# CS3516 (B10) HELP Session 2

*Presented by Lei Cao*

# Outline

- Project 4 Overview
- Unix Network Programming
  - UDP/TCP Client
  - Server
- Communication with the net oracle
- Additional suggestions / tips

# CS3516 Project4

- Your programs should compile and work on ccc.wpi.edu computers, which are running Linux.
- Net oracle(named oracle) is running on cccWORK4.wpi.edu only.
- Programs should be written in C or C++
- If your program is developed on another platform or machine, you should test the software on ccc before turning in the assignment.
- Make sure you have the correct #include files in your program.

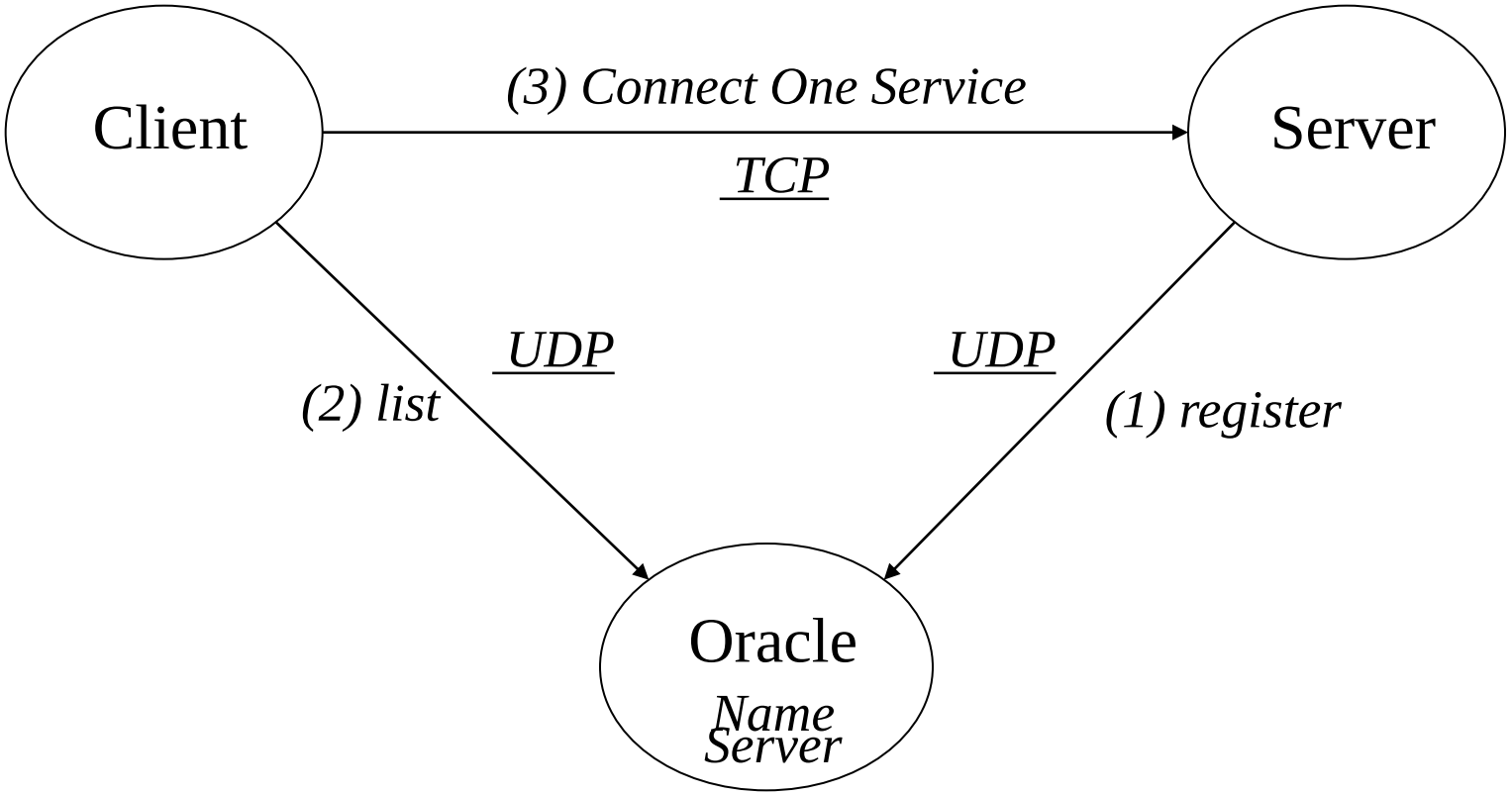
# What is the Net Oracle?

- Net Oracle is a name server that maps service names into their transport level address.
- Net Oracle allows you to register the service you have developed.
- You register services by sending a message to the oracle containing the server name together with the transport address
- Given the service name, the client can request the service address or remove the service you have registered.

