LAB MANUAL OF
COMPUTER NETWORKS LAB
ETCS 354

Maharaja Agrasen Institute of Technology, PSP area,
Sector – 22, Rohini, New Delhi – 110085
(Affiliated to Guru Gobind Singh Indraprastha University,
New Delhi)

INDEX OF THE CONTENTS

1. Introduction to the Lab

Lab Objective

The objective of this lab is making student know how to design and analyze computer networks. To become acquainted with network programming and some of the important GUI based computer networking tools.

Course Outcomes

At the end of the course, a student will be able to:

C354.1 Simulate the Discrete Event Systems using various network tools.

C354.2 Design solutions for real life situations in form of communication networks.

C354.3 Evaluate all the possibilities of wired as well as wireless networks (Zigbee, Wi-Max,Wi-PAN, IEEE 802.11 a,b,c,g) by using routers, switches and various topologies.

C354.4 Analyze and evaluate the network results using different open source logger tools (Wireshark, TCPDump and NS3 NetAnim package).

C354.5 Implement Sliding window and congestion avoidance protocols.

C354.6 Explore the possible research opportunities and difficulties within the course scope.

2. LAB REQUIREMENTS

Hardware Detail

Intel i3/C2D Processor/2 GB RAM/500GB HDD/MB/Lan Card/ Key Board/ Mouse/CD Drive/15" Color Monitor/ UPS	24 Nos
LaserJet Printer	1 No

Software Detail

Linux, Network Simulator v 2 & 3 and Wireshark.

3. LIST OF EXPERIMENTS

(As prescribed by G.G.S.I.P.U)

1. Introduction to Computer Network laboratory, introduction to Discrete Event Simulation and Discrete Event Simulation Tools
2. Introduction to NS3 and its comparison with NS2.
3. Install NS3 on Linux.
4. Using Free Open Source Software tools *ns3,* design and implement two nodes topology.
5. Using Free Open Source Software tools *ns3,* design and implement three nodes topology considering one node as a central node.
6. Using Free Open Source Software tools *ns3,* design and Implement star topology using StarHelperClass.
7. Using Free Open Source Software tools *ns3,* design and implement a bus topology using CSMA.
8. Using Free Open Source Software tools *ns3,* design and implement hybrid topology connecting multiple routers and nodes.
9. Install and configure NetAnim.

4. LIST OF EXPERIMENTS (Beyond the syllabus)

1. Using Free Open Source Software tools *ns3,* design and implement FTP using TCP bulk transfer.
2. Analyze network traces using Wireshark software

5. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Paper Code

Font should be (Size 20", italics bold, Times New Roman)

Faculty name

Student name

Roll No.:

Semester:

Font should be (12", Times Roman)

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18", Times Roman)

Index

Exp. no	Experiment Name	Date of performance	Date of checking	Marks	Signature

6. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

i. Internal Practical Exam
ii. External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the respective faculty of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 10 marks per practical is as follows:

	Rubrics for : Laboratory (General)			
Sr No.	Experiment Component (LAC)	Max. Marks	Grading Rubrics	
			2 marks	1 mark
1	Practical Performance	2	Completeness of practical, exhibits proficiency in using different types of inputs.	Incomplete practical, unformatted, lacks comments, Demonstrates no proficiency.
2	Output and Validation	2	Output is free of errors and output is obtained. Demonstrates excellent understanding of the concepts relevant to the experiment.	Output contains few logical errors and/or no output is obtained. Demonstrates partial understanding of the concepts relevant to the experiment.
3	Attendance and Viva Questions Answered	4	1. Four marks for answering more than 75% questions. 2. Three marks for answering more than 60% questions. 3. Two mark for answering more than 50% questions. 4. One mark for answering less than 50% questions.	
4	Timely Submission of Lab Record	2	On time submission	Late submission

Each experiment will be evaluated out of 10 marks. At the end of the semester average of 8 best performed practical will be considered as marks out of 40.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student: 20
2. Viva Voice: 15
3. Experiment performance: 15
4. File submitted: 10

NOTE:

- Internal marks + External marks = Total marks given to the students
  (40 marks) (60 marks) (100 marks)

- Experiments given to perform can be from any section of the lab.

Introduction to
Computer Networks Lab

The lab of Computer Networks gives in depth view of the computer networks working in real time and simulation of various topologies using NS3 tool.

ns-3 has been developed to provide an open, extensible network simulation platform, for networking research and education. In brief, *ns-3* provides models of how packet data networks work and perform & provides a simulation engine for users to conduct simulation experiments. Some of the reasons to use *ns-3* include to perform studies that are more difficult or not possible to perform with real systems, to study system behavior in a highly controlled, reproducible environment, and to learn about how networks work. Users will note that the available model set in *ns-3* focuses on modeling how Internet protocols and networks work, but *ns-3* is not limited to Internet systems; several users are using *ns-3* to model non-Internet-based systems.

Many simulation tools exist for network simulation studies. Below are a few distinguishing features of *ns-3* in contrast to other tools.

- *ns-3* is designed as a set of libraries that can be combined together and also with other external software libraries. While some simulation platforms provide users with a single, integrated graphical user interface environment in which all tasks are carried out, *ns-3* is more modular in this regard. Several external animators and data analysis and visualization tools can be used with *ns-3*. However, users should expect to work at the command line and with C++ and/or Python software development tools.
- *ns-3* is primarily used on Linux systems, although support exists for FreeBSD, Cygwin (for Windows), and native Windows Visual Studio support is in the process of being developed.

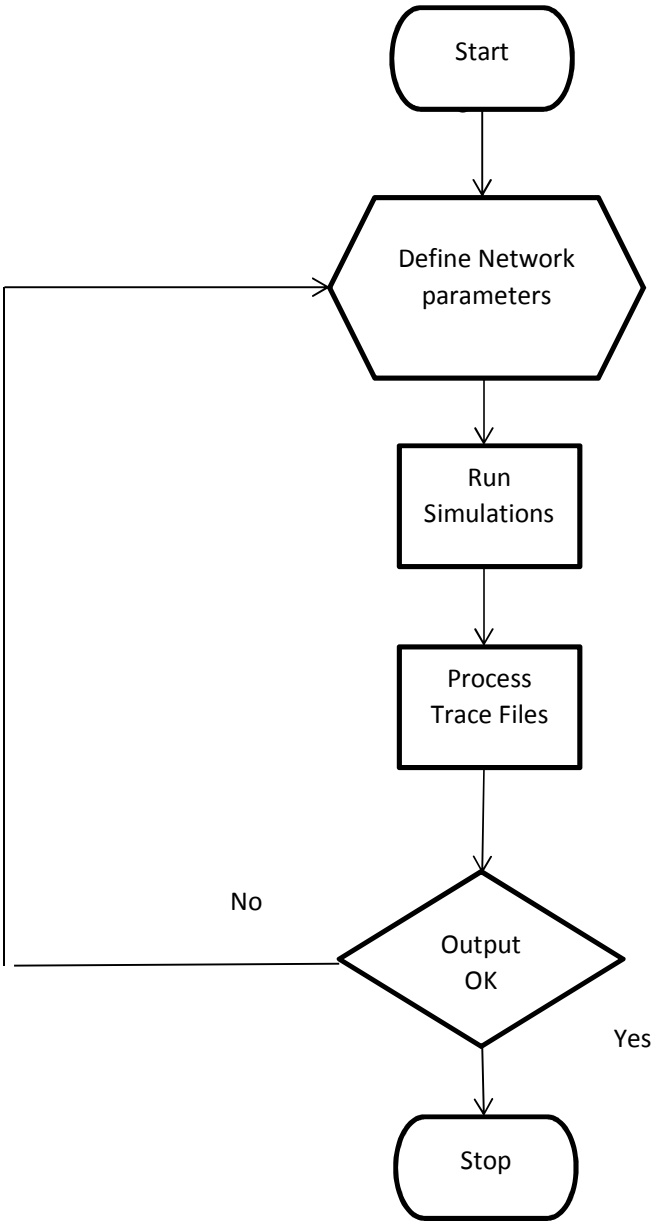


Fig 1: Simulation Process

INSTRUCTIONS FOR EACH LAB EXPERIMENT

EXPERIMENT 1

Aim: Introduction to Computer Network laboratory, introduction to Discrete Event Simulation and Discrete Event Simulation Tools

System:

A collection of entities that act and interact together toward the accomplishment of some logical end.

