MALNAD COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION UNDER VTU, BELAGAVI)
HASSAN, KARNATAKA 573202, INDIA

Department of Computer Science and Engineering

NETWORK LABORATORY MANUAL

CS705

0-1.5-1.5
)
topologies
POs
PO6,PO10

2.	Execute the socket programming and networking using network simulators and are documented.	PO2,PO6,PO10
3.	Depict built-in networking modules and design user defined topologies and modules	PO2,PO6,PO8

Course Contents:

Following set of programs are given for execution in lab, which will be helpful in understanding the basics of programming and serves as base for execution of Exercise Programs. These programs are not considered for CIE and SEE, but carry 10 marks that will be included with record mark. Cryptographic program needs to be executed in c/c++ and simulation programs in NS2 tool.

PRACTICE PROGRAMS (SELF STUDY COMPONENT)

Write and execute Programs for the below given problems before executing the corresponding programs of Exercise Part.

1.	Write and execute a program for error detecting code using CRC-CCITT (16- bits).
2.	Write and execute a program for congestion control using leaky bucket algorithm.
3.	Simulate a three nodes point – to – point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped.
4.	Simulate a three nodes point – to – point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets sent with different types of traffic.

EXERCISE PROGRAMS

Following set of programs are included in CIE and SEE, Students have to pick a program from lot of Programs in CIE and SEE.

1.	Write and execute a program for distance vector algorithm to find the suitable path for transmission between sender and receiver and also find the appropriate path if the link has been break-down.
2.	Write and execute a program to find 16-bit and 32-bit checksum Fletcher and Adler Checksum methods.
3.	Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present
4.	Write and execute a program for simple RSA algorithm to encrypt and decrypt the data where the input prime numbers should be very large and display all the possible values of encryption key.
5.	Simulate a four node point-to-point network with the links connected as follows: n0 – n2, n1 – n2 and n2 – n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP. And also plot the throughput graph for both TCP and UDP traffic.
6.	Simulate a point to point network using n nodes and set multiple traffic and plot Packet delivery ratio, End to end delay and throughput for different source / destination.
7.	Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion. And also plot the Congestion graph.
8.	Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput. And also plot the graph for different throughputs.

9.	Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
10.	Simulate simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets

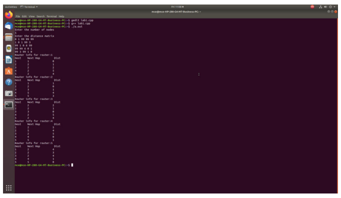
Programs

1. **Write and execute a program for distance vector algorithm to find the suitable path for transmission between sender and receiver and also find the appropriate path if the link has been break-down.**
Program:

```
#include<iostream>
#include<stdio.h>
using namespace std;
struct node {
int dist[20];
int from[20];
}route[10];
int main()
{
int dm[20][20],no;
cout<<"Enter the number of nodes"<<endl;
cin>>no;
cout<<"Enter the distance matrix"<<endl;
for(int i=0;i<no;i++)
{ for(int j=0;j<no;j++)
{ cin>>dm[i][j];
        dm[i][i]=0;
        route[i].dist[j]=dm[i][j];
        route[i].from[j]=j;
}
}
int flag;
do{
        flag=0;
        for(int i=0;i<no;i++)
        {       for(int j=0;j<no;j++)
                {       for(int k=0;k<no;k++)
                        { if((route[i].dist[j])>(route[i].dist[k]+route[k].dist[j]))
                          { route[i].dist[j]=route[i].dist[k]+route[k].dist[j];
                            route[i].from[j]=k;
                            flag=1;
                          }
                        }
                }
```

```
        }
}while(flag);
for(int i=0;i<no;i++)
{ cout<<"Router info for router:"<<i+1<<endl;
cout<<"Dest\tNext Hop\t Dist"<<endl;
        for(int j=0;j<no;j++)
         printf("%d\t%d\t\t%d\n",j+1,route[i].from[j]+1,route[i].dist[j]);
}
return 0;
}
```

Output:


