Network performance

An overview of key concepts.

Three yardsticks or performance measures:

- throughput: bps or b/s (bits-per-second)
- latency: msec, ms (millisecond)

  $\rightarrow$ signal propagation speed
- delay: msec

  $\rightarrow$ includes software processing overhead
- jitter: delay variation

  $\rightarrow$ standard deviation etc.

Bandwidth vs. throughput:

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

throughput—maximum data transmission rate achievable at the software level; overhead of network protocols inside OS is accounted for.

reliable throughput—maximum reliable data transmission rate achievable at the software level; effect of recovery from transmission errors and packet loss accounted for.

$\longrightarrow$ true measure of communication speed

$\longrightarrow$ "goodput" or "effective throughput"

$\longrightarrow$ vs. "raw throughput"

Trend on protocol implementation and overhead side:

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

network processors: programmable NICs

$\longrightarrow$ as opposed to ASIC based devices

$\longrightarrow$ trade-off between hardware & software

$\longrightarrow$ boundary between hardware & software blurred

With proliferation of wireless networks, lower layers have become important in network programming and system design

$\longrightarrow$ e.g., programming iPAQ with WLAN card

$\longrightarrow$ 802.11b WLAN: 11, 5.5, 2 and 1 Mbps

Meaning of "high-speed" networks:

- signal propagation speed is bounded by SOL (speed-of-light)

  → ∼300K km/s or ∼186K miles/s

  → optical fiber, copper: nearly same

  → latency: Purdue to West Coast

  → around 2000 miles: ∼10 msec (= 2000/186000)

  → lower bound

  → geostationary satellites: ∼22.2K miles/s

  → latency: ∼120 msec

  → end-to-end (one-way): ∼240 msec

  → round-trip: ∼480 msec

  → typically: ∼500 msec

- thus: can only increase “bandwidth”

  → analogous to widening highway, i.e., more lanes

  → simulatenous transmission

  → a single bit does not travel faster

  → “high-speed” $\Leftrightarrow$ “many lanes”

  → completion time of large files faster

  → in this sense, “higher” speed

  → more accurate term: broadband networks

Key issue with broadband/high-speed networks:

$\longrightarrow$ fat (broadband) and long pipes (coast-to-coast)

$\longrightarrow$ a lot of traffic in transit

$\longrightarrow$ total transit traffic: length $\times$ width

$\longrightarrow$ length $\mapsto$ delay and width $\mapsto$ bandwidth

$\longrightarrow$ called delay-bandwidth product

$\longrightarrow$ packet in transit: not under control of sender

$\longrightarrow$ significant damage before detection & recovery

$\longrightarrow$ reactive cost

$\longrightarrow$ limitation of feedback controls (e.g., TCP)

Some units:

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

10^9, 10^6, 10^3 bits per second; indicates data transmission rate; influenced by clock rate (MHz/GHz) of signaling hardware; soon Tbps.

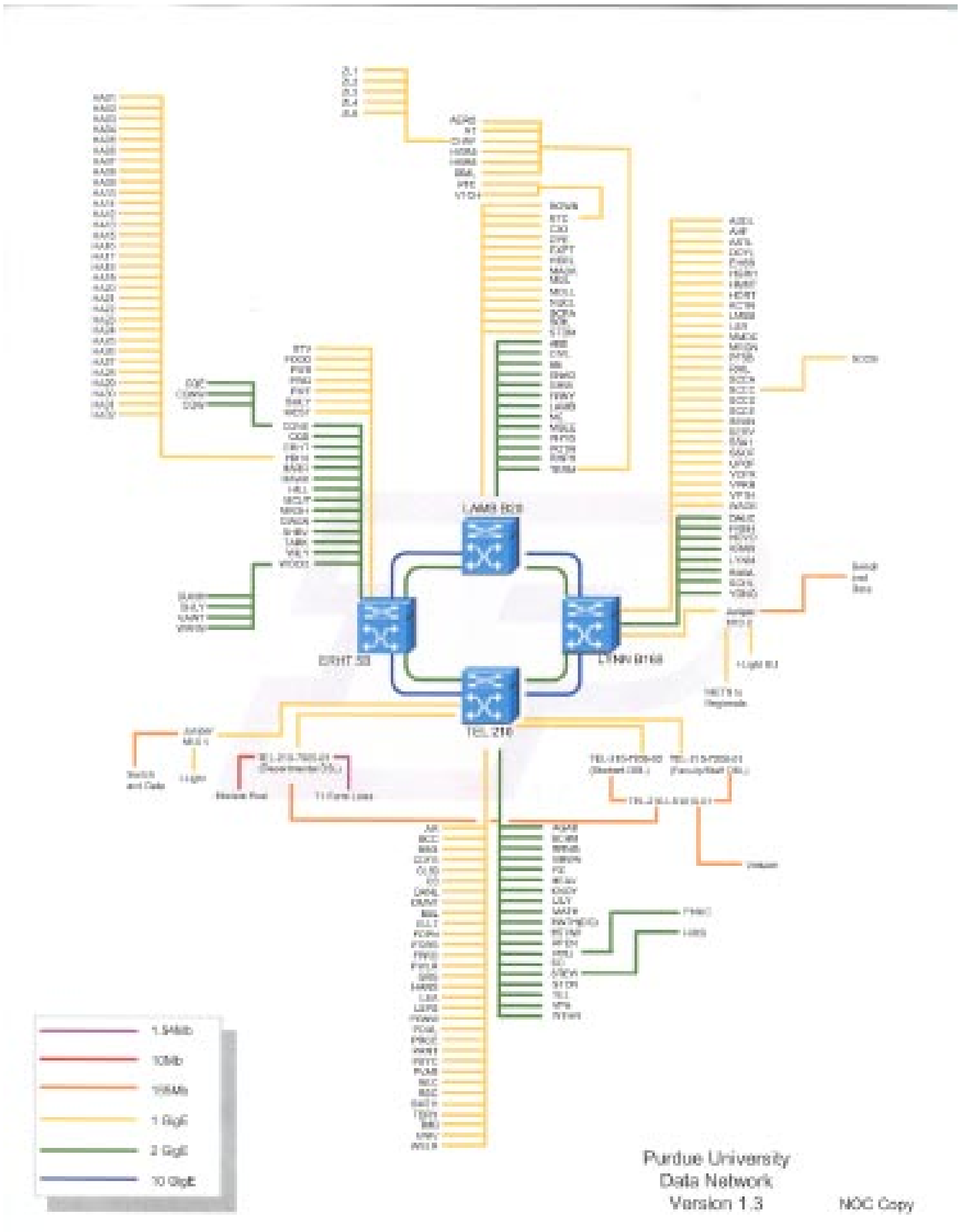
$\longrightarrow$ communication rate: factors of 1000

