Introduction

What is a computer network?

Components of a computer network:

- hosts (nodes, computers)
- links
- protocols
- applications (network services)
- humans and service agents

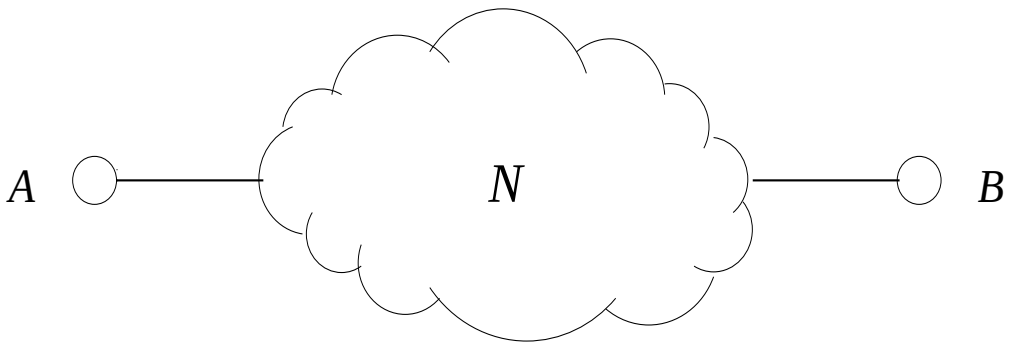
Hosts & links form the *hardware* side.

Protocols & applications form the *software* side.

Protocols can be viewed as the "glue" that binds everything else together.

Simplest instance of networking problem:

> Given two hosts A, B interconnected by some network N, facilitate communication between A & B.



Information abstraction

- objects (e.g., files)
- bytes & bits
- signals over physical media (e.g., electromagnetic waves)

Minimal functionality required of A, B

- encoding
- decoding

$\longrightarrow$ a form of translation

Additional functionalities may be required depending on properties of network N

- information corruption
- information loss
- information delay (to wait or not to wait)
- information security

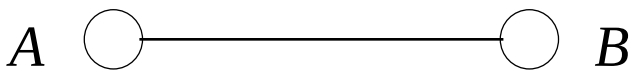
Network N can be

- point-to-point links
- multi-access links
- internetworks

  $\longrightarrow$ logical topology point-of-view

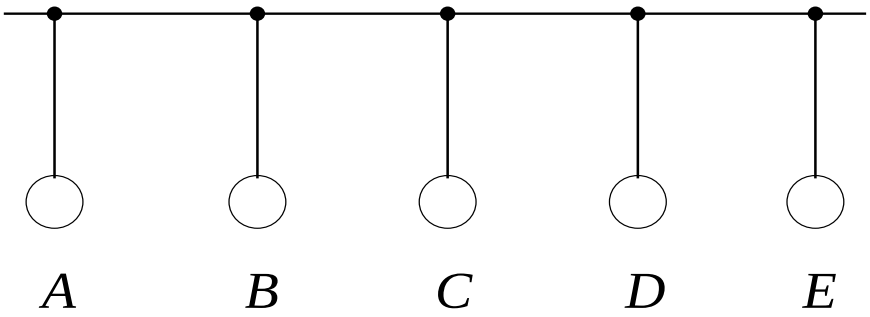
  $\longrightarrow$ may differ from physical topology

Point-to-point links



- various "cables"
- line of sight wireless communication
- no addressing necessary

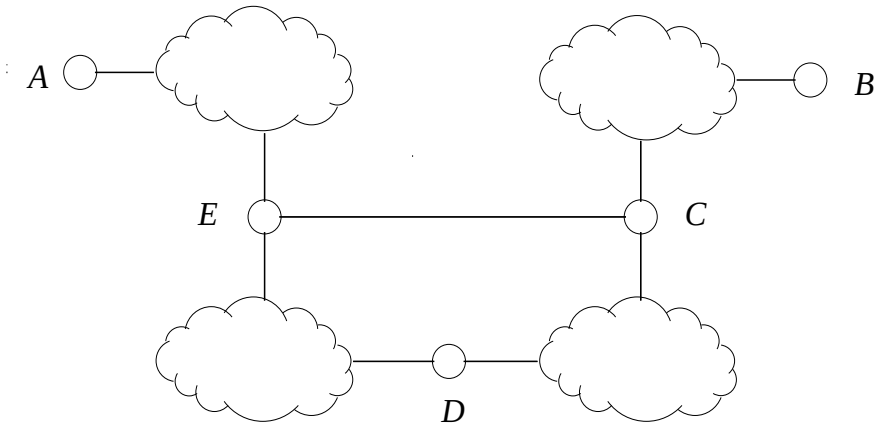
Multi-access links



- "buses," wireless media
- access control (bus arbitration)

  $\rightarrow$ resolve contention due to interference effects
- addressing necessary

Internetwork



- recursive definition
- addressing necessary
- path selection (routing)
- protocol translation (internetworking)

LAN/WAN distinction

- LAN: point-to-point, multi-access
- WAN: internetwork

  $\longrightarrow$ geographical distinction is secondary

Myriad of different local area network (LAN) technologies co-existing in a wide area network (WAN).

> e.g.: Ethernet, token bus, FDDI (Fiber Distributed Data Interface), packet radio, etc.

