PART-A

INTRODUCTION TO NS3

NS3 Simulator Basics

- NS-3 is a network simulator
- Developed for network research and education
- Developed afterns-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:
- AC/C++preprocessor for computing dependencies simulation programs are C++ executable sor Python scripts

Features

- It it a discrete event simulator
- Modular design / Open source
- Actively developed (ContrastNS-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 tar ball from www.nsnam.orgthe recent release in your directory. Go to that directory and untar the tar ball.

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

- Include necessary files
- Use appropriate namespace
- Set simulation time resolution(Optional)
- Enable logging for different modules(Optional)
- Create nodes
- Create net devices with MAC and PHY
- Attach Net devices to nodes and set interconnections
- Install protocol stack in nodes
- Set network address for interfaces
- Setup routing
- Install applications in nodes
- Setup tracing(Optional)
- Set application start and stop time
- Set simulation start time(Optional)
- Run simulation
- 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 Node Container 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 Node Container 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 andPHY

PointToPointHelperpointToPoint;

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 subsequentlycreates.

7. Attach Net devices to nodes and set interconnections

NetDeviceContainerdevices;

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 innodes

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 thenetwork 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 theway).

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 generatetraffic.

10. Setup routing

UdpEchoServerHelperechoServer (9);

ApplicationContainerserverApps = 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 usingSetAttribute.

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 ofimplicit 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 tenseconds.

11. Install applications in nodes

UdpEchoClientHelperechoClient (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-bytepacket.

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:

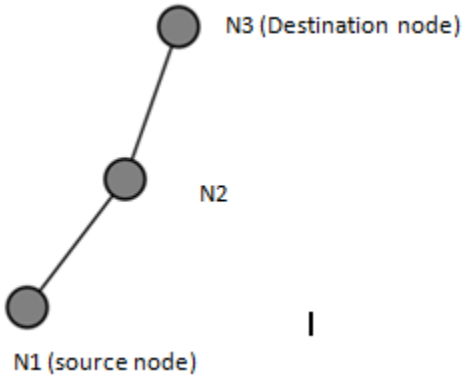
Simulator::Destroy ();

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.

Overview:

Point-to-point network topology is a simple topology that displays the network of exactly two hosts (computers, servers, switches or routers) connected with a cable. Point-to-point topology is widely used in the computer networking and computer architecture. It is also used in the telecommunications systems when we speak about the communication connection of two nodes or endpoints.

- Add 3 nodes
- Establish link between the nodes using TCP/UDP
- Show the number of packets dropped in pyviz and trace route.



Networktopology

```
      10.1.1.0       10.1.2.0
n0--------------n1 ..... n2
Point-to-point
```

Inthisprogramwehavecreated3pointtopointnodesn0, n1, n2.Node n0 has IP address 10.1.1.1 and n3 has 10.1.2.2. Node n1 has 2 interfaces(10.1.1.2and10.1.2.1). OnOffHelper application is used to generatethetrafficatsourcenoden0.Packetsmovefromn0ton2via 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 trace metrics(lab1.trfile).

Program

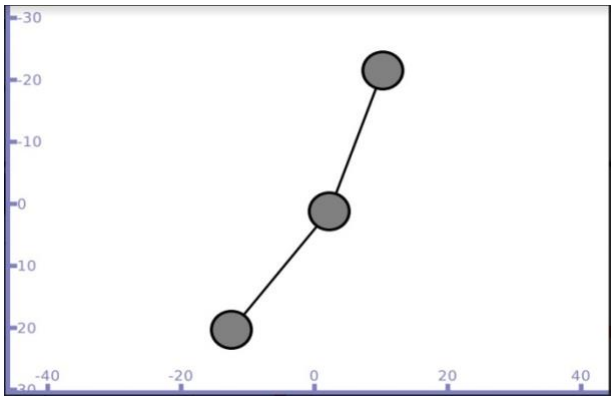
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include"ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include"ns3/traffic-control-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("Lab-Program-1");
```

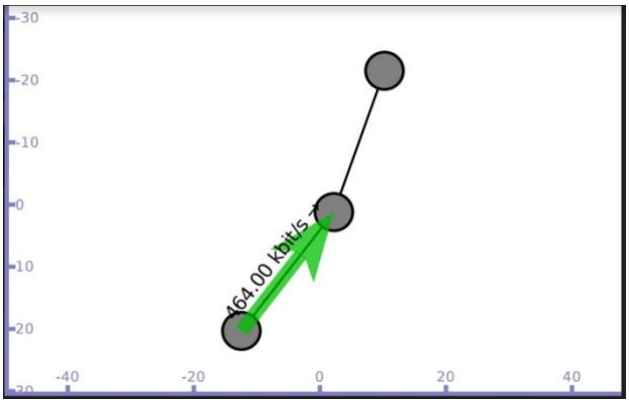
```
int main (int argc, char *argv[])
{
std::string socketType= "ns3::TcpSocketFactory";;

CommandLinecmd;
cmd.Parse (argc, argv);

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

InternetStackHelperstack;
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. PointToPointHelperp2p1;
p2p1.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
p2p1.SetChannelAttribute ("Delay", StringValue ("1ms"));

//Set the base address for the first network(nodes n0 andn1)
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

NetDeviceContainerdevices;
devices = p2p1.Install (nodes.Get (0), nodes.Get (1));
Ipv4InterfaceContainerinterfaces=address.Assign(devices);

//Setthebaseaddressforthesecondnetwork(nodesn1andn2) 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;
OnOffHelperonoff(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
AddresslocalAddress1(InetSocketAddress(Ipv4Address::GetAny(),port));
PacketSinkHelper packetSinkHelper1 (socketType, localAddress1);
ApplicationContainersinkApp1=packetSinkHelper1.Install(nodes.Get(2));
sinkApp1.Start (Seconds(0.0));
sinkApp1.Stop (Seconds (10));
```

