CEG7450: Lecture 2

Reading

- **[Ste97]** W. Stevens, “TCP congestion control”, RFC 2001, Jan 1997.

Overview

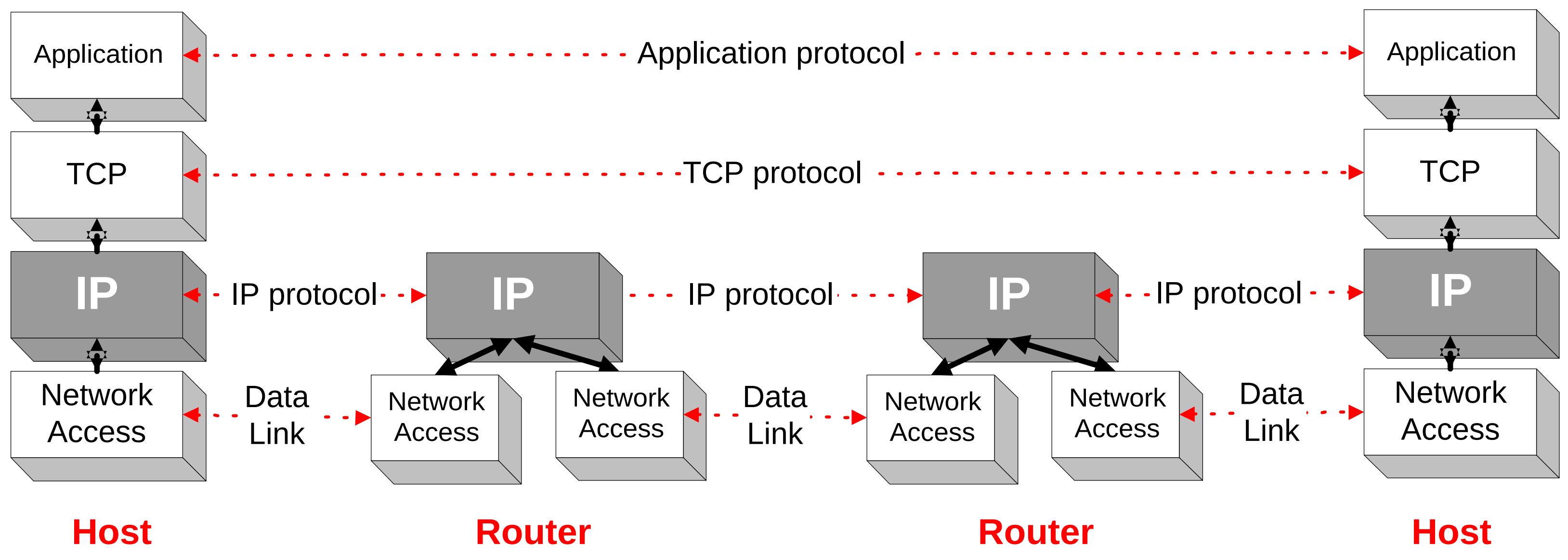
- ➢ Layering
- End-to-End Arguments
- A Case Study: the Internet

What is Layering?

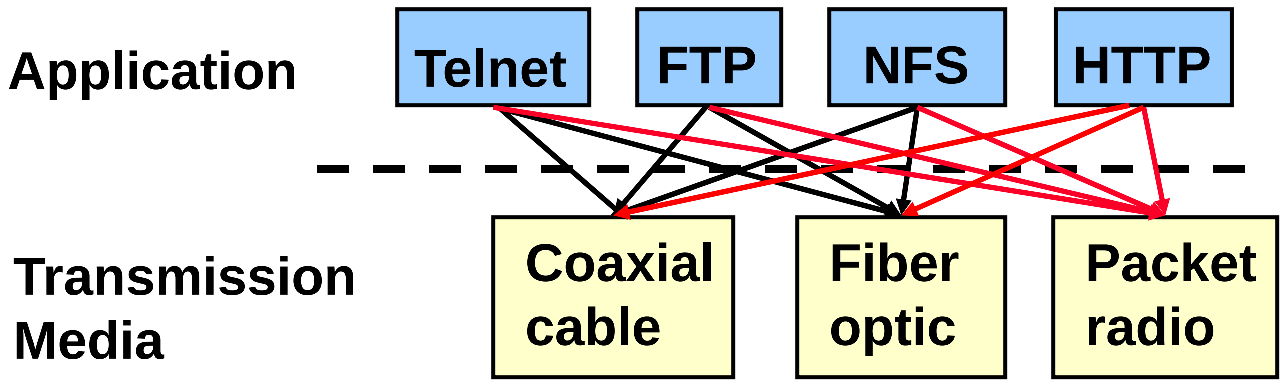
- A technique to organize a network system into a succession of logically distinct entities, such that the service provided by one entity is solely based on the service provided by the previous (lower level) entity

