Data Networks

UdS and IMPRS-CS

Lecture 3: Internet architecture

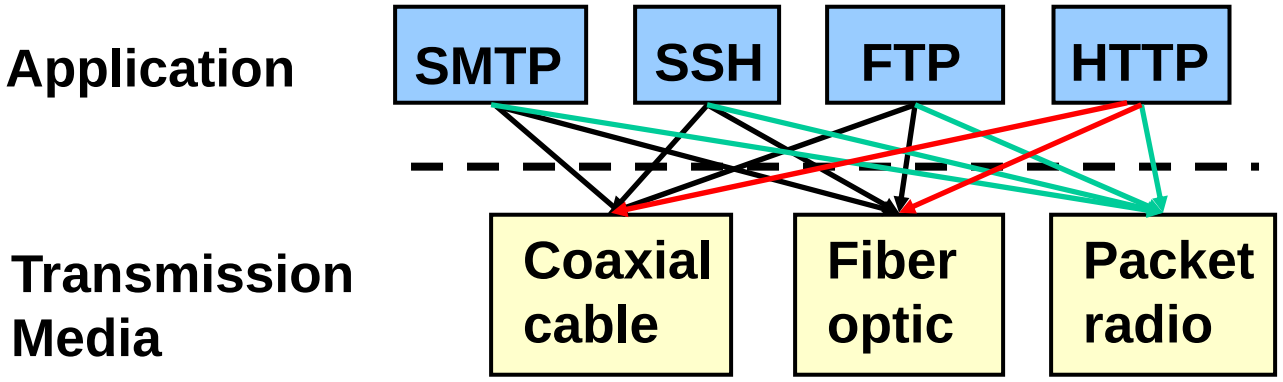
Thanks to Eugene Ng, Rice University

Organizing Network Functionality

- Many kinds of networking functionality
  - e.g., encoding, framing, routing, addressing, reliability, etc.
- Many different network styles and technologies
  - circuit-switched vs packet-switched, etc.
  - wireless vs wired vs optical, etc.
- Many different applications
  - ftp, email, web, P2P, etc.

- Network architecture
  - How should different pieces be organized?
  - How should different pieces interact?

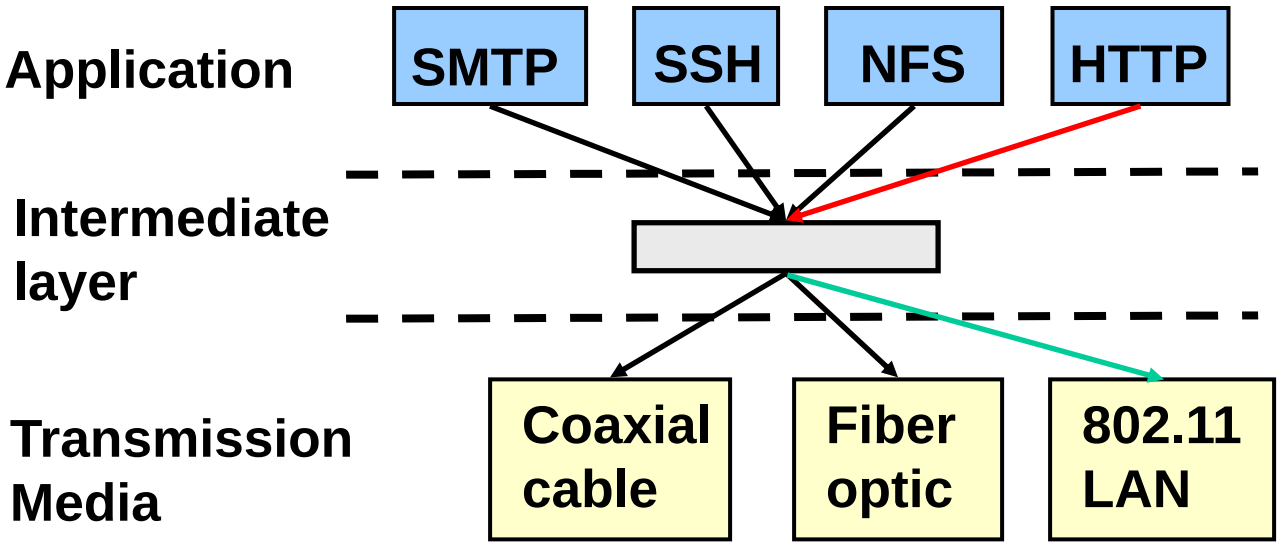
Problem



- new application has to interface to all existing media
  - adding new application requires O(m) work, m = number of media
- new media requires all existing applications be modified
  - adding new media requires O(a) work, a = number of applications
- total work in system O(ma) → eventually too much work to add apps/media
- Application end points may not be on the same media!

Solution: Indirection

- Solution: introduce an intermediate layer that provides a single abstraction for various network technologies
  - O(1) work to add app/media
  - Indirection is an often used technique in computer science



Network Architecture

- Architecture is not the implementation itself
- Architecture is how to “organize” implementations
  - what interfaces are supported
  - where functionality is implemented
- Architecture is the modular design of the network

Software Modularity

Break system into modules:

- Well-defined interfaces gives flexibility
  - can change implementation of modules
  - can extend functionality of system by adding new modules
- Interfaces hide information
  - allows for flexibility
  - but can hurt performance

Network Modularity

Like software modularity, but with a twist:

- Implementation distributed across routers and hosts
- Must decide both:
  - how to break system into modules
  - where modules are implemented

Outline

- How to break network functionality into modules
  - Layering
- Where to implement functionality
  - E.g. End-to-End Argument