```
//Install sender app on node 0
ApplicationContainerapps;
AddressValueremoteAddress(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;
}
```

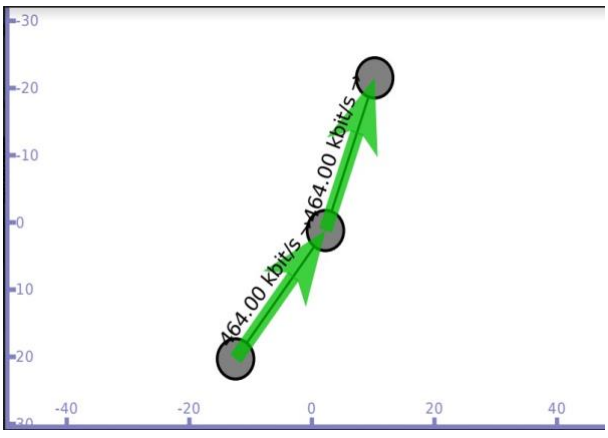
./waf - - run scratch/Program1 - -vis

Output

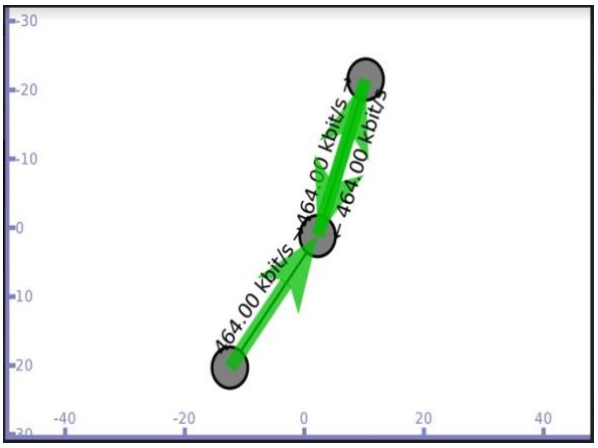




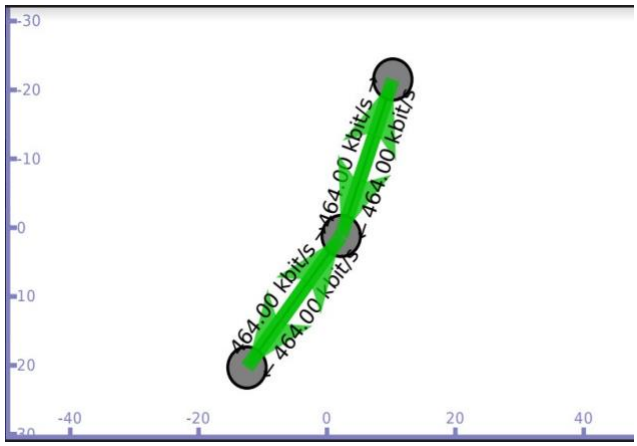
Packetsentfromn0ton1andthenton2

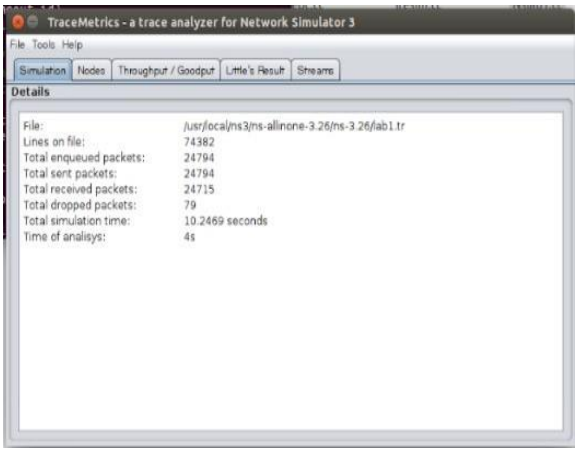


Acknowledgment sent fromn2



Flow details on trace file lab1.tr


-30
-20
-10
0
10
20
-40
-20
0
20
40
464.00 kbit/s
464.00 kbit/s
464.00 kbit/s
464.00 kbit/s


TraceMetrics - a trace analyzer for Network Simulator 3
File Tools Help
Simulation
Nodes
Throughput / Goodput
Little's Result
Streams
Details
File: /usr/local/ns3/ns-allinone-3.26/ns-3.26/lab1.tr
Lines on file: 74382
Total enqueued packets: 24794
Total sent packets: 24794
Total received packets: 24715
Total dropped packets: 79
Total simulation time: 10.2469 seconds
Time of analisys: 4s

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

OVERVIEW

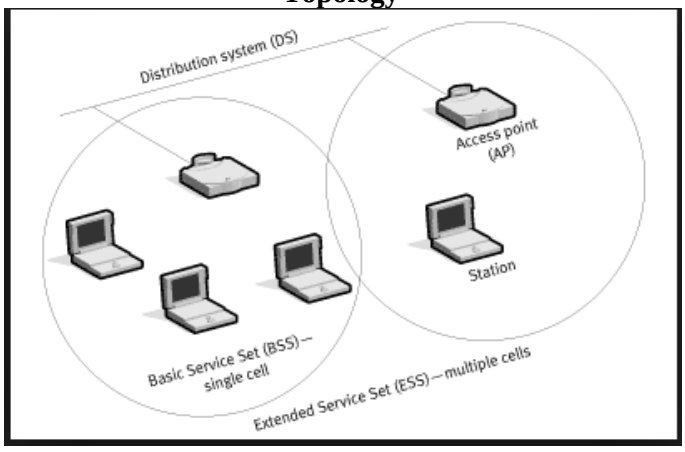
An extended service set (ESS) is one or more interconnected basic service sets (BSSs) and their associated LANs. Each BSS consists of a single access point (AP) together with all wireless client devices (stations, also called STAs) creating a local or enterprise 802.11 wireless LAN (WLAN).

The most basic BSS consists of one AP and one STA.An extended service set, consisting of a set of BSSs, must have a common service set identifier (SSID). The BSSs can all work on the same or different channels.

This helps to boost the signal throughout the wireless network.

A single service set consists of all STAs receiving signals from a given AP and creates an 802.11 wireless LAN (WLAN). Each STA may receive a signal from several APs within their range. Depending on its configuration each STA can, manually or automatically, select the network with which to associate. And multiple APs may share the same SSID as part of an extended service set.

Topology




```
Default Network Topology
|
Rank0    |   Rank1
------------------------ |----------------------------
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
pointtopointnodes(n0,n1)wheren0actsasaccesspointn1isabase 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
measuredintermsofthroughputofthenodes.Itcanbeverifiedusing
tracemetrics(Filesgenerated:    Tracefilewifides andTracefilewifisrc).
```

Program

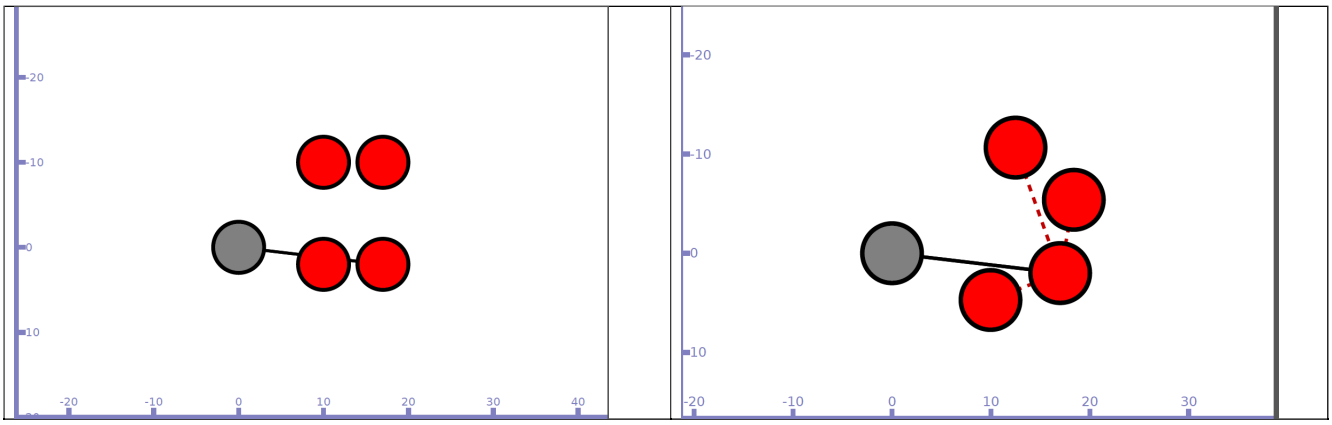
```
#include"ns3/core-module.h"
#include"ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include"ns3/wifi-module.h"
#include"ns3/mobility-module.h"
#include"ns3/internet-module.h"