Network Protocols

- Specify any function that requires cooperation between two or more network entities
  - specify the format of the information that is sent/received among routers and end-systems
  - specify timings and the actions that a node has to take when it receives special messages or special events occur



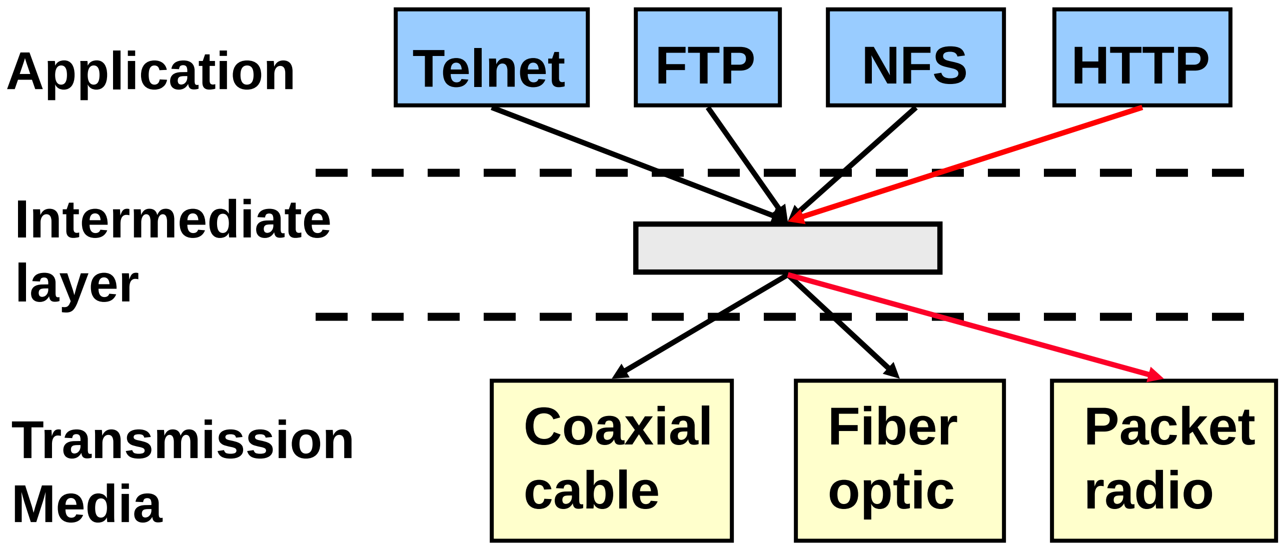
Why Layering?



- No layering: each new application has to be re-implemented for every network technology!

Why Layering?

- Solution: introduce an intermediate layer that provides a unique abstraction for various network technologies



