Networking

Alan L. Cox
alc@rice.edu

Some slides adapted from CMU 15.213 slides

Objectives

Be exposed to the basic underpinnings of the Internet

Be able to use network socket interfaces effectively

Be exposed to the basic underpinnings of the World Wide Web

The 2004 A. M. Turing Award Goes to...

The 2004 A. M. Turing Award Goes to...

Bob Kahn Vint Cerf

"For pioneering work on internetworking, including the design and implementation of the Internet's basic communications protocols, TCP/IP, and for inspired leadership in networking."

The first Turing Award given to recognize work in computer networking

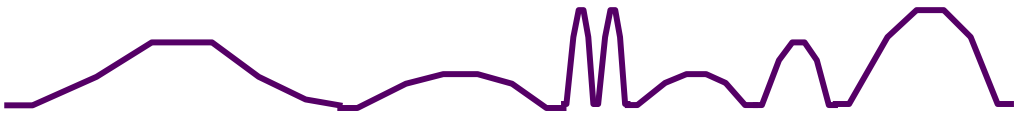
Telephony

Interactive telecommunication between people

Analog voice

- Transmitter/receiver continuously in contact with electronic circuit
- Electric current varies with acoustic pressure

Analog/Continuous Signal



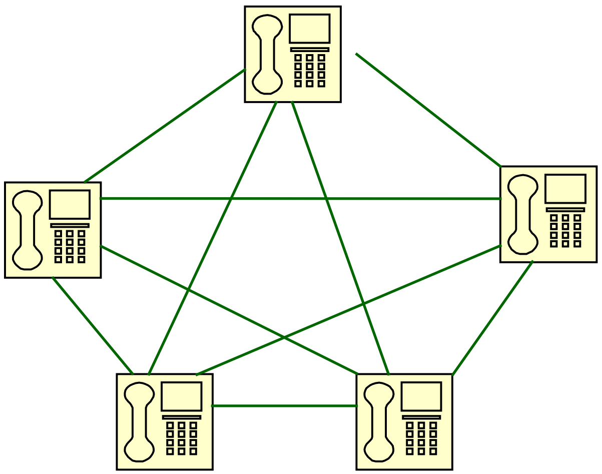
Over electrical circuits

Telephony Milestones

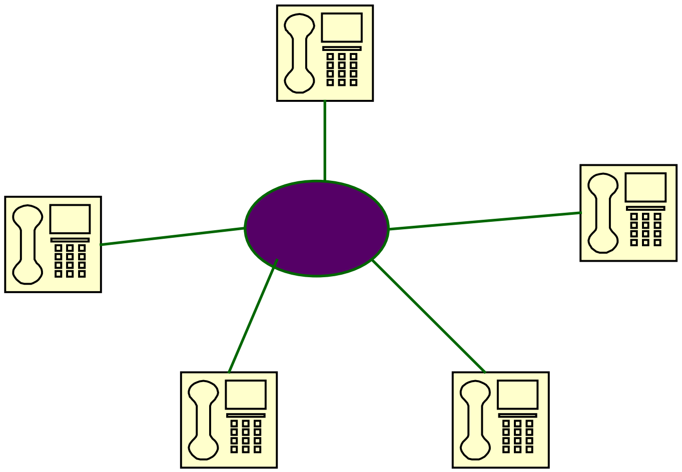
1876: Alexander Bell invented telephone

1878: Public switches installed at New Haven and San Francisco, public switched telephone network is born

- People can talk without being on the same wire!



Without Switch



With Switch

Telephony Milestones

1878: First telephone directory; White House line

1881: Insulated, balanced twisted pair as local loop

1885: AT&T formed

1892: First automatic commercial telephone switch

1903: 3 million telephones in U.S.

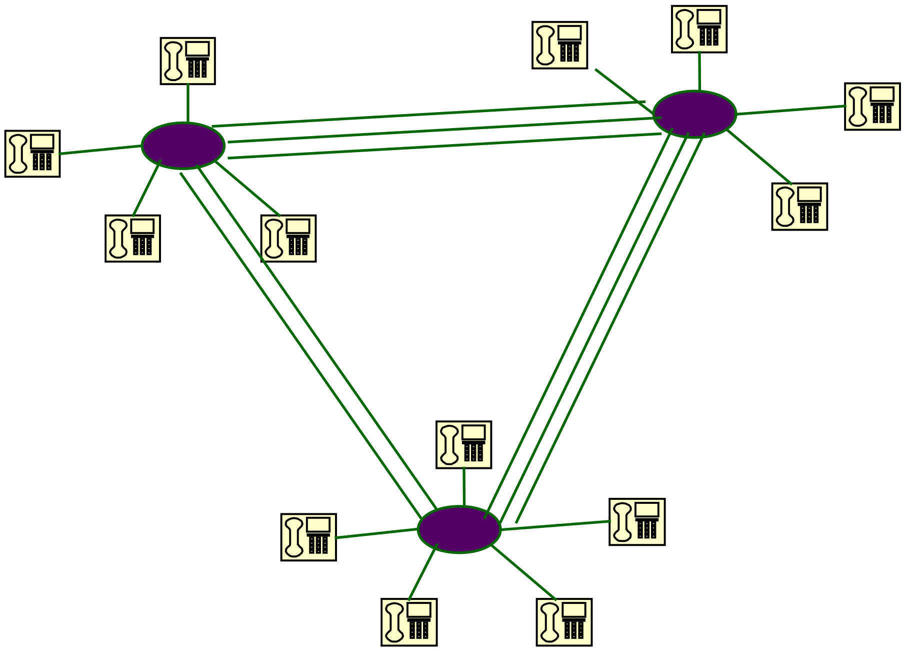
1915: First transcontinental telephone line

1927: First commercial transatlantic commercial service

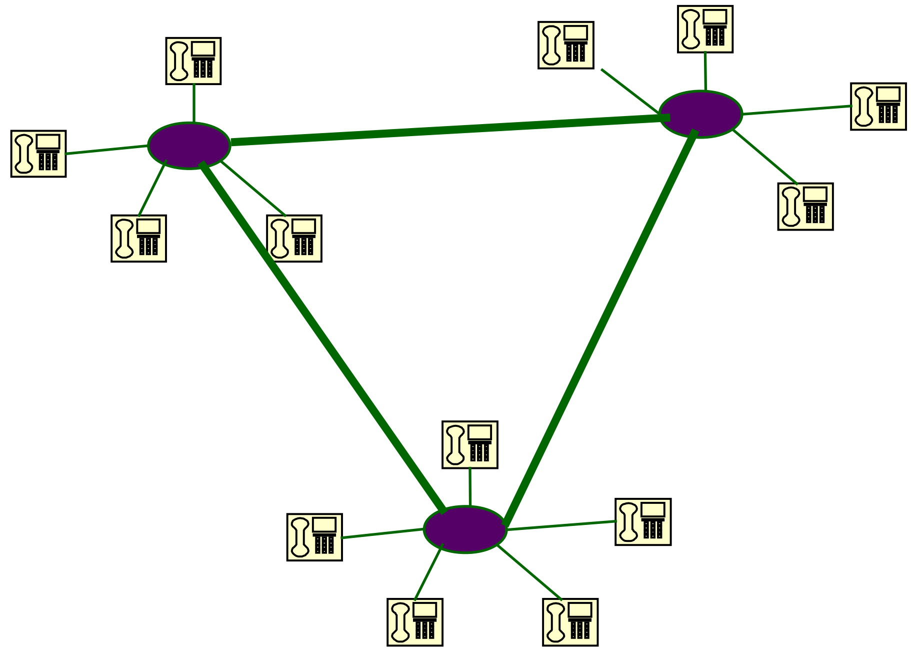
Telephony Milestones

1937: Multiplexing introduced for inter-city calls

- One link carries multiple conversations



Without Multiplexing



With Multiplexing

Data or Computer Networks

- Networks designed for computers to computers or devices
  - vs. communication between human beings
- Digital information
  - vs. analog voice

Digital/Discrete Signal

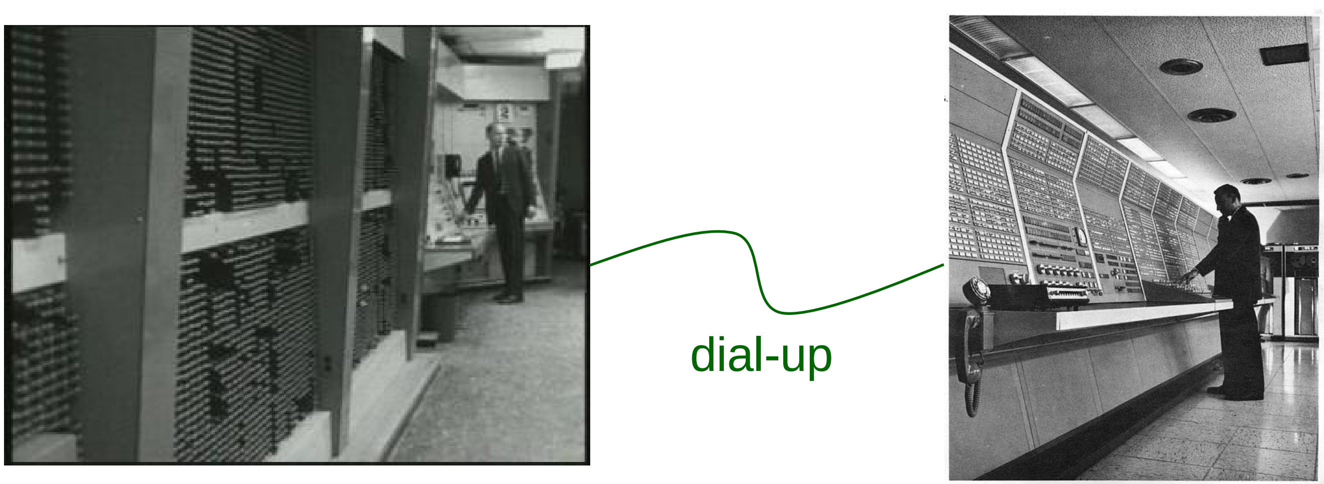
- Not a continuous stream of bits, rather, discrete “packets” with lots of silence in between
  - Dedicated circuit hugely inefficient
  - Packet switching invented

Major Internet Milestones

1960-1964 Basic concept of “packet switching” was independently developed by Baran (RAND), Kleinrock (MIT)

- AT&T insisted that packet switching would never work!

