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.

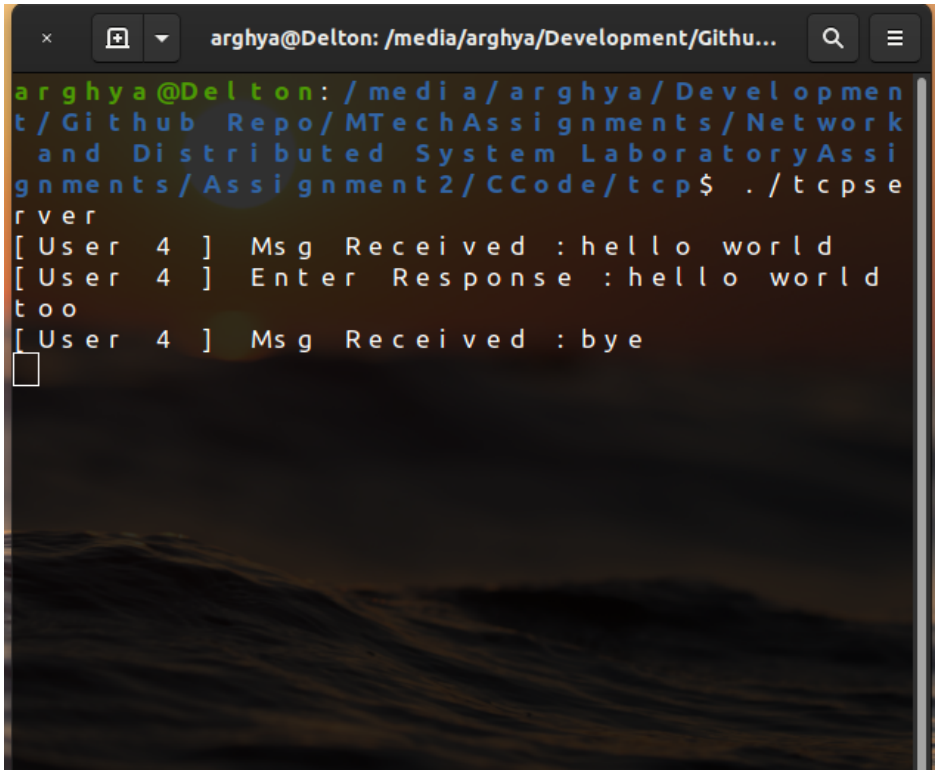
Answer.

```c
//This is the Server side implementation of TCP Chat

#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdio.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)
  {
    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;

    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;
  }

  return 0;
}

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

    //Create and Fill Address Structure for this Server
  struct  sockaddr_in server_addr;
```

