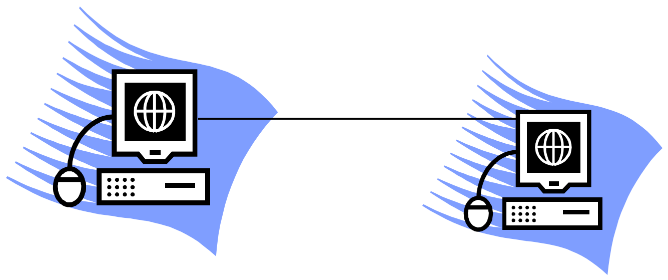
CSE 461: Introduction to Computer Communications Networks
Winter 2010

Module 1
Course Introduction

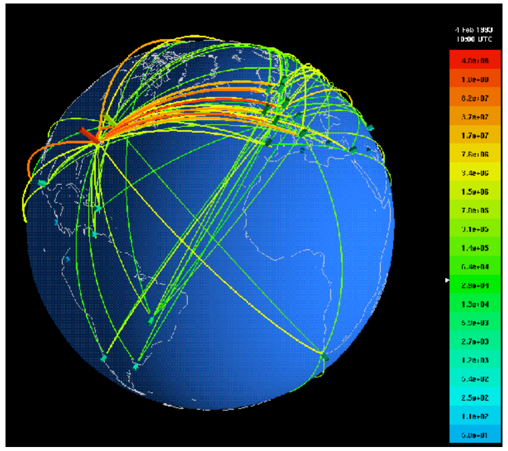
John Zahorjan
zahorjan@cs.washington.edu
534 Allen Center

A Network in 461

- A network is what you get anytime you connect two or more computers together by some kind of a link.



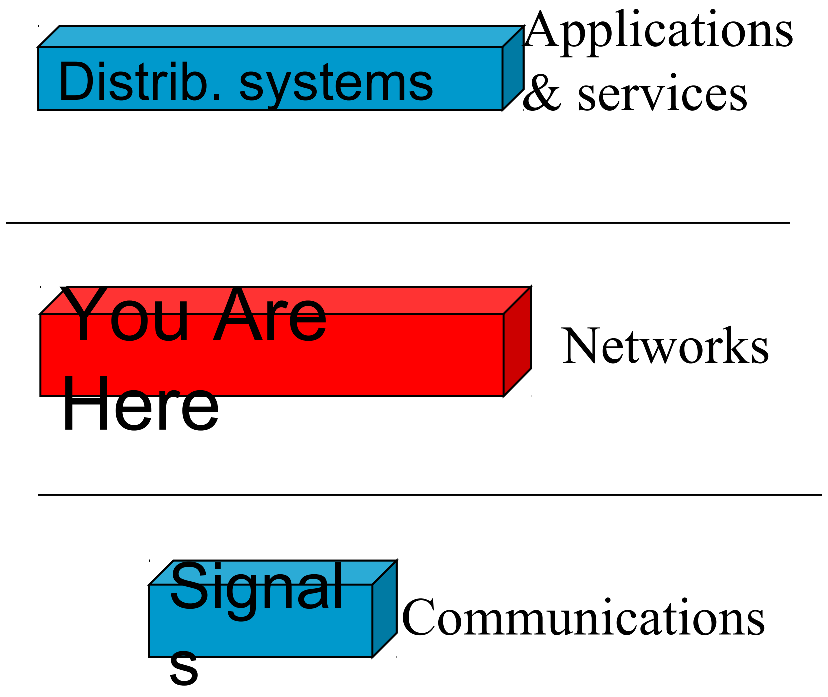
OR



Focus of this Course

- You will understand how to design and build *large, distributed computer* networks.
  - Fundamental problems
  - Design principles
  - Implementation technologies

- This is a systems course, not queuing theory, signals, or hardware design.
- We focus on networks, and a bit on applications or services that run on top of them.



Today's agenda

- Course Administration
  - Everything you need to know will be on the course web page:

    http://www.cs.washington.edu/461/

  - Most everything (lecture schedule, reading, assignments, section materials, …) is linked off the schedule
- Introduction to Course Content
  - Part 1: Generally useful principles and abstractions
  - Part 2: An overview of the Internet

Course goals

- Our primary goal is to understand how today's networks are built
- This involves a mixture of:
  - science: Is there an algorithm that meets some goal?
  - engineering: How cost effective are various alternatives likely to be?
  - experience: what has worked, what hasn't, and why?
  - measurement: are current networks working as intended? how are people using them?

Course goals (cont.)

- What is likely to be of lasting value to you?
  - Specific information: Many (most? all?) real applications involve networks.
  - General lessons: engineering a large, dynamic system
- The hope, as always, is to make all minutes you spend on the course worth your while
- Activities:
  - reading text, answering questions from text, taking exams
  - reading additional important papers, writing short analyses of them
  - Implementations…

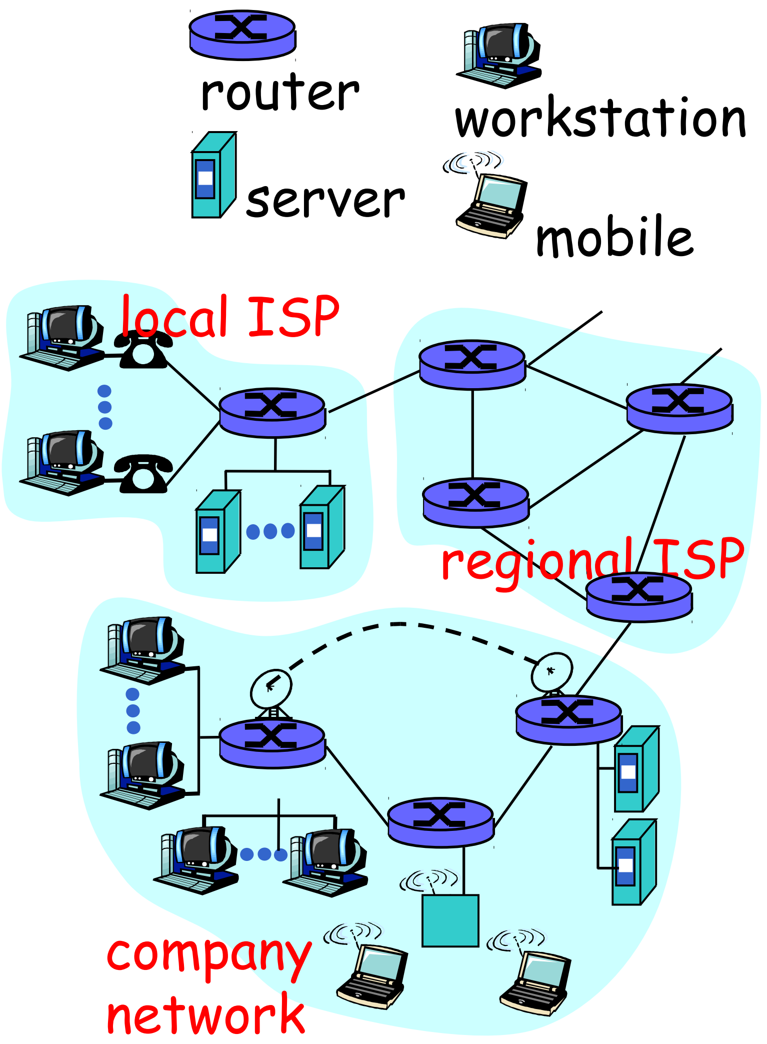
Last bit of course admin

- the text
  - Peterson & Davie, *Computer networks: a systems approach* (4th edition)
- other resources
  - many online; some of them will be required reading
- Policies
  - email
  - late policy
  - grading

Introduction: The Internet at large

- Internetworks = network of networks
- Hierarchical structure
- millions of connected computing devices: *hosts, end-systems*
  - pc's workstations, servers
  - PDA's phones, toasters

  running *network apps*
- *communication links*
  - fiber, copper, radio, satellite
- *routers:* forward packets (chunks) of data thru network



Introduction to Networking

- Understanding networking involves thinking in a way you're almost certainly not accustomed to
  - networks are *distributed*:
    - concurrent: there is more than one program/computer involved
    - possibly strange failure semantics
      - sure, part of the "application" can crash and others stay up, but…
      - part can operate incorrectly, or
      - part can go down and come back up while app is running…
  - network architectures are *deeply layered*:
    - not just 2 levels (process/kernel)
    - actual implementations favor function and efficiency over blind respect for layering
  - networks can have *immense scale and heterogeneity*
    - our most prominent network, the Internet, is so large and dynamic, and operated by so many distinct, entities that no one knows just exactly what it looks like, how it's being used, or how well it's working
  - networks *must work correctly*
    - is "five nine's" (99.999%) correctness enough?
    - Estimate: by 2015, one zettabyte/year traffic (one million million billion bytes)

Example Networking Problem

- Suppose you've amassed a 1TB (1024 GB) collection of "home movies" and you want to communicate a copy of them to a friend living in Walla Walla
- Q: What networking technology should you use?
  - A: A 1.5TB disk and the US Postal System
- This is not a joke…

TeraScale SneakerNet: Using Inexpensive Disks for Backup, Archiving, and Data Exchange.

Jim Gray, Wyman Chong, Tom Barclay, Alex Szalay, Jan Vandenberg

May 2002, Technical Report, MS-TR-02-54

Table 2: The raw price of bandwidth, the true price is more than twice this when staff, router, and support costs are included. Raw prices are higher in some parts of the world.

Context	Speed Mbps	Rent $/month	Raw $/Mbps	Raw $/TB sent	Time/TB days
home phone	0.04	40	1,000	3,086	6 years
home DSL	0.6	70	117	360	5 months
T1	1.5	1,200	800	2,469	2 months
T3	43	28,000	651	2,010	2 days
OC3	155	49,000	316	976	14 hours
100 Mpbs	100				1 day
Gbps	1000				2.2 hours
OC192	9600	1,920,000	200	617	14 minutes

Table 3: The relative cost of sneaker-net, using various media. The analysis assumes 6MBps tape, 10MBps CD/DVD and robots at each end to handle the media. Note that the price of media is less than the fixed robot cost.

	Media	Robot$	Media$	TB read + write time	ship time	TotalTime /TB	Mbps	Cost (10 TB)	$/TB shipped
CD	1500	2x800	240	60 hrs	24 hrs	6 days	28	$2,080	$208
DVD	200	2x8000	400	60 hrs	24 hrs	6 days	28	$20,000	$2,000
Tape	25	2x15,000	1000	92 hrs	24 hrs	5 days	18	$31,000	$3,100
DiskBrick	7	1,000	1,400	19 hrs	24 hrs	2 days	52	$2,600	$260

"Storage Brick"

Huh? Why not just send the disks?

"We began sending raw disks to one another, but that has the nasty problem that disks do not plug right into the network. So the recipient had to have exactly the right kind of disk reader (an ATA path that could read an NTFS file system with an SQL database). At a minimum, this excludes our Macintosh, Solaris, MVS, and Linux colleagues. Even within the select group that can read our favored disk format, we had many problems about master-slave settings on the ATA bus, about recognizing dynamic disks, and innumerable other details. So, sending magnetic disks around is problematic."

