CS 4400
Computer Systems

LECTURE 22

Client-server programming model

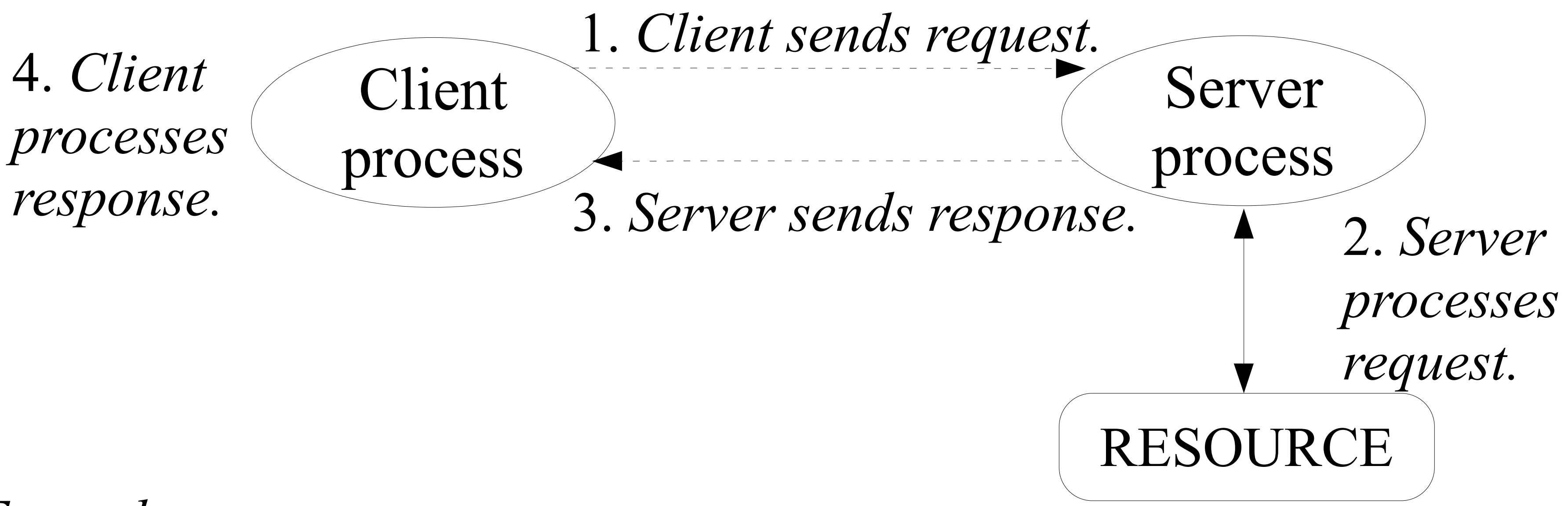
Networks

The Global IP Internet

Network Applications

- Web, email, popping up an X window, ...
- All network applications are based on the same basic client-server programming model.
  - an app consists of 1 server process and ≥ 1 client process(es)
  - the server manages some resource, providing some service for its clients by manipulating that resource
  - an FTP server manages disk files that it stores and retrieves on the behalf of its clients
  - an email server manages a spool file that it reads/updates ...
- Concepts we have already learned about play a role.
  - processes, signals, byte ordering, dynamic storage alloc, ...

Client-Server Transaction


1. *Client sends request.*
4. *Client processes response.*
Client process
Server process
3. *Server sends response.*
2. *Server processes request.*
RESOURCE

Example:

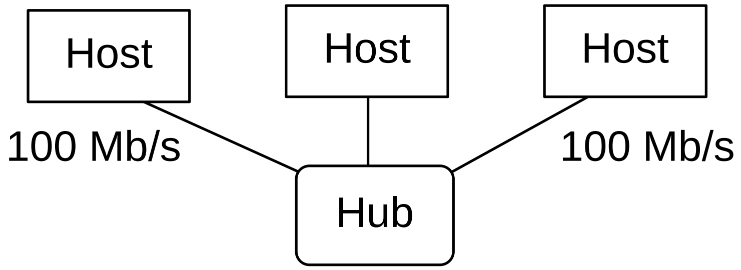
1. Web browser needs a file and sends a request to a Web server.
2. A Web server receives the request and reads a disk file.
3. A Web server sends the file back to the client.
4. After receiving the page from the server, the Web browser displays it to the screen.

