Assignment 2
Of
Network & Distributed System Lab (CS2051)
Masters of Technology in Computer Science And Engineering

submitted to
Dr Sujoy Saha
Assistant Professor
&
Dr Suvrojit Das
Associate Professor
Dept. of CSE

National Institute of Technology, Durgapur

submitted by
Arghya Bandyopadhyay
RollNo. 20CS4103

19 June 2021

1. Write TCP and UDP Chat Program.

Answer.

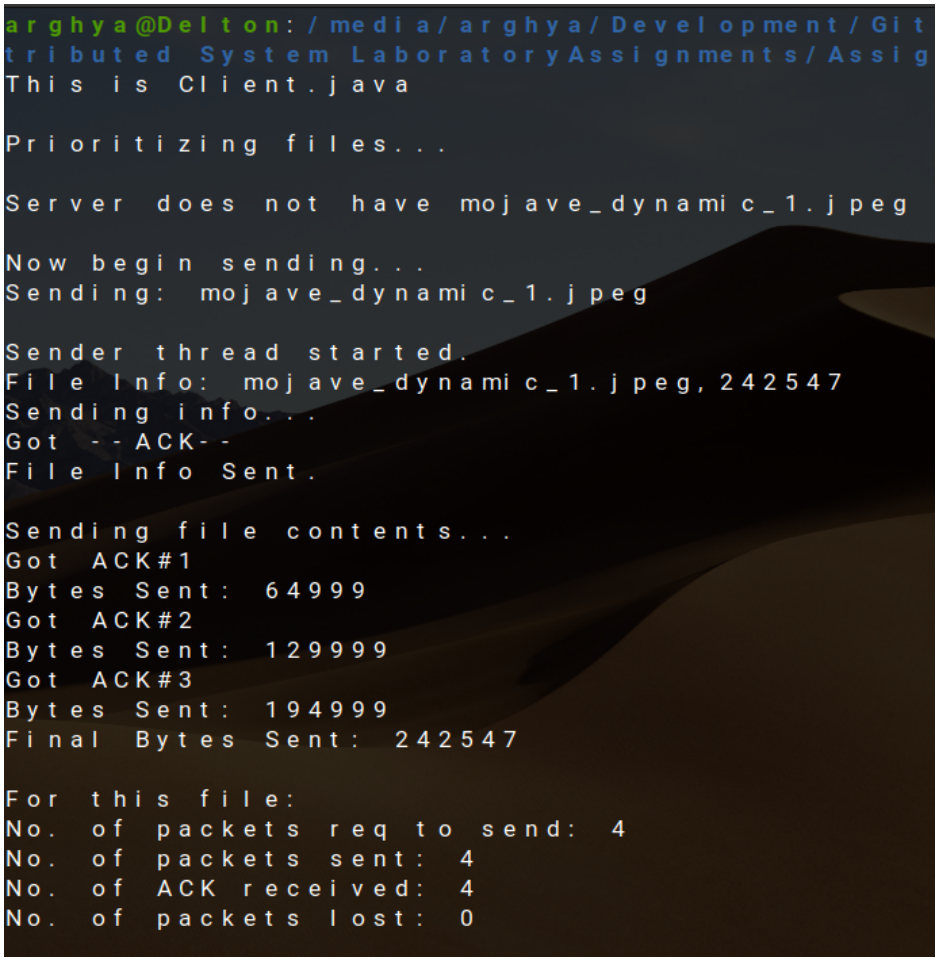
```
// TCP Chat Server (Concurrent)

//#include<iostream.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>                          //For sockaddr_in
#include<unistd.h>
//using namespace std;
#include<stdio.h>

//#include"Common.h"
#include <stdbool.h>


int TCP_ChatServer(int client_desc)
{
  const int BUFFER_SIZE = 4096;
  char msg[BUFFER_SIZE];
  int  msg_len = 0;

  while((msg_len = read(client_desc, msg, BUFFER_SIZE)) != 0)
  {
    //cout<<"[User "<<client_desc<<"] Msg Recieved :";
    printf("%s %d %s","[User",client_desc,"] Msg Received :");
    fflush(stdout);

    write(fileno(stdout), msg, msg_len);

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;

    //cout<<"[User "<<client_desc<<"] Enter Responce :";
    printf("%s %d %s","[User",client_desc,"] Enter Response :");

    fflush(stdout);


    msg_len = read(fileno(stdin), msg, BUFFER_SIZE);

    write(client_desc, msg, msg_len);

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;
  }

  // EOF
  return 0;
}

