p1.tcl

A1. Implement three nodes point–to–point networks with duplex links between them using NS2. Set the queue size, vary the bandwidth, and find the number of packets dropped.

```
set ns [new Simulator]
set nf [open p1.nam w]
$ns namtrace-all $nf
set tf [open p1.tr w]
$ns trace-all $tf

proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam p1.nam &
exit 0
}

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

$ns duplex-link $n0 $n2 4Mb 2ms DropTail
$ns duplex-link $n2 $n3 100kb 10ms DropTail
$ns queue-limit $n0 $n2 2

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packageSize_ 1000
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

$ns at 0.1 "$cbr0 start"
$ns at 1.0 "finish"
```

```
$ns run
```

p1.awk

```
BEGIN{c=0;

       r=0;
}

{
        if($1=="d")
        {
               c++;
         }
         else if($1=="r")
         {
           r++;
     }
}
END{ printf("The number of packets dropped=%d\n",c);
   printf("The number of packets recieved =%d\n",r); }
```

p2.tcl

Implement transmission of ping messages/traceroute over a network topology consisting of 6 nodes using NS2 and find the number of packets dropped due to congestion.

```
set ns [new Simulator]
set nf [open lab2.nam w]
$ns namtrace-all $nf
set tf [open lab2.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 0.1Mb 1ms DropTail
$ns duplex-link $n1 $n4 0.1Mb 1ms DropTail
```

```
$ns duplex-link $n2 $n4 0.1Mb 1ms DropTail
$ns duplex-link $n3 $n4 0.1Mb 1ms DropTail
$ns duplex-link $n4 $n5 0.1Mb 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
$p3 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 $p2 $p4
$ns connect $p3 $p5

proc finish {} {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab2.nam &
exit 0
}

$ns at 0.1 "$p2 send"
$ns at 0.2 "$p2 send"
$ns at 0.3 "$p2 send"
$ns at 0.4 "$p2 send"
$ns at 0.5 "$p2 send"
```

```
$ns at 0.6 "$p2 send"
$ns at 0.7 "$p2 send"
$ns at 0.8 "$p2 send"
$ns at 0.9 "$p2 send"
$ns at 1.0 "$p2 send"
$ns at 1.1 "$p2 send"
$ns at 1.2 "$p2 send"
$ns at 1.3 "$p2 send"
$ns at 1.4 "$p2 send"
$ns at 1.5 "$p2 send"
$ns at 1.6 "$p2 send"
$ns at 1.7 "$p2 send"
$ns at 1.8 "$p2 send"
$ns at 1.9 "$p2 send"
$ns at 2.0 "$p2 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 3.0 "finish"

$ns run
```

p2.awk

```
BEGIN{
      count=0;
}
{
      if($1=="d")
      {
            count++;
      }
}
END{ printf("The number of packets dropped =%d\n",count);
}
```

p4.tcl

Implement simple ESS and with transmitting nodes in wire-less LAN by simulation using NS2 and determine the performance with respect to the transmission of packets.

```
set ns [new Simulator]

set tf [open Program4.tr w]
$ns trace-all $tf

set topo [new Topography]
$topo load_flatgrid 1000 1000

set nf [open Program4.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 Program4.nam &
close $tf
exit 0
}

$ns at 250 "finish"
$ns  run
```

p4.awk

```
BEGIN{
pack1=0
pack2=0
time1=0
time2=0
}{
if($1=="r" && $3=="_1_" && $4=="AGT")
{
pack1=pack1+$8
time1=$2 }
if($1=="r" && $3=="_2_" && $4=="AGT")
{
pack2=pack2+$8
time2=$2 }
}
END{
printf("The Throught from n1 to n2: %f Mbps\n",
((pack1*8)/(time1*1000000)));
printf("The Throught from n1 to n2: %f Mbps\n",
((pack2*8)/(time2*1000000)));
}
```