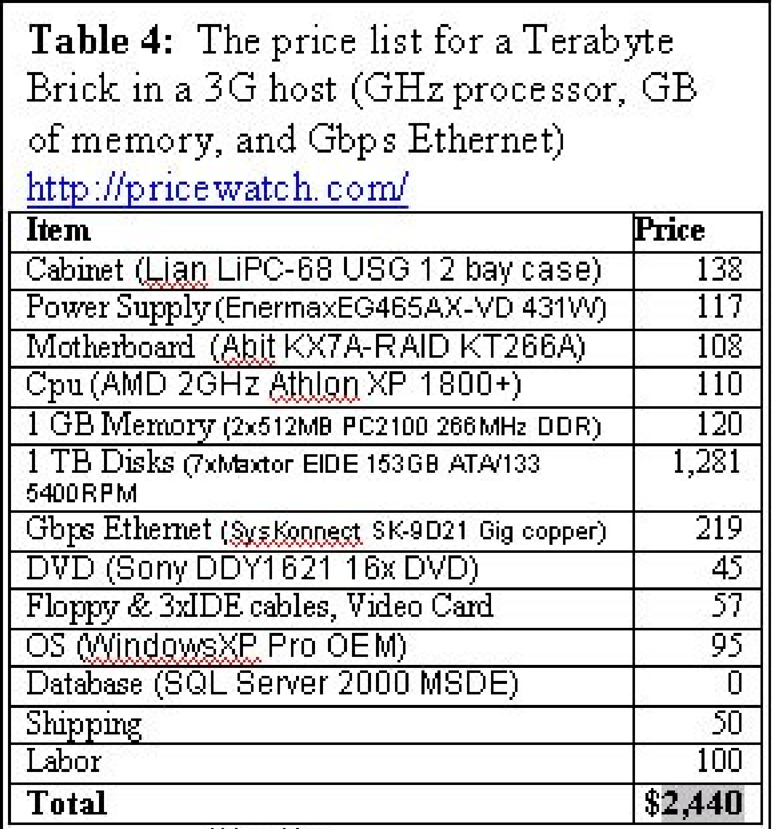
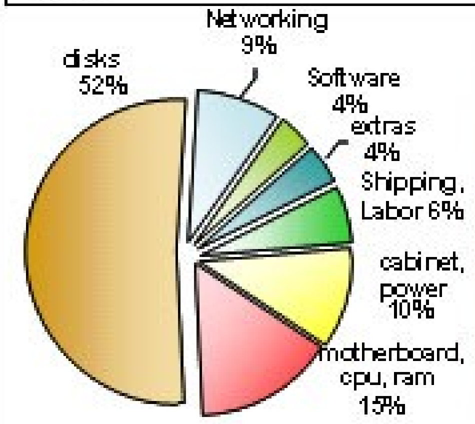


Table 4: The price list for a Terabyte Brick in a 3G host (GHz processor, GB of memory, and Gbps Ethernet) http://pricewatch.com/

Item	Price
Cabinet (Lian LiPC-68 USG 12 bay case)	138
Power Supply (EnermaxEG465AX-VD 431W)	117
Motherboard (Abit KX7A-RAID KT266A)	108
Cpu (AMD 2GHz Athlon XP 1800+)	110
1 GB Memory (2x512MB PC2100 266MHz DDR)	120
1 TB Disks (7xMaxtor EIDE 153GB ATA/133 5400RPM	1,281
Gbps Ethernet (SysKonnect SK-9D21 Gig copper)	219
DVD (Sony DDY1621 16x DVD)	45
Floppy & 3xIDE cables, Video Card	57
OS (WindowsXP Pro OEM)	95
Database (SQL Server 2000 MSDE)	0
Shipping	50
Labor	100
Total	**$2,440**



The USPS as a Network

- There is a *client* at each end of the connection
- USPS itself is a networking *service*
- The service exports an *API* that tells clients how to use it:
  - To send:
    - wrap your data in an envelope
    - put an address on the envelope in a format that we (USPS) dictate
    - enter your packet into our system
  - To receive:
    - check your mailbox every so often
    - remove the envelope from whatever you find
    - voila, the data that was sent to you

How does the USPS network work?

- *To be honest, I have almost no idea.*
- *Fortunately, for our purposes it doesn't matter – it's enough that our hypothetical explanation is plausible and could work*

1. All mail deposited into a mail box, no matter what its final destination, is first encapsulated in a new container (a mail truck) and routed to a local sorting facility.
2. The mail is unencapsulated (taken out of the truck). The destination address is examined, and it is re-encapsulated in a new container (e.g., mail headed to zip codes 993** is placed in a bag)
3. The bag of mail is un- and then re-encapsulated a number of times as it is transported over different physical media – a truck to Boeing Field, then an airplane to SFO, a truck to a sorting facility there, a truck back to SFO, a plane to Walla Walla, a truck to sorting facility there, a jeep/mailbag on the delivery route. At each stop, a decision is made about where to send it next.
4. As it nears its final destination, routing is based on its actual street address. It's eventually stuffed into the mailbox for your friend's address.
5. Your friend's roommates don't open it because part of the address includes his name. (Note that his name portion of the address was irrelevant up to this point.)

This is a lot like (but not exactly like) what happens in the Internet

Parallels to the Internet

- Division of responsibilities/capabilities intersecting at an API
  - At the highest level, you don't much care *how* USPS delivers your mail, all you care about is the API. (We'll look at what the API is in just a moment.)
  - Similarly, USPS doesn't care or know what your data is.
  - Moreover, you don't really have much control over how USPS delivers your mail.

- Delivery involves a number of hops, with *routing* decisions made at each.

- A number of different physical media (trucks, planes, feet) are used, with the lowest capacity media typically found near the sender and the destination and the fastest media in the middle (of the route).

- Addresses are places, not people
  - "1600 Pennsylvania Avenue NW, Washington, DC 20500"
    not
    "Barack Obama"
  - Why? Why do we care?

Parallels to the Internet (cont.)

- There is a loose hierarchy involved in choosing a delivery route – more precise location information is needed as the mail gets closer to its destination.

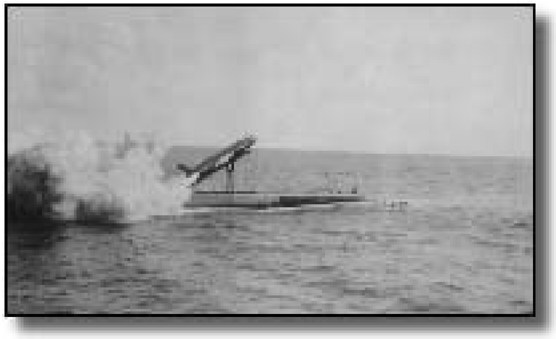
- Correspondingly, the useful part of the address changes at various stages of delivery, for example:
  - None of it is relevant in the first step (truck to local sorting facility)
  - The first three digits of the zip are relevant through a lot of the middle stages
  - The full street address is important in the second last stage
  - "Joe Smith" is relevant (only) once it has been delivered to the destination mailbox

- There is a maximum allowed size – if you want to send more than that, put whatever it is in multiple boxes, each not too big

- If it's Christmas, expect more problems than usual
  - The system capacity is set to give good performance most of the time, but can suffer during periods of unusual load

Parallels to the Internet (cont.)

- USPS is able to make use of new delivery technologies as they arise, without altering its API



Missile Mail Launch, 1959

Missle Mail

Throughout its history, the Postal Service enthusiastically has explored faster, more efficient forms of mail transportation. Technologies now commonplace -- railroads, automobiles, and airplanes -- were embraced by the Post Office Department at their radical birth, when they were considered new-fangled, unworkable contraptions by many. One such technology, however, remains only a footnote in the history of mail delivery. On June 8, 1959, in a move a postal official heralded as "of historic significance to the peoples of the entire world," the Navy submarine U.S.S. *Barbero* fired a guided missile carrying 3,000 letters at the Naval Auxiliary Air Station in Mayport, Florida. "Before man reaches the moon," the official was quoted as saying, "mail will be delivered within hours from New York to California, to Britain, to India or Australia by guided missiles."

History proved differently, but this experiment with missile mail exemplifies the pioneering spirit of the Post Office Department when it came to developing faster, better ways of moving the mail.

copied without permission from http://www.usps.com/history/his2_75.htm

One Last Parallel

- 1963: USPS rolls out zip codes
  - 5 digits => 100,000 different zips, that's plenty
- 1983: USPS rolls out zip+4
  - 9 digits => 1,000,000,000 different zips; this time we mean it

- early 1970's: IP developed
  - 32-bit address fields => 4 billion distinct addresses, that's plenty
- circa 1995: IPv6 standarized
  - 128-bit address fields => about 10^{38} distinct addresses (about 10^{24} per square meter of the earth)
- (circa 2009: IPv6 still not widely adopted)

The USPS API

- We know how to send/receive mail. But what are the semantics of those operations (i.e., what properties are guaranteed)?
  - Reliability?
    - Is everything sent eventually delivered?
    - Is it received? (What's the difference?)
    - Is it received by the person named in the address?
  - Failure notification?
    - Does USPS let me know if it got there or not?
  - Integrity?
    - If it arrives, are the contents undamaged? *(Does it arrive only once?)*
  - Latency (delay)
    - Is there a guaranteed upper bound? How much does it vary from one letter to the next?
  - Ordering?
    - If I send a letter a day to a single destination address, do they arrive in the order I sent them?
  - Security?
    - Is anyone reading your mail in transit?
    - Can you be sure who sent the mail?
    - Can you avoid having so much junk mail sent to you that there's no room for the mail you want?

USPS as Engineering

- Want the service to be widely useful
  - Has to accommodate lots of different client “decisions”
    - Most anything as an envelope (not a USPS designed one)
    - Any old handwriting (not printed in some specific font)
    - Contents of envelope are irrelevant to its delivery
  - Has to be “scalable” – able to deliver mail from any and to any of an ever increasing number of addresses
  - Has to be cheap enough that people will use it
    - So, has to be cheap to provide
- To be cheap, the API provides almost no guarantees
  - It’ll probably get there, it’ll probably take around a week, and it’ll probably not be damaged
  - We can’t tell you who sent it or how it got there or if the check is actually in the mail
  - We can’t guarantee the carrier won’t read your postcard

USPS Engineering Decisions

- This may sound as though the design is "the cheapest thing to implement possible." It's not.
  - "Once a day, everyone take mail you find in your mailbox that's not for you and put it in the next mailbox to the right…"
- Hopefully, it's an appropriate tradeoff:
  - to be any cheaper, some property would have to get a lot worse (e.g., delivery times in months, not days)
  - to provide any stronger properties (e.g., we can verify who the sender was) it would have to get a lot more expensive
  - the properties it does provide are good enough in practice for lots of uses
- A desirable feature: if in some cases it's worth it to the sender to have stronger properties, those properties can be built on top of the generic service at additional expense
  - If you want to know when the mail you sent arrives, phone the person you sent it to daily
  - if you don't want the postman reading your postcards, put them in envelopes

Moral of This Example

- What the Internet is doing isn't really all that complicated
- The fact that it isn't all that complicated is a stunning engineering accomplishment
  - Not that it could be built, but that they decided to build it this way
- If the pieces of the Internet architecture get a bit confusing, think about USPS – it's a pretty good analogy