```

```
int main()
{
    //Create Socket
  int  server_desc = socket(AF_INET,SOCK_STREAM,0);


  //CheckError(server_desc, "socket()");

    //Create and Fill Address Structure for this Server
  struct  sockaddr_in server_addr;
  server_addr.sin_family      = AF_INET;    //Address Family (AF_INET, AF_INET6, AF_LOCAL, ...)
    server_addr.sin_addr.s_addr = INADDR_ANY; //Internet Address (INADDR_ANY-> Accept connection at any IP Address)
    server_addr.sin_port        = htons(9000);//Port Number (htons -> h.HOST t.TO n.NETWORK s.SHORT , Ensures proper byte ordering)

  //Bind Socket Descriptor and Address Structure together
  int result = bind(server_desc,  (struct sockaddr*) &server_addr, sizeof(server_addr));
  //CheckError(result, "bind()");

  //Start Listioning (Tell kernel to accept connections directed towards this socket) (Puts socket into passive mode)
  listen(server_desc,4);




  //Server Loop
  bool RunServer = true;
  while(RunServer)
  {
    //Accept a Connection (Puts process in sleep mode if Connection Queue is Empty)
    int client_desc;
    client_desc = accept(server_desc,NULL,NULL);        //Listening Socket

    //CheckError(client_desc, "accept()");

    //Create Child Process to handle connection
    int pid = fork();

    if(pid > 0)                             //Parent Process
    {
      //Close Client Socket
      close(client_desc);
      continue;
    }
    else
    if(pid == 0)                            //Child Process
    {
      //Close Listening Socket
      close(server_desc);

      TCP_ChatServer(client_desc);

      //Close Connection
      close(client_desc);
```

