M.S. RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous institute affiliated to VTU)

VIDYA SOUDHA, M.S.R NAGAR, M.S.R.I.T. POST, BANGALORE-560054

RAMAIAH

INSTITUTE OF
TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE ENGINEERING

Computer Networks Lab Manual

(2015-2016)

Semester: 5

Course coordinator:

Sandeep B L

Assistant Professor
Dept. of Information Science and Engineering
M S Ramaiah Institute of Technology, Bangalore

Manual prepared by:

Sandeep B L, S R Mani Sekhar and Suresh Kumar K R

Syllabus

Course code and Title : IS531L – CN Lab	Course Credits : 0:0:1
CIE : **50 Marks**	SEE : **50 Marks**
Total No of Lab Hours : 14 Labs (Each of 2 hours)	

Prerequisites

Prerequisite Courses with codes
IS432 – Data Communication

Course Objectives

- Design program for file transfer using the concepts of Socket programming.
- Build programs to implement congestion control techniques.
- Construct programs to build optimal routing table.
- Build programs to implement error detection.
- Design sample networks with different topologies and configurations
- Analyze the network behavior with respect to different parameters and conditions.

Syllabus

Part A: Implement the following using C/C++:

1. Using TCP/IP sockets, write a client-server program to make client send the file name and the server to send back the contents of the requested file name “sample.txt” with the following contents: “Hello we are at Computer Networks Lab” Display suitable error message in case the file is not present in the server.
2. Write a program to archive Traffic management at Flow level by implementing Leaky Bucket Algorithm.
3. Given a graph, each node A knows the shortest path to node Z and node A can determine its shortest path to Z by calculating the minimum cost. Now when packet flows through a path it incurs some cost to the network, find shortest paths from src to all nodes in the given graph using Bellman Ford Algorithm. The graph may contain negative weight edges.
4. Given a graph find shortest paths from source to all nodes in the graph using Dijkstra’s shortest path algorithm.

5. Write a program for implementing the error detection technique for data transfer in unreliable network code using CRC (16-bits) Technique.
6. Write a program to implement internet checksum for error correction and detection.
7. Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Implement priority queuing as a technique to improve Quality of Service.

Part B: Simulation Using NS-2

1. Simulate three nodes point-to-point networks with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped
2. Simulate the different types of internet traffic such as FTP and TELNET over network and analyze the throughput
3. Simulate a four-node point-to-point network, and connect the links as follows: n0->n2, n1->n2 and n2->n3. Apply TCP agent changing the parameters and determine the number of packets sent/received by TCP/UDP
4. Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
5. Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and determine collision across different nodes.
6. Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput.

Text books

1. Behrouz A. Forouzan, Data Communications and Networking, Fourth Edition, Tata McGraw-Hill, 2006
2. William Stallings, Cryptography and Network security, Principles and Practices, Third Edition, PHI, 2005

Course Outcomes

The students will be able to-

1. Design and implement the functionalities of various layers of the OSI model.
2. Simulate and analyze the network behavior against various parameters using NS2.

Course Assessment

	What		To whom	When/ Where (Frequency in the course)	Max marks	Evidence collected	Contributing to Course Outcomes
Direct Assessment Methods	CIE	Internal assessment tests	Students	Twice(Part A + Part B)	20+10	Data sheets	1, 2
		Continuous Evaluation		Regular Labs	10	Records	1,2
		Viva Voce		Once	10	Online result sheet	1, 2
	SEE	Standard examination		End of course	50	Answer scripts	1, 2
Indirect Assessment Methods	Students feedback		Students	Middle of the course	-	Feedback forms	1, 2
	End of course survey			End of course	-	Feedback forms	1, 2

Program 1: Using TCP/IP sockets, write a client-server program to make client send the file name and the server to send back the contents of the requested file name “sample.txt” with the following contents: “Hello we are at Computer Networks Lab” Display suitable error message in case the file is not present in the server.

server.c

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>

int main()

