Announcements

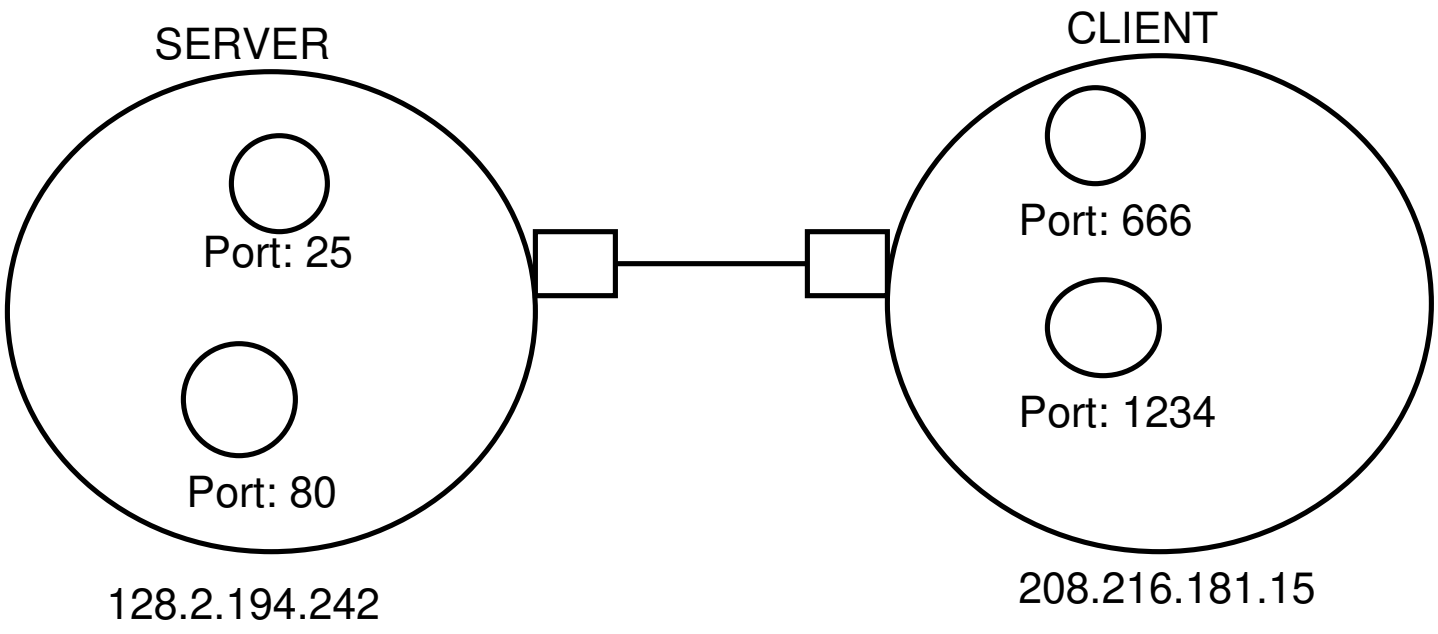
- Program #4 on the web
- Reading
  - Notes (Today) also see the posted Tutorial on the web page
  - Notes (Thursday)
- Project 3
  - code coverage – see the grades.cs.umd.edu
    - did not count the print routines
    - 90% (of other code) and above got 10 points

Internet Abstractions

- Domain Name System
  - character strings to represent names
  - mashie.cs.umd.edu
- IP Address
  - 32 bit value that describes a network card
  - Example address: 128.8.129.30
- Port (TCP)
  - 16 bit endpoint for communication on a machine
  - Well known ports:
    - used to connect to a service (e.g. 80 is HTTP)
  - Machine assigned:
    - used for incoming connections

Networking

- Unique identification of communication
  - 4 tuple of <[clientIp,clientPort], [serverIp, serverPort]>
  - example: <208.216.181.15:1234,128.2.194.242:80 >



