



File Systems and Network Programming

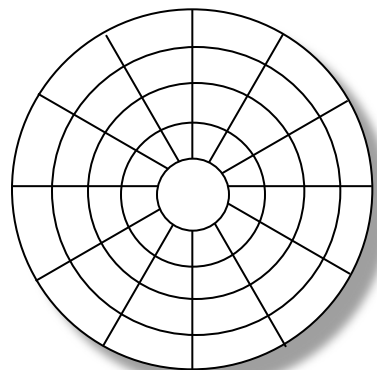
15-213/15-513/14-513: Introduction to Computer Systems
19th Lecture, Nov 11, 2025

Today

- **File Systems**
- Network types and structures
- Locating a host
- Setting up a connection

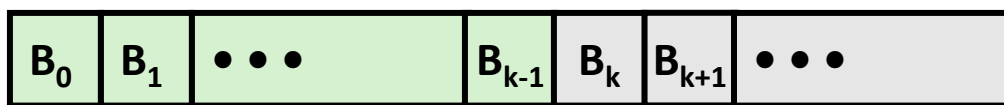
File System

- Manages disk blocks to provide a file abstraction



Surface* organized into tracks

Tracks divided into sectors



Current file position = k

***Durable storage has many architectures, but ultimately they expose “blocks”**

Making a File System

- **File systems start by formatting raw disk blocks**
 - Designate one (or more) blocks as “super”
 - Record the rest of the blocks as free

Managing a File System

- **“Super” block is the master block with information**
 - Type information
 - Size
 - Root directory
 - Free blocks

- **SFS has a flat directory structure, so the root directory is part of the superblock**

Finding a File

- A directory is a special file
 - Maps strings to files
 - Those files could also be directories

Index in directory

Max files in directory

Allocated?

```
for (fileEntry = 0; (unsigned long)fileEntry < FILE_COUNT_LIMIT;
    fileEntry++) {
    if (superBlock->files[fileEntry].first_block != 0 &&
        strcmp(superBlock->files[fileEntry].name, fileName) == 0) {
        return addOpenFileEntry(fileEntry);
    }
}
```

Check name

Opening a File

- **Find the file**
- **Create the three table entries**
 - Find an available file descriptor
 - Allocate an open file table entry
 - Pos, permissions, etc
 - Load file info into memory
 - *SFS is always in-memory, so this is implicit

Reading a File

- The file system will map file pos to disk blocks
- Lots of ways to map
 - Contiguous
 - Linked / FAT ← SFS
 - Indexed

Writing a File

- **Like reading, but the file could grow**
 - SFS preallocates space
 - Interesting synchronization

Deleting a File

- **Like free(), but ...**
 - Can open files be deleted?

- **Two steps:**
 - Removing the mapping
 - Putting the blocks into the free list

SFS Specific Notes

■ “Shark” File System

- Uses mmap to bring the entire “disk” file into memory
- Treats the disk as an array of 512-byte blocks
- Block 0 is the superblock, other references to 0 are NULLs
- Flat directory structure

Further Notes

- <https://tcpp.cs.gsu.edu/curriculum/sites/default/files/Edupar115.pdf>
- **Scope of assignment:**
 - Average of 200 lines of additional code
 - 13 hours to complete (9 days of assigned work)
- **The tricky part is identifying critical sections**
 - Critical sections are defined by the shared variable / resource
 - That can be two (or more) threads calling the same or different functions

Part of sfs_open

```

    sfs_filesystem_t *superBlock = accessSuperBlock();
    int fileEntry;
    int emptyEntry = -1;
    for (fileEntry = 0; (unsigned long)fileEntry <
FILE_COUNT_LIMIT;
        fileEntry++)
    {
        if (superBlock->files[fileEntry].first_block != 0 &&
            strcmp(superBlock->files[fileEntry].name, fileName)
== 0)
        {
            return addOpenFileEntry(fileEntry);
        }
        else if (emptyEntry == -1 &&
            superBlock->files[fileEntry].first_block == 0)
        {
            emptyEntry = fileEntry;
        }
    }
}

```

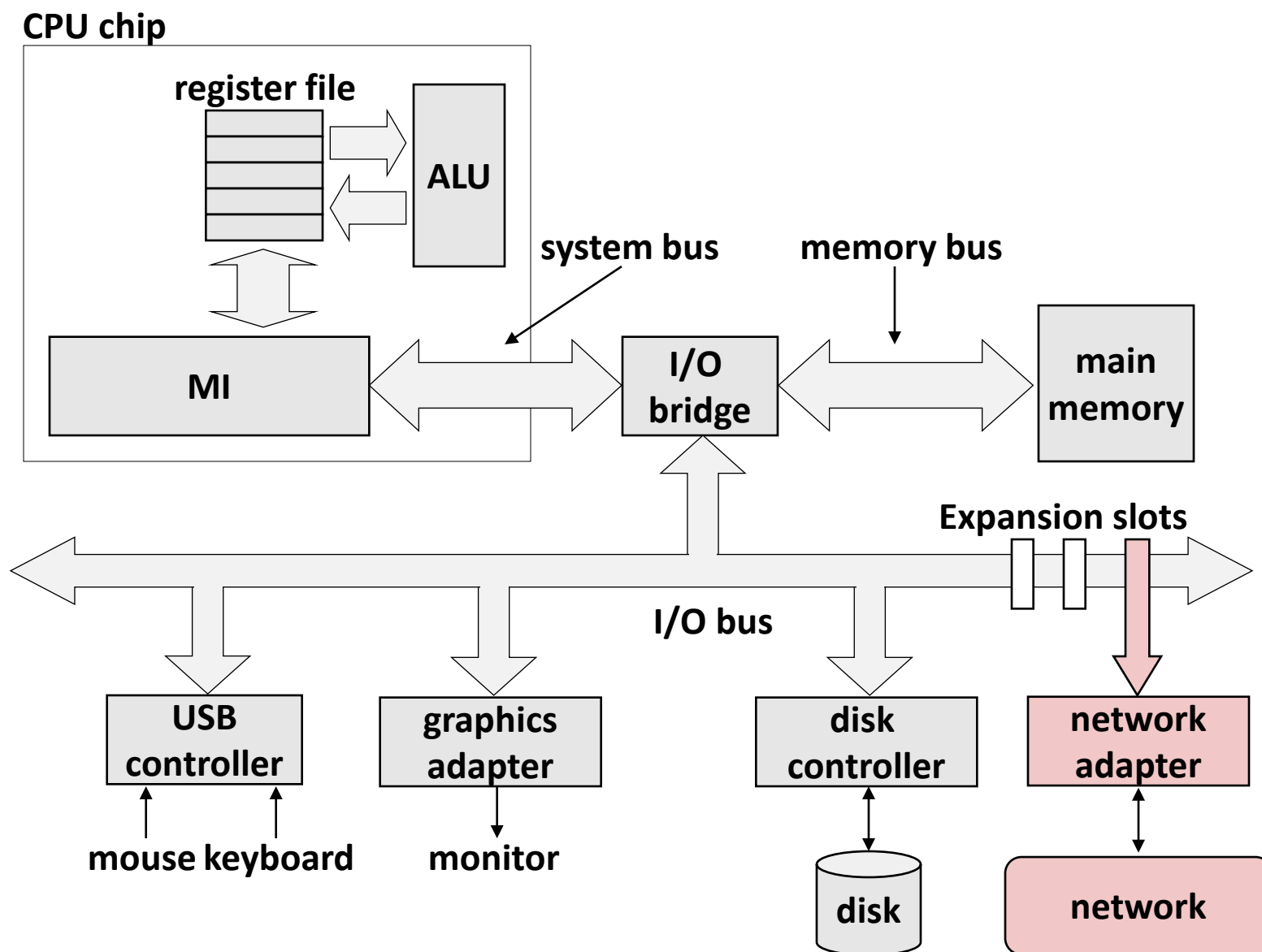
Today

- File Systems
- **Network types and structures**
- Locating a host
- Setting up a connection