Common bit rates:

- 10 Mbps (10BaseT), 100 Mbps (100BaseT), 1000 Mbps (1000BaseT)
- 11 Mbps (and 5, 2, 1 Mbps) for 802.11b WLAN

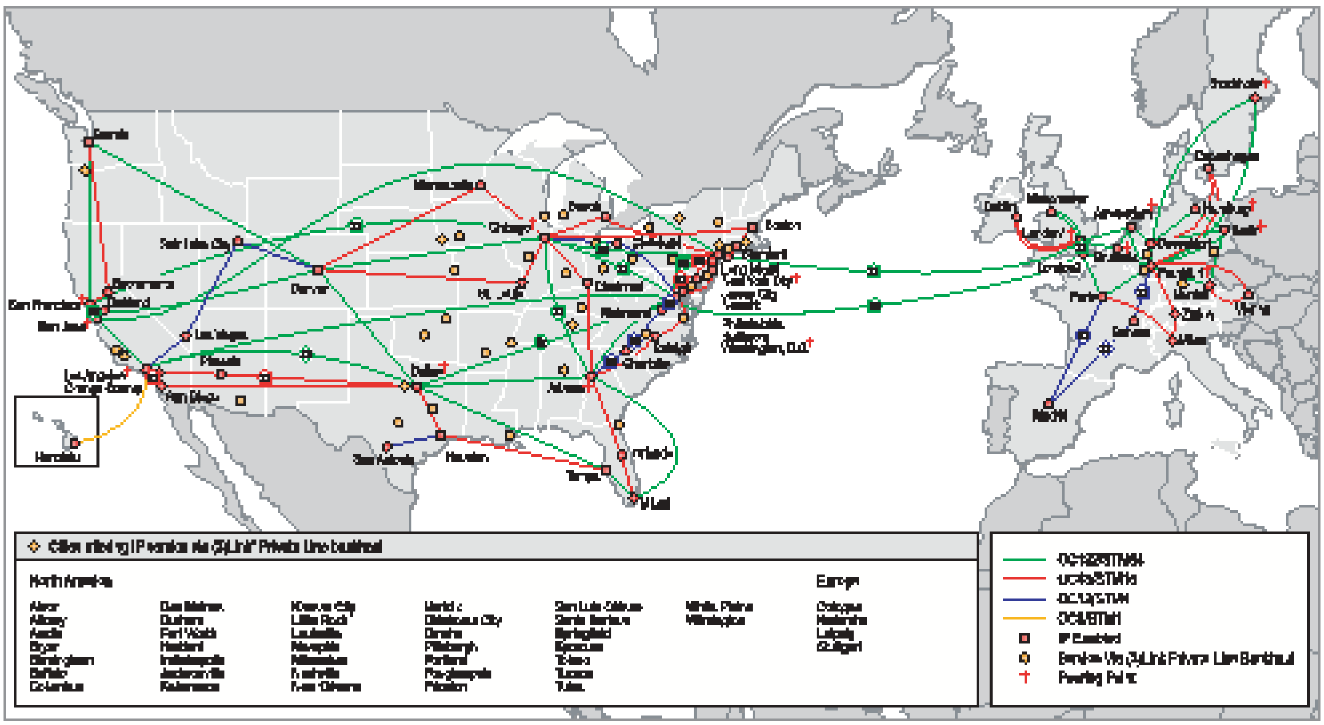
  $\rightarrow$ 5, 2 and 1 Mbps: fallback rates
- 54 Mbps (and 48, 36, 24, 18, 12, 9, 6 Mbps) for 802.11g/a WLAN
- 100 Mbps (FDDI)
- 64 kb/s (toll quality digitized voice)
- ~10 kbps (cell phone quality voice)
- 144kb/s (ISDN line 2B + D service)
- 1.544 Mbps (T1), 44.736 Mbps (T3)
- 155.52 Mbps (OC-3), 622.08 Mbps (OC-12)
- 1244.16 Mbps (OC-24), 2488.32 Mbps (OC-48)
- popular backbone speeds: 1 GigE and 9953.28 Mbps (OC-192)

Purdue's backbone network (Fall 2004): ITaP



Level3 backbone network: www.level3.com

LEVEL 3 IP BACKBONE



→ 10 Gbps backbone (green): same speed as Purdue

→ part of backbone (red): OC-48

GB, MB, kB:

2^{30}, 2^{20}, 2^{10} bytes; size of data being shipped; influenced by the memory structure of computer; already TB.

