ASSIGNMENT-4

Name: Sayantan Roy

Roll Number: GCECTB-R18-3025

Subject: Computer Network Lab

Dept: CSE

Year:3rd

ASSIGNMENT -4

UDP ECHO CLIENT SERVER PROGRAM

IN C

UDP ECHO CLIENT SERVER PROGRAM

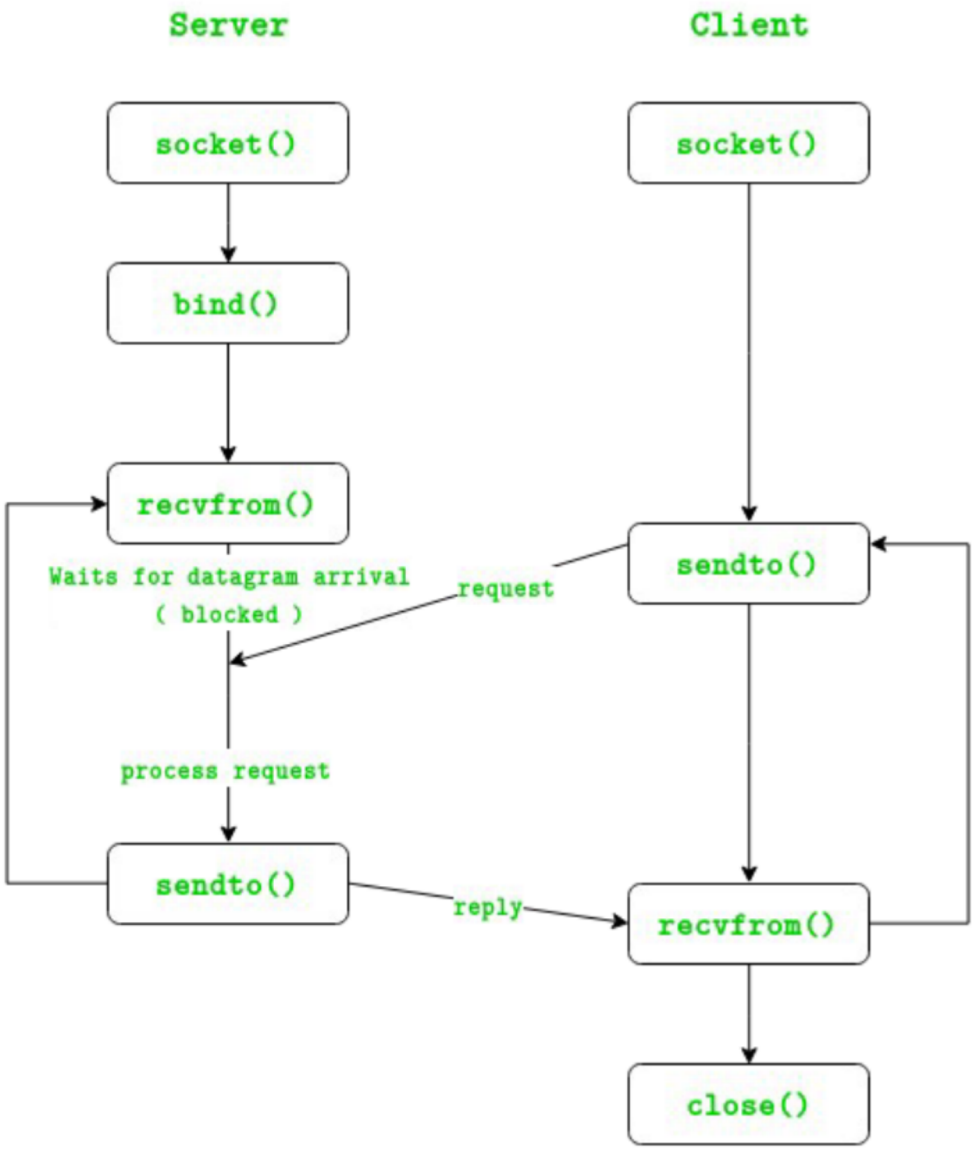
SYSTEM USED :
OPERATING SYSTEM : kali linux 2020.1
KERNEL: 4.19.153
GUI: DEFAULT XSESSION (XFCE4)

system owned and project done by :Sayantan Roy

UDP ECHO CLIENT SERVER PROGRAM

Theory:

In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of sender which the server uses to send data to the correct client.