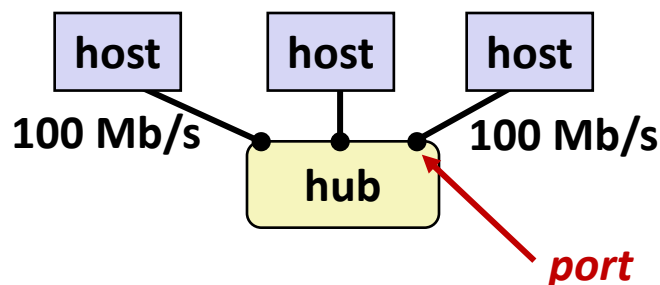
Computer Networks

- A ***network*** is a hierarchical system of boxes and wires organized by geographical proximity
 - 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 (mostly optical) links
 - Also: SAN (Storage area network), MAN (Metropolitan), etc., etc.
- 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

Hardware Organization of a Network Host

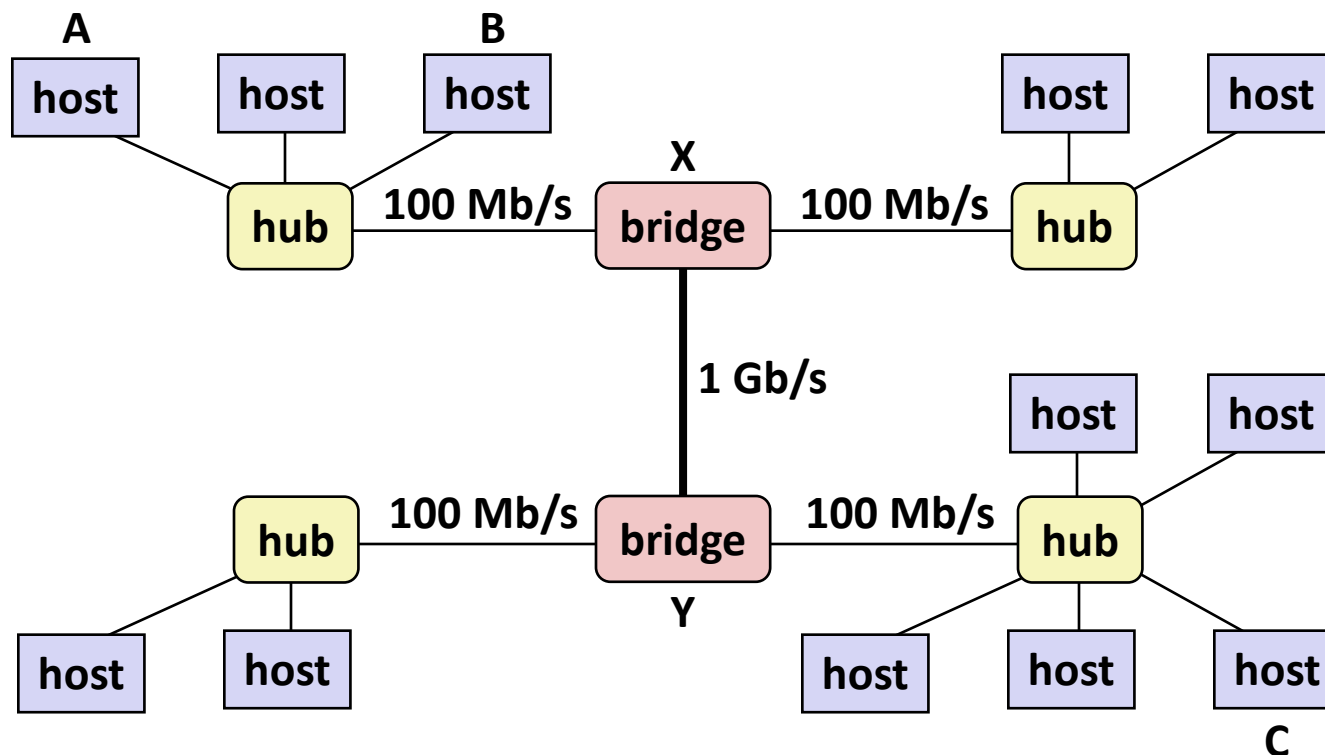


Example Lowest Level: Ethernet



- 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

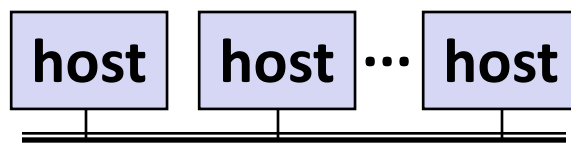
Next Level: Bridged Ethernet Segment



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

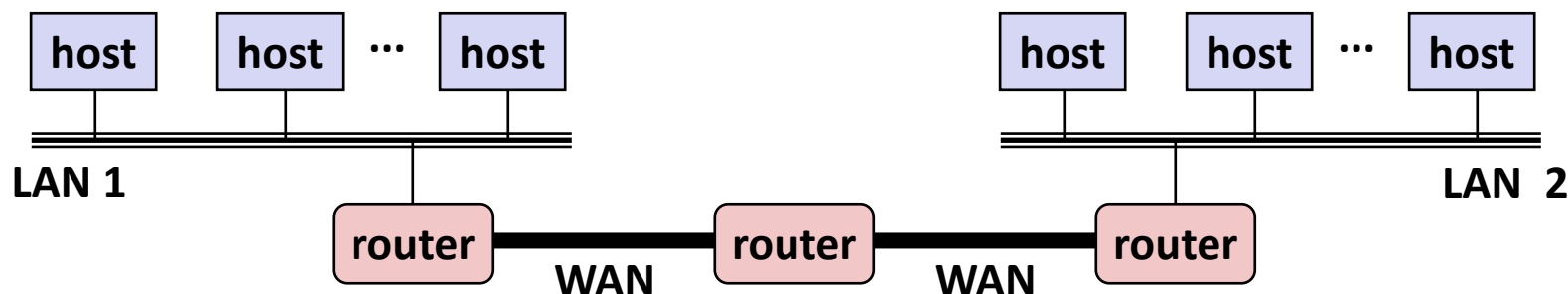
Conceptual View of LANs

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



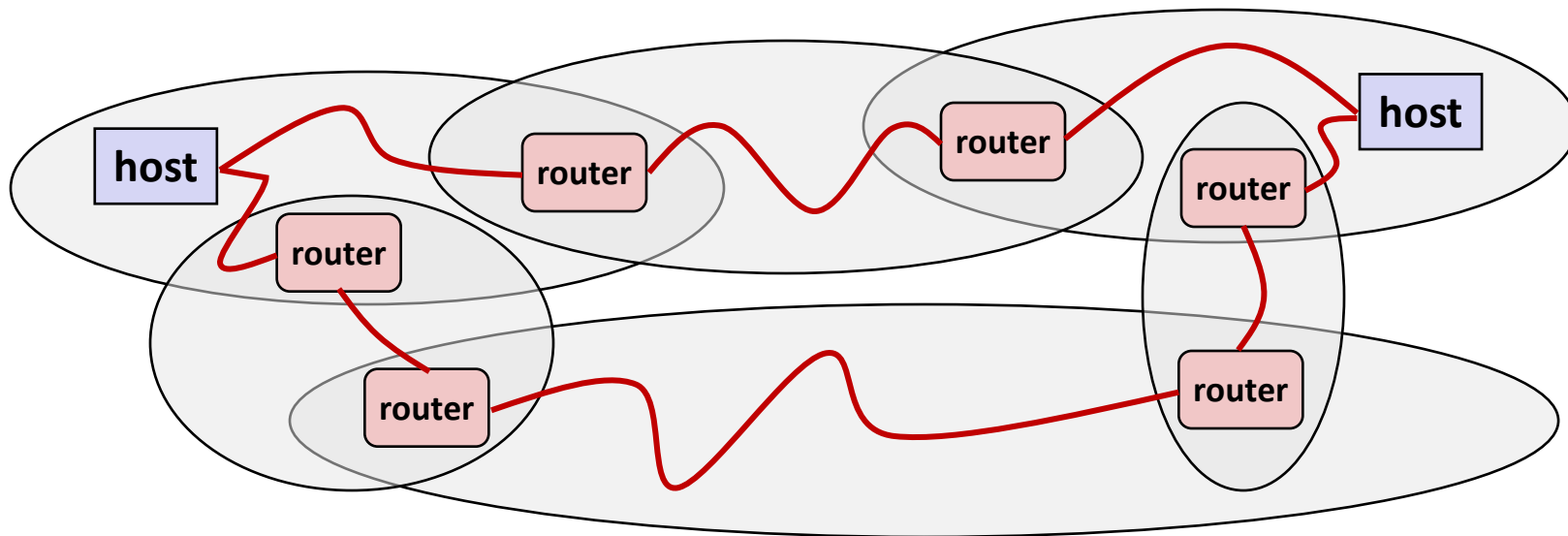
Next Level: internets

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

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?

