How to make sense of all this?

Study of networks can be divided into three aspects:

• architecture
  → system design or blueprint

• algorithms
  → how do the components work

• implementation
  → 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)

→ the what over how

Provides the “skeleton” for everything else.
... speaking of *standards*,

- 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.
- 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:

\[ \]
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Outdated; still semi-useful as historical reference point.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)

→ 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 constrained data

  $\rightarrow$  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 reduce context-switch overhead
- multi-threading to hide communication latency

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.

![Diagram showing time intervals]

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
  → vertical design
- use horizontal design to achieve parallelism
  → multi-threading
User space memory management.

- data structure: e.g., trie, hashing for IP table
- 300,000+ route entries
- garbage collection

Keep number of system calls small.

- system call is costly
- stay in user space, if possible

Disk I/O.