

## Computer Networks Lab

### Week-4

Aug 13, 2021

**Venkata Naga Sai Ram Nomula**

**RA1911033010021**

**L2 - SWE**

**Aim: To create simple udp client server communication**

#### **Procedure:**

STEP 1: CREATE A FOLDER (Regno)

STEP 2: CREATE a filename server.c

STEP 3: open or click server.c

STEP4: WRITE THE PROGRAM IN server.c

STEP5: CREATE a filename client.c

STEP6: open or click client.c

STEP7: Write the program for client.c

STEP8: OPEN A NEW TERMINAL

STEP9: Type cd foldername

STEP10: Type cc server.c

STEP11: Type ./a.out

STEP12: Open one more terminal

STEP13: Type cc client.c

STEP14: Type ./a.out 127.0.0.1

STEP15: Type any message, say hello in the client terminal

STEP16: Verify its received in the server

#### **Code:**

##### **Server.c**

```
#include<sys/socket.h>
```

```
#include<stdio.h>
```

```
#include<unistd.h>
```

```
#include<string.h>
```

```
#include<stdlib.h>
```

```
#include<netinet/in.h>
```

```
#include<netdb.h>
```

```

#include<arpa/inet.h>
#include<sys/types.h>
int main(int argc,char *argv[])
{
int sd;
char buff[1024];
struct sockaddr_in cliaddr,servaddr;
socklen_t clilen;
clilen=sizeof(cliaddr);
/*UDP socket is created, an Internet socket address structure is filled with wildcard
address & server's well known port*/
sd=socket(AF_INET,SOCK_DGRAM,0);
if (sd<0)
{
perror ("Cannot open Socket");
exit(1);
}
bzero(&servaddr,sizeof(servaddr));
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(1504);
/*Bind function assigns a local protocol address to the socket*/
if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
perror("error in binding the port");
exit(1);
}
printf("%s","Server is Running...\n");
while(1)
{
bzero(&buff,sizeof(buff));
/*Read the message from the client*/
if(recvfrom(sd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,&clilen)<0)
{

```

```

perror("Cannot rec data");
exit(1);
}
printf("%sMessage is received \n",buff);
/*Sendto function is used to echo the message from server to client side*/
if(sendto(sd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,clilen)<0)
{
perror("Cannot send data to client");
exit(1);
}
printf("Send data to UDP Client: %s",buff);
}
close(sd);
return 0;
}

```

## Client.c

```

#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<netinet/in.h>
#include<netdb.h>
int main(int argc,char*argv[])
{
int sd;
char buff[1024];
struct sockaddr_in servaddr;
socklen_t len;
len=sizeof(servaddr);
/*UDP socket is created, an Internet socket address structure is filled with
wildcard address & server's well known port*/
sd = socket(AF_INET,SOCK_DGRAM,0);
if(sd<0)

```

```

{
perror("Cannot open socket");
exit(1);
}
bzero(&servaddr,len);
/*Socket address structure*/
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(4000);
while(1)
{
printf("Enter Input data : \n");
bzero(buff,sizeof(buff));
/*Reads the message from standard input*/
fgets(buff,sizeof (buff),stdin);
/*sendto is used to transmit the request message to the server*/
if(sendto (sd,buff,sizeof (buff),0,(struct sockaddr*)&servaddr,len)<0)
{
perror("Cannot send data");
exit(1);
}
printf("Data sent to UDP Server:%s",buff);
bzero(buff,sizeof(buff));
/*Receiving the echoed message from server*/
if(recvfrom (sd,buff,sizeof(buff),0,(struct sockaddr*)&servaddr,&len)<0)
{
perror("Cannot receive data");
exit(1);
}
printf("Received Data from server: %s",buff);
}
close(sd);
return 0;
}

```

## Screenshot:

```
server.c
14 struct sockaddr_in cliaddr,servaddr;
15 socklen_t clien;
16 clien=sizeof(cliaddr);
17 /*UDP socket is created, an Internet socket address:
18 address & server's well known port*/
19 sd=socket(AF_INET,SOCK_DGRAM,0);
20 if (sd<0)
21 {
22     perror ("Cannot open Socket");
23     exit(1);
24 }
25 bzero(&servaddr,sizeof(servaddr));
26 /*Socket address structure*/
27 servaddr.sin_family=AF_INET;
28 servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
29 servaddr.sin_port=htons(1504);
30 /*Bind function assigns a local protocol address */
31 if(bind(sd,(struct sockaddr*)&servaddr,sizeof(serv
32 {
33     perror("error in binding the port");
34     exit(1);
35 }
36 printf("%s","Server is Running...\n");
37 while(1)
38 {
39     bzero(&buff,sizeof(buff));
40     /*Read the message from the client*/
41     if(recvfrom(sd,buff, 45,0, C and C++ Spaces: 1
42 {
43     perror("Cannot rec data");
```

```
client.c
14 socklen_t len;
15 len=sizeof(servaddr);
16 /*UDP socket is created, an Internet socket address:
17 wildcard address & server's well known port*/
18 sd = socket(AF_INET,SOCK_DGRAM,0);
19 if(sd<0)
20 {
21     perror("Cannot open socket");
22     exit(1);
23 }
24 bzero(&servaddr,len);
25 /*Socket address structure*/
26 servaddr.sin_family=AF_INET;
27 servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
28 servaddr.sin_port=htons(1504);
29 while(1)
30 {
31     printf("Enter Input data : \n");
32     bzero(buff,sizeof(buff));
33     /*Reads the message from standard input*/
34     fgets(buff,sizeof (buff),stdin);
35     /*sendto is used to transmit the request message */
36     if(sendto (sd,buff,sizeof (buff),0,(struct sockadd
37 {
38     perror("Cannot send data");
39     exit(1);
40 }
41 printf("Data sent to UDP Server:%s" buff);
42 bzero(buff,sizeof(b
43 /*Receiving the echoed message from server*/
```

```
./a.out - "ip-172-31-9-200" x
RA1911033010029:~/environment $ cd RA1911033010021
RA1911033010029:~/environment/RA1911033010021 $ ls
TCP  UDP
RA1911033010029:~/environment/RA1911033010021 $ cd UDP
RA1911033010029:~/environment/RA1911033010021/UDP $ cc se
rver.c
RA1911033010029:~/environment/RA1911033010021/UDP $ ./a.o
ut
Server is Running...
Experiment 4 done!
Message is received
Send data to UDP Client: Experiment 4 done!
```

```
./a.out - "ip-172-31-9-200" x
RA1911033010029:~/environment $ cd RA191103
3010021
RA1911033010029:~/environment/RA19110330100
21 $ cd UDP
RA1911033010029:~/environment/RA19110330100
21/UDP $ cc client.c
RA1911033010029:~/environment/RA19110330100
21/UDP $ ./a.out 127.0.0.1Enter Input data :
Experiment 4 done!
Data sent to UDP Server:Experiment 4 done!
Received Data from server: Experiment 4 don
e!
Enter Input data :
```