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

CommandLinecmd;
cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue("verbose","Tellechoapplicationstologiftrue",verbose);
cmd.Parse(argc,argv);

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

NodeContainerp2pNodes;
p2pNodes.Create (2); // 2 nodes are n0,n1 are created

PointToPointHelperpointToPoint;

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

NetDeviceContainerp2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

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

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

WifiHelperwifi;
wifi.SetRemoteStationManager("ns3::AarfWifiManager");//AARF=ratecontrol
algorithm

WifiMacHelpermac;
Ssidssid= Ssid ("ns-3-ssid");// ssid=service set identifier in802.11
mac.SetType ("ns3::StaWifiMac",
"Ssid", SsidValue (ssid),
"ActiveProbing", BooleanValue(false));

NetDeviceContainerstaDevices;
```

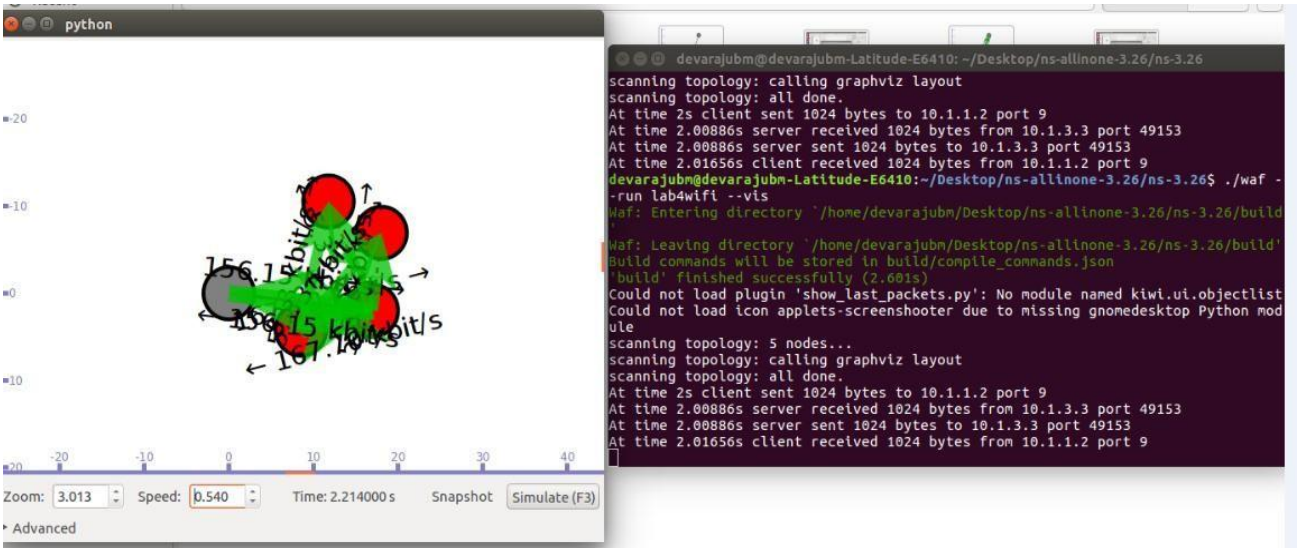
```
staDevices= wifi.Install (phy, mac, wifiStaNodes);

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

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

MobilityHelpermobility;

//2dimensionalgridtoinitiallyplacesta(stationarynodes)
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);

InternetStackHelperstack;
stack.Install(p2pNodes.Get(1));//stackinstalledonn1ofp2p
stack.Install (wifiApNode); //stack installed on access point
stack.Install(wifiStaNodes);//stackinstalledonmobilenodes

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
UdpEchoServerHelperechoServer(9);
ApplicationContainerserverApps= echoServer.Install (p2pNodes.Get(1));
serverApps.Start (Seconds(1.0));
serverApps.Stop (Seconds(10.0));

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

ApplicationContainerclientApps=
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;
}
```

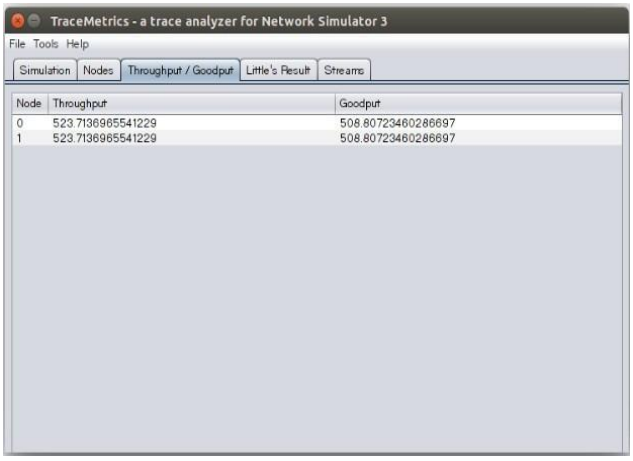
./waf - - run scratch/Program4 - -vis

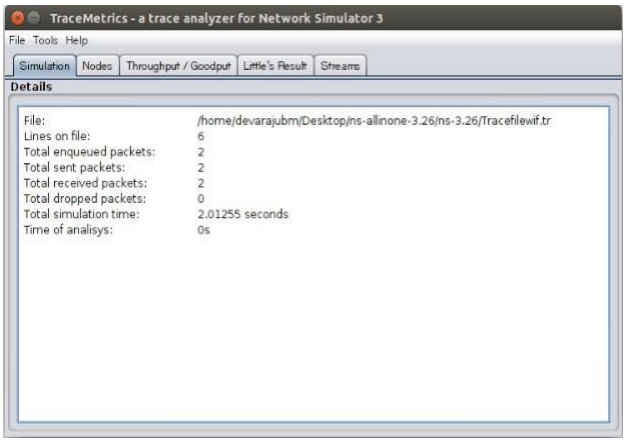
Output





Trace file is used to see the throughput by using TraceMetrics





3. 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.

Overview:

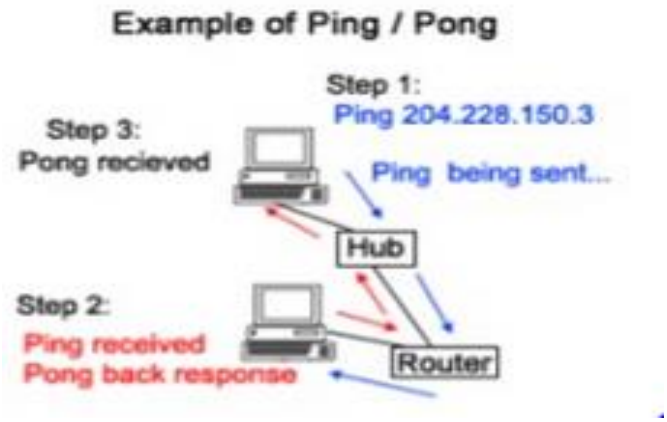
Ping - Packet Internet Groper

What is Ping?

- A computer network utility to determine whether a specific IP address is accessible
- Measures the round-trip time for messages sent from the originating host to a destination computer and back

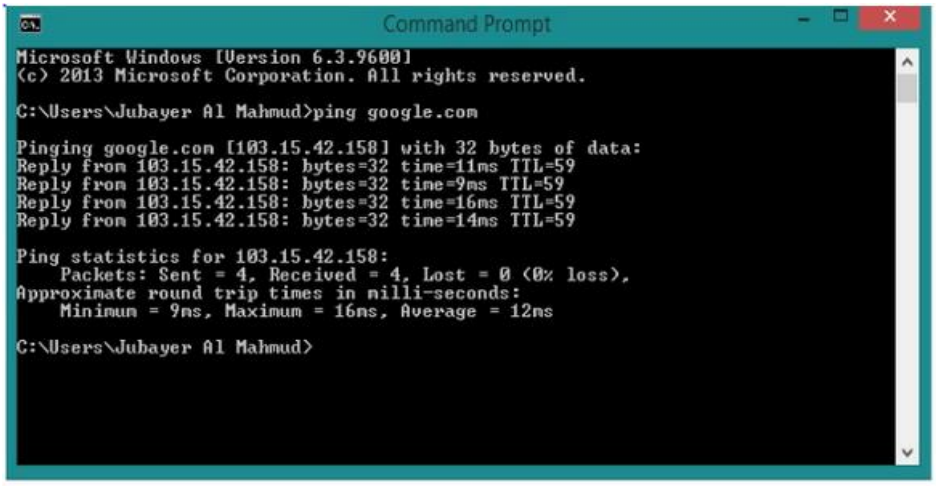
How ping works?

- Operates by sending ICMP echo request (ping) packets to the target host and waiting for an ICMP echo reply (pong)
- It measures the round-trip time from transmission to reception, reporting errors and packet loss

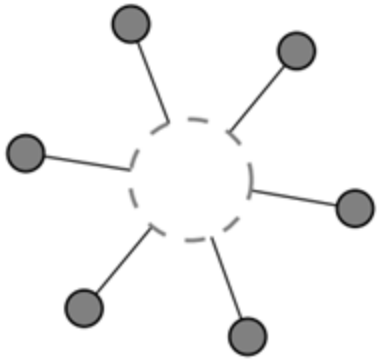


What does Ping test shows?

- a statistical summary of the response packets received
- Also includes minimum, maximum and mean RTT



Topology



Networktopology

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

Inthisprogramwehavecreated6CSMAnodesn0,n1,n2,n3,n4andn5 withIPaddresses10.1.1.1,10.1.1.2,10.1.1.3,10.1.1.4,10.1.1.5and 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 thenodesn0andn2usingUdpSocketFactorytogeneratetraffic.

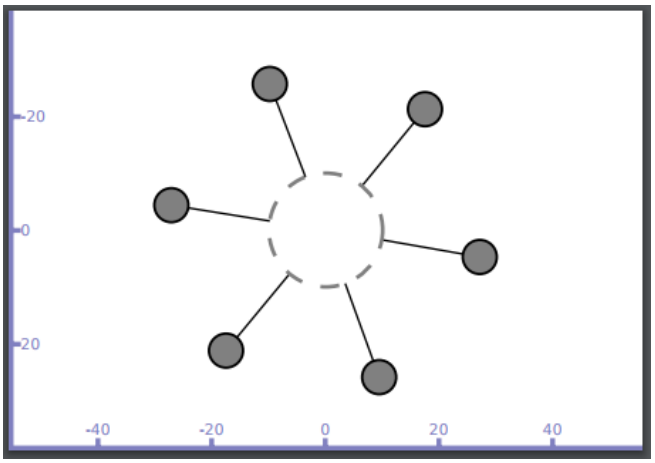
Program

