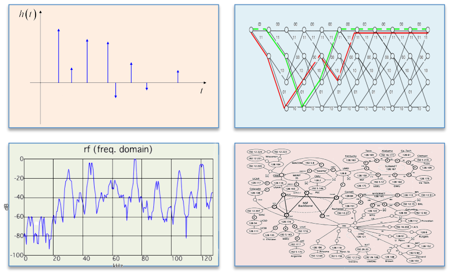


INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

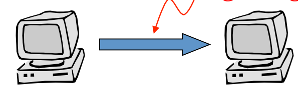
6.02 Spring 2011
Lecture #18

- multi-hop networks: design criteria
- network topologies
- circuit vs. packet switching
- queues, Little's Law

Next Steps…

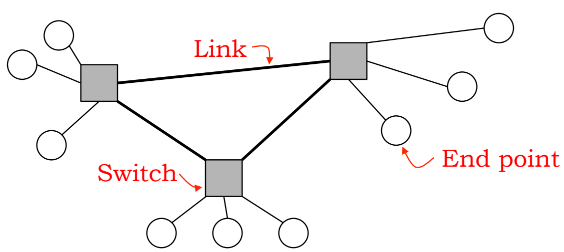
- Have: digital point-to-point

We've worked on link signaling, reliability, sharing



- Want: many interconnected points

Multi-hop Networks



- Switches orchestrate flow of information through the network, often multiplexing many logically-independent flows over a single physical link.
  - Appropriate sharing of network resources
  - Circuit switching vs. packet switching
- Design criteria
  - Reliability, scalability, performance, cost, security, …

Network Reliability

- Design engineers:
  - Low MTBF components, failure prediction
  - Easy to identify and fix problems
    - Remote observability and controllability
  - Replace/expand/evolve network incrementally
  - Defend against malicious users
- Redundancy
  - No single point of failure
  - Fail-safe, Fail-safe, Fail-hard
  - Automated adaptation to component failures
- Degradation, not failure
- Users:
  - High availability
  - Meaningful feedback on failure (e.g., busy signal)

Network Scalability

- Enable incremental build-out
  - increase in usage involves incremental costs (both at edges and in interior of network)
  - Address bottlenecks without fundamental changes
- Economies of scale
  - Larger number of users → less cost/user
  - "lose money on each customer, but make it up in volume"
- Slow growth of scale factors (N = number of users)

$2^N > N^2 > N \log(N) > N > \log(N) > \text{constant}$

Network Performance

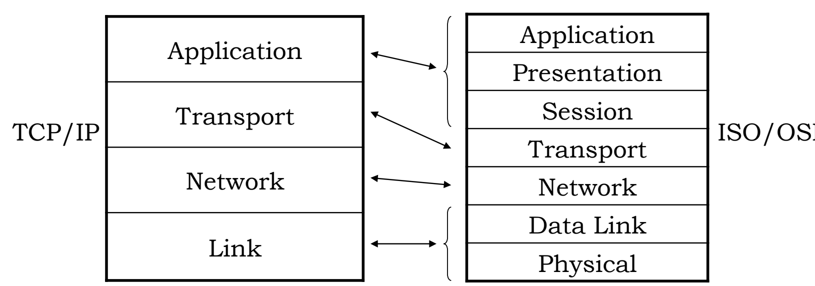
- Design Engineers
  - Utilization
  - Minimize protocol overhead
  - Quality of Service (performance tiers)
- User
  - Throughput (guaranteed minimum)
    - Opportunistic improvements
  - Latency (guaranteed maximum)
    - One-way, round-trip
  - Isochrony?