Layering

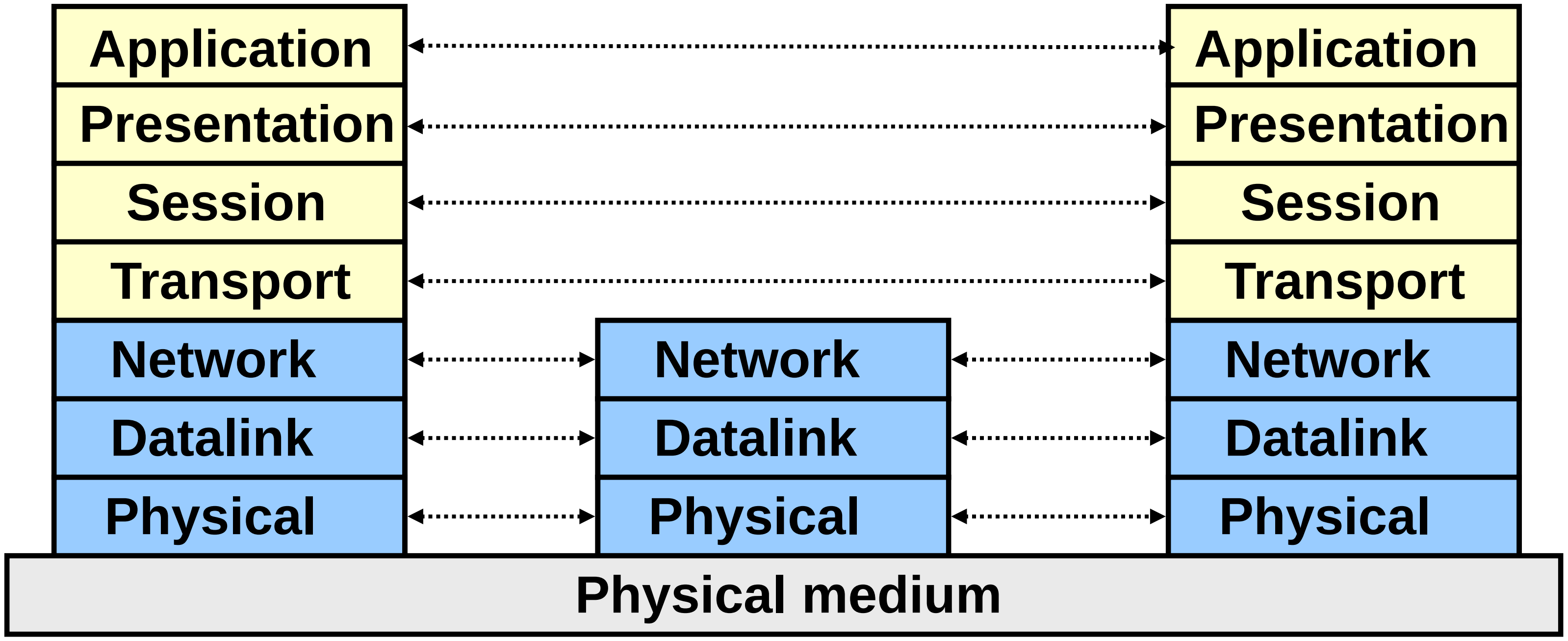
- Advantages
  - Modularity – protocols easier to manage and maintain
  - Abstract functionality –lower layers can be changed without affecting the upper layers
  - Reuse – upper layers can reuse the functionality provided by lower layers
- Disadvantages
  - Information hiding – inefficient implementations

ISO OSI Reference Model

- ISO – International Standard Organization
- OSI – Open System Interconnection
- Started in 1978; first standard 1979
  - ARPANET started in 1969; TCP/IP protocols ready by 1974
- Goal: a general open standard
  - allow vendors to enter the market by using their own implementation

