**COMPUTER NETWORKS LAB**

**(ETCS 354)**

**)**

Faculty name: Anupam Kumar Chander Jindal

06514802719 6th Semester

****

#### Maharaja Agrasen Institute of Technology, PSP area,

Sector – 22, Rohini, New Delhi – 110085

( Affiliated to Guru Gobind Singh Indraprastha University, New Delhi )

### 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.

### 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.

### 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.

**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

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exp. no** | **Experiment Name** | **Date of performance** | **Signature** |
| 1 | Introduction to Computer Network laboratory, introduction to Discrete Event Simulation and Discrete Event Simulation Tools | 4/3/2022 |  |
| 2 | Introduction to NS3 and its comparison with NS2. | 11/3/2022 |  |
| 3 | Install NS3 on Linux. | 25/3/2022 |  |
| 4 | Using Free Open Source Software tools ns3, design and implement two nodes topology. | 1/4/2022 |  |
| 5 | Using Free Open Source Software tools ns3, design and implement three nodes topology considering one node as a central node. | 22/4/2022 |  |
| 6 | Using Free Open Source Software tools ns3, design and Implement star topology using StarHelperClass. | 22/4/2022 |  |
| 7 | Using Free Open Source Software tools ns3, design and implement a bus topology using CSMA. | 6/5/2022 |  |
| 8 | Using Free Open Source Software tools ns3, design and implement hybrid topology connecting multiple routers and nodes. | 13/5/2022 |  |
| **Beyond The Syllabus** | | | |
| 1 | Using Free Open Source Software tools ns3, design and implement FTP using TCP bulk transfer. | 20/5/2022 |  |
|  |  |  |  |

##### 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.

Start

Fig. 1

Define Network

parameters

Run

Simulations

Process

Trace Files

No

Output

OK

Yes

Stop

*Fig 1: Simulation Process*

##### EXPERIMENT 1

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

**System:**

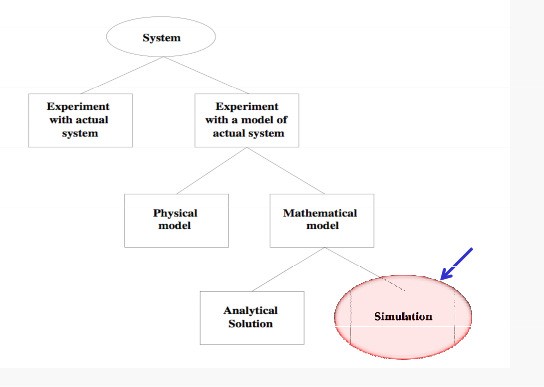
A collection of entities that act and interact together toward the accomplishment of some logical end.

**Discrete system:**

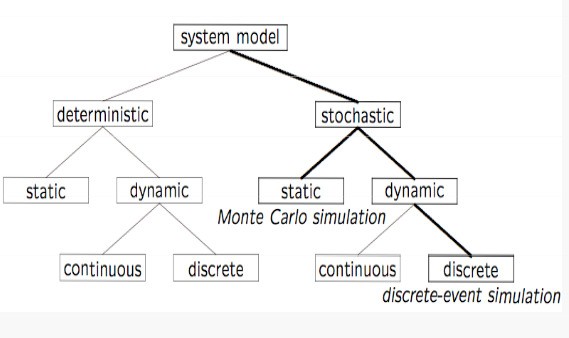
State variables change instantaneously at separated point in time, e.g., a bank, since state variables - number of customers, change only when a customer arrives or when a customer finishes being served and departs

**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:**

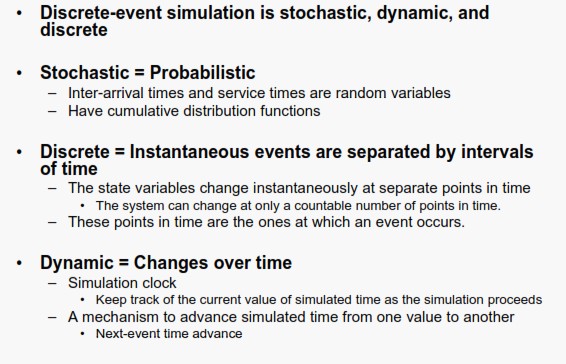
is composed of objects called entities that have certain properties called attributes.

**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



**VIVA Questions**

Q1. What is Discrete Event Simulation?

A discrete-event simulation (DES)models the operation of a system as a (discrete) sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system.

Q2. What is the importance and limitations of simulation?

Main advantages of simulation is Study the behavior of a system without building it.

Main disadvantages of simulation is Expensive to build a simulation model.

Q3. What is the difference between deterministic and stochastic model?

Stochastic model recognizes the random nature of variables, whereas,deterministic models does not include random variables. Stochastic models uses random numbers to do calculations and output determined is also random in nature,whereas,in deterministic model once the inputs are fixed output values can be determined which are also fixed in nature.

### EXPERIMENT 2

**Aim:** Introduction to NS3 and its comparison with NS2.

**Description:**

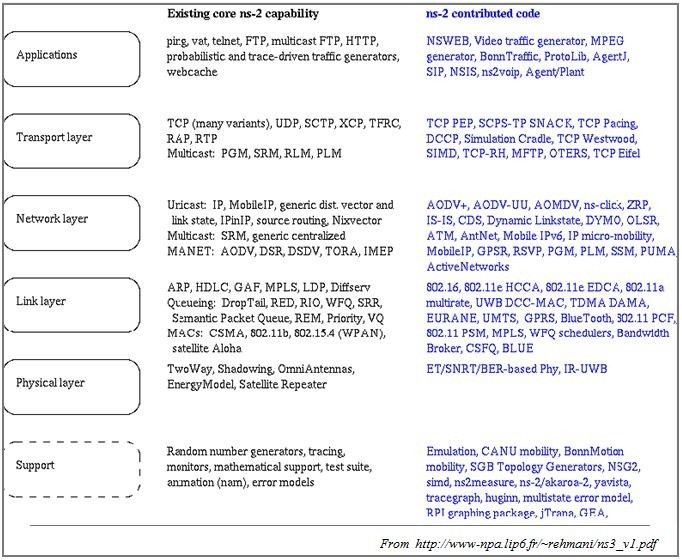
In this lab, we will be using the Network Simulator, NS3, available from [www.nsnam.org.](http://www.nsnam.org/) NS3 is a powerful program, however we will only be looking at some basic features. NS3 simulations are built in C++.

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

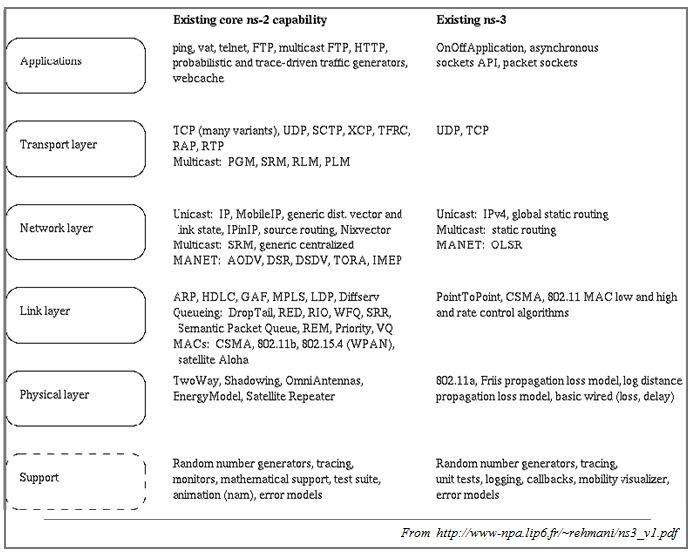
1. Programming Languages
2. Memory Management
3. Packets
4. Performance
5. Simulation Output

**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?

|  |  |
| --- | --- |
| Programming language | Scripting language |
| A programming language is a type of computer language that consists of a set of instructions for communicating with computers. | A scripting language is a kind of programming language that is used to automate the execution of operations in a runtime environment. |
| Traditional programming languages are based on low level languages. | Scripting languages generally prefers high level languages. |
| Execution of a program takes more time since they are compiled. | Execution of a script takes less time as scripts are generally short. |
| It is used to create a new program or piece of software from the scratch. | It is used to enhance an existing program or automate a specific task. A scripting language is used to control the behavior of a program. |
| A programming language generally requires many lines of code to accomplish a particular task. | A scripting language usually requires fewer lines than programming languages to accomplish a task. |
| Programming languages have high maintenance costs. | Scripting languages typically have a low maintenance cost. |
| Programming languages generate .exe files. | Scripting languages do not create .exe files. |
| User interface design and graphic design are often partially supported by programming languages. | User interface design, data types, and graphic design are all greatly aided by scripting languages. |
| Programming languages are self-executable. | Scripting languages requires a host. |
| All programming languages are not scripting languages. | All scripting languages are programming languages. |
| Examples include C, C++, Java, Python, etc. | Examples include Perl, PHP, JavaScript, etc. |

Q2. State the importance of NS3 over NS2?

