

Internetworking

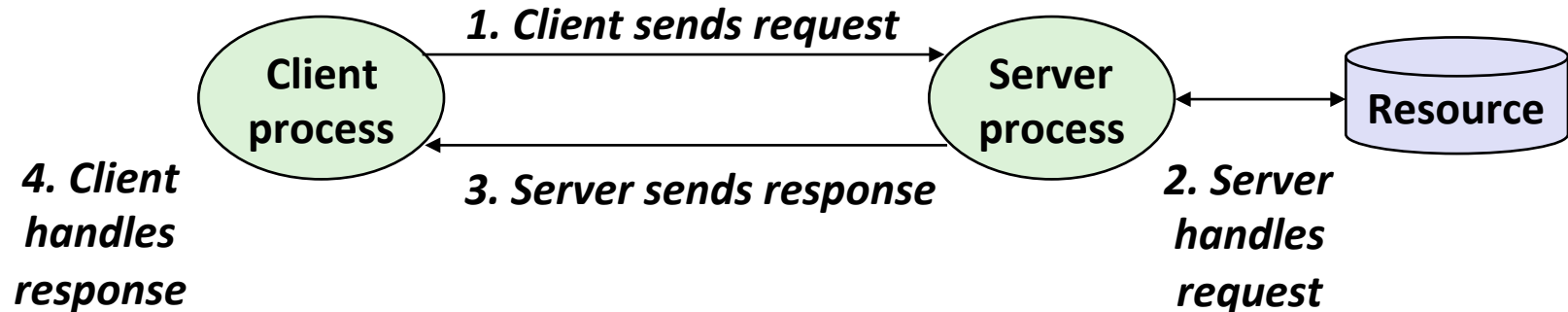
15-213/18-243: Introduction to Computer Systems

21th Lecture, Mar. 31, 2011

Instructors:

Gregory Kesden and Anthony Rowe

A Client-Server Transaction

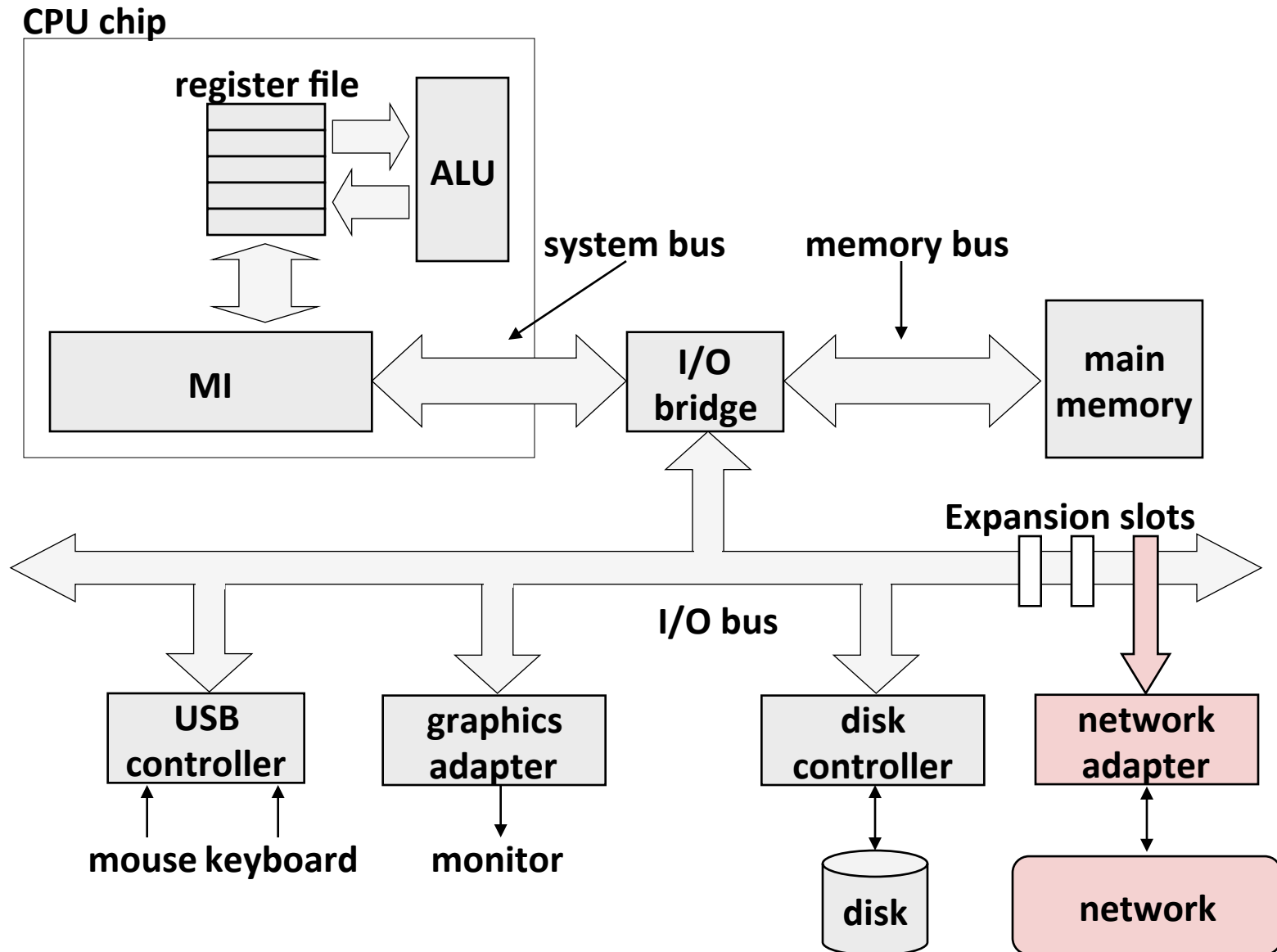


Note: clients and servers are processes running on hosts (can be the same or different hosts)

■ 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
- Server activated by request from client (vending machine analogy)