Thus, a network (internetwork), at the base level, is a collection of LANs connected together.

Each LAN, in general, speaks a different language (e.g., message formats etc.). Internetworking solves this problem by . . .

Notice that communicating entities are *processes* residing on A and B.

Thus an *address* must also identify which process a message is destined for (e.g., UNIX).

Fault-tolerance issues.

> the larger the network, the more things can go wrong.
>
> e.g.: node/link failures, message corruption, lost messages, outdated messages.
>
> probability of component failure is at least proportional to system size ($p \cdot n$ where p is prob. of single component failure and n is system size)
>
> oftentimes, it's superlinear

- issue of reliable communication
- reliable network services

  $\longrightarrow$ main principle: redundancy

Two examples:

- routing of messages—alternate routes
- domain name servers—duplication

Network security:

- denial of service (DoS) attack, virus, etc.
- along with fault-tolerance impacts QoS
- trade-off with overhead

Performance issues.

- excessive traffic can cause congestion (analogous to highways)

  $\longrightarrow$ differences?

- traffic volume exhibits large fluctuations
- multimedia traffic is single-source voluminous
- ubiquitous access (e.g., Internet)

  $\longrightarrow$ potential for bottleneck development.

Different kinds of applications or traffic require different levels of services (fast, slow, etc.).

$\longrightarrow$ how to achieve quality of service (QoS)

Network performance

Three yardsticks (performance measures):

$\longrightarrow$ throughput, delay (latency), jitter (variance).

Bandwidth vs. throughput:

bandwidth—maximum data transmission rate achievable at the hardware level; determined by signalling rate of physical link and network interface card (NIC).

throughput (*effective*)—maximum data transmission rate achievable at the software level; overhead of network protocols is accounted for.

actually, even more than that; effect of resource limitations (e.g., mismatch) is also taken into account.

$\longrightarrow$ "true" measure of communication speed.

Trend on protocol overhead side:

migration of protocol software functionality into NICs; NIC is becoming a powerful, semi-autonomous device.

spurned, in part, by speed, compression, and encryption requirements.

Meaning of "high-speed" networks:

- signal propagation speed is bounded by SOL (speed-of-light)
- can only increase bandwidth (analogous to widening highway)

$\longrightarrow$ increase in *delay-bandwidth product*

Units:

Gbps (Gb/s), Mbps (Mb/s), kbps (kb/s)

$\longrightarrow$ 10^6 bits per second, 10^3 bits per second; indicates data transmission rate; influenced by clock rate (MHz) of signalling hardware.

e.g.: 10 Mbps (10BaseT), 100 Mbps (FDDI), 64kb/s (digitized voice), 144kb/s (ISDN line 2B + D service), 1.544 Mbps (T1), 44.736 Mbps (T3), 155.52 Mbps (OC-3), 622.08 Mbps (OC-12), OC-24, etc.

GB, MB, kB

$\longrightarrow$ 2^{20} bytes, 2^{10} bytes; indicates size of data being shipped; influenced by the memory structure of computer.

e.g.: 512 B, 1 kB (TCP segment size), 64 kB (maximum IP packet size), 53 B (ATM cell), 810 B (SONET frame), etc.

Packet, frame, cell, datagram, message, etc.

$\longrightarrow$ *packet* most generic term

Conventional usage

- frame: LAN-level
- datagram: IP packets
- cell: ATM packets
- packet: generic
- message: high-level (e.g., e-mail)

Characteristics of message delay:

Given a message M of size s, its *delay* (or *latency*), $d(M)$, is given by

$$d(M) = D_P + D_T(M) + D_Q(M, \mathcal{S})$$

where D_P is the propagation time (link latency), $D_T(M)$ is the transmission time, $D_Q(M, \mathcal{S})$ is the queueing delay, and $\mathcal{S}$ represents the network state.

Individually,

$$D_P \propto L/c$$

where L is the total distance from source to destination and c is the speed of light.

$\longrightarrow$ D_P is independent of M.

Trivial lower bound: $d(M) > D_P$. (E.g., satellite communication, coast-to-coast.)

For transmission time,

$$D_T(M) \propto s/BW$$

where BW is the link bandwidth.

Queueing delay, $D_Q(M, \mathcal{S})$, generally, does not have a closed-form expression, being a complicated function of $\mathcal{S}$. Sometimes, $\mathcal{S}$ represents the level of congestion.

$\longrightarrow$ however, it is the most important factor.

$*$ *Implicit in the previous formulae: infinite (sufficiently large) buffer size. Why?*

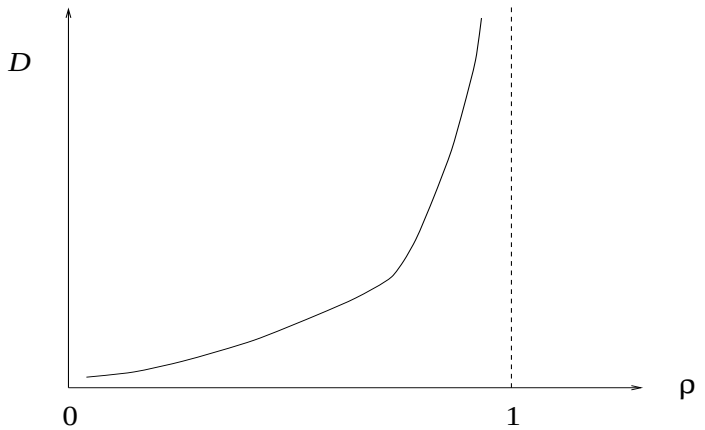
Throughput vs. delay relationship.

