

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CSB4241 – COMPUTER NETWORKING LABORATORY**

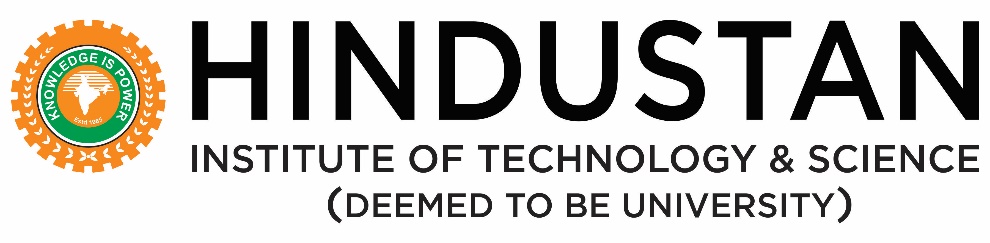
NAME : KONDREDDY SUMANTH KUMAR REDDY

REGISTER NUMBER : 19113105

SEMESTER : **IV**

YEAR : **II**

BRANCH/SECTION : **CSE - B**



**LABORATORY RECORD**

19113105

**REG NO:**

Name of the lab  **COMPUTER NETWORKING**  in the Department of **CSE**

Certified that this is a bonafide record of the work done by

**K. Sumanth Kumar Reddy** of **CSE 3B** Class in the

**virtual** laboratory during the year **2020 - 21**

……………………………

Signature of

Staff-in-charge

………………………….... …………………………...

Internal Examiner External Examiner

Name of the Examination: **University Practical Lab Examination**

Register No. : **19113105**

Date of the Examination : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Date** | **Name of the**  **Experiment** | **Pages** |
| 1 | 18/02/2021 | Simulation of ARP/RARP | 04-07 |
| 2 | 04/03/2021  11/03/2021 | a) Bit-stuffing  b) CRC | 08-14 |
| 3 | 16/03/2021 | Client and Server chat Application | 15-19 |
| 4 | 25/03/2021 | Simulation of BGP / OSPF routing protocol | 20-22 |
| 5 | 01/04/2021 | Sliding Window Protocol | 23-28 |
| 6 | 10/05/2021 | Develop a Client that contacts a given DNS Server to resolve a given host name | 29-31 |
| 7 | 06/04/2021 | Write a Client to download a file from a HTTP Server. | 32-36 |
| 8 | 06/05/2021 | Case study on a network simulator. | 37-39 |

**Program No:** 1

**AIM:**  
         To write a program for Simulation of ARP/RARP.  
  
**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: Establish the connection with the client  
    STEP 8: Display the file name  
    STEP 9: Dispaly the input message  
    STEP 10: Dispaly the output message after bit stuffing  
    STEP 11: Close the socket  
    STEP 12: 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 file name  
    STEP 7: Send the file name to the server  
    STEP 8: Close the socket  
    STEP 9: Stop

**SOURCE CODE:  
  
SERVER:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/shm.h>

#include<string.h>

int main()

{

int shmid, a, i;

char \*ptr, \*shmptr;

shmid=shmget(1000,10,IPC\_CREAT | 0666);

shmptr=shmat(shmid,NULL,0);

ptr=shmptr;

for(i=0;i<3;i++)

{

puts("enter the mac");

scanf("%s",ptr);

a=strlen(ptr);

printf("string length:%d",a);

ptr[a]= ' ' ;

puts("\nenter ip");

ptr=ptr+a+1;

scanf("%s",ptr);

ptr[a]='\n' ;

ptr= ptr+a+1;

}

ptr[strlen(ptr)]= '\0';

printf("\n ARP table at serverside is=\n%s", shmptr);

}

**CLIENT:**

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<sys/shm.h>

#include<stdlib.h>

int main()

{

int shmid,a;

char \*ptr, \*shmptr;

char ptr2[51], ip[12], mac[26];

shmid=shmget(1000,10,0666);

shmptr=shmat(shmid,NULL,0);

puts("the arp table is");

printf("%s",shmptr);

printf("\n1.ARP\n2.RARP\n3.EXIT\n");

scanf("%d",&a);

switch(a)

{

case 1:

puts("enter ip address");

scanf("%s",ip);

ptr=strstr(shmptr, ip);

ptr-=4;

sscanf(ptr,"%s%\*s",ptr2);

printf("mac address is %s",ptr2);

case 2:

puts("\nenter mac address");

scanf("%s",mac);

ptr=strstr(shmptr, mac);

sscanf(ptr,"%\*s%s",ptr2);

printf("ip address is %s\n",ptr2);

case 3:

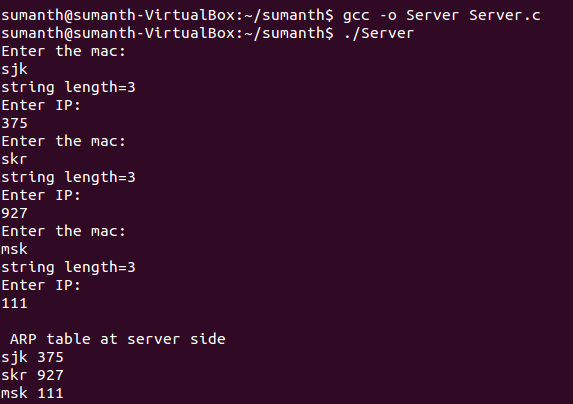
exit(1);

}

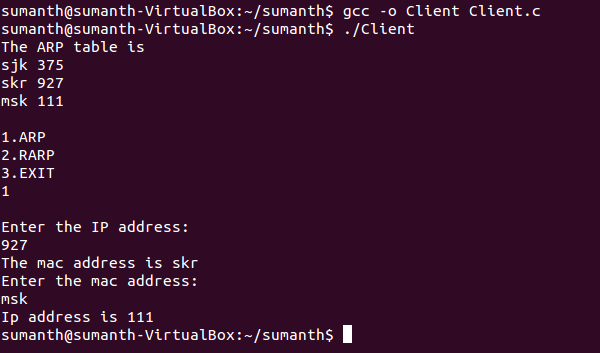
}

**OUTPUT:**

**SERVER:**

****

**CLIENT:**

****

**RESULT:**

Thus the program for Simulation of ARP/RARP was executed and the output was verified.

