**­COMPUTER NETWORKS LAB**

**ETCS-354**

**Faculty Name: Ms. Sakshi Jha Student name: AKSHAT AJAY**

**Roll No.: 75714802719**

**Semester: 6th**

**Batch: 6C9**



**Maharaja Agrasen Institute of Technology, PSP Area,**

**Sector – 22, Rohini, New Delhi – 110085**

**INDEX**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Exp. No.** | **Experiment Name** | **Date of Performing** | **Date of checking** | **R1**  **(3)** | **R2**  **(3)** | **R3**  **(3)** | **R4**  **(3)** | **R5**  **(3)** | **Total Marks**  **(15)** | **Signature** |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Exp. No.** | **Experiment Name** | **Date of Performing** | **Date of checking** | **R1**  **(3)** | **R2**  **(3)** | **R3**  **(3)** | **R4**  **(3)** | **R5**  **(3)** | **Total Marks**  **(15)** | **Signature** |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

**EXPERIMENT-1**

**AIM-1:** Introduction to Discrete event simulation tools and its tools.

**Theory:**

A computer network is a system that connects numerous independent computers in order to share information (data) and resources. The integration of computers and other different devices allows users to communicate more easily.

**SYSTEM:**

System is a collection of entities that act and interact together to watch to watch the accomplishment of some logical end.

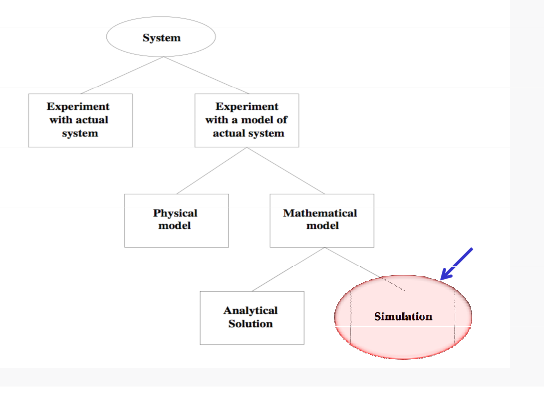
**Discrete System:**

A discrete system is one in which the state variable(s) change only at a discrete set of points 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

E.g. customers arrive at 3:15, 3:23, 4:01, etc.

**Continuous System:**

A continuous system is one in which the state variable(s) change continuously over time. E.g. the amount of water flow over a dam, or an airplane moving through air since state variables position and velocity change continuously with respect to time.



**Ways to study system**

**SIMULATION:**

Network simulation is a technique whereby a software program replicates the behavior of a real network. This is achieved by calculating the interactions between the different network entities such as routers, switches, nodes, access points, links, etc. Simulation is a research field that deals with experimentation of models to make predictions about the behaviour and the performance of actual systems.

*“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:**

A discrete-event simulation (DES) models the operation of a [system](https://en.wikipedia.org/wiki/System) as a ([discrete](https://en.wikipedia.org/wiki/Discrete_time)) [sequence of events](https://en.wikipedia.org/wiki/Sequence_of_events) in time. Each event occurs at a particular instant in time and marks a change of [state](https://en.wikipedia.org/wiki/State_(computer_science)) in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation time can directly jump to the occurrence time of the next event, which is called next-event time progression. It is a method used to model real world systems that can be decomposed into a set of logically separate processes that autonomously progress through time. Each event occurs on a specific process, and is assigned a logical time (a timestamp).

In addition to next-event time progression, there is also an alternative approach, called fixed-increment time progression, where time is broken up into small time slices and the system state is updated according to the set of events/activities happening in the time slice. Because not every time slice has to be simulated, a next-event time simulation can typically run much faster than a corresponding fixed-increment time simulation.

**System:**

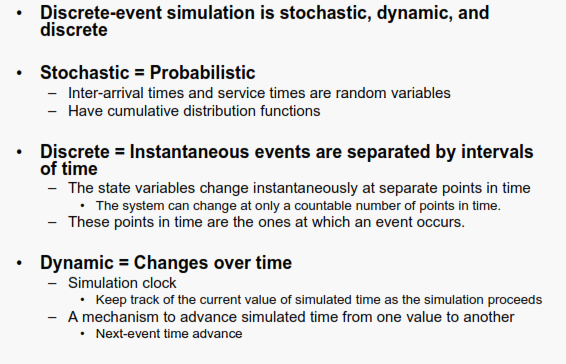
It is composed of objects called entities that have certain properties called attributes.

