Computer Networks

Name : M. Sai Saranya

Regno: 22BAI1471

Course Title : Computer Networks

Course code : BCSE308P

Slot : L45-46

Faculty : Dr Neelanarayanan V

S.No	Experiment Name	Date	Page No.	Marks
1.	**Basic Network Configuration Commands**	**10-01-2024**		
2.	**Client-Server Application Echo**	**17-01-2024**		
3.	**IP Address Validation and Simple application of ATM using TCP**	**24-01-2024**		
4.	**CRC code generator using socket programming**	**07-02-2024**		
5.	**Echo programming using UDP**	**21-02-2024**		

Experiment No. 5

Experiment Name: Client-Server Application (Echo client-server)

Date: 21-2-2024

Problem Statement

Design a simple client-server application named Echo client-server using c program in UDP protocol and execute in Linux.

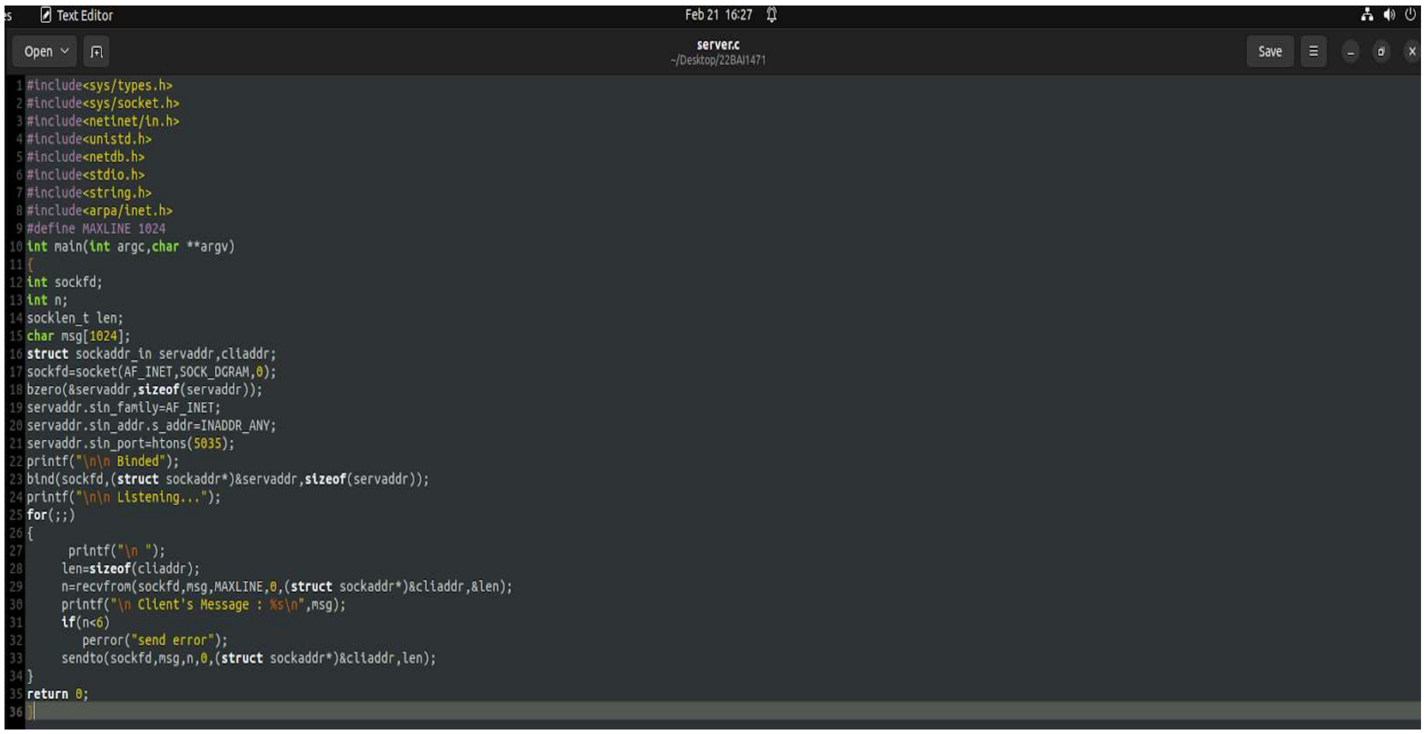
Aim

To write a c program for echo client-server application (UDP protocol) and execute in Linux environment.

