CS4514 (C04) HELP Session 1

Introduction to Network Programming (v1.3)

Speaker: Frank Posluszny

WPI

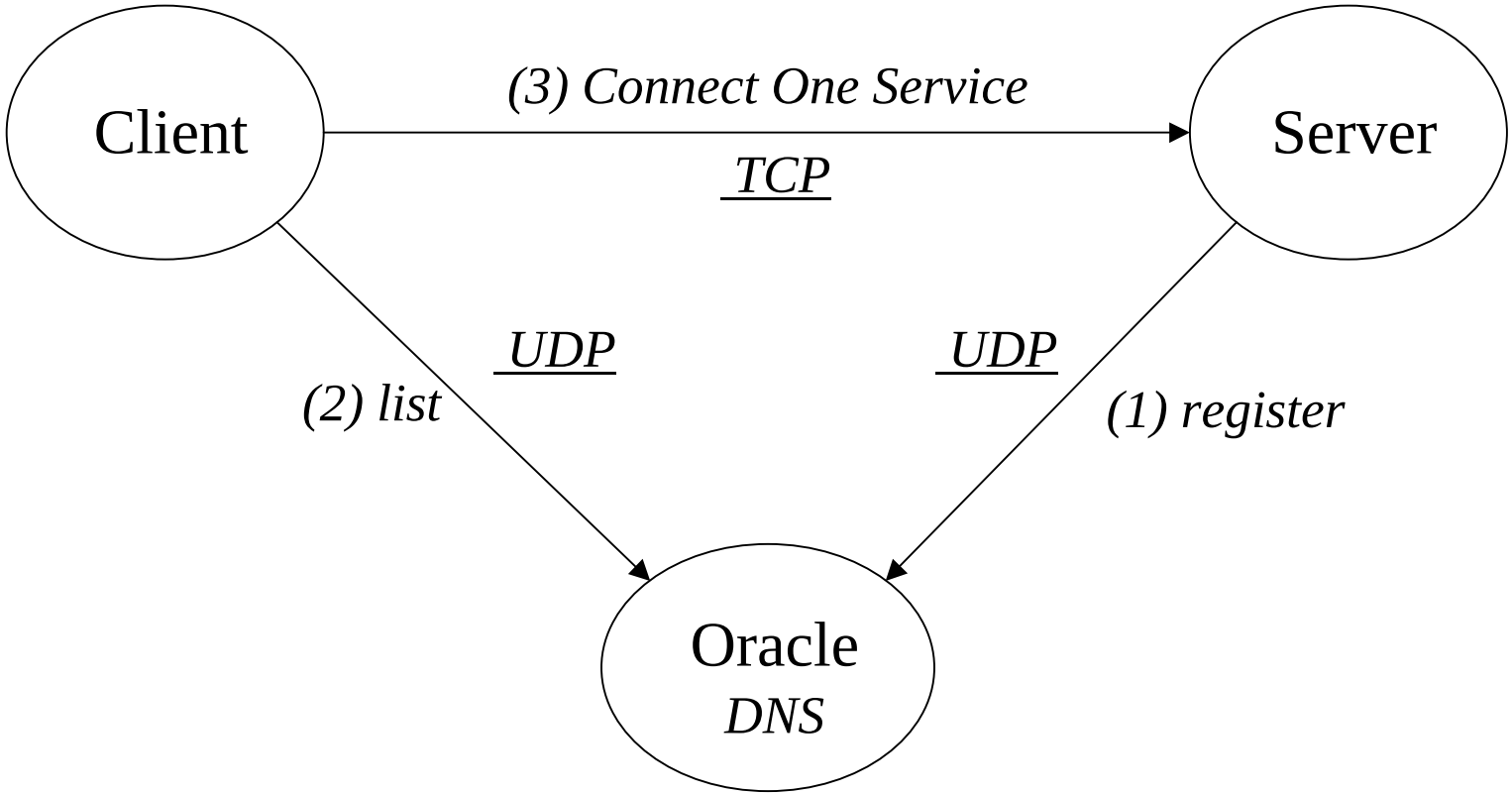
Outline

- Project 1 Overview
- Unix Network Programming
  - Client
  - Server
  - Communication with netoracle
- Project Turnin
- How to find help
- Additional suggestions / tips

CS4514 Project1

- Your programs should compile and work on ccc.wpi.edu computers, which are running Linux.
- Netoracle is running on ccc3.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 in your program.
- We have registered a couple of services in oracle that can be used to debug your client.

