CompSci 356: Computer Network Architectures

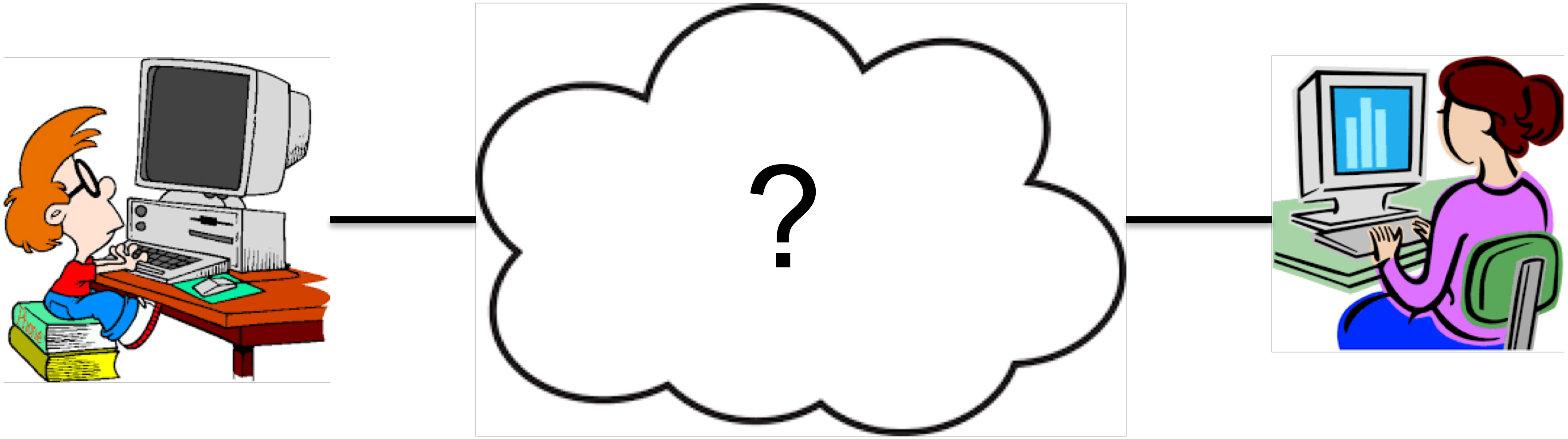
Lecture 2: Network Architectures

Xiaowei Yang

xwy@cs.duke.edu

Overview

- Design requirements of the original Internet
  - Where to places functions
  - Ref:
- Concepts of Network Architectures



1^{st} Mission of this course

- Understand the concepts and design principles that make the Internet work
- Design process
  - Identify requirements, brainstorm design choices/mechanisms, make design decisions
  - What requirements make sense to you?
    - Scalable connectivity
    - Cost-effective resource sharing
    - Support for different types of services
    - Manageability
    - …
  - It remains an open challenge how to incorporate other requirements such as security into the Internet design

Features of computer networks

- Generality
- Carrying many different types of data
- Supporting an unlimited range of applications

What’s the Internet?

- The **Internet** is a large-scale general-purpose computer network.
  - Run more than one application

- The Internet transfers data between computers.

- The Internet is a network of networks.

Reference: The Design Philosophy of the DARPA Internet Protocols

By David Clark

Design requirements and techniques to meet them

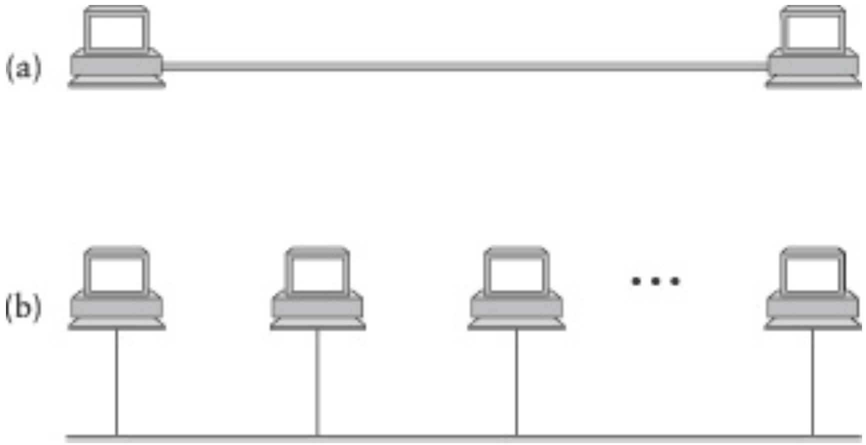
1. Scalable connectivity
2. Cost-effective resource sharing
3. Support for common services
4. Manageability

1. Scalable connectivity

- A network must provide connectivity among a set of computers
  - Open vs close: to connect all computers or a subset of them?
  - Internet is an open network

- Scalability: A system is designed to grow to an arbitrary large size is said to scale
  - How to connect an arbitrary large number of computers on a network?

Connectivity recursively occurs at different levels

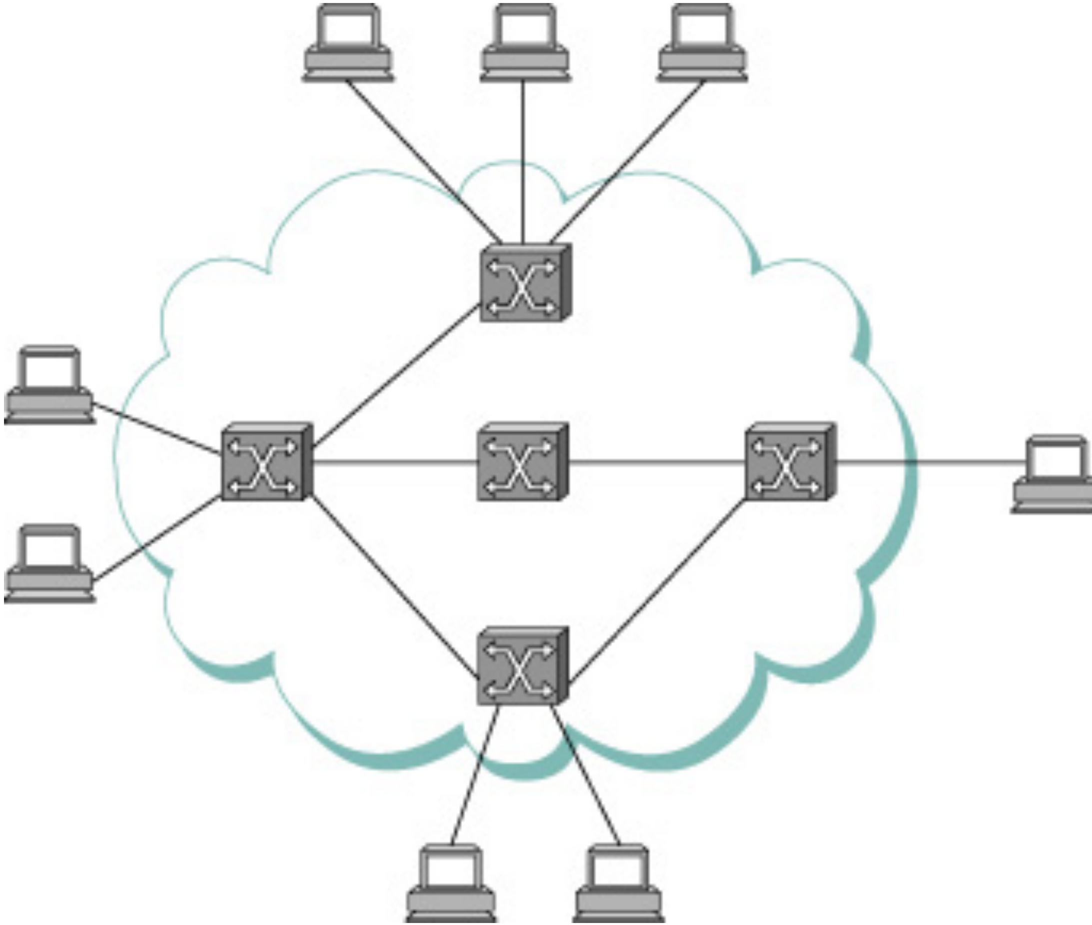


Point-to-Point

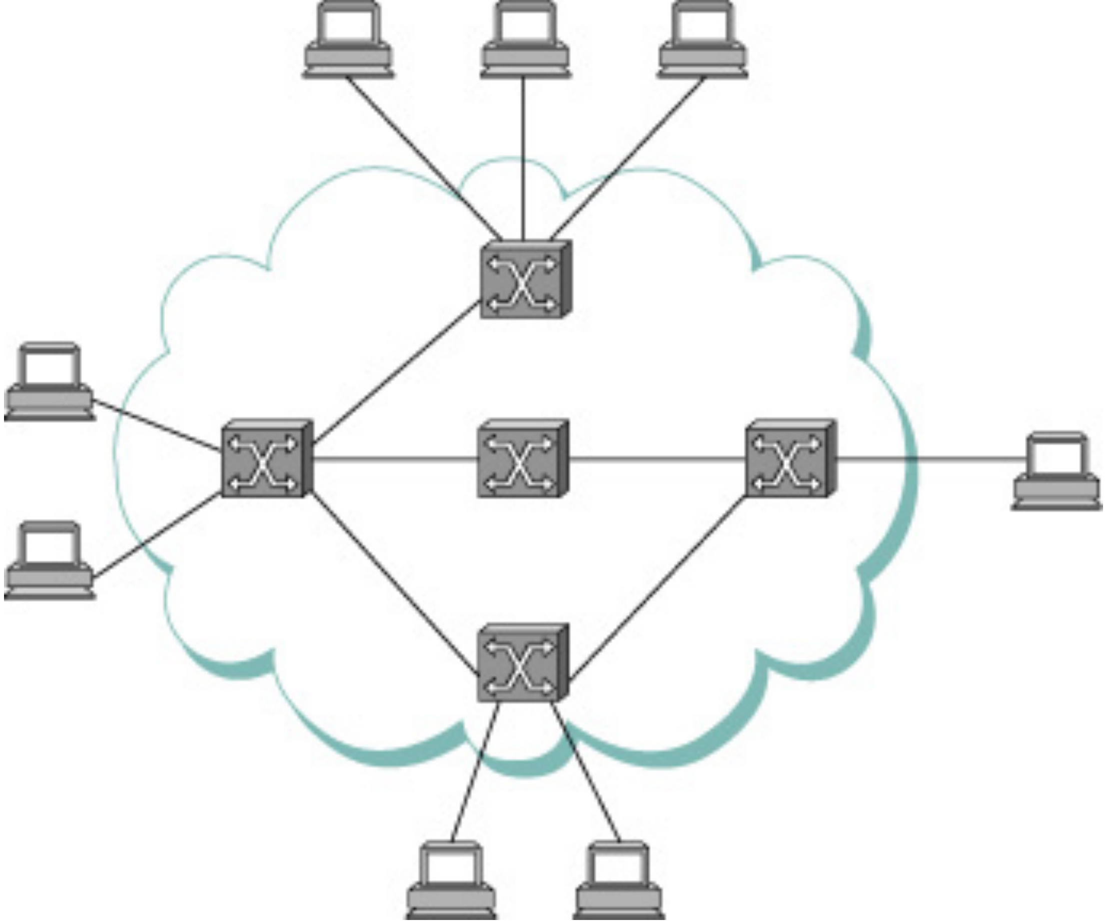
Multiple-Access

- Link-level: connect two or more computers via a physical medium or electromagnetic waves
- Computers are referred to as nodes
- The physical medium is referred to as a link

