CS 268: Lecture 2
(Layering & End-to-End Arguments)

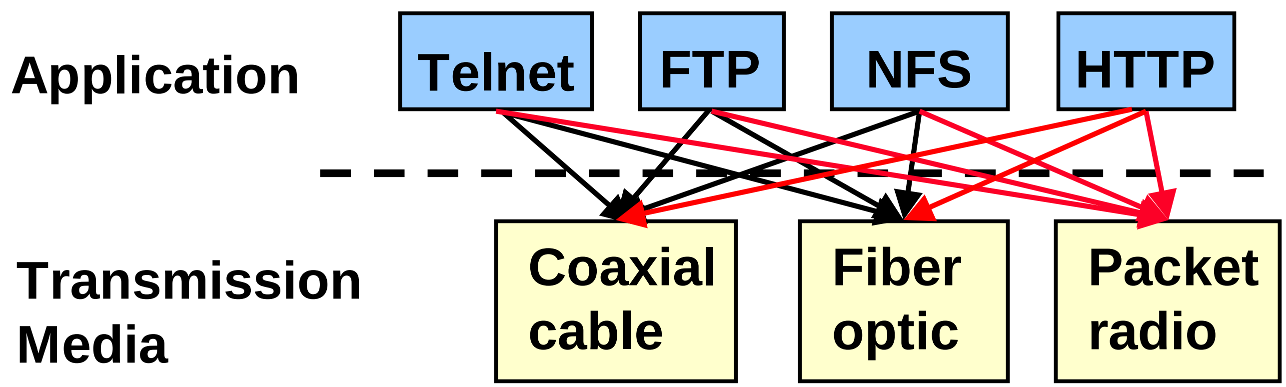
Overview

- ➢ 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

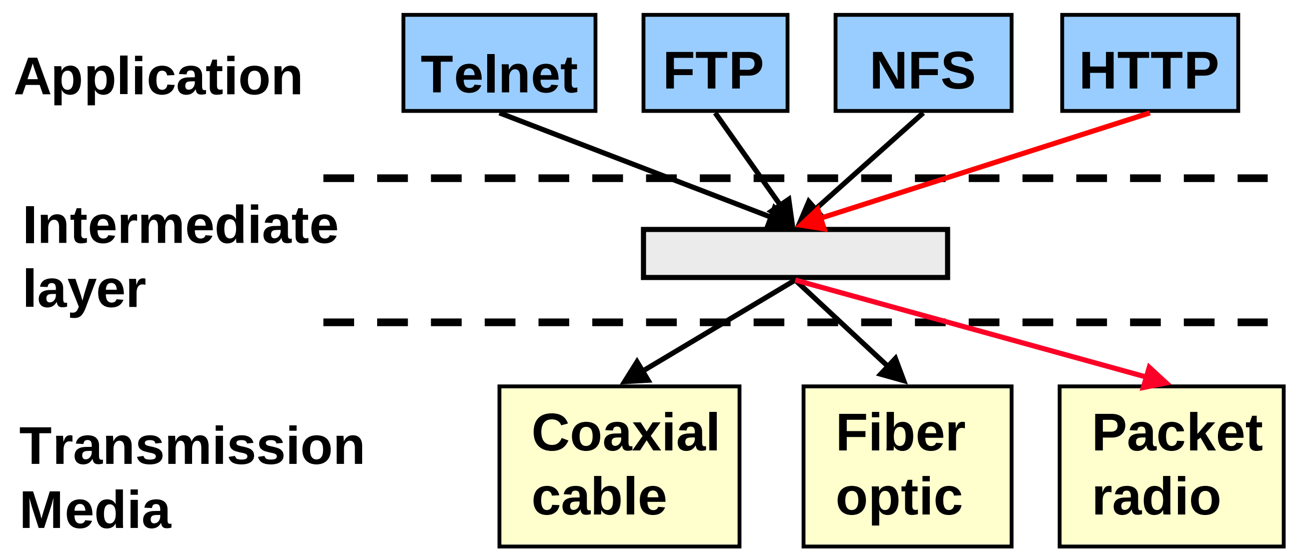
Why Layering?



