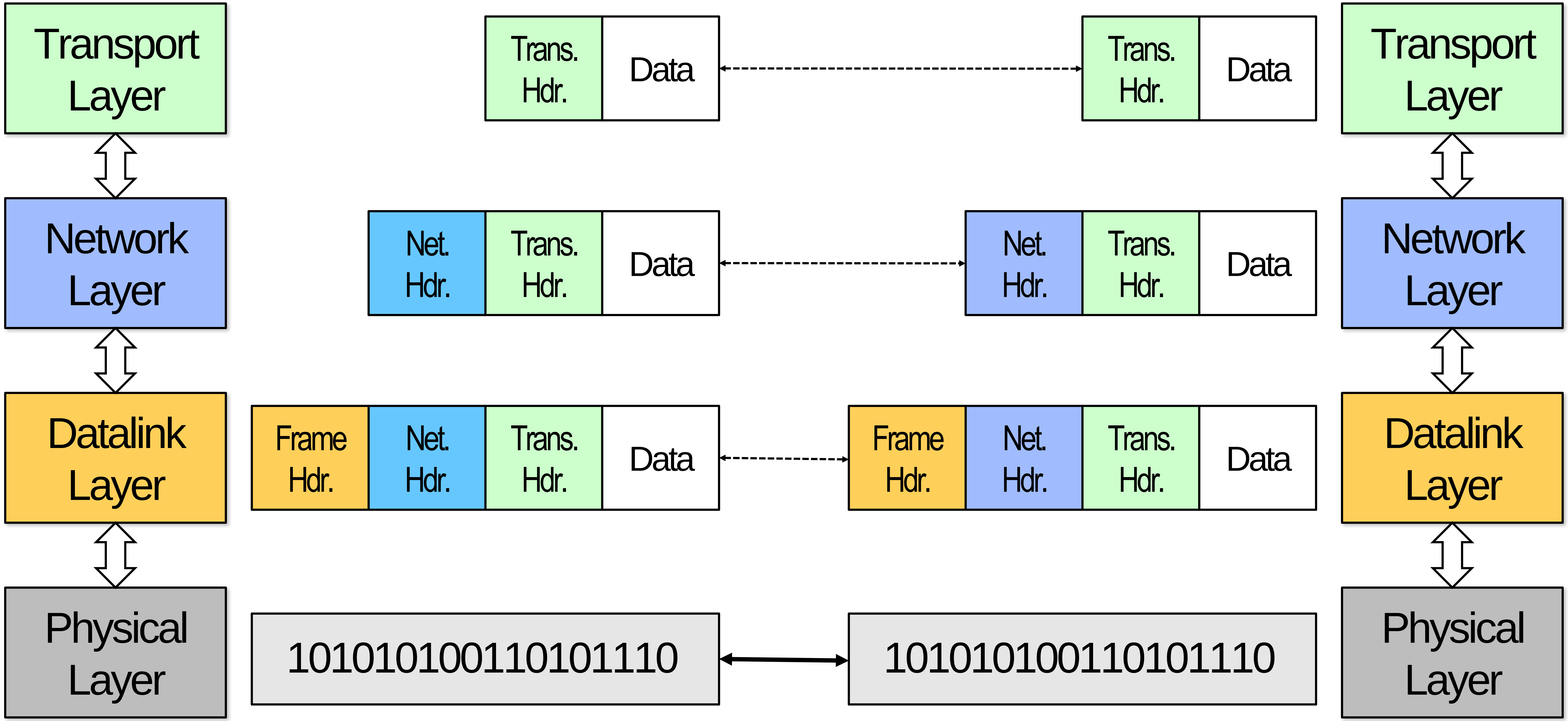
Transport Layer (4)