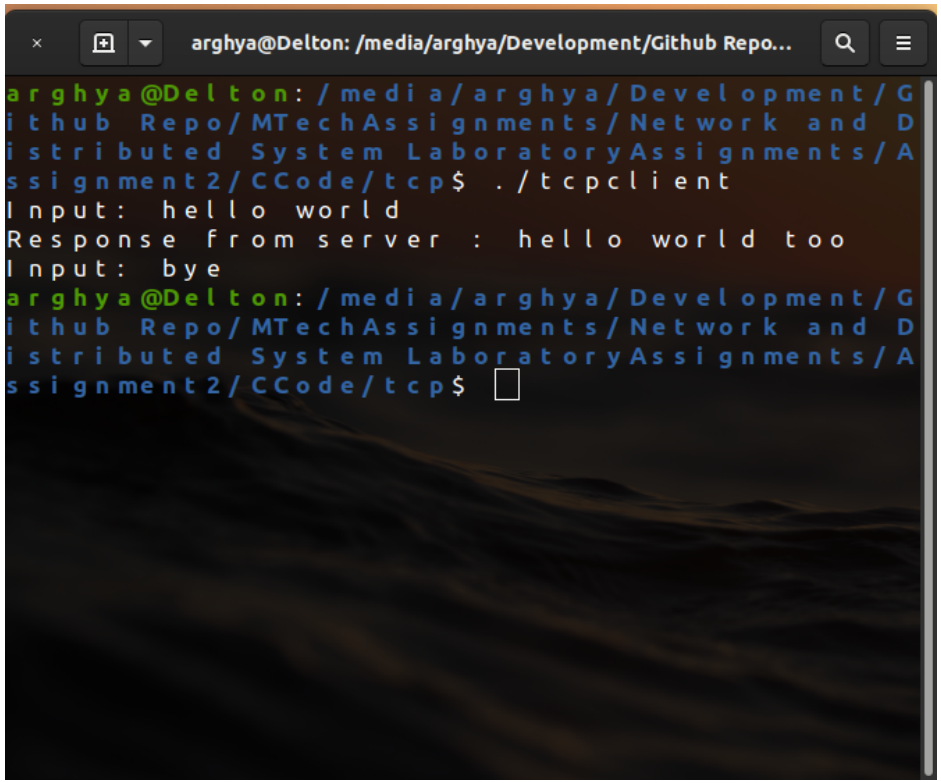
```
  server_addr.sin_family      = AF_INET;    //Address Family (AF_INET, AF_INET6, AF_LOCAL, ...)
    server_addr.sin_addr.s_addr = INADDR_ANY; //Internet Address (INADDR_ANY-> Accept connection at any IP Address)
    server_addr.sin_port        = htons(9000);//Port Number (htons -> h.HOST t.TO n.NETWORK 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));

  //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

    //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);
      break;                              //Work Done! Exit Child Process.
    }
    else
    {
      printf("%s ","fork() Error!!!");
      break;
    }
  }


  return 0;
}
```

```c
//This is the client side implementation of TCP Chat

#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/wait.h>
#include<netinet/in.h>
#include <stdbool.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>


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

  while(true)
  {
    printf("%s ","Input:"); fflush(stdout);
    len = read(fileno(stdin), msg, BUFFER_SIZE);

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

    len = write(server_desc, msg, len);

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

    len = read(server_desc, msg, BUFFER_SIZE);

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

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

  }
  return 0;
}

int main()
{
  int sock = socket(AF_INET, SOCK_STREAM, 0);

  struct sockaddr_in  server_addr;
```

```
  server_addr.sin_family        = AF_INET;
  server_addr.sin_addr.s_addr   = inet_addr("127.0.0.1");
  server_addr.sin_port          = htons(9000);

  int result = connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));

  TCP_ChatClient(sock);


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



(a) TCPServer



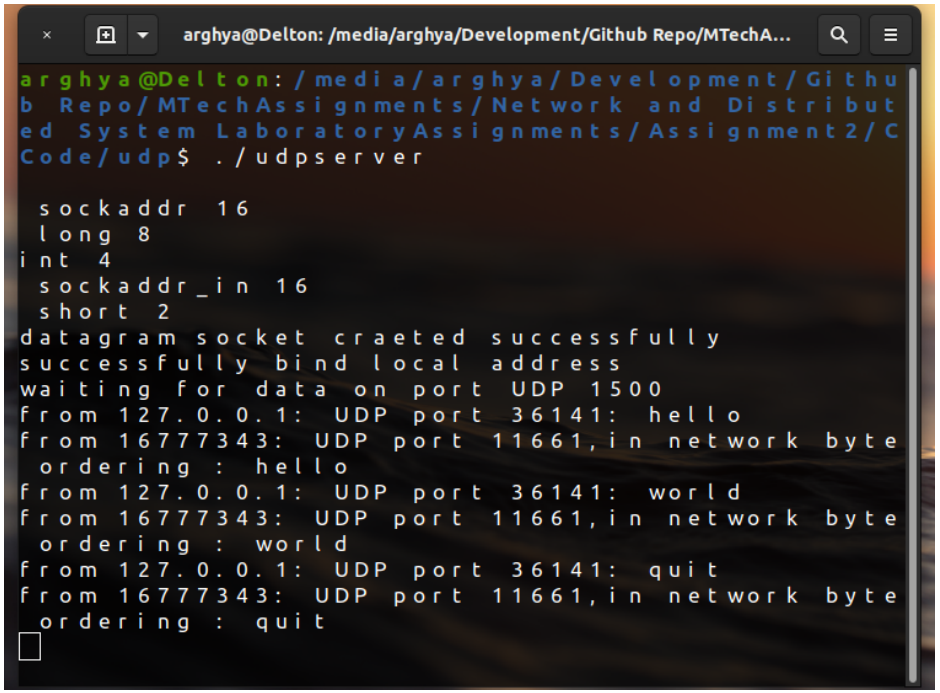
(b) TCPClient

Figure 1: Output:TCP

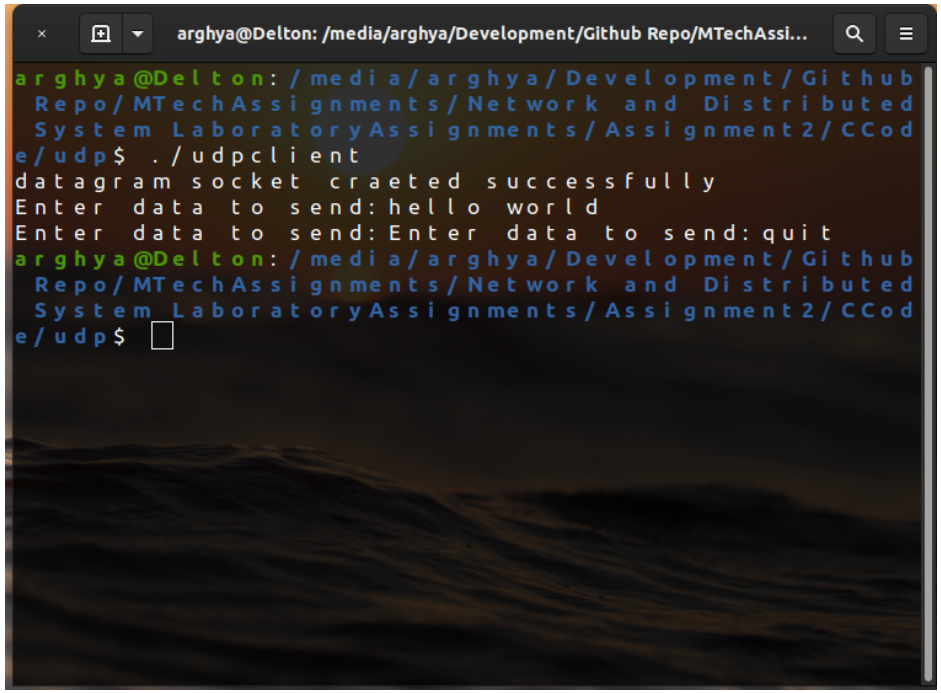
```c
//This is the Server side implementation of UDP Chat