Network Costs

- NRE (non-recurring expenses, ie, one-time costs)
- Basic infrastructure
- Per connection
- Per message transported
- Economies of scale, amortization

Dealing with System Complexity

- Manage complexity with abstraction layers
  - Easier to design if details have been abstracted away
  - Desired behavior implemented using only component behaviors (not component implementation details), but check that correct behavior actually happened!
  - Change implementation without changing behavior
- Communication Network Layers



Abstraction Layers

Application Layer:
HTTP, Mail, FTP, Telnet, RPC, DNS, NFS

Transport Layer (eg, TCP or UDP):
Using packets to create illusion of continuous stream of data. In-order, connection-based vs. connectionless

Network Layer (eg, IP):
Addressing and routing fixed-length datagrams through a multi-hop network, breaking up larger packets

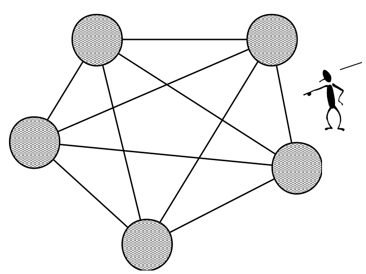
Link Layer:
Channel access, how bits are transmitted on the channel, framing of data, channel-specific addressing, channel coding.

Network Topologies

Networks built from two types of channels:

- Point-to-point channels
  - Simplex: one-way communication
  - Half-duplex: bidirectional, but one-way at a time
  - Full-duplex: simultaneous bidirectional communications
    - Symmetric and asymmetric bandwidths
- Multiple-access channels (eg, wireless, ethernet)
  - Sharing mechanism:
    - Regimented: Fixed allocations in time or frequency
    - Ad-hoc: Contend for use, eg, collision detect/back-off
- Local-area (LANs) vs. wide-area (WANs)
  - Appropriate technologies? topologies?

Fully-Connected Graph

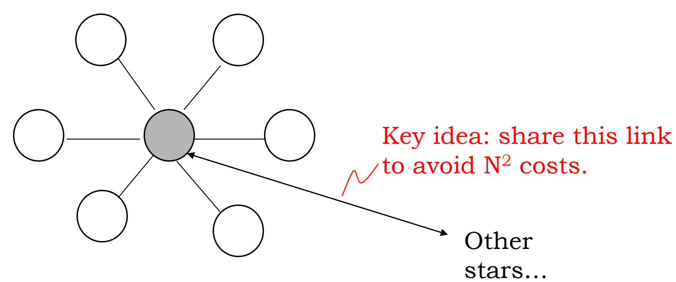


Asymptotic analysis: Each point-to-point link requires 1 unit of hardware, each communication, 1 unit of time

- Throughput: $O(N^2)$
- Latency: O(1)
- Cost: $O(N^2)$ overall, O(N) per node
- no single points of failure, many alternate paths
- no blocking/congestion
- Expensive, doesn't easily scale to large regions

**But in the real world, hardware lives in 3-space, so N nodes take at least $O(N^{1/3})$ space which means that comm. time grows as $O(N^{1/3})$*

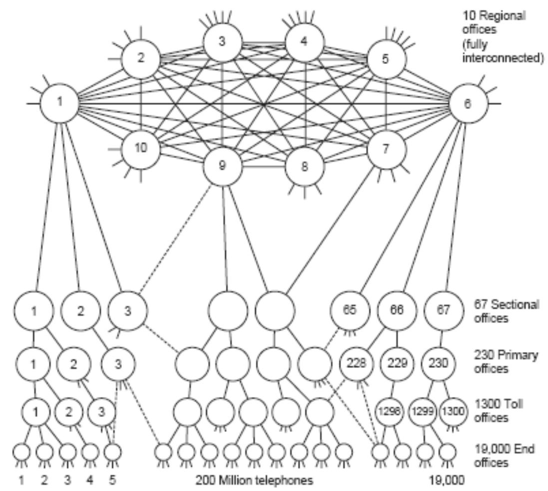
Star



- Throughput: O(N)
- Latency: O(1)*
- Cost: O(N) for center node, but only O(1) for leaf node
- single point of failure!
- congested link affects all communications to that node
- Inexpensive, long links possible