Discrete system:

State variables change instantaneously at separated point in time, e.g., a bank, since state variables - number of customers, change only when a customer arrives or when a customer finishes being served and departs

Continuous system:

State variable change continuously with respect to time, e.g., airplane moving through the air, since state variables - position and velocity change continuously with respect to time

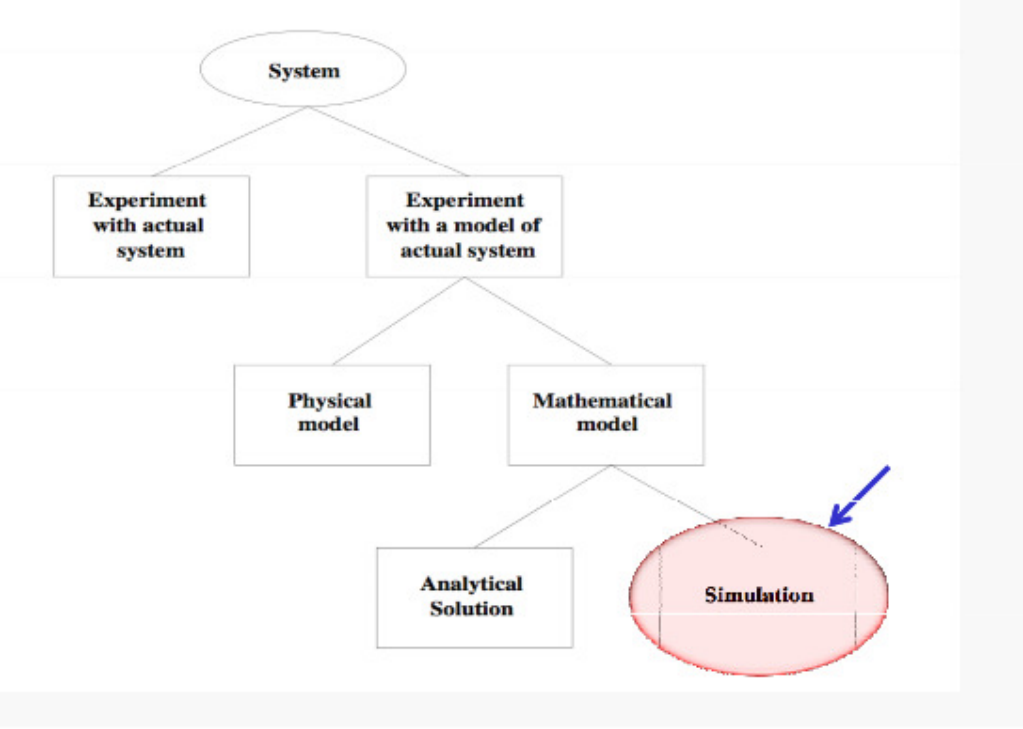


Fig 2: System Implementation and Study

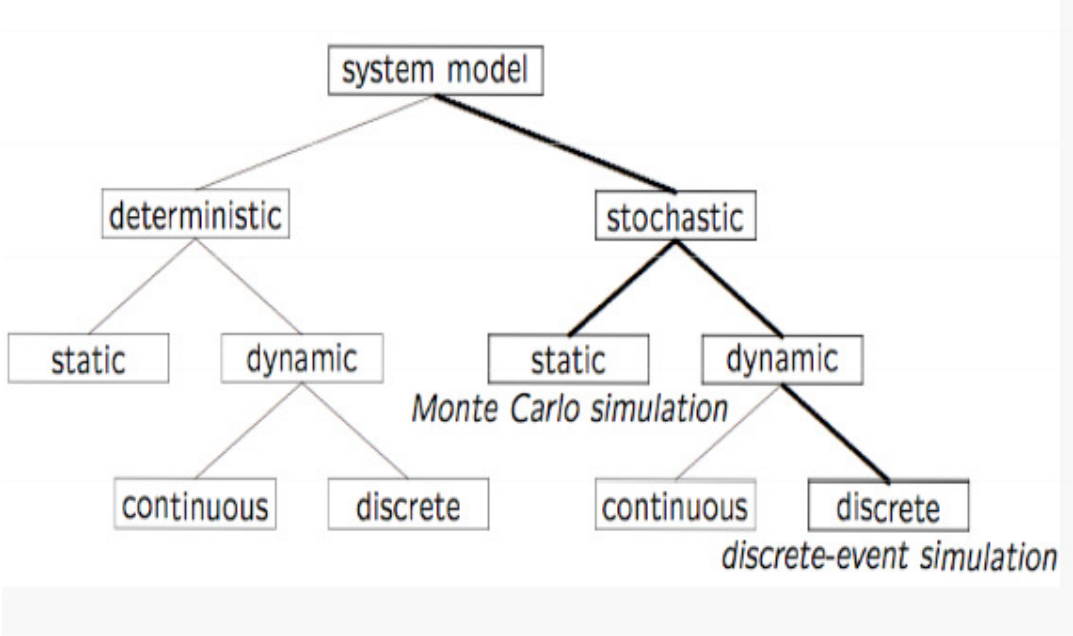


Fig 3: Model Taxonomy

Why Simulation?

• Many systems are highly complex, precluding the possibility of analytical solution

•The analytical solutions are extraordinarily complex, requiring vast computing resources

•Thus, such systems should be studied by means of simulation numerically exercising the model for inputs in question to see how they affect the output measures of performance

> *"Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of a system."*

Discrete-Event Simulation (DES)

A discrete-event simulation
Models a system whose state may change only at discrete point models a system whose state may change only at discrete point in time

System:
is composed of objects called entities that have certain properties called attributes.

State:
a collection of attributes or state variables that represent the entities of the system.

Event:
an instantaneous occurrence in time that may alter the state of the system

- **Discrete-event simulation is stochastic, dynamic, and discrete**
- **Stochastic = Probabilistic**
  - Inter-arrival times and service times are random variables
  - Have cumulative distribution functions
- **Discrete = Instantaneous events are separated by intervals of time**
  - The state variables change instantaneously at separate points in time
    - The system can change at only a countable number of points in time.
  - These points in time are the ones at which an event occurs.
- **Dynamic = Changes over time**
  - Simulation clock
    - Keep track of the current value of simulated time as the simulation proceeds
  - A mechanism to advance simulated time from one value to another
    - Next-event time advance

VIVA Questions

Q1. What is Discrete Event Simulation?

Q2. What is the importance and limitations of simulation?

Q3. What is the relationship between state, system and an event?

Q4. What is the difference between deterministic and stochastic model?

Q5. What is proof of correctness? Why mathematical models are more reliable than simulations?

EXPERIMENT 2

Aim: Introduction to NS3 and its comparison with NS2.

Description:
In this lab, we will be using the Network Simulator, NS3, available from www.nsnam.org. NS3 is a powerful program, however we will only be looking at some basic features. NS3 simulations are built in C++.

Compare NS2 and NS3 on the basis of the following parameters:

1. Programming Languages
2. Memory Management
3. Packets
4. Performance
5. Simulation Output

Some Important Points about NS3:

1. NS3 is not backward compatible with NS2; it's built from the scratch to replace NS2.
2. NS3 is written in C++, Python Programming Language can be optionally used as an interface.
3. NS3 is trying to solve problems present in NS2.
4. There is very limited number of contributed codes made withNS3 compared to NS2
5. In NS2, bi-language system makes debugging complex (C++/Tcl), but for NS3 only knowledge of C++ is enough (single-language architecture is more robust in the long term).
6. NS3 has an emulation mode, which allows for the integration with real networks.

