Review of Networking and Design Concepts

There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies

--- CAR Hoare

- Networking and Design concepts
- Layering: Reference Models
- Data link/MAC:
  - Ethernet/IEEE 802.3 LANs, SLIP, PPP
- Interconnection Devices

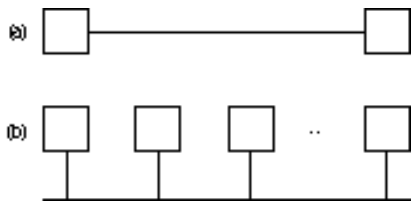
Many of these concepts are taught in CCN (ECSE-4670)

Information, Computers, Networks

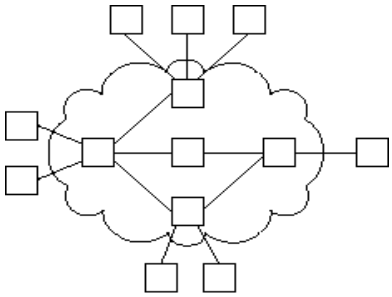
- Information: anything that is represented in ***bits***
  - *Form* (can be represented) vs *substance* (cannot)
  - All "*knowledge*" and "*control*" of physical systems
  - Theory: Information created when you reduce *uncertainty*
- Properties:
  - Infinitely replicable
  - Computers can "***manipulate***" information
  - Networks create "***access***" to information
- Potential of networking:
  - move bits ***everywhere***, ***cheaply***, and with desired ***performance characteristics***

Network provides access or *connectivity*...

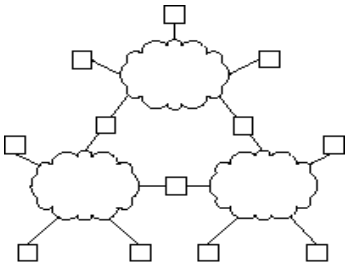
- Building Blocks
  - links: coax cable, optical fiber...
  - nodes: general-purpose workstations...
- Direct Links
  - point-to-point
  - multiple access



- ***Indirect*** Connectivity
  - switched networks



  - inter-networks



What is "*connectivity*" ?

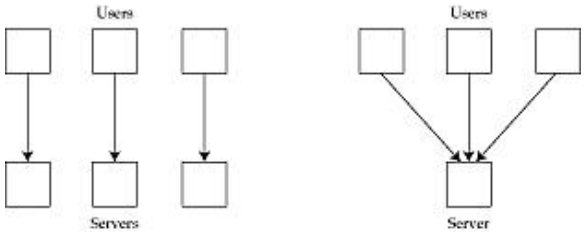
- ***Direct or indirect "access" to every other node in the network***
- ***Access*** is not the same as having a pt-pt link
  - What you get is a ***"virtual channel"*** between nodes, which does not necessarily have the same performance characteristics of a physical link.
  - For example, a virtual channel may minimally provide only "***best-effort connectivity on a packet-by-packet basis"*** whereas a link provides an ***"always-connected, fixed bandwidth, fixed delay and near zero-jitter"*** channel.
- Virtualization is made up of two parts:
  - Multiplexing
  - Indirection
- But lets take a break and understand impt design concepts

System Design Ideas

- *Resources:*
  - space
  - time
  - computation
  - money
  - labor
  - Design a system to ***tradeoff cheaper resources against expensive ones*** (for a gain)

Multiplexing

- Multiplexing = sharing
  - Trades time and space for money
  - ***Cost***: waiting time (delay), buffer space, loss (if not enough buffer)
  - ***Gain***: Money ($$) => Overall system costs less
  - Eg: Time-Division Multiplexing (TDM), Frequency-Division Multiplexing (FDM)



Statistical Multiplexing

- Reduce resource requirements by *exploiting statistical knowledge* of the system. Specifically, choose service rate such that:
  - average rate <= service rate <= peak rate
  - *Multiplexing Gain* = peak rate/service rate.
  - Cost: buffering, delays for applications. Tradeoff space and time resources for money
  - *Useful only if peak rate differs significantly from average rate.*

Example system: Circuit Switching

- *Circuit-switching*: resource (circuit) reservation followed by time-bound transmission.
  - Resources wasted if unused: expensive.
  - Straightforward to assure quality for voice (150 ms round trip delay, 64 Kbps bandwidth).
  - Time slots have no meta-data (header) associated. All relevant meta-data is inferred from timing and state installed during circuit/connection-setup.