Example: Telephone Network

ATT North America, c. mid-1990's



Wide Area Networks

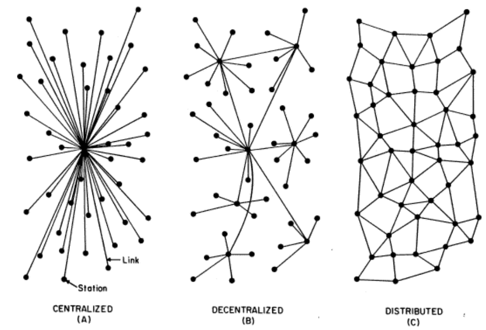
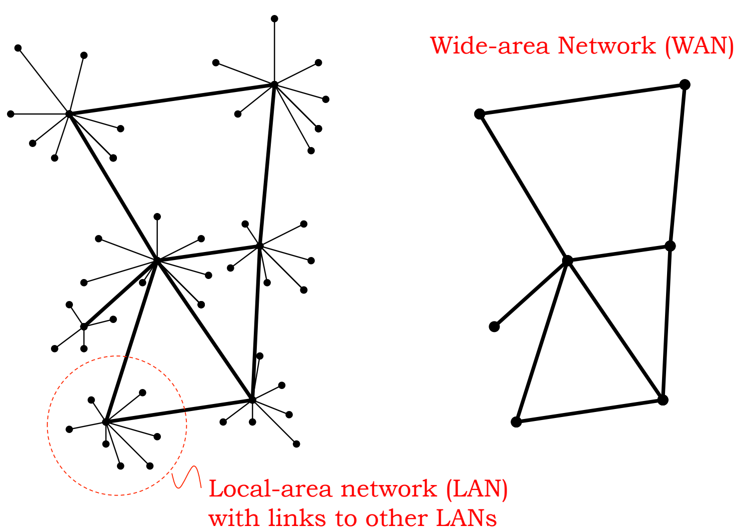


FIG. 1 — Centralized, Decentralized and Distributed Networks

The pioneering research of Paul Baran in the 1960s, who envisioned a communications network that would survive a major enemy attack. The sketch shows three different network topologies described in his RAND Memorandum, "On Distributed Communications: 1. Introduction to Distributed Communications Network" (August 1964). The distributed network structure was judged to offer the best survivability.

Modern Networks: LANs + WANs



Sharing the internetwork links

Consider a single node in our wide-area network:

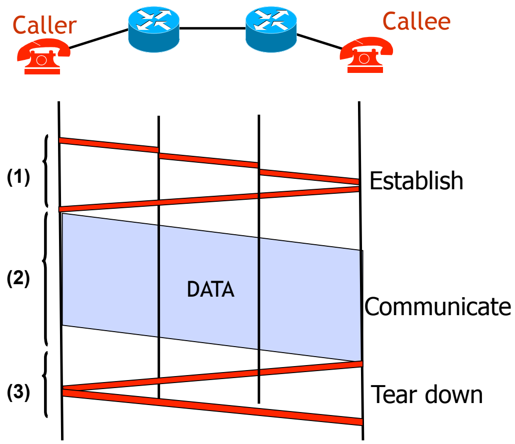
We have many (low-bandwidth) application-level connections that need to mapped onto a small number of (high-bandwidth) channels. Some originate/terminate at the current node, some are just passing through on their way to another node.

How should we share the link between all the connections?

