Fundamentals of Computer Networks

Guevara Noubir

Textbook: Computer Networks: A Systems Approach,
L. Peterson, B. Davie, Morgan Kaufmann

What is CS4700/CS5700 about?

- Goal:
  - Convey the principles and mechanisms to build a computer network that can scale:
    - Grow to a global proportion
    - Support diverse applications
  - Special attention is given to the Internet protocols and architecture

- How? a combination of
  - Lectures on principles/concepts, mechanisms/algorithms, performance analysis techniques
  - Practical assignments to develop network protocols and applications
  - Conceptual assignment to understand how performance is optimized

Course Outline

- Introduction to internetworking: principles, concepts, architecture, services
- Direct link networks
- Routing
- End to end protocols: e.g., TCP, UDP
- Congestion control/congestion avoidance
- Multicast and overlay routing over the internet
- Wireless and mobile networks
- Internet security
- Applications: www, mail, content distribution networks, p2p
- Queuing theory: dimensioning and performance prediction

Outline of Lecture 1

- Requirements
- Architecture
- Implementation

Read Chapter* 1 *of textbook

Requirements

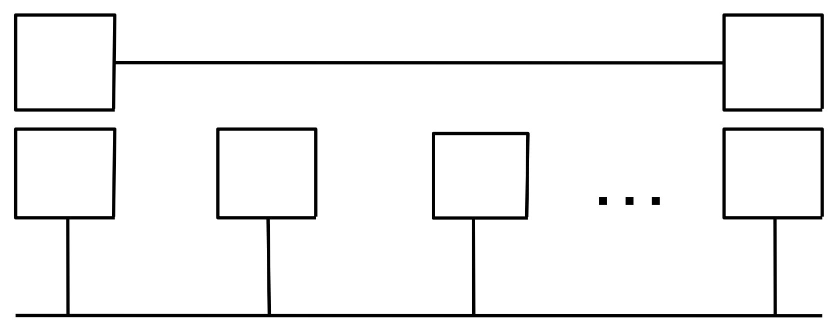
- Computer networks are different from classical networks:
  - General
  - Not optimized for a specific application
- Requirements differ according to the perspective:
  - Application programmer: services
  - Network designer: resource efficiency and fairness
  - Network provider: administration, manageability, accountability, security

Requirements

- Connectivity
- Cost-Effective Resource Sharing
- Support for Common Services
- Performance