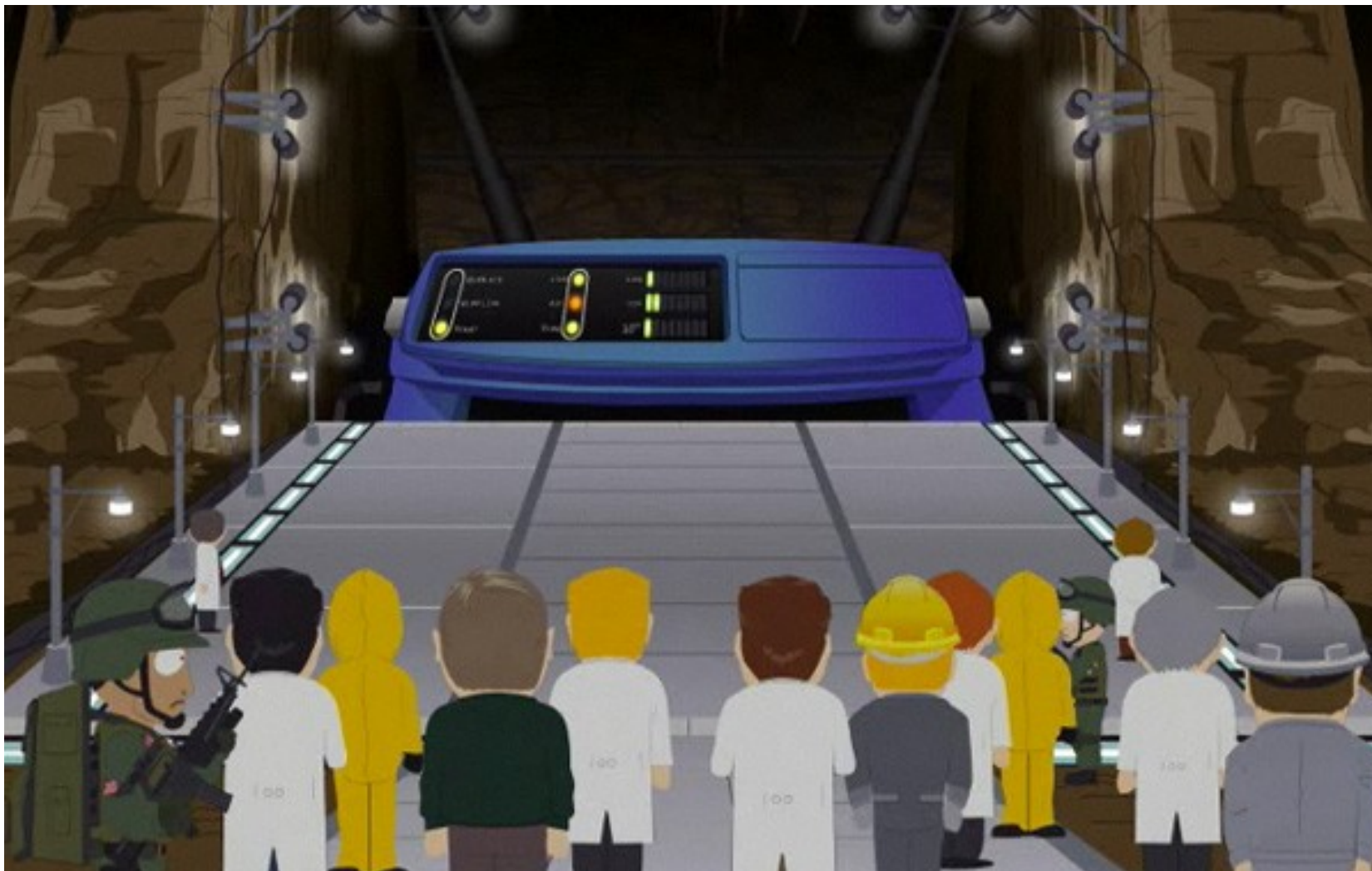
Hardware Organization of a Network Host



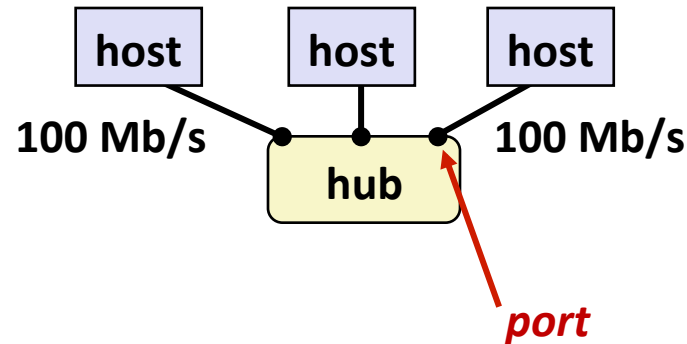
Computer Networks

- A ***network*** is a hierarchical system of boxes and wires organized by geographical proximity
 - SAN (System Area Network) spans cluster or machine room
 - Switched Ethernet, Quadrics QSW, ...
 - LAN (Local Area Network) spans a building or campus
 - Ethernet is most prominent example
 - WAN (Wide Area Network) spans country or world
 - Typically high-speed point-to-point phone lines
- 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”)
- Let’s see how an internet is built from the ground up

What is the Internet?

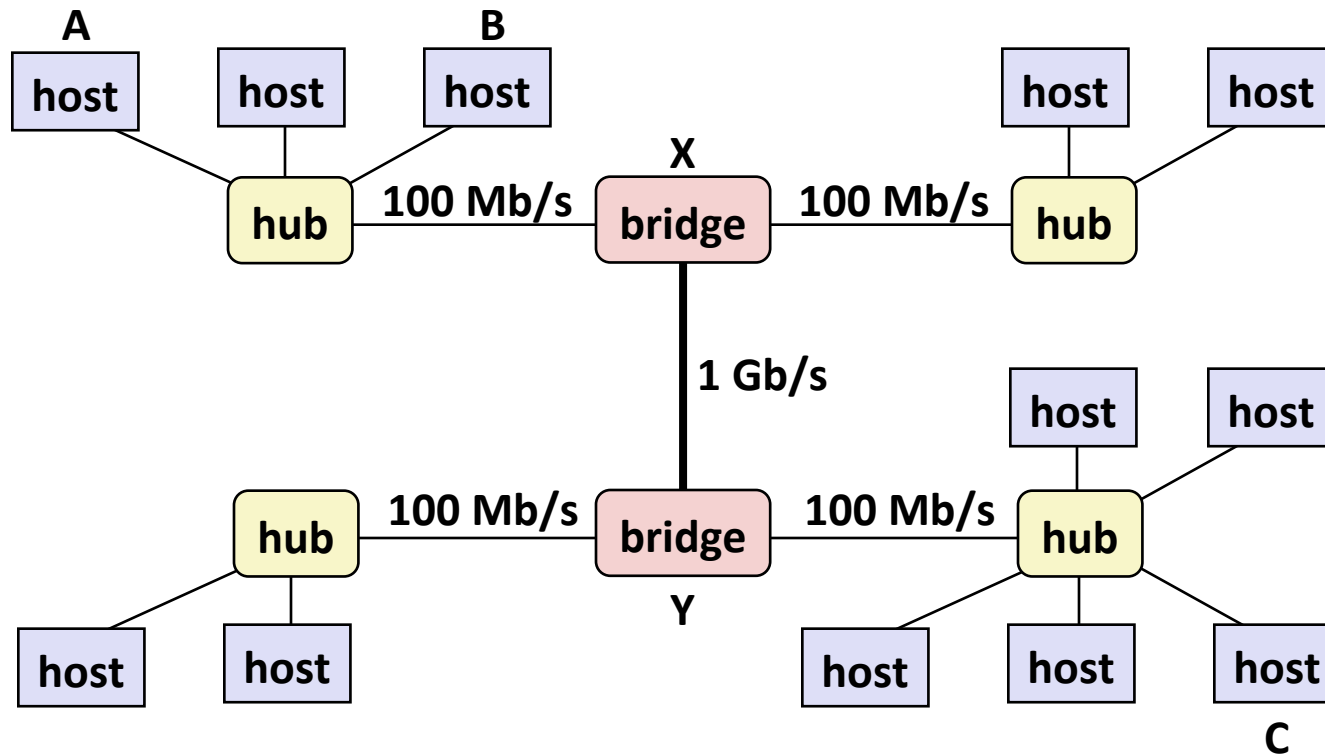


Lowest Level: Ethernet Segment



- Ethernet segment consists of a collection of *hosts* connected by wires (twisted pairs) to a *hub*
- Spans room or floor in a building
- Operation
 - Each Ethernet adapter has a unique 48-bit address (MAC address)
 - E.g., 00:16:ea:e3:54:e6
 - 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 on their way out. Bridges (switches, routers) became cheap enough to replace them (means no more broadcasting)

Next Level: Bridged Ethernet Segment



- Spans building or campus
- Bridges cleverly learn which hosts are reachable from which ports and then selectively copy frames from port to port