**State:**

A collection of attributes or state variables that represent the entities of the system.

**Event:**

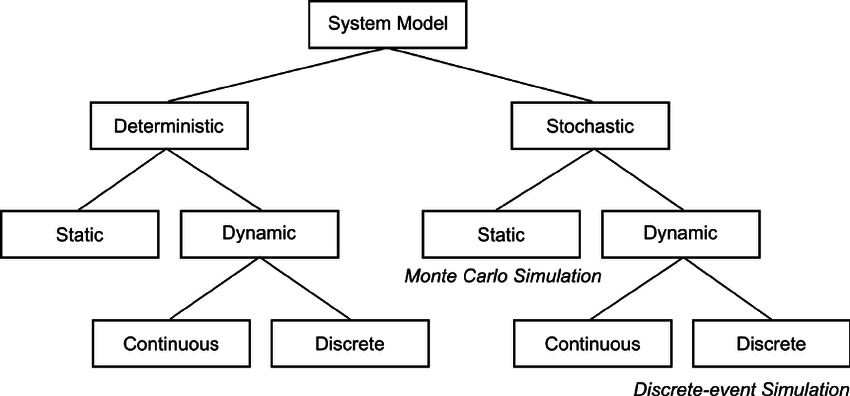
An instantaneous occurrence in time that may alter the state of the system



**Introduction to Discrete Event Simulation**

**MODEL TAXONOMY:**

A *model* is defined as a representation of a system for the purpose of studying the system. Many times one can't experiment with a real system such as a bank, or a highway system. One has to expriment with a *model* of the real system. A model is often not exactly the same as the real system it presents. Rather, it includes a few (or majority of) key aspects of the real system. It is an *abstraction* of the real system.



**Model Taxonomy**

**EXPERIMENT-2**

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

**THEORY:**

**NS3 :** Ns3 (network simulator version 3) has been developed to provide an open, extensible network simulation platform, for networking research and education. It provides models of how packet data networks work and perform and provides a simulation engine for users to conduct simulation experiments. Some of the reasons to use ns-3 include performing 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. 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.

Ns3 scripts are written in C++ code and saved with the extension .c which is built using: ./waf command in the terminal.

Ns-3 provides helper functionality that wraps the low-level tracing system to help with the details involved in configuring some easily understood packet traces. If this functionality is enabled, one will see the output in an ASCII file-thus the name.

**Differences between NS3 and NS2 :**

1. **Programming Languages Supported :**

* NS3 is written using C++ while NS2 is written with the help of TCL and C++.
* Python support is available for scripting language in NS3 while only TCL is used for scripting language in NS2.
* A simulation script can be written in NS3 while it is not possible in NS2.

1. **Memory Management**

* NS3 provides good memory allocation as compared to that in NS2
* NS3 frees the memory that is used to store the packets, while NS2 never reuse the memory until it is terminated.

1. **Packets**

* In NS3 , the Information needed to send through the packet can be added at the header ,trailer, buffer ,etc while The header part of the NS2 includes all the information of header parts in the specified protocol
* The packet of NS3 are composed of single buffer and small tags while the packet of NS2 has headers and data for payload.

1. **Performance**

* Total computation time is less in NS3 as compared to NS2.
* In NS3 , unnecessary parameters storage is prevented by the system while it is not prevented in NS2.
* NS3 performs better than NS2 in terms of memory management

1. **Simulation Output**

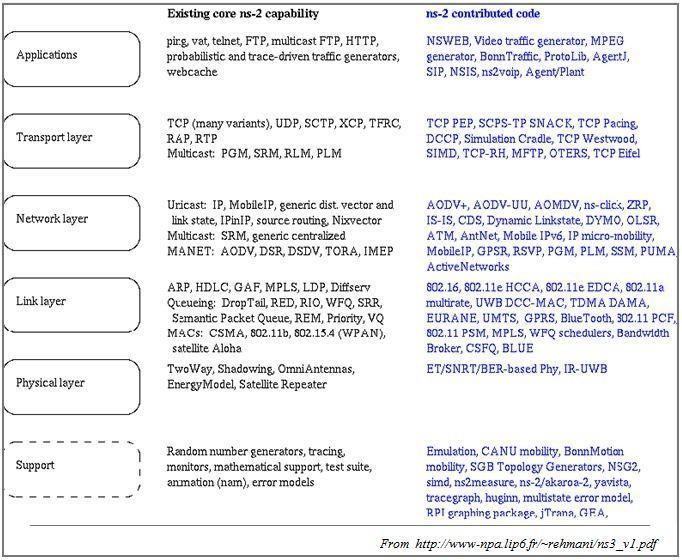
* NS3 employs a package known as PyViz, which is a python based realtime visualization package while NS2 comes with a package called NAM (Network Animator), it's a Tcl based animation system that produces a visual representation of the network described