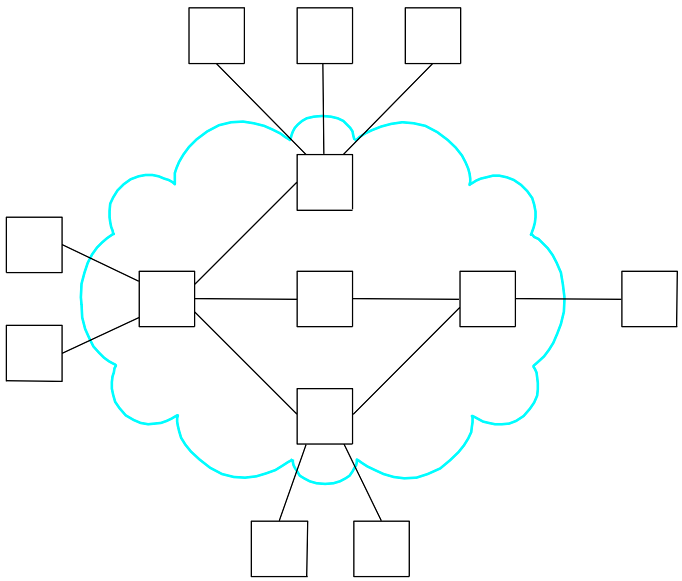
Connectivity

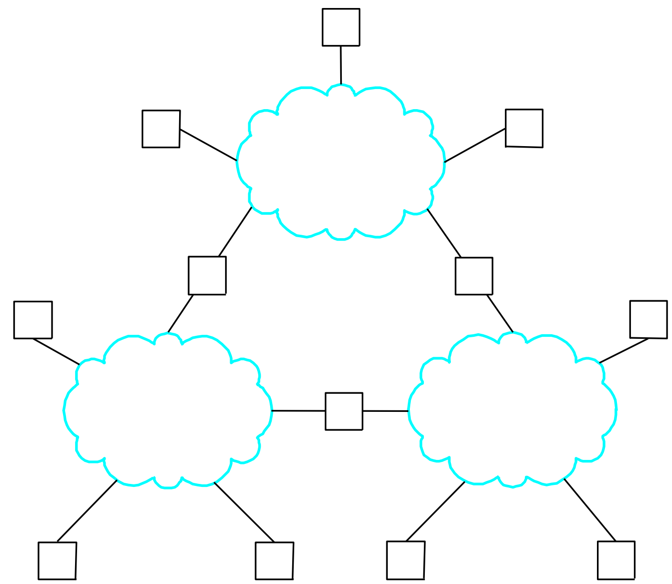
- Goal: allow machines to communicate
  - Exceptions?
- Building blocks:
  - Nodes: PC, workstations, special-purpose hardware…
    - hosts
    - switches
  - Links: coax cable, optical fiber, wireless…
    - point-to-point
    - multiple access
    (generally limited in size)



Connectivity: Switched Networks

- A network can be defined recursively as...
  - two or more nodes connected by links, or
  - two or more networks connected by two or more nodes





Switching Strategies

- Circuit switching: carry bit streams
  - On session establishment a path from source to destination is selected. Resources are allocated over all the links of the path. Route does not change during session life.
  - Links can be shared by different sessions through mechanisms such *time-division multiplexing* (TDM) or *frequency-division multiplexing* (FDM)
  - Guarantees: rate and packets delivery in order.
  - Example: original telephone network
- Packet switching: store-and-forward messages
  - Links are shared on a "*demand basis*" vs. fixed allocation
  - Packets wait in a queue before being transmitted
  - E.g., Internet mainly made out of packet switching

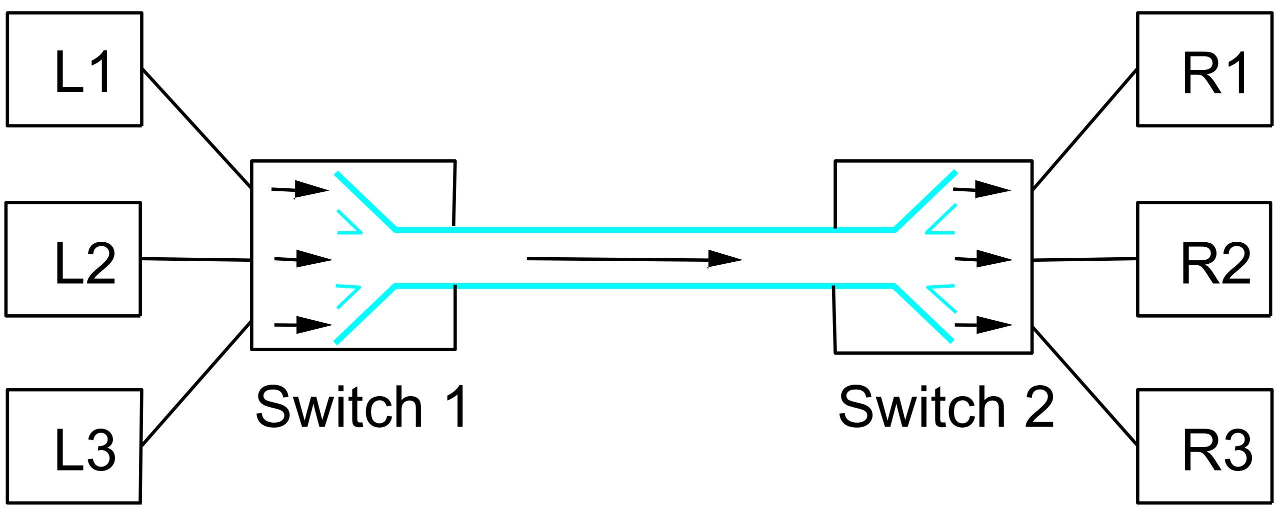
Addressing and Routing

- Address: byte-string that identifies a node
  - usually unique
- Routing: process of forwarding messages to the destination node based on its address
- Types of addresses
  - unicast: node-specific
  - broadcast: all nodes on the network
  - multicast: some subset of nodes on the network