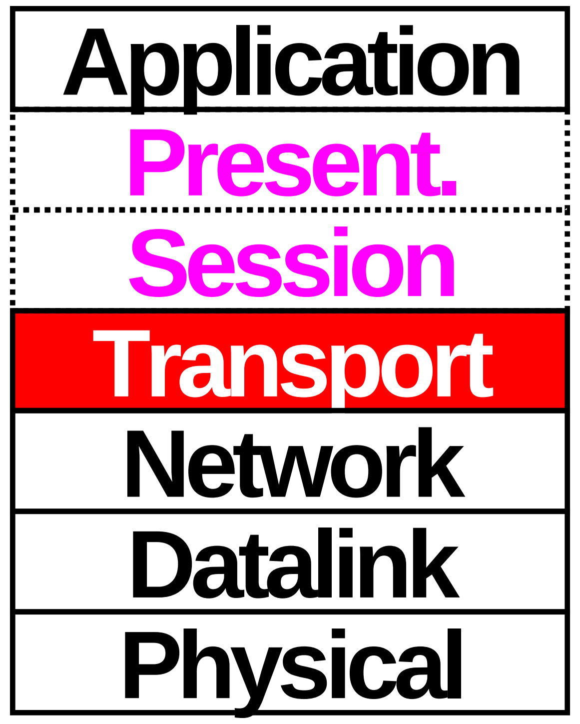
- **Service**:
  - Provide end-to-end communication between processes
  - Demultiplexing of communication between hosts
  - Possible other services:
    - Reliability in the presence of errors
    - Timing properties
    - Rate adaption (flow-control, congestion control)
- **Interface**: send message 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
- Examples: TCP and UDP

Port Numbers



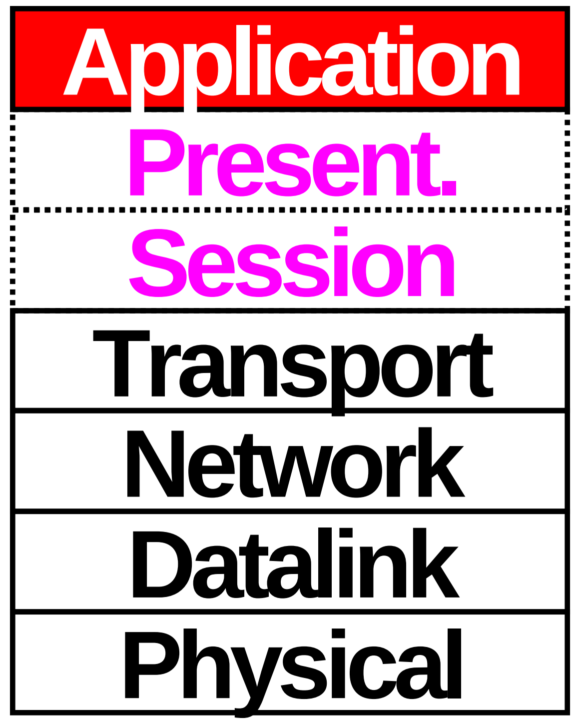
- Port number: 16-bit number identifying the end-point of a transport connection