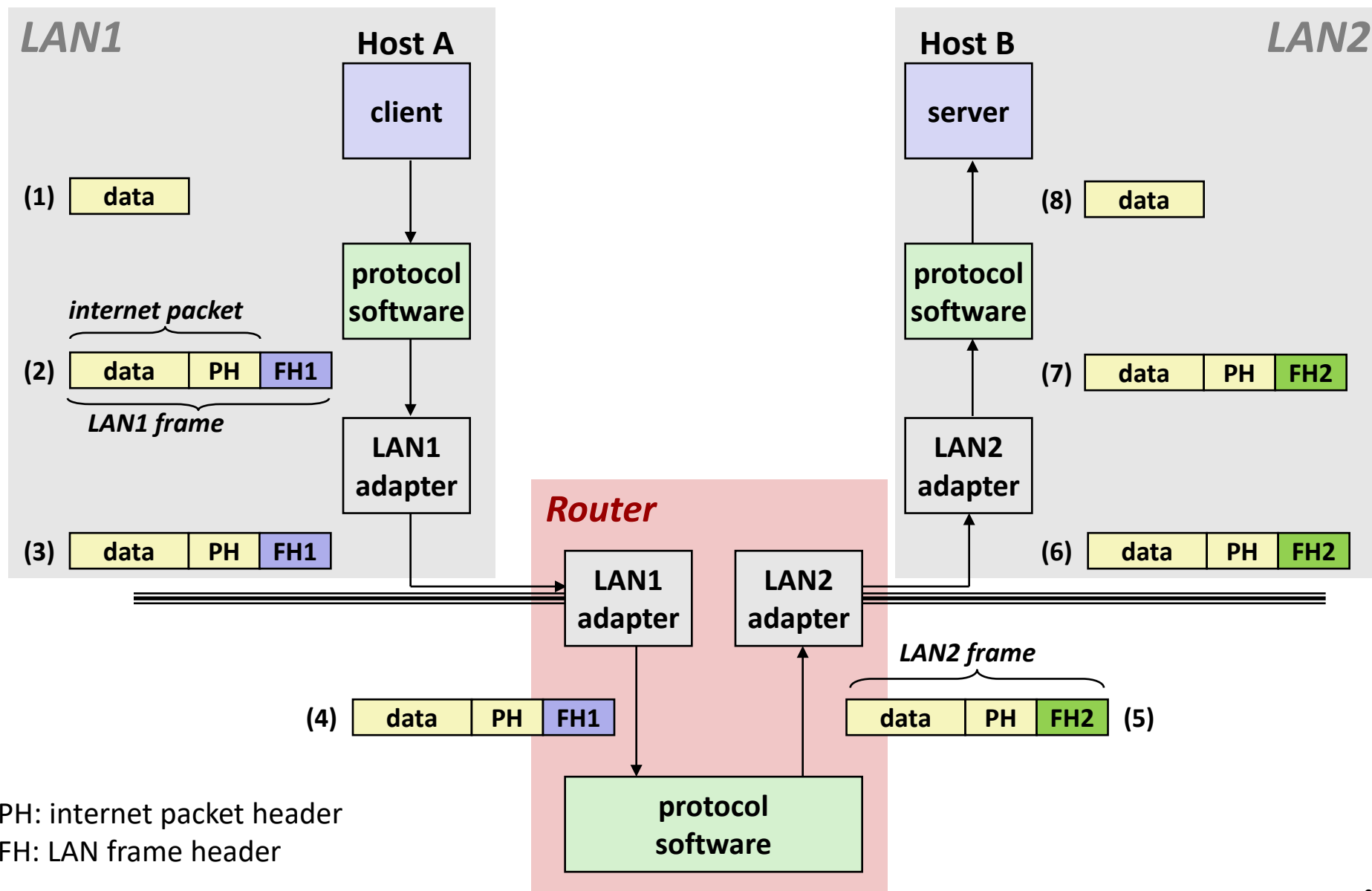
■ Provides a *naming scheme*

- 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

Transferring internet Data Via Encapsulation



Other Issues

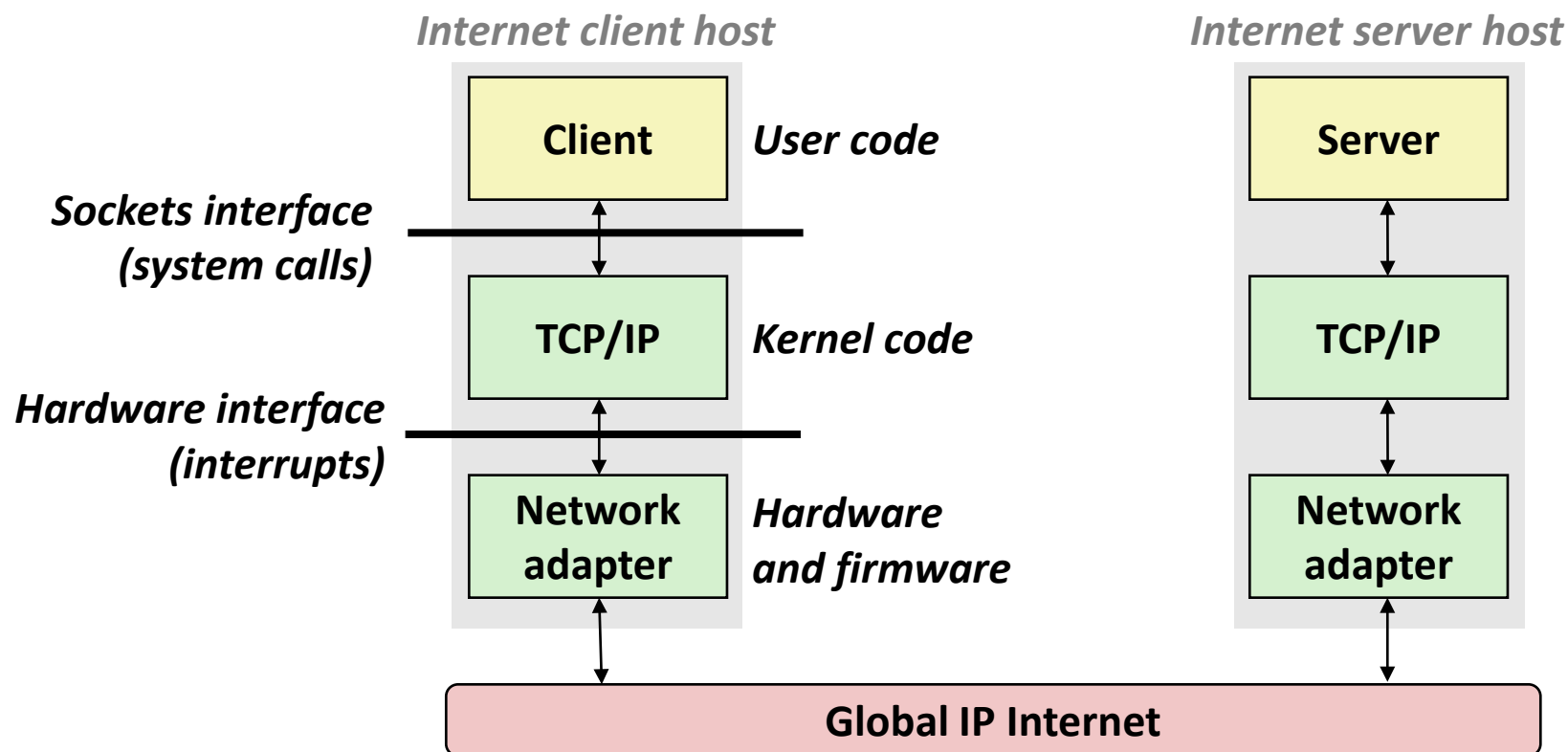
- **We are glossing over a number of important questions:**
 - What if different networks have different maximum frame sizes? (segmentation)
 - How do routers know where to forward frames?
 - How are routers informed when the network topology changes?
 - What if packets get lost?

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

Global IP Internet (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 (User 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



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. As a convenience for humans, the Domain Name System maps a set of identifiers called Internet *domain names* to IP addresses:

- www.cs.cmu.edu “resolves to” 128.2.217.3

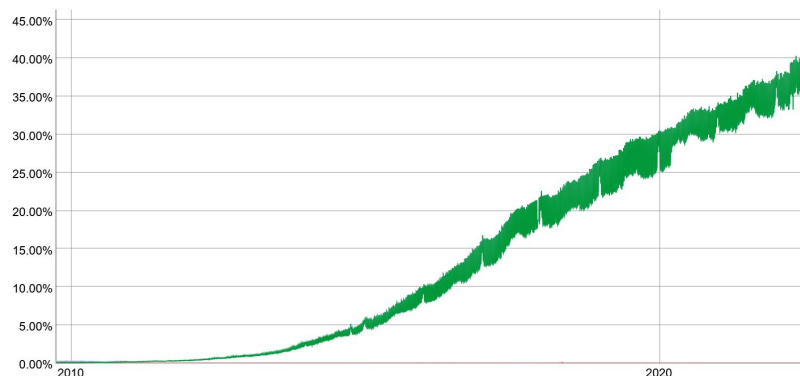
3. A process on one Internet host can communicate with a process on another Internet host over a *connection*

Aside: IPv4 and IPv6

