TDTS21 Advanced Networking

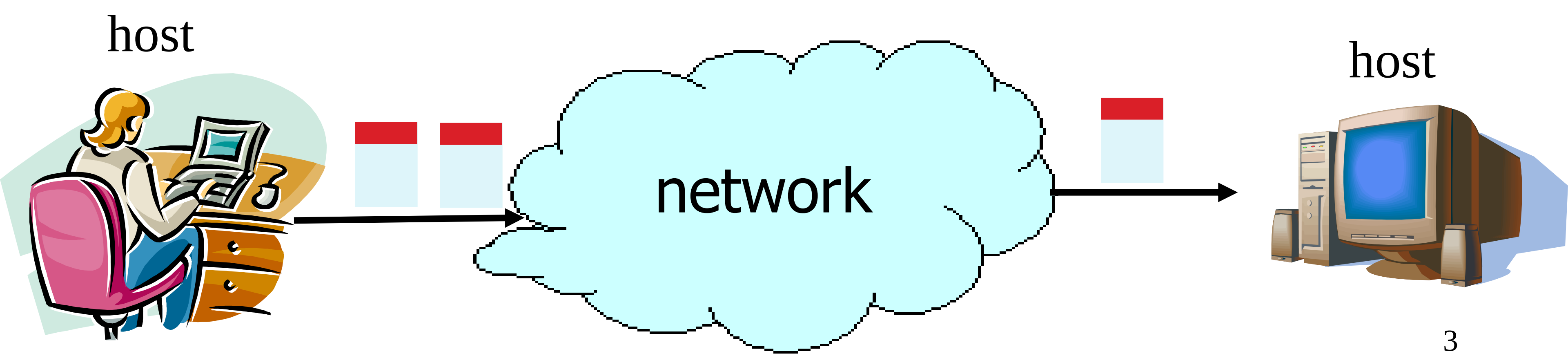
Lecture 2: Hosts, the Internet architecture, and the E2E arguments …

Based on slides from D. Choffnes, P. Gill, and J. Rexford
Revised Spring 2015 by N. Carlsson

End hosts ...

Host-Network Division of Labor

- Network
  - Best-effort packet delivery
  - Between two (or more) end-point addresses
- Hosts
  - Everything else



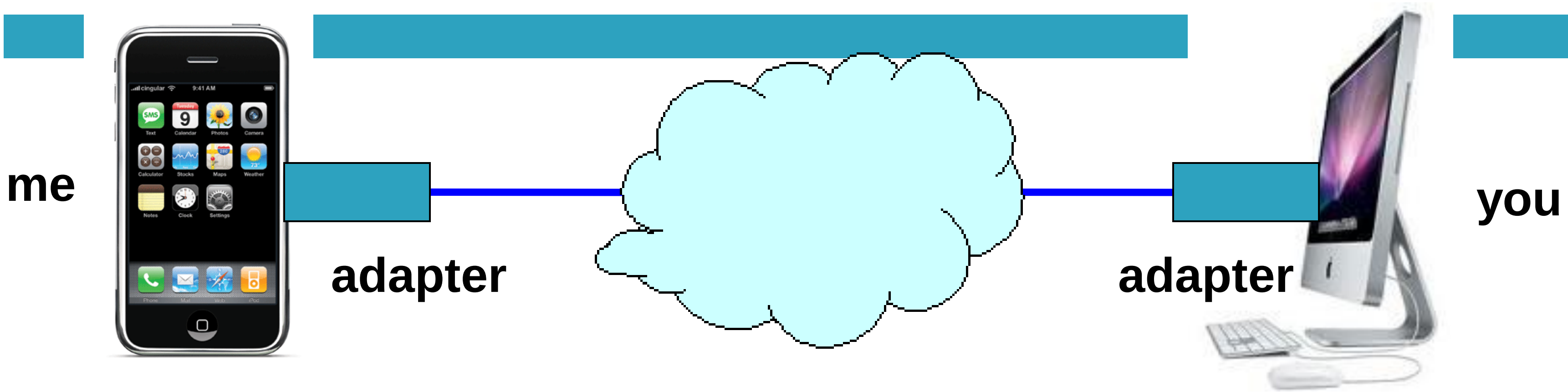
The Role of the End Host

- Network discovery and bootstrapping
  - How does the host join the network?
  - How does the host get an address?
- Interface to networked applications
  - What interface to higher-level applications?
  - How does the host realize that abstraction?
- Distributed resource sharing
  - What roles does the host play in network resource allocation decisions?

Three Kinds of Identifiers

	Host Name	IP Address	MAC Address
Example	www.cs.princeton.edu	128.112.7.156	00-15-C5-49-04-A9
Size	Hierarchical, human readable, variable length	Hierarchical, machine readable, 32 bits (in IPv4)	Flat, machine readable, 48 bits
Read by	Humans, hosts	IP routers	Switches in LAN
Allocation, top-level	Domain, assigned by registrar (e.g., for .edu)	Variable-length prefixes, assigned by ICANN, RIR, or ISP	Fixed-sized blocks, assigned by IEEE to vendors (e.g., Dell)
Allocation, low-level	Host name, local administrator	Interface, by DHCP or an administrator	Interface, by vendor

Learning a Host's Address

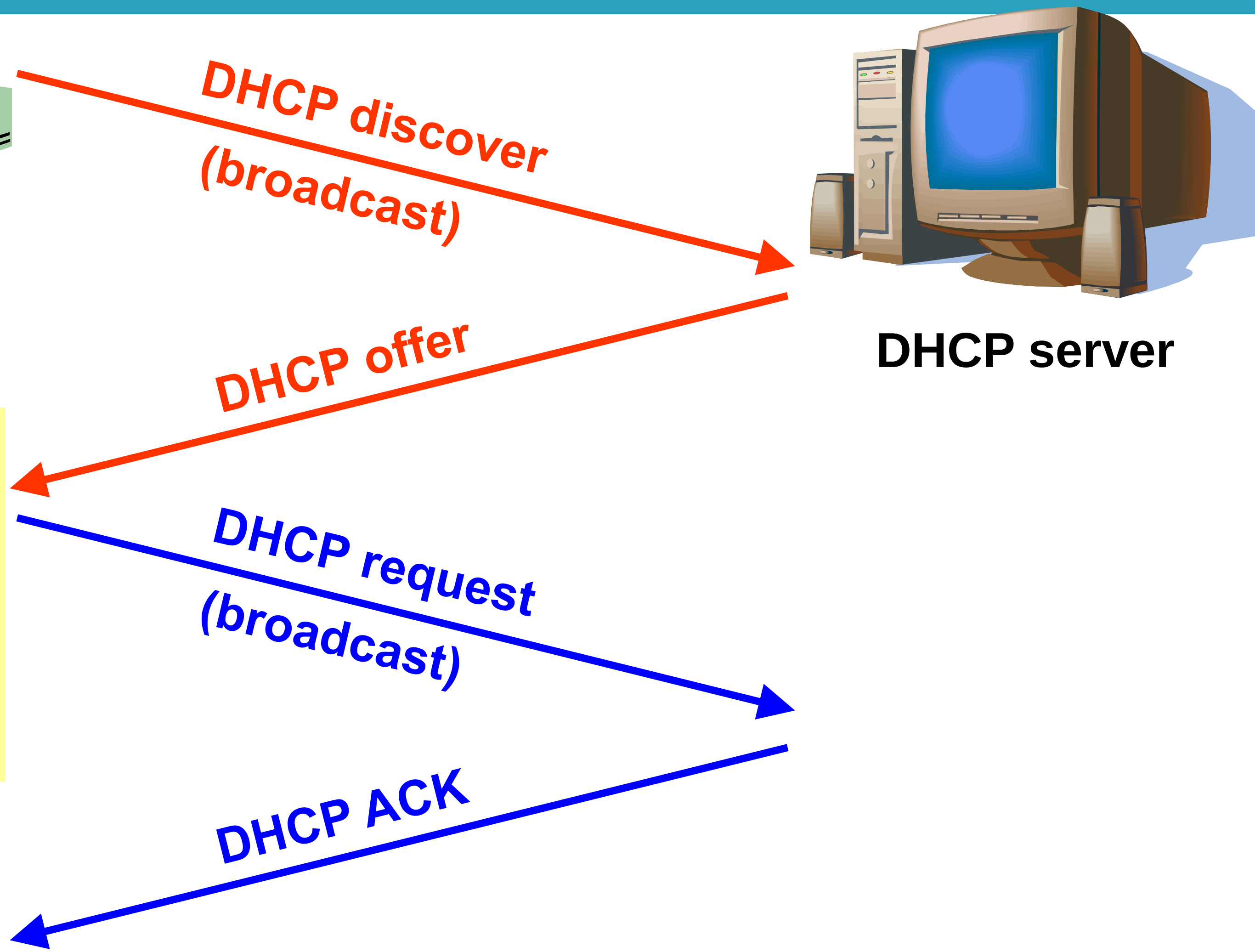


- Who am I?
  - Hard-wired: MAC address
  - Static configuration: IP interface configuration
  - Dynamically learned: IP address configured by DHCP
- Who are you?
  - Hard-wired: IP address in a URL, or in the code
  - Dynamically looked up: ARP or DNS

Mapping Between Identifiers

- Dynamic Host Configuration Protocol (DHCP)
  - Given a MAC address, assign a unique IP address
  - … and tell host other stuff about the Local Area Network
  - To automate the boot-strapping process
- Address Resolution Protocol (ARP)
  - Given an IP address, provide the MAC address
  - To enable communication within the Local Area Network
- Domain Name System (DNS)
  - Given a host name, provide the IP address
  - Given an IP address, provide the host name

Dynamic Host Configuration Protocol



<u>Host learns</u>
IP address,
Subnet mask,
Gateway address,
DNS server(s),
and a lease time.