Switching

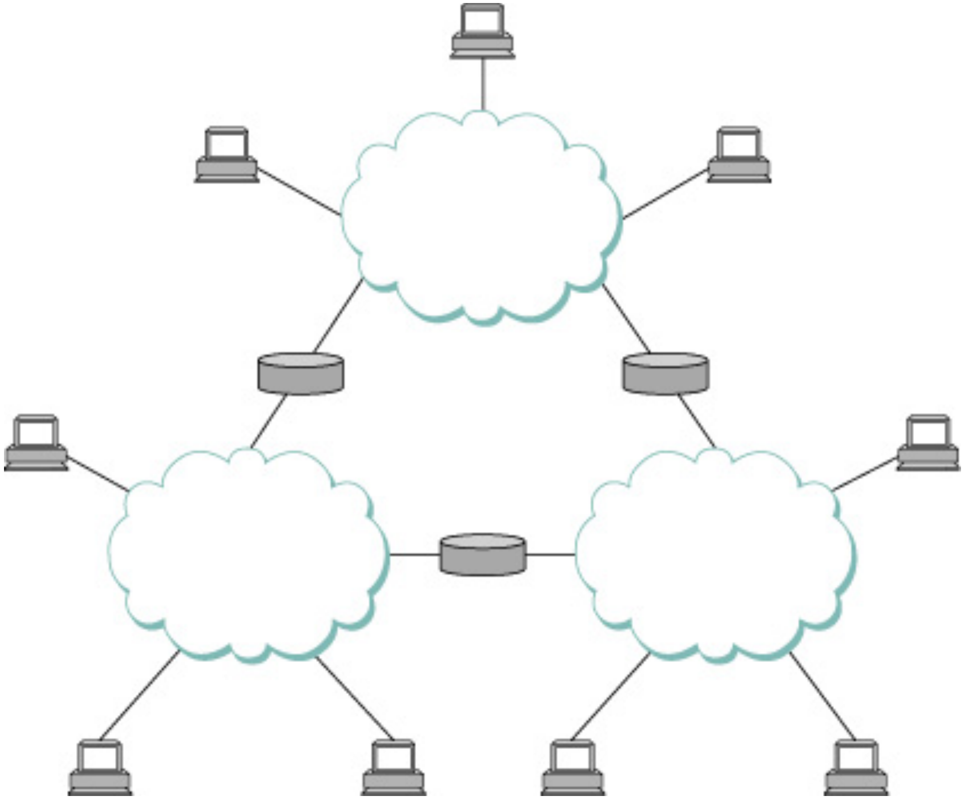


- Switching is a mechanism to achieve connectivity
- Nodes that are attached to at least two links forward data from one link to another link
- They are called switches
- Computers outside the cloud are called hosts



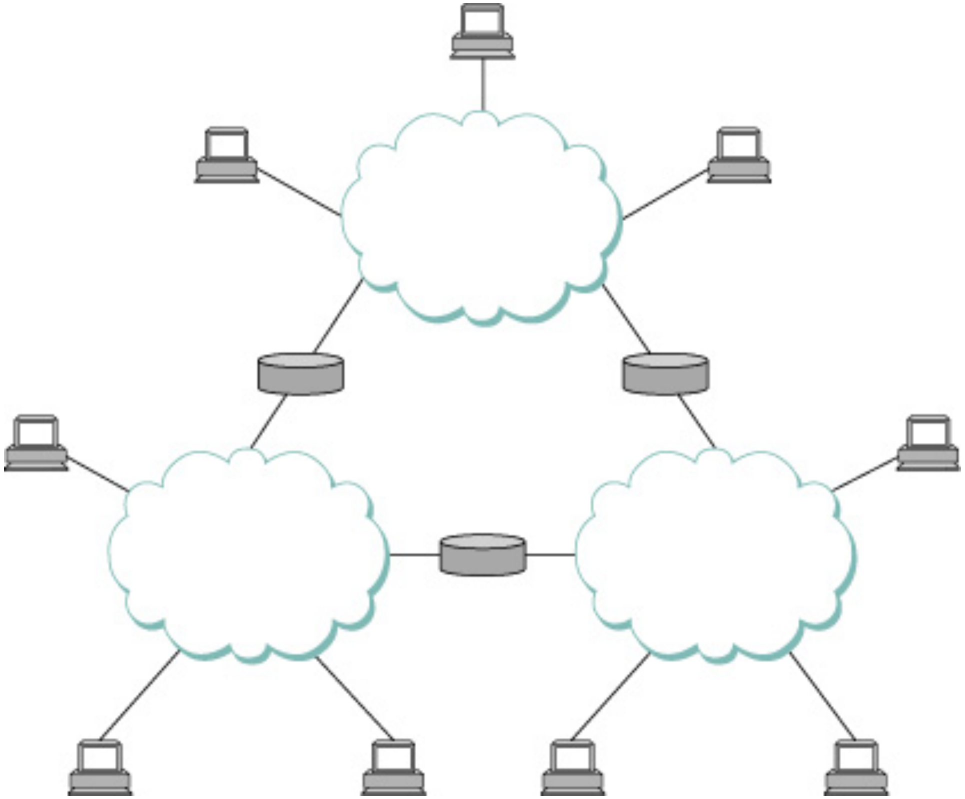
- Circuit switching
  - Sets up a circuit before nodes can communicate
  - Switches connect circuits on different links
- Virtual circuit switching
- Packet switching
  - Data are split into blocks of data called packets
  - Store and forward
  - Nodes send packets and switches forward them

Inter-networking: Another way to achieve connectivity



- An internetwork of networks
  - Each cloud is a network/a multiple-access link
  - A node that is connected to two or more networks is commonly called a router
    - Speaks different protocols than switches
  - An internet can be viewed as a "cloud." We can recursively build larger clouds by connecting smaller ones

Addressing and routing



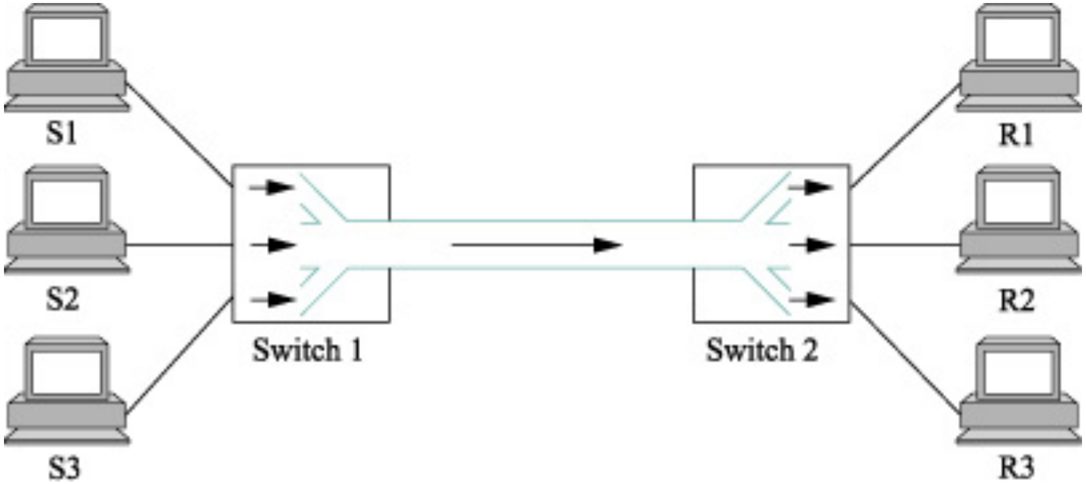
- Physical connectivity != connectivity
- Addressing and routing are mechanisms to achieve connectivity
- Nodes are assigned addresses
- Routers compute how to reach them by running routing protocols

2. Cost-effective resource sharing

- Question: how do all the hosts share the network when they want to communicate with each other?
  - Use at the same time
  - Fair

- Multiplexing: a system resource is shared among multiple users
  - Analogy: CPU sharing

- Mechanisms to multiplexing
  - Time-division multiplexing (TDM)
  - Frequency-division multiplexing (FDM)
  - Statistical multiplexing
  - …



Multiplex

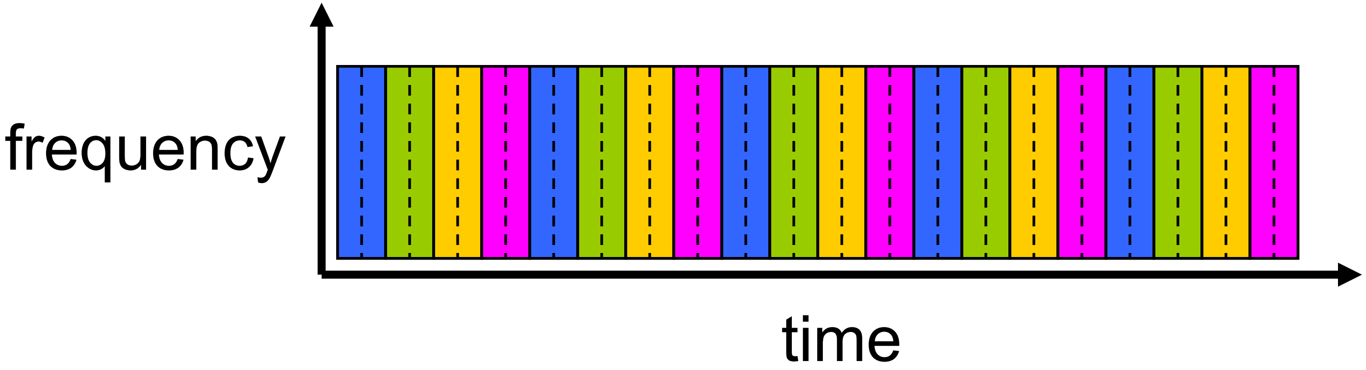
Demultiplex

TDM and FDM

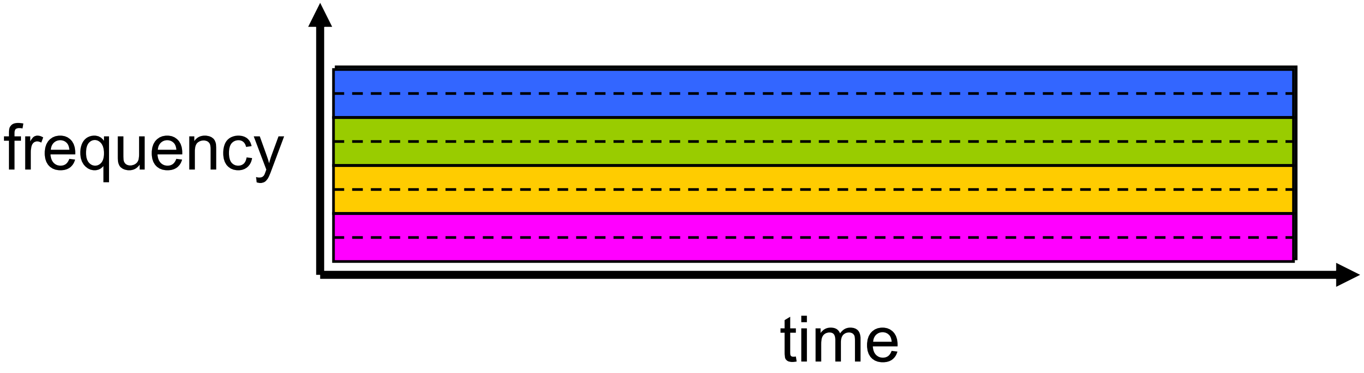
Example:

