RNS INSTITUTE OF TECHNOLOGY

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)

UG Programs - CSE, ECE, ISE, EIE and EEE have been Accredited by NBA for three Academic years

DR. VISHNUVARDHAN ROAD, CHANNASANDRA, RR NAGAR POST, BENGALURU – 560 0

ESTD : 2001

An Institute with a Difference

Department of Computer Science and Engineering

COMPUTER NETWORKS LABORATORY MANUAL

For Fifth Semester B.E
[VTU/CBCS, 2018-19 syllabus]

Subject Code – 18CSL57

NAME   : _____

BRANCH : _____

SECTION : _____

USN   : _____

VISION AND MISSION OF INSTITUTION
Vision
Building RNSIT into a World Class Institution

Mission
To impart high quality education in Engineering, Technology and Management with a Difference, Enabling Students to Excel in their Career by

1. Attracting quality Students and preparing them with a strong foundation in fundamentals so as to achieve distinctions in various walks of life leading to outstanding contributions.
2. Imparting value based, need based, choice based and skill based professional education to the aspiring youth and carving them into disciplined, World class Professionals with social responsibility.
3. Promoting excellence in Teaching, Research and Consultancy that galvanizes academic consciousness among Faculty and Students.
4. Exposing Students to emerging frontiers of knowledge in various domains and make them suitable for Industry, Entrepreneurship, Higher studies, and Research & Development.
5. Providing freedom of action and choice for all the Stake holders with better visibility.

VISION AND MISSION OF CSE DEPARTMENT
Vision

Preparing better computer professionals for a real world

Mission
The Department of Computer Science and Engineering will make every effort to promote an intellectual and an ethical environment in which the strengths and skills of Computer Professionals will flourish by

1. Imparting Solid foundations and Applied aspects in both Computer Science Theory and Programming practices.
2. Providing Training and encouraging R&D and Consultancy Services in frontier areas of Computer Science with a Global outlook.
3. Fostering the highest ideals of Ethics, Values and creating Awareness on the role of Computing in Global Environment.
4. Educating and preparing the graduates, highly Sought-after, Productive, and Well-respected for their work culture.
5. Supporting and inducing Lifelong Learning practice

ACKNOWLEDGMENT

TABLE OF CONTENTS

COMPUTER NETWORKS LABORATORY [18CSL57]

Subject Code: 18CSL57   IA Marks: 40
Hours/Week: 01I + 02P   Exam Marks: 60
Total Hours: 40   Exam Hours: 03

Course objectives: This course will enable students to achieve the following:
1. Demonstrate operation of network and its management commands
2. Simulate and demonstrate the performance of GSM and CDMA
3. Implement data link layer and transport layer protocols.

Course outcomes: At the end of the course the students should be able to:

CO 1: Understand error detection technique using CRC
CO 2: Analyze and compare different routing protocols.
CO 3: Implement connection-oriented and connectionless protocols in the network
CO 4: Demonstrate security features in networks using RSA algorithm
CO 5: Analyze techniques to avoid congestion in the network
CO 6: Implement, analyze and evaluate networking protocols using NS-3 Tool

PART A

Implement the following using NS2/NS3

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.
2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source /destination.
4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

PART B

Implement the following in Java:

7. Write a program for error detecting code using CRC-CCITT (16-bits).
8. Write a program to find the shortest path between vertices using bellman-ford algorithm.

9. 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 ifpresent.

10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.
11. Write a program for simple RSA algorithm to encrypt and decrypt the data.
12. Write a program for congestion control using leaky bucket algorithm.

PART – A

INTRODUCTION TO NS3

NS3 Simulator Basics

- NS-3 is a network simulator
- Developed for network research and education
- Developed after ns-2
- ns-3 is written in C++
- Bindings in Python
- ns-3 uses the waf build system

waf is a build automation tool designed to assist in the automatic compilation and installation of computer software. It is written in Python.

waf features:

- Portable to Unix and non-Unix systems
- Lightweight
- Offers a Turing-complete programming language (similar to SCons)
- Support for standard targets: configure, build, clean, distclean, install, and uninstall
- Parallel builds
- Colored output and progress bar display
- Scripts are Python modules
- XML script front-end and a dedicated, easy-to-parse "IDE output" mode to ease the interaction with integrated development environments
- Modular configuration scheme with customizable command-line parsing
- Daemon mode for background recompilation
- Find source files intelligently (glob()-like) to ease script maintenance
- Support for global object cache to avoid unnecessary recompilations
- Support for unit tests run on programs at the end of builds
  Waf supports:
- A C/C++ preprocessor for computing dependencies simulation programs are C++ executables or python scripts

Features

- It it a discrete event simulator
- Modular design / Open source
- Actively developed (Contrast NS-2)
- Developed in C++. Python binding available.
- Live visualizer
- Logging facility for debugging
- Tracing facility for getting output
- Can be connected to a real network
- Direct Code Execution (DCE)

How to install ns3?

Download tarball from www.nsnam.org the recent release in your directory. Go to that directory and untar the tarball.

    tar xvfj ns3.tar

It creates the directory for ns3 change to it.

    cd ns3

For compiling and installing

   ./build.py --enable-examples --enable-tests

How to run ns3 script?

To test the installation copy one example available in the distribution to scratch directory and build and run the same using the commands below:

   cd ns3.26
   cp examples/tutorial/first.cc scratch/first.cc
   ./waf --run scratch/first

Steps in writing scripts

1. Include necessary files
2. Use appropriate name space
3. Set simulation time resolution(Optional)
4. Enable logging for different modules(Optional)
5. Create nodes
6. Create net devices with MAC and PHY
7. Attach Net devices to nodes and set interconnections
8. Install protocol stack in nodes
9. Set network address for interfaces
10. Setup routing
11. Install applications in nodes
12. Setup tracing(Optional)
13. Set application start and stop time
14. Set simulation start time(Optional)
15. Run simulation
16. Release resources at end of simulation

Tutorial : First.cc

   Simple point to point (Wired network) link between server and client is established here. This program is in your NS3 repository. (example/tutorial/first.cc)

Note : To know about NS3, you must have the base knowledge in C++ and OOPS concept.

1. Module Includes

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

Each of the ns-3 include files is placed in a directory called ns3 (under the build directory) during the build process to help avoid include file name collisions. The ns3/core-module.h file corresponds to the ns-3 module you will find in the directory src/core in your downloaded release distribution. If you list this directory you will find a large number of header files. When you do a build, Waf will place public header files in an ns3 directory under the appropriate build/debug or build/optimized directory depending on your configuration. Waf will also automatically generate a module include file to load all of the public header files.

2. NS3 Namespace

using namespace ns3;

The ns-3 project is implemented in a C++ namespace called ns3. This groups all ns-3-related declarations in a scope outside the global namespace, which we hope will help with integration with other code. The C++ using statement introduces the ns-3 namespace into the current (global) declarative region. This is a fancy way of saying that after this declaration, you will not have to type ns3:: scope resolution operator before all of the ns-3 code in order to use it. If you are unfamiliar with namespaces, please consult almost any C++ tutorial and compare the ns3 namespace and usage here with instances of the std namespace and the using namespace std; statements you will often find in discussions of cout and streams.

3. Set simulation time resolution

*int main (int argc, char *argv[])*
This is just the declaration of the main function of your program (script). Just as in any C++ program, you need to define a main function that will be the first function run. There is nothing at all special here. Your ns3 script is just a C++ program.
The next line sets the time resolution to one nanosecond, which happens to be the default value:
Time::SetResolution (Time::NS);
The resolution is the smallest time value that can be represented (as well as the smallest represent able difference between two time values). You can change the resolution exactly once. The mechanism enabling this flexibility is somewhat memory hungry, so once the resolution has been set explicitly we release the memory, preventing further updates. (If you don't set the resolution explicitly, it will default to one nanosecond, and the memory will be released when the simulation starts.)

4. Enable logging for different modules

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
this line declares a logging component called FirstScriptExample that allows you to enable and disable console message logging by reference to the name.

5. Create nodes

NodeContainer nodes;
nodes.Create (2);
The NodeContainer topology helper provides a convenient way to create, manage and access any Node objects that we create in order to run a simulation. The first line above just declares a NodeContainer which we call nodes. The second line calls the Create method on the nodes object and asks the container to create two nodes.

6. Create net devices with MAC and PHY

PointToPointHelper pointToPoint;

It instantiates a PointToPointHelper object on the stack. From a high-level perspective the next line,

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

Above line tells the PointToPointHelper object to use the value "5Mbps" (five megabits per second) as the "DataRate" when it creates a PointToPointNetDevice object. From a more detailed perspective, the string "DataRate" corresponds to what we call an Attribute of the PointToPointNetDevice.

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

It tells the PointToPointHelper to use the value "2ms" (two milliseconds) as the value of the transmission delay of every point to point channel it subsequently creates.

