CSCI-351
Network Fundamentals

Lecture 3: Internet Architecture (Layer cake and an hourglass)

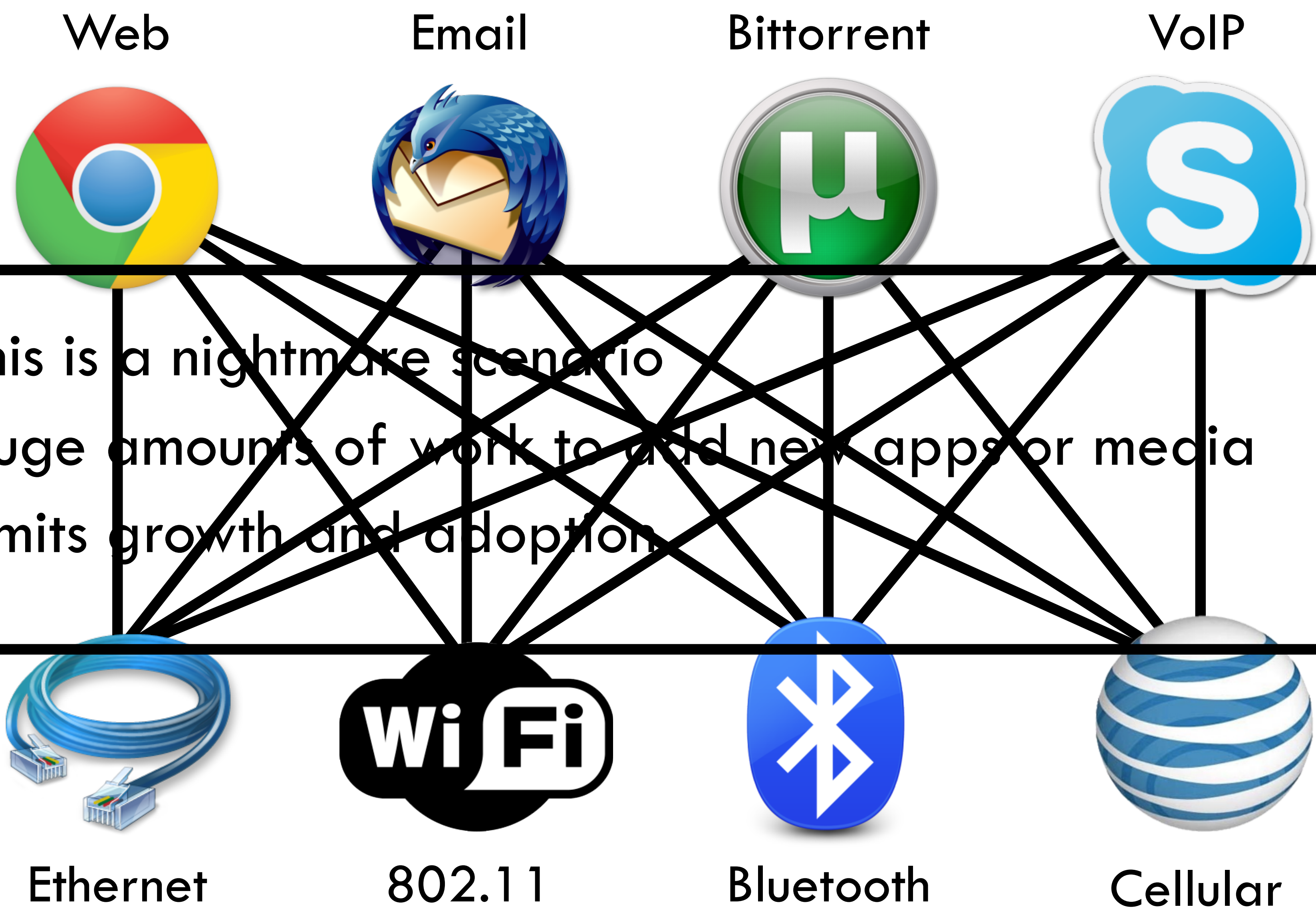
The slide is built with the help of Prof. Alan Mislove, Christo Wilson, and David Choffnes's class