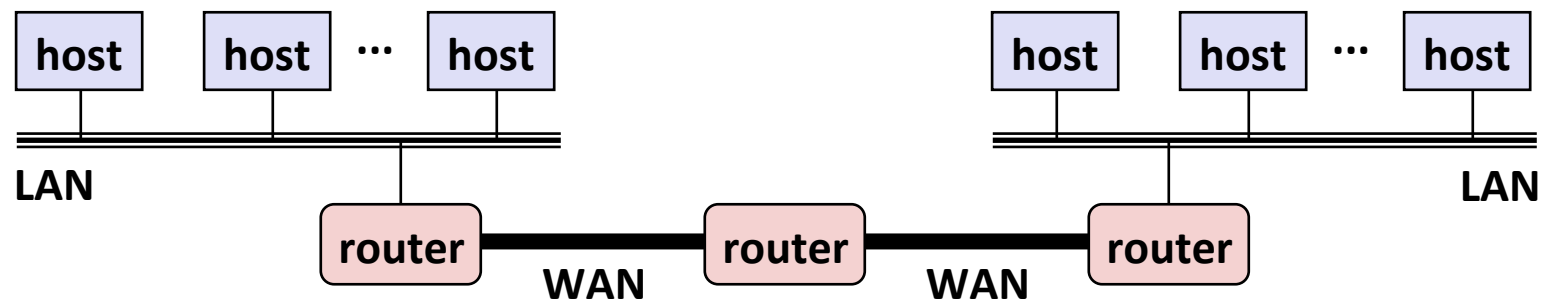
Conceptual View of LANs

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



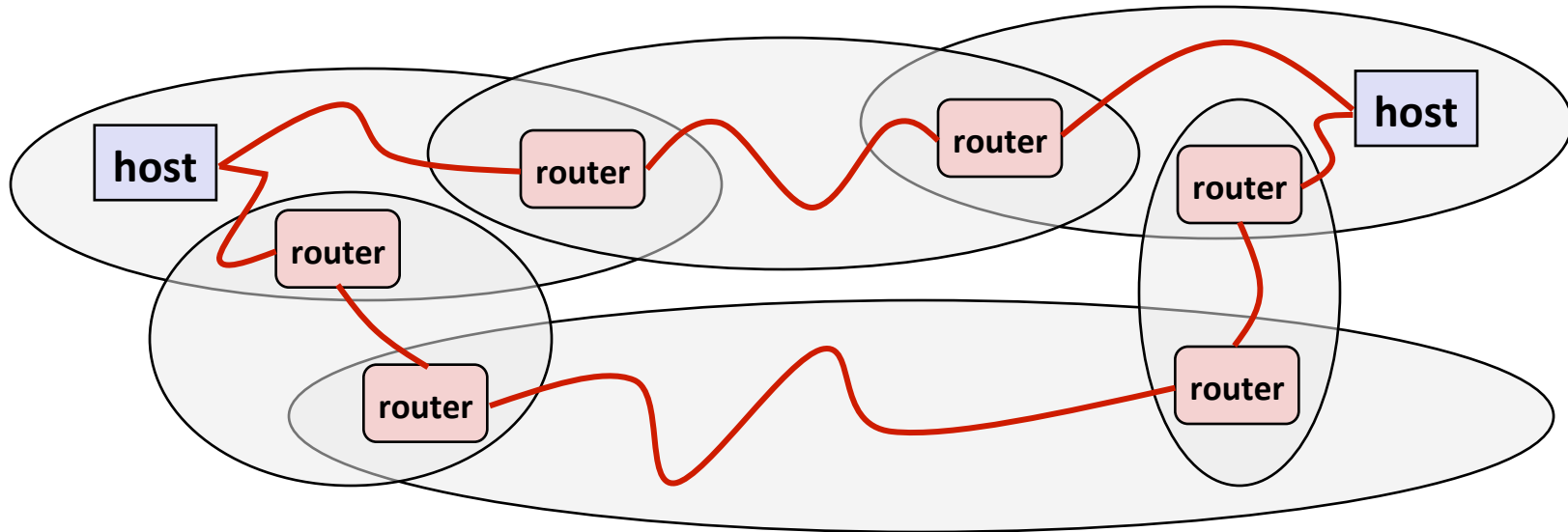
Next Level: internets

- 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 (e.g., Ethernet and Wifi, 802.11, T1-links, DSL, ...)*

Logical Structure of an internet



- **Ad hoc interconnection of networks**
 - No particular topology
 - Vastly different router & link capacities
- **Send packets from source to destination by hopping through networks**
 - Router forms bridge from one network to another
 - Different packets may take different routes

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
 - smooths out the differences between the different networks

- **Implements an internet protocol (i.e., set of rules)**
 - 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?