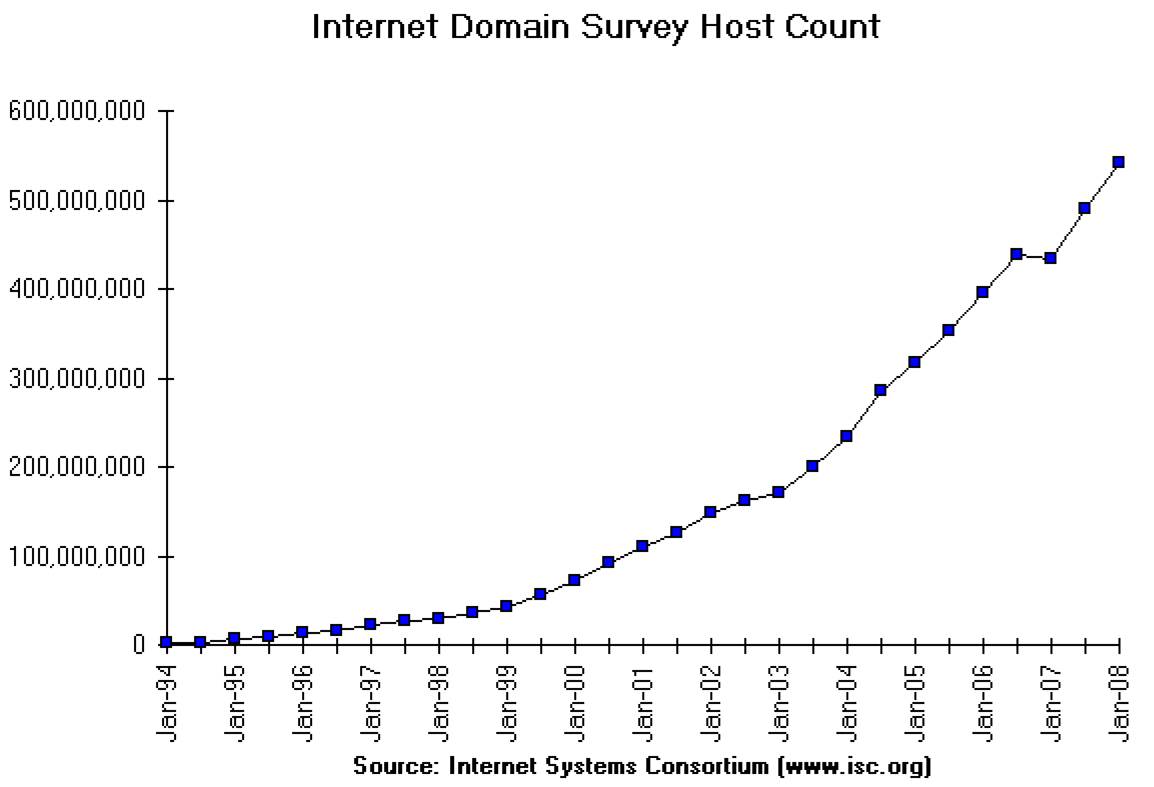
Part II: Overview of Computer Networking

1. Scalability / Implications of scale
2. The API
3. Internet overview
4. Layering / The OSI Model

Part 1: Network Scalability

- For this course, a “network” is what connects two or more computers. (What’s a “computer”?)
- We are interested in network architectures that are “scalable” – continue to work efficiently even as the size of the system grows by orders of magnitude

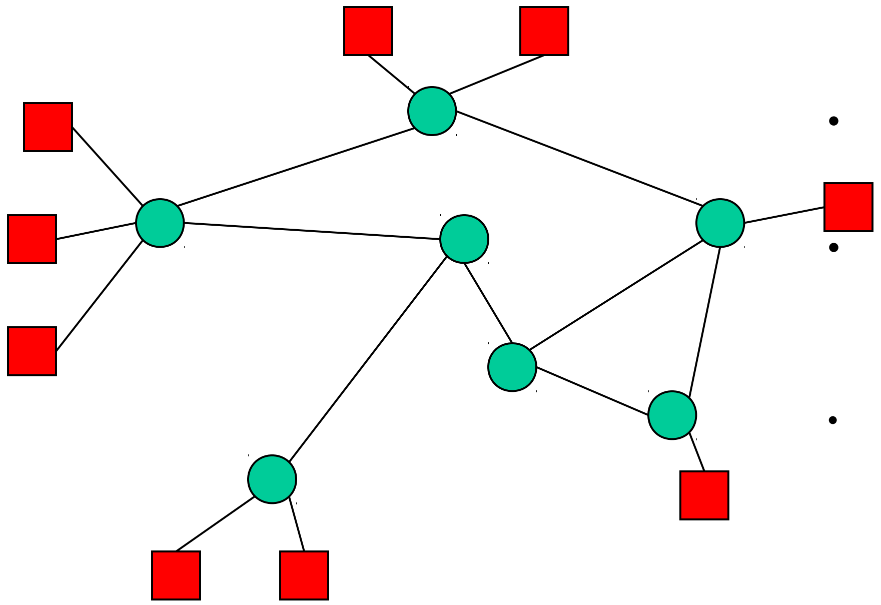
Why is scalability important?



- The basic network design happened in the 1970's.
  - There were maybe 10,000's of computers in the world at the time
- Not only *could* the design scale, it provided a combination of cost and benefit that drove demand

Implication of Scale I: Sharing

- It's clearly infeasible to interconnect an ever growing number of machines by running a wire/fiber/radio wave between every pair



- Links carry information (bits)
  - Wire, wireless, fiber optic, smoke signals …
  - May be point-to-point or broadcast
- Switches move bits between links
  - Routers, gateways,bridges, CATV headend, PABXs, …
- Hosts are the communication endpoints
  - PC, PDA, cell phone, tank, toaster, …
  - Hosts have names

- Much other terminology: channels, nodes, intermediate systems, end systems, ...

Implication of Scale II: Intrinsic Unreliability

- Information sent from one place to another
  - May not arrive
  - May arrive more than once
  - May arrive in garbled fashion
  - May arrive out of order
  - May be read by others
  - May be modified by others
- Why build intrinsically unreliable networks?

Implication of Scale III: Distributed

"A distributed system is a system in which I can't do my work because some computer has failed that I've never even heard of." – Lamport

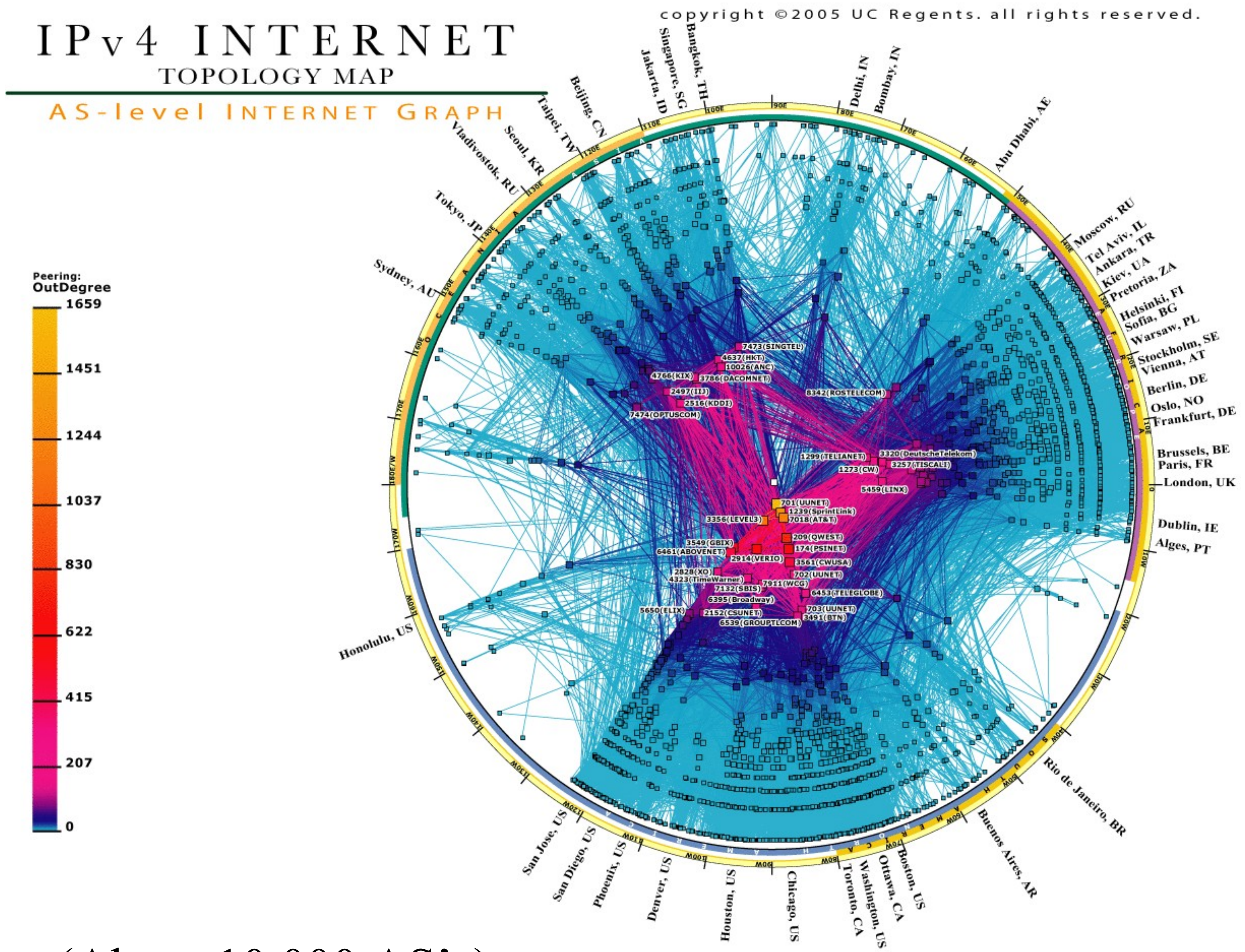
- (Hopefully) independent failure modes
- Exposed and hidden dependencies

Impl. Of Scale IV: Heterogeneous HW/SW

- Heterogeneous: Made up of different kinds of stuff
  - vs Homogeneous: Made up of the same kind of stuff
- Principles
  - Homogeneous networks are easier to deal with
  - Heterogeneous networks promote innovation and scale
  - Consider telephone network vs. Internet
  - Reasons?

Implications of Scale V: Autonomous Authorities

- The Internet is basically an interconnection of networks owned and operated by different people/corporations
  - I own/operate the network in my house
  - My ISP owns the network my network directly connects to
  - My ISP is connected to the network owned by the UW
  - The UW’s network is connected to the network owned by CSE

