Ayush Goyal
190905522 CSE D Roll 62

Computer Networks Lab 1

UDP Communication between Client and Server

We create a file called “udpser1.c” for server side connection ,the content of which is as follows :

```
// server program for udp connection

#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5004
#define MAXLINE 1000

// Driver code
int main()
{
        char buffer[100];
        char *message = "Hello Client";
        int listenfd, len,n;
        struct sockaddr_in servaddr, cliaddr;
        bzero(&servaddr, sizeof(servaddr));

        // Create a UDP Socket
        listenfd = socket(AF_INET, SOCK_DGRAM, 0);
        //servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
   servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;

        // bind server address to socket descriptor
        bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

        //receive the datagram
        len = sizeof(cliaddr);

        n = recvfrom(listenfd, buffer, sizeof(buffer),0, (struct sockaddr*)&cliaddr,&len);
        buffer[n] = '\0';
        puts(buffer);
   //getchar();


        // send the response
   //getchar();
```

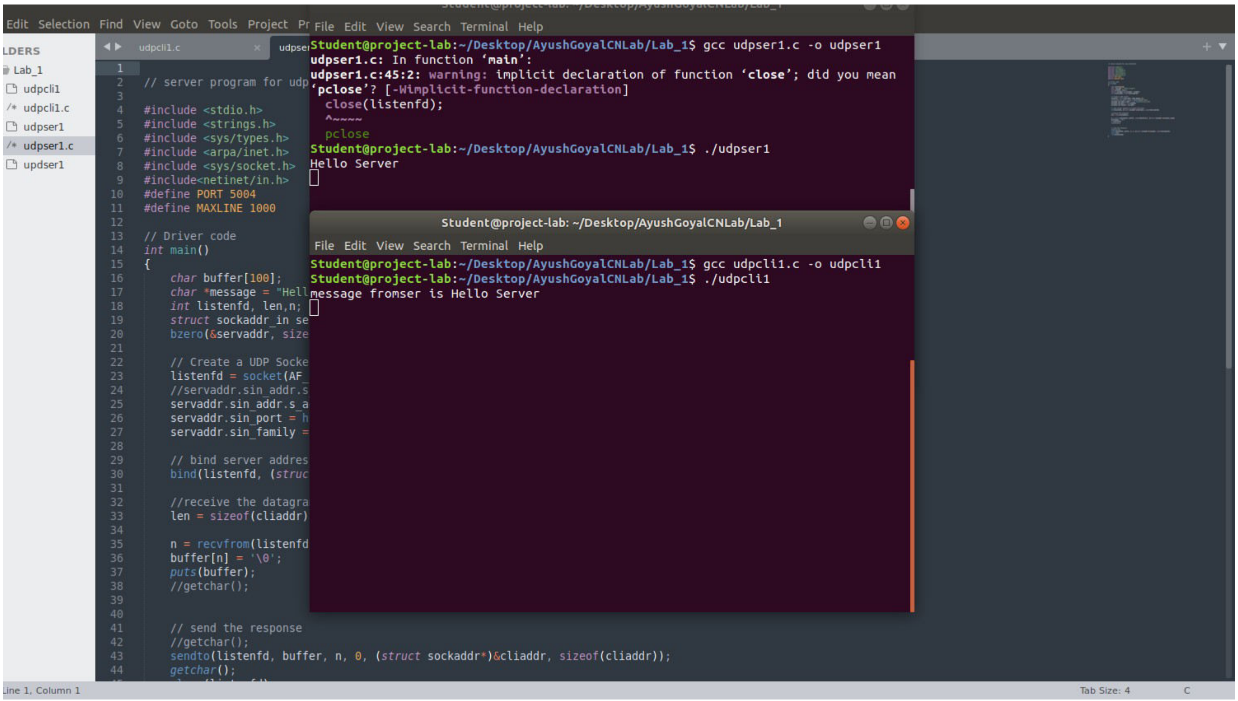
```
        sendto(listenfd, buffer, n, 0, (struct sockaddr*)&cliaddr, sizeof(cliaddr));
        getchar();
        close(listenfd);
}
```