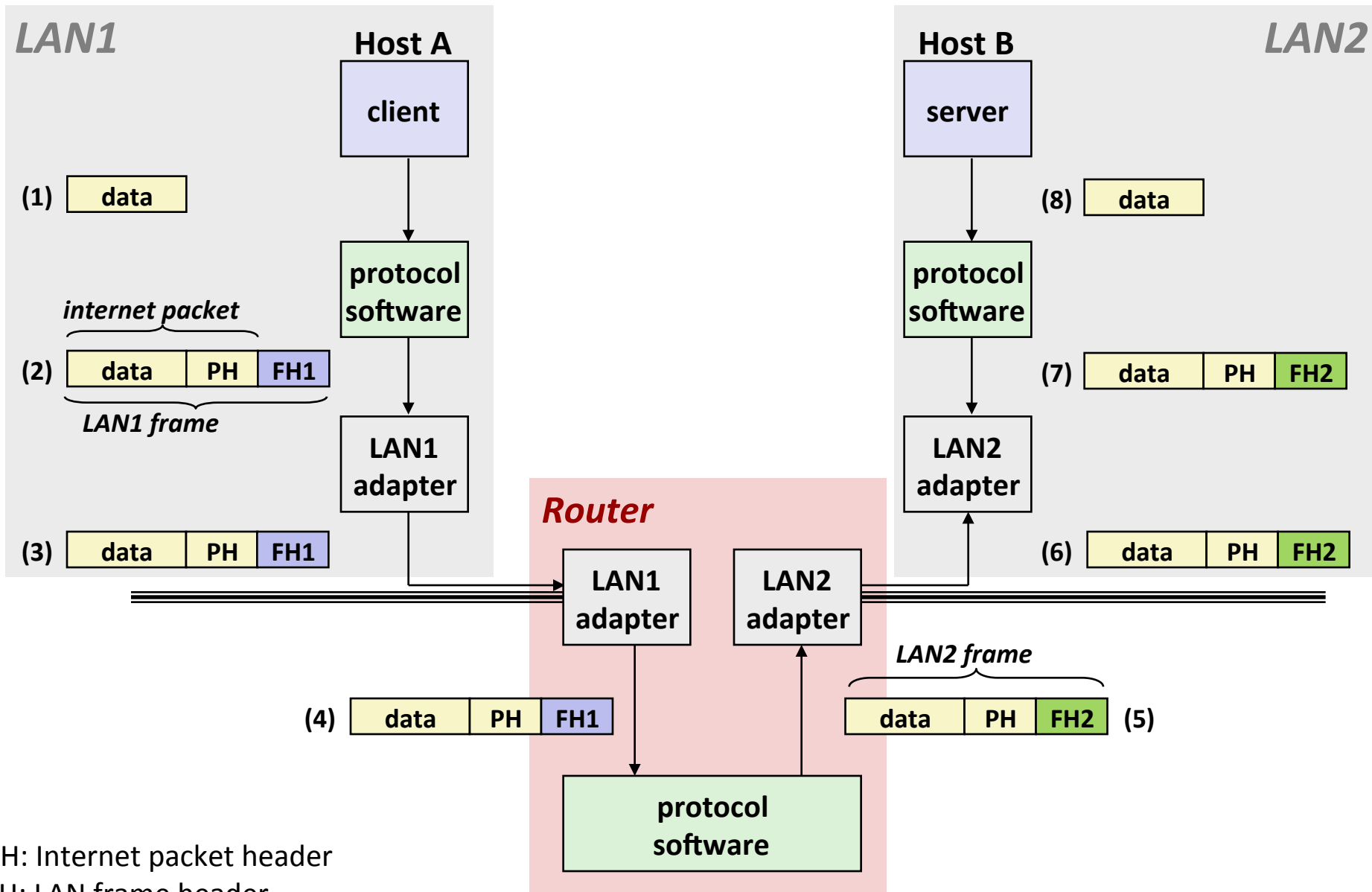
■ Provides a naming scheme

- 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

■ 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

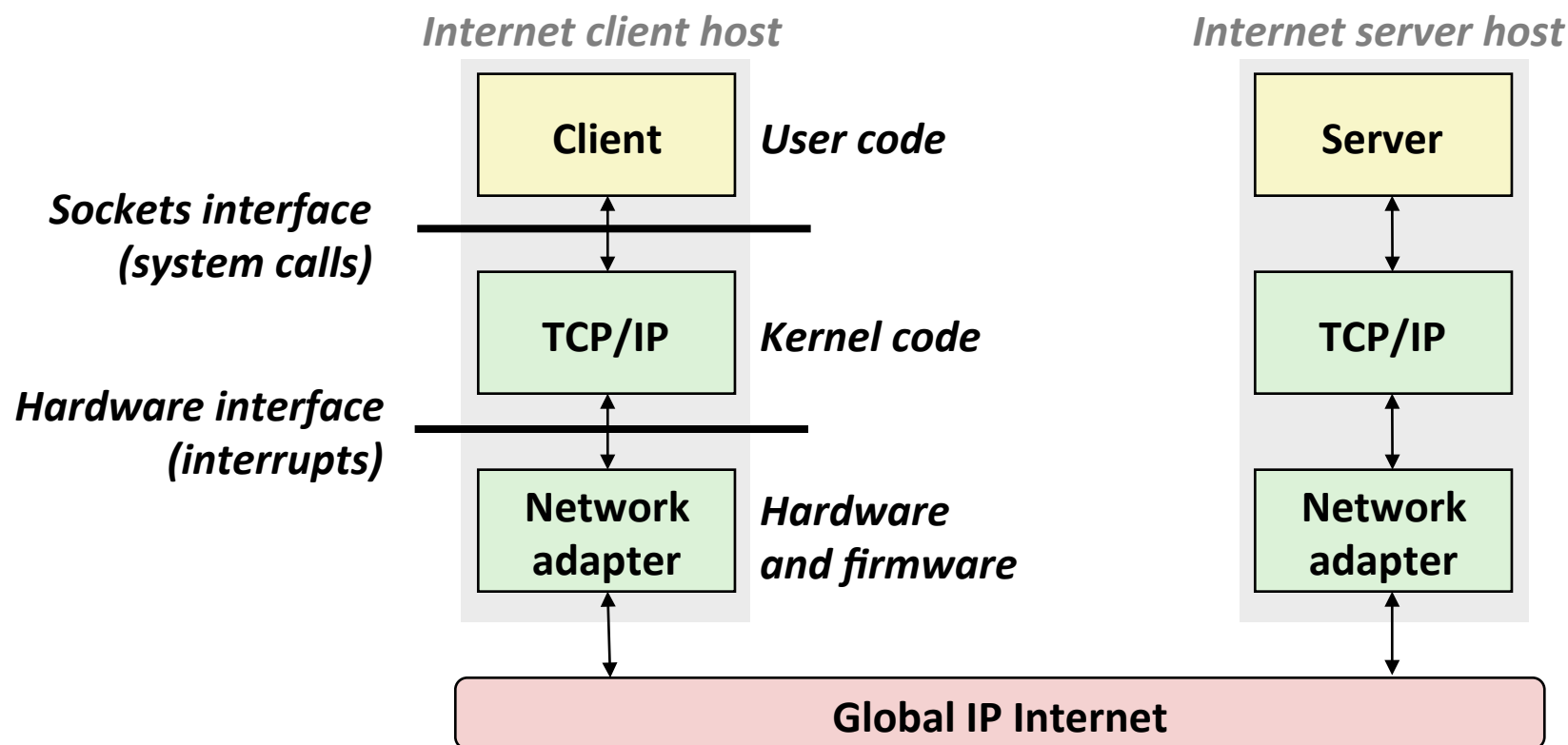
- **We are glossing over a number of important questions:**
 - What if different networks have different maximum frame sizes? (segmentation)
 - How do routers know where to forward frames?
 - How are routers informed when the network topology changes?
 - What if packets get lost?

- **These (and other) questions are addressed by the area of systems known as *computer networking***

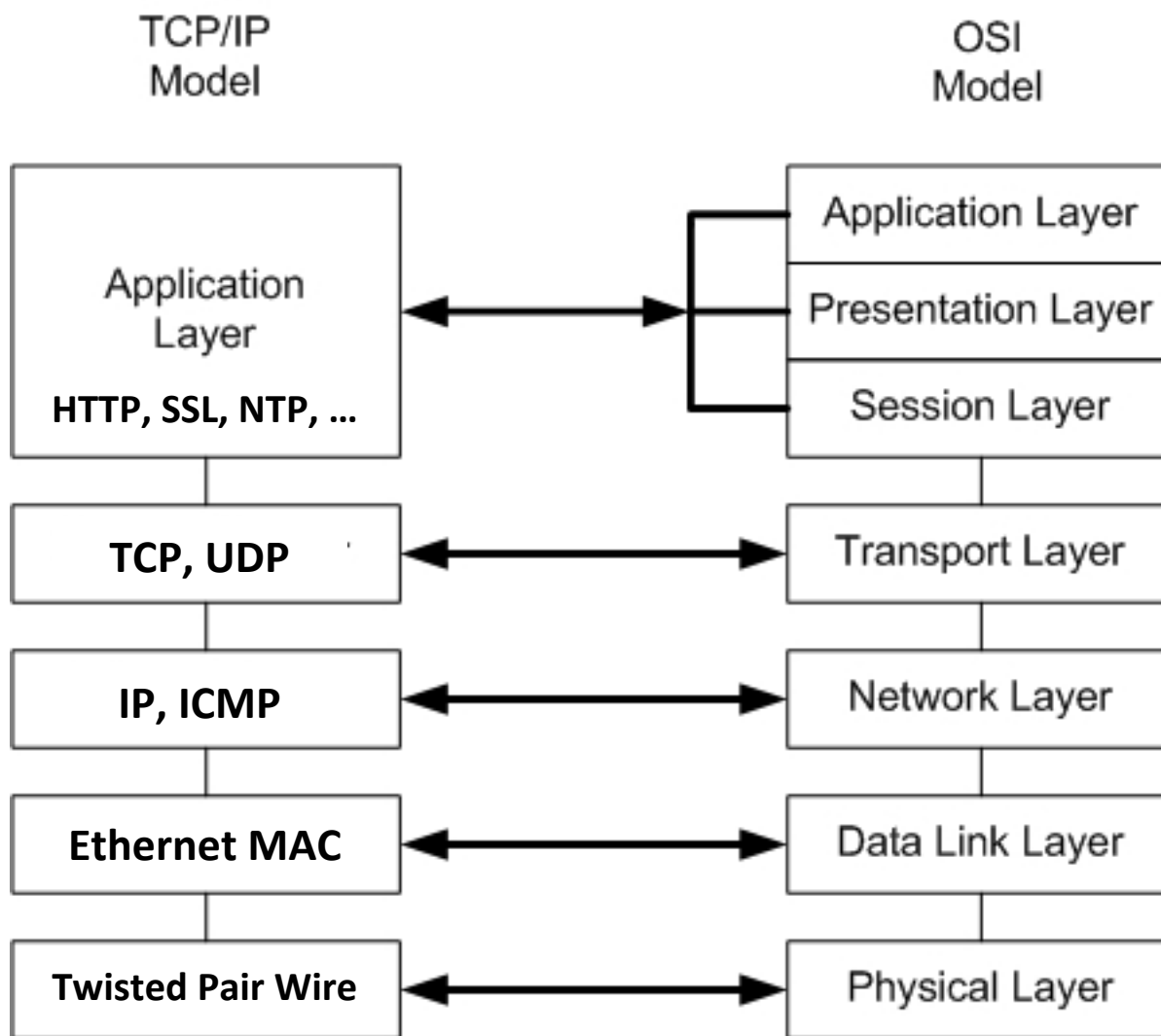
Global IP Internet

- Most famous example of an 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*

