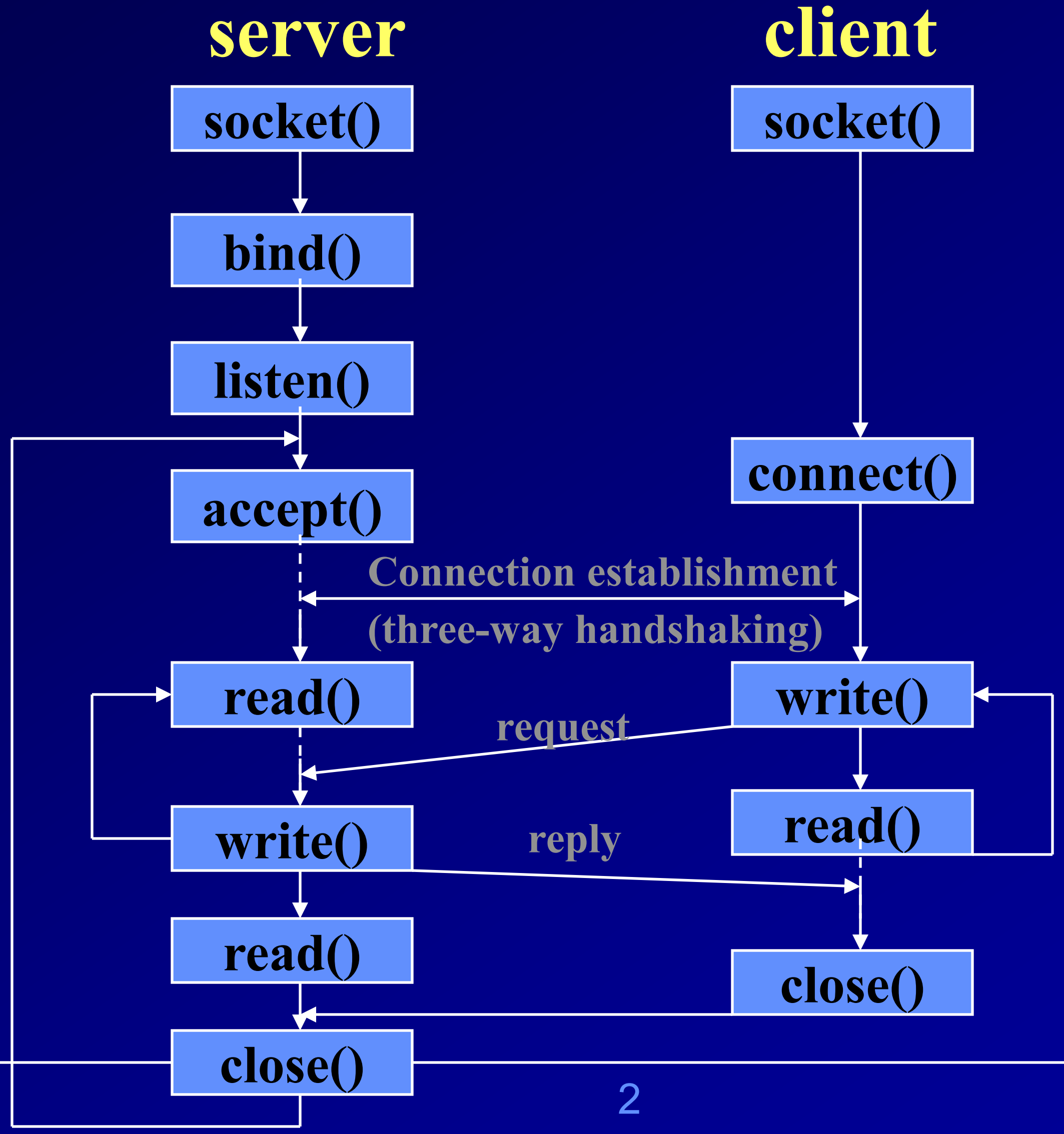
Socket Programming

CSEE, Handong Univ.