**Program No:** 2a

**AIM:**  
         To write a program for bit stuffing.  
  
**ALGORITHM:**

**SERVER:**  
    STEP 1: Start the process  
    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: Establish the connection with the client  
    STEP 8: Display the file name  
    STEP 9: Display the input message  
    STEP 10: Display the output message after bit stuffing  
    STEP 11: Close the socket  
    STEP 12: Stop the process

**CLIENT:**  
    STEP 1: Start the process  
    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 file name  
    STEP 7: Send the file name to the server  
    STEP 8: Close the socket  
    STEP 9: Stop the process  
  
**SOURCE CODE:**

**SERVER:**

#include<stdio.h>

#include<arpa/inet.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<netdb.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#define SERV\_TCP\_PORT 5035

#define MAX 60

void stuff();

int i,j,tem;

char buff[4096],t;

FILE \*f1;

int main(int arg,char \*\*argv)

{

int sockfd,newsockfd;

unsigned int clength;

struct sockaddr\_in serv\_addr,cli\_addr;

char t[MAX],str[MAX];

strcpy(t,"exit");

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=INADDR\_ANY;

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT);

printf("\nBinded");

bind(sockfd,(struct sockaddr\*)&serv\_addr,sizeof(serv\_addr));

printf("\nListening...\n");

listen(sockfd,5);

clength=(sizeof(cli\_addr));

newsockfd=accept(sockfd,(struct sockaddr\*)&cli\_addr,&clength);

close(sockfd);

read(newsockfd,&str,MAX);

printf("\nClient Message \nFile Name:\t%s\n",str);

f1 =fopen(str,"r");

while(fgets(buff,4096,f1)!=NULL)

{

stuff();

write(newsockfd,buff,MAX);

printf("\n");

}

fclose(f1);

return 0;

}

void stuff()

{

i=j=tem=0;

printf("\nInput : %s",buff);

printf("\nOutput : ");

do

{

if(buff[i]=='1')

{

if(j==4)

{

tem=i-4;

t='0';

for(;tem<=i;tem++)

printf("%c",buff[tem]);

j=-1;

printf("%c",t);

}

i+=1;

j+=1;

}

else

{

tem=i;

tem=tem-j;

for(;tem<=i;tem++)

printf("%c",buff[tem]);

i+=1;

j=0;

}

}while(buff[i]!='\0');

}

**CLIENT:**  
#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<netdb.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#define SERV\_TCP\_PORT 5035

#define MAX 60

int main(int argc,char \*\*argv)

{

int sockfd,n;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

char send[MAX],recvline[MAX],s[MAX],name[MAX];

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=INADDR\_ANY;

serv\_addr.sin\_port=htons(SERV\_TCP\_PORT);

connect(sockfd,(struct sockaddr\*)&serv\_addr,sizeof(serv\_addr));

printf("\nEnter the source file name:\n");

scanf("%s",send);

write(sockfd,send,MAX);

while((n=read(sockfd,recvline,MAX))!=0)

{

printf("%s",recvline);

}

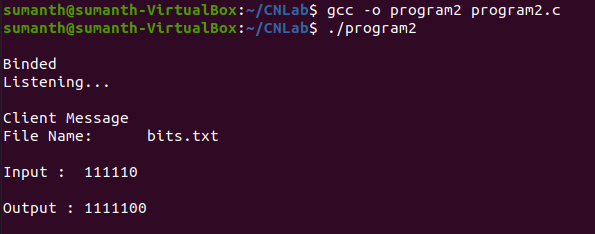
close(sockfd);

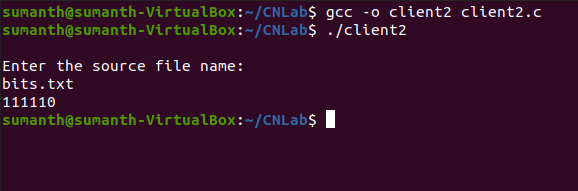
return 0;

}

**OUTPUT:**

**SERVER:**



**CLIENT:  
**

**RESULT:**

Thus the program for bit stuffing was executed and the output was verified.

**Program No:** 2b

**AIM:**  
         To write a program for CRC(Circular Redundancy Check).  
  
**ALGORITHM:**

**Calculation of CRC at Sender Side:**

1) A string of n 0’s is appended to the data unit to be transmitted.

2) Here, n is one less than the number of bits in CRC generator.

3) Binary division is performed of the resultant string with the CRC generator.

4) After division, the remainder so obtained is called as CRC.

5) It may be noted that CRC also consists of n bits.

**Appending CRC To Data Unit at sender side:**

1. The CRC is obtained after the binary division.
2. The string of n 0’s appended to the data unit earlier is replaced by the CRC remainder.

**Transmission To Receiver:**

* The newly formed code word (Original data + CRC) is transmitted to the receiver.

