1

```
#include<stdio.h>    #include<stdlib.h>
#include<math.h>    #include<string.h>
unsigned long modexp(unsigned long msg,
      unsigned long exp,unsigned long n)   {
    unsigned long i,k=1;
    for(i=0; i<exp; i++)
        k=(k*msg)%n;       return k;      }
int main()     {
    unsigned long p,q,e,d,n,z,i,m,c;
    int len;
    char data[100];
    printf("enter the value of p & q
               such that p*q>255\n");
    scanf("%lu%lu",&p,&q);
    n=p*q;
    z=(p-1)*(q-1);
    for(i=1; i<z; i++)     {
        if((z%i)==0)
            continue;
        else        break;      }
    e=i;
    printf("\nencryption key is=%lu",e);
    for(i=1; i<z; i++)   {
        if(((e*i-1)%z)==0)
            break;     }     d=i;
    printf("\ndecryption key is=%lu",d);
    printf("\nenter the msg:");
    scanf("%s",data);
    len=strlen(data);
    for(i=0; i<len; i++)        {
        m=(unsigned long) data[i];
        c=modexp(m,e,n);
        printf("\nencrypted key and its
             representation is %lu\t%c\n",c,c);
        m=modexp(c,d,n);
printf("\ndecrypted ----||---- %lu\t%c\n",m,m);
    }
printf("\n decrypted msg %s\n%lu\n%lu",data,c,m);   }
```

2 #include<iostream>

```
using namespace std;
class dj   {
    int n,cost[10][10],d[10],p[10],v[10];
public:
    void read_matrix();
    void short_path(int);
    void display(int);      };
void dj::read_matrix()      {
    int i,j;
    cout<<"Enter the number of vertices\n";
    cin>>n;
    cout<<"Enter the cost adjacency matrix\n";
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            cin>>cost[i][j];
}
void dj::short_path(int src)
{
    int i,j,min,u,s;
    for(i=0; i<n; i++)
    {
        d[i]=cost[src][i];
        v[i]=0;
        p[i]=src;
    }
    v[src]=1;
    for(i=0; i<n; i++)
    {
        min=99;
        u=0;
        for(j=0; j<n; j++)
        {
            if(!v[j])
                if(d[j]<min)
                {
                    min=d[j];
                    u=j;
                }
        }
        v[u]=1;
        for(s=0; s<n; s++)
            if(!v[s]&&(d[u]+cost[u][s]<d[s]))
            {
                d[s]=d[u]+cost[u][s];
                p[s]=u;
            }      }       }
void dj::display(int src)
{
    int i,k,parent;
    for(i=0; i<n; i++)
    {
        if(i==src)
            continue;
        cout<<"The shortest path from "<<src<<" to "<<i<<" is "<<endl;
        k=i;
        cout<<k<<"<----";
        while(p[k]!=src)
        {
            cout<<p[k]<<"<---";
            k=p[k];
        }
        cout<<src<<endl;
        cout<<"and the distance is "<<d[i]<<endl;
    }
}
int main()
{
    int source;
    dj dij;
    dij.read_matrix();
    cout<<"enter the source"<<endl;
    cin>>source;
    dij.short_path(source);
    dij.display(source);
    return 0;     }
```