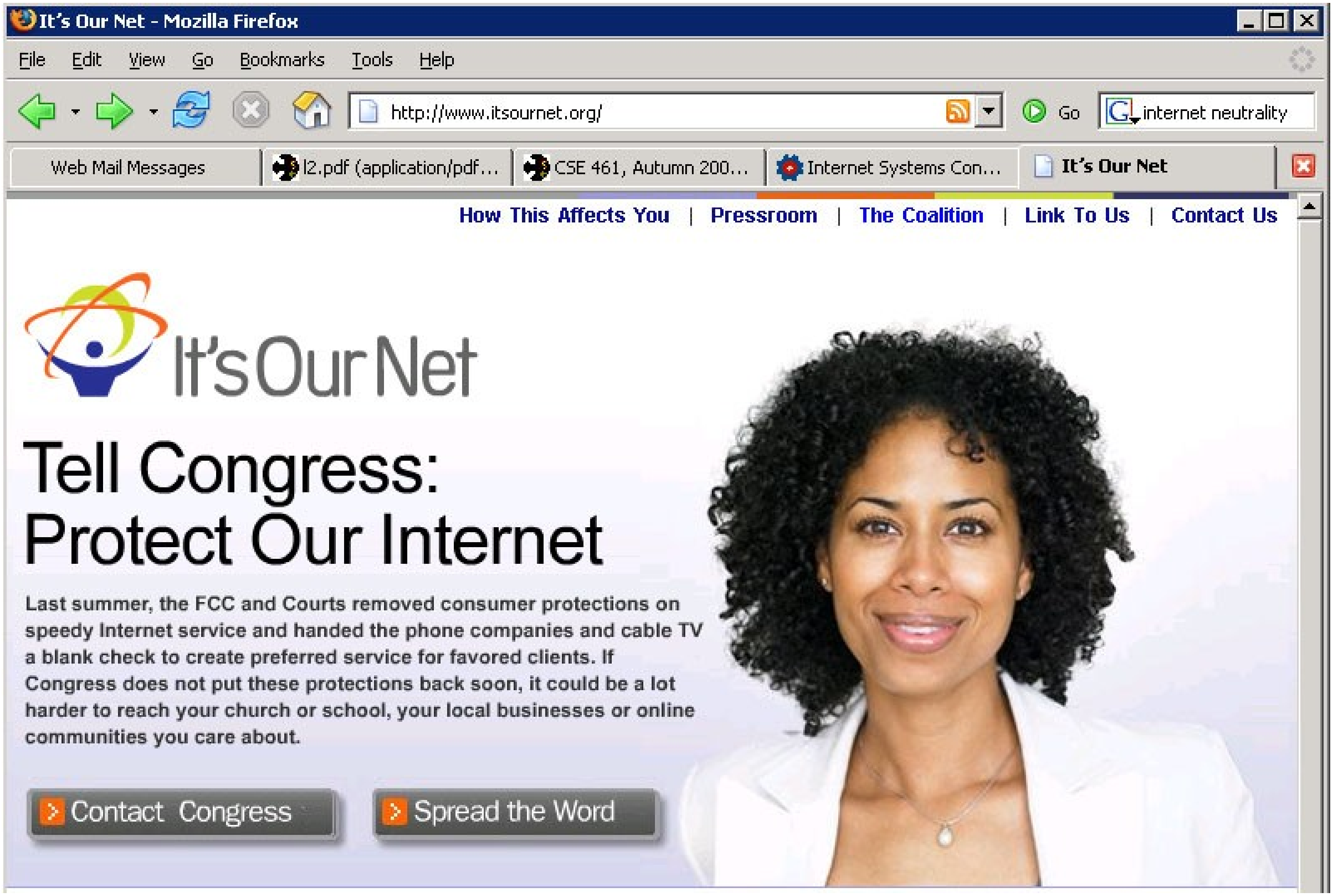


(About 10,000 AS's)

Implications of Confederation/Autonomy

- HW/SW heterogeneity (which was inevitable due to scale anyway)
- "Okay everybody, start using 64-bit addresses NOW."
- Policy/goals heterogeneity
  - So what?

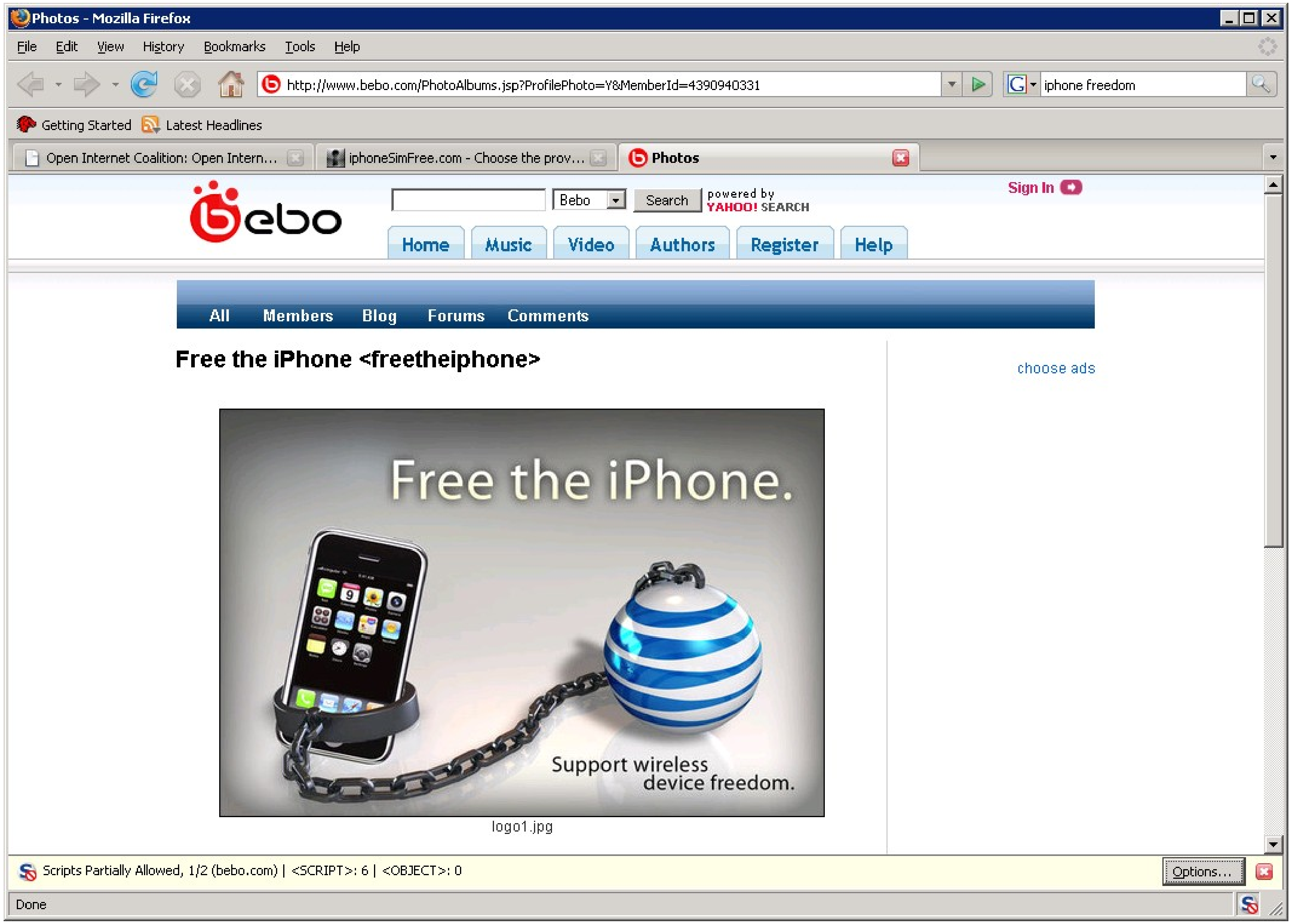
Summer 2006: Internet Neutrality


It's Our Net - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.itsournet.org/
Go
internet neutrality
Web Mail Messages
l2.pdf (application/pdf…
CSE 461, Autumn 200…
Internet Systems Con…
It's Our Net
How This Affects You | Pressroom | The Coalition | Link To Us | Contact Us
It's Our Net
Contact Congress
Spread the Word

Tell Congress: Protect Our Internet

Last summer, the FCC and Courts removed consumer protections on speedy Internet service and handed the phone companies and cable TV a blank check to create preferred service for favored clients. If Congress does not put these protections back soon, it could be a lot harder to reach your church or school, your local businesses or online communities you care about.

2007: Cell Service Neutrality



Photos - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.bebo.com/PhotoAlbums.jsp?ProfilePhoto=Y&MemberId=4390940331
iphone freedom
Getting Started
Latest Headlines
Open Internet Coalition: Open Intern...
iphoneSimFree.com - Choose the prov...
Photos
bebo
Bebo
Search
powered by YAHOO! SEARCH
Sign In
Home
Music
Video
Authors
Register
Help
All
Members
Blog
Forums
Comments
choose ads
Free the iPhone.
Support wireless device freedom.
Options...
Done

Free the iPhone <freetheiphone>

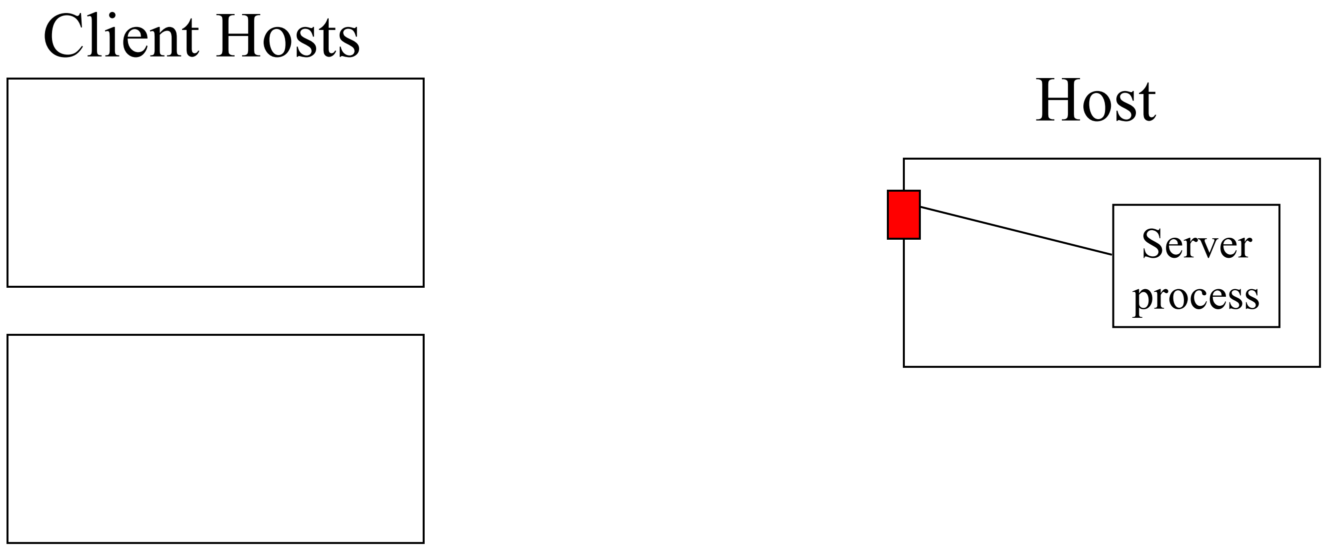
logo1.jpg

Scripts Partially Allowed, 1/2 (bebo.com) | <SCRIPT>: 6 | <OBJECT>: 0

Part II: The API

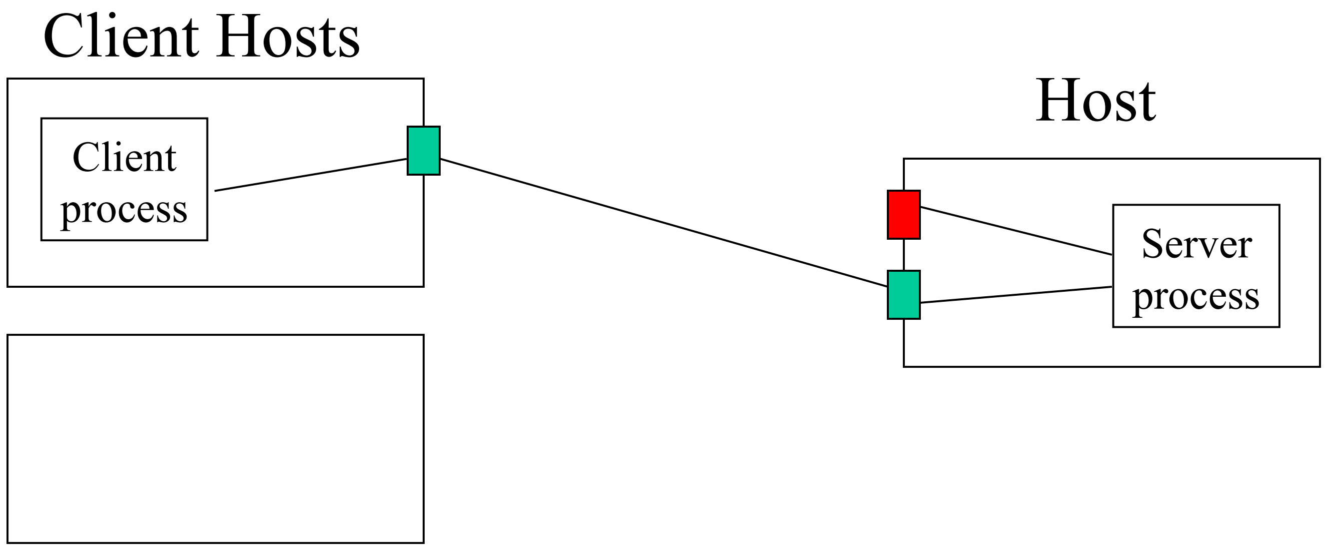
- Just as we want the network service software to run on top of many kinds of hardware, we'd like many kinds of applications to run on top of the network service
- The API is most commonly exposed through a *socket interface*
- A socket is a communication endpoint

