PART-A

Implement the following Computer Networks concepts using C/C++

A-1. Write a program for distance vector algorithm to find suitable path for transmission.

SOURCE CODE:

FILENAME: pA_1.c

```
#include<stdio.h>
#include<stdlib.h>
void rout_table();
int d[10][10],via[10][10];
int i,j,k,l,m,n,g[10][10],temp[10][10],ch,cost;
int main()
{
      system("clear");
      printf("enter the value of no. of nodes\n");
      scanf("%d",&n);
      rout_table();
      for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                  temp[i][j]=g[i][j];
      for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                  via[i][j]=i;
      while(1)
      {
            for(i=0;i<n;i++)
                  for(j=0;j<n;j++)
                 if(d[i][j])
                     for(k=0;k<n;k++)
                         if(g[i][j]+g[j][k]<g[i][k])
                         {
                             g[i][k]=g[i][j]+g[j][k];
                             via[i][k]=j;
                         }
            for(i=0;i<n;i++)
        {
            printf("table for router %c\n" ,i+97);
            for(j=0;j<n;j++)
                printf("%c:: %d via %c\n"
,j+97,g[i][j],via[i][j]+97);
        }
        break;
    }
}
```

```
void rout_table()
{
    printf("\nEnter the routing table : \n");
    printf("\t|");
    for(i=1;i<=n;i++)
    {
        printf("%c\t",i+96);
        printf("\n");
    }
    for(i=0;i<=n;i++)
    {
        printf("-------");
        printf("\n");
    }
    for(i=0;i<n;i++)
    {
        printf("%c|",i+97);
        for(j=0;j<n;j++)
        {
            scanf("%d",&g[i][j]);
            if(g[i][j]!=999)
                d[i][j]=1;
        }
    }
}
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pA_1$ gcc pA_1.c
nanmolrao@aloo:~/mca_1_network_lab/pA_1$ ./a.out
enter the value of no. of nodes
4

Enter the routing table :
        |a      b       c       d
-----------------------------------
a |0 5 1 4
b |5 0 6 2
c |1 6 0 3
d |4 2 3 0
table for router a
a:: 0 via a
b:: 5 via a
c:: 1 via a
d:: 4 via a
table for router b
a:: 5 via b
b:: 0 via b
c:: 5 via d
d:: 2 via b
table for router c
a:: 1 via c
b:: 5 via d
c:: 0 via c
d:: 3 via c
table for router d
a:: 4 via d
b:: 2 via d
c:: 3 via d
d:: 0 via d
```

A-2. 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.

SOURCE CODE:

FILENAME: pA_2_c.c(Client)

```
#include<stdlib.h>
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#define SERV_TCP_PORT 6879
#define SERV_HOST_ADDR "127.0.0.1"
int main()
{
    int sockfd;
    struct sockaddr_in serv_addr,cli_addr;
    char filename[100],buf[1000];
    int n;
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(SERV_HOST_ADDR);
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
        {
            printf("Client:cant open stream socket\n");
            exit(1);
        }
    else
        printf("Client:stream socket opened successfully\n");
    if(connect(sockfd,(struct sockaddr
*)&serv_addr,sizeof(serv_addr))<0)
    {
        printf("Client:cant connect to server\n");
        exit(1);
    }
```

```
else
        {
            printf("Client:connected to server
successfully\n");
            printf("\n Enter the file name to be displayed
:");
            scanf("%s",filename);
            write(sockfd,filename,strlen(filename));
            printf("\n filename transfered to server\n");
            n=read(sockfd,buf,1000);
            if(n < 0)
                printf("\n error reading from socket");
            else
                printf("\n Client : Displaying file content
of %s\n",filename);
                fputs(buf,stdout);
                close(sockfd);
            exit(0);
        }
}
```