p5.java

Write a program for error detecting code using CRC-CCITT (16- bits).

```
import java.util.Scanner;

public class CRCb {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter message bits:");
        String message = sc.nextLine();
        System.out.println("Enter generator (16 bits):");
        String generator = sc.nextLine();
```

```java
        // Convert message and generator strings to arrays of integers
        int[] data = new int[message.length() + generator.length() - 1];
        int[] divisor = new int[generator.length()];
        for (int i = 0; i < message.length(); i++)
            data[i] = Integer.parseInt(message.charAt(i) + "");
        for (int i = 0; i < generator.length(); i++)
            divisor[i] = Integer.parseInt(generator.charAt(i) + "");

        // Perform CRC division
        for (int i = 0; i < message.length(); i++) {
            if (data[i] == 1) {
                for (int j = 0; j < divisor.length; j++)
                    data[i + j] ^= divisor[j];
            }
        }

        // Generate checksum code
        System.out.println("The checksum code is:");
        for (int i = 0; i < data.length; i++)
            System.out.print(data[i]);
        System.out.println();

        // Check validity of data stream
        System.out.println("Enter received data bits:");
        String receivedData = sc.nextLine();
        data = new int[receivedData.length() + generator.length() - 1];
        for (int i = 0; i < receivedData.length(); i++)
            data[i] = Integer.parseInt(receivedData.charAt(i) + "");

        // Perform CRC division on received data
        for (int i = 0; i < receivedData.length(); i++) {
            if (data[i] == 1) {
                for (int j = 0; j < divisor.length; j++)
                    data[i + j] ^= divisor[j];
            }
        }
        boolean valid = true;
        for (int i = 0; i < data.length; i++) {
            if (data[i] == 1) {
                valid = false;
                break;
            }
        }
        if (valid)
            System.out.println("Data stream is valid.");
        else
            System.out.println("Data stream is invalid. CRC error has occurred.");
```

```
    }
}
```

p6.java

Write a program to find the shortest path between vertices using the bellman-ford algorithm

```
import java.util.Scanner;

public class BellmanFord
{
      private int D[];
      private int num_ver;
      public static final int MAX_VALUE = 999;
      public BellmanFord(int num_ver)
      {
            this.num_ver = num_ver;
            D = new int[num_ver + 1];
      }

      public void BellmanFordEvaluation(int source, int A[][])
      {
            for (int node=1;node <= num_ver; node++)
            {
                  D[node] = MAX_VALUE;
            }
            D[source] = 0;
            for (int node=1; node<=num_ver - 1; node++)
            {
                  for (int sn=1;sn<=num_ver;sn++)
                  {
                        for (int dn=1; dn <= num_ver;dn++)
                              {
                                    if(A[sn][dn] !=MAX_VALUE)
                                    {
                                                if (D[dn] > D[sn] + A[sn]
[dn])
                                                      D[dn] = D[sn] + A[sn]
[dn];
                                    }
                              }

                  }
            }
            for (int sn=1;sn<=num_ver; sn++)
            {
```

```
            for (int dn=1;dn<=num_ver;dn++)
                {
                if (A[sn][dn] != MAX_VALUE)
                {
                    if (D[dn] > D[sn] + A[sn][dn])
                    {
                        System.out.println("The graph contains
negative edge cycle");
                        return;
                    }
                }
            }
        }

        for (int vertex = 1; vertex<=num_ver;vertex++)
        {
            System.out.println("distance of source" + source + "
to " + vertex + " is " + D[vertex]);
        }
    }

    public static void main(String[] args)
    {
        int num_ver=0;
        int source;
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of vertices");
        num_ver=scanner.nextInt();
        int A[][]=new int[num_ver + 1 ][num_ver + 1];
        System.out.println("Enter the adjacency matrix");
        for (int sn=1;sn<=num_ver;sn++)
        {
            for (int dn=1;dn<=num_ver;dn++)
            {
                A[sn][dn] = scanner.nextInt();
                if(sn == dn)
                {
                    A[sn][dn] = 0;
                    continue;
                }
                if (A[sn][dn] == 0)
                {
                    A[sn][dn] = MAX_VALUE;
                }
            }
        }
        System.out.println("Enter the source vertex");
        source = scanner.nextInt();
        BellmanFord b= new BellmanFord(num_ver);
```

```
           b.BellmanFordEvaluation(source,A);
           scanner.close();
     }
}
```

