Internetworking

Introduction to Computer Systems
22th Lecture, Dec. 4, 2019

Instructors:

Class 1: Chen Xiangqun, Sun Guangyu

Class 2: Guan Xuetao

Class 3: Lu Junlin

Outline

- **Computer networking**

- Network protocols

- Global IP Internet

- Programmer's view of Internet

- Evolution of Internet

Know how, and know why

- **Using Internet?**
  - Web surfing
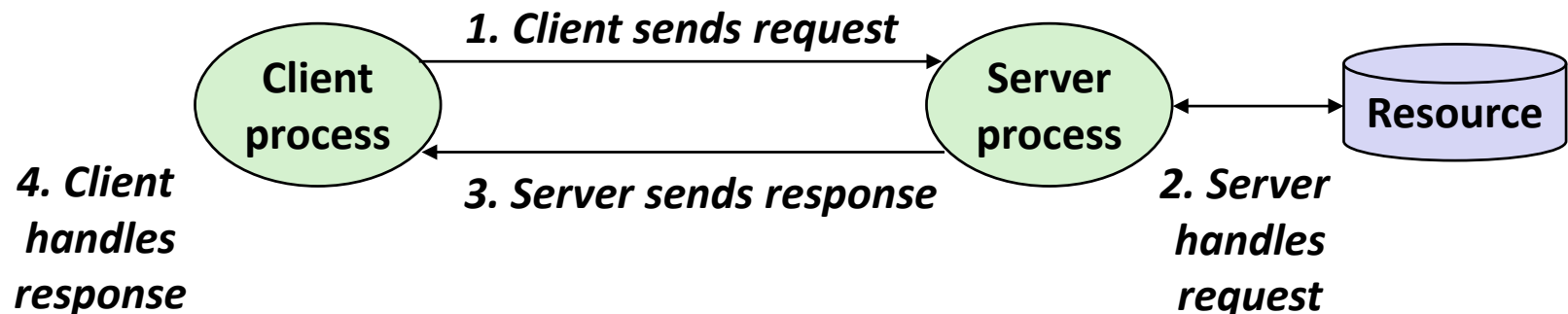 - IM (Instant Message)
  - Online Games
- **Troubleshooting and Network programming?**
  - What is computer network?
  - Socket interface
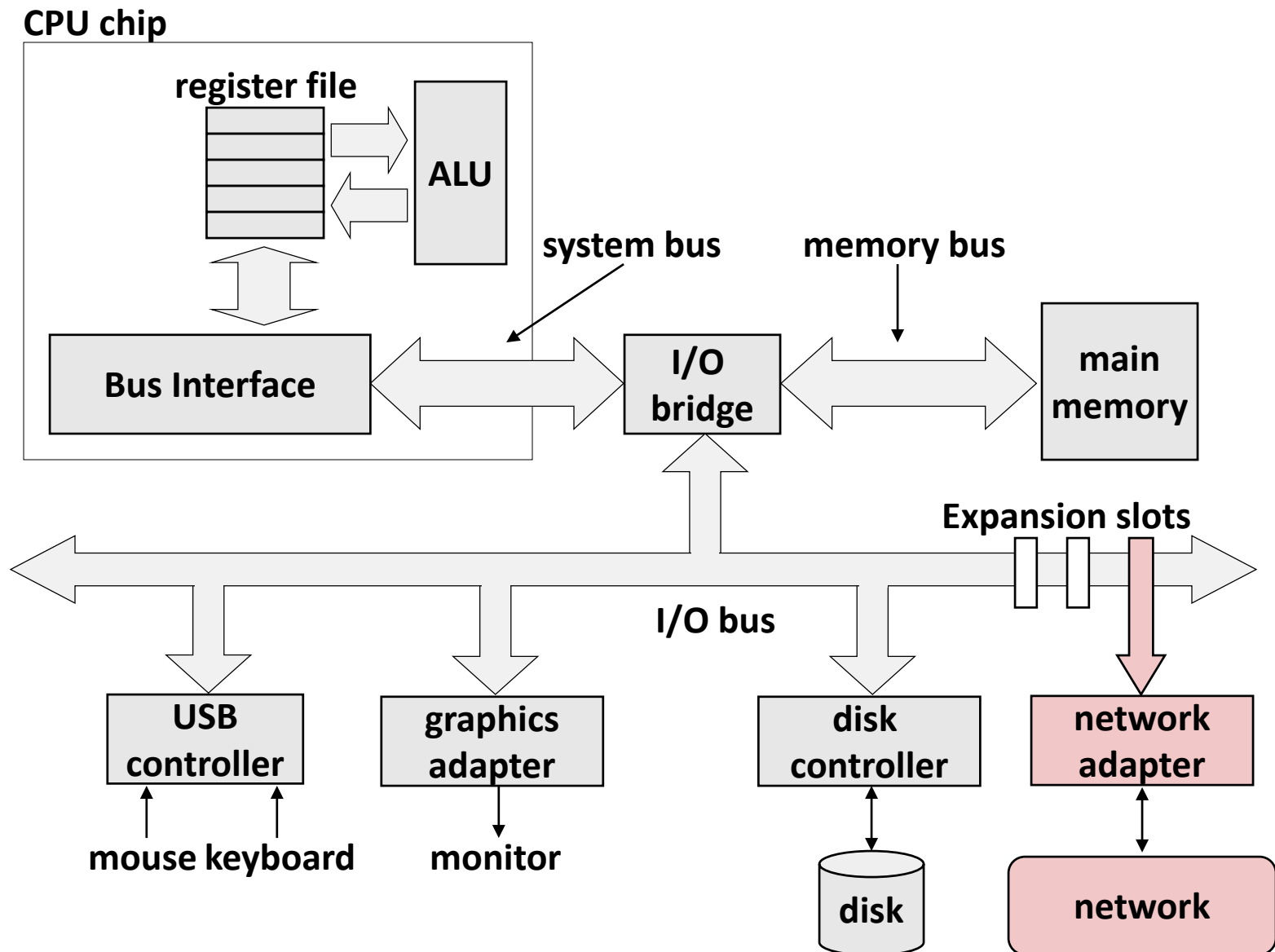 - Web server

