



COMPUTER NETWORKS

EXP 4



UDP CLIENT SERVER COMMUNICATION

AUGUST 13, 2021

ROEHIT RANGANATHAN
RA1911033010017 | L2

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(3000);
/*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");
int count=0;
    int packet=0;
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);
}
{
++count;
packet = strlen(buff);
printf("Total message=%d and size of last packet=%d\n",count,packet);
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(3000);
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;

}

```

OUTPUT:

```

server.c
serverCustom.c
client.c

38  perror("Cannot send data");
39  exit(1);
40  }
41  printf("Data sent to UDP Server:%s",buff);
42  bzero(buff,sizeof(buff));
43  /*Receiving the echoed message from server*/
44  if(recvfrom (sd,buff,sizeof(buff),0,(struct sockaddr*)&servaddr,&len)<0)
45  {
46      perror("Cannot receive data");
47      exit(1);
48  }

/a.out - "p-172-31-9-200" x
RA1911033010029:~/environment/RA1911033010017/UDP $ cc serverCustom.c
RA1911033010029:~/environment/RA1911033010017/UDP $ ./a.out
Server is Running...
hello
Message is received
Total message-1 and size of last packet-6
Send data to UDP Client: hello
this is Roohit
Message is received
Total message-2 and size of last packet-15
Send data to UDP Client: this is Roohit
voill
Message is received
Total message-3 and size of last packet-5
Send data to UDP Client: voill

/a.out - "p-172-31-9-200" x
RA1911033010029:~/environment/RA1911033010017/UDP $ cc client.c
RA1911033010029:~/environment/RA1911033010017/UDP $ ./a.out 127.0.0.1
Enter Input data :
hello
Data sent to UDP Server:hello
Received Data from server: hello
Enter Input data :
this is Roohit
Data sent to UDP Server:this is Roohit
Received Data from server: this is Roohit
Enter Input data :
voill
Data sent to UDP Server:voill
Received Data from server: voill
Enter Input data :

```