1965 First time two computers talked to each other using packets (Roberts, MIT; Marill, SDC)

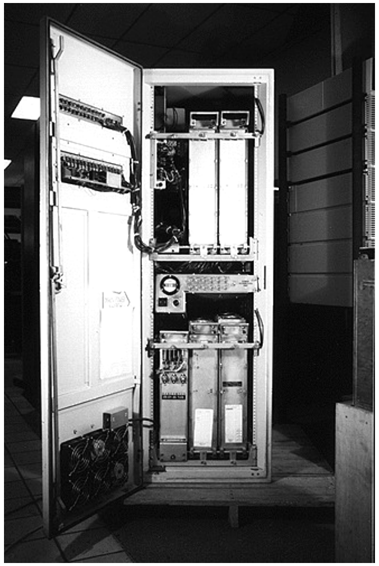


MIT TX-2

SDC Q32

Major Internet Milestones

1968 BBN group proposed to use Honeywell 516 mini-computers for the Interface Message Processors (i.e. packet switches)



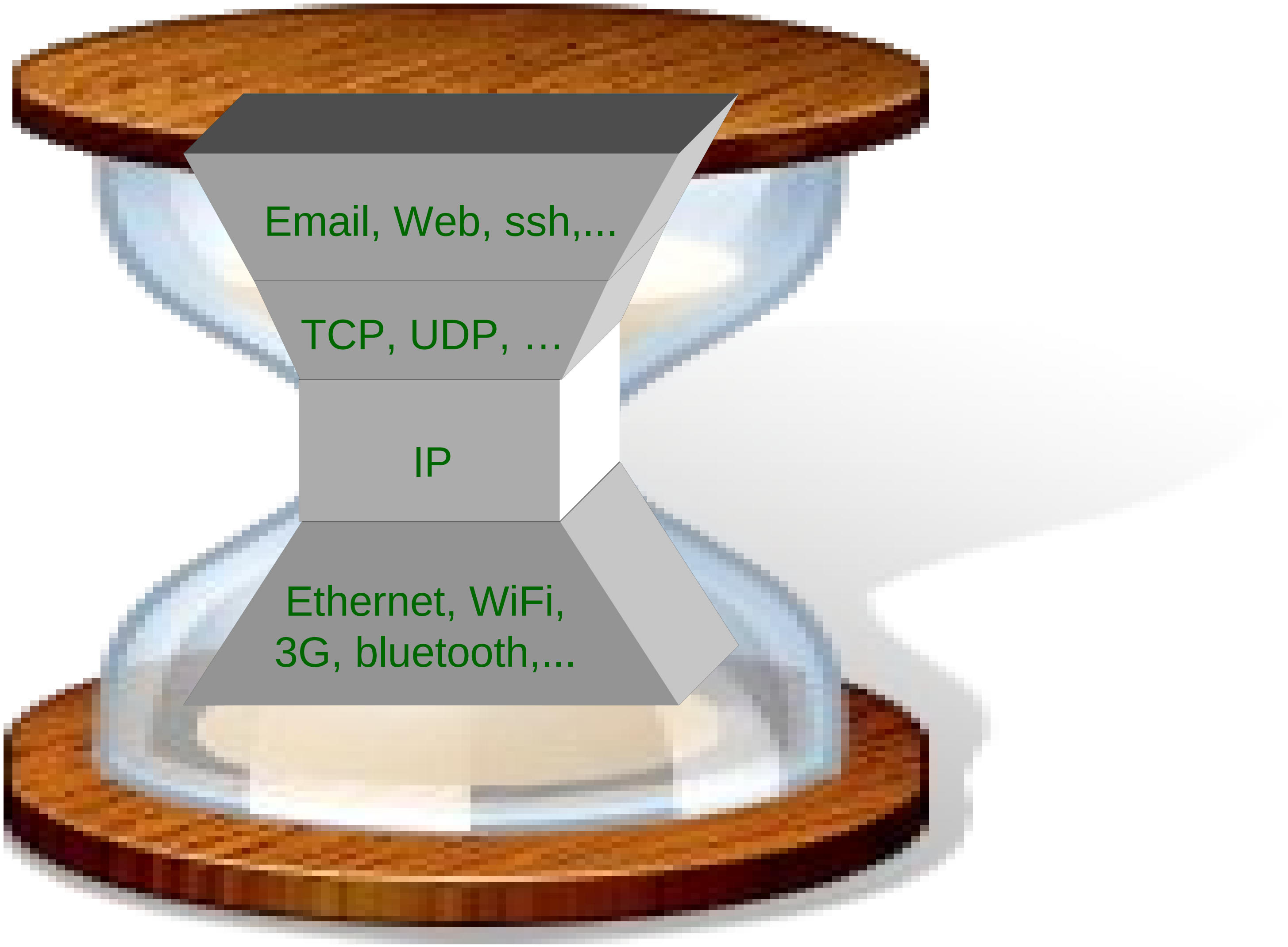
1969 The first ARPANET message transmitted between UCLA (Kleinrock) and SRI (Engelbart)

- We sent an “L”, did you get the “L”? Yep!
- We sent an “O”, did you get the “O”? Yep!
- We sent a “G”, did you get the “

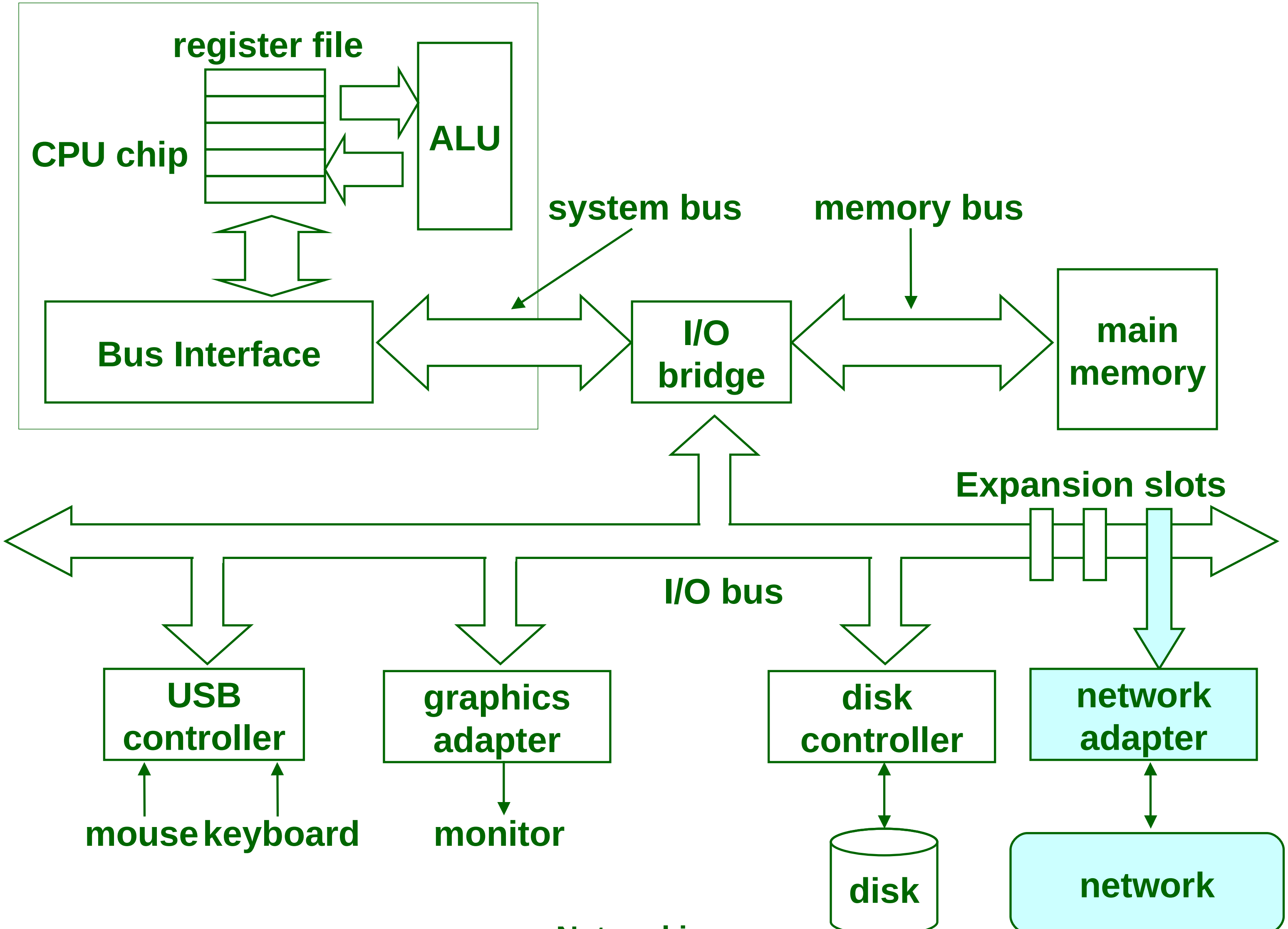
Major Internet Milestones

- 1970 First packet radio network ALOHANET (Abramson, U Hawaii)
- 1973 Ethernet invented (Metcalfe, Xerox PARC)
- Why is it called the “Inter-net”?
- 1974 “A protocol for Packet Network Interconnection” published by Cerf and Kahn
  - First internetworking protocol TCP
  - This paper was cited for their Turing Award
- 1977 First TCP operation over ARPANET, Packet Radio Net, and SATNET
- 1985 NSF commissions NSFNET backbone
- 1991 NSF opens Internet to commercial use

Internet Hourglass Architecture



Network Hardware



Computer Networks

A network is a hierarchical system of "boxes" and "wires" organized by geographical proximity

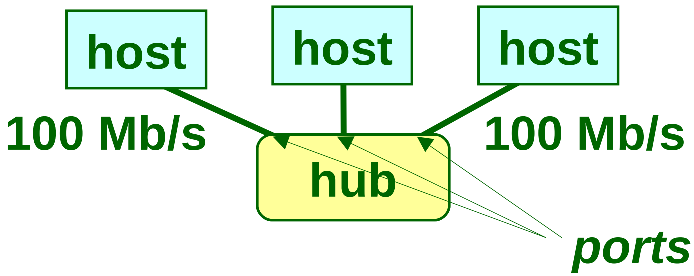
- Cluster network spans a rack or room
  - Ethernet, Infiniband, WiFi, …
- LAN (local area network) spans a building or campus
  - Switched Ethernet is most prominent example
- WAN (wide-area network) spans very long distance
  - A high-speed point-to-point link
  - Leased line or SONET/SDH circuit, or MPLS/ATM circuit

An internetwork (internet) is an interconnected set of networks

- The Global IP Internet (uppercase "I") is the most famous example of an internet (lowercase "i")

Lowest Level: Ethernet Segment

Ethernet segment consists of a collection of hosts connected by wires (twisted pairs) to a hub