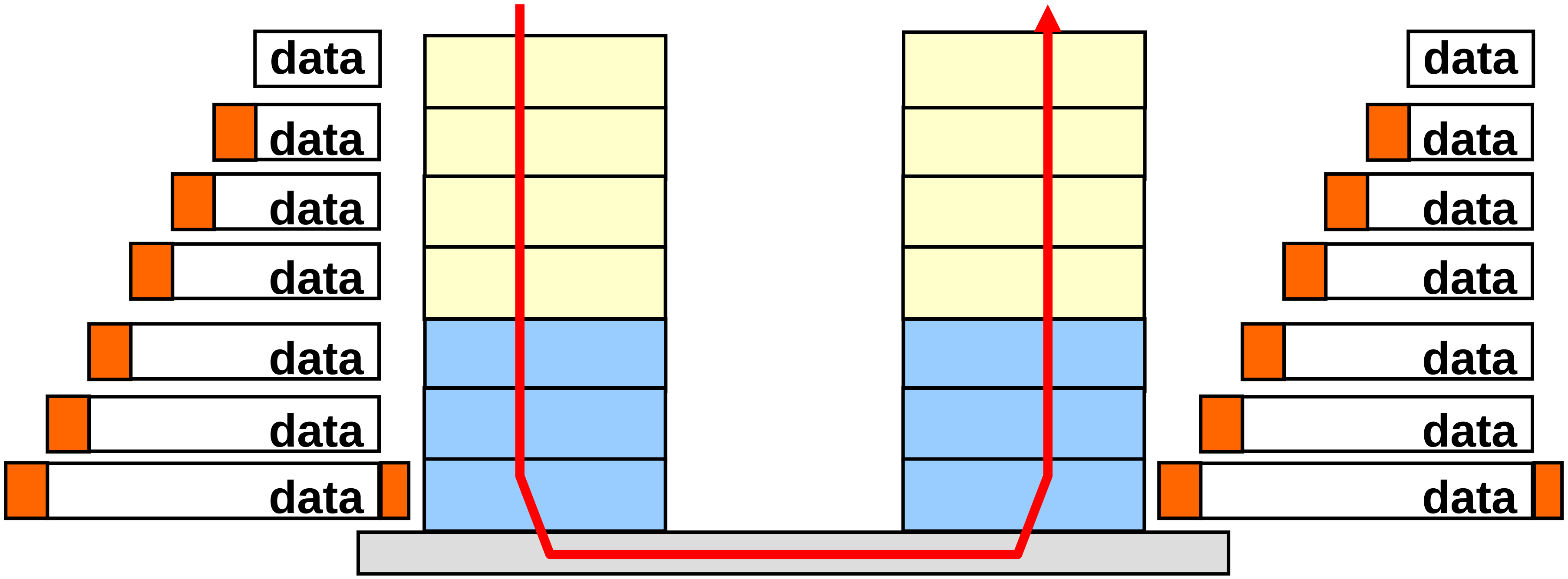
ISO OSI Reference Model

- Seven layers
  - Lower three layers are peer-to-peer
  - Next four layers are end-to-end



Data Transmission

- 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



OSI Model Concepts

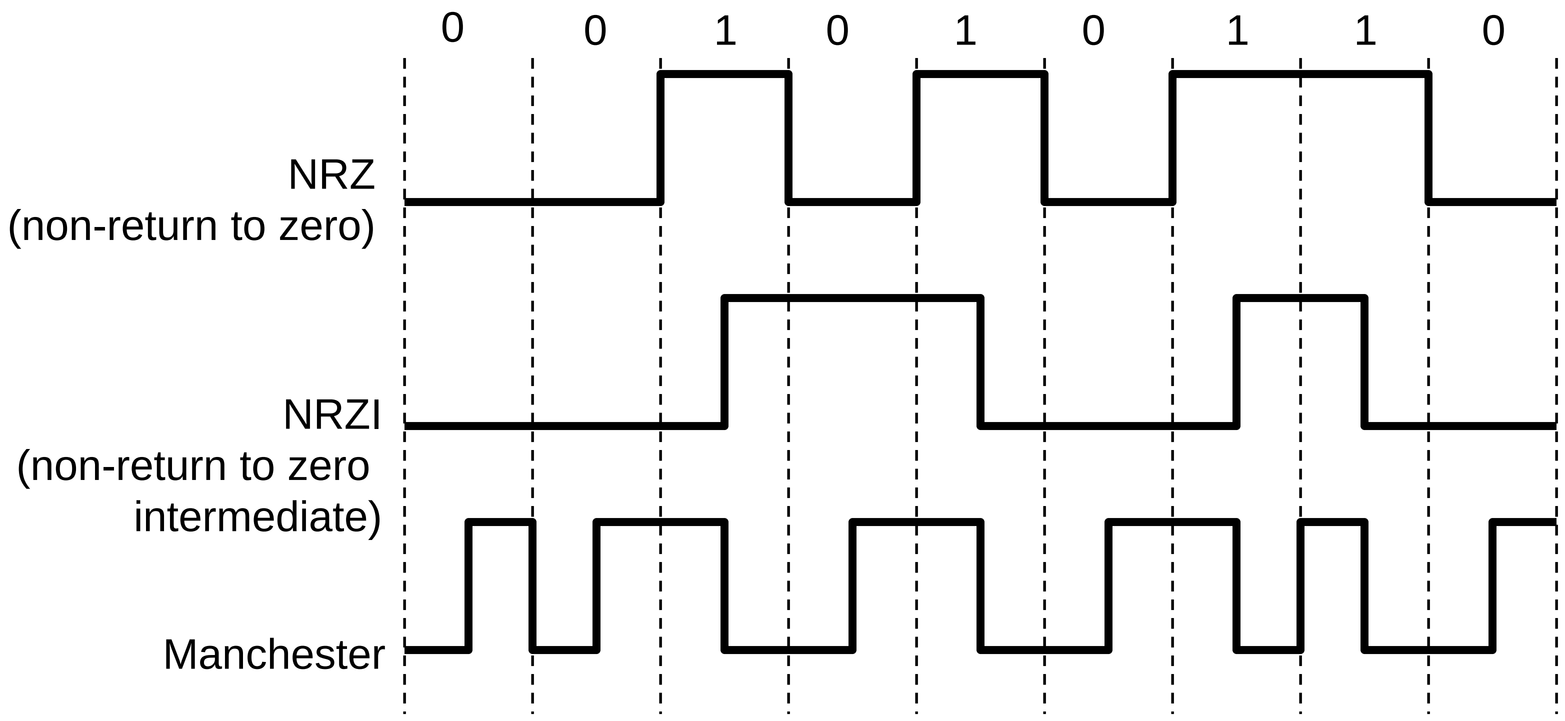
- Service – says what a layer does
- Interface – says how to access the service
- Protocol – says how is the service implemented
  - a set of rules and formats that govern the communication between two peers

