Lab Assignment 4
Network and Communication

Name: Kulvir Singh
Reg. No.: 19BCE2074

Question 1:

Client Server Communication using UDP

Aim :

To implement the client server communication using UDP protocol in linux environment.

Algorithm :

```
UDP Server :
1.Create UDP socket.
2.Bind the socket to server address.
3.Wait until datagram packet arrives from client.
4.Process the datagram packet and send a reply to client.
5.Go back to Step 3.
UDP Client :
1.Create UDP socket.
2.Send message to server.
3.Wait until response from server is recieved.
4.Process reply and go back to step 2, if necessary.
5.Close socket descriptor and exit.
```

Code Text :

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT     8080
```

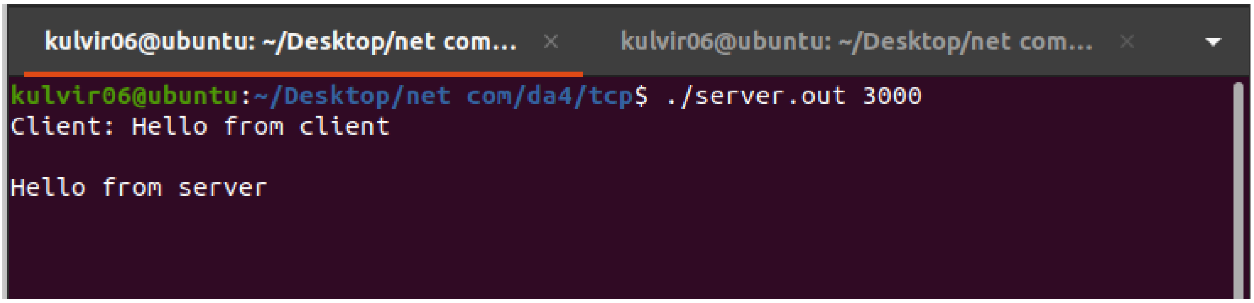
```
#define MAXLINE 1024
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Filling server information
    servaddr.sin_family = AF_INET; // IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Bind the socket with the server address
    if ( bind(sockfd, (const struct sockaddr *)&servaddr,
            sizeof(servaddr)) < 0 )
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    int len, n;

    len = sizeof(cliaddr); //len is value/resuslt

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,
                MSG_WAITALL, ( struct sockaddr *) &cliaddr,
                &len);
    buffer[n] = '\0';
    printf("Client : %s\n", buffer);
    sendto(sockfd, (const char *)hello, strlen(hello),
        MSG_CONFIRM, (const struct sockaddr *) &cliaddr,
            len);
    printf("Hello message sent.\n");

    return 0;
}
```

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#define PORT     8080
#define MAXLINE 1024
int main() {
        int sockfd;
        char buffer[MAXLINE];
        char *hello = "Hello from client";
        struct sockaddr_in         servaddr;

        // Creating socket file descriptor
        if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
                perror("socket creation failed");
                exit(EXIT_FAILURE);
        }

        memset(&servaddr, 0, sizeof(servaddr));

        // Filling server information
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(PORT);
        servaddr.sin_addr.s_addr = INADDR_ANY;

        int n, len;

        sendto(sockfd, (const char *)hello, strlen(hello),
                MSG_CONFIRM, (const struct sockaddr *) &servaddr,
                        sizeof(servaddr));
        printf("Hello message sent.\n");

        n = recvfrom(sockfd, (char *)buffer, MAXLINE,
                                MSG_WAITALL, (struct sockaddr *) &servaddr,
                                &len);
        buffer[n] = '\0';
        printf("Server : %s\n", buffer);

        close(sockfd);
        return 0;
}
```