- 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

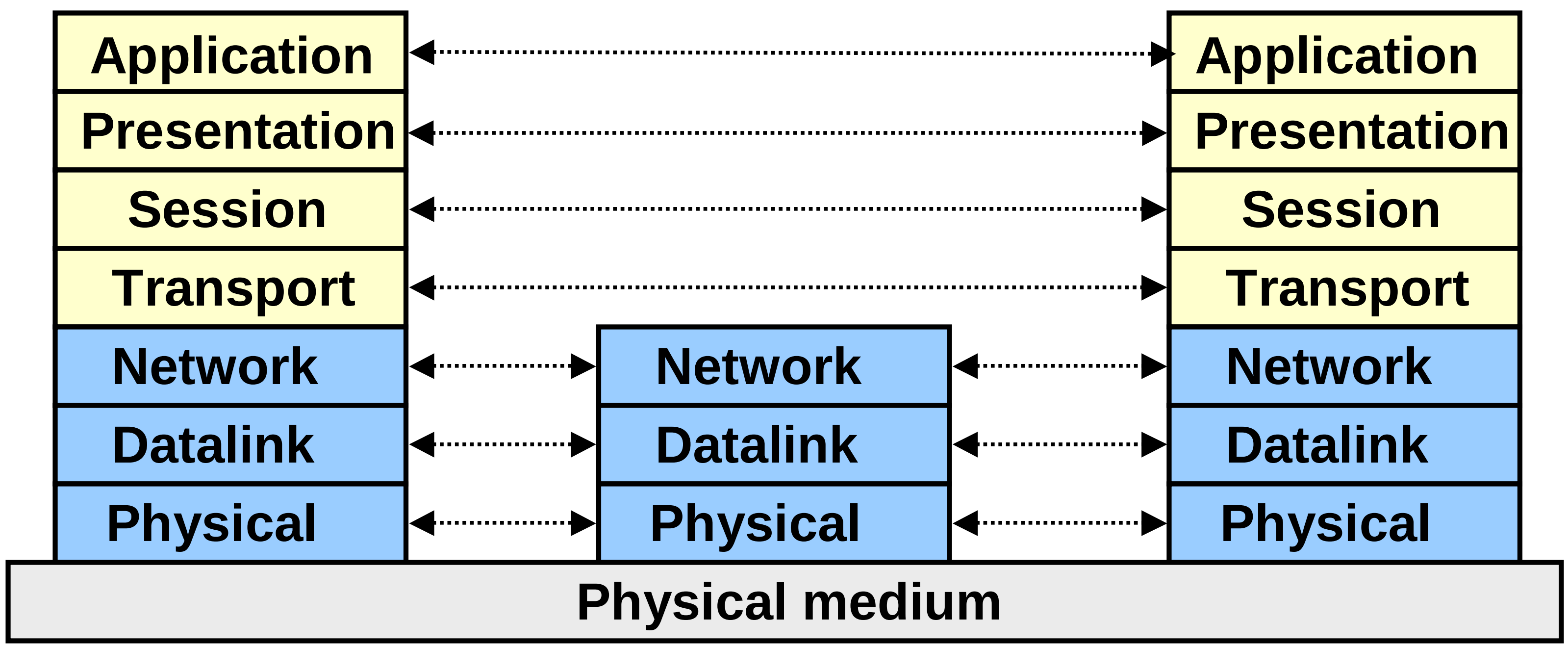
- Advantages
  - Modularity – protocols easier to manage and maintain
  - Abstract functionality –lower layers can be changed without affecting the upper layers
  - Reuse – upper layers can reuse the functionality provided by lower layers
- Disadvantages
  - Information hiding – inefficient implementations