- **IPv4 (Internet Protocol version 4) specified 1981**
 - 32-bit host addresses (192.0.2.43)
 - Known to not be enough for everyone since ~1990
- **IPv6 (Internet Protocol version 6) specified 1996**
 - 128-bit addresses (2001:0db8:0:0:0:0:cafe:la7e)
 - Intended to replace IPv4
 - Very slow adoption due to need to replace routers (CMU's network doesn't support IPv6 at all!)
- **Application programmers mostly don't have to care**
 - Sockets API makes it easy to write code that seamlessly uses either, as necessary

IPv6 traffic to Google

<https://www.google.com/intl/en/ipv6/statistics.html>



(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) */  
};
```

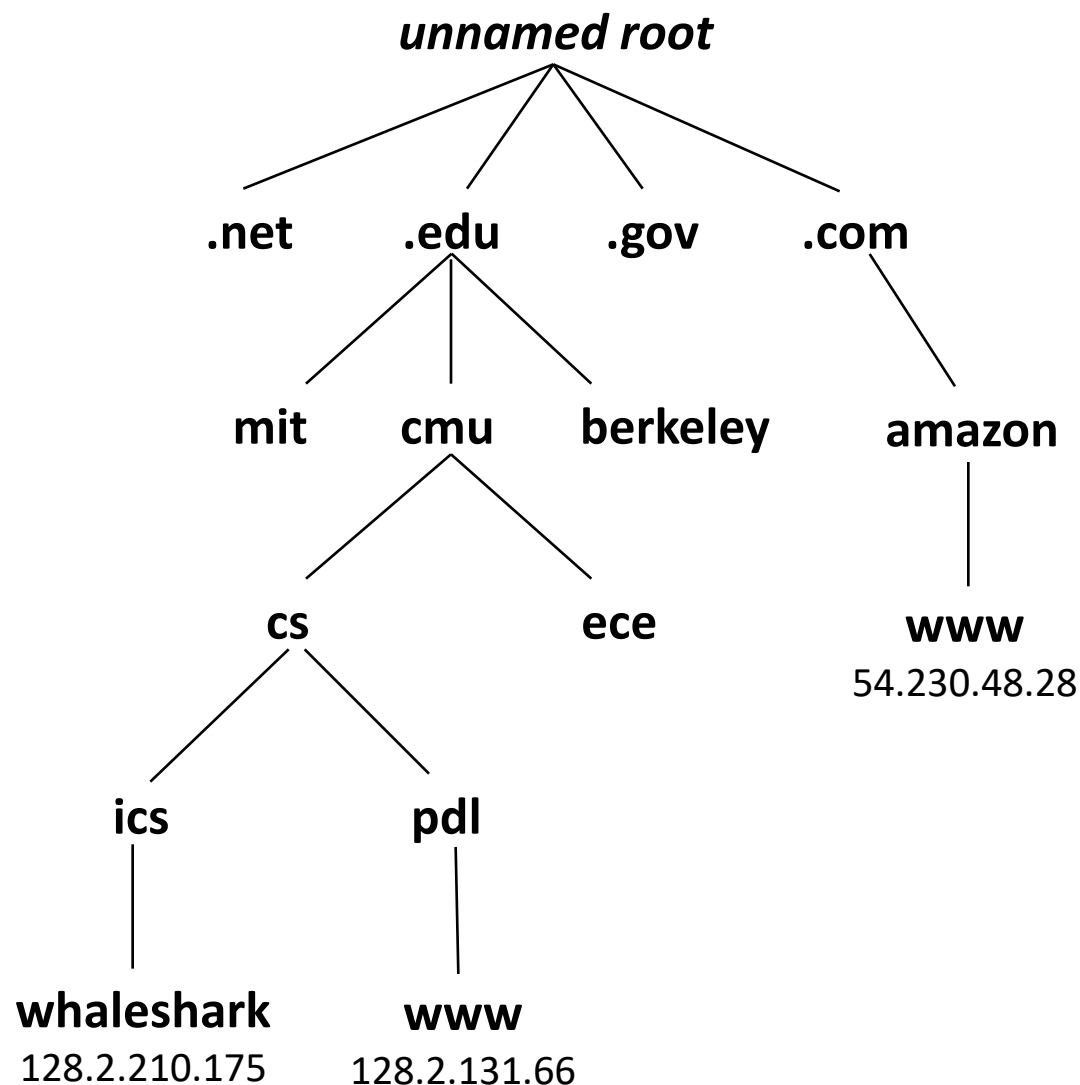
Dotted Decimal Notation

- 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.

Today

- File Systems
- Network types and structures
- **Locating a host**
- Setting up a connection

(2) Internet Domain Names



Domain Naming System (DNS)

- The Internet maintains a mapping between IP addresses and domain names in a 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
```