```
#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
("Createnodes.");
```

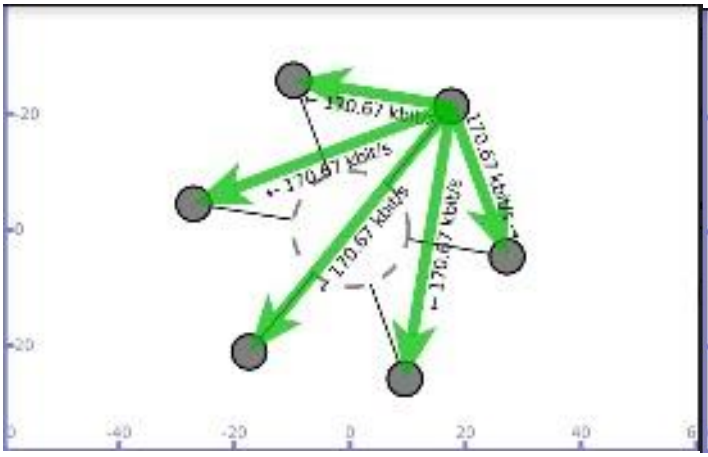
```
NodeContainerc;
c.Create (6);

// connect all our nodes to a shared channel.
NS_LOG_INFO
("BuildTopology.");
CsmaHelpercsma;
csma.SetChannelAttribute("DataRate",DataRateValue(DataRate(10000)));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (0.2)));
NetDeviceContainerdevs= csma.Install(c);

// add an ip stack to
allnodes. NS_LOG_INFO
("Add ip stack.");
InternetStackHelperipStac
k; ipStack.Install (c);

// assign ip addresses

NS_LOG_INFO ("Assign
ipaddresses.");
Ipv4AddressHelper ip;
ip.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer addresses = ip.Assign

(devs); NS_LOG_INFO ("Create Sink.");

//CreateanOnOffapplicationtosendUDPdatagramsfromnodezerotonode1.
NS_LOG_INFO ("CreateApplications.");
uint16_t port=9;     // Discard port (RFC863)

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

ApplicationContainerapp = 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
PacketSinkHelpersink ("ns3::UdpSocketFactory",
Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
app = sink.Install (c.Get (2));
app.Start (Seconds (0.0));

NS_LOG_INFO ("Create pinger");
V4PingHelperping=V4PingHelper(addresses.GetAddress(2));
NodeContainerpingers;
pingers.Add (c.Get(0));
pingers.Add (c.Get(1));

ApplicationContainerapp
s; 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));
```

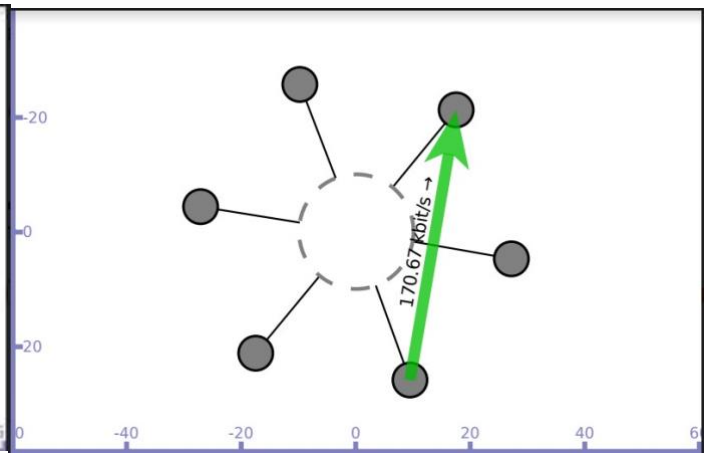
```
NS_LOG_INFO

("RunSimulation.");