7. Attach Net devices to nodes and set interconnections

NetDeviceContainer devices;

devices = pointToPoint.Install (nodes);

The first line declares the device container mentioned above and the second does the heavy lifting. The Install method of the PointToPointHelper takes a NodeContainer as a parameter. Internally, a NetDeviceContainer is created. For each node in the NodeContainer (there must be exactly two for a point-to-point link) a PointToPointNetDevice is created and saved in the device container.

  A PointToPointChannel is created and the two PointToPointNetDevices are attached. When objects are created by the PointToPointHelper, the Attributes previously set in the helper are used to initialize the corresponding Attributes in the created objects. After executing the pointToPoint.Install (nodes) call we will have two nodes, each with an installed point-to-point net device and a single point-to-point channel between them. Both devices will be configured to transmit data at five megabits per second over the channel which has a two millisecond transmission delay.

8. Install protocol stack in nodes

The only user-visible API is to set the base IP address and network mask to use when performing the actual address allocation (which is done at a lower level inside the helper).

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

It declares an address helper object and tell it that it should begin allocating IP addresses from the network 10.1.1.0 using the mask 255.255.255.0 to define the allocatable bits. By default the addresses allocated will start at one and increase monotonically, so the first address allocated from this base will be 10.1.1.1, followed by 10.1.1.2, etc. The low level ns3 system actually remembers all of the IP addresses allocated and will generate a fatal error if you accidentally cause the same address to be generated twice (which is a very hard to debug error, by the way).

9. Set network address for interfaces

Ipv4InterfaceContainer interfaces = address.Assign (devices);

It performs the actual address assignment. In ns-3 we make the association between an IP address and a device using an Ipv4Interface object. Just as we sometimes need a list of net devices created by a helper for future reference we sometimes need a list of Ipv4Interface objects. The Ipv4InterfaceContainer provides this functionality. Now we have a point-to-point network built, with stacks installed and IP addresses assigned. What we need at this point are applications to generate traffic.

10. Setup routing

UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));

The first line of code in the above snippet declares the UdpEchoServerHelper. As usual, this isn't the application itself, it is an object used to help us create the actual applications. One of our conventions is to place required Attributes in the helper constructor. In this case, the helper can't do anything useful unless it is provided with a port number that the client also knows about. Rather than just picking one and hoping it all works out, we require the port number as a parameter to the constructor. The constructor, in turn, simply does a SetAttribute with the passed value. If you want, you can set the "Port" Attribute to another value later using SetAttribute.
  Similar to many other helper objects, the UdpEchoServerHelper object has an Install method. It is the execution of this method that actually causes the underlying echo server application to be instantiated and attached to a node. Interestingly, the Install method takes a NodeContainter as a parameter just as the other Install methods we have seen. This is actually what is passed to the method even though it doesn't look so in this case. There is a C++ implicit conversion at work here that takes the result of nodes.Get (1) (which returns a smart pointer to a node object — Ptr<Node>) and uses that in a constructor for an unnamed NodeContainer that is then passed to Install. If you are ever at a loss to find a particular method signature in C++ code that compiles and runs just fine, look for these kinds of implicit conversions.

serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

Above lines cause the echo server application to Start (enable itself) at one second into the simulation and to Stop (disable itself) at ten seconds into the simulation. By virtue of the fact that we have declared a simulation event (the application stop event) to be executed at ten seconds, the simulation will last at least ten seconds.

11. Install applications in nodes

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
For the echo client, however, we need to set five different Attributes. The first two Attributes are set during construction of the UdpEchoClientHelper. We pass parameters that are used (internally to the helper) to set the "RemoteAddress" and "RemotePort" Attributes in accordance with our convention to make required Attributes parameters in the helper constructors.

The zeroth interface in the interfaces container is going to correspond to the IP address of the zeroth node in the nodes container. The first interface in the interfaces container corresponds to the IP address of the first node in the nodes container. So, in the first line of code (from above), we are creating the helper and telling it so set the remote address of the client to be the IP address assigned to the node on which the server resides. We also tell it to arrange to send packets to port nine.

The "MaxPackets" Attribute tells the client the maximum number of packets we allow it to send during the simulation.

The "Interval" Attribute tells the client how long to wait between packets, and the "PacketSize" Attribute tells the client how large its packet payloads should be. With this particular combination of Attributes, we are telling the client to send one 1024-byte packet.

12. Set application start and stop time

serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

First it will run the event at 1.0 seconds, which will enable the echo server application (this event may, in turn, schedule many other events). Then it will run the event scheduled for t=2.0 seconds which will start the echo client application. Again, this event may schedule many more events. The start event implementation in the echo client application will begin the data transfer phase of the simulation by sending a packet to the server.

13. Run simulation

Simulator::Run ();

When Simulator::Run is called, the system will begin looking through the list of scheduled events and executing them. Eventually, since we only send one packet (recall the MaxPackets Attribute was set to one), the chain of events triggered by that single client echo request will taper off and the simulation will go idle. Once this happens, the remaining events will be the

Stop events for the server and the client. When these events are executed, there are no further events to process and Simulator::Run returns. The simulation is then complete.

14. Release resources at end of simulation

All that remains is to clean up. This is done by calling the global function Simulator::Destroy. As the helper functions (or low level ns-3 code) executed, they arranged it so that hooks were inserted in the simulator to destroy all of the objects that were created. You did not have to keep track of any of these objects yourself — all you had to do was to call Simulator::Destroy and exit. The ns-3 system took care of the hard part for you. The remaining lines of our first ns-3 script, first.cc, do just that:

Simulator::Destroy ();

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

```
Network topology


      10.1.1.0        10.1.2.0


n0 -------------- n1..........n2


point-to-point

In this program we have created 3 point to point nodes n0, n1, n2.
Node n0 has IP address 10.1.1.1 and n3 has 10.1.2.2. Node n1 has 2
interfaces(10.1.1.2 and 10.1.2.1).  OnOffHelper application is used
to generate the traffic at source node n0. Packets move from n0 to n2
via n1. Acknowledgment is sent from n2 to n0 via n1. Details of the
flow(Number of packets sent, received and dropped) can be verified by
using tracemetrics(lab1.tr file).
```

Program
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/traffic-control-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Lab-Program-1");

int main (int argc, char *argv[])
{
std::string socketType= "ns3::TcpSocketFactory";;

CommandLine cmd;
cmd.Parse (argc, argv);

NodeContainer nodes;
nodes.Create (3);            //3 point-to-point nodes are created


InternetStackHelper stack;
stack.Install (nodes);     //TCP-IP layer functionality configured on
all nodes


//Bandwidth and delay set for the point-to-point channel. Vary these
parameters to  //see the variation in number of packets
sent/received/dropped.
PointToPointHelper p2p1;
p2p1.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
p2p1.SetChannelAttribute ("Delay", StringValue ("1ms"));