Networks

- Clients and servers often run on separate hosts and communicate using a computer network.
- To a host, a network is just another I/O device.
  - data received from the network can be copied into memory, and data can be copied from memory to the network
- Physically, a network is a hierarchical system, organized by geographical proximity.
- Local Area Network (LAN) is at the lowest level.
  - *example*: spans a building or a campus
  - Ethernet is the most popular LAN technology

Ethernet

- An Ethernet segment consists of wires and a hub.
  - typically spans small areas like a room or a building floor
- Each wire attaches an adapter on a host to a port on the hub.
- The hub copies every bit that it receives on each port to every other port.
  - every host sees every bit



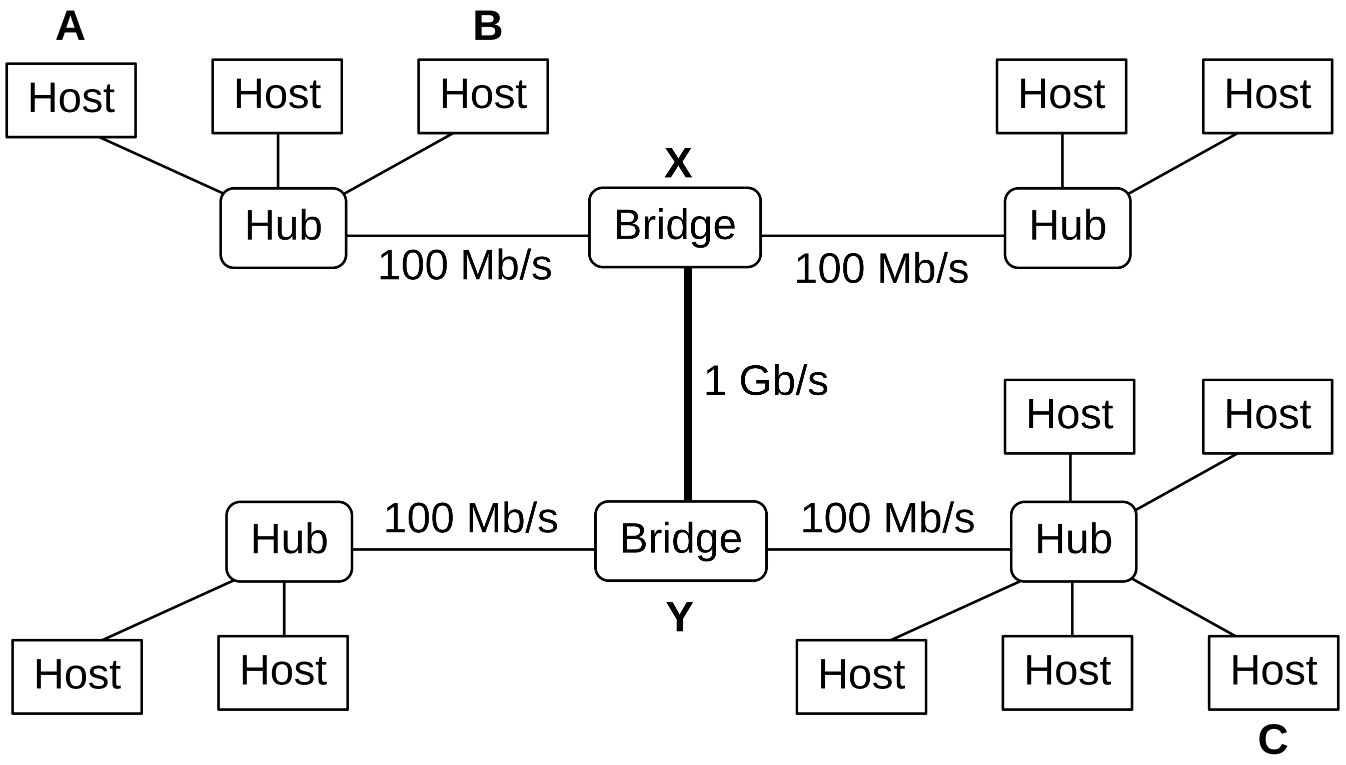
Frames

- Each Ethernet adapter has a globally unique 48-bit address stored in non-volatile memory on the adapter.
- A host can send a chunk of bits called a *frame* to any other host on the segment.
- Each frame includes a header and a payload.
  - header bits identify the source and destination of the frame and give the frame length
- Every host adapter sees the frame, but only the destination host actually reads it.