Cost-Effective Resource Sharing

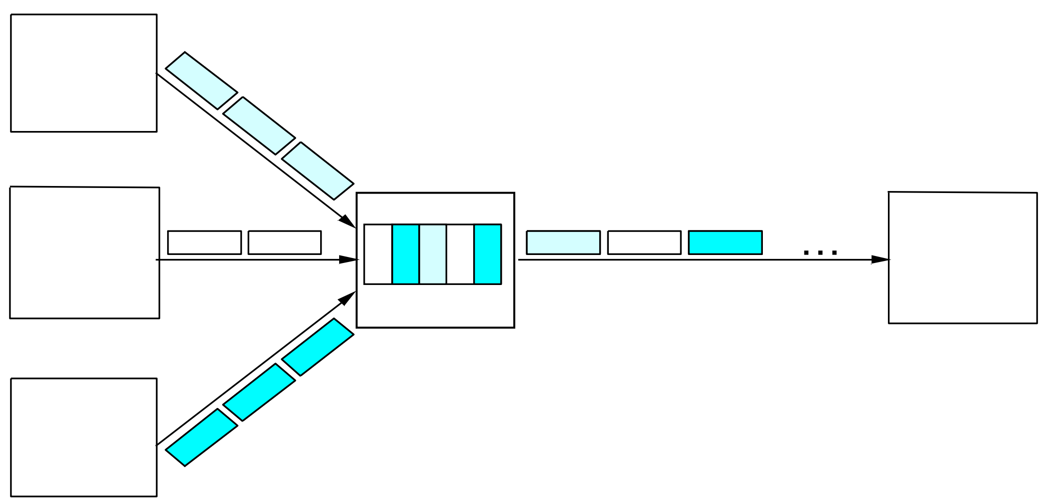
Multiplexing

- Time-Division Multiplexing (TDM)
- Frequency-Division Multiplexing (FDM)



Statistical Multiplexing

- On-demand time-division
- Schedule link on a per-packet basis
- Packets from different sources interleaved on link
- Buffer packets that are *contending* for the link
- Buffer (queue) overflow is called *congestion*



Support for Common Services

- A computer network provides more than packet delivery between nodes
- We don't want application developers to rewrite for each application higher layer networking services
- The channel is a pipe connecting two applications
- How to fill the gap between the underlying network capability and applications requirements?
- Problem: identify a set of common services
  - Delivery guarantees, security, delay

Communication Patterns: Types of Applications

- Interactive terminal and computer sessions:
  - Small packet length, small delay, high reliability
- File transfer:
  - High packet length, high delay, high reliability
- Voice application:
  - Small packet length, small delay, small reliability, high arrival rate
- Video-on-demand:
  - Variable/high packet length, fixed delay, small reliability
- Video-conferencing
  - Variable/high packet length, small delay, small reliability

Basic Channels

- Request/Reply channel:
  - Guarantees single copy message delivery
  - Can provide confidentiality and integrity
  - Used for file transfer and digital library applications
- Message Stream channel:
  - Supports one/two-way traffic, multicast
  - Parameterized for different delays
  - Does not need to guarantee message delivery
  - Guarantees order of delivered messages
  - Used for video-conferencing, video-on-demand

Reliability: What goes wrong in the network?

- Bit-level errors (electrical interference)
- Packet-level errors (congestion)
- Link and node failures

- Messages are delayed
- Messages are deliver out-of-order

- Security:
  - Authentication, confidentiality, integrity, availability,

Performance Metrics

- Bandwidth (throughput)
  - data transmitted per time unit
  - link versus end-to-end
  - notation
    - KB = 2^{10} bytes
    - Mbps = 10^6 bits per second
- Latency (delay)
  - time to send message from point A to point B
  - one-way versus round-trip time (RTT)
  - components

    Latency = Propagation + Transmit + Queue

    Propagation = Distance / c

    Transmit = Size / Bandwidth
  - Examples of RTT: LAN, Cross-country link, Satellite