We create a file called “udpcli1.c” for client side connection, the content of which is as follows :

```
// udp client driver program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>

#define PORT 5004
#define MAXLINE 1000

// Driver code
int main()
{
        char buffer[100];
        char *message = "Hello Server";
        int sockfd, n,len;
        struct sockaddr_in servaddr, cliaddr;

        // clear servaddr
        bzero(&servaddr, sizeof(servaddr));
        //servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
   servaddr.sin_addr.s_addr =  inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;

        // create datagram socket
        sockfd = socket(AF_INET, SOCK_DGRAM, 0);


        sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)&servaddr, sizeof(servaddr));
        len=sizeof(cliaddr);
   //getchar();
        // waiting for response
        n=recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&cliaddr,&len );
   buffer[n]='\0';
        printf("message fromser is %s \n",buffer);
   getchar();

        // close the descriptor
        close(sockfd);
}
```

When we compile and run the server program, it starts running and waits for the client side connection. Meanwhile when we compile and run the client program and connect the client side and send message, the server responds with appropriate message as shown in the screenshot below :

The overall picture :

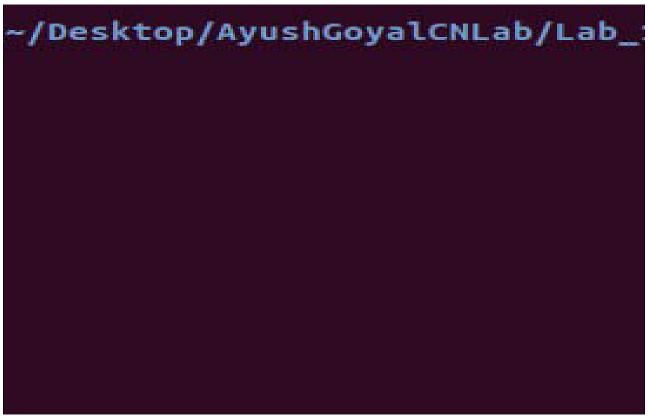

Edit Selection Find View Goto Tools Project Pr
LDERS
Lab_1
udpcli1
udpcli1.c
udpser1
udpser1.c
updser1
udpcli1.c
udpser
Line 1, Column 1
Tab Size: 4
C
File Edit View Search Terminal Help
Student@project-lab: ~/Desktop/AyushGoyalCNLab/Lab_1

```

// server program for udp

#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5004
#define MAXLINE 1000

// Driver code
int main()
{
    char buffer[100];
    char *message = "Hell
    int listenfd, len,n;
    struct sockaddr_in se
    bzero(&servaddr, size

    // Create a UDP Socke
    listenfd = socket(AF_
    //servaddr.sin_addr.s
    servaddr.sin_addr.s_a
    servaddr.sin_port = h
    servaddr.sin_family =

    // bind server addres
    bind(listenfd, (struc

    //receive the datagra
    len = sizeof(cliaddr)

    n = recvfrom(listenfd
    buffer[n] = '\0';
    puts(buffer);
    //getchar();


    // send the response
    //getchar();
    sendto(listenfd, buffer, n, 0, (struct sockaddr*)&cliaddr, sizeof(cliaddr));
    getchar();
```

```
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ gcc udpser1.c -o udpser1
udpser1.c: In function ‘main’:
udpser1.c:45:2: warning: implicit declaration of function ‘close’; did you mean
‘pclose’? [-Wimplicit-function-declaration]
  close(listenfd);
  ^~~~~
  pclose
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ ./udpser1
Hello Server
```

```
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ gcc udpcli1.c -o udpcli1
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ ./udpcli1
message fromser is Hello Server
```

The Server Side :



```
~/Desktop/AyushGoyalCNLab/Lab_
```

The Client Side :