3

```
#include<stdio.h>
#include<string.h>
Char data[100],concatdata[117],src_crc[17],
dest_crc[17],frame[120],divident[18],
divisor[18]="10001000000100001",
res[17]="0000000000000000"  ;
void crc_cal(int node)
{
    int i,j;
    for(j=17; j<=strlen(concatdata); j++)
    {
        if(divident[0]=='1')
        {
            for(i=1; i<=16; i++)
                if(divident[i]!=divisor[i])
                    divident[i-1]='1';
                else
                    divident[i-1]='0';
        }
        else
        {
            for(i=1; i<=16; i++)
                divident[i-1]=divident[i];
        }
        if(node==0)
            divident[i-1]=concatdata[j];
        else
            divident[i-1]=frame[j];
    }
    divident[i-1]='\0';
    printf("\ncrc is %s\n",divident);
    if(node==0)
    {
        strcpy(src_crc,divident);
    }
    else
        strcpy(dest_crc,divident);
}
```

```
int main()
{
    int i,len,rest;
    printf("\n\t\t\tAT SOURCE NODE\n\n
     enter the data to be send :");
    gets(data);
    strcpy(concatdata,data);
    strcat(concatdata,"0000000000000000");
    for(i=0; i<=16; i++)
        divident[i]=concatdata[i];
    divident[i+1]='\0';
    crc_cal(0);
    printf("\ndata is :\t");
    puts(data);
    printf("\nthe frame transmitted is :\t");
    printf("\n%s%s",data,src_crc);
    printf("\n\t\tSOURCE NODE
             TRANSMITTED THE FRAME ---->");
    printf("\n\n\n\n\t\t\tAT DESTINATION
            NODE\nenter the received frame:\t");
    gets(frame);
    for(i=0; i<=16; i++)
        divident[i]=frame[i];
    divident[i+1]='\0';
    crc_cal(1);
    if(strcmp(dest_crc,res)==0)
        printf("\nreceived frame is error free ");
    else
        printf("\nreceived frame has
           one or more error");
    return 1;
}
```

4 #include<stdio.h>

```
struct rtable
{
    int dist[20],nextnode[20];
} table[20];
int cost[10][10],n;
void distvector()
{
    int i,j,k,count=0;
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            table[i].dist[j]=cost[i][j];
            table[i].nextnode[j]=j;
        }
    }
do{
        count=0;
        for(i=0; i<n; i++)
        {
            for(j=0; j<n; j++)
            {
                for(k=0; k<n; k++)
                {
                    if(table[i].dist[j]>cost[i][k]+table[k].dist[j])
                    {
                        table[i].dist[j]=table[i].dist[k]+table[k].dist[j];
                        table[i].nextnode[j]=k;
                        count++;
           }    }     }    }    }  while(count!=0);
}
int main()
{
    int i,j;
    printf("\nenter the no of vertices:\t");
    scanf("%d",&n);
    printf("\nenter the cost matrix\n");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d",&cost[i][j]);
    distvector();
    for(i=0; i<n; i++)
    {
        printf("\nstate value for router %c \n",i+65);
        printf("\ndestnode\tnextnode\tdistance\n");
        for(j=0; j<n; j++)
        {
            if(table[i].dist[j]==99)
                printf("%c\t\t-\t\tinfinite\n",j+65);
            else
                printf("%c\t\t%c\t\t%d\n",j+65,
                   table[i].nextnode[j]+65,table[i].dist[j]);
        }
    }
    return 0;
}
```

5 SERVER #include<stdio.h>

```
#include<sys/types.h>    #include<sys/socket.h>
#include<netinet/in.h>     #include <stdlib.h>
#include<string.h>
void error(char *msg)
{
    perror(msg);
    exit(1);
}
int main(int argc,char *argv[])
{
    int sockfd,newsockfd,portno,clilen,n,i=0;
    char buffer[256],c[2000],ch;
    struct sockaddr_in serv_addr,cli_addr;
    FILE *fd;
    if(argc < 2)
    {
        fprintf(stderr,"ERROR,no port provided\n");
        exit(1);
    }
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0)
        error("ERROR opening socket");
    bzero((char*) &serv_addr,sizeof(serv_addr));
    portno=atoi(argv[1]);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(portno);
    if(bind(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr))<0)
           error("ERROR on binding");
    listen(sockfd,5);
    clilen=sizeof(cli_addr);
    printf("SERVER:Waiting for client....\n");
    newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clilen);
    if(newsockfd<0)
           error("ERROR on accept");
    bzero(buffer,256);
    n=read(newsockfd,buffer,255);
    if(n<0)
        error("ERROR reading from socket");
    printf("SERVER:%s \n",buffer);
    if((fd=freopen(buffer,"r",stdin))!=NULL)
    {
        printf("SERVER:%s found! \n Transfering the contents ...\n",buffer);
        while((ch=getc(stdin))!=EOF)
            c[i++]=ch;
        c[i]='\0';
        printf("File content %s\n",c);
        n=write(newsockfd,c,1999);
        if(n<0)
            error("ERROR in writing to socket");
    }
    else
    {
        printf("SERVER:File not found!\n");
        n=write(newsockfd,"File not found!",15);
        if(n<0)
            error("ERROR writing to socket");
    }
    return 0;        }
```