//Set the base address for the first network(nodes n0 and n1)
```

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");


NetDeviceContainer devices;
devices = p2p1.Install (nodes.Get (0), nodes.Get (1));
Ipv4InterfaceContainer interfaces = address.Assign (devices);


//Set the base address for the second network(nodes n1 and n2)
devices = p2p1.Install (nodes.Get (1), nodes.Get (2));
address.SetBase ("10.1.2.0", "255.255.255.0");
interfaces = address.Assign (devices);


//RateErrorModel allows us to introduce errors into a Channel at a
given rate.
//Vary the error rate value to see the variation in number of packets
dropped
Ptr<RateErrorModel>em = CreateObject<RateErrorModel> ();
em->SetAttribute ("ErrorRate", DoubleValue (0.00002));
devices.Get (1)->SetAttribute ("ReceiveErrorModel", PointerValue
(em));


//create routing table at all nodes
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();


uint32_t payloadSize = 1448;
OnOffHelper onoff (socketType, Ipv4Address::GetAny ());


//Generate traffic by using OnOff application
onoff.SetAttribute ("OnTime",  StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
onoff.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute ("PacketSize", UintegerValue (payloadSize));
onoff.SetAttribute ("DataRate", StringValue ("50Mbps")); //bit/s


uint16_t port = 7;
//Install receiver (for packetsink) on node 2
Address localAddress1 (InetSocketAddress (Ipv4Address::GetAny (),
port));
PacketSinkHelper packetSinkHelper1 (socketType, localAddress1);
ApplicationContainer sinkApp1 = packetSinkHelper1.Install (nodes.Get
(2));
sinkApp1.Start (Seconds (0.0));
sinkApp1.Stop (Seconds (10));


//Install sender app on node 0
ApplicationContainer apps;
AddressValue remoteAddress (InetSocketAddress (interfaces.GetAddress
(1), port));
onoff.SetAttribute ("Remote", remoteAddress);
apps.Add (onoff.Install (nodes.Get (0)));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (10));


Simulator::Stop (Seconds (10));


AsciiTraceHelper ascii;
```
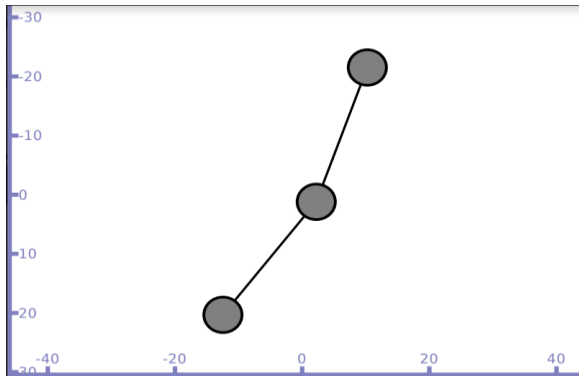
```
p2p1.EnableAsciiAll (ascii.CreateFileStream ("lab1.tr"));

//Run the simulator
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```
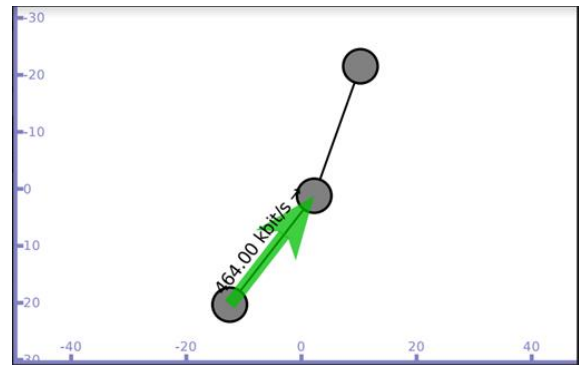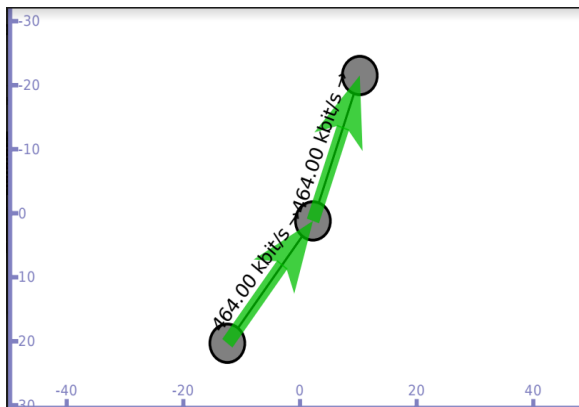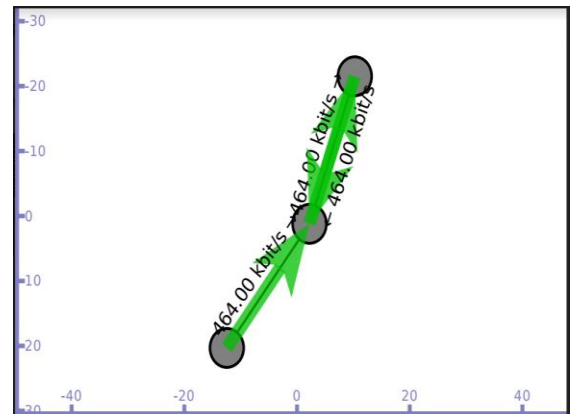
./waf - - run scratch/Program1 - -vis
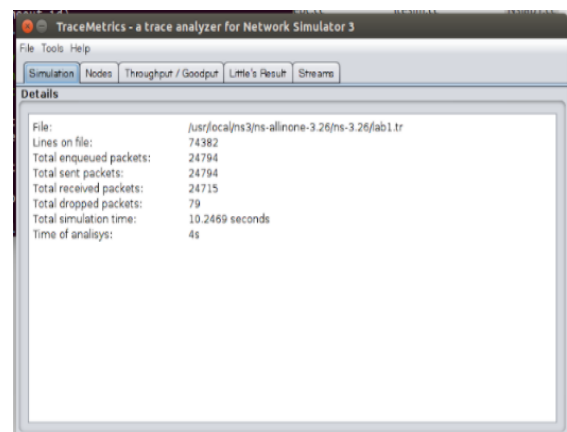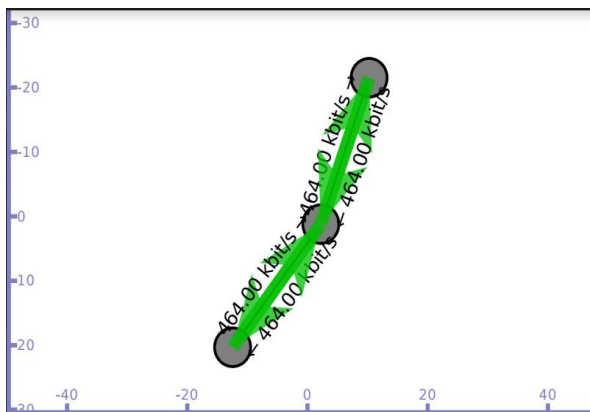
Output

Packet sent from n0 to n1 and then to n2

Acknowledgment sent from n2

Flow details on trace file lab1.tr

2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

```
Network topology

n0    n1    n2    n3    n4    n5
|     |     |     |     |     |
============================
CSMA channel with base IP 10.1.1.0

In this program we have created 6 CSMA nodes n0, n1, n2, n3, n4 and
n5 with IP addresses 10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4, 10.1.1.5
and 10.1.1.6 respectively.
Nodes n0 and n1 ping node n2, we can visualize the ping messages
transferred between the nodes. Data transfer is also simulated
between the nodes n0 and n2 using UdpSocketFactory to generate
traffic.
```

Program

```cpp
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/internet-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Lab-Program-2");

static void PingRtt (std::string context, Time rtt)
{
std::cout << context <<""<< rtt << std::endl;
}

int main (int argc, char *argv[])
{
CommandLine cmd;
cmd.Parse (argc, argv);

// Here, we will explicitly create six nodes.
NS_LOG_INFO ("Create nodes.");
NodeContainer c;
c.Create (6);

// connect all our nodes to a shared channel.
NS_LOG_INFO ("Build Topology.");
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (10000)));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (0.2)));
NetDeviceContainer devs = csma.Install (c);

// add an ip stack to all nodes.
NS_LOG_INFO ("Add ip stack.");
InternetStackHelper ipStack;
ipStack.Install (c);
```

```
  // assign ip addresses
  NS_LOG_INFO ("Assign ip addresses.");
  Ipv4AddressHelper ip;
  ip.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer addresses = ip.Assign (devs);

  NS_LOG_INFO ("Create Sink.");

  // Create an OnOff application to send UDP datagrams from node zero to
  node 1.
  NS_LOG_INFO ("Create Applications.");
  uint16_t port = 9;   // Discard port (RFC 863)

  OnOffHelper onoff ("ns3::UdpSocketFactory",
  Address (InetSocketAddress (addresses.GetAddress (2), port)));
  onoff.SetConstantRate (DataRate ("500Mb/s"));

  ApplicationContainer app = onoff.Install (c.Get (0));
  // Start the application
  app.Start (Seconds (6.0));
  app.Stop (Seconds (10.0));

  // Create an optional packet sink to receive these packets
  PacketSinkHelper sink ("ns3::UdpSocketFactory",
  Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
  app = sink.Install (c.Get (2));
  app.Start (Seconds (0.0));

  NS_LOG_INFO ("Create pinger");
  V4PingHelper ping = V4PingHelper (addresses.GetAddress (2));
  NodeContainer pingers;
  pingers.Add (c.Get (0));
  pingers.Add (c.Get (1));

  ApplicationContainer apps;
  apps = ping.Install (pingers);
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (5.0));

  // finally, print the ping rtts.
  Config::Connect ("/NodeList/*/ApplicationList/*/$ns3::V4Ping/Rtt",
  MakeCallback (&PingRtt));

  NS_LOG_INFO ("Run Simulation.");

  AsciiTraceHelper ascii;
  csma.EnableAsciiAll (ascii.CreateFileStream ("ping1.tr"));