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

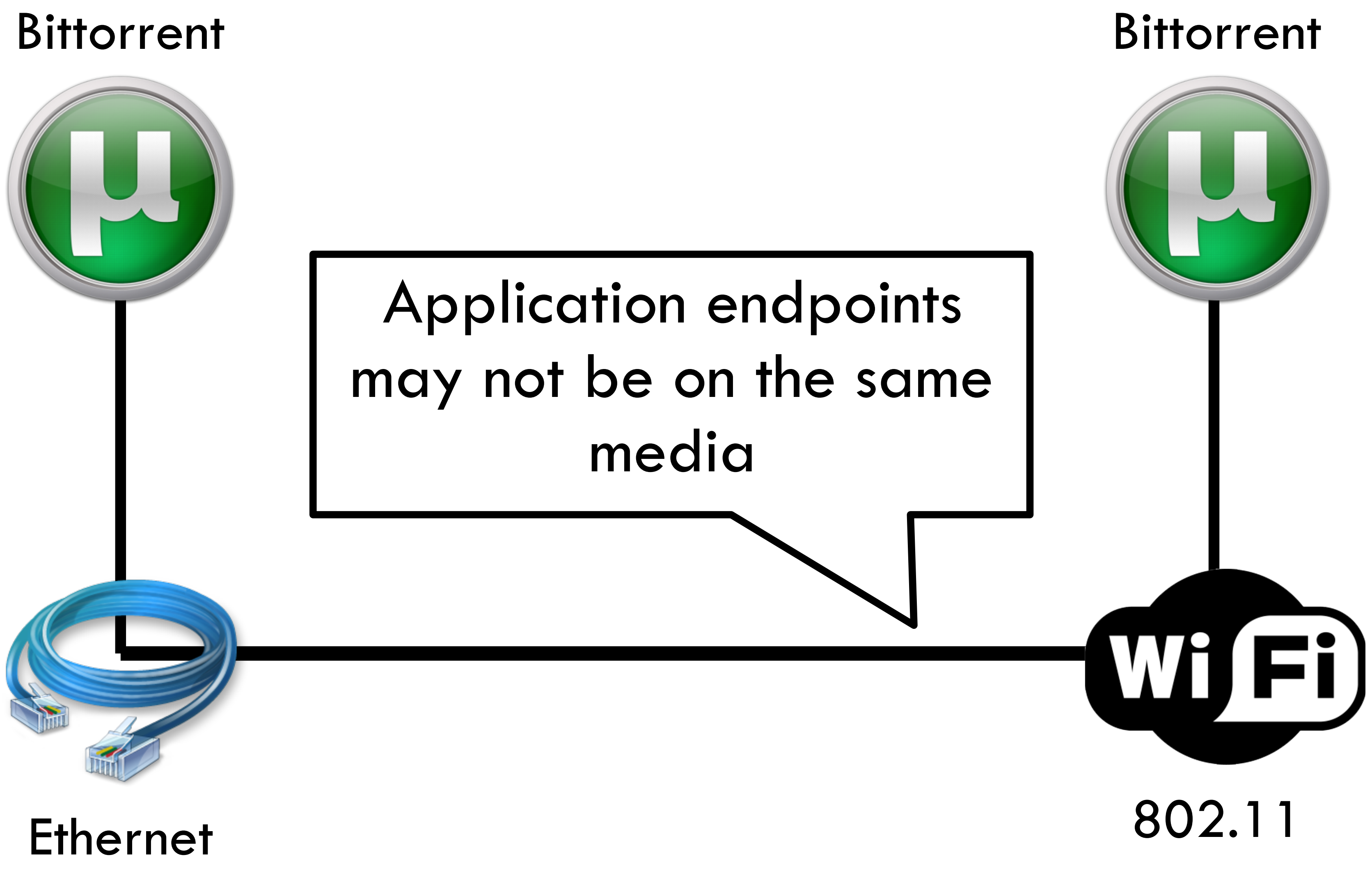


This is a nightmare scenario

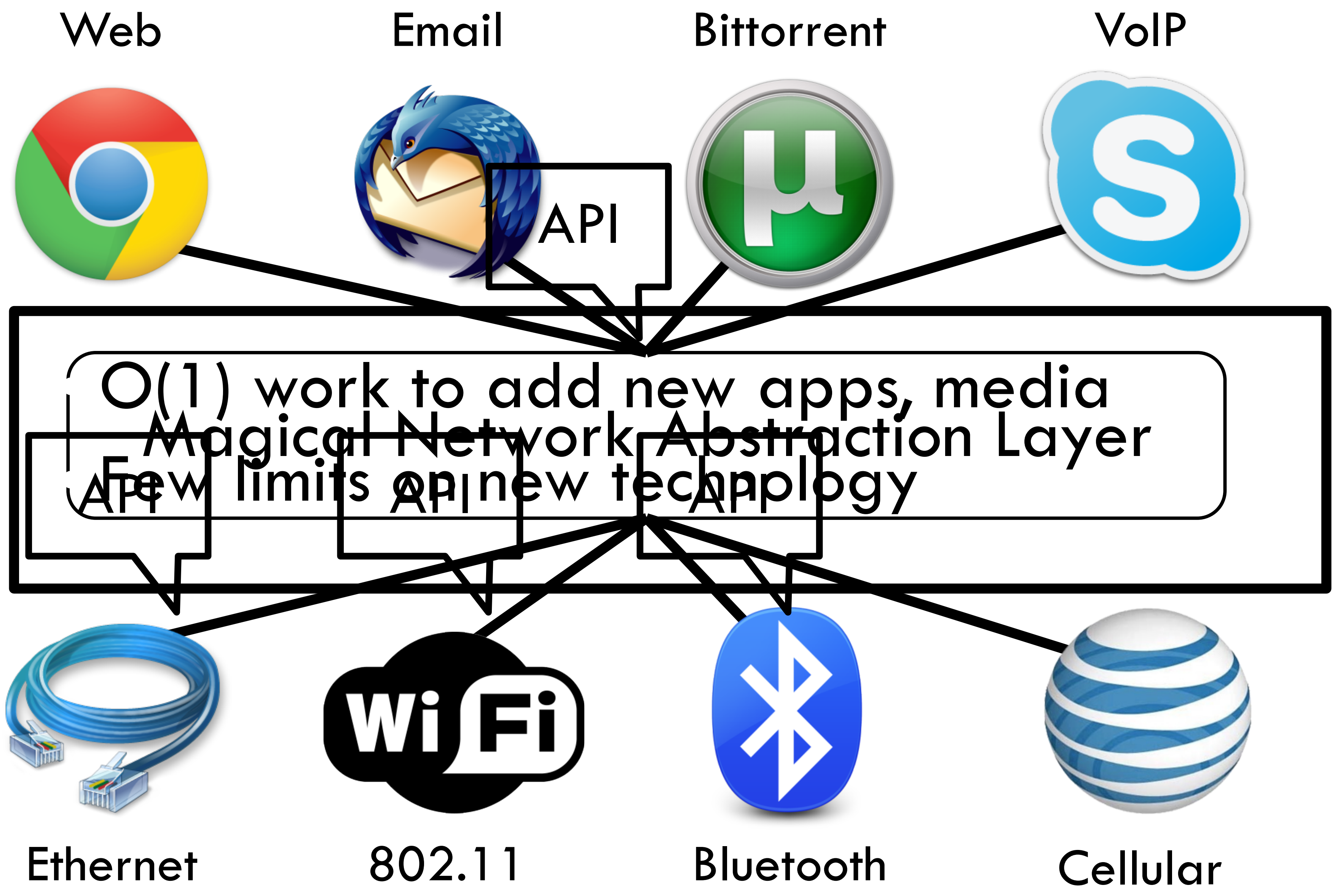
Huge amounts of work to add new apps or media

Limits growth and adoption

More Problems



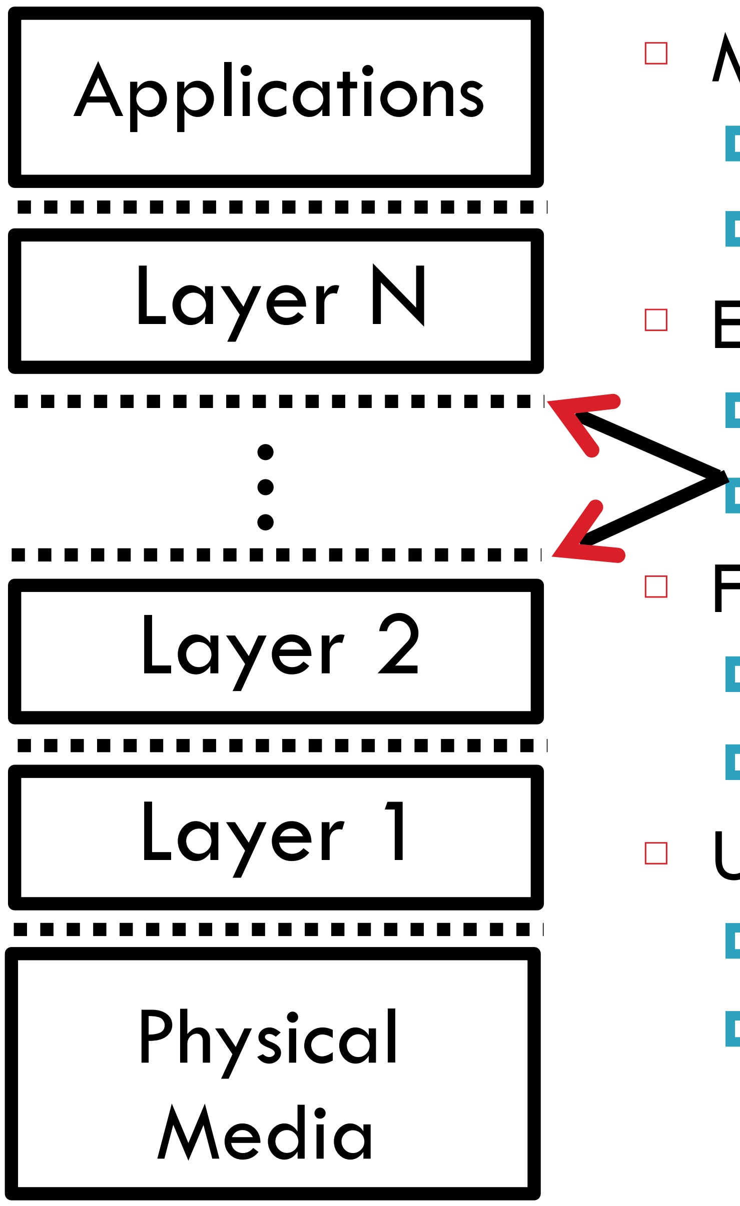
Solution: Use Indirection



O(1) work to add new apps, media

Few limits on new technology

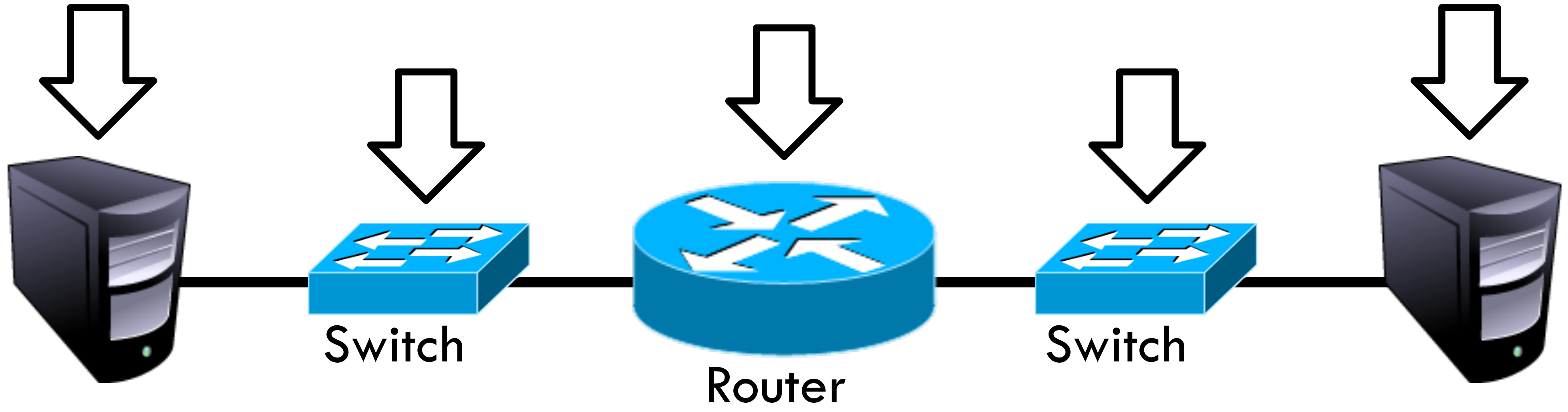
Layered Network Stack



- 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?