FILENAME: pA_2_s.c(Server)

```
#include<stdlib.h>
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<string.h>
#define SERV_TCP_PORT 6879
#define SERV_HOST_ADDR "127.0.0.1"
int main()
{
    int sockfd,newsockfd,clilen;
    struct sockaddr_in cli_addr,serv_addr;
    char filename[25],buf[1000];
    int n,m=0;
    int fd;
    if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
        {
            printf("server:cant open stream socket\n");
```

```
exit(1);
        }
    else
        printf("server:stream socket opened successfully\n");
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    if((bind(sockfd,(struct sockaddr
*)&serv_addr,sizeof(serv_addr)))<0)
        {
            printf("server:cant bind local address\n");
            exit(1);
        }
    else
        printf("server:bound to local address\n");
    listen(sockfd,5);
    printf("\n SERVER : Waiting for client...\n");
    for(;;)
    {
        clilen=sizeof(cli_addr);
        newsockfd=accept(sockfd,(struct sockaddr *)
&cli_addr,&clilen);
        if(newsockfd<0)
            {
                printf("server:accept error\n");
                exit(1);
            }
        else
            printf("server:accepted\n");
        n=read(newsockfd,filename,25);
        filename[n]='\0';
        printf("\n SERVER : %s is found and ready to
transfer\n",filename);
        fd=open(filename,O_RDONLY);
        n=read(fd,buf,1000);
        buf[n]='\0';
        write(newsockfd,buf,n);
        printf("\n transfer success\n");
        close(newsockfd);
        exit(0);
    }
}
```

OUTPUT:

TERMINAL-1

```
nanmolrao@aloo:~/mca_1_network_lab/pA_2$ gcc pA_2_s.c -o server.out
nanmolrao@aloo:~/mca_1_network_lab/pA_2$ ./server.out
server:stream socket opened successfully
server:bound to local address

 SERVER : Waiting for client...
```

TERMINAL-2

```
nanmolrao@aloo:~/mca_1_network_lab/pA_2$ gcc pA_2_c.c -o client.out
nanmolrao@aloo:~/mca_1_network_lab/pA_2$ ./client.out
Client:stream socket opened successfully
Client:connected to server successfully

 Enter the file name to be displayed :TEXTFILE.txt

 filename transfered to server

 Client : Displaying file content of TEXTFILE.txt
What is a paragraph?
Paragraphs are the building blocks of papers.
Many students define paragraphs in terms of length: a paragraph is a
group of at least five sentences, a paragraph is half a page long, etc.
In reality, though, the unity and coherence of ideas among sentences is
what constitutes a paragraph.
A paragraph is defined as “a group of sentences or a single sentence
that forms a unit” (Lunsford and Connors 116).
Length and appearance do not determine whether a section in a paper is
a paragraph.
For instance, in some styles of writing, particularly journalistic
styles, a paragraph can be just one sentence long.
Ultimately, a paragraph is a sentence or group of sentences that
support one main idea.
In this handout, we will refer to this as the “controlling idea,”
because it controls what happens in the rest of the paragraph.
```

TERMINAL-1

```
nanmolrao@aloo:~/mca_1_network_lab/pA_2$ ./server.out
server:stream socket opened successfully
server:bound to local address

 SERVER : Waiting for client...
server:accepted

 SERVER : TEXTFILE.txt is found and ready to transfer

 transfer success
```

A-3. Write a program for Hamming code generation for error detection and correction.

SOURCE CODE:

FILENAME: pA_3.c

```
#inlcude<stdlib.h>
#include<stdio.h>
#include<stdlib.h>
char data[5];
int encoded[8],edata[7],syndrome[3];
int hmatrix[3][7] = {
1,0,0,0,1,1,1,
0,1,0,1,0,1,1,
0,0,1,1,1,0,1
};
char gmatrix[4][8]={"0111000","1010100","1100010","1110001"};
int main()
{
    int i,j;
    system("clear");
    printf("\nHamming code----- Encoding\n");
    printf("Enter 4 bit data : ");
    scanf("%s",data);
    printf("\nGenerator matrix\n");
    for(i=0;i<4;i++)
        printf("%s\n",gmatrix[i]);
    printf("\nEncoded data ");
    for(i=0;i<7;i++)
    {
        for(j=0;j<4;j++)
        encoded[i]+=((data[j]-'0')*(gmatrix[j][i]-'0'));
        encoded[i]=encoded[i]%2;
        printf("%d ",encoded[i]);
    }
    printf("\nHamming code----- Decoding\n");
    printf("Enter encoded bits as recieved : ");
    for(i=0;i<7;i++)
        scanf("%d",&edata[i]);
    for(i=0;i<3;i++)
    {
        for(j=0;j<7;j++)
        syndrome[i]+=(edata[j]*hmatrix[i][j]);
        syndrome[i]=syndrome[i]%2;
    }
    for(j=0;j<7;j++)
        if((syndrome[0]==hmatrix[0][j]) &&
(syndrome[1]==hmatrix[1][j])&& (syndrome[2]==hmatrix[2][j]))
            break;
        if(j==7)
            printf("\nError free\n");
```

```
else
        {
            printf("\nError recieved at bit number %d of data\n",j+1);
            edata[j]=!edata[j];
            printf("\nCorrect data should be : ");
            for(i=0;i<7;i++)
                printf("%d",edata[i]);
        }
        return 0;
}
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pA_3$ gcc pA_3.c
nanmolrao@aloo:~/mca_1_network_lab/pA_3$ ./a.out
Hamming code----- Encoding
Enter 4 bit data : 1101

Generator matrix
0111000
1010100
1100010
1110001

Encoded data 0 0 1 1 1 0 1
Hamming code----- Decoding
Enter encoded bits as recieved : 0 0 1 1 1 0 1

Error free
```