Physical Layer (1)

- Service: move the 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

Encoding Schemes



- 4B/5B encode each 4 bits in a 5-bit code such that there is at most one leading 0 and at most two trailing zero and use NRZI encoding

Datalink Layer (2)

- Service:
  - framing, i.e., attach frames separator
  - send data frames between peers
  - others:
    - arbitrate the access to common physical media
    - ensure reliable transmission
    - provide flow control
- Interface: send a data unit (packet) to a machine connected to same physical media
- Protocol: layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)…

Marking the Beginning/Ending of a Frame

- Use a special bit sequence
- Problem: what happens if this sequence appears in the data payload?
- Use bit stuffing technique; e.g., assume both encoding, decoding bit sequences are 01111110
  - sender: inserts a 0 after five consecutive 1s
  - receiver: when it sees five 1s makes decision on next two bits
    - if next bit 0 (this is a stuffed bit), remove it
    - if next bit 1, look at the next bit
      - if 0 this is end-of-frame (receiver has seen 01111110)
      - if 1 this is an error, discard the frame (receiver has seen 01111111)

Network Layer (3)

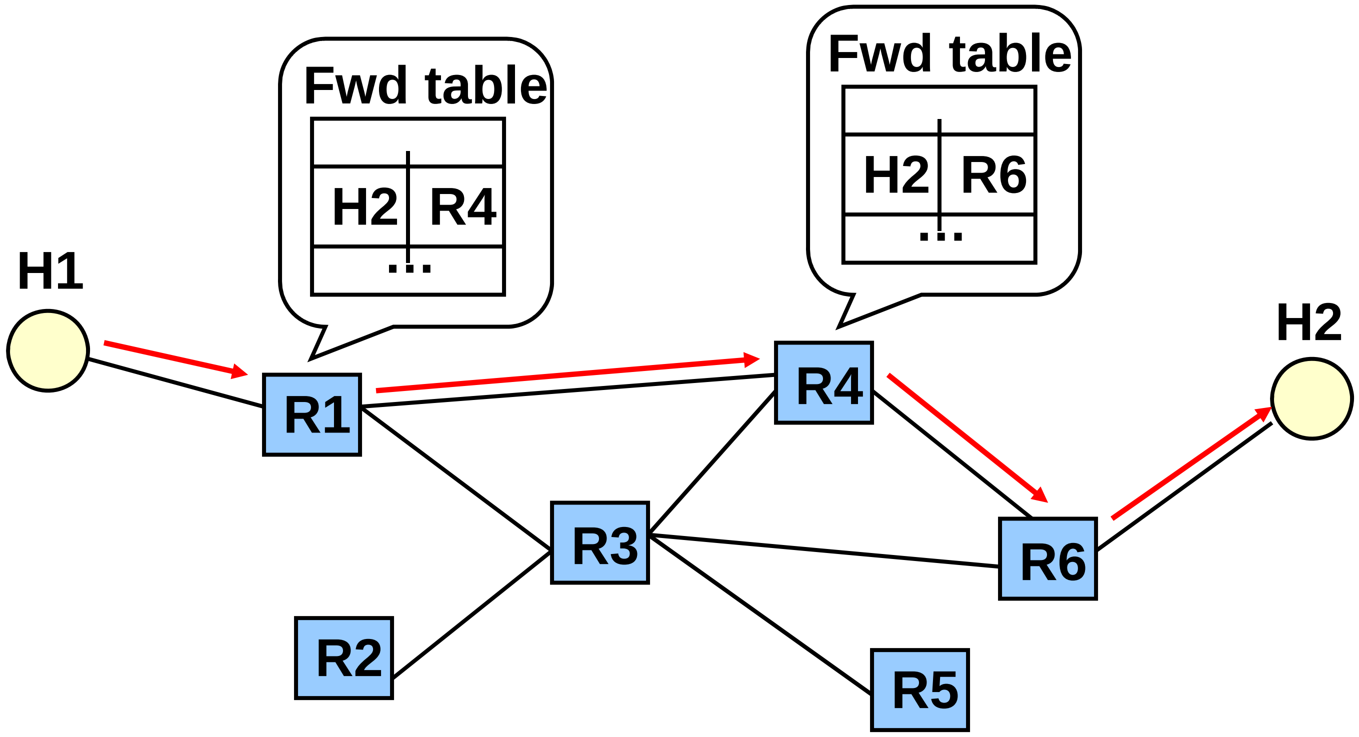
- Service:
  - deliver a packet to specified destination
  - perform segmentation/reassembly
  - others:
    - packet scheduling
    - buffer management
- Interface: send a packet to a specified destination
- Protocol: define global unique addresses; construct routing tables