Layering

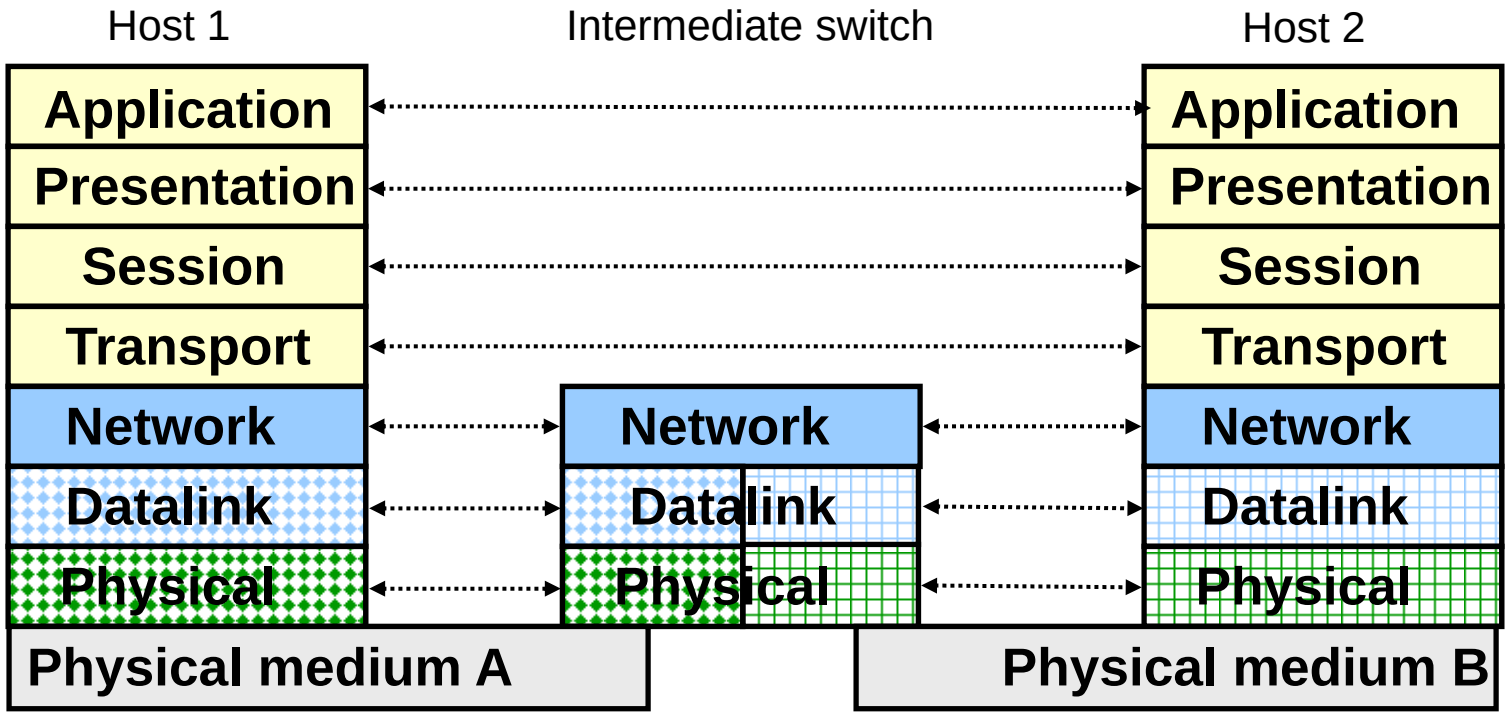
- Layering is a particular form of modularization
- The system is broken into a vertical hierarchy of logically distinct entities (layers)
- The service provided by one layer is based solely on the service provided by layer below
- Rigid structure: easy reuse, performance suffers

ISO OSI Reference Model

- ISO – International Standard Organization
- OSI – Open System Interconnection
- Goal: a general open standard
  - allow vendors to enter the market by using their own implementation and protocols

ISO OSI Reference Model

- Seven layers
  - Lower two layers are peer-to-peer
  - Network layer involves multiple switches
  - Next four layers are end-to-end



Layering Solves Problem

- Application layer doesn't know about anything below the presentation layer, etc.
- Information about network is hidden from higher layers
- This ensures that we only need to implement an application once!

Key Concepts

- Service – says what a layer does
  - Ethernet: unreliable subnet unicast/multicast/broadcast datagram service
  - IP: unreliable end-to-end unicast datagram service
  - TCP: reliable end-to-end bi-directional byte stream service
  - Guaranteed bandwidth/latency unicast service
- Service Interface – says how to access the service
  - E.g. UNIX socket interface
- Protocol – says how is the service implemented
  - a set of rules and formats that govern the communication between two peers

Physical Layer (1)

- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send a bit
- **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**:
  - framing (attach frame separators)
  - send data frames between peers
  - others:
    - arbitrate the access to common physical media
    - per-hop reliable transmission
    - per-hop flow control
- **Interface**: send a data unit (packet) to a machine connected to the same physical media
- **Protocol**: layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)…

Network Layer (3)

- **Service**:
  - deliver a packet to specified network destination
  - perform segmentation/reassemble
  - others:
    - packet scheduling
    - buffer management
- **Interface**: send a packet to a specified destination
- **Protocol**: define global unique addresses; construct routing tables

Transport Layer (4)

- **Service**:
  - Multiplexing/demultiplexing
  - optional: error-free and flow-controlled delivery
- **Interface**: send message to specific destination
- **Protocol**: implements reliability and flow control
- Examples: TCP and UDP

Session Layer (5)

- **Service**:
  - full-duplex
  - access management (e.g., token control)
  - synchronization (e.g., provide check points for long transfers)
- **Interface**: depends on service
- **Protocol**: token management; insert checkpoints, implement roll-back functions