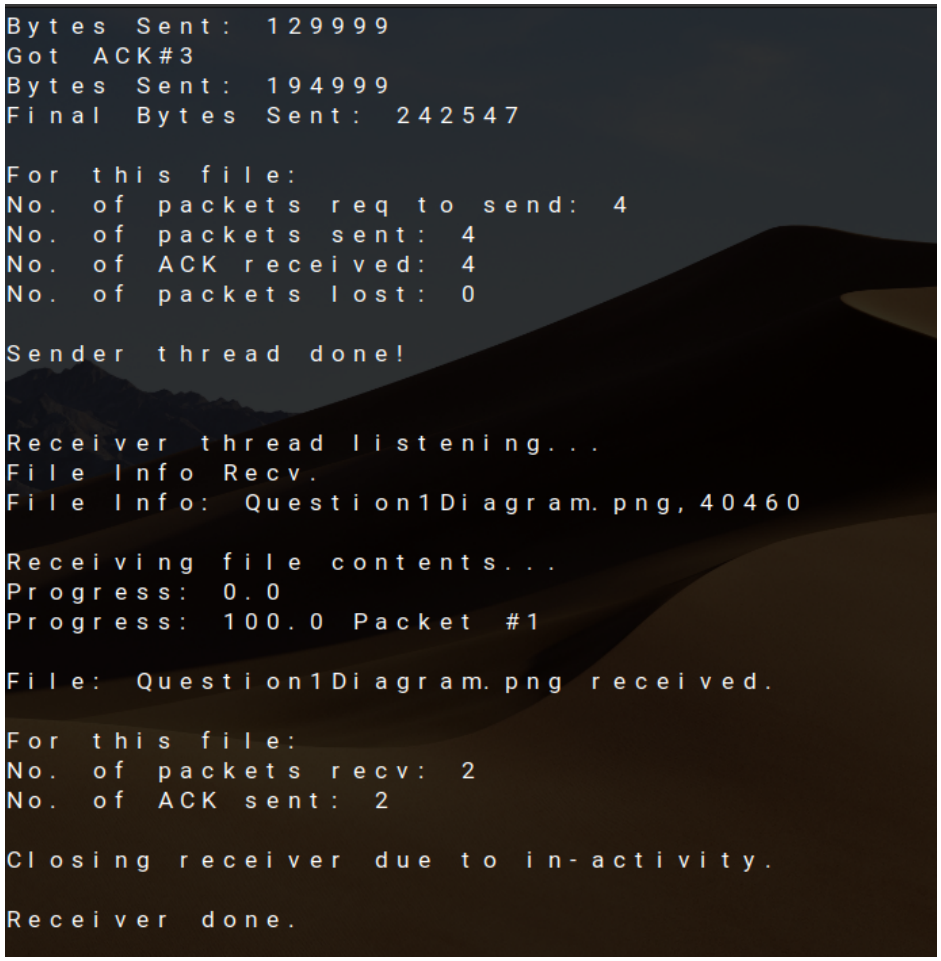
```
103         break;                                  //Work Done! Exit Child Process.
104       }
105       else
106       {
107         printf("%s ","fork() Error!!!");
108         break;
109       }
110     }
111
112
113     return 0;
114 }
```

```
//A TCP Chat Client

//#include"Common.h"
#include<string.h>
#include<time.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#include<signal.h>
#include<errno.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/wait.h>
#include <stdbool.h>


int TCP_ChatClient(int server_desc)
{
  const int BUFFER_SIZE = 4096;
  char msg[BUFFER_SIZE];
  int  len = 0;

  while(true)
  {
  //  cout<<"\n Input : ";fflush(stdout);
  printf("%s ","Input:"); fflush(stdout);
    len = read(fileno(stdin), msg, BUFFER_SIZE);
    //CheckError(len, "read()");

    if(len == 0) //EOF
      return 0;

    len = write(server_desc, msg, len);
    //CheckError(len, "write()");

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;

    len = read(server_desc, msg, BUFFER_SIZE);
    //CheckError(len, "read()");

    //cout<<" Responce From Server :"; fflush(stdout);

    printf("%s ","Response from server :"); fflush(stdout);
    len = write(fileno(stdout), msg, len);
    //CheckError(len, "write()");

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;

  }
```

```
54    return 0;
55  }
56
57  int main()
58  {
59    int sock = socket(AF_INET, SOCK_STREAM, 0);
60    //CheckError(sock, "socket()");
61
62    struct sockaddr_in  server_addr;
63    server_addr.sin_family        = AF_INET;
64    server_addr.sin_addr.s_addr   = inet_addr("127.0.0.1");
65    server_addr.sin_port          = htons(9000);
66
67    int result = connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
68    //CheckError(result, "connect()");
69
70    TCP_ChatClient(sock);
71
72
73    close(sock);
74    return 0;
75  }
```