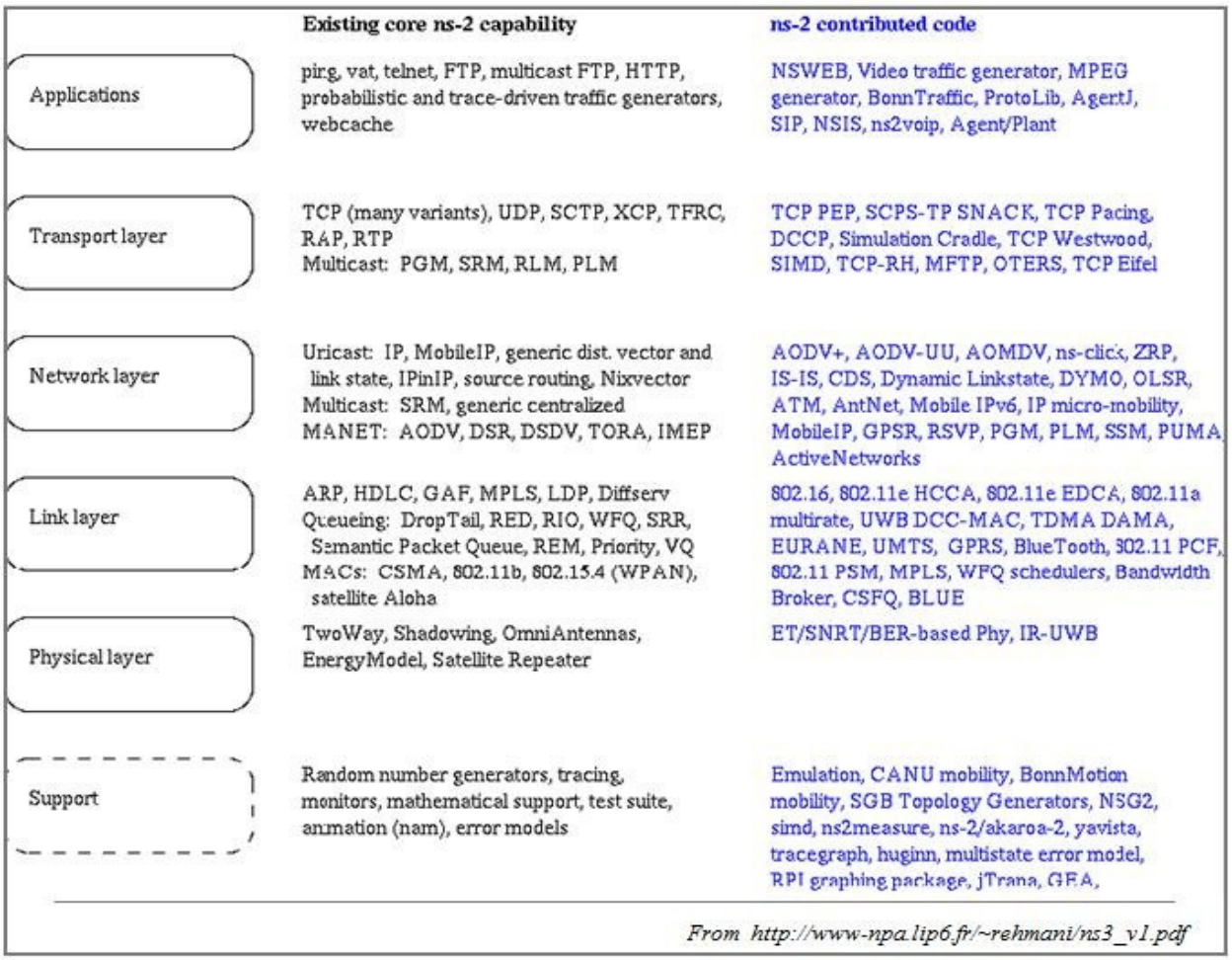


Fig 4: NS2contributed code

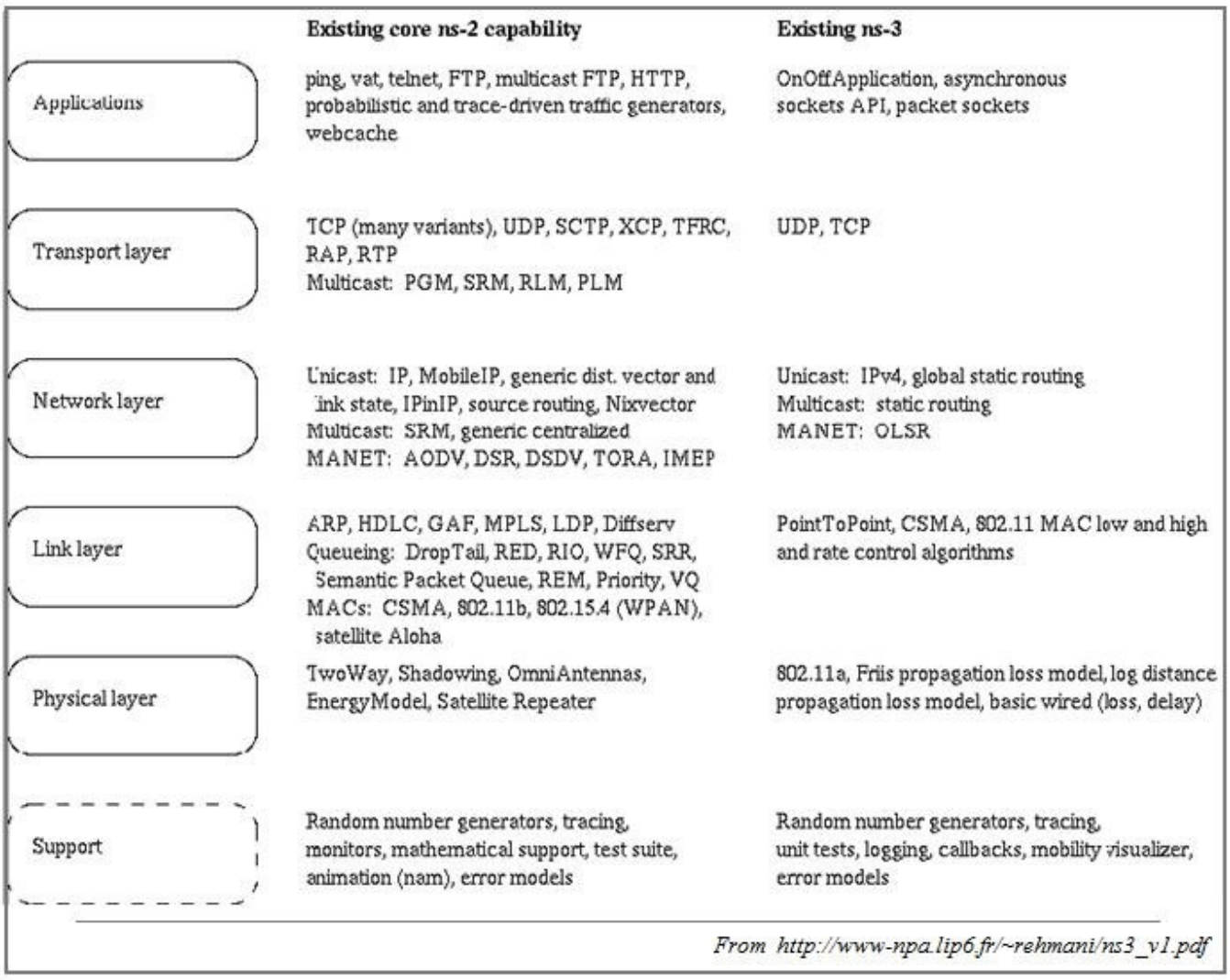


Fig 5: NS2 and NS3 existing core capabilities

VIVA Questions

Q1. How the scripting languages are different from Programming languages?

Q2. State the importance of NS3 over NS2?

Q3. Why it is debated to work on NS2 for research works instead of NS3 for research works?

Q4. What are the parameters on which the NS2 can be differentiated with the NS3?

EXPERIMENT 3

Aim: Install NS3 on Linux

Description:

Following are the basic steps which must be followed for installing NS3

1. Install prerequisite packages
2. Download ns3 codes
3. Build ns3
4. Validate ns3

Prerequisite packages for Linux are as follows:

1. Minimal requirements for Python: gcc g++ python
2. Debugging and GNU Scientific Library (GSL) support: gdbpython-dev,valgrind gsl-bin,libgsl0-dev,libgsl0ldbl, Network Simulation Cradle (nsc): flex,bison
3. Reading pcap packet traces: tcpdump
4. Database support for statistics framework: sqlite,sqlite3
5. XML-based version of the config store: libxml2
6. A GTK-based configuration system: libgtk2.0-0
7. Experimental with virtual machines and ns-3: vtun,lxc

Detail steps are as follows:

1. $sudo apt-get update / dnf update
2. $sudo apt-get upgrade / dnf upgrade
3. Once ubuntu/fedora is installed run following command opening the terminal(ctrl+alt+T) window.
4. To install prerequisites dependency packages- Type the following command in terminal window.

5. $sudo apt-get/ dnf install gcc g++ python python-dev mercurial bzr gdb valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick texlive texlive-

latex-extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlive-latex-extra texlive-extra-utils texlive-generic-recommended texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev python-pygccxml

6. After downloading NS3 on the drive, extract all the files in the NS3 folder, which you have created.

7. Then you can find build.py along with other files in NS3 folder.
   Then to build the examples in ns-3 run :
   $./build.py --enable-examples –enable-tests
   If the build is successful then it will give output
   "Build finished successfully".
8. Now run the following command on the terminal window to configure with waf (build tool)
   $./waf -d debug --enable-examples --enable-tests configure
   To build with waf (optional)
   $./waf
9. To test everything all right run the following command on the terminal window,
   $./test.py
   If the tests are ok the installation is done
10. Now after installing ns3 and testing it run some programs first to be ns3 user:
    make sure you are in directory where waf script is available then run

VIVA Questions

Q1 What protocols does ns support?

Q2. How should one can start doing something (like implementing a new protocol or trying an experiment)?

Q3. What is waf and its importance in the simulation process?

Q4. How we can test that NS3 has been installed properly?

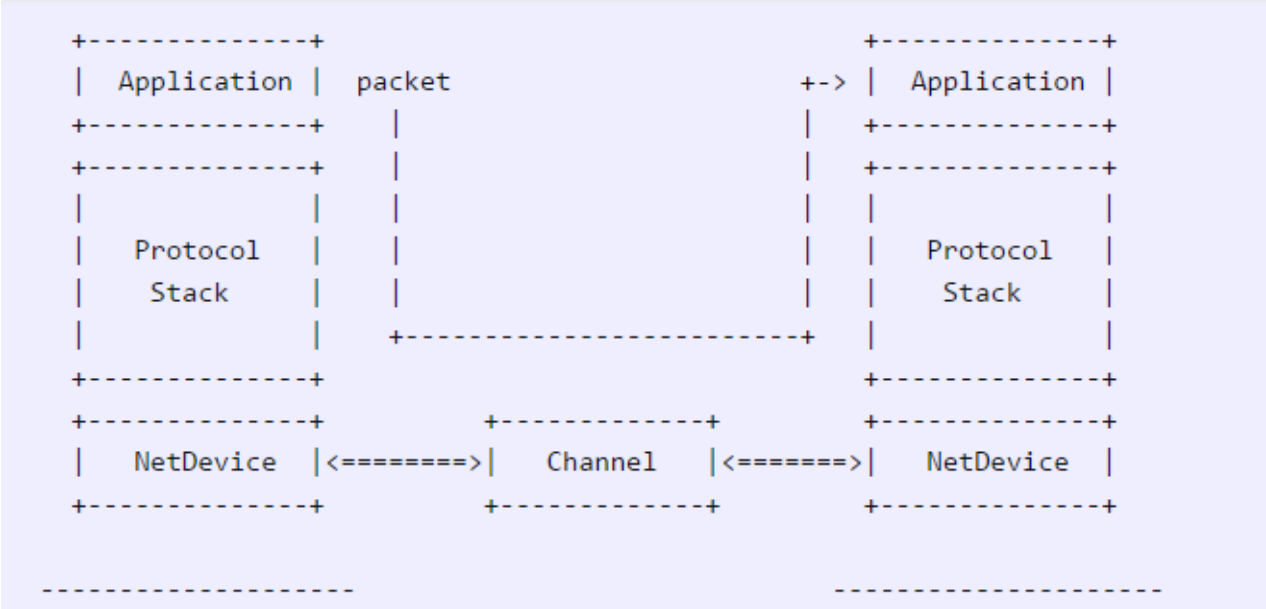
Q5. Why prerequisite packages are needed before the installation of NS3?