Example system: Packet-switching

- *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).
  - Cost: self-descriptive header per-packet, buffering and delays for applications. (tradeoff space and time for money)

Spatial and Temporal Multiplexing

- Spatial muxing: Decrease resource sizing expecting smaller set of sources to be active at any time instant.
  - Cost: call-blocking (time)
- Temporal muxing: even if many are active at any particular time instant, expect that the average over time will be much smaller.
  - Cost: buffers and meta-data (headers) in packets (space).
- Note: We need packet switching to exploit both spatial and temporal gains.

Virtualization

- The multiplexed shared resource may seem like a *unshared virtual resource* provided we have a level of *indirection*
  - I.e. Multiplexing + indirection = virtualization
  - We can “refer” to the virtual resource as if it were the physical resource.
  - Indirection requires *binding and unbinding*…
- Examples:
  - A queueing system: a virtual server for each named source
  - A Switch/Router/Bridge with a forwarding table (indirection) provide *virtual links* between any two ports.
  - The Internet uses control protocols like routing/ARP to establish indirections (name->address->lower layer address->next hop) to create virtual links between any two users/apps

Virtualization

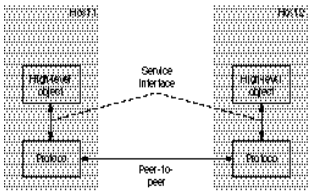
- Virtualization: If Quality of Service (QoS) is met, the multiplexed shared resource may seem like a *unshared virtual resource*.
- Multiplexing + indirection = virtualization, i.e., refer the virtual resource as if it were the physical resource itself.
  - Eg: virtual memory, virtual circuit, socket ports in BSD, telephone call.
  - Indirection requires binding and unbinding...

Circuit, Virtual-ckt, Connection-Oriented, Connectionless

- ***Circuit***: Telephone system
  - Path setup and resources reserved before data is sent
  - Data need not have meta-info at all. Only timing.
- ***Virtual Circuit***: ATM networks
  - Multiple circuits on one wire.
- ***Connection-Oriented***: TCP
  - Have an association between end-points
- ***Connectionless:*** Also known as datagram. IP, postage service
  - Complete address on each packet
  - The address decides the next hop at each routing point

Formal framework: Protocols

- Building blocks of a network architecture
- Each protocol object has two different interfaces
  - *service interface*: defines operations on this protocol
  - *peer-to-peer interface*: defines messages exchanged with peer



- Term "protocol" is overloaded
  - specification of peer-to-peer interface
  - module that implements this interface

Formal framework: Interface design

- Interface between layers is also called the "***architecture***"
  - Use abstractions to hide complexity
  - Allows a subroutine abstraction between a layer and its adjacent layers.
- Interface design crucial because interface outlives the technology used to implement the interface.
- Driven by three factors:
  - ***Functionality*** (what features the customer wants)
  - ***Technology*** (what's possible. Building blocks and techniques)
  - ***Performance*** (How fast etc… User, Designer, Operator)

Techniques:Pipelining and Parallelism

- Goal: trading computation for (gain in) time.
- Degree of parallelism: response time x throughput
- Linear speedup: split up task into N independent subtasks, each requiring same amount of time.
  - Response time speedup of N. Throughput constant. Degree = N
- Pipelining: Can't independently split subtasks - the subtasks may be serially dependent.
  - We can get speedup in throughput, but NOT in response time by using pipelining

Techniques: Batching

- *Goal:* trading response time for (gain in) throughput
- Batching is good when:
  - overhead per task increases less than linearly w/ number of tasks
  - time to accumulate a batch is not too long.
  - Response time can be traded off
- Eg: Interrupt handling, Silly window avoidance in TCP
- TCP also has triggers to avoid batching for telnet packets -- when response time is important

Techniques: Randomization

- Goal: Trade computation for (response) time
- Used in 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.