(a) TCPServer



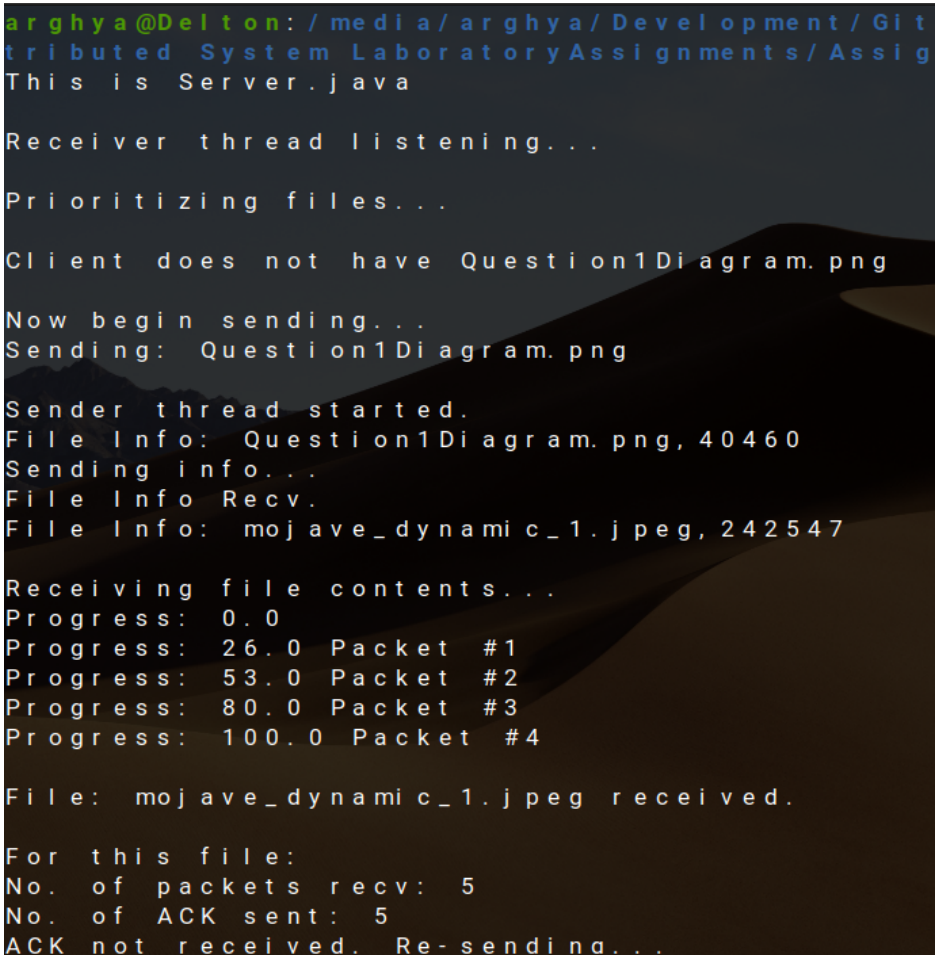
(b) TCPClient

Figure 1: Output:TCP

```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<stdio.h>
#include<unistd.h>
#include<string.h>

#define MAX_MSG 100
#define SERVER_ADDR "127.0.0.1"
#define SERVER_PORT 1500

int main()
{
int sd,rc,n,cliLen;
struct sockaddr x;
struct sockaddr_in cliAddr,servAddr;
char msg[MAX_MSG];

printf("\n sockaddr %ld",sizeof(x));
printf("\n long %ld",sizeof(long));
printf("\nint %ld",sizeof(int));
printf("\n sockaddr_in %ld",sizeof(cliAddr));
printf("\n short %ld\n",sizeof(short));

// build server address structure/*

bzero((char *)&servAddr,sizeof(servAddr));
servAddr.sin_family=AF_INET;
servAddr.sin_addr.s_addr=inet_addr(SERVER_ADDR);
servAddr.sin_port=htons(SERVER_PORT);
//CREATE DATAGRAM SOCKET

sd=socket(AF_INET,SOCK_DGRAM,0);
printf("datagram socket craeted successfully\n");
//BIND LOCAL PORT NUMBER


bind(sd,(struct sockaddr*)&servAddr,sizeof(servAddr));
printf("successfully bind local address\n");

printf("waiting for data on port UDP %u\n",SERVER_PORT);

while(1)
{
//init buffer

memset(msg,0x0,MAX_MSG);

//Receive data from client

cliLen=sizeof(cliAddr);
```

```
54
55 n=recvfrom(sd,msg,MAX_MSG,0,(struct sockaddr *) &cliAddr,&cliLen);
56
57 printf("from %s: UDP port %u: %s \n",inet_ntoa(cliAddr.sin_addr),ntohs(cliAddr.sin_port),msg);
58 printf("from %ld: UDP port %ld,in network byte ordering : %s \n",cliAddr.sin_addr,cliAddr.sin_port,msg);
59
60 }
61
62 return 0;
63
64 }
```

```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/time.h>


#define MAX_MSG 100
#define SERVER_ADDR "127.0.0.1"
#define SERVER_PORT 1500

int main()
{
int sd,rc,n,templen;
struct sockaddr x;
struct sockaddr_in cliAddr,tempAddr,remoteServAddr;
char msg[MAX_MSG];

bzero((char *)&remoteServAddr,sizeof(remoteServAddr));
remoteServAddr.sin_family=AF_INET;
remoteServAddr.sin_addr.s_addr=inet_addr(SERVER_ADDR);
remoteServAddr.sin_port=htons(SERVER_PORT);

sd=socket(AF_INET,SOCK_DGRAM,0);
printf("datagram socket craeted successfully\n");

do{
//send data to server

printf("Enter data to send:");
scanf("%s",msg);

sendto(sd,msg,strlen(msg)+1,0,(struct sockaddr *)&remoteServAddr,sizeof(remoteServAddr));

}while(strcmp(msg,"quit"));

close(sd);

}
```