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

1. Client sends request

Client process → **Server process** ↔ **Resource**

3. Server sends response

4. Client handles response

2. Server handles request

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

Hardware Organization of a Network Host

CPU chip

register file

ALU

system bus

memory bus

Bus Interface

I/O bridge

main memory

Expansion slots

I/O bus

USB controller

graphics adapter

disk controller

network adapter

mouse keyboard

monitor

**disk**

network

5

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**

* Not to be confused with a Storage Area Network

Lowest Level: Ethernet Segment
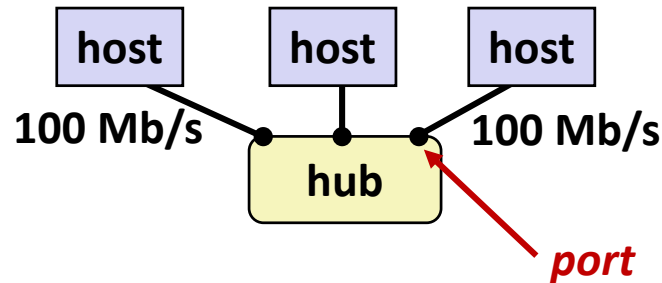
- **Ethernet segment consists of a collection of *hosts* connected by wires (twisted pairs) to a *hub***

- **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 obsolete. Bridges (switches, routers) became cheap enough to replace them)

Next Level: Bridged Ethernet Segment
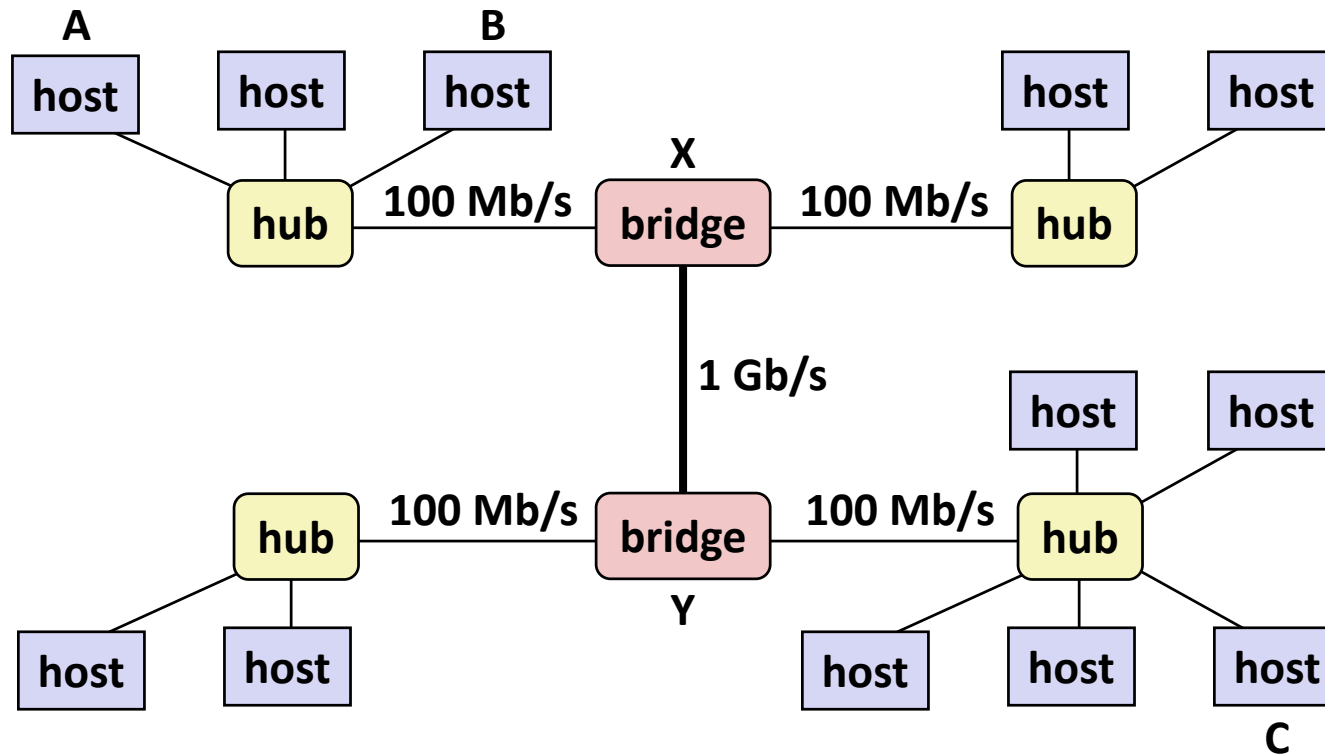
- **Spans building or campus**

- **Bridges cleverly learn which hosts are reachable from which ports and then selectively copy frames from port to port**
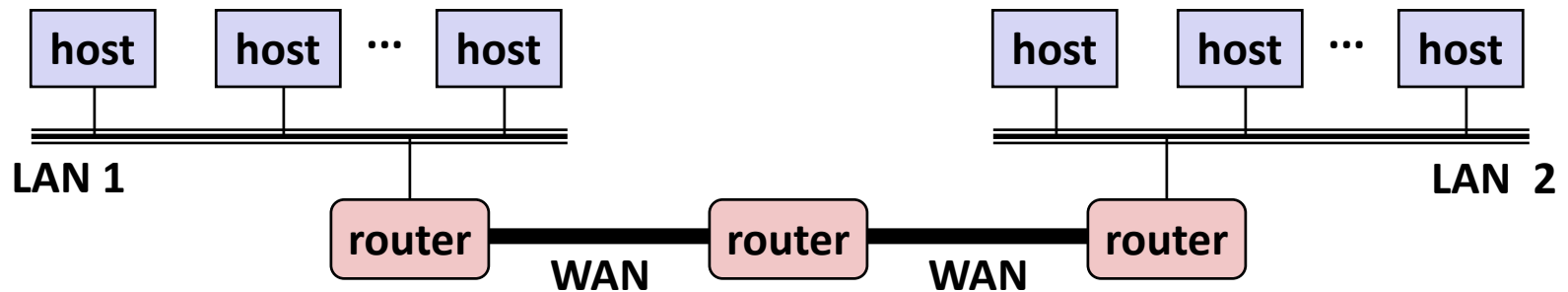
Conceptual View of LANs