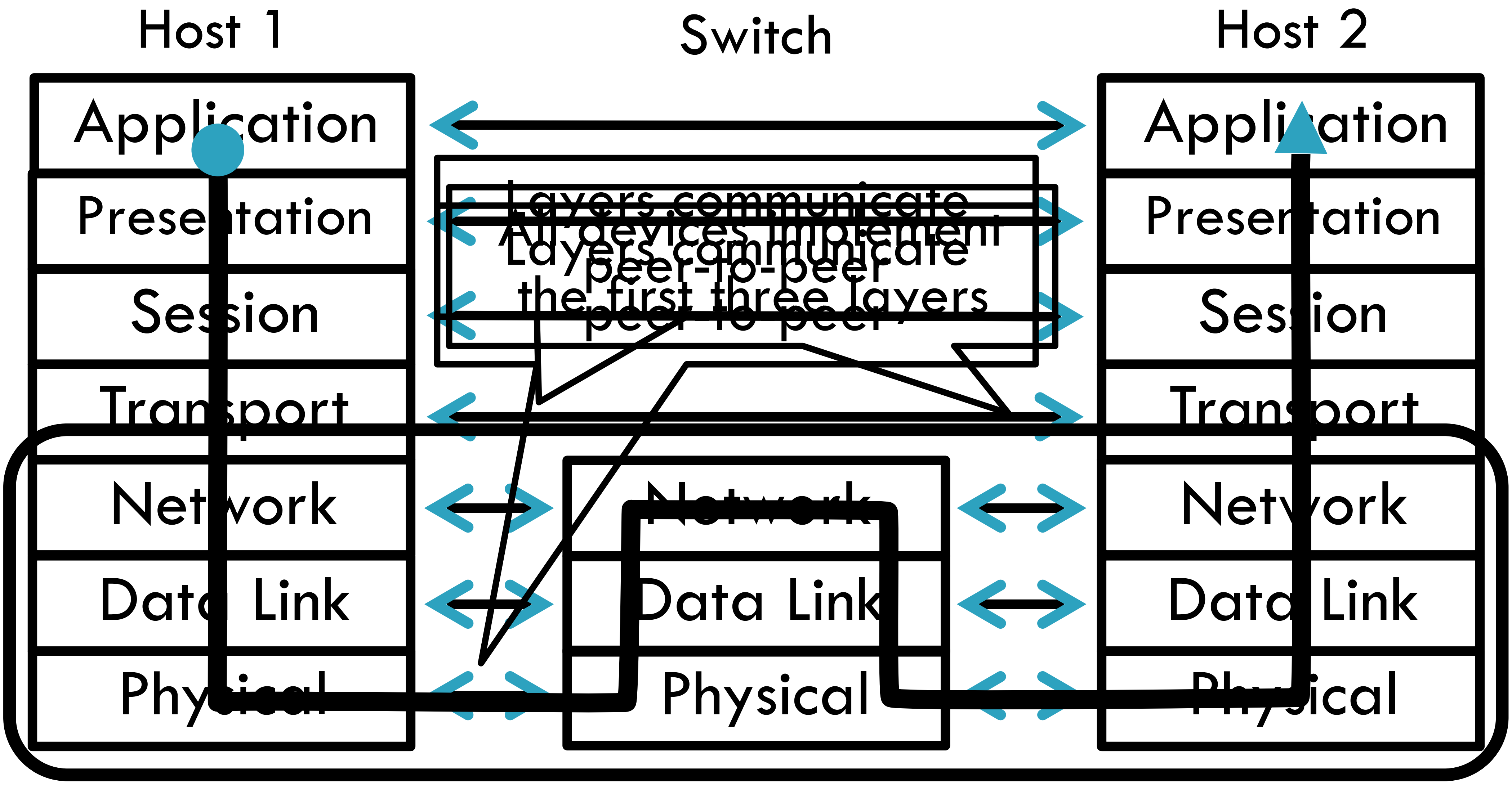


Outline

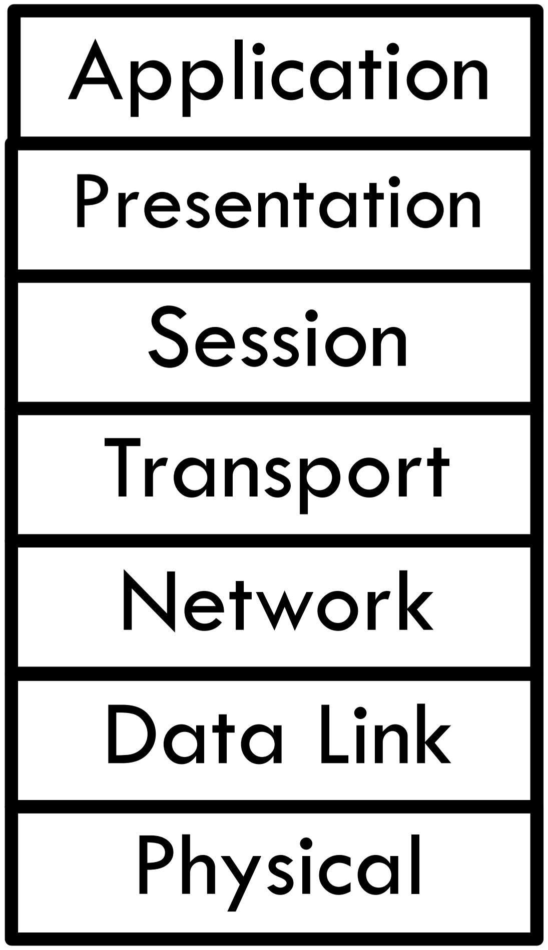
- Layering
  - The OSI Model
- Communicating
  - The End-to-End Argument

The ISO OSI Model

OSI: Open Systems Interconnect Model

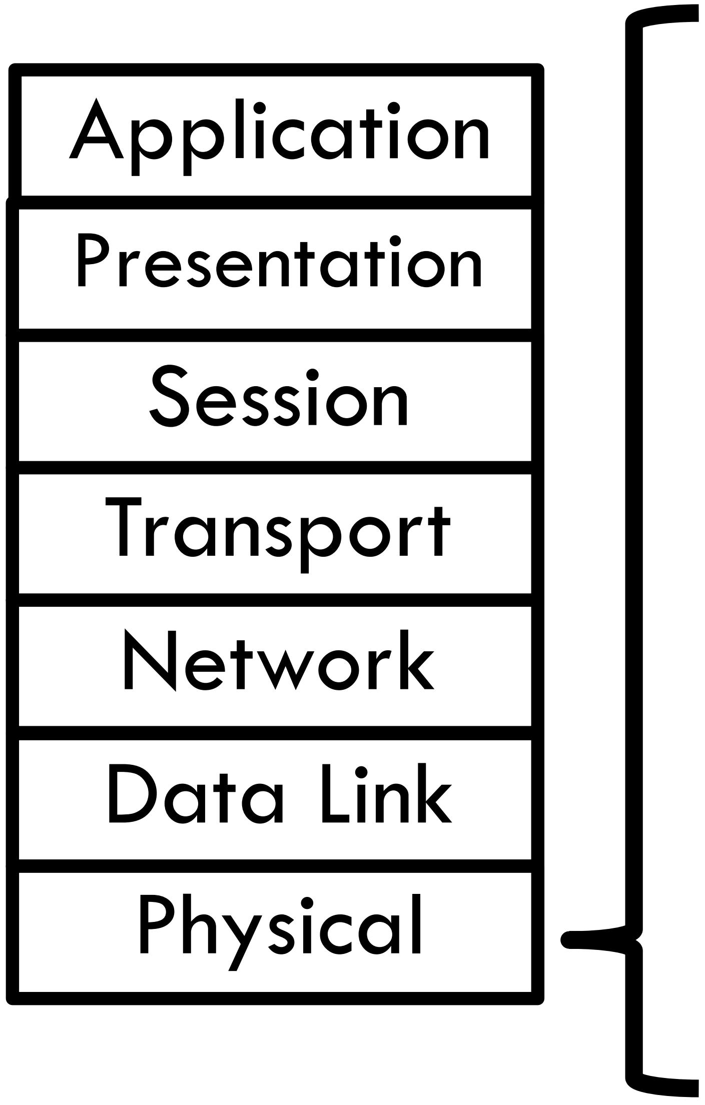


Layer Features



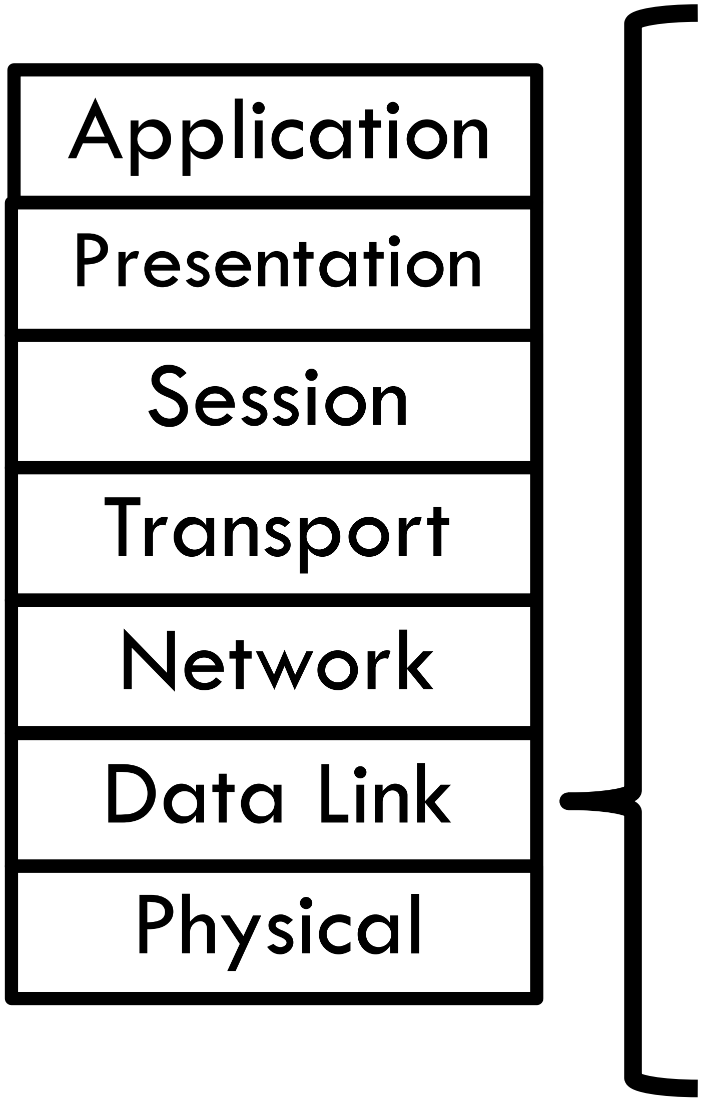
- Service
  - What does this layer do?
- Interface
  - How do you access this layer?
- Protocol
  - How is this layer implemented?

Physical Layer



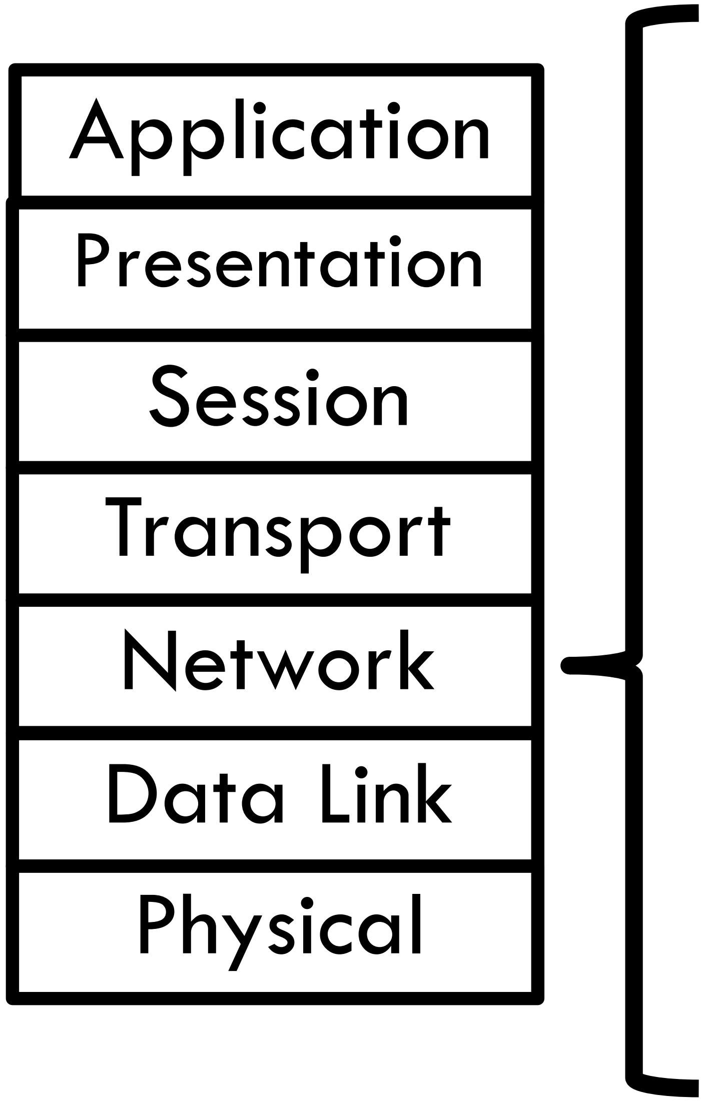
- Service
  - Move information between two systems connected by a physical link
- Interface
  - Specifies how to send one bit
- Protocol
  - Encoding scheme for one bit
  - Voltage levels
  - Timing of signals
- Examples: coaxial cable, fiber optics, radio frequency transmitters

