1 Point to point network with duplex link between them

```
# Letter S is capital
set ns [new Simulator]
# open a nam trace file in write mode
# nf nam filename
# tf trace filename
set nf [open /home/mca/sandhya/1.nam w]
# clears trace file contents
$ns namtrace-all $nf
set tf [open /home/mca/sandhya/1.tr w]
$ns trace-all $tf
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam 1.nam &amp;amp;
exit 0
}
# # creates 3 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
# establishing links
# you need to modify the bandwidth to observe the variation in packet.
$ns duplex-link $n0 $n1 200Mb 10ms DropTail
$ns duplex-link $n1 $n2 100Kb 1000ms DropTail
#set queue size
$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10
set udp [new Agent/UDP]
```

```
$ns attach-agent $n0 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packet_size_ 500
# attaching transport layer protocols

set null [new Agent/Null]
$ns attach-agent $n2 $null
$ns connect $udp $null
# creating sink(destination) node
# attaching application layer protocols
$ns at 0.1 &quot;$cbr start&quot;
$ns at 1.0 &quot;finish&quot;
$ns run
```

2 Bit Stuffing Program in C

```
#include&lt;stdio.h&gt;
#include&lt;string.h&gt;
int main()
{
int a[20],b[30],i,j,k,count,n;
printf(&quot;Enter frame size (Example: 8):&quot;);
scanf(&quot;%d&quot;,&amp;n);
printf(&quot;Enter the frame in the form of 0 and 1 :&quot;);
for(i=0; i&lt;n; i++)
scanf(&quot;%d&quot;,&amp;a[i]);
i=0;
count=1;
j=0;
while(i&lt;n)
{
if(a[i]==1)
{
b[j]=a[i];
for(k=i+1; a[k]==1 &amp;&amp; k&lt;n &amp;&amp; count&lt;5; k++)
{
j++;

b[j]=a[k];
count++;
if(count==5)
{
j++;
b[j]=0;
```

```
}
i=k;
}
}
else
{
b[j]=a[i];
}
i++;
j++;
}
printf(&amp;quot;After Bit Stuffing :&amp;quot;);
for(i=0; i&amp;lt;j; i++)
printf(&amp;quot;%d&amp;quot;,b[i]);
return 0;
}
```

CHARACTER / BYTE STUFFING:

```
#include&amp;lt;stdio.h&amp;gt;
#include&amp;lt;conio.h&amp;gt;
#include&amp;lt;string.h&amp;gt;

int main()
{
char sdel[]=&amp;quot;DELSTX&amp;quot;,data[100]=&amp;quot;&amp;quot;,sdata[100]=&amp;quot;&amp;quot;;
int i=0,j=0;
printf(&amp;quot;enter the message:&amp;quot;);
scanf(&amp;quot;%s&amp;quot;,data);

// converting user input to uppercase
```

```
for(int k=0;data[k]!=&#39;\0&#39;;k++) {
if(data[k] &gt;= &#39;a&#39; &amp;&amp; data[k] &lt;= &#39;z&#39;){
data[k] = data[k] - 32;
}
}
printf(&quot;original message: %s \n&quot;, data);

if(strlen(data) &lt; 3){
strcat(sdel,data);
strcat(sdel,&quot;DLEETX&quot;);
printf(&quot;Message after character stuffing is : %s&quot;, sdel);
}
else{
while(data[i] != &#39;\0&#39;){
if(data[i] == &#39;D&#39; &amp;&amp; data[i+1] == &#39;L&#39; &amp;&amp;
data[i+2]==&#39;E&#39;){
strcat(sdata,&quot;DLEDLE&quot;);
i = i+3;
j = j+6;
continue;
}
sdata[j] = data[i];
j++;

i++;
}
strcat(sdel,sdata);
strcat(sdel,&quot;DLEETX&quot;);
printf(&quot;Message after character stuffing is : %s&quot;, sdel);
}
```

return 0;

}

3. Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP.