Internet Transport Protocols

- 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



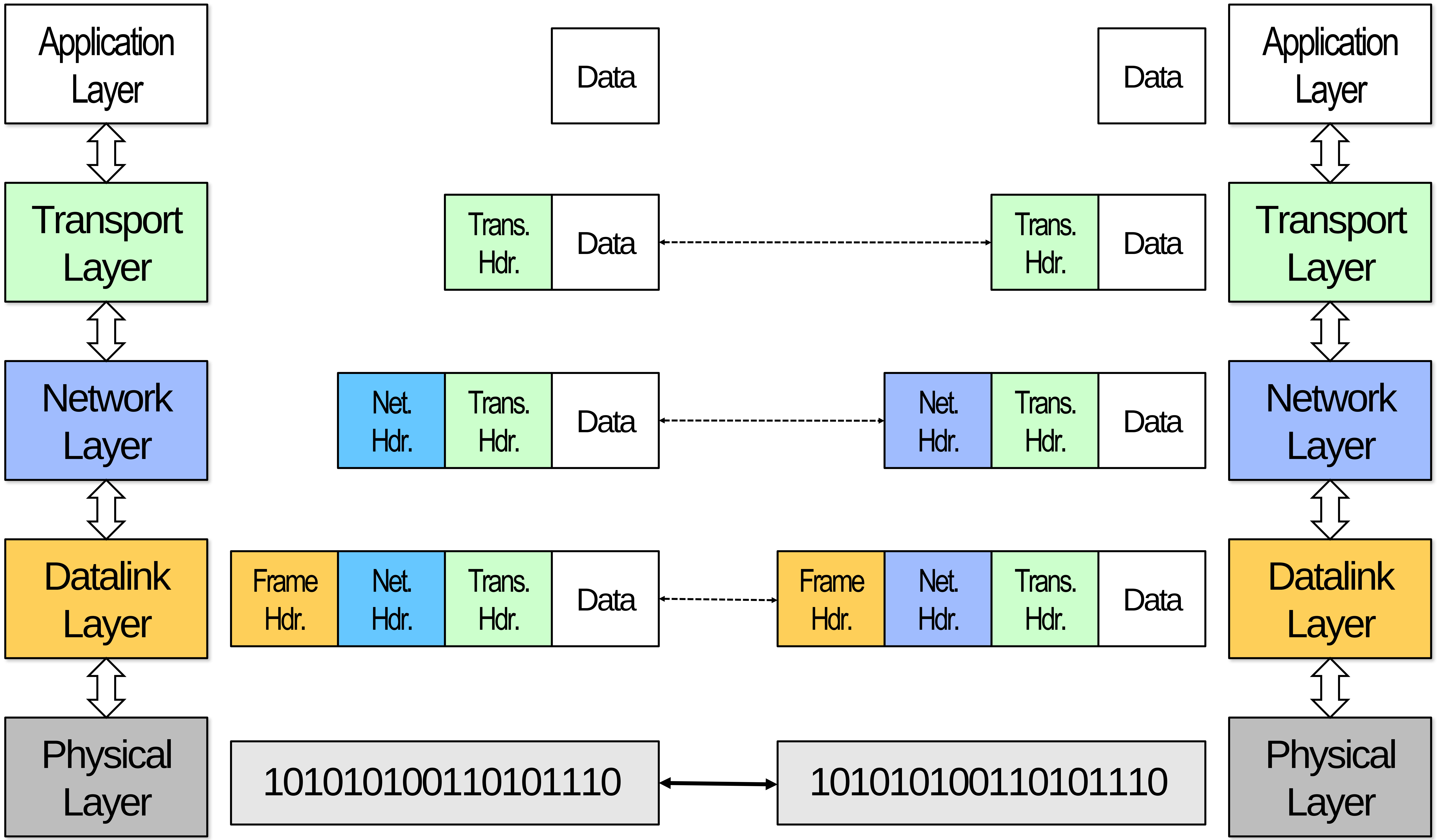
Application Layer (7 - not 5!)

- **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 …

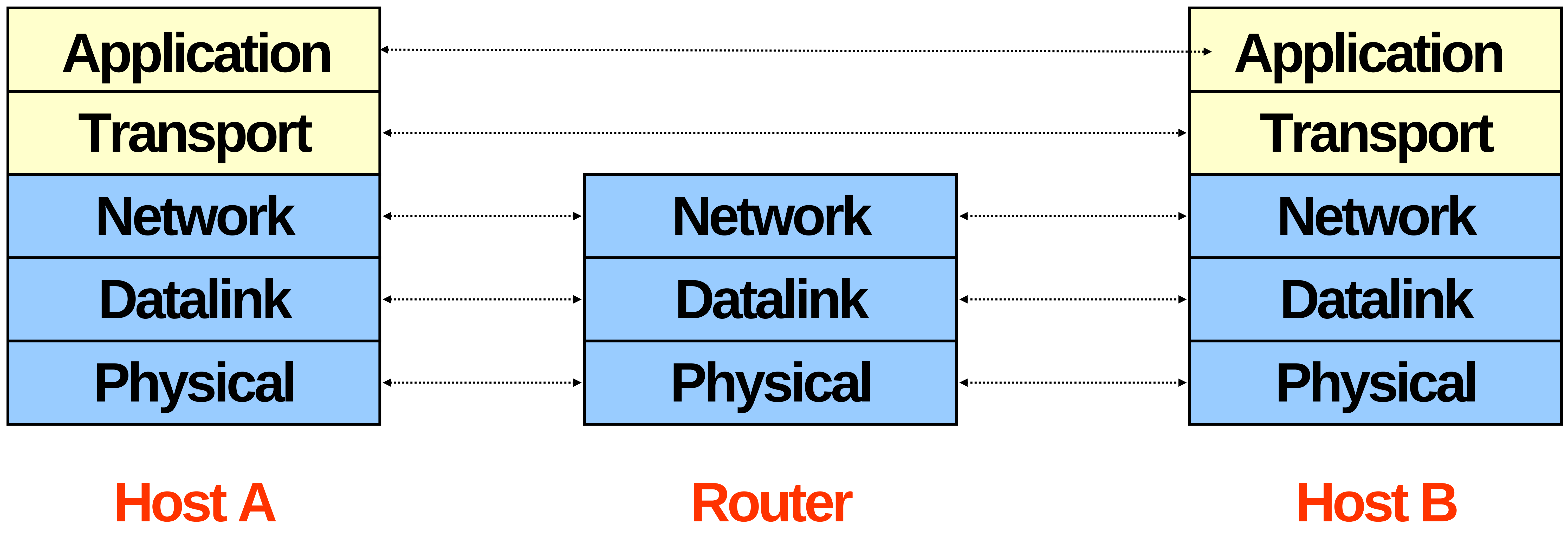
- What happened to layers 5 & 6?
  - “Session” and “Presentation” layers
  - Part of ***OSI*** architecture, but not Internet architecture
  - Their functionality is provided by application layer