CLIENT

```
void error(char *msg)
{
    perror(msg);
    exit(0);
}
int main(int argc,char *argv[])
{
    int sockfd,portno,n;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char filepath[256],buf[3000];
    if(argc < 3)
    {
        fprintf(stderr,"usage %s hostname port\n",argv[0]);
        exit(0);
    }
    portno=atoi(argv[2]);
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0)
        error("\nerror in opening socket");
    printf("\nclient online");
    server=gethostbyname(argv[1]);
    if(server==NULL)
    {
        fprintf(stderr,"error ,no such host");
        exit(0);
    }
    printf("\n server online");
    bzero((struct sockaddr_in *)
         &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(portno);
    if(connect(sockfd,(struct sockaddr_in*)
         &serv_addr,sizeof(serv_addr))<0)
        error("error writing to socket");
    printf("\nclient:enter path with filename:\n");
    scanf("%s",filepath);
    n=write(sockfd,filepath,strlen(filepath));
    if(n<0)
        error("\nerror writing to socket");
    bzero(buf,3000);
    n=read(sockfd,buf,2999);
    if(n<0)
        error("\nerror reading to socket");
    printf("\nclient:displaying from socket");
    fputs(buf,stdout);
    return 0;
}
```

6

```
#include<stdio.h>
#include<math.h>
void genhamcode();
void makeerror();
void correcterror();
int h[12];
int main()
{
    int i,ch;
    printf("\n enter the message in bits\n");
    for(i=1; i<12; i++)
        if(i==3||i==5||i==6||i==7||i==9||i==10||i==11)
            scanf("%d",&h[i]);
    for(i=1; i<12; i++)
        printf("%d",h[i]);
    genhamcode();
    printf("\n do you want to make error\n(0 or 1)\n");
    scanf("%d",&ch);
    if(ch)
    {
        makeerror();
        correcterror();
    }
    else
        printf("\n no error");
    return(0);
}
void genhamcode()
{
    int temp,i;
    temp=h[3]+h[5]+h[7]+h[9]+h[11];
    (temp%2!=0)?(h[1]=1):(h[1]=0);
    temp=h[3]+h[6]+h[7]+h[10]+h[11];
    (temp%2!=0)?(h[2]=1):(h[2]=0);
    temp=h[5]+h[6]+h[7];
    (temp%2!=0)?(h[4]=1):(h[4]=0);
    temp=h[9]+h[10]+h[11];
    (temp%2!=0)?(h[8]=1):(h[8]=0);
    printf("\n transmitted codeword is:\n");
    for(i=1; i<12; i++)
        printf(" %d ",h[i]);
}
void makeerror()
{
    int pos,i;
    printf("\n enter the position you want to make error\n");
    scanf("%d",&pos);
    if(h[pos]==1)
        h[pos]=0;
    else
        h[pos]=1;
    printf("\n Error occured and the error codeword is\n");
    for(i=1; i<12; i++)
        printf(" %d ",h[i]);
}
void correcterror()
{
    int r1,r2,r4,r8,i,errpos;
    r1=(h[1]+h[3]+h[5]+h[7]+h[9]+h[11])%2;
    r2=(h[2]+h[3]+h[6]+h[7]+h[10]+h[11])%2;
    r4=(h[4]+h[5]+h[6]+h[7])%2;
    r8=(h[8]+h[9]+h[10]+h[11])%2;
    errpos=r8*8+r4*4+r2*2+r1*1;
    printf("\n Error occured in pos %d\n",errpos);
    printf("\n\n............ correction starts now........\n");
    if(h[errpos]==1)
        h[errpos]=0;
    else
        h[errpos]=1;
    printf("\n Original codeword is :");
    for(i=1; i<12; i++)
        printf(" %d ",h[i]);     }
```