**Checking at Receiver Side:**

1. The transmitted code word is received.
2. The received code word is divided with the same CRC generator.
3. On division, the remainder so obtained is checked.

The following two cases are possible-

**Case-01: Remainder = 0**

1. Receiver assumes that no error occurred in the data during the transmission.
2. Receiver accepts the data.

**Case-02: Remainder ≠ 0**

1. Receiver assumes that some error occurred in the data during the transmission.

Receiver rejects the data and asks the sender for retransmission.

**SOURCE CODE:**

#include<stdio.h>

#include<math.h>

#include<string.h>

int main()

{

int i,j,k,m,n,cl;

char a[10],b[100],c[100];

printf("\n Enter the Original Message:");

scanf("%s",a);

printf("\n Enter the crc to be checked:");

scanf("%s",b);

m=strlen(a);

n=strlen(b);

for(i=0;i<m;i++) /\* To eliminate first zeros in polynomial \*/

{

if(a[i]=='1')

{

m=m-i;

break;

}

}

for(i=0;i<n;i++) /\* To copy the original frame to c[]\*/

c[i]=b[i];

for(i=n;i<cl;i++) /\* To add n-1 zeros at the end of frame \*/

c[i]='0';

c[i]='\0'; /\*To make it as a string \*/

for(i=0;i<n;i++) /\* To set polynomial remainder at end of c[]\*/

if(c[i]=='1')

{

for(j=i,k=0;k<m;k++,j++)

if(a[k]==c[j])

c[j]='0';

else c[j]='1';

}

for(i=0;i<n;i++) /\* To copy original data in c[] \*/

c[i]=b[i];

printf("\n THE MESSAGE IS: %s",c);

if(c[i]=='1')

{

printf("\n THERE IS SOME ERROR IN MESSAGE\n");

}

else

{

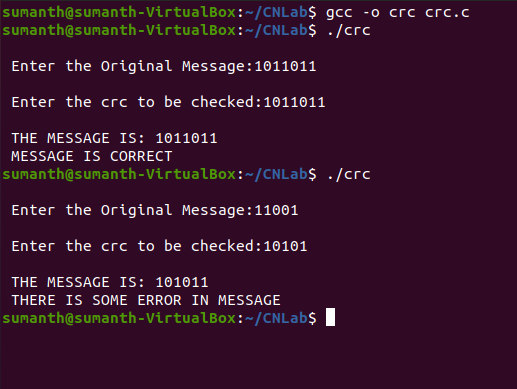
printf("\n MESSAGE IS CORRECT \n");

}

return 0;

}

**OUTPUT:**

****

**Result:**

Thus the program for crc was executed and the output was verified.

**Program No:** 3

**AIM:**  
         To write a program for Client and Server chat application.  
  
**ALGORITHM:**

**SERVER:**

STEP 1: Start the program.  
STEP 2: Declare the variables and structure for the socket.  
STEP 3: Create a socket using socket functions  
STEP 4: The socket is binded at the specified port.  
STEP 5: Using the object the port and address are declared.  
STEP 6: If the binding is successful write the message to the client.  
STEP 7: Close the socket if the client sends a goodbye message.

STEP 8: Execute the client program.

**CLIENT:**

STEP 1: Start the program.  
STEP 2: Declare the variables and structure.  
STEP 3: Socket is created and connects function is executed.  
STEP 4: If the connection is successful then server sends the message.  
STEP 5: The message from the server is responded by the client.  
STEP 6: Stop the program.

**SOURCE CODE:**

**SERVER:**

#include<sys/socket.h>

#include<stdio.h>

#include<string.h>

#include<netdb.h>

#include<stdlib.h>

int main()

{

char buf[100];

int k;

socklen\_t len;

int sock\_desc,temp\_sock\_desc;

struct sockaddr\_in server,client;

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

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

sock\_desc=socket(AF\_INET,SOCK\_STREAM,0);

if(sock\_desc==-1)

{

printf("Error in socket creation");

exit(1);

}

server.sin\_family=AF\_INET;

server.sin\_addr.s\_addr=INADDR\_ANY;

server.sin\_port=3002;

k=bind(sock\_desc,(struct sockaddr\*)&server,sizeof(server));

if(k==-1)

{

printf("Error in binding");

exit(1);

}

k=listen(sock\_desc,20);

if(k==-1)

{

printf("Error in listening");

exit(1);

}

len=sizeof(client);//VERY IMPORTANT

temp\_sock\_desc=accept(sock\_desc,(struct sockaddr\*)&client,&len);

if(temp\_sock\_desc==-1)

{

printf("Error in temporary socket creation");

exit(1);

}

while(1)

{

k=recv(temp\_sock\_desc,buf,100,0);

if(k==-1)

{

printf("Error in receiving");

exit(1);

}

printf("Message got from client is : %s",buf);

printf("\nEnter data to be send to client: ");

fgets(buf,100,stdin);

if(strncmp(buf,"end",3)==0)

break;

k=send(temp\_sock\_desc,buf,100,0);

if(k==-1)

{

printf("Error in sending");

exit(1);

}

}

exit(0);

return 0;

}

**CLIENT:**

#include<sys/socket.h>

#include<stdio.h>

#include<string.h>

#include<netdb.h>

#include<stdlib.h>

int main()

{

char buf[100];

int k;

int sock\_desc;

struct sockaddr\_in client;

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

sock\_desc=socket(AF\_INET,SOCK\_STREAM,0);

if(sock\_desc==-1)

{

printf("Error in socket creation");

exit(1);

