Review of Networking and Design Concepts

***Two ways** of constructing a software design:*

1) ***make it so simple that there are obviously no deficiencies, and***
2) ***make it so complicated that there are no obvious deficiencies***

--- CAR Hoare

Based in part upon slides of Prof. Raj Jain (OSU), S. Keshav (Cornell), L. Peterson (Princeton), J. Kurose (U Mass)

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

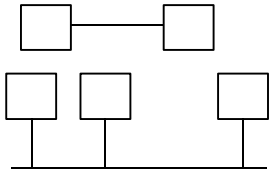
Many of these concepts are taught in CCN (ECSE-4670)

Information, Computers, Networks

- Information: anything that is represented in ***bits***
  - *Form* (can be represented) vs *substance* (cannot)
- Properties:
  - Infinitely replicable
  - Computers can *"**manipulate**"* information
  - Networks create *"**access**"* to information
- Potential of networking:
  - move bits ***everywhere***, ***cheaply***, and with desired ***performance characteristics***
  - ***Break the space barrier for information***

Connectivity...

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



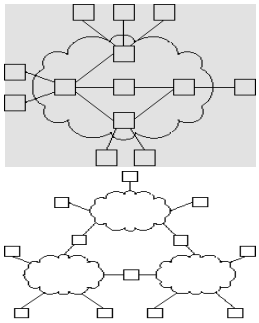
Connectivity… (Continued)

- *Indirect* Connectivity
  - switched networks

    => ***switches***

  - inter-networks

    => ***routers***

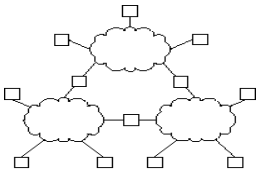


What is "*Connectivity*" ?

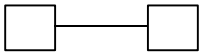
- *Direct or indirect access to every other node in the network*
- *Connectivity* is the *magic* needed to communicate if you do not have a link.
  - *Tradeoff: Performance characteristics worse!*

Connectivity …

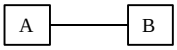
- Internet:
  - *Best-effort*
  - *(no performance guarantees)*
  - *Packet-by-packet*



- A pt-pt link:
  - *Always-connected*
  - *Fixed bandwidth*
  - *Fixed delay*
  - *Zero-jitter*



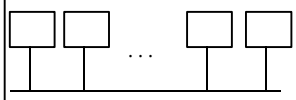
Point-to-Point Connectivity Issues



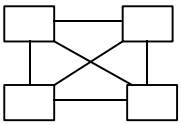
- **Physical** layer: coding, modulation etc
- **Link layer** needed if the link is shared bet'n apps; is unreliable; and is used sporadically
- *No need for protocol concepts* like addressing, names, routers, hubs, forwarding, filtering …
- What if I want to build a *network with N nodes* and *let N increase* ?

Connecting N users: *Directly* ...

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



Bus



Full mesh

- ***Address concept*** needed if we want the receiver alone to consume the packet!
  - Required in all topologies…

Scaling: *Filtering*

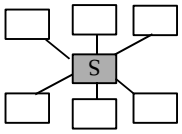
- Scaling: system allows the increase of a key parameter within tradeoffs. Eg: let N increase…
  - *Inefficiency limits scaling* …
  - *Direct connectivity: inefficient & does not scale*
- Mesh: inefficient in terms of # of links
- Bus architecture: 1 expensive link, N cheap links
  - Filtering: choose a subset of elements
  - Receivers need to "filter" out their packets
  - Packet "broadcast" on "bus"
  - Problem: broadcast is *bandwidth inefficient*

How to scale: *filtering, forwarding …*

- Filtering: choose a subset of elements from a set
  - A generic concept could apply to set of packets, links or nodes
  - *Filtering is the key to efficiency*
- Forwarding: actually sending packets to a filtered subset of link/node(s)
  - Packet sent to one link/node => efficient
    - Why ? Others can be used in parallel
  - *Parallel forwarding also leads to efficiency*
- Solution: Build nodes which filter/forward and connect indirectly => "switches" & "routers"

Connecting N users: *Indirectly* ...