Application Layer (5)



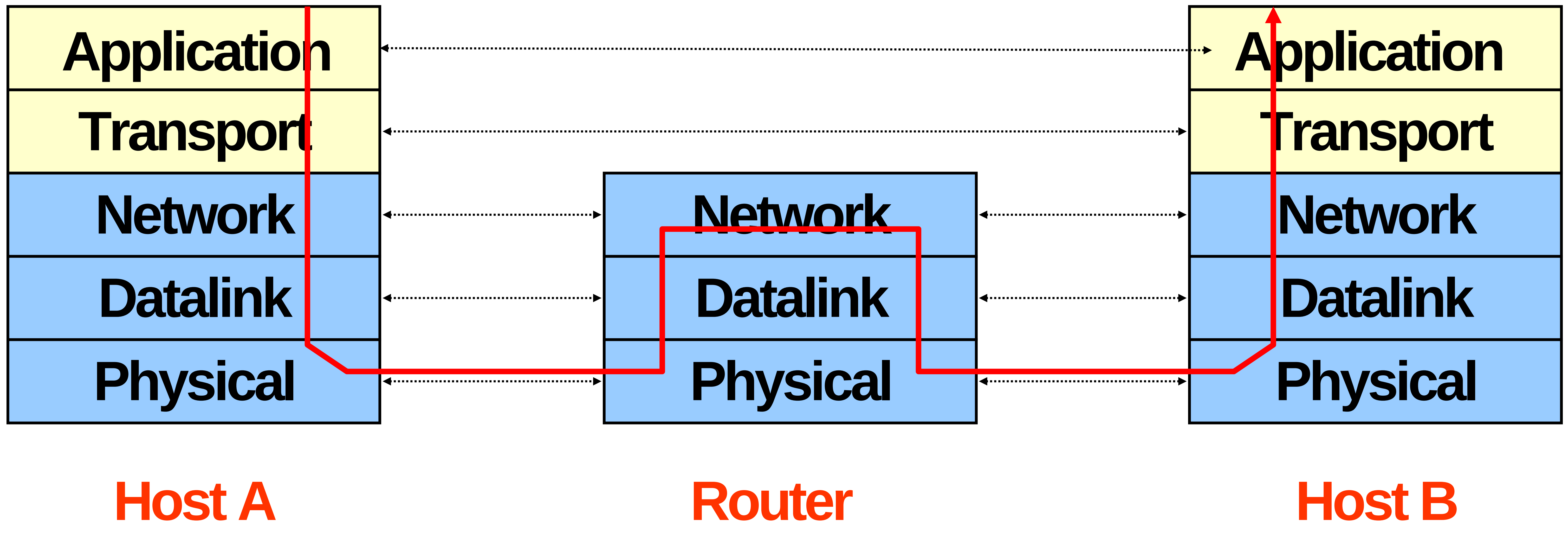
Five Layers Summary

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

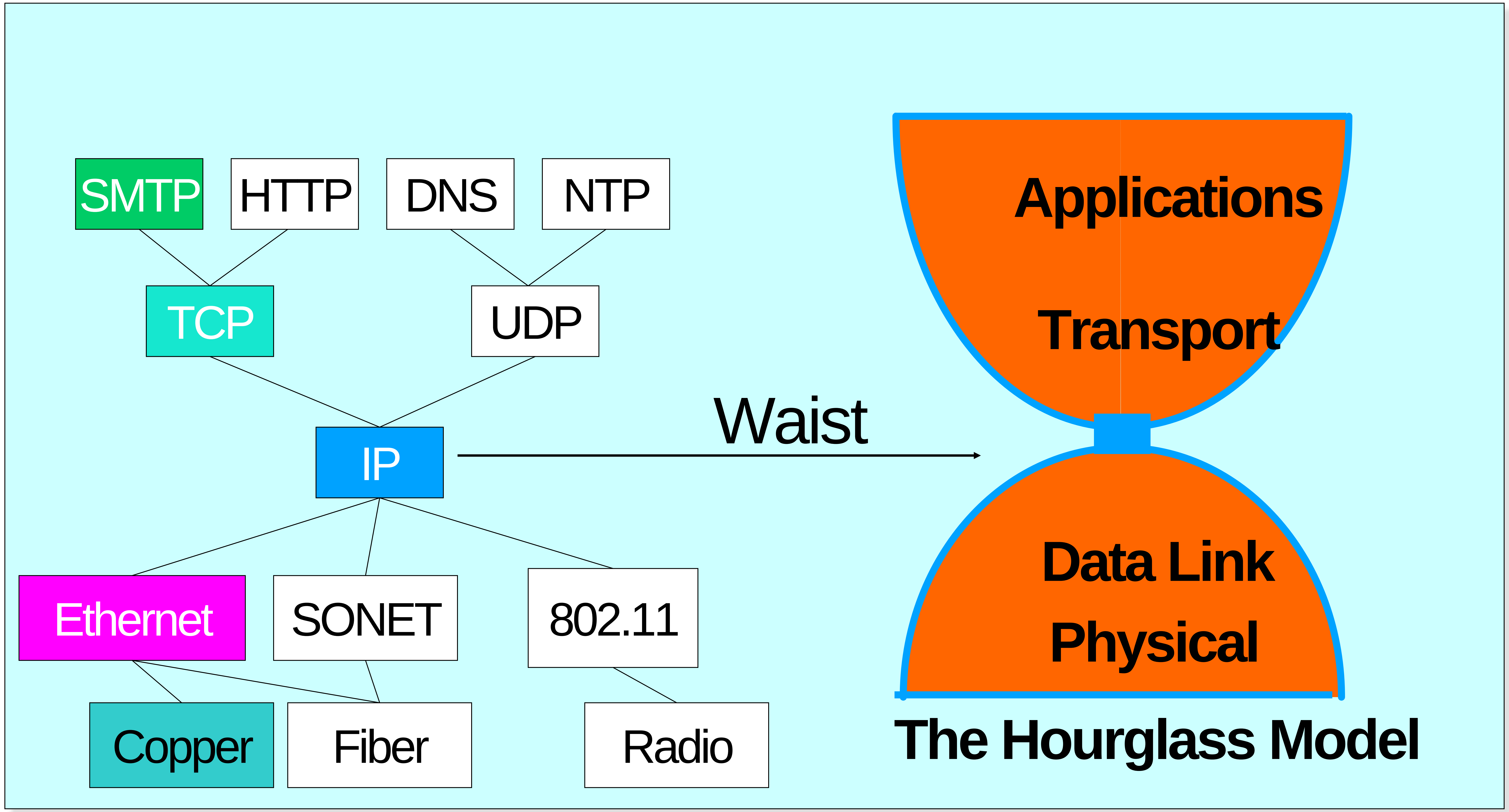


Physical Communication

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



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** is very complicated and slow

Drawbacks of Layering

- Layering can hurt performance
  - E.g., hiding details about what is really going on
- Headers start to get really big
  - Sometimes header bytes >> actual content
- 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

Summary

- 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