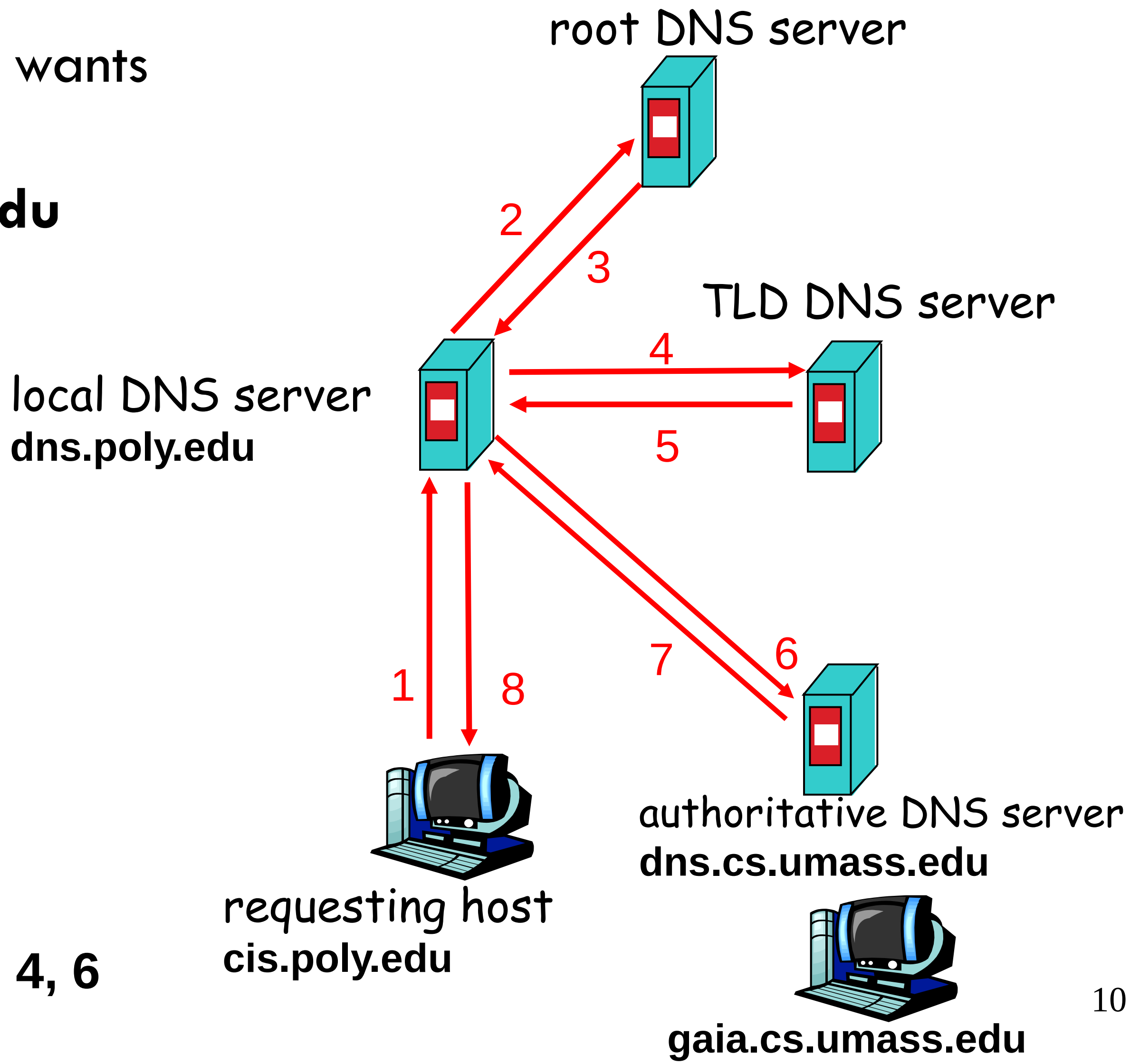
Address Resolution Protocol (ARP)

- Every host maintains an ARP table
  - (IP address, MAC address) pair
- Consult the table when sending a packet
  - Map destination IP address to destination MAC address
  - Encapsulate and transmit the data packet

- But, what if the IP address is not in the table?
  - Sender broadcasts: "Who has IP address 1.2.3.156?"
  - Receiver responds: "MAC address 58-23-D7-FA-20-B0"
  - Sender caches the result in its ARP table

Domain Name System

Host at cis.poly.edu wants IP address for **gaia.cs.umass.edu**



Recursive query: #1

Iterative queries: #2, 4, 6

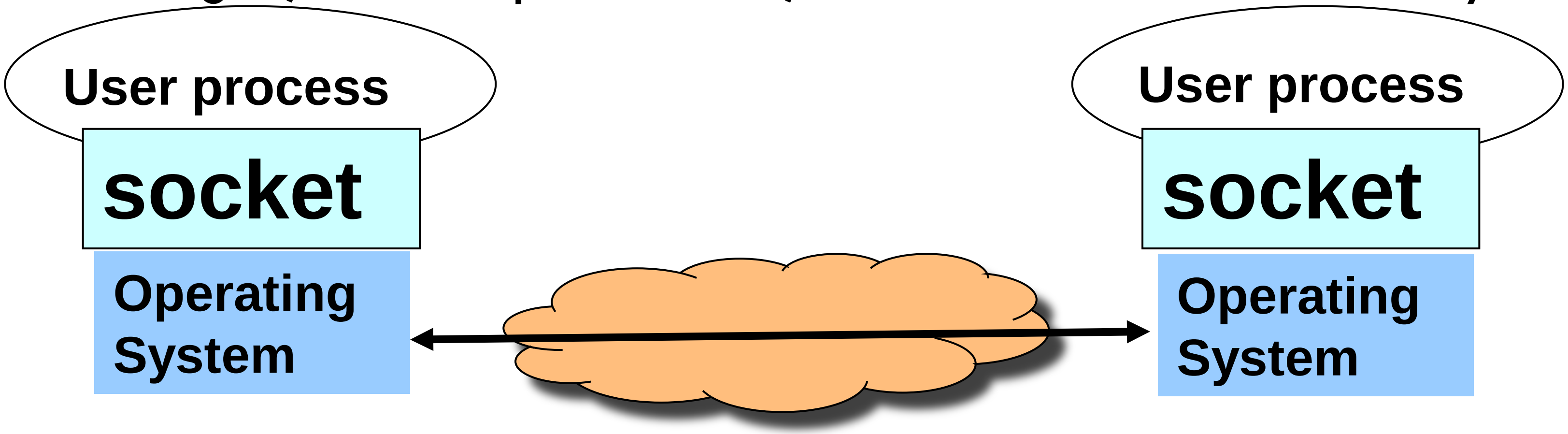
Questions

- Should addresses correspond to the interface (point of attachment) or to the host?
- Why do we have all three identifiers? Do we need all three?
- What should be done to prevent spoofing of addresses?

INTERFACE TO APPLICATIONS

Socket Abstraction

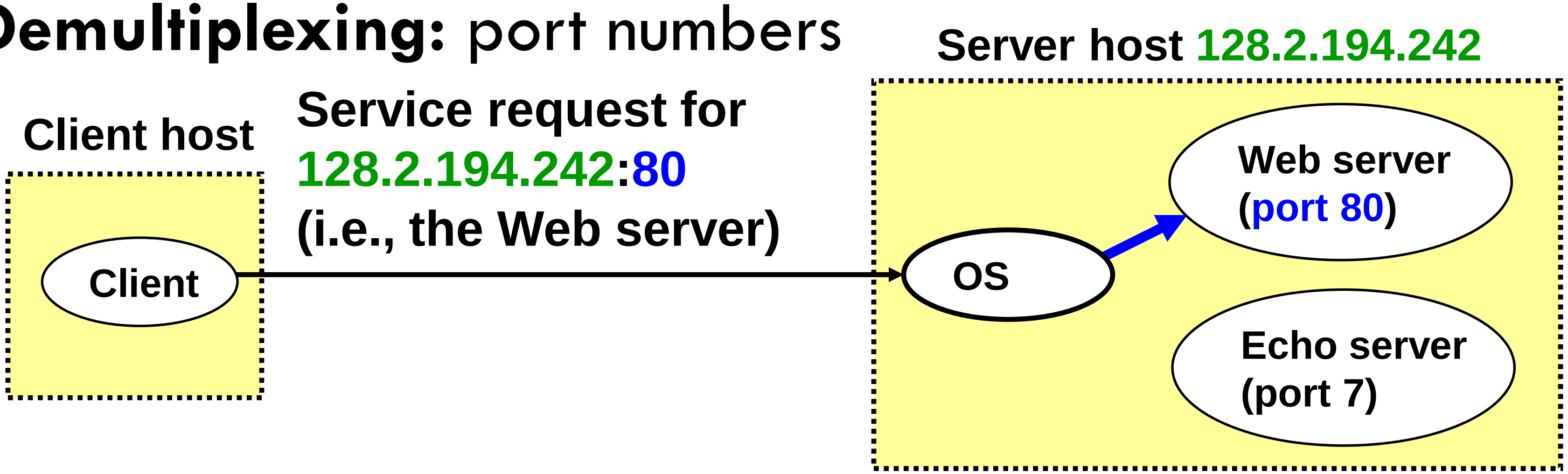
- Best-effort packet delivery is a clumsy abstraction
  - Applications typically want higher-level abstractions
  - Messages, uncorrupted data, reliable in-order delivery



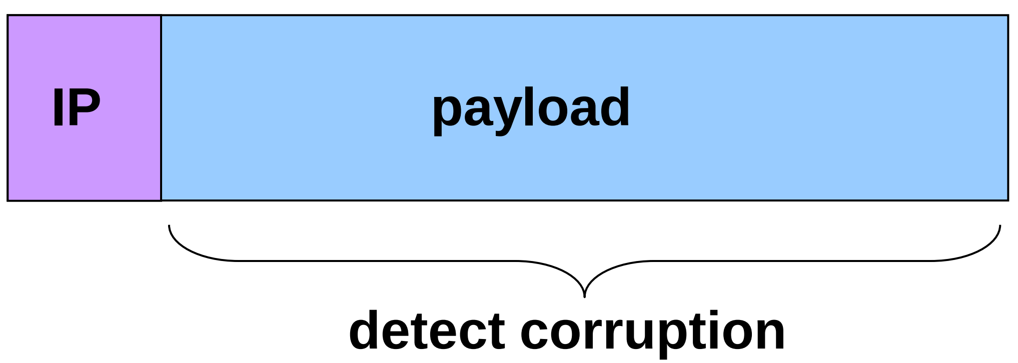
- Applications communicate using "sockets"
  - Stream socket: reliable stream of bytes (like a file)
  - Message socket: unreliable message delivery

Two Basic Transport Features

- **Demultiplexing:** port numbers



- **Error detection:** checksums

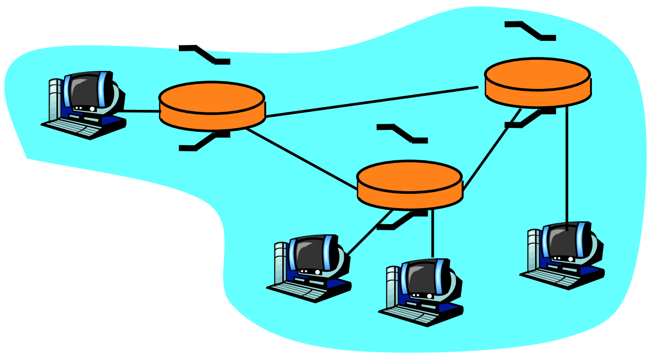


Two Main Transport Layers

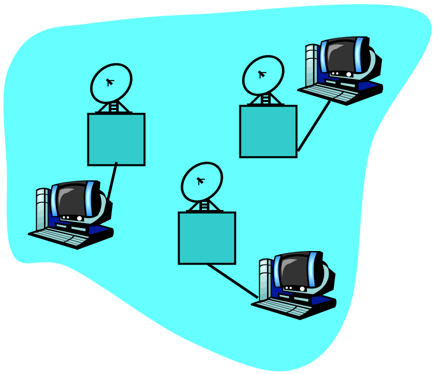
- User Datagram Protocol (UDP)
  - Just provides demultiplexing and error detection
  - Header fields: port numbers, checksum, and length
  - Low overhead, good for query/response and multimedia
- Transmission Control Protocol (TCP)
  - Adds support for a "stream of bytes" abstraction
  - Retransmitting lost or corrupted data
  - Putting out-of-order data back in order
  - Preventing overflow of the receiver buffer
  - Adapting the sending rate to alleviate congestion
  - Higher overhead, good for most stateful applications

Life in the 1970s…

- Multiple unconnected networks
  - ARPAnet, data-over-cable, packet satellite (Aloha), packet radio, …
- Heterogeneous designs
  - Addressing, max packet size, handling of lost/corrupted data, fault detection, routing, …



ARPAnet



satellite net

Handling Heterogeneity

- Where to handle heterogeneity?
  - Application process? End hosts? Packet switches?