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

Simulator::Run ();
Simulator::Destroy
(); NS_LOG_INFO
("Done.");
}
```

./waf - - run scratch/Program2 - -vis

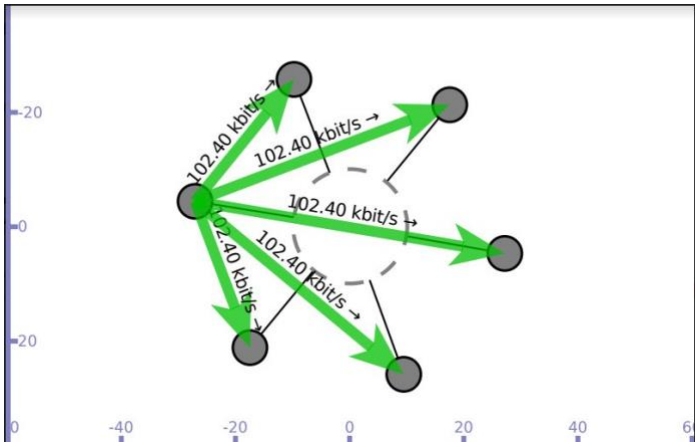


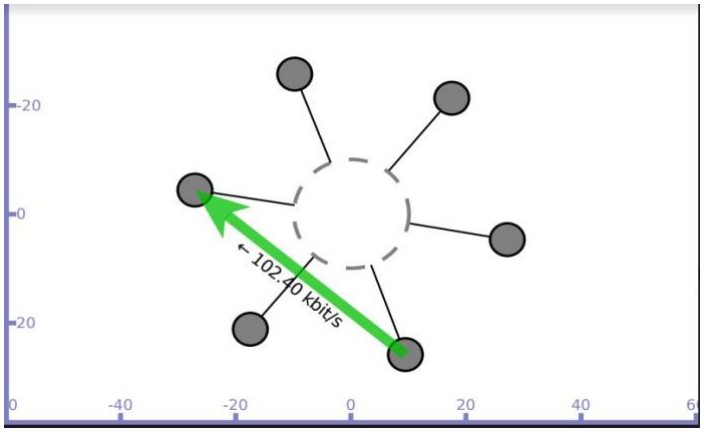
Noden1 sends ping message to n2 (Broadcast message is generated) and only n2 responds to n1



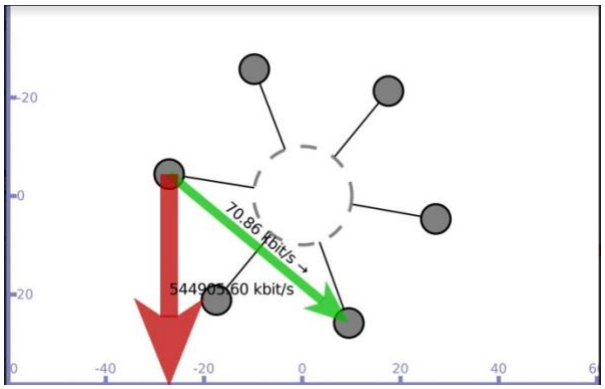


Noden0 sends ping message to n2 (Broadcast message is generated) and only n2 responds to n0

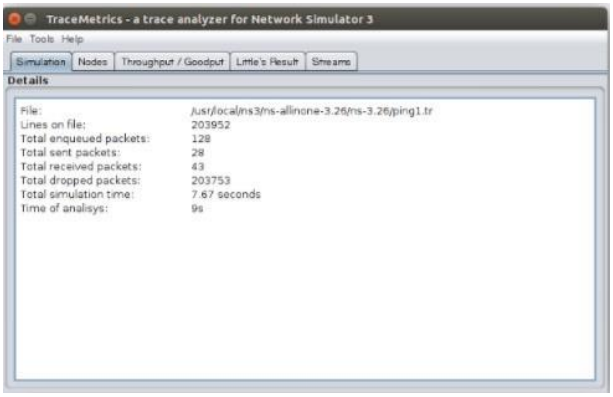




Datatransfersimulatedbetweennodesn0andn2



Trace file(ping1.tr)generated



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

OVERVIEW

Network congestion in data **networking** is the reduced quality of service that occurs when a **network** node or link is carrying more data than it can handle. Typical effects include queuing delay, packet loss or the blocking of new connections.

TCP Slow Start is part of the congestion control algorithms put in place by TCP to help control the amount of data flowing through to a network. This helps regulate the case where too much data is sent to a network and the network is incapable of processing that amount of data, thus resulting in network congestion.

When transmission of data from sender to receiver begins in a network, there may be unknown conditions as to what the network can handle. Slow Start helps to mitigate the pitfalls of this unknown by implementing the following functionality.

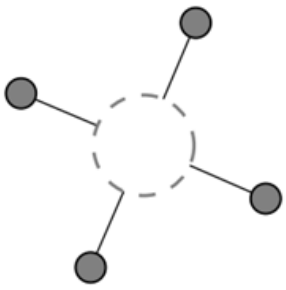
1. A sender begins transmissions to a receiver by slowly probing the network with a packet that contains its initial congestion window (cwnd).

2. The client receives the packet and replies with its maximum buffer size, also known as the receiver's advertised window (rwnd).

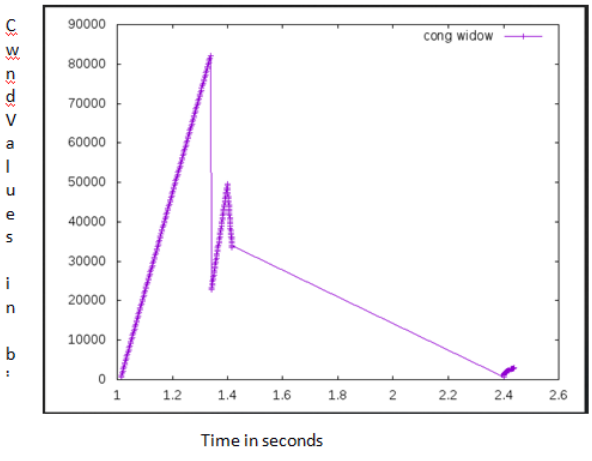
3. If the sender receives an acknowledgement (ACK) from the client, it then doubles the amount of packets to send to the client.

4. Step 3 is repeated until the sender no longer receives ACK from the receiver which means either congestion is detected, or the client's window limit has been reached.

Topology



Congestion graph



Networktopology

```
n0    n1   n2   n3
|     |    |    |
====================== CSMA channel with base IP10.1.1.0
                         Source node-n0      sink node -n1
```

Inthisprogramwehavecreated4CSMAnodesn0,n1,n2andn3withIP addresses10.1.1.1,10.1.1.2,10.1.1.3and10.1.1.4respectively.Data transmissionissimulatedbetweennodesn0andn1.Oncethecwndvalues are generated, they are exported to .dat file and congestion graph is plot usinggnuplot.

Program

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

using namespacens3;

NS_LOG_COMPONENT_DEFINE ("3rd LabProgram");

using namespacens3;

NS_LOG_COMPONENT_DEFINE ("3rd Lab Program");

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

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

  CsmaHelpercsma;
  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));

  InternetStackHelperstack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainerinterfaces=address.Assign(devices);

  uint16_t sinkPort =8080;

  AddresssinkAddress(InetSocketAddress(interfaces.GetAddress(1),
sinkPort));
```