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

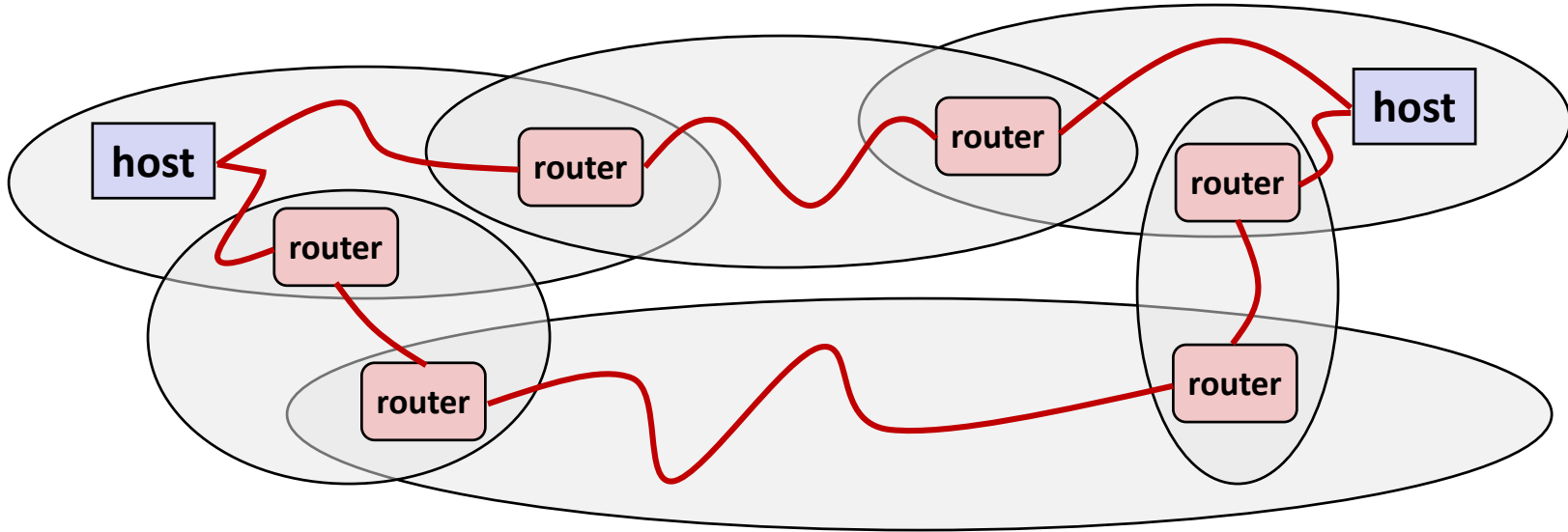| host	host	⋯	host
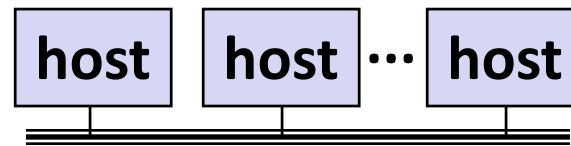
Next Level: internets

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

LAN 1 and LAN 2 might be completely different, totally incompatible

(e.g., Ethernet, Fibre Channel, 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

Outline

- **Computer networking**

- **Network protocols**

- **Global IP Internet**

- **Programmer's view of Internet**

- **Evolution of Internet**

What is a protocol?

- **Protocol = Pre-agreed rules**

  - Smile = Happiness

  - Cry = Sadness

  - Nod one's head = YES

  - Shake one's head = NO

- **Human protocols:**

  - What's the time?

  - Specific msgs sent

  - Specific actions taken when msgs received, or other events

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**

- Protocol is a set of rules that governs how hosts and routers should cooperate when they transfer data from network to network.

- Smooths out the differences between the different networks

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

15

Transferring internet Data Via Encapsulation

LAN1

Host A

client

(1) | data |

internet packet

(2) | data | PH | FH1 |

LAN1 frame

protocol software

LAN1 adapter

(3) | data | PH | FH1 |

Router

LAN1 adapter **LAN2 adapter**

(4) | data | PH | FH1 |

LAN2 frame

| data | PH | FH2 | (5)