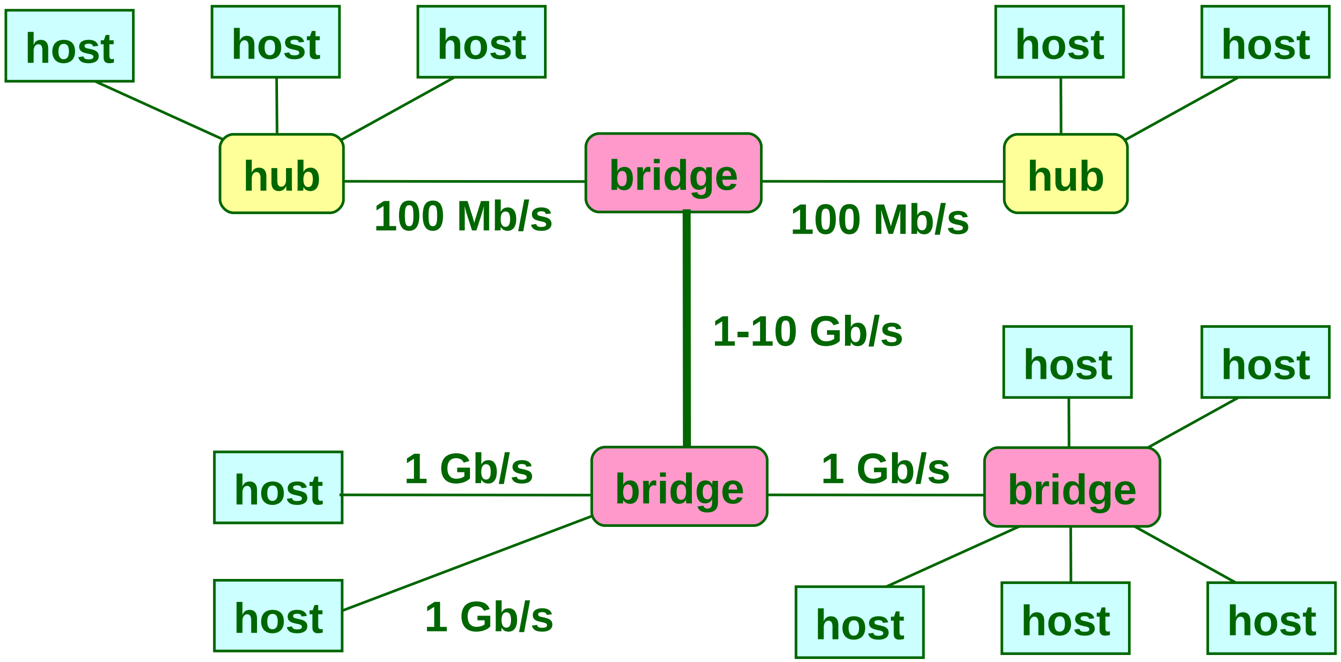
Operation

- Each Ethernet adapter has a unique 48-bit address
- Hosts send bits to any other host in chunks called frames
- Hub slavishly copies each bit from each port to every other port
  - Every host sees every bit
- Note: Hubs are largely obsolete
  - Bridges (switches, routers) became cheap enough to replace them (don’t broadcast all traffic)

Next Level: Bridged Ethernet Segment

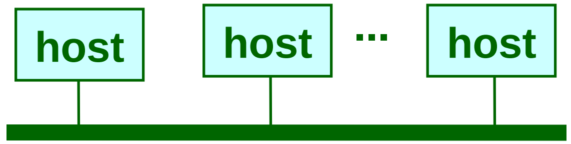
Spans room, building, or campus

Bridges cleverly learn which hosts are reachable from which ports and then selectively copy frames from port to port



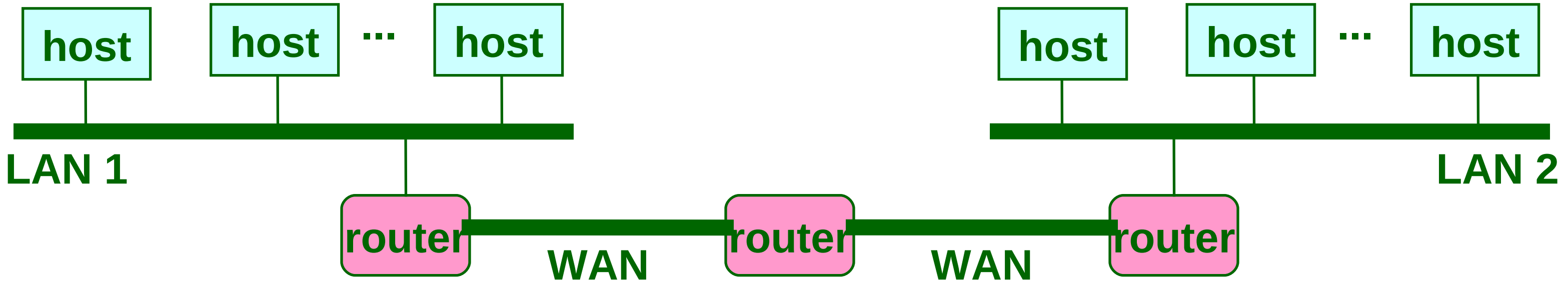
Conceptual View of LANs

For simplicity, hubs, bridges, and wires are often shown as a collection of hosts attached to a single wire:



Next Level: internets

- **Multiple incompatible LANs can be physically connected by specialized computers called routers**
- **The connected networks are called an internet**



LAN 1 and LAN 2 might be completely different, totally incompatible LANs (e.g., Ethernet and WiFi, 802.11*, T1-links, DSL, …)

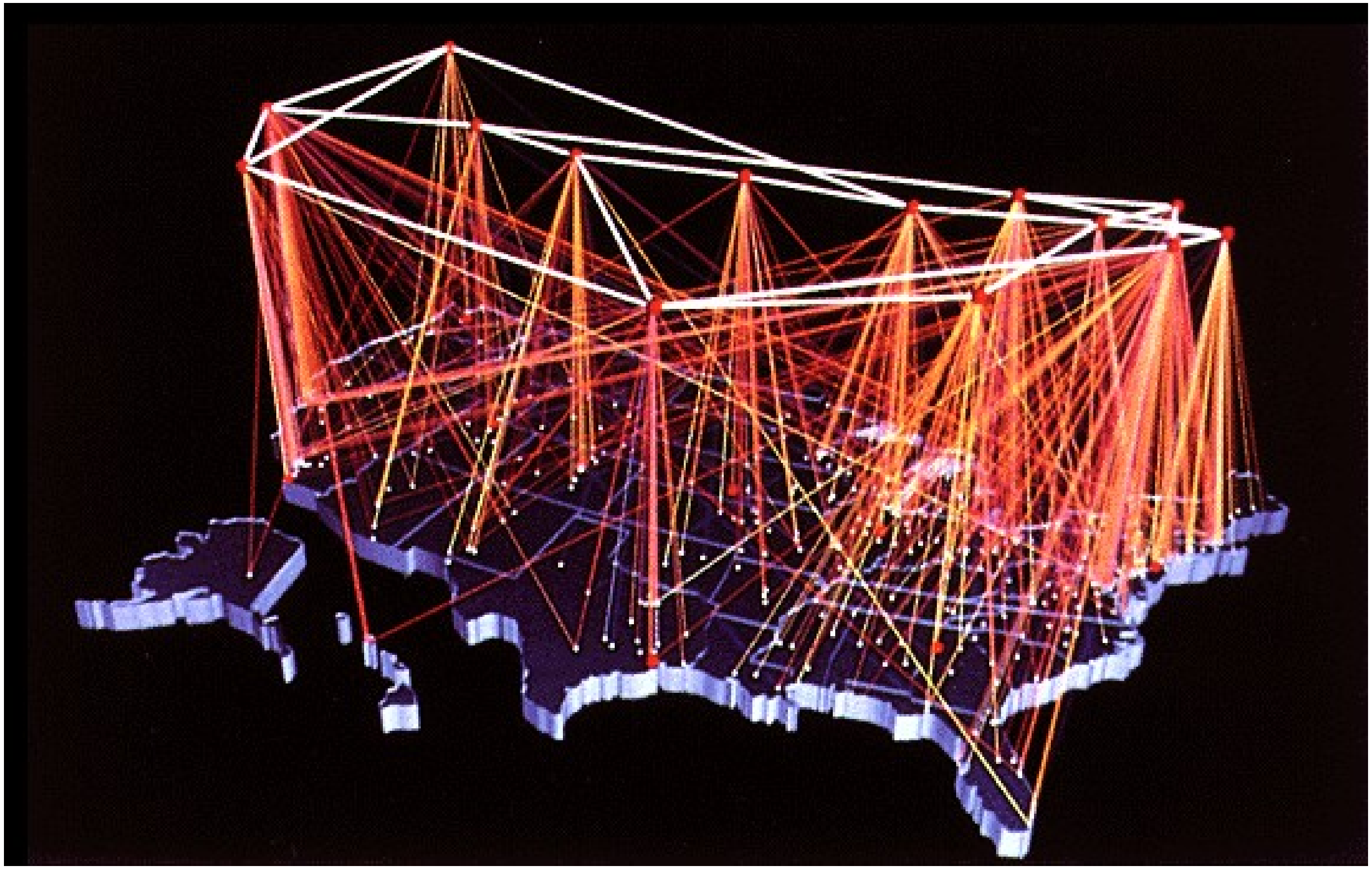
The Internet Circa 1986

In 1986, the Internet consisted of one backbone (NSFNET) that connected 13 sites via 45 Mbps T3 links

- Merit (Univ of Mich)
- NCSA (Illinois)
- Cornell Theory Center
- Pittsburgh Supercomputing Center
- San Diego Supercomputing Center
- John von Neumann Center (Princeton)
- BARRNet (Palo Alto)
- MidNet (Lincoln, NE)
- WestNet (Salt Lake City)
- NorthwestNet (Seattle)
- SESQUINET (Rice)
- SURANET (Georgia Tech)

Connecting to the Internet involved connecting one of your routers to a router at a backbone site, or to a regional network that was already connected to the backbone