{

  int cs,ns,fd,n;
  int bufsize=1024;
  char *buffer=malloc(bufsize);
  struct sockaddr_in address;
  char fname[255];
  address.sin_family=AF_INET;
  address.sin_port=htons(15000);
  address.sin_addr.s_addr=
INADDR_ANY;
  cs=socket(AF_INET,SOCK_STREAM,0);
  bind(cs,(struct sockaddr *)&address,sizeof(address));
  listen(cs,3);
  ns=accept(cs,(struct sockaddr *)NULL,NULL);
  recv(ns,fname,255,0);
  fd=open(fname,O_RDONLY);
  n=read(fd,buffer,bufsize);
  send(ns,buffer,n,0);
  close(ns);
  return close(cs);

}
```

client.c

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>

int main(int argc,char **argv)
{
  int cs,n;
  int bufsize=1024;
  char *buffer=malloc(bufsize);
  char fname[255];
  struct sockaddr_in address;
  address.sin_family=AF_INET;
  address.sin_port=htons(15000);
  inet_pton(AF_INET,argv[1],&address.sin_addr);
  cs=socket(AF_INET,SOCK_STREAM,0);
  connect(cs,(struct sockaddr *)&address,sizeof(address));
  printf("\nEnter filename: ");scanf("%s",fname);
  send(cs,fname,255,0);
  while((recv(cs,buffer,bufsize,0))>0)
  printf("%s",buffer);
  printf("\nEOF\n");
  return close(cs);
}
```

steps to execute:

```
-->netstat -tulnp
-->gcc server.c
-->./a.out 631
```

open another terminal

```
-->gcc client.c
-->./a.out 127.0.0.1
```

Program 2: Write a program to archive Traffic management at Flow level by implementing Leaky Bucket Algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#define MIN(x,y) (x>y)?y:x

int main()
{
int orate,drop=0,cap,x,count=0,
inp[10]={0},i=0,nsec,ch;
printf(" \n enter bucket size : ");
scanf("%d",&cap);
printf("\n enter output rate :");
scanf("%d",&orate);
do{
printf("\n enter number of packets coming at second %d : ",i+1);
scanf("%d",&inp[i]);
i++;
printf("\n enter 1 to contiue or 0 to quit..........");
scanf("%d",&ch);
}while(ch);
nsec=i;
printf("\n second \t recieved \t sent \t dropped \t remained \n");
for(i=0;count || i<nsec;i++)
{
printf("%d",i+1);
printf(" \t %d\t ",inp[i]);
printf(" \t %d\t ",MIN((inp[i]+count),orate));
if((x=inp[i]+count-orate)>0)
{
if(x>cap)
{
count=cap;
drop=x-cap;
}
else
{
count=x;
drop=0;
}
}
else
{
drop=0;
count=0;
}
printf(" \t %d \t %d \n",drop,count);
}
return 0;
}
```

Program 3: Given a graph, each node A knows the shortest path to node Z and node A can determine its shortest path to Z by calculating the minimum cost. Now when packet flows through a path it incurs some cost to the network, find shortest paths from src to all nodes in the given graph using Bellman Ford Algorithm. The graph may contain negative weight edges.

```
#include<stdio.h>
struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];
int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);//Enter the nodes
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j];
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {
                    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                    rt[i].from[j]=k;
                    count++;
                }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
           printf("\n\n For router %d\n",i+1);
           for(j=0;j<nodes;j++)
           {
               printf("\t\nnode %d via %d Distance %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
           }
        }
    printf("\n\n");

}
```

Program 4: Given a graph find shortest paths from source to all nodes in the graph using Dijkstra's shortest path algorithm.

```
#include <stdio.h>
#define infinity 999
void dij(int n,int v,int cost[10][10],int dist[])
{
 int i,u,count,w,flag[10],min;
 for(i=1;i<=n;i++)
  flag[i]=0,dist[i]=cost[v][i];
 count=2;
 while(count<=n)
 {
  min=99;
  for(w=1;w<=n;w++)
   if(dist[w]<min && !flag[w])
    min=dist[w],u=w;
  flag[u]=1;
  count++;
  for(w=1;w<=n;w++)
   if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
    dist[w]=dist[u]+cost[u][w];
 }
}

void main()
{
 int n,v,i,j,cost[10][10],dist[10];

