Internet Architecture and Assumptions

David Andersen
CMU Computer Science

Course status

- 27 registered (goal: 24)
- 24 on waitlist (goal: 0)
- So – still not looking so good.
  - If you're dropping, remember to actually drop!
- Remember: Project groups!

Internet Architecture

- Background
  - "The Design Philosophy of the DARPA Internet Protocols" (David Clark, 1988).
- Fundamental goal: Effective network interconnection
- Goals, *in order of priority:*
  1. Continue despite loss of networks or gateways
  2. Support multiple types of communication service
  3. Accommodate a variety of networks
  4. Permit distributed management of Internet resources
  5. Cost effective
  6. Host attachment should be easy
  7. Resource accountability

Priorities

- Technical Lessons
  - Packet switching
  - Fate Sharing/Soft state
- The effects of the order of items in that list are still felt today
  - E.g., resource accounting is a hard, current research topic
- Let's look at them in detail

Fundamental Goal

- "technique for **multiplexed utilization** of **existing interconnected networks**"

- **Multiplexing** (sharing)
  - Shared use of a single communications channel
- **Existing networks** (interconnection)
  - Tries to define an "easy" set of requirements for the underlying networks to support as many as possible

Sharing and Multiplexing

- Question #1: How do you avoid an all-to-all network topology?
  - Multiplexing!
  - How can you do it? TDMA, FDMA, CDMA
  - And you can do statistical multiplexing
- Stat mux: Efficient sharing of resources
  - A link can *always* transmit when it has data!

Datagram Switching

- Information for forwarding traffic is contained in destination address of packet
- No state established ahead of time (helps fate sharing)
- Basic building block – must build things like TCP on top
- Pretty much implies statistical multiplexing
- Alternatives:
- **Circuit Switching:** Signaling protocol sets up entire path out-of-band. (cf. the phone network)
- **Virtual Circuits:** Hybrid approach. Packets carry "tags" to indicate path, forwarding over IP
- **Source routing:** Complete route is contained in each data packet

Preview: An Age-Old Debate

- Circuits vs Packets?
- Circuits: Guaranteed QoS, dedicated connection, easy accounting
- Packets: Efficiency, simplicity

It is held that packet switching was one of the Internet's greatest design choices.
Of course, there are constant attempts to shoehorn the best aspects of circuits into packet switching.

Examples: Capabilities, MPLS,ATM, IntServ QoS, etc.

Survivability

- If network disrupted and reconfigured
  - Communicating entities should not care!
  - No higher-level state reconfiguration
  - Ergo, transport interface only knows “working” and “not working.” Not working == complete partition.
- How to achieve such reliability?
  - Where can communication state be stored?

	Network	Host
Failure handing	Replication	“Fate sharing”
Net Engineering	Tough	Simple
Switches	Maintain state	Stateless
Host trust	Less	More

Fate Sharing

- Lose state information for an entity if (and only if?) the entity itself is lost.
- Examples:
  - OK to lose TCP state if one endpoint crashes
    - NOT okay to lose if an intermediate router reboots
  - Is this still true in today’s network?
    - NATs and firewalls
- Survivability compromise: Heterogenous network -> less information available to end hosts and Internet level recovery mechanisms

Types of Service

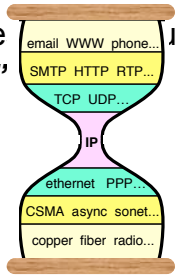
- TCP vs. UDP
  - Elastic apps that need reliability: remote login or email
  - Inelastic, loss-tolerant apps: real-time voice or video
  - Others in between, or with stronger requirements
  - Biggest cause of delay variation: reliable delivery
    - Today's net: ~100ms RTT
    - Reliable delivery can add *seconds*.
- Original Internet model: "TCP/IP" one layer
  - First app was remote login…
  - But then came debugging, voice, etc.
  - These differences caused the layer split, added UDP
- No QoS support assumed from below
  - In fact, some underlying nets only supported reliable delivery
    - Made Internet datagram service less useful!
  - Hard to implement without network support
  - QoS is an ongoing debate…

Varieties of Networks

- Interconnect the ARPANET, X.25 networks, LANs, satellite networks, packet networks, serial links…
- Mininum set of assumptions for underlying net
  - Minimum packet size
  - Reasonable delivery odds, but not 100%
  - Some form of addressing unless point to point
- Important non-assumptions:
  - Perfect reliability
  - Broadcast, multicast
  - Priority handling of traffic
  - Internal knowledge of delays, speeds, failures, etc.
- Much engineering then only has to be done once

So, how do you support them?

- Need to interconnect many existing networks
- Hide underlying technology from applications
- Decisions:
  - Network provide [illegible]nctionality
  - ***"Narrow waist"***



Applications

Technology

Tradeoff: **No assumptions, no guarantees.**

The "Curse of the Narrow Waist"

- IP over anything, anything over IP
  - Has allowed for much innovation both above and below the IP layer of the stack
  - An IP stack gets a device on the Internet
- **Drawbacks:**
  - difficult to make changes to IP
  - But…people are trying (cf GENI)
  - Only a small amount of information available about lower levels. (cf wireless)

Goal #4: Distributed Management

- Independently managed as a set of independent "Autonomous Systems"
  - ISPs
  - CMU
  - Etc.
- BGP (Border Gateway Protocol) connects ASes together
  - Completely (well…) decentralized routing
  - Is this a good thing? (wait two slides)