(TCP) Socket API – The Typical Case



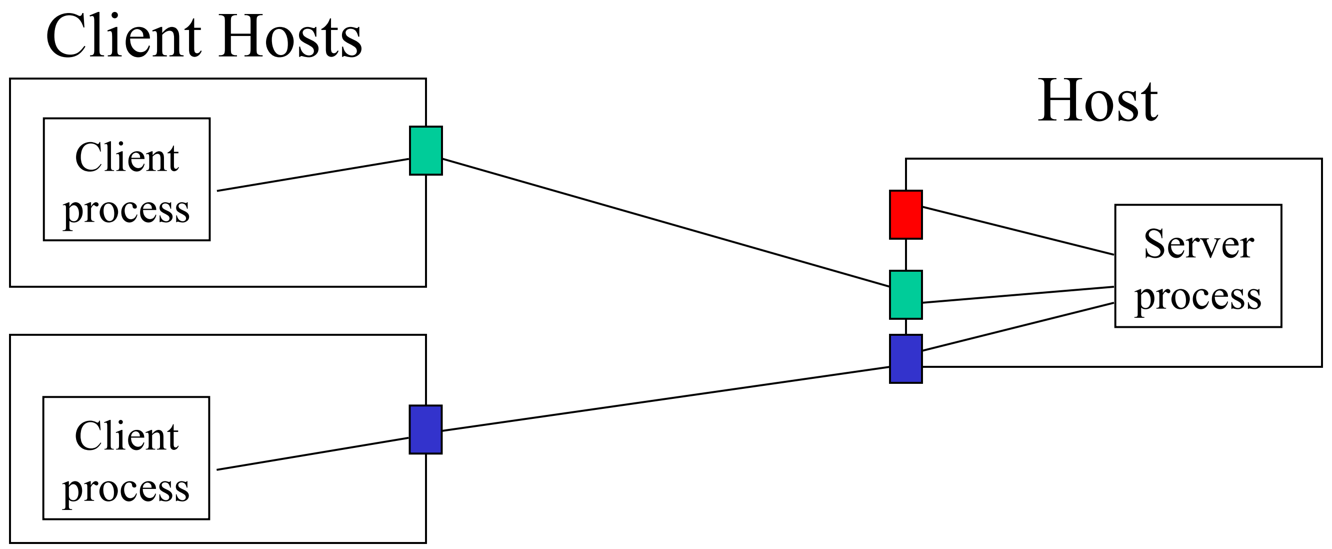
1. Server process is launched, creates a socket, and waits someone to connect to it.

Socket API (2)



1. Server process is launched, creates a socket, and waits someone to connect to it.
2. Client process is launched on some host, creates a socket, and causes it to be contact the server-side socket. This creates a new socket at the server, representing this particular connection.

Socket API (3)

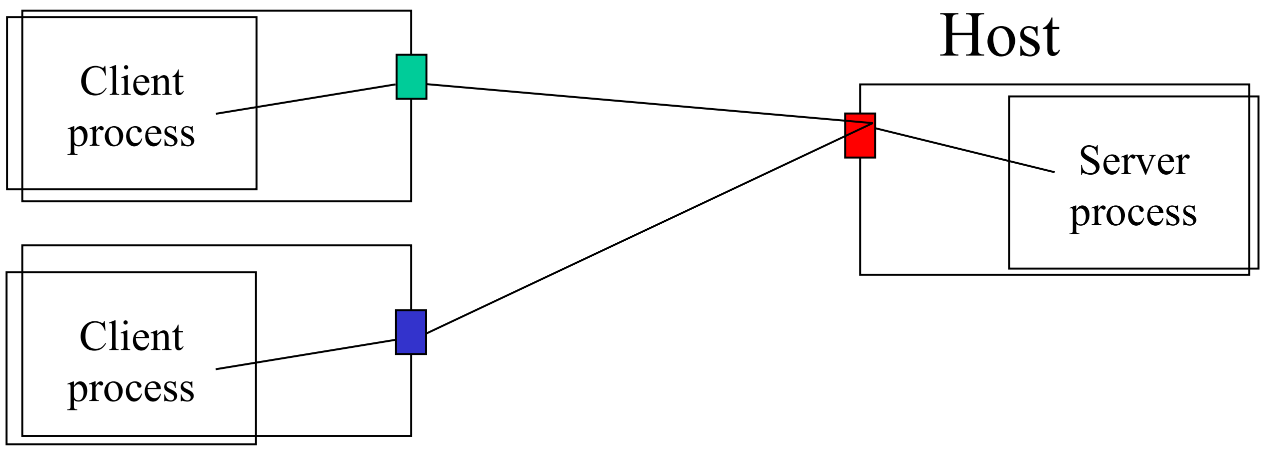


1. Server process is launched, creates a socket, and waits someone to connect to it.
2. Client process is launched on some host, creates a socket, and causes it to be contact the server-side socket. This creates a new socket at the server, representing this particular connection.
3. Another client does the same thing…

Socket API

- Somewhat more detail in Chapter 1

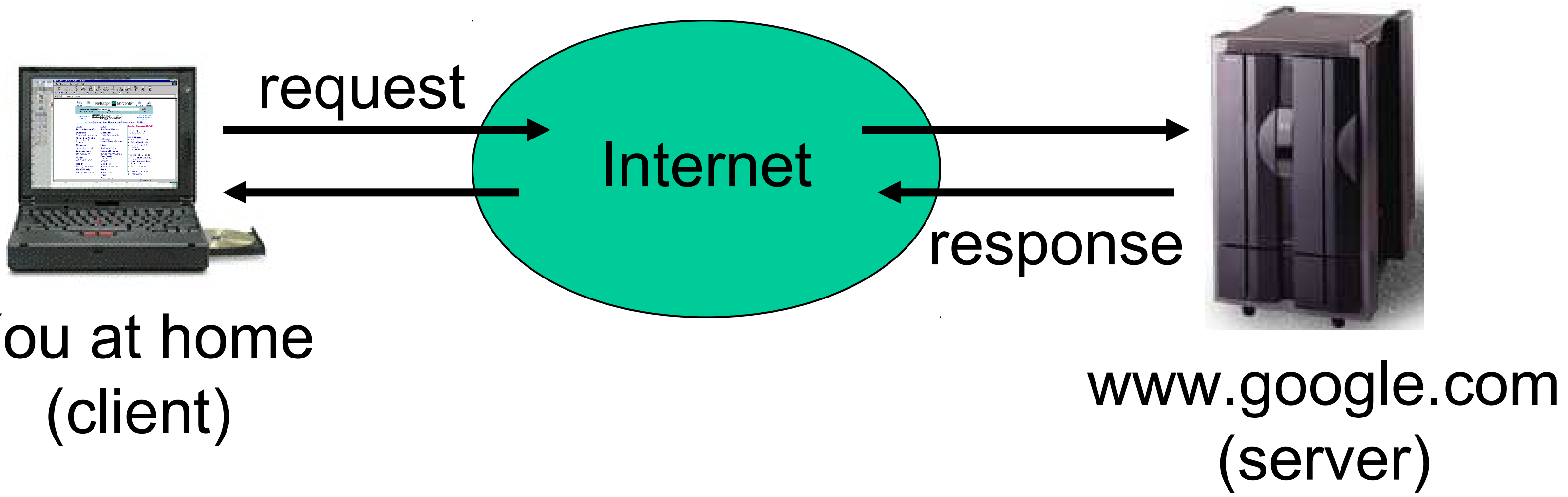
- Somewhat more detail as part of HW 0 (except we will be using Java...)



- What value is there to putting sockets between processes? (Why not connect to the server process directly?)

Part III: A Brief Tour of the Internet

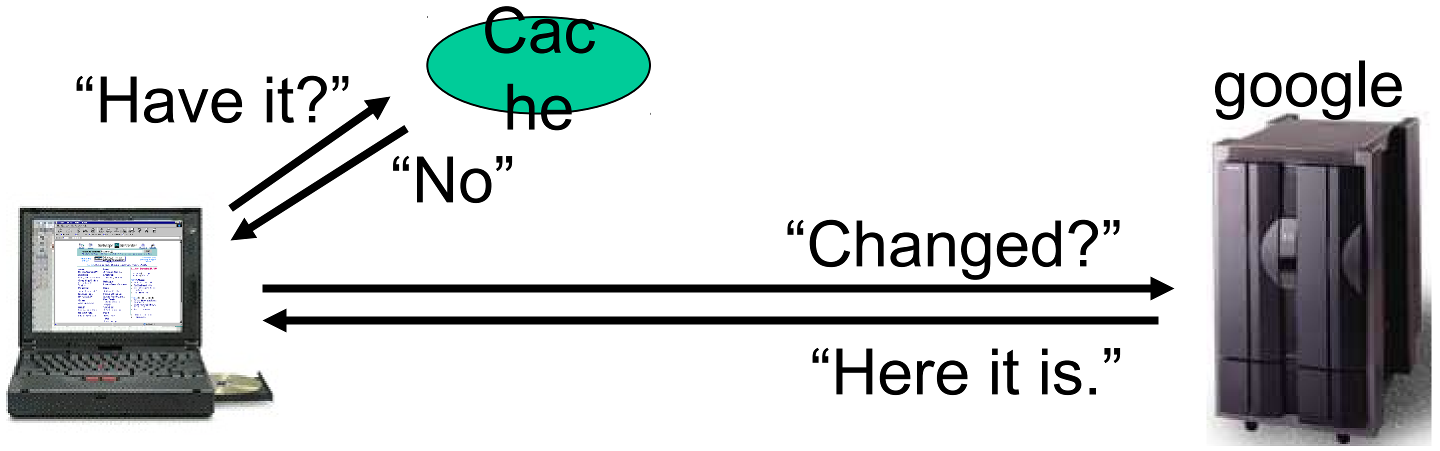
- What happens when you “click” on a web link?