EXPERIMENT 4

AIM: Using Free Open Source Software tools ns3, design and implement two nodes topology.

Fig 6: Two Node Topology

Description:



Node

Because in any network simulation, we will need nodes. So ns-3 comes with `NodeContainer` that you can use to manage all the nodes (Add, Create, Iterate, etc.).

```
// Create two nodes to hold.
```

```
NodeContainer nodes;
nodes.Create (2);
```

Channel and NetDevice

In the real world, they correspond to network cables (or wireless media) and peripheral cards (NIC). Typically, these two things are intimately tied together. In the first example, we are using `PointToPointHelper` that wraps the Channel and NetDevice.

```
// Channel: PointToPoint, a direct link with `DataRate` and `Delay` specified.
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

Then we need to install the devices. The internal of `Install` is actually more complicated, but for now, let's just skip the magic behind the scene.

```
// NetDevice: installed onto the channel
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Protocols

Internet and IPv4. Since Internet is the current largest network to study, ns-3 has a particular focus on it. The `InternetStackHelper` will install an Internet Stack (TCP, UDP, IP, etc.) on each of the nodes in the node container.

```
// Protocol Stack: Internet Stack
InternetStackHelper stack;
stack.Install (nodes);
```

To assign IP addresses, use a helper and set the base. The low-level ns-3 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.

```
// Since IP Address assignment is so common, the helper does the dirty work!
// You only need to set the base.
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

// Assign the address to devices we created above
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Applications

Every application needs to have `Start` and `Stop` function so that the simulator knows how to schedule it. Other functions are application-specific. We will use `UdpEchoServer` and `UdpEchoClient` for now

```
// Application layer: UDP Echo Server and Client
// 1, Server:
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

// 2, Client:
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
```

```
clientApps.Stop (Seconds (10.0));
```

Simulation

```
// Start Simulation
Simulator::Run ();
Simulator::Destroy ();
        return 0;
```

EXPECTED OUTPUT

```
[root@mta-192-180-1-110 ns-3.24.1]# ./waf --run udpro8
Waf: Entering directory `/home/mait/Downloads/ns-allinone-3.24.1/ns-3.24.1/build
'
Waf: Leaving directory `/home/mait/Downloads/ns-allinone-3.24.1/ns-3.24.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.566s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 9
[root@mta-192-180-1-110 ns-3.24.1]#
```

VIVA Questions

Q1 which probable protocols can come into play when 2 nodes are connected?

Q2. What is the difference between UDP and TCP?

Q3. What is a Point to Point Connection? What are its characteristics?

Q4. What will happen if the client starts first then the server?

Q5. What is the difference between IPv4 and IPv6?

EXPERIMENT 5

AIM: Using Free Open Source Software tools ns3, design and implement three nodes topology considering one node as a central node.

Description:

Comment: Similar to Two Node program student can perform an experiment to simulate a 3 node topology considering one node as Server (Central Node) and two nodes as client nodes.

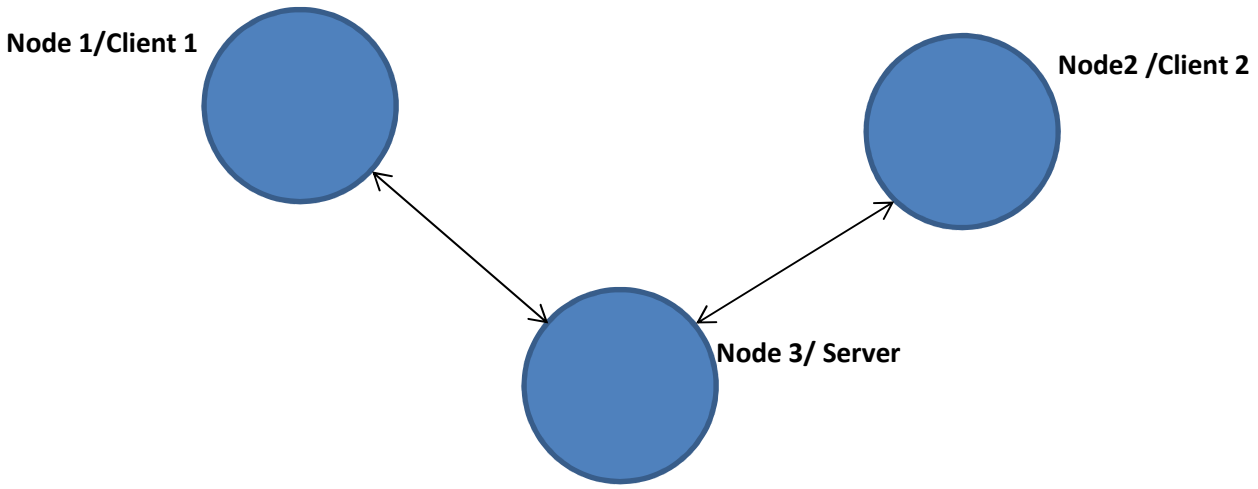


Fig 7: Three Node Topology (with two clients and one server)

Step 1: Create three nodes.

Step 2: Set the attributes of one Point to Point link.

Step 3: Declare one NetDevice Container.

Step 4: Install the Point to Point link on Devices (Nodes).

Step 5: Install the Internet stack on nodes.

Step 6: Install the Internet Stack on nodes using node containers.

Step 7: Assign the IP addresses using a helper and set the base.

Step 8: Set the applications using UDP Echo Server and UDP Echo Client applications.

Step 9: Install the server application on one node and set the Port No. for accessing the services.

Step 10: Install the client applications on two nodes; set the communication attributes in terms of ***Packet Size*** and ***Interval*** while client is communicating with server with the defined port no.

Step 11: As for simulations each application is required to Start and Stop after a time interval; Start and Stop the Ser4ver and the Clients.

Step 12: Finally run the simulation and destroy in order to release the resources.

EXPECTED OUTPUT