Data Link Layer



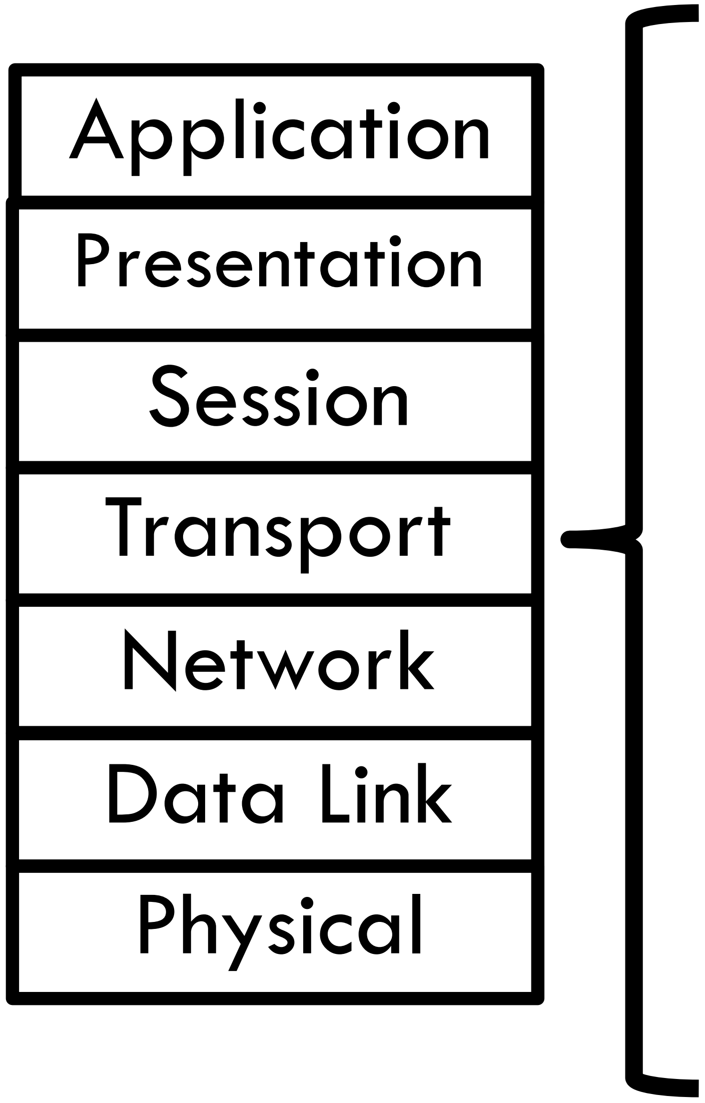
- Service
  - Data framing: boundaries between packets
  - Media access control (MAC)
  - Per-hop reliability and flow-control
- Interface
  - Send one packet between two hosts connected to the same media
- Protocol
  - Physical addressing (e.g. MAC address)
- Examples: Ethernet, Wifi, DOCSIS

Network Layer



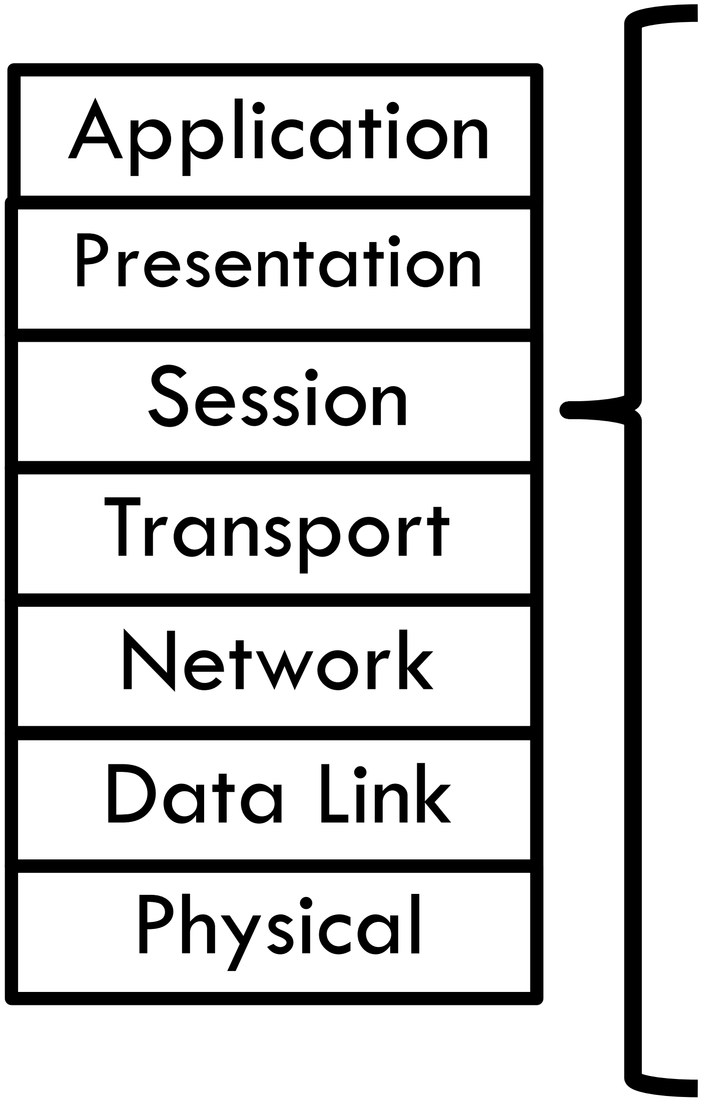
- Service
  - Deliver packets across the network
  - Handle fragmentation/reassembly
  - Packet scheduling
  - Buffer management
- Interface
  - Send one packet to a specific destination
- Protocol
  - Define globally unique addresses
  - Maintain routing tables
- Example: Internet Protocol (IP), IPv6

Transport Layer



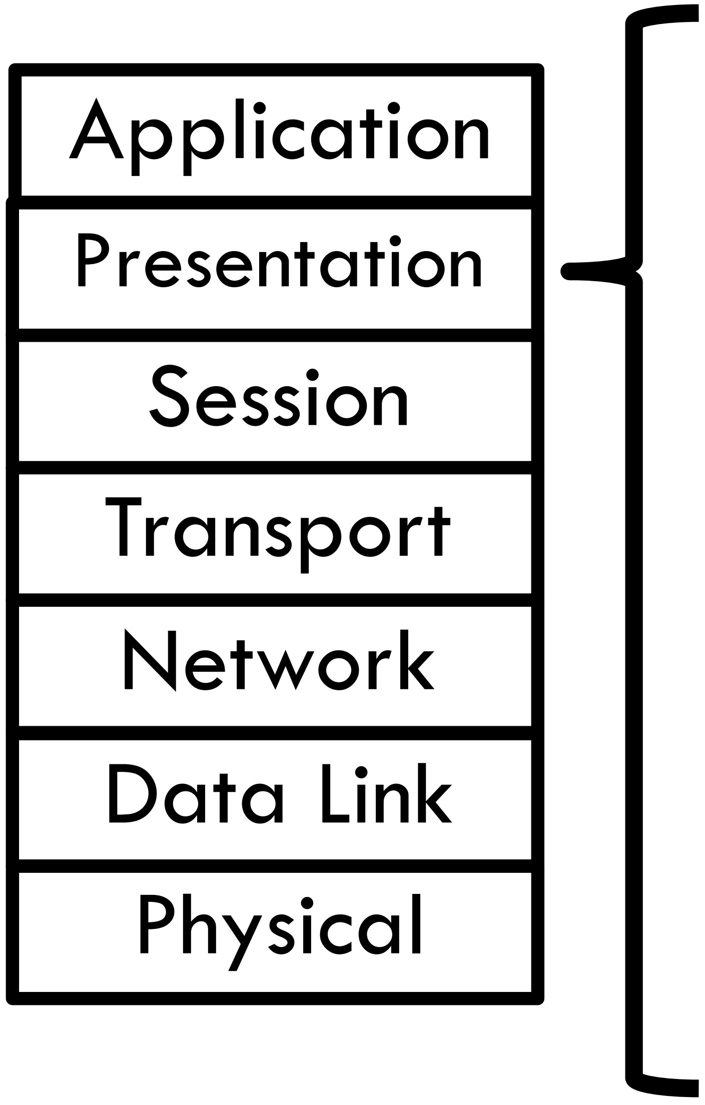
- Service
  - Multiplexing/demultiplexing
  - Congestion control
  - Reliable, in-order delivery
- Interface
  - Send message to a destination
- Protocol
  - Port numbers
  - Reliability/error correction
  - Flow-control information
- Examples: UDP, TCP