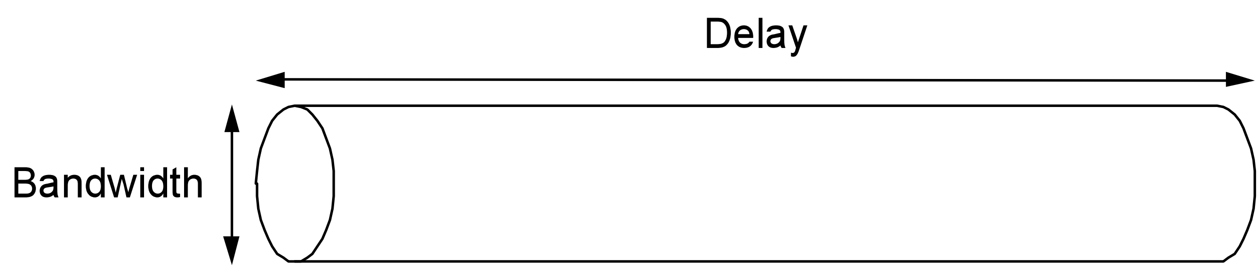
Bandwidth versus Latency

- Relative importance (e.g., fetch application)
  - 1-byte: 1ms vs 100ms (latency) dominates 1Mbps vs 100Mbps (bandwidth)
  - 25MB: 1Mbps vs 100Mbps dominates 1ms vs 100ms

- Infinite bandwidth
  - RTT dominates
    - Throughput = TransferSize / TransferTime
    - TransferTime = RTT + TransferSize /Bandwidth
  - 1-MB *file* to 1-Gbps link as 1-KB *packet* to 1-Mbps link (RTT=100ms)

Delay x Bandwidth Product

- Amount of data "in flight" or "in the pipe"
- Example: 100ms x 45Mbps = 560KB



- Why is it important to know Delay x Bandwidth product?

Network Architecture

- Layering and Protocols
- ISO Architecture
- Internet architecture

Layering

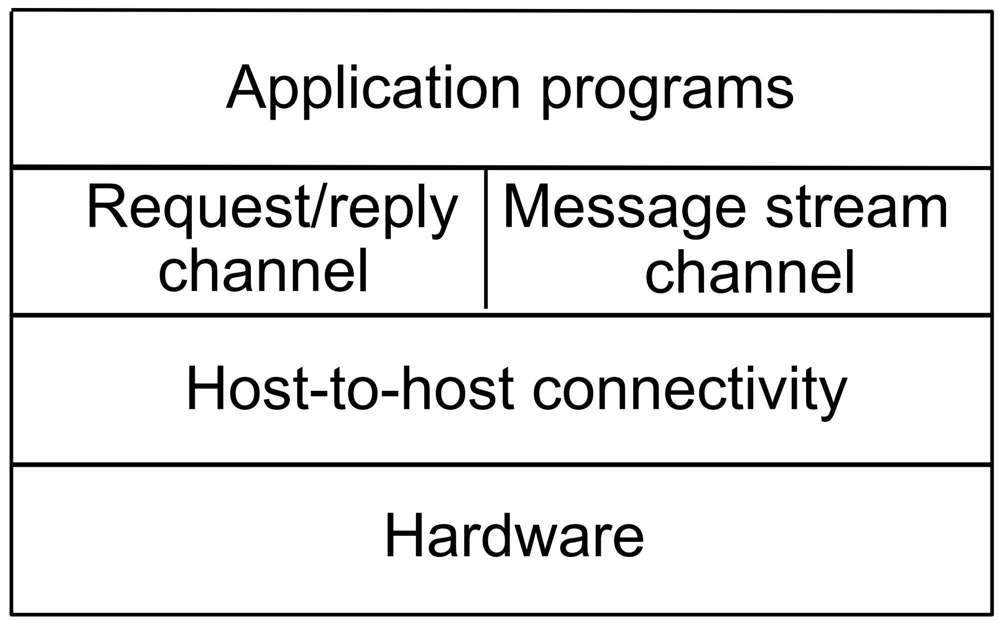
- Use abstractions to hide complexity
- Abstractions naturally lead to layering
- Alternative abstractions at each layer

<u>Advantages</u>:

- Solve small problems vs. monolithic software
- Modularity: easily add new services