$\longrightarrow$ data size: factors of 1024

Common data sizes:

- 512 B, 1 kB (TCP segment size)
- 64 kB (maximum IP packet size)
- 53 B (ATM cell)
- 810 B (SONET frame)

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

$\longrightarrow$ "packet": most generic term

Conventional usage

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

What is traveling on the wires?

Mixed data:

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

$\longrightarrow$ $> 85\%$ of Internet traffic is bulk TCP traffic

$\longrightarrow$ due to Web/HTTP

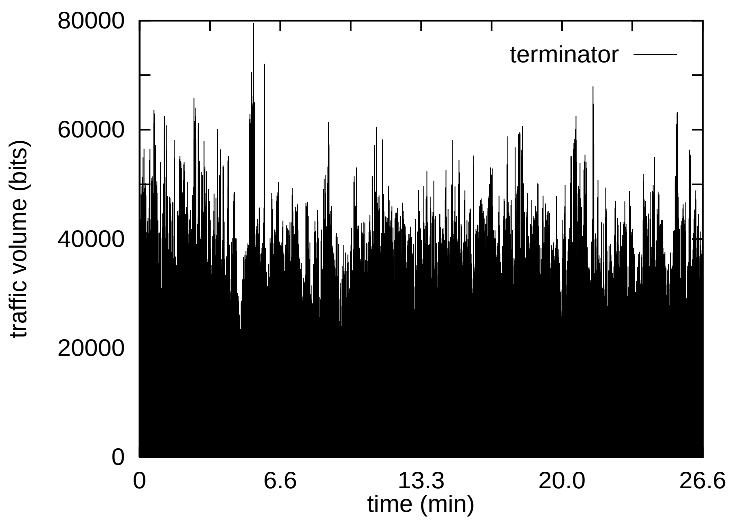
$\longrightarrow$ barriers to streaming traffic implosion

$\longrightarrow$ technical and other

Tilting toward multimedia data; i.e., traffic with QoS requirements including real-time constraints.

Internet traffic is bursty:

⟶ multimedia: MPEG compressed video



Why?

- pattern of scene changes in movies

  → within a scene few changes

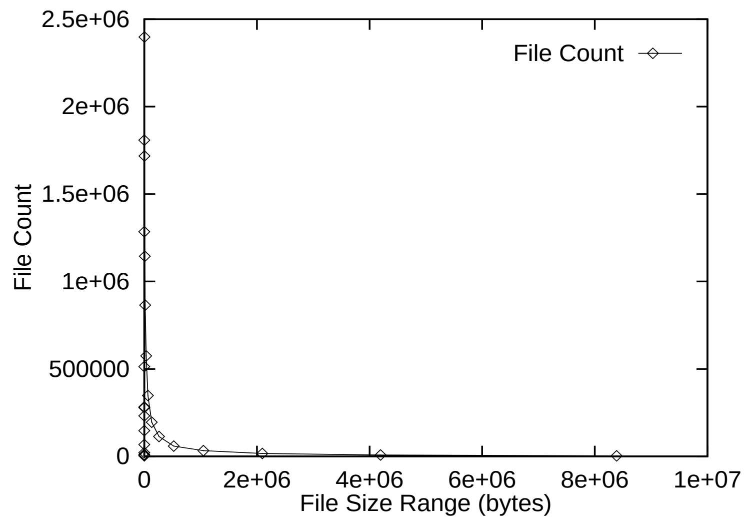
- across scenes, significant scenary changes

  → e.g., action movies

- video compression

  → utilize inter-frame compression

⟶ file sizes on file servers



Why?

- bulk data: 80/20 rule-of-thumb
- majority of files are small, a few very large

  → disproportionate contribution to total traffic

  → "elephants and mice"

Usage pattern in the real-world: uneven or "unfair"

Given mixed payload:

Data networks capable of carrying diverse payload on the same network is a recent phenomenon.

$\longrightarrow$ killer app: VoIP (voice-over-IP)

$\longrightarrow$ other?

$\longrightarrow$ e.g., on-demand audio/video (iTunes)

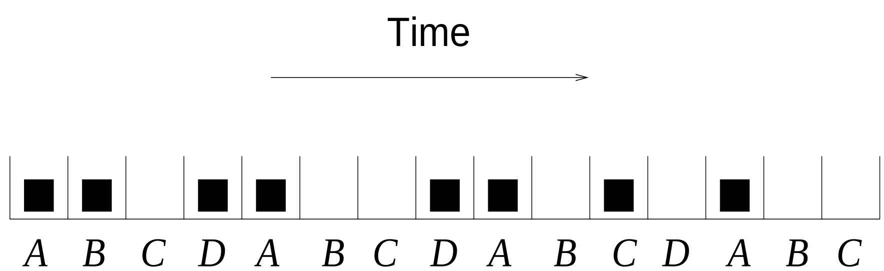
But, even today, much of voice traffic (telephony) is carried on an entirely separate communication network vis-à-vis data traffic, operating under different internetworking principles from the latter.

$\longrightarrow$ time-division multiplexing (TDM) for telephony

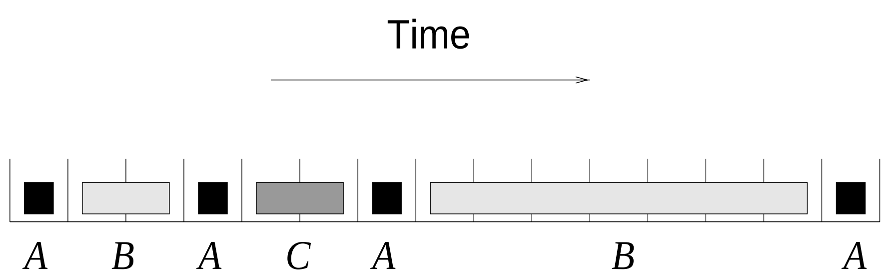
$\longrightarrow$ packet switching for data networks

How is time—viewed as a resource—shared?