**NS2 vs NS3:**

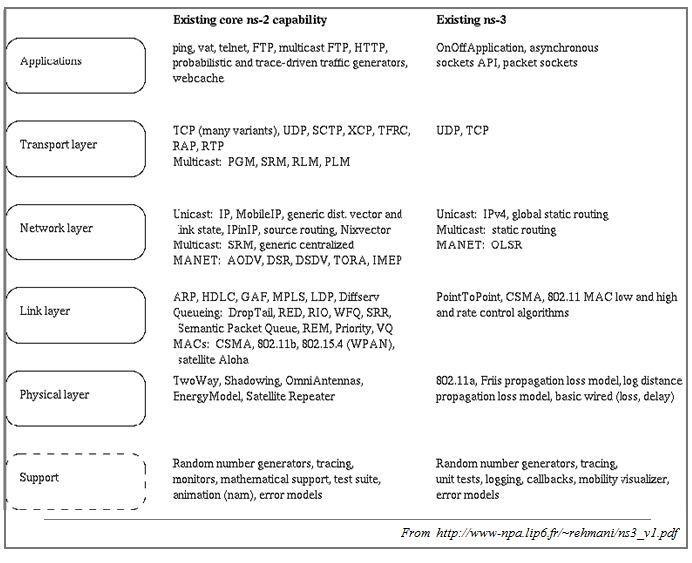
|  |  |  |
| --- | --- | --- |
|  | **NS2** | **NS3** |
| **Programming languages** | ∙NS2 is implemented using combination of oTCL (for scripts describing the network topology) and C++(The core of the simulator).  ∙This system was chosen in the early  1990s to avoid the recompilation of  C++ as it was very time consuming  using the hardware available at that  time, oTCL recompilation takes  less time than C++.  ∙TCL disadvantage: there is overhead introduced with large simulations.  . | ∙NS3isimplemented using C++  ∙With modern hardware capabilities, compilation time was not an issue like for NS2, NS3 can be developed with C++entirely.  ∙A simulation script can be written as  a C++ program, which is not possible  in NS2.  ∙There is a limited support for Python in scripting and visualization. |
| **MemoryManagement** | ∙NS2 requires basic manual C++  memory management functions. | ∙Because NS3 is implemented in C++, all normal C++ memory management functions such as new, delete, malloc, and free are still available.  ∙Automatic de-allocation of objects is supported using reference counting (track number of pointers to an object);this is useful when dealing with Packet objects. |
| **Packets** | ∙A packet consists of 2 distinct  regions; one for headers, and the  second stores payload data.  ∙NS2 never frees memory used to  store packets until the simulation  terminates, it just reuses the  allocated packets repeatedly, as a  result,  the header region of any  packet includes all headers defined  as part of the used protocol even if  that particular packet won't use that  particular header, but just to be  available when this Packet  allocation is reused. | ∙A packet consists of a single buffer of bytes, and optionally a collection of small tags containing meta-data.  ∙The buffer corresponds exactly to the stream of bits that would be sent over are al network.  ∙ Information is added to the packet  by using subclasses; Header, which  adds information to the beginning  of the buffer, Trailer, which adds to  the end.    ∙Unlike NS2, there is generally easy way to determine if a specific header is attached. |
| **Performance** | ∙The total computation time required  to run a simulation scales better in  NS3 than NS2.  ∙This is due to the removal of the  overhead associated with  interfacing oTcl with C++,and the  overhead associated with the oTcl  interpreter. | ∙NS3 performs better than NS2  in terms of memory  management.  ∙The aggregation system  prevents unneeded parameters  from being stored, and packets  don't contain unused reserved  header space. |
| **Simulation output** | ∙NS2 comes with a package called NAM (Network Animator), it's aTcl based animation system that produces a visual representation of the network described. | ∙NS3 employs a package known as PyViz, which is a python based real- time visualization package |

Some Important Points about NS3:

1. NS3isnotbackwardcompatiblewithNS2;it's builtfromthescratchtoreplaceNS2.
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 with NS3 compared to NS2
5. In NS2, bi-language system make 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.