Activities
Terminal ▾
Fri 11:00 ●
mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

```
mce@mce-HP-280-G4-MT-Business-PC:~$ gedit lab1.cpp
mce@mce-HP-280-G4-MT-Business-PC:~$ g++ lab1.cpp
mce@mce-HP-280-G4-MT-Business-PC:~$ ./a.out
Enter the number of nodes
5
Enter the distance matrix
0 1 99 99 99
1 0 1 99 3
99 1 0 6 99
99 99 6 0 1
99 3 99 1 0
Router info for router:1
Dest    Next Hop         Dist
1       1               0
2       2               1
3       2               2
4       2               5
5       2               4
Router info for router:2
Dest    Next Hop         Dist
1       1               1
2       2               0
3       3               1
4       5               4
5       5               3
Router info for router:3
Dest    Next Hop         Dist
1       2               2
2       2               1
3       3               0
4       2               5
5       2               4
Router info for router:4
Dest    Next Hop         Dist
1       2               5
2       5               4
3       2               5
4       4               0
5       5               1
Router info for router:5
Dest    Next Hop         Dist
1       2               4
2       2               3
3       2               4
4       4               1
5       5               0
mce@mce-HP-280-G4-MT-Business-PC:~$
```

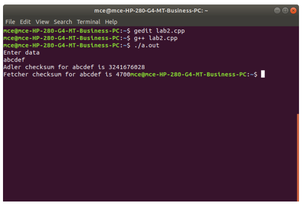
2. **Write and execute a program to find 16-bit and 32-bit checksum Fletcher and Adler Checksum methods**

Program:

```
#include <iostream>
#include<inttypes.h>
#include<stdio.h>
#include<stdint.h>
#include<stdio.h>

using namespace std;
const uint32_t MOD_ADLER = 65521;

uint32_t adler32(unsigned char *data, size_t len)
/*
where data is the location of the data in physical memory and
len is the length of the data in bytes
*/
{
uint32_t a = 1;
uint32_t b = 0;
size_t index;

// Process each byte of the data in order
for (index = 0; index < len; ++index)
{
a = (a + data[index]) % MOD_ADLER;
b = (b + a) % MOD_ADLER;
}

return (b << 16) | a;
}
uint16_t Fletcher16( uint8_t *data, int count )
{
uint16_t sum1 = 0;
uint16_t sum2 = 0;
int index;

for ( index = 0; index < count; ++index )
{
sum1 = (sum1 + data[index]) % 255;
sum2 = (sum2 + sum1) % 255;
}

return (sum2 << 8) | sum1;
}
int main()
```

```
{ uint8_t data[20];
cout<<"Enter data\n";
cin>>data;
uint16_t fletcher;
uint32_t adler;
fletcher=Fletcher16(data,16);
adler=adler32(data,32);
cout<<"Adler checksum for "<<data<<" is "<<adler;
cout<<"\nFetcher checksum for "<<data<<" is "<<fletcher;
}
```

Output:


mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

```
mce@mce-HP-280-G4-MT-Business-PC:~$ gedit lab2.cpp
mce@mce-HP-280-G4-MT-Business-PC:~$ g++ lab2.cpp
mce@mce-HP-280-G4-MT-Business-PC:~$ ./a.out
Enter data
abcdef
Adler checksum for abcdef is 3241676028
Fetcher checksum for abcdef is 4700mce@mce-HP-280-G4-MT-Business-PC:~$
```

3. **Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present**

Client.c

```
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/fcntl.h>
#include<arpa/inet.h>

int main(int argc, char*argv[])
{
        int sockfd,portno,n;
        struct sockaddr_in serv_addr;
        char buffer[4096],*servip;
        if(argc<4)
        {
                fprintf(stderr,"usage%s serverip filename port\n",argv[0]);
                exit(0);
        }
        servip=argv[1];
        portno=atoi(argv[3]);
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd<0)
                perror("error openingt socket");
        printf("Client Online\n");
        bzero((char*)&serv_addr,sizeof(serv_addr));
        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=inet_addr(servip);
        serv_addr.sin_port=htons(portno);
        if(connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr))<0)
                perror("error connecting");
        write(sockfd,argv[2],strlen(argv[2])+1);
        bzero(buffer,4096);
        n=read(sockfd,buffer,4096);
        if(n<=0)
        {
                perror("file not found");
                exit(0);
        }
        write(1,buffer,n);
}
```