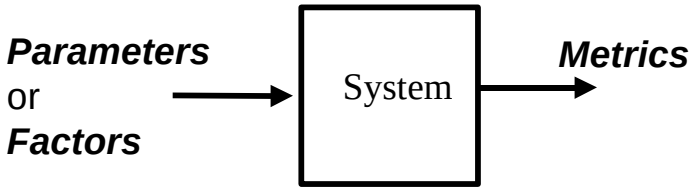
Techniques: Locality and Hierarchy

- *Locality*: Critical in exploiting smaller, faster resource to create an illusion of a larger, faster resource.
  - The larger, slower resource, is accessed when item is not found in the smaller resource.
- *Hierarchy*: for scalability.
  - Loose hierarchies more efficient than strict ones (eg: children can interconnect).
  - Eg: managing name space or address allocation and forwarding.

Miscellaneous techniques

- Different types of hierarchy: topological, routing, traffic management, organizational.
- *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 in telephony.
  - Eg: separate routing protocols in Internet.
- *Extensibility:* hooks for future growth. Eg: version field, reserved fields.

Performance, Metrics and Parameters



- **Performance:**
  - ***How fast*** does computer A run MY program ?
  - Is machine A ***generally faster than*** machine B, and if so, ***how much faster*** ?
- **Parameters**: clock rate, poisson inter-arrivals ...
- **Metrics**: throughput, response time, queue length ...
  - Metric should characterize the design tradeoffs adequately
  - Metrics are usually functions of many factors. Use of one factor alone may be misleading.

More on Metrics/Parameters

- User metrics:
  - How fast does MY program run => we need a measure of execution *time* ?
- System metrics:
  - How much is the system utilized ?
  - How much buffers do I need to provision ?
  - How many programs is it able to execute per second ?
  - => Need a measure of *throughput, queue length*
- Eg: Execution Time = Instrns/pgm *avg clock cycles/Instruction * time/clock cycle.
- T = I /P * CPI *Clock cycle time
- All three factors *combine* to affect the metric.

Workloads, Benchmarks

- ***Workload:*** a test case for the system
- ***Benchmark:*** A set of workloads which together is representative of "MY program". Should be reproduceable
  - Problem: combining metrics from each test case.
  - Pitfalls: ratio games.

Machine / Test case	A	B
1	1s	10s
2	100s	10s

Which is faster, A or B ?

Effect on Design: Amdahl's law

- Execution time after improvement =

  Execution time ***affected*** by improvement / speedup + ***Unaffected*** execution time
- ***Point:*** Speedup the common case !!