//PacketSink Application is used on the destination node to receiveTCP
//connections and data. Creates objects in an abstract way and associates
//type-id along with the object.

```
  PacketSinkHelperpacketSinkHelper("ns3::TcpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
```

//Install sinkapp on node 1

```
  ApplicationContainersinkApps=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 NetworkApplication (Class present innetwork-
//application-helper.h)

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

//Next statement tells 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 thoseevents.

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

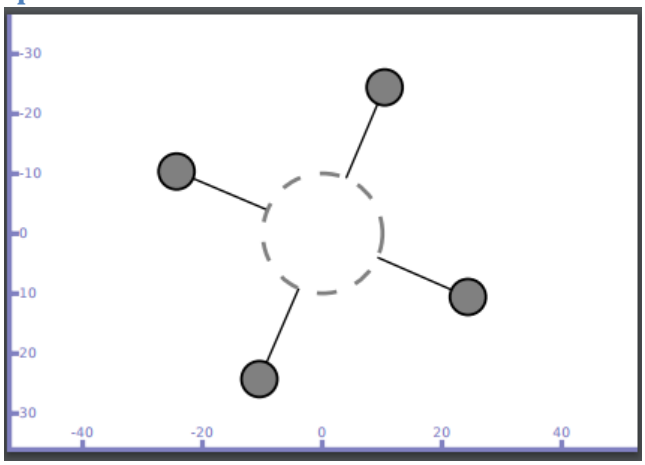
//Displays packet drop msg

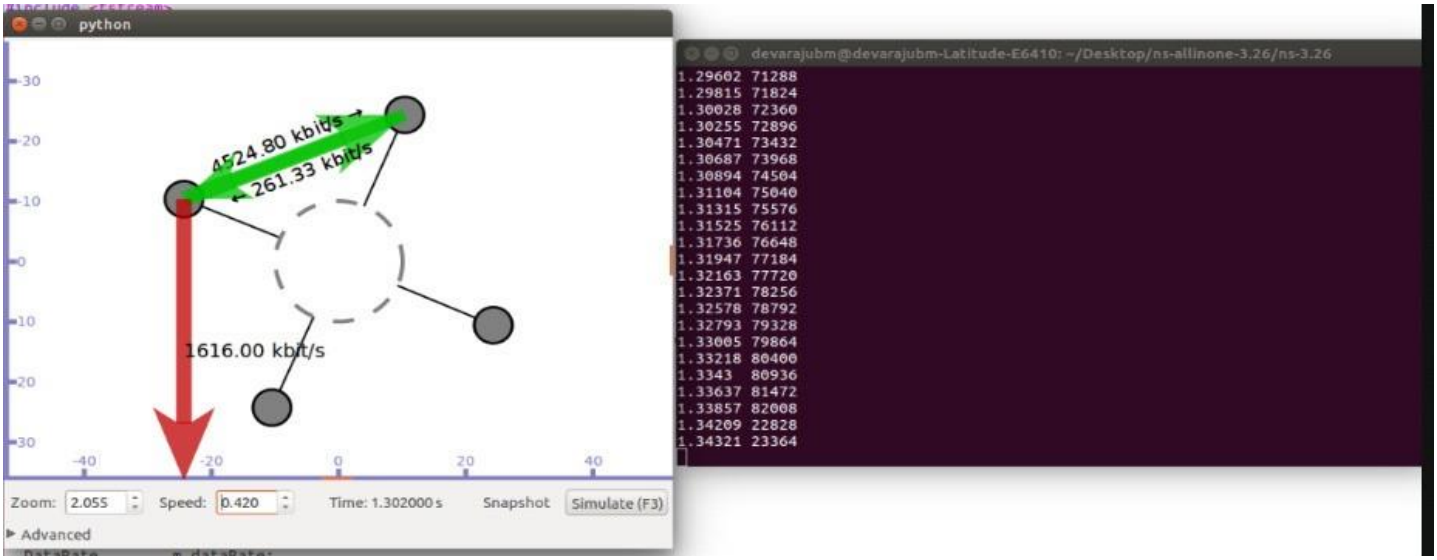
```
  devices.Get (1)->TraceConnectWithoutContext("PhyRxDrop", MakeCallback
(&RxDrop));

  AsciiTraceHelperascii;
  csma.EnableAsciiAll(ascii.CreateFileStream("3lan.tr"));
  csma.EnablePcapAll (std::string ("3lan"), true);
```

```
  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

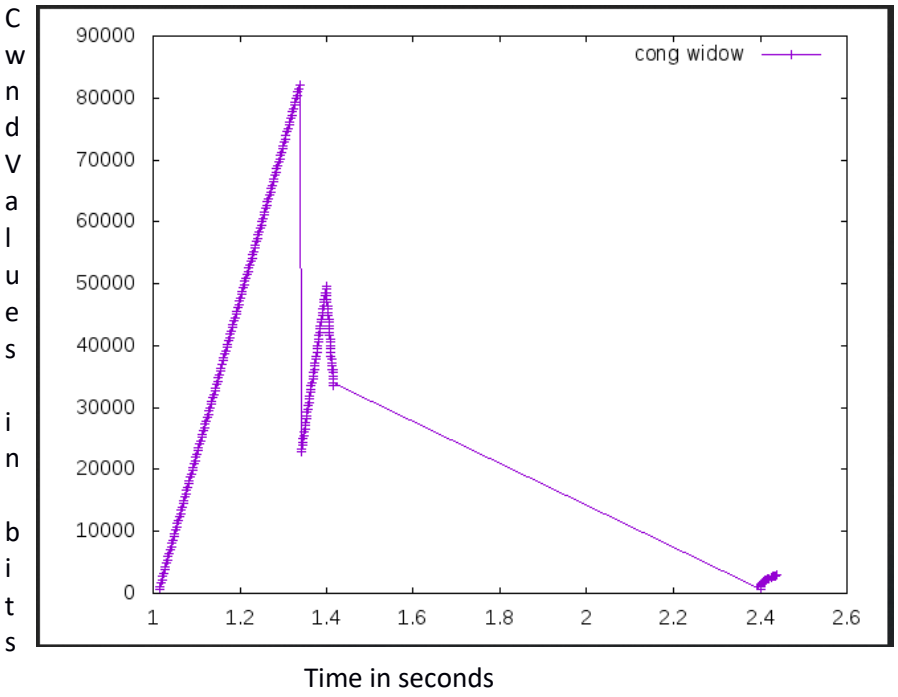
./waf --run scratch/Program3 > cwnd.dat 2>&1

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 linespointsgnuplot> exit


Cwnd Values in bits
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
cong widow
Time in seconds

PART-B

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

Overview:

The **accurate** implementations (long-hand and programmed) of the 16-bit CRC-CCITT specification, is as follows:

- Width = 16 bits
- Truncated polynomial = 0x1021
- Initial value = 0xFFFF
- Input data is NOT reflected
- Output CRC is NOT reflected
- No XOR is performed on the output CRC

Theoretical Concepts:

- Important features of a standard CRC are that it:
- Can be used to validate data
- Is reproducible by others

The first feature above is easy to realize in a closed system if corruption of data is infrequent (but substantial when it occurs). The term "closed system" refers to a situation where the CRC need not be communicated to others. A correct implementation of a 16-bit CRC will detect a change in a single bit in a message of over 8000 bytes. An erroneous CRC implementation may not be able to detect such subtle errors. If errors are usually both rare and large (affecting several bits), then a faulty 16-bit CRC implementation may still be adequate in a closed system.