Server.c

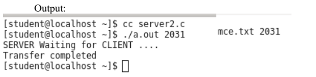
```
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/fcntl.h>
int main(int argc, char*argv[])
{
        int fd,sockfd,newsockfd,clilen,portno,n;
        struct sockaddr_in serv_addr,cli_addr;
        char buffer[4096];
        if(argc<2)
        {
                fprintf(stderr,"no port\n");
                  exit(1);
        }
        portno=atoi(argv[1]);
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd<0)
                error("error opening socket");
        bzero((char*)&serv_addr,sizeof(serv_addr));
        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=(htonl)INADDR_ANY;
        serv_addr.sin_port=htons(portno);
        if(bind(sockfd,(struct sockaddr*) &serv_addr,sizeof(serv_addr))<0)
                perror("error bindind");
        listen(sockfd,5);
        clilen=sizeof(cli_addr);
        printf("SERVER Waiting for CLIENT ....\n");
        while(1)
        {       newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&clilen);
                if(newsockfd<0)
                        perror("error on accept");
                bzero(buffer,4096);
                read(newsockfd,buffer,4096);
                fd=open(buffer,O_RDONLY);
                if(fd<0)
                {  write(newsockfd,"file does not exist in server",30);
                   perror("File not found");
                   exit(0);
                }
                else
                { while(1)
```

```
                        { n=read(fd,buffer,4096);
                          if(n<=0)
                                exit(0);
                          write(newsockfd,buffer,n);
                          printf("Transfer completed\n");
                        }
                close(fd);
                close(newsockfd);
        }
        }
        return 0;
}
```

To execute :
Run Server program first then client program

Server side	Client side
~$ **vi tcpserver.c** ~$ cc **tcpserver.c** ~$ ./**a.out** <***port_number***> **Ex:** ./a.out **2020** SERVER waiting for the CLIENT.... Transfer Completed…	~$ **vi client.c** ~$ cc **tcpclient.c** ~$ ./**a.out** <*localhost_ip_address*> <*sample.txt*> <*sender_port_number*> **EX:** ./a.out **127.0.0.1 rit.txt 2020** Client online Welcome to RIT…..

Output:



```
[student@localhost ~]$ cc server2.c
[student@localhost ~]$ ./a.out 2031          mce.txt 2031
SERVER Waiting for CLIENT ....
Transfer completed
[student@localhost ~]$ 
```

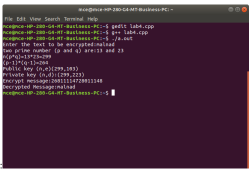
4. **Write and execute a program for simple RSA algorithm to encrypt and decrypt the data where the input prime numbers should be very large and display all the possible values of encryption key.**

Program:

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<string.h>
using namespace std;
long int gcd(long int a, long int b)
{ if(a==0)
return b;
if(b==0)
        return a;
return gcd(b,a%b);
}
long int isprime(long int a)
{
        int i;
        for(i=2;i<a;i++)
        { if((a%i)==0)
          return 0;
        }
        return 1;
}
long int encrypt( char ch, long int n,long int e)
{       int i;
        long int temp=ch;
        for(i=1;i<e;i++)
                temp=(temp*ch)%n;
        return temp;
}
char decrypt(long int ch,long int n, long int d)
{       int i;
        long int temp=ch;
        for(i=1;i<d;i++)
        ch=(temp*ch)%n;
        return ch;
}
int main()
{
        long int i,len;
        long int p,q,n,phi,e,d,cipher[50];
        char text[50];
        cout<<"Enter the text to be encrypted:";
```

