CSCI-1680 - Computer Networks

Rodrigo Fonseca (rfonseca)

http://www.cs.brown.edu/courses/cs168

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti, Peterson & Davie

Cast

- **Instructor: Rodrigo Fonseca (rfonseca)**
- **HTA: Sam Steffl(ssteffl)**
- **UTA: Xueyang Hu (xhu3)**
- **How to reach us: Piazza**

https://piazza.com/brown/fall2016/cs168

Overview

- **Goal: learn concepts underlying networks**
  - How do networks work? What can one do with them?
  - Gain a basic understanding of the Internet
  - Gain experience writing *protocols*
  - Tools to understand new protocols and applications

"From 2 communicating machines to the entire Internet"

Prerequisites

- **CSCI-0330 (or equivalent).**
  - We assume basic OS concepts (kernel/user, threads/processes, I/O, scheduling)
- **Low-level programming or be willing to learn quickly**
  - threads, locking, explicit memory management, …
- **We allow any* language**
  - No high-level networking APIs, though (unless you write them yourself)
  - You will be bit twiddling and byte packing…

Administrivia

- **All assignments will be on the course page**
  http://www.cs.brown.edu/courses/cs168/f16
- **Texts (not required):**
  – Peterson and Davie, Computer Networks - A Systems Approach, 4th or 5th editions *or*
  – Kurose and Ross, 'Computer Networking: A Top-Down Approach (6th or 7th editions)
- **You are responsible to check the web page!**
  – All announcements will be there
  – Textbook chapters corresponding to lectures: read them before class
  – Handouts, due dates, programming resources, *etc…*
  – *Subject to change* (reload before checking assignments)

Grading

- **"Written" component**
  - Exams: Midterm (15%) and Final (25%)
  - Homework: 3 written assignments (15%)
    - Short answer and design questions
- **4 Programming Projects (45%)**
  - Snowcast: streaming music server
  - IP, as an overlay, on top of UDP
  - TCP, on top of *your* IP
  - Final (short, fun, to be decided)
- **Must pass two components individually**

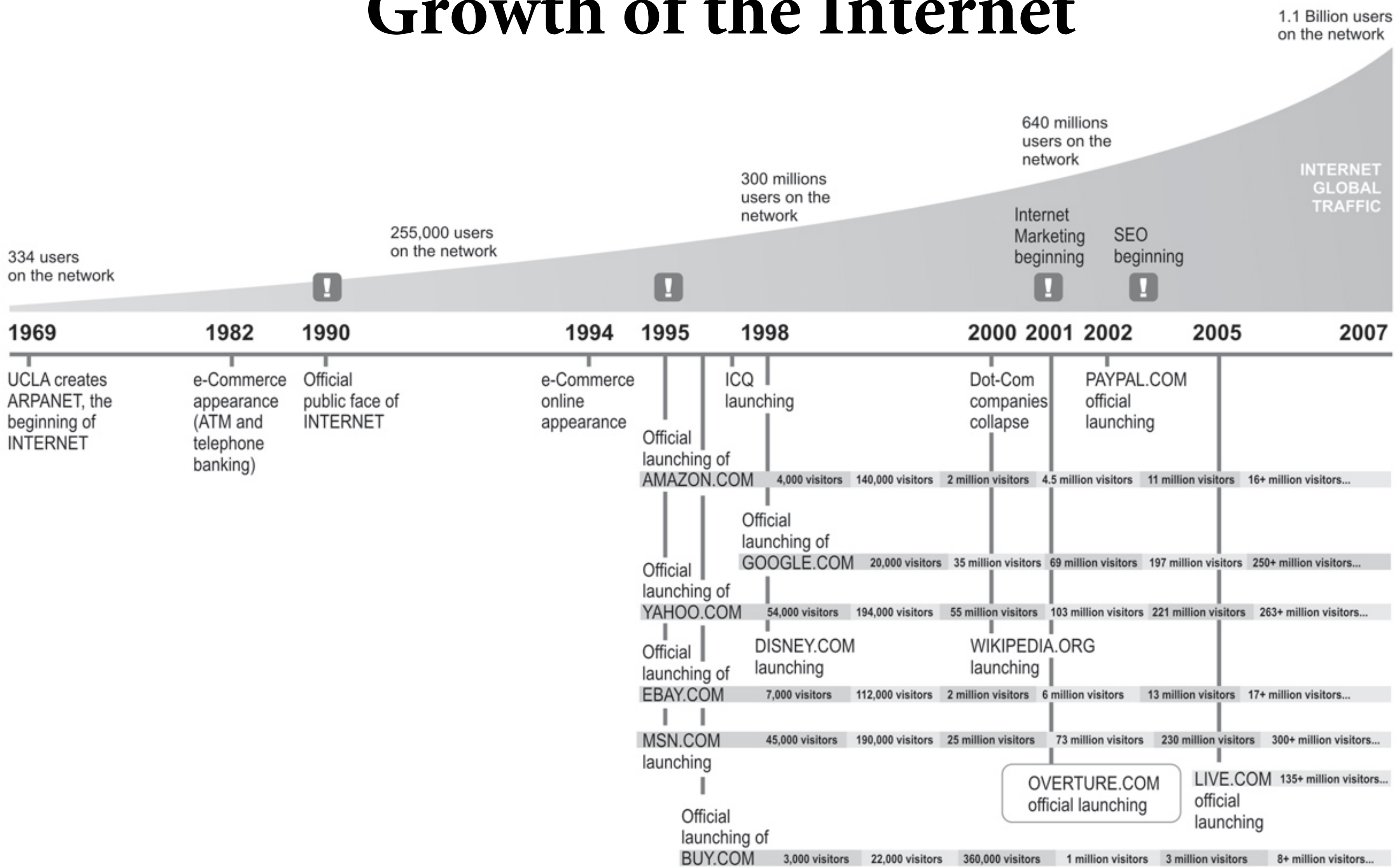
Networks

- **What is a network?**
  - System of lines/channels that interconnect
  - *E.g.*, railroad, highway, plumbing, postal, telephone, social, computer
- **Computer Network**
  - Moves information
  - Nodes: general-purpose computers (most nodes)
  - Links: wires, fiber optics, EM spectrum, composite…