Hardware and Software Organization of an Internet Application



Protocol Layers



Basic Internet Components

■ Internet backbone:

- collection of routers (nationwide or worldwide) connected by high-speed point-to-point networks

■ Network Access Point (NAP):

- router that connects multiple backbones (often referred to as peers)

■ Regional networks:

- smaller backbones that cover smaller geographical areas (e.g., cities or states)

■ Point of presence (POP):

- machine that is connected to the Internet

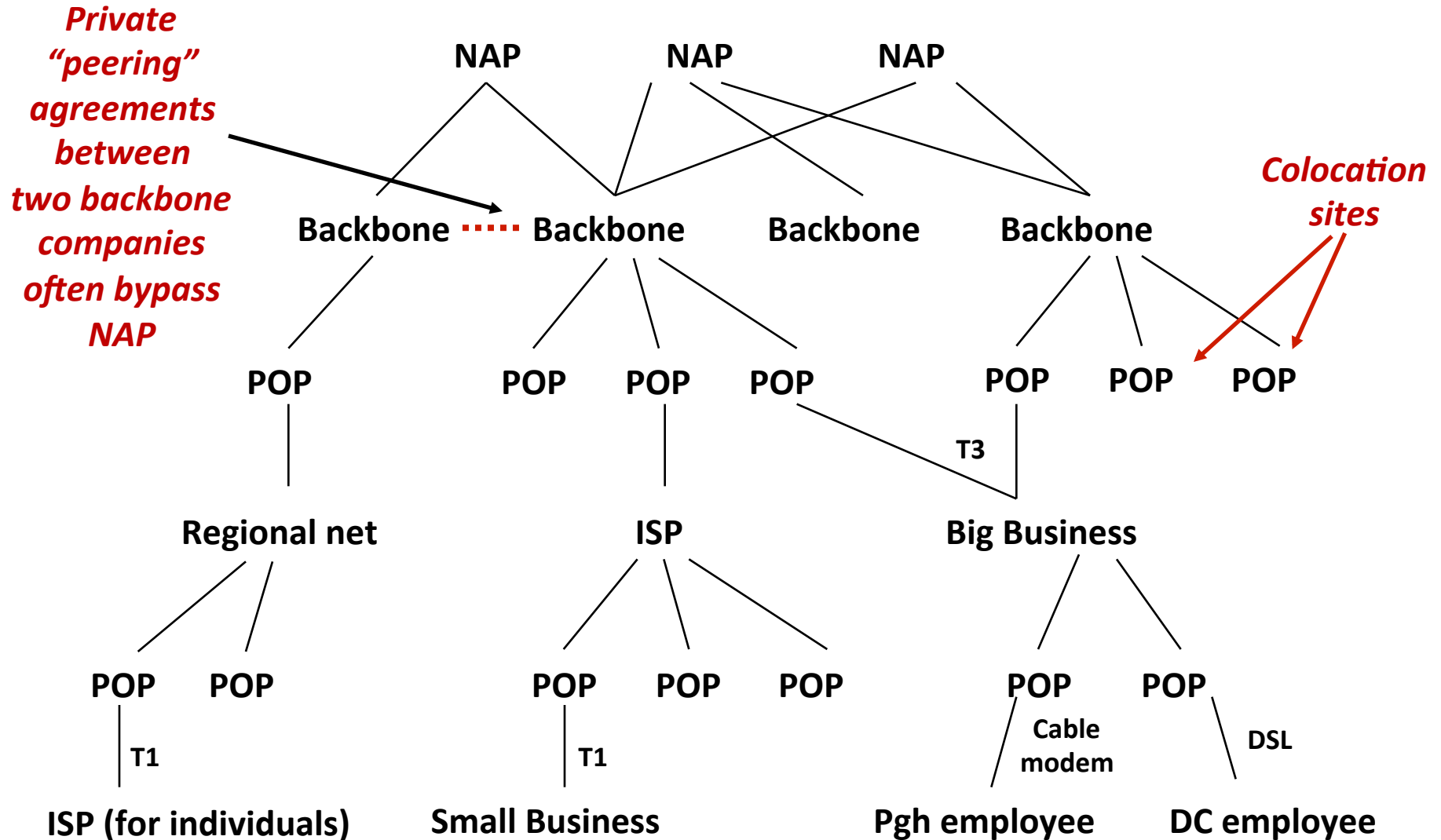
■ Internet Service Providers (ISPs):

- provide dial-up or direct access to POPs

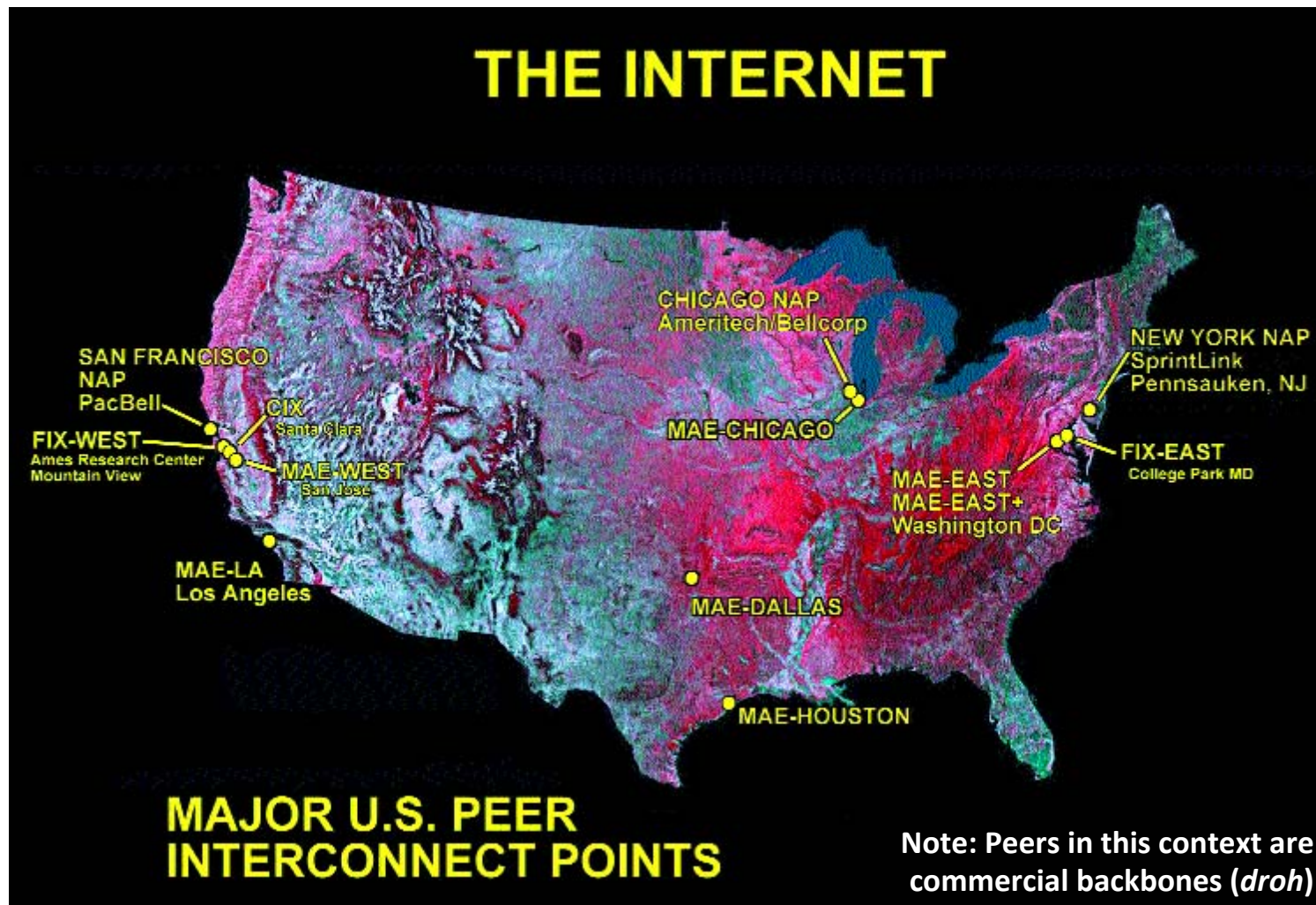
NAP-Based Internet Architecture

- **NAPs link together commercial backbones provided by companies such as AT&T and Worldcom**
- **Currently in the US there are about 50 commercial backbones connected by ~12 NAPs (peering points)**
- **Similar architecture worldwide connects national networks to the Internet**