//Program Code: // Program for Cyclic Redundency Check

//Program Code: // Program for Cyclic Redundency Check

#include<stdio.h>

#include<string.h>

#include<conio.h>

#define N strlen(g)

char t[128],cs[128],g[28];

int a, e,c;

void xor()

{

for(c=1;c<N;c++)

cs[c]=((cs[c]==g[c])?'0':'1');

}

void crc()

{

for(e=0;e<N;e++)

cs[e]=t[e];

do

{

if(cs[0]=='1')

xor();

```
for(c=0;c&lt;N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}while(e&lt;=a+N-1);
}
int main()
{
int b;
printf(&quot;\n1.crc12\n2.crc16\ncrc ccit\n4.exit\n\nEnter your option.&quot;);

scanf(&quot;%d&quot;,&amp;b);
switch(b)
{
case 1:strcpy(g,&quot;1100000001111&quot;);
break;
case 2:strcpy(g,&quot;11000000000000101&quot;);
break;
case 3:strcpy(g,&quot;10001000000100001&quot;);
break;
case 4:return 0;
}
printf(&quot;\n enter data:&quot;);
scanf(&quot;%s&quot;,t);
printf(&quot;\n----------\n&quot;);
printf(&quot;\n generating polynomial:%s&quot;,g);
a=strlen(t);
for(e=a;e&lt;a+N-1;e++)
t[e]=&#39;0&#39;;
printf(&quot;\n----------\n&quot;);
printf(&quot;mod-ified data is:%s&quot;,t);
printf(&quot;\n----------\n&quot;);
```

```
crc();
printf(&quot;checksum is:%s&quot;,cs);
for(e=a;e&lt;a+N-1;e++)
t[e]=cs[e-a];
printf(&quot;\n----------\n&quot;);
printf(&quot;\n final codeword is : %s&quot;,t);
printf(&quot;\n----------\n&quot;);
return 0 ;
}
```

4 Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

```
#include&lt;stdio.h&gt;
int main()
{
int w,i,f,frames[50];
printf(&quot;Enter window size: &quot;);
scanf(&quot;%d&quot;,&amp;w);
printf(&quot;\nEnter number of frames to transmit: &quot;);
scanf(&quot;%d&quot;,&amp;f);
printf(&quot;\nEnter %d frames: &quot;,f);
for(i=1;i&lt;=f;i++)
scanf(&quot;%d&quot;,&amp;frames[i]);
printf(&quot;\nWith sliding window protocol the frames will be sent in the
following manner (assuming no corruption of frames)\n\n&quot;);
printf(&quot;After sending %d frames at each stage sender waits for
acknowledgement sent by the receiver\n\n&quot;,w);
```

```
for(i=1;i&lt;=f;i++)
{
if(i%w==0)
{
printf(&quot;%d\n&quot;,frames[i]);
printf(&quot;Acknowledgement of above frames sent is received by
sender\n\n&quot;);
}
else

printf(&quot;%d &quot;,frames[i]);
}

if(f%w!=0)
printf(&quot;\nAcknowledgement of above frames sent is received by
sender\n&quot;);

return 0;
}
```

5. Implement Dijsktra's algorithm to compute the shortest path through a network

```
#include&lt;stdio.h&gt;
#include&lt;conio.h&gt;
```

```
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
int G[MAX][MAX],i,j,n,u;
printf(&amp;quot;Enter no. of vertices:&amp;quot;);
scanf(&amp;quot;%d&amp;quot;,&amp;amp;n);
printf(&amp;quot;\nEnter the adjacency matrix:\n&amp;quot;);
for(i=0;i&amp;lt;n;i++)
for(j=0;j&amp;lt;n;j++)
scanf(&amp;quot;%d&amp;quot;,&amp;amp;G[i][j]);
printf(&amp;quot;\nEnter the starting node:&amp;quot;);
scanf(&amp;quot;%d&amp;quot;,&amp;amp;u);
dijkstra(G,n,u);
return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{

int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;
//pred[] stores the predecessor of each node
//count gives the number of nodes seen so far
//create the cost matrix
for(i=0;i&amp;lt;n;i++)
for(j=0;j&amp;lt;n;j++)
```

```
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
for(i=0;i&lt;n;i++)
{
distance[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count&lt;n-1)
{
mindistance=INFINITY;
//nextnode gives the node at minimum distance
for(i=0;i&lt;n;i++)
if(distance[i]&lt;mindistance&amp;&amp;!visited[i])
{
mindistance=distance[i];
nextnode=i;
}
//check if a better path exists through nextnode

visited[nextnode]=1;
for(i=0;i&lt;n;i++)
if(!visited[i])
if(mindistance+cost[nextnode][i]&lt;distance[i])
{
```

```
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}

//print the path and distance of each node
for(i=0;i&lt;n;i++)
if(i!=startnode)
{
printf(&quot;\nDistance of node%d=%d&quot;,i,distance[i]);
printf(&quot;\nPath=%d&quot;,i);
j=i;
do
{
j=pred[j];
printf(&quot;&lt;-%d&quot;,j);
}while(j!=startnode);
}
}
```