NS3 simulator is mainly designed for the purpose of education as well as research. When compared with the Ns2 type, it uses Python to work in a better way because of the low-level of abstraction. The modules of Ns3 include protocols and network devices, written in the languages of C++, Python.

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

NS2 is an object-oriented kind of simulator mainly used for simulating the protocols of networking as well as routing protocols for the networks like wired & wireless. These can be implemented through OTCl & C++.

**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

1. After downloading NS3 on the drive, extract all the files in the NS3 folder, which you have created.
2. 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".

1. 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)

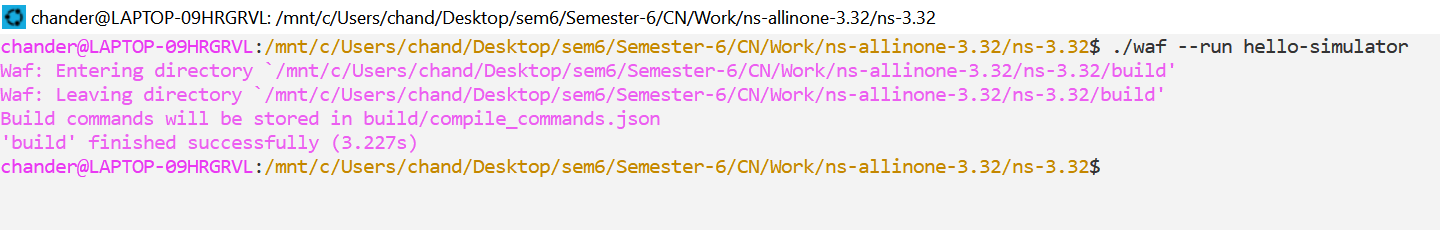
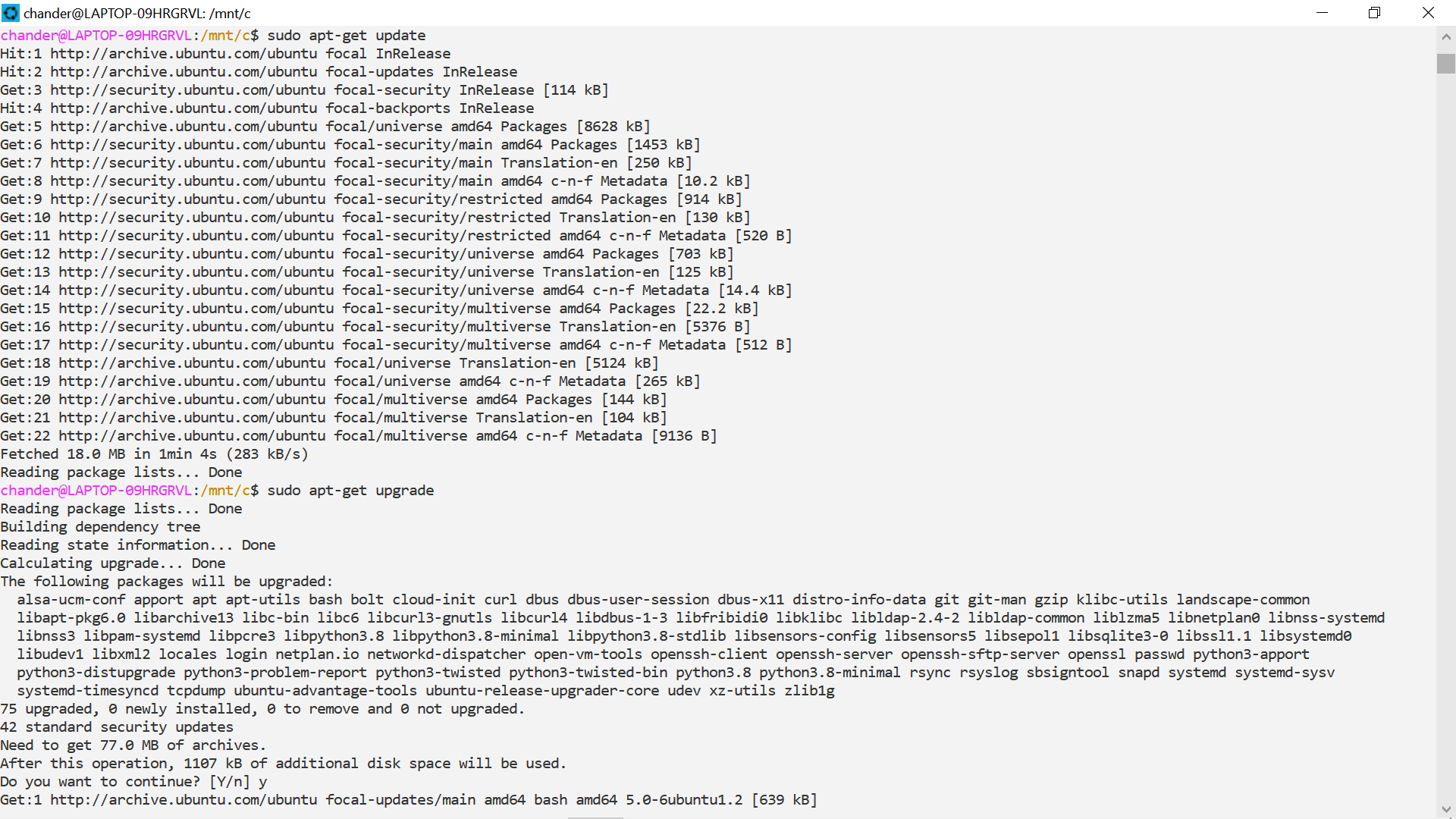
$./waf

1. To test everything all right run the following command on the terminal window,

$./test.py

If the tests are ok the installation is done

1. 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?

Almost all variants of TCP, several forms of multicast, wired networking, several ad hoc routing protocols and propagation models (but not cellular phones), data diffusion, satellite, and other stuff.

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

One should go through the Tutorial, Documentation and Examples given when trying to implementing a new protocol or trying an experiment

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

WAF security detects and filters out threats which could degrade, compromise, or expose online applications to denial-of-service (DoS) attacks. WAF security examines HTTP traffic before it reaches the application server. They also protect against unauthorized transfer of data from the server.

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

Try running a simple program like “Hello Simulator”.

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

Prerequisites The core of ns-3 requires a gcc/g++ installation of 4.9 or greater (Linux), or a recent version of clang compiler (OS X, Linux, or BSD), and Python 3.5 or greater. As mentioned above, different options require additional support. This is a list of packages (for Debian/Ubuntu systems) that are needed to support different ns-3 options.

## EXPERIMENT 4

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

**Node 1**

**Node 2**

###### Description:

**Node**

*Fig 6: Two Node Topology*

NodeContainer

Because in any network simulation, we will need nodes. So ns-3 comes with 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.