10

```
#include <stdio.h>
#include <math.h>
void main()
{
    int q,alpha,xa,xb,ya,yb,ka,kb, x,y,z,count,ai[20][20];
    printf("Enter a Prime Number \"q\":");
    scanf("%d",&q);
    printf("Enter a No \"xa\" which is lessthan value of q:");
    scanf("%d",&xa);
    printf("Enter a No \"xb\" which is lessthan value of q:");
    scanf("%d",&xb);
    for(x=0; x<q-1; x++)
        for(y=0; y<q-1; y++)
            ai[x][y] = ((int)pow(x+1,y+1))%q;
    for(x=0; x<q-1; x++)        {
        count = 0;
        for(y=0; y<q-2; y++)        {
            for(z=y+1; z<q-1; z++)
                if(ai[x][y] == ai[x][z])            {
                    count = 1;
                    break;                }
            if(count == 1)
                break;            }
        if (count == 0 )
        {
            alpha = x+1;
            break;          }     }
    printf("alpha = %d\n",alpha);
    ya = ((int)pow(alpha,xa))%q;
    yb = ((int)pow(alpha,xb))%q;
    ka = ((int)pow(yb,xa))%q;
    kb = ((int)pow(ya,xb))%q;
    printf("ya = %d\nyb = %d\nka = %d\nkb = %d\n",ya,yb,ka,kb);
    if(ka == kb) printf("The keys exchanged are same");
    else printf("The keys exchanged are not same");
}
```

11 #include<stdio.h>

```
#include<stdlib.h>    #include<string.h>
#include<sys/types.h>    #include<error.h>
#include<sys/stat.h>    #include<unistd.h>
#define min(x,y)((x)<(y)?(x):(y))
#define max(x,y)((x)>(y)?(x):(y))
#define MAX 25
int main()
{
    int cap,oprt,cont,i=0,inp[MAX],ch,nsec,drop;
    printf("LEAKY BUCKET ALGORITM\n");
    printf("\nEnter the bucket size:\n");
    scanf("%d",&cap);
    printf("\nEnter the output rate:");
    scanf("%d",&oprt);
    do
    {
        printf("\nEnter the number of packets
                entering at %d seconds\n",i+1);
        scanf("%d",&inp[i]);
        i++;
        printf("\nEnter 1 to insert packet or 0 to quit\n");
        scanf("%d",&ch);
    }
    while(ch);
    nsec=i;
    printf("\n(SECOND):(PACK RECVD):(PACK SENT):
      (PACK LEFT IN BUCKET):(PACK DROPPED)\n");
    cont=0;
    drop=0;
    for(i=0; i<nsec; i++)
    {
        cont+=inp[i];
        if(cont>cap)
        {
            drop=cont-cap;
            cont=cap;
        }
        printf("(%d): ",i+1);
        printf("\t\t(%d): ",inp[i]);
        printf("\t\t(%d): ",min(cont,oprt));
        cont=cont-min(cont,oprt);
        printf("\t\t(%d)",cont);
        printf("\t\t(%d)\n",drop);
    }
    for(; cont!=0; i++)
    {
        if(cont>cap)
            cont=cap;
        drop=0;
        printf("(%d): ",i+1);
        printf("\t\t(0): ");
        printf("\t\t(%d): ",min(cont,oprt));
        cont=cont-min(cont,oprt);
        printf("\t\t(%d)",cont);
        printf("\t\t(%d)\n",drop);
    }
    return(0);
}
```

TCL file

```
set ns [ new Simulator ]
set tf [ open lab1.tr w ]
$ns trace-all $tf
set nf [ open lab1.nam w ]
$ns namtrace-all $nf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns color 1 "red"
$ns color 2 "blue"
$n0 label "Source/udp0"
$n1 label "Source/udp1"
$n2 label "Router"
```

```
$n3 label "Destination/Null"
$ns duplex-link $n0 $n2 10Mb 300ms DropTail
$ns duplex-link $n1 $n2 10Mb 300ms DropTail
$ns duplex-link $n2 $n3 1Mb 300ms DropTail
$ns set queue-limit $n0 $n2 10
$ns set queue-limit $n1 $n2 10
$ns set queue-limit $n2 $n3 5
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$udp0 set class_ 1
$udp1 set class_ 2
$ns connect $udp0 $null3
$ns connect $udp1 $null3
$cbr1 set packetSize_ 500Mb
$cbr1 set interval_ 0.005
proc finish { } {
global ns nf tf
$ns flush-trace
exec nam lab1.nam &
close $tf
close $nf
exit 0
}
$ns at 0.1 "$cbr0 start"
$ns at 0.1 "$cbr1 start"
$ns at 10.0 "finish"
$ns run
```

awk file

```
BEGIN{
```

```
count=0
}
{
if($1=="d") #d stands for the packets drops.
count++
}
END{
printf("The Total no of Packets Dropped
        due to Congestion : %d\n\n", count)
}
```