p7.java

Write a program for congestion control using a leaky bucket algorithm.

```
import java.util.Scanner;
public class lab7
{
  public static void main(String[] args)
  {
    int i;
    int a[]=new int[20];
    int buck_rem=0,buck_cap=4,rate=3,sent,recv;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of packets ");
    int n = in.nextInt();
    System.out.println("Enter the packets ");
    for(i=1;i<=n;i++) {
      a[i]=in.nextInt();
    }
    System.out.println("Clock \t Packet size \t Accept \t Sent \t
Remaining");
    for(i=1;i<=n;i++) {
      if(a[i]!=0) {
        if(buck_rem + a[i] > buck_cap) {
          recv = -1;
        }
        else {
          recv=a[i];
          buck_rem=+a[i];
        }
      }
      else {
        recv=0;
      }
      if(buck_rem!=0)
      {
        if(buck_rem<rate)
        {
          sent=buck_rem;
          buck_rem=0;
        }
        else{
          sent=rate;
          buck_rem=buck_rem-rate;
        }}
        else
          sent=0;
        if(recv==-1)
          System.out.println(+i+"\t\t"+a[i]+"\t\tdropped\t"+sent+"\
t"+buck_rem);
        else
          System.out.println(+i+"\t\t"+a[i]+"\t\t"+recv+"\t"+sent+"\
t"+buck_rem);
        }
      }
    }
```

p8a FileClient.java

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

```
import java.io.*;
import java.net.*;
public class FileClient
{
    public static void main(String[] args)
    {
        new FileClient();
    }
    public FileClient(){
        BufferedReader bufReader=new BufferedReader(new
InputStreamReader(System.in));
        try{
            Socket clientsocket=new Socket("localhost",8000);
            System.out.println("Connecting to server...");
            DataInputStream input=new
DataInputStream(clientsocket.getInputStream());
            DataOutputStream output = new
DataOutputStream(clientsocket.getOutputStream());
            System.out.println("Enter file name:");
            String Name = bufReader.readLine();
            output.writeUTF(Name);
            String EcFile = input.readUTF();
            System.out.println(EcFile);
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
    }
}
```

p8b FileServer.java

```
import java.io.*;
import java.net.*;
public class FileServer
{
    public static void main(String[] args)
    {
        new FileServer();
    }
    public FileServer()
    {
        DataOutputStream output;
        DataInputStream input;
        Socket socket;
        ServerSocket serversocket;
        BufferedReader br;
        String everything;
        try
        {
            serversocket=new ServerSocket(8000);
            System.out.println("Server Started.......");
            socket=serversocket.accept();
            input=new DataInputStream(socket.getInputStream());
            output=new DataOutputStream(socket.getOutputStream());
            while(true)
            {
                String str =input.readUTF();
                System.out.println(str);
                try{
                    URL url = getClass().getResource(str);
                    InputStream istream= url.openStream();

                    br=new BufferedReader(new
InputStreamReader(istream));
                    StringBuilder sb=new StringBuilder();
                    String line = br.readLine();
                    while(line!=null)
                    {
                        sb.append(line);
                        line=br.readLine();
                    }
                    everything=sb.toString();
                }
                catch(Exception ex)
                {
                    everything = "File Not Found";
                }
                output.writeUTF(everything);
            }
        }
        catch(Exception ex)
        {
            everything="Error";
        }
        finally{
        }
    }
}
```