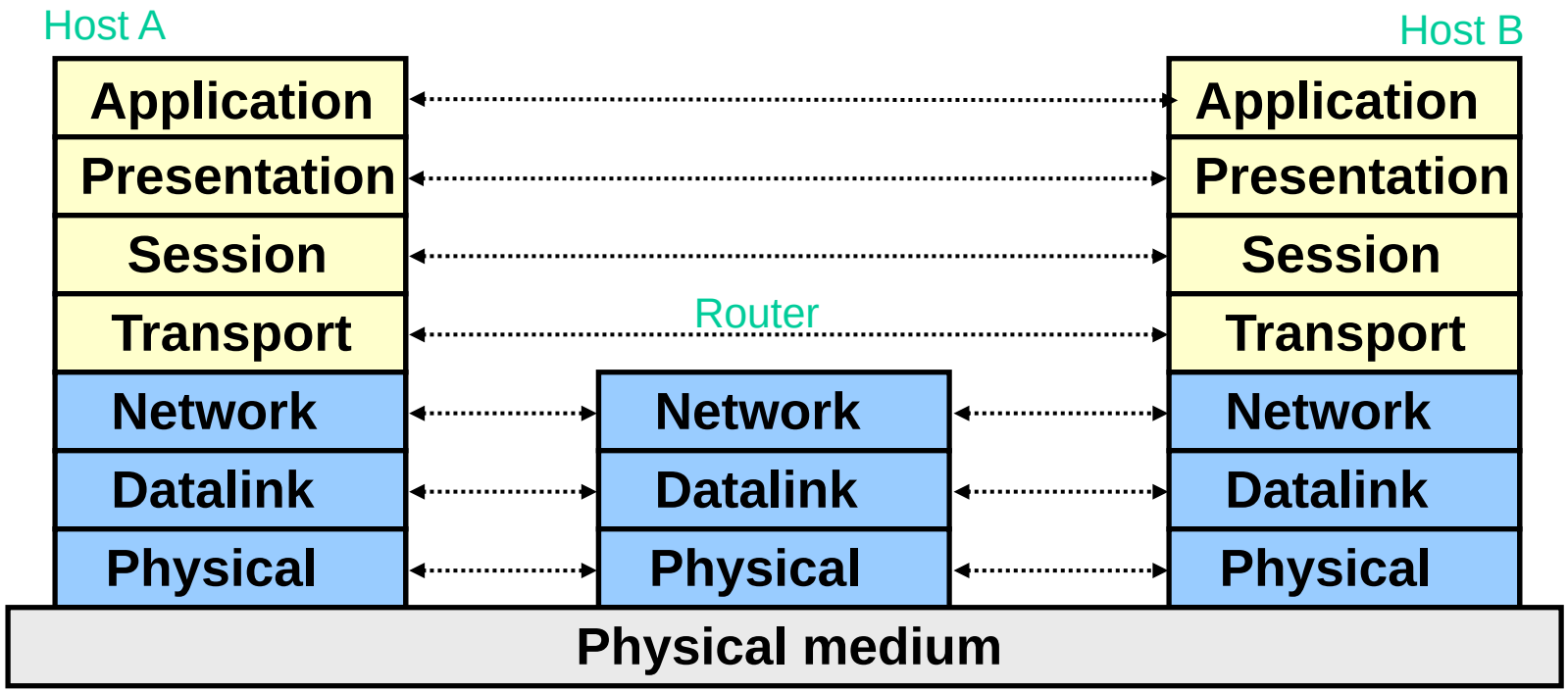
Presentation Layer (6)

- **Service**: convert data between various representations
- **Interface**: depends on service
- **Protocol**: define data formats, and rules to convert from one format to another

Application Layer (7)

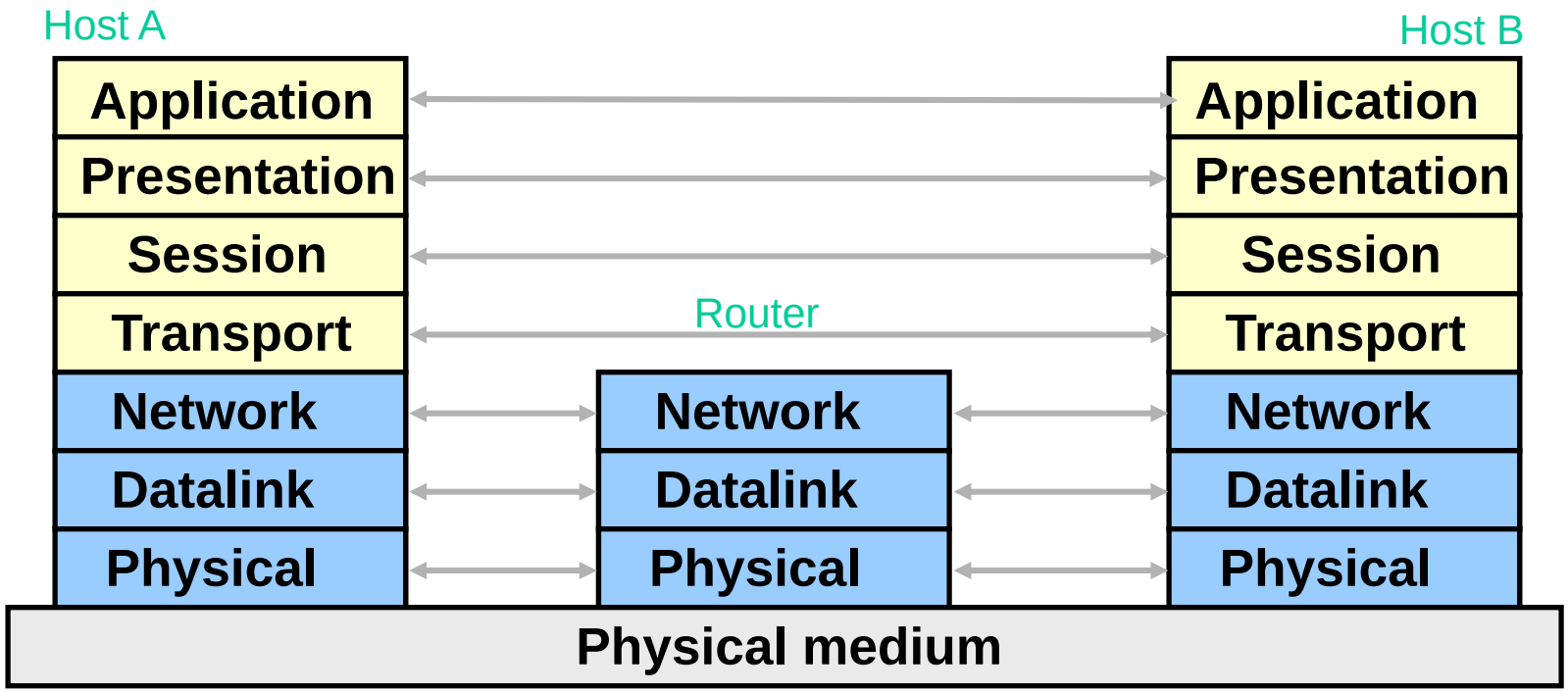
- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application
- Examples: FTP, Telnet, WWW browser