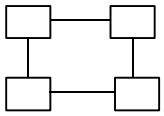
- **Star:** One-hop path to any node, reliability, forwarding function
- "Switch" S.can filter and forward!
  - Switch may forward multiple pkts in parallel !
- Forwarding without filtering => "hub"
  - Emulates "bus" + needs filtering at hosts



Star

Connecting N users: *Indirectly* …

- **Ring:** *Reliability* to link failure, *near-minimal* links
- All nodes need "forwarding" and "filtering"
- Sophistication of forward/filter lesser than switch



Ring

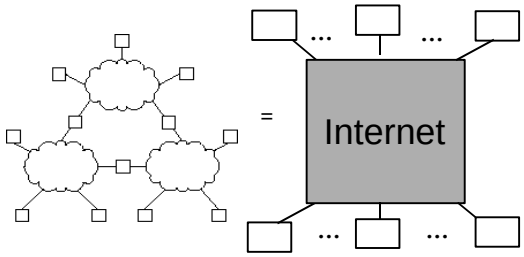
Multi-Access LANs

- ***Hybrid topologies:*** direct & indirect
  - Limited scalability due to limited filtering
- ***Topology issues***: Cost, reliability, manageability, deployability, scalability, complexity
- ***Medium Access Protocols***:
  - ALOHA, CSMA/CD (Ethernet), Token Ring …
  - *Key: Use a single protocol in network*
- ***Concepts***: address, forwarding (and forwarding table), bridge, switch, hub, token, medium access control (MAC) protocols

Inter-Networks: Networks of Networks

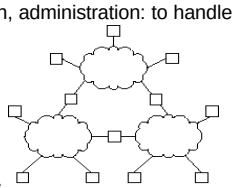
- What is it ?
  - "Connect many disparate physical networks and make them function as a coordinated unit … " - Douglas Comer
  - *Many => scale*
  - *Disparate => heterogeneity*
- Result: *Universal connectivity!*
  - ***The inter-network looks like one large switch, I.e.***
  - ***User interface is sub-network independent***

Inter-Networks: Networks of Networks



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*

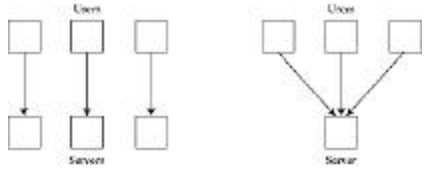


System Design Ideas

- *Resources:*
  - Space
  - Time
  - Computation
  - Money
  - Labor
- Design a system to ***tradeoff cheaper resources against expensive ones*** (for a gain)

Building blocks: *Multiplexing*

- Multiplexing = sharing
  - Trades time and space for money
  - ***Cost***: waiting time (delay), buffer space & loss
  - ***Gain***: Money ($$) => Overall system costs less
- Eg: Time-Division Multiplexing (TDM), Frequency-Division Multiplexing (FDM)



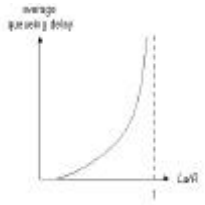
Statistical Multiplexing

- Reduce resource requirements by *exploiting statistical knowledge* of the system.
  - Eg: average rate <= service rate <= peak rate
  - *Multiplexing Gain* = peak rate/service rate.
    - Service rate: *much lower* than peak rate
  - Cost: buffering, queuing delays, losses.
  - Tradeoff space and time resources for money
  - *Useful only if peak rate differs significantly from average rate.*

What's a *tradeoff* ? Eg: Queuing delay

- R=link bandwidth (bps)
- L=packet length (bits)
- a=average packet arrival rate

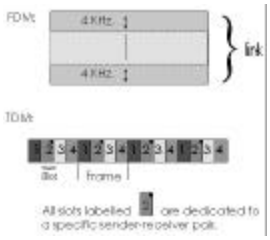
traffic intensity = La/R



- La/R ~ 0: average queuing delay small
- La/R -> 1: delays become large
- La/R > 1: more "work" (*demand*) arriving than can be serviced (*capacity*), average delay infinite (*service degrades unboundedly*)!