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: 18.25.0.23  
linux> nslookup eeecs.mit.edu  
Address: 18.25.0.23
```

- And backwards:

```
linux> nslookup 18.25.0.23  
23.0.25.18.in-addr.arpa      name = eeecs.mit.edu.
```

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 www.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)
```

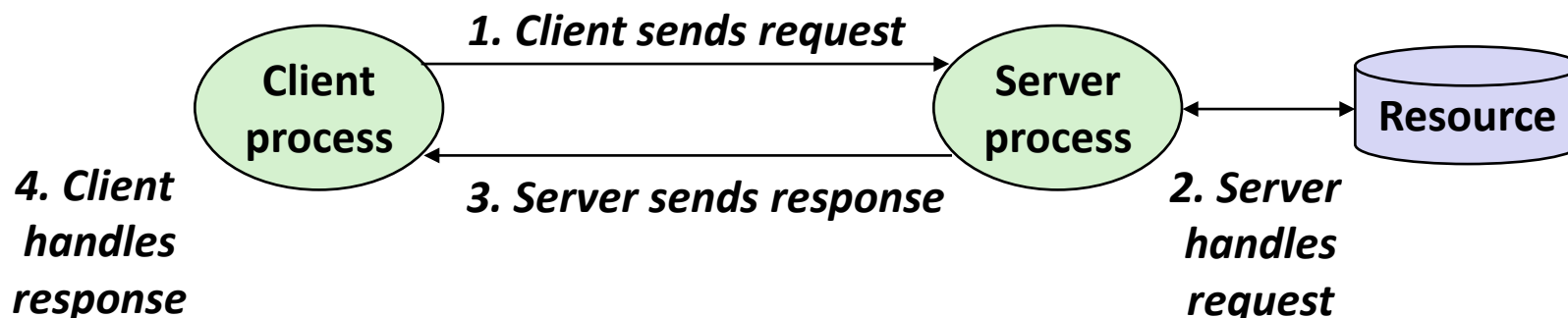
Today

- File Systems
- Network types and structures
- Locating a host
- **Setting up a connection**

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)



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

(3) Internet Connections

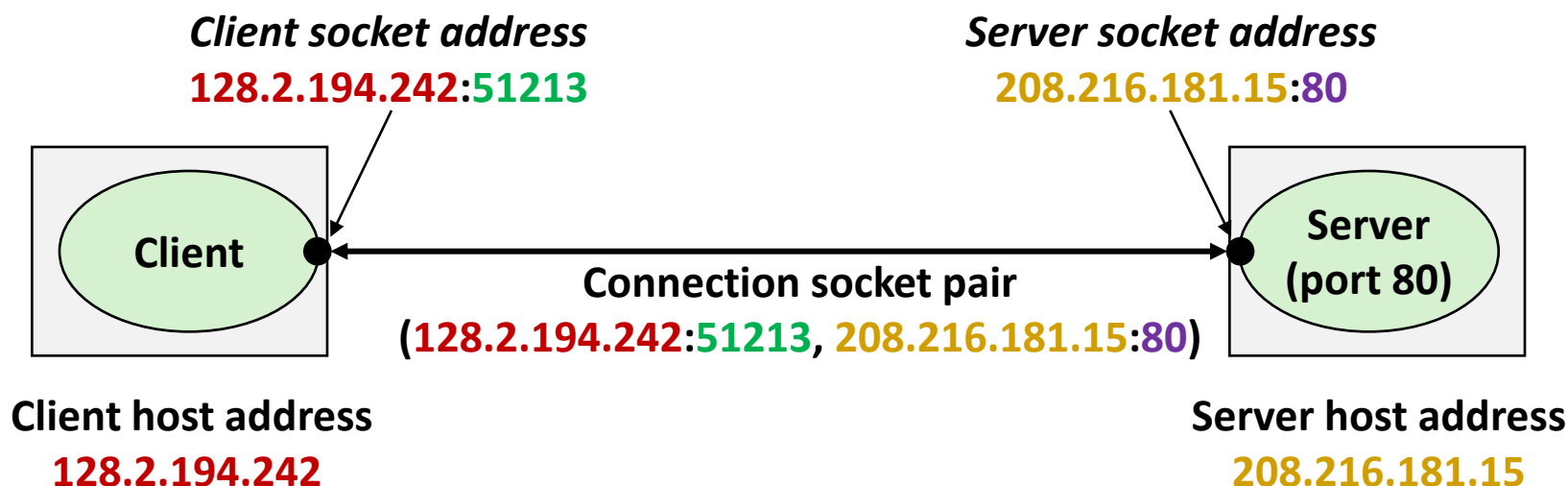
- Clients and servers most often communicate by sending streams of bytes over TCP *connections*. Each connection is:
 - *Point-to-point*: connects a pair of processes.
 - *Full-duplex*: data can flow in both directions at the same time,
 - *Reliable*: stream of bytes sent by the source is eventually 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
 - Unencrypted Web servers: http 80
 - SSL/TLS encrypted Web: https 443
- Mappings between well-known ports and service names is contained in the file `/etc/services` on each Linux machine.

Anatomy of a Connection

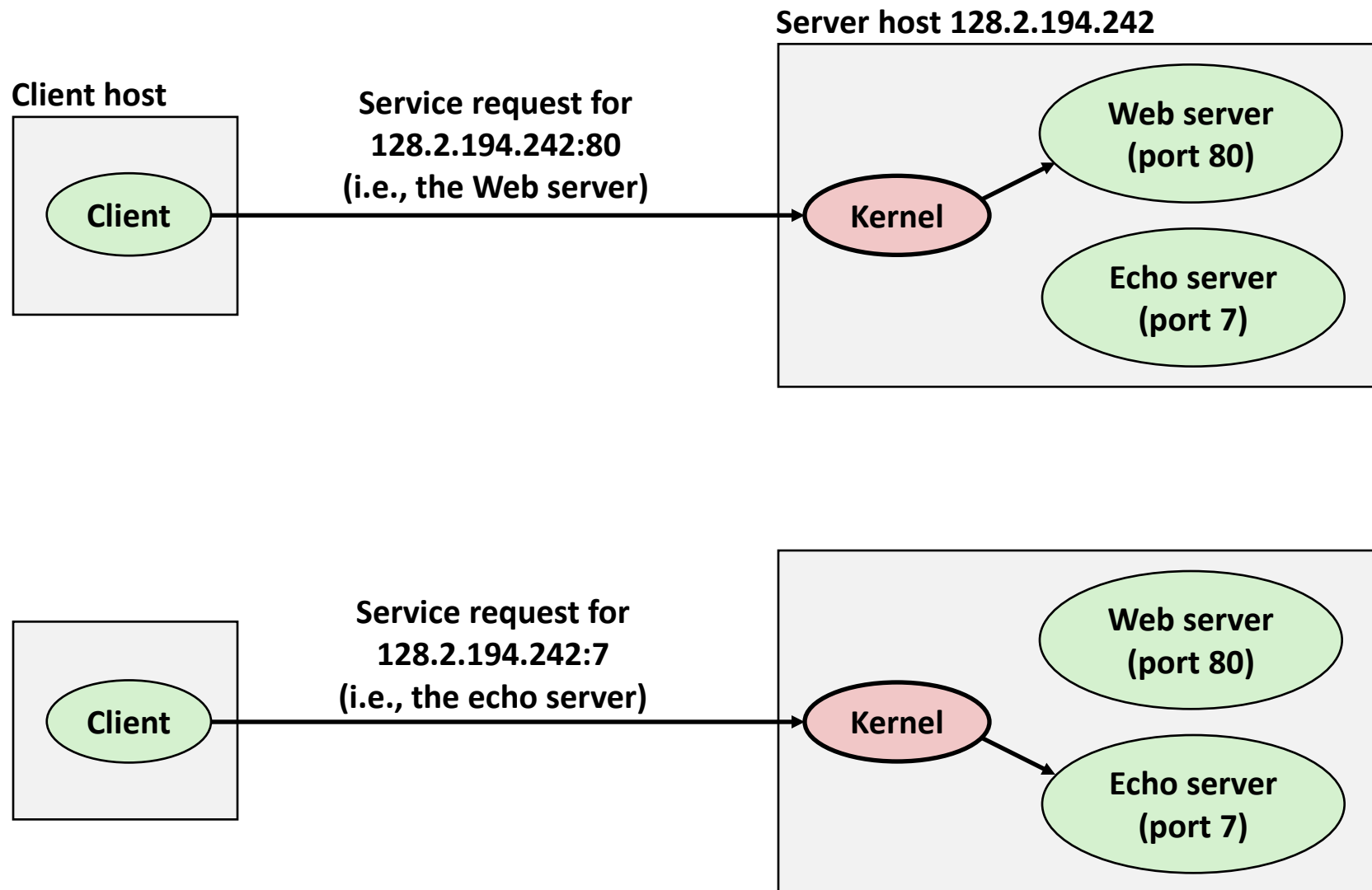
- A connection is uniquely identified by the socket addresses of its endpoints (*socket pair*)
 - (cliaddr:cliport, servaddr:servport)