**NS2 contributed code**



**NS2 and NS3 existing core capabilities (in July 2010)**

**EXPERIMENT-3**

**AIM:** Installation of NS3 simulator.

**THEORY:**

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

3. valgrind gsl-bin libgsl0-dev libgsl0ldbl  Network Simulation Cradle (nsc): flex bison

   Reading pcap packet traces: tcpdump

4. Database support for statistics framework: sqlite sqlite3

5. Xml-based version of the config store: libxml2

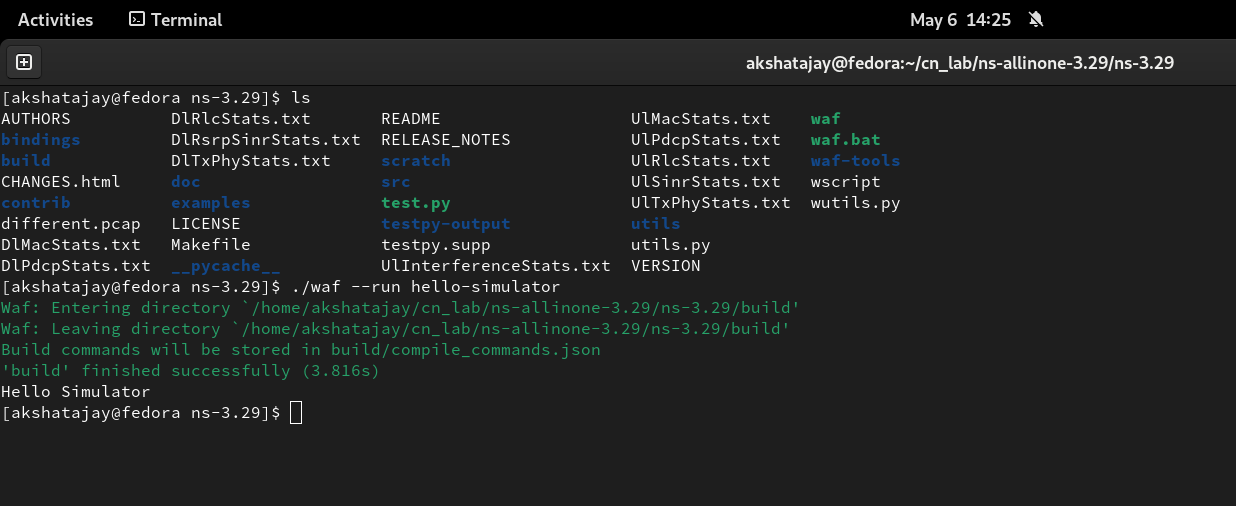
6. A GTK-based configuration system: libgtk2.0-0

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 texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlivelatex-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

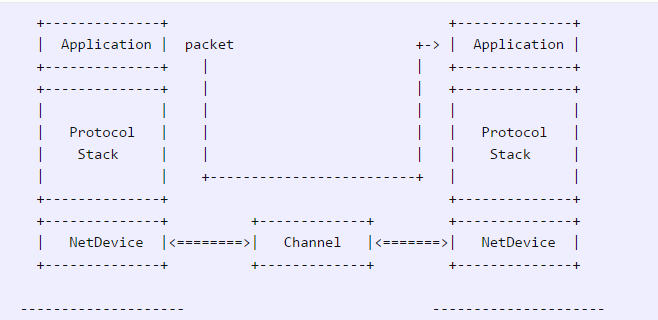
**SCREENSHOT**



**EXPERIMENT-4**

**AIM:** Program in NS3 to connect two nodes.

**THEORY:**



**Node**

Because in any network simulation, we will need nodes. So ns-3 comes with [NodeContainer](https://www.nsnam.org/doxygen/classns3_1_1_node_container.html) 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 usingPointToPointHelper 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 UdpEchoClientfor 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;

**SOURCE CODE:**

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE("FirstScriptExample");

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

{

CommandLine cmd;

cmd.Parse(argc, argv);

Time::SetResolution(Time::NS);

LogComponentEnable("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

LogComponentEnable("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

NodeContainer nodes;

nodes.Create(2);

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer devices;

devices = pointToPoint.Install(nodes);