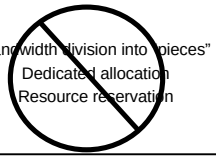
Example: *Circuit-Switching*

- *Circuit-switching*:
  - Divide link bandwidth into "pieces"
  - Reserve pieces of the resource (circuit)
  - Resources wasted if unused: expensive.
  - But, simple to assure quality for voice
  - No meta-data (header)
  - Inferred from timing and circuit state



Example: *Packet-Switching*

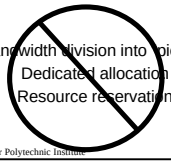
- *Packet-switching:*
  - Chop up data to be transmitted into "packets"
    - *Packets: data + meta-data (header)*
  - "Switch" packets at intermediate nodes
    - *Store-and-forward* if bandwidth is not immediately available.



Packet Switching (continued)

Each end-end data stream divided into *packets*

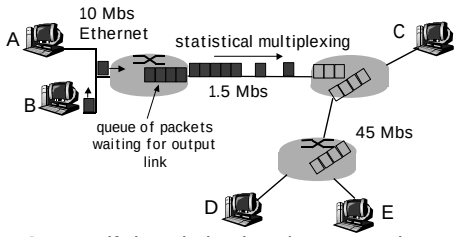
- user A, B packets *share* network resources
- each packet *uses full link bandwidth*
- resources used *as needed*,



Resource *contention*:

- aggregate resource demand can *exceed* amount available
- *congestion:* packets queue, wait for link use
- *store and forward:* packets move one hop at a time
  - transmit over link
  - wait turn at next link

Packet Switching



- Cost: self-descriptive header per-packet, buffering and delays for applications.
- Tradeoff space and time for money

Spatial vs Temporal Multiplexing

- Spatial multiplexing: Chop up resource into chunks. Eg: bandwidth, cake …
- Temporal multiplexing: resource is shared over time, I.e. queue up jobs and provide access to resource over time. Eg: FIFO queueing, packet switching
- Packet switching can exploit *both* spatial & temporal gains.
- *Packet switching is more efficient and hence more scalable !*

Virtualization

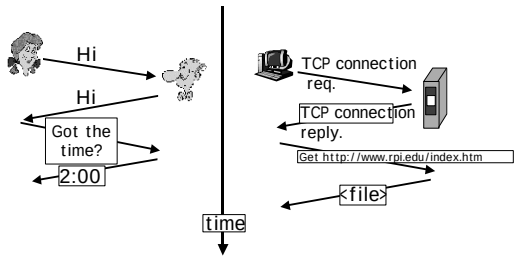
- The multiplexed shared resource with a level of indirection will seem like a *unshared virtual resource*!
  - I.e. Multiplexing + indirection = virtualization
  - We can "refer" to the virtual resource as if it were the physical resource.
    - Pure magic !
  - Eg: virtual memory, virtual circuits…
  - Connectivity: a virtualization created by the Internet!
  - Indirection requires *binding and unbinding*…

Degrees of *virtualization…*

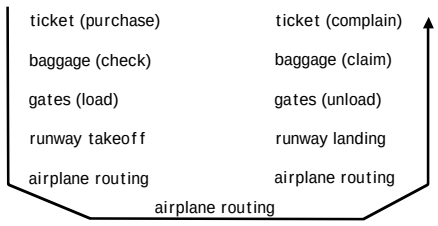
- ***Circuit***: Telephone system
  - **Path & resources reserved before data is sent**
  - **Data has no meta-info at all. Only timing!**
- ***Virtual Circuit***: ATM networks
  - **Multiple virtual circuits mapped to one wire.**
- ***Connection-Oriented***: TCP
  - **Have an association between end-points**
- ***Connectionless/datagram:*** IP, postage service
  - **Complete address on each packet**
  - **The address finds next hop at each routing point**