4 users 🟦🟩🟨🟪

TDM



FDM

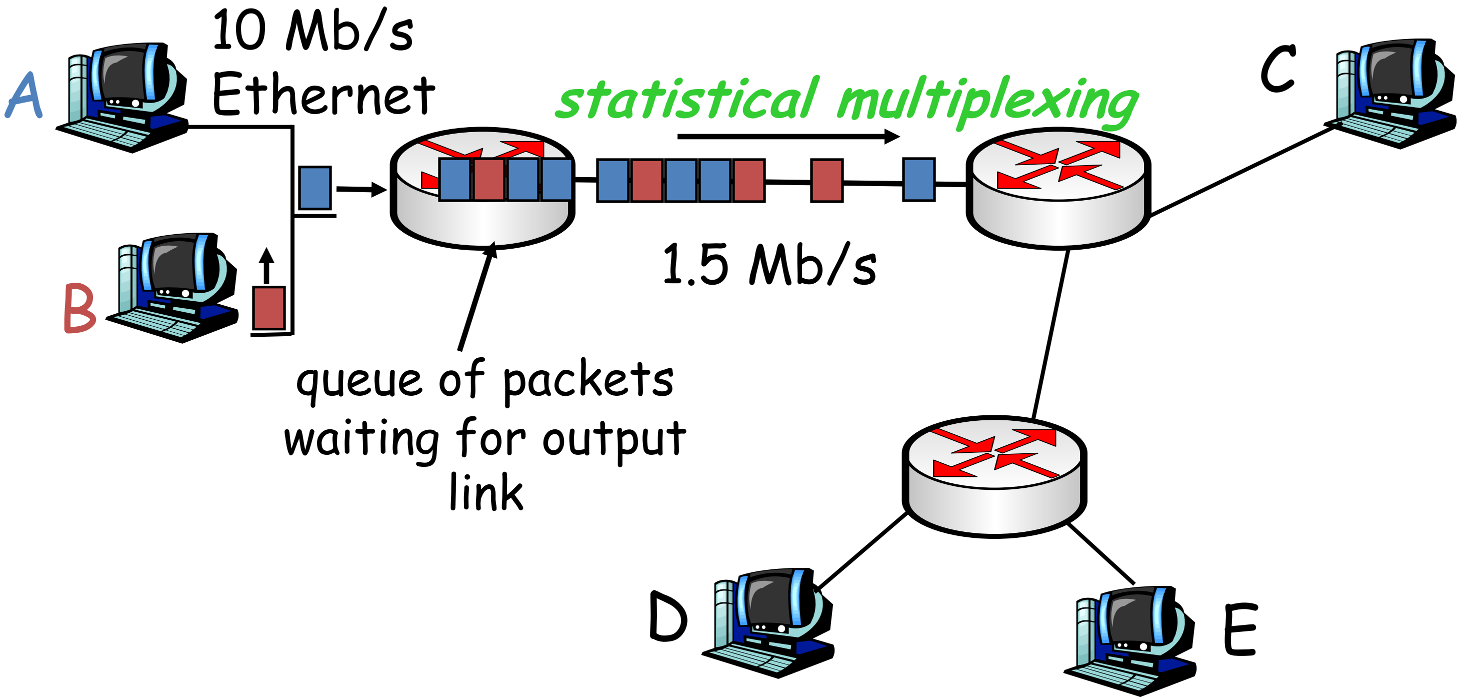


Problems with FDM and TDM

- What if a user does not have data to send all the time (Over-provision)?
  - Consider web browsing
  - → Inefficient use of resources

- Max # of flows is fixed and known ahead of time (Under-provision)
  - Not practical to change the size of quantum or add additional quanta for TDM
  - Nor add more frequencies in FDM

Statistical Multiplexing



- The physical link is shared over time (like TDM)
- But does not have fixed pattern. This is called ***statistical multiplexing***
  - Sequence of A & B packets are sent on demand, not predetermined slots

Pros and Cons

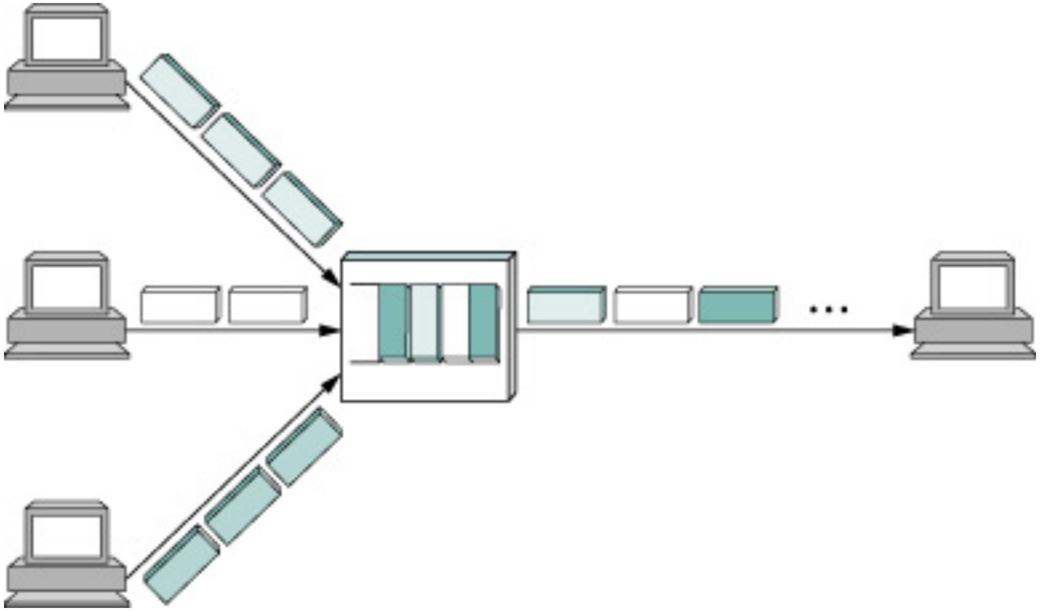
- Assumption: traffic is largely bursty
- Pros: Resources are not wasted when hosts are idle
- Cons: No guarantee flows would have their turns to transmit
- Some possible fixes:
  - Limit maximum packet size
  - Scheduling which packets got transmitted, e.g., fair queuing

Maximum Packet Size

- Divide an application message into blocks of data → packets
  - Names at different layer: Segments, frames

- Maximum packet size limit
  - Flows sent on demand
  - Must give each flow its turn to send
  - Solution: defines an upper bound on the size of the block of data

Packet scheduling



- Scheduling: which packet to send
  - – First come first serve (FIFQ)
  - – Weighted fair queuing

Switching vs multiplexing

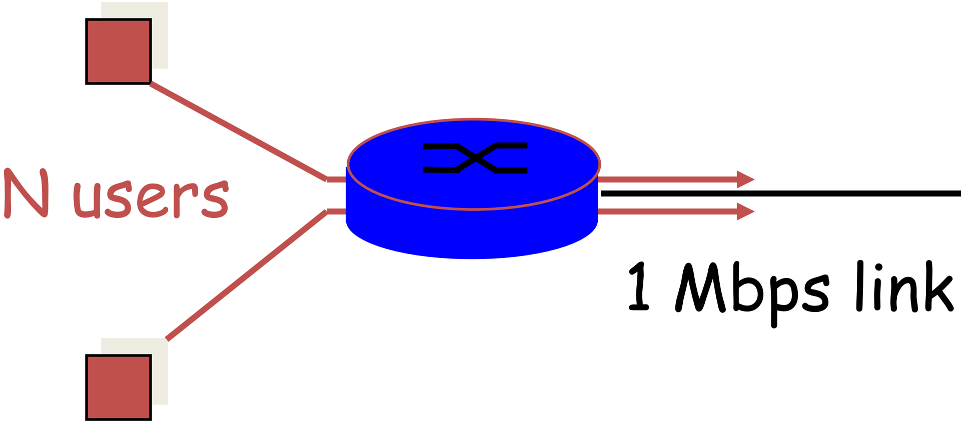
- TDM and FDM are used in circuit switching networks
  - Require a setup as max # of flows is fixed

- SM is used in packet switching networks

Statistical switching versus circuit switching

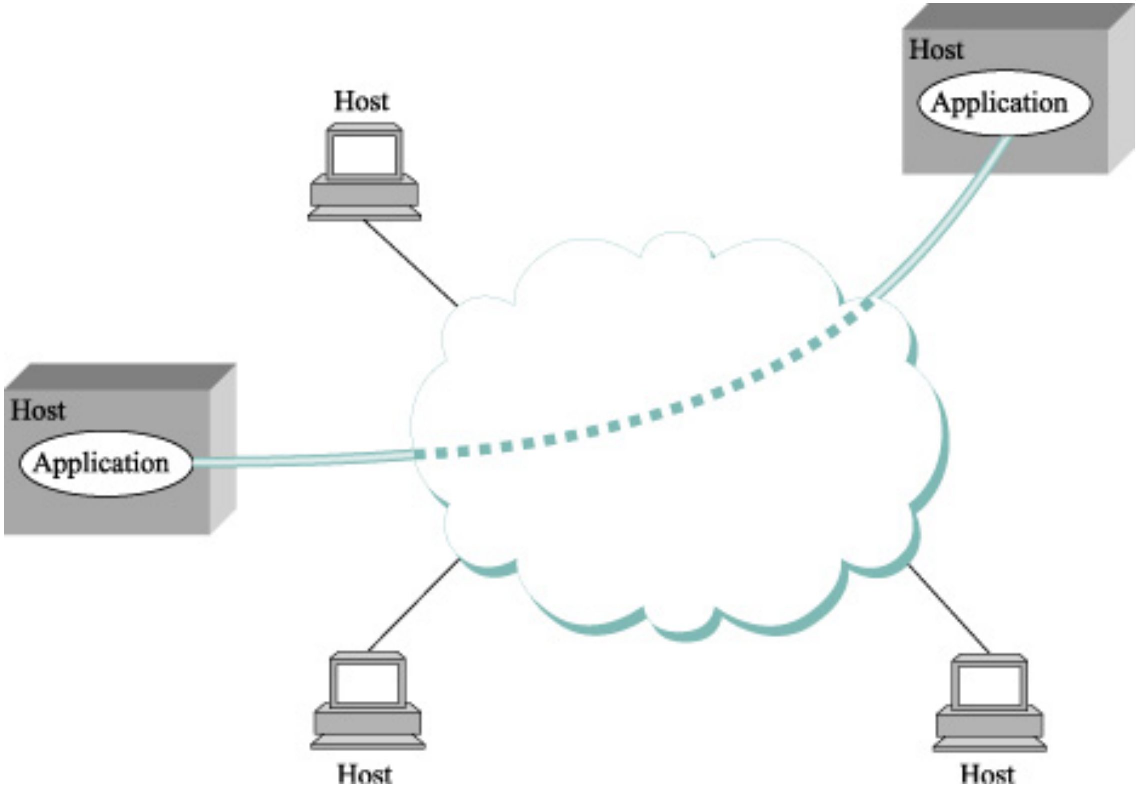
Statistical switching allows more users to use the network!

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time
- circuit-switching: fixed capacity
  - 10 users
- statistical switching
  - When they are 35 users
  - Not congested when 0, 1, …, 10 users are active at the same time
  - Congested = 1- $\sum_{i=0}^{10} C_{35}^{i}\, 0.1^{i} 0.9^{35-i}$<= 0.000424298



3. Support for common services

- Application developers want a network to provide services that make application programs communicate with each other, not just sending packets
  - E.g. reliably delivering an email message from a sender to a receiver
- Many complicated things need to happen
  - Can you name a few?
- Design choices
  - Application developers build all functions they need
  - Network provides common services → a layered network architecture
    - Build it once, and shared many times



- Interactive request/reply
- Streaming of data
- Bulk data transfer
- …

- Key challenges: what services/channels to provide that can satisfy most applications at lowest costs?

- Approach: identify common patterns, then decide
  - What functions to implement
  - Where to implement those functions

4. Manageability

- Manage the network as it grows and when things go wrong
- An open research challenge
  - Datacenter networks
  - Backbones
  - Home networks
    - IP cameras, printers, network attached storage
  - Software defined networking

Overview

- Design requirements of the original Internet
- Concepts of Network Architectures
  - How are we going to meet those requirements?