<u>Drawback:</u>

- May hide important information



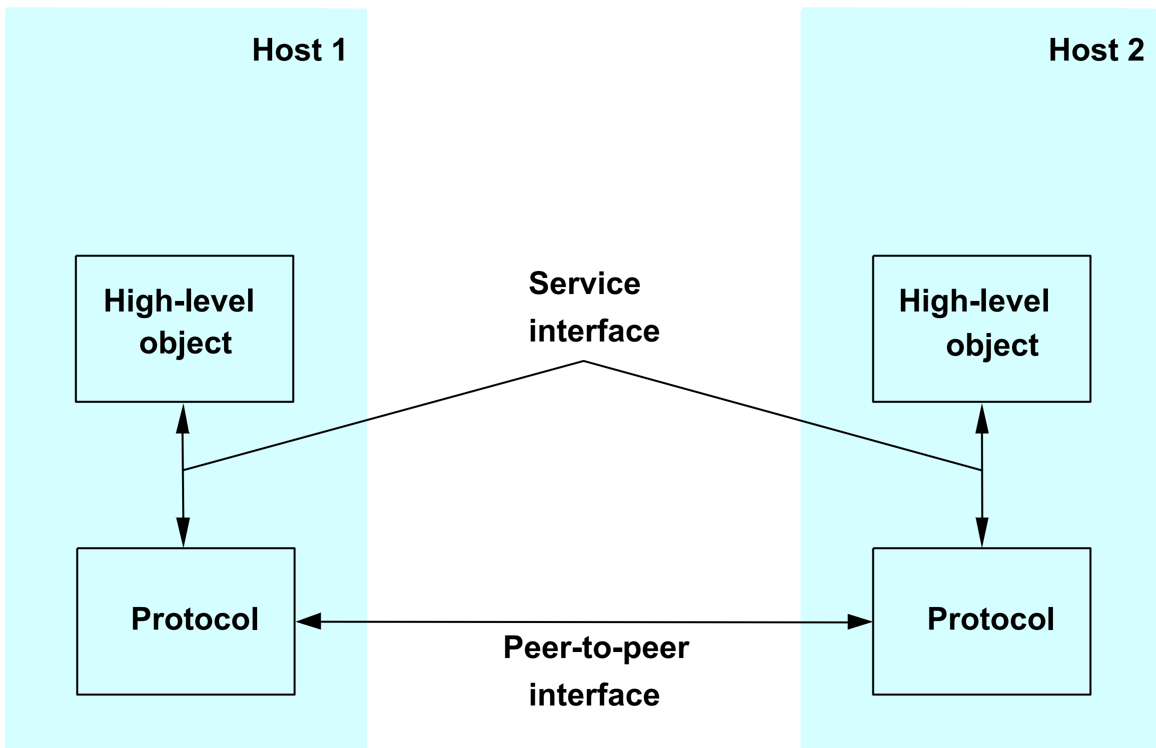
Protocols

- Building blocks of a network architecture
- Term "protocol" is overloaded
  - specification of peer-to-peer interface (rules)
  - module that implements this interface

- Each protocol object has two different interfaces
  - *service interface*: operations on this protocol (SAP)
  - *peer-to-peer interface*: form/meaning of messages exchanged with peer