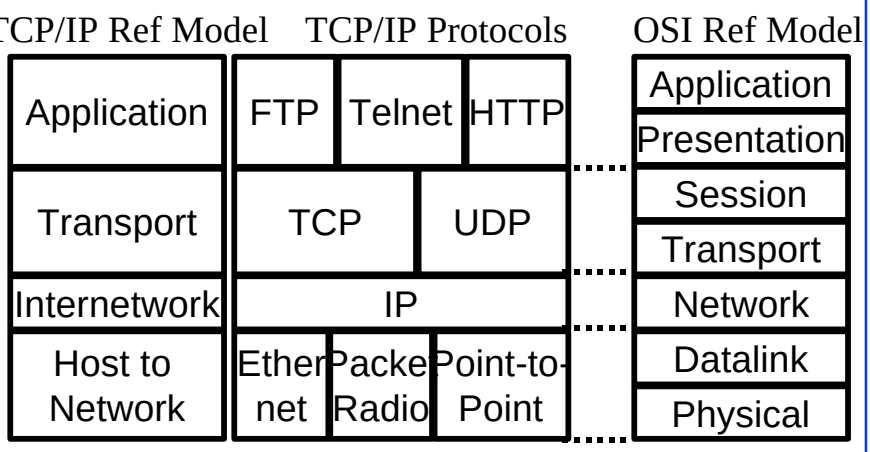
Perspective

- ***Network users:*** services and performance that their *applications need*, e.g., guarantee that each message it sends will be delivered without error within a certain amount of time
- ***Network designers:*** *cost-effective design* e.g., that network resources are efficiently utilized and fairly allocated to different users
  - Users + designer perspectives enough to meet 3 factors of interface/architecture design. But ...
- ***Network providers:*** system that is easy to *administer and manage* e.g., that faults can be easily isolated and it is easy to account for usage
  - Require management tools and interfaces.
  - Often considered to the basic protocol interface design

Key networking concepts

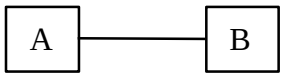
- **Layering:** interfaces, services, peer-peer protocols.
- **Encapsulation:** (de)multiplexing, overlays
- **Addressing/Naming:** address/name resolution, hierarchical vs flat addresses, routing, conn'n setup
- **Communication type:** message passing, packet switching, connection-oriented, connectionless
- **Error & flow control:** framing, CRC, retransmission, backoff, sliding window ARQ protocols.
- **Media access:** shared, switched, collision, carrier sense, collision detect, multiple access, slots, tokens
- **Interconnect devices:** repeaters, hubs, bridges, switches, routers, application gateways

Reference Models for Layering



Where did the problems these layers solve spring up from ?

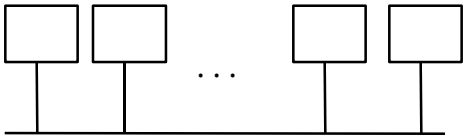
Issues in a point-to-point network ...



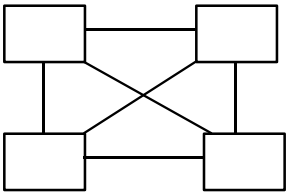
- ***Physical*** layer: coding, modulation etc
- ***Link*** layer: framing, protocol multiplexing, error recovery, flow control…
- No need for protocol flab like addressing, names, routers, hubs etc

Connecting N users: Directly ...

- ***Bus***: broadcast, collisions, media access control
- ***Full mesh***: Cost, simplicity



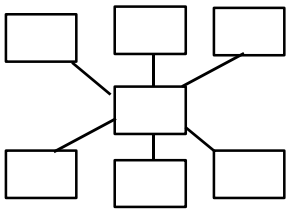
Bus



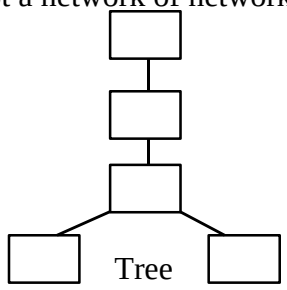
Full mesh

Connecting N users: Indirectly ...

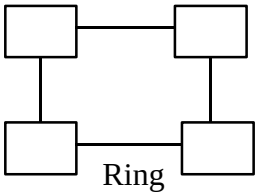
- Star: One-hop path to any node, reliability, forwarding function
- Tree: Minimal links, multiple hop-paths, distributed load
- Ring: Reliability to link failure, near-minimal links etc
- Hybrid
- All these topologies ("multi-access networks") assume a single physical network, not a network of networks



Star



Tree



Ring

Multi-access networks (contd)

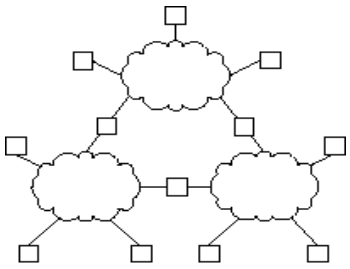
- ***Topology issues***: Cost, reliability, manageability, deployability, scalability, software complexity
- ***Medium Access Protocols***:
  - ALOHA
  - CSMA/CD (Ethernet)
  - Token Ring
  - Wireless
- ***New concepts***: address, forwarding (and forwarding table), bridge, switch, hub, token, medium access control (MAC) protocols