Network API Types

- struct in_addr
  - stores IP address
  - network byte order
    - bigendian byte order
- struct hostent
  - represents a host
  - struct hostent gethostbyname(char *name);
    - returns the information about the passed host name
- struct sockaddr
  - end point for communication
  - can be used for IP or other protocols too
- struct sockaddr_in
  - defines an endpoint on an IP network
  - contains fields for src/dest IP addr and src/dest port
- file descriptor
  - used to refer to an active socket

```
struct in_addr{
   unsigned int s_addr;
};
```

```
struct hostent{
   char *h_name;
   char **h_aliases;
   int h_addrtype;
   int h_length;
   char **h_addr_list
};
```

```
struct sockaddr{
   unsigned short sa_family;
   char sa_data[14];
};
```

```
struct sockaddr_in{
   unsigned short sin_family;
   unsigned short sin_port;
   struct in_addr sin_addr
   unsigned char sin_zero[8];
};
```

Socket API Call (Client & Server)

- int socket(int domain, int type, int protocol);
  - create a new socket
  - typical usage:
    - fd = socket(AF_INET, SOCK_STREAM, 0);
    - three parameters: domain, type and protocol
    - creates a TCP socket
    - trying to read or write before connection would be an error

Socket API Call (Client)

- int connect(int sockfd, struct sockadrr*server, int len);
  - connect to the remote host/port described by sockaddr
  - three parameters:
    - sockfd must be a previously created socket (the fd)
    - can either be a sockaddr or a sockaddr_in (but these are different sizes so we also need the length)
    - len is the size of struct sockaddr

Network Code Example (Client)

```
int openserverfd(char *host, unsigned short port) {
    int clientfd, port;
    struct sockaddr_in serveraddr;
    struct hostent *hp;

    clientfd = socket(AF_INET, SOCK_STREAM, 0);
    if (clientfd <0) exit(-1);
    hp = gethostbyname(host);
    if (!hp) exit(-1);
    memset((void *) &serveraddr, '\0', sizeof(struct sockaddr));
    serveraddr.sin_family = AF_INET;
    memcpy(&serveraddr.sin_addr.s_addr, hp->h_addr, hp->h_length);
    serveraddr.sin_port = htons(port);

    return connect(clientfd, &serveraddr, sizeof(serveraddr));
}
```

Socket API Calls (Server)

- int bind(int sockfd, struct sockaddr *addr, int len);
  - bind a local port to the passed socket
  - port is specificed in the passed sockaddr
- int listen(int sockfd, int backlog);
  - inform OS to let connections happen on the passed sock
  - allow at most backlog pending connections
- int accept(int sockfd, struct sockaddr *addr, int len);
  - blocks until a connection arrives on passed sockfd
  - returns a new file descriptor for the active connection
  - fill out addr with information about established connection
  - again len because the sockaddr could be other detailed types

Network Code Example (Server)

```
int readyServer(int port) {
    int listenfd, optval=1;
    sturct sockaddr_in serveraddr;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenfd <0) return -1;
    setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR);

    memset((void *) &serveraddr, '\0', sizeof(struct sockaddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serveraddr.sin_port = htons((unsigned short) port);
    ret = bind(listenfd, &serveraddr, sizeof(serveraddr);
    if (ret < 0) return -1;

    return listen(listenfd, 10);
}
```

Network Code Example (Server), Cont.

```
int listenfd;
struct sockaddr_in clientaddr;
struct hostent *hp;
char *haddrp;
 ….
listenfd = readyServer(MY_PORT);

while (1) {
    connfd = accept(listenfd, &clientaddr, sizeof(clientaddr));
    if (connfd < 0) exit(-1);
    hp = gethostbyaddr(&clientaddr.sin_addr.s_addr,
                sizeof(clientaddr.sin_addr.s_addr), AF_INET);
    haddrp = inet_ntoa(clientaddr.sin_addr);
    printf("connection from %s (ip %s)\n", hp->h_name, haddrp);
}
```

Network API - Using and Open Socket (client)

- Sockets can be read/written just like other fd

```
FILE *ofp, *ifp;
int serverfd;

serverfd = openserverfd("www.dyninst.org", 80) ;
ofp = fdopen(serverfd, "w");
if (!ofp) exit (-1);
ifp = fdopen(serverfd, "r");
if (!ifp) exit(-1);
fprintf(ofp, "GET index.html\n");
while (fgets(line, sizeof(line), ifp)) {
        printf(line);
}
```

Other Socket API Details

- When done with a socket
  - use close to cleanup