  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
  }
```
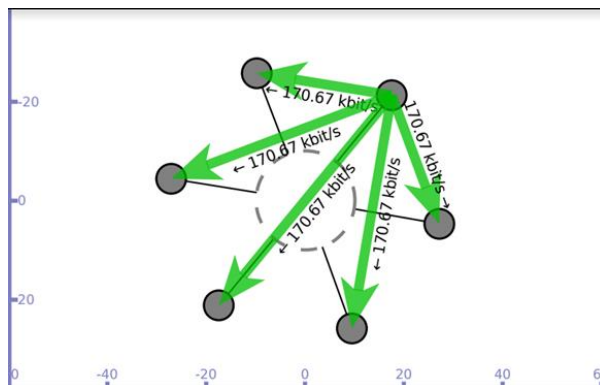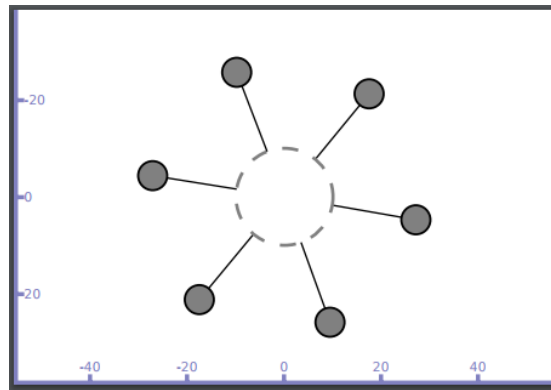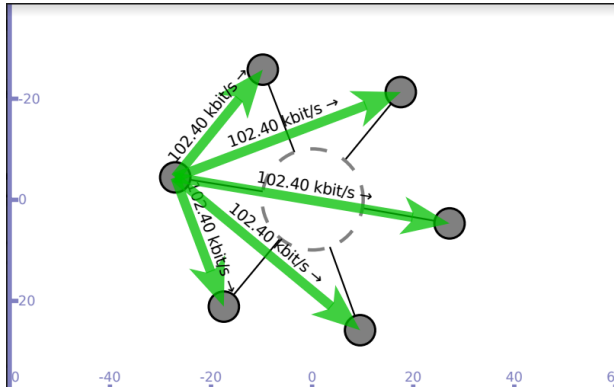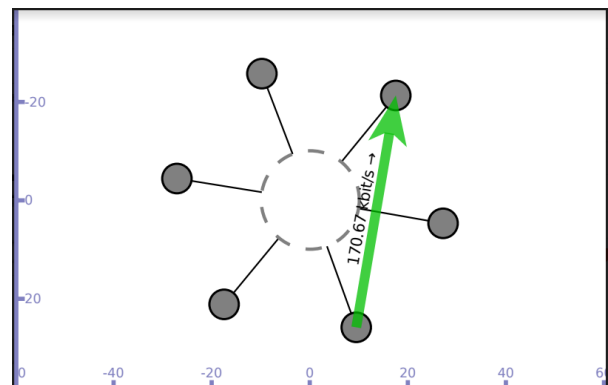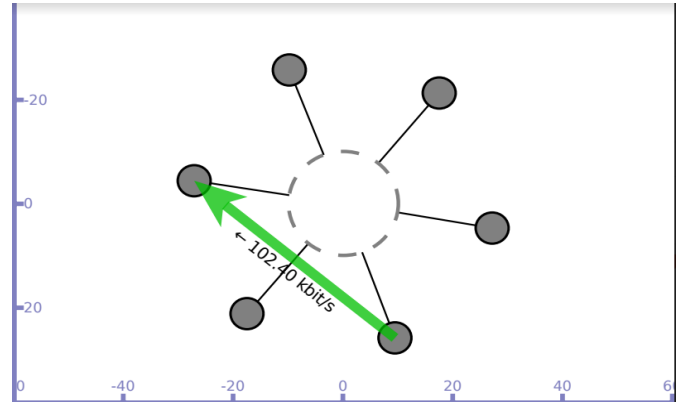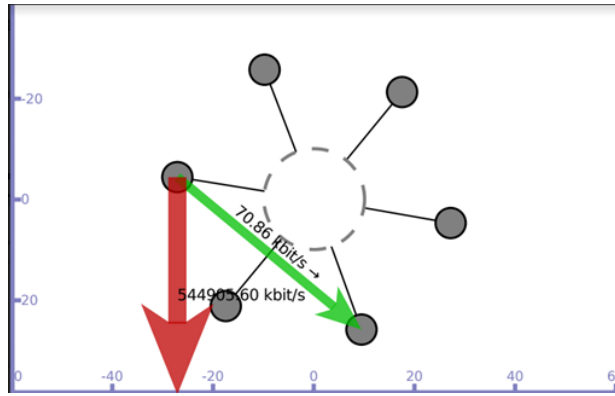
./waf - - run scratch/Program2 - -vis

Node n1 sends ping message to n2 (Broadcast message is generated) and only n2 responds to n1

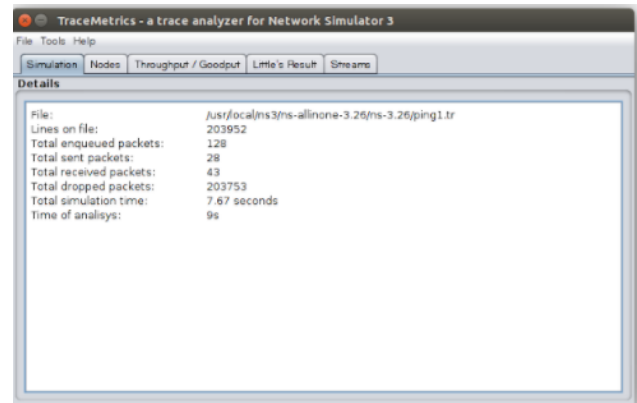**Node n0 sends ping message to n2 (Broadcast message is generated) and only n2 responds to n0**

Data transfer simulated between nodes n0 and n2

Trace file (**ping1.tr**) generated

3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
Network topology

```
n0    n1    n2    n3
|     |     |     |
==================== CSMA channel with base IP 10.1.1.0
                        Source node – n0    sink node - n1



In this program we have created 4 CSMA nodes n0, n1, n2 and n3 with
IP addresses 10.1.1.1, 10.1.1.2, 10.1.1.3 and 10.1.1.4 respectively.
Data transmission is simulated between nodes n0 and n1. Once the cwnd
values are generated, they are exported to .dat file and congestion
graph is plot using gnuplot.
```

Program

```cpp
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include <iostream>
#include "ns3/csma-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("3rd Lab Program");

//MyApp class inherits the ns-3 Application class defined in
//src/network/model/application.h.
//The MyApp class is obligated to override the StartApplication and
//StopApplication methods. These methods are automatically called when MyApp
is //required to start and stop sending data during the simulation.
class MyApp : public Application
{
public:

MyApp ();
virtual ~MyApp();

void Setup (Ptr<Socket> socket, Address address, uint32_t packetSize,
uint32_t nPackets, DataRate dataRate);

private:
virtual void StartApplication (void);
virtual void StopApplication (void);

void ScheduleTx (void);
void SendPacket (void);

Ptr<Socket>     m_socket;
Address         m_peer;
uint32_t        m_packetSize;
uint32_t        m_nPackets;
DataRate        m_dataRate;
EventId         m_sendEvent;
bool            m_running;
```

```
uint32_t        m_packetsSent;
};
MyApp::MyApp ()          // constructor
: m_socket (0),
m_peer (),
m_packetSize (0),
m_nPackets (0),
m_dataRate (0),
m_sendEvent (),
m_running (false),
m_packetsSent (0)
{
}
MyApp::~MyApp()          // destructor
{
m_socket = 0;
}


// initialize member variables.
void MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t packetSize,
uint32_t nPackets, DataRate dataRate)
{
m_socket = socket;
m_peer = address;
m_packetSize = packetSize;
m_nPackets = nPackets;
m_dataRate = dataRate;
}


// Below code is the overridden implementation of
Application::StartApplication. It //does a socket bind operation and
establishes TCP connection with the address //specified in m_peer.

void MyApp::StartApplication (void)
{
m_running = true;
m_packetsSent = 0;
m_socket->Bind ();
m_socket->Connect (m_peer);
SendPacket ();
}


//The next bit of code explains to the Application how to stop creating
simulation  //events.

void MyApp::StopApplication (void)
{
m_running = false;
if (m_sendEvent.IsRunning ())
{
Simulator::Cancel (m_sendEvent);
}
if (m_socket)
{
m_socket->Close ();
}
}
//StartApplication calls SendPacket to start the chain of events that
describes the //Application behavior.
void MyApp::SendPacket (void)
{
Ptr<Packet> packet = Create<Packet> (m_packetSize);
m_socket->Send (packet);
```

RNSIT-CSE
Bengaluru

```
if (++m_packetsSent < m_nPackets)
{
ScheduleTx ();
}
}
```
//It is the responsibility of the Application to keep scheduling the chain of //events, so the next lines call ScheduleTx to schedule another transmit event
//(*a SendPacket*) until the Application decides it has sent enough.

```
void MyApp::ScheduleTx (void)
{
if (m_running)
{
Time tNext (Seconds (m_packetSize * 8 / static_cast<double>
(m_dataRate.GetBitRate ())));
m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket, this);
}
}
```
//Below function logs the current simulation time and the new value of the congestion window every time it is changed.

```
static void CwndChange (uint32_t oldCwnd, uint32_t newCwnd)
{
NS_LOG_UNCOND (Simulator::Now ().GetSeconds () <<"\t"<< newCwnd);
}
```
//trace sink to show where packets are dropped

```
static void RxDrop (Ptr<const Packet> p)
{
NS_LOG_UNCOND ("RxDrop at "<< Simulator::Now ().GetSeconds ());
}
```
//main function

```
int main (int argc, char *argv[])
{
CommandLine cmd;
cmd.Parse (argc, argv);

NS_LOG_INFO ("Create nodes.");
NodeContainer nodes;
nodes.Create (4); //4 csma nodes are created

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("5Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (0.0001)));

NetDeviceContainer devices;
devices = csma.Install (nodes);
```
//RateErrorModel allows us to introduce errors into a Channel at a given *rate*.