Data and Control Planes

- Data plane: concerned with
  - packet forwarding
  - buffer management
  - packet scheduling
- Control Plane: concerned with installing and maintaining state for data plane

Example: Routing

- Data plane: use Forwarding Table to forward packets
- Control plane: construct and maintain Forwarding Tables (e.g., Distance Vector, Link State protocols)



Transport Layer (4)

- Service:
  - provide an error-free and flow-controlled end-to-end connection
  - multiplex multiple transport connections to one network connection
  - split one transport connection in multiple network connections
- Interface: send a packet to specify destination
- Protocol: implement 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
- Protocols: 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

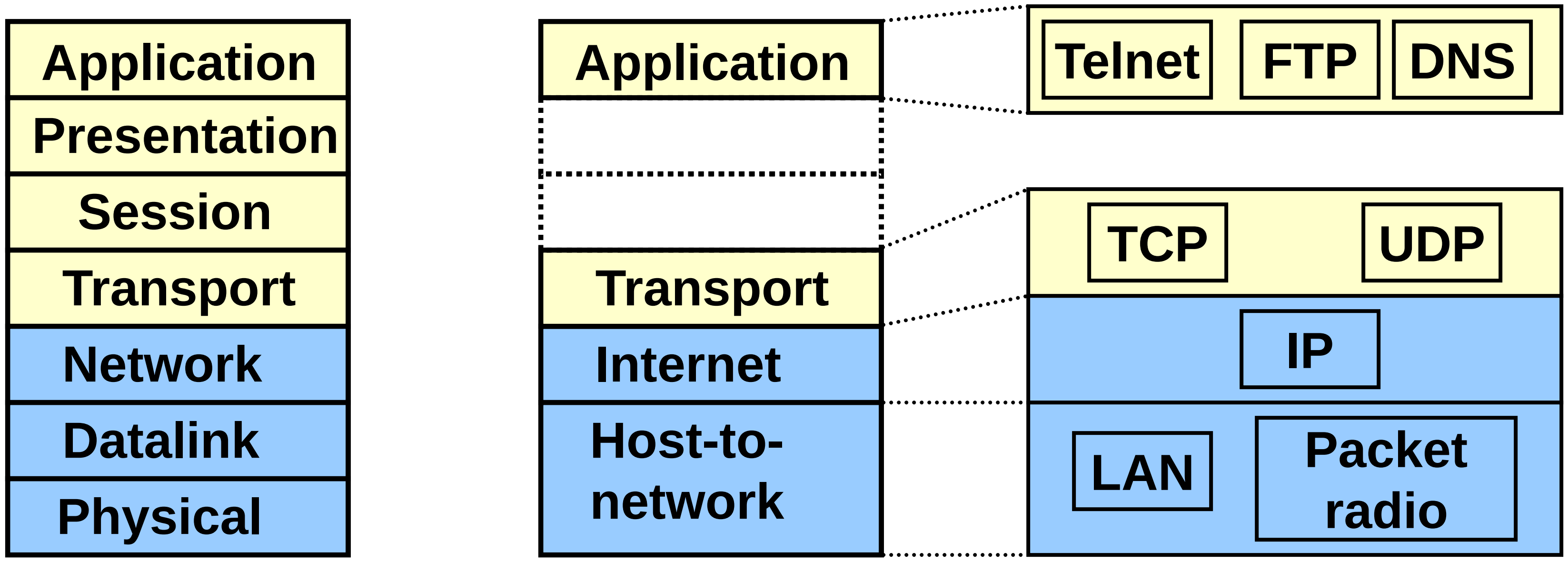
Application Layer (7)

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

- Examples: FTP, Telnet, WWW browser

OSI vs. TCP/IP

- OSI: conceptually define: service, interface, protocol
- Internet: provide a successful implementation



Key Design Decision

- How do you divide functionality across the layers?