Jong-won Lee

ljw@handong.edu

TCP Client/Server Interaction



TCP: echo_server.c

```
#define BUFSIZE 1024

void error_handling(char *message);

int main(int argc, char **argv)
{
   int serv_sock;
   int clnt_sock;
   char message[BUFSIZE];
   int str_len;

   struct sockaddr_in serv_addr;
   struct sockaddr_in clnt_addr;
   int clnt_addr_size;

   if(argc!=2){
       printf("Usage : %s <port>\n", argv[0]);
       exit(1);
    }
```

TCP: echo_server.c

```
serv_sock = socket(PF_INET, SOCK_STREAM, 0);
if(serv_sock == -1)
    error_handling("socket() error");

memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(atoi(argv[1]));

if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error");

if(listen(serv_sock, 5) == -1)
    error_handling("listen() error");

clnt_addr_size = sizeof(clnt_addr);
clnt_sock = accept(serv_sock, struct sockaddr*)&clnt_addr,&clnt_addr_size);
if(clnt_sock == -1)
    error_handling("accept() error");
```

TCP: echo_server.c

```
    /* receive data & send*/
   while( (str_len = read(clnt_sock,message, BUFSIZE)) != 0){
        write(clnt_sock, message, str_len);
        write(1, message, str_len);
   }

   close(clnt_sock);
   close(serv_sock);
   return 0;
}

void error_handling(char *message)
{
   fputs(message, stderr);
   fputc('\n', stderr);
   exit(1);
}
```

TCP: echo_client.c

```
#define BUFSIZE 1024

void error_handling(char *message);

int main(int argc, char **argv)
{
   int sock;
   char message[BUFSIZE];
   int str_len;
   struct sockaddr_in serv_addr;

   if(argc != 3){
         printf("Usage : %s <IP> <port>\n", argv[0]);
         exit(1);
   }

   sock = socket(PF_INET, SOCK_STREAM, 0);
   if(sock == -1)
         error_handling("socket() error");
```

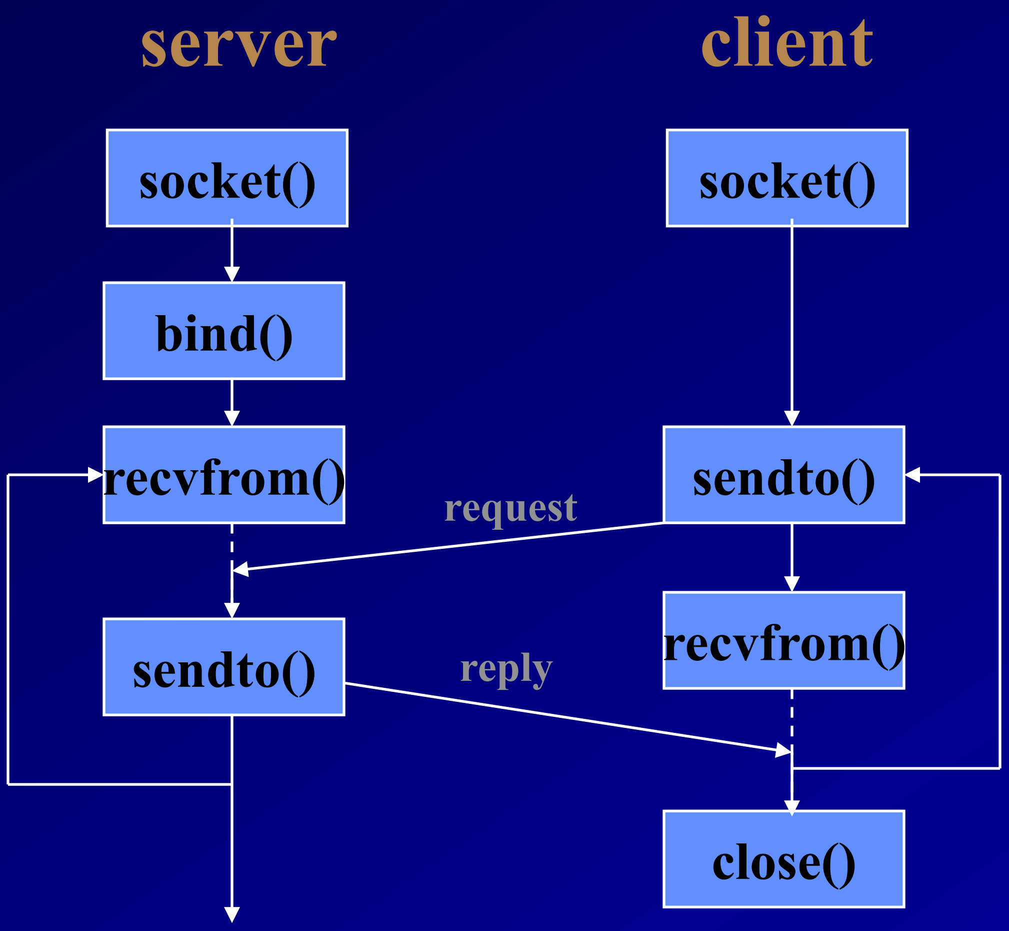
TCP: echo_client.c

```
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
serv_addr.sin_port=htons(atoi(argv[2]));

if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("connect() error!");

while(1) {
    /* message input & send */
    fputs("Enter a message to send(q to quit) : ", stdout);
    fgets(message, BUFSIZE, stdin);

    if(!strcmp(message,"q\n"))     break;
    write(sock, message, strlen(message));

    /* print messages received */
    str_len = read(sock, message, BUFSIZE-1);
    message[str_len]=0;
    printf("A message from the server : %s \n", message);
```