```
        cin.getline(text,sizeof(text));
        len=strlen(text);
        do
        { p=rand()%30;
        }while(!isprime(p));
        do{ q=rand()%30;
        }while(!isprime(q));
        n=p*q;
        phi=(p-1)*(q-1);
        do{ e=rand()%phi;
        }while(gcd(phi,e)!=1);
        do{ d=rand()%phi;
        }while(((d*e)%phi)!=1);
        cout<<"two prime number (p and q) are:"<<p<<" and "<<q<<endl;
        cout<<"n(p*q)="<<p<<"*"<<q<<"="<<p*q<<endl;
        cout<<"(p-1)*(q-1)="<<phi<<endl;
        cout<<"Public key (n,e)("<<n<<","<<e<<")\n";
        cout<<"Private key (n,d):("<<n<<","<<d<<")\n";
        for(i=0;i<len;i++)
        cipher[i]=encrypt(text[i],n,e);
        cout<<"Encrypt message:";
        for(i=0;i<len;i++)
        cout<<cipher[i];
        for(i=0;i<len;i++)
        text[i]=decrypt(cipher[i],n,d);
        cout<<endl;
        cout<<"Decrypted Message:";
        for(i=0;i<len;i++)
                cout<<text[i];
        cout<<endl;
        return 0;
}
```

Output:


mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

```
mce@mce-HP-280-G4-MT-Business-PC:~$ gedit lab4.cpp
mce@mce-HP-280-G4-MT-Business-PC:~$ g++ lab4.cpp
mce@mce-HP-280-G4-MT-Business-PC:~$ ./a.out
Enter the text to be encrypted:malnad
two prime number (p and q) are:13 and 23
n(p*q)=13*23=299
(p-1)*(q-1)=264
Public key (n,e)(299,103)
Private key (n,d):(299,223)
Encrypt message:26811114728011148
Decrypted Message:malnad
mce@mce-HP-280-G4-MT-Business-PC:~$
```

5. **Simulate a four node point-to-point network with the links connected as follows: n0 – n2, n1 – n2 and n2 – n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP. And also plot the throughput graph for both TCP and UDP traffic**

Program:

lab5.tcl

```
set ns [new Simulator]
set ntrace [open lab5.tr w]
$ns trace-all $ntrace
set file3 [open lab5.nam w]
$ns namtrace-all $file3
proc finish {} {
global ns ntrace
$ns flush-trace
close $ntrace
exec nam lab5.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 10Mb 1ms DropTail
$ns duplex-link $n2 $n3 10Mb 1ms DropTail
$ns duplex-link $n2 $n1 10Mb 1ms DropTail
$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$ns connect $tcp0 $sink0
$ns connect $udp0 $null0
$ns at 0.0 "$ftp0 start"
$ns at 0.2 "$cbr0 start"
$ns at 0.8 "finish"
```

```
$ns run
```

AWK file:*(Open a new editor using "vi command" and write awk file and save with ".awk" extension)*

lab5.awk

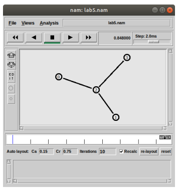
```
BEGIN{
d=0;
tcp=0;
udp=0;
pkt_t=0;
time_t=0;
pkt_u=0;
time_u=0;
}
{
{
if(($1=="r"&& $3=="0"&& $4=="2"&& $5=="tcp")||($1=="r"&& $3=="2"&&
$4=="3"&& $5=="tcp"))
 { pkt_t=pkt_t+$6;
  time_t=$2;
 printf("%f\t%f\n",pkt_t,time_t);
 }
if(($1=="r"&& $3=="1"&& $4=="2"&& $5=="cbr")|| ($1=="r"&& $3=="2"&&
$4=="3"&& $5=="cbr"))
 { pkt_u=pkt_u+$6;
  time_u=$2;
        printf("%f\t%f\n",pkt_u,time_u);
 }
}
}

END{
printf("Throughput of TCP :%f  Mbps\n",((pkt_t/time_t)*(8/1000000)));
printf("Throughput of UDP :%f Mbps\n",((pkt_u/time_u)*(8/1000000)));
}
```

Commands for execution:

- To open tcl file **: #gedit filename.tcl**
- To run tcl file **: #ns filename.tcl**
- To open awk file **: #gedit filename.awk**
- To run awk file and save contents in trace file: **awk –f filename.awk filename.tr**
- To plot the graph **: awk –f filename.awk filename.tr >a1**
    - Open the file a1. Remove the last two lines which contains the throughput statements and save the file
    - To display graph # **xgraph a1**

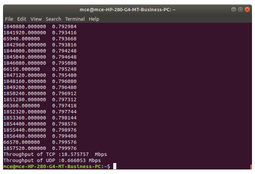
Output:

1. Simulator



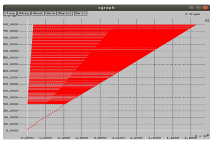
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

2. Awk file output


mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