(a) UDPServer

(b) UDPClient

Figure 2: Output:UDP

2. Write a program to broadcast a message with UDP.

Answer.

```c
/*************************************************************************/
 /*    This sends a BROADCAST Limited message                             */
 /*                                                                       */
 /*    The program send message given as argument to the port given as    */
 /*        second argument.                                               */
 /*                                                                       */
 /*                                                                       */
 /*                                                                       */
 /*                                                                       */
 /*                                                                       */
 /*    Eddie Aronovich                                                    */
 /*************************************************************************/

    #include <stdio.h>
    #include <stdlib.h>
    #include <errno.h>
    #include <string.h>
    #include <sys/types.h>
    #include <netinet/in.h>
    #include <netdb.h>
    #include <sys/socket.h>
    #include <sys/wait.h>
    #include <arpa/inet.h>

    #define MAXBUFLEN 100    /* the port users will be connecting to */

    int main(int argc, char *argv[])
    {
        int sockfd;
        struct sockaddr_in their_addr;  /* connector's address information */
        struct sockaddr_in my_addr;   /* connector's address information */
        int numbytes;
  int optval;     /*Used to build the options for the broadcast */
  int optlen;
        char buf[MAXBUFLEN];            /*The buffer that we read / write each time    */
  int addr_len;     /* Address length for the network functions
                                                        that require that    */
  unsigned long int net_id; /*The network id    */
  long int    host_id;  /*The hpst id in the network  */


        if (argc != 3) {
            fprintf(stderr,"usage: %s message port\n",argv[1]);
            exit(1);
        }

        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {  /* The socket should be changed to broadcast  */
                /* This part demands root permissions   */
            perror("socket");
```