#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);

n=recvfrom(sd,msg,MAX_MSG,0,(struct sockaddr *) &cliAddr,&cliLen);

printf("from %s: UDP port %u: %s \n",inet_ntoa(cliAddr.sin_addr),ntohs(cliAddr.sin_port),msg);
printf("from %ld: UDP port %ld,in network byte ordering : %s \n",cliAddr.sin_addr,cliAddr.sin_port,msg);

}

return 0;

}
```

```c
//This is the client side implementation of UDP Chat

#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 is the server side implementation of UDP Broadcast service for sending message passed in the parameter to the clients

#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");
        exit(1);
    }

  optval=1;                       /*Prepare the options of the socket for Broadcast */
  optlen=sizeof(int);

  if(setsockopt(sockfd,SOL_SOCKET,SO_BROADCAST,(char *) &optval,optlen)){
      perror("Error setting socket to BROADCAST mode");
        exit(1);
  }

```

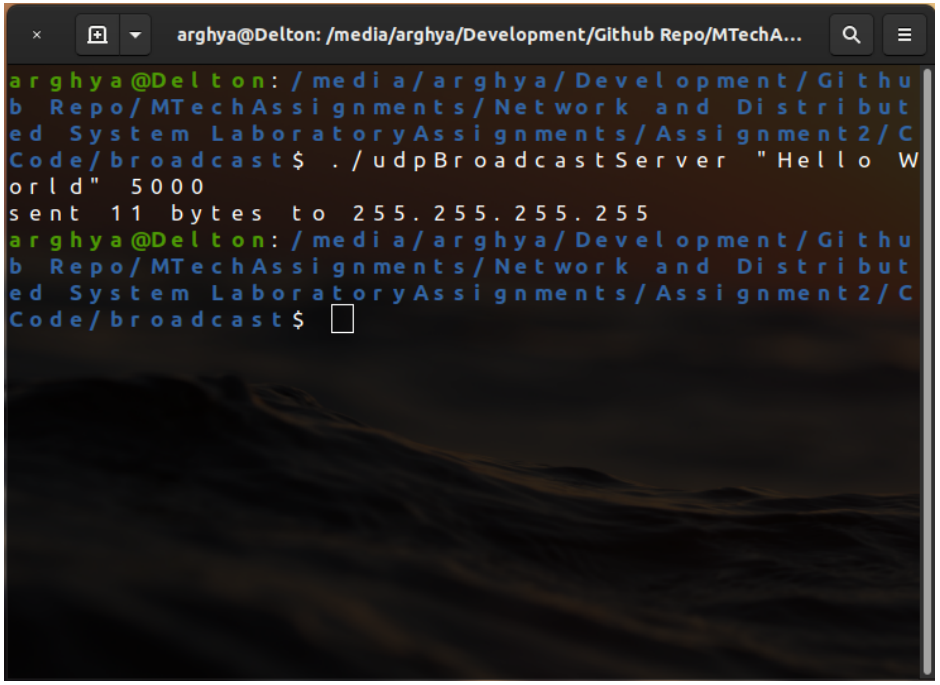
```
    their_addr.sin_family = AF_INET;            /* Protocol family - host byte order */
  their_addr.sin_port=htons((unsigned short) atoi(argv[2]));  /* port - short, network byte order */
  their_addr.sin_addr.s_addr=htonl(INADDR_BROADCAST);     /* send to all */
  bzero(&(their_addr.sin_zero), 8);     /* zero the rest of the struct */