NSFNET Internet Backbone



source: www.eef.org

After NSFNET

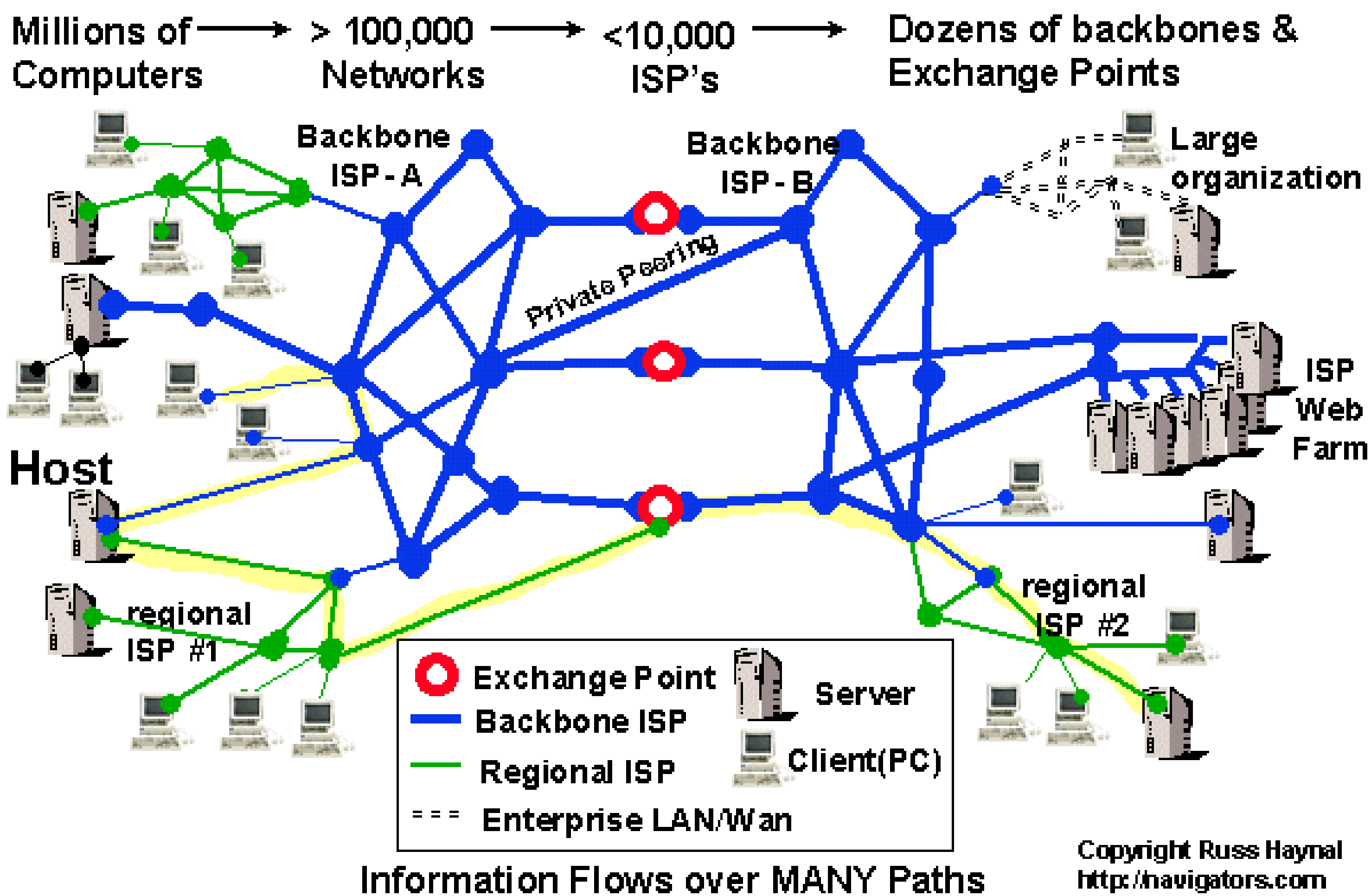
Early 90s

- Commercial enterprises began building their own high-speed backbones
- Backbone would connect to NSFNET, sell access to companies, ISPs, and individuals

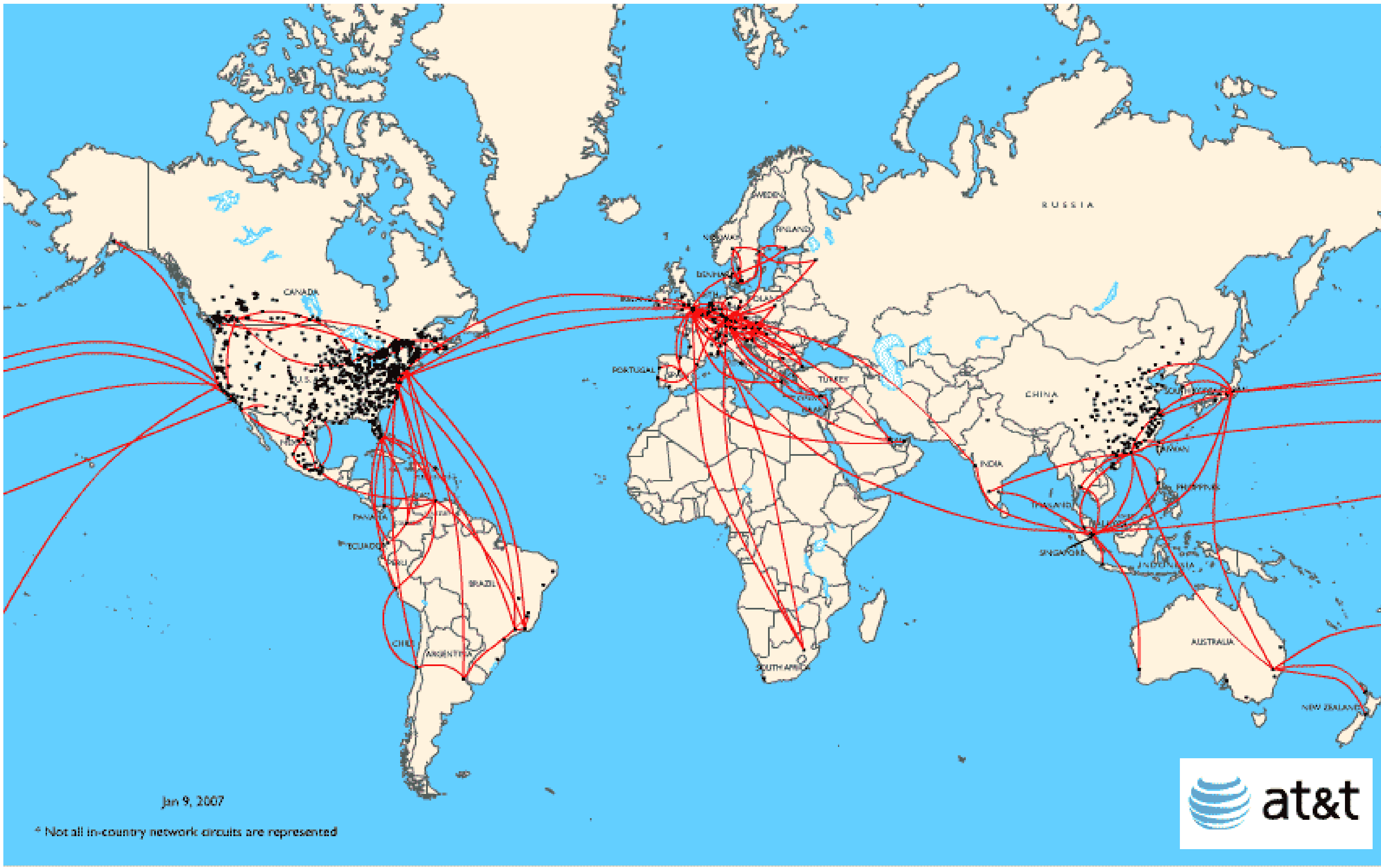
1995

- NSFNET decommissioned
- NSF fostered the creation of network access points (NAPs) to interconnect the commercial backbones

Current Internet Architecture



AT&T Backbone



The Notion of an internet Protocol

How is it possible to send bits across incompatible LANs and WANs?

Solution: protocol software running on each host and router smoothes out the differences between the different networks

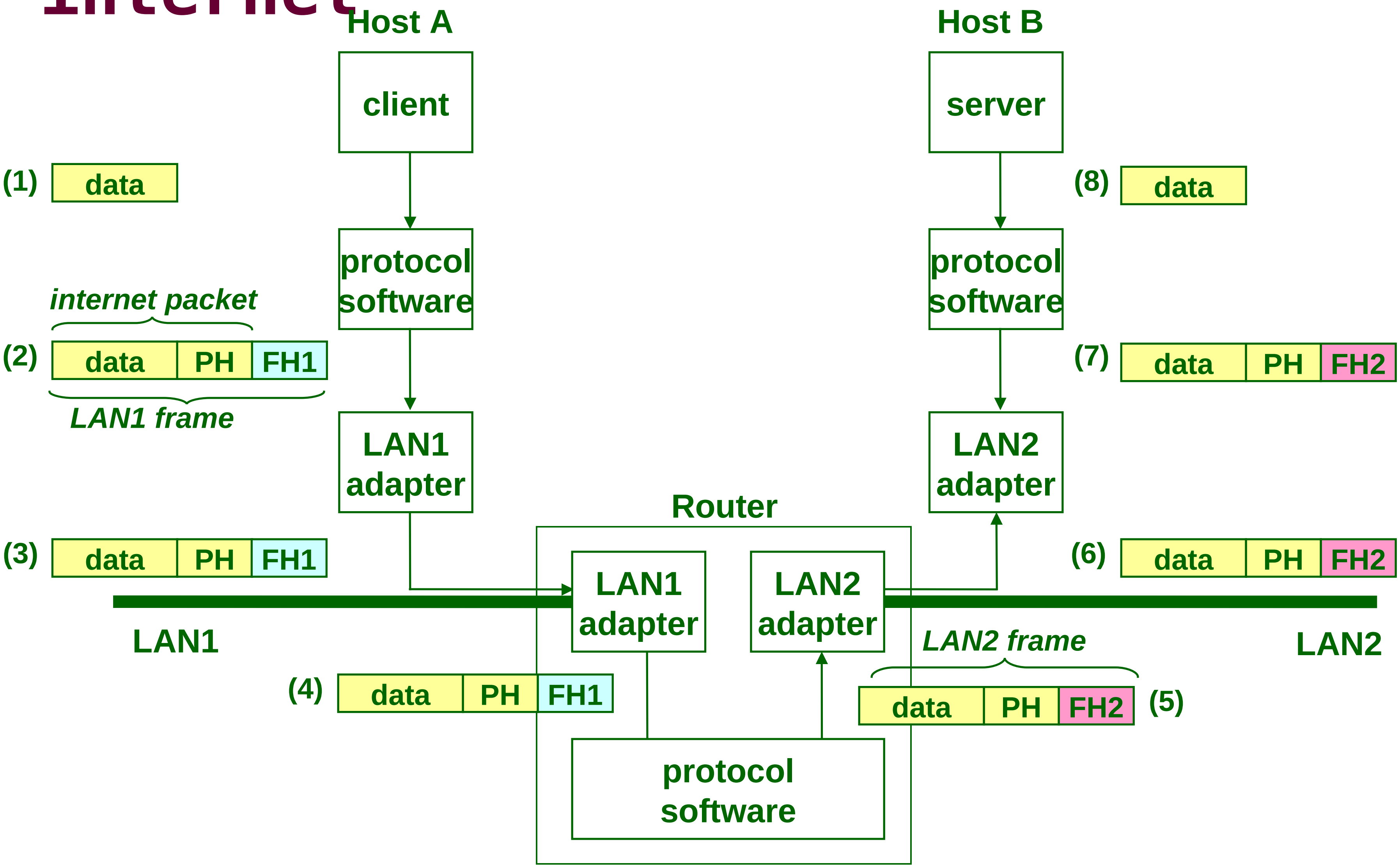
Implements an internet protocol (i.e., set of rules) that governs how hosts and routers should cooperate when they transfer data from network to network

- TCP/IP is the protocol for the global IP Internet

What Does an internet Protocol Do?