Bridged Ethernet Segments

- Multiple Ethernet segments can be connected into larger LANs, using wires and *bridges*.
    - typically span entire buildings or campuses
    - some wires connect bridges to bridges, others bridges to hubs
- Bridges selectively copy frames from one port to another port only when necessary.
    - makes better use of wire bandwidth than hubs
- *Example*: If host *A* sends a frame to host *B* (on the same segment), then bridge *X* will throw away the frame when it arrives at its input port.

Example: Bridged Ethernet



If host *A* sends a frame to host to the port *C*, then bridge *X* will copy the frame only to the port connected to bridge *Y*, which will copy the frame only to the port connected to bridge *C*'s segment.

Internet

- An *internet* is multiple LANs connected by specialized computers (called *routers*).
- Each router has a port for each network it is connected to.
- An internet can consist of LANs with radically different and incompatible technologies.
- A layer of protocol software running on each host and router smooths out the differences.
- Such software implements a protocol that governs how hosts and routers cooperate in order to transfer data.

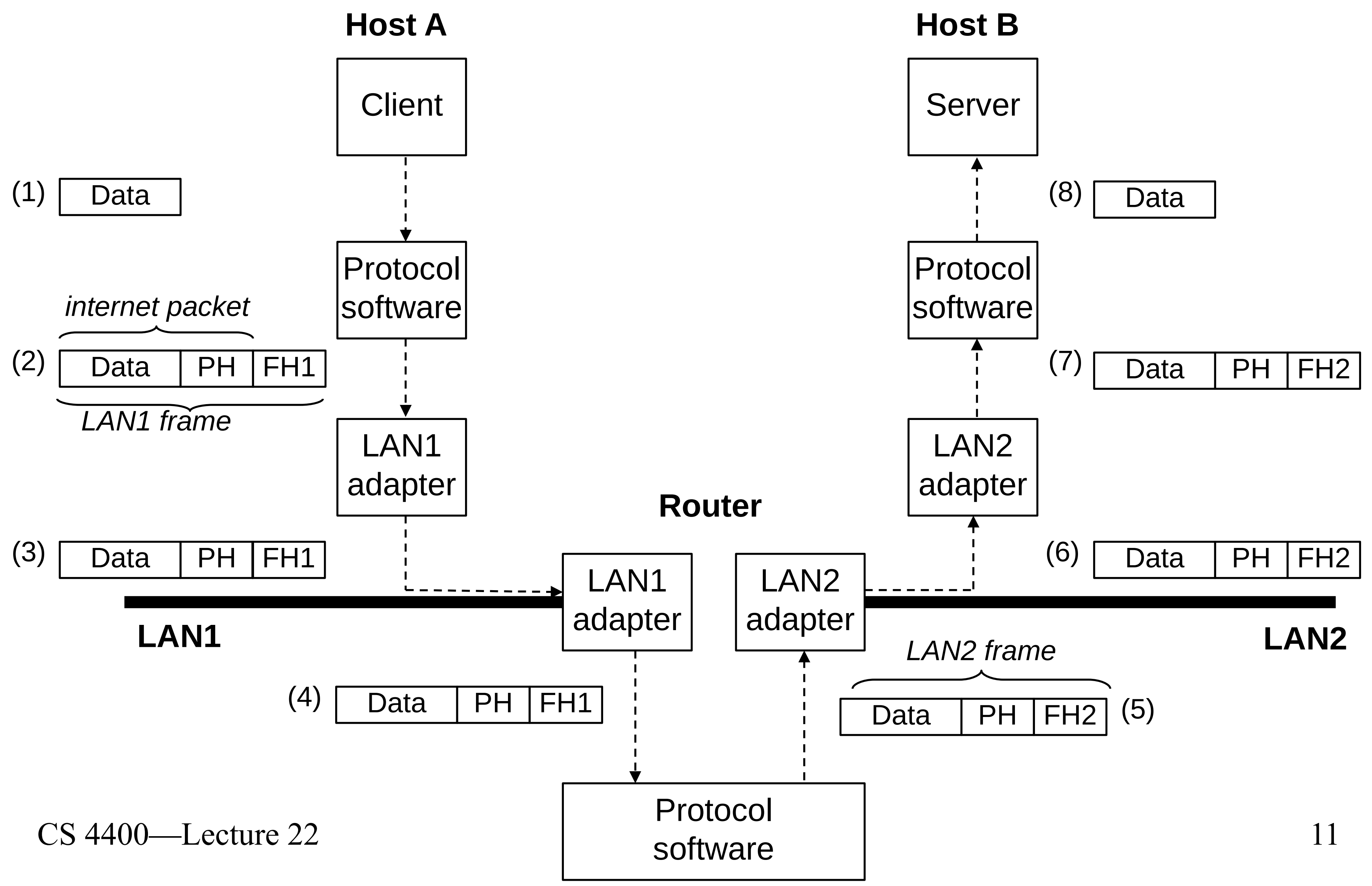
Protocol

The protocol must provide two basic capabilities.

- *Naming scheme*
  - Different LAN technologies have different and incompatible ways of assigning addresses to hosts.
  - The internet protocol defines a uniform host address format.
- *Delivery mechanism*
  - Different technologies have different and incompatible ways of encoding bits on wires and packaging bits into frames.
  - The internet protocol defines a uniform way to bundle up data bits into discrete chunks (called *packets*).
  - A packet consists of header (size, src/dst address) and payload.

Transfer of Data Across LANs

A client sends a sequence of data bytes to a server:



Transfer of Data Across LANs

1. Client invokes a system call that copies the data from the client's virtual address space into a kernel buffer.

2. Host *A*'s protocol software creates a LAN1 frame and passes it to the LAN1 adapter.
   - appends internet header and a LAN1 frame header to data
   - internet header is addressed to host *B*
   - LAN1 frame header is addressed to the router
   - *encapsulation*: the payload of the internet packet is the data, and the payload of the LAN1 frame is the internet packet

3. The LAN1 adapter copies the frame to the network.

Transfer of Data Across LANs

4. The router's LAN1 adapter reads the frame and passes it to the protocol software.

5. The router fetches the destination internet address from the internet packet header. Then it strips off the old LAN1 frame header and prepends the new LAN2 header. Finally, the router passes the frame to the LAN2 adapter.
    - the destination address is used as an index into a routing table
    - the LAN2 frame header is addressed to host *B*