Network Architectures

- Many ways to build a network
- Use network architectures to characterize different ways of building a network
- The general blueprints that guide the design and implementation of networks are referred to as network architectures

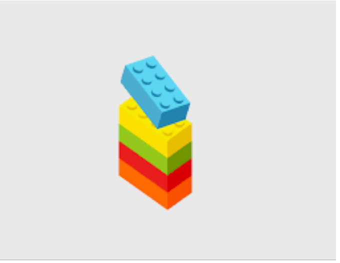
Central concepts

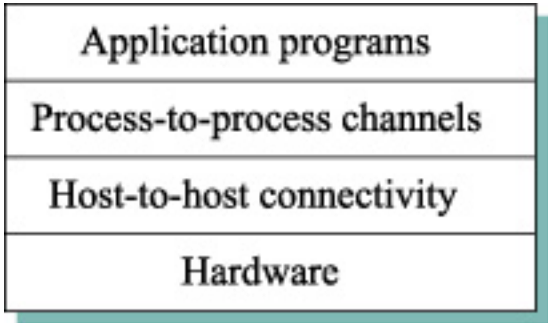
- Layering
- Protocols

A layered architecture

- Many sub-tasks need to be accomplished
  - Find a path to the destination, reliably transfer information
- The complexity of the communication task is reduced by using multiple protocol layers:
    - Each protocol is implemented independently
    - Each protocol is responsible for a specific subtask
    - Protocols are grouped in a hierarchy
- The old divide-and-conquer principle!

Layering





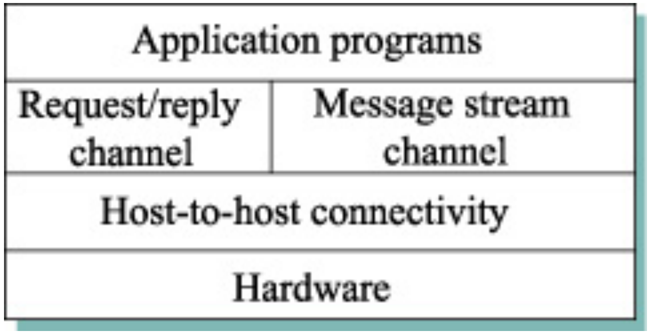
Application programs

Process-to-process channels

Host-to-host connectivity

Hardware

Not so strict



Application programs

Request/reply channel | Message stream channel

Host-to-host connectivity

Hardware

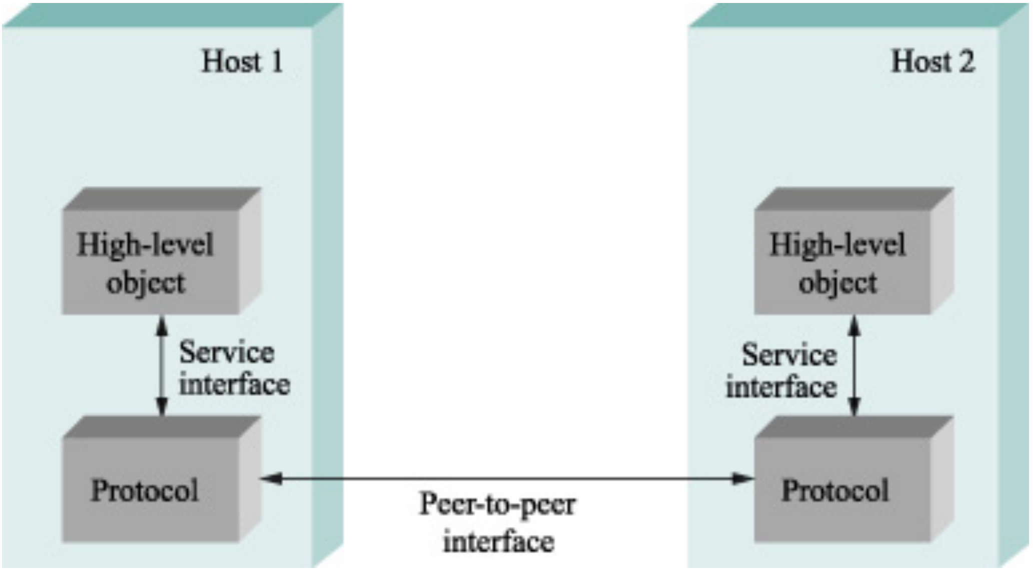
- An abstraction to handle complexity
  - A unifying model that capture important aspect of a system
  - Encapsulate the model in an object that has an interface for others to interact with
  - Hide the details from the users of the object

Advantages of layering

- Simplify the design tasks
  - Each layer implements simpler functions

- Modular design
  - Can provide new services by modifying one layer

Protocols



- The abstract objects that make up the layers of a network system are called protocols

- Each protocol defines two different interfaces
  - Service interface
  - Peer interface

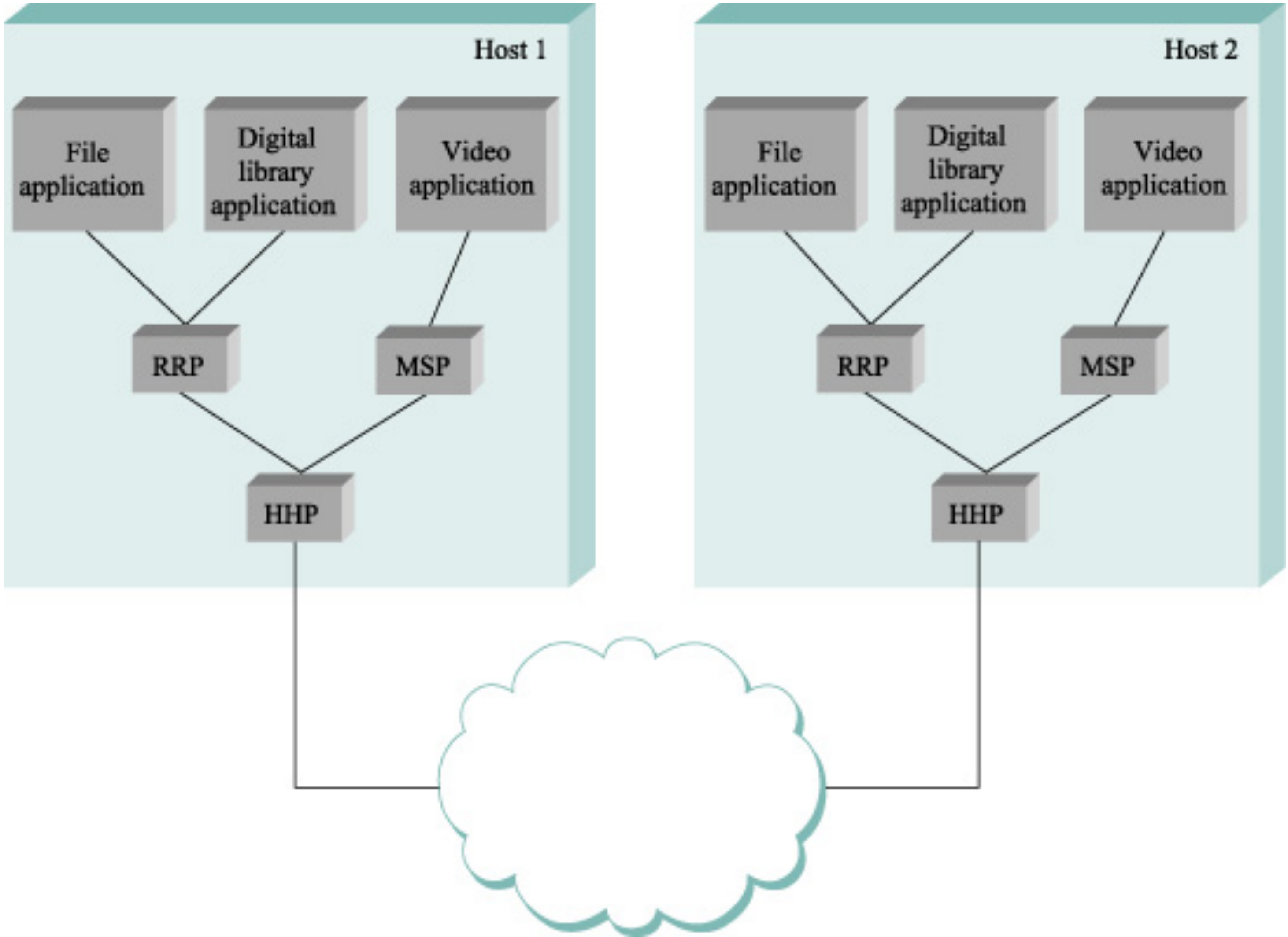
A protocol graph

- Peer-to-peer communication is indirect
  - Except at the hardware level

- Potentially multiple protocols at each level

- Show the suite of protocols that make up a network system with a protocol graph

A sample protocol graph



Protocol standardization

- Standard bodies such as IETF govern procedures for introducing, validating, and approving protocols
  - The Internet protocol suite uses open standard

- Set of rules governing the form and content of a protocol graph are called a network architecture

We reject kings, presidents, and voting. We believe in rough consensus and running code