1. Provides a naming scheme
   - An internet protocol defines a uniform format for host addresses
   - Each host (and router) is assigned at least one of these internet addresses that uniquely identifies it
2. Provides a delivery mechanism
   - An internet protocol defines a standard transfer unit (packet)
   - Packet consists of header and payload
     - Header: contains info such as packet size, source and destination addresses
     - Payload: contains data bits sent from source host

Transferring Data Over an internet



Other Issues

We are glossing over a number of important questions:

- What if different networks have different maximum frame sizes? (segmentation)
- How do routers know where to forward frames?
- How are routers informed when the network topology changes?
- What if packets get lost?

We'll leave the discussion of these question to computer networking classes (COMP 429)

Global IP Internet

Based on the TCP/IP protocol family

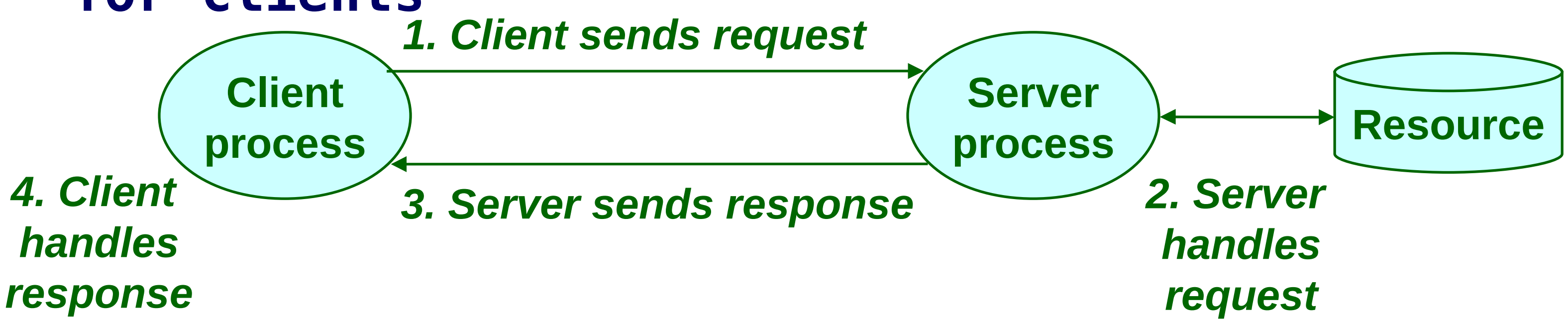
- IP (Internet protocol) :
  - Provides basic naming scheme and unreliable delivery capability of packets (datagrams) from host-to-host
- UDP (Unreliable Datagram Protocol)
  - Uses IP to provide unreliable datagram delivery from process-to-process
- TCP (Transmission Control Protocol)
  - Uses IP to provide reliable byte streams from process-to-process over connections

Accessed via a mix of Unix file I/O and functions from the sockets interface

A Client-Server Transaction

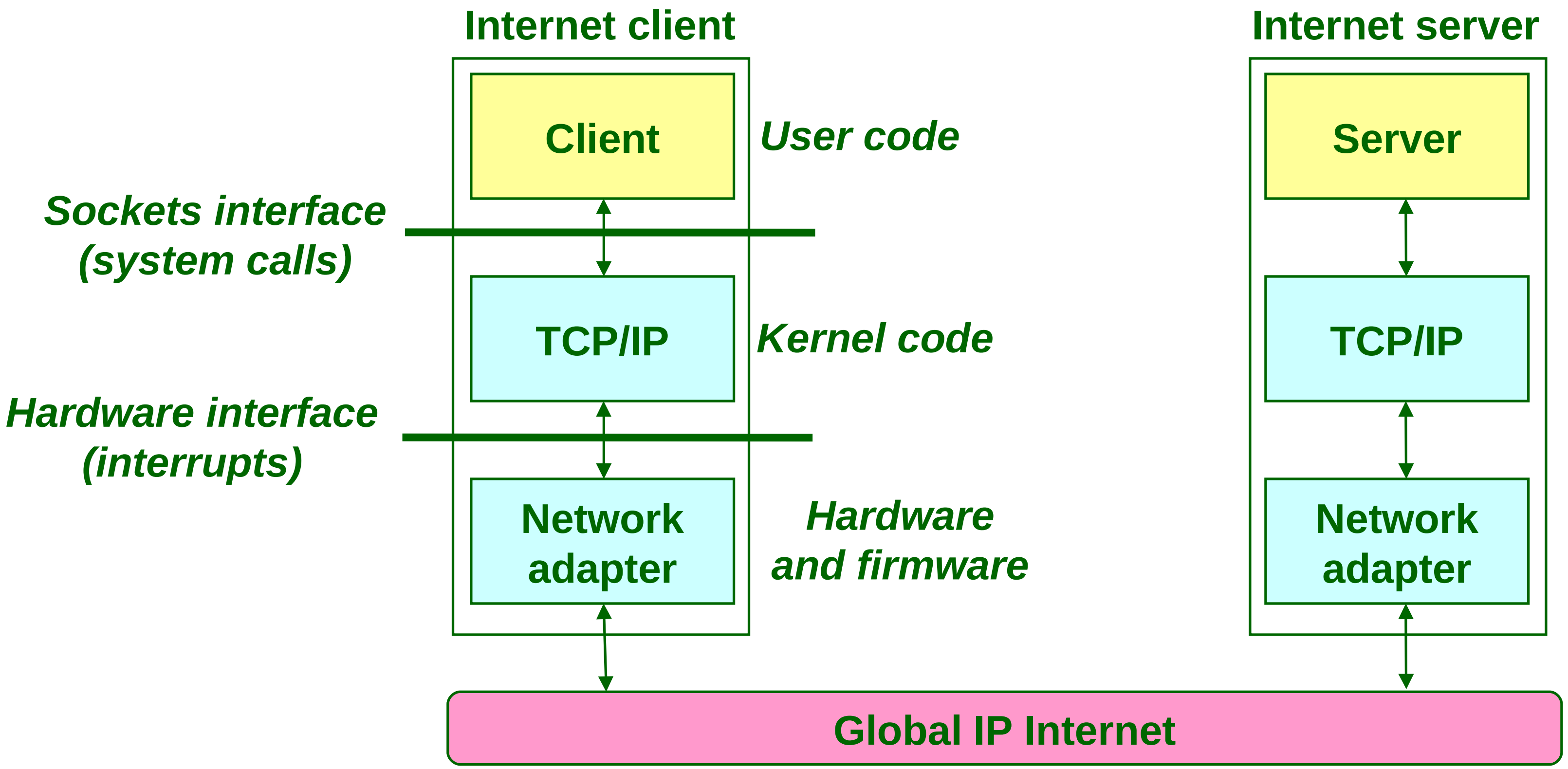
Most network applications are based on the client-server model:

- A server process and one or more client processes
- Server manages some resource
- Server provides service by manipulating resource for clients



Note: clients and servers are processes running on hosts (can be the same or different hosts)

Organization of an Internet Application



A Programmer’s View of the Internet

Hosts are mapped to a set of 32-bit IP addresses

- 128.42.128.17 (4 * 8 bits)

A set of identifiers called Internet domain names are mapped to the set of IP addresses for convenience

- www.cs.rice.edu is mapped to 128.42.128.17

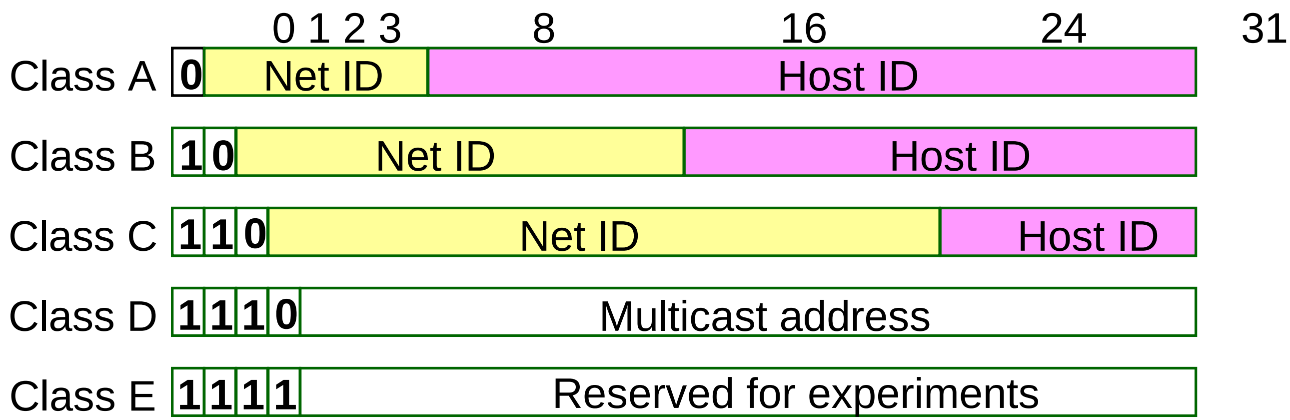
A process on one Internet host can communicate with a process on another Internet host over a connection

Dotted Decimal Notation

- By convention, each byte in a 32-bit IP address is represented by its decimal value and separated by a period
  - IP address 0x8002C2F2 = 128.2.194.242
- Functions for converting between binary IP addresses and dotted decimal strings:
  - inet_pton: converts a dotted decimal string to an IP address in network byte order
  - inet_ntop: converts an IP address in network by order to its corresponding dotted decimal string
  - “n” denotes network representation, “p” denotes presentation representation

IP Address Structure

IP (V4) Address space divided into classes:



Special Addresses for routers and gateways (all 0/1’s)

Loop-back address: 127.0.0.1

Unrouted (private) IP addresses:

- 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

Dynamic IP addresses (DHCP)

Domain Naming System (DNS)