- Compatible process and host conventions
  - Obviate the need to support all combinations
- Retain the unique features of each network
  - Avoid changing the local network components
- Introduce the notion of a gateway

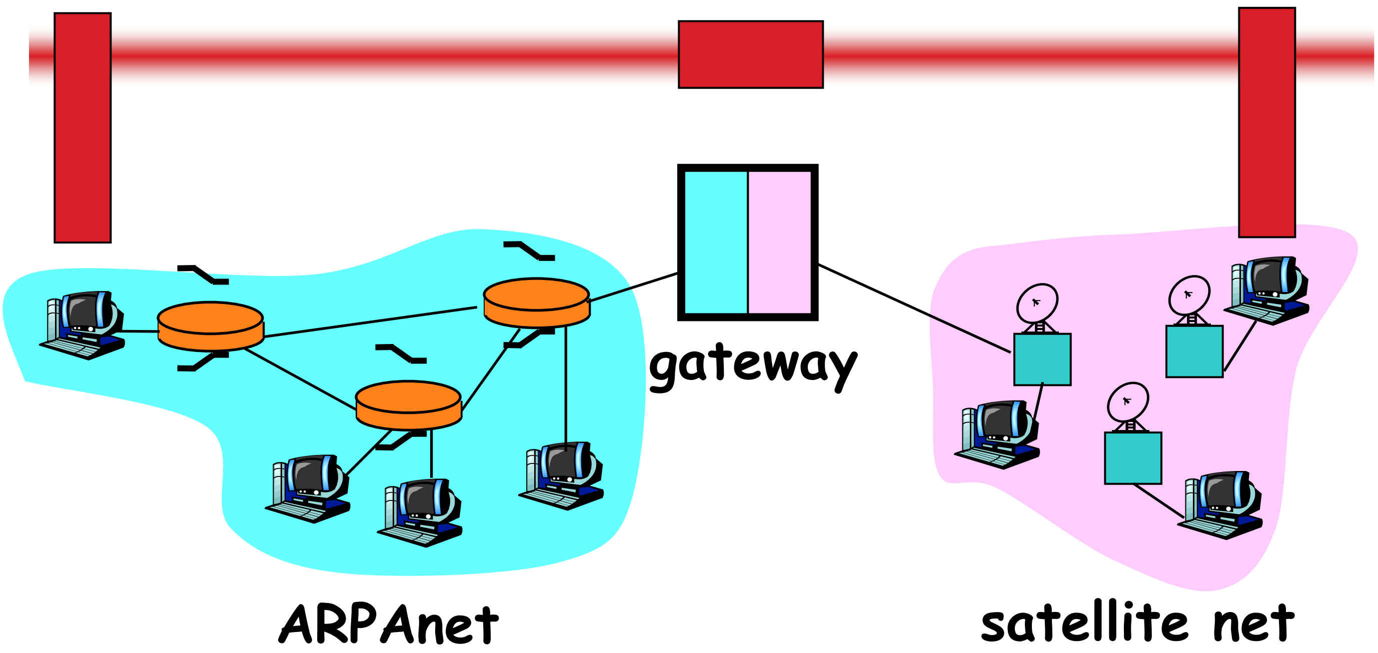
Internetwork Layer and Gateways

Internetwork Layer

- Internetwork appears as a single, uniform entity
- Despite the heterogeneity of the local networks
- Network of networks

Gateway

- "Embed internetwork packets in local packet format or extract them"
- Route (at internetwork level) to next gateway



THE DESIGN PHILOSOPHY OF THE DARPA INTERNET PROTOCOLS

CLARK '88

Goals of the Internet Architecture (Clark ‘88)

1. **Connect existing networks**
2. Robust in face of failures (not nuclear war…)
3. Support multiple types of services
4. Accommodate a variety of networks
5. Allow distributed management
6. Easy host attachment
7. Cost effective
8. Allow resource accountability

Robust

1. As long as the network is not partitioned, two endpoints should be able to communicate
2. Failures (excepting network partition) should not interfere with endpoint semantics (why?)

- Maintain state only at end-points
  - Fate-sharing, eliminates network state restoration
    - If information associated with an entity is lost, then the entity itself must have been lost
  - stateless network architecture (no per-flow state)

- Routing state is held by network (why?)
- No failure information is given to ends (why?)

Types of Services

- Use of the term "communication services" already implied that they wanted application-neutral network
- Realized TCP wasn't needed (or wanted) by some applications
- Separated TCP from IP, and introduced UDP
  - What's missing from UDP?

Variety of Networks

- Incredibly successful!
  - Minimal requirements on networks
  - No need for reliability, in-order, fixed size packets, etc.

- IP over everything
  - Then: ARPANET, X.25, DARPA satellite network..
  - Now: ATM, SONET, WDM…

Real Goals

1. **Something that works.....**
2. Connect existing networks
3. Survivability (not nuclear war…)
4. Support multiple types of services
5. Accommodate a variety of networks
6. Allow distributed management
7. Easy host attachment
8. Cost effective
9. Allow resource accountability

Internet Motto

We reject kings, presidents, and voting.

We believe in ***rough consensus and running code.***"

David Clark

Questions

- What priority order would a commercial design have?
- What would a commercially invented Internet look like?
- What goals are missing from this list?
- Which goals led to the success of the Internet?

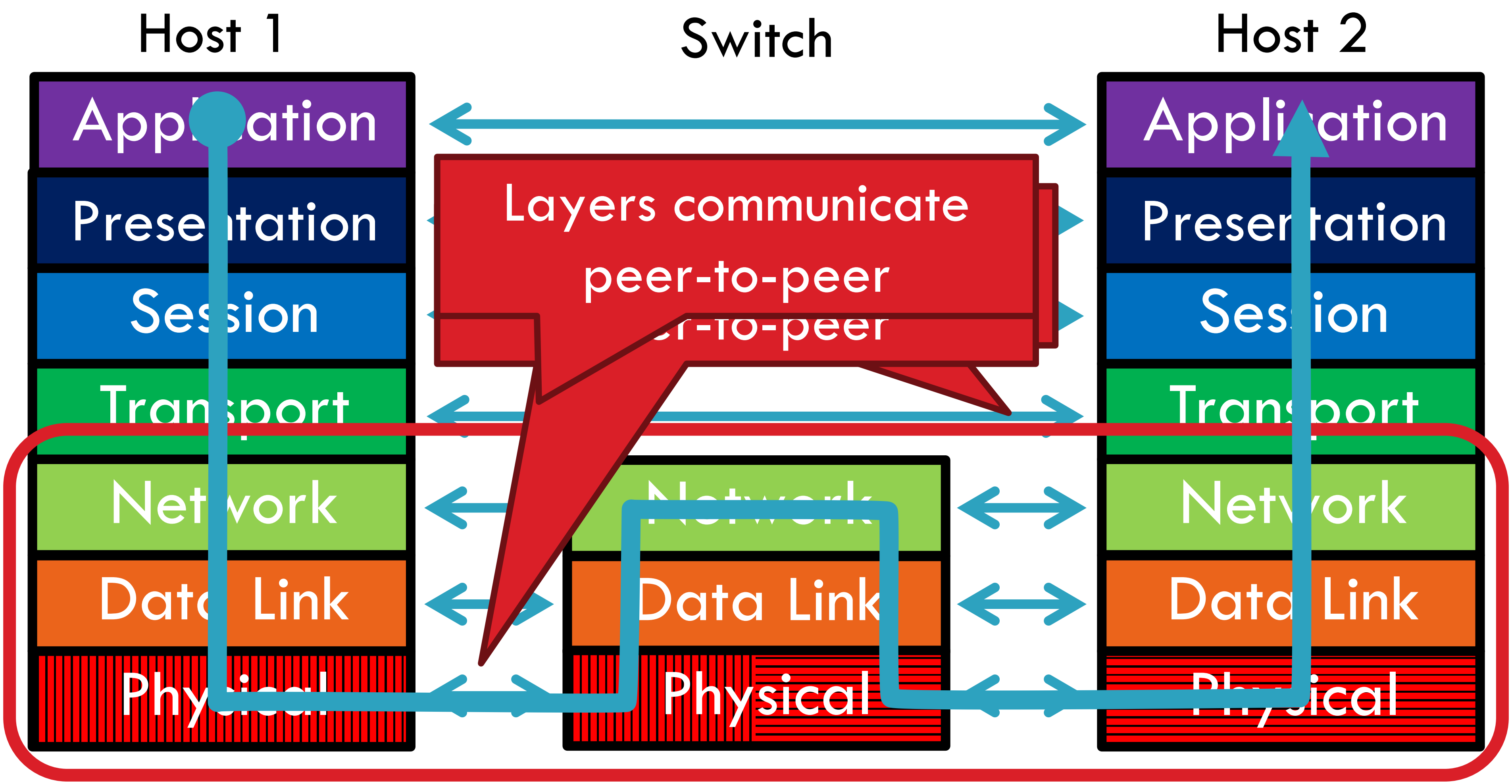
1. **Something that works…..**
2. Connect existing networks
3. Survivability (not nuclear war…)
4. Support multiple types of services
5. Accommodate a variety of networks
6. Allow distributed management
7. Easy host attachment
8. Cost effective
9. Allow resource accountability