TCP Communication between Client and Server

We create a file called “tcpser1.c” for server side connection , the content of which is as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
int main()
{
            int sd,nd,n,len,reult;
            struct sockaddr_in seraddress, cliaddr;
            char buf[256];
            sd=socket(AF_INET, SOCK_STREAM,0);
            seraddress.sin_family=AF_INET;
            seraddress.sin_addr.s_addr=INADDR_ANY;
            seraddress.sin_port=htons(10200);
            bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));
            listen(sd,5);
            len=sizeof(cliaddr);
            nd=accept(sd,(struct sockaddr*)&cliaddr,&len);
            n=read(nd,buf,sizeof(buf));
        buf[n]='\0';
        printf("Message from Client : %s\n",buf);
        n=write(nd,buf,strlen(buf));
            getchar();
        close(nd);
        close(sd);
}
```

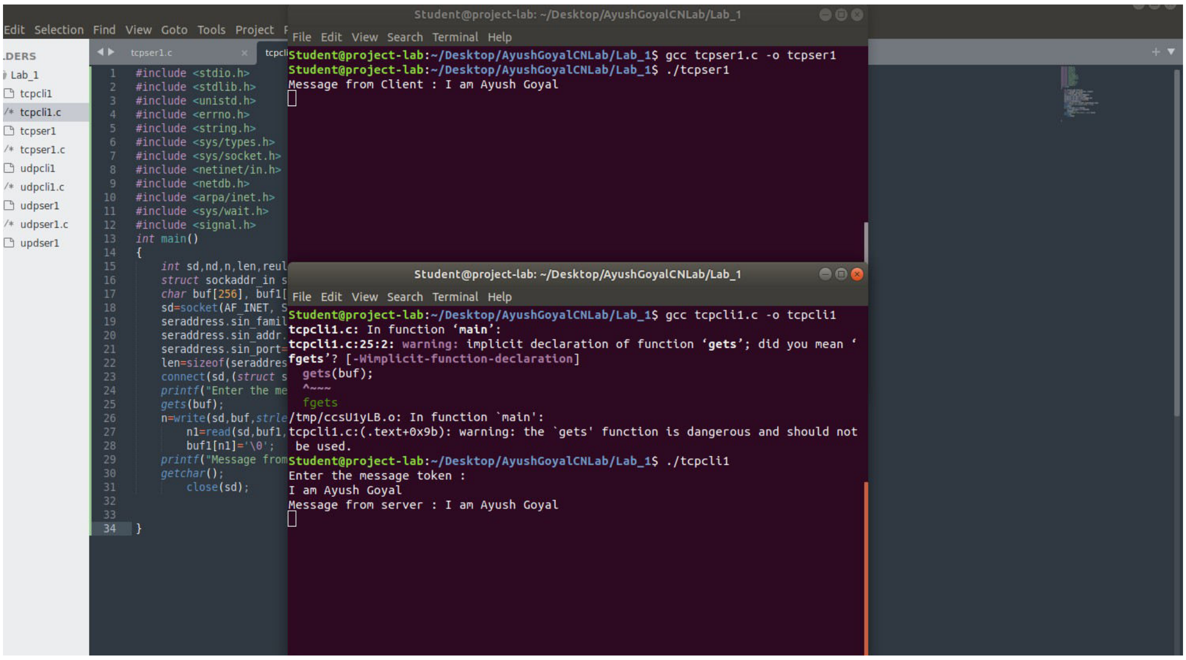
We create a file called “tcpcli1.c” for client side connection, the content of which is as follows :

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

```
#include <sys/wait.h>
#include <signal.h>
int main()
{
        int sd,nd,n,len,reult,n1;
        struct sockaddr_in seraddress, cliaddr;
        char buf[256], buf1[256];
        sd=socket(AF_INET, SOCK_STREAM,0);
        seraddress.sin_family=AF_INET;
        seraddress.sin_addr.s_addr=INADDR_ANY;
        seraddress.sin_port=htons(10200);
        len=sizeof(seraddress);
        connect(sd,(struct sockaddr*)&seraddress,len);
        printf("Enter the message token : \n");
        gets(buf);
        n=write(sd,buf,strlen(buf));
      n1=read(sd,buf1,sizeof(buf1));
      buf1[n1]='\0';
        printf("Message from server : %s\n",buf1);
        getchar();
      close(sd);