51213 is an ephemeral port allocated by the kernel

80 is a well-known port associated with Web servers

Using Ports to Identify Services



Quiz

<https://canvas.cmu.edu/courses/49105/quizzes/150047>

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
- Using the FD abstraction lets you reuse code & interfaces

■ 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

Socket Programming Example

■ Echo server and client

■ Server

- Accepts connection request
- Repeats back lines as they are typed

■ Client

- Requests connection to server
- Repeatedly:
 - Read line from terminal
 - Send to server
 - Read reply from server
 - Print line to terminal

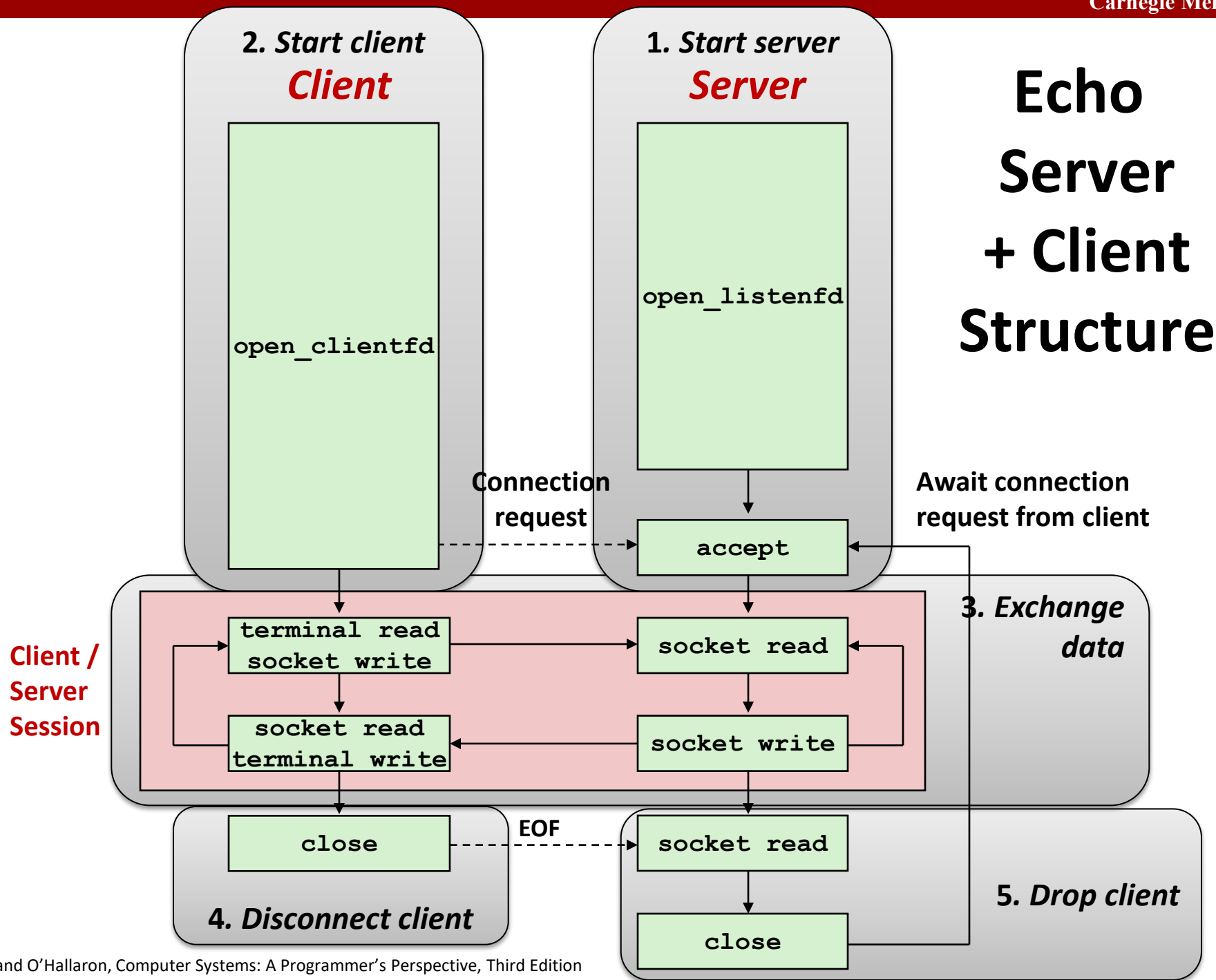
Echo Server/Client Session Example

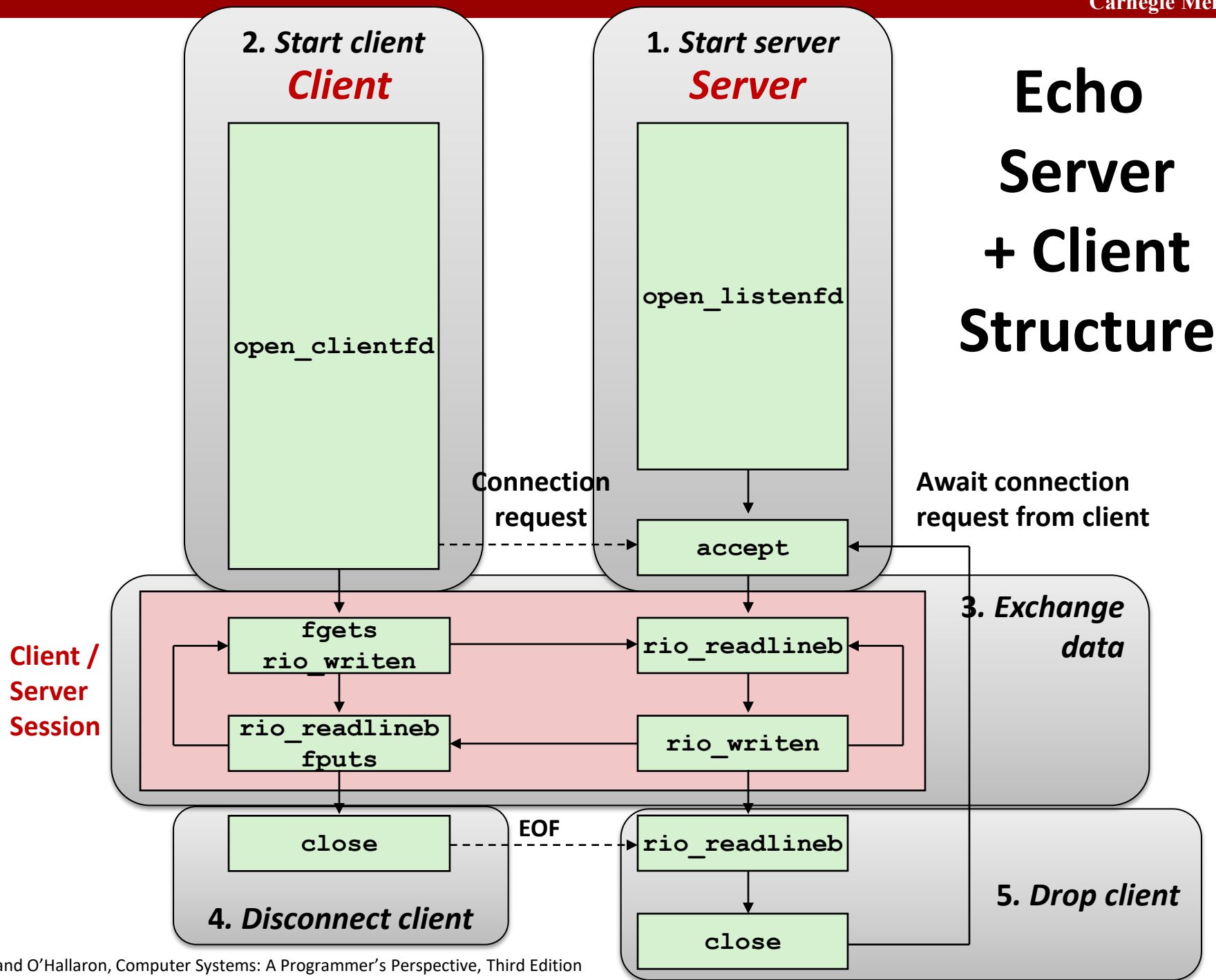
Client

```
bambooshark: ./echoclient whaleshark.ics.cs.cmu.edu 6616      (A)
This line is being echoed                                     (B)
This line is being echoed
This one is, too                                             (C)
This one is, too
^D
bambooshark: ./echoclient whaleshark.ics.cs.cmu.edu 6616      (D)
This one is a new connection                                 (E)
This one is a new connection
^D
```

Server

```
whaleshark: ./echoserveri 6616
Connected to (BAMBOOSHARK.ICS.CS.CMU.EDU, 33707)             (A)
server received 26 bytes                                     (B)
server received 17 bytes                                     (C)
Connected to (BAMBOOSHARK.ICS.CS.CMU.EDU, 33708)             (D)
server received 29 bytes                                     (E)
```