 printf("\n Enter the number of nodes:");
 scanf("%d",&n);
 printf("\n Enter the cost matrix:\n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
  {
   scanf("%d",&cost[i][j]);
   if(cost[i][j]==0)
    cost[i][j]=infinity;
  }
 printf("\n Enter the source node:");
 scanf("%d",&v);
 dij(n,v,cost,dist);
 printf("\n Shortest path:\n");
 for(i=1;i<=n;i++)
  if(i!=v)
 printf("%d->%d,cost=%d\n",v,i,dist[i]);
}
```

Program 5: Write a program for implementing the error detection technique for data transfer in unreliable network code using CRC (16-bits) Technique.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<time.h>

void divide(char agdtw[],char divs[],char remd[])

{
   int i,r,l,a,t;
   r=strlen(divs);
   t=strlen(agdtw)-r+1;
   char divd[18],rem[18];
   strncpy(divd,agdtw,r);
   divd[r]='\0';
   l=0;
   memset(rem, 0,18);
   while(l<t)
   {
      a=0;
      memset(rem, 0,18);
      if(divd[0]==divs[0])
      {
         for(i=1;i<r;i++)
         {
            if(divd[i]==divs[i])
               rem[a++]='0';
            else
               rem[a++]='1';
         }
         rem[a]='\0';
         strcpy(divd,rem);
      }
      else
      {
         strncpy(divd,&divd[1],strlen(divd)-1);
         divd[r-1]='\0';
      }
      int o=strlen(divd);
      divd[o]=agdtw[l+r];
```

```
        divd[r]='\0';
        l++;
    }

    strncpy(remd,divd,r-1);
    remd[r-1]='\0';
}

void binary(char letter,char bin[])
{
    int t,c,i=7;
    c=(int)letter;
    while(i>=0)
    {
        t=c%2;
        c=c/2;
        bin[i--]=t+'0';
    }
    bin[8]='\0';
}

char ascii(char bin[])
{
    int t=0,c,i=7;
    while(i>=0)
    {
        t=t+pow(2,7-i)*(bin[i]-'0');
        i--;
    }
    return t;
}

void main()
{
char dw[126],augdw[1018],div[18],rem[18],cw[1018],rcw[1018],bin[9],rdw[1001],msg[126];
    printf("Enter a Message to be sent (Max 125 Char)\n");
    fgets(dw, sizeof(dw), stdin);
    binary(dw[0],bin);
    strcpy(augdw,bin);
    int j,k,e;
    for(j=1;j<strlen(dw);j++)
    {
        binary(dw[j],bin);
        strcat(augdw,bin);
    }
    strcat(augdw,"0000000000000000");
    printf("\nEnter Divisor (generator) of 17 bits\n");
    scanf("%s",div);
    divide(augdw,div,rem);
    strcpy(cw,augdw);
    strcpy(&cw[strlen(augdw)-16],rem);
    strcpy(rcw,cw);
    printf("\nEnter no. of errors to be introduced during transmission :");
    scanf("%d",&e);
    srand(time(0));
    for(j=0;j<e;j++)
    {
        k=rand()%strlen(rcw)-1;
        if(rcw[k]=='0')
            rcw[k]='1';
        else
            rcw[k]='0';
        printf("Error Generated at %d th bit %d thcharacter\n",k,(k/8)+1);
    }
    divide(rcw,div,rem);
    if(strcmp(rem,"0000000000000000")!=0)
        printf("\n\nErroneous Transmission detected!\n");
    strncpy(rdw,rcw,strlen(rcw)-16);
    rdw[strlen(rcw)-16]='\0';
    for(j=0,k=0;j<strlen(rdw);j=j+8)
    {
        strncpy(bin,&rdw[j],8);
        bin[8]='\0';
        msg[k++]=ascii(bin);
    }
    msg[k]='\0';
    printf("\nRecieved Message = %s\n\n",msg);
}
```

```
gcc prog5.c -lm
```