ALGORITHM:

SERVER:

STEP 1: Start
STEP 2:Declare the variables for the socket
STEP 3: Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Bind the IP address and Port number
STEP 6: Listen and accept the client's request for the connection
STEP 7: Read and Display the client's message
STEP 8: Stop

CLIENT:

STEP 1: Start
STEP 2: Declare the variables for the socket
STEP 3: Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Call the connect() function
STEP 6: Read the input message
STEP 7: Send the input message to the server
STEP 8: Display the server's echo
STEP 9: Close the socket
STEP 10: Stop

Arguments :

domain –Specifies the communication
domain (AF_INET for IPv4/ AF_INET6 for IPv6)

type –Type of socket to be created
(SOCK_STREAM for TCP / SOCK_DGRAM for UDP)

protocol –Protocol to be used by socket.
0 means use default protocol for the address family.

bind : assigns address to the unbound socket

sockfd –File descriptor of socket.

addr –Structure in which address to be binded to is specified

Functions:

socket(int domain, int type, int protocol)

```
Creates an unbound socket in the specified domain.
Returns socket file descriptor.
```

SOURCE CODE:
FILENAME:udpserver.c
SERVER:

```
#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;
}
```

FILENAME:udpclient.c
CLIENT:

```
#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 : ");
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;
}
```