}

client.sin\_family=AF\_INET;

client.sin\_addr.s\_addr=INADDR\_ANY;

client.sin\_port=3002;

k=connect(sock\_desc,(struct sockaddr\*)&client,sizeof(client));

if(k==-1)

{

printf("Error in connecting to server");

exit(1);

}

while(1)

{

printf("\nEnter data to be send to server: ");

fgets(buf,100,stdin);

if(strncmp(buf,"end the chat",3)==0)

break;

k=send(sock\_desc,buf,100,0);

if(k==-1)

{

printf("Error in sending");

exit(1);

}

k=recv(sock\_desc,buf,100,0);

if(k==-1)

{

printf("Error in receiving");

exit(1);

}

printf("Message got from server is : %s",buf);

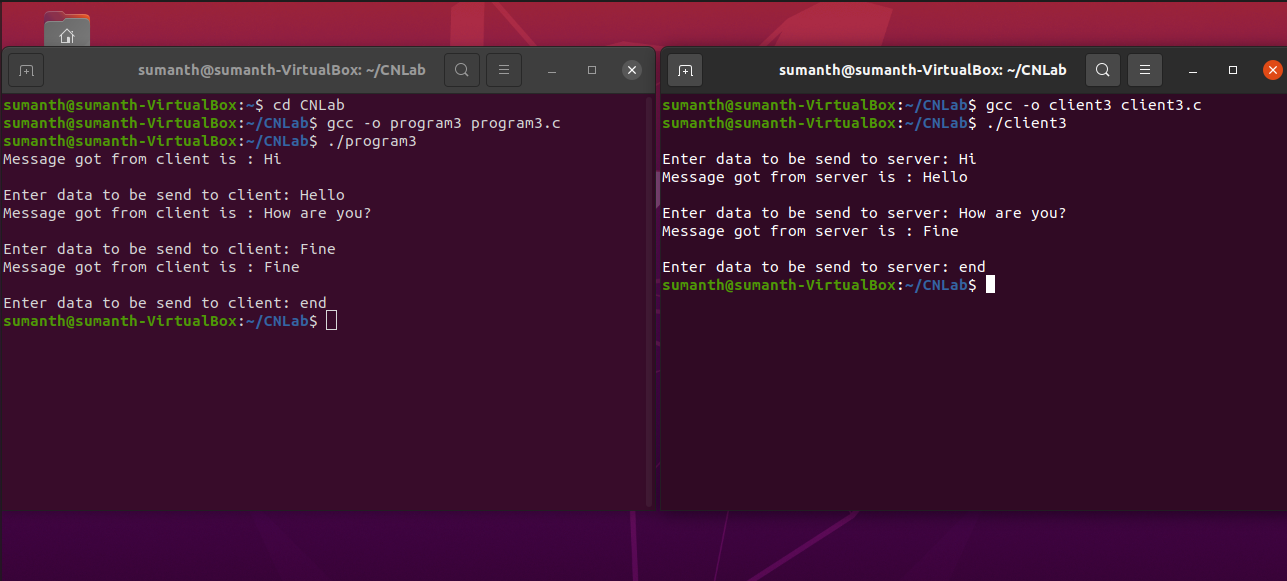
}

exit(0);

return 0;

}

**OUTPUT:**

****

**RESULT:**

Thus the program for Client and Server Chat Application was executed and the output was verified.

**Program No:** 4

**Aim:**  
         To write a program for Simulation of BGP / OSPF routing protocol.  
  
**Algorithm:**

STEP 1: Start the program.  
STEP 2: Read the no. of nodes.  
STEP 3: Read the distance between one node to the other for every node.  
STEP 4: Assign the distance from a node to itself as zero.  
STEP 5: Find the shortest path from the given node to other node and assign the same as distance.  
STEP 6: Print the distance from one node to every other node in form of a matrix.  
STEP 7: Stop.

**Source Code:**

#include <stdio.h>

int main()

{

int n;

int i,j,k;

int a[10][10],b[10][10];

printf("\n Enter the number of nodes:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("\n Enter the distance between the host %d - %d:",i+1,j+1);

scanf("%d",&a[i][j]);

}}

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d\t",a[i][j]);

}

printf("\n");

}

for(k=0;k<n;k++)

{

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

if(a[i][j]>a[i][k]+a[k][j])

{

a[i][j]=a[i][k]+a[k][j];

}}}}

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

b[i][j]=a[i][j];

if(i==j)

{

b[i][j]=0;

}

}}

printf("\n The output matrix:\n");

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d\t",b[i][j]);

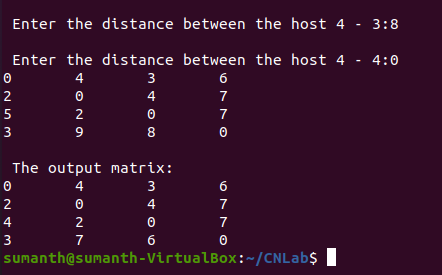
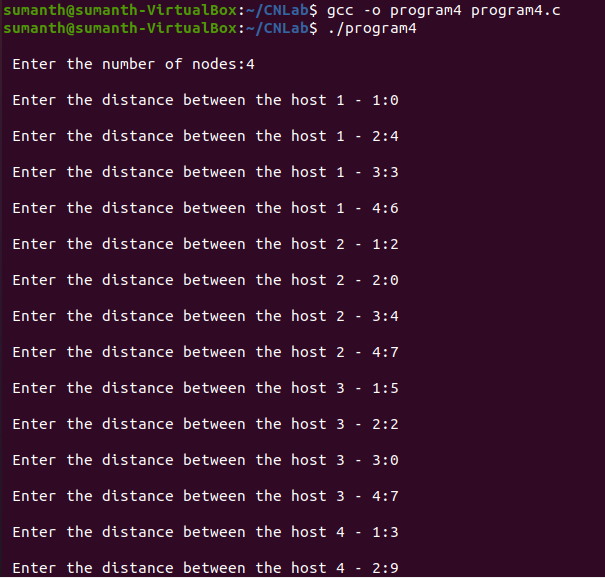
}

printf("\n");

}

}

**OUTPUT:**

****

**RESULT:**

Thus, the program for Simulation of BGP / OSPF routing protocol was executed and the output was verified.