Program 6: Write a program to implement internet checksum for error correction and detection.

```
#include<stdio.h>
#include<string.h>
int checksum(int fl)
{
char in[100];
int buf[25];
int i,sum=0,n,temp,temp1;
scanf("%s",in);
if(strlen(in)%2!=0)
        n=(strlen(in)+1)/2;
else
        n=n=(strlen(in))/2;
for(i=0;i<n;i++)
    {
temp=in[i*2];
temp=(temp*256)+in[(i*2)+1];
sum=sum+temp;
    }
if(fl==1)
    {
printf("Enter the checksum value \n");
scanf ("%x", &temp);
sum+=temp;
    }
if(sum%65536!=0)
    {
        n=sum%65536;
sum=(sum/65536) + n;
    }
sum=65535-sum;
printf("%x\n",sum);
return sum;
}
void main()
{
int ch,sum;
do{
printf("1.Encode \n2.Decode \n3.Exit \n");
scanf("%d",&ch);
switch(ch)
        {
```

```
case 1: printf("Enter the string \n");
sum=checksum(0);
printf("Checksum to append is:%x \n",sum);
break;
case 2: printf("Enter the string \n");
sum=checksum(1);
if(sum!=0)
printf("The data has been tampered with or invalid checksum\n");
else
printf("The checksum is valid \n");
break;
case 3: break;
default: printf("Invalid option, try again \n");
        }
    }
while(ch!=3);
}
```

Program 7: Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Implement priority queuing as a technique to improve Quality of Service.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
void insert_by_priority(int);
void delete_by_priority(int);
void create();
void check(int);
void display_pqueue();

int pri_que[MAX];
int front, rear;

void main()
{
   int n, ch;

   printf("\n1 - Insert an element into queue");
   printf("\n2 - Delete an element from queue");
   printf("\n3 - Display queue elements");
   printf("\n4 - Exit");

   create();

   while (1)
   {
      printf("\nEnter your choice : ");
      scanf("%d", &ch);

      switch (ch)
      {
      case 1:
         printf("\nEnter value to be inserted : ");
         scanf("%d",&n);
         insert_by_priority(n);
         break;
      case 2:
         printf("\nEnter value to delete : ");
         scanf("%d",&n);
         delete_by_priority(n);
         break;
      case 3:
         display_pqueue();
         break;
      case 4:
         exit(0);
      default:
         printf("\nChoice is incorrect, Enter a correct choice");
      }
   }
}

/* Function to create an empty priority queue */
void create()
{
   front = rear = -1;
}

/* Function to insert value into priority queue */
void insert_by_priority(int data)
{
   if (rear >= MAX - 1)
   {
      printf("\nQueue overflow no more elements can be inserted");
      return;
   }
   if ((front == -1) && (rear == -1))
   {
      front++;
      rear++;
      pri_que[rear] = data;
      return;
   }
   else
      check(data);
   rear++;
}

/* Function to check priority and place element */
void check(int data)
{
   int i,j;

   for (i = 0; i <= rear; i++)
   {
      if (data >= pri_que[i])
      {
         for (j = rear + 1; j > i; j--)
         {
            pri_que[j] = pri_que[j - 1];
         }
         pri_que[i] = data;
         return;
      }
   }
   pri_que[i] = data;
}

/* Function to delete an element from queue */
void delete_by_priority(int data)
{
   int i;

   if ((front==-1) && (rear==-1))
   {
      printf("\nQueue is empty no elements to delete");
      return;
   }

   for (i = 0; i <= rear; i++)
   {
      if (data == pri_que[i])
      {
         for (; i < rear; i++)
         {
            pri_que[i] = pri_que[i + 1];
         }

      pri_que[i] = -99;
      rear--;

      if (rear == -1)
         front = -1;
      return;
      }
   }
   printf("\n%d not found in queue to delete", data);
}

/* Function to display queue elements */
void display_pqueue()
{
   if ((front == -1) && (rear == -1))
   {
      printf("\nQueue is empty");
      return;
   }

   for (; front <= rear; front++)
```

```
    {
        printf(" %d ", pri_que[front]);
    }

    front = 0;
}
```

SIMULATION USING NS-2

Program 1: Simulate three nodes point-to-point networks with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped

1.tcl

```
set ns [new Simulator]
set nf [open prog1.nam w]
$ns namtrace-all $nf
set nd [open prog1.tr w]
$ns trace-all $nd

proc finish { } {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog1.nam &
exit 0
}

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

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 512Kb 10ms DropTail
$ns queue-limit $n1 $n2 5

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $udp0 $sink

$ns at 0.2 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

1.awk

```
BEGIN {
dcount = 0;
rcount = 0;
}
{
event = $1;
if(event == "d")
{
dcount++;
}
if(event == "r")
{
rcount++;
}
}
END {
printf("The no.of packets dropped  : %d\n ",dcount);
printf("The no.of packets recieved : %d\n ",rcount);
}
```