Outline

- Layering
  - The OSI Model
- Communicating
  - The End-to-End Argument

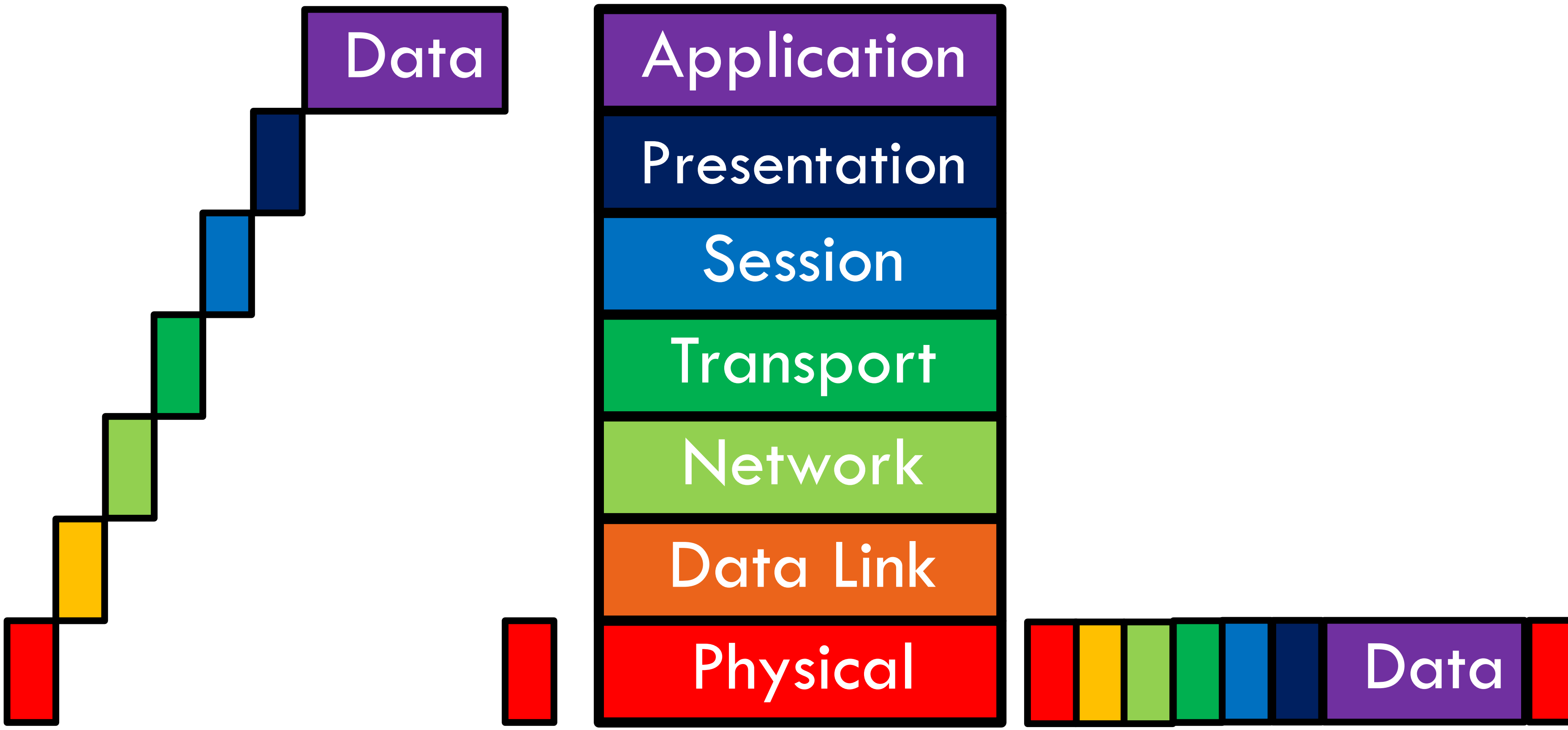
The ISO OSI Model

OSI: Open Systems Interconnect Model

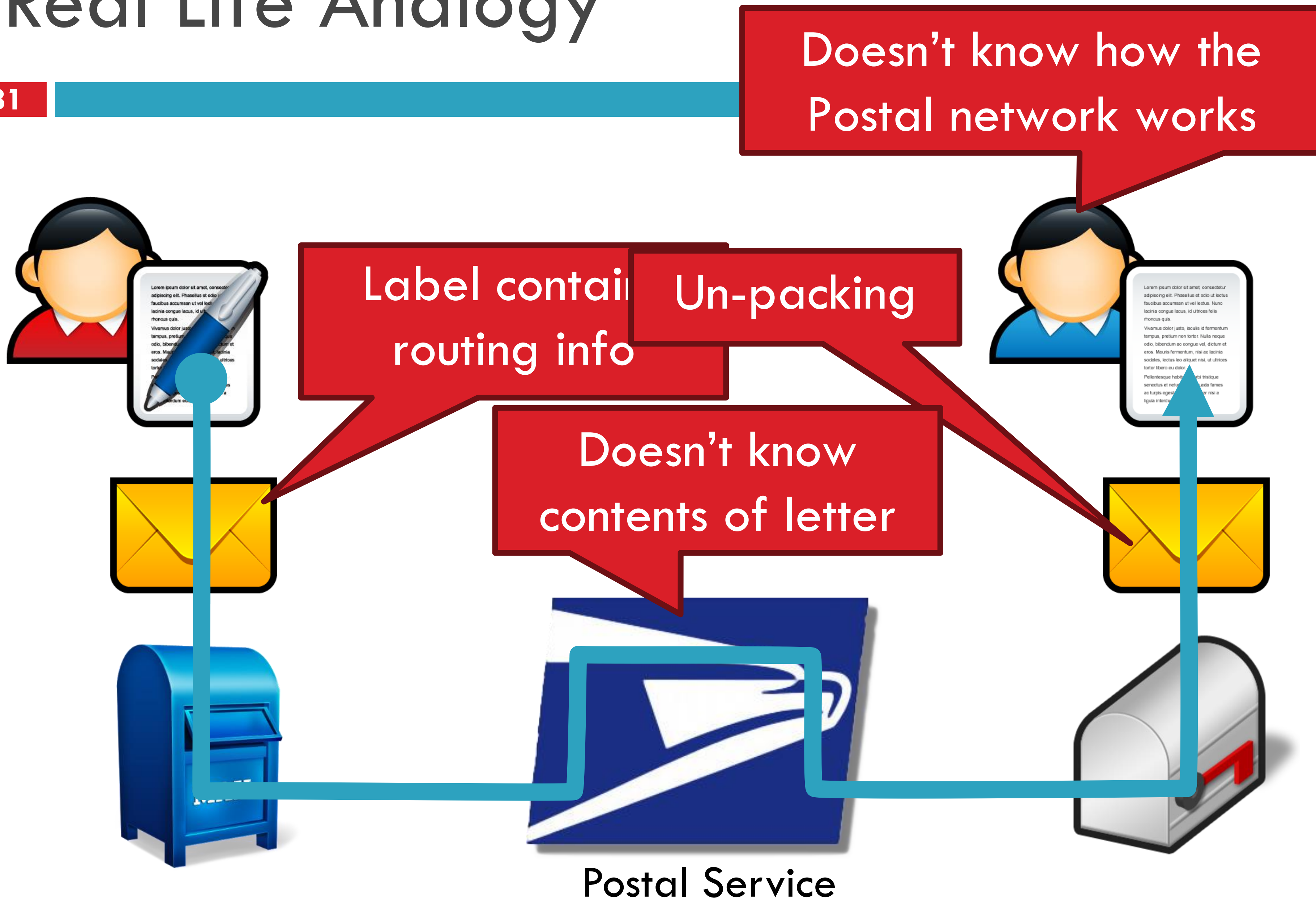


Encapsulation

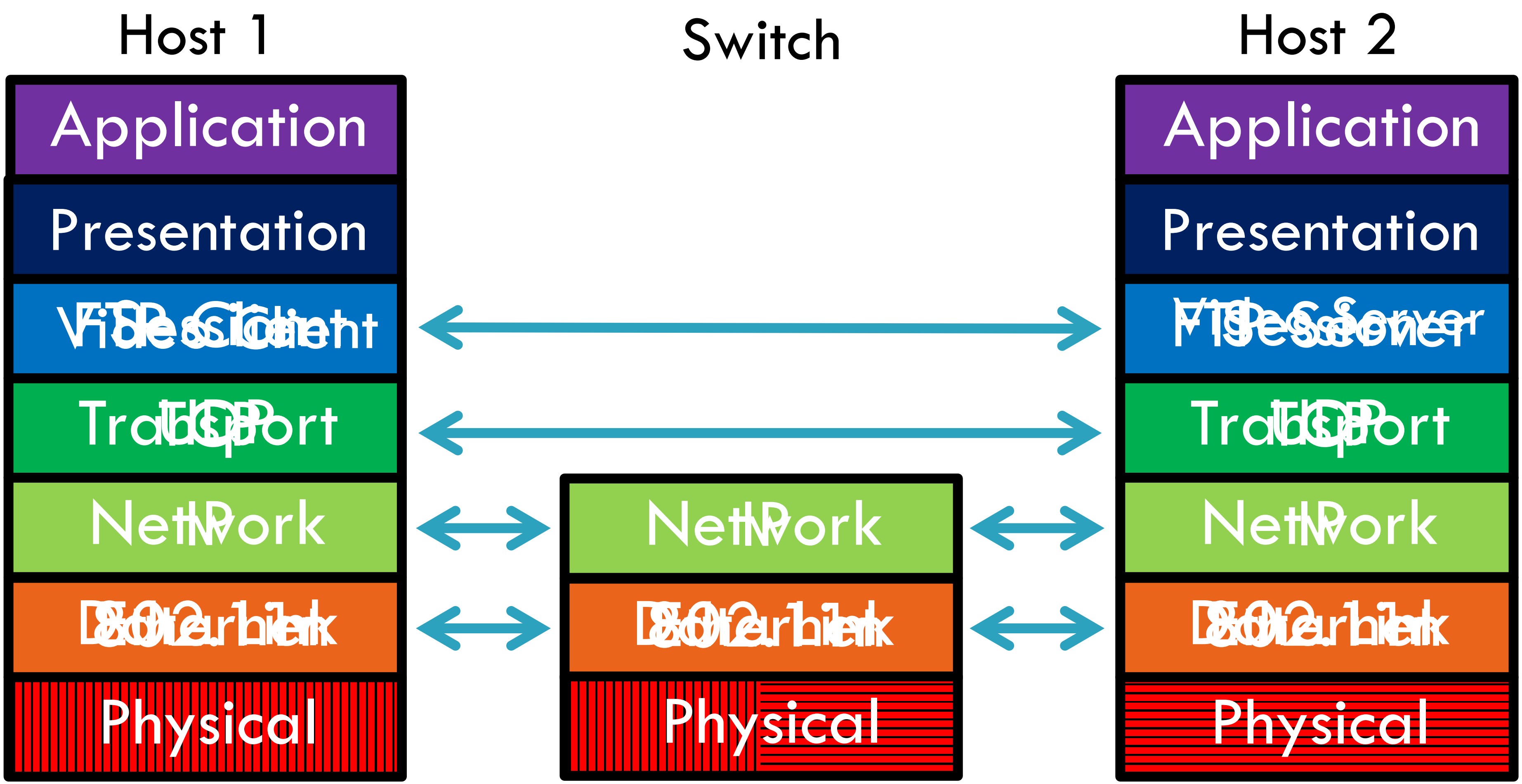
How does data move through the layers?