- Protocol stack: set of consecutive layers
- Interoperability problems

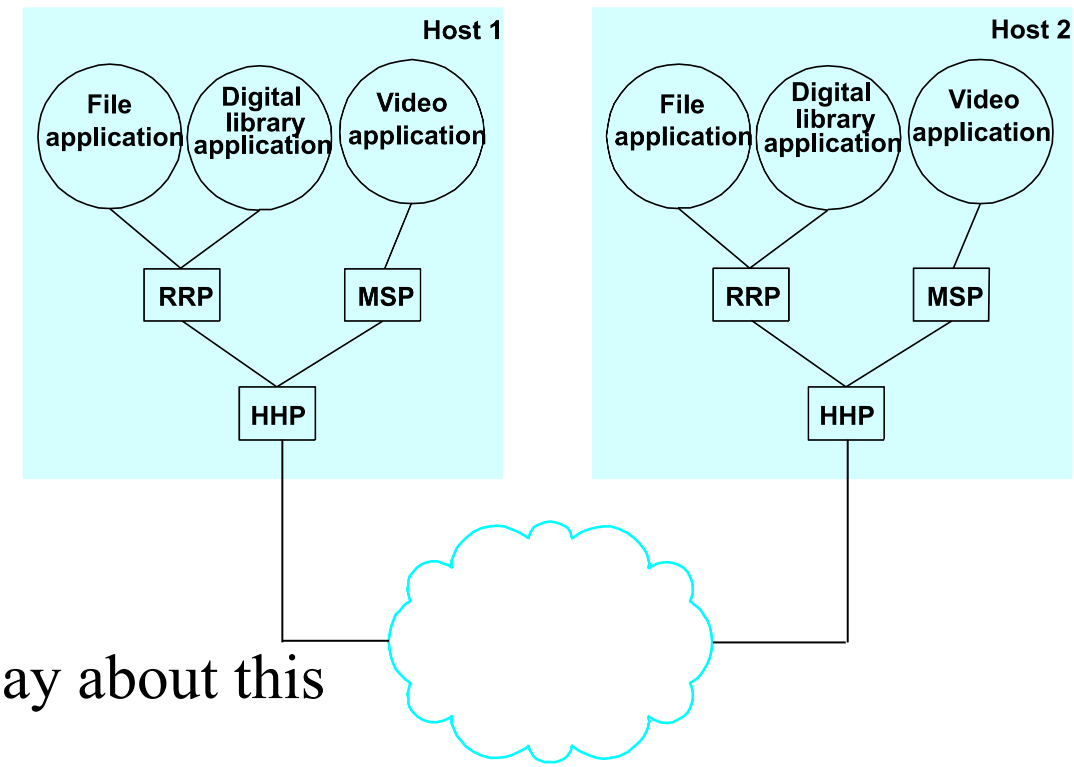
Interfaces



Host 1
Host 2
High-level object
High-level object
Service interface
Protocol
Protocol
Peer-to-peer interface

Protocol Machinery

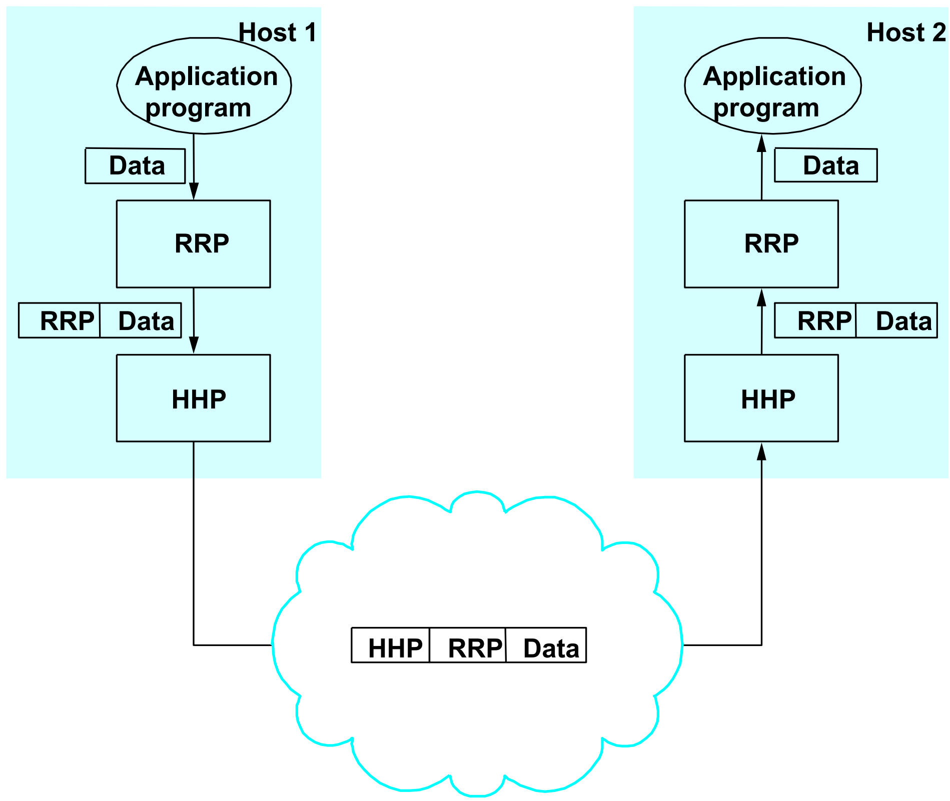
- Protocol Graph
  - most peer-to-peer communication is indirect
  - peer-to-peer is direct only at hardware level