Internet Connection Hierarchy

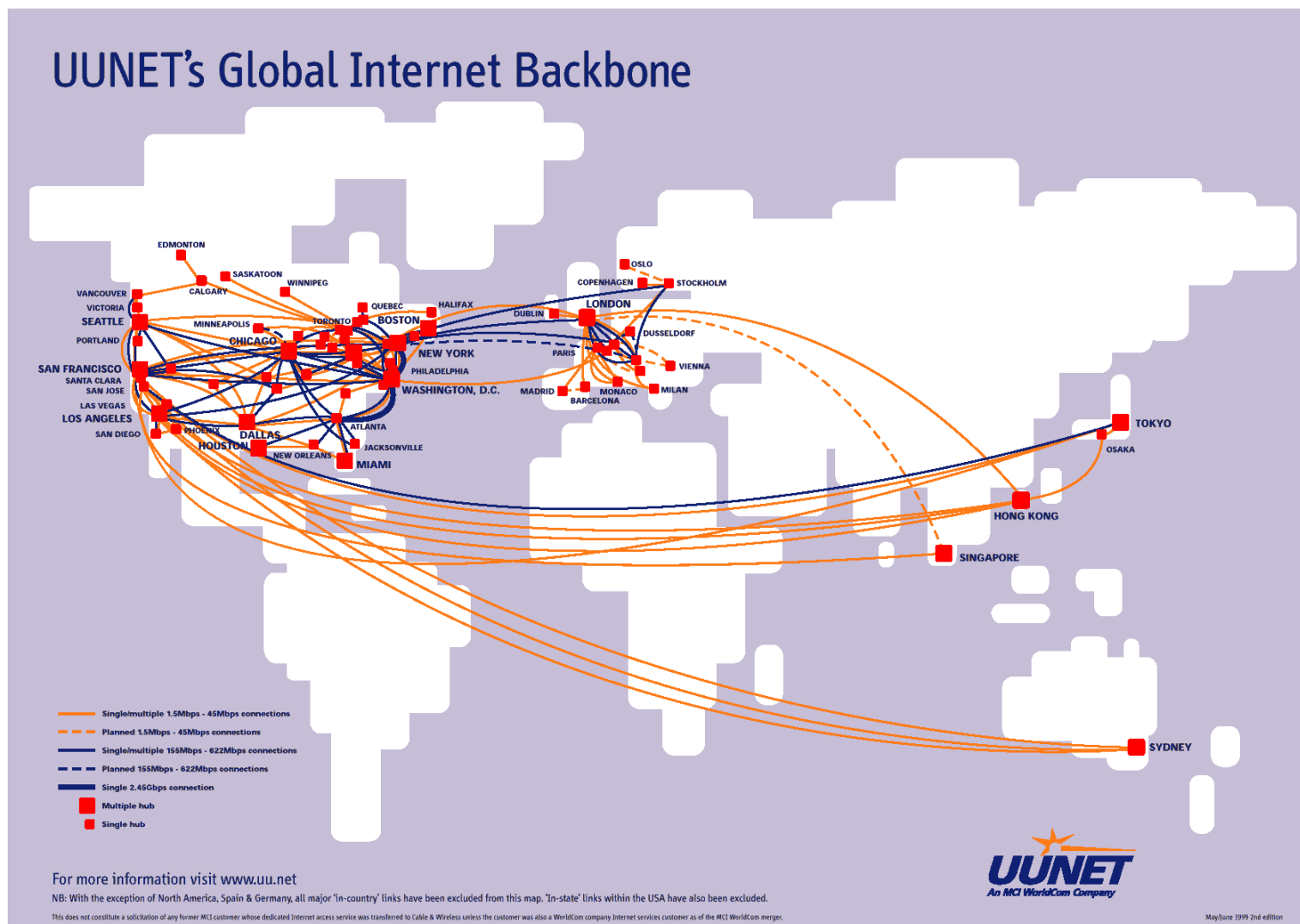


Network Access Points (NAPs)



Source: Boardwatch.com

MCI/WorldCom/UUNET Global Backbone



Source: <http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/>

Naming and Communicating on the Internet

■ Original Idea

- Every node on Internet would have unique IP address
 - Everyone would be able to talk directly to everyone
- No secrecy or authentication
 - Messages visible to routers and hosts on same LAN
 - Possible to forge source field in packet header

■ Shortcomings

- There aren't enough IP addresses available
- Don't want everyone to have access or knowledge of all other hosts
- Security issues mandate secrecy & authentication

Evolution of Internet: Naming

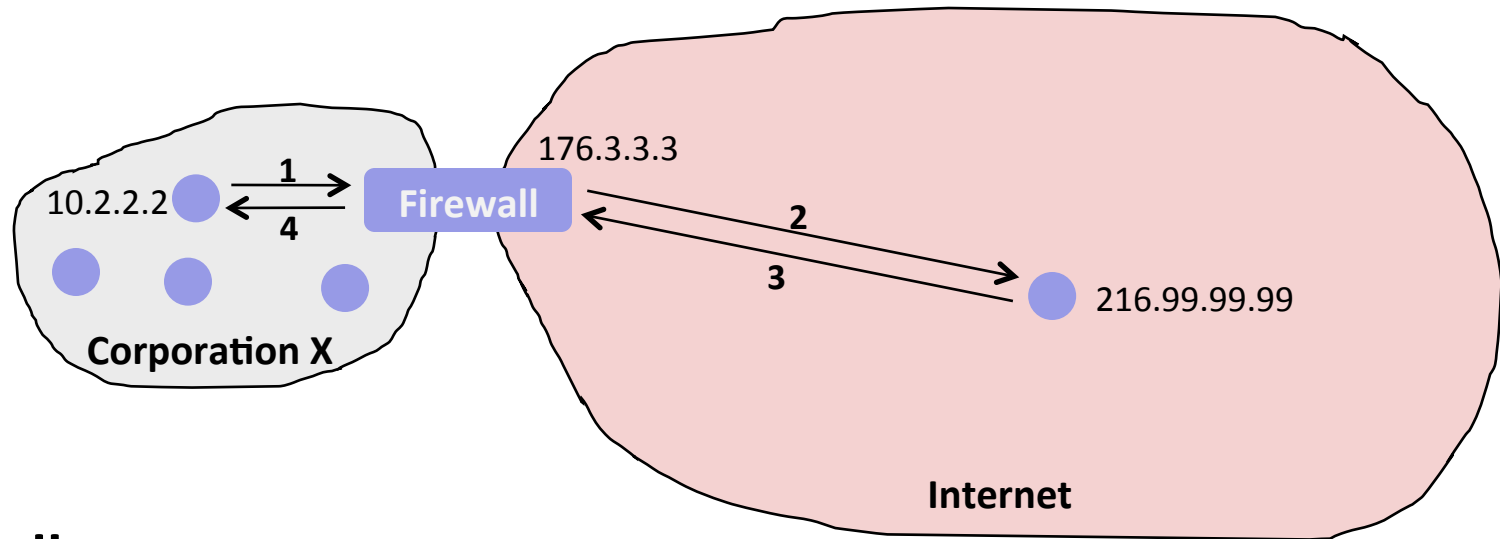
■ Dynamic address assignment

- Most hosts don't need to have known address
 - Only those functioning as servers
- DHCP (Dynamic Host Configuration Protocol)
 - Local ISP assigns address for temporary use