    if ((numbytes=sendto(sockfd, argv[1], strlen(argv[1]), 0, \
         (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1) {
        perror("sendto");
        exit(1);
    }

    printf("sent %d bytes to %s\n",numbytes,inet_ntoa(their_addr.sin_addr));

    close(sockfd);

    return 0;
}
```

```c
//This is the client side implementation of UDP Broadcast service for receiving message passed by the server

#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);
    }

  printf("Wait for packet \n");

    addr_len = sizeof(struct sockaddr);


```

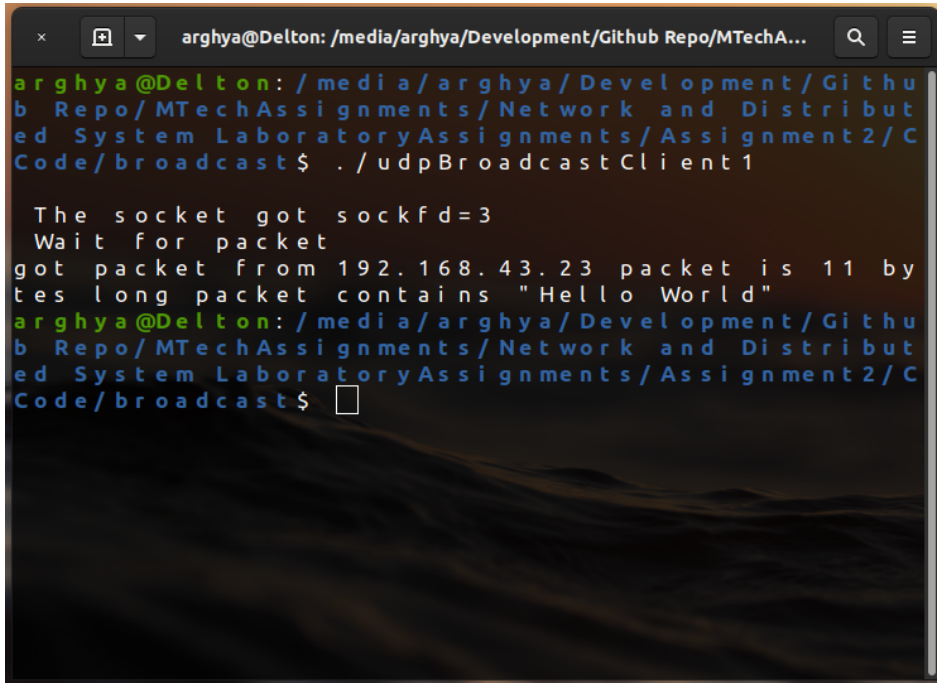
```
  if ((numbytes=recvfrom(sockfd, buf, MAXBUFLEN, 0, \
                            (struct sockaddr *)&their_addr, &addr_len)) == -1) {
                        perror("recvfrom");
                        exit(1);
                }

    printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
    printf("packet is %d bytes long ",numbytes);
    buf[numbytes] = '\0';
    printf("packet contains \"%s\"\n",buf);


    close(sockfd);

  return 0;
}
```

```c
//This is the client side implementation of UDP Broadcast service for receiving message passed from the server