A-4. Write a program for congestion control using leaky bucket algorithm.

SOURCE CODE:

FILENAME: pA_4.c

```
#include<stdlib.h>
#include<stdio.h>
#include<strings.h>
#include<stdio.h>
int min(int x,int y)
{
if(x<y)
    return x;
else
    return y;
}

int main()
{
    int drop=0,mini,nsec,cap,count=0,i,inp[25],process;
    system("clear");
    printf("Enter The Bucket Size\n");
    scanf("%d",&cap);
    printf("Enter The Operation Rate\n");
    scanf("%d",&process);
    printf("Enter The No. Of Seconds You Want To Stimulate\n");
    scanf("%d",&nsec);
    for(i=0;i<nsec;i++)
    {
        printf("Enter The Size Of The Packet Entering At %dsec\n",i+1);
        scanf("%d",&inp[i]);
    }
    printf("\nSecond|Packet Recieved|Packet Sent|PacketLeft|Packet
Dropped|\n");
    printf("-----------------------------------------------------------
--\n");
    for(i=0;i<nsec;i++)
    {
        count+=inp[i];
        if(count>cap)
        {
            drop=count-cap;
            count=cap;
        }
        printf("%d",i+1);
        printf("\t%d",inp[i]);
        mini=min(count,process);
        printf("\t\t%d",mini);
        count=count-mini;
        printf("\t\t%d",count);
```

```
        printf("\t\t%d\n",drop);
        drop=0;
    }
    for(;count!=0;i++)
    {
        if(count>cap)
        {
            drop=count-cap;
            count=cap;
        }
        printf("%d",i+1);
        printf("\t0");
        mini=min(count,process);
        printf("\t\t%d",mini);
        count=count-mini;
        printf("\t\t%d",count);
        printf("\t\t%d\n",drop);
    }
}
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pA_4$ gcc pA_4.c
nanmolrao@aloo:~/mca_1_network_lab/pA_4$ ./a.out
Enter The Bucket Size
5
Enter The Operation Rate
2
Enter The No. Of Seconds You Want To Stimulate
3
Enter The Size Of The Packet Entering At 1sec
5
Enter The Size Of The Packet Entering At 2sec
4
Enter The Size Of The Packet Entering At 3sec
3

Second|Packet Recieved|Packet Sent|PacketLeft|Packet Dropped|
-------------------------------------------------------------
1       5               2               3               0
2       4               2               3               2
3       3               2               3               1
4       0               2               1               0
5       0               1               0               0
```

PART-B

Simulate the following Computer Networks concepts using any network simulators

B- 1. 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.