# Project 4 Steps

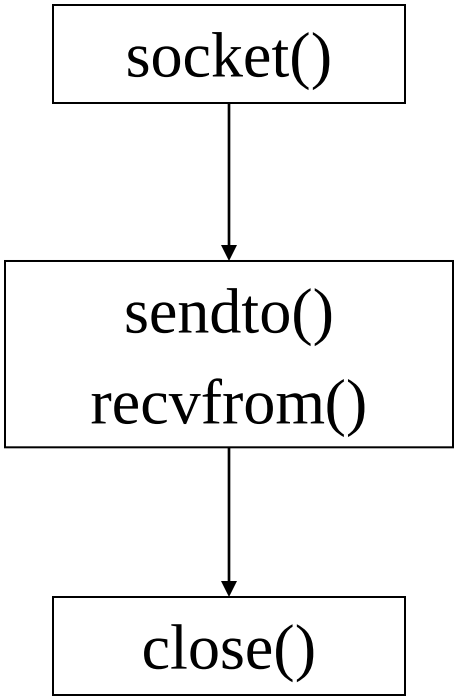
- Client:
  1. Wait on user's commands.
  2. List all services registered on net oracle(using list command).
  1. Connect to service using the transport address returned from net oracle( by connect service [uid] command).
  2. Terminate connection with the server when it receives an end-of-file.
  3. Terminate the client program(by quit command)
- Server:
  1. Register services with the net oracle.
  2. Wait for connections and provide name service to the clients.

# Project 4 Communication Model



# UDP Transmission (Client)

- Connectionless
  - Specify transport address every time you send/recv data
- Unreliable Protocol
  - Data lost, bit errors



# Example: UDP Client

```
struct hostent *hp;              /* /usr/include/netdb.h */
struct sockaddr_in server;       /* /usr/include/netinet/in.h */
int sd, lserver = sizeof( server );

/* prepare a socket */
if ( (sd = socket( AF_INET, SOCK_DGRAM, 0 )) < 0 ) {
        perror( strerror(errno) );
        exit(-1);
}
```

# Example: UDP Client (Continued)

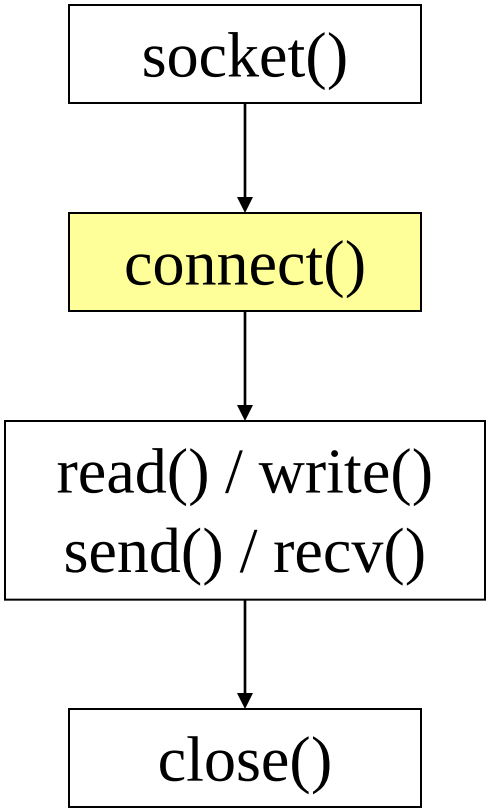
```
/* prepare server address */
bzero( (char*)&server, sizeof(server) );
server.sin_family = AF_INET;
server.sin_port = htons( SERVER_PORT );  //endian convert
if ( (hp = gethostbyname(SERVER_NAME)) == NULL) {
        perror( strerror(errno) );
        exit(-1);
}
bcopy( hp->h_addr, (char*)&server.sin_addr, hp->h_length);
```

# Example: UDP Client (Continued)

```
/* prepare your message */
…
/* send/receive data */
sendto( sd, sBuf, data_size, 0, (struct sockaddr*)&server,
        lserver );
recvfrom( sd, rBuf, MAXLEN, 0, (struct sockaddr*)&server,
          &lserver );
/* close socket */
close( sd );
```

# TCP Connection (Client)

- Connection Oriented
  - Specify transport address once at connection
- Use File Operations
  - read() / write()
- Reliable Protocol



# Example: TCP Client

```
int sd;
struct hostent *hp;              /* /usr/include/netdb.h */
struct sockaddr_in server;       /* /usr/include/netinet/in.h */

/* prepare a socket */
if ( (sd = socket( AF_INET, SOCK_STREAM, 0 )) < 0 ) {
        perror( strerror(errno) );
        exit(-1);
}
```