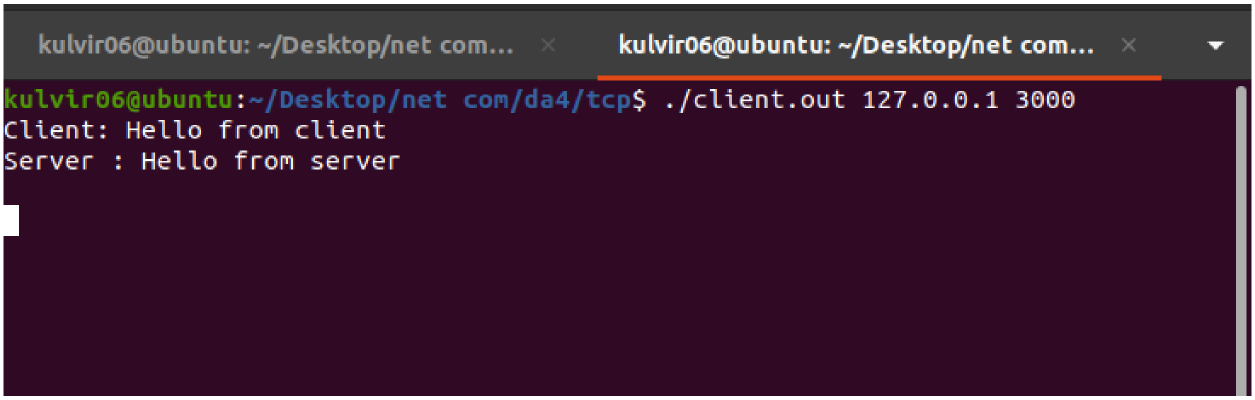
Output Screenshots :

Server:


kulvir06@ubuntu: ~/Desktop/net com...
kulvir06@ubuntu: ~/Desktop/net com...

```
kulvir06@ubuntu:~/Desktop/net com/da4/tcp$ ./server.out 3000
Client: Hello from client

Hello from server
```

Client:


kulvir06@ubuntu: ~/Desktop/net com...
kulvir06@ubuntu: ~/Desktop/net com...

```
kulvir06@ubuntu:~/Desktop/net com/da4/tcp$ ./client.out 127.0.0.1 3000
Client: Hello from client
Server : Hello from server
```

Question 2:

CHAT Application Using TCP

Aim :

To implement a chat application using TCP protocol in linux environment

Algorithm :

TCP Server –

using create(), Create TCP socket.

using bind(), Bind the socket to server address.

using listen(), put the server socket in a passive mode, where it waits for the client to approach the server to make a connection using accept(), At this point, connection is established between client and server, and they are ready to transfer data.

Go back to Step 3.

TCP Client –

Create TCP socket.

connect newly created client socket to server.

Code Text :

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
void error(const char *msg)
{
perror(msg);
exit(1);
}
int main(int argc, char *argv[])
{
  int socket1, newsocket1, portNum;
  socklen_t clilen;
  char buffer[255];
  struct sockaddr_in serv_addr, cli_addr;
  int n;
  if (argc < 2) {
  fprintf(stderr,"ERROR, port not given\n");
  exit(1);
  }
  socket1 = socket(AF_INET, SOCK_STREAM, 0);
  if (socket1 < 0)
  error("ERROR can't open socket");
  bzero((char *) &serv_addr, sizeof(serv_addr));
  portNum = atoi(argv[1]);

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = INADDR_ANY;
  serv_addr.sin_port = htons(portNum);
  if (bind(socket1, (struct sockaddr *) &serv_addr,
  sizeof(serv_addr)) < 0)
  error("ERROR can't bind");
```