SOURCE CODE:

FILENAME: pB_1.tcl

```
set ns [new Simulator]

$ns color 2 red

$ns rtproto Static
set traceFile [open pB_1.tr w]
$ns trace-all $traceFile
set namFile [open pB_1.nam w]
$ns namtrace-all $namFile
proc finish {} {
      global ns namFile traceFile
      $ns flush-trace
      close $traceFile
      close $namFile
      exec nam pB_1.nam &
      exit 0
}

set n(1) [$ns node]
set n(2) [$ns node]
set n(3) [$ns node]

$ns duplex-link $n(1) $n(2) 0.5Mb 20ms DropTail
$ns duplex-link $n(2) $n(3) 0.5Mb 20ms DropTail
$ns queue-limit $n(1) $n(2) 10
$ns queue-limit $n(2) $n(3) 10

$n(1) shape hexagon
$n(1) color red
$n(3) shape square
$n(3) color blue

set udp0 [new Agent/UDP]
$ns attach-agent $n(1) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 512
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

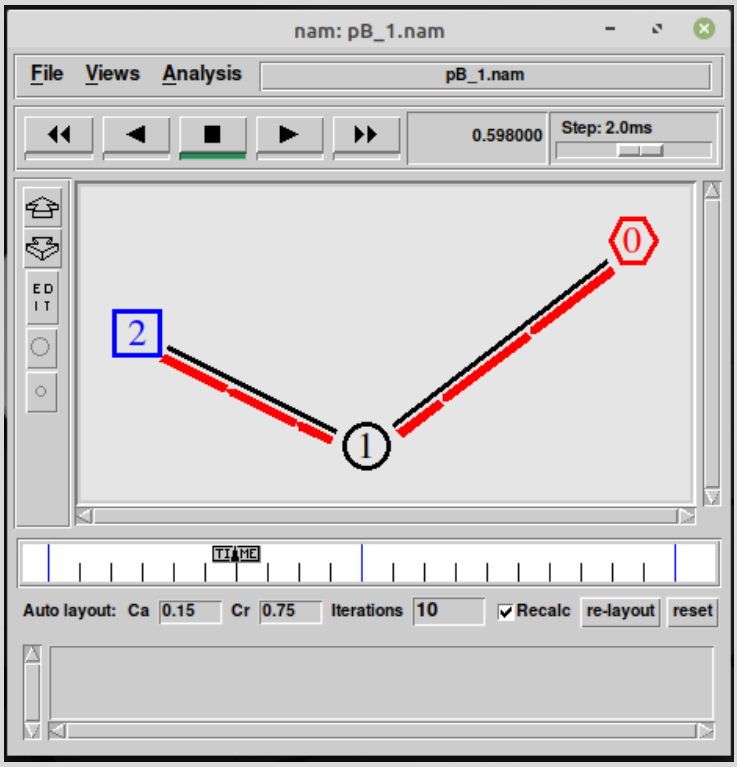
```
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

$ns connect $udp0 $null0
$udp0 set fid_ 2
$ns at 0.5 "$cbr0 start"
$ns at 2.0 "$cbr0 stop"
$ns at 2.0 "finish"

$ns run
```

FILENAME: pB_1.awk

```
BEGIN{ c=0;}
{
      if($1=="d")
      {
            c++;
            printf("%s\t%s\n",$5,$11);
      }
}
END{ printf("The number of packets dropped : %d\n",c);
      }
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pB_1$ ns pB_1.tcl
```



```
nanmolrao@aloo:~/mca_1_network_lab/pB_1$ awk -f pB_1.awk pB_1.tr
cbr   24
cbr   26
cbr   29
cbr   31
cbr   34
cbr   36
cbr   39
cbr   42
cbr   44
cbr   47
cbr   49
cbr   52
cbr   54
cbr   57
cbr   60
cbr   62
cbr   65
cbr   67
cbr   70
cbr   72
cbr   75
cbr   77
cbr   80
cbr   83
cbr   85
cbr   88
cbr   90
cbr   93
cbr   95
cbr   98
cbr   101
cbr   103
cbr   106
cbr   108
cbr   111
cbr   113
cbr   116
cbr   119
cbr   121
cbr   124
cbr   126
cbr   129
cbr   131
cbr   134
cbr   137
cbr   139
cbr   142
cbr   144
cbr   147
```

```
cbr   149
cbr   152
cbr   154
cbr   157
cbr   160
cbr   162
cbr   165
cbr   167
cbr   170
cbr   172
cbr   175
cbr   178
cbr   180
cbr   183
cbr   185
cbr   188
cbr   190
cbr   193
cbr   196
cbr   198
cbr   201
cbr   203
cbr   206
cbr   208
cbr   211
cbr   214
cbr   216
cbr   219
cbr   221
cbr   224
cbr   226
cbr   229
cbr   231
cbr   234
cbr   237
cbr   239
cbr   242
cbr   244
cbr   247
cbr   249
cbr   252
cbr   255
cbr   257
cbr   260
cbr   262
cbr   265
cbr   267
cbr   270
cbr   273
cbr   275
```