ISO OSI Reference Model

- ISO – International Standard Organization
- OSI – Open System Interconnection
- Started to 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 and protocols

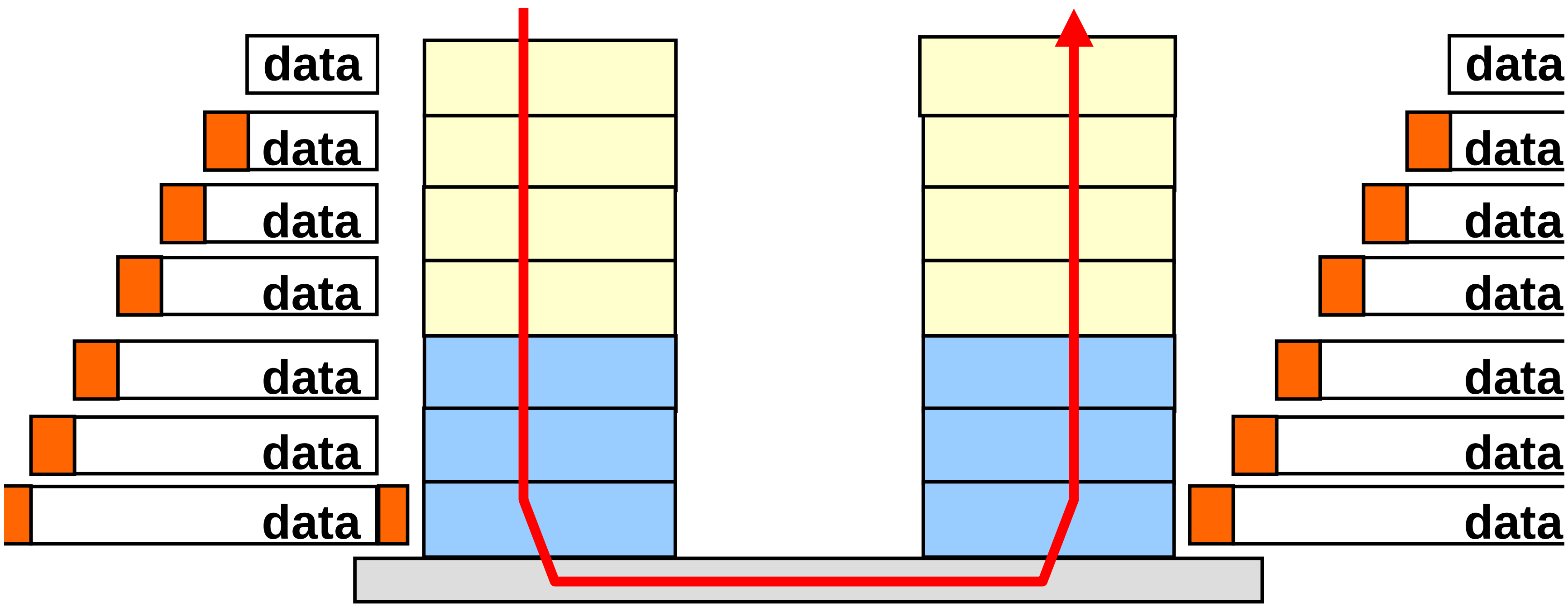
ISO OSI Reference Model

- 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

Datalink Layer (2)

- Service:
  - Framing, i.e., attach frames separator
  - Send data frames between peers attached to the same physical media
  - Others (optional):
    - Arbitrate the access to common physical media
    - Ensure reliable transmission
    - Provide 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)