Why are computer networks cooler?

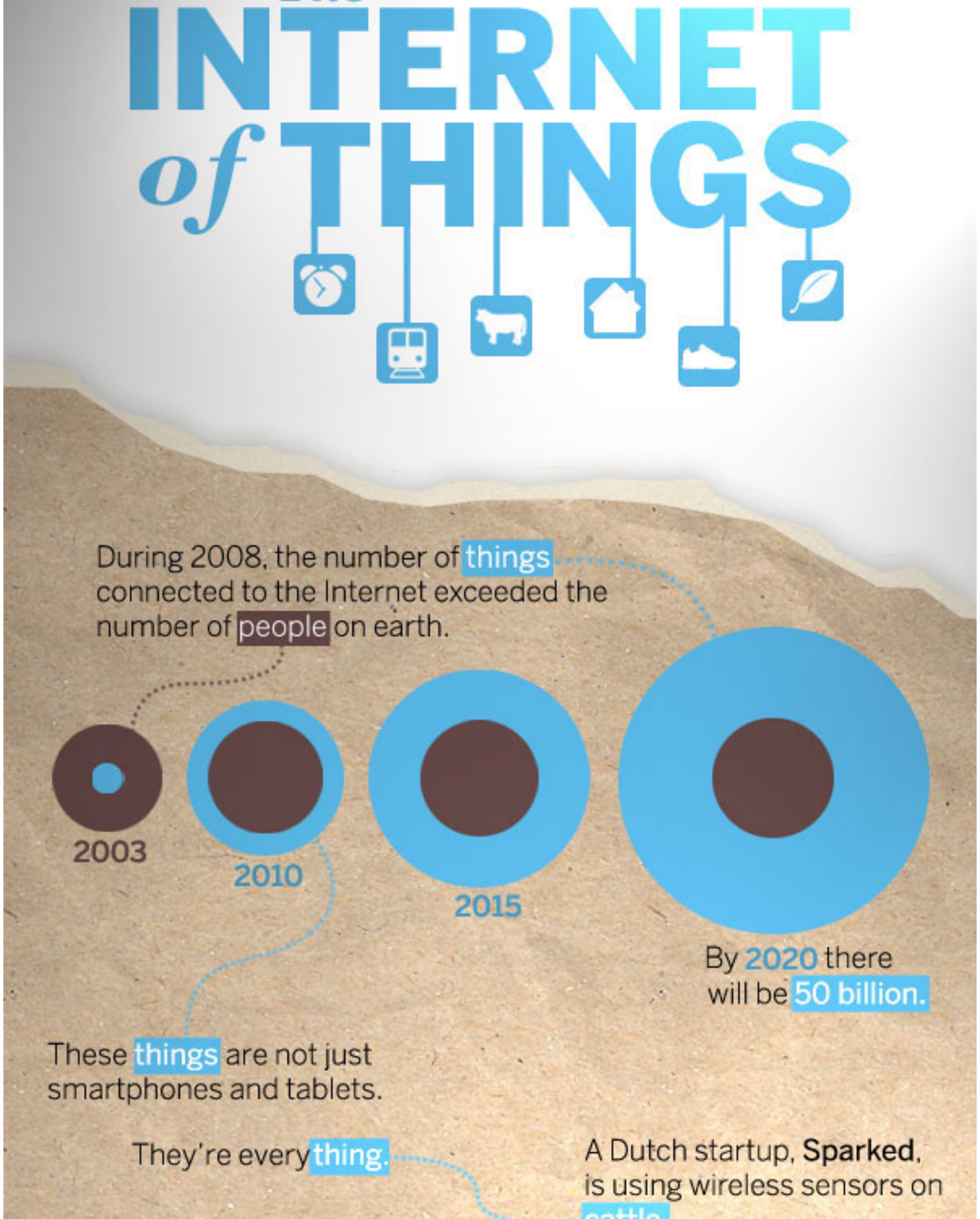
- **Many nodes are general-purpose computers**
- **Very easy to innovate and develop new uses of the network: *you* can program the nodes**
- **Contrast with the ossified Telephone network:**
  - Can't program most phones
  - Intelligence in the network, control by parties vested in the *status quo*, …