```
Ptr<RateErrorModel>em = CreateObject<RateErrorModel> ();
em->SetAttribute ("ErrorRate", DoubleValue (0.00001));
devices.Get (1)->SetAttribute ("ReceiveErrorModel", PointerValue (em));

InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
```

```
Ipv4InterfaceContainer interfaces = address.Assign (devices);

uint16_t sinkPort = 8080;
```

//PacketSink Application is used on the destination node to receive TCP connections //and data.

```
Address sinkAddress (InetSocketAddress (interfaces.GetAddress (1),
sinkPort));
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));

ApplicationContainer sinkApps = packetSinkHelper.Install (nodes.Get (1));
sinkApps.Start (Seconds (0.));
sinkApps.Stop (Seconds (20.));
```

//next two lines of code will create the socket and connect the trace source.

```
Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (nodes.Get (0),
TcpSocketFactory::GetTypeId ());
ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow", MakeCallback
(&CwndChange));
```

//creates an Object of type MyApp

```
Ptr<MyApp> app = CreateObject<MyApp> ();
```

//tell the Application what Socket to use, what address to connect to, how much //data to send at each send event, how many send events to generate and the rate at //which to produce data from those events.

```
app->Setup (ns3TcpSocket, sinkAddress, 1040, 1000, DataRate ("50Mbps"));
nodes.Get (0)->AddApplication (app);
app->SetStartTime (Seconds (1.));
app->SetStopTime (Seconds (20.));

devices.Get (1)->TraceConnectWithoutContext ("PhyRxDrop", MakeCallback
(&RxDrop));

Simulator::Stop (Seconds (20));
Simulator::Run ();
Simulator::Destroy ();

return 0;
}
```

./waf - - run scratch/Program3 - -vis

Output
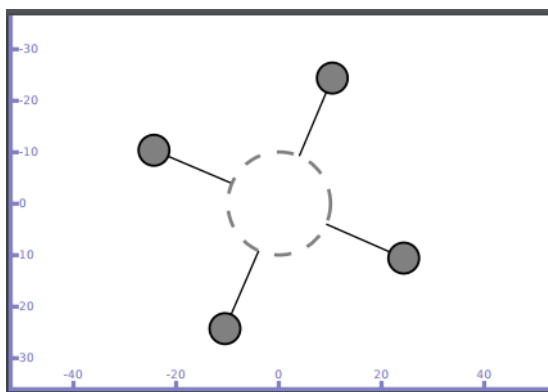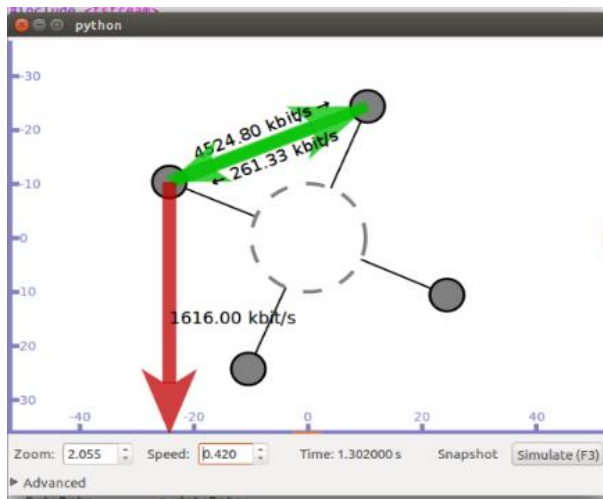
Redirect the output to a file called cwnd.dat

`./waf --run scratch/Program3 > cwnd.dat 2>&1`

Now run gnuplot

gnuplot> set terminal png size 640,480

gnuplot> set output "cwnd.png"

gnuplot> plot "cwnd.dat" using 1:2 title 'Congestion Window' with linespoints

gnuplot> exit
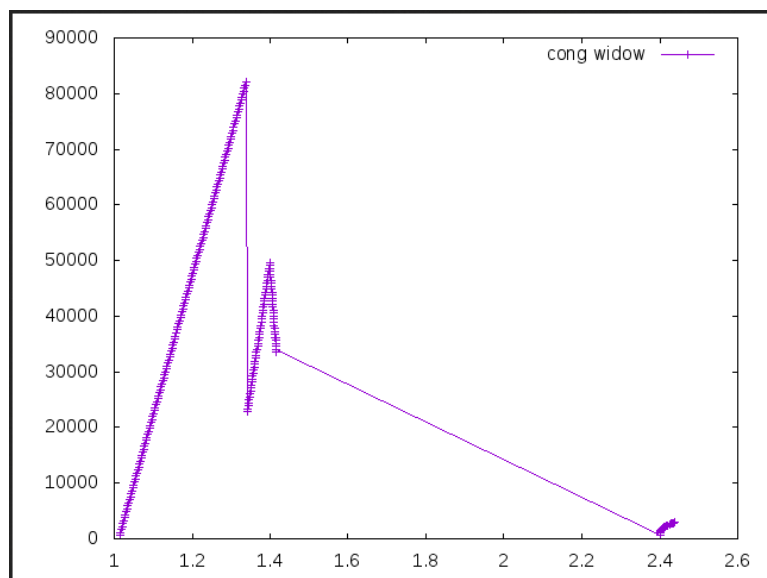
Time in seconds

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

```
Default Network Topology
|
Rank 0   |   Rank 1
---------------------- |---------------------------
Wifi 10.1.3.0
AP
*    *    *    *
|    |    |    |    10.1.1.0
n2   n3   n4   n0   n1
point-to-point  |

In this program we have created 3 wifi (STA/mobile)nodes(n2,n3,n4), 2
point to point nodes(n0,n1) where n0 acts as access point n1 is a
base station. This program establishes connection between
n2(10.1.3.3) and n1(10.1.1.2) through access point(10.1.1.1). The
Performance is measured in terms of throughput of the nodes. It can
be verified using  tracemetrics(Files generated :  Tracefilewifides
and Tracefilewifisrc).
```

Program

```cpp
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int main (int argc, char *argv[])
{
bool verbose = true;
uint32_t nWifi = 3; // 3 wi-fi nodes are created

CommandLine cmd;
cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue ("verbose", "Tell echo applications to log if true",
verbose);
cmd.Parse (argc,argv);

if (verbose)
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}

NodeContainer p2pNodes;
p2pNodes.Create (2); // 2 nodes are n0,n1 are created

PointToPointHelper pointToPoint;
```

```
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);
// 1st node of p2p is also access point

// default PHY layer configuration is used for wifi
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");//AARF= rate
control algorithm

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");// ssid=service set identifier in 802.11
mac.SetType ("ns3::StaWifiMac",
"Ssid", SsidValue (ssid),
"ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