Session Layer



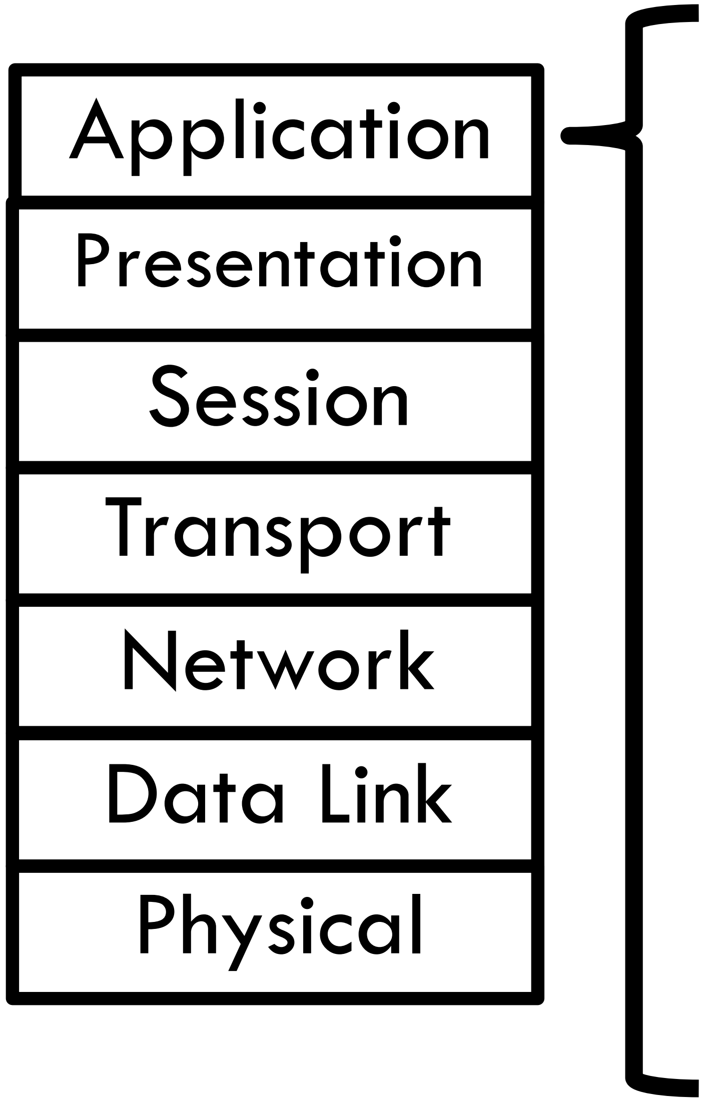
- Service
  - Access management
  - Synchronization
- Interface
  - It depends…
- Protocol
  - Token management
  - Insert checkpoints
- Examples: none

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

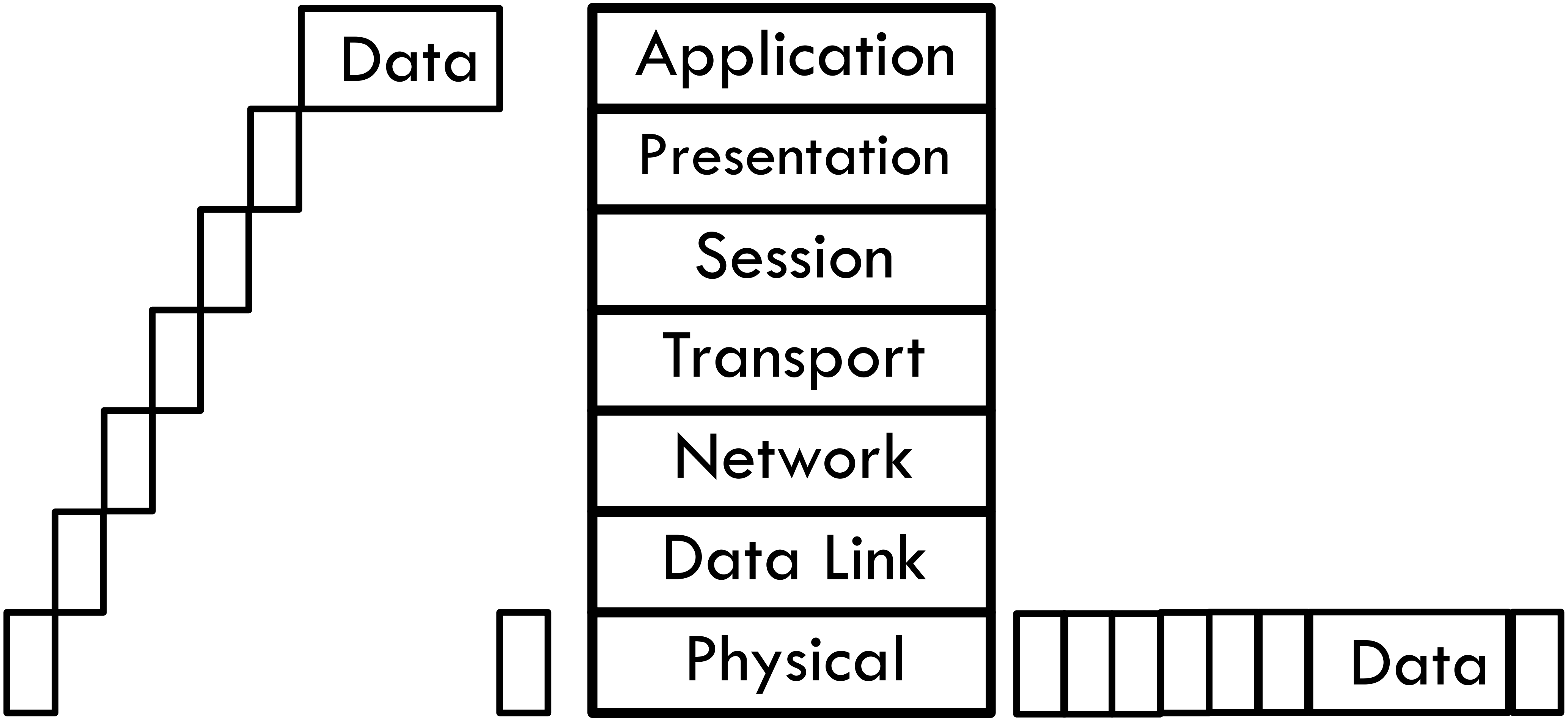
Application Layer



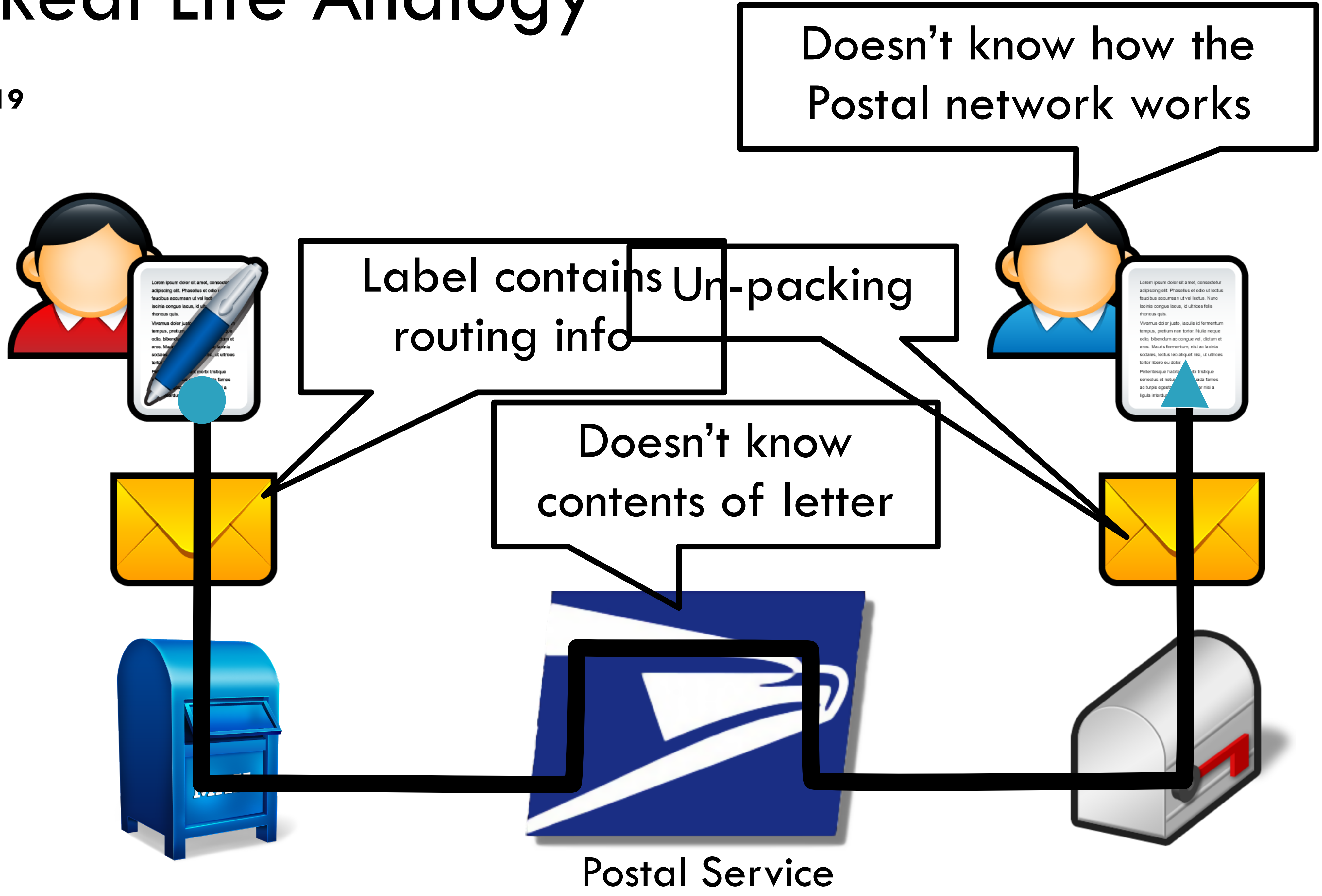
- Service
  - Whatever you want :)
- Interface
  - Whatever you want :D
- Protocol
  - Whatever you want ;)
- Examples: turn on your smartphone and look at the list of apps