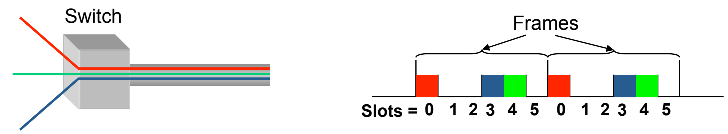
Circuit switching (isochronous)
Packet switching (asynchronous)

Circuit Switching

- First establish a *circuit* between end points
  - E.g., done when you dial a phone number
  - Message propagates from caller toward callee, establishing some state in each switch
- Then, ends send data ("talk") to each other
- After call, *tear down* (close) circuit
  - Remove state



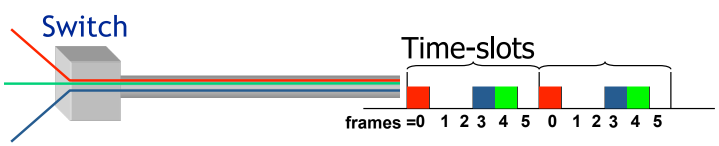
Multiplexing/Demultiplexing



One sharing technique: time-division multiplexing (TDM)

- Time divided into frames and frames divided into slots
  - Number of slots = number of concurrent conversations
- Relative slot position inside a frame determines which conversation the data belongs to
  - E.g., slot 0 belongs to the red conversation
  - Mapping established during setup, removed at tear down
- Forwarding step at switch: consult table

TDM Shares Link Equally, But Has Limitations

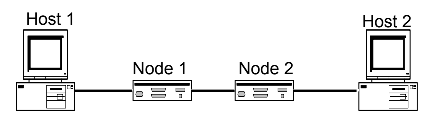


- Suppose link capacity is C bits/sec
- Each communication requires R bits/sec
- #frames in one "epoch" (one frame per communication) = C/R
- Maximum number of concurrent communications is C/R
- What happens if we have more than C/R communications?
- What happens if the communication sends less/more than R bits/sec?

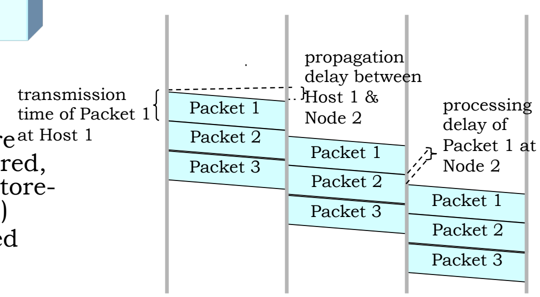
→ Design is unsuitable when traffic arrives in *bursts*

Packet Switching

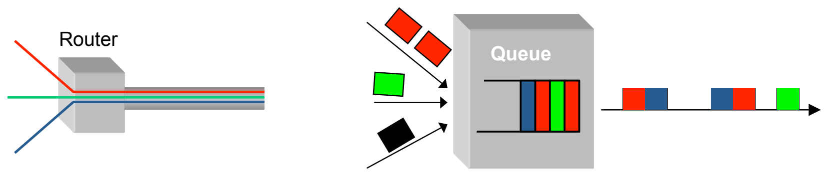
- Used in the Internet
- Data is sent in packets (header contains control info, e.g., source and destination addresses)



- Per-packet routing
- At each node the entire packet is received, stored, and then forwarded (store-and-forward networks)
- No capacity is allocated



Packet Switching: Multiplexing/Demultiplexing



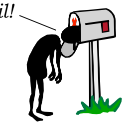
- Router has a routing table that contains information about which link to use to reach a destination
- For each link, packets are organized using a queue
  - If queue is full, packets will be dropped
- Demultiplex using information in packet header
  - Header has destination

"Best Efforts" Delivery

No Guarantees!

- Each packet is individually routed
  - May arrive at final destination in any order
- No time guarantee for delivery
  - Delays through the network vary packet-to-packet
- No guarantee of delivery at all!
  - Packets get dropped (due to corruption or congestion)
  - Use Acknowledgement/Retransmission protocol to recover
    - How to determine when to retransmit? Timeout?
- If packet is re-transmitted too soon → duplicate

Sounds like the US Mail!