7 SENDER
```
#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>
int main()      {
 int sock,bytes_received,connected,true=1,i=1,s,f=0,sin_size;
 char send_data[1024],data[1024],c,fr[30]=" ";
 struct sockaddr_in server_addr,client_addr;
 if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)      {
 perror("Socket not created");
 exit(1);        }
 if(setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,&true,sizeof(int))==-1)
 {
 perror("Setsockopt");
 exit(1);        }
 server_addr.sin_family=AF_INET;
 server_addr.sin_port=htons(17000);
 server_addr.sin_addr.s_addr=INADDR_ANY;
 if(bind(sock,(struct sockaddr *)&server_addr,sizeof(struct sockaddr))==-1)
 {
 perror("Unable to bind");
 exit(1);       }
 if(listen(sock,5)==-1)        {
 perror("Listen");
 exit(1);        }
 fflush(stdout);
 sin_size=sizeof(struct sockaddr_in);
 connected=accept(sock,(struct sockaddr *)&client_addr,&sin_size);
 while(strcmp(fr,"exit")!=0)         {
 printf("Enter Data Frame %d:(Enter exit for End):",i);
 scanf("%s",fr);
 send(connected,fr,strlen(fr),0);
 recv(sock,data,1024,0);
 if(strlen(data)!=0)
 printf("I got an acknowledgment : %s\n",data);
 fflush(stdout);       i++;       }
 close(sock);          return(0);      }
```

7 RECEIVER

```
int main()     {
 int sock,bytes_received,i=1;
 char receive[30];
 struct hostent *host;
 struct sockaddr_in server_addr;
host=gethostbyname("127.0.0.1");
 if((sock=socket(AF_INET,SOCK_STREAM,0))==-1)         {
 perror("Socket not created");          exit(1);      }
 printf("Socket created");
 server_addr.sin_family=AF_INET;
 server_addr.sin_port=htons(17000);
 server_addr.sin_addr=*((struct in_addr *)host->h_addr);
 bzero(&(server_addr.sin_zero),8);
 if(connect(sock,(struct sockaddr *)&server_addr,sizeof(struct sockaddr))==-1)
{
 perror("Connect");
 exit(1);       }
 while(1)       {
 bytes_received=recv(sock,receive,20,0);
 receive[bytes_received]='\0';
 if(strcmp(receive,"exit")==0)       {
 close(sock);     break;       }
 else      {
 if(strlen(receive)<10)        {
printf("\nFrame %d data %s received\n",i,receive);
send(0,receive,strlen(receive),0);       }
 else        {
send(0,"negative",10,0);      }
 i++;      }       }  close(sock);
 return(0);       }
```

output

<u>At terminal 1</u>   <u>At terminal 2</u>

```
$ gcc 7sender.c         $ gcc 7receiver.c
./a.out                   $ ./a.out
```