Project 1 Communication Model

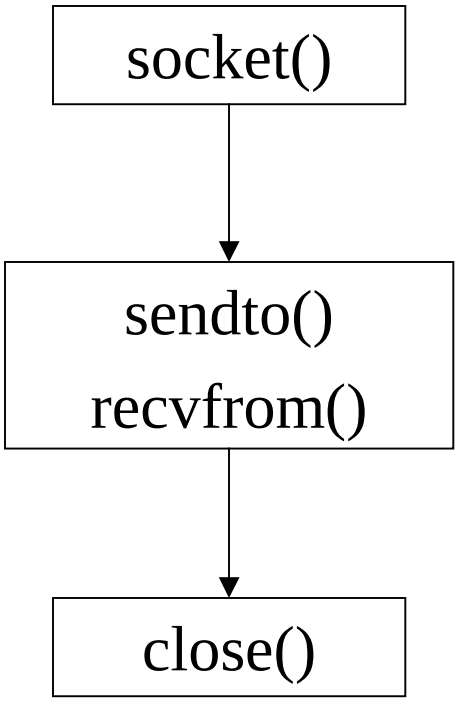


Project 1 missions (in handout)

- Client:
  1. Wait on user's commands.
  2. List all services registered on netoracle.
  3. Connect to service using the transport address returned from netoracle.
- Server:
  1. Register service to netoracle.
  2. Wait for connections and provide service.

UDP Transmission (Client)

- Stevens (Page 212)
  - Connectionless
    - Specify transport address every time you send/recv data
  - Unreliable Protocol
    - Data lost, bit errors
  - Stevens (Page 216)
    - Simple UDP echo client
    - Lenon page78 (robust)



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)