**Program No:** 5

**Aim:**  
         To write a program for Sliding Window protocol.

**Algorithm:**

**Server Side:**  
    STEP 1: Start  
    STEP 2: Declare the variables for the socket  
    STEP 3: Specify the family, protocol, 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: Establish the connection with the client

STEP 8: Implement the variables and function with the while loop to receives the frames for the sender side.

STEP 9: if it receives all the frames close the socket

STEP 10: stop

**Client side:**

STEP 1: Start the program.  
STEP 2: Declare the variables and structure.  
STEP 3: Socket is created and connects function is executed.  
STEP 4: If the connection is successful .

STEP 5: enter the size of frames to be send to receiver.

STEP 6: start sending the frames.

STEP 7: if the frames are sent after time out it needs to get acknowledgement from the receiver that the frame to be resent from the sender side.

if(e<f)

{

printf("\nTime Out, Resent Frame %d onwards",e);

}

Step 8: if all the frames are sent correctly to receiver close the socket

Step 9: stop

**Source Code:**

**Server:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<time.h>

#include<stdlib.h>

#include<ctype.h>

#include<arpa/inet.h>

#define W 5

#define P1 50

#define P2 10

char a[10];

char b[10];

void alpha9(int);

int main()

{

struct sockaddr\_in ser,cli;

int s,sock,i,j,c=1,f;

unsigned int s1,n;

s=socket(AF\_INET,SOCK\_STREAM,0);

ser.sin\_family=AF\_INET;

ser.sin\_port=6500;

ser.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

bind(s,(struct sockaddr \*) &ser, sizeof(ser));

listen(s,1);

n=sizeof(cli);

sock=accept(s,(struct sockaddr \*)&cli, &n);

printf("\nTCP Connection Established.\n");

s1=(unsigned int) time(NULL);

srand(s1);

strcpy(b,"Time Out ");

recv(sock,a,sizeof(a),0);

f=atoi(a);

while(1)

{

for(i=0;i<W;i++)

{

recv(sock,a,sizeof(a),0);

if(strcmp(a,b)==0)

{

break;

}

}

i=0;

while(i<W)

{

j=rand()%P1;

if(j<P2)

{

send(sock,b,sizeof(b),0);

break;

}

else

{

alpha9(c);

if(c<=f)

{

printf("\nFrame %s Received ",a);

send(sock,a,sizeof(a),0);

}

else

{

break;

}

c++;

}

if(c>f)

{

break;

}

i++;

}

}

return 0;

}

void alpha9(int z)

{

int k,i=0,j,g;

k=z;

while(k>0)

{

i++;

k=k/10;

}

g=i;

i--;

while(z>0)

{

k=z%10;

a[i]=k+48;

i--;

z=z/10;

}

a[g]='\0';

}

**Client:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<time.h>

#include<stdlib.h>

#include<ctype.h>

#define W 5

char a[10];

char b[10];

void alpha9(int);

int main()

{

int s,f,wl,c=1,x,i=0,j,n,p=0,e=0;

struct sockaddr\_in ser;

s=socket(AF\_INET,SOCK\_STREAM,0);

ser.sin\_family=AF\_INET;

ser.sin\_port=6500;

connect(s,(struct sockaddr \*) &ser, sizeof(ser));

printf("\nTCP Connection Established.\n");

printf("\nEnter the number of Frames: ");

scanf("%d",&f);

alpha9(f);

send(s,a,sizeof(a),0);

strcpy(b,"Time Out ");

while(1)

{

for(i=0;i<W;i++)

{

alpha9(c);

send(s,a,sizeof(a),0);

if(c<=f)

{

printf("\nFrame %d Sent",c);

c++;

}

}

i=0;

wl=W;

while(i<W)

{

recv(s,a,sizeof(a),0);

p=atoi(a);

if(strcmp(a,b)==0)

{

e=c-wl;

if(e<f)

{

printf("\nTime Out, Resent Frame %d onwards",e);

}

break;

}

else

{

if(p<=f)

{

printf("\nFrame %s Acknowledged",a);

wl--;

}

else

{

break;

}

}

if(p>f)

{

break;

}

i++;

}

if(wl==0 && c>f)

{

send(s,b,sizeof(b),0);

break;

}

else

{

c=c-wl;

wl=W;

}

}

return 0;

}

void alpha9(int z)

{

int k,i=0,j,g;

k=z;

while(k>0)

{

i++;

k=k/10;

}

g=i;

i--;

while(z>0)

{

k=z%10;

a[i]=k+48;

i--;