Growth of the Internet



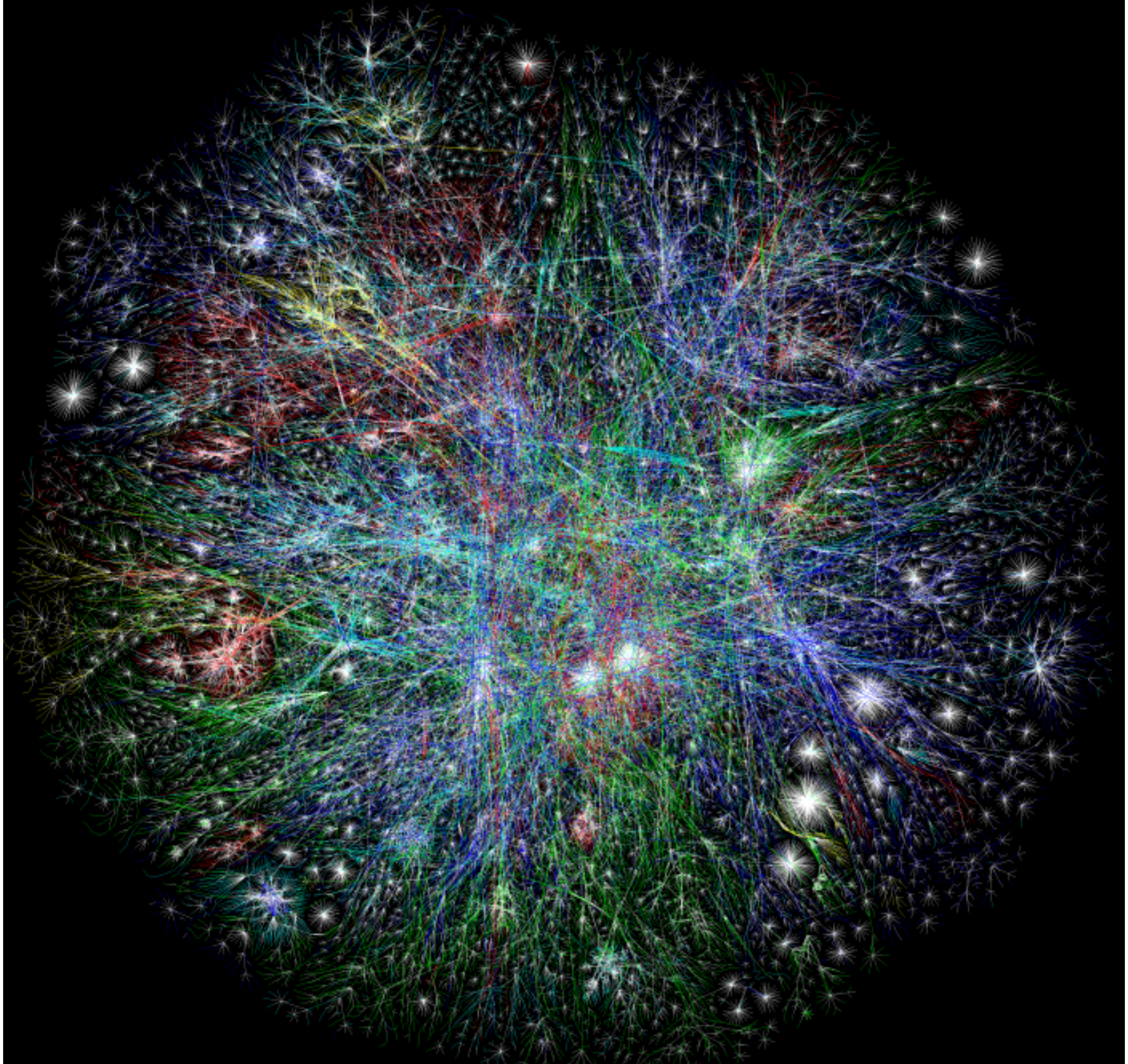
* User traffic calculation per day

Source: Miguel Angel Todaro



Source: Cisco

Source: Facebook



Traceroute map of the Internet, ~5 million edges, circa 2003. opte.org

Why should you take this course?

- **Impact**
  - Social, economic, political, educational, …
  - Why should you care about NetNeutrality?
  - What does it mean to run out of IP addresses?
  - How could Egypt shut down the Internet internally?
  - How could Pakistan shut down Youtube *globally*?
- **Continuously changing and evolving**
  - Incredible complexity
  - Any fact you learn will be inevitably out of date
  - Learn general underlying *principles*
- **Learn to program the network**
- **Networks are cool!**

Roadmap

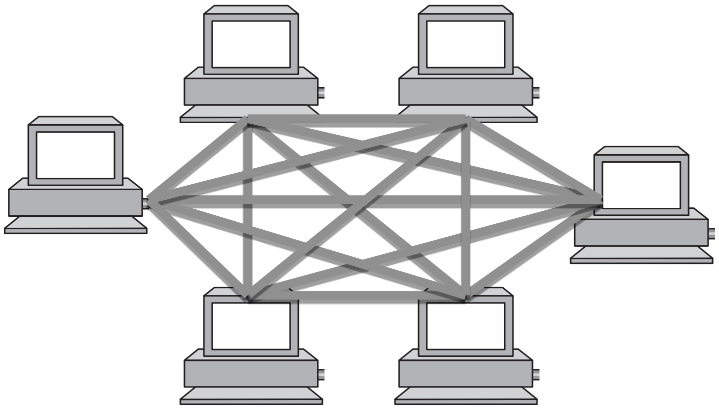
- **Assignments: learn by implementing**
  - Warm up: Snowcast, a networked music server
    - Get a feel for how applications use the network
- **Build knowledge from the ground up**
  - Link individual nodes
  - Local networks with multiple nodes
  - IP: Connect hosts across several networks
  - Transport: Connect processes on different hosts
  - Applications
- **A few cross-cutting issues**
  - Security, multimedia, overlay networks, P2P…