The Internet maintains a mapping between IP addresses and domain names in a huge worldwide distributed database called DNS

- Conceptually, programmers can view the DNS database as a collection of millions of addrinfo structures:

```
struct addrinfo {
    int          ai_flags;     /* flags for getaddrinfo */
    int          ai_family;    /* address type (AF_INET or AF_INET6) */
    int          ai_socktype;  /* the socket type */
    int          ai_protocol;  /* the type of protocol */
    size_t       ai_addrlen;   /* length of ai_addr */
    struct sockaddr   *ai_addr;     /* pointer to a sockaddr struct */
    char         *ai_canonname;/* the canonical name */
    struct addrinfo *ai_next; /* pointer to the next addrinfo struct */
};
```

Functions for retrieving host entries from DNS:

- getaddrinfo: query DNS using domain name or IP
- getnameinfo: query DNS using sockaddr struct

Properties of DNS Host Entries

Each host entry is an equivalence class of domain names and IP addresses

Each host has a locally defined domain name localhost which always maps to the *loopback* address 127.0.0.1

Different kinds of mappings are possible:

- Simple case: 1 domain name maps to one IP address:
  - water.clear.rice.edu maps to 128.42.208.6
- Multiple domain names mapped to the same IP address:
  - www.cs.rice.edu, ececs.cs.rice.edu, and bianca.cs.rice.edu all map to 128.42.128.17
- Multiple domain names mapped to multiple IP addresses:
  - aol.com and www.aol.com map to multiple IP addresses
- Some valid domain names don’t map to any IP address:
  - for example: clear.rice.edu

Internet Connections

Clients and servers communicate by sending streams of bytes over connections:

- Point-to-point, full-duplex (2-way communication), and reliable

A socket is an endpoint of a connection

- Socket address is an IP address, port pair

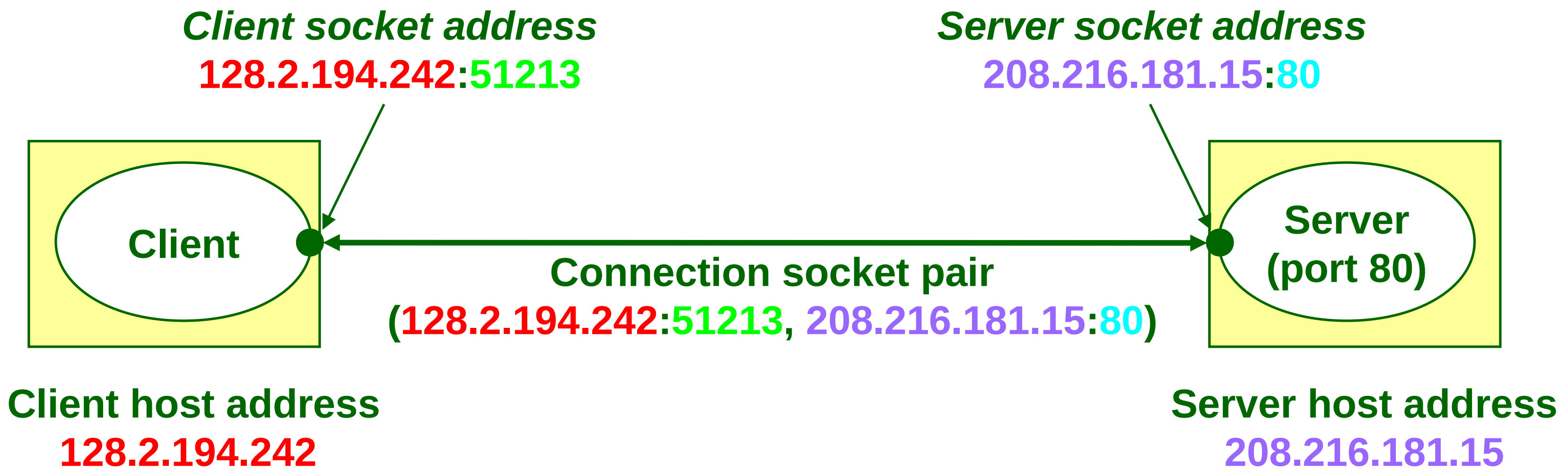
A port is a 16-bit integer that identifies a process:

- Ephemeral port: Assigned automatically on client when client makes a connection request
- Well-known port: Associated with some service provided by a server (e.g., port 80 is associated with Web servers)

A connection is uniquely identified by the socket addresses of its endpoints (socket pair)

- (cliaddr:cliport, servaddr:servport)

Putting it all Together: Anatomy of an Internet Connection



Clients

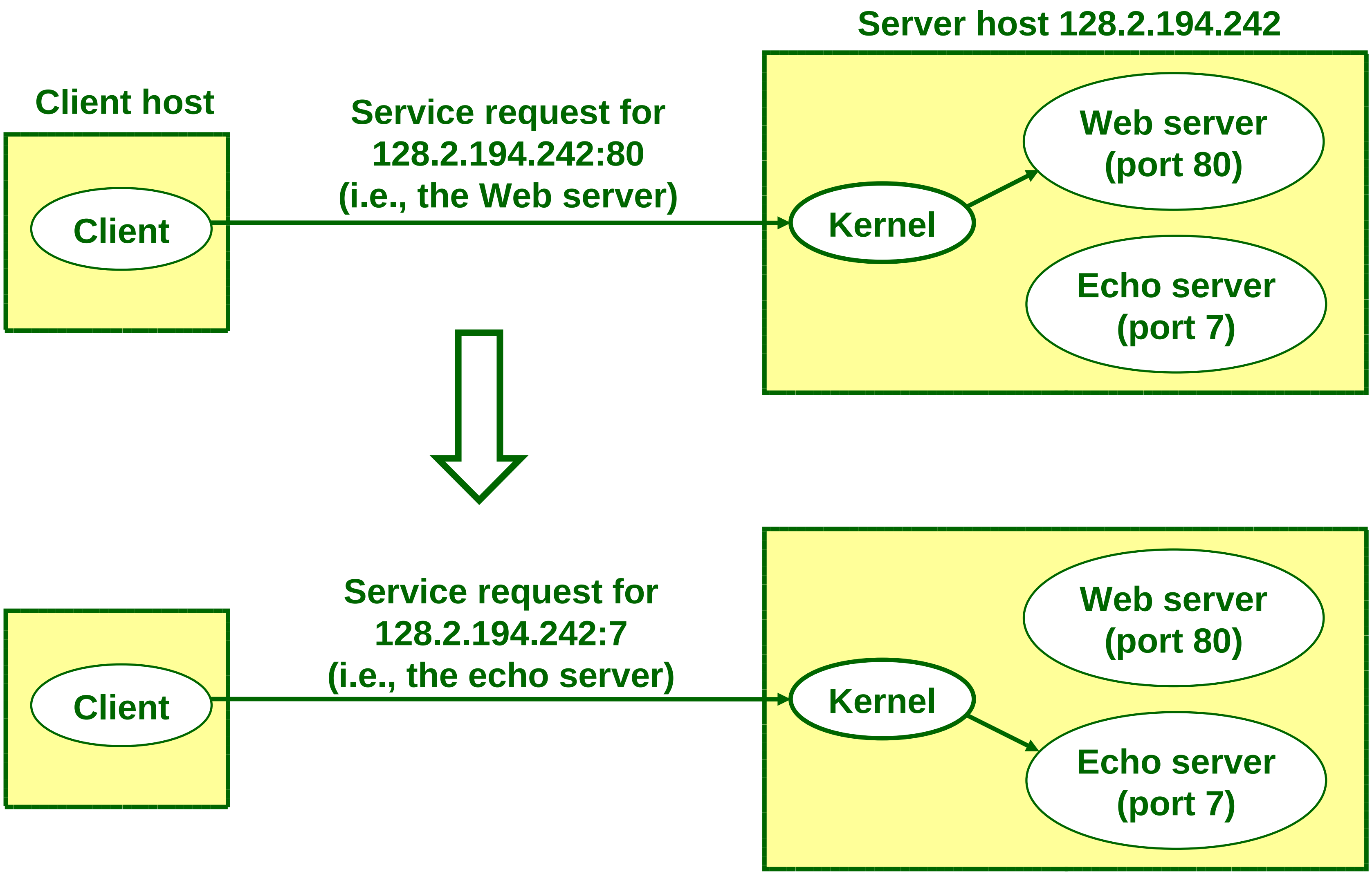
Examples of client programs

- Web browsers, email, ssh

How does a client find the server?

- The IP address in the server socket address identifies the host (more precisely, an adapter on the host)
- The (well-known) port in the server socket address identifies the service, and thus implicitly identifies the server process that performs that service
- Examples of well known ports
  - Port 7: Echo server
  - Port 22: Ssh server
  - Port 25: Mail server
  - Port 80: Web server

Using Ports to Identify Services



Servers

- Servers are long-running processes (daemons)
  - Created at boot-time (typically) by the init process (process 1)
  - Run continuously until the machine is turned off
- Each server waits for requests to arrive on a well-known port associated with a particular service
  - Port 7: echo server
  - Port 22: ssh server
  - Port 25: mail server
  - Port 80: HTTP (“Web”) server
- A machine that runs a server process is also often referred to as a “server”

Server Examples

Web server (port 80)

- Resource: files/compute cycles (CGI programs)
- Service: retrieves files and runs CGI programs on behalf of the client

> See `/etc/services` for a comprehensive list of the services available on a UNIX machine

Ssh server (port 22)

- Resource: terminal
- Service: proxies a terminal on the server machine

Mail server (port 25)

- Resource: email “spool” file
- Service: stores mail messages in spool file

Sockets Interface

Created in the early 80’s as part of the original Berkeley distribution of Unix that contained an early version of the Internet protocols