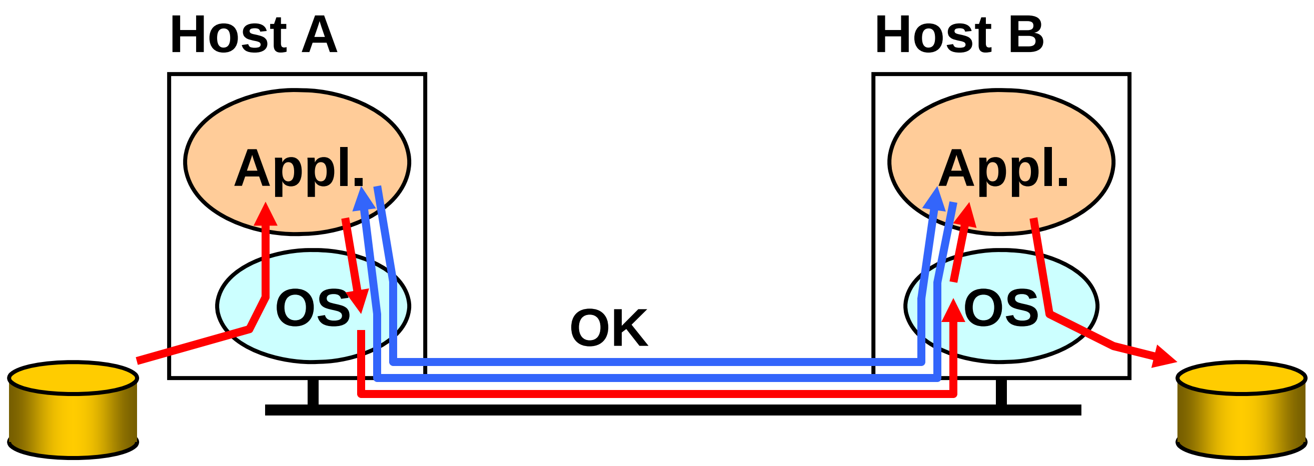
Overview

- Layering
- ➢ End-to-End Arguments
- A Case Study: the Internet

End-to-End Argument

- Think twice before implementing a functionality that you believe that is useful to an application at a lower layer
- If the application can implement a functionality correctly, implement it at a lower layer only as a performance enhancement

Example: Reliable File Transfer



- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry

Discussion

- Solution 1 not complete
  - What happens if the sender or/and receiver misbehave?
- The receiver has to do the check anyway!
- Thus, full functionality can be entirely implemented at application layer; no need for reliability from lower layers
- Is there any need to implement reliability at lower layers?

Discussion

- Yes, but only to improve performance
- Example:
  - assume a high error rate on communication network
  - then, a reliable communication service at data link layer might help

Trade-offs

- Application has more information about the data and the semantic of the service it requires (e.g., can check only at the end of each data unit)
- A lower layer has more information about constraints in data transmission (e.g., packet size, error rate)

- Note: these trade-offs are a direct result of layering!

Rule of Thumb

- Implementing a functionality at a lower level should have minimum performance impact on the application that do not use the functionality

Other Examples

- Secure transmission of data
- Duplicate message suppression
- Transaction management
- RISC vs. CISC