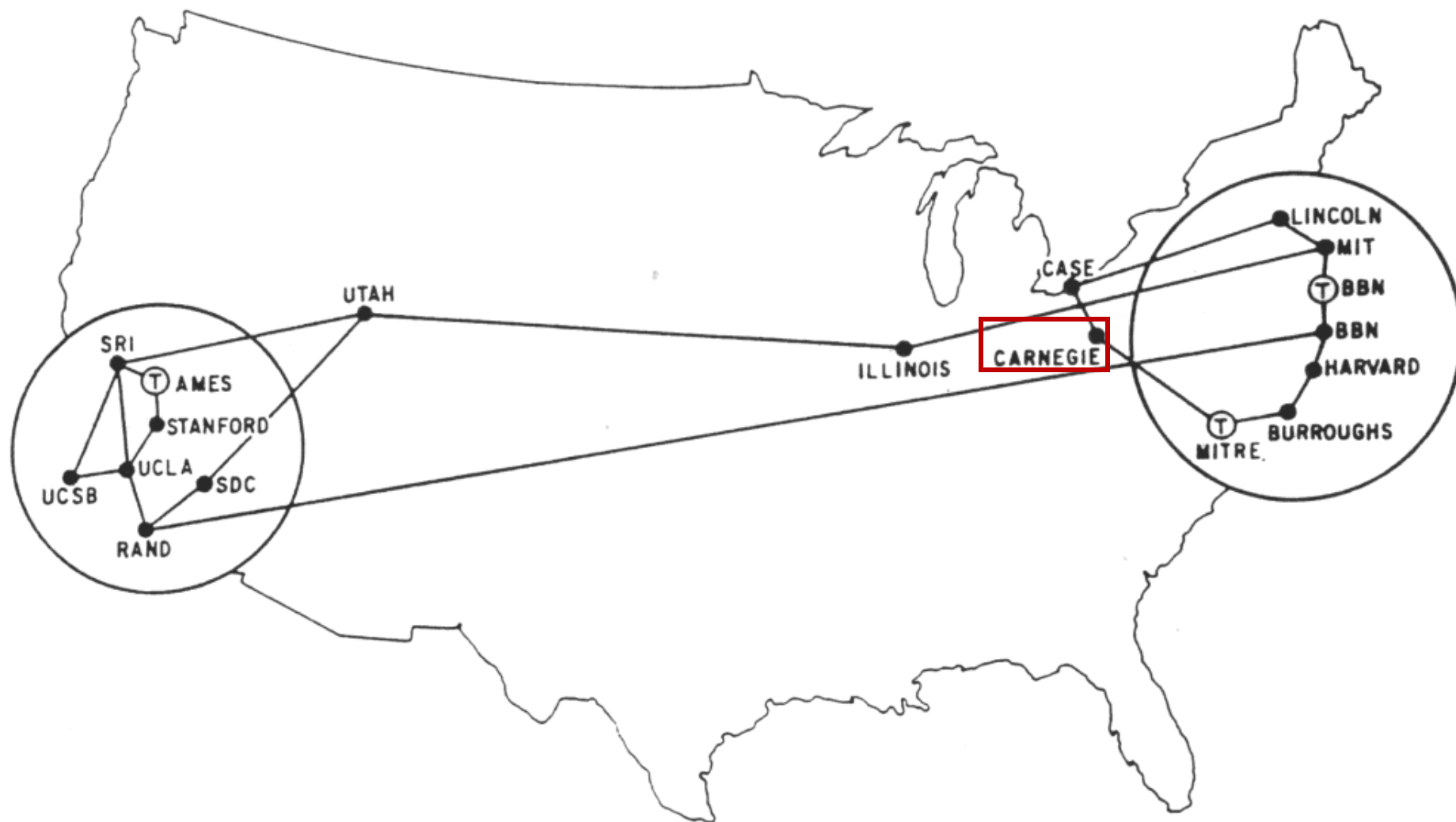




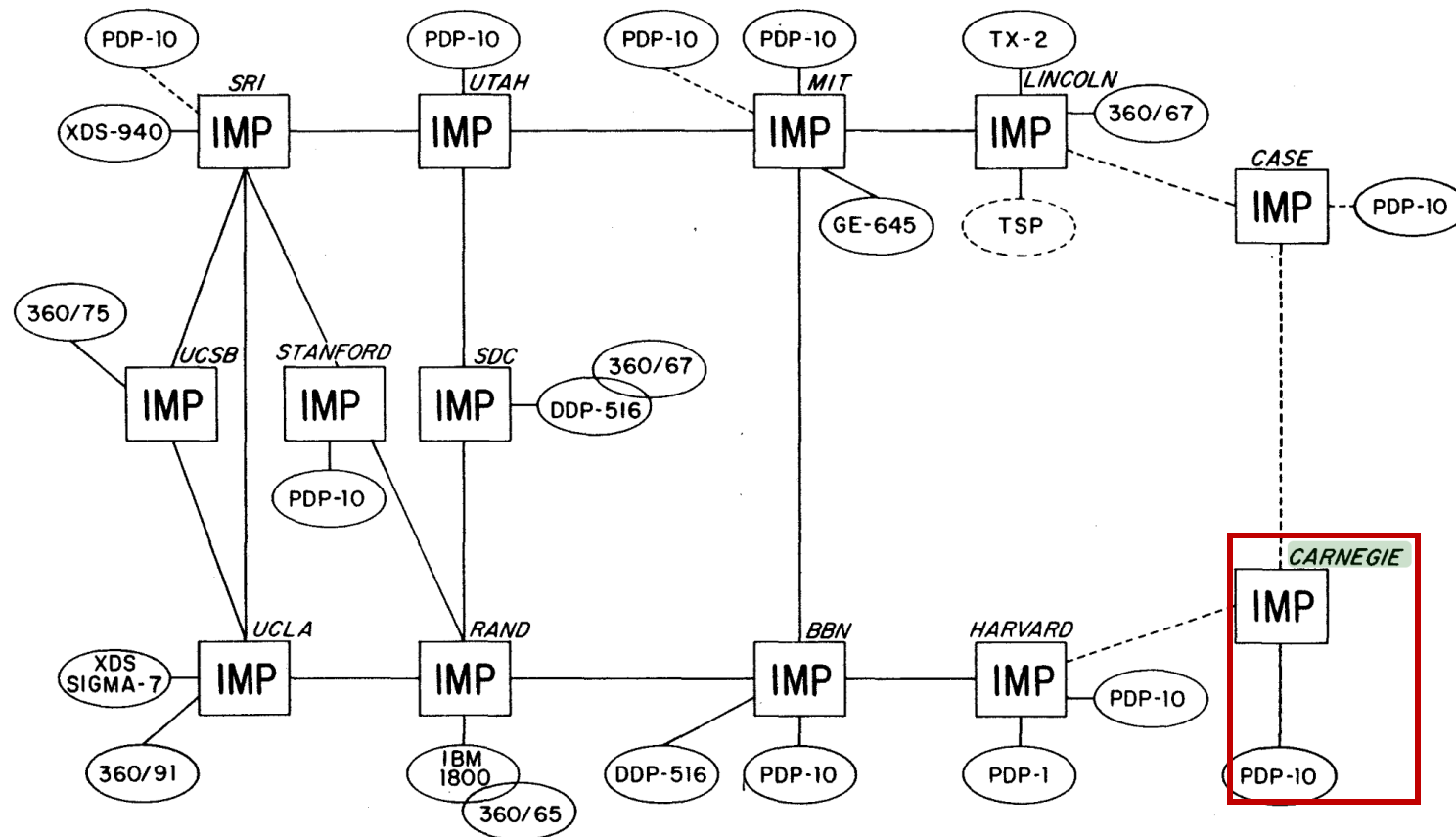
Appendix



The ARPANET in December 1969



The ARPANET in December 1970

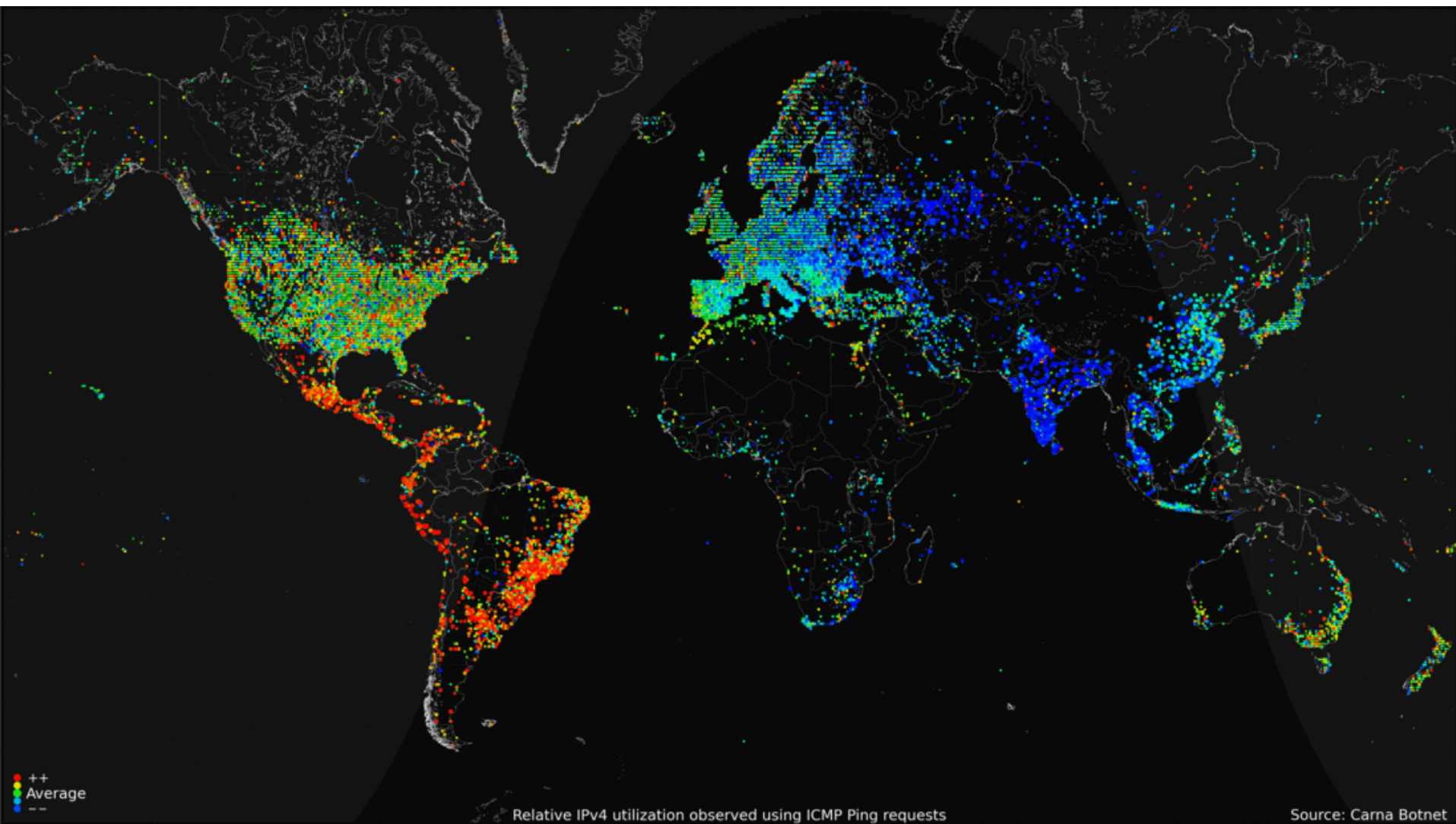


ARPA NET, DECEMBER 1970

TeleGeography | telecomegypt



A Map of 460 Billion Device Connections to the Internet collected by the Carna Botnet



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