Program 2: Simulate the different types of internet traffic such as FTP and TELNET over network and analyze the throughput

tcl

```
#create Simulator
set ns [new Simulator]

#Open Trace and NAM Trace File
set ntrace [open ex3.tr w]
$ns trace-all $ntrace
set namfile [open ex3.nam w]
$ns namtrace-all $namfile

#Finish Procedure
proc Finish {} {
global ns ntrace namfile

#Dump all trace data and close the files
$ns flush-trace
close $ntrace
close $namfile

#Execute the nam animation file
exec nam ex3.nam &
exit 0
}

$ns color 1 Blue
$ns color 2 Red

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 1Mb 10ms DropTail
$ns simplex-link $n3 $n2 1Mb 10ms DropTail

#Set queue size and Monitor the queue
$ns queue-limit $n0 $n2 10
```

```
$ns simplex-link-op $n0 $n2 queuePos 0.5

#Set TCP Connection between n0 and n3
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
$tcp0 set fid_ 1

#Attach FTP Application over TCP
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

#Set TCP Connection between n1 and n3
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 2

#Attach Telnet Application over UDP
set telnet [new Application/Telnet]
$telnet attach-agent $tcp1
$telnet set type_ Telnet

#Schedule Events
$ns at 0.5 "$telnet start"
$ns at 0.5 "$ftp0 start"
$ns at 24.5 "$telnet stop"
$ns at 24.5 "$ftp0 stop"
$ns at 25.0 "Finish"

$ns run
```

----

awk

```
BEGIN {
numTCP1=0;
tcpSize1=0;
numTCP2=0;
tcpSize2=0;
totaltcp1=0;
totaltcp2=0;
}
{
event=$1;
pkttype= $5;
fromnode=$9;
tonode=$10;
pktsize=$6;
if(event == "r" && pkttype == "tcp" && fromnode == "0.0" && tonode == "3.0")
{
numTCP1++;
tcpSize1 = pktsize;
}
if(event == "r" && pkttype == "tcp" && fromnode == "1.0" && tonode == "3.1")
{
numTCP2++;
tcpSize2 = pktsize;
}
}
END {
totaltcp1=numTCP1*tcpSize1*8;
totaltcp2=numTCP2*tcpSize2*8;
throughputtcp1= totaltcp1/24; # because simulation time is 24.5  0.5 = 24
throughputtcp2= totaltcp2/24; # because simulation time is 24.5  0.5 = 24
printf("The Throughput of FTP application is %d \n", throughputtcp1);
printf("The Throughput of TELNET application is %d \n", throughputtcp2);
}
----------------------------------------------------------------------------------------------------------------
```

Program 3: Simulate a four-node point-to-point network, and connect the links as follows: n0->n2, n1->n2 and n2->n3. Apply TCP agent changing the parameters and determine the number of packets sent/received by TCP/UDP

```
set ns [new Simulator]
set nf [open prog2.nam w]
$ns namtrace-all $nf
set nd [open prog2.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
exec nam prog2.nam &
exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#set color to the nodes
 $n1 color blue
$n0 color red
$n2 color purple
$n3 color orange


$ns color 1 blue

$n0 label TCP
$n1 label UDP
$n3 label NULL-TCPSINK

$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms 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
$tcp0 set fid_ 1

#$tcp0 set class_ 1

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
$ns connect $udp0 $null0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

$ns at 0.2 "$cbr0 start"
$ns at 0.1 "$ftp0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 4.4 "$ftp0 stop"

$ns at 5.0 "finish"
$ns run
```

----

awk

```
BEGIN {
ctcp=0;
cudp=0;
}
{
pkt=$5;
if(pkt=="cbr") { cudp++;}
if(pkt=="tcp") { ctcp++;}
}
END {
printf("No of packets sent\nTcp : %d\nUdp : %d\n",ctcp,cudp);
}
```