- This is the view from 10,000 ft …

9,000 ft: Scalability

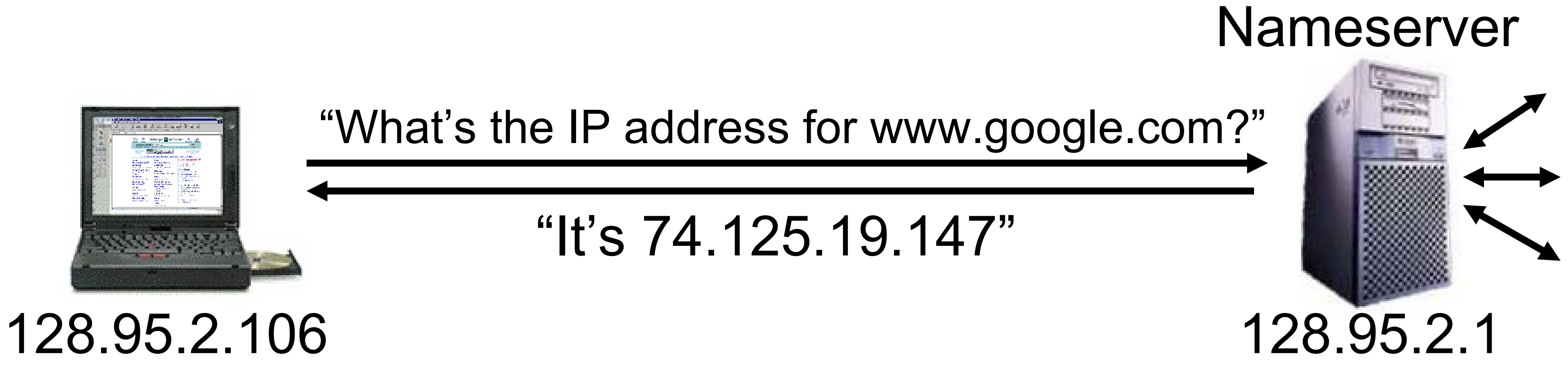
- Caching improves scalability



- We cut down on transfers:
  - Check cache (local or proxy) for a copy
  - Check with server for a new version

8,000 ft: Naming (DNS)

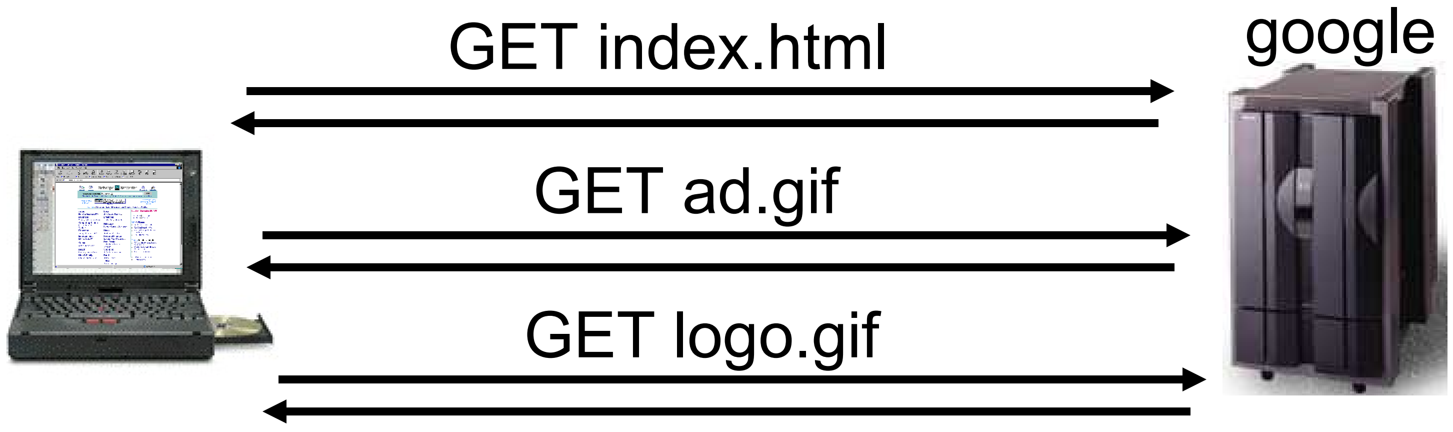
- Map domain names to IP network addresses



- All messages are sent using IP addresses
  - So we have to translate names to addresses first
  - But we cache translations to avoid doing it next time

7,000 ft: Sessions (HTTP)

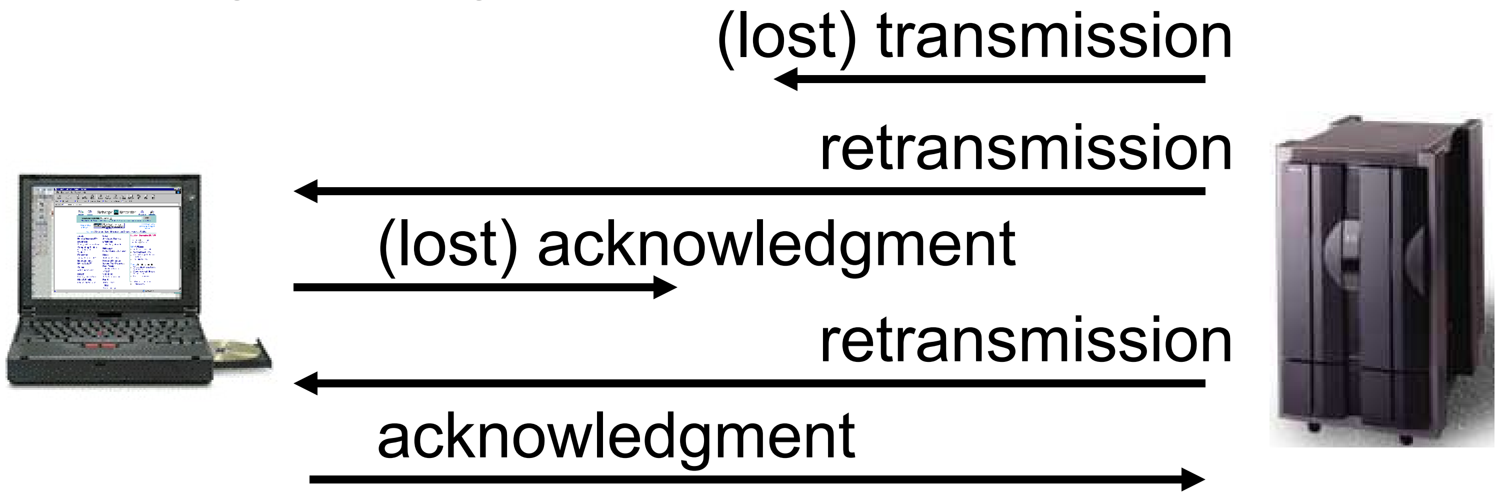
- A single web page can be multiple "objects"



- Fetch each "object"
  - either sequentially or in parallel

6,000 ft: Reliability (TCP)

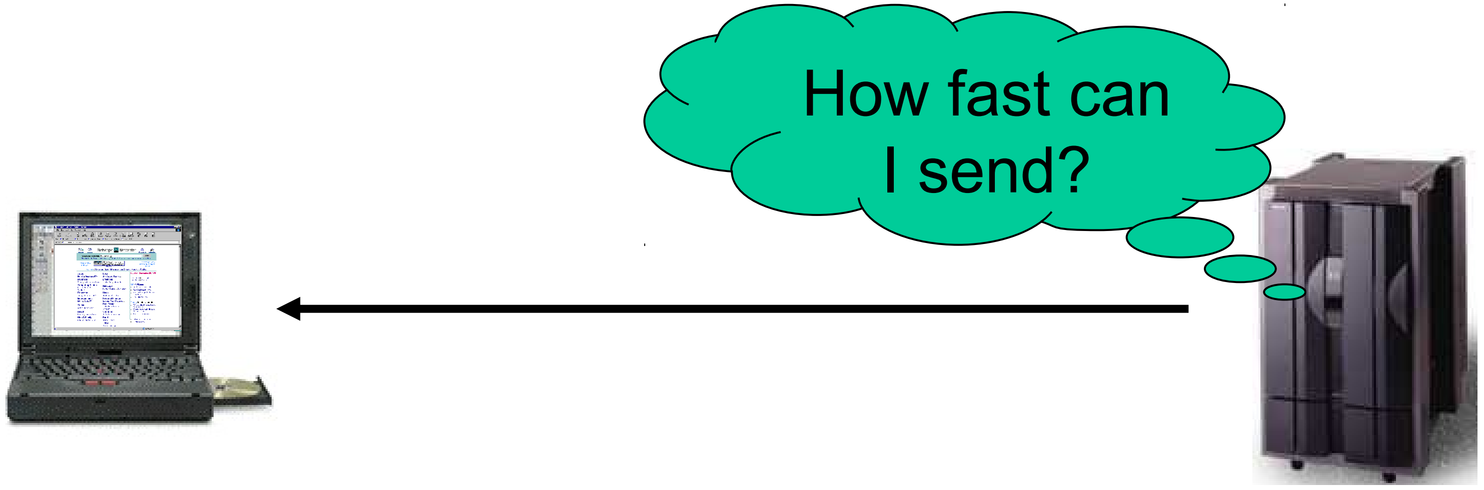
- Messages can get lost



- We acknowledge successful receipt and detect and retransmit lost messages (e.g., timeouts)

5,000 ft: Congestion (TCP)

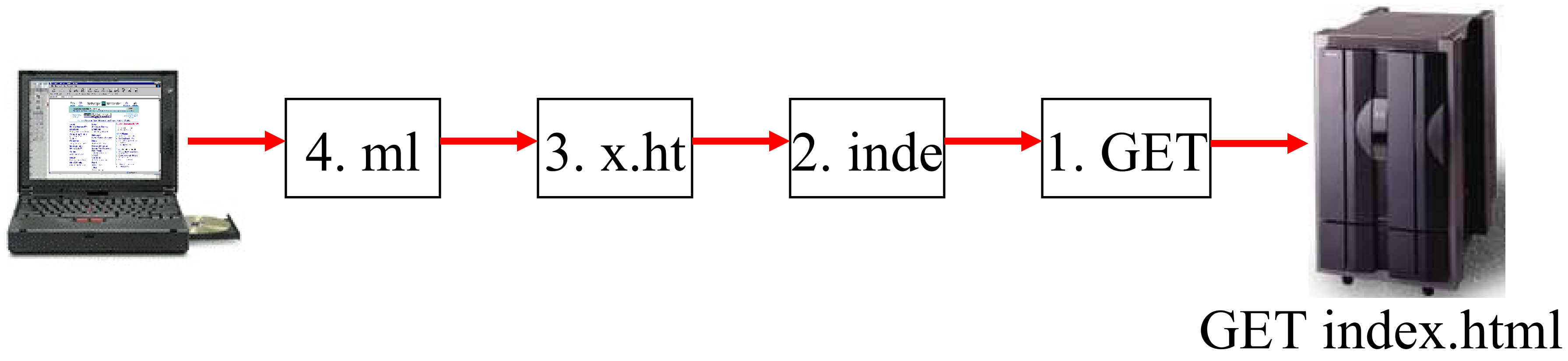
- Need to allocate bandwidth among users



- Senders balance available and required bandwidths by probing network path and observing the response

4,000 ft: Packets (TCP/IP)