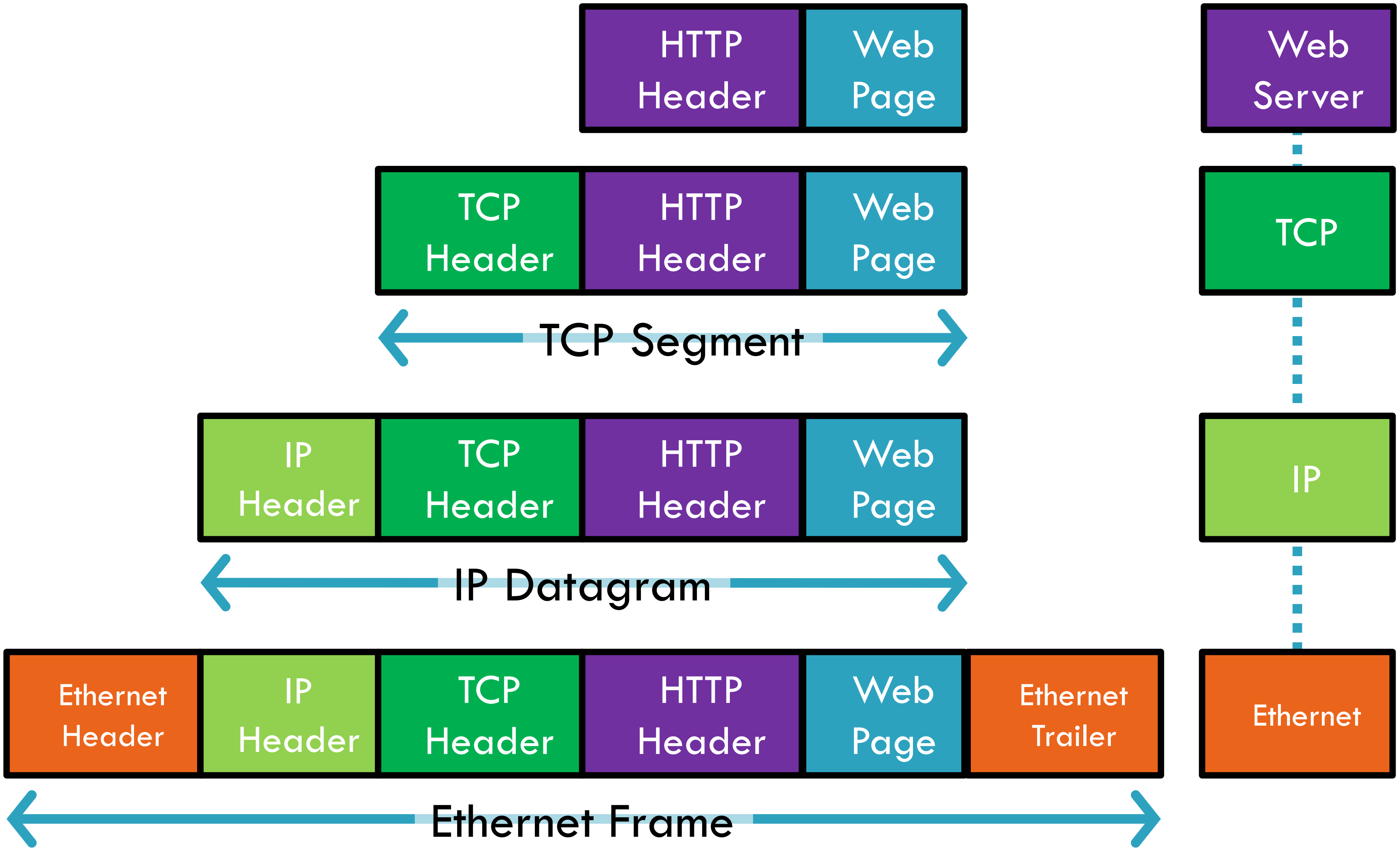
Real Life Analogy



Network Stack in Practice



Encapsulation, Revisited



The Hourglass

HTTP, FTP, RTP, IMAP, Jabber, …

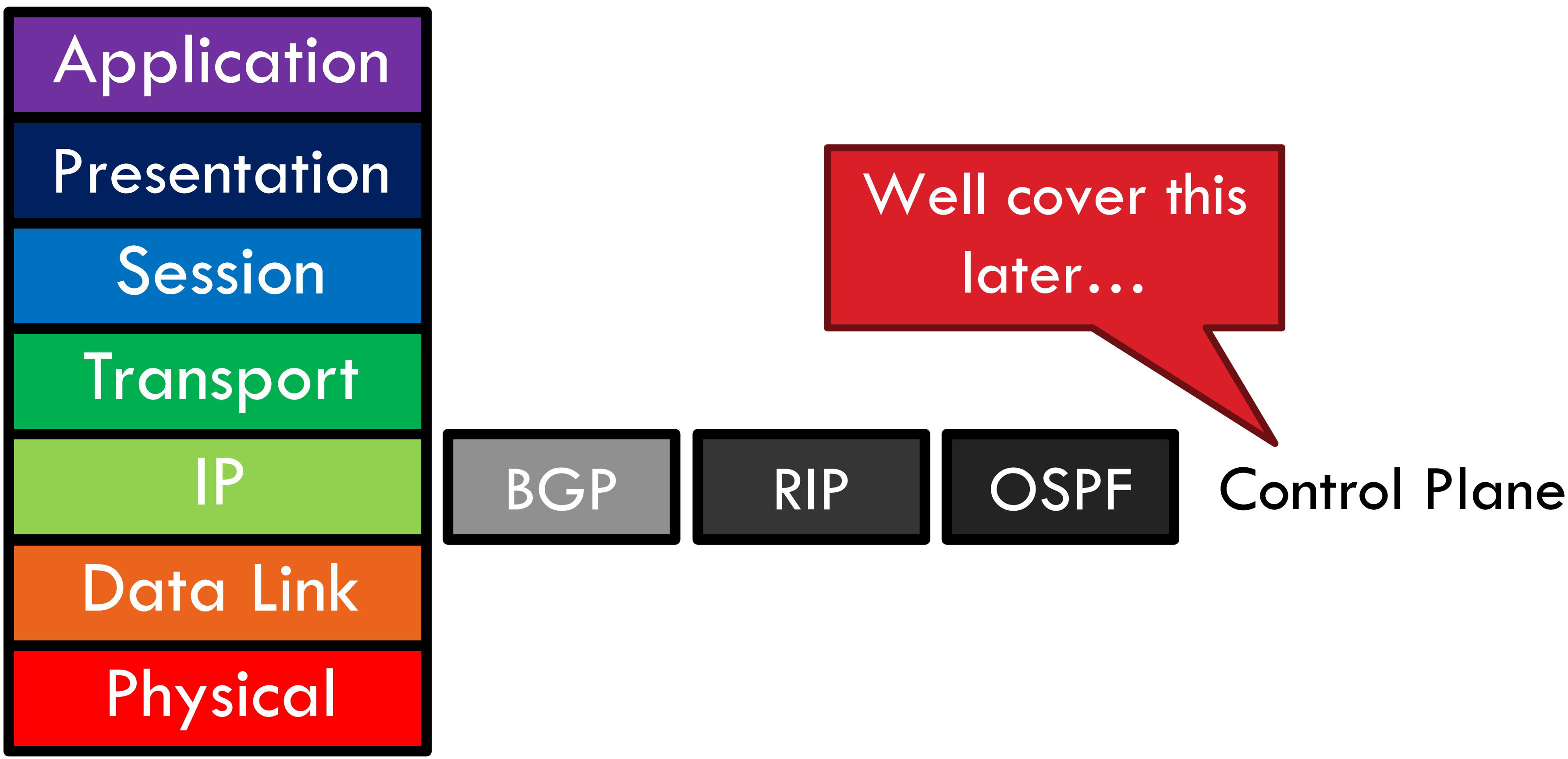
- One Internet layer means all networks interoperate
- All applications function on all…
- Room for development above
- But, changing IP is insanely hard

Think about the difficulty of deploying IPv6…

Fiber, Coax, Twisted Pair, Radio, …

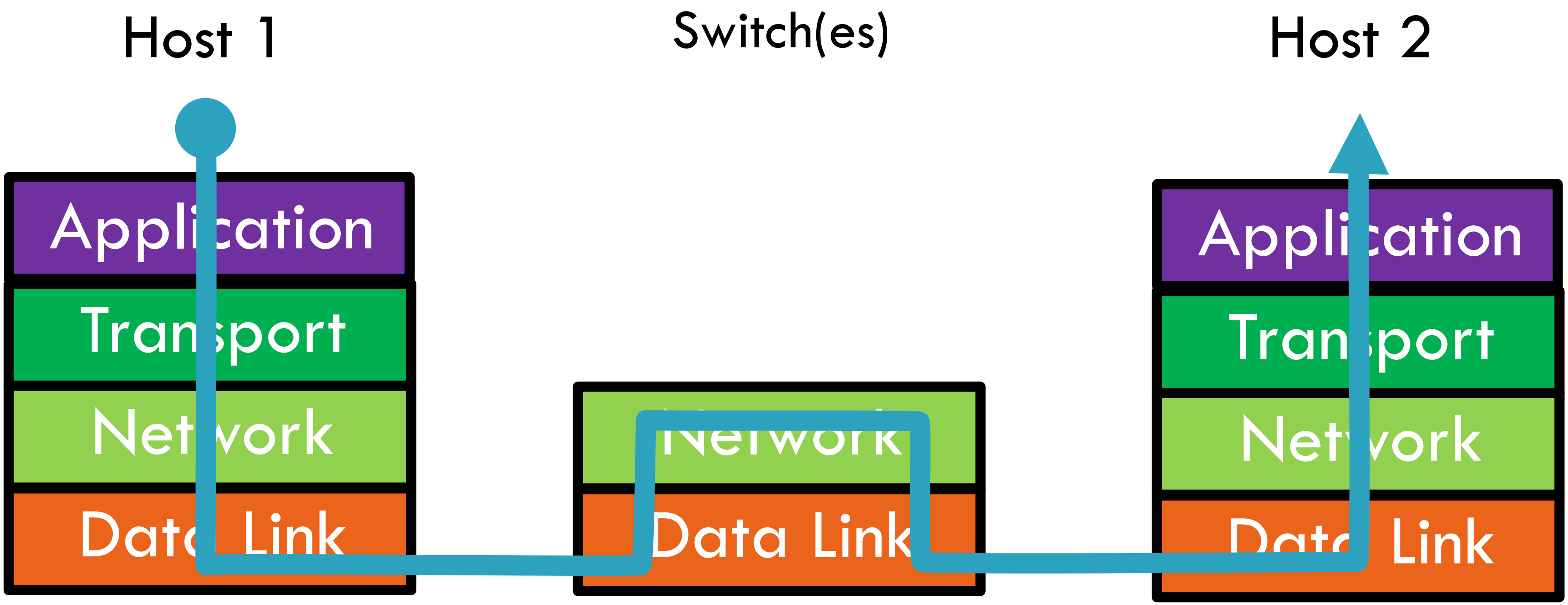
Orthogonal Planes

Control plane: How **Internet paths** are established



Orthogonal Planes

Data plane: How data is **forwarded** over Internet paths



Reality Check

- The layered abstraction is very nice
- Does it hold in reality?

No.

Firewalls

- Analyze application layer headers

Transparent Proxies

- Simulate application endpoints within the network

NATs

- Break end-to-end network reachability

Outline

- ~~Layering~~
  - ~~The OSI Model~~
- Communicating
  - The End-to-End Argument

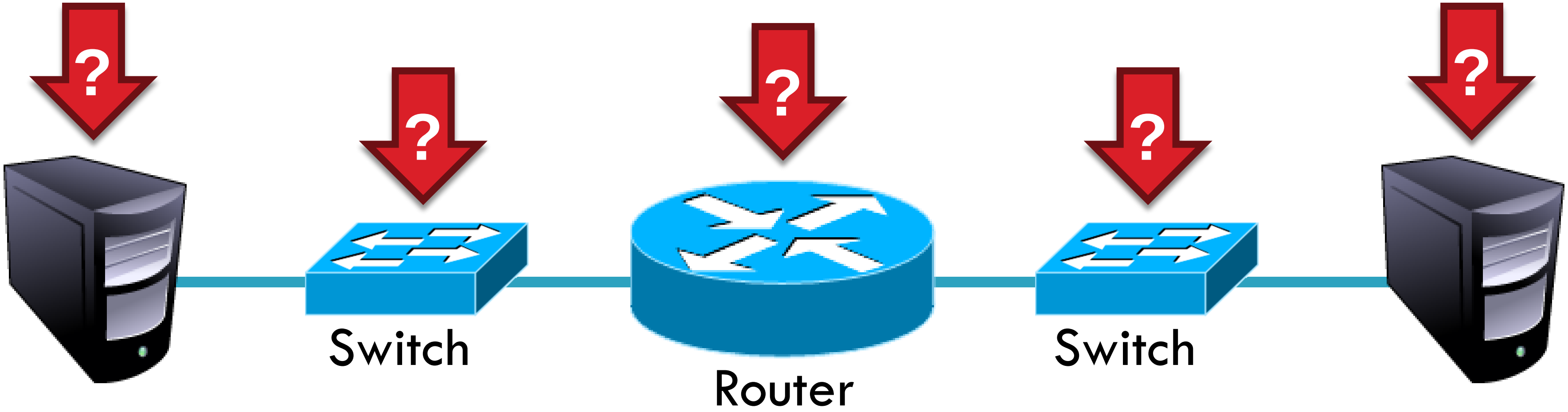
From Layers to Eating Cake

- IP gives us best-effort datagram forwarding
  - So simple anyone can do it
  - Large part of why the Internet has succeeded
  - …but it sure isn't giving us much

- Layers give us a way to **compose** functionality
  - Example: HTTP over TCP for Web browsers with reliable connections
- …but they do not tell us where (in the network) to implement the functionality

Where to Place Functionality

- How do we distribute functionality across devices?
  - Example: who is responsible for security?