```
cbr   280
cbr   283
cbr   285
cbr   288
cbr   291
cbr   293
cbr   296
cbr   298
The number of packets dropped : 108
```

B-2. Simulate the network with five nodes n0, n1, n2, n3, n4, forming a star topology. The node n4 is at the centre. Node n0 is a TCP source, which transmits packets to node n3 (a TCP sink) through the node n4. Node n1 is another traffic source, and sends UDP packets to node n2 through n4. The duration of the simulation time is 10 seconds.

SOURCE CODE:

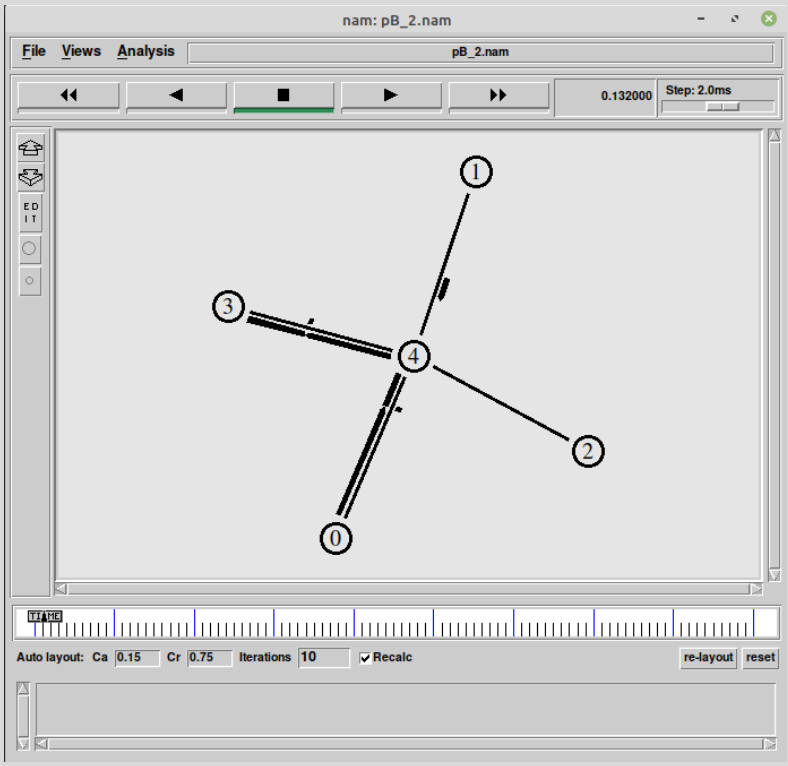
FILENAME: pB_2.tcl

```
set ns [new Simulator]

set nf [open pB_2.nam w]
$ns namtrace-all $nf
set tf [open pB_2.tr w]
$ns trace-all $tf

proc finish { } {
global ns tf nf
$ns flush-trace
close $nf
close $tf
exec nam pB_2.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n4 10Mb 1ms DropTail
$ns duplex-link $n1 $n4 10Mb 1ms DropTail
$ns duplex-link $n4 $n3 10Mb 1ms DropTail
$ns duplex-link $n4 $n2 10Mb 1ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0

$ns connect $tcp0 $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 $n2 $null0
```

```
$ns connect $udp0 $null0

set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp0
$ns at 0.0 "$cbr1 start"
$ns at 0.0 "$ftp0 start"
$ns at 9.0 "$cbr1 stop"
$ns at 9.0 "$ftp0 stop"
$ns at 10.0 "finish"

$ns run
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pB_2$ ns pB_2.tcl
```