part c1.tcl

Implement an Ethernet LAN using n nodes and set multiple traffic nodes using NS2 and plot conges the window for different source/destination.

```
set ns [new Simulator]
set tf [open lab3.tr w]
$ns trace-all $tf

set nf [open lab3.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"

set n1 [$ns node]
$n1 color "magenta"
$n1 label "src2"

set n2 [$ns node]

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 $n4" 100Mb 10ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n2 $n3 1Mb 1ms DropTail
$ns queue-limit $n2 $n3 5
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
$ns queue-limit $n4 $n5 3

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 $n1 $tcp2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.000

set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp2 $sink3

set file1 [open file1.tr w]
$tcp0 attach $file1
$tcp0 trace cwnd_

set file2 [open file2.tr w]
$tcp2 attach $file2
$tcp2 trace cwnd_

proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
exec nam lab3.nam &
exit 0
}

$ns at 0.1 "$ftp0 start"
$ns at 14 "$ftp0 stop"
$ns at 0.2 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run
```

part c1.awk

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

part c2.py

Implement IPv4 address classifier (A, B, C, D, and E) using any programming language.

```
def classify_and_identify_ipv4(ip_address : str):
    octets = ip_address.split('.')

    if len(octets) != 4:
        return "Invalid Ipv4 address"

    first_octet = int(octets[0])

    if 1 <= first_octet <= 127:
        classification = "Class A"
        network_id = octets[0]
        host_id = '.'.join(octets[1:])
    elif 128 <= first_octet <= 191:
        classification = "Class B"
        network_id = '.'.join(octets[:2])
        host_id = '.'.join(octets[2:])
    elif 192 <= first_octet <= 223:
        classification = "Class C"
        network_id = '.'.join(octets[:3])
        host_id = '.'.join(octets[3:])
    elif 224 <= first_octet <= 239:
        classification = "Class D"
        network_id = "N/A"
        host_id = "N/A"
    elif 240 <= first_octet <= 255:
        classification = "Class E"
        network_id = "N/A"
        host_id = "N/A"
    else:
        return "Invalid IPv4 address"

    return f"Classification: {classification} \n Network ID:
{network_id}\n Host ID: {host_id}"


user_input = input("Enter an IPv4 address: ")
results = classify_and_identify_ipv4(user_input)
print(results)
```

Open Ended

Create IPv4 or IPv6 packets using any programming language.

```
import socket

# Take input for IPv4 header fields
ttl = int(input("Enter TTL (Time To Live): "))
source_ip = input("Enter Source IP address: ")
destination_ip = input("Enter Destination IP address: ")

# TCP header fields
source_port = int(input("Enter Source Port: "))
destination_port = int(input("Enter Destination Port: "))

# IPv4 header fields
version = 4
header_length = 5
protocol = 6  # TCP protocol

# Constructing the IPv4 packet
ipv4_header = bytearray()
ipv4_header += ((version << 4) + header_length).to_bytes(1, 'big')
ipv4_header += ttl.to_bytes(1, 'big')
ipv4_header += protocol.to_bytes(1, 'big')
ipv4_header += socket.inet_aton(source_ip)
ipv4_header += socket.inet_aton(destination_ip)
```

```python
# TCP header
tcp_header = bytearray()
tcp_header += source_port.to_bytes(2, 'big')
tcp_header += destination_port.to_bytes(2, 'big')
tcp_header += b'\x00\x00\x00\x00'  # Sequence number (4 bytes)
tcp_header += b'\x00\x00\x00\x00'  # Acknowledgment number (4 bytes)
tcp_header += b'\x50\x02'  # Data offset, Reserved, and Flags (2
bytes)
tcp_header += b'\xff\xff'  # Window size (2 bytes)
tcp_header += b'\x00\x00'  # Checksum (2 bytes)
tcp_header += b'\x00\x00'  # Urgent pointer (2 bytes)

# Displaying the constructed IPv4 packet
print("IPv4 Packet:")
print("Version:", version)
print("Header Length:", header_length)
print("TTL:", ttl)
print("Protocol:", protocol)
print("Source IP:", source_ip)
print("Destination IP:", destination_ip)
print("Raw Bytes (IPv4 header):", ipv4_header.hex())
print("Raw Bytes (TCP header):", tcp_header.hex())
```