}
```

When we compile and run the server program, it starts running and waits for the client side connection. Meanwhile when we compile and run the client program and connect the client side, it asks us to send a message token. When we send a message token, the server responds with appropriate message as shown in the screenshot below :

The overall picture :

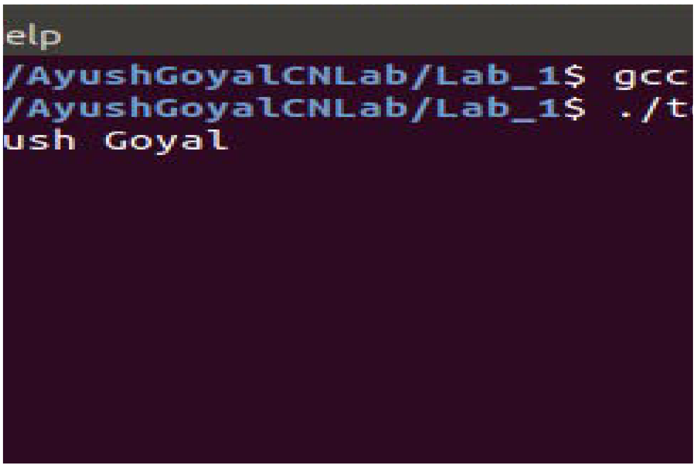

Edit Selection Find View Goto Tools Project
DERS
Lab_1
tcpcli1
tcpcli1.c
tcpser1
tcpser1.c
udpcli1
udpcli1.c
udpser1
udpser1.c
updser1
tcpser1.c
Student@project-lab: ~/Desktop/AyushGoyalCNLab/Lab_1
File Edit View Search Terminal Help

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <signal.h>
int main()
{
    int sd,nd,n,len,reul
    struct sockaddr_in s
    char buf[256], buf1[
    sd=socket(AF_INET, S
    seraddress.sin_famil
    seraddress.sin_addr.
    seraddress.sin_port=
    len=sizeof(seraddres
    connect(sd,(struct s
    printf("Enter the me
    gets(buf);
    n=write(sd,buf,strle
        n1=read(sd,buf1,
        buf1[n1]='\0';
    printf("Message from
    getchar();
        close(sd);


}
```

```
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ gcc tcpser1.c -o tcpser1
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ ./tcpser1
Message from Client : I am Ayush Goyal
```

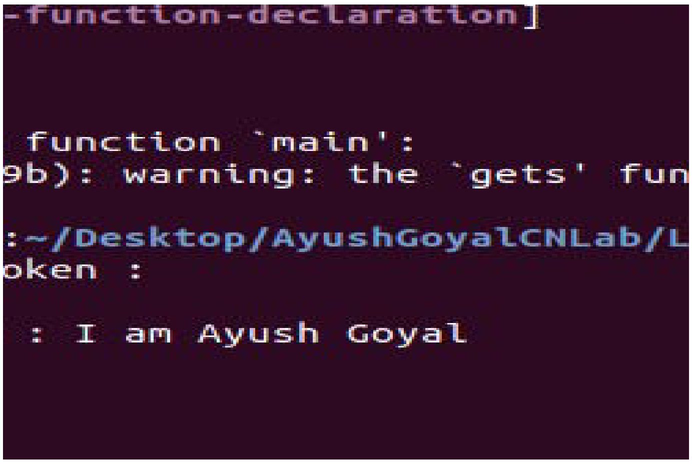
```
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ gcc tcpcli1.c -o tcpcli1
tcpcli1.c: In function ‘main’:
tcpcli1.c:25:2: warning: implicit declaration of function ‘gets’; did you mean ‘
fgets’? [-Wimplicit-function-declaration]
  gets(buf);
  ^~~~
  fgets
/tmp/ccsU1yLB.o: In function `main':
tcpcli1.c:(.text+0x9b): warning: the `gets' function is dangerous and should not
 be used.
Student@project-lab:~/Desktop/AyushGoyalCNLab/Lab_1$ ./tcpcli1
Enter the message token :
I am Ayush Goyal
Message from server : I am Ayush Goyal
```

The Server Side :



```
elp
/AyushGoyalCNLab/Lab_1$ gcc
/AyushGoyalCNLab/Lab_1$ ./t
ush Goyal
```

The Client Side :



```
-function-declaration]


 function `main':
9b): warning: the `gets' fun

:~/Desktop/AyushGoyalCNLab/L
oken :

 : I am Ayush Goyal
```

Above were the demonstrartion of both UDP and TCP client server communication protocols.

THE END