Internetworking

- What is it ?
  - "Connect many disparate physical networks and make them function as a coordinated unit … " - Douglas Comer
- Results:
  - Universal Interconnection
  - User interface is network independent
  - All sub-networks are equal in the eyes of TCP/IP
  - Killer apps: Email, WWW

Inter-networks: networks of networks

- Internetworking involves two fundamental problems: ***heterogeneity and scale***
- ***Concepts***:
  - Translation, overlays, address & name resolution, fragmentation: to handle *heterogeneity*
  - Hierarchical addressing, routing, naming, address allocation, administration: to handle *scaling*

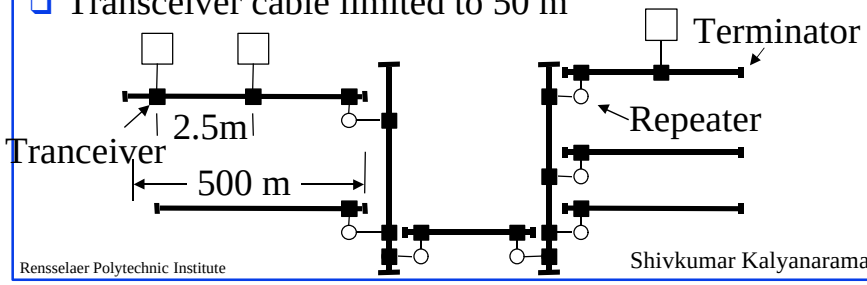


Multiple Access Protocols

- Aloha at University of Hawaii:
  Transmit whenever you like
  Worst case utilization = 1/(2e) =18%
- CSMA: Carrier Sense Multiple Access
  Listen before you transmit
- CSMA/CD: CSMA with Collision Detection
  Listen while transmitting.
  Stop if you hear someone else.
- Ethernet uses CSMA/CD.
  Standardized by IEEE 802.3 committee.

10Base5 Cabling Rules

- Thick coax
- Length of the cable is limited to 2.5 km, no more than 4 repeaters between stations
- No more than 500 m per segment ⇒ 10Base5
- No more than 2.5 m between stations
- Transceiver cable limited to 50 m

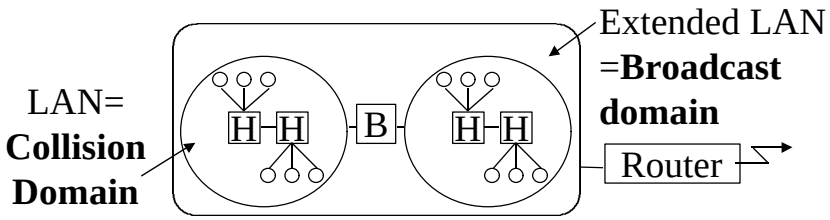


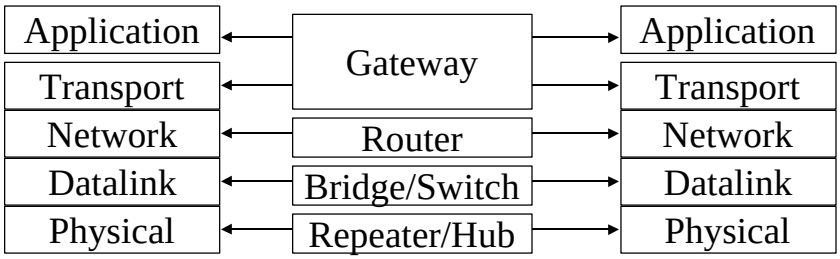
Interconnection Devices

- Repeater: PHY device that restores data and collision signals: a digital amplifier
- Hub: Multi-port repeater + fault detection
- Bridge: Data link layer device connecting two or more collision domains. MAC multicasts are propagated throughout “extended LAN.”
- Router: Network layer device. IP, IPX, AppleTalk. Does not propagate MAC multicasts.
- Switch: Multi-port bridge with parallel paths

