Networking fundamentals:

Reference:
S. Keshav, "An Engineering Approach to Computer Networking: ATM Networks, the Internet and the Telephone Network", Addison-Wesley Professional Computing Series, ISBN 0-201-63442-2

==========

Fundamental ***problems*** in networking:

- Coding, transmission
- Multiple access, location tracking
- Switching and scheduling
- Naming, Addressing, Routing
- Error control, flow control

==========

Fundamental ***concepts and techniques*** in networking to attack these problems:

- Coding and transmission:
  - Digitization/sampling, CRC for error control, framing or timing for synchronization.
- Multiple access:
  - Media characteristics: Shared (broadcast) or switched media;
  - Basic access: time division (TDMA), Frequency division (FDMA), Code-division or frequency hopping (CDMA)
  - Basic techniques: collision, carrier sense, collision detect, multiple access, slots, tokens, cells, frequency reuse, handoffs, base-stations.
- Switching and scheduling:
  - Alternative to network of switches: billions of wires to your home !
  - Two parts: Switch controller (control plane) and switching hardware (data plane). Separate network of switching controllers is an overlay network.. In the Internet switch (router) controller protocols also use IP.
  - Connecting inputs to outputs: Space division and time division
    - Space division: data paths from input to output separated in space
      - Eg: crossbar: inputs = row, outputs = column; crosspoints
    - Time division: n inputs stored in buffer; switch reads n times faster and writes to outputs in *proper order*. Order swapping equipment called time slot interchange (TSI) -- core of PBXs. Combination of time-division and space-division is used in real telephony switches.
  - Packet switching involves buffering, and an intelligent switching core known as the fabric. Simple fabric = bus. Complex: batcher-banyan networks.

    - Buffering also necessitates a header (meta-data) in packets.
    - Where there is buffering, there is an issue of scheduling to determine buffer, bandwidth and delay allocation to competing packets/flows. Scheduling is not so important in circuit switching…
- Naming/Addressing:
    - name/address resolution protocols: mapping names/addresses between different protocol layers
    - hierarchical addresses/names for manageability. Geographic hierarchy or provider-based hierarchy…
    - Address reuse and translation schemes: managing a scarce address space.
    - Separation between routing/connection-setup and forwarding
- Routing:
    - Intra-domain: connectivity, shortest-path access
    - Inter-domain: connectivity, policy-based access.
    - Large transit domains (ISPs): traffic engineering, QoS support
- Error control and flow control:
    - Error detection (CRC) + discard or error correction (FEC etc)
    - Loss/congestion detection + Retransmission
    - Backoff, Window or rate-based transmission

==========

General System Design ideas:

- ***Protocol:*** Set of rules and formats that govern the communication between peers. Specification involves: message semantics and actions taken while sending/receiving them. Interface between layers is also called the architecture. Interface design crucial because interface outlives the technology used to implement the interface.
- ***Layering and encapsulation:*** manage complexity through the divide-and-conquer approach. Allows a subroutine abstraction between a layer and its adjacent layers. Layering allows pipelined implementations (if not fully parallel ones).
    - ***Application layer framing:*** packet format at every layer flexible and partly application-defined
- ***Circuit-switching*** (resource (circuit) reservation followed by time-bound transmission). Resources wasted if unused: expensive. Straighforward to assure quality for voice (150 ms round trip delay, 64 Kbps bandwidth). Observe that time slots have no meta-data (header) associated. All relevant meta-data is inferred from timing and state installed during circuit/connection-setup.

- ***Packet-switching*** (packets with meta-data (header) and store-and-forward type transmission). Very efficient – can exploit multiplexing gains both in space and time (see below), at the cost of a self-descriptive header per-packet, buffering and delays for applications. Great for data. Switch design for packets different from those for voice samples. ..
- ***Statistical Multiplexing***: Reduce resource requirements by exploiting statistical knowledge of the system. Specifically, setup server such that:

  average rate <= service rate <= peak rate
  Muxing Gain = peak rate/service rate.
  Cost: buffering, delays for applications. Useful only if peak rate differs significantly from average rate.
  Spatial muxing: Decrease resource sizing expecting smaller set of sources to be active at any time instant. Cost: call-blocking.
  Temporal muxing: even if many are active at any particular time instant, expect that the average over time will be much smaller. Add buffers. Cost: buffers and meta-data (headers) in packets => need packet switching to exploit both spatial and temporal gains.
  Tradeoff space and time resources for (gain in) money (i.e. optimize use of expensive resource).
  Virtualization: If QoS is met, multiplexed
  shared resource may seem like a unshared virtual resource. In fact, multiplexing can allow us to convert a physical resource into multiple virtual resources. Specifically, Multiplexing + indirection = virtualization, i.e., refer the virtual resource as if it were the physical resource itself. Eg: virtual memory, virtual circuit, socket ports in BSD, telephone call. But, indirection requires binding and unbinding...
  Economics: Assumption: buffer cheaper than bandwidth => tradeoff the latter for the former. With WDM and new wire installation techniques, cost of bandwidth dropping => economic drive for spatial statistical muxing reducing: eg: Qwest.
- ***Pipelining and parallelism:*** trading computation for (gain in) time. Can split up task into N independent subtasks, each requiring same amount of time. Note that both throughput and response time are speeded up by a factor of N. Pipelining: Can't independently split subtasks - the subtasks may be serially dependent (eg: ordering

requirement of layering, instruction execution or on TCP packets). We can still get factor-of-N speedup in throughput, but NOT in response time by using pipelining

- ***Batching:*** trading response time for (gain in) throughput Batching is good when overhead per task increases less than linearly w/ number of tasks and time to accumulate a batch is not too long. Eg: Interrupt handling of back to back packets can be batched. Silly window avoidance in TCP is an application of batching. TCP also has triggers to avoid batching for telnet packets -- when response time is important and cannot be traded off any further.
- ***Randomization***: breaking ties without biases or high probability of repeat of tie. Eg: Use of exponential backoff in broadcast multiple access (ethernet), avoidance of ACK or NAK implosion in reliable multicast, or in some routing algorithms.
- ***Locality***: Critical in exploiting smaller, faster resource to create an illusion of a larger, faster resource. The real larger, slower resource, is used as backup when the access to the smaller resource fails.
- ***Hierarchy***: Scales well. Loose hierarchies more efficient than strict ones (eg: children can interconnect). Eg: managing name space or address allocation and forwarding. Different types of hierarchy: topological, routing, traffic management, organizational. Make sure you are dealing with the right one.
- ***Separating data and control:*** Per-packet actions are part of critical data path -- fewer control actions => greater forwarding speed. Greater separation of data and control => need to install more state in the network. Eg: separate CCIS channel for signaling in telephony networks with the SS7 protocol that enables international calling. In the internet, routing protocols setup tables (control), which are then simply read during forwarding (data).
- ***Extensibility:*** hooks for future growth. Eg: version field, reserved fields.

==========