The second feature above -- that the CRC is reproducible by others -- is crucial in an open system; that is, when the CRC must be communicated to others. If the integrity of data passed between two applications is to be verified using a CRC defined by a particular standard, then the implementation of that standard must produce the same result in both applications -- otherwise, valid data will be reported as corrupt.

Reproducibility may be satisfied by even a botched implementation of a standard CRC in most cases -- if everyone uses the same erroneous implementation of the standard. But this approach:

- Modifies the standard in ways that are both unofficial and undocumented.
- Creates confusion when communicating with others who have not adopted the botched implementation as the implied standard.

The CRC value for the 9-byte reference string, "123456789" is **0xE5CC**.

The need to focus on the 16-bit CRC-CCITT (polynomial 0x1021) and not CRC-16 (polynomial 0x8005),are as follows :

- Is a straightforward 16-bit CRC implementation in that it doesn't involve:
- reflection of data
- reflection of the final CRC value

- Starts with a non-zero initial value -- leading zero bits can't affect the CRC16 used by LHA, ARC, etc., because

its initial value is zero.

- It requires no additional XOR operation after everything else is done. The CRC32 implementation used by Ethernet, Pkzip, etc., requires this operation; less common 16-bit CRCs may require it as well.

The need to use a 16-bit CRC instead of a 32-bit CRC is as follows :

- Can be calculated faster than a 32-bit CRC.
- Requires less space than a 32-bit CRC for storage, display or printing.
- Is usually long enough if the data being safeguarded is fewer than several thousand

	CRC-CCITT	**CRC-16**	**CRC-32**
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101 **(IBM-CRC-16 (ANSI)**	10000010011000001000111011011 0111

bytes in length, e.g., individual records in a database.

Some CRC polynomials that are actually used

	CRC-8	**CRC-CCITT**	**CRC-32**
CRC	x^8+x^2+x+1	$x^{16}+x^{12}+x^5+1$	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
Uses	Used in:802.16 (alongwith error*correction*).	Used in:HDLC, SDLC,PPP	Used in:Ethernet, PPP

Theory

It does error checking via polynomial division. In general, a bit string $b_{n-1}b_{n-2}b_{n-3}\ldots b_2b_1b_0$

Asb$_{n-1}X^{n-1} + b_{n-2}X^{n-2} + b_{n-3}X^{n-3} + \ldots b_2X^2 + b_1X^1 + b_0$

Ex: -

10010101110 As

$X^{10} + X^7 + X^5 + X^3 + X^2 + X^1$

All computations are done in modulo 2

Algorithm:-

1. Given a bit string, append 0S to the end of it (the number of 0s is the same as the degree of the generator polynomial) let B(x) be the polynomial corresponding to B.

2. Divide B(x) by some agreed on polynomial G(x) (generator polynomial) and determine the remainder R(x). This division is to be done using Modulo 2 Division.

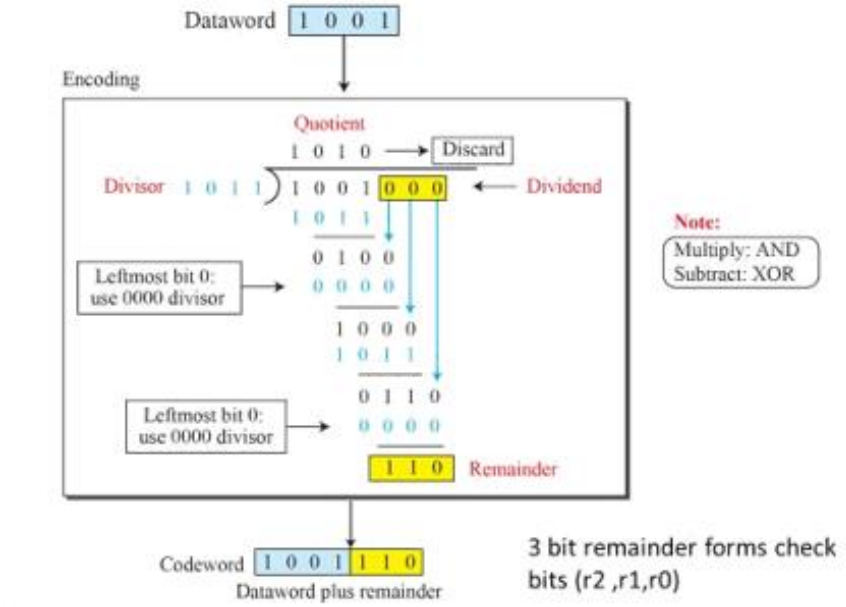
3. Define T(x) = B(x) –R(x)

(T(x)/G(x) => remainder 0)

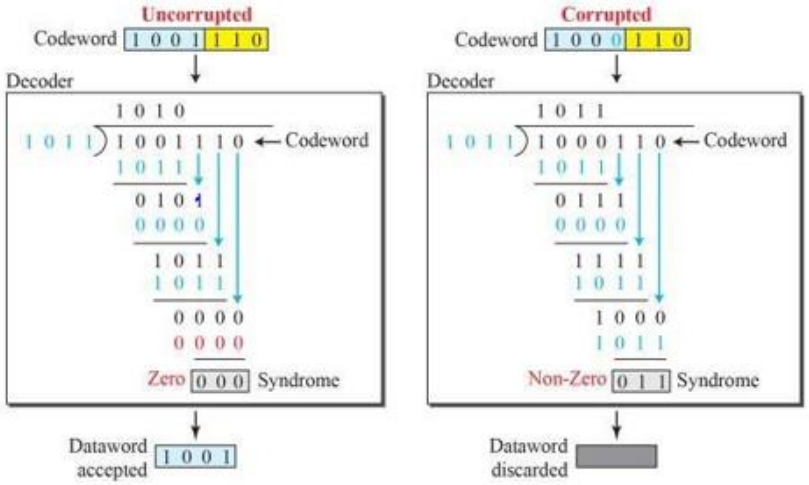
4. Transmit T, the bit string corresponding to T(x).

5. Let T' represent the bit stream the receiver gets and T'(x) the associated polynomial. The receiver divides T1(x) by G(x). If there is a 0 remainder, the receiver concludes T = T' and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission.