TCP: echo_client.c

```
   close(sock);
   return 0;
}

void error_handling(char *message)
{
   fputs(message, stderr);
   fputc('\n', stderr);
   exit(1);
}
```

UDP Client/Server Interaction



UDP: becho_server.c

```
#define BUFSIZE 30
void error_handling(char *message);

int main(int argc, char **argv){
    int serv_sock;
    char message[BUFSIZE];
    int str_len, num=0;

    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    int clnt_addr_size;

    if(argc != 2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
     }

     serv_sock = socket(PF_INET, SOCK_DGRAM, 0);
     if(serv_sock == -1)
         error_handling("UDP socket error");
```

UDP: becho_server.c

```
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(atoi(argv[1]));

if (bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) == -1)
    error_handling("bind() error");

    sleep(5);
    while(1) {
         clnt_addr_size = sizeof(clnt_addr);
         sleep(1);
         str_len = recvfrom (serv_sock, message, BUFSIZE, 0,
                                    (struct sockaddr*)&clnt_addr, &clnt_addr_size);
         printf("The number of messages: %d \n", num++);
        sendto (serv_sock, message, str_len, 0, (struct sockaddr*)&clnt_addr, sizeof(clnt_addr));
    }

    return 0;
}
```

UDP: becho_client.c

```
#define BUFSIZE 30
void error_handling(char *message);

int main(int argc, char **argv){
    int sock;
    char message[BUFSIZE];
    int str_len, addr_size, i;

    char MSG1[] = "Good ";
    char MSG2[] = "Evening ";
    char MSG3[] = "Everybody!";

    struct sockaddr_in serv_addr;
    struct sockaddr_in from_addr;

    if(argc!=3){
        printf("Usage : %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if(sock == -1)  error_handling("UDP socket error");
```

UDP: becho_client.c

```
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
    serv_addr.sin_port=htons(atoi(argv[2]));
    sendto(sock, MSG1, strlen(MSG1), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
    sendto(sock, MSG2, strlen(MSG2), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
    sendto(sock, MSG3, strlen(MSG3), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

    for(i=0; i<3; i++)
    {
        addr_size = sizeof(from_addr);
        str_len = recvfrom(sock, message, BUFSIZE, 0, (struct sockaddr*)&from_addr, &addr_size);
         message[str_len]='\0';
         printf("The message %d from the server: %s \n", i, message);
    }
    close(sock);
    return 0;
}
```