Two-minutes for stretching…
(and an opportunity to sneak out if you are shopping)

Building Blocks

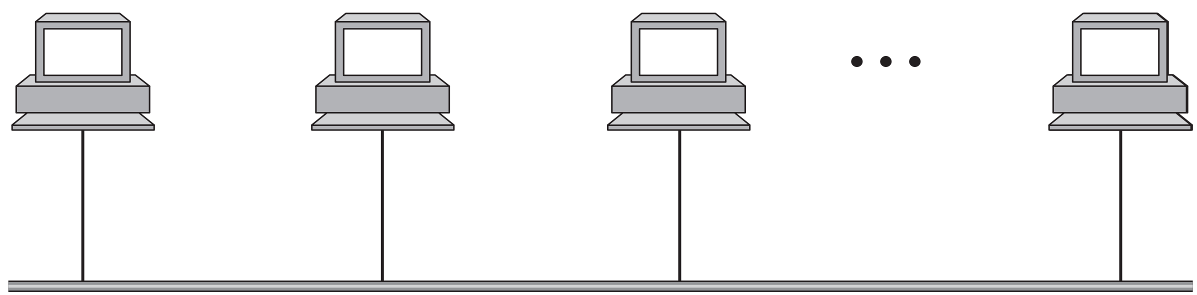
- **Nodes: Computers (hosts), dedicated routers, …**
- **Links: Coax, twisted pair, fiber, radio, …**

How to connect more nodes?

