The Internet Architecture [Clark88a]

CSci551: Computer Networks
SP2006 Thursday Section
John Heidemann

Key ideas

- motivation behind Internet architecture
  - why things are they way they are
  - hints and some alternative designs (and why they weren't taken)
- not complete
  - main goal of internet
    - building on existing networks
  - secondary goals
    - reliability / availability
  - approaches
    - starting with simple building blocks
    - datagram

What is the Internet?

- *inter*net: composition of many small neteworks
- a common network protocol IP
- multiple protocols: TCP, UDP, etc.
- multiple services: HTTP (web), SMTP (mail)
- dynamic routing protocols
  - routes can change automatic (different links)
  - routes change do to network conditions
    - change due to SOME network conditions
      - link failures
      - evalute and change to congestion infrequently: traffic engineering
    - not so much due to congestion

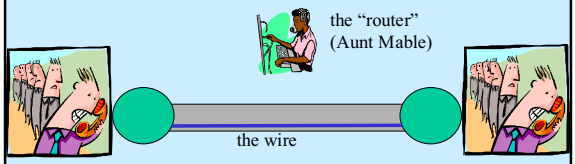
The Internet Architecture

- primary goals:
  - connecting heterogeneous networks
  - packet switching
- secondary goals:
  - robusteness
  - (rest are in list in paper)

Main Goals

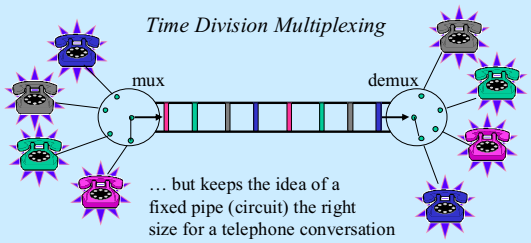
- heterogeneous link-layers
  - why? already out there (built on telephone network), and wanted to let things evolve
  - some examples: wireless 802.11, Ethernet, X.25, optical, satellites, token rings, …
- multiplexing
  - packet switching fundamentally different from circuit switching

Back in the Old Days...



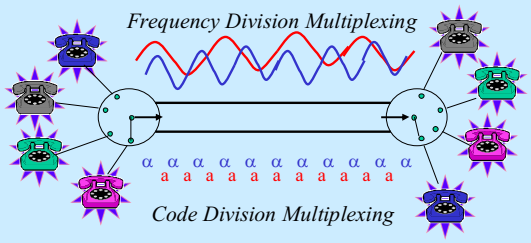
1920s telephony: *circuits*---a physical wire from one end to the other

Then Came TDM...

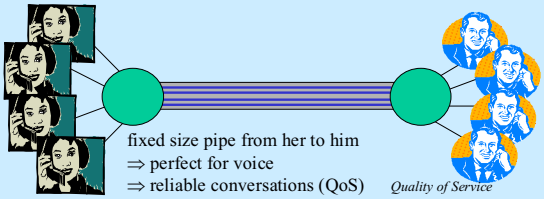


… but keeps the idea of a fixed pipe (circuit) the right size for a telephone conversation

And FDM and CDM...

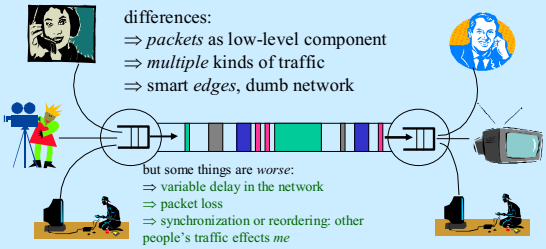


Logical Network View



fixed size pipe from her to him
⇒ perfect for voice
⇒ reliable conversations (QoS) *Quality of Service*
⇒ provisioning, good engineering
⇒ dumb & cheap end points, smart network
⇒ evolved for 100 years (analog to digital)

Packet Switching (Internet)



differences:
⇒ *packets* as low-level component
⇒ *multiple* kinds of traffic
⇒ smart *edges*, dumb network

but some things are *worse*:
⇒ variable delay in the network
⇒ packet loss
⇒ synchronization or reordering: other people's traffic effects *me*
⇒ much smarter end-host => more expensive end host