```
[student@localhost ns-3.27]$ ./waf --run first
Waf: Entering directory `/home/student/ns-allinone-3.27/ns-3.27/build'
[2267/2667] Compiling examples/tutorial/first.cc
[2518/2667] Linking build/examples/tutorial/ns3.27-first-debug
Waf: Leaving directory `/home/student/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.942s)
At time 2s client sent 1024 bytes to 10.1.1.1 port 9
At time 2s client sent 100 bytes to 10.1.1.3 port 9
At time 2.00521s server received 100 bytes from 10.1.1.4 port 49153
At time 2.00521s server sent 100 bytes to 10.1.1.4 port 49153
At time 2.00669s server received 1024 bytes from 10.1.1.2 port 49153
At time 2.00669s server sent 1024 bytes to 10.1.1.2 port 49153
At time 2.01042s client received 100 bytes from 10.1.1.3 port 9
At time 3s client sent 100 bytes to 10.1.1.3 port 9
At time 3.00521s server received 100 bytes from 10.1.1.4 port 49153
At time 3.00521s server sent 100 bytes to 10.1.1.4 port 49153
At time 3.01042s client received 100 bytes from 10.1.1.3 port 9
At time 4s client sent 100 bytes to 10.1.1.3 port 9
At time 4.00521s server received 100 bytes from 10.1.1.4 port 49153
At time 4.00521s server sent 100 bytes to 10.1.1.4 port 49153
At time 4.01042s client received 100 bytes from 10.1.1.3 port 9
At time 5s client sent 100 bytes to 10.1.1.3 port 9
At time 6s client sent 100 bytes to 10.1.1.3 port 9
At time 7s client sent 100 bytes to 10.1.1.3 port 9
At time 8s client sent 100 bytes to 10.1.1.3 port 9
At time 9s client sent 100 bytes to 10.1.1.3 port 9
[student@localhost ns-3.27]$
```

VIVA Questions

Q1. What if all the 3 nodes are the part of the same network and each interface is assigned the IP address of the same range?

Q2. What is the port number? What is its significance?

Q3. What is socket address? What is the role of IP address in it?

Q4. How a NS3 program is simulated? Explain various stages.

Q5. What is the difference between IP address and physical address?

EXPERIMENT 6

Aim: Using Free Open Source Software tools ns3, design and Implement star topology using StarHelperClass.

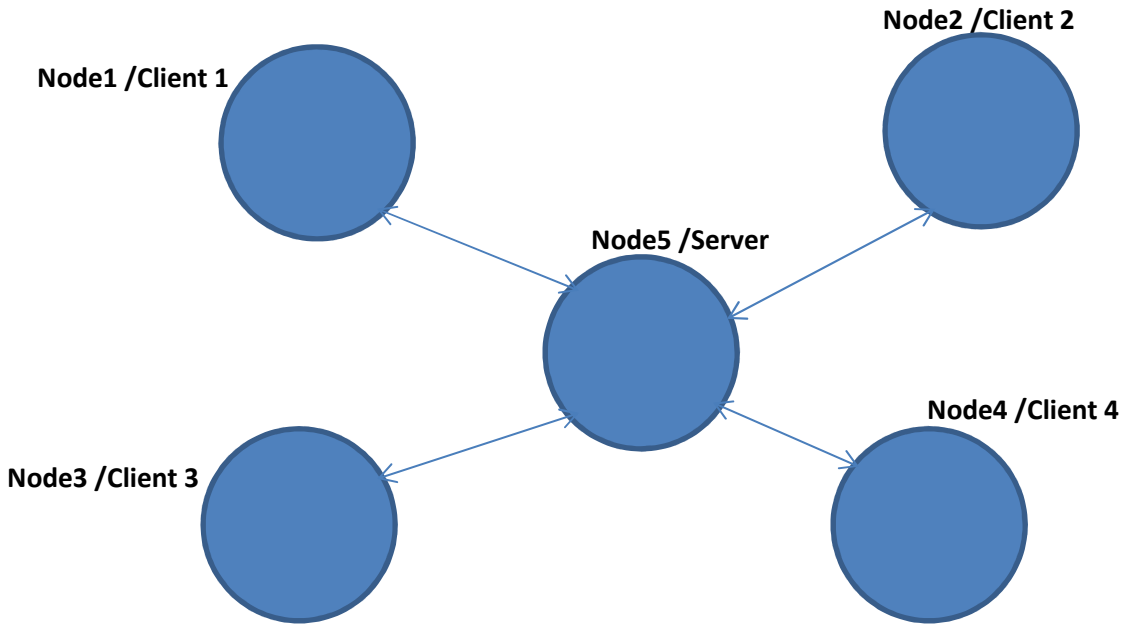


Fig 8: Star Topology (using Star Helper Class)

Theory:

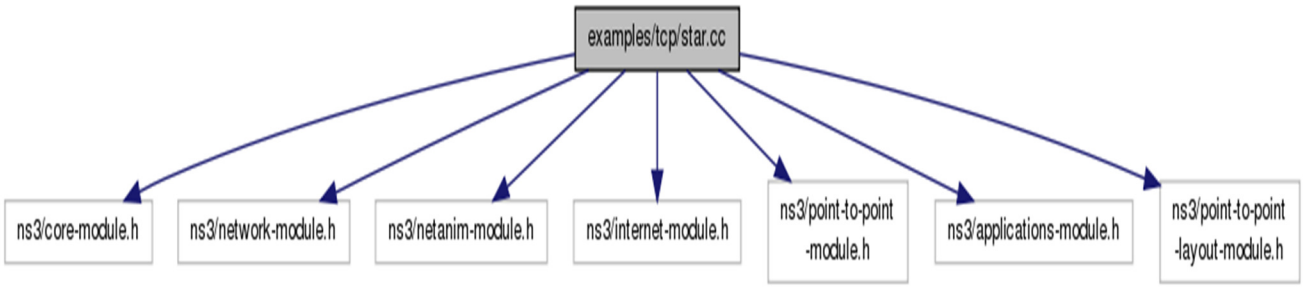


Fig 9: Star Helper Class

ns3::PointToPointStarHelper Class Reference

- A helper to make it easier to create a star topology with PointToPoint links.
- Create a **PointToPointStarHelper** in order to easily create star topologies using p2p links.
- **Parameters**
  - **numSpokes** the number of links attached to the hub node, creating a total of numSpokes + 1 nodes
  - **p2pHelper** the link helper for p2p links, used to link nodes together

Function calling details of the class:

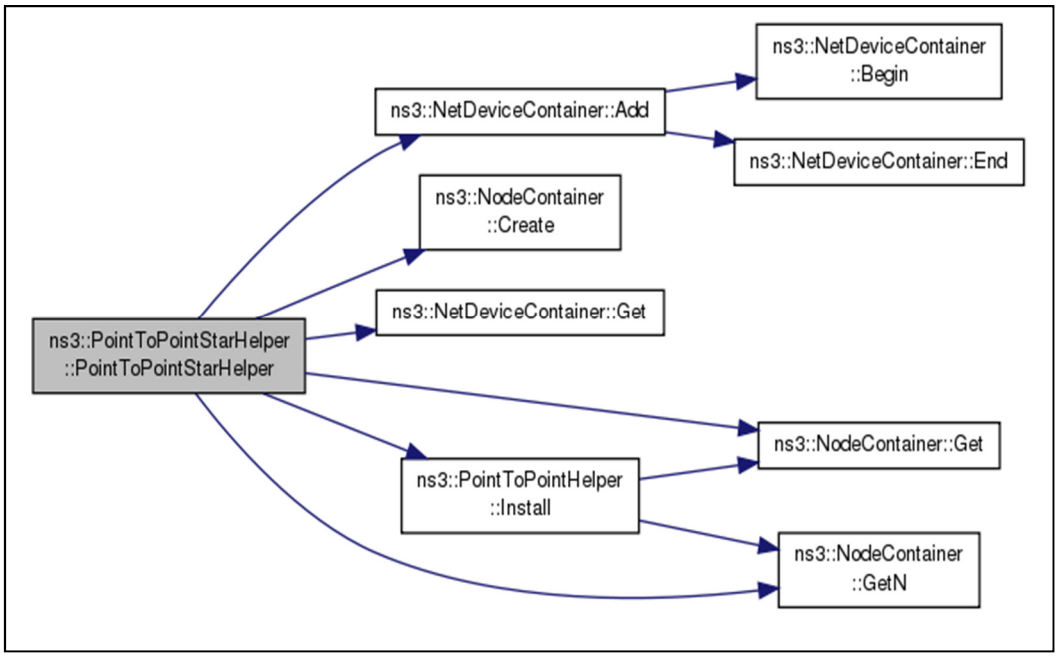


Fig 10: Star Helper Class calling Hierarchy

Member Functions:

Function 1:

void ns3::PointToPointStarHelper::AssignIpv4Addresses(Ipv4AddressHelper *address*)

Parameters

Address Ipv4AddressHelper which is used to install **Ipv4** addresses on all the node interfaces in the star

Function 2:

void ns3::PointToPointStarHelper::InstallStack (InternetStackHelper *stack*)

Parameters

stack an **InternetStackHelper** which is used to install on every node in the star

Function 3:

uint32_t ns3::PointToPointStarHelper::SpokeCount ( ) const

Returns

The total number of spokes in the star

Step 1: Define the No of spokes (No of nodes connecting in Star Topology)

```
uint32_t nSpokes = 8;
```

Step 2: Set the attributes of one Point to Point link

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

Step 3 : Install the Internet stack on nodes

```
NS_LOG_INFO ("Install internet stack on all nodes.");
InternetStackHelper internet;
star.InstallStack (internet);
```

Step 4: Assign the IP addresses using a helper and set the base.

```
NS_LOG_INFO ("Assign IP Addresses.");
star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"))
```

Step5: Create a packet sink on the star "hub" to receive packets using Packet Sink Helper Class

```
uint16_t port = 50000;
Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", hubLocalAddress);
ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());
hubApp.Start (Seconds (1.0));
hubApp.Stop (Seconds (10.0));
```

Step 6: Create On Off applications to send TCP to the hub, one on each spoke node using OnOff Helper class

```
OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
```

Step7: Turn on global static routing so that nodes data can actually be routed across the star.

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

NS_LOG_INFO ("Enable pcap tracing.");
```

Step 8: Enable pcap tracing on all point-to-point devices on all nodes by EnablePcapAll feature.

```
pointToPoint.EnablePcapAll ("star");
```

Step 9: Finally run the simulation and destroy in order to release the resources.

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

VIVA Questions

Q1 What is network topology?

Q2. What is the difference between Star and Bus topology?

Q3. What is role of socket address in networking?

Q4. Why server is started before the client in a NS3 program?

Q5. What is the role of command line arguments in a NS3 program?

EXPERIMENT 7

AIM: Using Free Open Source Software tools ns3, design and implement a bus topology using CSMA.

Description:

Carrier Sense Multiple Access (CSMA) Channel:

This represents a simple CSMA channel that can be used when many nodes are connected to one wire. It uses a single busy flag to indicate if the channel is currently in use. It does not take into account the distances between stations or the speed of light to determine collisions.

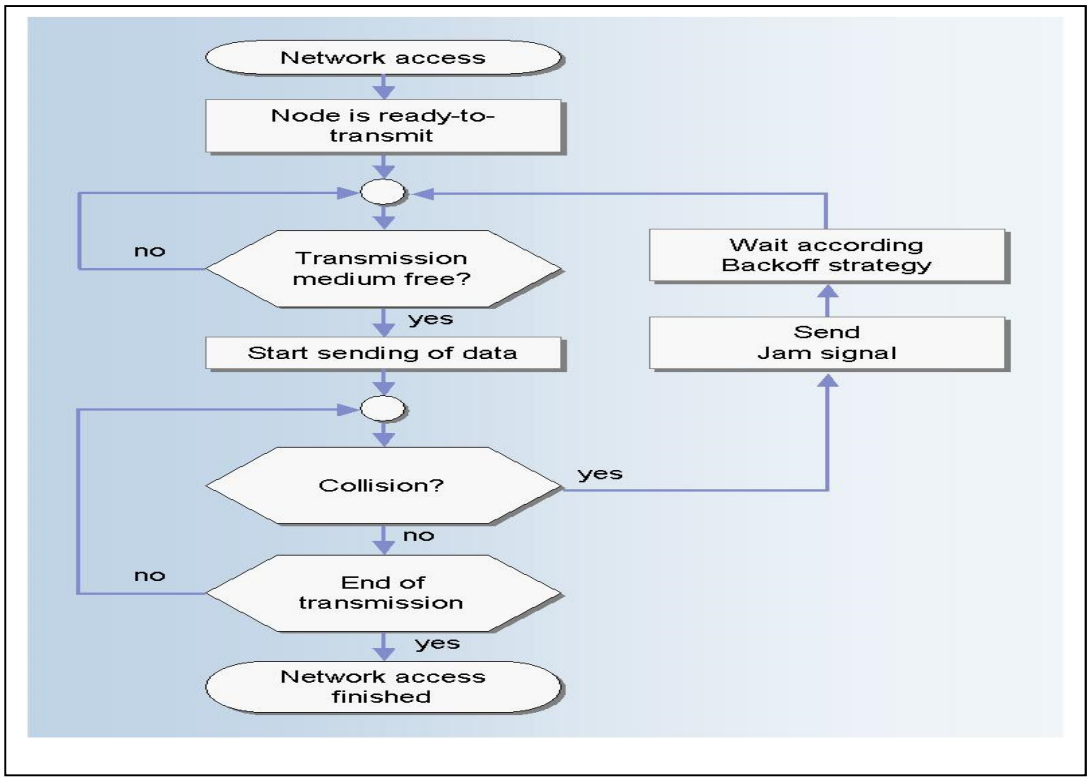


Fig 11: CSM/CD Protocol (Channel Transmission)