Encapsulation

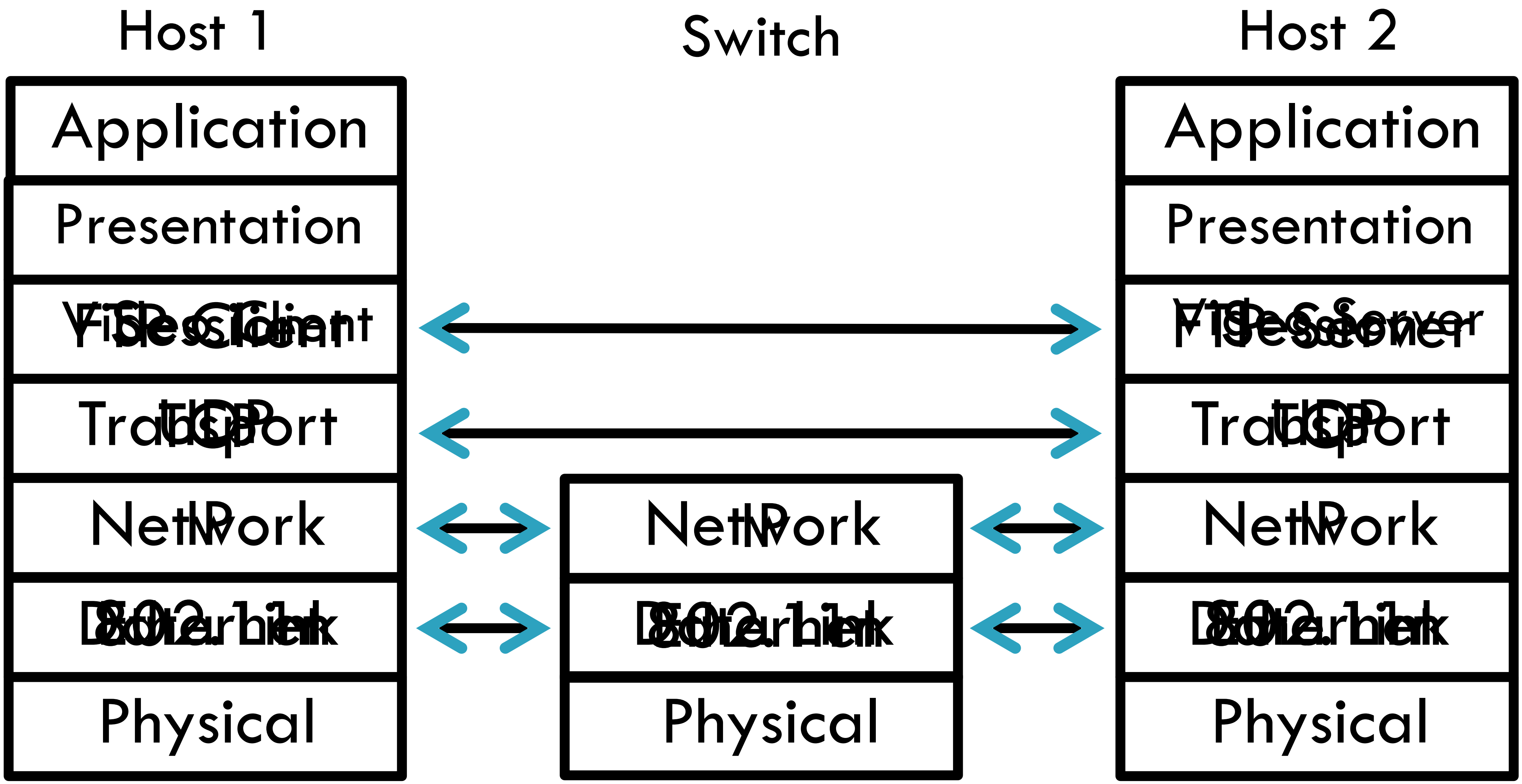
How does data move through the layers?



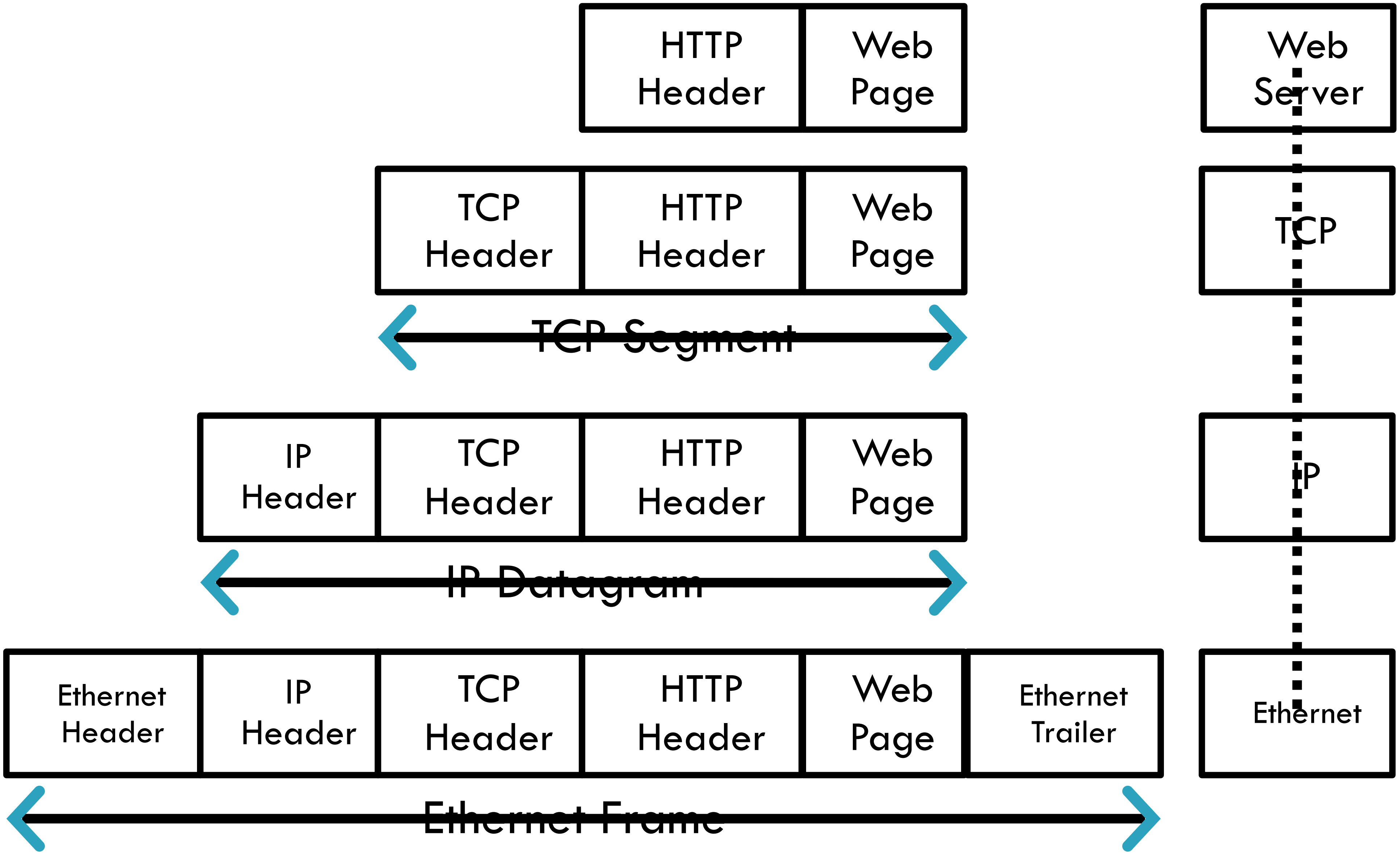
Real Life Analogy


Doesn't know how the Postal network works
Label contains routing info
Un-packing
Doesn't know contents of letter
Postal Service