protocol software

Host B

server

LAN2

(8) | data |

protocol software

(7) | data | PH | FH2 |

LAN2 adapter

(6) | data | PH | FH2 |

PH: Internet packet header
FH: LAN frame header

16
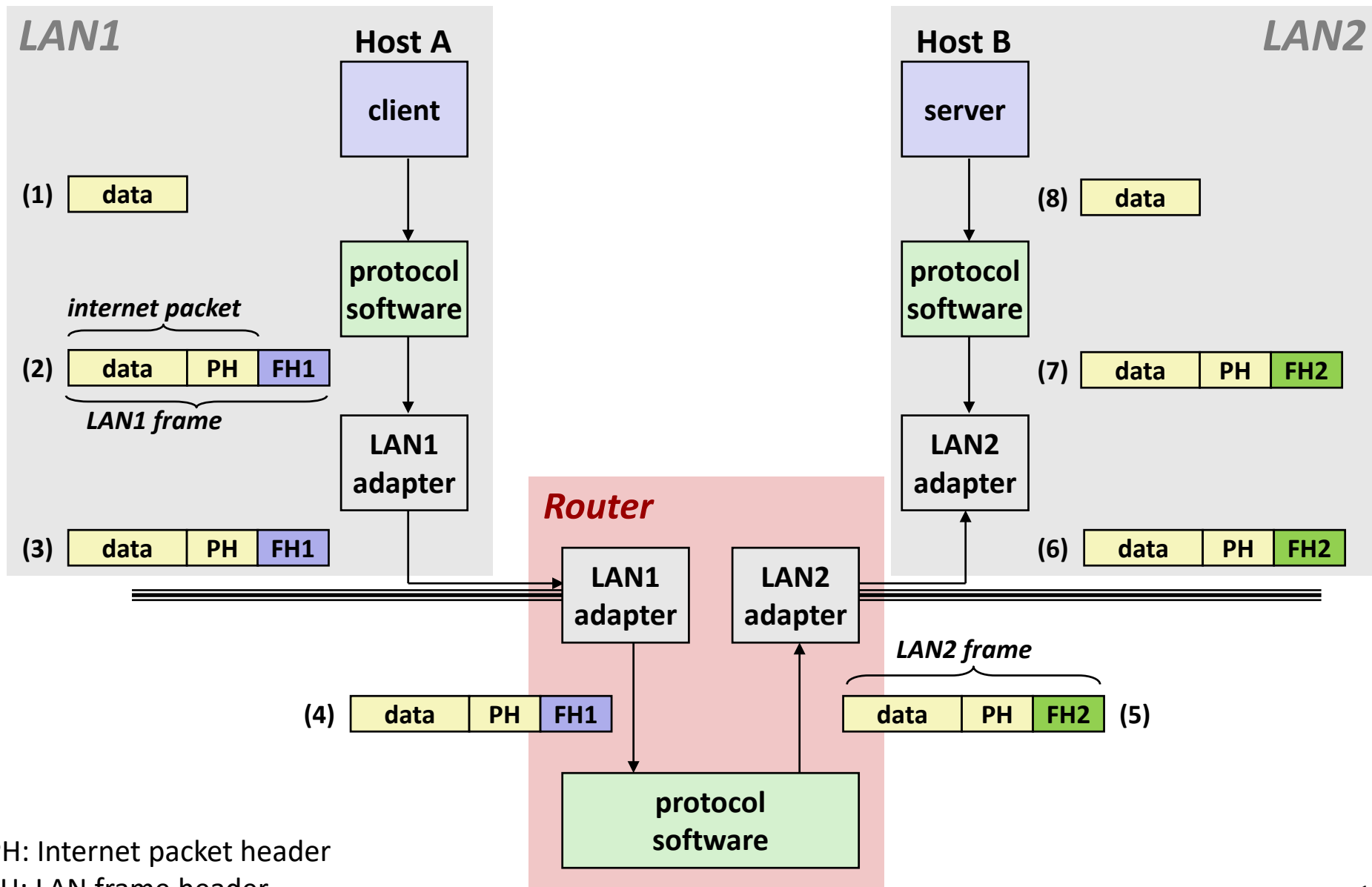
Other Issues

- **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***
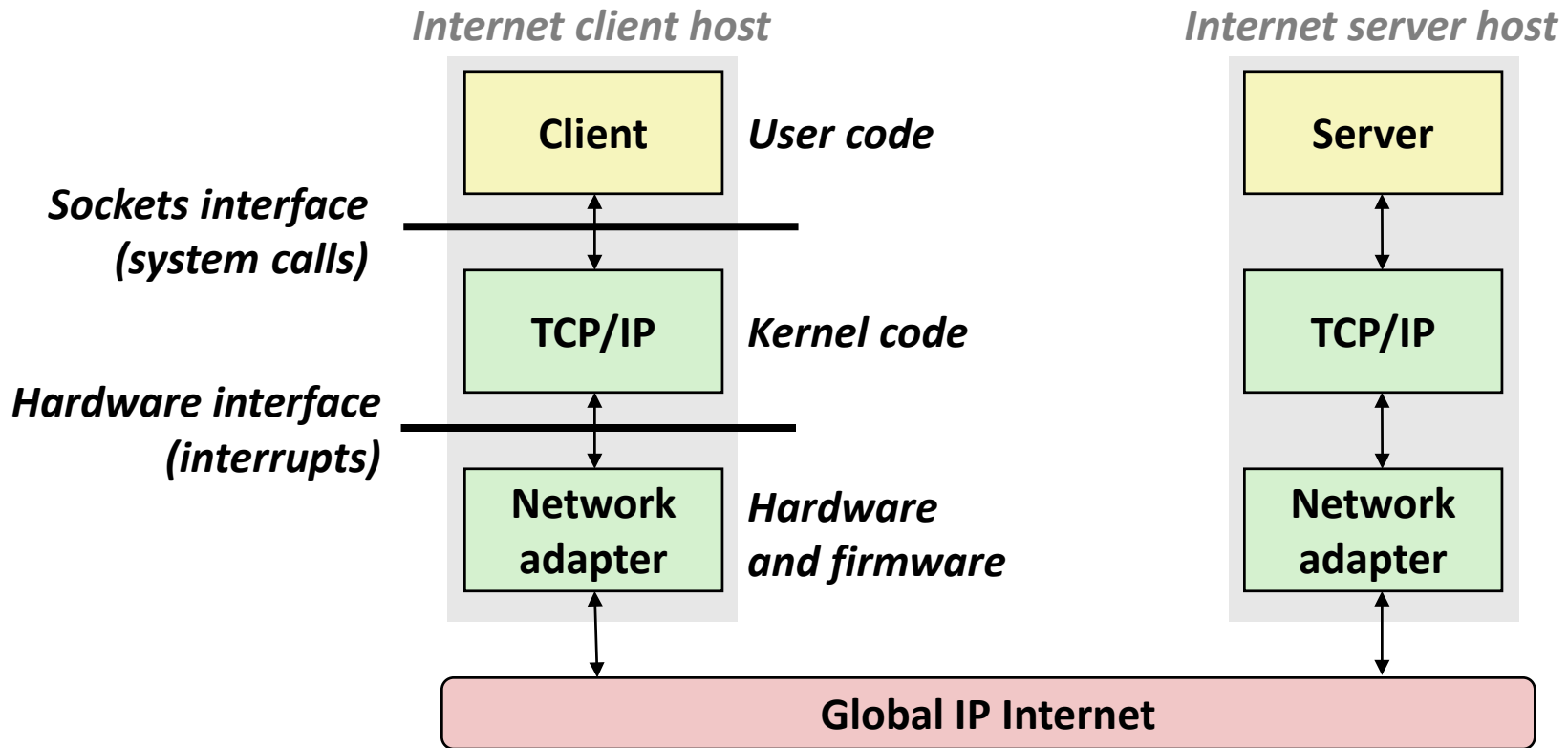
Outline

- **Computer networking**

- **Network protocols**

- **Global IP Internet**

- **Programmer's view of Internet**

- **Evolution of Internet**

Global IP Internet (upper case)

- **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

Internet client host

Internet server host

Client	User code

Sockets interface (system calls)

TCP/IP	Kernel code

Hardware interface (interrupts)

Network adapter	Hardware and firmware

Server

TCP/IP

Network adapter

Global IP Internet

Basic Internet Components

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

- **Internet Exchange Points (IXP):**
  - router that connects multiple backbones (often referred to as peers)
  - Also called Network Access Points (NAP)