Function 1:

NetDeviceContainer ns3::CsmaHelper::Install (Ptr< Node > node) const

Description:

This method creates an **ns3::CsmaChannel** with the attributes configured by **CsmaHelper::SetChannelAttribute**, and **ns3::CsmaNetDevice** with the attributes configured by **CsmaHelper::SetDeviceAttribute**, then adds the device to the node and attaches the channel to the device.

Parameters

node The node to install the device in

Returns

A container holding the added net device.

Function 2:

NetDeviceContainer ns3::CsmaHelper::Install (Ptr< Node >

std::string channelName

) const

Description:

This method creates an **ns3::CsmaNetDevice** with the attributes configured by **CsmaHelper::SetDeviceAttribute** and then adds the device to the node and attaches the provided channel to the device.

Parameters

node The node to install the device in

channelName The name of the channel to attach to the device.

Returns

A container holding the added net device.

Step 1: **Define the No of connecting in CSMA**

```
uint32_t nCsma = 3;
```

Step 2: **Set the attributes of one Point to Point link**

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

Step 3: Declare one NetDevice Container.

```
NetDeviceContainer p2pDevices;
```

Step 4: Install the Point to Point link on Devices (Nodes).

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

Step 5 : Set CSMA channel attributes and install on nodes

```
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
csmaDevices = csma.Install (csmaNodes);
```

Step 6 : Install the Internet stack on nodes

```
stack.Install (p2pNodes.Get (0));
stack.Install (csmaNodes);
```

Step 7: Assign the IP addresses using a helper and set the base.

```
p2pInterfaces = address.Assign (p2pDevices);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
```

Step8: Set the applications using UDP Echo Server and UDP Echo Client applications. Install the

server application on one node and set the Port No. for accessing the services. Install the client applications on two nodes; set the communication attributes in terms of *Packet Size* and *Interval* while client is communicating with server with the defined port no.

```
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
```

Step9: Turn on global static routing so that nodes data can actually be routed across the star.

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Step 10: Enable pcap tracing on all point-to-point devices on all nodes by EnablePcapAll feature.

```
pointToPoint.EnablePcapAll ("second");
csma.EnablePcap ("second", csmaDevices.Get (1), true);
```

Step 11: Finally run the simulation and destroy in order to release the resources.

```
Simulator::Run ();
Simulator::Destroy ();
```

VIVA Questions

Q1 What is the difference between CSMA/CA and CSMA/CD protocol?

Q2 Explain the flowchart of CSMA protocol?

Q3. Explain the persistence strategy and the types of persistence strategies?

Q4. Which persistence strategy is the best and why?

Q5. What is the difference between Star and Bus topology?

EXPERIMENT-8

AIM- Using Free Open Source Software tools ns3, design and implement hybrid topology connecting multiple routers and nodes.

BUILDING NETWORK TOPOLOGY

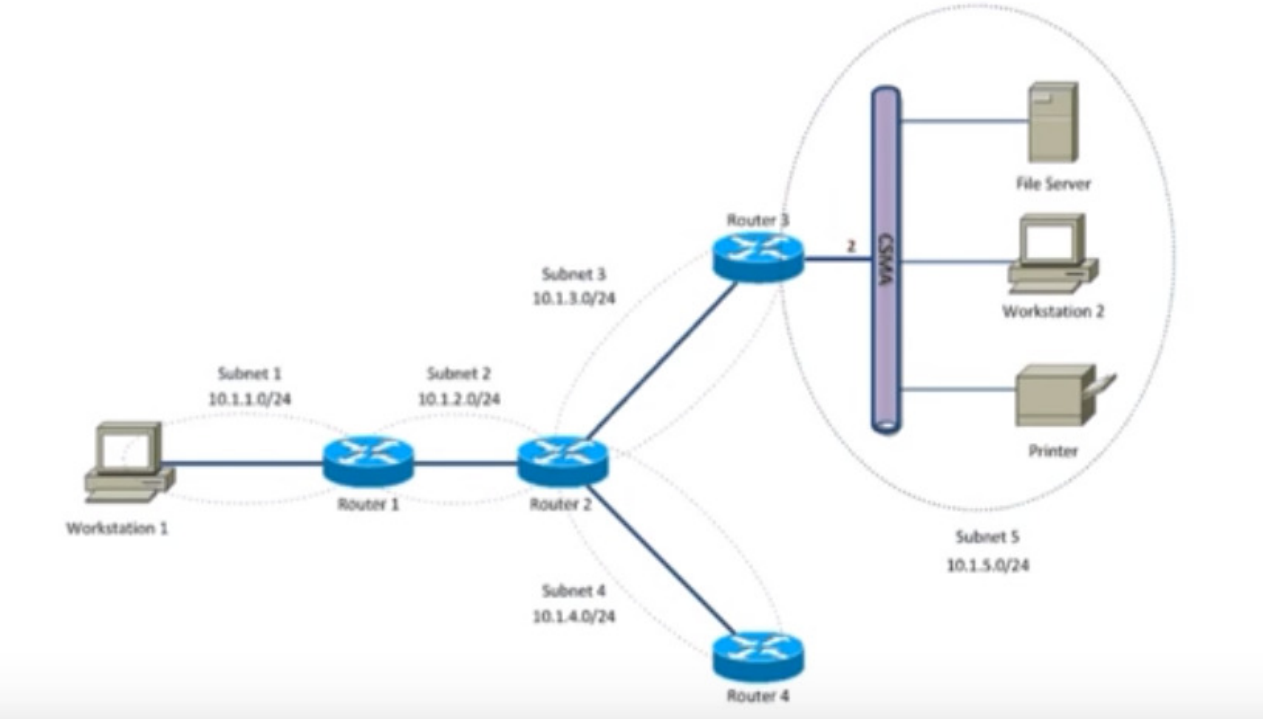


Fig 12: Hybrid Topology

Step 1: **Create nodes that include Host, Four Routers and Host1.**

```
NodeContainer host, router, host1;
host.Create (2);
router.Create (4);
```

Step2: Set subnets and assign host to each subnet.

```
NodeContainer subnet1;
subnet1.Add (host.Get (0));
```

```
subnet1.Add (router.Get (0));
```

Step 3: Set the attributes of one Point to Point link.

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

Step 4: Declare one NetDevice Container.

```
NetDeviceContainer subnet1Devices;
subnet1Devices = pointToPoint.Install (subnet1);
```

Step 5: Install the Point to Point link on Devices (Nodes).

```
subnet1Devices = pointToPoint.Install (subnet1);
```

Step 6: Install the Internet stack on nodes.

```
InternetStackHelper stack;
stack.Install (router);
stack.Install (host);
```

Step7: Assign the IP addresses using a helper and set the base.

```
Ipv4AddressHelper address1, address2, address3, address4, address5, address6,;
Address1.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer subnet1Interfaces;
subnet1Interfaces = address1.Assign (subnet1Devices);
```

Step8: Set the applications using UDP Echo Server and UDP Echo Client applications.

```
UdpEchoServerHelper echoServer (9);
UdpEchoClientHelper echoClient (subnet5Interfaces.GetAddress (1), 9);
```

Step 9: Install the server application on node and set the Port No. for accessing the services.

```
ApplicationContainer serverApps = echoServer.Install (subnet5.Get (1));
```

Step 10: Install the client applications on two nodes; set the communication attributes in terms of *Packet Size* and *Interval* while client is communicating with server with the defined port no.

```
echoClient.SetAttribute ("MaxPackets", UintegerValue (3));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (subnet1.Get (0));
```

Step 11: As for simulations each application is required to Start and Stop after a time interval; Start and Stop the Ser4ver and the Clients.

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

Step12: Turn on global static routing to populate the routing tables.