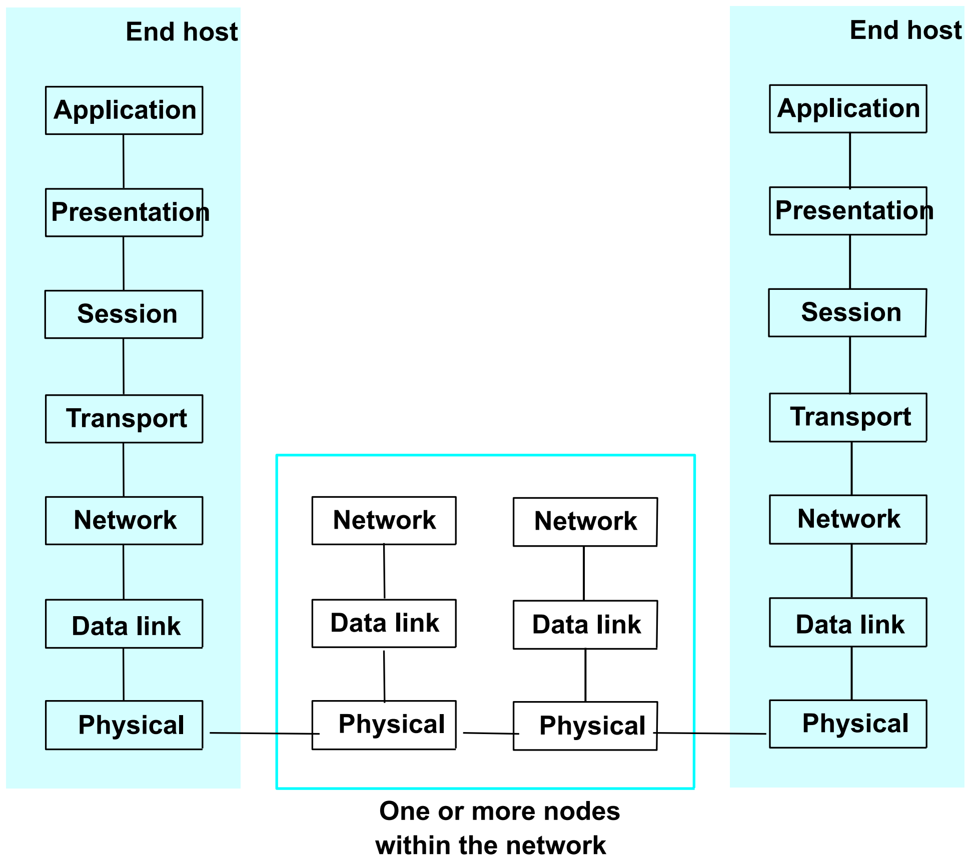
What can we say about this graph?

Machinery (cont)

- Multiplexing and Demultiplexing (demux key)
- Encapsulation (header/body)



Open System Interconnect (OSI) Architecture: Reference Model



Physical Layer

- Function: provides a "*virtual bit pipe*"
- How: maps bits into electrical/electromagnetic signals appropriate for the channel
- The physical layer module is called a *modem* (modulator/demodulator)
- Important issues:
  - Timing: synchronous, intermittent synchronous, asynchronous (characters)
  - Interfacing the physical layer and DLC (e.g., RS-232, Ethernet, IEEE802.x)

Data Link Control Layer (DLC)

- Receives packets from the network layer and transforms them into bits transmitted by the physical layer. ***Generally*** guarantees order and correctness.
- Mechanisms of the DLC:
  - Framing: header, trailer to separate packets, detect errors…
  - Multiple access schemes: when the link is shared by several nodes there is a need for addressing and controlling the access (this entity is called MAC sublayer)
  - Error detection and retransmission (LLC sublayer)

Network Layer

- Provides naming/addressing, routing, flow control, and scheduling/queuing in a multi-hop network
- Makes decisions based on packet header (e.g., destination address) and module stored information (e.g., routing tables)
- General comment: each layer looks only at its corresponding header (here packet header)
- Routing is different on virtual circuit networks than on datagram networks