```
1840880.000000  0.792984
1841920.000000  0.793416
65940.000000    0.793668
1842960.000000  0.793816
1844000.000000  0.794248
1845040.000000  0.794648
1846080.000000  0.795080
66150.000000    0.795248
1847120.000000  0.795480
1848160.000000  0.796080
1849200.000000  0.796480
1850240.000000  0.796912
1851280.000000  0.797312
66360.000000    0.797418
1852320.000000  0.797744
1853360.000000  0.798144
1854400.000000  0.798576
1855440.000000  0.798976
1856480.000000  0.799408
66570.000000    0.799576
1857520.000000  0.799976
Throughput of TCP :18.575757  Mbps
Throughput of UDP :0.666053 Mbps
mce@mce-HP-280-G4-MT-Business-PC:~$
```

3. Graph



6. Simulate a point-to-point network using n nodes and set multiple traffic and plot Packet delivery ratio , End to End delay and throughput for different source/destination.

Program:

```
set ns [new Simulator]
set nf [open first.nam w]
$ns namtrace-all $nf
set tf [open p6.tr w]
$ns trace-all $tf
proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam first.nam &
exit 0
}
set n0 [$ns node]
$n0 label "s"
set n1 [$ns node]
$n1 label "d"
set n2 [$ns node]
$n2 label "i1"
set n3 [$ns node]
$n3 label "i2"
$ns duplex-link $n0 $n2 10Mb 1ms DropTail
$ns duplex-link $n1 $n2 10Mb 1ms DropTail
$ns duplex-link $n2 $n3 10Mb 1ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$ns connect $tcp0 $sink0
$ns connect $udp1 $null0
$ns at 0.1 "$cbr1 start"
$ns at 0.1 "$ftp0 start"
$ns at 0.5 "finish"
$ns run
```

Awk file:

Throughput.awk

```
BEGIN{
d=0;
tcp=0;
udp=0;
```

```
pkt_t=0;
time_t=0;
pkt_u=0;
time_u=0;
}
{
{
if(($1=="r"&& $3=="0"&& $4=="2"&& $5=="tcp")||($1=="r"&& $3=="2"&& $4=="3"&& $5=="tcp"))
        { pkt_t=pkt_t+$6;
         time_t=$2;
        printf("%f\t%f\n",pkt_t,time_t);
        }
if(($1=="r"&& $3=="1"&& $4=="2"&& $5=="cbr")|| ($1=="r"&& $3=="2"&& $4=="3"&& $5=="cbr"))
        { pkt_u=pkt_u+$6;
         time_u=$2;
                printf("%f\t%f\n",pkt_u,time_u);
        }
}
}
END{
printf("Throughput of TCP :%f Mbps\n",((pkt_t/time_t)*(8/1000000)));
}
```

2.End to End delay

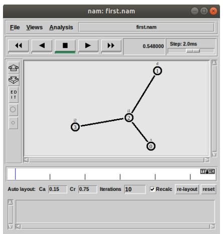
```
BEGIN {
a=0.0;
s=0.0;
n=0;
e=0;
}
{
if($3 == "0"&& $4=="2")
{
n++;
}
if($3 == "0"&& $1=="+")
{
a=a+$2;
}
if($1=="r")
{
s=s+$2;
}
e=(a-s)/n;
printf("%d\t%d\n", a,s);
}
END {
printf("arrival time=%f\n",a);
printf("sent time=%f\n",s );
printf("End to ENd delay=%f for %d connections\n",e,n);
}
```

3. PDR.awk