```
    listen(socket1,5);
    clilen = sizeof(cli_addr);
    newsocket1 = accept(socket1,
    (struct sockaddr *) &cli_addr,
    &clilen);
    if (newsocket1 < 0)
    error("ERROR can't accept");
    while(1)
    {
    bzero(buffer,255);
    n = read(newsocket1,buffer,255);
    if (n < 0) error("ERROR ,can't read");
    printf("Client: %s\n",buffer);
    bzero(buffer,255);
    fgets(buffer,255,stdin);
    n = write(newsocket1,buffer,strlen(buffer));
    if (n < 0) error("ERROR ,can't write");
    int i=strncmp("Goodbye" , buffer, 7);
    if(i == 0)
    break;
    }
    close(newsocket1);
    close(socket1);
    return 0;
}
```

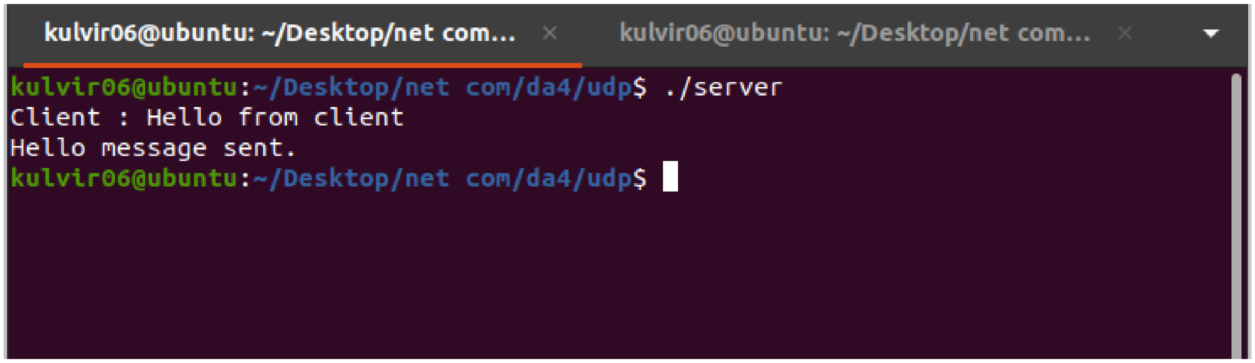
Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
void error(const char *msg)
{
perror(msg);
exit(0);
}
int main(int argc, char *argv[])
```

```
{
int socket1, portnum, n;
struct sockaddr_in serv_addr;
struct hostent *server;
char buffer[256];
if (argc < 3) {
fprintf(stderr,"usage %s hostname port\n", argv[0]);
exit(0);
}
portnum = atoi(argv[2]);
socket1 = socket(AF_INET, SOCK_STREAM, 0);
if (socket1 < 0)
error("ERROR, can't open socket");
server = gethostbyname(argv[1]);
if (server == NULL) {
fprintf(stderr,"ERROR, there exists no such host\n");

exit(0);
}
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr,
(char *)&serv_addr.sin_addr.s_addr,
server->h_length);
serv_addr.sin_port = htons(portnum);
if (connect(socket1,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
error("ERROR,can't connect");
printf("Client: ");
while(1)
{
bzero(buffer,256);
fgets(buffer,255,stdin);
n = write(socket1,buffer,strlen(buffer));
if (n < 0)
error("ERROR, writing to socket");
bzero(buffer,256);
n = read(socket1,buffer,255);
if (n < 0)
error("ERROR, reading from socket");
printf("Server : %s\n",buffer);
int i = strncmp("Goodbye" , buffer , 7);
if(i == 0)
break;
}
}
```

Output Screenshots :

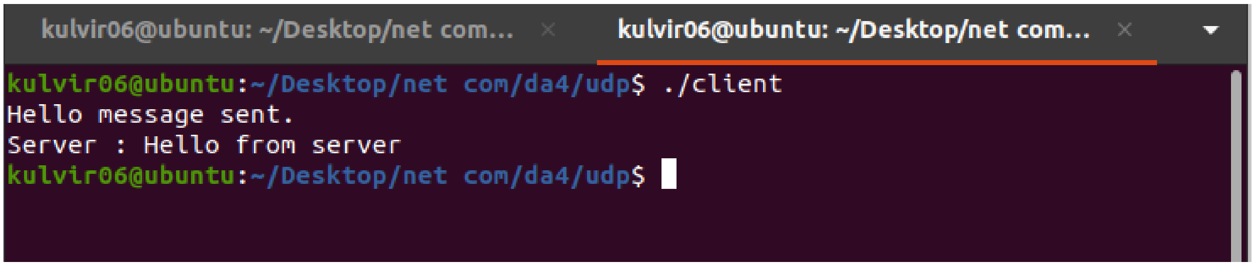
Server:



kulvir06@ubuntu: ~/Desktop/net com...
kulvir06@ubuntu: ~/Desktop/net com...

```
kulvir06@ubuntu:~/Desktop/net com/da4/udp$ ./server
Client : Hello from client
Hello message sent.
kulvir06@ubuntu:~/Desktop/net com/da4/udp$ 
```

Client:



kulvir06@ubuntu: ~/Desktop/net com...
kulvir06@ubuntu: ~/Desktop/net com...

```
kulvir06@ubuntu:~/Desktop/net com/da4/udp$ ./client
Hello message sent.
Server : Hello from server
kulvir06@ubuntu:~/Desktop/net com/da4/udp$ 
```

Question 3:

Security Protocol

At Sender Side:

Plaintext + Key=> Cipher Text

At Receiver Side:

Cipher Text-Key=> Plain Text

Aim :

To implement a Cesar Cipher encryption and decryption program in C which takes key and message as input and encodes and decodes the same.