- David Clark

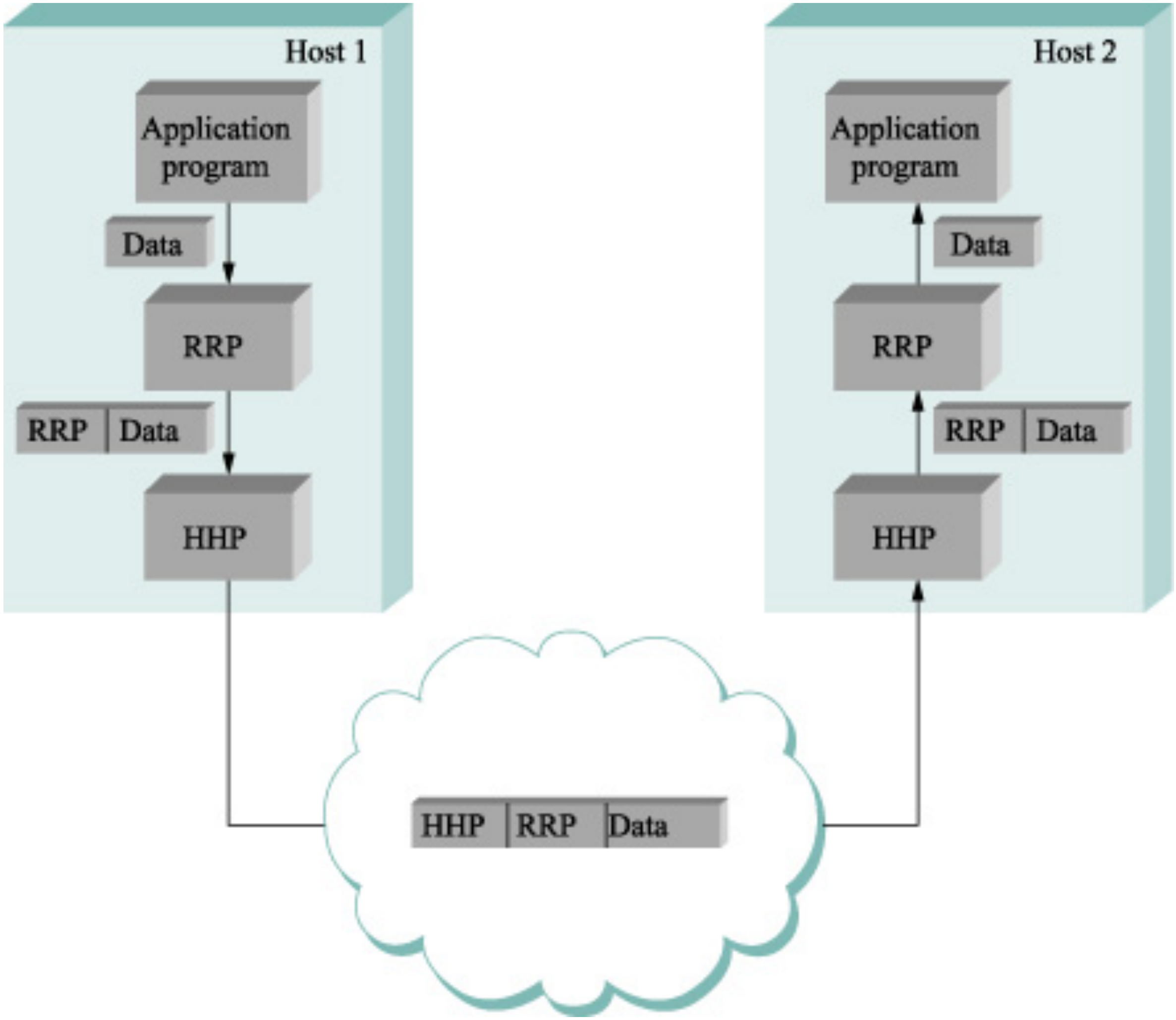
Encapsulation

- Upper layer sends a message using the service interface

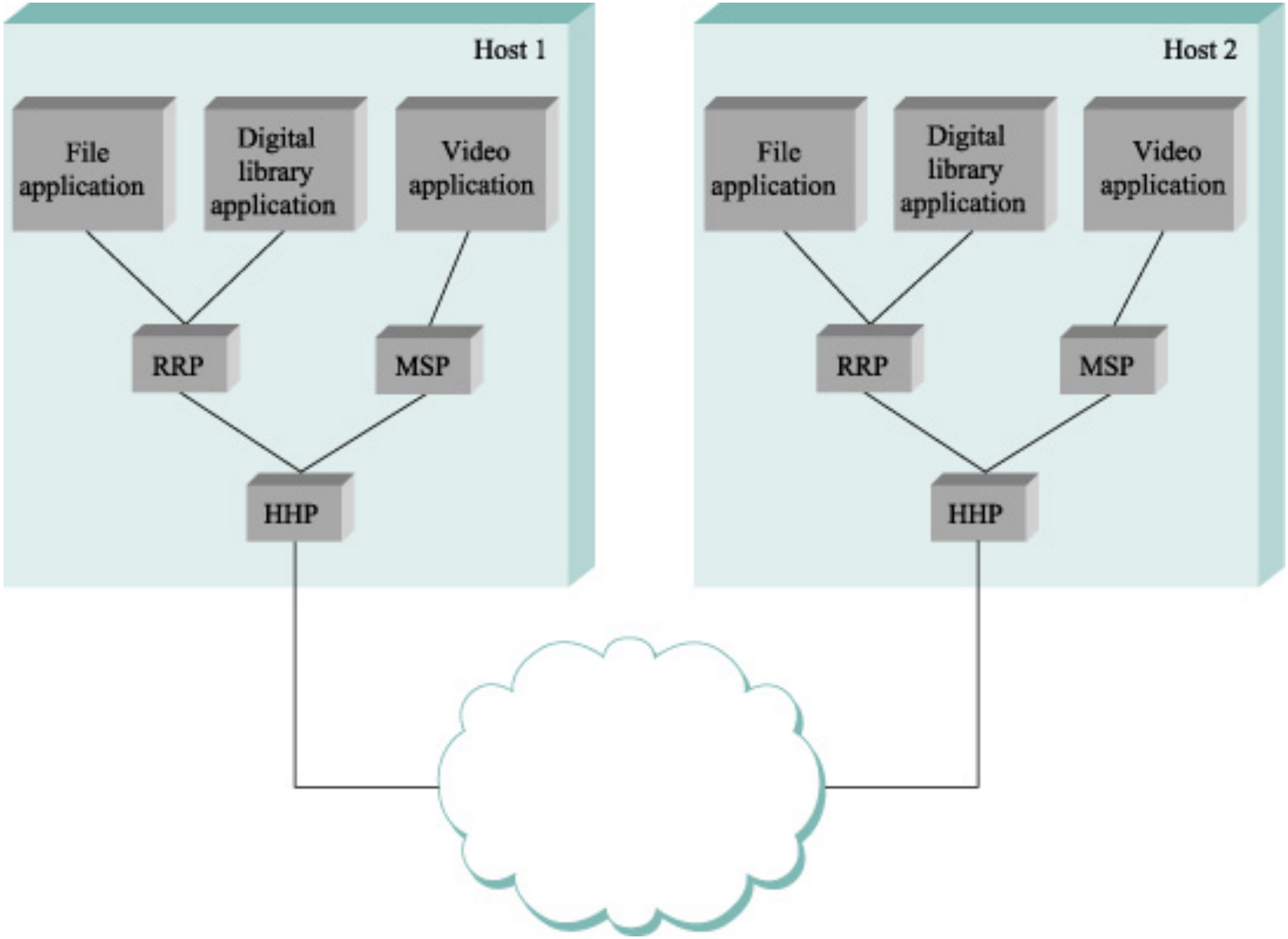
- A header, a small data structure, to add information for peer-to-peer communication, is attached to the front message
  - Sometimes a trailer is added to the end

- Message is called payload or data

- This process is called encapsulation


Host 1
Application program
Data
RRP
RRP | Data
HHP
Host 2
Application program
Data
RRP
RRP | Data
HHP
HHP | RRP | Data

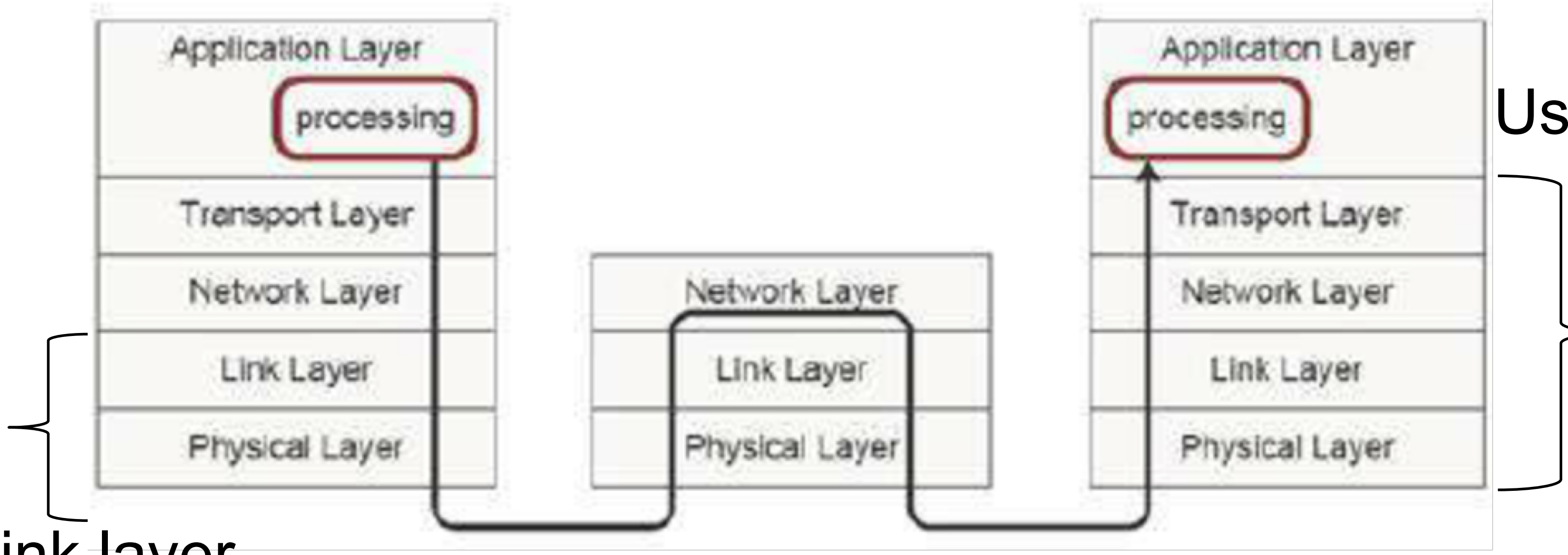
Multiplexing & Demultiplexing



- Same ideas apply up and down the protocol graph

Examples of Network Architectures

The Internet Protocol Suite



User space

OS

Link layer

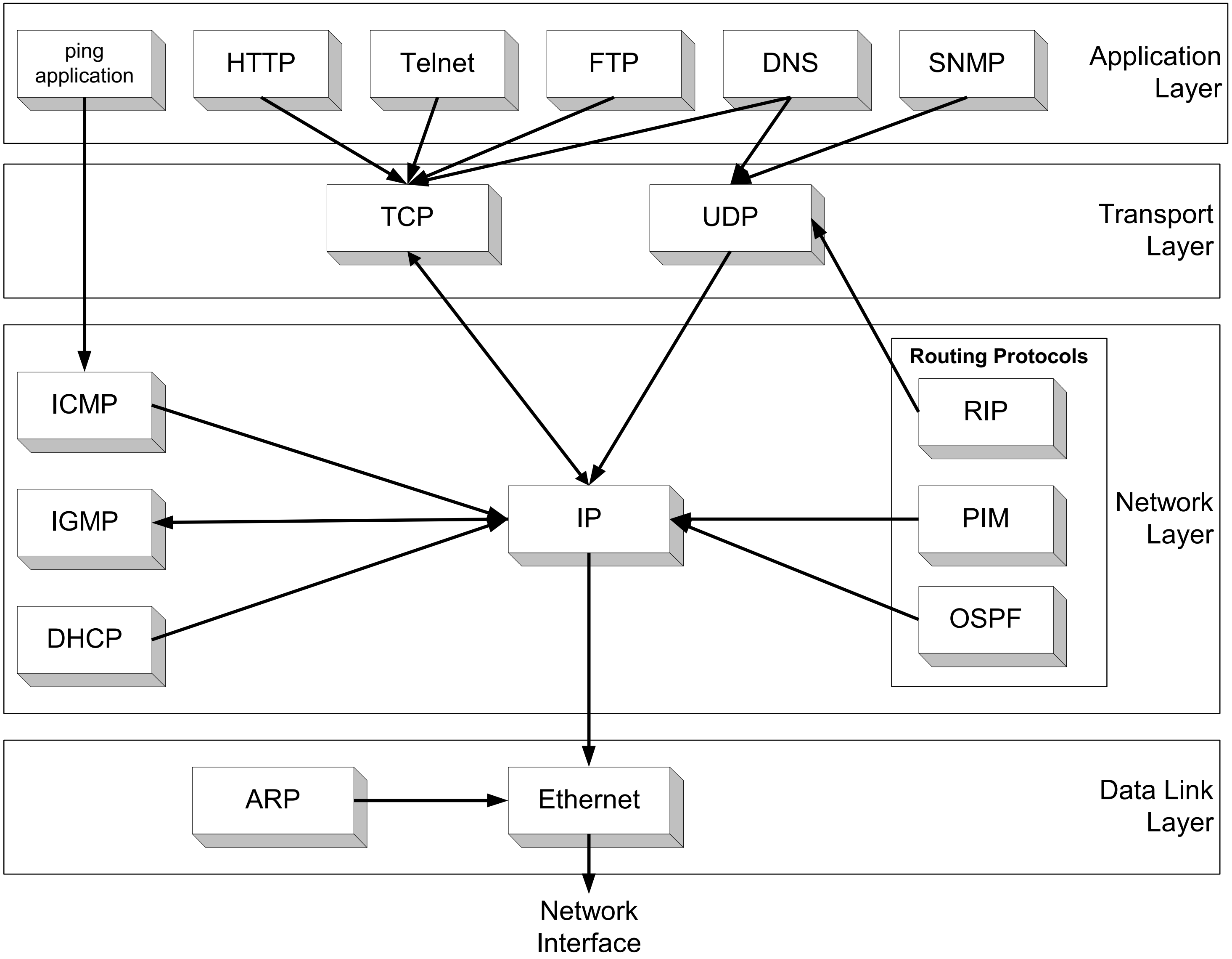
- The Internet architecture has four layers: Application, Transport, Network, and Data Link Layer (logical link layer, and physical link layer)