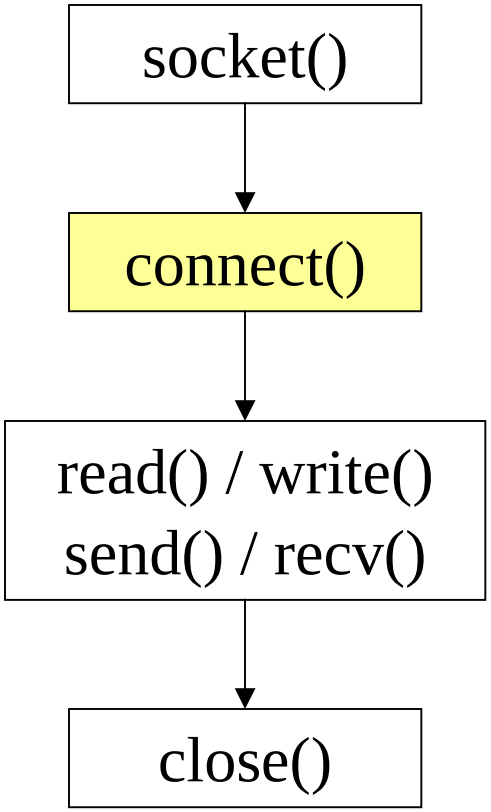
```
/* prepare server address */
bzero( (char*)&server, sizeof(server) );
server.sin_family = AF_INET;
server.sin_port = htons( SERVER_PORT );  //endian convert
if ( (hp = gethostbyname(SERVER_NAME)) == NULL) {
        perror( strerror(errno) );
        exit(-1);
}
bcopy( hp->h_addr, (char*)&server.sin_addr,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

- Stevens (Page 86)
  - 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)

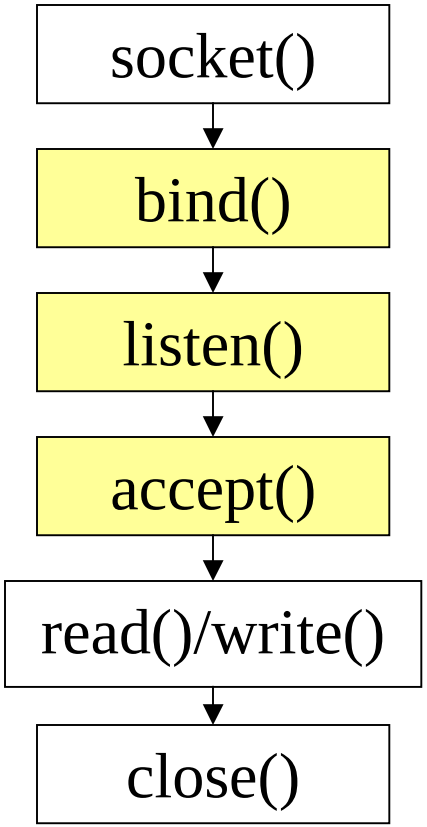
```
/* connect to the server */
if (connect( sd, (struct sockaddr*) &server, sizeof(server) ) < 0 ) {
        perror( strerror(errno) );
        exit(-1);
}
/* send/receive data */
while (1) {
        read/write();
}
/* close socket */
close( sd );
```

When to read()

- Your client needs to take input from both the network *and* stdin
- How to tell when to do which one?
- Use the *select()* function!

TCP Connection (Server)

- Stevens (Page 86)
  - 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 );
```

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 (om struct)

- **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 = ?

- **Clear a service:**

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

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”
```

Turnin Your Files

- Turnin Command
  - Create a directory to hold your files:
    mkdir proj1
  - Copy all files in the current directory to your new directory:
    *cp * proj1*
  - Create a single, compressed archive file containing all of the files in your new directory:
    tar -czf proj1.tgz proj1
  - Submit the archive file:
    /cs/bin/turnin submit cs4514 proj1 proj1.tgz

Turnin Your Files (Continued)

- Files should include
  - All source code (including a Makefile)
  - A documentation file (include your compile command if you don't offer a Makefile)
  - A result *script* showing the running result
  - Any custom include files that you used, including oracle.h if you have not used

    #include "/cs/cs4514/pub/lib/oracle.h"

HELP

- Bring printouts to office hours.
- Email TA's (cs4514-ta@cs.wpi.edu) with questions.
- You CAN email a specific TA or SA, but do not expect immediate results, better to use the TA mailing list.
- We do have a class mailing list that could be used as a last resort.

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)

UNIX Programming

- Some functions that you may need:
  - bind
  - listen
  - accept
  - select
  - sendto/send
  - recvfrom/recv
  - gethostbyname
  - getservbyname
  - gethostname
  - getsockname
  - fork
  - strlen, strtok

Other resources

- Use man pages for help on a particular command or function (Give the section number).

  > man sendto

  > man 2 bind //show bind(2)

- Internet: Beej's Guide to Network Programming

  http://www.ecst.csuchico.edu/~beej/guide/net/

UNIX Debugging

- GDB -- GNU Project Debugger
- Compile program with –g flag
  - g++ -g –o program program.cc
  - gcc –g –o program program.c
- gdb program {core}
  - set args (command arguments)
  - run, where, list, step, break
  - continue inspect, help, quit
- Can examine specific data in program

UNIX Debugging (Continued)

- There are many more options use “help” to learn more.
- Also look at “man gdb”.
- This will be useful to find out where a program crashes or seg faults.
- You can set breakpoints to stop at specific line or function.
- You can set specific data values in program.