Algorithm :

Traverse the given text one character at a time .

For each character, transform the given character as per the rule, depending on whether we're encrypting or decrypting the text.

Return the new string generated.

Code Text :

Encryption.c

```
#include<stdio.h>
int main()
{
        char message[100], ch;
        int i, key;
        printf("Enter a message to encrypt: ");
        gets(message);
        printf("Enter key: ");
        scanf("%d", &key);
        for(i = 0; message[i] != '\0'; ++i){
                ch = message[i];
                if(ch >= 'a' && ch <= 'z'){
                        ch = ch + key;
                        if(ch > 'z'){
                                ch = ch - 'z' + 'a' - 1;
                        }
                        message[i] = ch;
                }
                else if(ch >= 'A' && ch <= 'Z'){
                        ch = ch + key;

                        if(ch > 'Z'){
                                ch = ch - 'Z' + 'A' - 1;
                        }
                        message[i] = ch;
                }
        }
        printf("Encrypted message: %s\n", message);
        return 0;
}
```

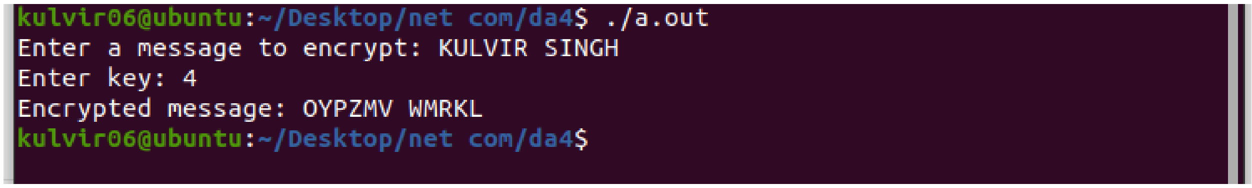
Decryption.c

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

```
int main()
{
    char message[100], ch;
    int i, key;
    printf("Enter a message to decrypt: ");
    gets(message);
    printf("Enter key: ");
    scanf("%d", &key);
    for(i = 0; message[i] != '\0'; ++i){
        ch = message[i];
        if(ch >= 'a' && ch <= 'z'){
            ch = ch - key;
            if(ch < 'a'){
                ch = ch + 'z' - 'a' + 1;
            }
            message[i] = ch;
        }
        else if(ch >= 'A' && ch <= 'Z'){
            ch = ch - key;

            if(ch < 'A'){
                ch = ch + 'Z' - 'A' + 1;
            }
            message[i] = ch;
        }
    }

    printf("Decrypted message: %s\n", message);
    return 0;
}
```

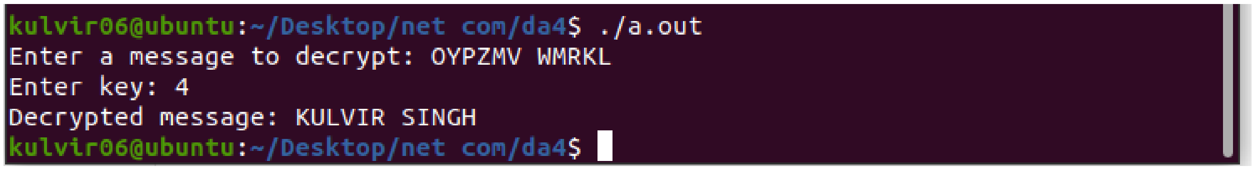
Output Screenshots :

Encryption :



```
kulvir06@ubuntu:~/Desktop/net com/da4$ ./a.out
Enter a message to encrypt: KULVIR SINGH
Enter key: 4
Encrypted message: OYPZMV WMRKL
kulvir06@ubuntu:~/Desktop/net com/da4$
```

Decryption :