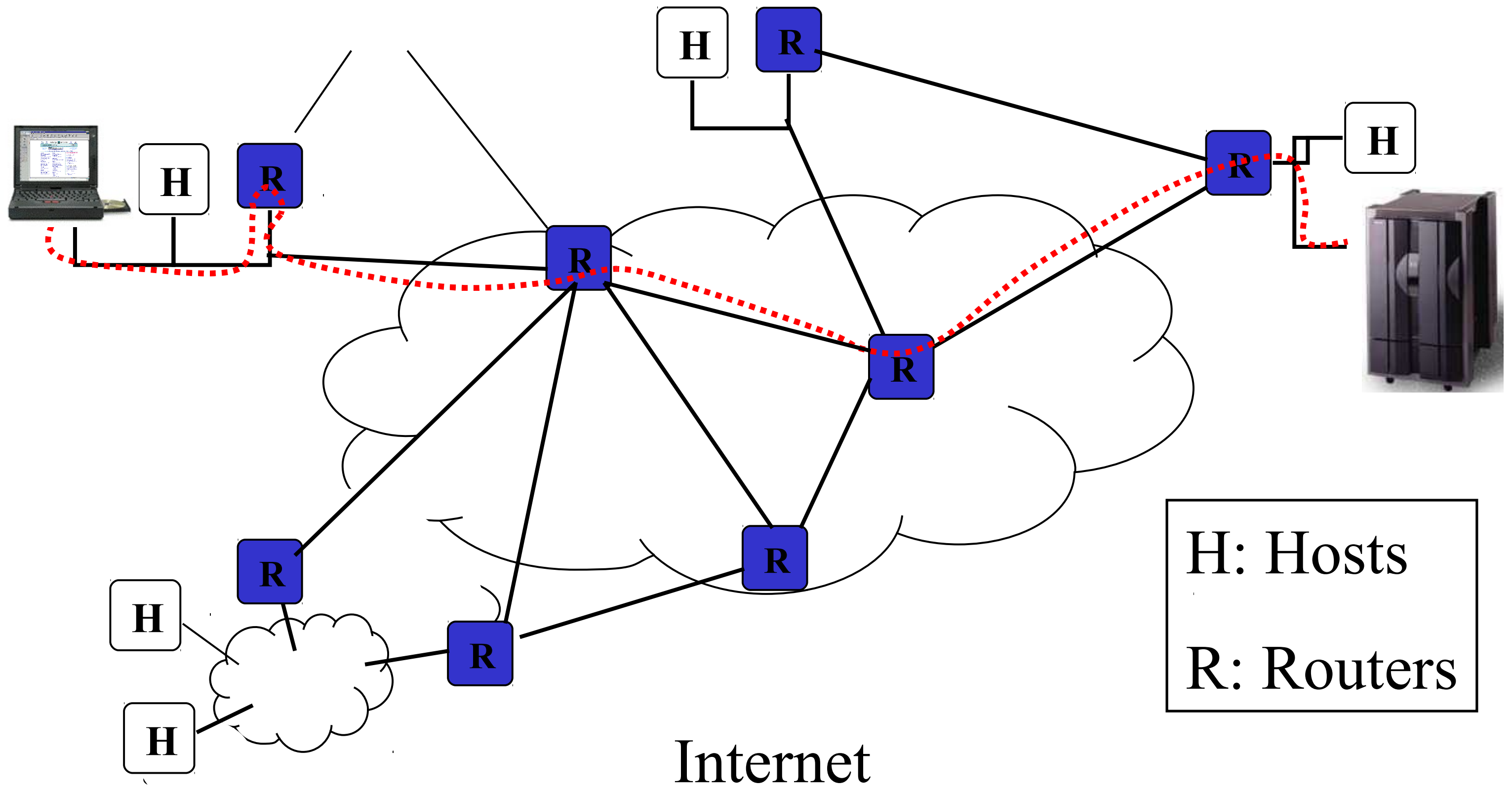
- Long messages are broken into packets
  - Maximum Ethernet packet is 1.5 Kbytes
  - Typical web page is 10 Kbytes



- Number the segments for reassembly

3,000 ft: Routing (IP)

- Packets are directed through many routers

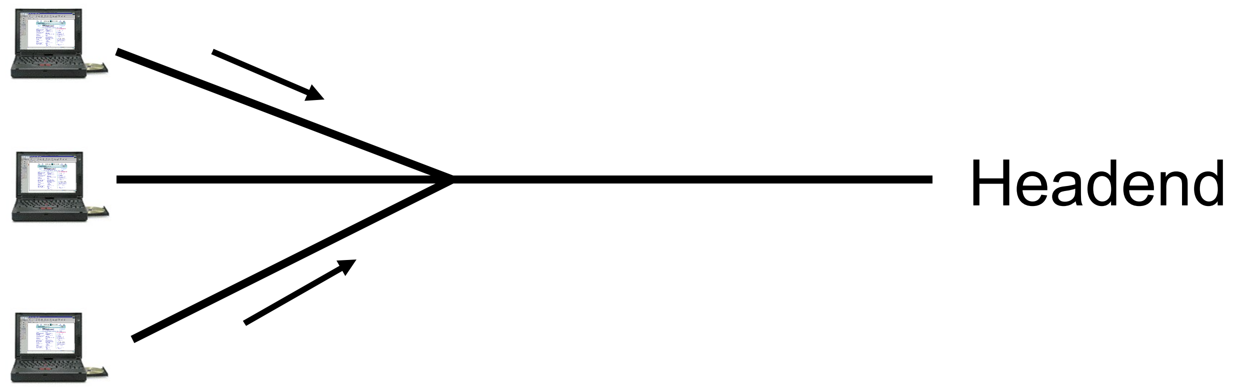


H: Hosts

R: Routers

2,000 ft: Multi-access (e.g., Cable)

- May need to share links with other senders



- Poll headend to receive a timeslot to send upstream
  - Headend controls all downstream transmissions
  - A lower level of addressing (than IP addresses) is used … why?

1,000 ft: Framing/Modulation

- Protect, delimit and modulate payload as signal

Sync / Unique	Header	Payload w/ error correcting code

- E.g, for cable, take payload, add error protection, header and framing, then turn into a signal
  - Modulate data to assigned channel and time (upstream)
  - Downstream, 6 MHz (~30 Mbps), Upstream ~2 MHz (~3 Mbps)

Part 3. Protocols and Layering

- We need abstractions to handle all this system complexity

A protocol is an agreement dictating the form and function of data exchanged between parties to effect communication

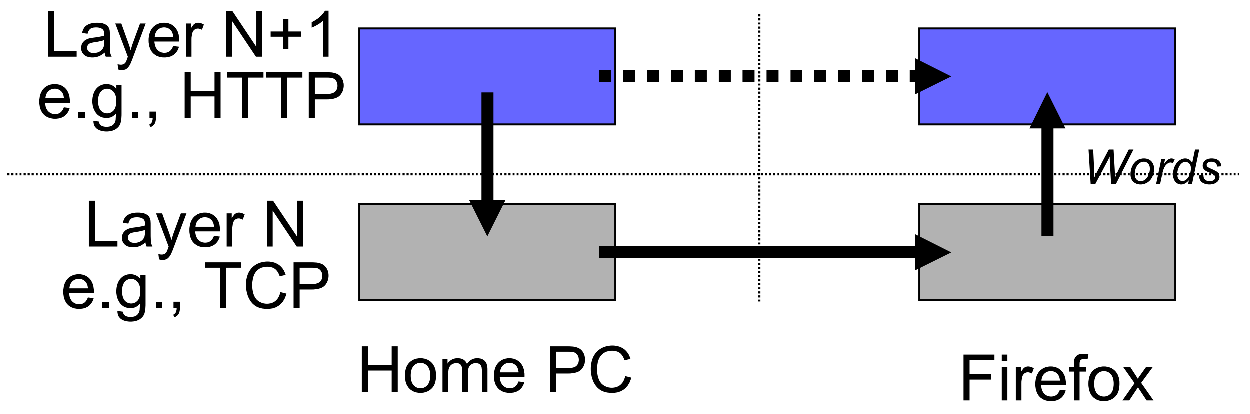
- Two parts:
  - Syntax: format -- where the bits go
  - Semantics: meaning -- what the words mean, what to do with them
- Examples:
  - Ordering food from a drive-through window
  - TCP/IP, the Internet protocol
  - HTTP, for the Web

Protocol Standards

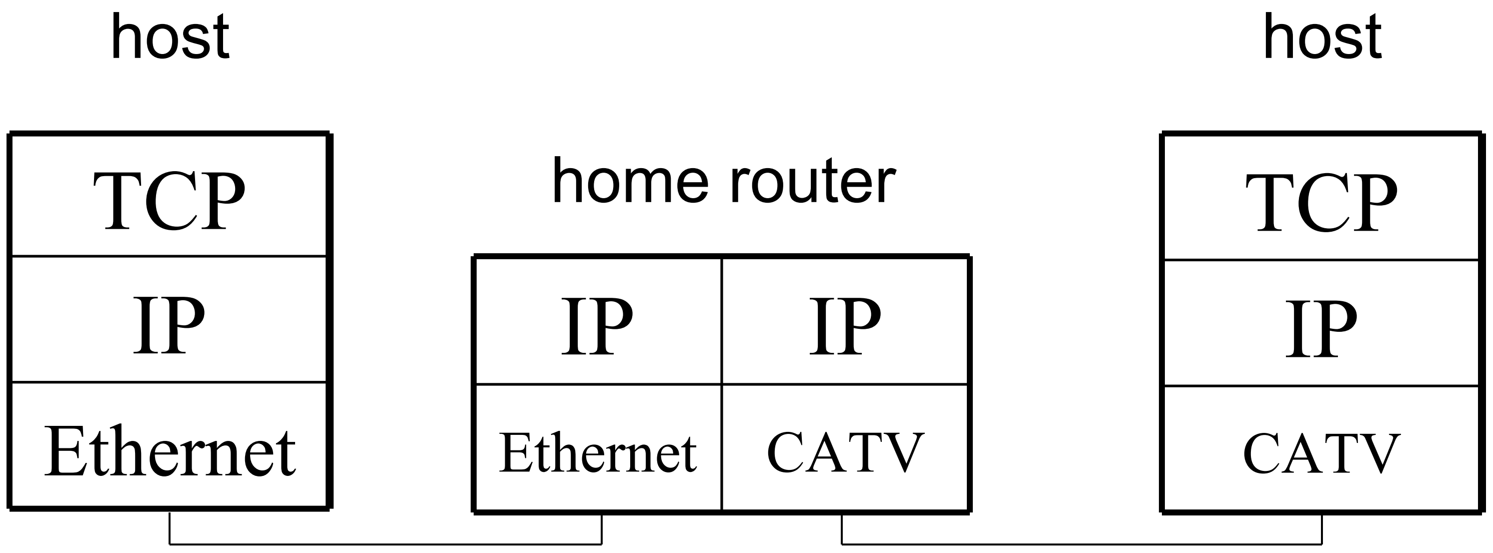
- Different functions require different protocols
- Thus there are many, many protocol standards
  - E.g., IP, TCP, UDP, HTTP, DNS, FTP, SMTP, NNTP, ARP, Ethernet/802.3, 802.11, RIP, OSPF, 802.1D, NFS, ICMP, IGMP, DVMRP, IPSEC, PIM-SM, BGP, …
  - every distributed application requires a protocol…
- Organizations: IETF, IEEE, ITU
- IETF (www.ietf.org) specifies Internet-related protocols
  - RFCs (Requests for Comments)
  - “We reject kings, presidents and voting. We believe in rough consensus and running code.” – Dave Clark.