Who Does What?



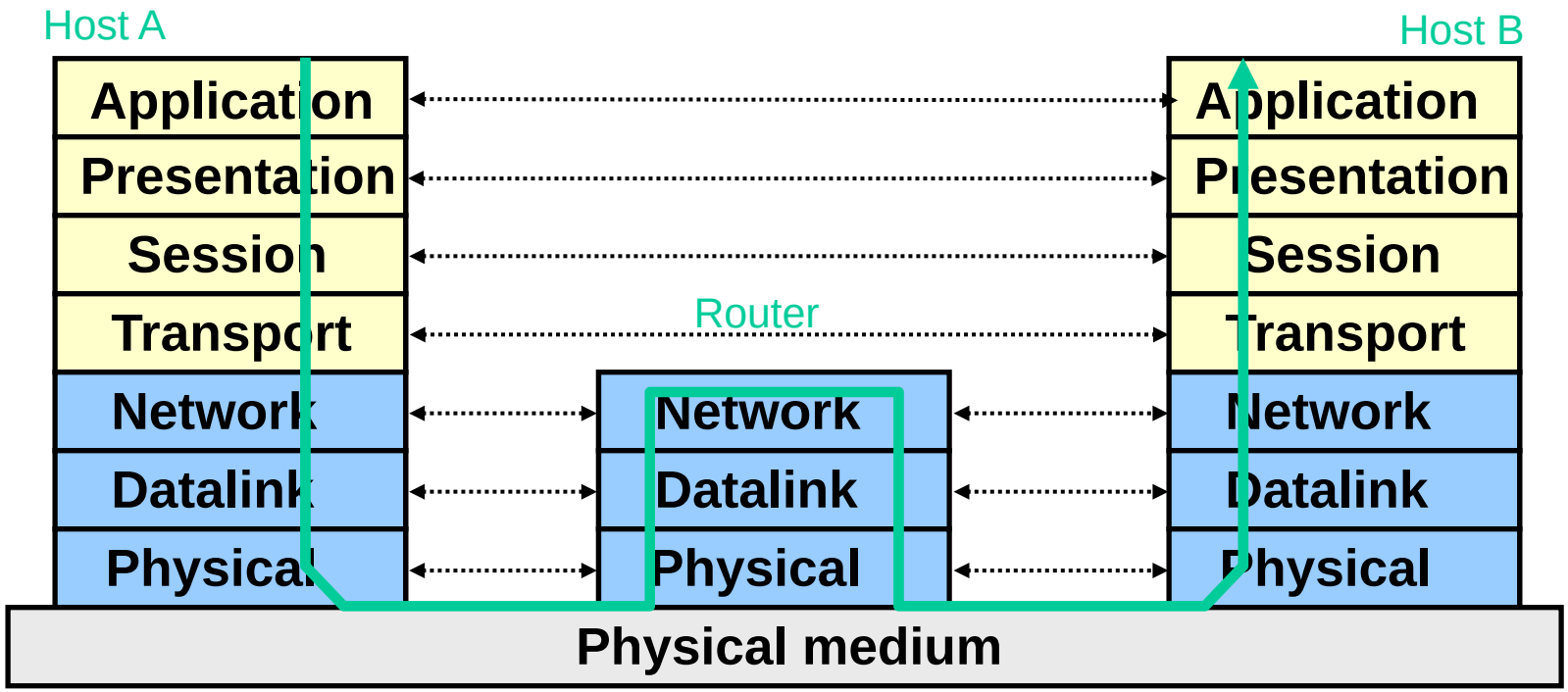
Logical Communication

- Layers interacts with corresponding layer on peer



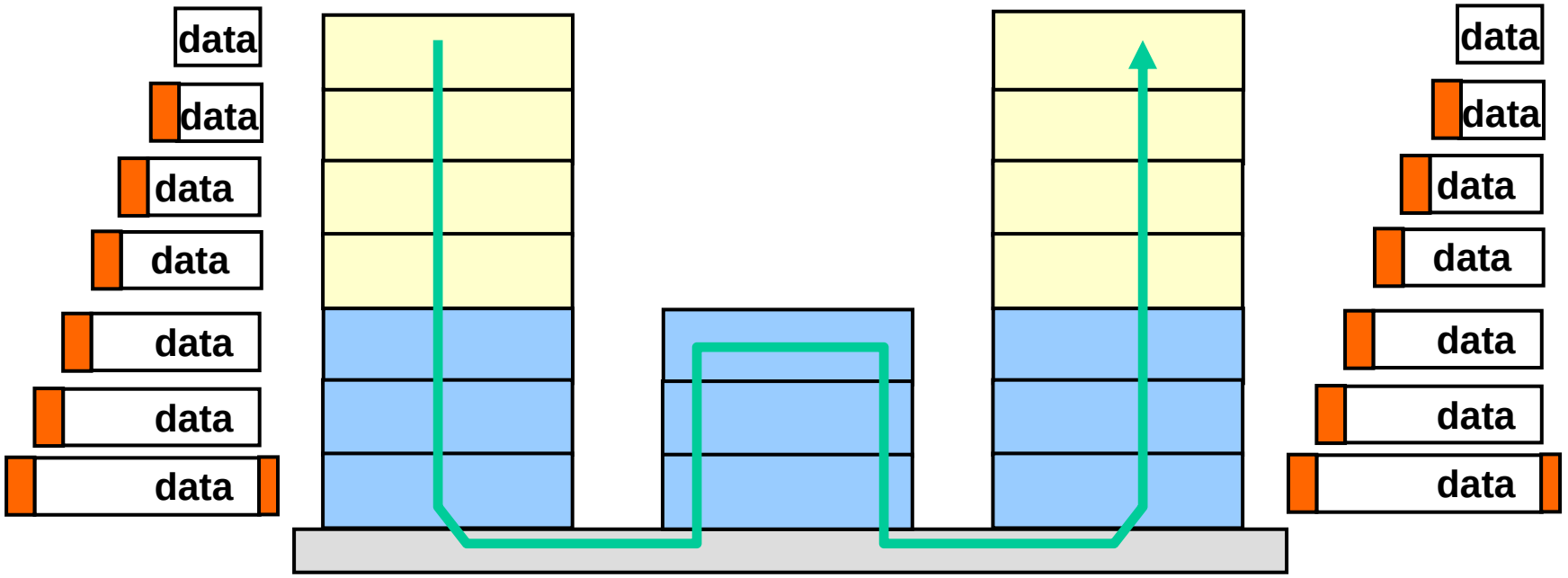
Physical Communication

- Communication goes down to physical network, then to peer, then up to relevant layer



Encapsulation

- A layer can use only the service provided by the layer immediate below it
- Each layer may change and add a header to data packet



Example: Postal System

Standard process (historical):

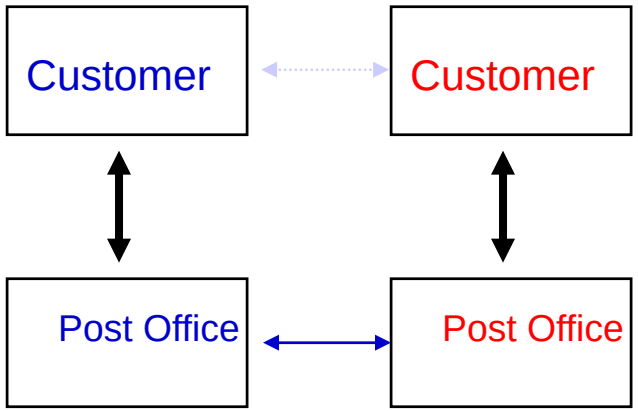