Important lesson from queueing theory:

> Other things being equal, if traffic flow is *stochastic* (with certain properties), then there exists a trade-off relationship between throughput and delay.

Definition of *utilization* ρ (for now):

$$\rho = \text{throughput}/BW.$$



Why is this relationship important?

> *Multimedia data* (audio/voice, video/image, real-time interactive data) need to meet certain quality of service (QoS) requirements to be useful to the user.
>
> $\longrightarrow$ small delay & high thoughput is desirable.
>
> Trade-off relationship points toward *fundamental* limitations to achieving both objectives at the same time.

Other performance measures:

- packet (cell) loss rate
- delay variation—"jitter"
- throughput variation—"burstiness"
- power (= throughput / delay)

What is traveling on the wires?

Mixed data:

$\longrightarrow$ bulk data, audio/voice, video/image, real-time interactive data, etc.

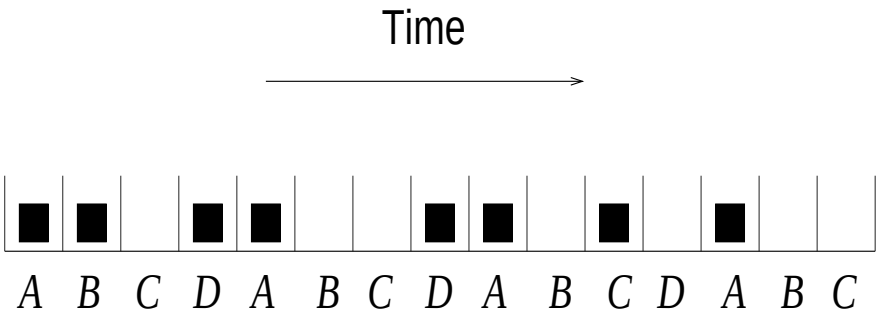
Tilting more and more every day toward *multimedia* data; i.e., traffic with real-time constraints.

Data networks capable of carrying all types of data at the same time is a *new* phenomenon.

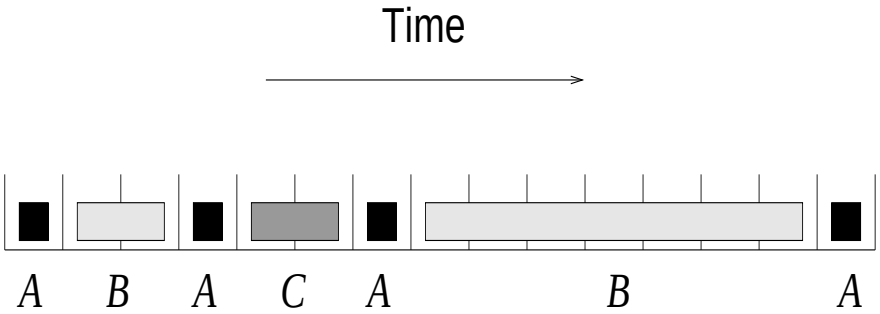
Even today, *voice traffic* (telephone) is carried on an entirely separate communication network vis-à-vis *data traffic*, operating under different internetworking principles from the latter.

$\longrightarrow$ Time-division multiplexing (TDM) for telephony vs. packet switching for data networks.

Time-division multiplexing:



Packet/circuit switching:

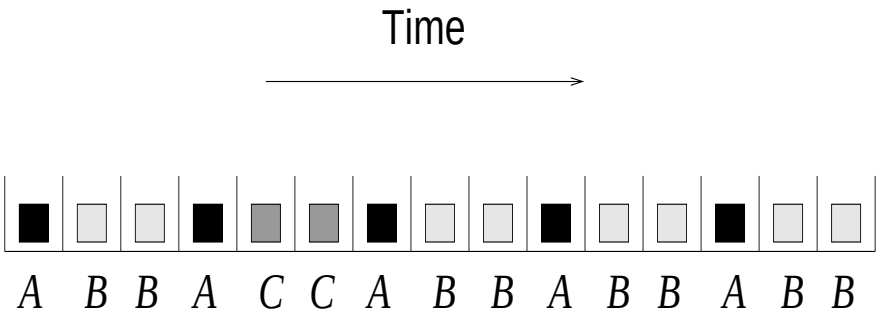


Goal: would like to merge to increase functionality, system utilization and efficiency.

Immediate impediment:

$\longrightarrow$ "Bully phenomenon." What to do?

Asynchronous transfer mode (ATM) :



$\longrightarrow$ 53 byte packet or *cell*.

Synergy of all forms of data, audio, video, bulk, etc. One unified network with "integrated" services.

$\longrightarrow$ solves bully problem but ...

Still many problems

- overhead (48 + 5)
- function duplication (e.g., IP over ATM)
- switching speed and programmability
- circuit switching approach (telephony influence)
- conflict resolution and QoS provision
  - how to guarantee QoS
  - who gets what/how much (fairness)
  - efficiency

$\longrightarrow$ *performance* question.

$\longrightarrow$ significant deployment base, but on the way out?