```
BEGIN{
pdr=0.0;
s=0.0;
r=0.0;
}
{ {
        if($1=="+")
        { s++;
        }
else if($1=="r")
        { r++;
        }
pdr=r/s;
}
}
END{
printf("Sent=%d\n",s);
printf("Recv packets=%d\n",r);
printf("PDR=%f\n",pdr);
}
```

Output:

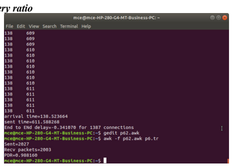
1. Simulator



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

2. Awk file output

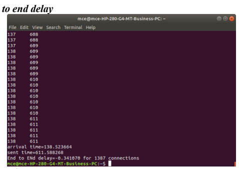
I. Packet delivery ratio



```
138     609
138     609
138     609
138     610
138     610
138     610
138     610
138     610
138     610
138     610
138     611
138     611
138     611
138     611
138     611
arrival time=138.523664
sent time=611.588268
End to ENd delay=-0.341070 for 1387 connections
mce@mce-HP-280-G4-MT-Business-PC:~$ gedit p62.awk
mce@mce-HP-280-G4-MT-Business-PC:~$ awk -f p62.awk p6.tr
Sent=2027
Recv packets=2003
PDR=0.988160
mce@mce-HP-280-G4-MT-Business-PC:~$
```

mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

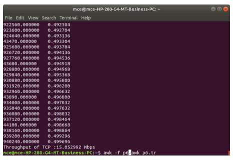
II. End to end delay



```
137     608
137     608
137     609
138     609
138     609
138     609
138     609
138     609
138     610
138     610
138     610
138     610
138     610
138     610
138     610
138     611
138     611
138     611
138     611
138     611
arrival time=138.523664
sent time=611.588268
End to ENd delay=-0.341070 for 1387 connections
mce@mce-HP-280-G4-MT-Business-PC:~$
```

mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

III. Throughput



```
922560.000000   0.492304
923600.000000   0.492704
924640.000000   0.493136
43470.000000    0.493304
925680.000000   0.493704
926720.000000   0.494136
927760.000000   0.494536
43680.000000    0.494918
928800.000000   0.494968
929840.000000   0.495368
930880.000000   0.495800
931920.000000   0.496200
932960.000000   0.496632
43890.000000    0.496800
934000.000000   0.497032
935040.000000   0.497632
936080.000000   0.498032
937120.000000   0.498464
44100.000000    0.498668
938160.000000   0.498864
939200.000000   0.499296
940240.000000   0.499696
Throughput of TCP :15.052992 Mbps
mce@mce-HP-280-G4-MT-Business-PC:~$ awk -f p6.awk p6.tr
```

mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

7. Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion. And also plot the Congestion graph.

```
set ns [new Simulator]
set nf [open lab7.nam w]
$ns namtrace-all $nf
set tf [open lab7.tr w]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
set p1 [new Agent/Ping]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001
set p2 [new Agent/Ping]
$ns attach-agent $n1 $p2
set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p1 set interval_ 0.00001
set p4 [new Agent/Ping]
$ns attach-agent $n3 $p4
set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instproc recv { from rtt } {
$self instvar node_
puts "node [ $node_ id ] recieved answer from $from with round trip time $rtt msec"
}
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
exec nam lab7.nam &
exit 0
}
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
```

```
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "$p1 send"
$ns at 2.1 "$p1 send"
$ns at 2.2 "$p1 send"
$ns at 2.3 "$p1 send"
$ns at 2.4 "$p1 send"
$ns at 2.5 "$p1 send"
$ns at 2.6 "$p1 send"
$ns at 2.7 "$p1 send"
$ns at 2.8 "$p1 send"
$ns at 2.9 "$p1 send"
$ns at 0.1 "$p3 send"
$ns at 0.2 "$p3 send"
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"
$ns at 0.5 "$p3 send"
$ns at 0.6 "$p3 send"
$ns at 0.7 "$p3 send"
$ns at 0.8 "$p3 send"
$ns at 0.9 "$p3 send"
$ns at 1.0 "$p3 send"
$ns at 1.1 "$p3 send"
$ns at 1.2 "$p3 send"
$ns at 1.3 "$p3 send"
$ns at 1.4 "$p3 send"
$ns at 1.5 "$p3 send"
$ns at 1.6 "$p3 send"
$ns at 1.7 "$p3 send"
$ns at 1.8 "$p3 send"
$ns at 1.9 "$p3 send"
$ns at 2.0 "$p3 send"
$ns at 2.1 "$p3 send"
$ns at 2.2 "$p3 send"
$ns at 2.3 "$p3 send"
$ns at 2.4 "$p3 send"
$ns at 2.5 "$p3 send"
$ns at 2.6 "$p3 send"
$ns at 2.7 "$p3 send"
```

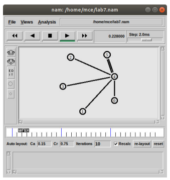
```
$ns at 2.8 "$p3 send"
$ns at 2.9 "$p3 send"
$ns at 3.0 finish
$ns run
```

Awk File:

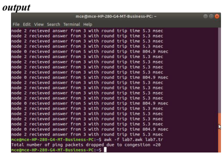
```
BEGIN{
drop=0;
}
{
if($1=="d")
{ drop++;
}
}
END{
printf("Total number of %s packets dropped due to congestion =%d \n",$5,drop);
}
```

Output:

1. Simulator



2. Awk file output