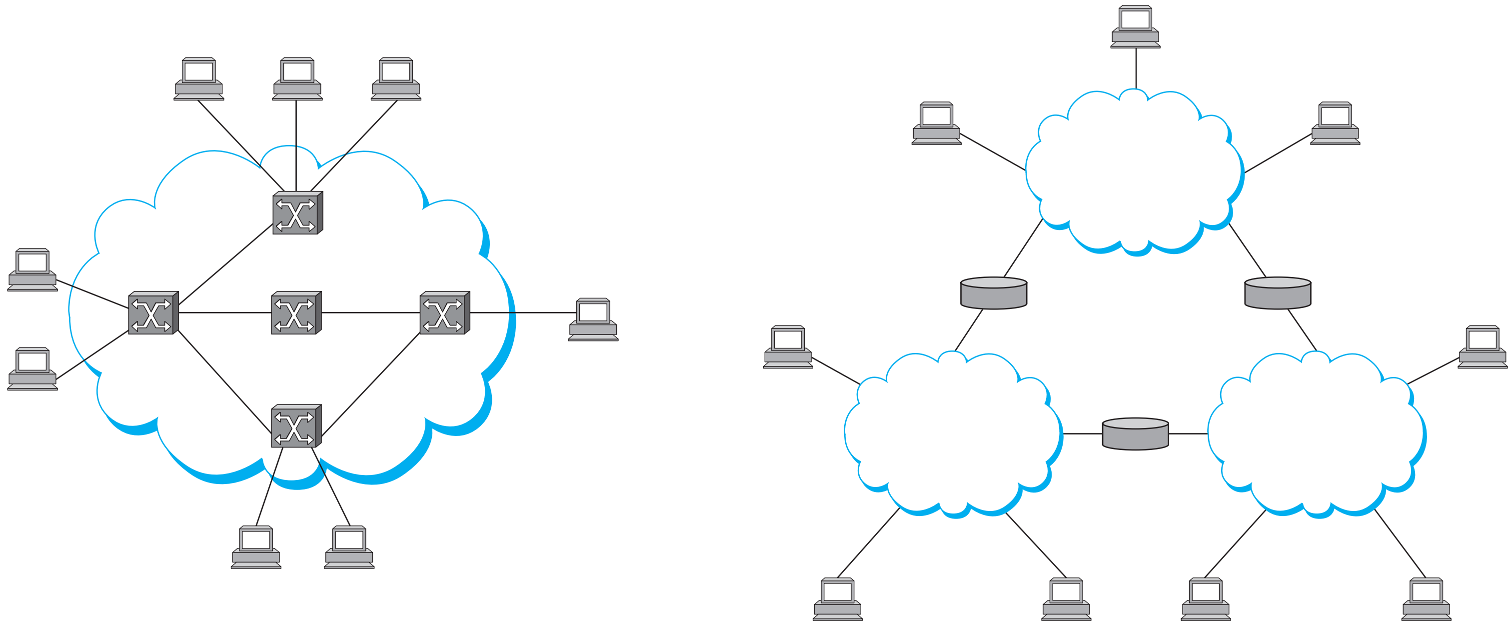


Multiple wires

Shared medium



From Links to Networks



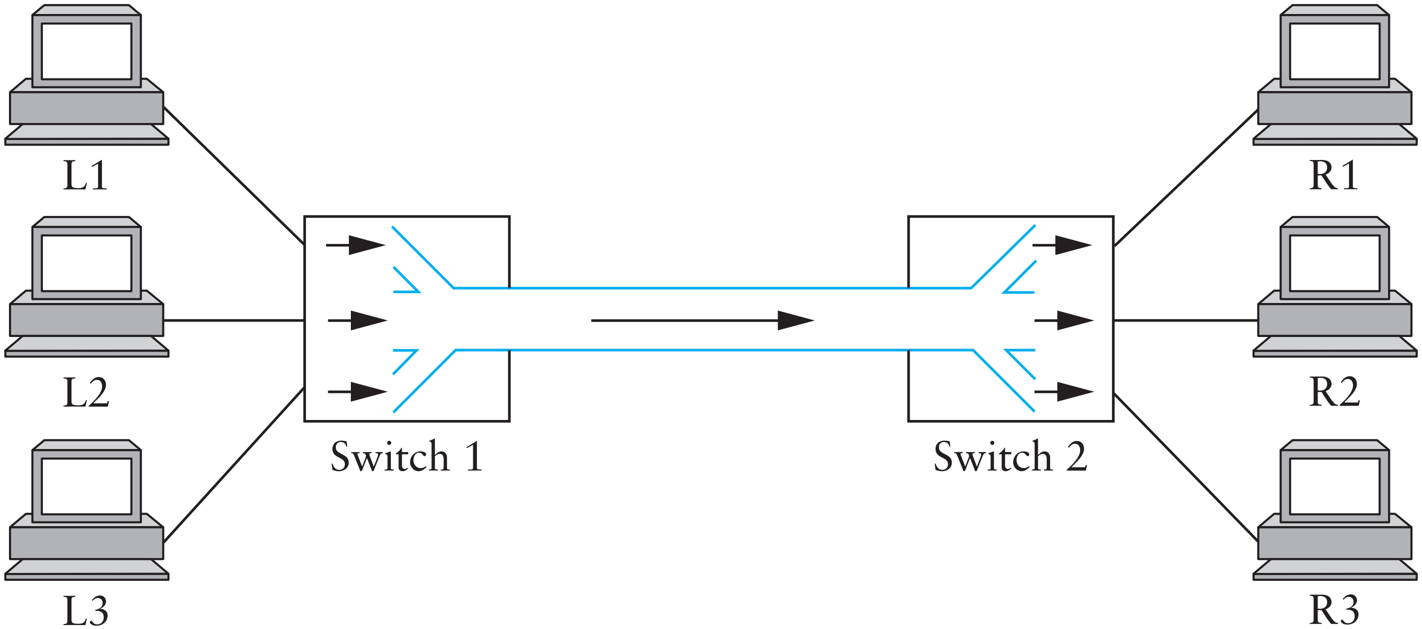
- **To scale to more nodes, use *switching***
  - Nodes can connect to multiple other nodes
  - Recursively, one node can connect to multiple networks

Switching Strategies

- **Circuit Switching – virtual link between two nodes**
  - Set up circuit (*e.g.* dialing, signaling) – may fail: busy
  - Transfer data at known rate
  - Tear down circuit
- **Packet Switching**
  - Forward bounded-size messages.
  - Each message can have different senders/receivers
  - Focus of this course

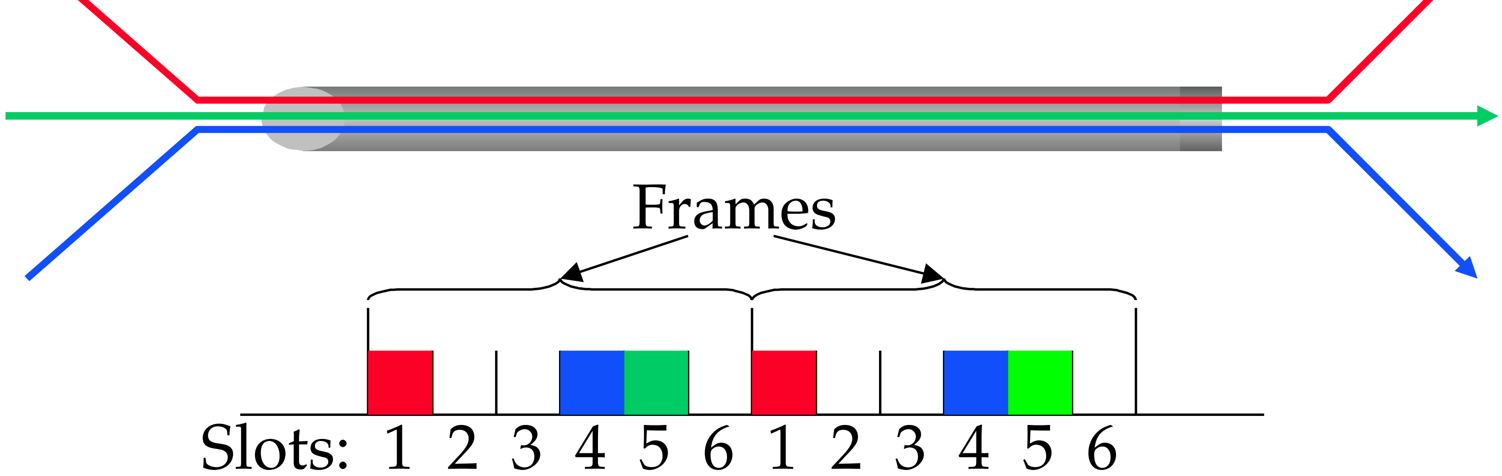
Analogy: circuit switching reserves the highway for a cross-country trip. Packet switching interleaves everyone's cars.