6 Implement data encryption and data decryption

```
#include &lt;stdio.h&gt;
int main()
{
```

```
  int i, x;
  char str[100];
  printf(&amp;quot;\nPlease enter a string:\t&amp;quot;);
  gets(str);
  printf(&amp;quot;\nPlease choose following options:\n&amp;quot;);
  printf(&amp;quot;1 = Encrypt the string.\n&amp;quot;);
  printf(&amp;quot;2 = Decrypt the string.\n&amp;quot;);
  scanf(&amp;quot;%d&amp;quot;, &amp;amp;x);
  //using switch case statements
  switch(x)
  {
  case 1:
    for(i = 0; (i &amp;lt; 100 &amp;amp;&amp;amp; str[i] != &amp;#39;\0&amp;#39;); i++)
      str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value
    printf(&amp;quot;\nEncrypted string: %s\n&amp;quot;, str);
    break;
  case 2:
    for(i = 0; (i &amp;lt; 100 &amp;amp;&amp;amp; str[i] != &amp;#39;\0&amp;#39;); i++)
      str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value
    printf(&amp;quot;\nDecrypted string: %s\n&amp;quot;, str);
    break;
  default:
    printf(&amp;quot;\nError\n&amp;quot;);
  }
  return 0;
}
```

7.simulate the network with nodes n0,n1,n2,n3,n4, forming a star topology. The node n4 is at the center. 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.

```
#Create a simulator object
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the NAM trace file
set nf [open 7.nam w]
$ns namtrace-all $nf
set tf [open 7.tr w]
$ns trace-all $tf
#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n1 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n3 1Mb 10ms DropTail
$ns duplex-link $n4 $n2 1Mb 10ms DropTail
#Setup a TCP connection
```

```
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
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 $n2 $null
$ns connect $udp $null
#Setup a CBR over UDP

set cbr [new
Application/Traffic/CBR]

$cbr attach-agent $udp
$cbr set packet_size_ 500
#Schedule events for the CBR and FTP
agents $ns at 0.0 &quot;$cbr start&quot;
$ns at 0.0 &quot;$ftp start&quot;
$ns at 20.0 &quot;$ftp stop&quot;
$ns at 20.0 &quot;$cbr stop&quot;
#Define a &#39;finish&#39; procedure

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

```
close $nf

close $tf
exec nam 7.nam &amp;amp;
exit 0
}
$ns at 10.0 &amp;quot;finish&amp;quot;
#Run the simulation
$ns run

7.awk:-
BEGIN{
udp=0;
tcp=0;
}
{
if($1 == &amp;quot;r&amp;quot; &amp;amp;&amp;amp; $5 == &amp;quot;cbr&amp;quot;)
{
udp++;
}
else if($1 == &amp;quot;r&amp;quot; &amp;amp;&amp;amp; $5 == &amp;quot;tcp&amp;quot;)
{

tcp++;
}
}
END{
printf(&amp;quot;Number of packets sent by TCP = %d\n&amp;quot;, tcp);
printf(&amp;quot;Number of packets sent by UDP=%d\n&amp;quot;,udp);
```

8. simulate to study transmission of packets over ethernet LAN and determine the number of packets drop destination.

```
set ns [new Simulator]
set nf [open 8b.nam w]
$ns namtrace-all $nf
set tf [open 8b.tr w]
$ns trace-all $tf
set n0 [$ns node]
$n0 color &amp;quot;red&amp;quot;
$n0 label &amp;quot;src1&amp;quot;
set n1 [$ns node]
set n2 [$ns node]
$n2 color &amp;quot;red&amp;quot;
$n2 label &amp;quot;src2&amp;quot;
set n3 [$ns node]
$n3 color &amp;quot;blue&amp;quot;
$n3 label &amp;quot;dest2&amp;quot;
set n4 [$ns node]
set n5 [$ns node]
$n5 color &amp;quot;blue&amp;quot;
$n5 label &amp;quot;dest1&amp;quot;
$ns make-lan &amp;quot;$n0 $n1 $n2 $n3 $n4&amp;quot; 1Mb 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Kb 1ms DropTail
$ns queue-limit $n4 $n5 1
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

proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam 8b.nam &amp;
exit 0
}
$ns at 0.1 &quot;$ftp0 start&quot;
$ns at 5 &quot;$ftp0 stop&quot;
$ns at 7 &quot;$ftp0 start&quot;
$ns at 0.2 &quot;$ftp2 start&quot;
$ns at 8 &quot;$ftp2 stop&quot;
```