InternetStackHelper stack;

stack.Install(nodes);

Ipv4AddressHelper address;

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

Ipv4InterfaceContainer interfaces = address.Assign(devices);

UdpEchoServerHelper echoServer(9);

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

serverApps.Start(Seconds(1.0));

serverApps.Stop(Seconds(10.0));

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));

Simulator::Run();

Simulator::Destroy();

return 0;

}

**Output :**

Graphical user interface, text

Description automatically generated

**EXPERIMENT-5**

**AIM:** Program in NS3 for connecting three nodes considering one node as a central node.

**Theory:**

#include "ns3/core-module.h"

# include "ns3/core-netanim.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("FirstScriptExample");

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

{

Time::SetResolution (Time::NS);

LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

NodeContainer nodes;

nodes.Create (3);

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer devices, devices1;

devices = pointToPoint.Install (nodes.Get(0),nodes.Get(1));

devices1 = pointToPoint.Install (nodes.Get(2),nodes.Get(1));

InternetStackHelper stack;

stack.Install (nodes);

Ipv4AddressHelper address;

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

Ipv4InterfaceContainer interfaces = address.Assign (devices);

Ipv4InterfaceContainer interfaces1 = address.Assign (devices1);

UdpEchoServerHelper echoServer (90);

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

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 90);

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));

UdpEchoClientHelper echoClient1 (interfaces1.GetAddress (1), 90);

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps1 = echoClient1.Install (nodes.Get (2));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

AnimationInterface anim(“threenode.xml”);

Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**Output:**

Graphical user interface

Description automatically generated

**EXPERIMENT-6**

**AIM:** Installation and configuration of NetAnim

**Theory:**

# Installing NetAnim

The website:

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

1. Install Mercurial:

apt−get/dnf install mercurial

1. Install QT4 development package:

apt−get/dnf install qt4−dev−tools

1. You can use Synaptic too, to install both the above packages.
2. Download NetAnim: hg clone http :// code .nsnam. org/netanim
3. Build NetAnim:

cd netanim

make clean

qmake NetAnim.

pro make

# Compiling code with NetAnim

So you will have to 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 file 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-3.29/ns-3.29/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-3.29/ns-3.29/scratch/).
4. Run the code using the command:

./ waf −−run <filename>

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

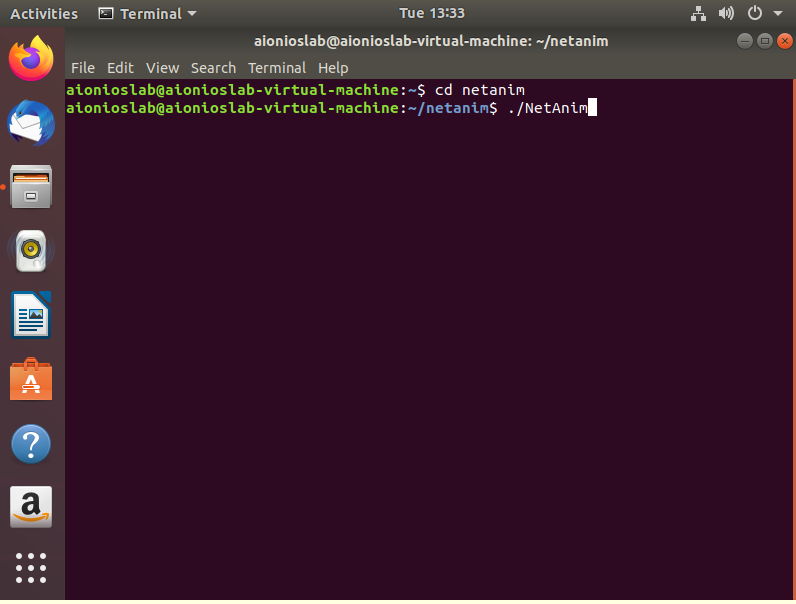
**To visualize on NetAnim:**

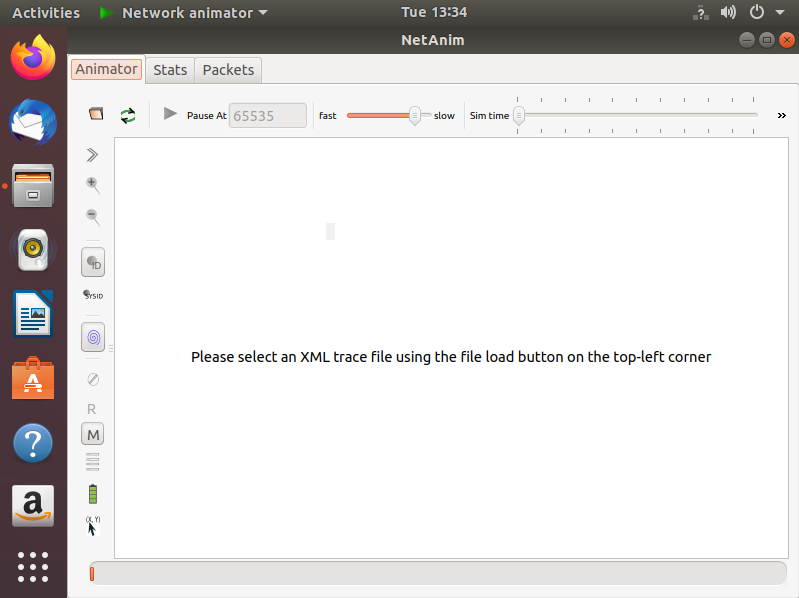
1. cd to the netanim folder (cd ~/netanim/).
2. Run Netanim:

./NetAnim

1. Include the .xml le generated in the ns-3.29 folder

**OUTPUT:**

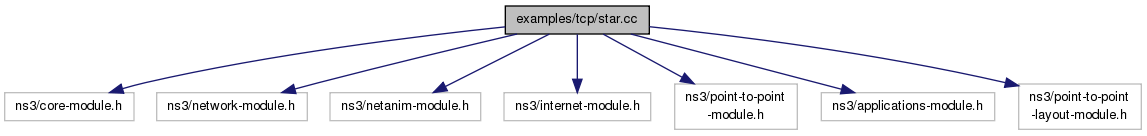




**EXPERIMENT-7**

**Aim:** Program in NS3 to implement star topology

**Theory:**

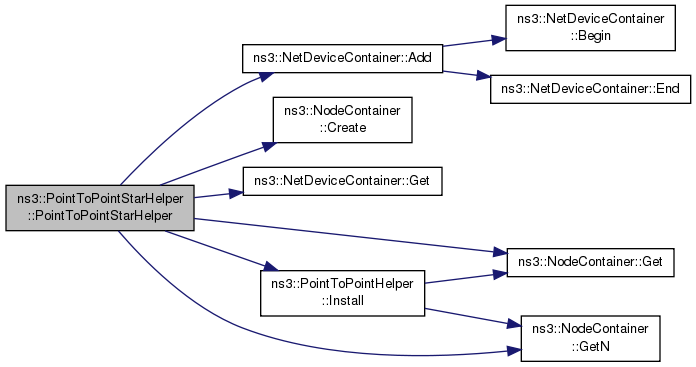


**ns3::PointToPointStarHelper Class Reference**

* A helper to make it easier to create a star topology with PointToPoint links.
* Create a [**PointToPointStarHelper**](https://www.nsnam.org/doxygen/classns3_1_1_point_to_point_star_helper.html) 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:**



Member Functions:

***Function 1:***

**void ns3::PointToPointStarHelper::AssignIpv4Addresses(**[**Ipv4AddressHelper**](https://www.nsnam.org/doxygen/classns3_1_1_ipv4_address_helper.html)**address)**

**Parameters**

|  |  |
| --- | --- |
| **address** | an **Ipv4AddressHelper** which is used to install [**Ipv4**](https://www.nsnam.org/doxygen/classns3_1_1_ipv4.html) addresses on all the node interfaces in the star |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Function 2:***  **void ns3::PointToPointStarHelper::InstallStack** | **(** | [**InternetStackHelper**](https://www.nsnam.org/doxygen/classns3_1_1_internet_stack_helper.html) | **stack** | **)** |  |

**Parameters**

|  |  |
| --- | --- |
| **stack** | an [**InternetStackHelper**](https://www.nsnam.org/doxygen/classns3_1_1_internet_stack_helper.html) 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

**Code :**

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/netanim-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

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

// Network topology (default)

//

// n2 n3 n4 .

// \ | / .

// \|/ .

// n1--- n0---n5 .

// /|\ .

// / | \ .

// n8 n7 n6 .

//

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE("Star");

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

{

Time::SetResolution(Time::NS);

LogComponentEnable("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

LogComponentEnable("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

//

// Set up some default values for the simulation.