// 2 dimensional grid to initially place sta(stationary nodes)
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
"MinX", DoubleValue (10.0),
"MinY", DoubleValue (-10.0),
"DeltaX", DoubleValue (7.0),
"DeltaY", DoubleValue (12.0),
"GridWidth", UintegerValue (3),
"LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
"Bounds",RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (p2pNodes.Get(1));// stack installed on n1 of p2p
stack.Install (wifiApNode); //stack installed on access point
stack.Install (wifiStaNodes); //stack installed on mobile nodes
```

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

//install echo server application on n1
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (p2pNodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

//install echo client application on n3
UdpEchoClientHelper echoClient (p2pInterfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps =
echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll (ascii.CreateFileStream
("Tracefilewifides.tr"));
phy.EnableAsciiAll (ascii.CreateFileStream ("Tracefilewifisrc.tr"));

Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```
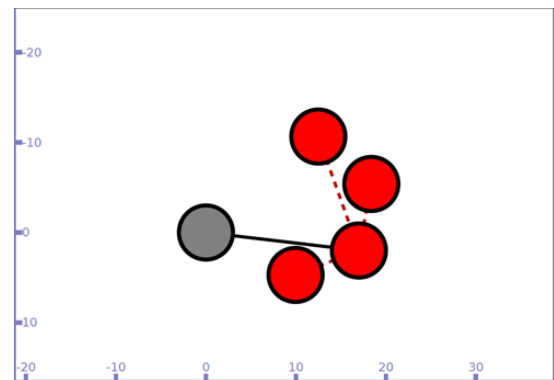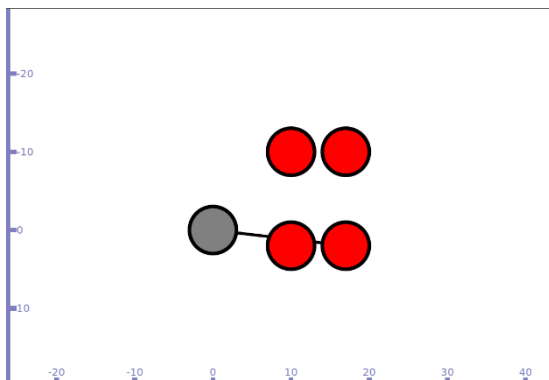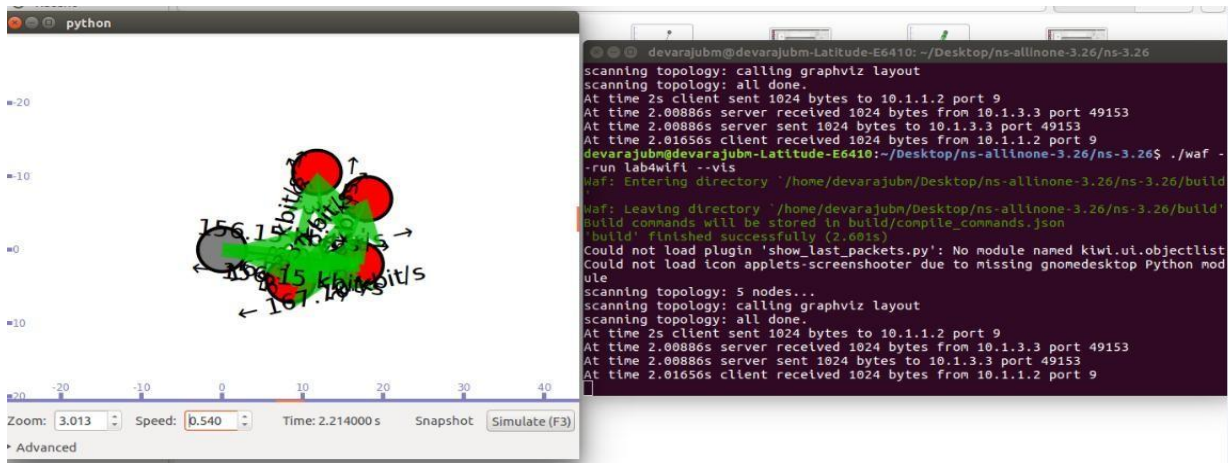
./waf - - run scratch/Program4 - -vis

Output

Trace file is used to see the throughput by using TraceMetrics

5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.

```cpp
#include "ns3/lte-helper.h"
#include "ns3/epc-helper.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/lte-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/config-store.h"
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("EpcFirstExample");

int
main (int argc, char *argv[])
{

  uint16_t numberOfNodes = 2;
  double simTime = 1.1;
  double distance = 60.0;
  double interPacketInterval = 100;

   CommandLine cmd;
   cmd.Parse(argc, argv);

  Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
  Ptr<PointToPointEpcHelper>  epcHelper = CreateObject<PointToPointEpcHelper> ();
  lteHelper->SetEpcHelper (epcHelper);
  ConfigStore inputConfig;
  inputConfig.ConfigureDefaults();
  cmd.Parse(argc, argv);

  Ptr<Node> pgw = epcHelper->GetPgwNode ();
  // Create a single RemoteHost
  NodeContainer remoteHostContainer;
  remoteHostContainer.Create (1);
  Ptr<Node> remoteHost = remoteHostContainer.Get (0);
  InternetStackHelper internet;
  internet.Install (remoteHostContainer);

  // Create the Internet
  PointToPointHelper p2ph;
  p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("100Gb/s")));
  p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
  p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
  NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
  Ipv4AddressHelper ipv4h;
  ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
  Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);
  // interface 0 is localhost, 1 is the p2p device
  Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress (1);
```

```
  Ipv4StaticRoutingHelper ipv4RoutingHelper;
  Ptr<Ipv4StaticRouting> remoteHostStaticRouting =
ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<Ipv4> ());
  remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address
("7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);

  NodeContainer ueNodes;
  NodeContainer enbNodes;
  enbNodes.Create(numberOfNodes);
  ueNodes.Create(numberOfNodes);

  // Install Mobility Model
  Ptr<ListPositionAllocator> positionAlloc =
CreateObject<ListPositionAllocator> ();
  for (uint16_t i = 0; i < numberOfNodes; i++)
    {
      positionAlloc->Add (Vector(distance * i, 100, 100));
    }
  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.SetPositionAllocator(positionAlloc);
  mobility.Install(enbNodes);
  mobility.Install(ueNodes);

  // Install LTE Devices to the nodes
  NetDeviceContainer enbLteDevs = lteHelper->InstallEnbDevice (enbNodes);
  NetDeviceContainer ueLteDevs = lteHelper->InstallUeDevice (ueNodes);

  // Install the IP stack on the UEs
  internet.Install (ueNodes);
  Ipv4InterfaceContainer ueIpIface;
  ueIpIface = epcHelper->AssignUeIpv4Address (NetDeviceContainer
(ueLteDevs));
  // Assign IP address to UEs, and install applications
  for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
    {
      Ptr<Node> ueNode = ueNodes.Get (u);
      // Set the default gateway for the UE
      Ptr<Ipv4StaticRouting> ueStaticRouting =
ipv4RoutingHelper.GetStaticRouting (ueNode->GetObject<Ipv4> ());
      ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress
(), 1);
    }

  // Attach one UE per eNodeB
  for (uint16_t i = 0; i < numberOfNodes; i++)
      {
        lteHelper->Attach (ueLteDevs.Get(i), enbLteDevs.Get(i));
        // side effect: the default EPS bearer will be activated
      }
```

```cpp
  // Install and start applications on UEs and remote host
  uint16_t dlPort = 1234;
  uint16_t ulPort = 2000;
  uint16_t otherPort = 3000;
  ApplicationContainer clientApps;
  ApplicationContainer serverApps;
  for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
    {
      ++ulPort;
      ++otherPort;
      PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), dlPort));
      PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), ulPort));
      PacketSinkHelper packetSinkHelper ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), otherPort));
      serverApps.Add (dlPacketSinkHelper.Install (ueNodes.Get(u)));
      serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
      serverApps.Add (packetSinkHelper.Install (ueNodes.Get(u)));

      UdpClientHelper dlClient (ueIpIface.GetAddress (u), dlPort);
      dlClient.SetAttribute ("Interval", TimeValue (MilliSeconds
(interPacketInterval)));
      dlClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper ulClient (remoteHostAddr, ulPort);
      ulClient.SetAttribute ("Interval", TimeValue (MilliSeconds
(interPacketInterval)));
      ulClient.SetAttribute ("MaxPackets", UintegerValue(1000000));

      UdpClientHelper client (ueIpIface.GetAddress (u), otherPort);
      client.SetAttribute ("Interval", TimeValue (MilliSeconds
(interPacketInterval)));
      client.SetAttribute ("MaxPackets", UintegerValue(1000000));

      clientApps.Add (dlClient.Install (remoteHost));
      clientApps.Add (ulClient.Install (ueNodes.Get(u)));
      if (u+1 < ueNodes.GetN ())
        {
          clientApps.Add (client.Install (ueNodes.Get(u+1)));
        }
      else
        {
          clientApps.Add (client.Install (ueNodes.Get(0)));
        }
    }
  serverApps.Start (Seconds (0.01));
  clientApps.Start (Seconds (0.01));
  lteHelper->EnableTraces ();
  // Uncomment to enable PCAP tracing
  p2ph.EnablePcapAll("lena-epc-first");

  AsciiTraceHelper ascii;
  p2ph.EnableAsciiAll(ascii.CreateFileStream("cdma.tr"));

  Simulator::Stop(Seconds(simTime));
  Simulator::Run();
```
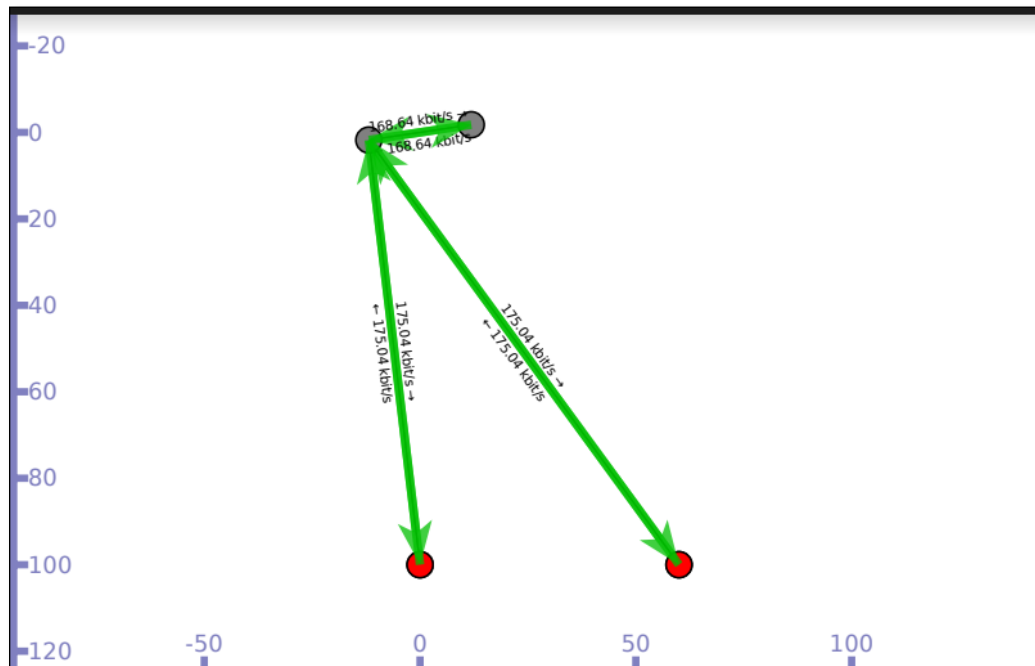
```
    /*GtkConfigStore config;
    config.ConfigureAttributes();*/

