6.033 Computer System Engineering
Spring 2009

6.033 Lecture 9: Networking Introduction

ideas for more content
  load-shedding / feedback options (pricing &c)
  why did the internet approach win?
    what was the other plan?
  voice / video on internet actually works
    despite best effort non-guarantee
    why? video is elastic.
    voice actually a lot harder despite low bitrate
  why QoS isn't needed (elastic adaptive applications?)

burstyness and flat-rate fixed-rate pipes make pricing complex
  my dsl company sells one way, buys in bulk another
  they take on a big risk -- what if i start downloading 24 hours per day?

old internet maps
  http://www.nthelp.com/maps.htm
  MCIWorldcom 2000 most hair-raising

First of 5 lectures on data networks.
  Overview today: identify the problems and design space
  Dig into details over next four lectures.
  * network are a useful systems building block
  * internal workings are a good case study of system design.
    Internet in particular an example of a very successful system.
    complex enough to be a subject of science, like the weather

What is the goal of data networking?
  Allow us to build systems out of multiple computers.
  Your problem may be too big for one computer.
  You may want multiple computers for structural reasons: client/server.
  more fundamental reason:
    A key use of computers is for communication between people, orgs.
    People are geographically distributed, along with their computers.
  So we're forced to deal with inter-computer comms.

System picture:
  Hosts. Maybe millions of them. All over the world.
  [No cloud yet, but leave room. circular hosts.]

What are the key design problems to be solved?
  I'll classify into three types:
  O: Organizational, to help human designers cope.
  E: Economic, to save operators money.
  P: Physics.

E: universality.
  Want one net, many uses
    Rather than lots of small incompatible/disconnected network worlds.
      e.g. Fedex builds its own data net for its customers to track packages
      private nets sometimes good for reliability, predictable service,
security
    The more people that use a network, the more useful it is for everybody.
    Value to me is proportional to N people using the same net.
    Value to society is proportional to N^2.

```
  What technical tools help us achieve universality?
    Standard protocols make it easy to build systems.
    But don't want to prevent evolution by freezing design w/ standards.
    Hard: standardize just what's required to make net generally useful,
      but not the things that might need to be changed later.
  One universal net means it's *not* part of each system design.
  It's a black box; simplifies design of systems that use it.
  Symbolically, a cloud that hides internal workings.
  [draw network cloud]

E: topology.
  [three new pictures; circular hosts, square switches]
  Look inside the cloud.
  Wire between every pair?
    pro: network (wires) is transparent, passes analog signals.
      It's never busy. Never any question of how to reach someone.
    con: host electronics. laying wires expensive.
  Star network. "switch". Federal Express in Memphis.
    pro: less wire cost. easy to find a path.
    con: load and cost and unreliability of hub.
  Mesh topology. Routers and links.
    pro: low wire cost. limited router fan-out. some path redundancy.
    con: shared links, higher load. complex routing.

O: routing.
  Find paths given dst "address".
  Harder in mesh than in star or hierarchy.
    [diagram: two available paths]
    change far away may require re-routing
    -> has a global element, thus scaling problems
  many goals:
    Efficiency: low delay, fast links
    Spread load
    Adapt to failures
    Minimize payments
    Cope w/ huge scale
  Routing is a key problem.

O: addressing.
  Hosts need to be able to specify destination to network.
  specifically, the address you give to the routing system
    18.7.22.69, not web.mit.edu
  ideal:
    every host has a unique ID, perhaps 128 bits assigned at random
    routing system can find address, wherever it is
    so i can connect to my PDA, whether i left it at home or in the office
  no-one knows how to build a routing system for large #s of flat addresses!
    maybe can layer on top of something else, but not directly
  In practice, encode location hints in addresses.
    hierarcical addresses, e.g. 18.7.whatever
    [diagram: 18 area, 18.7 area, ...]
    rest of inet knows only 18, not every host in 18
    Trouble if hosts move or topology changes.
    hard to exploit non-hierarchical links (rest of Inet can't use MIT-
Harvard)
  routing and addressing often very intertwined
```