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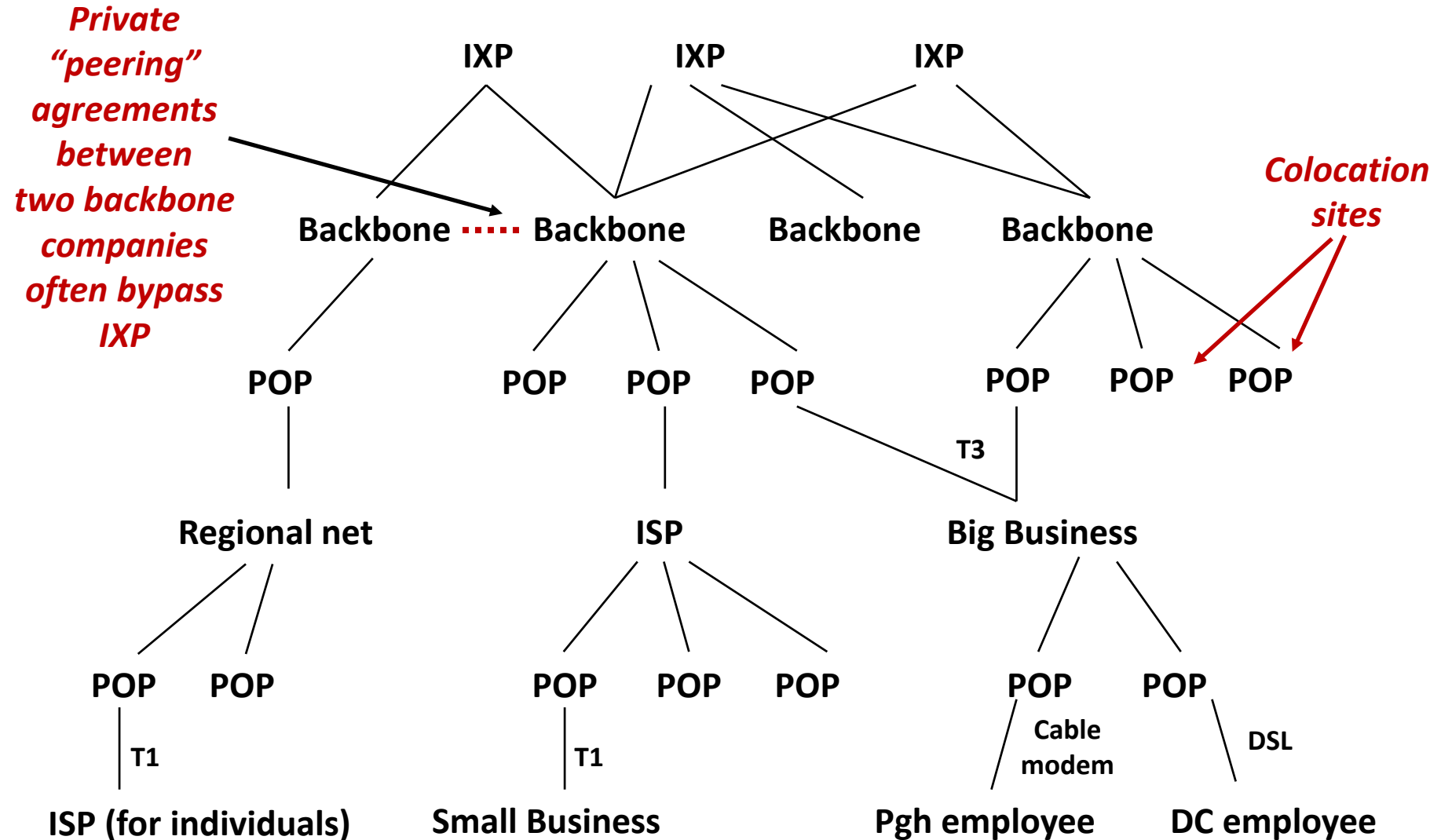 - smaller backbones that cover smaller geographical areas (e.g., cities or states)

- **Point of presence (POP):**
  - machine that is connected to the Internet

- **Internet Service Providers (ISPs):**
  - provide dial-up or direct access to POPs

Internet Connection Hierarchy

Private "peering" agreements between two backbone companies often bypass IXP

Colocation sites

IXP   IXP   IXP

Backbone ···· Backbone   Backbone   Backbone

POP   POP   POP   POP   POP   POP   POP

T3

Regional net   ISP   Big Business

POP   POP   POP   POP   POP   POP   POP

T1   T1   Cable modem   DSL

ISP (for individuals)   **Small Business**   **Pgh employee**   **DC employee**

Outline

- **Computer networking**

- **Network protocols**

- **Global IP Internet**

- **Programmer's view of Internet**

- **Evolution of Internet**

A Programmer's View of the Internet

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

- 128.2.203.179
- 127.0.0.1 (always *localhost*)

2. The set of IP addresses is mapped to a set of identifiers called Internet *domain names*

- 128.2.217.3 is mapped to www.cs.cmu.edu

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

Aside: IPv4 and IPv6

- **The original Internet Protocol, with its 32-bit addresses, is known as *Internet Protocol Version 4* (IPv4)**

- **1996: Internet Engineering Task Force (IETF) introduced *Internet Protocol Version 6* (IPv6) with 128-bit addresses**
  - Intended as the successor to IPv4

- **Majority of Internet traffic still carried by IPv4**
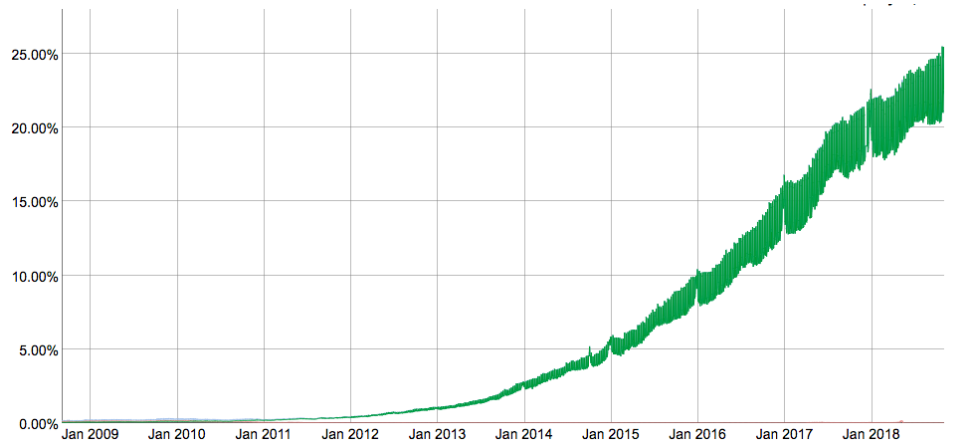
IPv6 traffic at Google

- **We will focus on IPv4, but will show you how to write networking code that is protocol-independent.**