8 SERVER

```
#include<stdio.h>      #include<stdlib.h>
#include<errno.h>      #include<string.h>
#include<fcntl.h>      #include<sys/types.h>
#include<sys/stat.h>     #include<unistd.h>
#define FIFO1_NAME "fifo1"
#define FIFO2_NAME "fifo2"
int main()    {
    char p[100],f[100],c[300],ch;
    int num,num2,f1,fd,fd2,i=0;
    mknod(FIFO1_NAME,S_IFIFO |0666,0);
    mknod(FIFO2_NAME,S_IFIFO |0666,0);
    printf("\nSERVER ONLINE");
    fd=open(FIFO1_NAME,O_RDONLY);
    printf("client online\nwaiting for request\n\n");
    while(1)         {
        if((num=read(fd,p,100))==-1)
            perror("\nread error");
        else             {
            p[num]='\0';
            if((f1=open(p,O_RDONLY))<0)          {
                printf("\nserver: %s not found",p);
                exit(1);                }
            else               {
                printf("\nserver:%s found \ntranfering the contents",p);
                stdin=fdopen(f1,"r");
                while((ch=getc(stdin))!=EOF)
                    c[i++]=ch;
                c[i]='\0';
                printf("\nfile contents %s\n ",c);
                fd2=open(FIFO2_NAME,O_WRONLY);
                if(num2=write(fd2,c,strlen(c))==-1)
                    perror("\ntranfer error");
                else
                    printf("\nserver :tranfer completed");                 }
            exit(1);       }     }       }
```

8 CLIENT

```
int main()      {
    char p[100],f[100],c[3000];
    int num,num2,f1,fd,fd2;
    mknod(FIFO1_NAME,S_IFIFO|0666,0);
    mknod(FIFO2_NAME,S_IFIFO|0666,0);
    printf("\n waiting for server...\n");
    fd=open(FIFO1_NAME,O_WRONLY);
    printf("\n SERVER ONLINE !\n CLIENT:Enter the path\n");
    while(gets(p),!feof(stdin))         {
        if((num=write(fd,p,strlen(p)))==-1)
            perror("write error\n");
        else            {
            printf("Waiting for reply....\n");
            fd2=open(FIFO2_NAME,O_RDONLY);
            if((num2=read(fd2,c,3000))==-1)
                perror("Transfer error!\n");
            else                {
                printf("File recieved! displaying the contents:\n");
                if(fputs(c,stdout)==EOF)
                    perror("print error\n");
                exit(1);        }     }     }      }
```

OUTPUT :

```
AT TERMINAL 1 :         AT TERMINAL 2 :
$ gcc 8server.c           $ gcc 8client.c
$ ./a.out                  $ ./a.out
```

9 SERVER

```
#include<sys/types.h>
#include<sys/socket.h>  #include<netinet/in.h>
#include<netdb.h>     #include<stdio.h>
#include<string.h>     #include<stdlib.h>
void error(char *msg)       {
    perror(msg);
    exit(0);     }
int main(int argc, char *argv[])     {
    int sock, length, fromlen, n;
    struct sockaddr_in server;
    struct sockaddr_in from;
    char buf[1024];
    if (argc < 2)         {
        fprintf(stderr, "ERROR, no port provided\n");
        exit(0);         }
    Sock=socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0)         {
        error("Opening socket");          }
    length = sizeof(server);
    bzero(&server,length);
    server.sin_family=AF_INET;
    server.sin_addr.s_addr=INADDR_ANY;
    server.sin_port=htons(atoi(argv[1]));
    if (bind(sock,(struct sockaddr *)&server,length)<0)         {
        error("binding");          }
    fromlen = sizeof(struct sockaddr_in);
    while (1)         {
        n = recvfrom(sock,buf,1024,0,(struct sockaddr *)&from,&fromlen);
        if (n < 0)            {
            error("recvfrom");            }
        write(1,"Received a datagram: ",21);
        write(1,buf,n);
        n = sendto(sock,"Got your message\n",17,
                   0,(struct sockaddr *)&from,fromlen);
        if (n < 0)            {
            error("sendto");      }       }       }
```

9 CLIENT

```
void error(char *);
int main(int argc, char *argv[])    {
    int sock, length, n;
    struct sockaddr_in server, from;
    struct hostent *hp;
    char buffer[256];
    if (argc != 3)        {
        printf("Usage: server port\n");
        exit(1);        }
    sock= socket(AF_INET, SOCK_DGRAM, 0);
    if(sock<0)        {
        error("socket");        }
    server.sin_family=AF_INET;
    hp=gethostbyname(argv[1]);
    if(hp==0)        {        error("Unknown host");        }
    bcopy((char *)hp->h_addr,(char *)&server.sin_addr,hp->h_length);
    server.sin_port = htons(atoi(argv[2]));
    length=sizeof(struct sockaddr_in);
    printf("Please enter the message: ");
    bzero(buffer,256);
    fgets(buffer,255,stdin);
    n=sendto(sock,buffer,strlen(buffer),0,&server,length);
    if (n < 0)        {          error("Sendto");        }
    n = recvfrom(sock,buffer,256,0,&from, &length);
    if (n < 0)        {
        error("recvfrom");        }
    write(1,"Got an ack: ",12);
    write(1,buffer,n);     }
void error(char *msg)     {
    perror(msg);
    exit(0);     }
```