Time-division multiplexing:

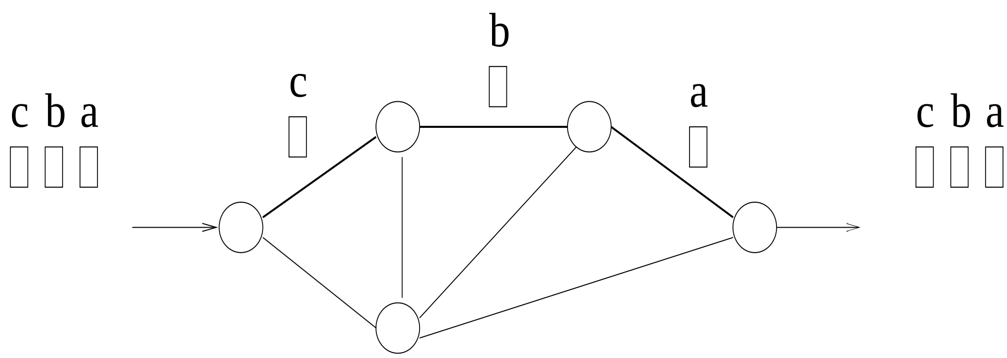


Packet switching:



How is "real estate" shared?

Circuit switching: Virtual channel is established and followed during the lifetime of an end-to-end connection.



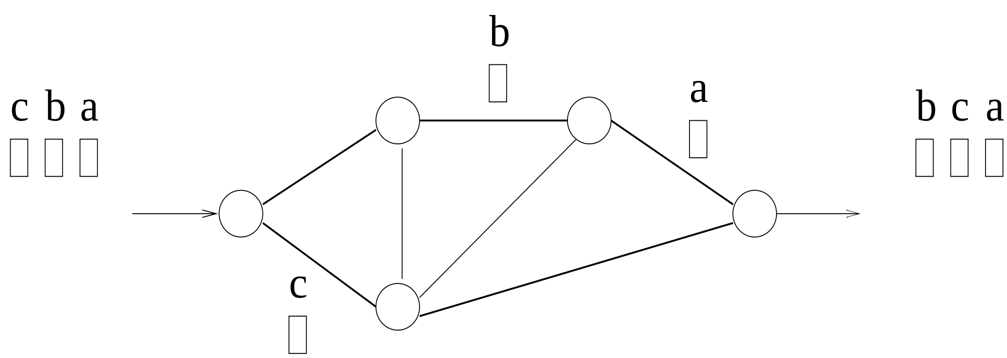
⟶ static route

⟶ in-order delivery

⟶ small routing table

Telephone networks (and ATM networks).

Packet switching: Every packet belonging to an end-to-end conversation is an independent entity; may take a different route from other packets in the same connection.



⟶ dynamic route

⟶ out-of-order delivery

⟶ larger route table

Trade-off between processing overhead and route goodness

Trend: convergence to packet-switched technology

$\longrightarrow$ layer 2 switching in the backbone

$\longrightarrow$ move away from IP due to overhead

$\longrightarrow$ IP critical at peering points

Yet another drawback of packet switching:

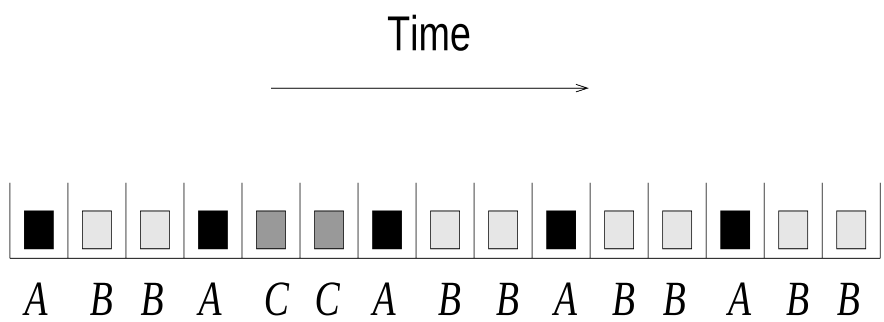
$\longrightarrow$ "bully phenomenon"

$\longrightarrow$ video: 24 frames-per-second (f/s)

$\longrightarrow$ voice: 8000 samples-per-second (s/s)

$\longrightarrow$ what to do?

Asynchronous transfer mode (ATM) :



⟶ 53 byte packet or *cell*.

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

Addresses bully problem but . . .

⟶ significant overhead (48 + 5)

⟶ why 48 bytes?

⟶ performs its own routing

⟶ function duplication

⟶ very complex (overloaded with features)

⟶ confusion with "real" ATM

⟶ peaked in mid-90s; crashed in late 90s

Much has migrated to new layer 2 switching technology

⟶ MPLS (multiprotocol label switching)

⟶ ATM community reincarnated as MPLS . . .

⟶ supporting role to IP

In the meantime, at routers receiving mixed payload . . .

Try to avoid packet loss, but no loss comes at a cost:

- fast memory (buffer) is not cheap
- packets may have to wait in line for their turn

  → queueing delay

  → who gets preference?

Depends on scheduling.

- FIFO (first-in-first-out)
- priority queue
- round robin + weighted fair queue

  → use TOS field of IPv4 header to encode priority

  → packet format: header + payload
- reservation

  → software-based “line leasing”

Is adding more and more buffer space a good solution?

$\longrightarrow$ no: related to “elephants and mice”

$\longrightarrow$ bandwidth is preferred (and, presently, cheaper)

When is it outright bad?