Install

Then we need to install the devices. The internal of for now, let’s just skip the magic behind the scene.

is actually more complicated, but

// Channel: PointToPoint, a direct link with `DataRate` and `Delay` specified. PointToPointHelper pointToPoint;

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

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

// 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

InternetStackHelper

focus on it. The

each of the nodes in the node container.

will install an Internet Stack (TCP, UDP, IP, etc.) on

// 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

Stop

Every application needs to have

and

Start

function so that the simulator knows how to

UdpEchoServer

schedule it. Other functions are application-specific. We will use 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));

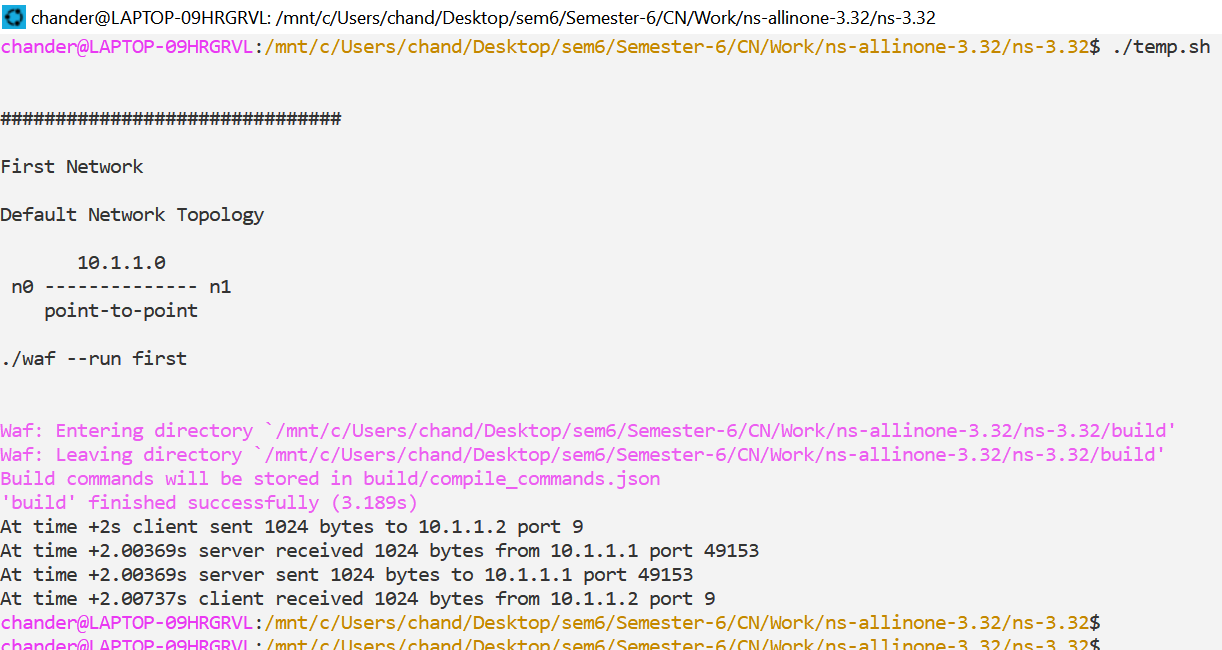
###### Simulation

// Start Simulation Simulator::Run ();

Simulator::Destroy ();

return 0;

**EXPECTED OUTPUT**

****

###### VIVA Questions

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

PPP, this computer network protocol used to transfer a datagram between two directly connected (point-to-point) computers. This protocol is used for a very basic level of connectivity providing data linkage between the computers.

Q2. What is the difference between UDP and TCP?

| Basis | Transmission control protocol (TCP) | User datagram protocol (UDP) |
| --- | --- | --- |
| Type of Service | TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data. | UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast types of network transmission. |
| Reliability | TCP is reliable as it guarantees the delivery of data to the destination router. | The delivery of data to the destination cannot be guaranteed in UDP. |
| Acknowledgment | An acknowledgment segment is present. | No acknowledgment segment. |
| Speed | TCP is comparatively slower than UDP. | UDP is faster, simpler, and more efficient than TCP. |
| Retransmission | Retransmission of lost packets is possible in TCP, but not in UDP. | There is no retransmission of lost packets in the User Datagram Protocol (UDP). |
| Header Length | TCP has a (20-60) bytes variable length header. | UDP has an 8 bytes fixed-length header. |
| Weight | TCP is heavy-weight. | UDP is lightweight. |
| Stream Type | The TCP connection is a byte stream. | UDP connection is message stream. |
| Overhead | Low but higher than UDP. | Very low. |

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

The point-to-point connection is aunicast connection. There is a dedicated link between an individual pair of sender and receiver. The capacity of the entire channel is reserved only for the transmission of the packet between the sender and receiver.

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

Client is a computer (Host) i.e. capable of receiving information or using a particular service from the service providers (Servers). Servers: Similarly, when we talk the word Servers, It mean a person or medium that serves something. If Client Starts before the server, the client can’t do anything as there is nothing for the client to interact with.

Q5. What is the difference between IPv4 and IPv6?

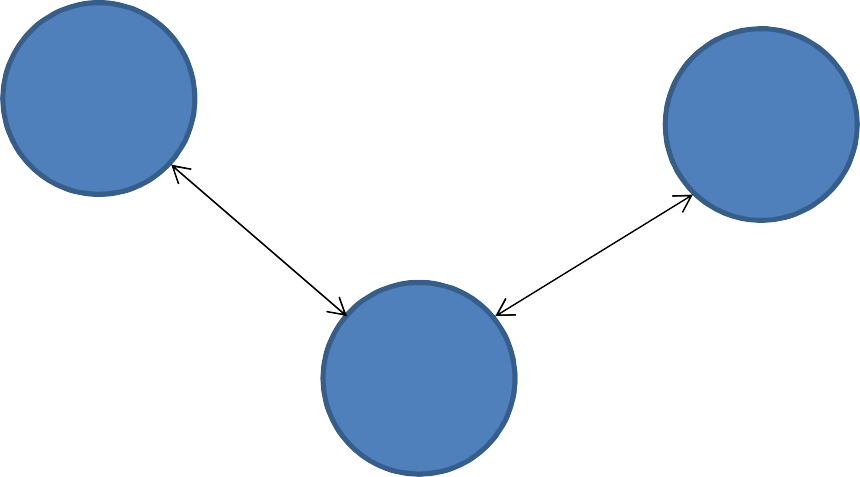
| Basis for differences | IPv4 | IPv6 |
| --- | --- | --- |
| Size of IP address | IPv4 is a 32-Bit IP Address. | IPv6 is 128 Bit IP Address. |
| Addressing method | IPv4 is a numeric address, and its binary bits are separated by a dot (.) | IPv6 is an alphanumeric address whose binary bits are separated by a colon (:). It also contains hexadecimal. |
| Number of header fields | 12 | 8 |
| Length of header filed | 20 | 40 |
| Type of Addresses | Unicast, broadcast, and multicast. | Unicast, multicast, and anycast. |
| Number of classes | IPv4 offers five different[classes of IP Address](https://www.guru99.com/ip-address-classes.html). Class A to E. | lPv6 allows storing an unlimited number of IP Address. |
| Fragmentation | Fragmentation is done by sending and forwarding routes. | Fragmentation is done by the sender. |
| Routing Information Protocol (RIP) | RIP is a routing protocol supported by the routed daemon. | RIP does not support IPv6. It uses static routes. |
| Security | Security is dependent on applications – IPv4 was not designed with security in mind. | IPSec(Internet Protocol Security) is built into the IPv6 protocol, usable with a proper key infrastructure. |
| Packet size | Packet size 576 bytes required, fragmentation optional | 1208 bytes required without fragmentation |
| Packet fragmentation | Allows from routers and sending host | Sending hosts only |

## 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.

**Node** **1/Client** **1**

**Node2** **/Client** **2**

**Node 3/ Server**

*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.