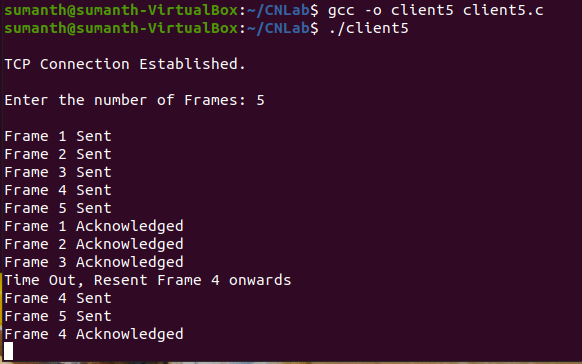
z=z/10;

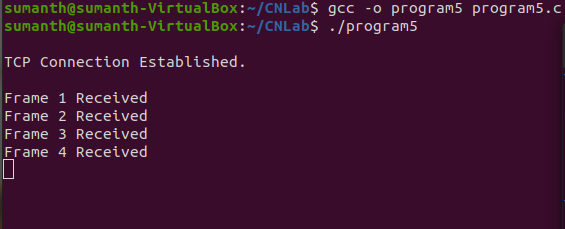
}

a[g]='\0';

}

**Output:**





**Result:**

Thus the sliding window protocol is created hence output is verified.

**Program No:** 6

**Aim:**  
         To write a program for to develop a Client that contacts a given DNS Server to resolve a given host name.

**Algorithm:**

**Server side:**Server Side Chat application algorithm  
STEP 1: Start the program.  
STEP 2: Declare the variables and structure for the socket.  
STEP 3: Create a socket using socket functions  
STEP 4: The socket is binded at the specified port.  
STEP 5: Using the object the port and address are declared.  
STEP 6: If the binding is successful.  
STEP 7: if the given host name is present it sends the ip address of given host name.

STEP 8: Execute the client program.

**Client Side:**STEP 1: Start the program.  
STEP 2: Declare the variables and structure.  
STEP 3: Socket is created and connects function is executed.  
STEP 4: If the connection is successful then server sends the host name to server.  
STEP 5: The message from the server is responded by the client.

STEP 6: it gets the ip address form server so print the host name and ip address of a host.  
STEP 7: Stop the program

**Server:**

#include<stdio.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<string.h>

int main()

{

FILE \*fp;

struct sockaddr\_in server,client;

int s;

unsigned int n;

char b1[100],b2[100],a[100];

s=socket(AF\_INET,SOCK\_DGRAM,0);

server.sin\_family=AF\_INET;

server.sin\_port=3000;

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

bind(s,(struct sockaddr \*)&server,sizeof(server));

n=sizeof(client);

while(1)

{

strcpy(b2,"");

fp=fopen("dns.txt","r");

recvfrom(s,b1,sizeof b1, 0,(struct sockaddr \*)&client,&n);

while(!feof(fp))

{

fscanf(fp,"%s",a);

if(strcmp(a,b1)==0)

{

fscanf(fp,"%s",b2);

break;

}

}

if(strcmp(b2,"")==0)

{

strcpy(b2,"Not found...");

}

fclose(fp);

sendto(s,b2,sizeof b2,0,(struct sockaddr \*)&client,n);

}

}

**Client side:**

#include<stdio.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<netinet/in.h>

int main()

{

struct sockaddr\_in server,client;

int s;

unsigned int n;

char b1[100],b2[100];

s=socket(AF\_INET,SOCK\_DGRAM,0);

server.sin\_family=AF\_INET;

server.sin\_port=3000;

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

n=sizeof(server);

printf("\nEnter Host address: ");

scanf("%s",b2);

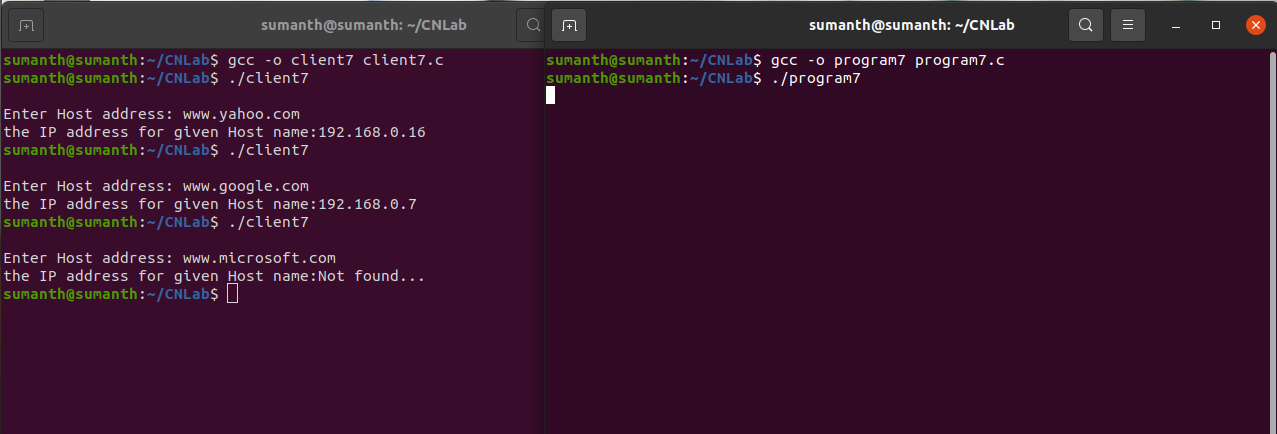
sendto(s,b2,sizeof(b2),0,(struct sockaddr \*)&server,n);

recvfrom(s,b1,sizeof(b1), 0,NULL,NULL);

printf("the IP address for given Host name:%s \n",b1);

}

**Output:**



**Result:**

Thus, the program to develop a Client that contacts a given DNS Server to resolve a given host name has been executed successfully.