■ Example:

- My laptop at CMU (wired connection)
 - IP address 128.2.213.29 (**bryant-tp4.cs.cmu.edu**)
 - Assigned statically
- My laptop at home
 - IP address 192.168.1.5
 - Only valid within home network

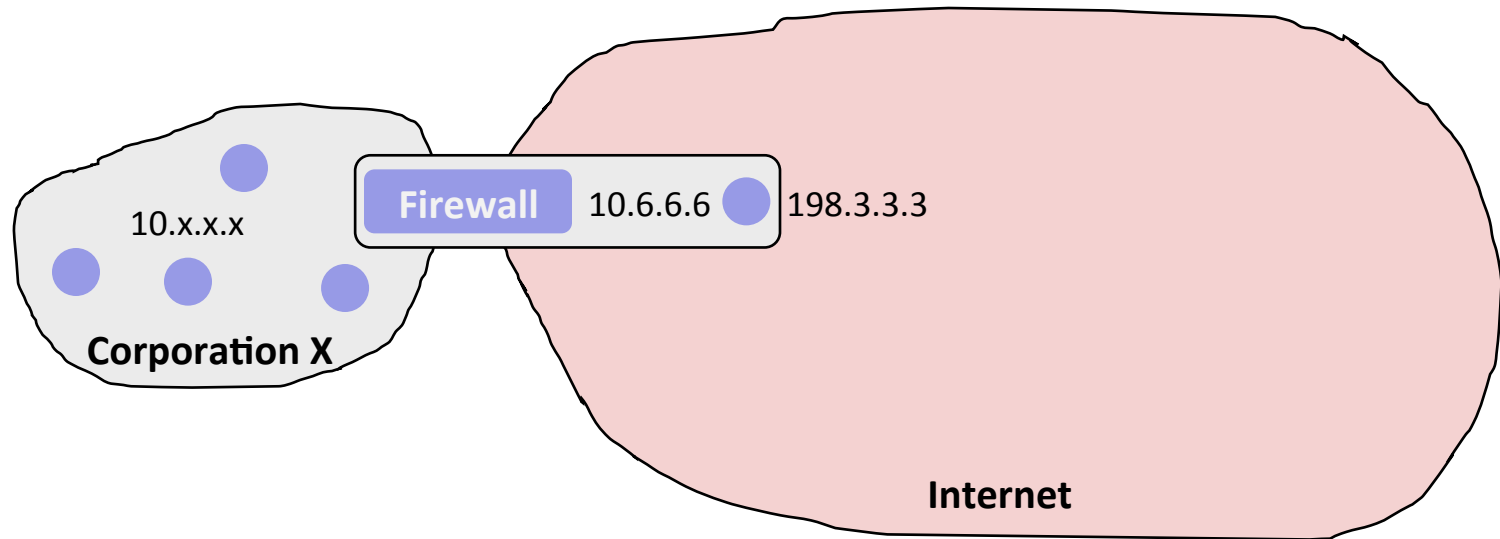
Evolution of Internet: Firewalls



■ Firewalls

- Hides organizations nodes from rest of Internet
- Use local IP addresses within organization
- For external service, provides proxy service
 1. Client request: src=10.2.2.2, dest=216.99.99.99
 2. Firewall forwards: src=176.3.3.3, dest=216.99.99.99
 3. Server responds: src=216.99.99.99, dest=176.3.3.3
 4. Firewall forwards response: src=216.99.99.99, dest=10.2.2.2

Virtual Private Networks



■ Supporting road warrior

- Employee working remotely with assigned IP address 198.3.3.3
- Wants to appear to rest of corporation as if working internally
 - From address 10.6.6.6
 - Gives access to internal services (e.g., ability to send mail)

■ Virtual Private Network (VPN)

- Overlays private network on top of regular Internet

A Programmer's View of the Internet

- Hosts are mapped to a set of 32-bit *IP addresses*
 - 128.2.203.179
- The set of IP addresses is mapped to a set of identifiers called Internet *domain names*
 - 128.2.203.179 is mapped to `www.cs.cmu.edu`
- A process on one Internet host can communicate with a process on another Internet host over a *connection*

IP Addresses

- **32-bit IP addresses are stored in an *IP address struct***
 - IP addresses are always stored in memory in network byte order (big-endian byte order)
 - True in general for any integer transferred in a packet header from one machine to another.
 - E.g., the port number used to identify an Internet connection.

```
/* Internet address structure */  
struct in_addr {  
    unsigned int s_addr; /* network byte order (big-endian) */  
};
```

Useful network byte-order conversion functions (“l” = 32 bits, “s” = 16 bits)

htonl: convert `uint32_t` from host to network byte order

htons: convert `uint16_t` from host to network byte order

ntohl: convert `uint32_t` from network to host byte order

ntohs: convert `uint16_t` from network to host byte order

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_aton`: dotted decimal string → IP address in network byte order
 - `inet_ntoa`: IP address in network byte order → dotted decimal string
 - “n” denotes network representation
 - “a” denotes application representation

IP Address Structure

■ IP (V4) Address space divided into classes:

	0	1	2	3	8	16	24	31	
Class A	0	Net ID			Host ID				
Class B	1	0	Net ID			Host ID			
Class C	1	1	0	Net ID			Host ID		
Class D	1	1	1	0	Multicast address				
Class E	1	1	1	1	Reserved for experiments				