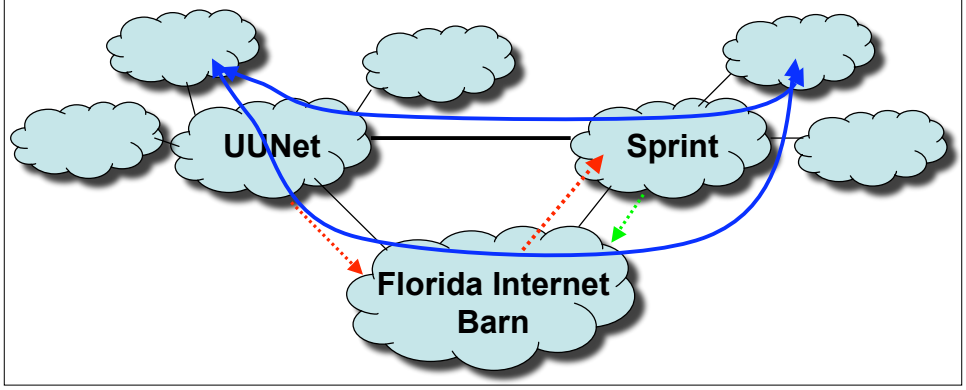
A problem: Management

- **"Some of the most significant problems with the Internet today relate to lack of sufficient tools for distributed management, especially in the area of routing."**
- The Internet is now a hugely complex beast
  - 18,000 constituent networks
  - Routing tables with 1,000,000+ entries
  - Gajillions of $$.
- Management and operational expenses becoming increasingly important
  - Remember that growth chart? Not just more b/w per user, but constantly more users & links

Local Actions, Global Consequences

"…a glitch at a small ISP… triggered a major outage in Internet access across the country. The problem started when MAI Network Services...passed bad router information from one of its customers onto Sprint."

-- *news.com*, April 25, 1997



Goal #5: Cost Effectiveness

- Packet headers introduce high overhead – but so does circuit setup
- End-to-end retransmission of lost packets
  - Potentially wasteful of bandwidth by placing burden on the edges of the network

Arguably a good tradeoff. Current trends are to exploit redundancy even more.

Bandwidth is becoming cheaper in many environments

Goal #6: Ease of Attachment

- IP is "plug and play" Anything with a working IP stack can connect to the Internet (hourglass model)
- A huge success!
  - **Lesson:** Lower the barrier to innovation/entry and people will get creative (*e.g.,* Cerf and Kahn probably did not think about IP stacks on phones, sensors, etc.)

Tradeoff: **Burden on end systems/programmers.**

- But….

Goal #7: Accountability

- Huge problem.
- Accounting
  - Billing? (mostly flat-rate. But phones are moving that way too - people like it!)
  - Inter-provider payments
    - Hornet's nest. Complicated. Political. Hard.
- Accountability and security
  - Huge problem.
  - Worms, viruses, etc.
    - Partly a host problem. But hosts very trusted.
  - Authentication
    - Purely optional. Many philosophical issues of privacy vs. security.

Stopping Unwanted Traffic is Hard

February 2000

March 2006

Some environments challenge the model

- Wireless
- Host mobility
- Ad hoc wireless networks
- Satellite
- Space
- Sensor networks
- Dial-up / store and forward
- Disconnection
- High availability requirements
- No QoS assumed from below
- Reasonable but non-zero loss rates
  - What's minimum recovery time?
    - 1rtt
  - But conservative assumptions end-to-end
    - TCP RTO - min(1s)!
- Interconnect independent networks
  - Federation makes things hard:
    - My network is good. Is yours? Is the one in the middle?
  - Scale
    - Routing convergence times, etc.

Design wrapup

- IP model: Stat mux, datagrams, fate sharing, narrow waist
  - Successes: IP on everything!
  - Drawbacks… but perhaps they're totally worth it in the context of the original Internet. Might not have worked without them!

"This set of goals might seem to be nothing more than a checklist of all the desirable network features. It is important to understand that these goals are in order of importance, and **an entirely different network architecture would result if the order were changed**."

Project Stuff

"These changes in the Internet design arose through the repeated pattern of implementation and testing that occurred before the standards were set" -- ddc, Design Philosophy

The RFCs required "rough consensus and working code" -- also coined by ddc

Some thoughts

- "Simulation is doomed to succeed" -- Rod Brooks (former MIT AI Lab director)
- Simulation depends on what you choose as input parameters.
  - Easy to account for them b/c you set them
  - Easy to be sensitive to things you forgot!
    - ex: wireless routing protocols (later)

Some resources

- See the list on the web page
- Writing: The Elements of Style
- Graphing: The Visual Display of Quantitative Information
- Research: Patterson "Bad Career" talk

Some tools

- Please learn LaTeX if you don't know it already.
- Pick one of gnuplot, ploticus, or jgraph
  - Each has advantages & disadvantages. I use gnuplot for most things and ploticus for really evil complex graphs.
- Script aggressively. Automate your analysis and eval from typing "run" to having graphs.
  - Shorten the design/eval/re-think cycle!
  - Up-front investment in time, but it pays off

Next Time: End-to-end Arguments, ALF

- Read the E2E paper if you haven't; think about the argument it's making.
- Some points to ponder:
- What functions can only be implemented correctly with the help of the endpoints?
- What functions can *not* be implemented without the help of the network?
- ALF: Application needs vs. what the network stack provides