- "The End-to-End Arguments in System Design"
  - Saltzer, Reed, and Clark
  - The Sacred Text of the Internet
  - Endlessly debated by researchers and engineers

"END-TO-END ARGUMENTS IN SYSTEM DESIGN"

(*ACM TRANS. ON COMPUTER SYSTEMS*, NOVEMBER 1984)

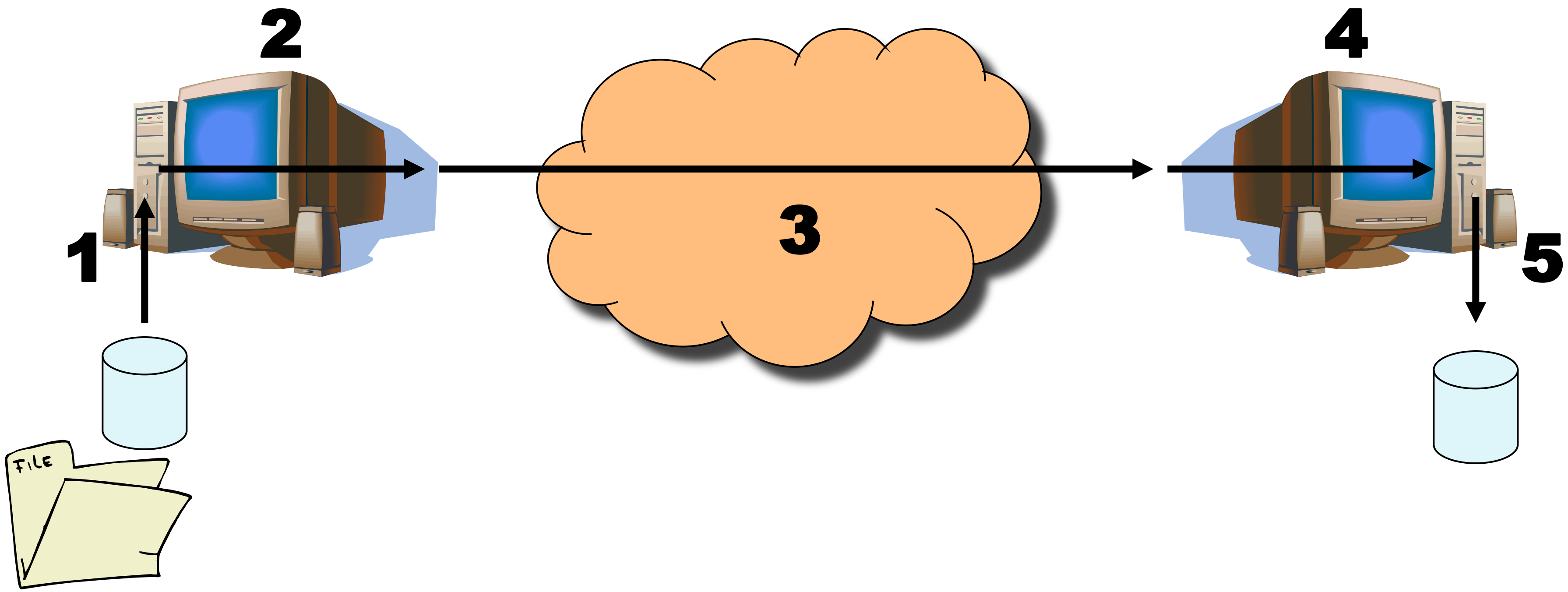
J. Saltzer, D. Reed, and D. Clark

Basic Observation

- Some applications have end-to-end requirements
  - Security, reliability, etc.
- Implementing this stuff inside the network is hard
  - Every step along the way must be fail-proof
  - Different applications have different needs
- End hosts…
  - Can't depend on the network
  - Can satisfy these requirements without network level support

End-to-End Argument

- Operations should occur only at the end points
- … unless needed for performance optimization



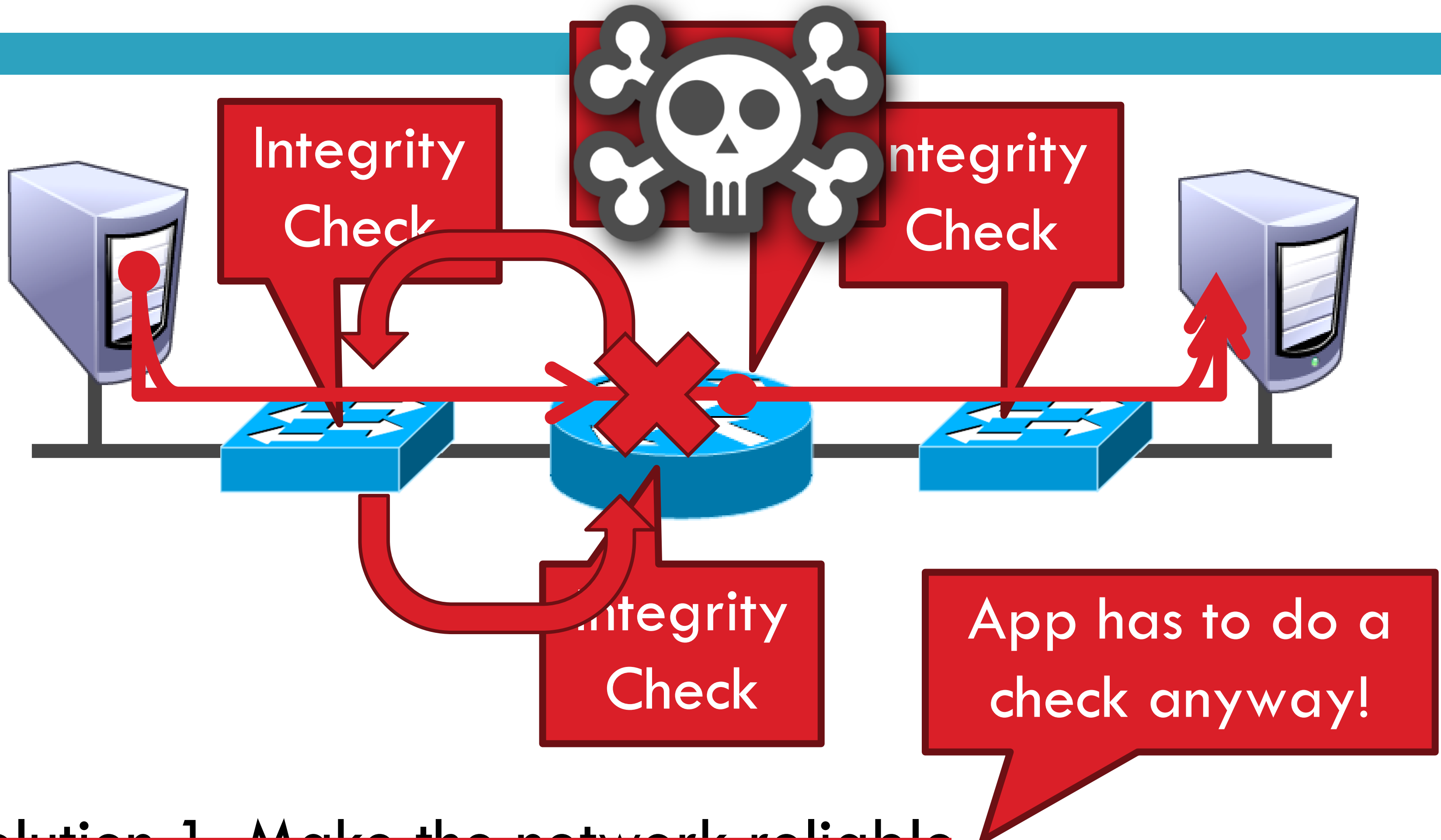
Many things can go wrong: disk errors, software errors, hardware errors, communication errors, …

Tradeoffs

- Put functionality at each hop
  - All applications pay the price
  - End systems *still* need to check for errors
- Place functionality only at the ends
  - Slower error detection
  - End-to-end retransmission wastes bandwidth

- Compromise solution?
  - Reliable end-to-end transport protocol (TCP)
  - Plus file checksums to detect file-system errors

Example: Reliable File Transfer



- ~~Solution 1: Make the network reliable~~
- Solution 2: App level, end-to-end check, retry on failure

Example: Reliable File Transfer

Please

- In-network implementation…
  - Doesn't reduce host complexity
  - Does increase network complexity
  - Increased overhead for apps that don't need functionality
- But, in-network performance may be better

- ~~Solution 1: Make the network reliable~~
- Solution 2: App level, end-to-end check, retry on failure

Conservative Interpretation

"Don't implement a function at the lower levels of the system unless it can be completely implemented at this level" (Peterson and Davie)

Basically, unless you can completely remove the burden from end hosts, don't bother

Radical Interpretation

- Don’t implement anything in the network that can be implemented correctly by the hosts
- Make network layer absolutely minimal
- Ignore performance issues

Moderate Interpretation

- Think twice before implementing functionality in the network
- If hosts can implement functionality correctly, implement it 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…
- …and if it doesn't cost too much $ to implement

Reality Check, Again

- Layering and E2E principals regularly violated

Firewalls

Transparent Proxies

NATs

- Conflicting interests
  - Architectural purity
  - Commercial necessity

Takeaways

- Layering for network functions
  - Helps manage diversity in computer networks
  - Not optimal for everything, but simple and flexible
- Narrow waist ensures interoperability, enables innovation
- E2E argument (attempts) to keep IP layer simple
- Think carefully when adding functionality into the network

More slides …

Questions

- Is a socket between two IP addresses the right abstraction?
  - Mobile hosts?
  - Replicated services?
- What does the network know about the traffic?
  - Inferring the application from the port numbers?
- Is end-to-end error detection and correction the right model?
  - High loss environments?
  - Expense of retransmitting over the entire path?

Organizing Network Functionality

- Networks are built from many components
  - Networking technologies
    - Ethernet, Wifi, Bluetooth, Fiber Optic, Cable Modem, DSL
  - Network styles
    - Circuit switch, packet switch
    - Wired, Wireless, Optical, Satellite
  - Applications
    - Email, Web (HTTP), FTP, BitTorrent, VoIP

- How do we make all this stuff work together?!