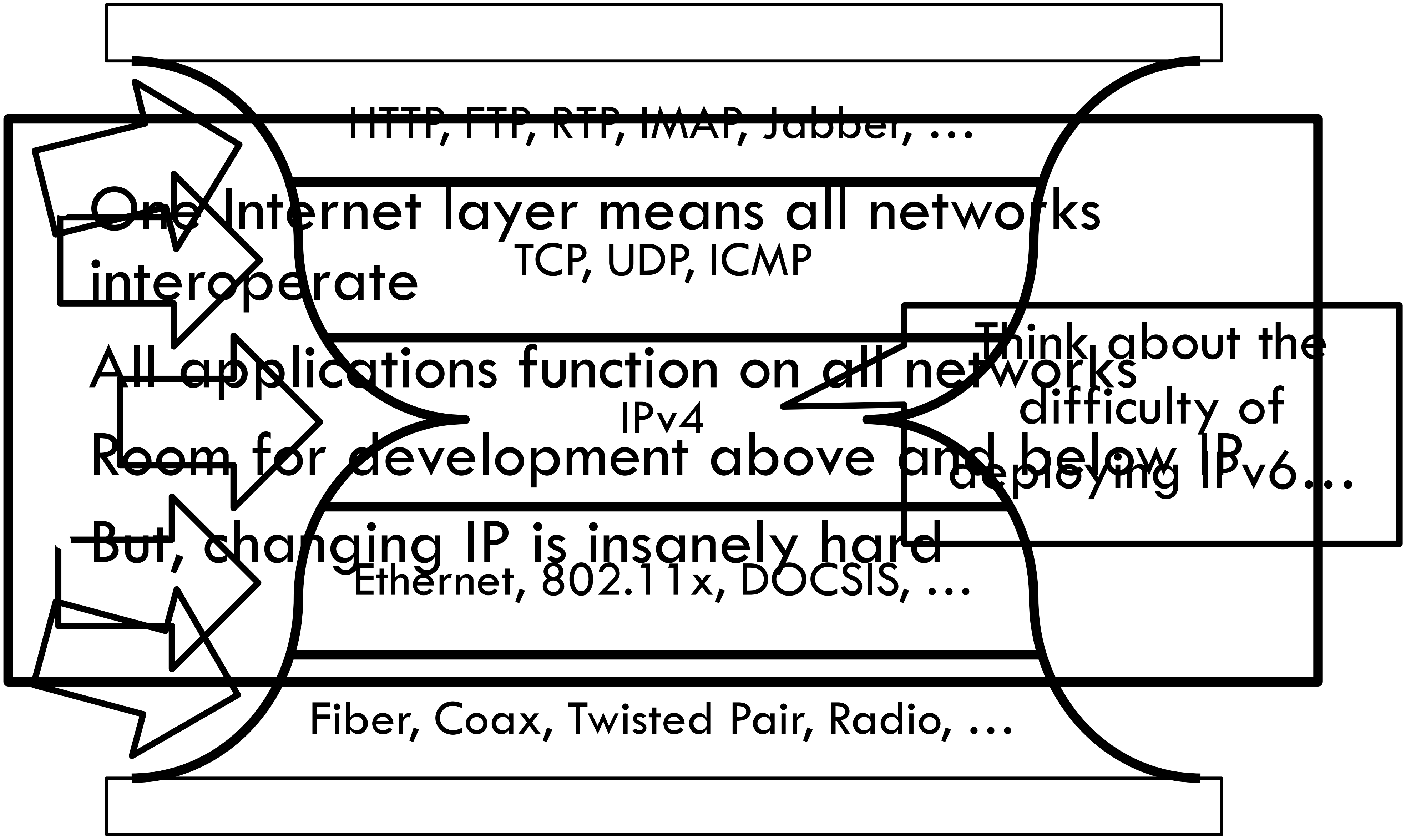
Network Stack in Practice



Encapsulation, Revisited



The Hourglass

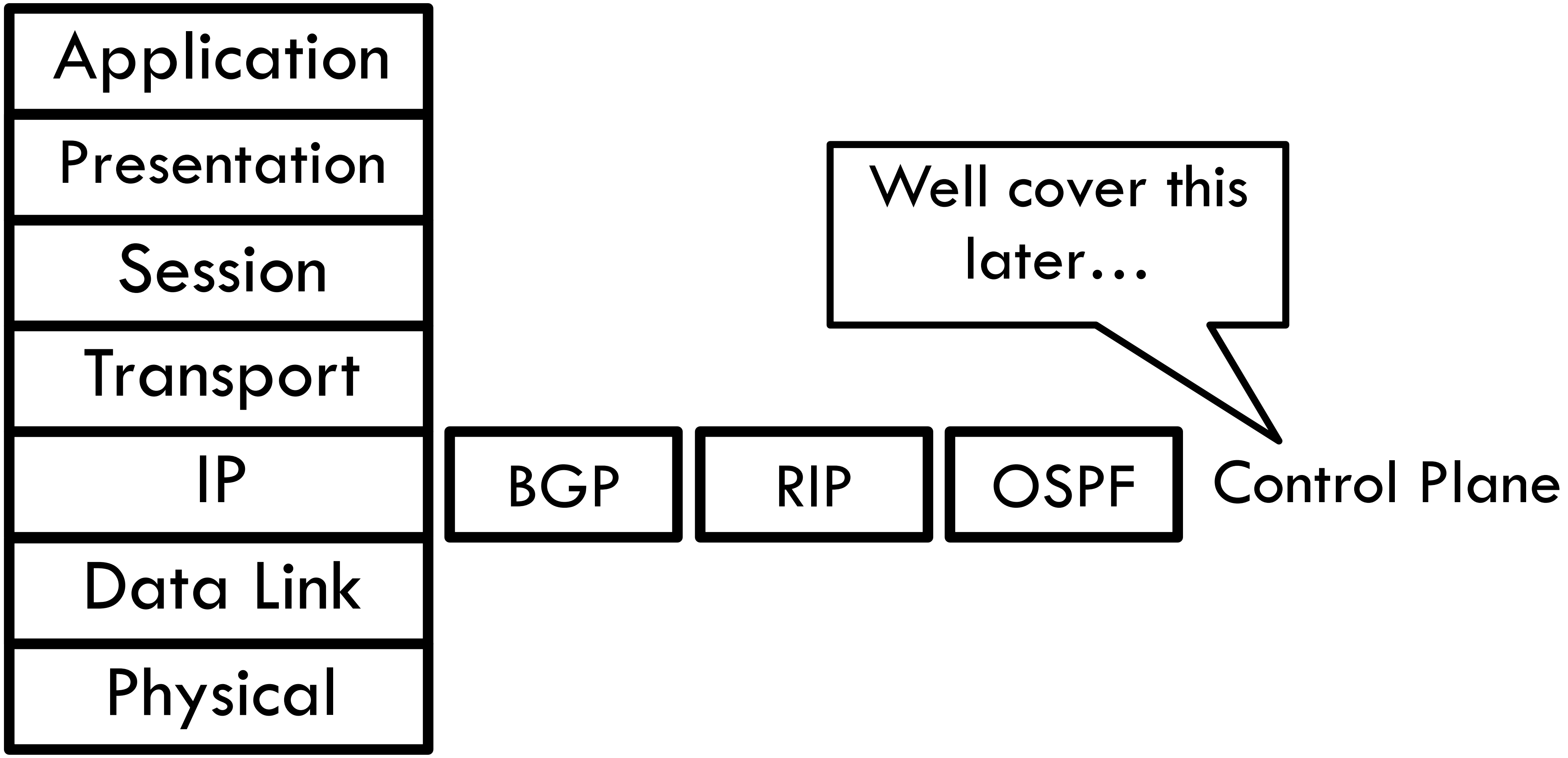

HTTP, FTP, RTP, IMAP, Jabber, …
TCP, UDP, ICMP
IPv4
Ethernet, 802.11x, DOCSIS, …
Fiber, Coax, Twisted Pair, Radio, …