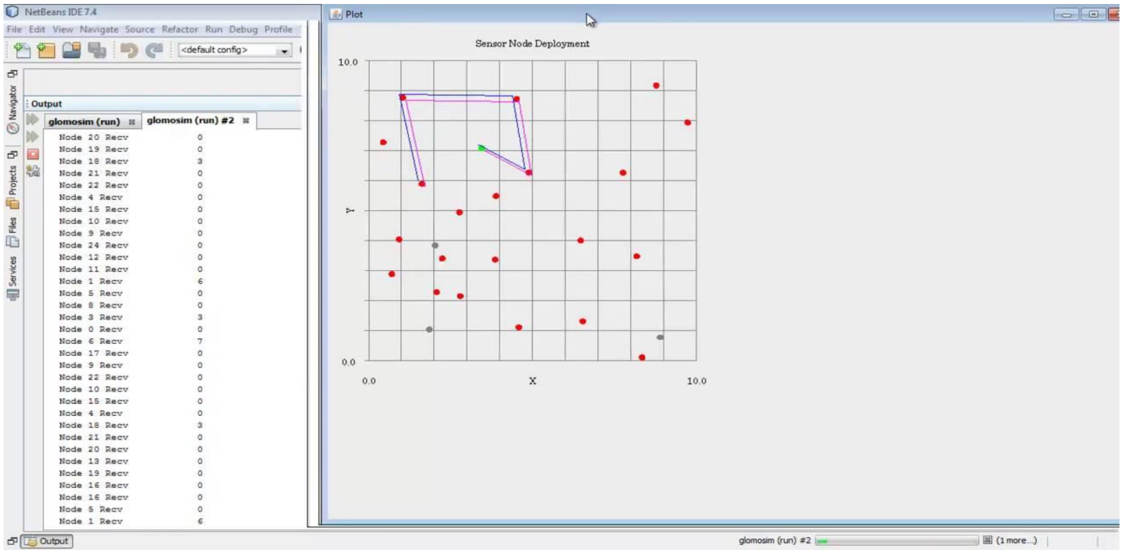
```
kulvir06@ubuntu:~/Desktop/net com/da4$ ./a.out
Enter a message to decrypt: OYPZMV WMRKL
Enter key: 4
Decrypted message: KULVIR SINGH
kulvir06@ubuntu:~/Desktop/net com/da4$
```

4. Study about any one Simulations Tools/Network Software

GloMoSim

Global Mobile Information System Simulator (GloMoSim) is a network protocol simulation software that simulates wireless and wired network systems. GloMoSim is designed using the parallel discrete event simulation capability provided by Parsec, a parallel programming language. GloMoSim currently supports protocols for a purely wireless network. It uses the Parsec compiler to compile the simulation protocols. Parsec is a C-based simulation language, developed by the Parallel Computing Laboratory at UCLA, for sequential and parallel execution of discrete-event simulation models.

A number of library based parallel and sequential network simulators have been designed. The paper describes a library, called GloMoSim (Global Mobile system Simulator), for parallel simulation of wireless networks. GloMoSim has been designed to be extensible and composable: the communication protocol stack for wireless networks is divided into a set of layers, each with its own API. Models of protocols at one layer interact with those at a lower (or higher) layer only via these APIs. The modular implementation enables consistent comparison of multiple protocols at a given layer. The parallel implementation of GloMoSim can be executed using a variety of conservative synchronization protocols, which include the null message and conditional event algorithms. The paper describes the GloMoSim library, addresses a number of issues relevant to its parallelization, and presents a set of experimental results on the IBM 9076 SP, a distributed memory multicomputer. These experiments use models constructed from the library modules. A model project in GloMoSim workspace looks like this.



```
Node 20 Recv          0
Node 19 Recv          0
Node 18 Recv          3
Node 21 Recv          0
Node 22 Recv          0
Node 4 Recv           0
Node 15 Recv          0
Node 10 Recv          0
Node 9 Recv           0
Node 24 Recv          0
Node 12 Recv          0
Node 11 Recv          0
Node 1 Recv           6
Node 5 Recv           0
Node 8 Recv           0
Node 3 Recv           3
Node 0 Recv           0
Node 6 Recv           7
Node 17 Recv          0
Node 9 Recv           0
Node 22 Recv          0
Node 10 Recv          0
Node 15 Recv          0
Node 4 Recv           0
Node 18 Recv          3
Node 21 Recv          0
Node 20 Recv          0
Node 13 Recv          0
Node 19 Recv          0
Node 16 Recv          0
Node 16 Recv          0
Node 5 Recv           0
Node 1 Recv           6
```