#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);
    }

   printf("Wait for packet \n");

    addr_len = sizeof(struct sockaddr);


```

```
    if ((numbytes=recvfrom(sockfd, buf, MAXBUFLEN, 0, \
                        (struct sockaddr *)&their_addr, &addr_len)) == -1) {
                    perror("recvfrom");
                    exit(1);
            }

     printf("got packet from %s ",inet_ntoa(their_addr.sin_addr));
     printf("packet is %d bytes long ",numbytes);
     buf[numbytes] = '\0';
     printf("packet contains \"%s\"\n",buf);


     close(sockfd);

    return 0;
}
```



(a) UDP Broadcast Server



(b) UDP Broadcast Client1

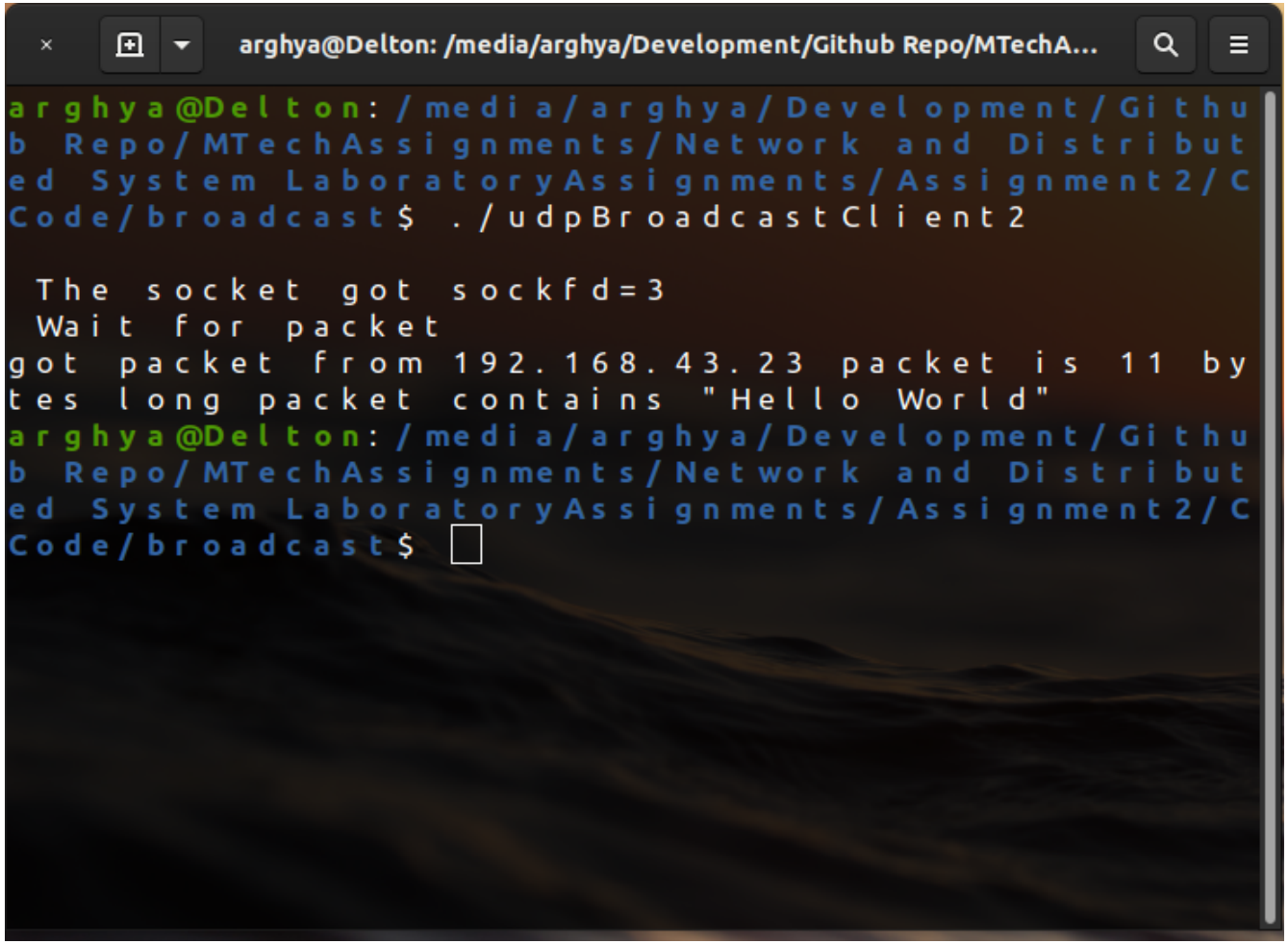


Figure 4: UDP Broadcast Client2