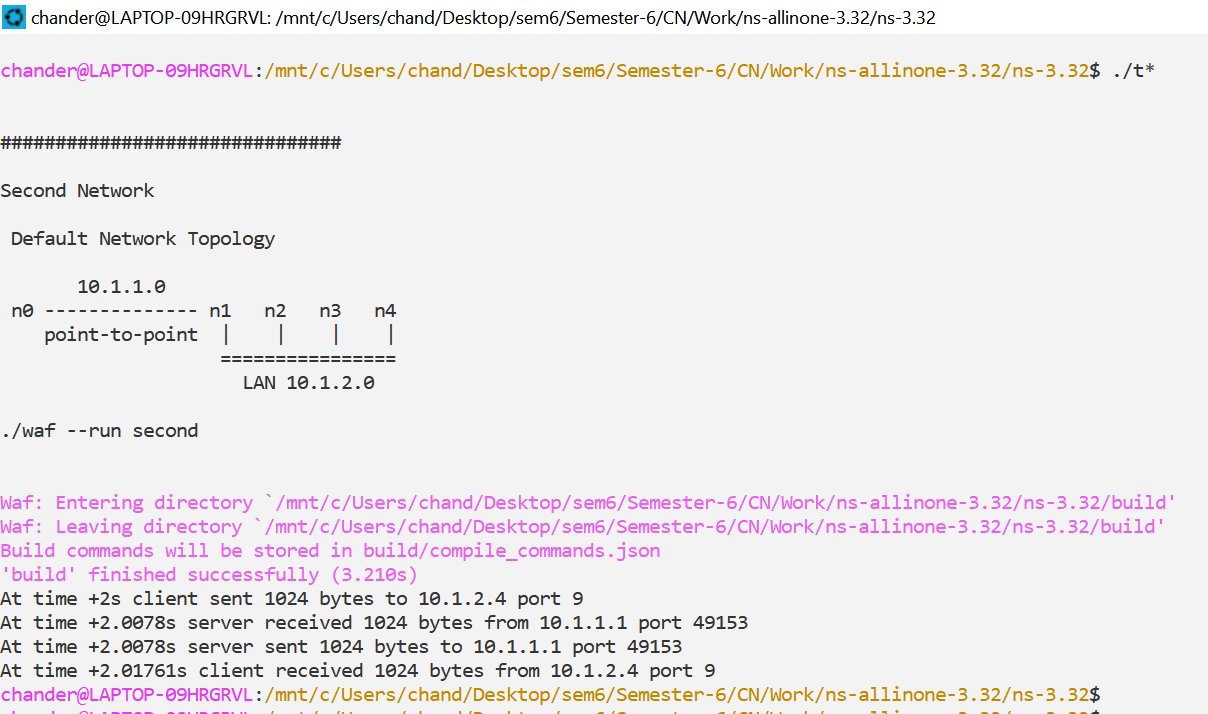
**Step 9:** Install the server application on one node and set the Port No. for accessing the services.

**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**

****

###### 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?

Then other nodes can process the data packet sent, even if the packet is not intended for them.

This because the nodes may identify each other as the source or destination of any given packet.

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

A port number uniquely identifies a network-based application on a computer. Each application/program is allocated a 16-bit integer port number. This number is assigned automatically by the OS, manually by the user or is set as a default for some popular applications.

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

Socket In a very over-simplified sense, a socket is simply the combination of an IP address and a port.

The IP address gets the requested information to the right computer, and the port gets the information to the right application or service running on that computer.

Q4. What is the difference between IP address and physical address?

An IP address is a logical address that is assigned by software residing in a server or router. The physical address is built into the hardware.