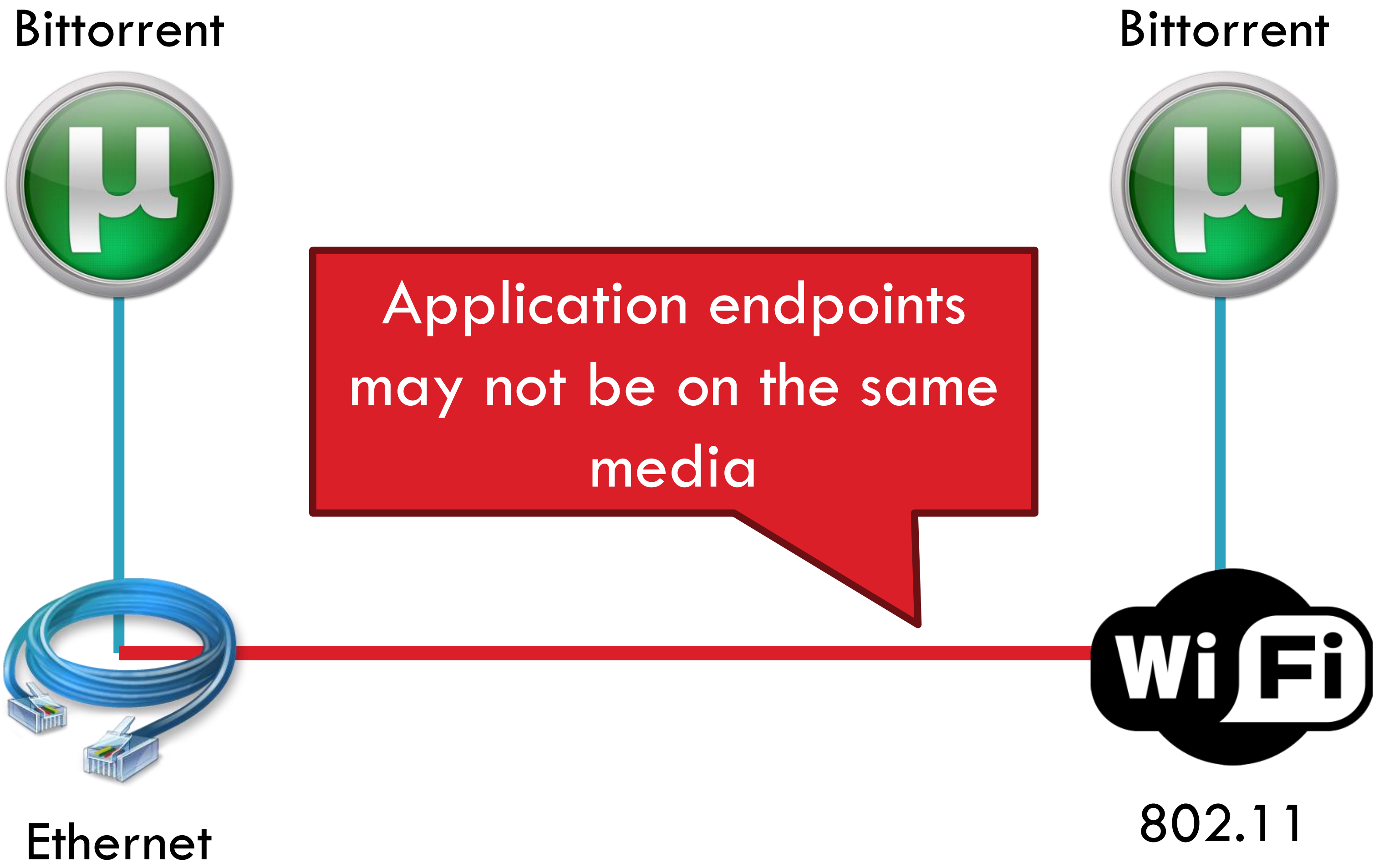
Problem Scenario

Web Email Bittorrent VoIP

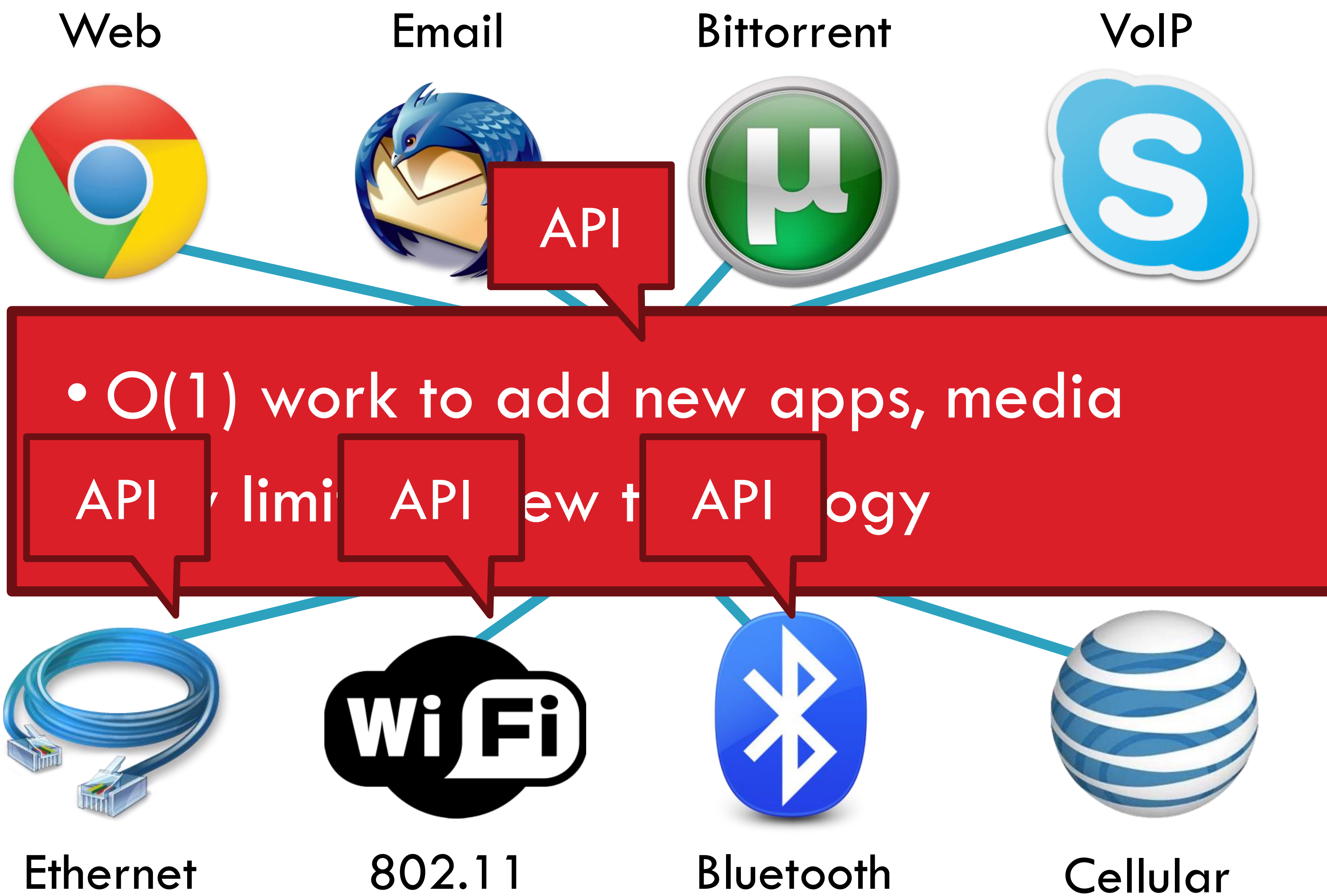
- This is a nightmare scenario
- Huge amounts of work to add new apps or media
- Limits growth and adoption

Ethernet 802.11 Bluetooth Cellular

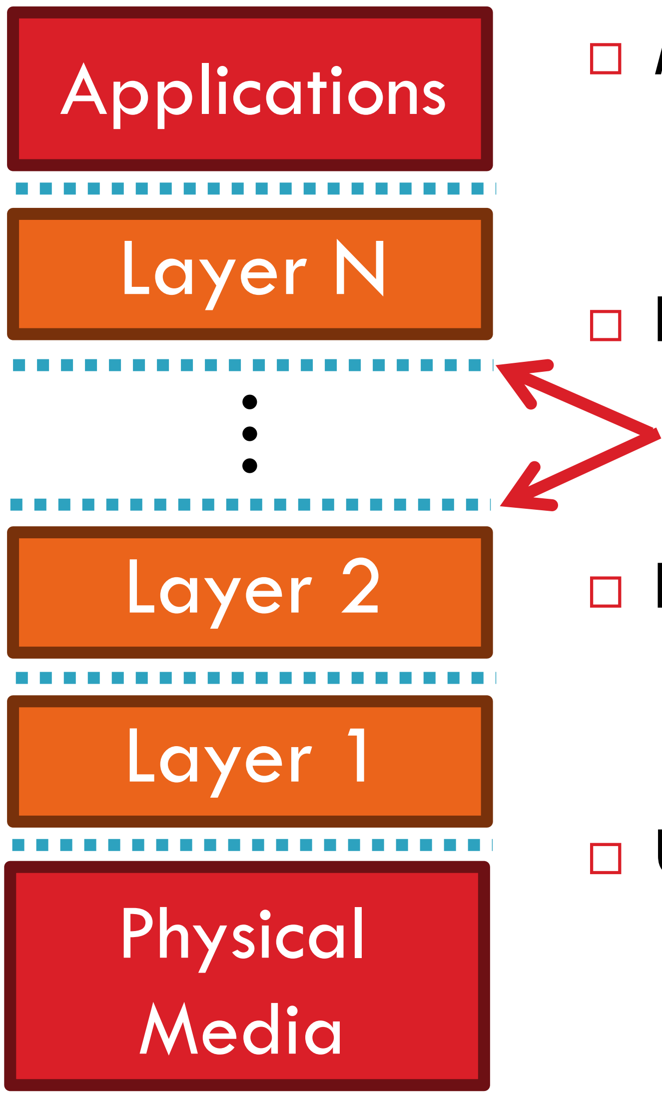
More Problems



Solution: Use Indirection



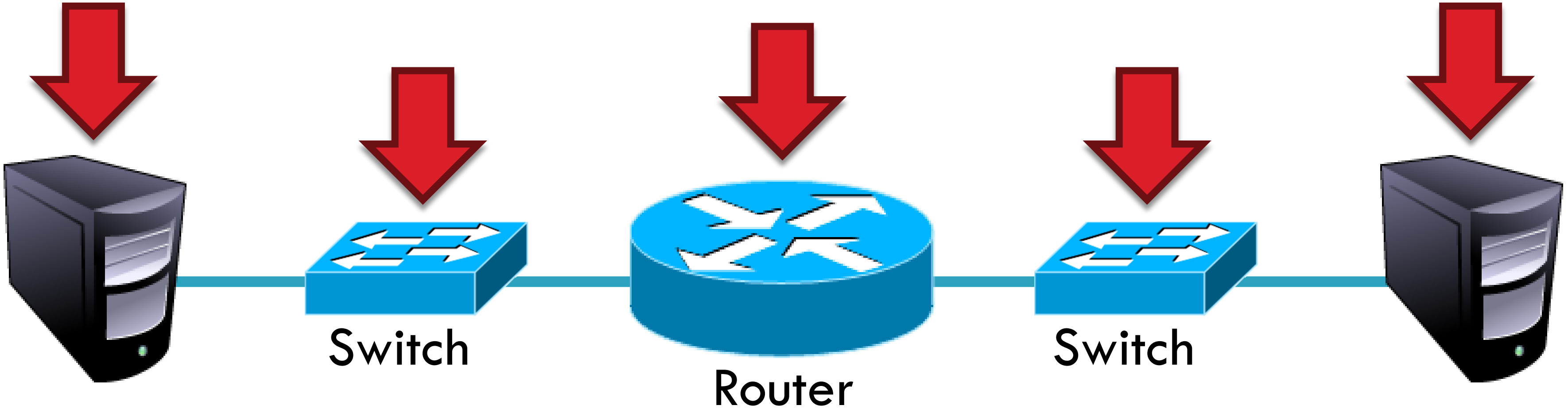
Layered Network Stack



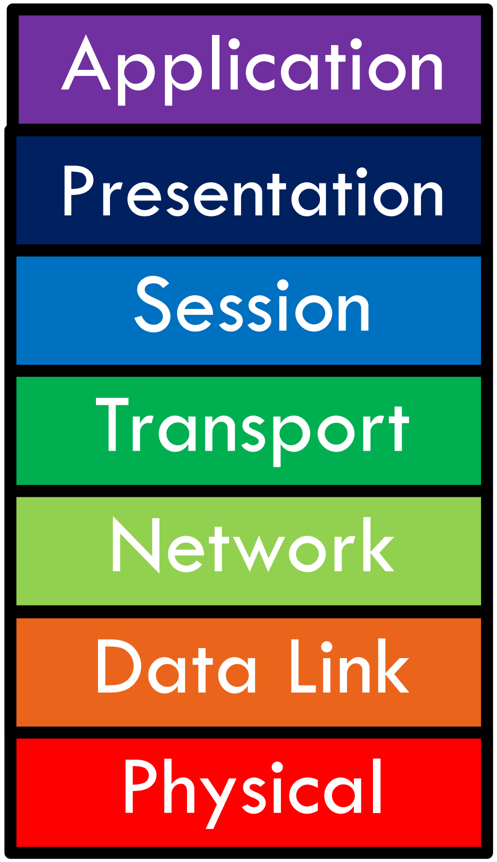
- Modularity
  - Does not specify an implementation
  - Instead, tells us how to organize functionality
- Encapsulation
  - Interfaces define cross-layer interaction
  - Layers only rely on those below them
- Flexibility
  - Reuse of code across the network
  - Module implementations may change
- Unfortunately, there are tradeoffs
  - Interfaces hide information
  - As we will see, may hurt performance…

Key Questions

- How do we divide functionality into layers?
  - Routing
  - Congestion control
  - Error checking
  - Security
  - Fairness
  - And many more…
- How do we distribute functionality across devices?
  - Example: who is responsible for security?