E: sharing.
  Must multiplex many conversations on each physical wire.
  1. how to multiplex?
     A. isochronous.
       [input links, router1, router2, link, repeating cycle]
       reserved b/w, predictable delay, originally designed for voice
       called TDM
     B. asynchronous
       data traffic tends to be bursty - not evenly spaced
         you think, then click on a link that loads lots of big images
       wasteful to reserve b/w for a conversation
       so send and forward whenever data is ready
       [diagram: input links, router1, link, router2, output links]
       divide data into packets, each with header w/ info abt dst
  2. how to keep track of conversations?
     A. connections
        like phone system
        you tell network who you want to talk to
        figures out path in advance, then you talk
        reqired for isochronous traffic
        can allow control of load balance for async traffic
        maybe allow smaller packet headers (just small connection ID)
        connection setup complex, forwarding simpler
     B. connectionless / datagrams
        many apps don't match net-level connections, e.g. DNS lookups
        each packet self-contained, treated separately
        packet contains full src and dst addresses
        each may take a different route through network!
        shifts burden from network to end hosts (connection abstraction)

E: Overload
  more demand than capacity
  consequence of sharing and growth and bursty sources
  how does it appear?
  isochronous net:
    new calls blocked, existing calls undisturbed
    makes sense if apps have constant unchangeable b/w requirements
  async net:
    [diagram: router with many inputs]
    overload is inside the net, not apparent to sending hosts.
    net must do something with excess traffic
    depends on time scale of demand > capacity
    very long: buy faster links
    short: tell senders to slow down
      feedback
      elastic applications (file xfer, video with parameterized compression)
      can always add more users, just gets slower for everyone
      often better than blocking calls
    very short: don't drop -- buffer packets -- "queuing"
      demand fluctuates over time
      [graph: varying demand, line for fixed link capacity]
      buffers allow you to delay peaks into valleys
      but only works if valley comes along pretty quick!
      adds complexity. must drop if overload too much. source of most Inet
loss.

Behavior with overload is key.
[Graph of in rate vs out rate, knee, collapse.]
Collapse: resources consumed coping with overload.
This is an important and hard topic: congestion control

E: high utilization.
How much expensive capacity do we have to pay for?
Capacity >= peak demand would keep users happy.
What would that mean?
single-user traffic is bursty!
[typical user time vs load graph -- avg low, peak high]
Worst case is 100x average, so network would be 99% wasted!
e.g. my one-megabit DSL line is almost entirely idle
too expensive in core to buy Nusers * peak
But when you aggregate async users -- peaks and valleys cancel.
[a few graphs, w/ more and more users, smoother and smoother]
peak gets closer and closer to average
100 users in << 100x capacity needed for 1 user.
less and less idle capacity, lower cost per bit transmitted
"Statistical multiplexing."
Hugely important to economics of Internet
Assumes independent users
mostly true -- but day/night, flash crowds
50% average daytime utilization maybe typical on core links

Once you buy capacity, you want to get as close to 100% as possible.
fixed cost, want to maximize revenue/work
may need feedback to tell senders to speed up!
want to stay at first knee of in/out graph.
Or maybe less to limit delay
[graph: load vs queuing delay]

P: errors and failures.
Communication wires are analog, suffer from noise.
Backhoes, unreliable control computers.
General fix: detect, send redundant info (rxmt), or re-route.
How to divide responsibility?
You pay your telecommunications provider, so they better be perfect?
Leads to complex, expensive networks: each piece 100% reliable.
You don't trust the network, so you detect and fix problems yourself.
Leads to complex host software.
Decision of smart network vs smart hosts turns out to be very important.
Locus of complexity pays costs but also has power.
"Best design" depends on whether you are operator or user.

P: speed of light.
foot per nanosecond.
14 ms coast to coast, maybe 40 ms w/ electronics.
Request/response delay. Byte per rtt -> 36 bytes/second.
[picture of pkt going over wire -- and idle wire]
Need to be more efficient.
Worse: delay hurts control algorithms. Change during delay.
Example: Slow down / speed up.
Result: Oscillation between overload and wastefully idle.

O: wide range of adaptability.

One design, many situations.
Applications: file xfer, games, voice, video.
Delay: machine room vs satellite.
Link bit rate: modem vs fiber optics.
Management: a few users, whole Internet.
Solution: Adaptive mechanisms.

Internet in design space
Asynchronous (no reservations)
Packets (no connections)
No b/w or delay guarantees
May drop, duplicate, re-order, or corrupt packets -- and often does
Not of much direct use! End hosts must fix
but gives host flexibility, room for innovation
"best effort"

Next lecture: start to talk about solutions.