```
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 0 recieved answer from 5 with round trip time 804.9 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 0 recieved answer from 5 with round trip time 804.9 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 0 recieved answer from 5 with round trip time 804.9 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
node 0 recieved answer from 5 with round trip time 804.9 msec
node 2 recieved answer from 3 with round trip time 5.3 msec
mce@mce-HP-280-G4-MT-Business-PC:~$ awk -f lab7.awk lab7.tr
Total number of ping packets dropped due to congestion =20
mce@mce-HP-280-G4-MT-Business-PC:~$
```

mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

8. Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput. And also plot the graph for different throughputs.

```
set ns [new Simulator]
set tf [open lab8.tr w]
$ns trace-all $tf
set nf [open lab8.nam w]
$ns namtrace-all $nf
$ns color 0 blue

set n0 [$ns node]
$n0 color "red"
set n1 [$ns node]
$n1 color "red"
set n2 [$ns node]
$n2 color "red"
set n3 [$ns node]
$n3 color "red"
set n4 [$ns node]
$n4 color "magenta"
set n5 [$ns node]
$n5 color "magenta"
set n6 [$ns node]
$n6 color "magenta"
set n7 [$ns node]
$n7 color "magenta"
$ns make-lan "$n0 $n1 $n2 $n3" 100Mb 300ms LL Queue/DropTail Mac/802_3
$ns make-lan "$n4 $n5 $n6 $n7" 100Mb 300ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n3 $n4 100Mb 300ms DropTail
$ns duplex-link-op $n3 $n4 color "green"

set err [new ErrorModel]
$ns lossmodel $err $n3 $n4
$err set rate_ 0.1
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set fid_ 0
$cbr set packetSize_ 1000
$cbr set interval_ 0.0001
set null [new Agent/Null]
$ns attach-agent $n7 $null
$ns connect $udp $null
proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab8.nam &
exit 0
```

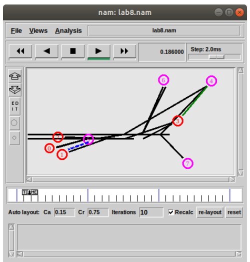
```
}
$ns at 0.1 "$cbr start"
$ns at 3.0 "finish"
$ns run
```

Awk File

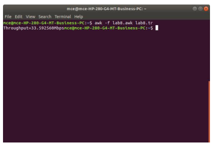
```
BEGIN{
pkt=0;
time=0;
}
{ if($1=="r"&& $3=="9"&& $4=="7"){
pkt=pkt+$6;
time=$2;
}
}
END{
printf("Throughput=%fMbps",((pkt/time)*(8/1000000)));
}
```

Output:

1. Simulator



2. Awk file output