```
nanmolrao@aloo:~/mca_1_network_lab/pB_2$ awk -f pB_2.awk pB_2.tr
Number of packets sent by TCP: 21638
Number of packets sent by UDP: 4800
```

B-3. Simulate to study transmission of packets over Ethernet LAN and determine the number of packets drop destination.

SOURCE CODE:

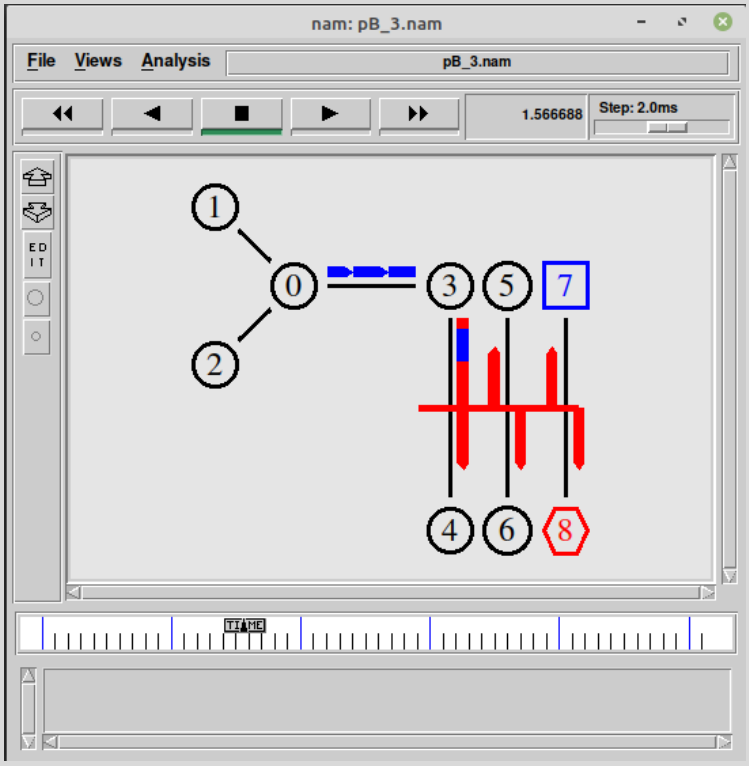
FILENAME: pB_3.tcl

```
set ns [new Simulator]

set nf [open pB_3.nam w]
$ns namtrace-all $nf

set nd [open pB_3.tr w]
$ns trace-all $nd

$ns color 1 Blue
$ns color 2 Red

proc finish { } {
      global ns nf nd
      $ns flush-trace
      close $nf
      close $nd
      exec nam pB_3.nam &
      exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

$n7 shape box
$n7 color Blue
$n8 shape hexagon
$n8 color Red

$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail

$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL Queue/DropTail
Mac/802_3

$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right
```

```
$ns queue-limit $n0 $n3 20

set tcp1 [new Agent/TCP/Vegas]
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1
$tcp1 set packetsize_ 55

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set tfile [open pB_3_tcp.tr w]
$tcp1 attach $tfile
$tcp1 trace pB_3_tcp_

set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2
$ns connect $tcp2 $sink2

$tcp2 set class_ 2
$tcp2 set packetsize_ 55

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

set tfile2 [open pB_3_tcp2.tr w]
$tcp2 attach $tfile2
$tcp2 trace pB_3_tcp2_

$ns at 0.5 "$ftp1 start"
$ns at 1.0 "$ftp2 start"
$ns at 5.0 "$ftp2 stop"
$ns at 5.0 "$ftp1 stop"