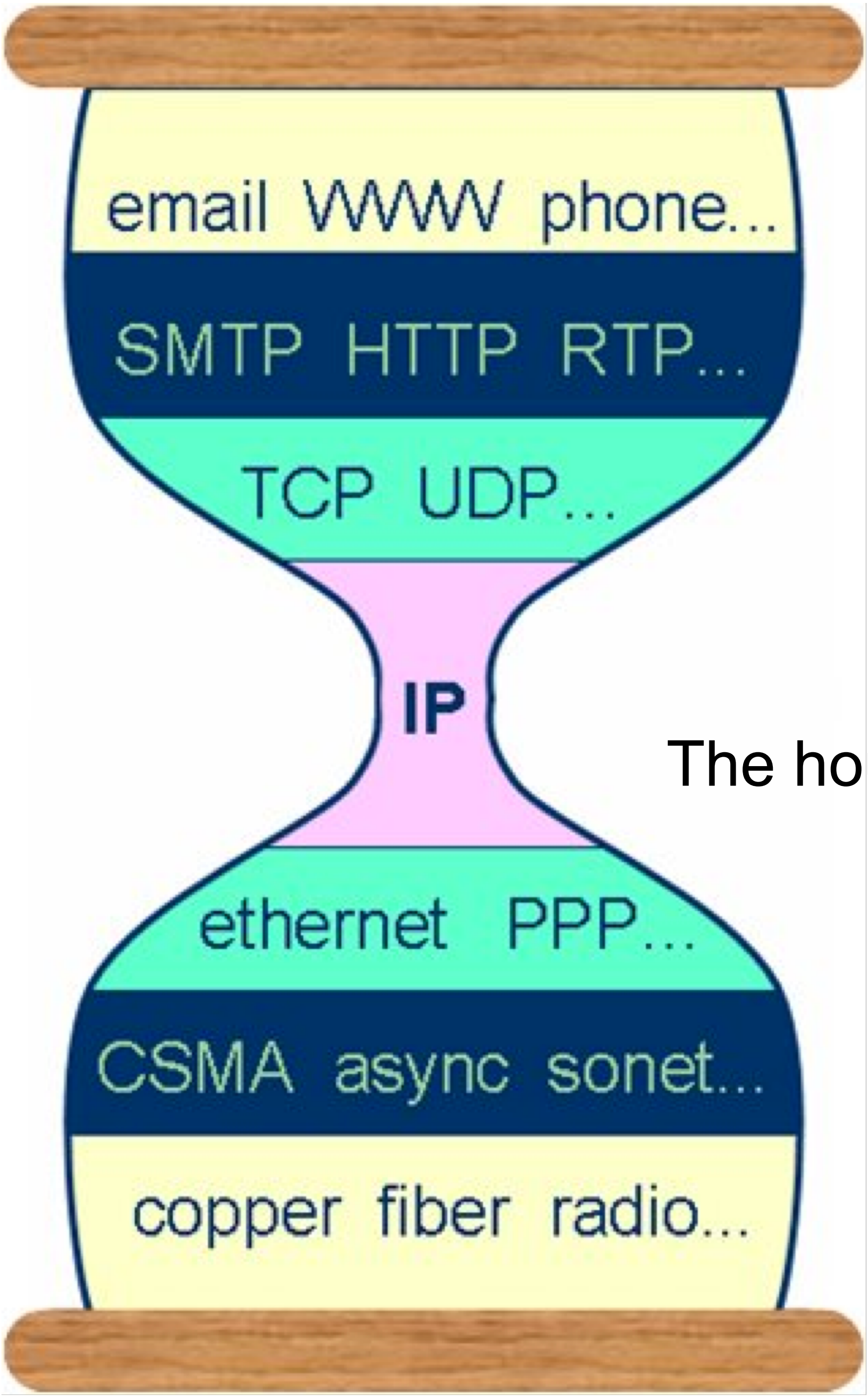
- Sending or receiving a packet from end systems (hosts) may involve actions of all four layers. Packet forwarding by (Routers) only involve the bottom three layers.

Functions of the Layers

- Data Link Layer: (layer 2)
  - Service: Reliable transfer of frames over a link
Media Access Control on a LAN
  - Functions: Framing, media access control, error checking
- Network Layer: (layer 3)
  - Service: Move packets from source host to destination host
  - Functions: Routing, addressing
- Transport Layer: (layer 4)
  - Service: Delivery of data between hosts
  - Functions: Connection establishment/termination, error control, flow control
- Application Layer:
  - Service: Application specific (delivery of email, retrieval of HTML documents, reliable transfer of file)
  - Functions: Application specific

Assignment of Protocols to Layers

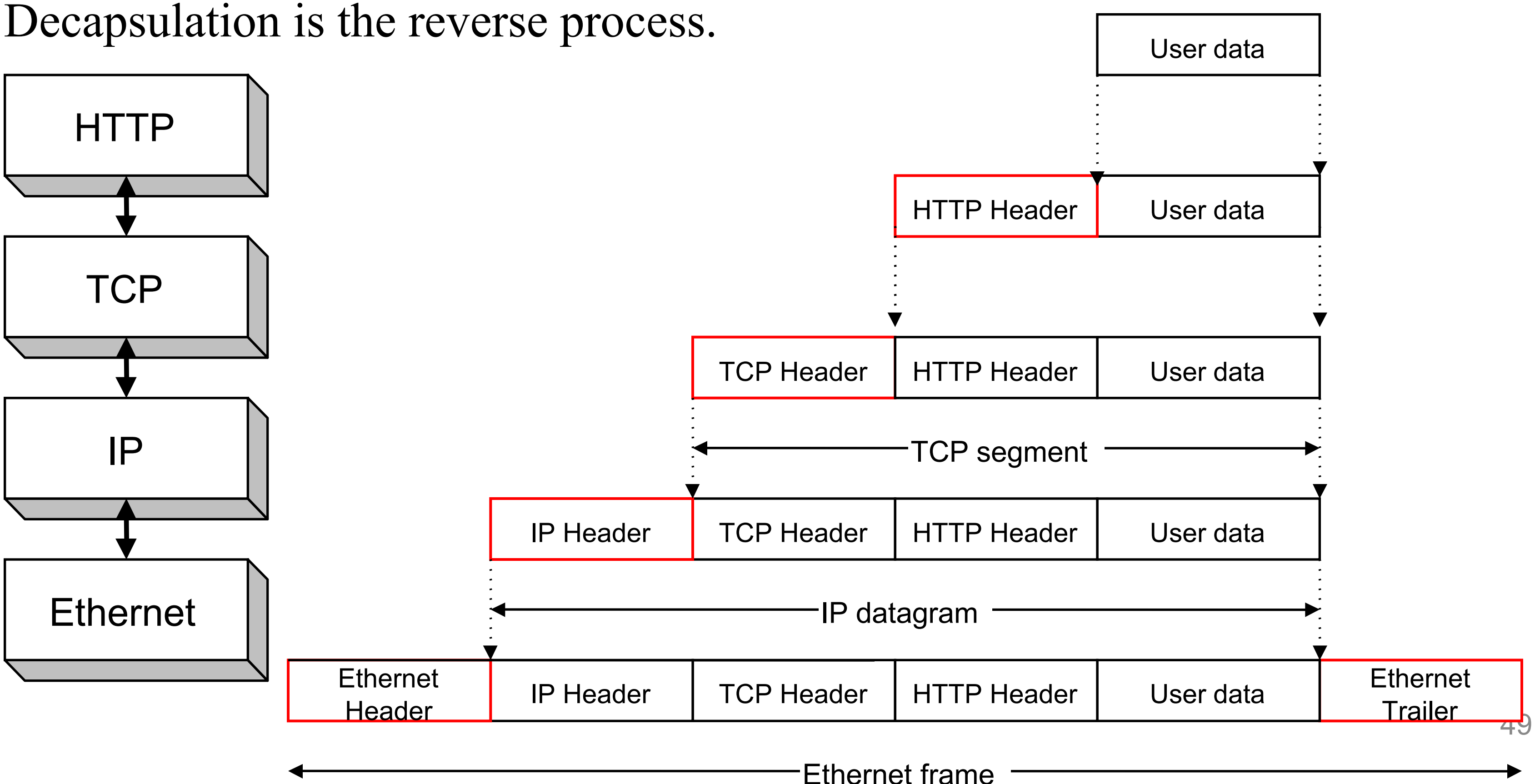




The hourglass model

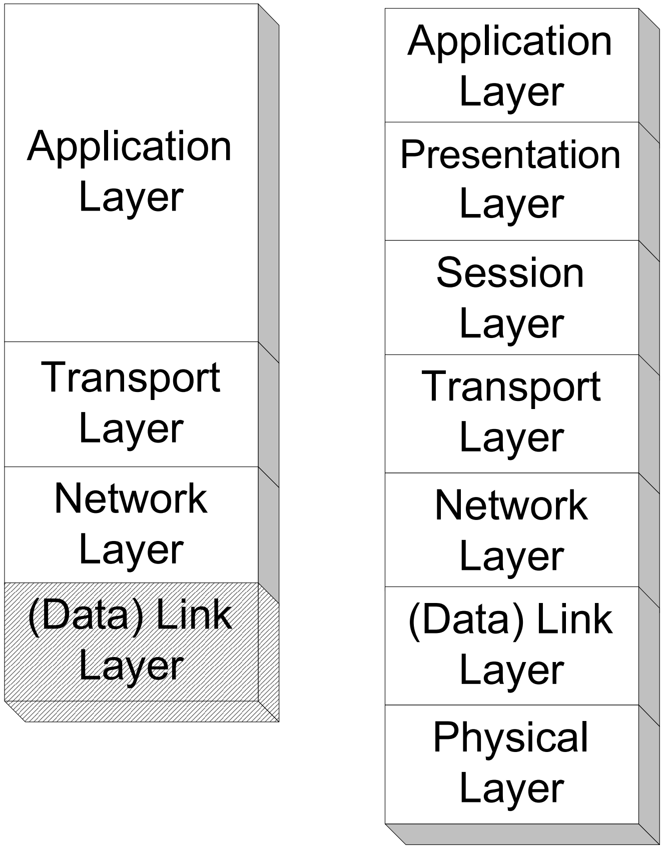
Use Encapsulation and Decapsulation to demultiplex

- Encapsulation: As data is moving down the protocol stack, each protocol is adding layer-specific control information.
- Decapsulation is the reverse process.