$\longrightarrow$ real-time multimedia payload

How to make sense of all this?

Study of networks can be divided into three aspects:

- architecture

  $\longrightarrow$ system design or blueprint

- algorithms

  $\longrightarrow$ how do the components work

- implementation

  $\longrightarrow$ how are the algorithms implemented

Architecture

- hardware
  - communication or data link technology (e.g., Ethernet, SONET, CDMA/DSSS, TDMA)
  - 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)

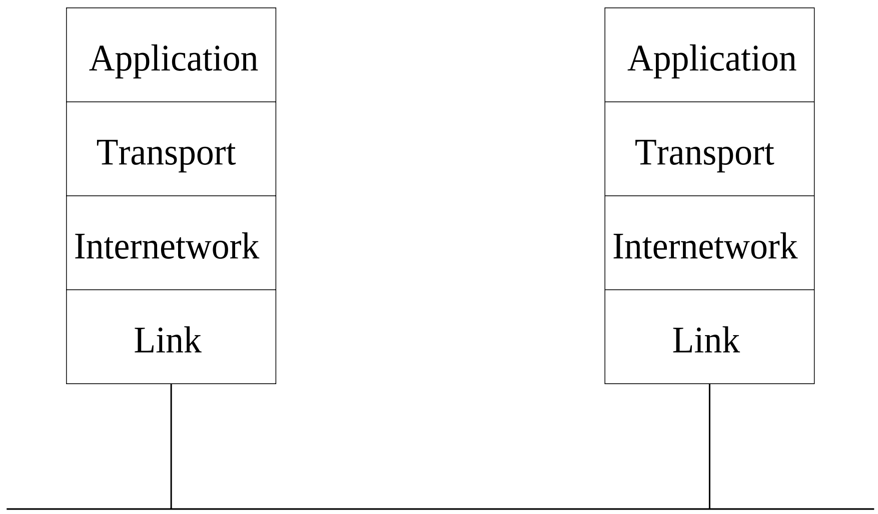
$\longrightarrow$ the *what* over *how*

Provides the "skeleton" for everything else.

... speaking of *standards*,

- ISO (International Standards Organization). ISO/OSI 7-layer reference model.
- ITU (International Telecommunications Union). Successor of CCITT (used to be parent organization), U.N.-chartered.
- IEEE. Professional society, LAN standards; e.g., IEEE 802.3 (Ethernet), IEEE 802.11 (WLAN), IEEE 802.5 (token ring).
- IETF (Internet Engineering Task Force). Internet protocol standardization body.
- W3C. World Wide Web consortium. Application layer web protocols and representations.
- ATM Forum. Industry organization (defunct).
- many others ...

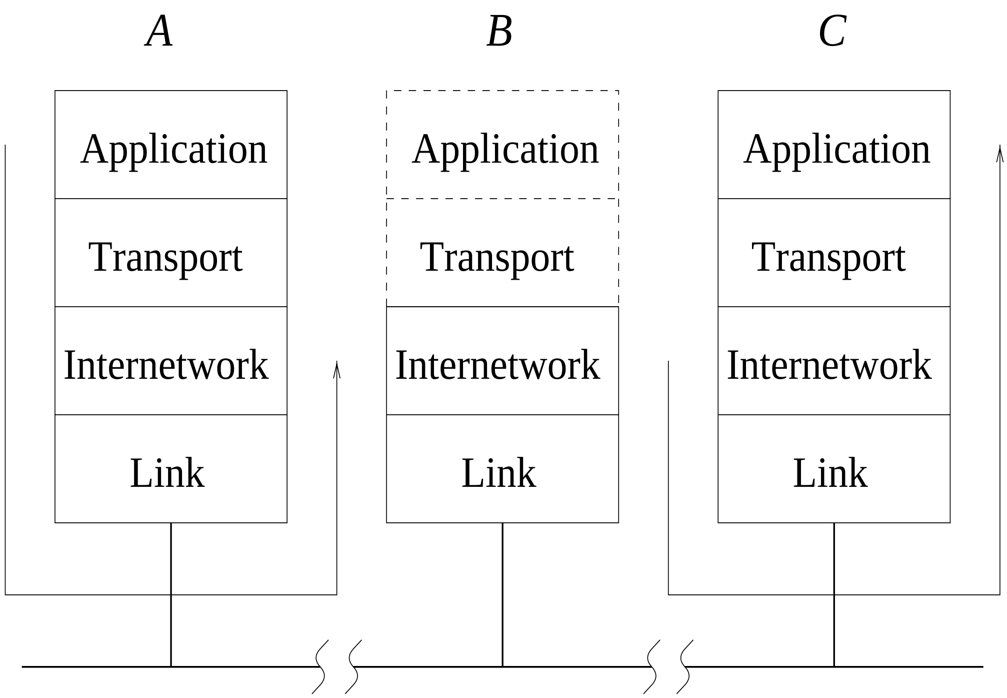
Layering: protocol stack



Achieves abstraction, modularization; two types of interfaces:

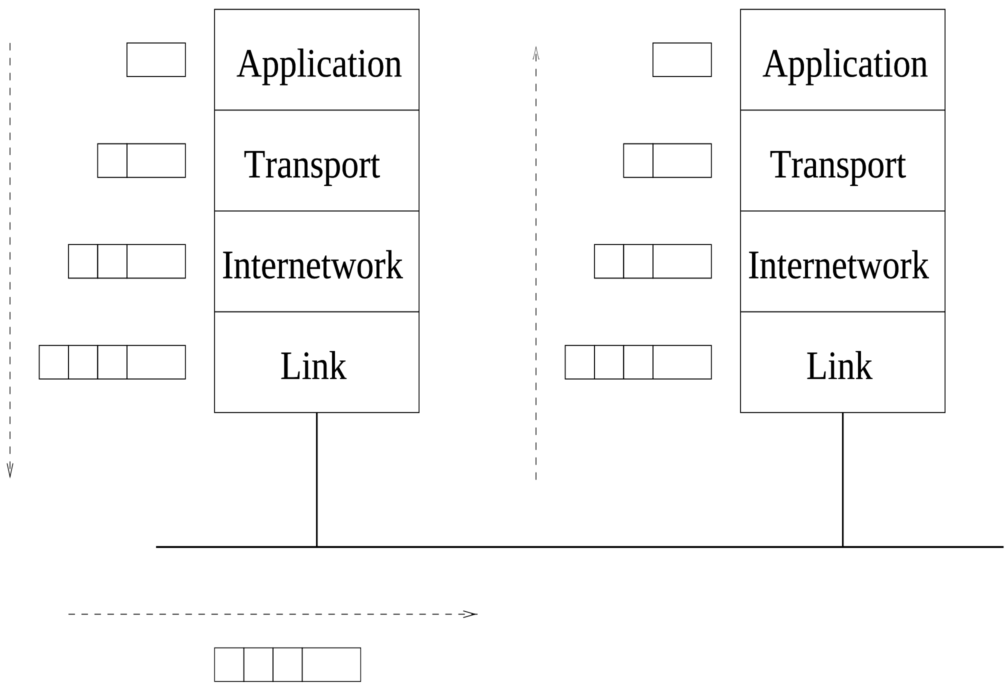
- vertical: inter-layer communication
  - SAP (service access point)
  - PDU (protocol data unit)
- horizontal: peer-to-peer

Internetworking example:



$\longrightarrow$ note processing of packet at B

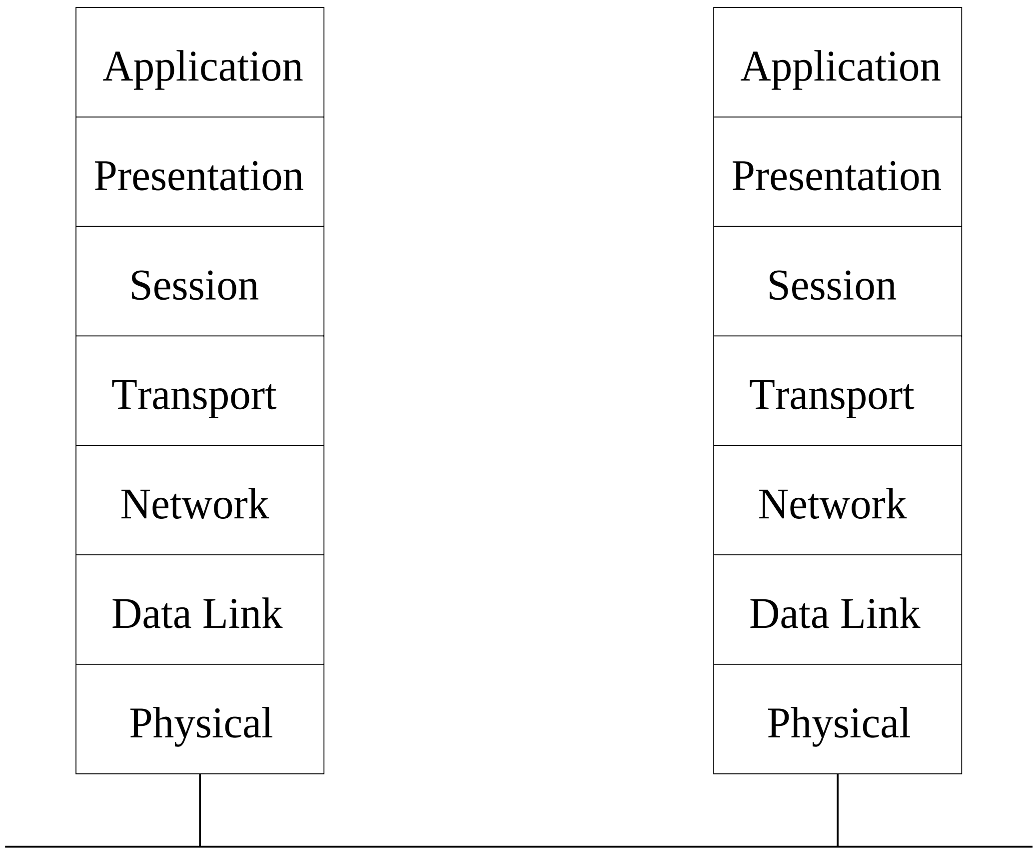
Encapsulation:



- protocol stack (push/pop)
- header/trailer overhead

  → e.g., addressing, error detection
- segmentation/fragmentation and reassembly

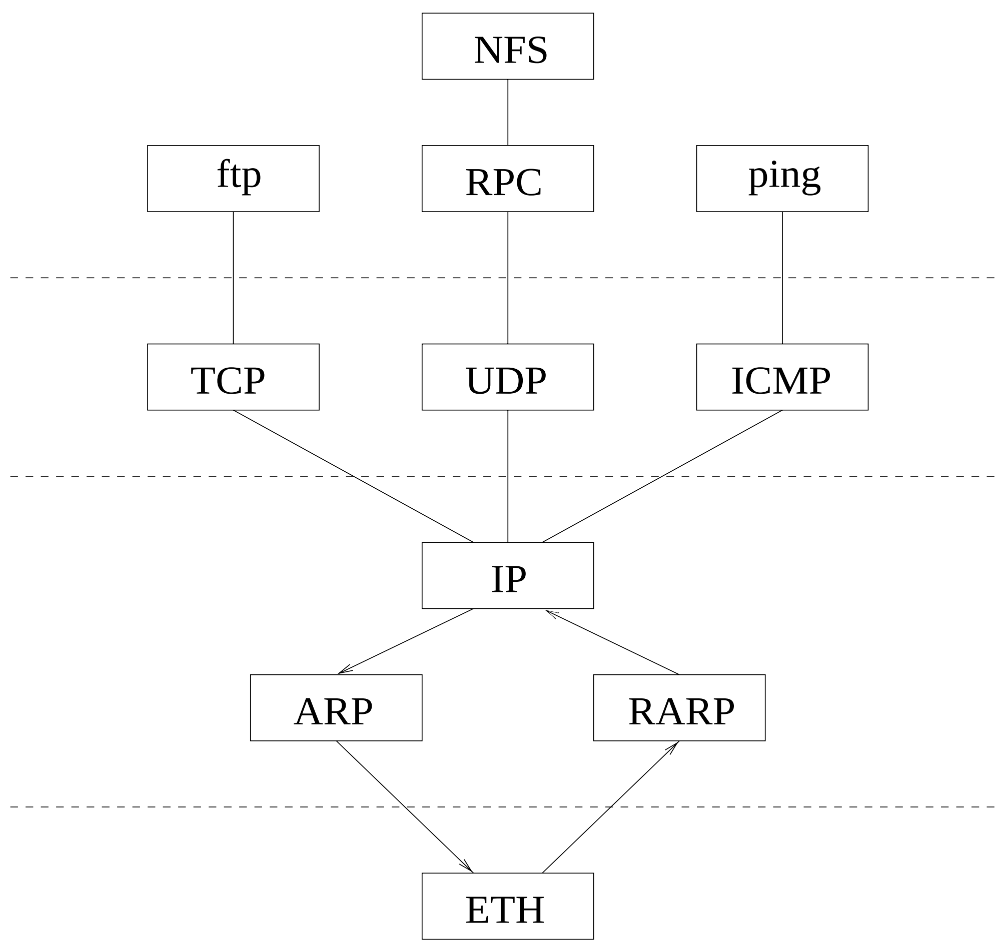
ISO/OSI 7-layer reference model:



Outdated; still semi-useful as historical reference point.

Protocol graph:

Shows logical relationship between protocol modules in the protocol stack.



Algorithms

- error detection and correction (e.g., checksum, CRC)
- medium access control (e.g., CSMA/CD, token ring, CSMA/CA)
- routing (e.g., shortest path—Dijkstra, Bellman & Ford; policy based)
- congestion control (e.g., TCP window control, multimedia rate control)
- scheduling (e.g., FIFO, priority, WFQ)
- traffic shaping and admission control
- packet filtering (e.g., firewalls)
- overlay networks (e.g., VPNs)

$\longrightarrow$ *how* aspect of computer networks

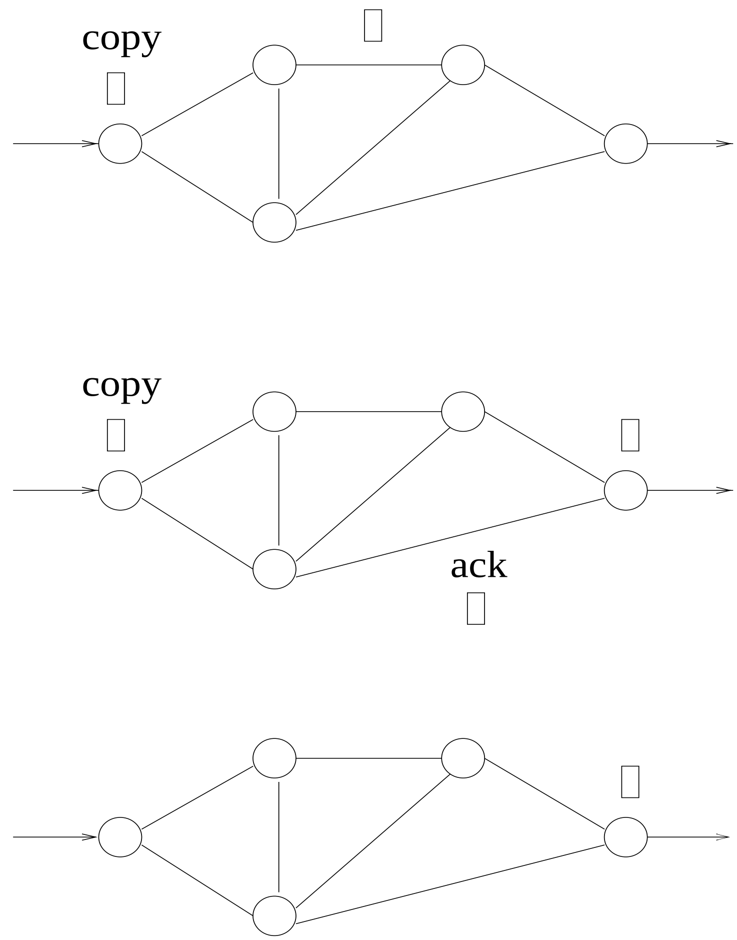
Impacts network performance by controlling the underlying resources provided by the network architecture.