Provides a user-level interface to the network

Underlying basis for all Internet applications

Based on client/server programming model

Sockets

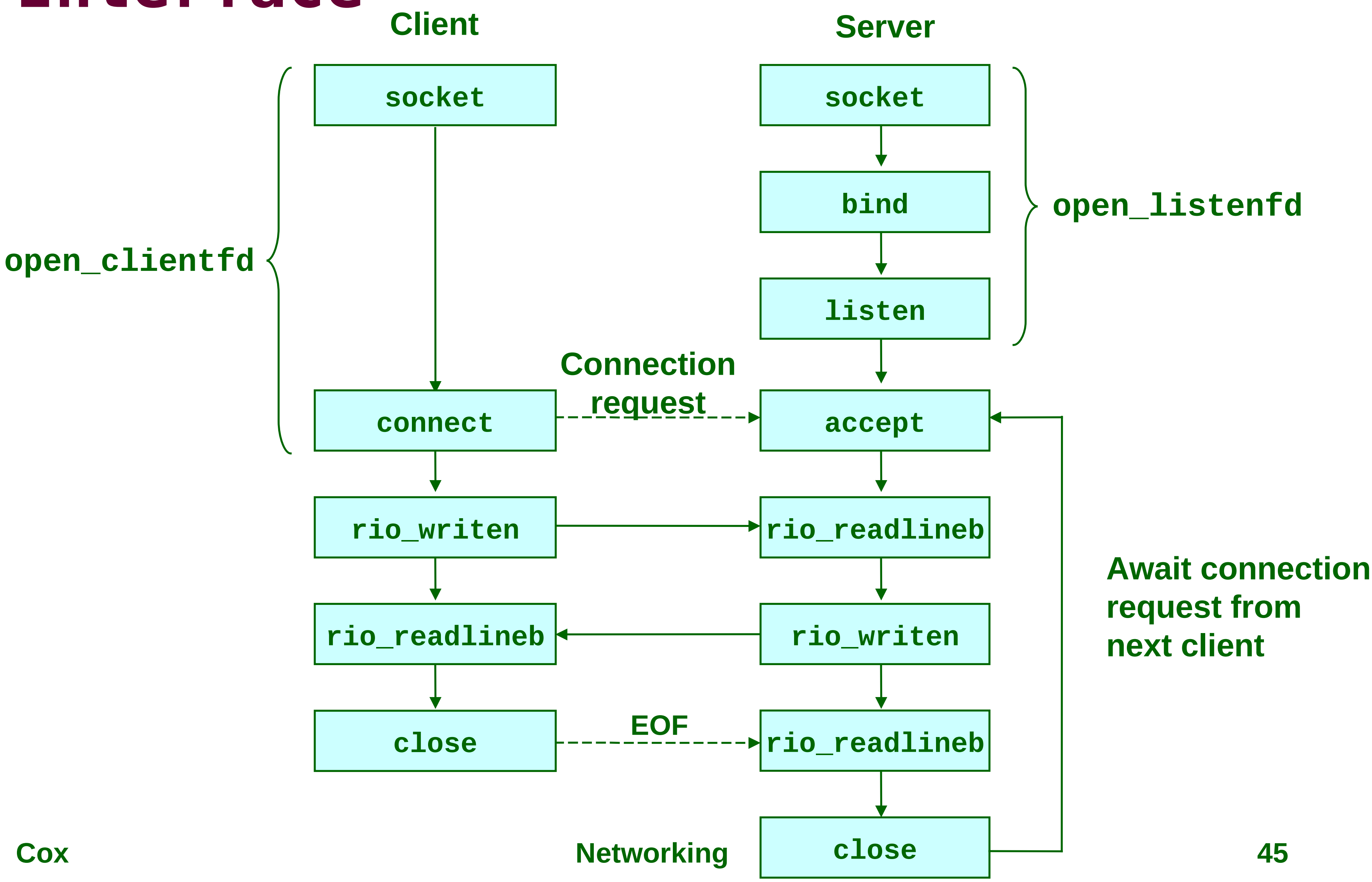
What is a socket?

- To the kernel, a socket is an endpoint of communication
- To an application, a socket is a file descriptor that lets the application read/write from/to the network
  - Remember: all Unix I/O devices, including networks, are modeled as files

Clients and servers communicate with each other by reading from and writing to socket descriptors

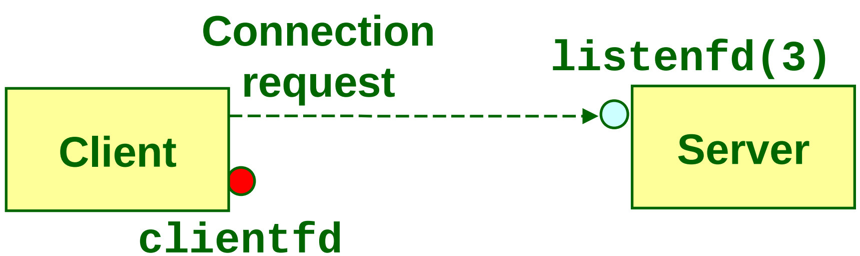
The main distinction between regular file I/O and socket I/O is how the application “opens” the socket descriptors

Overview of the Sockets Interface



Echo Server: `accept` Illustrated

1. Server blocks in `accept`, waiting for connection request on listening descriptor `listenfd`



2. Client makes connection request by calling and blocking in `connect`

3. Server returns `connfd` from `accept`. Client returns from `connect`. Connection is now established between `clientfd` and `connfd`

Connected vs. Listening Descriptors

Listening descriptor

- End point for client connection requests
- Created once and exists for lifetime of the server

Connected descriptor

- End point of the connection between client and server
- A new descriptor is created each time the server accepts a connection request from a client
- Exists only as long as it takes to service client

Why the distinction?

- Allows for concurrent servers that can communicate over many client connections simultaneously
  - E.g., Each time we receive a new request, we fork a child to handle the request

Web History

1945:

- Vannevar Bush, “As we may think”, Atlantic Monthly, July, 1945
  - Describes the idea of a distributed hypertext system
  - A “memex” that mimics the “web of trails” in our minds

1989:

- Tim Berners-Lee (CERN) writes internal proposal to develop a distributed hypertext system
  - Connects “a web of notes with links”
  - Intended to help CERN physicists in large projects share and manage information

1990:

- Tim Berners-Lee writes a graphical browser for Next machines

Web History (cont)

1992

- NCSA server released
- 26 WWW servers worldwide

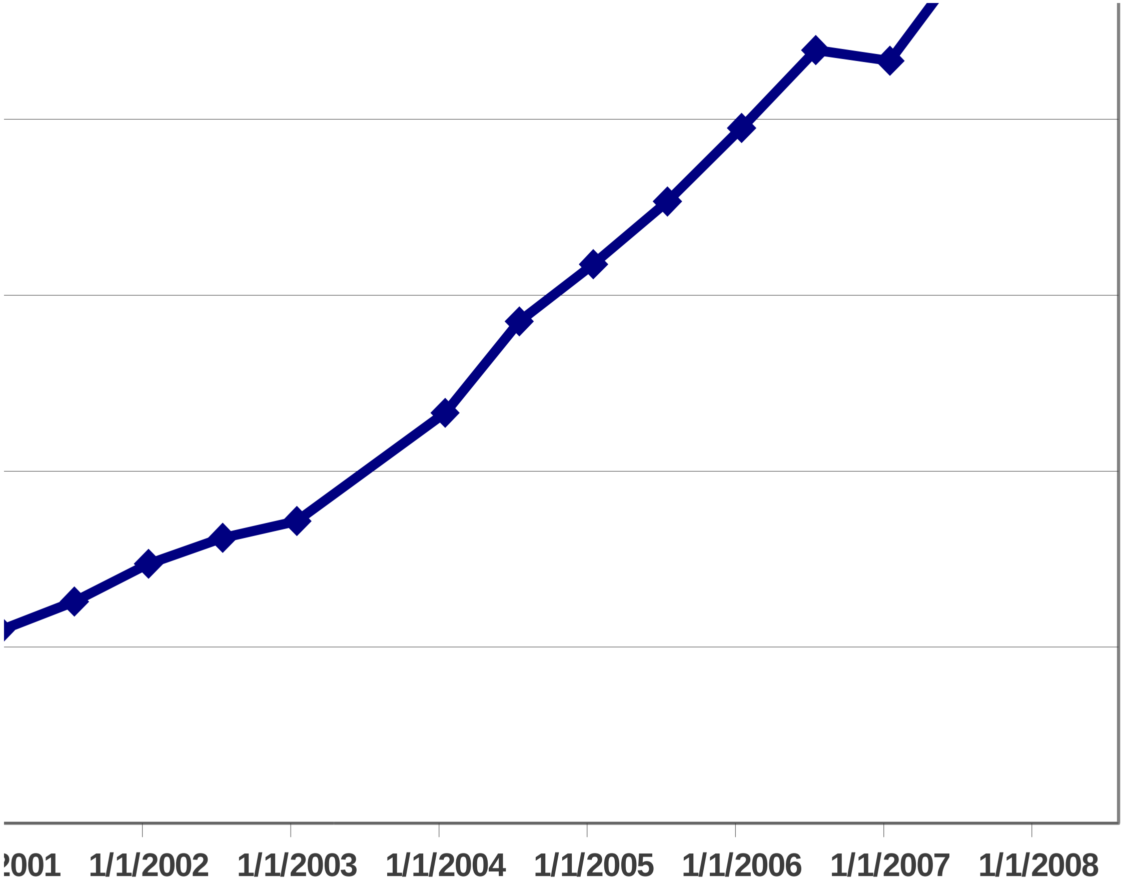
1993

- Marc Andreessen releases first version of NCSA Mosaic browser
- Mosaic version released for (Windows, Mac, Unix)
- Web (port 80) traffic at 1% of NSFNET backbone traffic
- Over 200 WWW servers worldwide

1994

- Andreessen and colleagues leave NCSA to form "Mosaic Communications Corp" (became Netscape, then part of AOL)

Internet Hosts

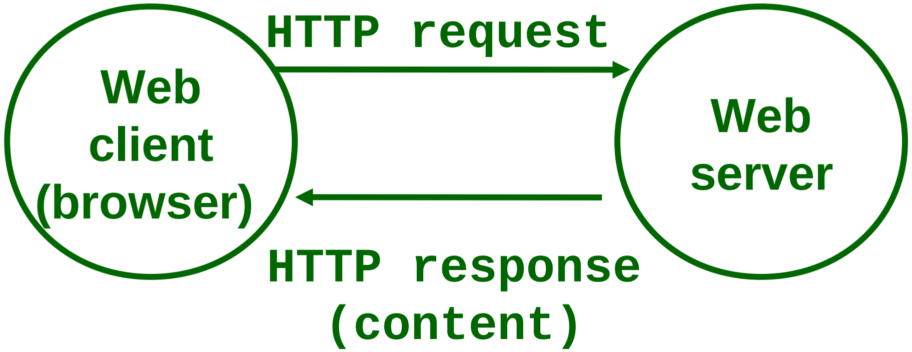


Source: www.isc.org

Web Servers

Clients and servers communicate using the HyperText Transfer Protocol (HTTP)

- Client and server establish TCP connection
- Client requests content
- Server responds with requested content
- Client and server (may) close connection



HTTP/1.1 is still the most widely used

- RFC 2616, 1999
- HTTP/2, 2015

Web Content

Web servers return content to clients

- content: a sequence of bytes with an associated MIME (Multipurpose Internet Mail Extensions) type

Example MIME types

- text/html HTML document
- text/plain Unformatted text
- application/postscript Postcript document
- image/gif Binary image (GIF format)
- image/jpeg Binary image (JPEG format)