```
$ns at 14 &quot;$ftp0 stop&quot;
$ns at 10 &quot;$ftp2 start&quot;
$ns at 15 &quot;$ftp2 stop&quot;
$ns at 16 &quot;finish&quot;
$ns run
```

8b.awk

```
#awk file run in new terminal
BEGIN{
 c=0;
}
{
if($1==&quot;d&quot;)
{
c++;
}
}
END{
printf(&quot;The number of %s packets dropped =%d\n&quot;,$5,c);  }
```

output:-

the number of ack packets dropped=3

9.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.

```
set ns [new Simulator]
set nf [open 9.nam
w]
$ns namtrace-all
$nf
set tf [open 9.tr w]
$ns trace-all
$tf
proc finish {} {
global ns nf tf
$ns flush-
trace
close $nf
close $tf
exec nam 9.nam
&amp;
exit 0
}
set n0 [$ns
node]
$n0 label &quot;ftp/src1&quot;
set n1 [$ns node]
$n1 label
&quot;telnet/src2&quot;
set n2 [$ns node]
set n3 [$ns node]

$n3 label
```

```
&quot;tcpsink/dest1&quot;
set n4 [$ns node]
$n4 label &quot;tcpsink/dest2&quot;
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
$ns duplex-link $n4 $n2 1Mb 10ms
DropTail
$ns queue-limit $n1 $n2 1
$ns queue-limit $n0 $n2
3
set tcp [new
Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new
Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set tcp0 [new
Agent/TCP]
$ns attach-agent $n1 $tcp0
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n3 $tcpsink0
$ns connect $tcp0 $tcpsink0

$tcp0 set fid_ 2
set telnet0 [new Application/Telnet]
$telnet0 attach-agent $tcp0
```

$ns at 0.5 "$ftp start"

$ns at 4.5 "$ftp stop"

$ns at 0.5 "$telnet0 start"

$ns at 4.5 "$telnet0 stop"

$ns at 5.0 "finish"

$ns run

9.awk

BEGIN{

ftpsend=0;

telnetsend=0

; ftpDrop=0;

telnetDrop=0

;

telnetrecv=0;

ftprecv=0;

}

{

event=$1; src=$9;

dest=$10;

pktype=$5;

if((event=="+") && (src=="0.0") &&
(dest=="4.0") &&pktype="tcp")

{

ftpsend++;

}

if((event=="r") && (src=="0.0") &&
(dest=="4.0") &&pktype="tcp")

{

ftprecv++;

}

```
if((event==&quot;+&quot;) &amp;&amp; (src==&quot;1.0&quot;) &amp;&amp;
(dest==&quot;3.0&quot;) &amp;&amp;pktype=&quot;tcp&quot;)

{

telnetsend++;

}

if((event==&quot;r&quot;) &amp;&amp; (src==&quot;1.0&quot;) &amp;&amp;
(dest==&quot;3.0&quot;) &amp;&amp;pktype=&quot;tcp&quot;)

{

telnetrecv++;

}


if((event==&quot;d&quot;) (src==&quot;0.0&quot;) &amp;&amp; (dest==&quot;4.0&quot;)
&amp;&amp;pktype=&quot;tcp&quot;)

{

ftpDrop++;

}


if((event==&quot;d&quot;) &amp;&amp; (src==&quot;1.0&quot;) &amp;&amp;
(dest==&quot;3.0&quot;) &amp;&amp;pktype=&quot;tcp&quot;)

{

telnetDrop++;

}

} END{

print &quot;\n Number of Packets Sent by ftp : &quot; ftpsend

print &quot;\n Number of Packets Sent by tenet : &quot; telnetsend print

&quot;\n Number of Packets Received by ftp: &quot; ftprecv

print &quot;\n Number of Packets Received by telnet: &quot; telnetrecv print

&quot;\n Packet Delivery Ratio by FTP : &quot; (ftprecv/ftpsend)*100

print &quot;\n Packet Delivery Ratio by telnet : &quot; (telnetrecv/telnetsend)*100 print

&quot;\n Packet drop by ftp:&quot; ftpDrop

print &quot;\n Packet drop by telnet:&quot; telnetDrop

}
```