    Simulator::Destroy();
    return 0;

}
```

./waf - - run scratch/Program5 - -vis

Output

PART-B

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

```java
import java.io.*;
class crc
{
    public static void main(String args[]) throws IOException
    {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        int[] data;
        int[] div;
        int[] divisor;
        int[] rem;
        int[] crc;
        int data_bits, divisor_bits, tot_length;

        System.out.println("Enter number of data bits : ");
        data_bits=Integer.parseInt(br.readLine());
        data=new int[data_bits];

        System.out.println("Enter data bits : ");
        for(int i=0; i<data_bits; i++)
            data[i]=Integer.parseInt(br.readLine());

        System.out.println("Enter number of bits in divisor : ");
        divisor_bits=Integer.parseInt(br.readLine());
        divisor=new int[divisor_bits];

        System.out.println("Enter Divisor bits : ");
        for(int i=0; i<divisor_bits; i++)
            divisor[i]=Integer.parseInt(br.readLine());


        tot_length=data_bits+divisor_bits-1;

        div=new int[tot_length];
        rem=new int[tot_length];
        crc=new int[tot_length];
    /*----------------- CRC GENERATION----------------------*/
        for(int i=0;i<data.length;i++)
            div[i]=data[i];

        System.out.print("Dividend (after appending 0's) are : ");
        for(int i=0; i< div.length; i++)
            System.out.print(div[i]);
        System.out.println();

        for(int j=0; j<div.length; j++){
            rem[j] = div[j];
        }
        rem=divide(div, divisor, rem);
        for(int i=0;i<div.length;i++)          //append dividend and
ramainder
        {
            crc[i]=(div[i]^rem[i]);
        }
        System.out.println();
        System.out.println("CRC code : ");
        for(int i=0;i<crc.length;i++)
            System.out.print(crc[i]);

    /*------------------ERROR DETECTION-------------------*/
        System.out.println();
```

```
        System.out.println("Enter CRC code of "+tot_length+" bits : ");
        for(int i=0; i<crc.length; i++)
            crc[i]=Integer.parseInt(br.readLine());

        for(int j=0; j<crc.length; j++){
            rem[j] = crc[j];
        }
        rem=divide(crc, divisor, rem);

        for(int i=0; i< rem.length; i++)
        {
            if(rem[i]!=0)
            {
                System.out.println("Error");
                break;
            }
            if(i==rem.length-1)
                System.out.println("No Error");
        }

        System.out.println("THANK YOU.... :)");
    }

    static int[] divide(int div[],int divisor[], int rem[])
    {
        int cur=0;
        while(true)
        {
            for(int i=0;i<divisor.length;i++)
                rem[cur+i]=(rem[cur+i]^divisor[i]);

            while(rem[cur]==0 && cur!=rem.length-1)
                cur++;

            if((rem.length-cur)<divisor.length)
                break;
        }
        return rem;
    }
}


OUTPUT :
Enter number of data bits :
7
Enter data bits :
1
0
1
1
0
0
1
Enter number of bits in divisor :
3
Enter Divisor bits :
1
0
1
Data bits are : 1011001
divisor bits are : 101
Dividend (after appending 0's) are : 101100100
```

```
CRC code :
101100111
Enter CRC code of 9 bits :
1
0
1
1
0
0
1
0
1
crc bits are : 101100101
Error
THANK YOU.... :)
Press any key to continue...
2) Enter number of data bits :
7
Enter data bits :
1
0
1
1
0
0
1
Enter number of bits in divisor :
3
Enter Divisor bits :
1
0
1
Dividend (after appending 0's) are : 101100100

CRC code :
101100111
Enter CRC code of 9 bits :
1
1
1
1
0
0
1
0
1
No Error
THANK YOU.... :)
```

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

```java
package bellmanford;
import java.util.*;
public class Bellmanford {
static int n, dest;
static double[] prevDistanceVector, distanceVector;
static double[][] adjacencyMatrix;
public static void main(String[] args) {
 // TODO code application logic here
Scanner scanner = new Scanner(System.in);
System.out.println("Enter number of nodes");
n = scanner.nextInt();
adjacencyMatrix = new double[n][n];
System.out.println("Enter Adjacency Matrix (Use 'Infinity' for No Link)");
for (int i = 0; i< n; i++)
   for (int j = 0; j < n; j++)
       adjacencyMatrix[i][j] = scanner.nextDouble();
       System.out.println("Enter destination vertex");
       dest = scanner.nextInt();
       distanceVector = new double[n];
       for (int i = 0; i< n; i++)
             distanceVector[i] = Double.POSITIVE_INFINITY;
             distanceVector[dest - 1] = 0;

             bellmanFordAlgorithm();

       System.out.println("Distance Vector");
         for (int i = 0; i< n; i++) {
             if (i == dest - 1) {
                 continue;
             }
System.out.println("Distance from " + (i + 1) + " is " + distanceVector[i]);
         }
System.out.println();
    }

    static void bellmanFordAlgorithm() {
        for (int i = 0; i< n - 1; i++) {
prevDistanceVector = distanceVector.clone();
            for (int j = 0; j < n; j++) {
                double min = Double.POSITIVE_INFINITY;
                for (int k = 0; k < n; k++) {
                    if (min >adjacencyMatrix[j][k] + prevDistanceVector[k])
{
                        min = adjacencyMatrix[j][k] + prevDistanceVector[k];
                    }
                }
distanceVector[j] = min;
            }
        }
    }
}
```

OUTPUT
run:
Enter number of nodes
6
Enter Adjacency Matrix (Use 'Infinity' for No Link)
0 3 2 5 99 99
3 0 99 1 4 99
2 99 0 2 99 1

```
5 1 2 0 3 99
99 4 99 3 0 2
99 99 1 99 2 0
Enter destination vertex
6
Distance Vector
Distance from 1 is 3.0
Distance from 2 is 4.0
Distance from 3 is 1.0
Distance from 4 is 3.0
Distance from 5 is 2.0
```

9. 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.

Server
```java
package org.tcp;
import java.net.*;
import java.io.*;

public class tcpServer {
    public static void main(String[] args)
    {
        try
        {
            ServerSocket se = new ServerSocket(1537);
            System.out.println("Server waiting");

            Socket server = se.accept();
            System.out.println("Connection established");

            BufferedReader k = new BufferedReader (new
InputStreamReader(server.getInputStream()));
            String filename = k.readLine();
            FileReader f = null;
            BufferedReader ff = null;
            DataOutputStream sendToClient = new
DataOutputStream(server.getOutputStream());
            File file = new File(filename);
            if(file.exists())
            {
                sendToClient.writeBytes("Yes\n");
                f = new FileReader(filename);
                ff = new BufferedReader(f);
                String string;
                while((string = ff.readLine())!=null)
                    sendToClient.writeBytes(string+"\n");
            }
            else
            {
                sendToClient.writeBytes("No\n");
            }
            server.close();
            k.close();
            sendToClient.close();
            f.close();
            ff.close();
            se.close();
        }
        catch(Exception ex) {}
    }
}
```

Client
```java
package org.tcp;
import java.net.*;
import java.io.*;

public class tcpClient {

    public static void main(String[] args)
    {
        try
        {
```

```
                Socket client = new Socket("localhost", 1537);
                BufferedReader k = new BufferedReader(new
InputStreamReader(System.in));
                System.out.println("Enter file location:");
                String filename = k.readLine();
                DataOutputStream sendToServer = new
DataOutputStream(client.getOutputStream());
                sendToServer.writeBytes(filename+"\n");

                BufferedReader i = new BufferedReader(new
InputStreamReader(client.getInputStream()));
                String string = i.readLine();
                if(string.equals("Yes"))
                {
                        while((string=i.readLine())!=null)
                                System.out.println(string);
                }
                else
                        System.out.println("File not found");

                k.close();
                client.close();
                sendToServer.close();
                i.close();
        }
        catch(Exception ex) {}
    }
}
```

OUTPUT
Server Console
Server waiting
Connection established

Client Console
Enter file location:
F:\test.txt
Testing TCP socket

10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.