Formal Framework: *Protocols*

Human protocol vs Computer network protocol:



Analogy: Organization of air travel



- a series of steps

Organization of air travel: *a different view*

ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing
airplane routing	

Layers: each layer implements a *service*
- via its own internal-layer actions (I.e. *technology*)
- *relying on* services provided by layer below

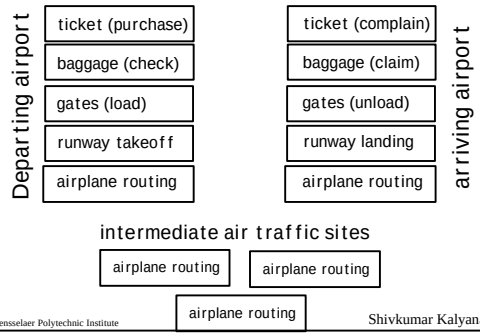
Layered air travel: *services*

- Counter-to-counter delivery of person+bags
- baggage-claim-to-baggage-claim delivery
- people transfer: loading gate to arrival gate
- runway-to-runway delivery of plane
- airplane routing from source to destination

So, why *layering*?

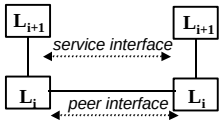
- *Explicit structure* allows *identification, relationship of complex system's pieces*
  - layered reference model
- *Modularization* eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- Layering considered *harmful*?

Distributed implementation of layers

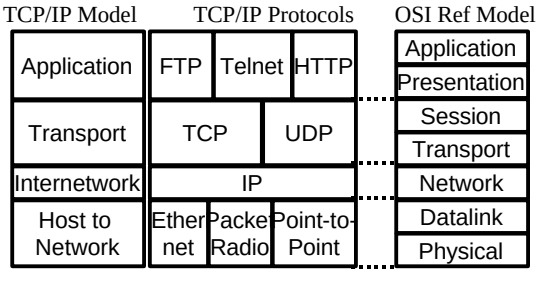


Formal Framework: *Protocols*

- 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



Reference Models for Layering



Where did the problems these layers solve spring up from ?

Formal Framework: *Interface Design*

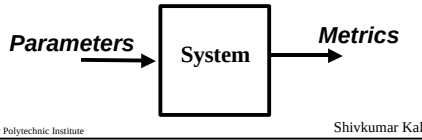
- Interface between layers is also called the "***architecture***"
  - Use abstractions to hide complexity
  - Allows a subroutine abstraction between a layer and its adjacent layers.
- Interface design crucial because *interface outlives the technology* used to implement the interface.

Formal Framework: *Interface Design*

- 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 views of performance ..

Performance evaluation

- **Performance questions*:***
  - Absolute: ***How fast*** …
  - Relative: Is A ***faster than*** B and ***how much faster***?
- **Define system as a black box.**
  - Parameters: input; Metrics: output
- Parameters: only those the system is *sensitive to*
- Metrics: must *reflect the system design tradeoff*



Effect on Design: *Amdahl's law*

- Performance after improvement =
  Performance ***affected*** by improvement ***/*** speedup ***+ Unaffected*** performance
  - ***Lesson:*** *Speedup the common case I.e. the parts that matter most* !!
- *Amdahl's law guides the definition of tradeoffs, parameters, test cases and metrics !*

Perspectives on Performance/Design

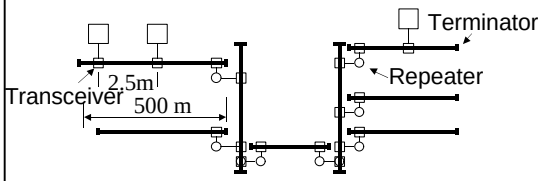
- ***Network users:*** services and performance that their *applications need*,
- ***Network designers:*** *cost-effective design*
- ***Network providers:*** system that is easy to *administer and manage*
  - Need to balance these three needs

Review: *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.