Transfer of Data Across LANs

6. The router's LAN2 adapter copies the frame to the network.

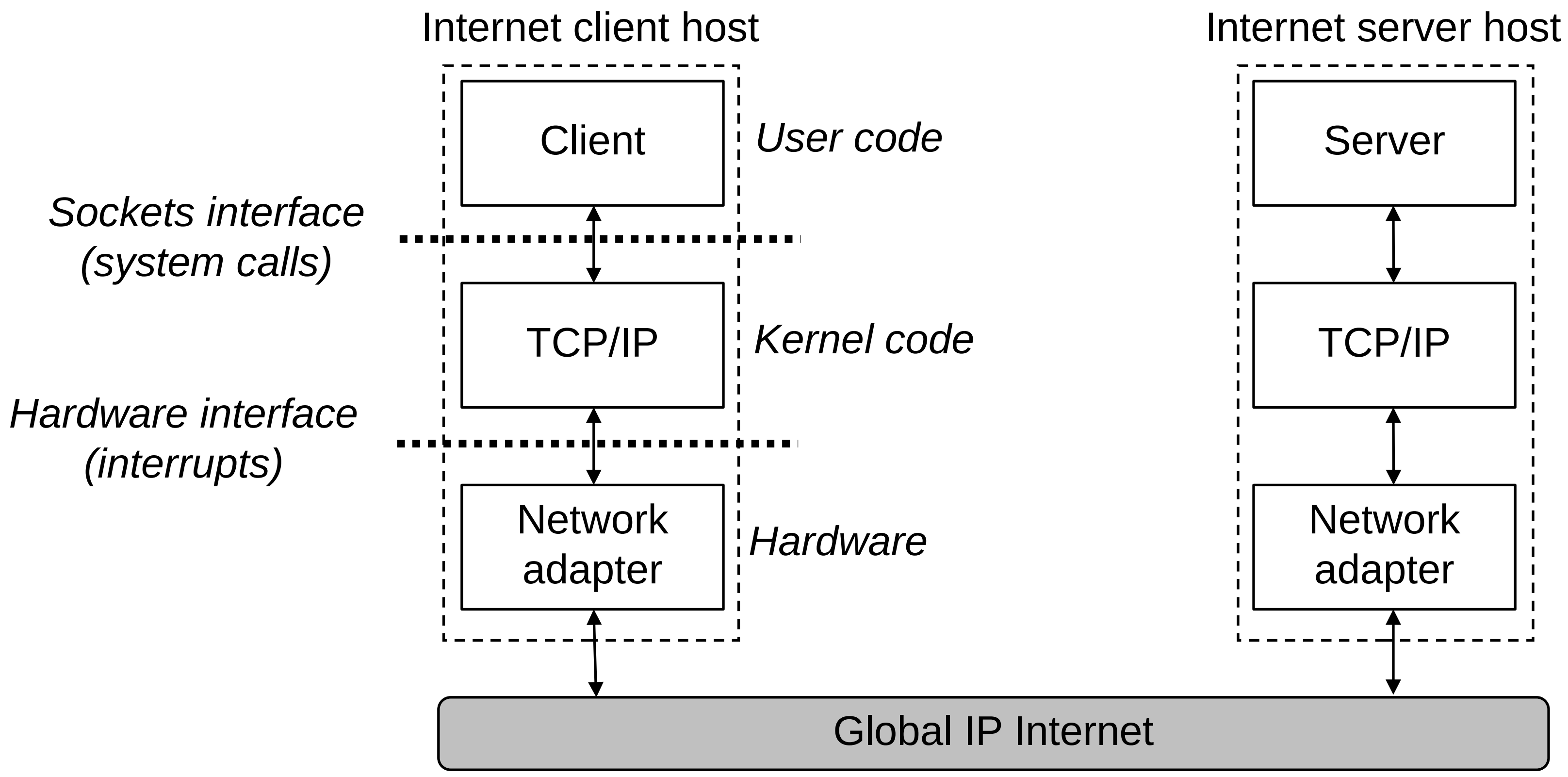
7. When the frame reaches host *B*, its adapter reads the frame and passes it to the protocol software.

8. Host *B*'s protocol software strips off the packet header and frame header. When the server invokes a system call that reads the data, the protocol software will copy the data into the server's virtual address space.

The Global IP Internet

The most famous and successful implementation of an internet.

- has existed in various forms since 1969
- since the early 80s, the organization of client-server apps has remained remarkably stable



Host Properties

A programmer can think of the Internet as a worldwide collection of hosts with these properties.

- The set of hosts is mapped to a set of 32-bit IP *addresses*.
- The set of IP addresses is mapped to a set of identifiers called *Internet domain names*.
- A process on one Internet host can communicate with a process on any other Internet host over a *connection*.

IP Addresses

```
/* Internet address structure */
struct in_addr {
  unsigned int s_addr;  /* the IP address */
};
```

- TCP/IP defines a uniform network byte ordering (big endian) for any integer data item carried across the network in a packet header.
- Unix provides functions from converting between network and host byte order.
  - `unsigned long int htonl(unsigned long int hostlong);`
  - `unsigned long int ntohl(unsigned long int netlong);`
  - versions for 16-bit ints: `htons` and `ntohs`

Dotted-Decimal Notation

- IP addresses are presented to humans with each byte represented by its decimal valued and separated from the other bytes by a period.

-
  ```
  lab1-12> hostname -i
  155.98.111.61
  ```
  address 0x9b626f3d

- Functions for converting between IP addresses and dotted-decimal strings.
  - `int inet_aton(const char* cp, struct in_addr* inp);`
  - `char* inet_ntoa(struct in_addr in);`