Algorithm or Procedure

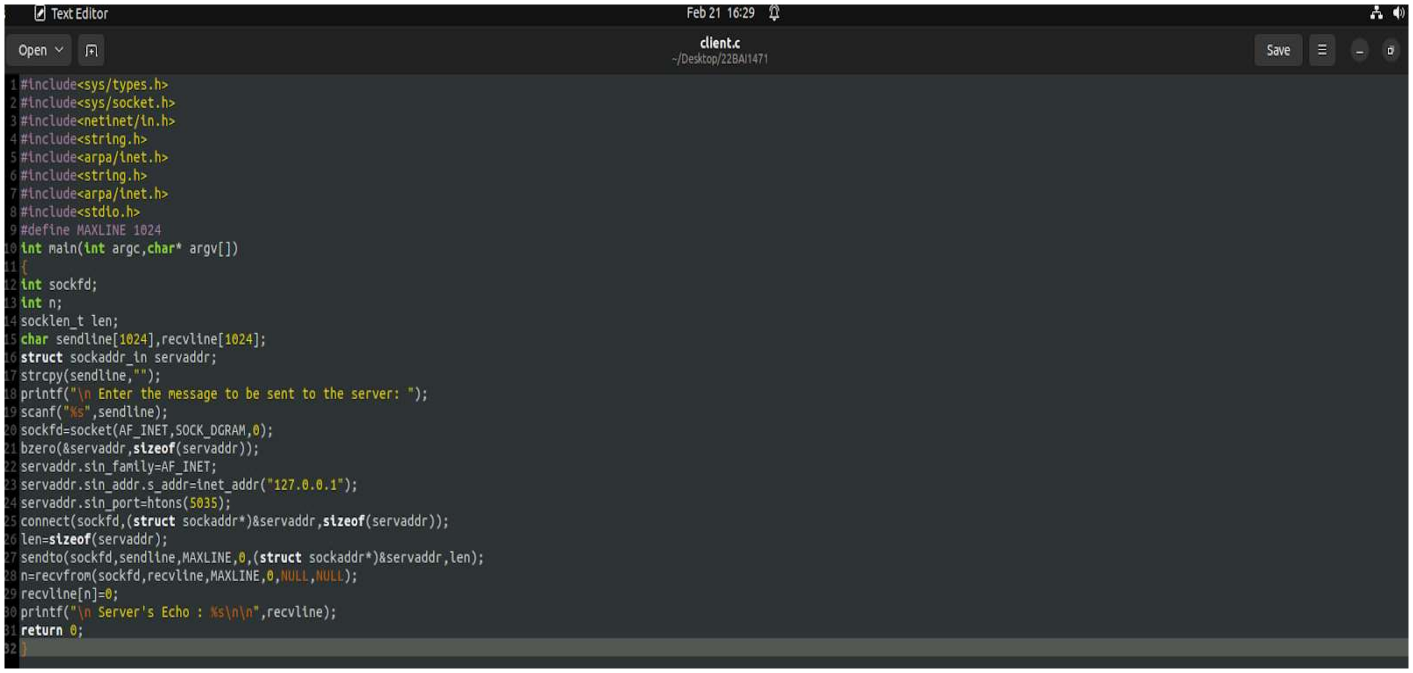
1. Start
2. Writing client and server files separately using socket programming and by using User Datagram Protocol(UDP)
3. Create a UDP socket
4. Assign a port to the socket
5. Communicate with server and client simultaneously
6. Close the socket
7. Execute client and server files separately or simultaneously in two parallel linux terminal windows.
8. Giving the message that is to be return back by server through client.
9. Obtaining desired output
10. Stop

Server side program:



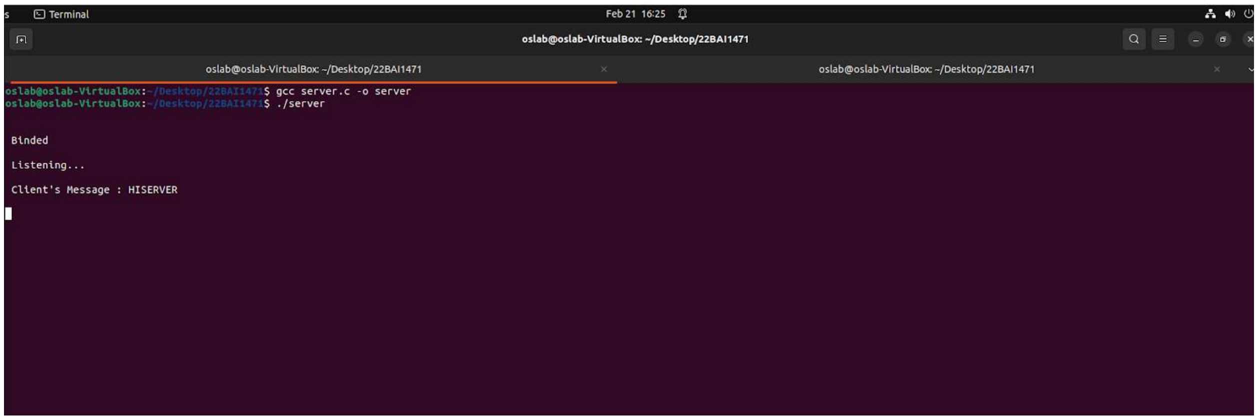
```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<netdb.h>
#include<stdio.h>
#include<string.h>
#include<arpa/inet.h>
#define MAXLINE 1024
int main(int argc,char **argv)
{
int sockfd;
int n;
socklen_t len;
char msg[1024];
struct sockaddr_in servaddr,cliaddr;
sockfd=socket(AF_INET,SOCK_DGRAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(5035);
printf("\n\n Binded");
bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
printf("\n\n Listening...");
for(;;)
{
      printf("\n ");
     len=sizeof(cliaddr);
     n=recvfrom(sockfd,msg,MAXLINE,0,(struct sockaddr*)&cliaddr,&len);
     printf("\n Client's Message : %s\n",msg);
     if(n<6)
        perror("send error");
     sendto(sockfd,msg,n,0,(struct sockaddr*)&cliaddr,len);
}
return 0;
}
```

Client side programming:



```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<arpa/inet.h>
#include<string.h>
#include<arpa/inet.h>
#include<stdio.h>
#define MAXLINE 1024
int main(int argc,char* argv[])
{
int sockfd;
int n;
socklen_t len;
char sendline[1024],recvline[1024];
struct sockaddr_in servaddr;
strcpy(sendline,"");
printf("\n Enter the message to be sent to the server: ");
scanf("%s",sendline);
sockfd=socket(AF_INET,SOCK_DGRAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
servaddr.sin_port=htons(5035);
connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
len=sizeof(servaddr);
sendto(sockfd,sendline,MAXLINE,0,(struct sockaddr*)&servaddr,len);
n=recvfrom(sockfd,recvline,MAXLINE,0,NULL,NULL);
recvline[n]=0;
printf("\n Server's Echo : %s\n\n",recvline);
return 0;
}
```

Output at server side


Terminal
Feb 21 16:25
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471

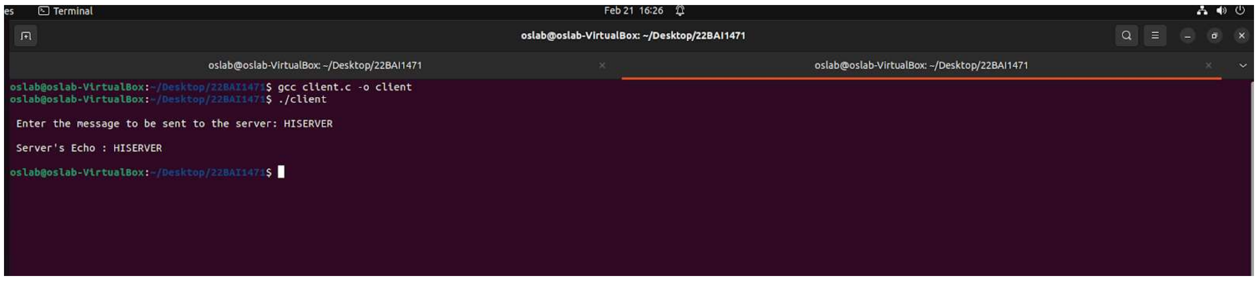
```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ gcc server.c -o server
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./server


 Binded

 Listening...

 Client's Message : HISERVER
```

Output at client side


Terminal
Feb 21 16:26
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471

```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ gcc client.c -o client
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./client

 Enter the message to be sent to the server: HISERVER

 Server's Echo : HISERVER

oslab@oslab-VirtualBox:~/Desktop/22BAI1471$
```

Conclusion

Linux terminal : Output

When a message is entered in the command prompt, the same message is reflected or sent back to the client by the server thus making an echo. This is achieved by using UDP protocol.(no-feedback method).