(1) 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 {
    uint32_t  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`

- **Use `getaddrinfo` and `getnameinfo` functions (described later) to convert between IP addresses and dotted decimal format.**

- **Functions for converting between binary IP addresses and dotted decimal strings:**
  - `inet_pton`: dotted decimal string → IP address in network byte order
  - `inet_ntop`: IP address in network byte order → dotted decimal string
  - "n" denotes network, "p" denotes presentation
  - Out-of-date: `inet_aton` & `inet_ntoa`

IP Address Structure

■ **IP (V4) Address space divided into classes:**

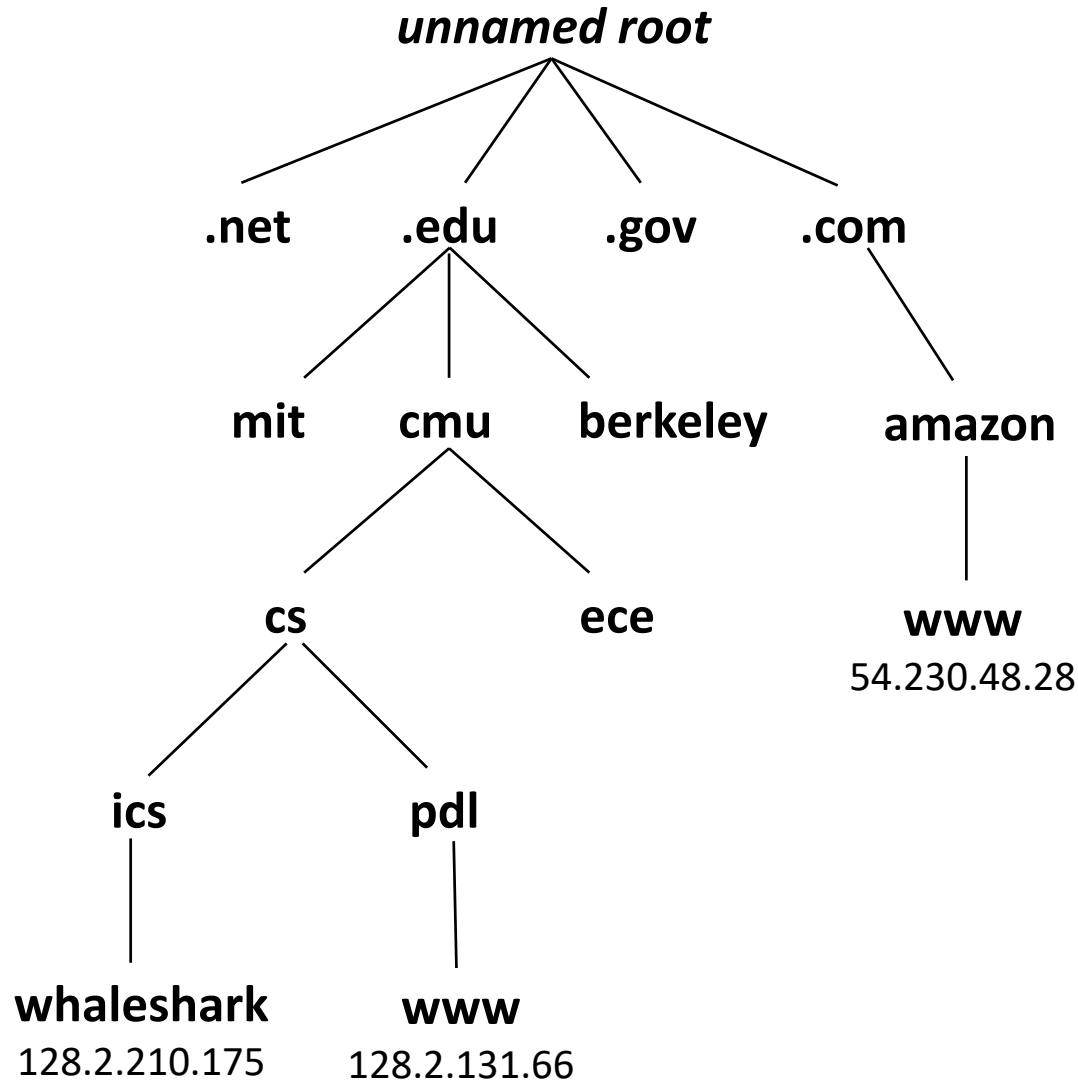|	0 1 2 3	8	16	24	31
Class A	**0** Net ID		Host ID		
Class B	**1 0** Net ID			Host ID	
Class C	**1 1 0** Net ID				Host ID
Class D	**1 1 1 0** Multicast address				
Class E	**1 1 1 1** Reserved for experiments				

■ **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, PKU written as 162.105.0.0/16

    ▪ Class B address

■ **Un-routed (private) IP addresses:**

  10.0.0.0/8   172.16.0.0/12   192.168.0.0/16

(2) Internet Domain Names

unnamed root

.net   .edu   .gov   .com   *First-level domain names*

mit   cmu   berkeley   amazon   *Second-level domain names*

cs   ece   www   *Third-level domain names*
54.230.48.28

ics   pdl

whaleshark   www
128.2.210.175   128.2.131.66

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 entries.***
  - Each host entry defines the mapping between a set of domain names and IP addresses.
  - In a mathematical sense, a host entry is an equivalence class of domain names and IP addresses.

Properties of DNS Mappings

- **Can explore properties of DNS mappings using `nslookup`**

  - (Output edited for brevity)

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

  ```
  linux> nslookup localhost
  Address: 127.0.0.1
  ```

- **Use `hostname` to determine real domain name of local host:**

  ```
  linux> hostname
  whaleshark.ics.cs.cmu.edu
  ```

31

Properties of DNS Mappings (cont)

- **Simple case: one-to-one mapping between domain name and IP address:**

```
linux> nslookup whaleshark.ics.cs.cmu.edu
Address: 128.2.210.175
```

- **Multiple domain names mapped to the same IP address:**

```
linux> nslookup cs.mit.edu
Address: 18.62.1.6
linux> nslookup eecs.mit.edu
Address: 18.62.1.6
```

Properties of DNS Mappings (cont)

■ **Multiple domain names mapped to multiple IP addresses:**

```
linux> nslookup www.twitter.com
Address: 104.244.42.65
Address: 104.244.42.129
Address: 104.244.42.193
Address: 104.244.42.1

linux> nslookup twitter.com
Address: 104.244.42.129
Address: 104.244.42.65
Address: 104.244.42.193
Address: 104.244.42.1
```

■ **Some valid domain names don't map to any IP address:**

```
linux> nslookup ics.cs.cmu.edu
(No Address given)
```

Properties of DNS Host Entries (obsolete)

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

■ **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.