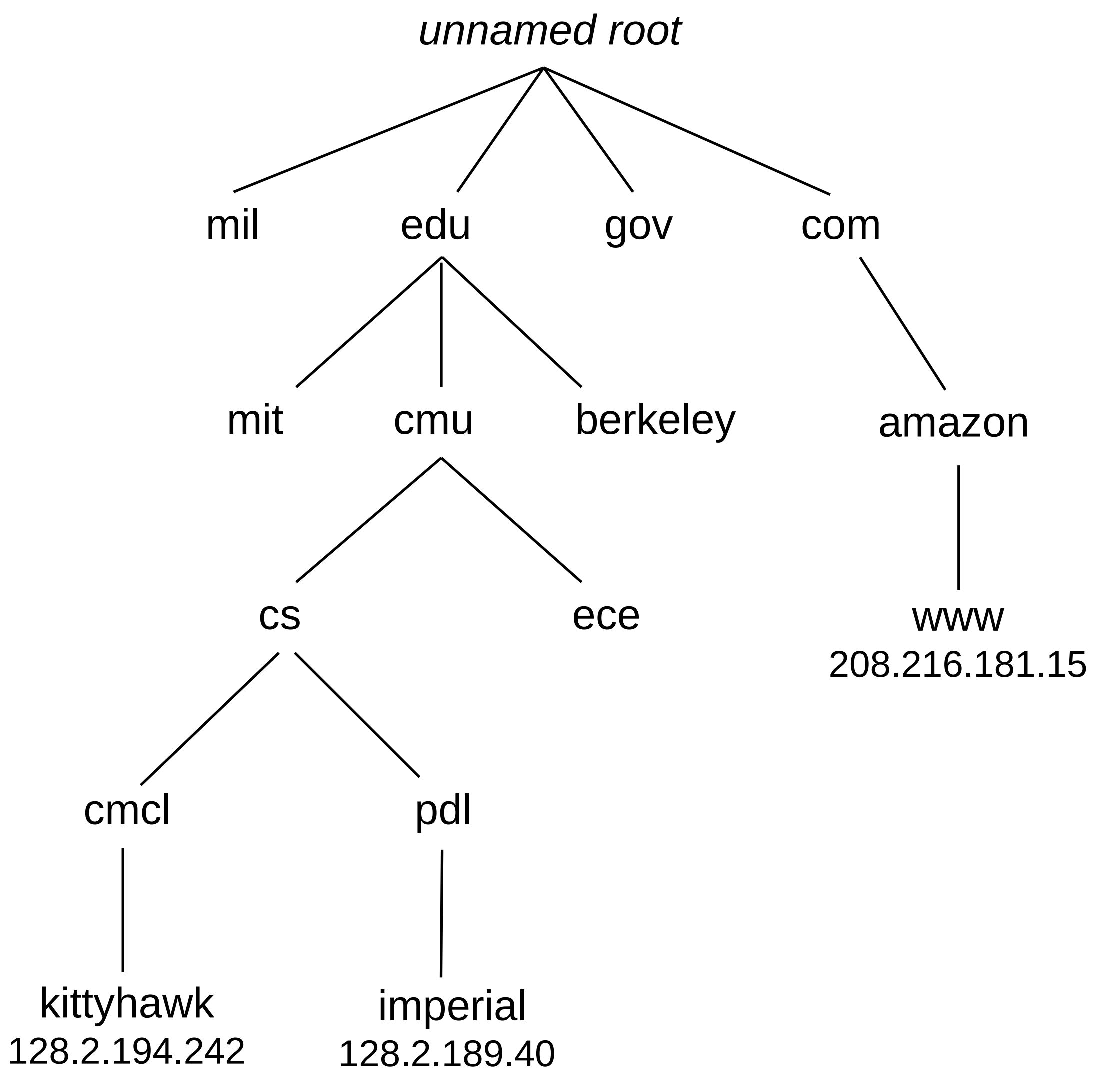
Exercise: IP Addresses

- Hex address `0x0` has dotted-decimal address `0.0.0.0`.
- Dotted-decimal address for `0xffffffff`?
- Dotted-decimal address for `0xef000001`?
- Hex address for `254.16.70.11`?

Internet Domain Names

Domain names are more human-friendly than IP addresses.