Statistical Multiplexing Gain

Assumptions:
1 Mb/s link
user: 0.1Mb/s when transmitting, but 10% duty cycle

- Circuit switching: can support 10 users, 100% reliable
- Packet switching: with 35 users, probability that ≥10 are transmitting at the same time = 0.0004

$$p = .1, N = 35, T = 10$$

$$Pr[n \geq T] = 1 - \sum_{i=0}^{i=T} \binom{N}{i} p^i (1-p)^{(N-i)}$$

a. Robust to Failures

- app should not see transient failures
- what kinds of failures?
  - (talked about earlier today)
- how does anything still work?
  - fate-sharing
    - soft-state
    - if we put state just at the ends, and we lose and end, we don't care
  - end-to-end argument
    - saving state at the end hosts
  - (replication)

b. Multiple Types of Service

- originally just NCP, but split to {TCP,UDP}/IP
- why?
  - to get different types of service
  - because different apps need different things from the network
    - ex: with voice, you don't want relibability because you can just ask the other person to repeat
    - downside of reliability: delay
- what?
- made different protocols
- TCP
  - reliable, end-to-end, connection oriented, byte stream
- UDP
  - unreliable, end-to-end, connectionless, packet

Other protocols (than TCP/UDP/IP)?

- RTP: Real Time Protocol
  - checksum, connectionless (?), standard frame of timing information
- RTSP: Real Time Streaming Protocol
- SCTP: Stream Control Transmission Protocol
  - streaming like TCP, but without reliability
- application protocols (above transport)
  - SIP: not really transport
  - HTTP, …

Non-TCP/UDP protocols

- RDP: Reliable Delivery Protocol
  - message-based
  - allows out-of-order delivery
  - RFC-908
- SCTP: Stream Control Transmission Protocol
  - intended for telephony signaling over IP
  - multiplexes multiple "streams" per connection
  - in-sequence per stream, out-of-sequence between streams
  - reliable
  - RFC-3286
- DCCP: Datagram Congestion Control Protocol
  - add congestion control to UDP
  - in progress
- TFRC: TCP-Friendly Rate Control
  - defines how to do rate-control as a function of loss rate
  - RFC-3448
  - we study TCP-friendliness later
- XCP: Explicit Control Protocol
  - high-speed streaming with explicit router rate feedback
  - presented at SIGCOMM
  - we'll study it later

b.2. multiple *applications*

- classes:
  - interactive multimedia
    - voice conversations
    - [real time]
    - gaming
  - one-way multimedia
    - video streaming
  - bulk data transfer
    - file transfer
    - peer-to-peer
    - e-mail
  - interactive non-multimedia
    - remote login
    - chat
  - [e-commerce]
- requirements:
  - low latency
  - high bandwidth
  - reliablity
  - security
  - jitter
  - interactive vs. background
- this is not a perfect, orthogonal list

c. multiple kinds of networks

- IP over X
- compare to integrated stacks:
  - ISO
  - ATM
  - cell phones (CPDP & WAP, maybe??)
  - fibre channel, Apple Desktop Bus, USB, Firewire
- but a counter example: SCSI and now SCSI over IP
- requirements of X:
  - reasonable size packets
    - but fragmentation and reassembly
  - reasonable reliablity
    - but workarounds
  - addressing
- *non*-requirements of X:
  - reliable, in-order, broadcast, QoS, etc.

d. other goals

- distributed management
  - some work, and today policy routing exists
  - but limitations (ex. address space portability)
- cost effective
  - today quite cheap
  - but for small devices? for keyboard?

d. other other goals

- effort to deploy end host
  - for him in '88: *cost of implementing stack*
  - today: cost of administering machine
    - much lower today (DHCP, etc.)
    - but still lots of manual configuration "futzing"
- accountability
  - tracking resource usage, money, identity of user and reprucussions of bad usage (spam, p2p file sharing, zombies and denial-of-service)
  - basically nothing then
    - today: not much, but today there is authentication (and identity) at many network connections, but not everywhere
    - possibly a major focus of internet research will be look security

Architecture and Implementation

- *realization:* an instance of the Internet class
  - him: 1200b/s modem vs. 1Mb/s LAN
  - today: from sensor net nodes with 8-bit CPUs and 20kb/s radios…to supercomputers
  - today: the Internet can't do X because it is Y
    - ex. can't do System Area Networks over IP because it's too slow, so we need Fibre Channel
    - *alternative:* build a fast Internet realization
  - corollary: not every realization is appropriate for every app
  - also: custom stack will get last 5% of performance, but is it worth it?

TCP Alternative Choices

- byte stream vs. message stream
- flow control
- congestion control came later
- PSH flag
  - a weak record boundry
  - but *the* reason the Plan-9 people didn't use TCP

Other Components of IP Success

- a good, free implementation
  - BSD Unix in the mid-80's
  - compare to OSI where impls were late
- a good API
  - BSD socket API
  - not perfect, but good
  - compare to OSes where Unix and Windows have very different APIs to open/rename/etc. files

Where are we now?

- this paper: The Internet in 1988 (!)
- much has changed since then (as discussed next in [Deering98a]
- what are the big challenges *today?*
  - think about this question… we'll come back to it

Other questions/observations?

- xxx