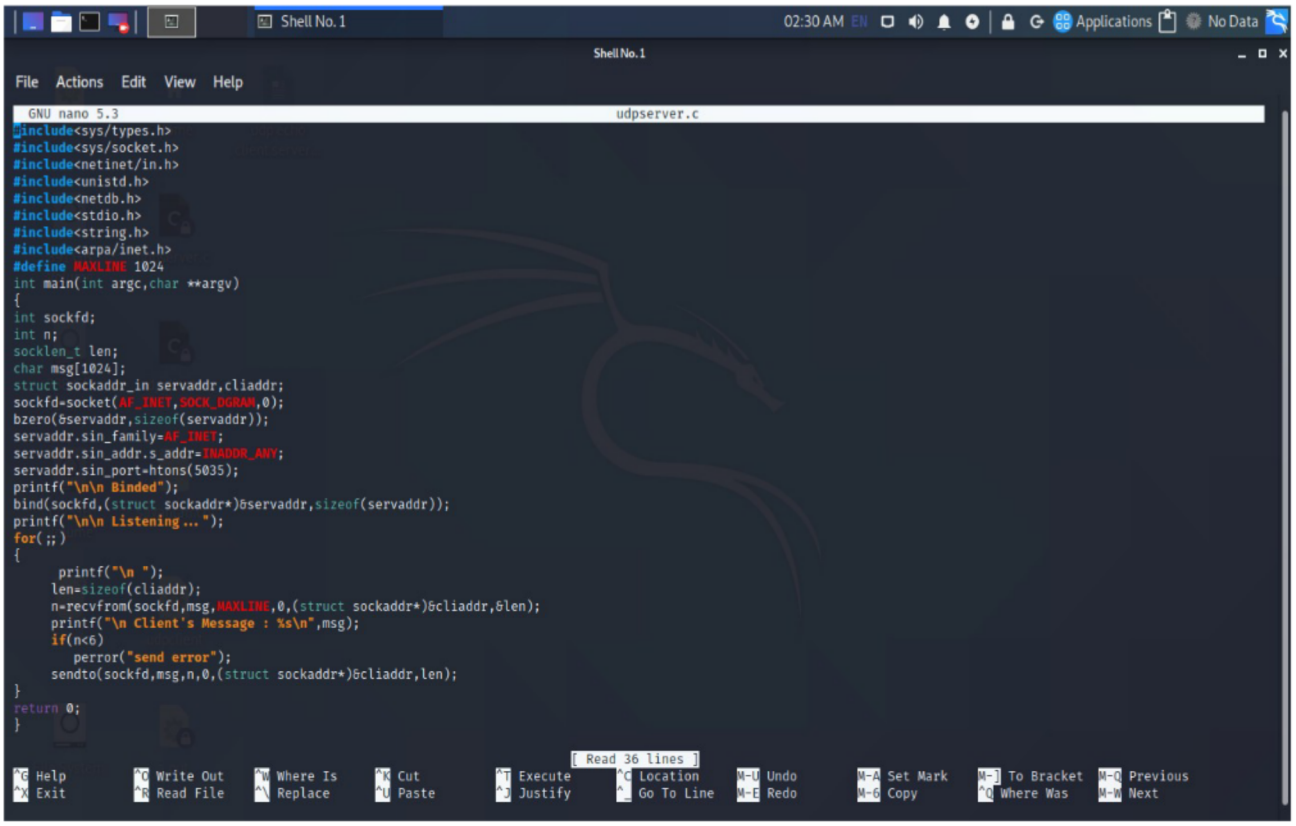


fig: udpserver.c

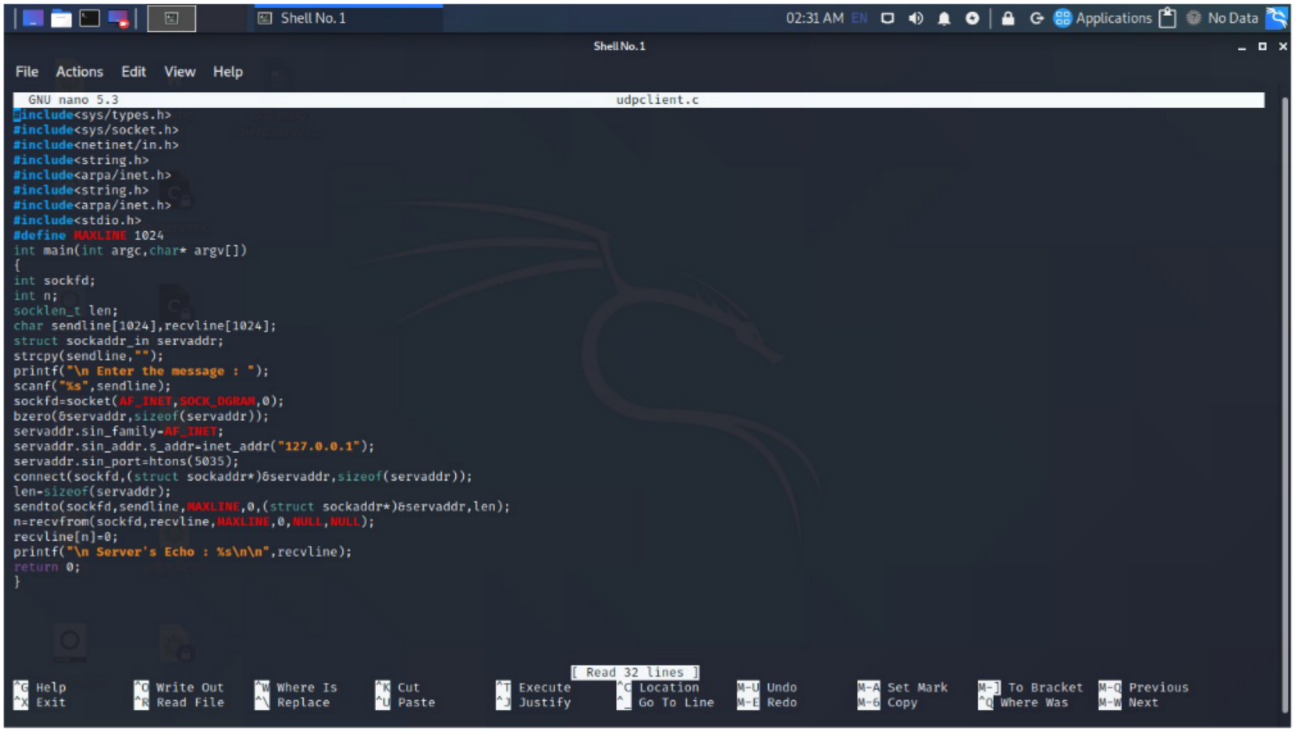


fig:udpclient.c

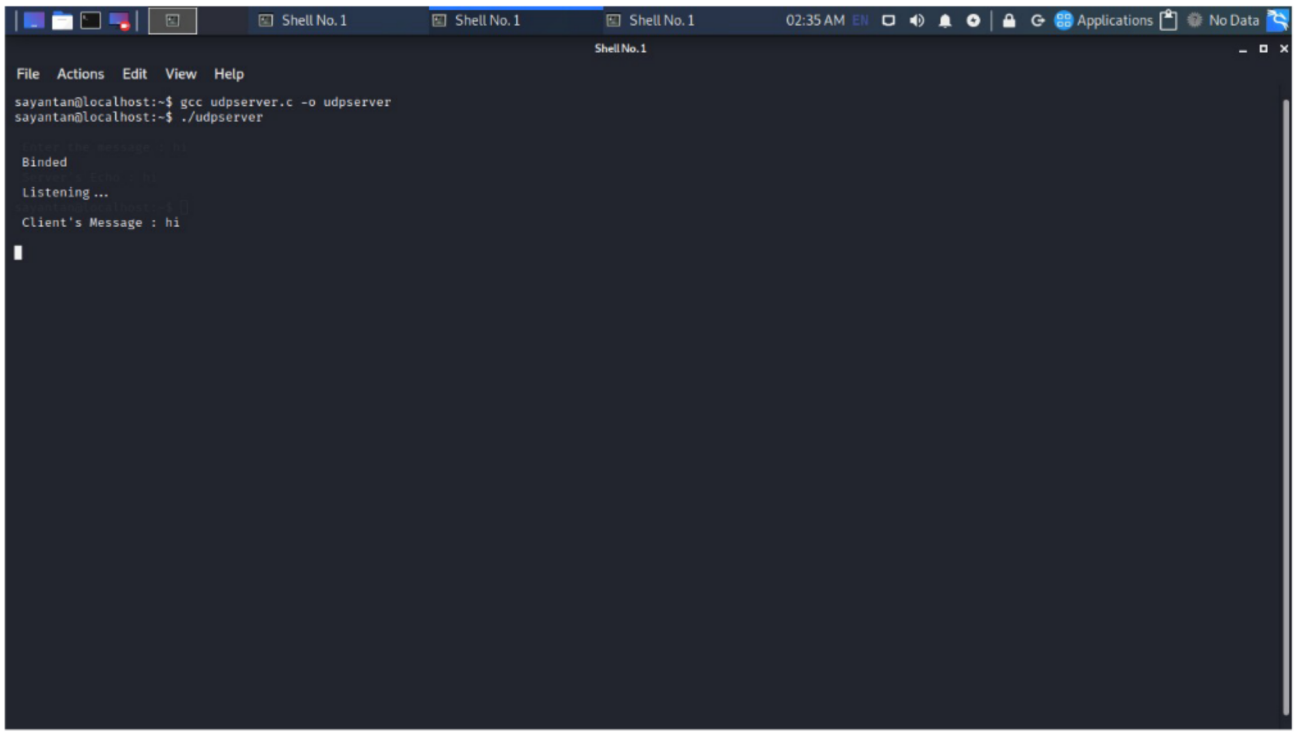