TCP/IP Suite vs OSI Reference Model

The TCP/IP protocol stack does not define the lower layers of a complete protocol stack



Application Layer

Transport Layer

Network Layer

(Data) Link Layer

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

(Data) Link Layer

Physical Layer

TCP/IP Suite

OSI Reference Model

- International Telecommunications Union (ITU) publishes protocol specs based on the OSI reference model
  - X dot series

- Physical layer: handles raw bits
- Data link layer: aggregate bits to frames. Network adaptors implement it
- Network layer: handles host-to-host packet delivery. Data units are called packets
- Transport: implements process channel. Data units are called messages
- Session layer: handles multiple transport streams belong to the same applications
- Presentation layer: data format, e.g., integer format, ASCII string or not
- Application layer: application specific protocols

Summary

- The design requirement of the Internet
- Network architectures that meet the design requirement
- New terms
  - Scalability, nodes, links, switches, routers, multiplexing/demultiplexing, circuit switching, packet switching, statistical multiplexing, layering, protocols, encapsulation/decapsulation, network architetures

End-to-End Argument

- Extremely influential
- "…functions placed at the lower levels may be *redundant* or of *little value* when compared to the cost of providing them at the lower level…"
- "…sometimes an *incomplete* version of the function provided by the communication system (lower levels) may be useful as a *performance enhancement*…"

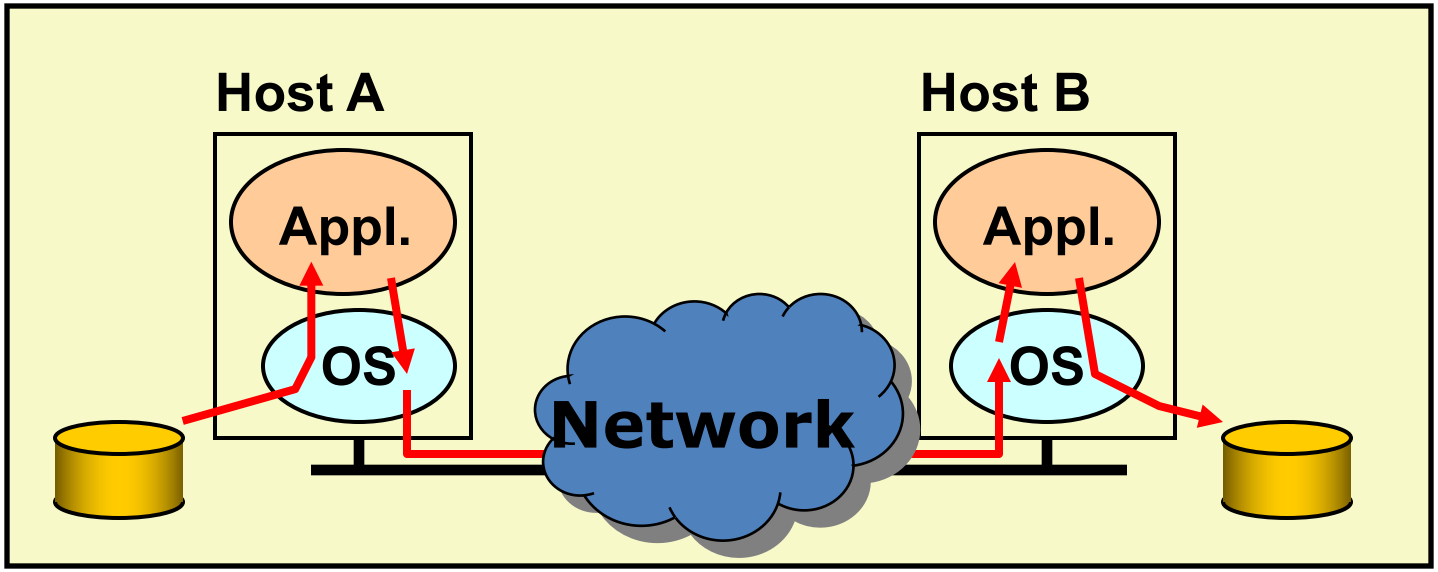
The counter argument

- Modularity argument:
  - It is tempting to implement functions at lower layers so that higher level applications can reuse them
- The end-to-end argument:
  - "The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of communication."
  - "Centrally-provided versions of each of those functions will be incomplete for some applications, and those applications will find it easier to build their own version of the functions starting with datagrams."

Techniques used by the authors

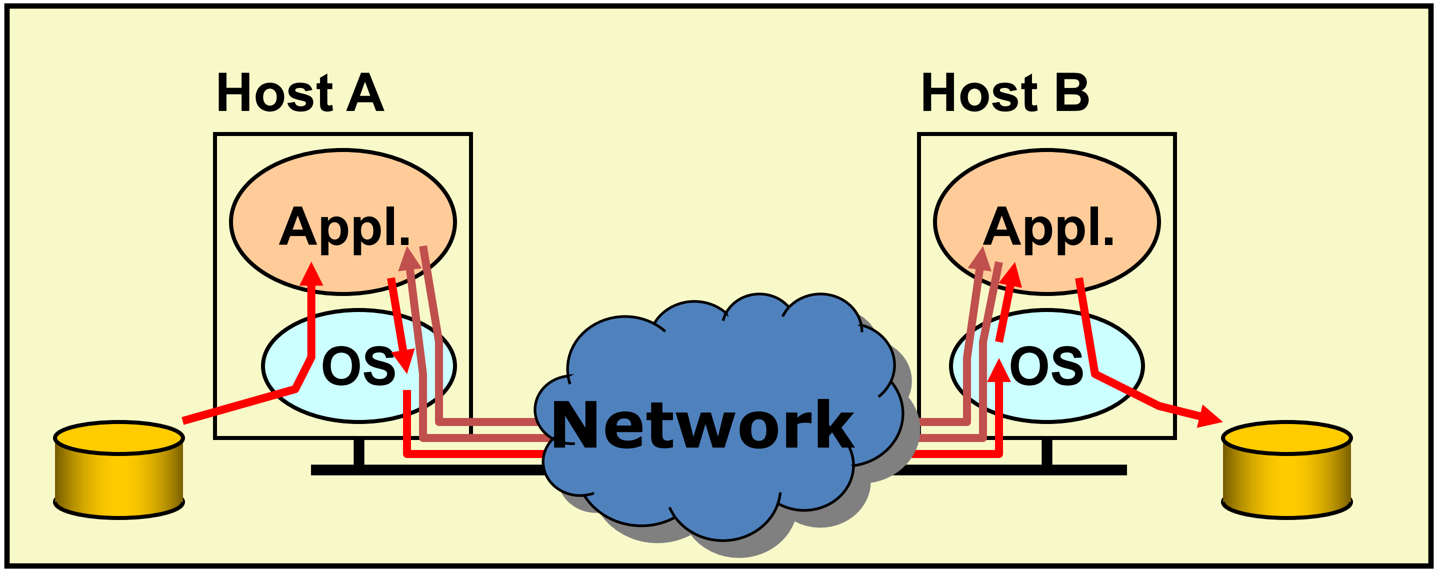
- The authors made their argument by analyzing examples
  - Reliable file transfer
  - Delivery guarantees
  - Secure data transmission
  - Duplicate message suppression
  - FIFO
  - Transaction management
  - *Can you think of more examples to argue for or against the end-to-end argument*?
- Can be applied generally to system design

Example: Reliable File Transfer



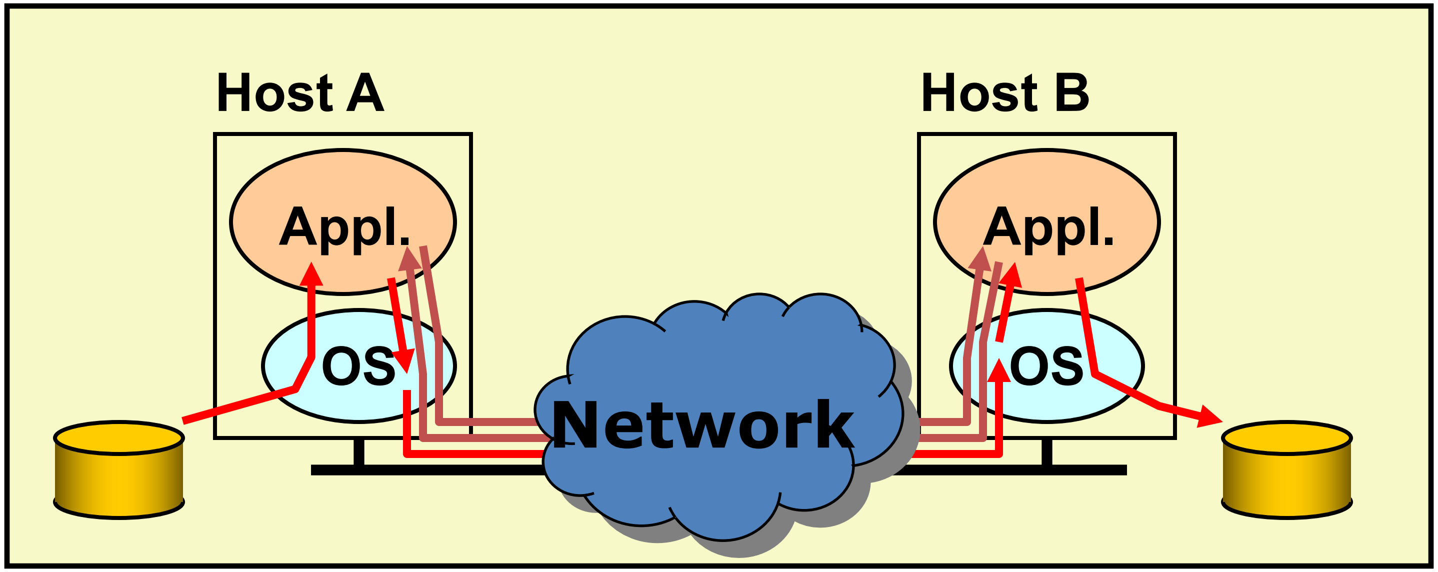
- Solution 1: make each step reliable, and then concatenate them
  - Uneconomical if each step has small error probability

Example: Reliable File Transfer



- Solution 2: end-to-end check and retry
  - Correct and complete

Example: Reliable File Transfer



- An intermediate solution: the communication system provides internally, a guarantee of reliable data transmission, e.g., a hop-by-hop reliable protocol
  - Only reducing end-to-end retries
  - No effect on correctness

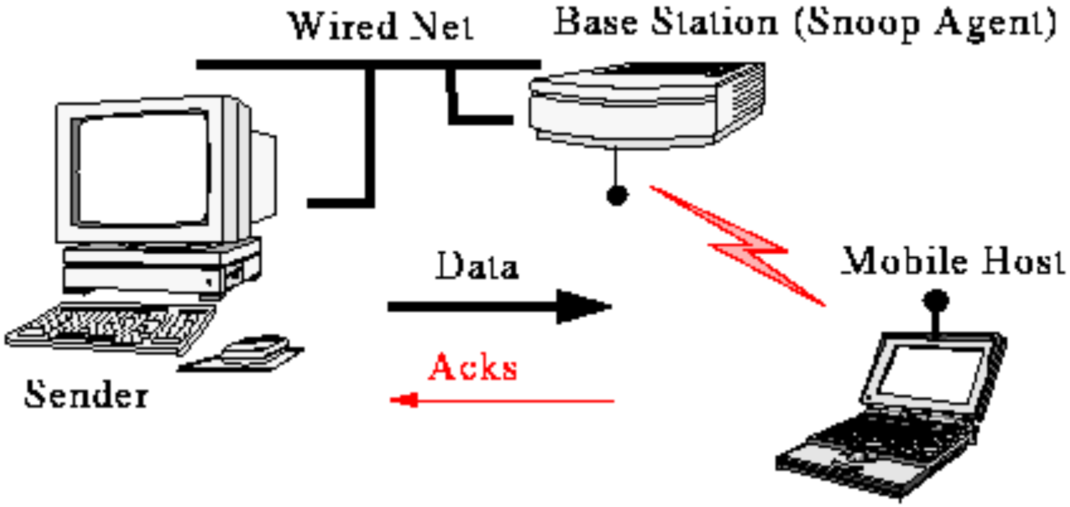
Question: should lower layer play a part in obtaining reliability?

- Answer: it depends
  - Example: extremely lossy link
    - One in a hundred packets will be corrupted
    - 1K packet size, 1M file size
    - Prob of no end-to-end retry: $(1-1/100)^{1000}$ ~ 4.3e-5

Performance enhancement

- “put into reliability measures within the data communication system is seen to be an engineering tradeoff based on performance, rather than a requirement for correctness.”