```java
package labcn;
import java.net.*;
import java.util.*;

public class udpser {

    public static void main(String[] args)
    {
        try
        {
            DatagramSocket server = new DatagramSocket(1537);

            System.out.println("Enter server message:");
            Scanner scan = new Scanner(System.in);

            while(true)
            {
                byte[] buffer = new byte[1000];
                DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
                server.receive(request);

                String message = scan.nextLine();
                byte[] sendMessage = message.getBytes();
                DatagramPacket reply = new
DatagramPacket(sendMessage, sendMessage.length, request.getAddress(),
request.getPort());
                server.send(reply);

                server.close();
                scan.close();
            }
        }
        catch(Exception ex) {}
    }

}

package labcn;
import java.net.*;
public class udpcli {

    public static void main(String[] args)
    {
        try
        {
            DatagramSocket client = new DatagramSocket();
            int serverSocket = 1537;
            InetAddress host = InetAddress.getByName("127.0.0.1");

            String message = "Text Message";
            byte[] sendMessage = message.getBytes();
            DatagramPacket request = new DatagramPacket(sendMessage,
sendMessage.length, host, serverSocket);
            client.send(request);

            byte[] buffer = new byte[1000];
            DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
            client.receive(reply);
```

```
                    System.out.println("Client received:\n "+new
String(reply.getData()));

                    client.close();
            }
            catch(Exception ex) {}
        }

}
```

OUTPUT
Sever console
Enter server message:
Hello udp client

Client console
Client received:
Hello udp client

11. Write a program for simple RSA algorithm to encrypt and decrypt the data.

```java
package rsa4;
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

public class Rsa4 {
     static BigInteger p, q, n, phi_n, e, d;
    static SecureRandomsecureRandom;
    static int bitLength = 64;

    static String encrypt(String msg) {
        return new BigInteger(msg).modPow(e, n).toString();
    }

    static String decrypt(String cipher) {
        return new BigInteger(cipher).modPow(d, n).toString();
    }

        // TODO code application logic here

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        secureRandom = new SecureRandom();

        p = BigInteger.probablePrime(bitLength, secureRandom);
        q = BigInteger.probablePrime(bitLength, secureRandom);
        n = p.multiply(q);
        phi_n =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

        e = BigInteger.probablePrime(bitLength / 2, secureRandom);
        while (e.gcd(phi_n).compareTo(BigInteger.ONE) != 0
&&e.compareTo(phi_n) < 0) {
            e = e.add(BigInteger.ONE);
        }

        d = e.modInverse(phi_n);

        System.out.println("P assigned as: " + p);
        System.out.println("Q assigned as: " + q);
        System.out.println("N assigned as: " + n);
        System.out.println("PHI_N assigned as: " + phi_n);
        System.out.println("\nEnter Message");
        String msg = scanner.next();
        String encryptedMessage = encrypt(msg);
        System.out.println("Encrypted Message: " + encryptedMessage);
        String decryptedMessage = decrypt(encryptedMessage);
        System.out.println("Decrypted Message: " + decryptedMessage);

    }
}
```
OUTPUT
P assigned as: 13330063847181728989
Q assigned as: 10448652783677772539
N assigned as: 139281208723457810526403094838284433071
PHI_N assigned as: 139281208723457810502624378207424931544

Enter Message
5432101

```
Encrypted Message: 6074209369875310577291580727 0189084958
Decrypted Message: 5432101
```

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

```java
package lb;
import java.util.*;
public class Lb{
        public static void main(String[] args) {
            System.out.println("enter the number of time intervals");
            Scanner sc=new Scanner(System.in);
            int n=sc.nextInt();
            int t[]=new int[n];
            System.out.println("enter the time intervals");
            for(int i=0;i<n;i++)
                    t[i]=sc.nextInt();
            System.out.println("enter i and l");
            int i=sc.nextInt();
            int l=sc.nextInt();
            int lct=t[0];
            int x=0,y=0;
            for(int j=0;j<n;j++)
            {
                    y=x-(t[j]-lct);
                    if(y>l)
                    {
                            System.out.println("nonconforming packet"+t[j]);
                    }
                            else
                                        x=y+i;
                                lct=t[j];
                                System.out.println("conforming packet"+t[j]);
                    }
            }
        }
}
```

OUTPUT
enter the number of time intervals
11
enter the time intervals
1 2 3 5 6 8 11 12 13 15 19
enter i and l
4
4
conforming packet1
conforming packet2
nonconforming packet3
conforming packet3
nonconforming packet5
conforming packet5
nonconforming packet6
conforming packet6
nonconforming packet8
conforming packet8
conforming packet11
nonconforming packet12
conforming packet12
nonconforming packet13
conforming packet13
nonconforming packet15
conforming packet15
conforming packet19

Appendix-A

This section lists the viva questions.

1. What are functions of different layers?

2. Differentiate between TCP/IP Layers and OSI Layers

3. Differentiate between TCP and UDP.

4. Differentiate between Connectionless and connection oriented connection.

5. What is meant by subnet?

6. What is meant by Gateway?

7. What is an IP address?

8. What is MAC address?

9. Define Socket

10. Why IP address is required when we have MAC address?

11. What is meant by port?

12. What are ephemerical port number and well known port numbers?

13. What is a socket?

14. What are the parameters of socket()?

15. Describe bind(), listen(), accept(),connect(), send() and recv().

16. What are system calls? Mention few of them.

17. What is IPC? Name three techniques.

18. What type of protocol is BGP?

19. What type of protocol is OSPF?

20. Difference between ARP and RARP.

21. What is Distance Vector Routing

22. What is flooding.

23. What is three way handshake.

24. Disadvantages of Stop and wait protocol.

25. Differentiate bridges from switches.

26. What is a Router.

27. What is routing.

28. What is the role of DNS.

29. What type of transport protocol is used for DNS.

30. What are functions of different layers?

31. Differentiate between TCP/IP Layers and OSI Layers

32. Why header is required?

33. What is the use of adding header and trailer to frames?

34. What is encapsulation?

35. Why fragmentation requires?

36. What is MTU?

37. Which layer imposes MTU?

38. Differentiate between flow control and congestion control.

39. Differentiate between Point-to-Point Connection and End-to-End connections.

40. What are protocols running in different layers?

41. What is Protocol Stack?

42. Differentiate between TCP and UDP.

43. Differentiate between Connectionless and connection oriented connection.

44. Why frame sorting is required?

45. What is meant by subnet?

46. What is meant by Gateway?

47. What is an IP address?

48. What is MAC address?

49. Why IP address is required when we have MAC address?

50. What is meant by port?

51. What are ephemerical port number and well known port numbers?

52. What is a socket?

53. What are the parameters of socket()?

54. Describe bind(), listen(), accept(),connect(), send() and recv().

55. What are system calls? Mention few of them.

56. What is IPC? Name three techniques.

57. Explain open(), close() with parameters.

58. What is meant by file descriptor?

59. What is meant by traffic shaping?

60. How do you classify congestion control algorithms?

61. Differentiate between Leaky bucket and Token bucket.

62. How do you implement Leaky bucket?

63. How do you generate busty traffic?

64. What is the polynomial used in CRC-CCITT?

65. What are the other error detection algorithms?

66. What is difference between CRC and Hamming code?

67. Why Hamming code is called 7,4 code?

68. What is odd parity and even parity?

69. What is meant by syndrome?

70. What is generator matrix?

71. What is spanning tree?

72. Where Pirm's algorithm does finds its use in Networks?

73. Differentiate between Prim's and Kruskal's algorithm.

74. What are Routing algorithms?

75. How do you classify routing algorithms? Give examples for each.

76. What are drawbacks in distance vector algorithm?

77. How routers update distances to each of its neighbor?

78. How do you overcome count to infinity problem?

79. What is cryptography?

80. How do you classify cryptographic algorithms?

81. What is public key?

82. What is private key?

83. What are key, ciphertext and plaintext?

84. What is simulation?

85. What are advantages of simulation?

86. Differentiate between Simulation and Emulation.

87. What is meant by router?

88. What is meant by bridge?

89. What is meant by switch?

90. What is meant by hub?

91. Differentiate between route, bridge, switch and hub.

92. What is ping and telnet?

93. What is FTP?

94. What is BER?

95. What is meant by congestion window?

96. What is BSS?

97. What is incoming throughput and outgoing throughput?

98. What is collision?

99. How do you generate multiple traffics across different sender-receiver pairs?

100. How do you setup Ethernet LAN?

101. What is meant by mobile host?

102. What is meant by ns3 ?

103. Name few other Network simulators

104. Differentiate between logical and physical address.

105. Which address gets affected if a system moves from one place to another place?

106. What is ICMP? What are uses of ICMP? Name few.

107. Which layer implements security for data?