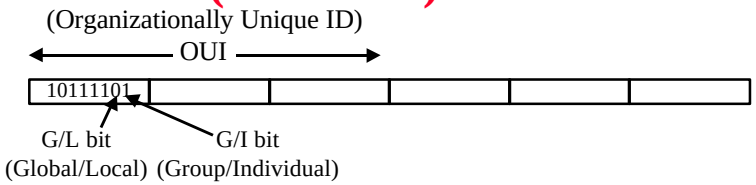
These are functions. Packaging varies.

Interconnection Devices





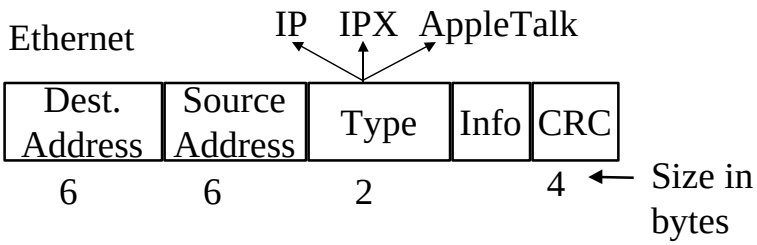
Ethernet (IEEE 802) Address Format



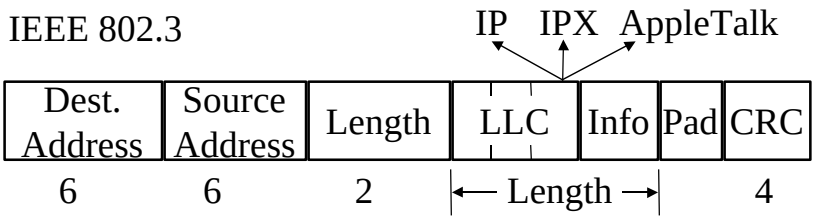
- G/L bit:
  - Global: unique worldwide; assigned by IEEE
  - Local: Software assigned
- G/I: bit:
  - I: unicast address
  - G: multicast address
  - Eg: Multicast = “To all bridges on this LAN”
  - All 1s => Broadcast = “To all stations”

Frame Format

- Ethernet



- IEEE 802.3



Serial IP (SLIP)

- Simple: only framing = Flags + byte-stuffing
- Compressed headers (CSLIP) for efficiency on low speed links for interactive traffic.
- Problems:
  - Need other end's IP address a priori (can't dynamically assign IP addresses)
  - No "type" field => no multi-protocol encapsulation
  - No checksum => all errors detected/corrected by higher layer.
- RFCs: 1055, 1144

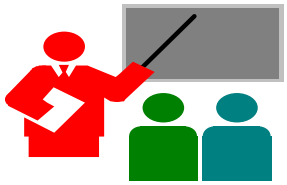
PPP

- Frame format similar to HDLC
- Multi-protocol encapsulation, CRC, dynamic address allocation possible
  - key fields: flags, protocol, CRC (fig 2.3)
- Asynchronous and synchronous communications possible
- Link and Network Control Protocols (LCP, NCP) for flexible control & peer-peer negotiation
- Can be mapped onto low speed (9.6Kbps) and high speed channels (SONET)
- RFCs: 1548, 1332

MTU

- Maximum Transmission Unit
- Key link layer characteristic which affects IP performance.
- (IP datagram size > MTU) => fragment => inefficient
- Path MTU: smallest MTU on any traversed link on path => TCP/IP can be more efficient knowing this.
- Reducing MTU for a low speed CSLIP line can lead to lesser transmission/propagation times for interactive traffic

Summary: Laundry List of Problems



- Topologies, Framing, Error control, Flow control
- Multiple access
  - How to share a wire
- Switching, bridging, routing
- Naming, addressing
- Resolution (name/address), fragmentation
- Congestion control, traffic management, Reliability
- Network Management