**Program No:** 7

**Aim:**  
         Write a Client program to download a file from a HTTP Server.

**Algorithm:**

STEP 1: Set the command line argument.

STEP 2: Check the command line arguments and allocate yes or no for the rest of argv[1]=’1’;

STEP 3: Copy ‘1’ info host and info integers.

STEP 4: Otherwise assign post to argv[1]

STEP 5: Print the host and the request.

STEP 6: Create a socket and connect the socket.

STEP 7: Print the download page into a buffer.

STEP 8: Open a file in a write mode, write the download page into the fill.

STEP 9: Close the socket and the file.

**Program:**

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <errno.h>

#include <arpa/inet.h>

#include <string.h>

int ReadHttpStatus(int sock){

char c;

char buff[1024]="",\*ptr=buff+1;

int bytes\_received, status;

printf("Begin Response ..\n");

while(bytes\_received = recv(sock, ptr, 1, 0)){

if(bytes\_received==-1){

perror("ReadHttpStatus");

exit(1);

}

if((ptr[-1]=='\r') && (\*ptr=='\n' )) break;

ptr++;

}

\*ptr=0;

ptr=buff+1;

sscanf(ptr,"%\*s %d ", &status);

printf("%s\n",ptr);

printf("status=%d\n",status);

printf("End Response ..\n");

return (bytes\_received>0)?status:0;

}

//the only filed that it parsed is 'Content-Length'

int ParseHeader(int sock){

char c;

char buff[1024]="",\*ptr=buff+4;

int bytes\_received, status;

printf("Begin HEADER ..\n");

while(bytes\_received = recv(sock, ptr, 1, 0)){

if(bytes\_received==-1){

perror("Parse Header");

exit(1);

}

if(

(ptr[-3]=='\r') && (ptr[-2]=='\n' ) &&

(ptr[-1]=='\r') && (\*ptr=='\n' )

) break;

ptr++;

}

\*ptr=0;

ptr=buff+4;

//printf("%s",ptr);

if(bytes\_received){

ptr=strstr(ptr,"Content-Length:");

if(ptr){

sscanf(ptr,"%\*s %d",&bytes\_received);

}else

bytes\_received=-1; //unknown size

printf("Content-Length: %d\n",bytes\_received);

}

printf("End HEADER ..\n");

return bytes\_received ;

}

int main(void){

char domain[] = "sstatic.net", path[]="stackexchange/img/logos/so/so-logo-med.png";

int sock, bytes\_received;

char send\_data[1024],recv\_data[1024], \*p;

struct sockaddr\_in server\_addr;

struct hostent \*he;

he = gethostbyname(domain);

if (he == NULL){

herror("gethostbyname");

exit(1);

}

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0))== -1){

perror("Socket");

exit(1);

}

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(80);

server\_addr.sin\_addr = \*((struct in\_addr \*)he->h\_addr);

bzero(&(server\_addr.sin\_zero),8);

printf("Connecting ...\n");

if (connect(sock, (struct sockaddr \*)&server\_addr,sizeof(struct sockaddr)) == -1){

perror("Connect");

exit(1);

}

printf("Sending data ...\n");

snprintf(send\_data, sizeof(send\_data), "GET /%s HTTP/1.1\r\nHost: %s\r\n\r\n", path, domain);

if(send(sock, send\_data, strlen(send\_data), 0)==-1){

perror("send");

exit(2);

}

printf("Data sent.\n");

//fp=fopen("received\_file","wb");

printf("Recieving data...\n\n");

int contentlengh;

if(ReadHttpStatus(sock) && (contentlengh=ParseHeader(sock))){

int bytes=0;

FILE\* fd=fopen("test.png","wb");

printf("Saving data...\n\n");

while(bytes\_received = recv(sock, recv\_data, 1024, 0)){

if(bytes\_received==-1){

perror("recieve");

exit(3);

}

fwrite(recv\_data,1,bytes\_received,fd);

bytes+=bytes\_received;

printf("Bytes recieved: %d from %d\n",bytes,contentlengh);

if(bytes==contentlengh)

break;

}

fclose(fd);

}

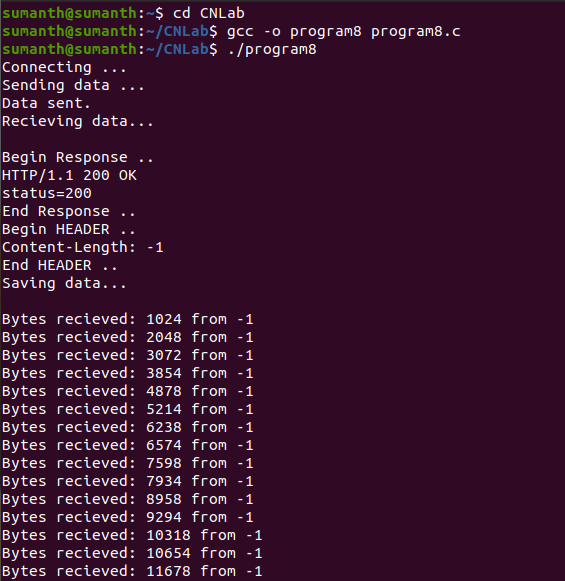
close(sock);

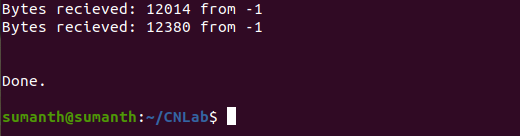
printf("\n\nDone.\n\n");

return 0;

}

**OUTPUT:**

****

****

**RESULT:**

Thus, the program for client to download a file from a HTTP Server was executed and the output was verified.