- Write letter
- Drop an addressed letter off in your local mailbox
- Postal service delivers to address
- Addressee reads letter (and perhaps responds)

Postal Service as Layered System

Layers:
- Letter writing/reading
- Delivery

Information Hiding:
- Network need not know letter contents
- Customer need not know how the postal network works

Encapsulation:
- Envelope



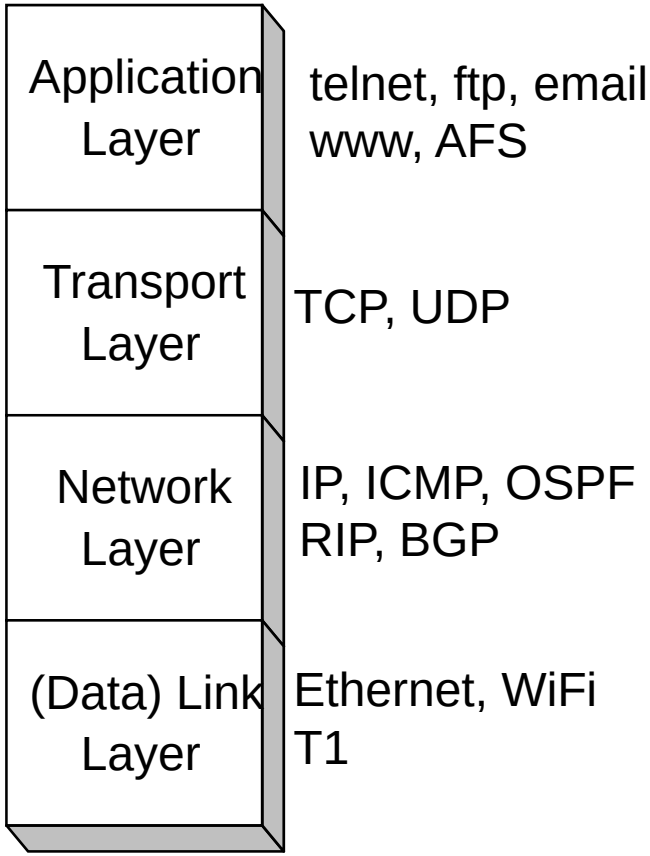
Functions of the Layers

- Service: Handles details of application programs.
- Functions:

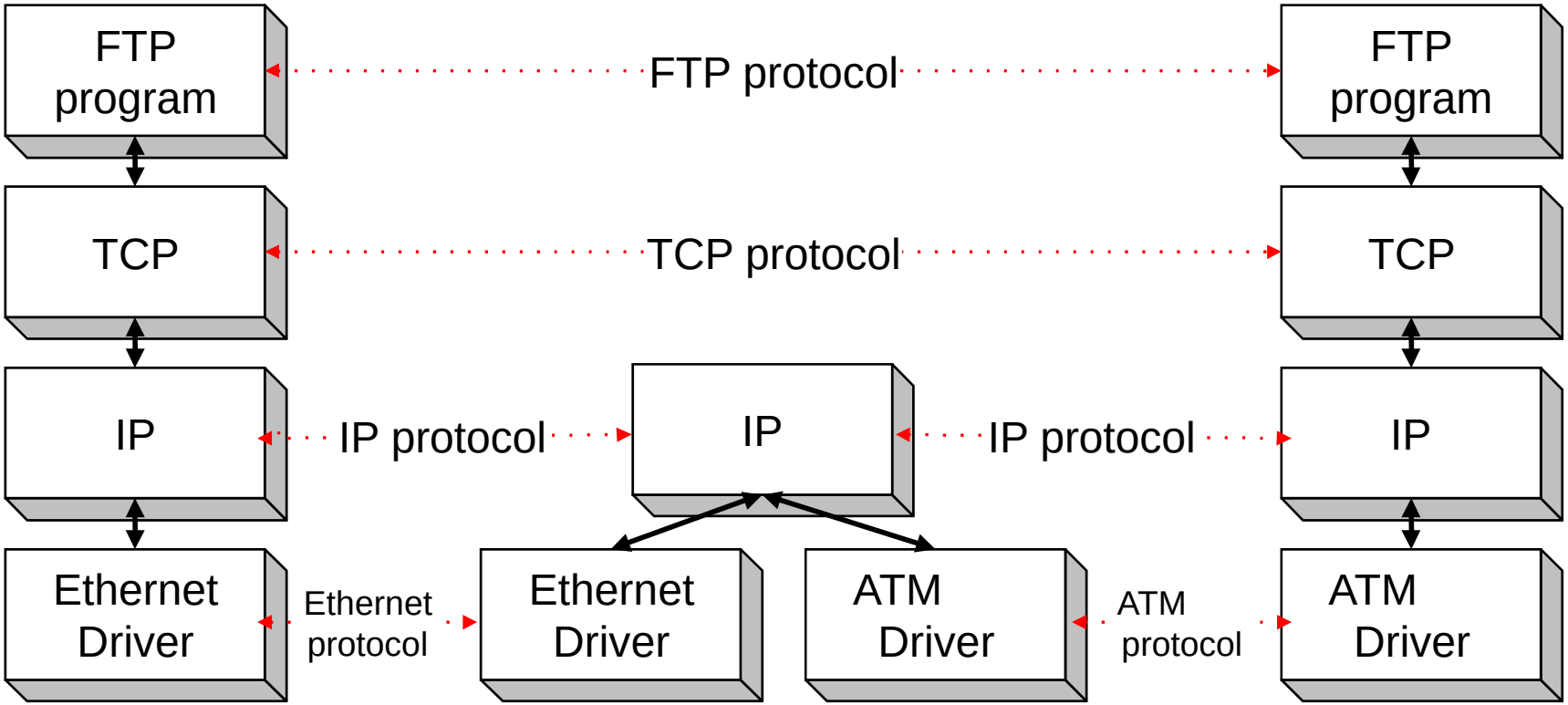
- Service: Controls delivery of data between hosts.
- Functions: Connection establishment/termination, error control, flow control, congestion control, etc.

- Service: Moves packets inside the network.
- Functions: Routing, addressing, switching, etc.

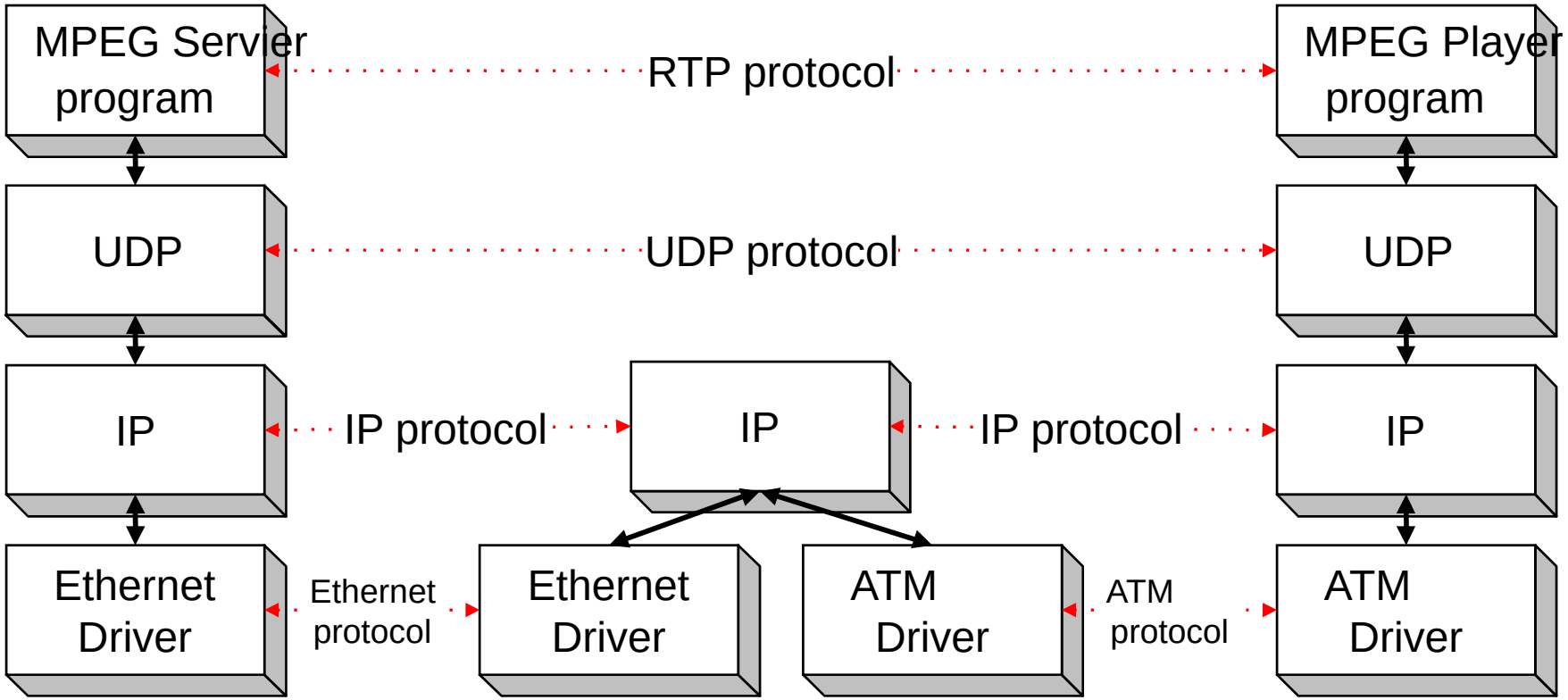
- Service: Reliable transfer of frames over a link.
- Functions: Synchronization, error control, flow control, etc.