```
mce@mce-HP-280-G4-MT-Business-PC:~$ awk -f lab8.awk lab8.tr
Throughput=33.592560Mbpsmce@mce-HP-280-G4-MT-Business-PC:~$
```

mce@mce-HP-280-G4-MT-Business-PC: ~
File Edit View Search Terminal Help

9. Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
set ns [ new Simulator]
set tf [ open lab9.tr w]
$ns trace-all $tf
set nf [open lab9.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
$n0 color "magenta"
$n0 label "SRC1"
set n1 [$ns node]
set n2 [$ns node]
$n2 color "magenta"
$n2 label "SRC2"
set n3 [$ns node]
$n3 color "blue"
$n3 label "DEST2"
set n4 [$ns node]
set n5 [$ns node]
$n5 color "blue"
$n5 label "DEST1"
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
$ns connect $tcp0 $sink5
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp2 $sink3
set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
$tcp2 attach $file2

$tcp0 trace cwnd_
$tcp2 trace cwnd_
proc finish {} {
global ns nf tf
$ns flush-trace
```

```
close $tf
close $nf
exec nam lab9.nam &
exit 0
}
$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp2 start"
$ns at 8 "$ftp2 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run
```

Awk File

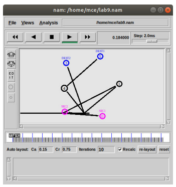
```
BEGIN{
}
{ if($6=="cwnd_")
printf("%f\t%f\t\n",$1,$7);
}
END{
}
```

Steps for Execution

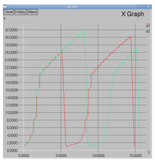
1. Run Simulator
2. In terminal
   awk –f lab9.awk lab9.tr> a1
   awk –f lab9.awk lab9.tr> a2
3. To plot graph
   xgraph a1 a2

Output:

1. Simulator



3. xgraph



10. Simulate simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

```
set ns [new Simulator]
set tf [open lab10.tr w]
$ns trace-all $tf
set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open lab10.nam w]
$ns namtrace-all-wireless $nf 1000 1000
$ns node-config -adhocRouting DSDV \
            -llType LL \
            -macType Mac/802_11 \
            -ifqType Queue/DropTail \
            -ifqLen 50 \
            -phyType Phy/WirelessPhy \
            -channelType Channel/WirelessChannel \
            -propType Propagation/TwoRayGround \
            -antType Antenna/OmniAntenna \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON
create-god 3
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 label "tcp0"
$n1 label "sink1/tcp1"
$n2 label "sink2"
$n0 set X_ 50
$n0 set Y_ 50
```

```
$n0 set Z_ 0
$n1 set X_ 100
$n1 set Y_ 100
$n1 set Z_ 0
$n2 set X_ 600
$n2 set Y_ 600
$n2 set Z_ 0
$ns at 0.1 "$n0 setdest 50 50 15"
$ns at 0.1 "$n1 setdest 100 100 25"
$ns at 0.1 "$n2 setdest 600 600 25"
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink1
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink2
$ns connect $tcp1 $sink2
$ns at 5 "$ftp0 start"
$ns at 5 "$ftp1 start"
$ns at 100 "$n1 setdest 550 550 15"
$ns at 190 "$n1 setdest 70 70 15"
proc finish {} {
global ns nf tf
$ns flush-trace
exec nam lab10.nam &
close $tf
close $nf
exit 0awk
}
$ns at 250 "finish"
$ns run
```

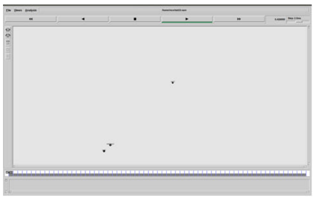
Awk File:

```
BEGIN{
count1=0;
count2=0;
pack1=0;
pack2=0;
time1=0;
time2=0;
}
{ if($1=="r"&& $3=="_1_"&& $4=="AGT")
{ count1++;
pack1=pack1+$8;
time1=$2;
}
```

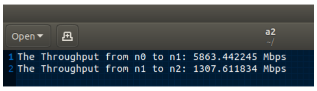
```
if($1=="r"&& $3=="_2_"&& $4=="AGT")
{ count2++;
pack2=pack2+$8;
time2=$2;
}
}
END{
printf("The Throughput from n0 to n1: %f Mbps\n",((count1*pack1*8)/(time1*1000000)));
printf("The Throughput from n1 to n2: %f Mbps\n",((count2*pack2*8)/(time2*1000000)));
}
```

Output:

1. Simulator



2. Awk file output



```
The Throughput from n0 to n1: 5863.442245 Mbps
The Throughput from n1 to n2: 1307.611834 Mbps
```