**CSE STUDY ON OPNET**

**OPNET:**

OPNET Network simulator is a tool to simulate the behavior and performance of any type of network. The main difference Opnet Network Simulator comparing to other simulators lies in its power and versatility. IT Guru provides pre-built models of protocols and devices. It allows you to create and simulation different network topologies. The set of protocols/devices is fixed – you cannot create new protocols nor modify the behavior of existing ones.

It provides a comprehensive development environment supporting the modeling of communication networks and also distributed systems.Both behavior and performance of modeled systems can also analyzed by performing discrete event simulations.C is a main programming language in OPNET and also use GUI for initial configurations. The simulation scenario requires c or C++.

**Structure of OPNET**:

OPNET consists of high level user interface, which is constructed from C and C++ source code blocks with a huge library of OPNET specific functions

Hierarchical structure: modeling is divided into three main domains:

* Network domain

-Networks + sub-networks, network topologies, geographical coordinates, mobility

* Node domain

-Single network nodes (e.g., routers, workstations, mobile devices, … )

* Process domain

-Single modules and source code inside network nodes (e.g., data traffic source model, IP protocol, … )

* With OPNET it is also possible to run external code components (External System Domain, ESD)

**OPNET Features:**

* Hierarchical network models
* Finite state machine modeling
* Integrated analysis tools
* Comprehensive library of detailed protocol and application models and network devices Object-oriented modeling
* Wireless, point-to-point, and multipoint links
* Geographical and mobility modeling
* Animation
* Integrated debugger
* Financial cost attribute for devices

**Advantages of Opnet Network Simulator:**

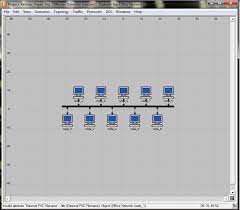
* Opnet Network Simulator is a open free software
* Large number of project scenarios that are offered information on Opnet Network Simulator
* Can be overlooked using Opnet Network Simulator.

**Uses of opnet simulator:**

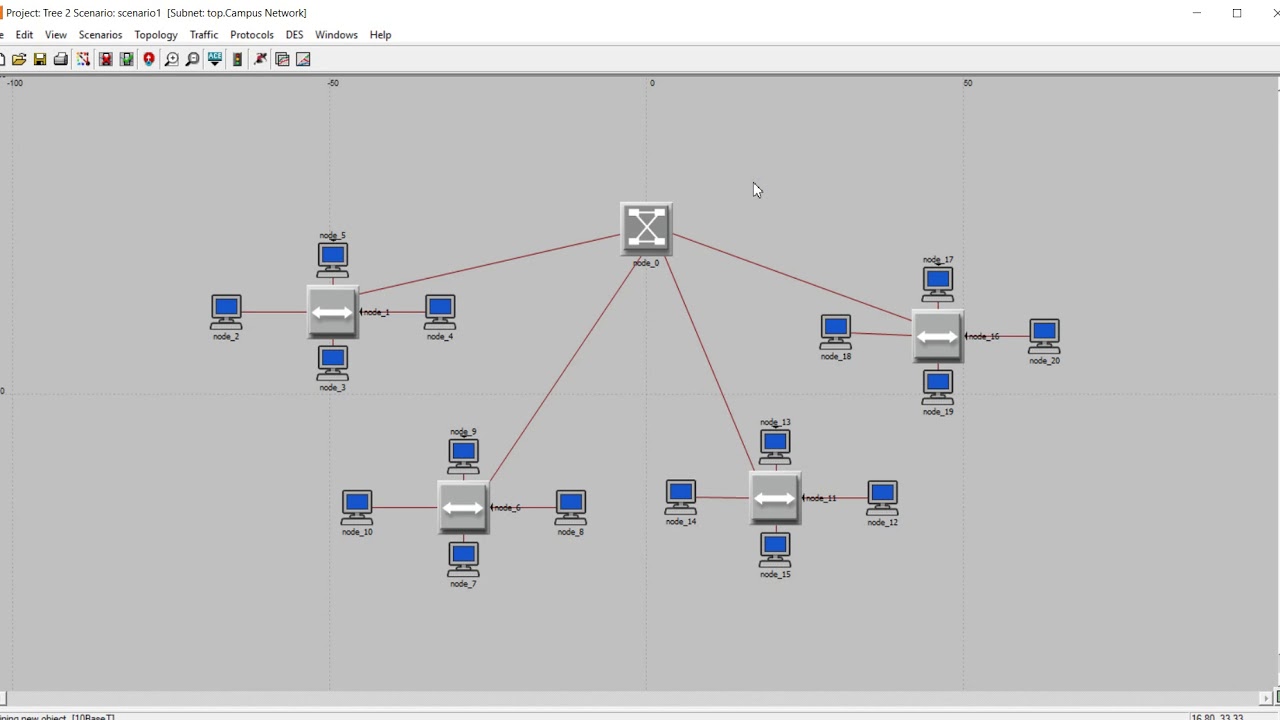
* Operational validation.
* Application troubleshooting.
* Network planning and design.
* Validating hardware architecture.
* Protocol modeling.
* Traffic modeling of telecommunication networks.
* Evaluating performance aspects of complex software systems.

**Discrete event simulation workflow:**

* Create/import topology/configuration.
* Create traffic.
* Choose statistics
* Run simulation.
* View results.
* Duplicate or create new scenario.
* Publish results.

**BUS TOPOLOGY  
**

**TREE TOPOLOGY**

****