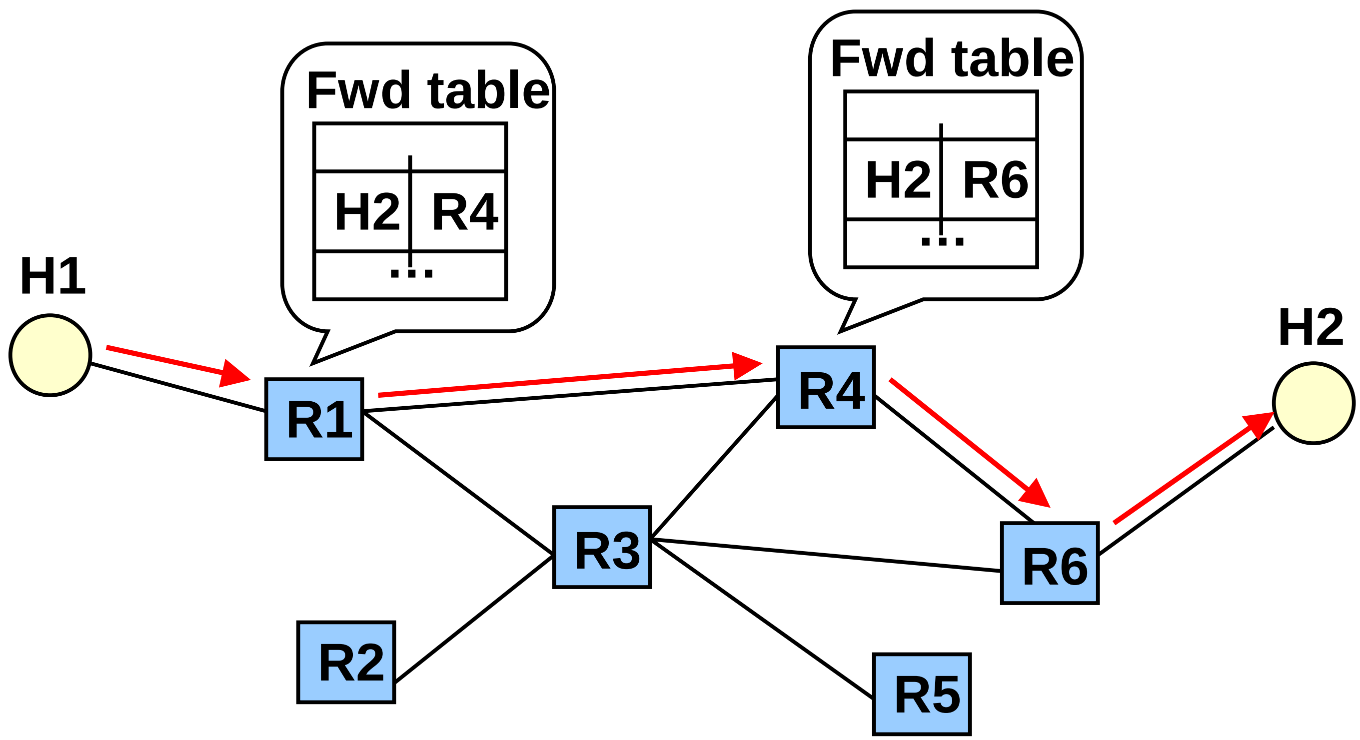
- Service:
  - Deliver a packet to specified destination
  - Perform segmentation/reassemble (fragmentation/defragmentation)
  - Others:
    - Packet scheduling
    - Buffer management
- Interface: send a packet to a specified destination
- Protocol: define global unique addresses; construct routing 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)