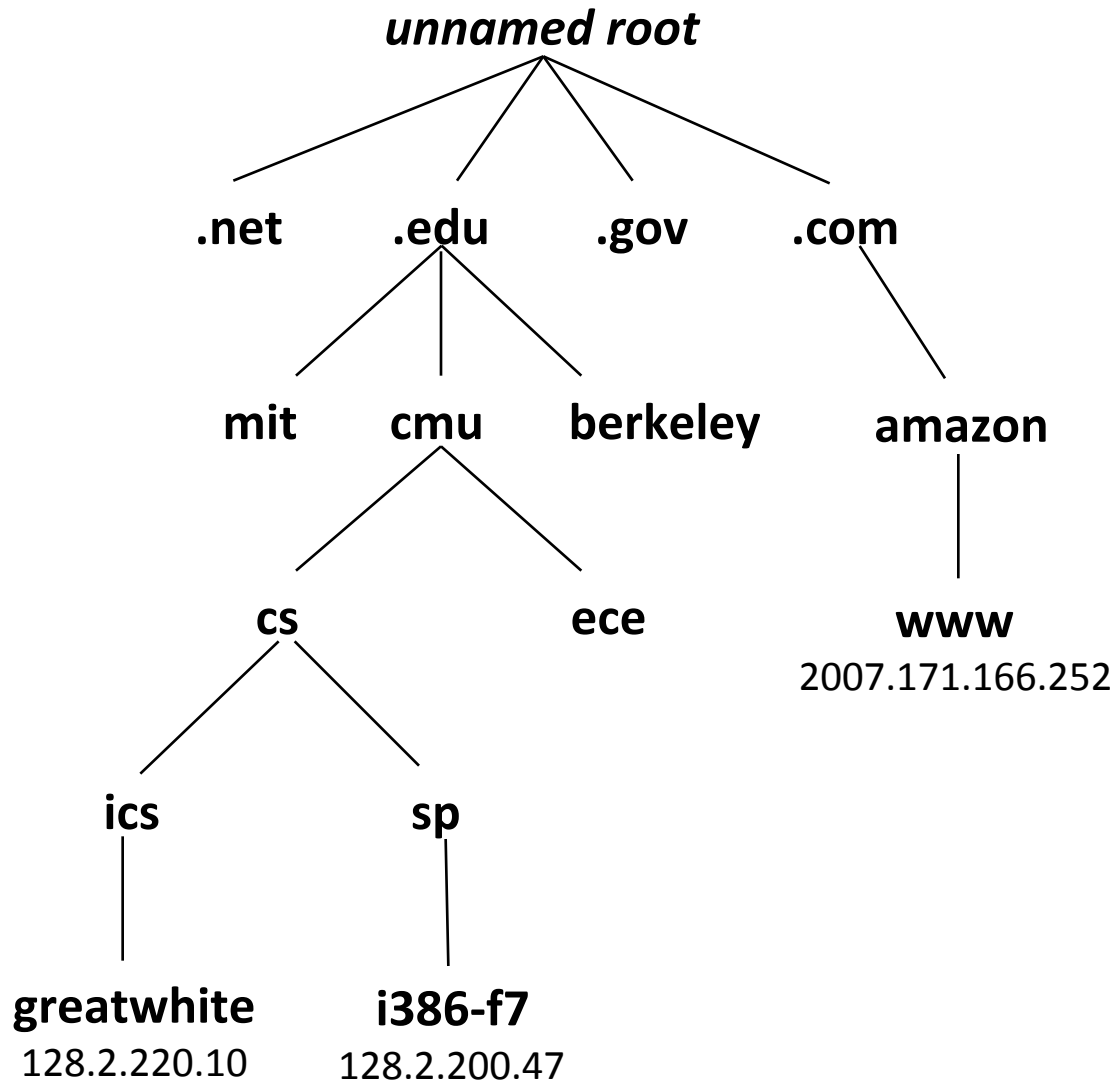
■ Network ID Written in form w.x.y.z/n

- n = number of bits in host address
- E.g., CMU written as 128.2.0.0/16
 - Class B address

■ Unrouted (private) IP addresses:

10.0.0.0/8 172.16.0.0/12 192.168.0.0/16

Internet Domain Names



First-level domain names

Second-level domain names

Third-level domain names

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 *host entry structures*:

```
/* DNS host entry structure */
struct hostent {
    char    *h_name;          /* official domain name of host */
    char    **h_aliases;     /* null-terminated array of domain names */
    int     h_addrtype;      /* host address type (AF_INET) */
    int     h_length;        /* length of an address, in bytes */
    char    **h_addr_list;   /* null-terminated array of in_addr structs
*/
};
```

- **Functions for retrieving host entries from DNS:**
 - **gethostbyname**: query key is a DNS domain name.
 - **gethostbyaddr**: query key is an IP address.

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: one-to-one mapping between domain name and IP address:
 - `greatwhile.ics.cs.cmu.edu` maps to `128.2.220.10`
 - Multiple domain names mapped to the same IP address:
 - `eecs.mit.edu` and `cs.mit.edu` both map to `18.62.1.6`
 - Multiple domain names mapped to multiple IP addresses:
 - `google.com` maps to multiple IP addresses
 - Some valid domain names don't map to any IP address:
 - for example: `ics.cs.cmu.edu`

A Program That Queries DNS

```
int main(int argc, char **argv) { /* argv[1] is a domain name */
    char **pp;                    /* or dotted decimal IP addr */
    struct in_addr addr;
    struct hostent *hostp;

    if (inet_aton(argv[1], &addr) != 0)
        hostp = Gethostbyaddr((const char *)&addr, sizeof(addr),
                               AF_INET);
    else
        hostp = Gethostbyname(argv[1]);
    printf("official hostname: %s\n", hostp->h_name);

    for (pp = hostp->h_aliases; *pp != NULL; pp++)
        printf("alias: %s\n", *pp);

    for (pp = hostp->h_addr_list; *pp != NULL; pp++) {
        addr.s_addr = ((struct in_addr *)*pp)->s_addr;
        printf("address: %s\n", inet_ntoa(addr));
    }
}
```

Using DNS Program

```
linux> ./dns greatwhite.ics.cs.cmu.edu
official hostname: greatwhite.ics.cs.cmu.edu
address 128.2.220.10
```

```
linux> ./dns 128.2.220.11
official hostname: ANGELSHARK.ICS.CS.CMU.EDU
address: 128.2.220.11
```

```
linux> ./dns www.google.com
official hostname: www.l.google.com
alias: www.google.com
address: 72.14.204.99
address: 72.14.204.103
address: 72.14.204.104
address: 72.14.204.147
linux> dig +short -x 72.14.204.103
iad04s01-in-f103.1e100.net.
```

Querying DIG

- Domain Information Groper (`dig`) provides a scriptable command line interface to DNS

```
linux> dig +short greatwhite.ics.cs.cmu.edu
128.2.220.10
linux> dig +short -x 128.2.220.11
ANGELSHARK.ICS.CS.CMU.EDU.
linux> dig +short google.com
72.14.204.104
72.14.204.147
72.14.204.99
72.14.204.103
linux> dig +short -x 72.14.204.103
iad04s01-in-f103.1e100.net.
```

More Exotic Features of DIG

- Provides more information than you would ever want about DNS

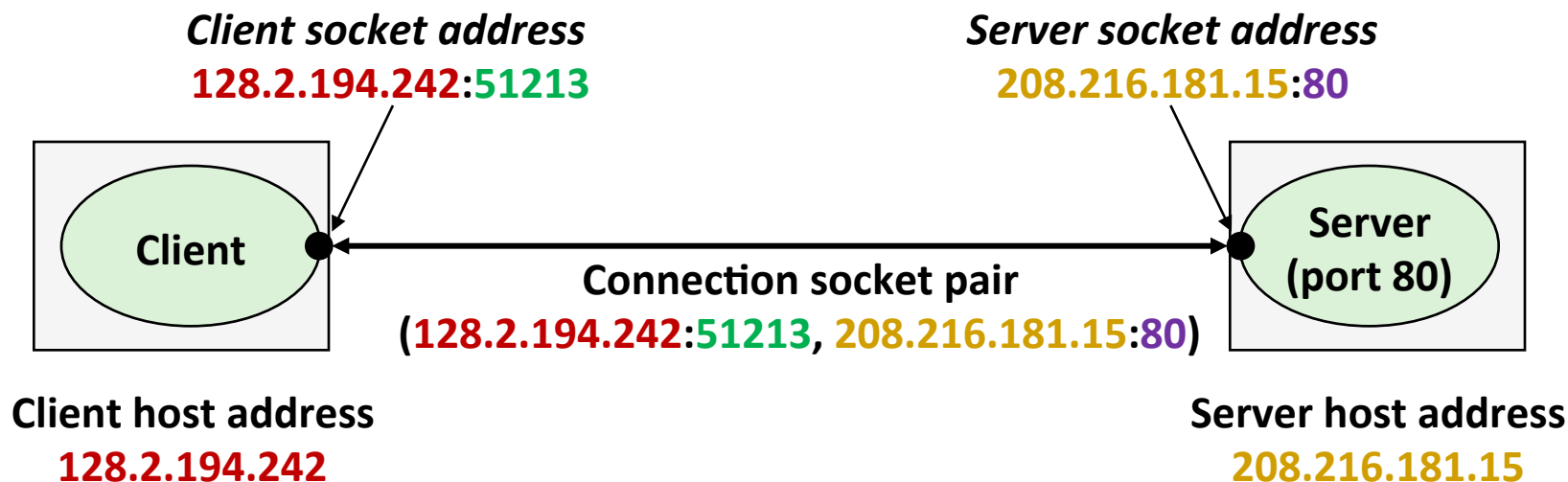
```
linux> dig www.phys.msu.ru a +trace  
128.2.220.10
```

```
linux> dig www.google.com a +trace
```

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 `IPAddress: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



Next Time

- **How to use the sockets interface to establish Internet connections between clients and servers**
- **How to use Unix I/O to copy data from one host to another over an Internet connection**