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

Think about the difficulty of deploying IPv6…

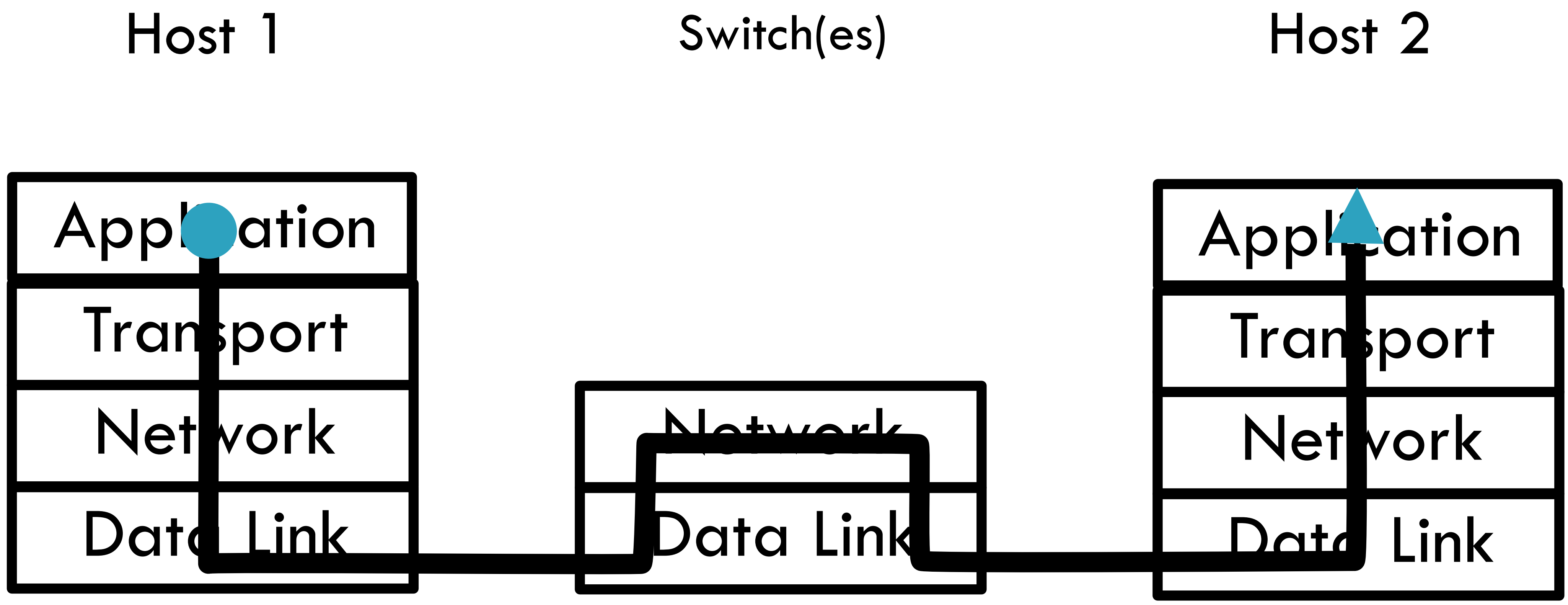
Orthogonal Planes

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

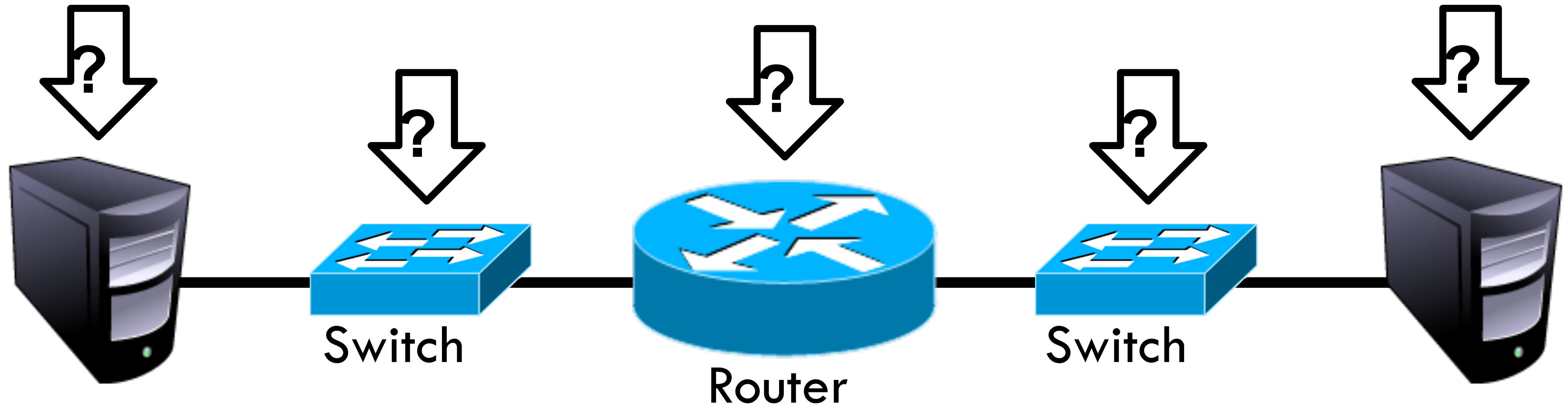
From Layers to Eating Cake

- 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?

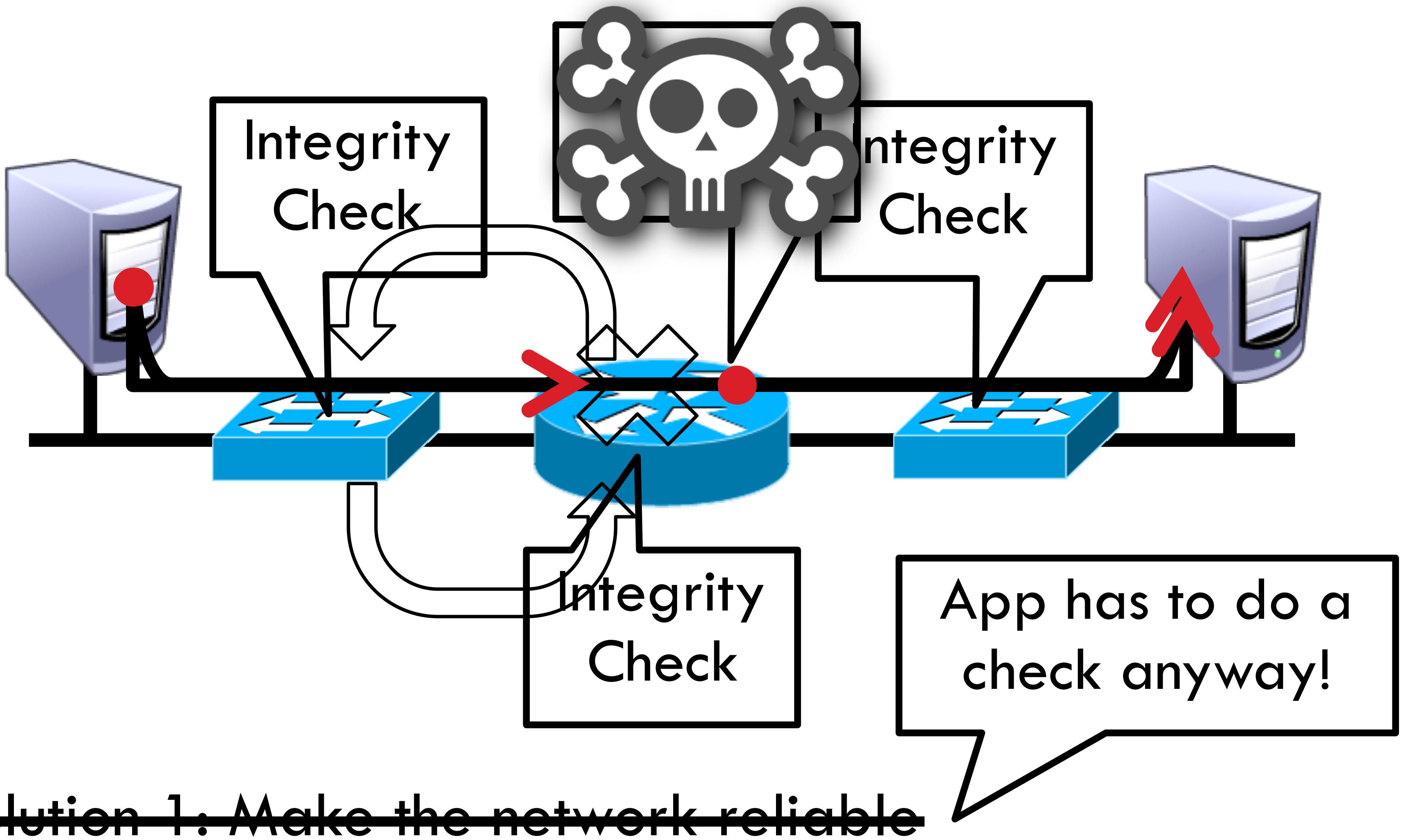


- "The End-to-End Arguments in System Design"
  - Saltzer, Reed, and Clark
  - The Sacred Text of the Internet
  - Endlessly debated by researchers and engineers

Basic Observation

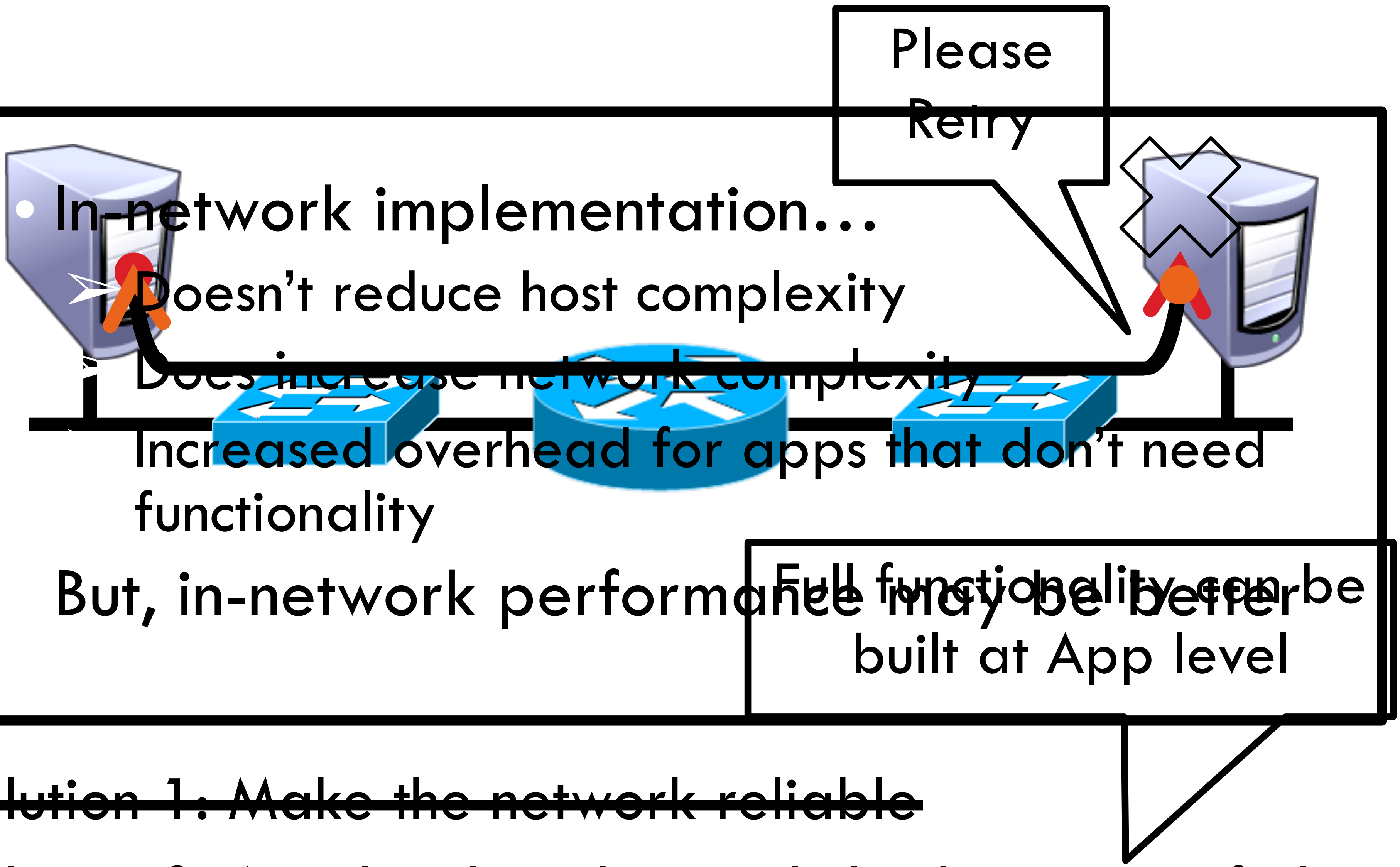
- Some applications have end-to-end requirements
  - Security, reliability, etc.
- Implementing this stuff inside the network is hard
  - Every step along the way must be fail-proof
  - Different applications have different needs
- End hosts…
  - Can't depend on the network
  - Can satisfy these requirements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

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



- 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

Pros vs. Cons?

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