A Program That Queries DNS (obsolete)

```
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;/* pointer to a DNS host entry structure */

    if (inet_aton(argv[1], &addr) != 0)
        hostp = Gethostbyaddr((const char *)&addr, sizeof(addr),
            AF_INET);           128.2.194.242 to 0x8002C2F2
    else
        hostp = Gethostbyname(argv[1]);
    printf("official hostname: %s\n", hostp->h_name);
                                // print host name
    for (pp = hostp->h_aliases; *pp != NULL; pp++)
        printf("alias: %s\n", *pp); // print all alias names

    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));
    } // print all addresses
}
```

Using DNS Program (obsolete)

```
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
```

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
```
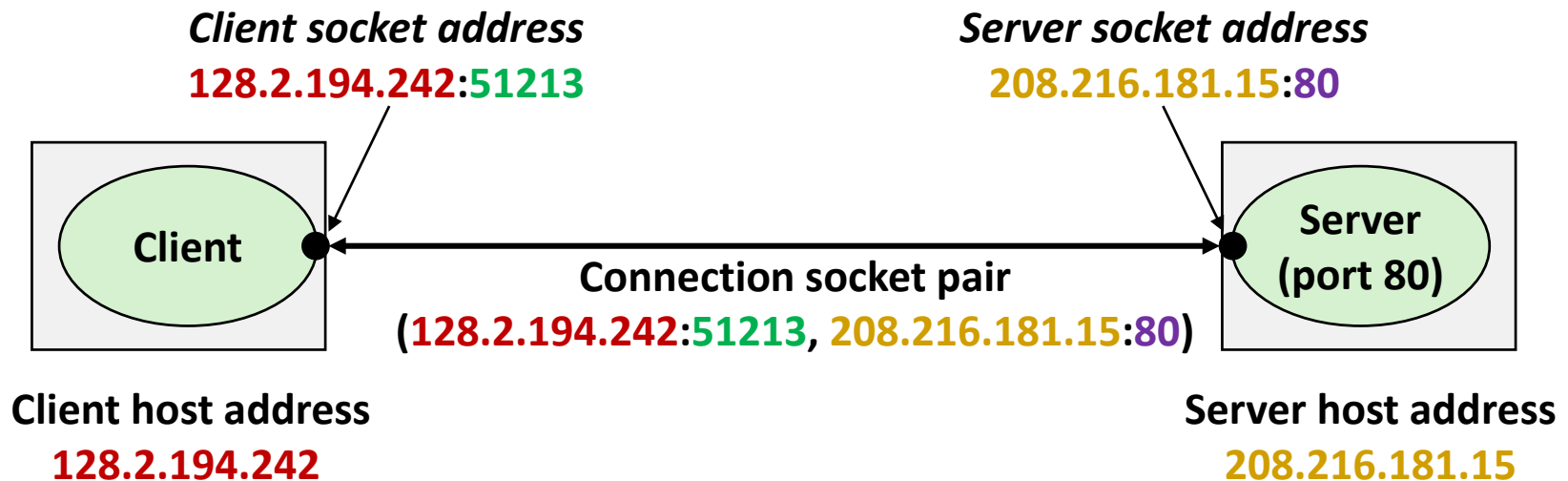
(3) Internet Connections

- **Clients and servers communicate by sending streams of bytes over *connections*. Each connection is:**
  - *Point-to-point*: connects a pair of processes.
  - *Full-duplex*: data can flow in both directions at the same time,
  - *Reliable*: stream of bytes sent by the source is eventually received by the destination in the same order it was sent.

- ***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 by client kernel 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)

Well-known Service Names and Ports

- **Popular services have permanently assigned *well-known ports* and corresponding *well-known service names*:**
  - echo servers: echo 7
  - ftp servers: ftp 21
  - ssh servers: ssh 22
  - email servers: smtp 25
  - Web servers: http 80

- **Mappings between well-known ports and service names is contained in the file `/etc/services` on each Linux machine.**
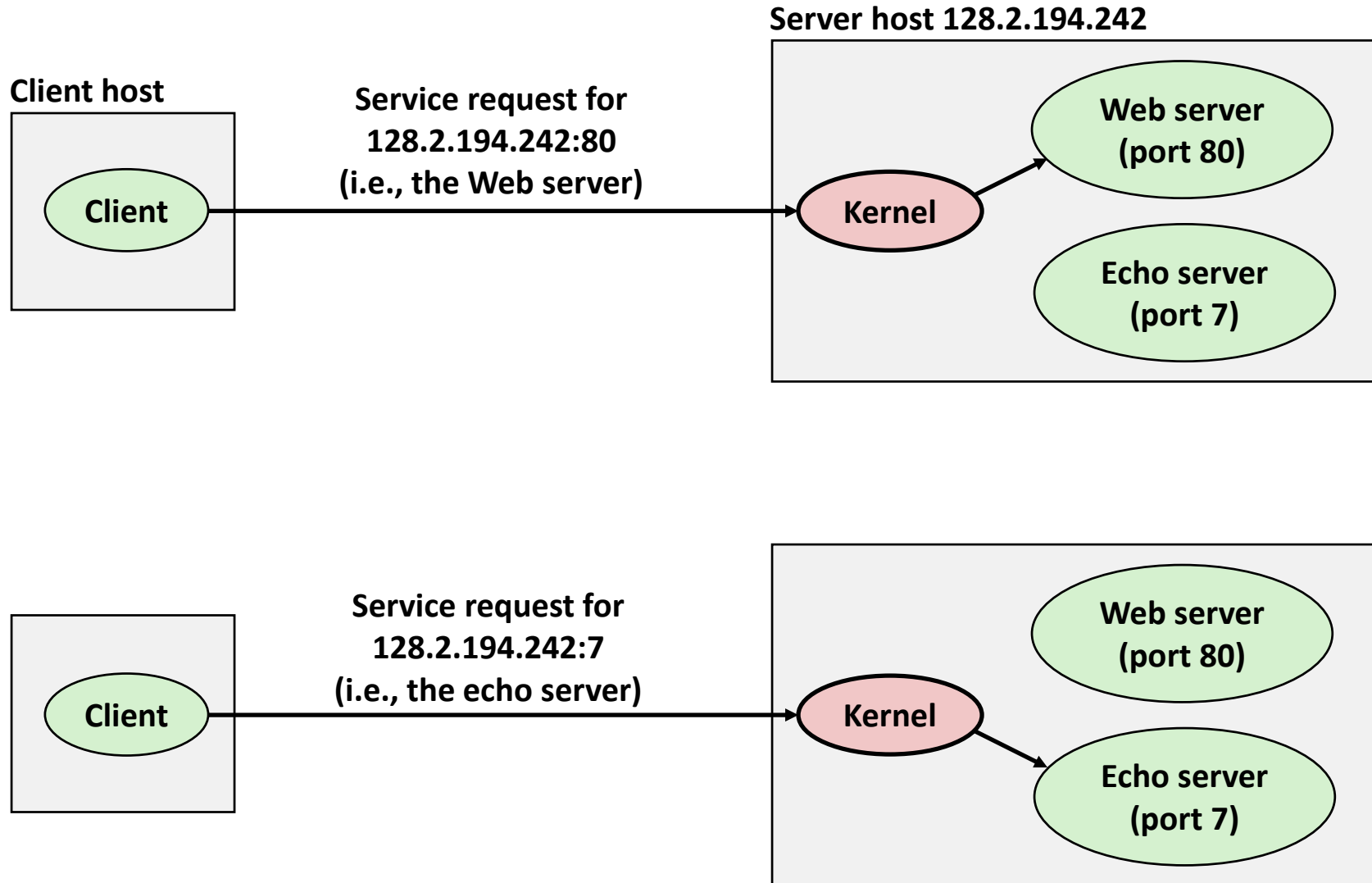
Anatomy of a Connection

- **A connection is uniquely identified by the socket addresses of its endpoints (*socket pair*)**
  - `(cliaddr:cliport, servaddr:servport)`

Client socket address