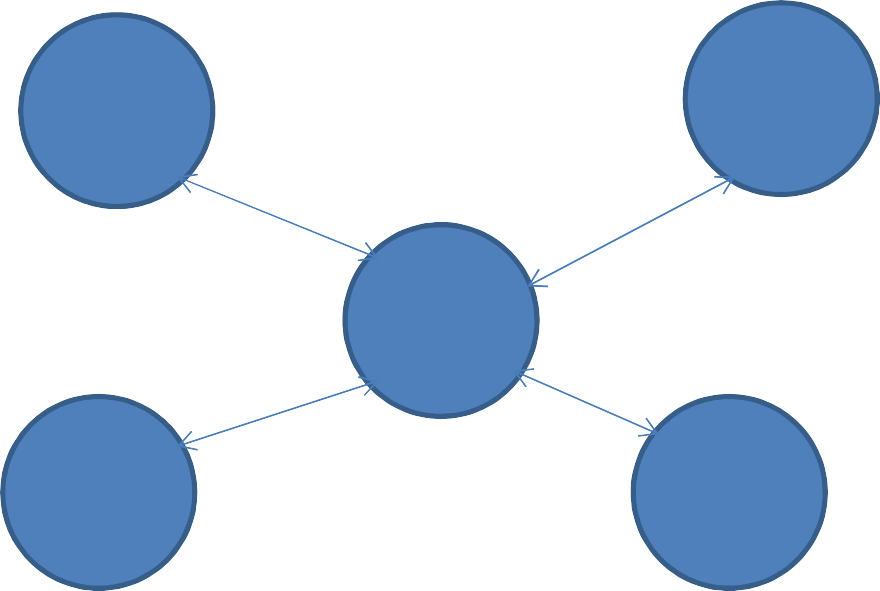
## EXPERIMENT 6

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



**Node2** **/Client** **2**

**Node1** **/Client** **1**

****

**Node3** **/Client** **3**



**Node5** **/Server**

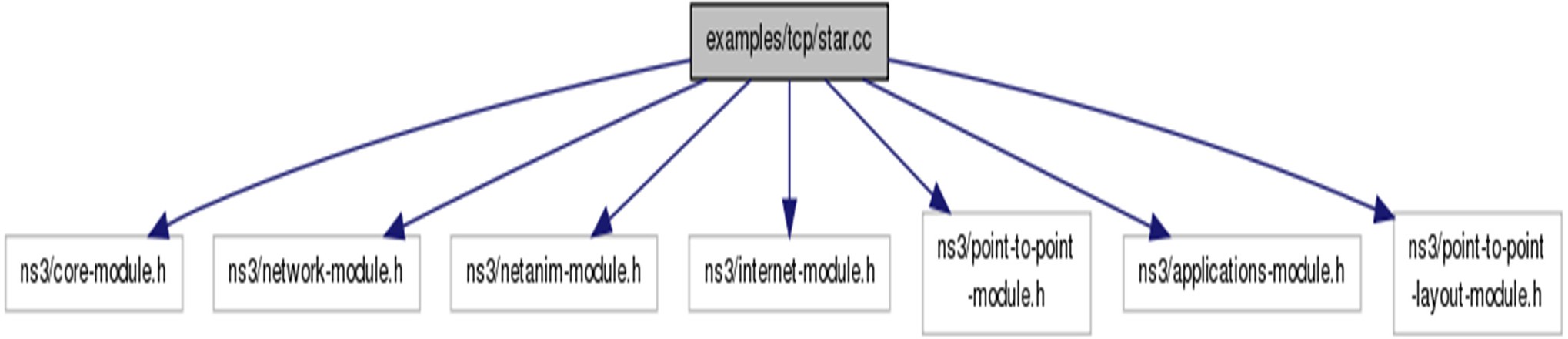
****



**Node4** **/Client** **4**

*Fig 8: Star Topology (using Star Helper Class)*

###### Theory:

****

*Fig 9: Star Helper Class*

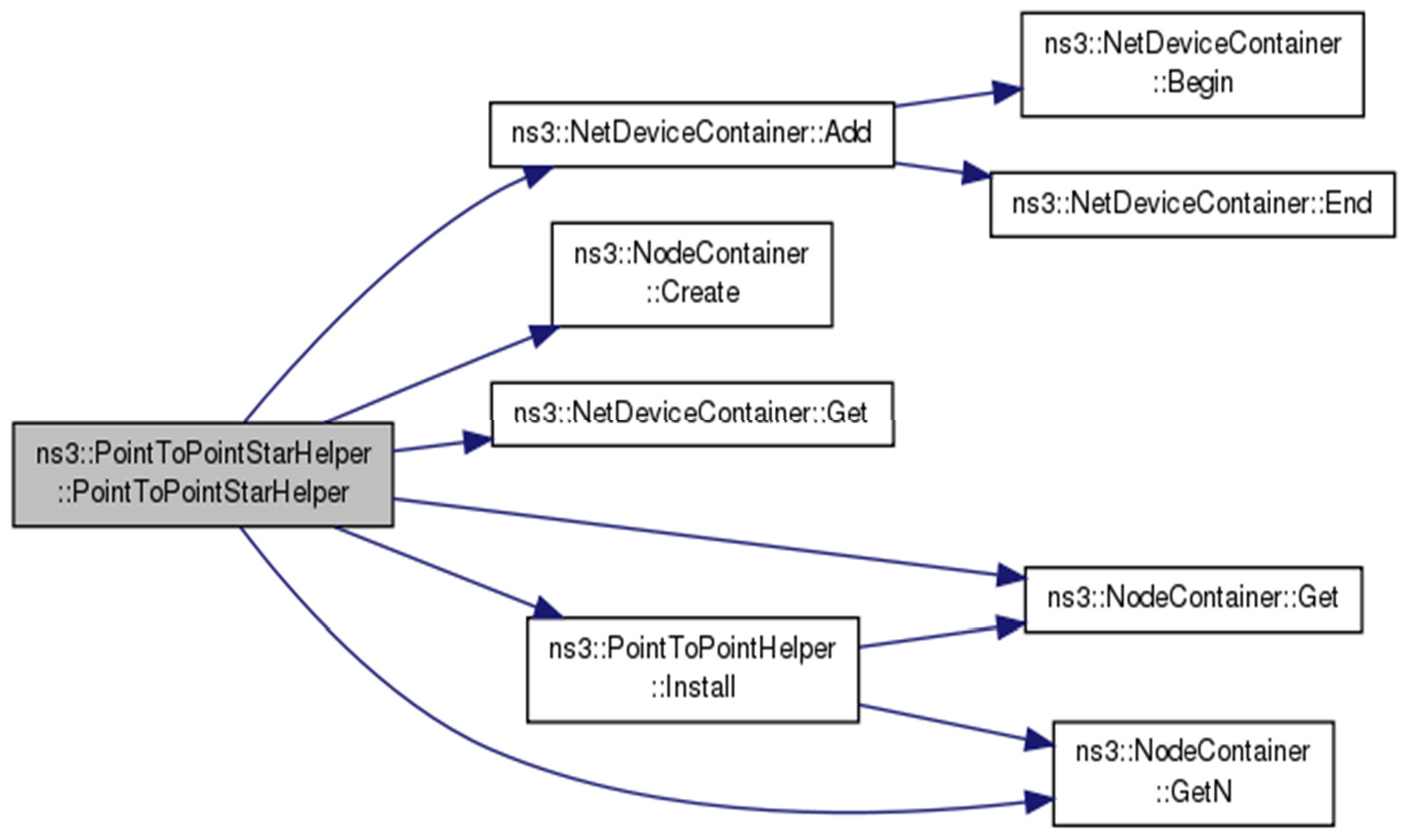
###### ns3::PointToPointStarHelper Class Reference

* + 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]"));

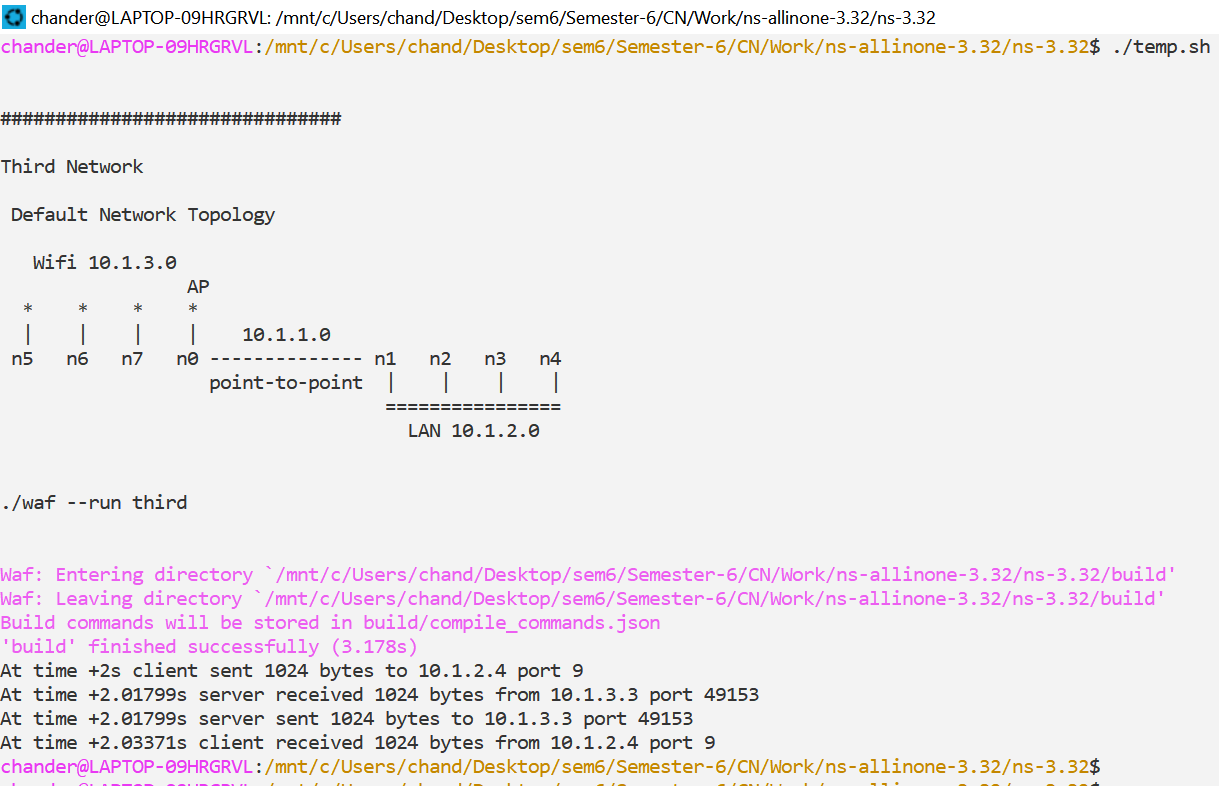
**Step7:** Turn on global static routing so that nodes data can actually be routed across the star.

###### 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?

Basically, it is made from two words, Network and Topology.

Network: Network is an arrangement where two or more computers ( also referred to as nodes ) communicate with each other. These computers or participants nodes actively contribute to the process of communication.

Topology: Topology is simply an arrangement of how these nodes will interact. It is a topology that governs data flow between respective nodes.

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

| S.NO. | Star Topology | Bus Topology |
| --- | --- | --- |
| 1. | Star topology is a topology in which all devices are connected to a central hub. | Bus topology is a topology where each device is connected to a single cable which is known as the backbone. |
| 2. | In star topology, if the central hub fails then the whole network fails. | In a Bus topology, the failure of the network cable will cause the whole network to fail. |
| 3. | Management of high traffic and performance of the network is highly dependent on the capacity of the central hub. | Bus topology can not effectively manage a terminator’s high amount of traffic as if there is high traffic then the performance of the network is affected. |
| 4. | Star topology does not have any terminator. | Bus topology has a terminators at both ends of the network. |
| 5. | Star topology has a high implementation cost because of the central hub and extra wires required for connection. | Bus topology is less expensive than a star topology. |
| 6. | Data transmission is faster in a star topology. | In a Bus topology, the data is transmitted slower as compared to a star topology. |
| 8. | Expansion is easier. | Expansion of network i.e. addition of new node is difficult. |

Q3. What is role of socket address in networking?

It is is used to externally identify it to other hosts.

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

A server program must be executed before the client program because once a client is run, it will attempt to initiate a connection with the server. If the server is down, then the client will not be able to make a connection.