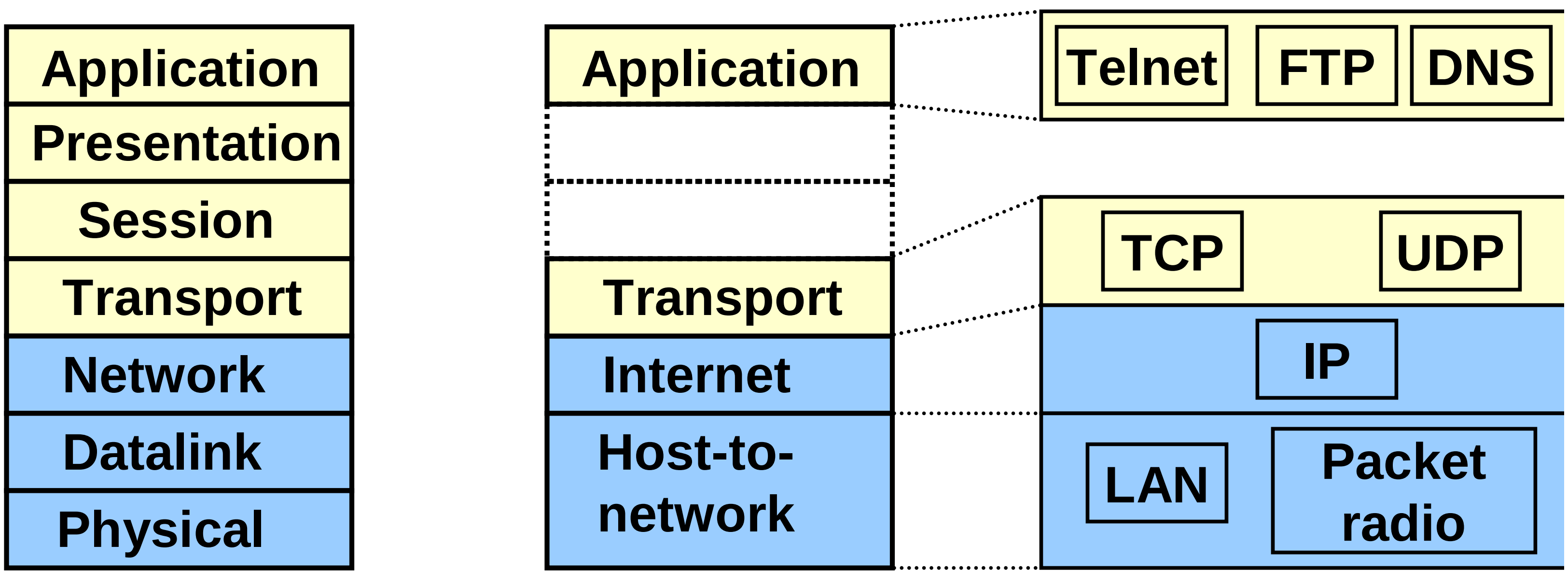
- Service: any service provided to the end user
- Interface: depends on the application
- Protocol: depends on the application

- 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?