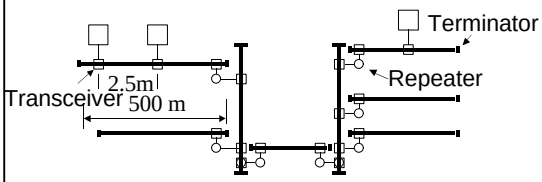
10Base5 Ethernet *Cabling* Rules

- 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"



10Base5 *Cabling* Rules (Continued)

- No more than 2.5 m between stations
- Transceiver cable limited to 50 m



Inter-connection Devices

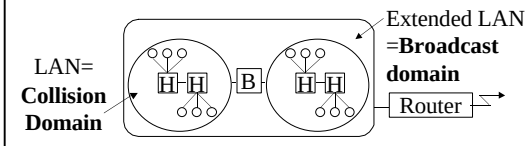
- **Repeater:** Layer 1 (PHY) device that restores data and collision signals: a digital amplifier
- **Hub:** Multi-port repeater + fault detection
  - Note: broadcast at layer 1
- **Bridge:** Layer 2 (Data link) device connecting two or more *collision domains*.
  - MAC multicasts are propagated throughout "*extended LAN*."
  - Note: Limited filtering and forwarding at layer 2

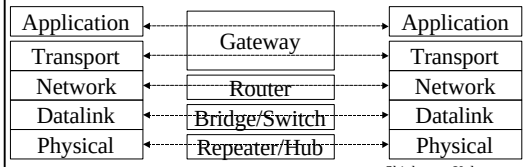
Interconnection Devices (Continued)

- **Router:** Network layer device. IP, IPX, AppleTalk. Interconnects *broadcast domains*.
  - Does not propagate MAC multicasts.
- **Switch:**
  - Key: has a switch fabric that allows parallel forwarding paths
  - *Layer 2 switch:* Multi-port bridge w/ fabric
  - *Layer 3 switch:* Router w/ fabric and per-port ASICs

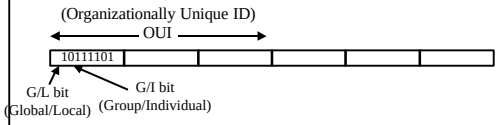
These are functions. Packaging varies.

Interconnection *Devices*



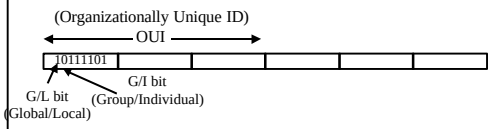


Ethernet (IEEE 802) *Address Format*



- 48-bit flat address => no hierarchy except for administrative purposes
  - Assumes that all destinations are (logically) directly connected.
  - *Address structure does not explicitly acknowledge indirect connectivity*

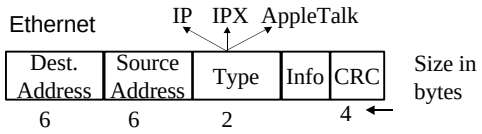
Ethernet (IEEE 802) *Address Format*



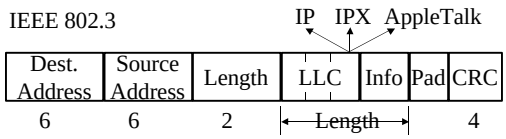
- G/L bit: *administrative*
  - Global: unique worldwide; assigned by IEEE
  - Local: Software assigned
- G/I: bit: *multicast*
  - I: unicast address
  - G: multicast address. Eg: "To all bridges on this LAN"

Ethernet & 802.3 *Frame Format*

- Ethernet



- IEEE 802.3



Review: 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

Review: PPP

- *Point-to-point protocol*
- 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

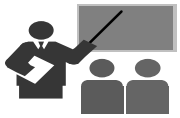
Review: PPP (Continued)

- 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

Review: *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



- Basics: Direct/indirect connectivity, topologies
- Link layer issues:
  - Framing, Error control, Flow control
- Multiple access & Ethernet:
  - Cabling, Pkt format, Switching, bridging vs routing
- Internetworking problems: Naming, addressing, Resolution, fragmentation, congestion control, traffic management, Reliability, Network Management