How to make sense of all this?

Study of networks can be divided into three aspects:

- architecture
- algorithms
- implementation

Architecture

- hardware
  - communication or data link technology (e.g., Ethernet, SONET, wireless)
  - hardware interface standards (e.g., EIA RS 232C—serial communication between DTE and DCE)
- software
  - conceptual organization (e.g., ISO/OSI 7 layer reference model, ATM network model)
  - protocol standards (e.g., IAB RFC—TCP, UDP, IP, Mobile IP; ISO MPEG)

$\longrightarrow$ the *what* over *how*

Provides the "shell" or "skeleton" to everything else.

Hardware/software distinction is not always clearcut.

... speaking of *standards*,

- ISO (International Standards Organization). ISO/OSI 7-layer reference model.
- ANSI (American National Standards Institute). U.S. Government representative at ISO.
- CCITT (International Telegraph and Telephone Consultative Committee). U.N.-chartered organization, principal international telecommunication standards organization.
- ITU (International Telecommunications Union). Successor of CCITT (used to be parent organization), U.N.-chartered.
- ATM Forum. Industry organization.
- IEEE. Professional society, LAN standards; e.g., IEEE 802.3 (Ethernet), IEEE 802.5 (token ring).
- IAB (Internet Architecture Board). Informal committee reporting to DoD, NSF.

- IETF (Internet Engineering Task Force). Subsidiary of IAB; main concern—short-term issues.
- IERF (Internet Research Task Force). IAB subsidiary; long-term issues.
- Internet Society. Elected trustees appoint IAB members.

Algorithms

- error detection and correction (e.g., checksum, CRC)
- medium access control (e.g., CSMA/CD, token ring)
- routing (e.g., shortest paths—Dijkstra, Bellman & Ford)
- congestion control or flow control (e.g., TCP window control and RTT estimation)
- traffic shaping (e.g., leaky bucket) and admission control
- queueing discipline (e.g., FCFS, weighted fair queueing)

Most important component; the *how* aspect of computer networks.

Directly impacts network performance by controlling the underlying resources provided by the network architecture.

Implementations

Same algorithm or protocol is implemented, i.e., *coded*, in several ways (e.g., TCP from different vendors).

$\longrightarrow$ network programming

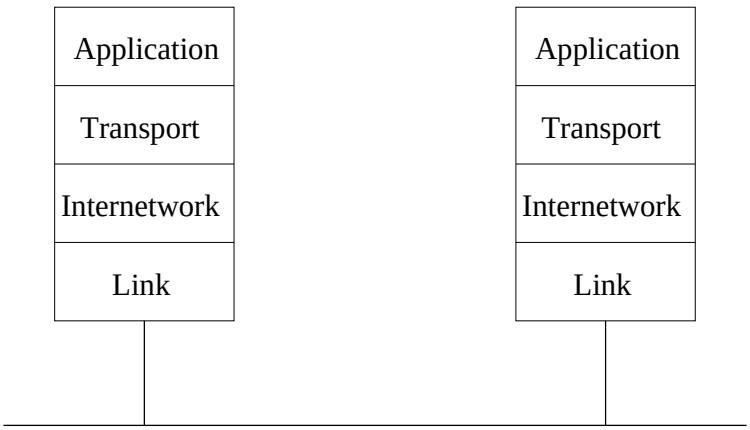
Main issue: *efficiency*. In particular, speed.

- reduce copying operation.
- header prediction.
- employ multi-threading to reduce context-switch time overhead; horizontal vs. vertical organization.
- employ multi-threading to hide communication latency.
- O.S.-support issues.

Although at times ugly, a *must* to squeeze the best out of performance.

Layering

Most fundamental organization of the software aspect of computer networks.



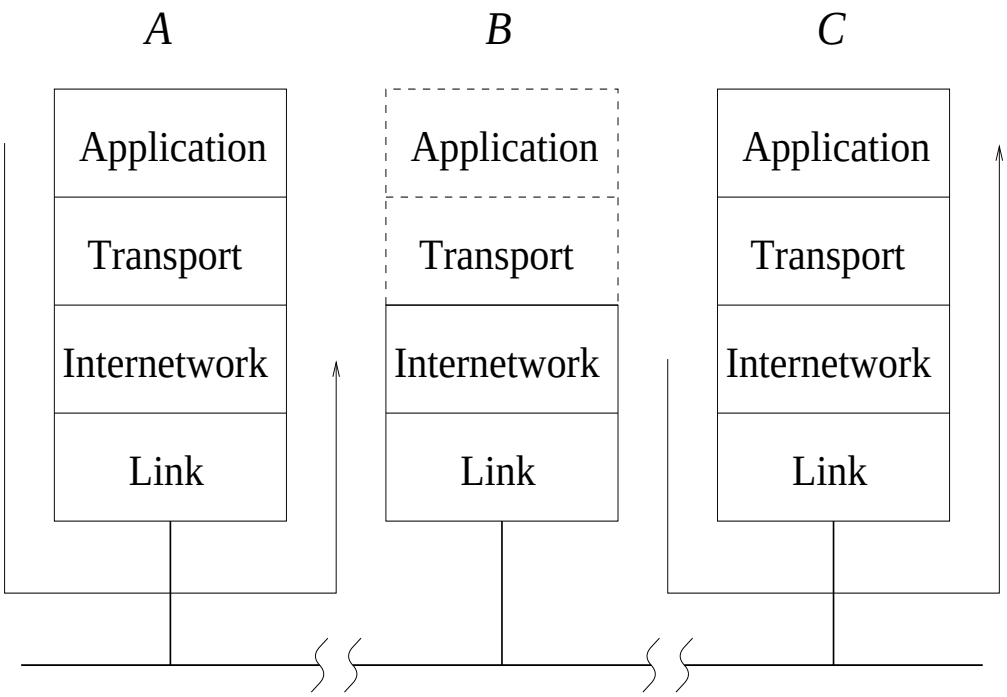
Achieves abstraction, modularization; two types of interfaces:

- peer-to-peer communication
- inter-layer communication (interface between layers)
  - SAP (service access point)
  - PDU (protocol data unit)

Functionalities:

- application layer—ftp, web browsers, etc.
- transport layer—end-to-end communication control (e.g., reliability, congestion control)
- internetwork layer—point-to-point control (e.g., routing, flow control)
- link layer—LAN access control

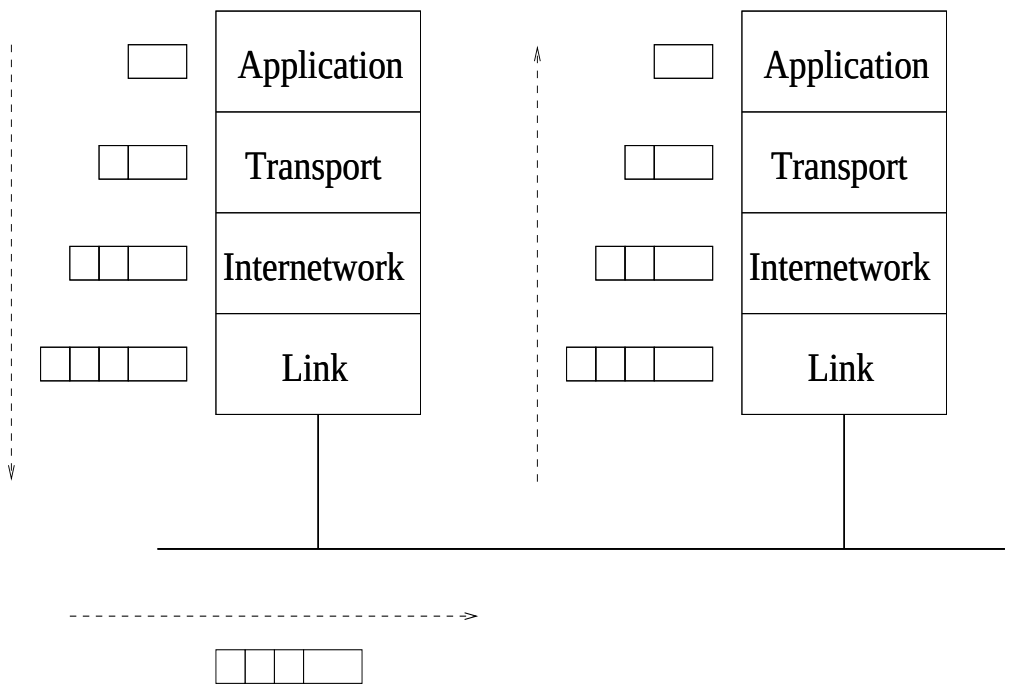
Internetworking example:



Non-peer-to-peer

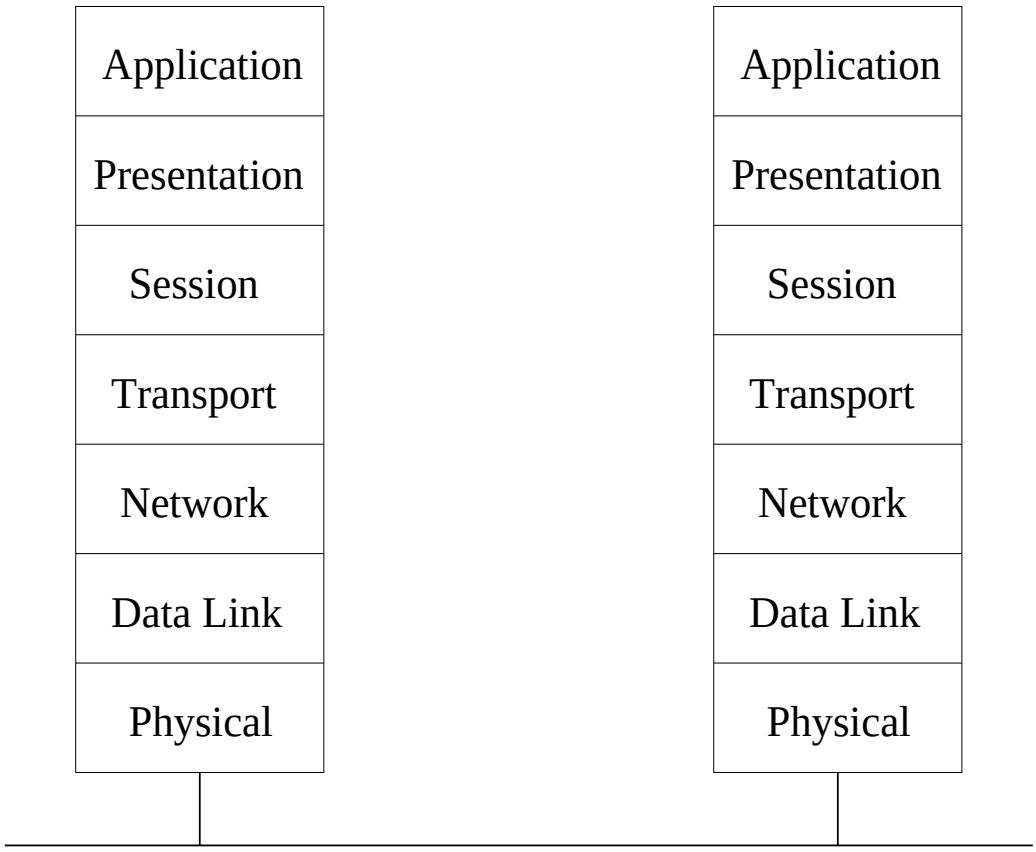
- application layers of A & B, B & C
- transport layers of A & B, B & C
- what about link layers between A & B?