Program 4: Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

```
set ns [new Simulator]
set nf [open prog4.nam w]
$ns namtrace-all $nf
set nd [open prog4.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog4.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]

$ns duplex-link $n1 $n0 1Mb 12ms DropTail
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link $n6 $n0 1Mb 11ms DropTail

Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] recieved ping answer from \
$from with round-trip-time $rtt ms."
}

set p1 [new Agent/Ping]
set p2 [new Agent/Ping]
set p3 [new Agent/Ping]
set p4 [new Agent/Ping]
set p5 [new Agent/Ping]
set p6 [new Agent/Ping]
```

```
$ns attach-agent $n1 $p1
$ns attach-agent $n2 $p2
$ns attach-agent $n3 $p3
$ns attach-agent $n4 $p4
$ns attach-agent $n5 $p5
$ns attach-agent $n6 $p6

$ns queue-limit $n0 $n4 3
$ns queue-limit $n0 $n5 2
$ns queue-limit $n0 $n6 2

$ns connect $p1 $p4
$ns connect $p2 $p5
$ns connect $p3 $p6

$ns at 0.1 "$p1 send"
$ns at 0.3 "$p2 send"
$ns at 0.5 "$p3 send"
$ns at 1.0 "$p4 send"
$ns at 1.2 "$p5 send"
$ns at 1.4 "$p6 send"
$ns at 2.0 "finish"
$ns run
---------------------
```

awk

```
BEGIN {
count=0;
}
{
event=$1;
if(event=="d")
{
count++;
}
}
END {
printf("No of packets dropped : %d\n",count);
}
----------------------------------------------------------------------------------------------------
```

Program 5: Simulate an Ethernet LAN using n nodes and set multiple traffic nodes and determine collision across different nodes.

tcl

```
#Lan simulation – mac.tcl
set ns [new Simulator]

#define color for data flows
$ns color 1 Blue
$ns color 2 Red

#open tracefile
set tracefile1 [open ex4.tr w]
$ns trace-all $tracefile1

#open nam file
set namfile [open ex4.nam w]
$ns namtrace-all $namfile

#define the finish procedure
proc finish {} {
global ns tracefile1 namfile
$ns flush-trace
close $tracefile1
close $namfile
exec nam ex4.nam &
exit 0
}

#create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# Specify color and shape for nodes
$n1 color Red
$n1 shape box
$n5 color Red
$n5 shape box
$n0 color Blue
$n4 color Blue

#create links between the nodes
```

```
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

# Create a LAN
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd
Channel]

#Give node position
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left

#setup TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set packet_size_ 552

#set ftp over tcp connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp

#setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2

#setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.05Mb
$cbr set random_ false

#scheduling the events
```

```
$ns at 0.0 "$n0 label TCP_Traffic"
$ns at 0.0 "$n1 label UDP_Traffic"
$ns at 0.3 "$cbr start"
$ns at 0.8 "$ftp start"
$ns at 7.0 "$ftp stop"
$ns at 7.5 "$cbr stop"
$ns at 8.0 "finish"
$ns run
------------------------------------------
```

awk

```
BEGIN {
pktdrp=0;
}

{
event=$1;
if(event == "d") {
pktdrp++; }
}

END {
printf("The number of packets dropped is %d\n",pktdrp);
}
 ---------------------------------------------------------------------------------------------------
```

Program 6: Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput.

tcl

```
set ns [new Simulator]
set nf [open prog5.nam w]
$ns namtrace-all $nf
set nd [open prog5.tr w]
$ns trace-all $nd

proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog5.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]

$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6" 0.2Mb 40ms LL Queue/DropTail
Mac/802_3

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns at 1.0 "$ftp start"
$ns at 5.0 "$ftp stop"
$ns at 5.5 "finish"
$ns run
------------------------------
```

----

awk

```
BEGIN {
sSize=0;
startTime = 5.0;
stopTime = 0.1;
Tput = 0;
}
{
event = $1;
time = $2;
from = $3;
to = $4;
pkt = $5;
size = $6;
fid = $7;
src = $8;
dst = $9;
seqn = $10;
pid = $11;
if (event == "+") {
if(time < startTime) {
startTime = time;
}
}
if (event == "r") {
if(time > stopTime) {
stopTime = time;
}
sSize+=size;
}
Tput = (sSize/(stopTime-startTime))*(8/1000);
printf("%f\t%.2f\n",time,Tput);
}
END {
}
```