Multiplexing



- **What to do when multiple flows must share a link?**

STDM

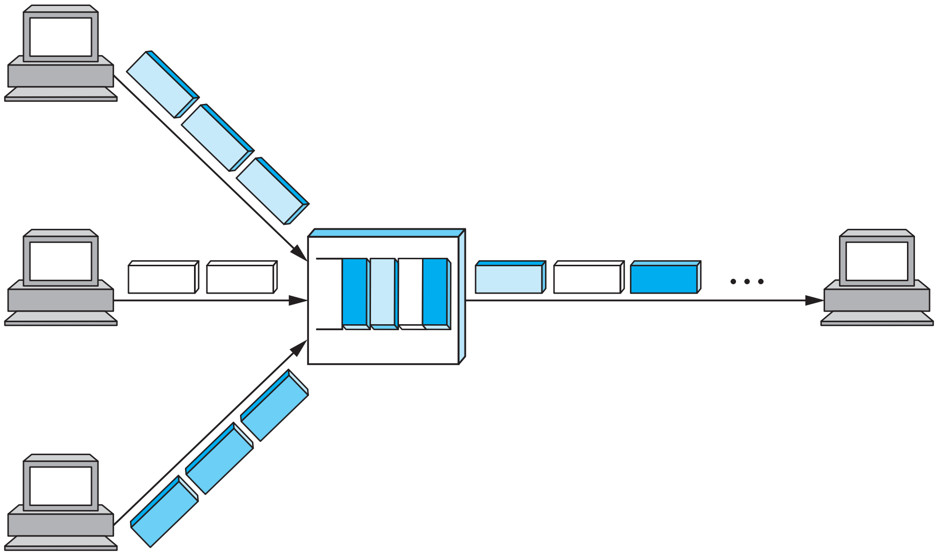


- **Synchronous time-division multiplexing**
  - Divide time into equal-sized quanta, round robin
  - Illusion of direct link for switched circuit net
  - But wastes capacity if not enough flows
  - Also doesn’t degrade gracefully when more flows than slots

FDM

- **Frequency-division multiplexing: allocates a frequency band for each flow**
  - Same TV channels and radio stations
- **Similar drawbacks to STDM**
  - Wastes bandwidth if someone not sending
  - Can run out of spectrum

Statistical Multiplexing



- **Idea: like STDM but with no pre-determined time slots (or order!)**
- **Maximizes link utilization**
  - Link is never idle if there are packets to send

Statistical Multiplexing

- **Cons:**
  - Hard to guarantee fairness
  - Unpredictable queuing delays
  - Packets may take different paths
- **Yet…**
  - This is the main model used on the Internet

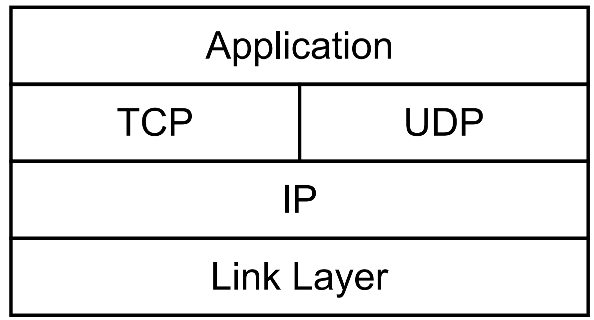
Traceroute map of the Internet, ~5 million edges, circa 2003. opte.org

Managing Complexity

- *Very* large number of computers
- Incredible variety of technologies
  - Each with very different constraints
- No single administrative entity
- Evolving demands, protocols, applications
  - Each with very different requirements!

- How do we make sense of all this?

Layering



- **Separation of concerns**
  - Break problem into separate parts
  - Solve each one independently
  - Tie together through common interfaces: abstraction
  - Encapsulate data from the layer above inside data from the layer below
  - Allow independent evolution
- **Example**
  - A *network layer* packet from A to D is put in *link layer* frames A to B, B, to C, C to D