$ns at 5.5 "finish"
$ns run
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pB_3$ ns pB_3.tcl
```



B-4. Simulate working of multicasting routing protocol and analyse the throughput of the network/protocol.

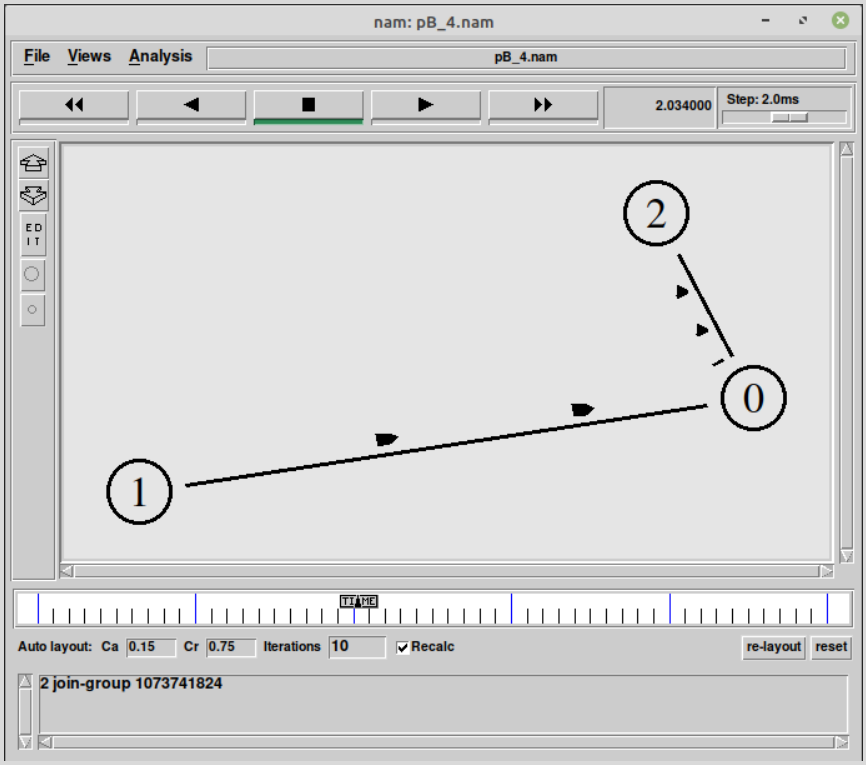
SOURCE CODE:

FILENAME: pB_4.tcl

```
set ns [new Simulator -multicast on];

set trace [open pB_4.tr w]
$ns trace-all $trace
set namtrace [open pB_4.nam w]
$ns namtrace-all $namtrace
set group [Node allocaddr];
set n0 [$ns node];
set n1 [$ns node]
set n2 [$ns node]

$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail

set mproto DM;
set mrthandle [$ns mrtproto $mproto];

set udp [new Agent/UDP]
$ns attach-agent $n0 $udp

set src [ new Application/Traffic/CBR]
$src attach-agent $udp

$udp set dst_addr_ $group
$udp set dst_port_ 0

set rcvr1 [new Agent/LossMonitor]
$ns attach-agent $n1 $rcvr1
set rcvr2 [new Agent/LossMonitor]
$ns attach-agent $n2 $rcvr2

$ns at 0.3 "$n2 join-group $rcvr2 $group"
$ns at 2.0 "$src start"
$ns at 3.3 "$n2 leave-group $rcvr2 $group"
$ns at 5.0 "$src stop"
proc finish { } {
global ns namtrace trace
$ns flush-trace
close $namtrace
close $trace
exec nam pB_4.nam &
exit 0
}
$ns at 10.0 "finish"
$ns run
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pB_4$ ns pB_4.tcl
```



B-5. Simulate the different types of internet traffic such as FTP and TELNET over a wired network and analyze the packet drop and packet delivery ratio in the network.

SOURCE CODE:

FILENAME: pB_5.tcl

```
set ns [new Simulator]

set nf [open pB_5.nam w]
$ns namtrace-all $nf

set nt [open pB_5.tr w]
$ns trace-all $nt

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000

set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1

$ns connect $tcp1 $sink1

set telnet0 [new Application/Telnet]
$telnet0 set interval_ 0.005
$telnet0 attach-agent $tcp1
```

```
proc finish { } {
global ns nf nt
$ns flush-trace
close $nf
close $nt
exec nam pB_5.nam &
exec awk -f pB_5.awk pB_5.tr &
exit 0
}

$ns at 0.5 "$telnet0 start"
$ns at 0.75 "$ftp0 start"
$ns at 4.5 "$telnet0 stop"
$ns at 4.75 "$ftp0 stop"
$ns at 5.0 "finish"
$ns run
```

FILENAME: pB_5.awk

```
BEGIN{ count=0; Rcount=0;}
{
      if($1=="r" && $5=="tcp")
      {
            count=count+1;
            if($6 >= 1000)
            {
                  Rcount=Rcount+1;
            }
      }
}
END{
      printf("Total Packets in Transmission: %d\n",count);
      printf("Recieved: %d\n",Rcount);
      printf("Loss: %d\n",count-Rcount);
}
```

OUTPUT:

```
nanmolrao@aloo:~/mca_1_network_lab/pB_5$ ns pB_5.tcl
nanmolrao@aloo:~/mca_1_network_lab/pB_5$ Total Packets in
Transmission: 1073
Recieved: 1069
Loss: 4
```