Encapsulation



- protocol stack (push/pop)
- header/trailer overhead
- segmentation/reassembly
- error detection etc.

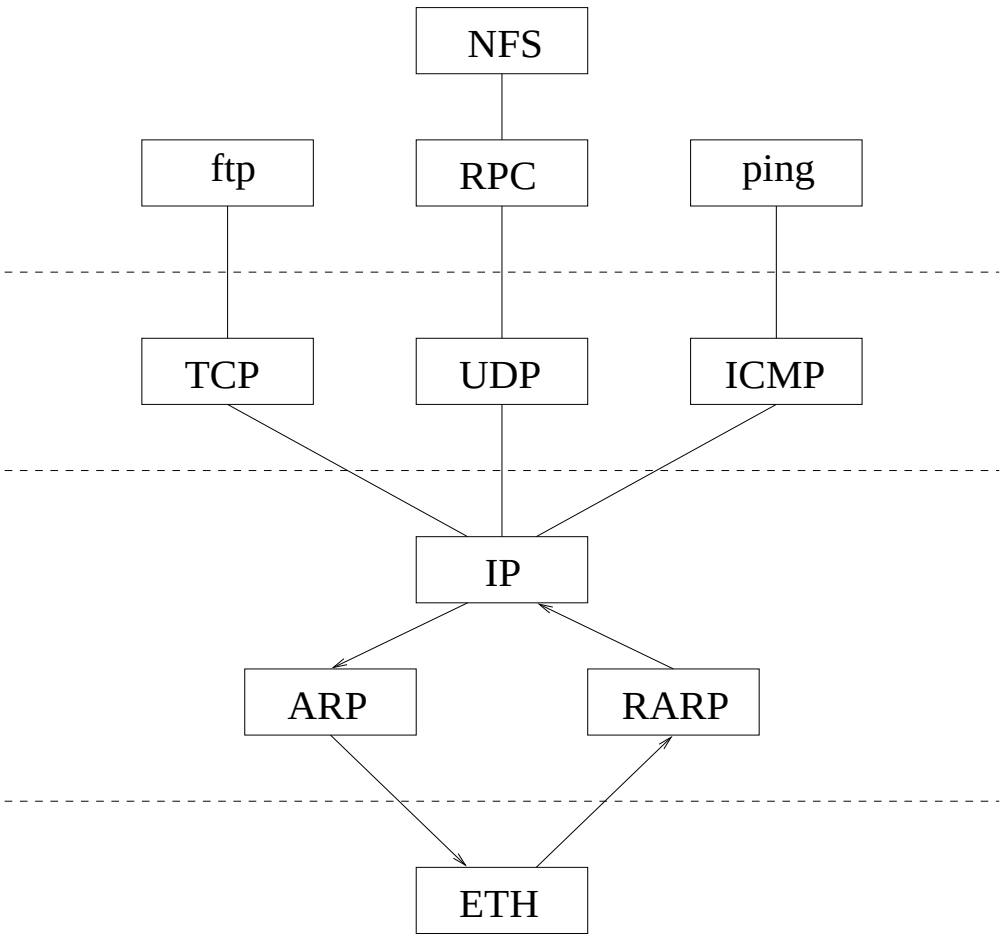
ISO/OSI 7-layer reference model



Outdated; however, still useful as logical reference point.

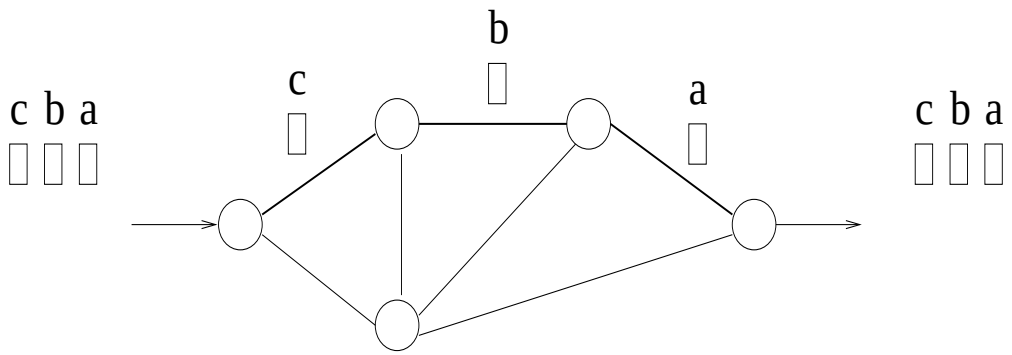
Protocol graph

Shows logical relationship between protocol modules in the protocol stack.