128.2.194.242:51213

Server socket address
208.216.181.15:80

Client

Server (port 80)

Connection socket pair
(128.2.194.242:51213, 208.216.181.15:80)

Client host address
128.2.194.242

Server host address
208.216.181.15

51213 is an ephemeral port allocated by the kernel

80 is a well-known port associated with Web servers

40

Using Ports to Identify Services

Server host 128.2.194.242

Client host

**Service request for
128.2.194.242:80
(i.e., the Web server)**

Client

Kernel

**Web server
(port 80)**

**Echo server
(port 7)**

**Service request for
128.2.194.242:7
(i.e., the echo server)**

Client

Kernel

**Web server
(port 80)**

**Echo server
(port 7)**

Sockets Interface

- **Set of system-level functions used in conjunction with Unix I/O to build network applications.**

- **Created in the early 80's as part of the original Berkeley distribution of Unix that contained an early version of the Internet protocols.**

- **Available on all modern systems**
    - Unix variants, Windows, OS X, IOS, Android, ARM

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**

Client ↔ Server

`clientfd`   `serverfd`

- **The main distinction between regular file I/O and socket I/O is how the application "opens" the socket descriptors**

Outline

- **Computer networking**

- **Network protocols**

- **Global IP Internet**

- **Programmer's view of Internet**

- **Evolution of Internet**

Evolution of 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:**
  - Laptop at CMU (wired connection)
    - IP address 128.2.213.29 (`bryant-tp4.cs.cmu.edu`)
    - Assigned statically
  - Laptop at home
    - IP address 192.168.1.5
    - Only valid within home network

Evolution of Internet: Firewalls

176.3.3.3

10.2.2.2 1 → **Firewall**

← 4

2 ←

3 →

216.99.99.99

Corporation X
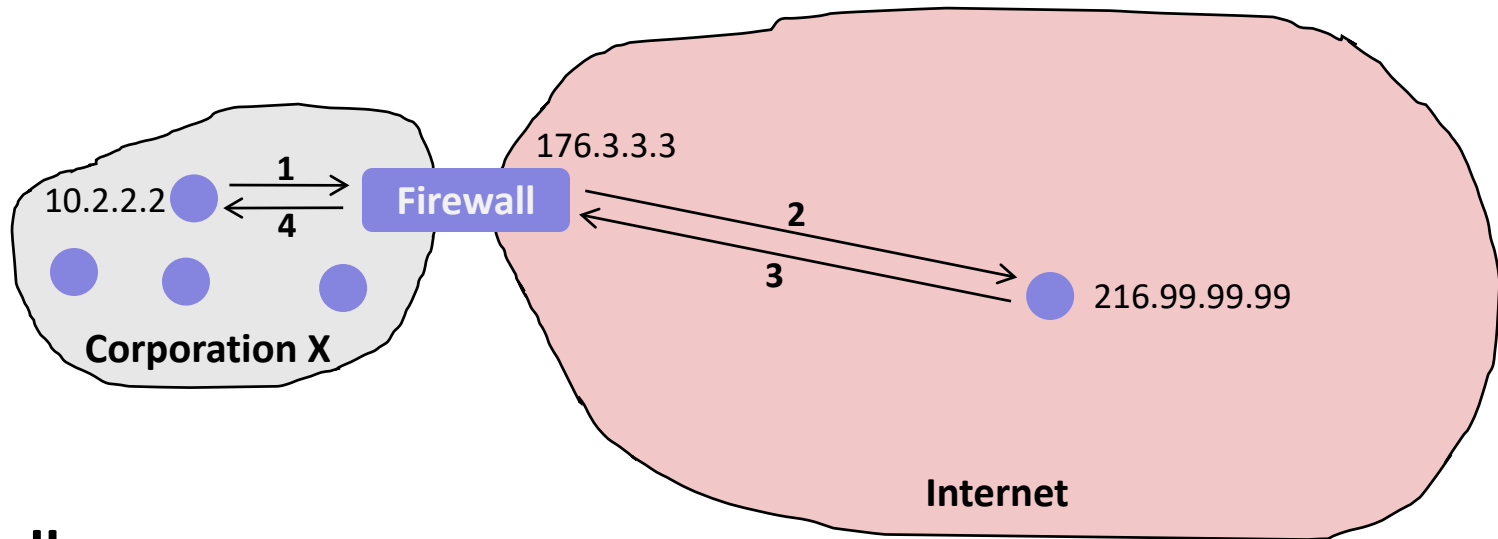
Internet

- **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

Next Lecture

- **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 established Internet connection**