- sequence of words (letters, numbers, dashes) separated by periods



First-level domain names

Second-level domain names

Third-level domain names

DNS Mapping

- The Internet defines a mapping between the set of domain names and the set of IP addresses.
  - until 1988, maintained manually in a single text file
  - now in a world-wide database, Domain Naming System (DNS)

-
  ```
  /* DNS host entry structure */
  struct hostent {
    char* h_name;  /* host's official domain name */
    char** h_aliases; /* null-term array of names */
    int h_addrtype;   /* host addr type (AF_INET) */
    int h_length;   /* number of bytes in address */
    char** h_addr_list;  /* in_addr structs array */
  };
  ```
- Functions `gethostbyname` and `gethostbyaddr` retrieve host entries from DNS.

```c
/* hostinfo.c – reads a domain name or dotted-decimal address from the
   command line and displays the corresponding host entry */
#include "csapp.h"

int main(int argc, char* argv[]) {
  char** pp;
  struct in_addr addr;     /* IP address */
  struct hostent* hostp;   /* host entry */

  if(argc != 2)
    /* ERROR, QUIT */

  if(inet_aton(argv[1], &addr) != 0)  /* dotted-decimal address */
    hostp = Gethostbyaddr((const char*)&addr, sizeof(addr), AF_INET);
  else
    hostp = Gethostbyname(argv[1]);   /* domain name */

  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 = *((unsigned int *)*pp);
    printf("address: %s\n", inet_ntoa(addr));
  }

  exit(0);
}
```

Example: Running `hostinfo.c`

- Each Internet host has the locally-defined domain name `localhost`, which always maps to address `127.0.0.1`.

```
lab1-12> ./hostinfo localhost
official hostname: localhost.localdomain
alias: localhost
address: 127.0.0.1
```

- *One-to-one mapping (simplest case):*

```
lab1-12> ./hostinfo shell.cs.utah.edu
official hostname: shell.cs.utah.edu
address: 155.98.65.55
```

- *One domain name mapped to multiple IP addresses:*

```
lab1-12> ./hostinfo cs.utah.edu
official hostname: cs.utah.edu
address: 155.98.64.249
address: 155.98.64.250
```

Example: Running `hostinfo.c`

- *Multiple domain names mapped to one IP address:*

```
lab1-12> ./hostinfo www.coe.utah.edu
official hostname: alonzi.eng.utah.edu
alias: www.coe.utah.edu
address: 155.98.111.119
```

- *Multiple domain names mapped to multiple IP addresses*

  (most general case):

```
lab1-12> ./hostinfo www.google.com
official hostname: www.l.google.com
alias: www.google.com
address: 173.194.33.20
address: 173.194.33.16
address: 173.194.33.17
address: 173.194.33.18
address: 173.194.33.19
```

Internet Connections

- A connection is *point-to-point* in the sense that it connects a pair of processes.
- It is *full-duplex* because data can flow in both directions at the same time.
- It is *reliable* in the sense that the stream of bytes sent by the source process is eventually received by the destination process in the same order it was sent.
- A *socket* is an end point of a connection. A socket address consists of an Internet address and a 16-bit integer *port*, denoted `address:port`.

Ports

- The port in the client's socket address (assigned by automatically by the kernel), aka an *ephemeral port*.
- The port in the server's socket address is typically some *well-known port* associated with the service.
  - web servers typically use port 80, email servers port 25
- A connection is uniquely identified by the socket address of its two end-points, a *socket pair*.
  - identified by `(cliaddr:cliport, servaddr:servport)`
  - *example*: `(128.2.194.242:51213, 128.216.181.15:80)`

The Sockets Interface

- The sockets interface is a set of functions used with Unix I/O functions to build network applications.
- To a Unix program, a socket is simply an open file with a corresponding descriptor.
- Functions for opening and closing socket descriptors are provided (`socket`, `connect`, `listen`, `accept`).
- Clients and servers communicate by reading and writing to socket descriptors.
- (Take a closer look at section 11.4 on your own.)

Web Servers

- Web servers and clients (browsers) communicate using the HTTP protocol.
- When a browser requests static content from the server, a file is fetched from the server's disk and returned to the client.
- For a request of dynamic content from the server, a program runs on the server (as a child process) and its output is returned to the client.
- (Take a closer look at sections 11.5-11.6 on your own.)