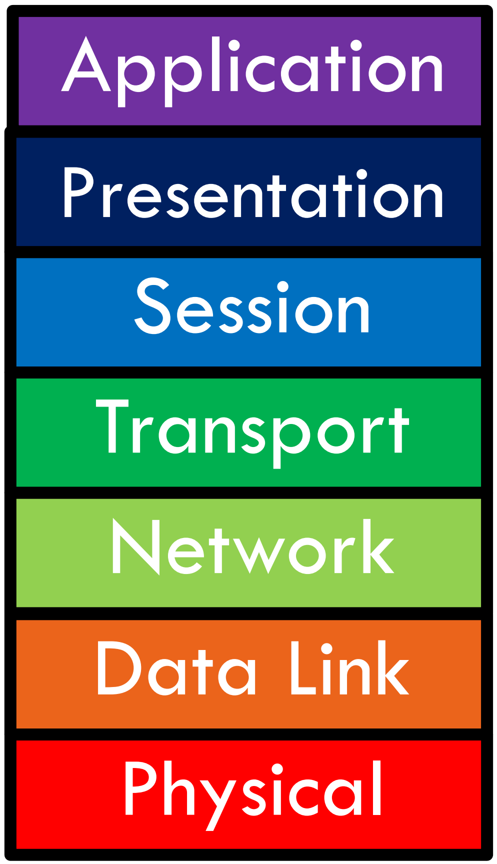


Layer Features



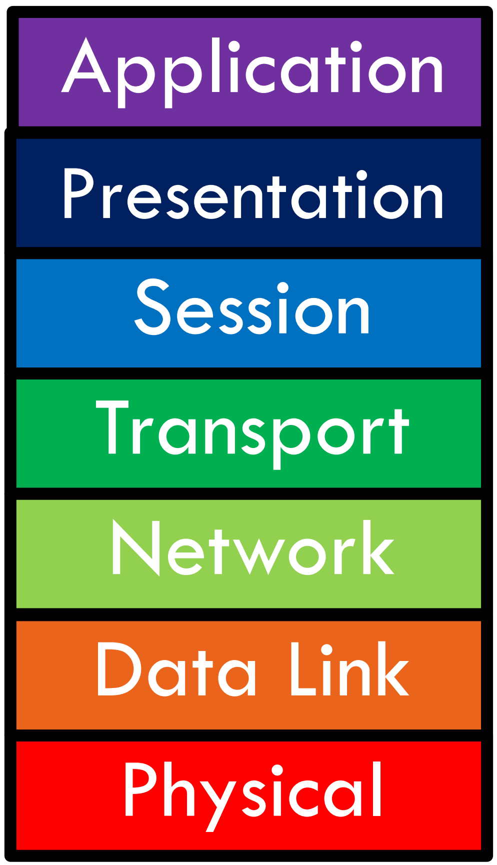
- Service
  - What does this layer do?
- Interface
  - How do you access this layer?
- Protocol
  - How is this layer implemented?

Physical Layer



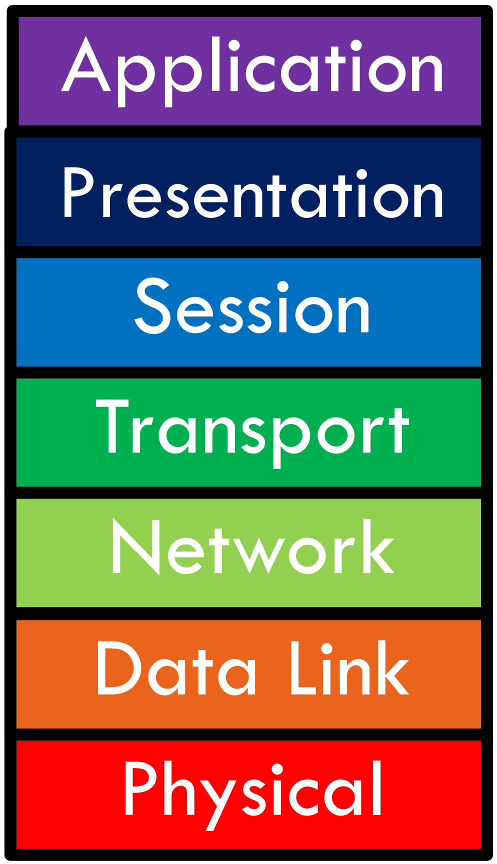
- Service
  - Move information between two systems connected by a physical link
- Interface
  - Specifies how to send one bit
- Protocol
  - Encoding scheme for one bit
  - Voltage levels
  - Timing of signals
- Examples: coaxial cable, fiber optics, radio frequency transmitters

Data Link Layer



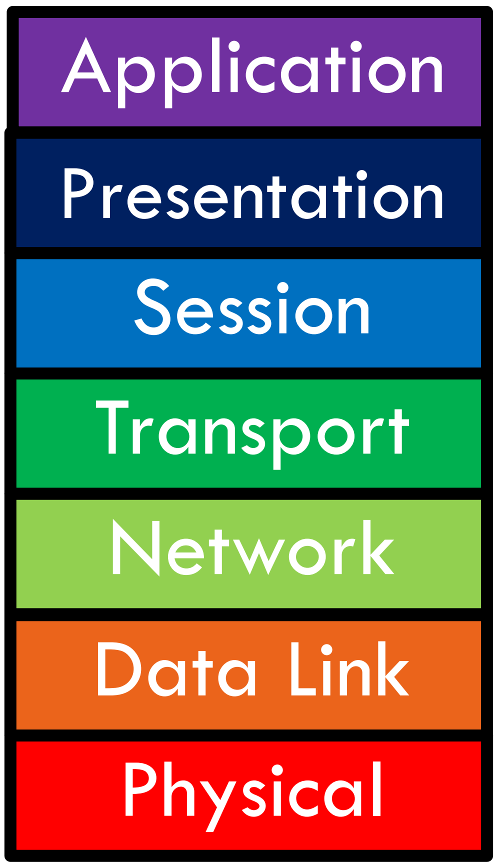
- Service
  - Data framing: boundaries between packets
  - Media access control (MAC)
  - Per-hop reliability and flow-control
- Interface
  - Send one packet between two hosts connected to the same media
- Protocol
  - Physical addressing (e.g. MAC address)
- Examples: Ethernet, Wifi, DOCSIS

Network Layer



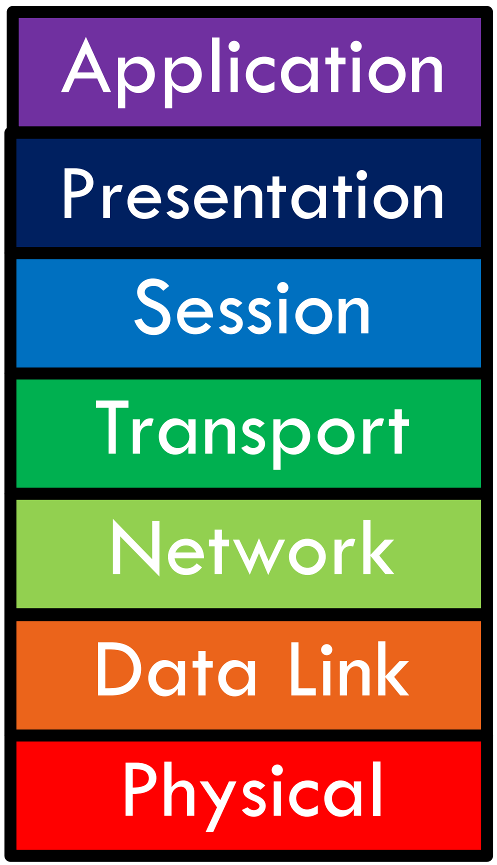
- Service
  - Deliver packets across the network
  - Handle fragmentation/reassembly
  - Packet scheduling
  - Buffer management
- Interface
  - Send one packet to a specific destination
- Protocol
  - Define globally unique addresses
  - Maintain routing tables
- Example: Internet Protocol (IP), IPv6

Transport Layer



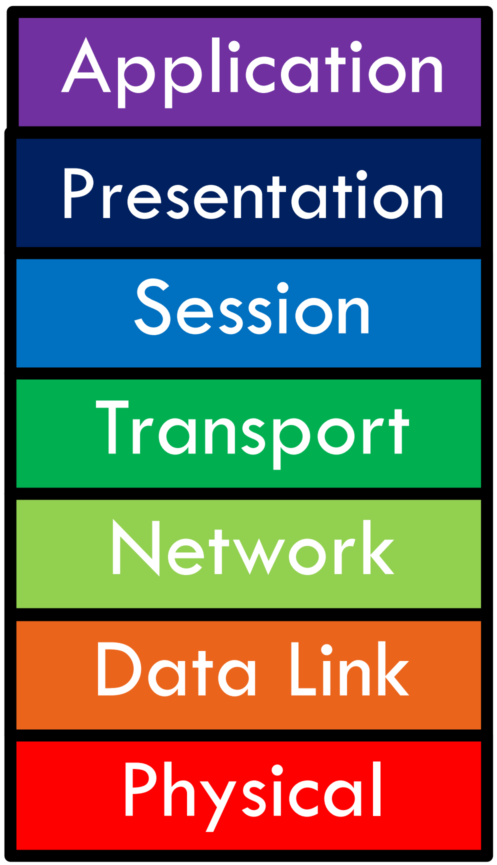
- Service
  - Multiplexing/demultiplexing
  - Congestion control
  - Reliable, in-order delivery
- Interface
  - Send message to a destination
- Protocol
  - Port numbers
  - Reliability/error correction
  - Flow-control information
- Examples: UDP, TCP

Session Layer



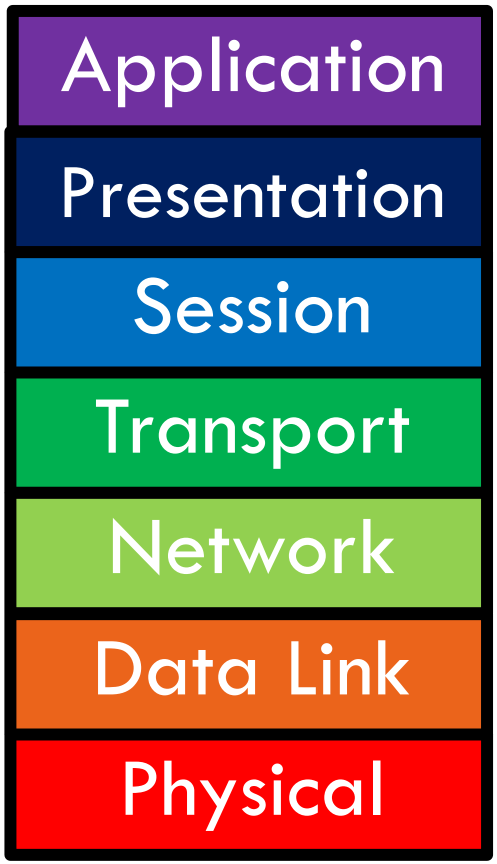
- Service
  - Access management
  - Synchronization
- Interface
  - It depends…
- Protocol
  - Token management
  - Insert checkpoints
- Examples: none

Presentation Layer



- Service
  - Convert data between different representations
  - E.g. big endian to little endian
  - E.g. Ascii to Unicode
- Interface
  - It depends…
- Protocol
  - Define data formats
  - Apply transformation rules
- Examples: none

Application Layer



- Service
  - Whatever you want :)
- Interface
  - Whatever you want :D
- Protocol
  - Whatever you want ;)
- Examples: turn on your smartphone and look at the list of apps