Circuit switching vs. packet switching

Circuit switching—virtual channel is established and followed during the duration of an end-to-end conversation or connection.

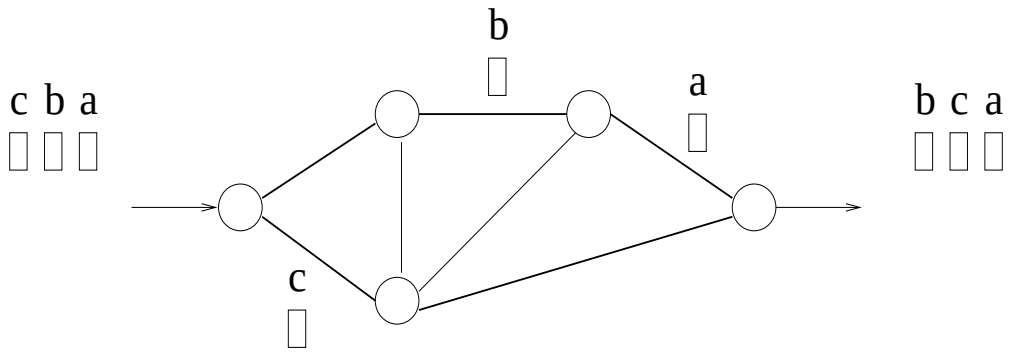


$\longrightarrow$ static route

$\longrightarrow$ in-order delivery

Telephone networks and ATM networks.

Packet switching—every packet belonging to an end-to-end conversation or connection is an independent entity and may take a different route from other packets in the same conversation.



$\longrightarrow$ dynamic route

$\longrightarrow$ out-of-order delivery

Trade-off between processing overhead and route overhead; if route is fixed, the probability of the virtual circuit becoming an inefficient route increases with duration of conversation.

Three modes of designating end point of communication:

- point-to-point
- multicast
- broadcast

$\longrightarrow$ in hardware or software

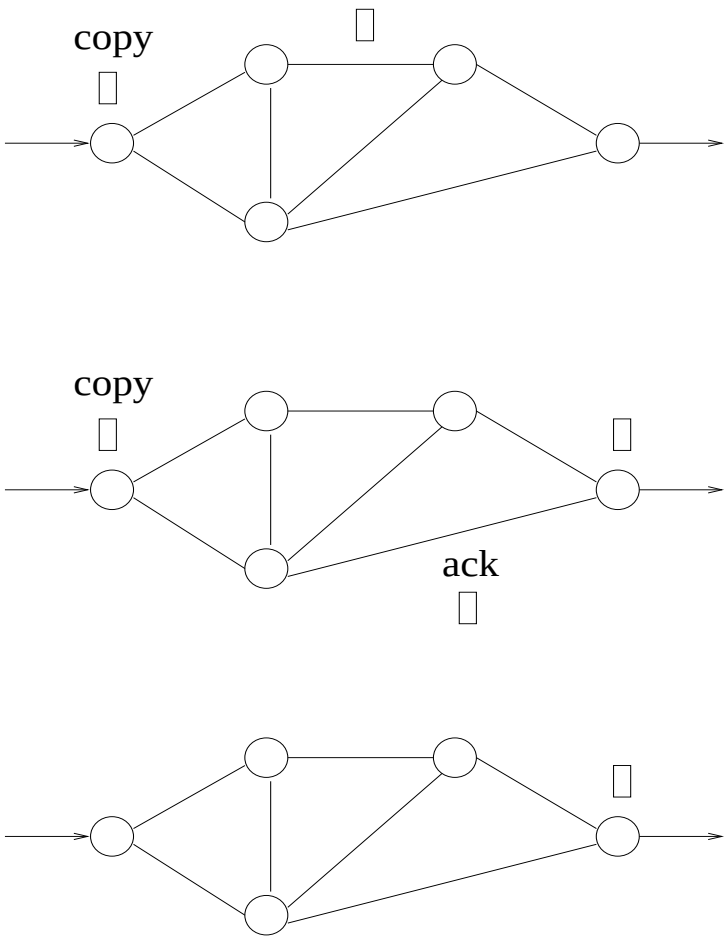
Reliable vs. unreliable communication

Packets may get

- corrupted due to errors (e.g., noise)
- dropped due to buffer overflow
- dropped due to aging or outdatedness—TTL (time-to-live field in IP)
- lost due to link or host failures

Internet philosophy: reliable transport (TCP) over unreliable internetwork (IP). Use retransmission and acknowledgment (ack).