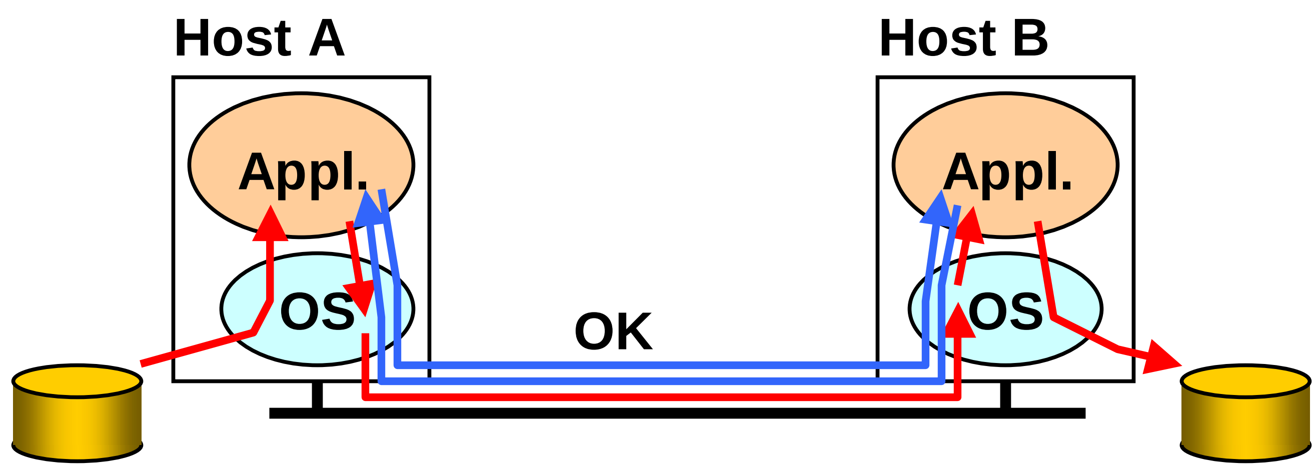
Overview

- 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 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 datalink 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

Internet & End-to-End Argument

- 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

Key Advantages

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

What About Other Services?

- Multicast?
- Quality of Service (QoS)?

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 Argument

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

Overview

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

Goals

0 **Connect existing networks**
  - initially ARPANET and ARPA packet radio network

1. Survivability
   - ensure communication service even in the presence of network and router failures
2. Support multiple types of services
3. Must accommodate a variety of networks
4. Allow distributed management
5. Allow host attachment with a low level of effort
6. Allow resource accountability

Connect Existing Networks

- Existing networks: ARPANET and ARPA packet radio
- Decision: packet switching
  - Existing networks already were using this technology
- Packet switching → store and forward router architecture

- Internet: a packet switched communication network consisting of different networks connected by store-and-forward routers

Survivability

- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - As long as the network is not partitioned, two endpoint should be able to communicate…moreover, any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain state only at end-points (fate-sharing)
  - Eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network architecture

Services

- At 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

Key Advantages

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

What About Other Services?

- Multicast?
- Quality of Service (QoS)?

Summary: Layering

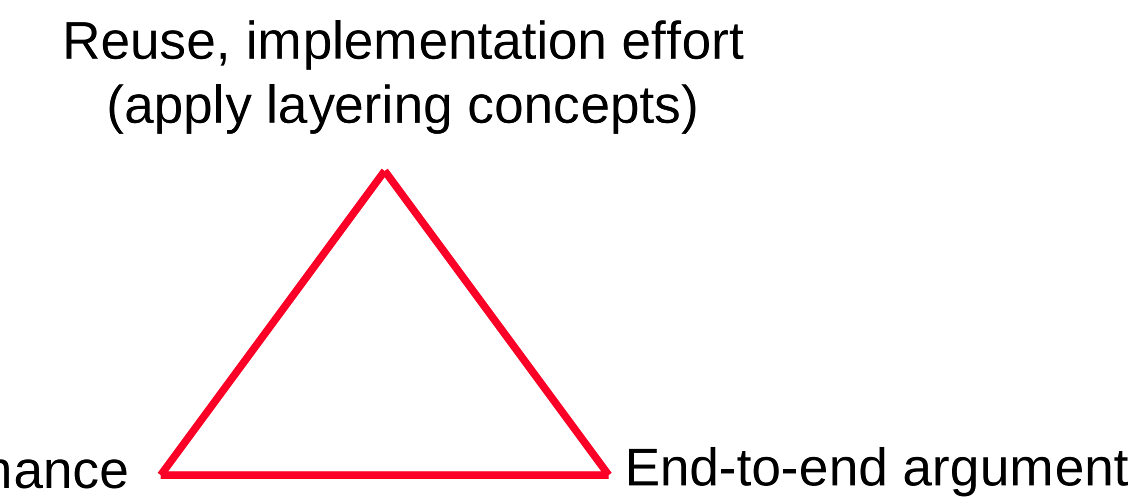
- Key technique to implement communication protocols; provides
  - Modularity
  - Abstraction
  - Reuse
- Key design decision: what functionality to put in each layer?

Summary: End-to-End Arguments

- If the application can do it, don’t do it at a lower layer -- anyway the application knows the best what it needs
  - add functionality in lower layers iff it is (1) used and improves performances 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!