Overview

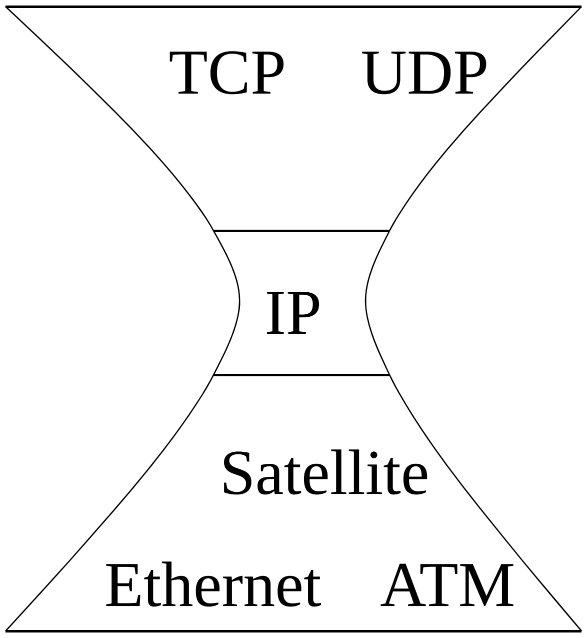
- Layering
- End-to-End Arguments
- ➢ A Case Study: the Internet

Internet & End-to-End Argument

- Network layer provides one simple service: best effort datagram (packet) delivery
- Only one higher level service implemented at transport layer: reliable data delivery (TCP)
  - performance enhancement; used by a large variety of applications (Telnet, FTP, HTTP)
  - does not impact other applications (can use UDP)
- Everything else implemented at application level

Internet Architecture

- Packet-switched datagram network
- IP is the glue
- Hourglass architecture
  - all hosts and routers run IP
- Stateless architecture
  - no per flow state inside network



Key Advantages

- The service can be implemented by a large variety of network technologies
- Does not require routers to maintain any fine grained state about traffic. Thus, network architecture is
  - Robust
  - Scalable
  - Stateless

What About Other Services?

- Multicast?
  - Multicast group management at routers?
  - End host based multicast?
- Quality of Service (QoS)?
  - Per flow state: resource reservation, classification, buffer management, scheduling
  - Stateless quality of service?

Summary: Layering

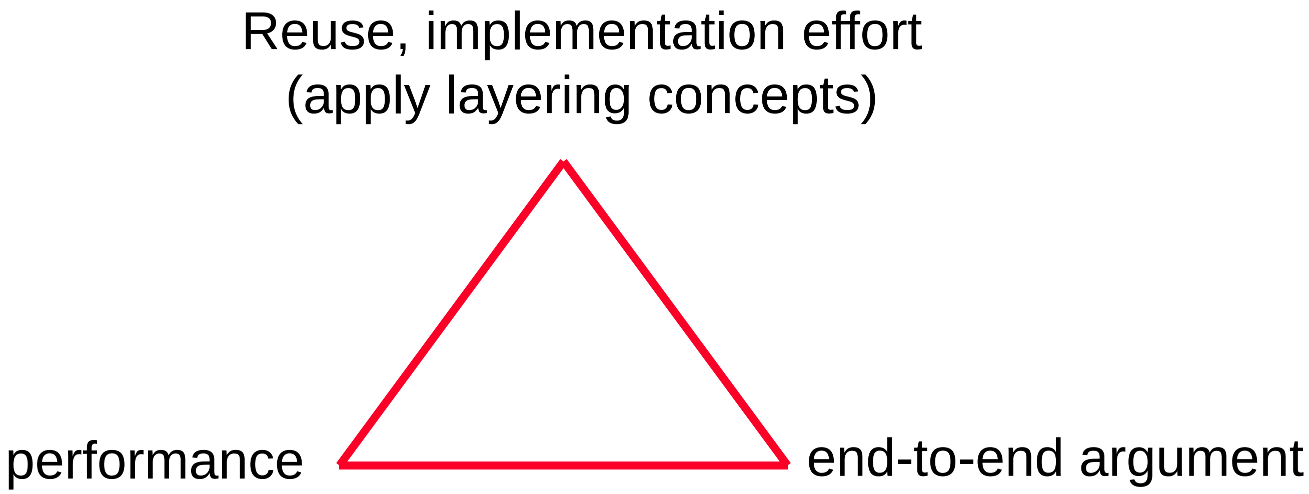
- Key technique to implement communication protocols; provides
  - Modularity
  - Abstraction
  - Reuse
- Key design decision: what functionality to put in each layer?

Summary: End-to-End Arguments

- If the application can do it, don’t do it at a lower layer
  - the application knows the best what it needs anyway
  - add functionality in lower layers iff it is
    - (1) used and improves performance of a large number of applications, and
    - (2) does not hurt other applications
- Success story: Internet

Summary

- Challenge of building a good (network) system: find the right balance between:



- No universal answer: the answer depends on the goals and assumptions!