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

Step 13: Finally run the simulation and destroy in order to release the resources.

```
Simulator::Run ();
Simulator::Destroy ();
```

EXPECTED OUTPUT

```
[root@mta-192-180-1-231 ns-3.24.1]# ./waf --run scratch/pragati2
Waf: Entering directory `/home/mait/Downloads/ns-allinone-3.24.1/ns-3.24.1/bu
Waf: Leaving directory `/home/mait/Downloads/ns-allinone-3.24.1/ns-3.24.1/bui
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.631s)
At time 1s client sent 1024 bytes to 10.1.5.2 port 9
At time 1.01475s server received 1024 bytes from 10.1.1.1 port 49153
At time 1.01475s server sent 1024 bytes to 10.1.1.1 port 49153
At time 1.02949s client received 1024 bytes from 10.1.5.2 port 9
At time 2s client sent 1024 bytes to 10.1.5.2 port 9
At time 2.01475s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.01475s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.02949s client received 1024 bytes from 10.1.5.2 port 9
At time 3s client sent 1024 bytes to 10.1.5.2 port 9
At time 3.01475s server received 1024 bytes from 10.1.1.1 port 49153
At time 3.01475s server sent 1024 bytes to 10.1.1.1 port 49153
At time 3.02949s client received 1024 bytes from 10.1.5.2 port 9
[root@mta-192-180-1-231 ns-3.24.1]#
```

VIVA Questions

Q1 What is the difference between amplifier and repeater?

Q2 What is hub and how it is different from router?

Q3. What is the role of animation in NS3 simulator?

Q4. Explain difference between TCP/IP protocol suite and OSI model?

Q5. How communication can be set between nodes of a ring network?

EXPERIMENT 9

AIM: To Install and configure NetAnim

Description:

Installing NetAnim

URL:

http://www.nsnam.org/wiki/index.php/NetAnim

1. **Install Mercurial:**

   $apt−get/dnf install mercurial

2. **Install QT4 development package:**

   $apt−get/dnf install qt4−dev−tools

3. **Use Synaptic alternatively to install above mentioned packages.**

4. **Download NetAnim: hg clone http://code.nsnam.org/netanim**

5. **Build NetAnim:**

   $cd netanim-3.xxx
   $make clean
   $qmake / qmake-qt?
   $NetAnim. Pro
   $make

Compiling code with NetAnim

Make the following changes to the code, in order to view the animation on NetAnim.

```
#include     " ... "
#include "ns3/netanim−module .h"                    //1 Include. . .
 int main ( int argc , char ∗argv [ ] )
  { std : : string animFile = "somename. xml";  //2 Name of f i l e for animation
            . . .
              AnimationInterface anim ( animFile );          //3 Animation     interface
            Simulator : : Run ();
            Simulator : : Destroy ();
            return 0;
}
```

To run the code:

1. Move the waf , waf.bat , wscript and wutils.py les in to the scratch folder (~/ns-allinone-3.24/ns-3.24/scratch/).
2. Move the example code to the scratch folder and make the changes required for NetAnim, as shown above.
3. Now cd to the scratch folder (cd ~/ns-allinone-3.24/ns-3.24/scratch/).
4. Run the code using the command:

   $./ waf --run <filename>

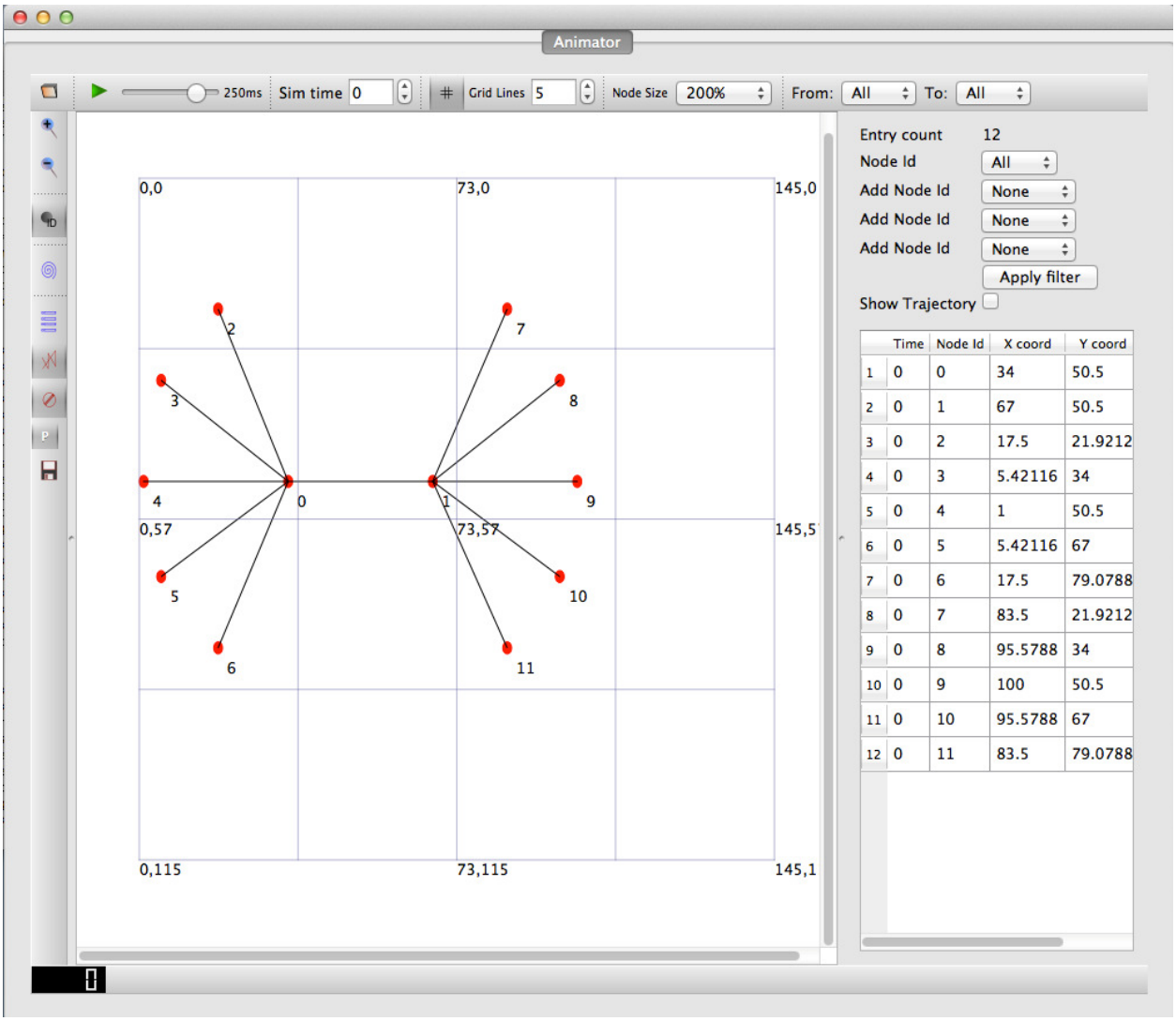
Note: < lename> should not contain the extension .cc

To visualize on NetAnim:

1. cd to the netanim folder ($cd ~/netanim/).
2. Run Netanim:

   $./NetAnim

3. Include the .xml file generated in the ns-3.24 folder (~/ns-allinone-3.17/ns3.24/).



	Time	Node Id	X coord	Y coord
1	0	0	34	50.5
2	0	1	67	50.5
3	0	2	17.5	21.9212
4	0	3	5.42116	34
5	0	4	1	50.5
6	0	5	5.42116	67
7	0	6	17.5	79.0788
8	0	7	83.5	21.9212
9	0	8	95.5788	34
10	0	9	100	50.5
11	0	10	95.5788	67
12	0	11	83.5	79.0788

Fig 13: NetAnim GUI

VIVA Questions

Q1 What is the role of NetAnim in NS3 simulator?

Q2 What is the role of scratch folder in NS3 program simulation?

Q3. What is the role of animation in NS3 simulator?

Q4. Explain TCP/IP protocol suite?

Q5. What is the hybrid network?

Sample Programs beyond Syllabus

EXPERIMENT 1

AIM: Using Free Open Source Software tools ns3, design and implement FTP using TCP bulk transfer.

Description:

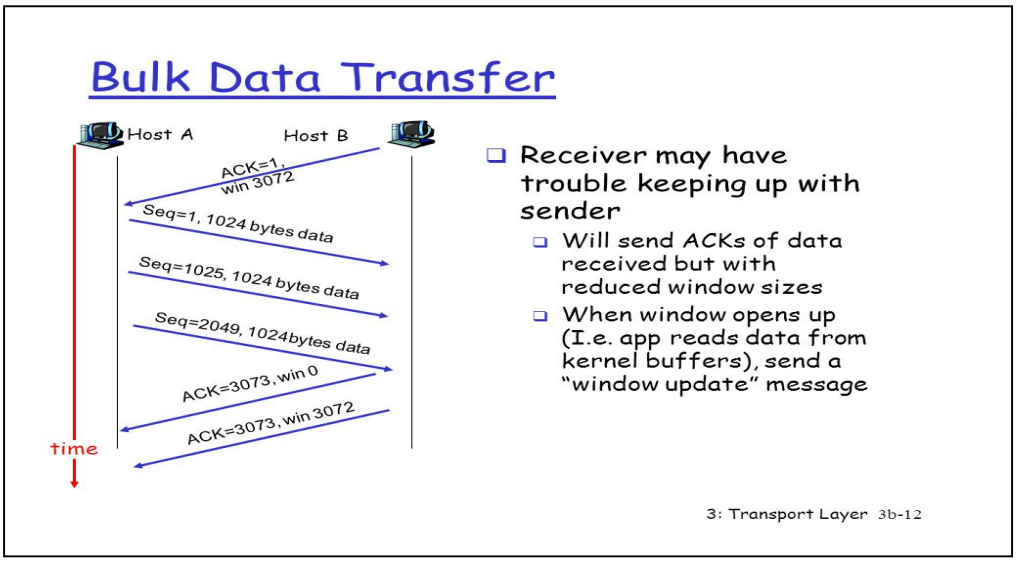


Fig 14: TCP Bulk Transfer

Topology Design:

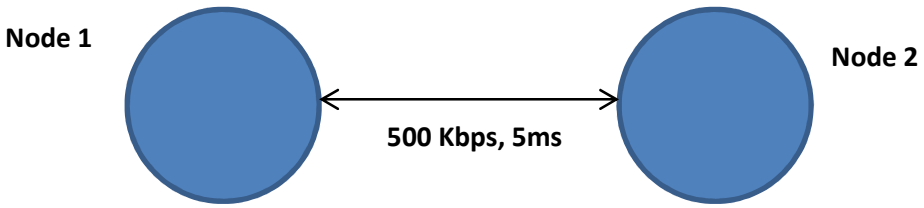


Fig 15: Data Transfer Using TCP with Size=500 Kbps, Delay=5ms

Function 1:

ns3::BulkSendHelper::BulkSendHelper (std::string protocol,

Address address

Description:)

Create an **BulkSendHelper** to make it easier to work with BulkSendApplications.