Example: reliable 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

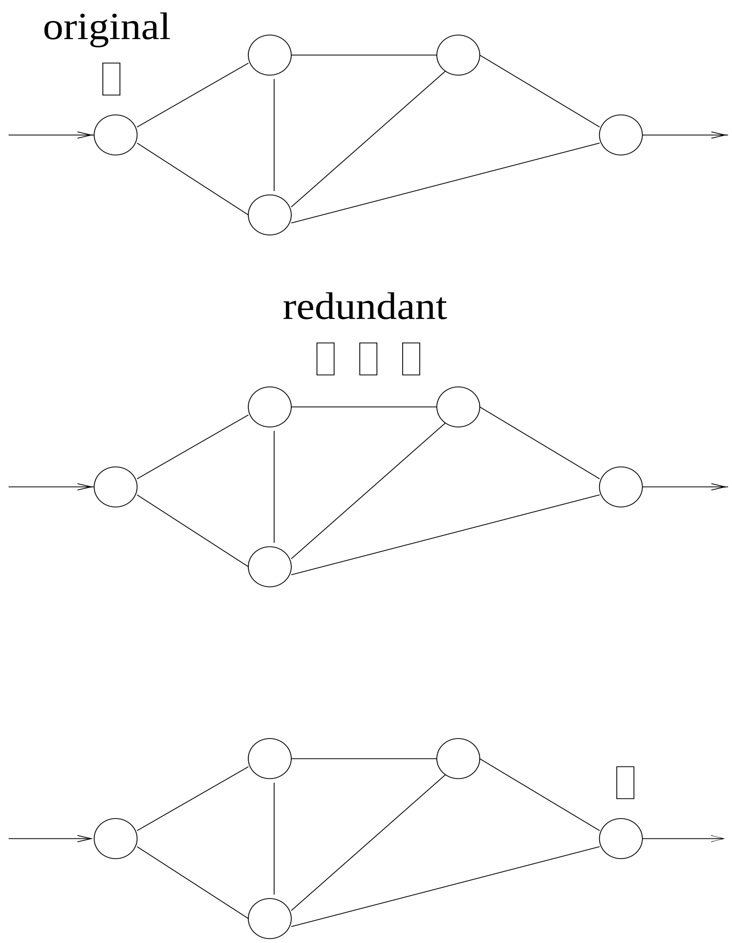
Internet philosophy: reliable transport (TCP) over unreliable internetwork (IP). Use retransmission and acknowledgment (ACK).



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

... or vice versa (negative ACK); 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 how losses occur
- appropriate for real-time contrained data

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

Pros/cons vis-à-vis retransmission ...

Implementations

Same algorithm can be implemented in different ways.

Key issue: *efficiency*.

- reduce copying operation

  → pass pointers instead of value

  → in-place processing
- locality of reference

  → packet trains
- multi-threading to hide communication latency
- multi-threading to reduce context-switch overhead

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

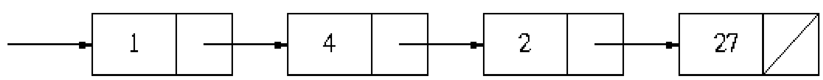
⟶ OO and modularity: secondary to performance

Software clock:

⟶ single hardware clock to emulate multiple clocks

⟶ timer for keeping track of events

Example: want to be notified at time 1 sec, 5 sec, 7 sec, 34 sec from now.



Hardware clock interrupt handling routine:

⟶ kept minimal

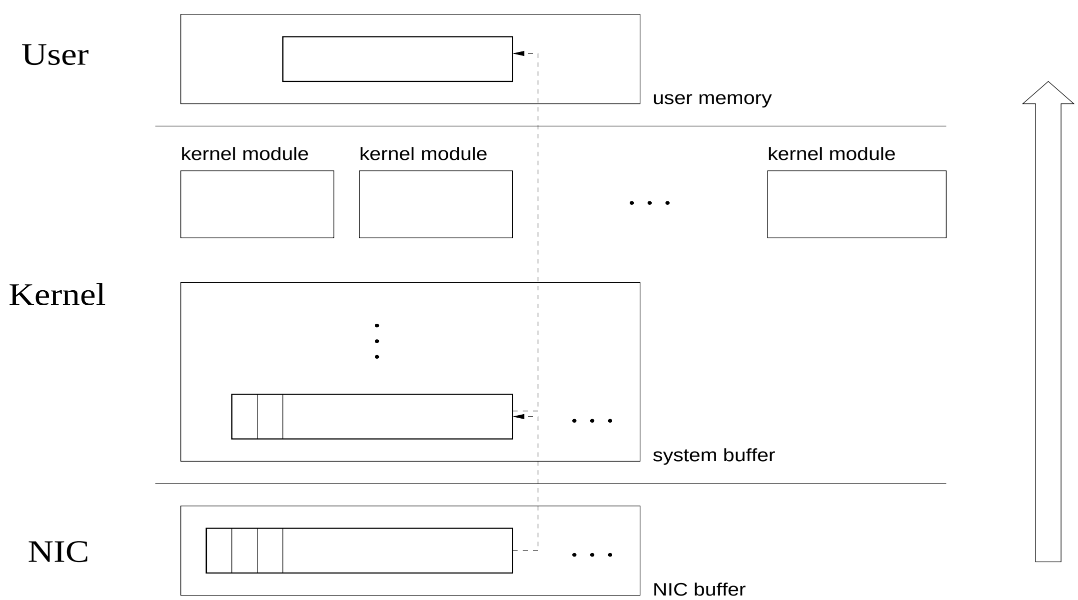
⟶ house-keeping chores through software clock

Vertical & horizontal design:

- keep copy operation to minimum
- use shared memory with pointers

  $\longrightarrow$ vertical design
- use horizontal design to achieve parallelism

  $\longrightarrow$ multi-threading



User space memory management.

$\longrightarrow$ data structure: e.g., trie, hashing for IP table

$\longrightarrow$ 300,000+ route entries

$\longrightarrow$ garbage collection

Keep number of system calls small.

$\longrightarrow$ system call is costly

$\longrightarrow$ stay in user space, if possible

Disk I/O.