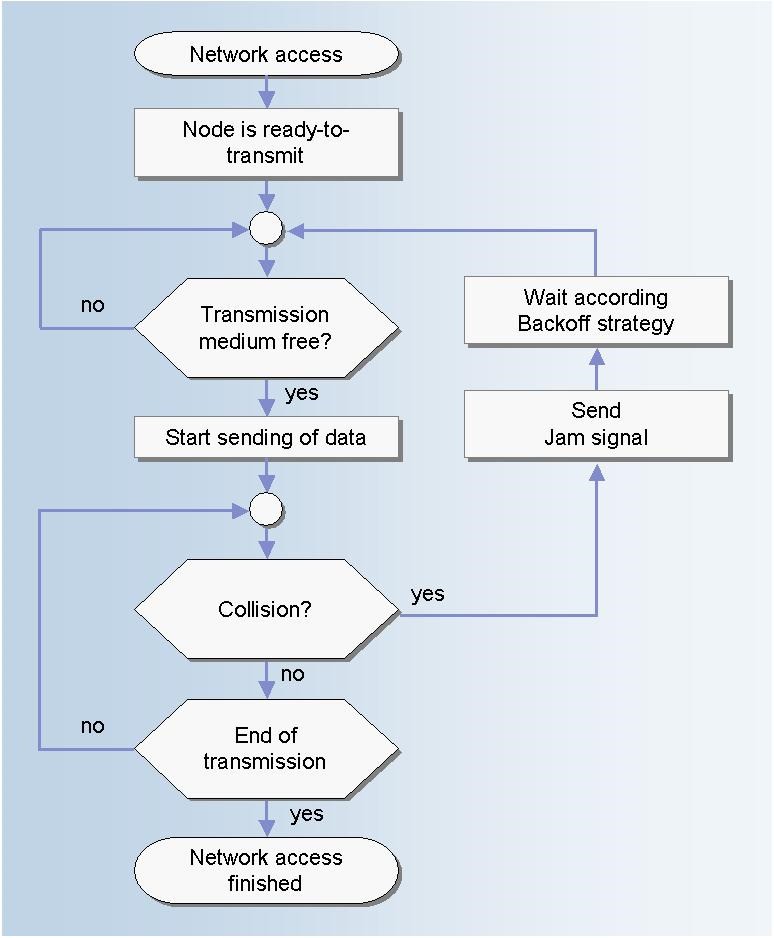
## EXPERIMENT 7

###### 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

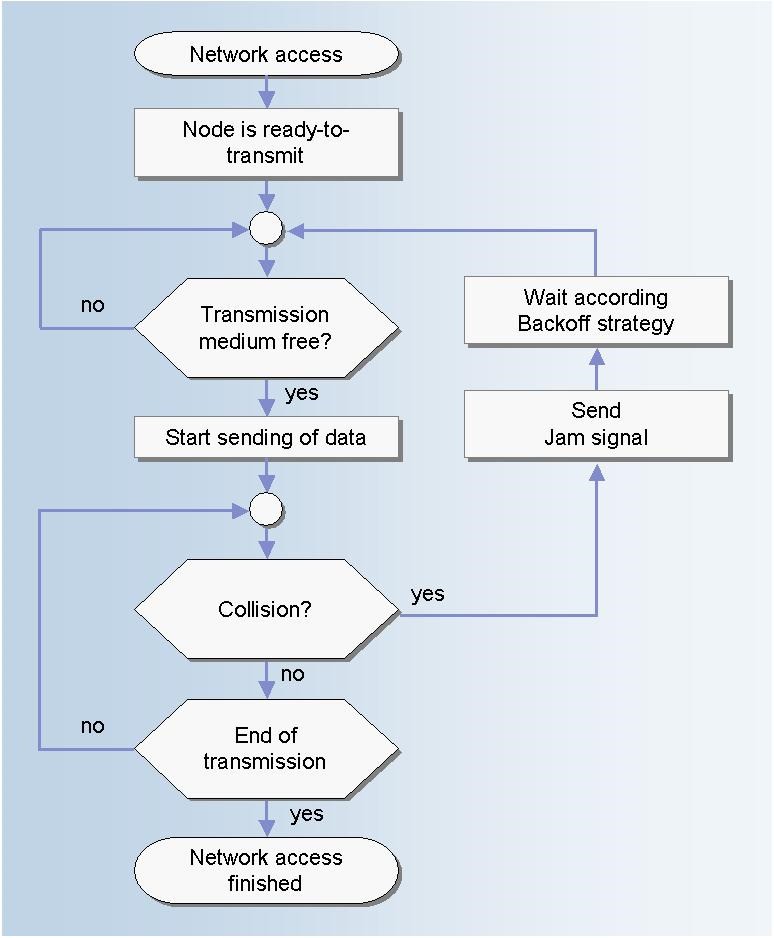
Q1 What is the difference between CSMA/CA and CSMA/CD protocol?

CSMA/CA prevents collisions from happening where as CSMA/CD deals with the aftermath of collisions.

CSMA/CA comes into play before collisions occur where as CSMA/CD comes into play after a collision has already occurred.

Q2 Explain the flowchart of CSMA protocol?

Here once a Signal access the network and a node is ready to transmit the signal.

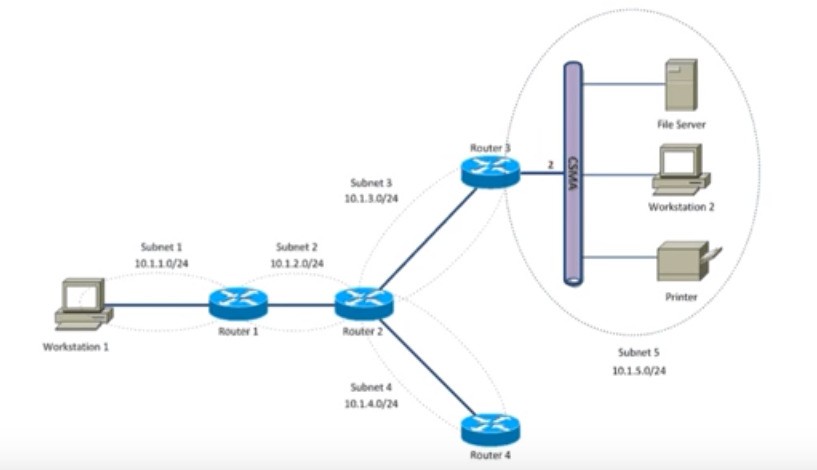


* Check if Transmission medium is free.
* If yes, transmit the signal.
* If no, signal waits till it is free.
* Check for Collisions, if any
* If Collision occurs, then Send Jam signal and wait as per Back off Strategy.
* If No Collisions, the Transmission is Successful.
* End the Network access for that signal.

## 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));

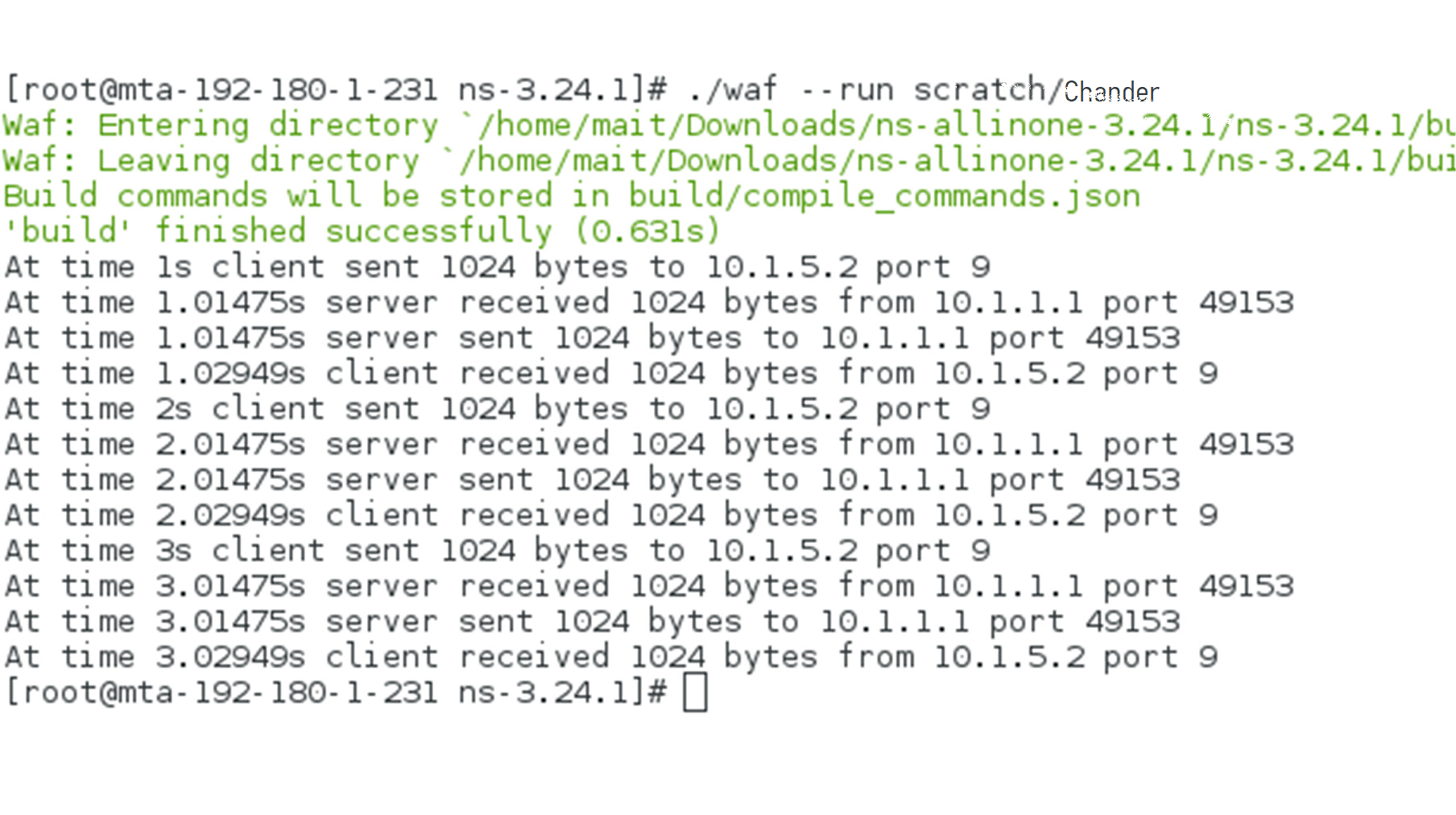
###### Step12: Turn on global static routing to populate the routing tables.