Internet Protocol Architecture

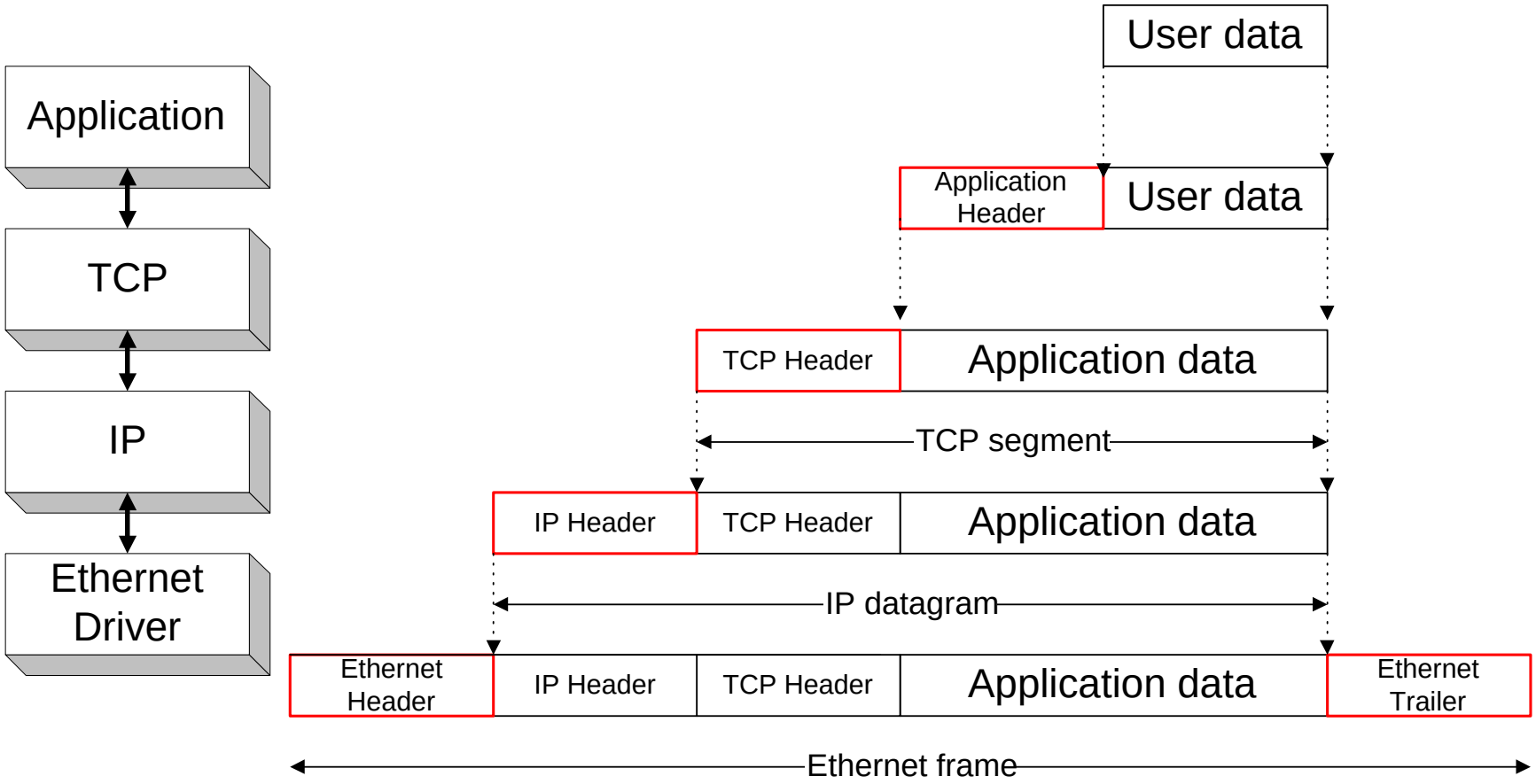


Internet Protocol Architecture

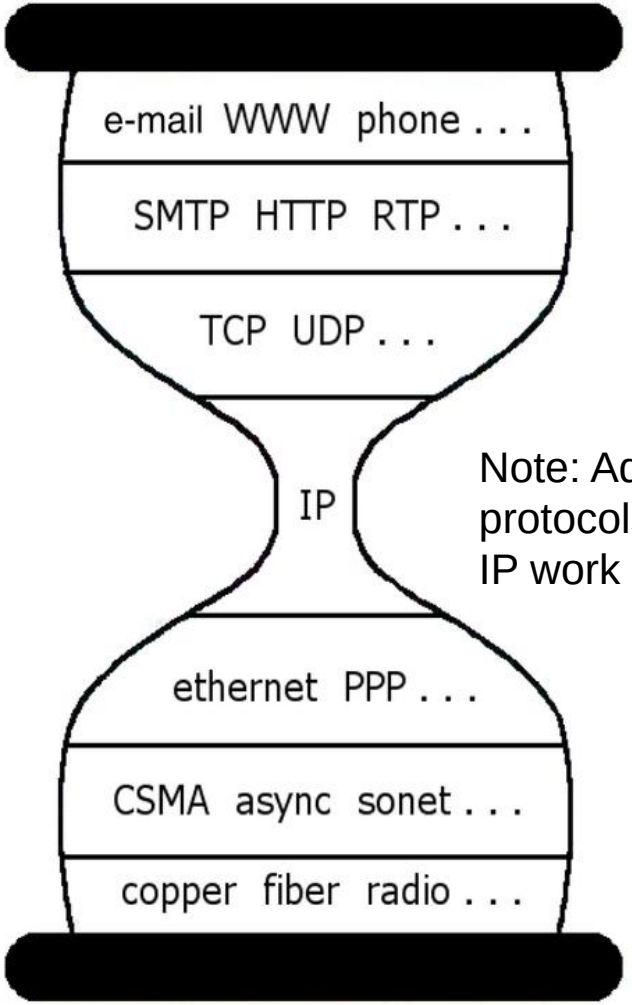


Encapsulation

- As data is moving down the protocol stack, each protocol is adding layer-specific control information.



Hourglass



Note: Additional protocols like routing protocols (RIP, OSPF) needed to make IP work

Implications of Hourglass

A single Internet layer module:

- Allows all networks to interoperate
  - all networks technologies that support IP can exchange packets
- Allows all applications to function on all networks
  - all applications that can run on IP can use any network
- Simultaneous developments above and below IP

Reality

- Layering is a convenient way to think about networks
- But layering is often violated
  - Firewalls
  - Transparent caches
  - NAT boxes

Placing Functionality

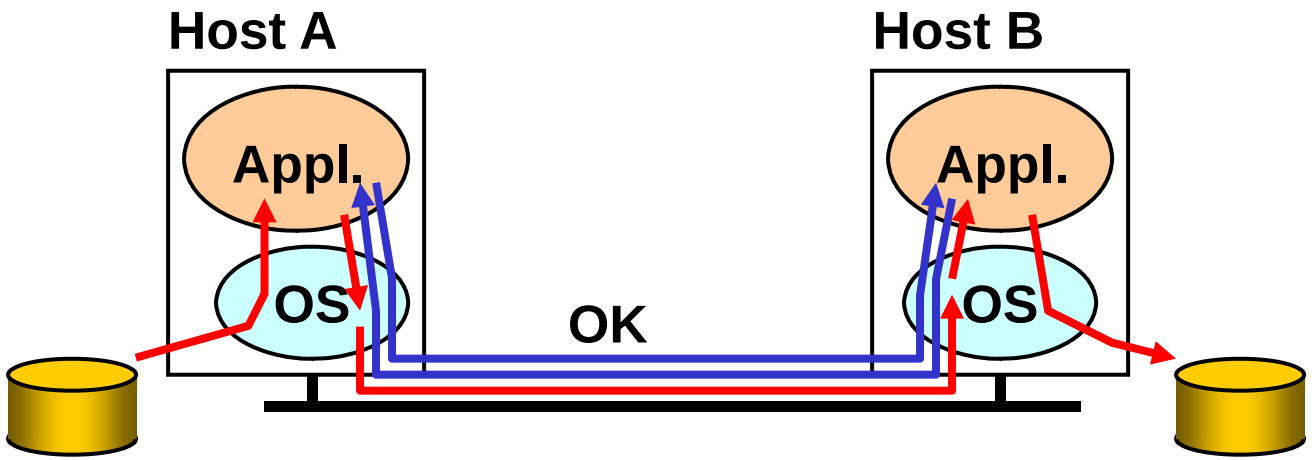
- Influential (and controversial) paper about placing functionality is “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark

- The “Sacred Text” of the Internet
  - endless disputes about what it means
  - everyone cites it as supporting their position

Basic Observation

- Some applications have end-to-end performance requirements
  - reliability, security, etc.
- Implementing these in the network is very hard:
  - every step along the way must be fail-proof
- The hosts:
  - can satisfy the requirement without the network
  - can’t depend on the network

Example: Reliable File Transfer



- Solution 1: make network reliable
- Solution 2: implement end-to-end check and retry if failed

Example (cont'd)

- Solution 1 not complete
  - What happens if any network element misbehaves?
  - The 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?

Conclusion

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
  - very lossy link

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)

- Unless you can relieve the burden from hosts, then 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

Reality

- Layering and E2E Principle regularly violated:
  - Firewalls
  - Transparent caches
  - Other middleboxes
- Battle between architectural purity and commercial pressures

Summary

- Layering is a good way to organize network functions
- Unified Internet layer decouples apps from networks
- E2E argument argues to keep IP simple
- Be judicious when thinking about adding to the network layer