fig : Here server got a message from client.

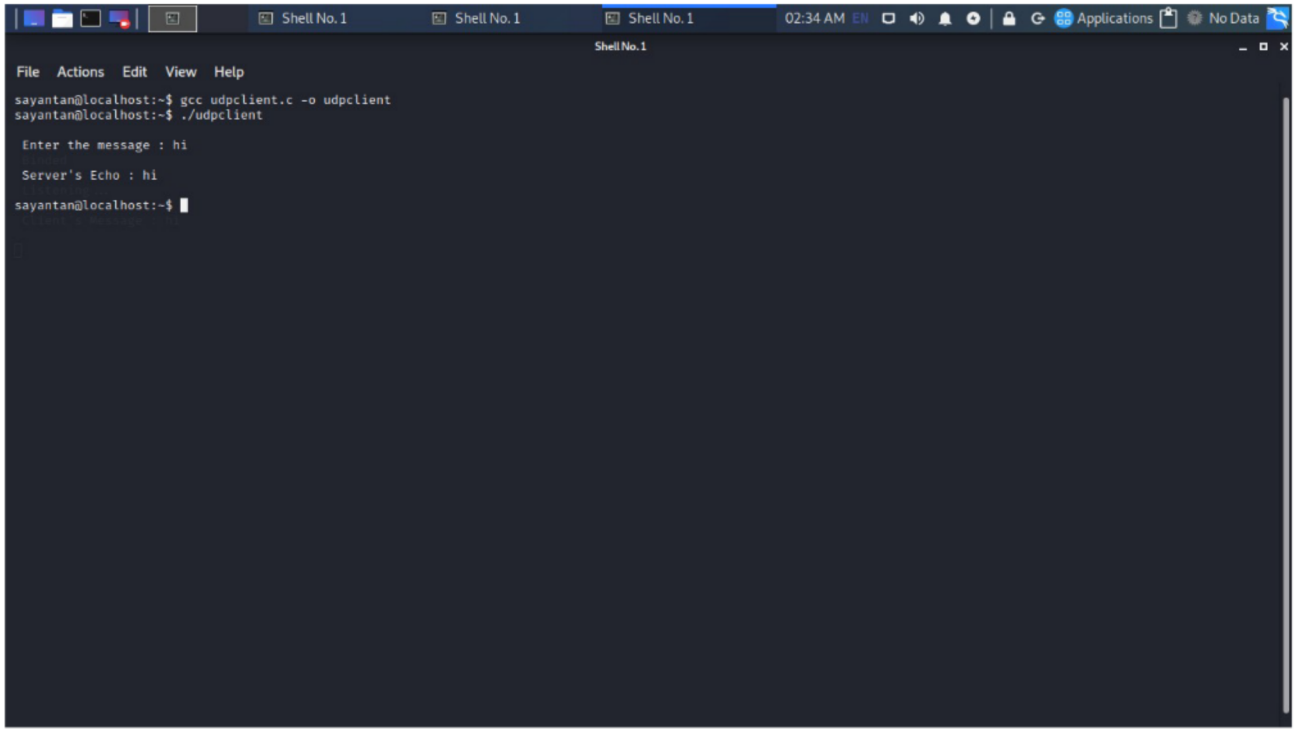


fig: message from client got echo from server