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

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

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

OUTPUT

****

**VIVA Questions**

Q1 What is the difference between amplifier and repeater?

An amplifier receives a signal weak or strong and forwards a strong signal, it doesn’t differentiate between signal and noise, where as, a repeater just repeats the signal, not noise, if the signal received is weak it forwards a weak signal, if signal received is strong it forwards strong signal.

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

Hub is a device that is used for connecting various Ethernet devices together and making them act as a single network segment. On the other hand router is a device that allow computers to communicate of pass information between two networks.

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

The class ns3::AnimationInterface is responsible for the creation the trace XML file. AnimationInterface uses the tracing infrastructure to track packet flows between nodes. AnimationInterface registers itself as a trace hook for tx and rx events before the simulation begins.

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

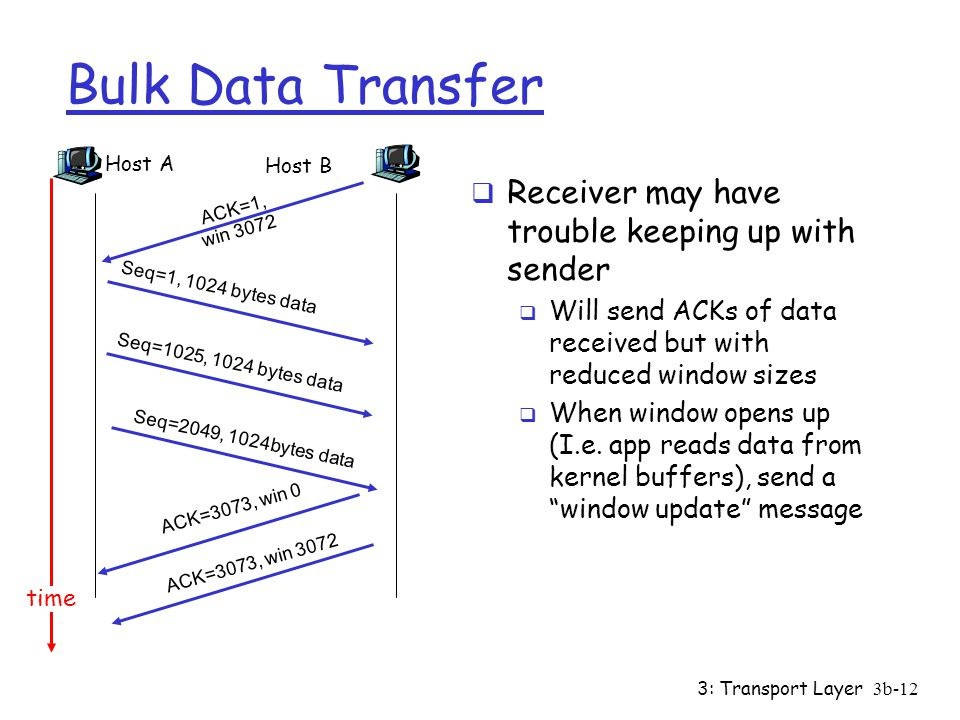
OSI is a conceptual framework while TCP/IP is a communication protocols suite. The OSI model offers seven layers with different functions that the TCP/IP model groups into four layers.

## 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:**

**Node 1**

**Node 2**

**500 Kbps, 5ms**

*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**

**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));

**BulkSendHelper** source(“ns3::TcpSocketFactory”,**InetSocketAddress** (I.**GetAddress** (1), **port));**

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

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

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

**PacketSinkHelper sink** ("ns3::TcpSocketFactory", **InetSocketAddress** (**Ipv4Address::GetAny** (), port)); **ApplicationContainer** sinkApps = **sink**.Install (nodes.**Get** (1));