Transport Layer

- Provides a reliable mean to transmit messages between two end-nodes through:
  - Messages fragmentation into packets
  - Packets reassembly in original order
  - Sessions multiplexing and splitting
  - Retransmission of lost packets
  - End-to-end flow control
  - Congestion control

Session Layer

- Was intended to handle the interaction between two end points in setting up a session:
  - multiple connections
  - Service location (e.g., would achieve load sharing)
  - Checkpointing
  - Control of access rights
- In many networks these functionalities are inexistent or spread over other layers

Presentation Layer

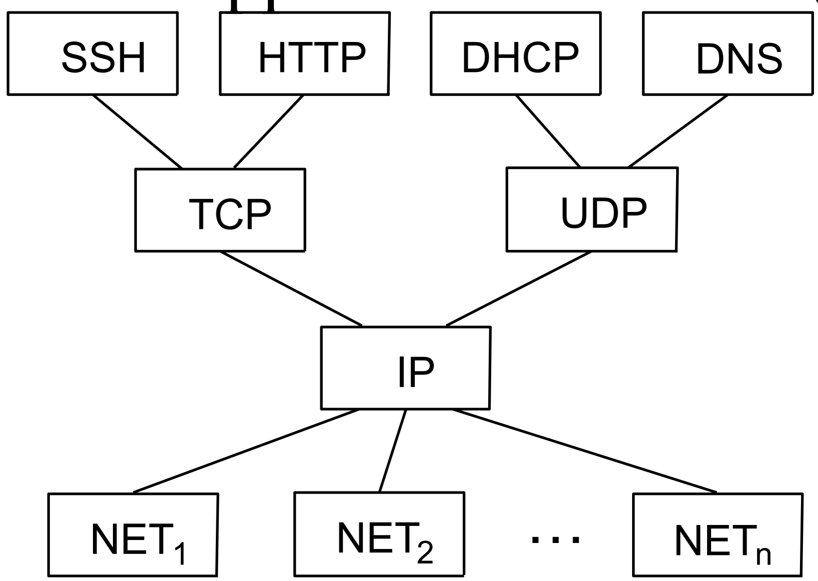
- Provides data encryption, data compression, and code conversion.

Application Layer

- What's left …
- Examples
  - http (web), smtp (mail), telnet, rtp (voip) …

Internet Architecture

- Defined by Internet Engineering Task Force (IETF)
- IETF requires working implementations for standard adoption
- Application vs. Application Protocol (SSH, HTTP)



Implementing Network Software

- Success of the internet is partially due to:
  - Minimal functionality within the network
  - Most of the functionality running as software over general-purpose computers
- Simple Application Programming Interface
- Efficient Protocol implementation

- Today, clean-slate initiatives e.g., GENI

Application Programming Interface

- Each OS can have a special interface exported by the network to the applications developer
- Most widely used network API is: socket interface
  - Initially developed by the Berkeley Distribution of Unix and today ported to almost all OS
  - ```
    int socket(  int domain, /* PF_INET, PF_UNIX */
                 int type, /* SOCK_STREAM, SOCK_DGRAM */
                 int protocol)
    ```
    ```

Client/Server Sockets (TCP)

- Client:
  - socket, connect, (send, recv)*, close
- Server:
  - socket, bind, listen, (accept, (recv, send)*, close)*

```
• int connect (int socket, struct sockaddr *address, int addr_len)
• int send (int socket, char *message, int msg_len, int flags)
• int recv (int socket, char *buffer, int buf_len, int flags)

• int bind (int socket, struct sockaddr *address, int addr_len)
• int listen (int socket, int backlog)
• int accept (int socket, struct sockaddr *address, int *addr_len)
```

Protocol Implementation Issues

- Process model:
  - Process-per-protocol vs. process-per-message
- Efficient message buffers management
  - Avoid copying
  - Encapsulation operations without copying: msgAddHdr, msgStripHdr
- Efficient events management:
  - Scheduling and cancellation

Summary

- The explosive growth of the internet (and computer networks) is mainly due to the easy way of adding new functionalities by running software on affordable general purpose computers
- Internet is a living systems
  - dominance of email, web, P2P, streaming
- Goal of the course is to:
  - understand the principles for designing and optimizing, networks mechanisms, and protocols
  - learn to develop networked applications
- Requirements, Architecture, Implementation