Static and Dynamic Content

The content returned in HTTP responses can be either *static* or *dynamic*

- Static content: content stored in files and retrieved in response to an HTTP request
  - Examples: HTML files, images, audio clips
- Dynamic content: content produced on-the-fly in response to an HTTP request
  - Example: content produced by a program executed by the server on behalf of the client (i.e., search results)

Bottom line: All Web content is associated with a file or program that is managed by the server

URLs

Each file managed by a server has a unique name called a URL (Universal Resource Locator)

URLs for static content:

- http://www.rice.edu:80/index.html
- http://www.rice.edu/index.html
- http://www.rice.edu
  - Identifies a file called index.html, managed by a Web server at www.rice.edu that is listening on port 80

URLs for dynamic content:

- http://www.cs.cmu.edu:8000/cgi-bin/adder?15000&213
  - Identifies an executable file called adder, managed by a Web server at www.cs.cmu.edu that is listening on port 8000, that should be called with two argument strings: 15000 and 213

How Clients and Servers Use URLs

Example URL: http://www.aol.com:80/index.html

Clients use prefix (http://www.aol.com:80) to infer:

- What kind of server to contact (http (Web) server)
- Where the server is (www.aol.com)
- What port the server is listening on (80)

Servers use suffix (/index.html) to:

- Determine if request is for static or dynamic content
  - No hard and fast rules for this
  - Historically executables resided in cgi-bin directory
- Find file on file system
  - Initial "/" in suffix denotes home directory for requested content
  - Minimal suffix is "/", which servers expand to some default home page (e.g., index.html)

Testing Servers Using telnet

The telnet program is invaluable for testing servers that transmit ASCII strings over Internet connections

- Our simple echo server
- Web servers
- Mail servers

Usage:

- unix> telnet <host> <portnumber>
- Creates a connection with a server running on <host> and listening on port <portnumber>

Anatomy of an HTTP Transaction

unix> ***telnet www.rice.edu 80***	*Client: open connection to server*
Trying 128.42.206.11...	*Telnet prints 3 lines to the terminal*
Connected to www.netfu.rice.edu.	
Escape character is '^]'.	
GET / HTTP/1.1	*Client: request line*
Host: www.rice.edu	*Client: required HTTP/1.1 HOST header*
	Client: empty line terminates headers
HTTP/1.1 200 OK	*Server: response line*
Date: Thu, 5 Apr 2018 <..snip..>	*Server: followed by 8 response headers*
Server: Apache/2.4.6 (Red Hat Enterprise Linux) <..snip..>	
Accept-Ranges: bytes	
<..snip..>	
Transfer-Encoding: chunked	
Content-Type: text/html; charset=UTF-8	
	Server: empty line ("\r\n") terminates hdrs
15e3	*Server: first line in response body*
<..snip..>	*Server: HTML content not shown.*
0	*Server: last line in response body*
Connection closed by foreign host.	*Server: closes connection*
unix>	*Client: closes connection and terminates*

HTTP Requests

HTTP request is a request line, followed by zero or more request headers

Request line: <method> <uri> <version>

- <method> is either GET, POST, OPTIONS, HEAD, PUT, DELETE, or TRACE
- <uri> is typically URL for proxies, URL suffix for servers
- <version> is HTTP version of request (HTTP/1.0 or HTTP/1.1)

HTTP Requests (cont)

HTTP methods:

- GET: Retrieve static or dynamic content
  - Arguments for dynamic content are in URI
  - Workhorse method (99% of requests)
- POST: Retrieve dynamic content
  - Arguments for dynamic content are in the request body
- OPTIONS: Get server or file attributes
- HEAD: Like GET but no data in response body
- PUT: Write a file to the server!
- DELETE: Delete a file on the server!
- TRACE: Echo request in response body
  - Useful for debugging

HTTP Requests (cont)

Request headers: <header name>: <header data>

- Provide additional information to the server

Major differences between HTTP/1.1 and HTTP/1.0

- HTTP/1.0 uses a new connection for each transaction
- HTTP/1.1 also supports persistent connections
  - Multiple transactions over the same connection
  - Connection: Keep-Alive
- HTTP/1.1 requires HOST header
  - Host: www.rice.com
- HTTP/1.1 adds additional support for caching

HTTP Responses

HTTP response is a response line followed by zero or more response headers

Response line: <version> <status code> <status msg>

- <version> is HTTP version of the response
- <status code> is numeric status
- <status msg> is corresponding English text
  - 200 OK Request was handled without error
  - 403 Forbidden Server lacks permission to access file
  - 404 Not found Server couldn’t find the file

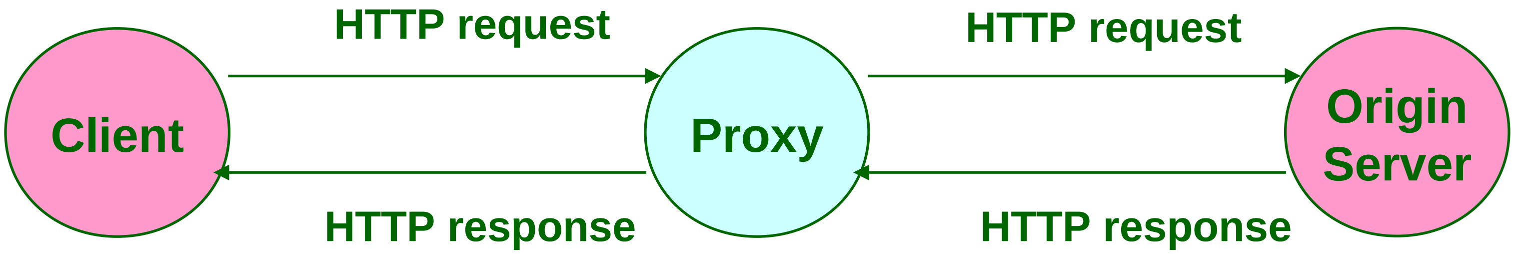
Response headers: <header name>: <header data>

- Provide additional information about response
- Content-Type: MIME type of content in response body
- Content-Length: Length of content in response body

Proxies

A *proxy* is an intermediary between a client and an origin server

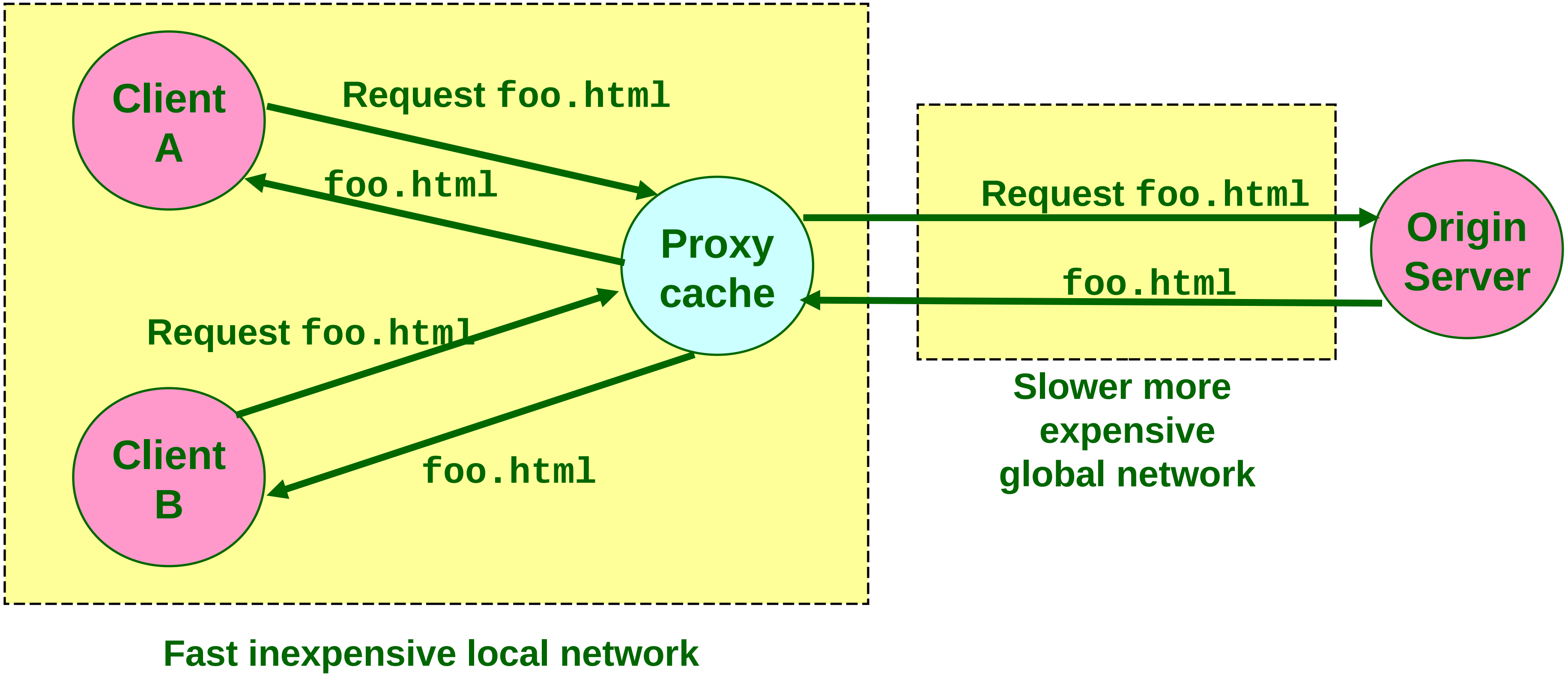
- To the client, the proxy acts like a server
- To the server, the proxy acts like a client



Why Proxies?

Can perform useful functions as requests and responses pass through

- **Examples: Caching, logging, anonymization**



A Client's Request Line To A Proxy

```
GET http://www.rice.edu/ HTTP/1.1
Host: www.rice.edu
Accept: text/html,application/xhtml+xml,application/xml;
 q=0.9,*/*;q=0.8
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cookie: SS_MID=4bff7079-1d9b-464e-a3ab-bee5762488a6i69kf1o6;
 __unam=bf980eb-15c0e074c94-a631ba-8;
 _ga=GA1.2.1181809510.1416263362; _gid=GA1.2.179111272.1523312176
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)
 AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1
 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
CRLF (\r\n)
```

The client's request line to a proxy must specify the full URL

For More Information

W. Richard Stevens, "Unix Network Programming: Networking APIs: Sockets and XTI", Volume 1, Second Edition, Prentice Hall, 1998.

- THE network programming bible

Complete versions of the echo client and server are developed in the text

- Available on the course web site
- You should compile and run them for yourselves to see how they work
- Feel free to borrow any of this code

Next Time

Concurrency