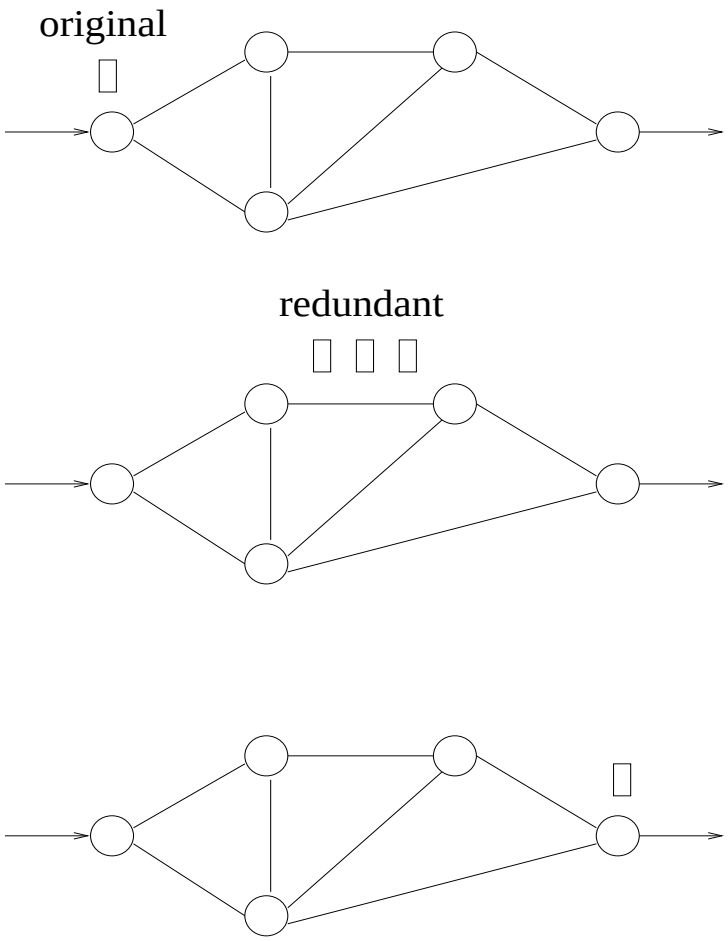
$\longrightarrow$ Automatic Repeat reQuest (ARQ)



- acknowledge receipt (positive ACK)
- absence of ACK indicates probable loss

. . . or vice versa (negative ACK scheme); when to use which.

Forward error-correction (FEC):



. . . works if at most two out of every three packets get dropped.

- send redundant information
- need to know properties of loss probability
- appropriate for real-time contrained data

  $\longrightarrow$ FEC vs. "BEC" (backward error-correction)

Main drawback vis-à-vis ARQ?

Reliability can also be implemented at the data link layer.

Trade-off between

- link-level overhead (fixed) and end-to-end reliability overhead (variable)
- simple vs. complex switch design

Still on-going debate.

More generally, design philosophy of where to impart functionality.

$\longrightarrow$ link vs. end-to-end control

Sharing of resources

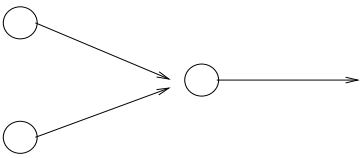
Networks can be viewed as the ultimate platform for sharing *resources*

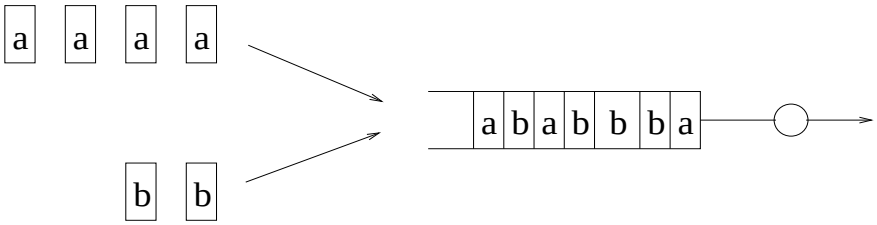
- software resources.
  - passive information—images, sound, text, etc.
  - active information—programs, network computing (e.g., Java)
- hardware resources.
  - system resources—printers, disk, CPU, tape, etc.
  - network resources—bandwidth, buffer space

Too much sharing: *resource contention*.

Resolution and prevention of resource contention is primary task of software layer.

Queueing and buffer overflow





Queueing or scheduling policy

- dequeueing policy (e.g., FCFS)
- enqueueing policy (reshuffling, who-to-drop, overwrite)

Based on the notion of *priorities* (e.g., UNIX scheduling of processes).

Determines

- queueing delay at switch (dwell period)
- packet loss rate at switch

IP has TOS (type-of-service) but not used; IPv6 (Next Generation IP) actively implements TOS.

One way to deal with quality of service issue

$\longrightarrow$ differentiated treatment of packets

Quality-of-service (QoS) is one of the main issues in ATM networks.

Resource allocation problem

Various forms of optimization problems

- network flow problem (e.g., routing and multi-commodity flow, routing and shortest path)
- queueing theory and dynamical systems (e.g., congestion control, QoS)
- game theory (e.g., QoS provision in selfish environments)
- scheduling theory

Issues: seek optimal *operating point* of the system while maintaining stability.

Stability is a big issue (e.g., routing, congestion control)

Questions to ask

- what are the assumptions
- are there intrinsic limitations (e.g., NP-hard)
- optimal/approximate/heuristic solution methods
- how fast
- centralized/decentralized
- how robust or fault-tolerant
- implementation efficiency