Single Link Communication

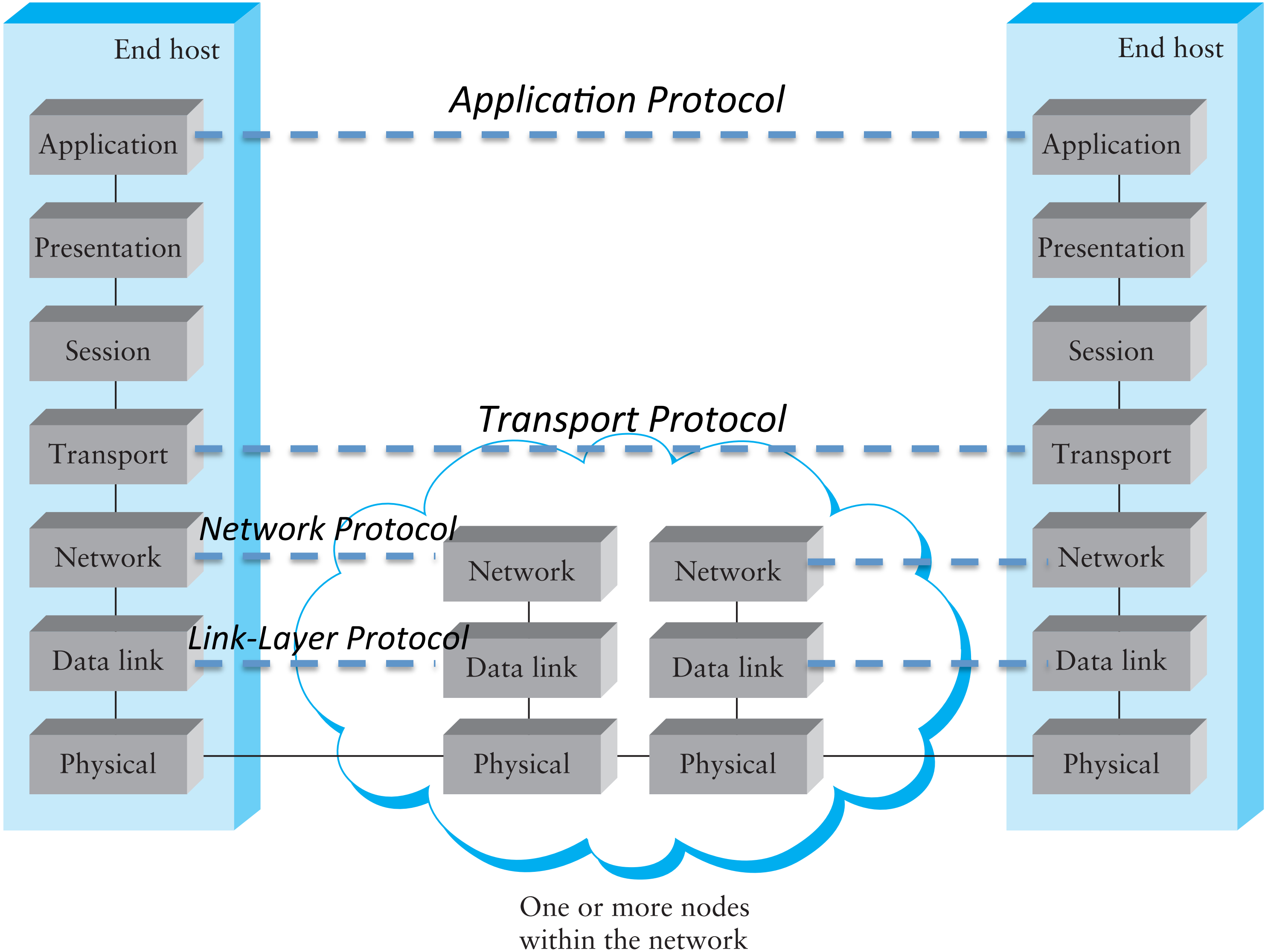
- **Physical Layer: Several questions:**
  - Encoding: voltage, frequency, phase,…
  - Medium: copper, fiber, radio, light,…
- **Link Layer: how to send data?**
  - When to talk
  - What to say (format, "language")
- **Examples: Ethernet, USB**

Stay tuned for lectures 3 and 4…

Layers

- **Application – what the users sees, *e.g.*, HTTP**
- **Presentation – crypto, conversion between representations**
- **Session – can tie together multiple streams (*e.g.*, audio & video)**
- **Transport – demultiplexes, provides reliability, flow and congestion control**
- **Network – sends *packets*, using *routing***
- **Data Link – sends *frames*, handles media access**
- **Physical – sends individual bits**

OSI Reference Model



Protocols

- **What do you need to communicate?**
  - Definition of message formats
  - Definition of the semantics of messages
  - Definition of valid sequences of messages
    - Including valid timings

Addressing

- **Each node typically has a unique* name**
  - When that name also tells you how to get to the node, it is called an *address*
- **Each layer can have its own naming/addressing**
- ***Routing* is the process of finding a path to the destination**
  - In packet switched networks, each packet must have a destination address
  - For circuit switched, use address to set up circuit
- **Special addresses can exist for broadcast/multicast/anycast**

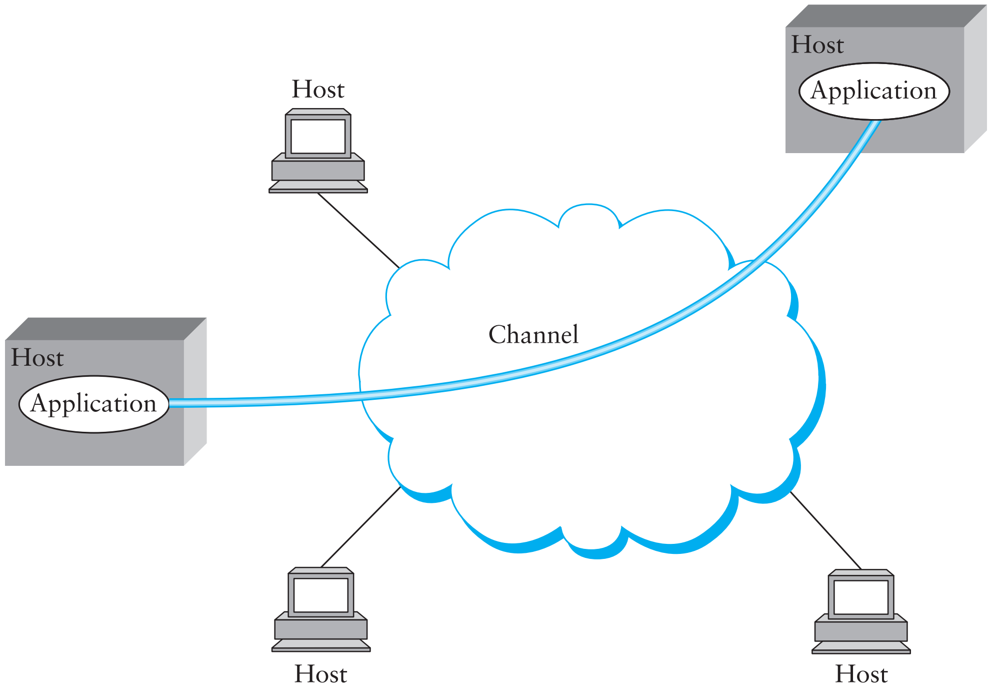
** or thinks it does, in case there is a shortage*

Network Layer: Internet Protocol (IP)

- **Used by most computer networks today**
  - Runs *over* a variety of physical networks, can connect Ethernet, wireless, modem lines, etc.
- **Every host has a unique* 4-byte IP address (IPv4)**
  - *E.g.*, www.cs.brown.edu → 128.148.32.110
  - The *network* knows how to route a packet to any address
- **Need more to build something like the Web**
  - Need naming (DNS)
  - Interface for browser and server software (next lecture)
  - Need demultiplexing within a host: which packets are for web browser, Skype, or the mail program?