Layering and Protocol Stacks

- Layering is how we combine protocols
  - Higher level protocols build on services provided by lower levels
  - Peer layers communicate with each other



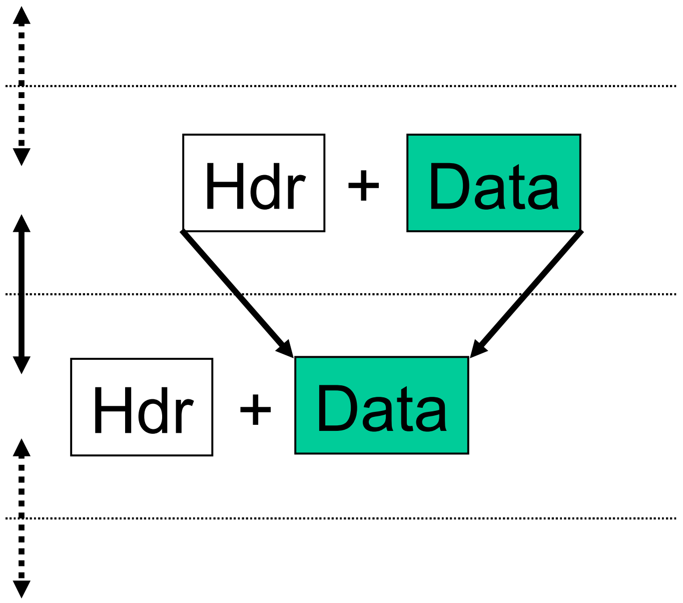
Example – Layering at work



Layering Mechanics

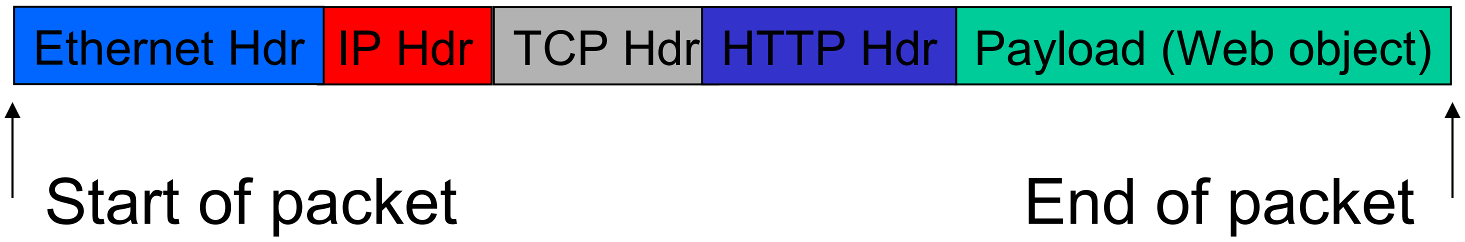
- Encapsulation and de(en)capsulation

Messages passed between layers



A Packet on the Wire

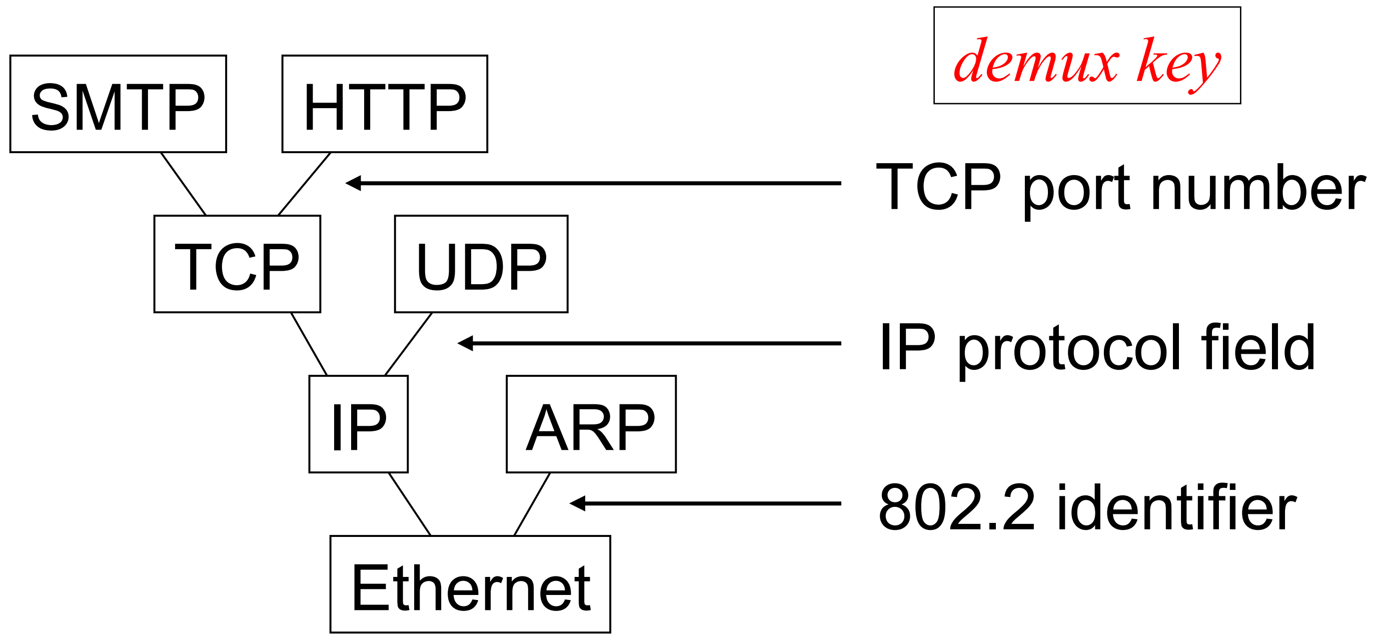
- Starts looking like an onion:



- This isn't entirely accurate
  - ignores segmentation and reassembly, Ethernet trailers, etc.
- But you can see that:
  - layering adds overhead
  - one protocol's header is another protocol's data

More Layering Mechanics

- Multiplexing and demultiplexing in a protocol graph



Part 4. OSI/Internet Protocol Stacks

Key Question: What functionality goes in which protocol?

- The "End to End Argument" (Reed, Saltzer, Clark, 1984):

  Functionality should be implemented at a lower layer only if it can be correctly and completely implemented. (Sometimes an incomplete implementation can be useful as a performance optimization.)

- Tends to push functions to the endpoints, which has aided the transparency and extensibility of the Internet.

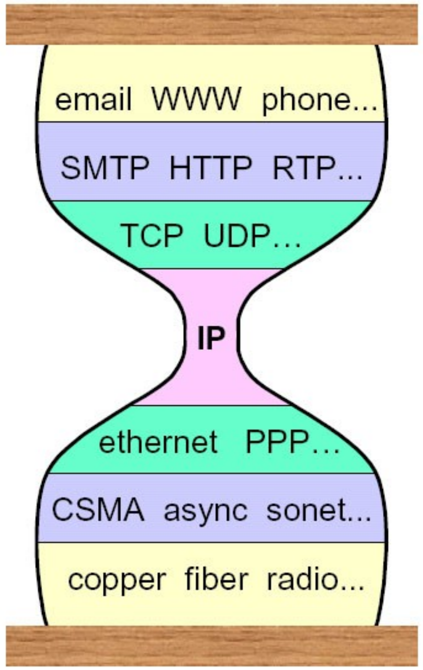
Internet Protocol Framework

Application
Transport
Network
Link

Model

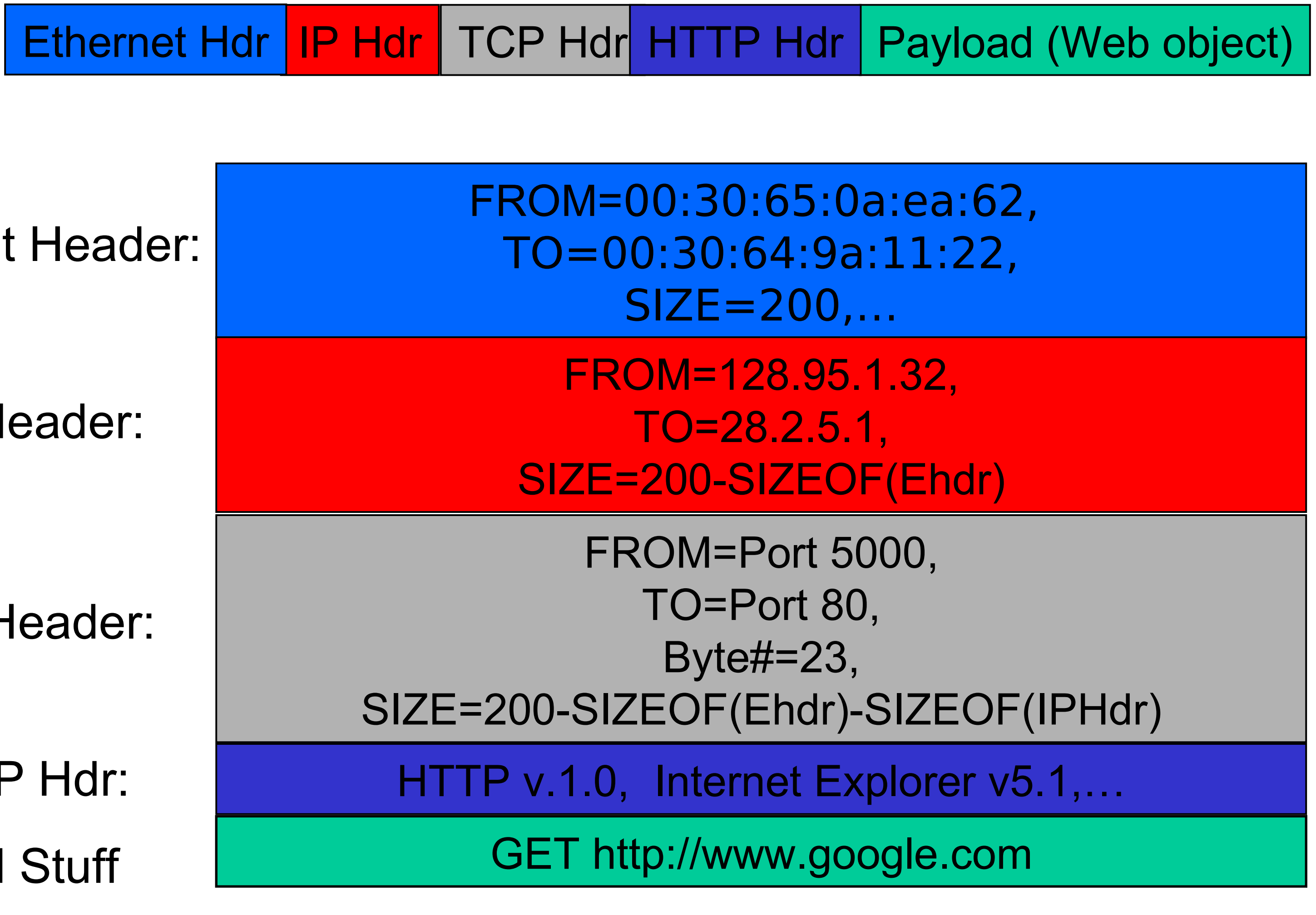
Many (HTTP,SMTP)
TCP / UDP
IP
Many (Ethernet, …)

Protocols

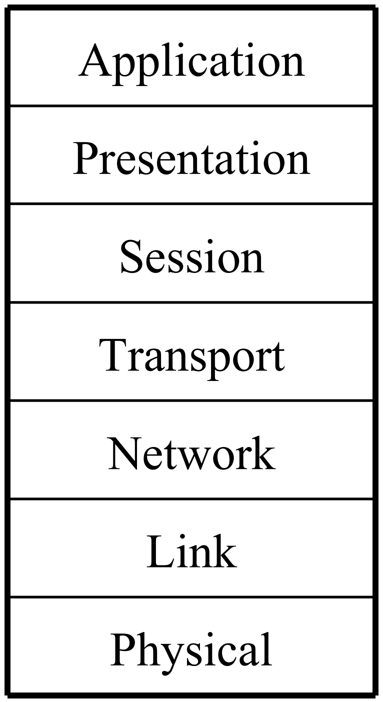


The “narrow waist”

What's Inside a Packet



OSI "Seven Layer" Reference Model



Their functions:

- Up to the application
- Encode/decode messages
- Manage connections
- Reliability, congestion control
- Routing
- Framing, multiple access
- Symbol coding, modulation

For next time

- Homework 0 is out (linked from syllabus page)
- Sections Thursday intended to help with the HW0
- For next class, please have read Chapter 1