OUTPUT :

```
AT TERMINAL 1 :          AT TERMINAL 2 :
$ gcc 9server.c          $ gcc 9client.c
~$ ./a.out 8080            $ ./a.out localhost 8080
```

3 TCL file

```
set ns [new Simulator]

set trace_file [open lab3.tr w]

$ns trace-all $trace_file

set nam_file [open lab3.nam w]

$ns namtrace-all $nam_file

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n0 label "Ping0"

$n4 label "Ping4"

$n1 label "Ping1"

$n5 label "Ping5"

$ns color 1 "blue"

$ns color 2 "orange"

$ns duplex-link $n0 $n2 0.5mb 10ms DropTail

$ns duplex-link $n1 $n2 0.5mb 10ms DropTail

$ns duplex-link $n2 $n3 0.5mb 10ms DropTail

$ns duplex-link $n3 $n4 0.5mb 10ms DropTail

$ns duplex-link $n3 $n5 0.5mb 10ms DropTail

set ping0 [new Agent/Ping]

$ns attach-agent $n0 $ping0

set ping4 [new Agent/Ping]

$ns attach-agent $n4 $ping4

set ping1 [new Agent/Ping]

$ns attach-agent $n1 $ping1

set ping5 [new Agent/Ping]

$ns attach-agent $n5 $ping5

$ping0 set packetSize_ 500

$ping0 set interval_ 0.001

$ping1 set packetSize_ 500

$ping1 set interval_ 0.001

$ping4 set packetSize_ 500

$ping4 set interval_ 0.001

$ping5 set packetSize_ 500
```

```
$ping5 set interval_ 0.001
set udp0 [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp0
$ns attach-agent $n4 $null
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 512
$cbr set interval_ 0.001
$cbr attach-agent $udp0
$ns connect $udp0 $null
$ping0 set class_ 1
$ping1 set class_ 2
$ns connect $ping0 $ping4
$ns connect $ping1 $ping5
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts " The node [$node_ id] received a reply
   from $from with round trip time of $rtt ms"   }
#define finish procedure
proc finish   { }   {
global ns nam_file trace_file
$ns flush-trace
exec nam lab3.nam &
close $trace_file
close $nam_file
exit 0     }
#schedule events to start sending the ping packets
$ns at 0.1 "$ping0 send"
$ns at 0.2 "$ping0 send"
$ns at 0.3 "$ping0 send"
$ns at 0.4 "$ping0 send"
$ns at 0.5 "$ping0 send"
$ns at 0.6 "$ping0 send"
$ns at 0.7 "$ping0 send"
$ns at 0.8 "$ping0 send"
$ns at 0.9 "$ping0 send"
$ns at 1.0 "$ping0 send"
$ns at 0.2 "$cbr start"
```

```
$ns at 4.0 "$cbr stop"
$ns at 0.1 "$ping1 send"
$ns at 0.2 "$ping1 send"
$ns at 0.3 "$ping1 send"
$ns at 0.4 "$ping1 send"
$ns at 0.5 "$ping1 send"
$ns at 0.6 "$ping1 send"
$ns at 0.7 "$ping1 send"
$ns at 0.8 "$ping1 send"
$ns at 0.9 "$ping1 send"
$ns at 1.0 "$ping1 send"
$ns at 5.5 "finish"
$ns run
```

Awk file

```
BEGIN{
#include<stdio.h>
count=0    }
if($1=="d")
{    count++    }
END     {
 printf("The Total no of Packets Dropped
     due toCongestion:%d ", count)     }
```

OUTPUT $ ns lab3.tcl

```
$ awk -f lab3.awk lab3.tr
```