# Example: TCP Client (Continued)

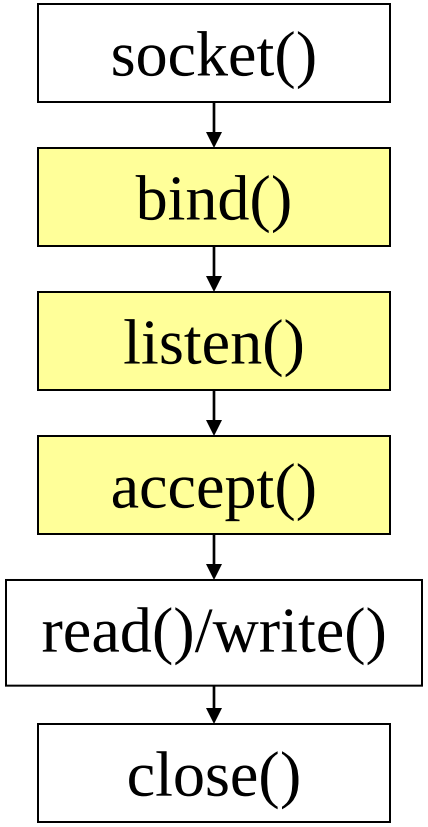
```
/* prepare server address */
bzero( (char*)&server, sizeof(server) );
server.sin_family = AF_INET;
server.sin_port = htons( SERVER_PORT );
if ( (hp = gethostbyname(SERVER_NAME)) == NULL) {
        perror( strerror(errno) );
        exit(-1);
}
bcopy( hp->h_addr, (char*)&server.sin_addr, hp->h_length);
```

# Example: TCP Client (Continued)

```
/* connect to the server */
if (connect( sd, (struct sockaddr*) &server, sizeof(server) ) < 0 ) {
        perror( strerror(errno) );
        exit(-1);
}
/* send/receive data */
while (1) {
        read/write();
}
/* close socket */
close( sd );
```

# TCP Connection (Server)

- Bind transport address to socket
- Listen to the socket
- Accept connection on a new socket



# Example: TCP Server

```
int sd, nsd;
struct sockaddr_in server;    /* /usr/include/netinet/in.h */

sd = socket( AF_INET, SOCK_STREAM, 0 );

bzero( (char*)&server, sizeof(server) );
server.sin_family = AF_INET;
server.sin_port = htons( YOUR_SERVER_PORT );
server.sin_addr.s_addr = htonl( INADDR_ANY );
```

# Example: TCP Server (Continued)

```
bind( sd, (struct sockaddr*) &server, sizeof(server) );

listen( sd, backlog );

while (1) {
   nsd = accept( sd, (struct sockaddr *) &client, sizeof(client) );
   read()/write();
   close( nsd );
}

close( sd );
```

# Your Server is also a Client

- Server has to register service with the net oracle via UDP.

# Oracle Commands

- enum cmd {
  cmdErr, /* An error occurred. See sbDesc for details */
  cmdGet, /* Get the address of a service */
  cmdAckGet, /* ACK for cmdGet message */
  cmdEnd, /* Last response to a cmdGet message */
  cmdPut, /* Register a new service */
  cmdAckPut, /* ACK for cmdPut message */
  cmdClr, /* Unregister a service */
  cmdAckClr /* ACK for cmdClr message */
- };

# Oracle Commands (Request)

- **Find a service:**

  serv.ver = verCur;
  serv.cmd = cmdGet;
  serv.uid = ?;
  serv.sbServ = ?;

- **Register a service:**

  serv.ver = verCur;
  serv.cmd = cmdPut;
  serv.uid = ?;
  serv.sbServ = ?;
  serv.sbDesc = ?;
  serv.sa(sockaddr_in) = ?
  Serv.ti = ?

- **Clear a service:**

  serv.ver = verCur;
  serv.cmd = cmdClr;
  serv.uid = ?;
  serv.sbServ = ?;

# Oracle Command (Response)

- The same structure as with Request
- Clear: serv.cmd = cmdAckClr;
- Register: serv.cmd = cmdAckPut;
- Get : serv.cmd = cmdAckGet;
  - Return two or more messages.
  - The last one: serv.cmd = cmdEnd;

# Oracle Communication Example

```
int sd;
struct sockaddr_in sa;        // you can use gethostbyname() and
                              // getservbyname() to get sa in your project.
struct om sendMsg, recvMsg;
size_t lom = sizeof(struct om);

sendMsg.ver = verCur;
sendMsg.cmd = cmdGet;

sendto( sd, (void *)&sendMsg, lom, 0, (struct sockaddr *) &sa, lsa );
recvfrom( sd, (void *)&recvMsg, lom, 0, (struct sockaddr *) &sa, &lsa );

// you can also use connect()/send()/recv() for UDP connection, for more
// information -- use “man connect”, “man send” and “man recv”
```

# Some Useful System Calls

- Gethostbyname: map hostname to IP addr

  struct hostent ***gethostbyname**( char *name )

- Getservbyname: look up service name given

  struct servent ***getservbyname**( const char *servname,
  const char *protocol )

- Gethostname: get own hostname

  int **gethostname**( char *name, size_t len )

- Getsockname: map sd to socket addr

  int **getsockname**( int sd, struct sockaddr *sa, size_t *lsa )

# Getservbyname()

- Get the address of the service
- The services should be registered in ./etc/services
- The registered name of net oracle is “oracle”
- struct servent ***getservbyname**( const char *servname,
  const char *protocol )

UDP

# Select() system call

- Your client needs to take input from both the network *and* stdin to close TCP connection or to support your interactive service.
- How to tell when to do which one?
- Use the *select()* function!
- Notes:

  #include <sys/select.h>

  man select