Parameters

protocol	the name of the protocol to use to send traffic by the applications. This string identifies the socket factory type used to create sockets for the applications. A typical value would be **ns3::UdpSocketFactory**.
address	the address of the remote node to send traffic to.

Function 2:

ApplicationContainer ns3::BulkSendHelper::Install (NodeContainer c) const

Description:

Install an **ns3::BulkSendApplication** on each node of the input container configured with all the attributes set with SetAttribute.

Parameters

c **NodeContainer** of the set of nodes on which an **BulkSendApplication** will be installed.

Returns

Container of **Ptr** to the applications installed.

Step1: create the nodes required by the topology

```
NodeContainer nodes;
nodes.Create (2);
```

Step2: create the point-to-point link required by the topology

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));
```

Step 3: Declare one NetDevice Container.

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Step4: Install the internet stack on the nodes

```
InternetStackHelper internet;
internet.Install (nodes);
```

Step5 : Install IP addresses.

```
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign (devices);
```

Step 6: Create a BulkSendApplication and install it on node 0

```
BulkSendHelper source ("ns3::TcpSocketFactory",InetSocketAddress (i.GetAddress (1), port));
```

Step 7: Set the amount of data to send in bytes. Zero is unlimited.

```
source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
```

Step 8: Declare the Application Container and Install it on Source Node

```
ApplicationContainer sourceApps = source.Install (nodes.Get (0));
sourceApps.Start (Seconds (0.0));
```

```
sourceApps.Stop (Seconds (10.0));
```

Step 9: Create a PacketSinkApplication and install it on node 1

```
PacketSinkHelper sink ("ns3::TcpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), port));
ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
sinkApps.Start (Seconds (0.0));
sinkApps.Stop (Seconds (10.0));
```

Step 10: Enable Tracing

```
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("tcp-bulk-send.tr"));
pointToPoint.EnablePcapAll ("tcp-bulk-send", false);
```

Step 11: Finally start the simulation

```
Simulator::Stop (Seconds (10.0));
Simulator::Run ();
Simulator::Destroy ();
Ptr<PacketSink> sink1 = DynamicCast<PacketSink> (sinkApps.Get (0));
std::cout << "Total Bytes Received: " << sink1->GetTotalRx () << std::endl;
```

VIVA Questions

Q1 What is TCP bulk transfer and how it is implemented?

Q2 What is FTP and its communication port number?

Q3. What is the role of waf in NS3 simulator?

Q4. Explain difference between TCP/IP protocol suite and OSI model?

Q5. Which topology you will design for your MAIT college internet network?

EXPERIMENT 2

AIM: To analyze network traces using Wireshark software.

Wireshark

Packet Capture (Pcap) File:

In the field of computer network administration, **pcap** (*packet capture*) consists of an application programming interface (API) for capturing network traffic.

It is a capture **file** saved in the format that libpcap and WinPcap use can be read by applications that understand that format, such as tcpdump, Wireshark, CA NetMaster, or Microsoft Network Monitor 3.x. The MIME type for the **file** format created and read by libpcap and WinPcap is application/vnd.tcpdump.**pcap**.

Wireshark:

Wireshark is a GUI network protocol analyzer. It lets you interactively browse packet data from a live network or from a previously saved capture file. **Wireshark**'s native capture file format is **pcap** format, which is also the format used by **tcpdump** and various other tools.

Wireshark is a free and open source packet analyzer. **Wireshark** is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using **pcap** to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows.

Wireshark can read in previously saved capture files. To read them, simply select the File → Open menu or toolbar item. Wireshark will then pop up the "File Open" dialog box.

The "Open Capture File" dialog box:

The "Open Capture File" dialog box allows you to search for a capture file containing previously captured packets for display in Wireshark. The following sections show some examples of the Wireshark "Open File" dialog box. The appearance of this dialog depends on the system.

- Select files and directories.
- Click the Open or OK button to accept your selected file and open it.
- Click the Cancel button to go back to Wireshark and not load a capture file.

Sample View:

Like other protocol analyzers, **Wireshark**'s main window shows 3 views of a packet. It shows a summary line, briefly describing what the packet is. A packet details display is shown, allowing you to drill down to exact protocol or field that you interested in. Finally, a hex dump shows you exactly what the packet looks like when it goes over the wire.

In addition, **Wireshark** has some features that make it unique. It can assemble all the packets in a TCP conversation and show you the ASCII (or EBCDIC, or hex) data in that conversation. Display filters in **Wireshark** are very powerful.

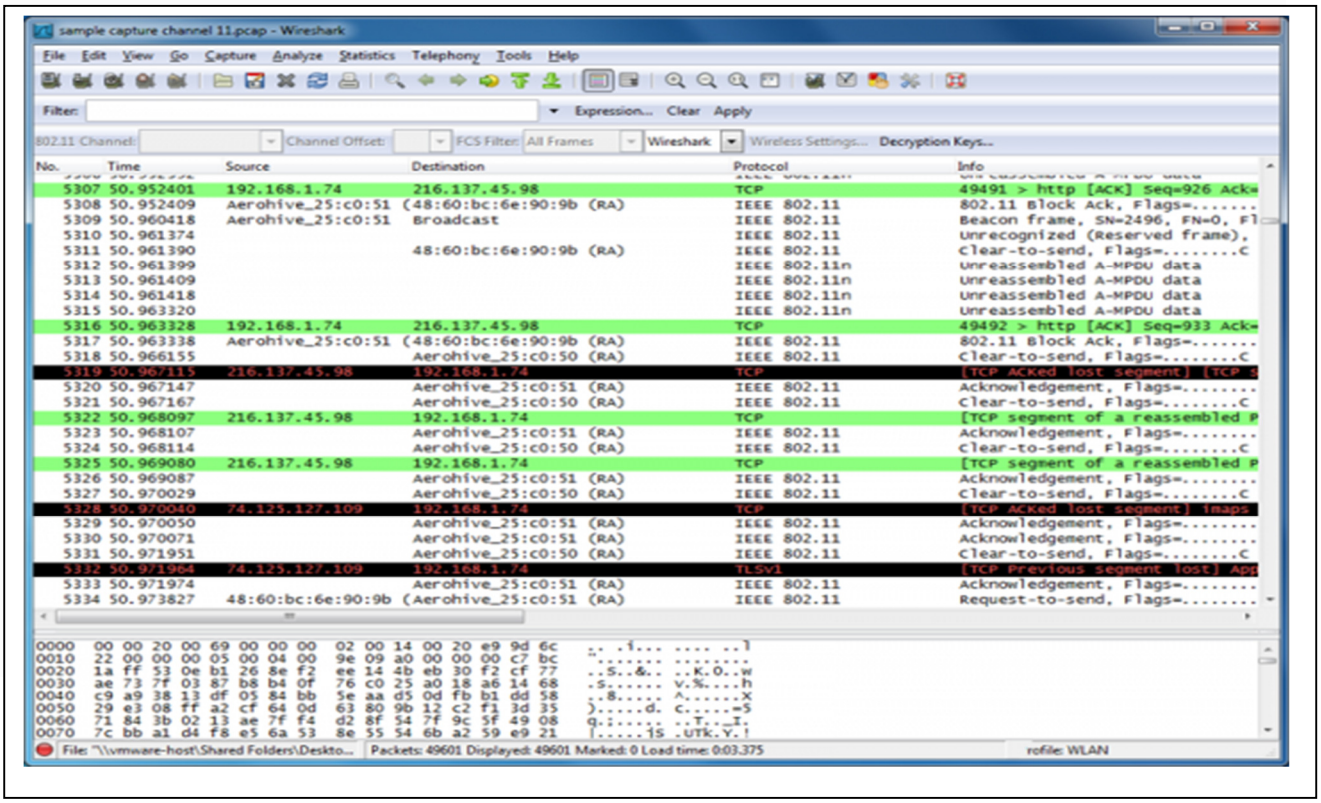


Fig 16: Wireshark GUI

VIVA Questions

Q1 What is Wireshark?

Q2 What is the role of Packet Capture file?

Q3. How you can available enable examples in NS3?

Q4. What is the role of router in any WAN?

Q5. Which topology you will design for any large academic institute?