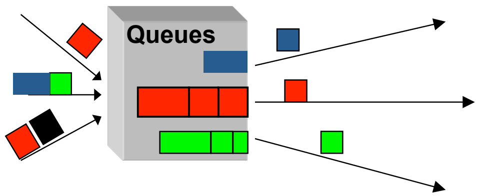
Comparison of Two Techniques

Circuit switching	Packet Switching
Guaranteed capacity	No guarantees (best effort)
Capacity is wasted if data is bursty	More efficient
Before sending data establishes a path	Send data immediately
All data in a single flow follow one path	Different packets might follow different paths
No reordering; constant delay; no dropped packets	Packets may be reordered, delayed, or dropped

We'll Explore Packet Switching

- Commonly used for data networks
  - Data traffic is "bursty", fixed BW allocation isn't best
  - makes most efficient use of communications link
- Routing: choose paths through network
  - "best" route changes dynamically
  - To preserve scalability prefer distributed, decentralized management of routing choices
- Transport: Deal with "best efforts" deficiencies via higher level acknowledgement/retransmission protocol
  - Greatly simplifies engineering at packet level
  - Example of end-to-end argument in engineering

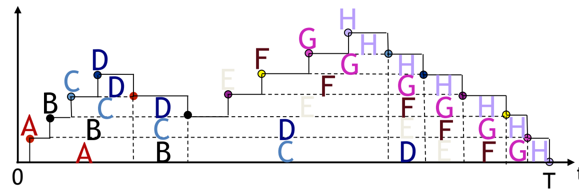
Queues are Essential



- Queues manage packets between arrival and departure
- They are a "necessary evil"
  - Needed to absorb bursts
  - But they add delay by making packets wait until link is available
- So they shouldn't be too big

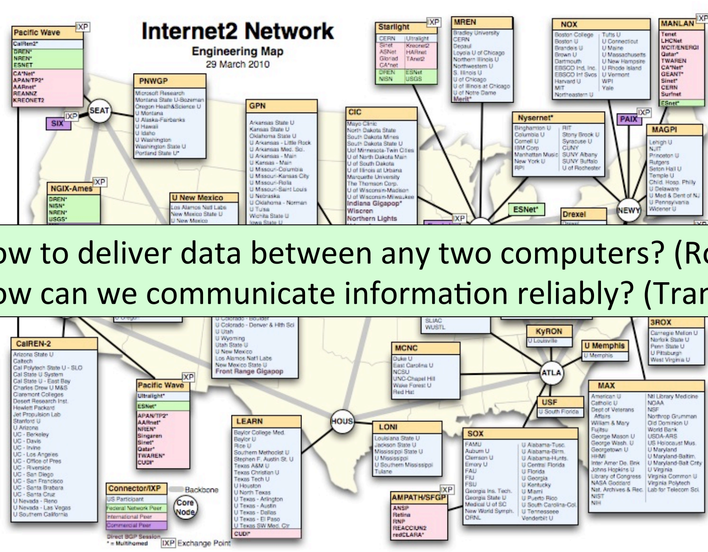
Little's Law

n(t) = # pkts at time t in queue

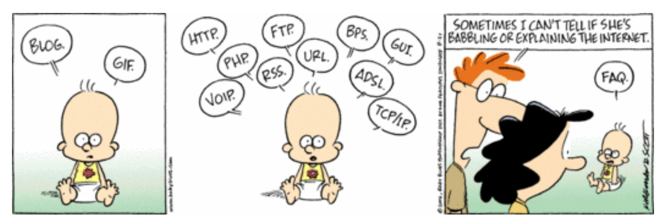


- P packets are forwarded in time T (assume T large)
- Rate = λ = P/T
- Let A = area under the n(t) curve from 0 to T
- Mean number of packets in queue = N = A/T
- A is aggregate delay weighted by each packet's time in queue. So, mean delay D per packet = A/P
- Therefore, **N = λ D** ← Little's Law
- For a given link rate, increasing queue size increases delay

Multi-Hop Packet Networks



How to deliver data between any two computers? (Routing)
How can we communicate information reliably? (Transport)



Baby Blues (Kirkman/Scott)