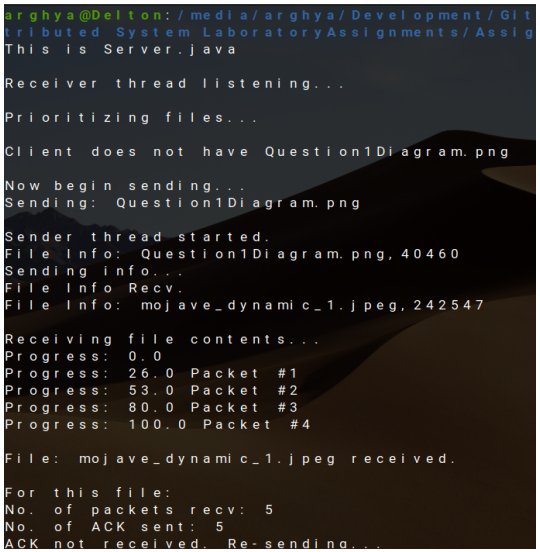
```
50              exit(1);
51          }
52
53      optval=1; /*Prepare the options of the socket for Broadcast */
54      optlen=sizeof(int);
55
56      if(setsockopt(sockfd,SOL_SOCKET,SO_BROADCAST,(char *) &optval,optlen)){
57          perror("Error setting socket to BROADCAST mode");
58            exit(1);
59      }
60
61            their_addr.sin_family = AF_INET;            /* Protocol family - host byte order */
62      their_addr.sin_port=htons((unsigned short) atoi(argv[2]));  /* port - short, network byte order */
63      their_addr.sin_addr.s_addr=htonl(INADDR_BROADCAST);     /* send to all */
64      /*their_addr.sin_addr.s_addr=inet_addr("enter here the IP addres in dot notation"); */  /* send to all */
65            bzero(&(their_addr.sin_zero), 8);     /* zero the rest of the struct */
66
67            if ((numbytes=sendto(sockfd, argv[1], strlen(argv[1]), 0, \
68                 (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
69                perror("sendto");
70                exit(1);
71            }
72
73            printf("sent %d bytes to %s\n",numbytes,inet_ntoa(their_addr.sin_addr));
74
75            close(sockfd);
76
77            return 0;
78        }
```

```c
/*************************************************************************/
/* UDP Broadcast listener                                                */
/*                                                                       */
/* This program is a UDP server that recieves message sent by            */
/* Broadcast                                                             */
/*                                                                       */
/* Prepared by Eddie Aronovich                                           */
/*************************************************************************/
    #include <stdio.h>
    #include <stdlib.h>
    #include <errno.h>
    #include <string.h>
    #include <sys/types.h>
    #include <netinet/in.h>
    #include <sys/socket.h>
    #include <sys/wait.h>
    #include <sys/time.h>
    #include <sys/unistd.h>
    #include <arpa/inet.h>

    #define MYPORT 5000    /* the port users will be sending to */

    #define MAXBUFLEN 100

    int main()
    {
        int sockfd;
        struct sockaddr_in my_addr;    /* my address information */
        struct sockaddr_in their_addr; /* connector's address information */
        int addr_len, numbytes;
        char buf[MAXBUFLEN];
  int option = 1;

        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
            perror("socket");
            exit(1);
        }
  else
    printf(" \n The socket got sockfd=%d \n ",sockfd);

  setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));

        my_addr.sin_family = AF_INET;         /* host byte order */
        my_addr.sin_port = htons(MYPORT);     /* short, network byte order */
        my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
        bzero(&(my_addr.sin_zero), 8);        /* zero the rest of the struct */


        if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) /* The bind command makes the ability to wait for messages */
                                                                == -1) {
            perror("bind");
            exit(1);
        }
```

```
54
55  printf("Wait for packet \n");
56
57        addr_len = sizeof(struct sockaddr);
58
59
60  if ((numbytes=recvfrom(sockfd, buf, MAXBUFLEN, 0, \
61                            (struct sockaddr *)&their_addr, &addr_len)) == -1) {
62                        perror("recvfrom");
63                        exit(1);
64                }
65
66        printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
67        printf("packet is %d bytes long ",numbytes);
68        buf[numbytes] = '\0';
69        printf("packet contains \"%s\"\n",buf);
70
71
72        close(sockfd);
73
74  return 0;
75    }
```

```c
/*************************************************************************/
/* UDP Broadcast listener                                                */
/*                                                                       */
/* This program is a UDP server that recieves message sent by            */
/* Broadcast                                                             */
/*                                                                       */
/* Prepared by Eddie Aronovich                                           */
/*************************************************************************/
    #include <stdio.h>
    #include <stdlib.h>
    #include <errno.h>
    #include <string.h>
    #include <sys/types.h>
    #include <netinet/in.h>
    #include <sys/socket.h>
    #include <sys/wait.h>
    #include <sys/time.h>
    #include <sys/unistd.h>
    #include <arpa/inet.h>

    #define MYPORT 5000    /* the port users will be sending to */

    #define MAXBUFLEN 100

    int main()
    {
        int sockfd;
        struct sockaddr_in my_addr;    /* my address information */
        struct sockaddr_in their_addr; /* connector's address information */
        int addr_len, numbytes;
        char buf[MAXBUFLEN];
  int option = 1;

        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
            perror("socket");
            exit(1);
        }
  else
    printf(" \n The socket got sockfd=%d \n ",sockfd);

  setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));

        my_addr.sin_family = AF_INET;         /* host byte order */
        my_addr.sin_port = htons(MYPORT);     /* short, network byte order */
        my_addr.sin_addr.s_addr = INADDR_ANY; /* auto-fill with my IP */
        bzero(&(my_addr.sin_zero), 8);        /* zero the rest of the struct */


        if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) /* The bind command makes the ability to wait for messages */
                                                                == -1) {
            perror("bind");
            exit(1);
        }
```

```
54
55  printf("Wait for packet \n");
56
57        addr_len = sizeof(struct sockaddr);
58
59
60  if ((numbytes=recvfrom(sockfd, buf, MAXBUFLEN, 0, \
61                            (struct sockaddr *)&their_addr, &addr_len)) == -1) {
62                        perror("recvfrom");
63                        exit(1);
64                }
65
66        printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
67        printf("packet is %d bytes long ",numbytes);
68        buf[numbytes] = '\0';
69        printf("packet contains \"%s\"\n",buf);
70
71
72        close(sockfd);
73
74  return 0;
75    }
```



(a) UDP Broadcast Server

(b) UDP Broadcast Client1

(c) UDP Broadcast Client2

Figure 3: Output:UDP Broadcast