* NATs make this story more complicated, we'll talk about this more

Inter-process Communication



- **Talking from host to host is great, but we want abstraction of inter-process communication**
- **Solution: *encapsulate* another protocol within IP**

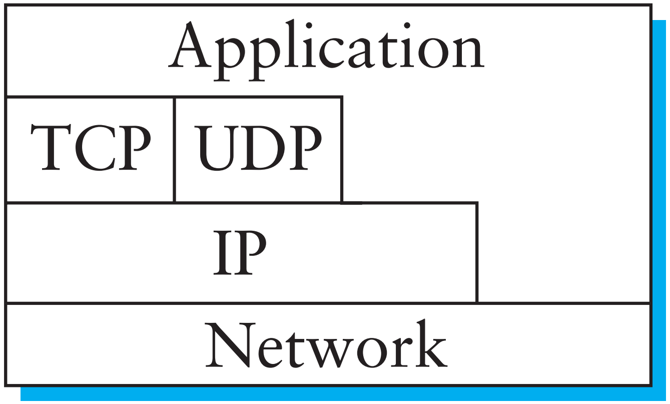
Transport: UDP and TCP

- **UDP and TCP most popular protocols on IP**
  - Both use 16-bit *port* number & 32-bit IP address
  - Applications *bind* a port & receive traffic on that port
- **UDP – User (unreliable) Datagram Protocol**
  - Exposes packet-switched nature of Internet
  - Sent packets may be dropped, reordered, even duplicated (but there is corruption protection)
- **TCP – Transmission Control Protocol**
  - Provides illusion of reliable 'pipe' or 'stream' between two processes anywhere on the network
  - Handles congestion and flow control

Uses of TCP

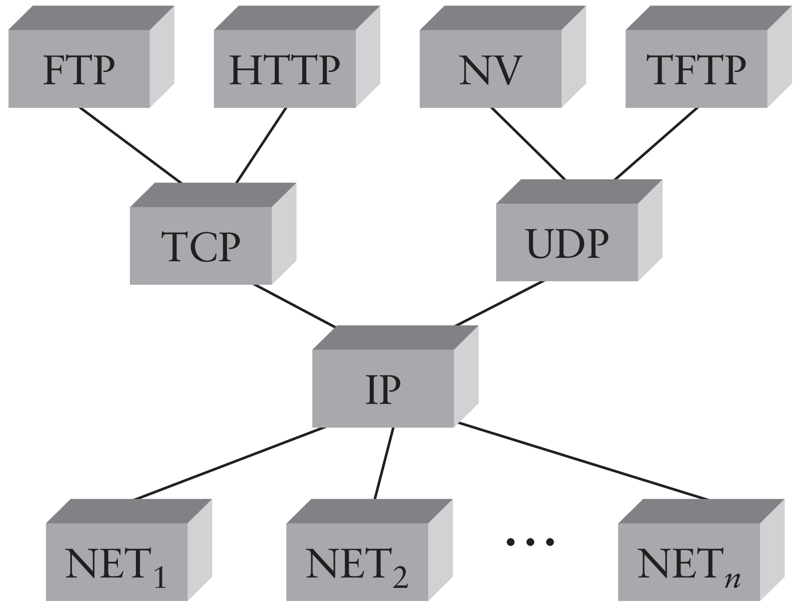
- **Most applications use TCP**
  - Easier to program (reliability is convenient)
  - Automatically avoids congestion (don't need to worry about taking down the network
- **Servers typically listen on well-know *ports*:**
  - SSH: 22
  - SMTP (email): 25
  - Finger: 79
  - HTTP (web): 80

Internet Layering



- **Strict layering not *required***
  - TCP/UDP "cheat" to detect certain errors in IP-level information like address
  - Overall, allows evolution, experimentation

IP as the Narrow Waist



- **Many applications protocols on top of UDP & TCP**
- **IP works over many types of networks**
- **This is the "Hourglass" architecture of the Internet.**
  - If every network supports IP, applications run over many different networks (*e.g.*, cellular network)

Roadmap

- **Assignments: learn by implementing**
  - Warm up: Snowcast, a networked music server
    - Get a feel for how applications use the network
- **Build knowledge from the ground up**
  - Link individual nodes
  - Local networks with multiple nodes
  - IP: Connect hosts across several networks
  - Transport: Connect processes on different hosts
  - Applications
- **A few cross-cutting issues**
  - Security, multimedia, overlay networks, P2P…

Coming Up

- **Snowcast: start TODAY!**
- **Next class: how do applications use the network?**
  - Introduction to programming with Sockets
  - Peterson & Davie 1.4
  - Beej's Guide to Network Programming (link on the course website)
- **Then…**
  - We start moving up the network stack, starting from how two computers can talk to each other.