Performance tradeoff is complex



- Example: reliability over a lossy link using retries

Performance tradeoffs

- Example: reliability over a lossy link using retries
  - But they wont help real time applications, applications with built-in error correction mechanisms
- Tradeoffs:
  - Applications that do not need them will pay the cost anyway
  - Low-level subsystems may not have as much information as the higher levels to do the job as efficiently

End-to-End Argument: Discussion

- The original end-to-end argument emphasizes correctness & completeness, not
  - complexity: is complexity at edges result in a “simpler” architecture?
  - evolvability, ease of introduction of new functionality: ability to evolve because easier/cheaper to add new edge applications than change routers?
  - Technology penetration: simple network layer makes it “easier” for IP to spread everywhere

Summary: End-to-End Arguments

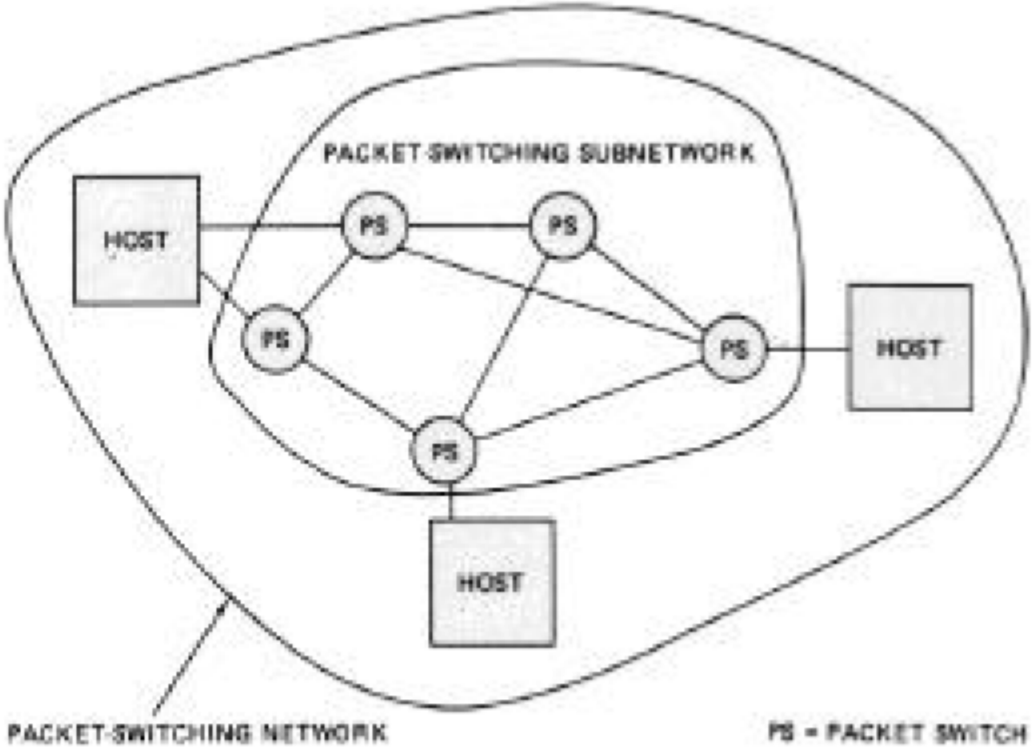
- If the application can do it, don't do it at a lower layer -- anyway the application knows the best what it needs
  - add functionality in lower layers iff it is (1) used and improves performances of a large number of applications, and (2) does not hurt other applications
- Success story: Internet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

Historic design

- The original TCP/IP design paper
  - A protocol for packet network intercommunication by Cerf and Karn, 1974
  - An excellent case study

Goal: Interconnecting different networks

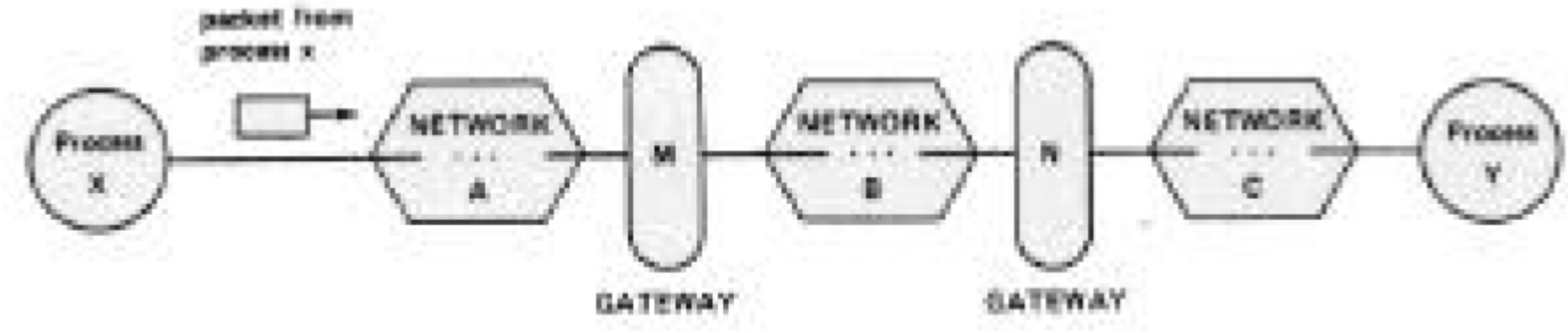
- Many different types of packet switch networks
  - ARPANET, packet satellite networks, ground-based packet radio networks, and other networks.
- Each has
  - Hosts, packet switches, processes
  - A protocol for communication
- Q: what would you do differently given such a design task?



Challenges

1. Different addressing schemes and host communication protocols
2. Different MTUs
3. Different success or failure indicators
4. End-to-end reliability: failures may occur at each subnet
5. Different control:
   - Status information, routing, fault detection/isolation

Inter- networking



- Gateways interface different networks
- A uniform data transmission control protocol (TCP)
- Uniform addressing (IP)

Inter-networking design alternatives

- Design alternative 1:
- Design alternative 2:

Inter-networking design alternatives

- Design alternative 1: one unified technology, a multi-media network
  - Restrictive
  - Not pratical: existing networks can’t be connected

- Design alternative 2: each host implements all other protocols
  - Expensive
  - Difficult to accommodate future developement

Maximum transmission unit size

- Solution: fragmentation
    - +
    - -
- Who reassembles?
- Design alternatives:

Maximum transmission unit size

- Fragmentation: gateways fragment packets into smaller units when MTU reduces
    - + packet sizes may vary in different networks
    - - complexity

- Who reassembles?
    - Destination
    - + gateway needs not handle the complexity, adding costs to all packets
    - + no need to reassemble again at downstream

MTU design alternative

- One max MTU for all networks
  - + simple
  - -. coupling: can't isolate internal max packet size of one network from other networks.
    - E.g. If a network can only support 512 bytes, then all networks can't exceed 512 bytes
  - - difficult to increase, requires agreements from all networks
  - - packet size may increase during transmission without source host knowing it.

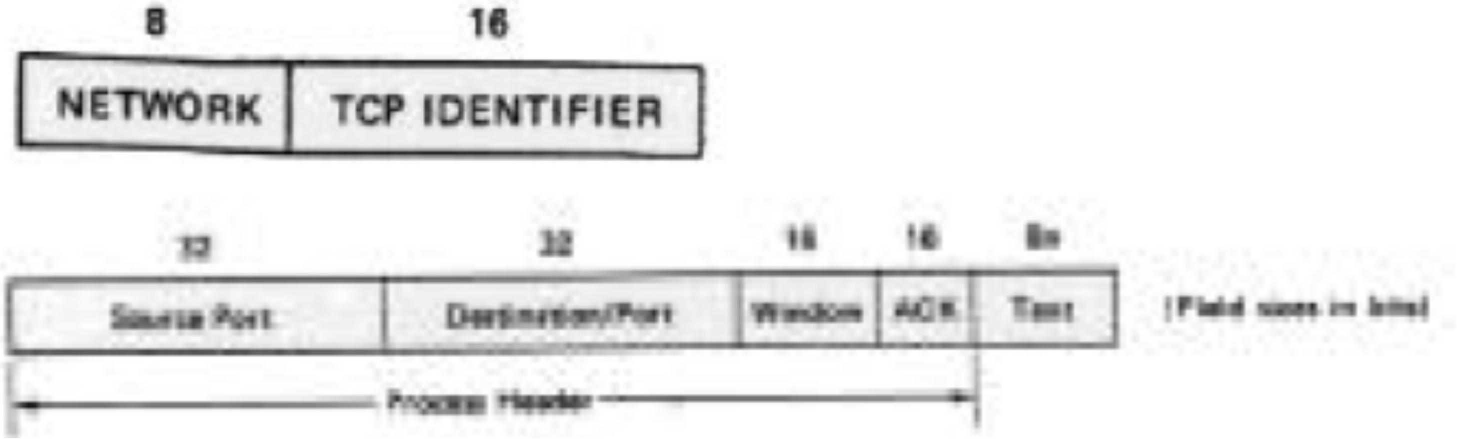
Process Level Communication

- Multiplexing and de-multiplexing messages among processes
- Design considerations: boundaries
  - Case 1: No process boundaries
    - +
    - -
  - Case 2: Separate messages from different processes
    - +
    - -

Process Level Communication

- Design: multiplexing and de-multiplexing messages among processes into segment streams
- Design considerations: boundaries
  - Case 1: No process boundaries
    - + More efficient, packing msgs to full segments
    - - interferences between processes
  - Case 2: Separate msgs from different processes
    - + process isolation
    - - overhead

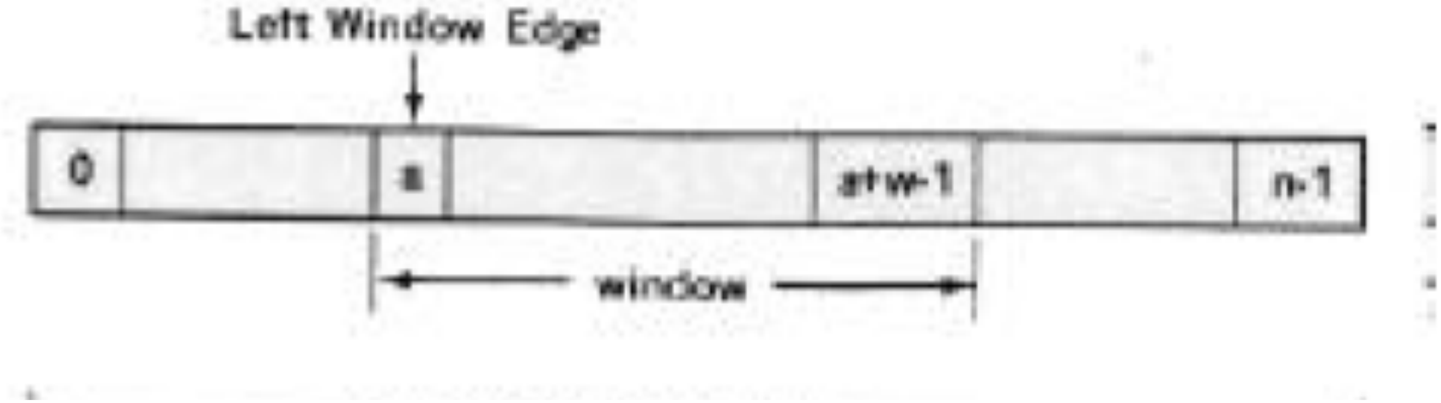
Addressing format



- A network address understood by all gateways
  - 8 bits
- A TCP identifier to identify a host
  - 16 bits
- Port number to identify a destination process
  - 16 bits

Other design considerations of TCP

- End to end reliability: sliding window



- Flow control: to prevent receiver buffer overflow
- Connection
  - Three-way SYN hand-shakes
  - Close

What are different today?

- Or what would you do differently?

What are different today?

- TCP/IP separation
  - Allows variety of services
- TCP implements byte stream interface
  - No notion of messages
  - Simple, and generic
- TCP has congestion control
  - We’ ll learn why soon
- Different addressing format, but same hierarchy
- It’ s amazing how much they got it right!