Division in CRC encoder



Division in CRC decoder



```
import java.io.*;
class crc
{
    public static void main(String args[]) throws IOException
    {
        BufferedReaderbr=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=newint[tot_length];
        rem=newint[tot_length];
        crc=newint[tot_length];
    /*------------------CRCGENERATION ----------------------- */
        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(inti=0;i<div.length;i++)            //append dividend andramainder
        {
            crc[i]=(div[i]^rem[i]);
        }
        System.out.println();
        System.out.println("CRC code :");
        for(inti=0;i<crc.length;i++)
            System.out.print(crc[i]);

    /*-------------------ERRORDETECTION -------------------- */
        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("NoError");
        }

        System.out.println("THANKYOU ......)");
    }

    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
THANKYOU ..... )
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
THANKYOU .....)

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

Theory

Routing algorithm is a part of network layer software which is responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagram internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the subnet uses virtual circuits internally, routing decisions are made only when a new established route is being set up. The latter case is sometimes called session routing, because a rout remains in force for an entire user session (e.g., login session at a terminal or a file).

Routing algorithms can be grouped into two major classes: adaptive and nonadaptive. Nonadaptive algorithms do not base their routing decisions on measurement or estimates of current traffic and topology. Instead, the choice of route to use to get from I to J (for all I and J) is compute in advance, offline, and downloaded to the routers when the network ids booted. This procedure is sometime called static routing.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ΔT sec, when the load changes, or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

Two algorithms in particular, distance vector routing and link state routing are the most popular. Distance vector routing algorithms operate by having each router maintain a table (i.e., vector) giving the best known distance to each destination and which line to get there. These tables are updated by exchanging information with the neighbors.

The distance vector routing algorithm is sometimes called by other names, including the distributed Bellman-Ford routing algorithm and the Ford-Fulkerson algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). It was the original ARPANET routing algorithm and was also used in the Internet under the RIP and in early versions of DECnet and Novell's IPX. AppleTalk and Cisco routers use improved distance vector protocols.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in subnet. This entry contains two parts: the preferred out going line to use for that destination, and an estimate of the time or distance to that destination. The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.

The router is assumed to know the "distance" to each of its neighbor. If the metric is hops, the distance is just one hop. If the metric is queue length, the router simply examines each queue. If the metric is delay, the router can measure it directly with special ECHO packets hat the receiver just time stamps and sends back as fast as possible.

The Count to Infinity Problem.

Distance vector routing algorithm reacts rapidly to good news, but leisurely to bad news. Consider a router whose best route to destination X is large. If on the next exchange neighbor A suddenly reports a short delay to X, the router just switches over to using the line to A to send traffic to X. In one vector exchange, the good news is processed.

To see how fast good news propagates, consider the five node (linear) subnet of following figure, where the delay metric is the number of hops. Suppose A is down initially and all the other outers know this. In other words, they have all recorded the delay to A as infinity.

A	B	C	D	E		A	B	C	D	E	
∞	∞	∞	∞	∞	Initially	1	2	3	4	Initially
1	∞	∞	∞	∞	After 1 exchange	3	2	3	4	After 1 exchange
1	2	∞	∞	∞	After 2 exchange	3	3	3	4	After 2 exchange
1	2	3	∞	∞	After 3 exchange	5	3	5	4	After 3 exchange
1	2	3	4	∞	After 4 exchange	5	6	5	6	After 4 exchange
						7	6	7	6	After 5 exchange
						7	8	7	8	After 6 exchange
							:			
						∞	∞	∞	∞	

DISTANCE VECTOR ALGORITHM:

One of the most popular & dynamic routing algorithms, the distance vector routing algorithms operate by using the concept of each router having to maintain a table(ie., a vector), that lets the router know the best or shortest distance to each destination and the next hop to get to there. Also know as Bellman Ford(1957) and Ford Fulkerson (1962). It was the original ARPANET algorithm.

Algorithm Overview:

Each router maintains a table containing entries for and indexed by each other router in the subnet. Table contains two parts:

1. A preferred outing line.

2. Estimate of time or distance to that destination.

The metric used here is the transmission delay to each destination. This metric maybe number of hops, queue length, etc.

The router is assumed to know the distance metric to each of its neighbors. Across the network the delay is calculated by sending echo packets to each of neighbors.

At each node, x:

```
1   Initialization:
2      for all destinations y in N:
3         Dx(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
4      for each neighbor w
5         Dw(y) = ? for all destinations y in N
6      for each neighbor w
7         send distance vector Dx = [Dx(y): y in N] to w
8
9   loop
10     wait (until I see a link cost change to some neighbor w or
11           until I receive a distance vector from some neighbor w)
12
13     for each y in N:
14        Dx(y) = minv{c(x,v) + Dv(y)}
15
16     if Dx(y) changed for any destination y
17        send distance vector Dx = [Dx(y): y in N] to all neighbors
18
19  forever
```

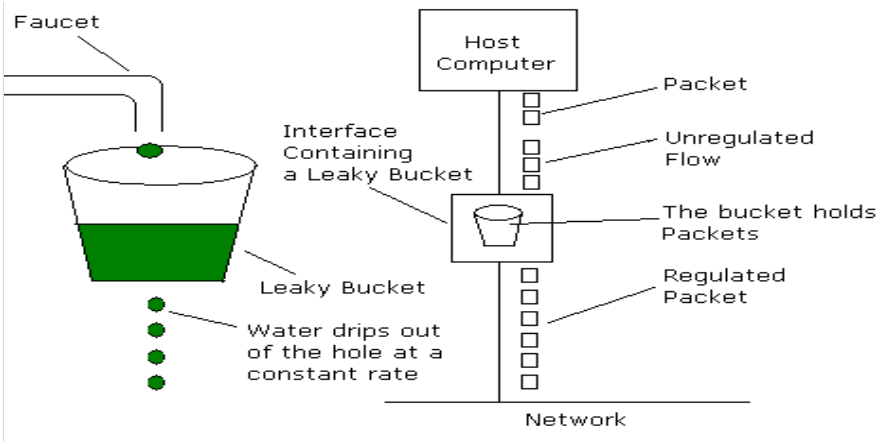
```
packagebellmanford;
importjava.util.*;
public class Bellmanford{ static
int n,dest;
static double[] prevDistanceVector,distanceVector;
static double[][] adjacencyMatrix;
public static void main(String[] args) {
 // TODO code application logic here
Scanner scanner = new Scanner(System.in);
System.out.println("Enter 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 destinationvertex");
      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 NoLink)
0 3 2 5 9999
3 0 99 1 499
2 99 0 2 991
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 is3.0
Distance from 2 is4.0
Distance from 3 is1.0
Distance from 4 is3.0
Distance from 5 is2.0
```

3. Write a program for congestion control using leaky bucketalgorithm.



LEAKY BUCKETALGORITHM

The leaky-bucket implementation is used to control the rate at which traffic is sent to the network. A leaky bucket provides a mechanism by which bursty traffic can be shaped to present a steady stream of traffic to the network, as opposed to traffic with erratic bursts of low-volume and high-volume flows.

Traffic analogy An appropriate analogy for the leaky bucket is a scenario in which four lanes of automobile traffic converge into a single lane. A regulated admission interval into the single lane of traffic flow helps the traffic move. The benefit of this approach is that traffic flow into the major arteries (the network) is predictable and controlled. The major liability is that when the volume of traffic is vastly greater than the bucket size, in conjunction with the drainage-time interval, traffic backs up in the bucket beyond bucket capacity and is discarded.

The Leaky-bucket algorithm

The algorithm can be conceptually understood as follows:

- Arriving packets (network layer PDUs) are placed in a bucket with a hole in the bottom.
- The bucket can queue at most *b* bytes. If a packet arrives when the bucket is full, the packet is discarded.
- Packets drain through the hole in the bucket, into the network, at a constant rate of *r* bytes per second, thus smoothing traffic bursts.

The size *b* of the bucket is limited by the available memory of the system.

Sometimes the leaky bucket and token algorithms are lumped together under the same name.

```
package lb;
import java.util.*;
public class Lb{
	public static void main(String[] args) {
		System.out.println("enter the number of timeintervals");
		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		int t[]=new int[n];
		System.out.println("enter the timeintervals");
		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
```