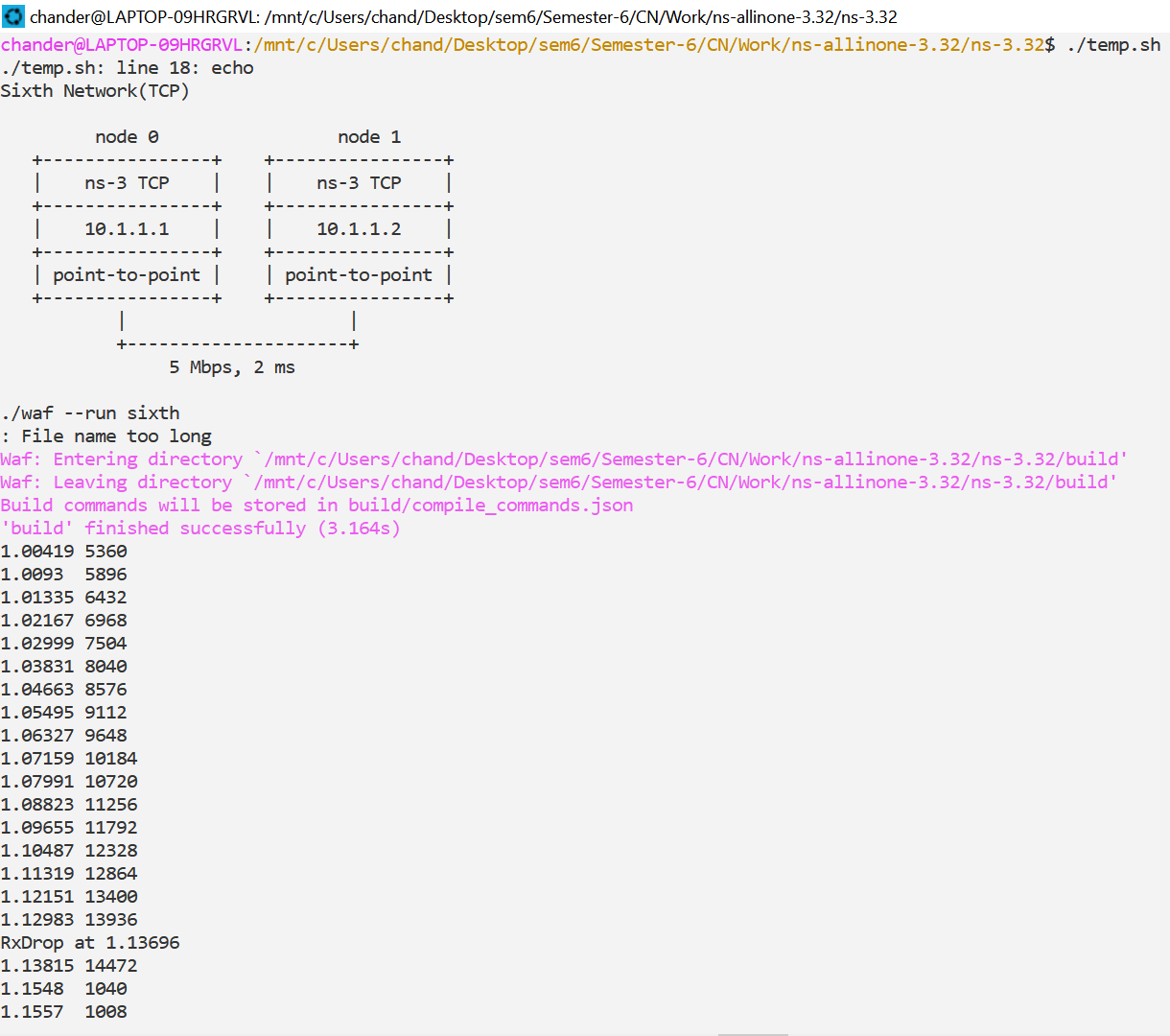
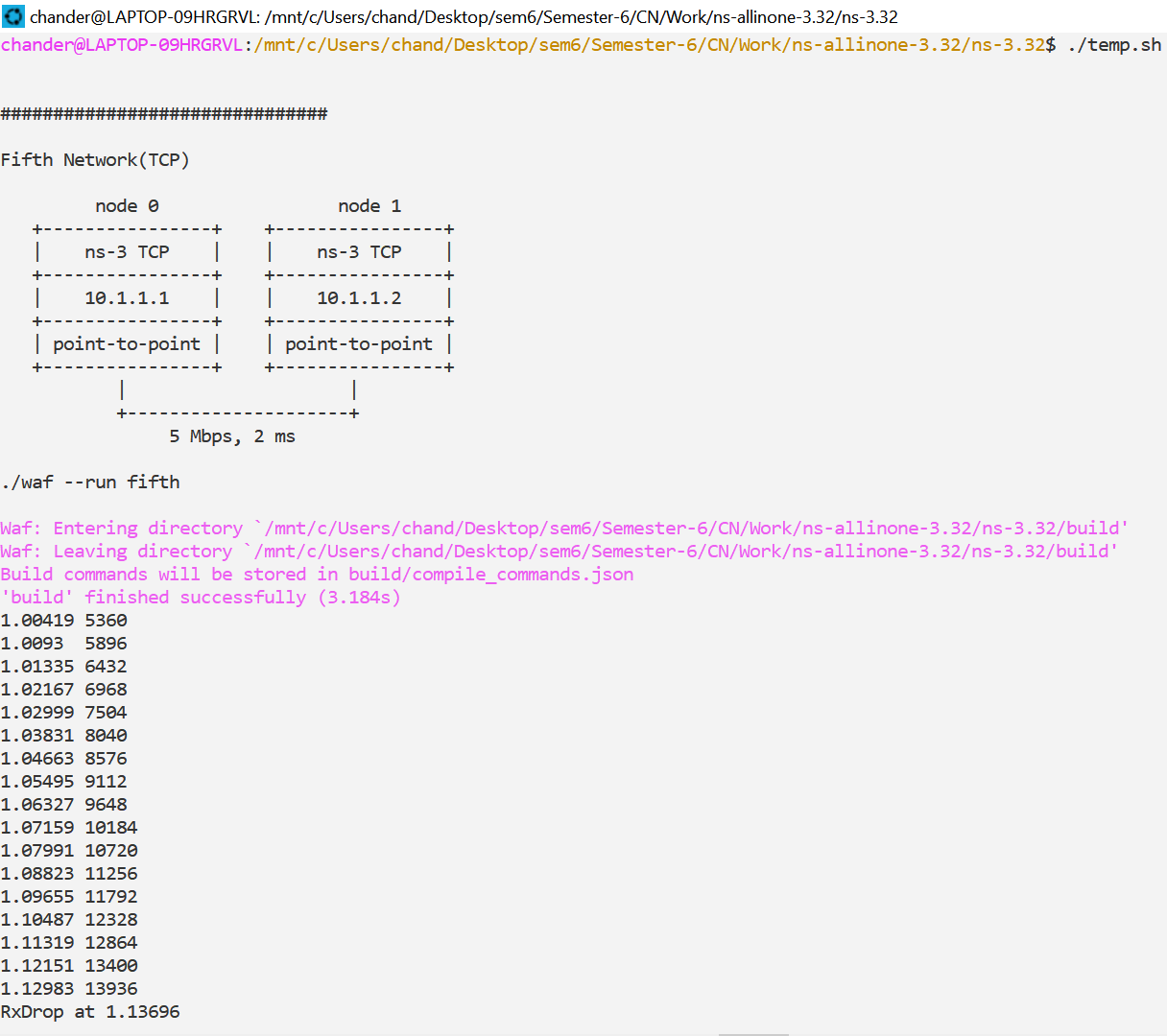
sinkApps.**Start** (**Seconds** (0.0));

**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;

Output



###### 

###### VIVA Questions

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

TCP-Based Bulk Data Transfer The movement of bulk data over long-haul networks has become integral to modern computational science. Raw instrument data, computational data sets, and visualization data must be moved between institutions to take advantage of computational and other resources.

Q2 What is FTP and its communication port number?

File Transfer Protocol (FTP), a standard for the exchange of program and data files across a network, it’s communication Port Number is 21.

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

Open Source Network Simulator lots of open source network simulators are available from that ns-3 is one of the efficient and moderate open source network simulator. Ns-3: In ns3 waf play vital role. Waf is a general purpose build system in ns-3.

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

OSI is a conceptual framework while TCP/IP is a communication protocols suite. The OSI model offers seven layers with different functions that the TCP/IP model groups into four layers.

Shell Script for the Experiments

sudo apt update #get update if any

sudo apt install build-essential autoconf automake libxmu-dev #Install #1

sudo apt install build-essential autoconf automake libxmu-dev python-pygraphviz cvs mercurial bzr git cmake p7zip-full python-matplotlib python-tk python-dev python-kiwi python-gnome2 python-gnome2-desktop qt4-dev-tools qt4-qmake qt4-qmake qt4-default gnuplot-x11 wireshark #Install #2

wget <https://www.nsnam.org/releases/ns-allinone-3.32.tar.bz2> #Get file from link

#git clone <https://gitlab.com/nsnam/ns-3-allinone.git> ← 2nd option

tar jxvf ns-allinone-3.32.tar.bz2 #unzip Linux style

./ns-allinone-3.32/build.py --enable-examples –enable-tests #Build in that directory

touch ns-allinone-3.32/ns-3.32/temp.sh #Make a temp file

echo "

echo 'Hello World!'

./waf --run hello-simulator

echo '

###############################

First Network

Default Network Topology

10.1.1.0

n0 -------------- n1

point-to-point

'

./waf --run first

echo '

###############################

Second Network

Default Network Topology

10.1.1.0

n0 -------------- n1 n2 n3 n4

point-to-point | | | |

================

LAN 10.1.2.0

'

./waf --run second

echo '

###############################

Third Network

Default Network Topology

Wifi 10.1.3.0

AP

\* \* \* \*

| | | | 10.1.1.0

n5 n6 n7 n0 -------------- n1 n2 n3 n4

point-to-point | | | |

================

LAN 10.1.2.0

'

./waf --run third

echo '

###############################

Fourth Experiment'

./waf --run fourth

echo '

###############################

Fifth Network(TCP)

node 0 node 1

+----------------+ +----------------+

| ns-3 TCP | | ns-3 TCP |

+----------------+ +----------------+

| 10.1.1.1 | | 10.1.1.2 |

+----------------+ +----------------+

| point-to-point | | point-to-point |

+----------------+ +----------------+

| |

+---------------------+

5 Mbps, 2 ms

'

./waf --run fifth

echo '

###############################

Sixth Network(TCP)

node 0 node 1

+----------------+ +----------------+

| ns-3 TCP | | ns-3 TCP |

+----------------+ +----------------+

| 10.1.1.1 | | 10.1.1.2 |

+----------------+ +----------------+

| point-to-point | | point-to-point |

+----------------+ +----------------+

| |

+---------------------+

5 Mbps, 2 ms

'

./waf --run sixth

echo '

###############################

Seventh Network (TCP)

node 0 node 1

+----------------+ +----------------+

| ns-3 TCP | | ns-3 TCP |

+----------------+ +----------------+

| 10.1.1.1 | | 10.1.1.2 |

+----------------+ +----------------+

| point-to-point | | point-to-point |

+----------------+ +----------------+

| |

+---------------------+

5 Mbps, 2 ms

'

./waf --run seventh

rm temp.sh

" > ns-allinone-3.32/ns-3.32/temp.sh

#Enter the commands to run there

echo "

###################

TYPE './temp.sh'

###################

"

cd ns-allinone-3.32/ns-3.32

$SHELL