//

Config::SetDefault("ns3::OnOffApplication::PacketSize", UintegerValue(137));

// ??? try and stick 15kb/s into the data rate

Config::SetDefault("ns3::OnOffApplication::DataRate", StringValue("14kb/s"));

//

// Default number of nodes in the star. Overridable by command line argument.

//

uint32\_t nSpokes = 8;

CommandLine cmd;

cmd.AddValue("nSpokes", "Number of nodes to place in the star", nSpokes);

cmd.Parse(argc, argv);

NS\_LOG\_INFO("Build star topology.");

PointToPointHelper pointToPoint;

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

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

PointToPointStarHelper star(nSpokes, pointToPoint);

NS\_LOG\_INFO("Install internet stack on all nodes.");

InternetStackHelper internet;

star.InstallStack(internet);

NS\_LOG\_INFO("Assign IP Addresses.");

star.AssignIpv4Addresses(Ipv4AddressHelper("10.1.1.0", "255.255.255.0"));

NS\_LOG\_INFO("Create applications.");

//

// Create a packet sink on the star "hub" to receive packets.

//

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));

//

// Create OnOff applications to send TCP to the hub, one on each spoke node.

//

OnOffHelper onOffHelper("ns3::TcpSocketFactory", Address());

onOffHelper.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));

onOffHelper.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));

ApplicationContainer spokeApps;

for (uint32\_t i = 0; i < star.SpokeCount(); ++i)

{

AddressValue remoteAddress(InetSocketAddress(star.GetHubIpv4Address(i), port));

onOffHelper.SetAttribute("Remote", remoteAddress);

spokeApps.Add(onOffHelper.Install(star.GetSpokeNode(i)));

}

spokeApps.Start(Seconds(1.0));

spokeApps.Stop(Seconds(10.0));

NS\_LOG\_INFO("Enable static global routing.");

//

// Turn on global static routing so we can actually be routed across the star.

//

Ipv4GlobalRoutingHelper::PopulateRoutingTables();

NS\_LOG\_INFO("Enable pcap tracing.");

//

// Do pcap tracing on all point-to-point devices on all nodes.

//

pointToPoint.EnablePcapAll("star");

NS\_LOG\_INFO("Run Simulation.");

Simulator::Run();

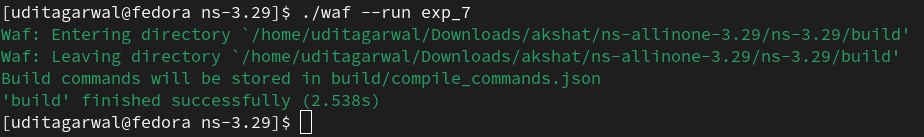
Simulator::Destroy();

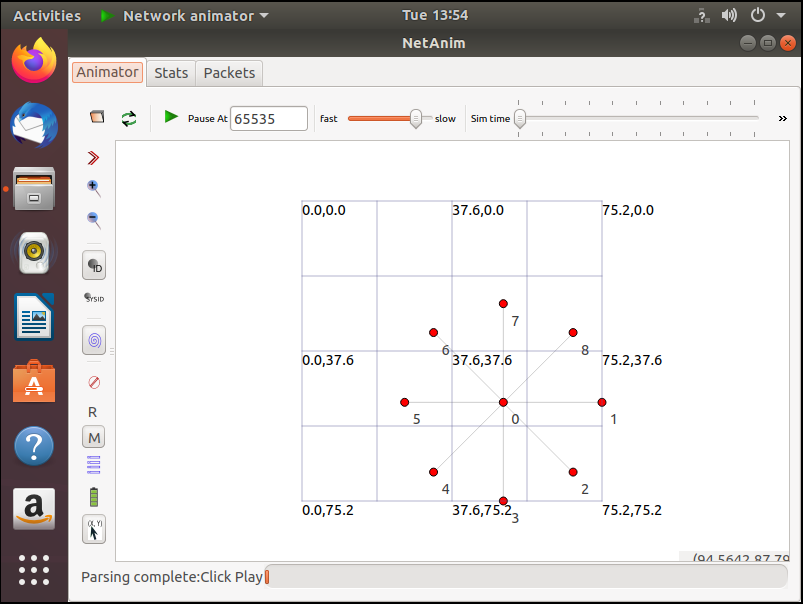
NS\_LOG\_INFO("Done.");

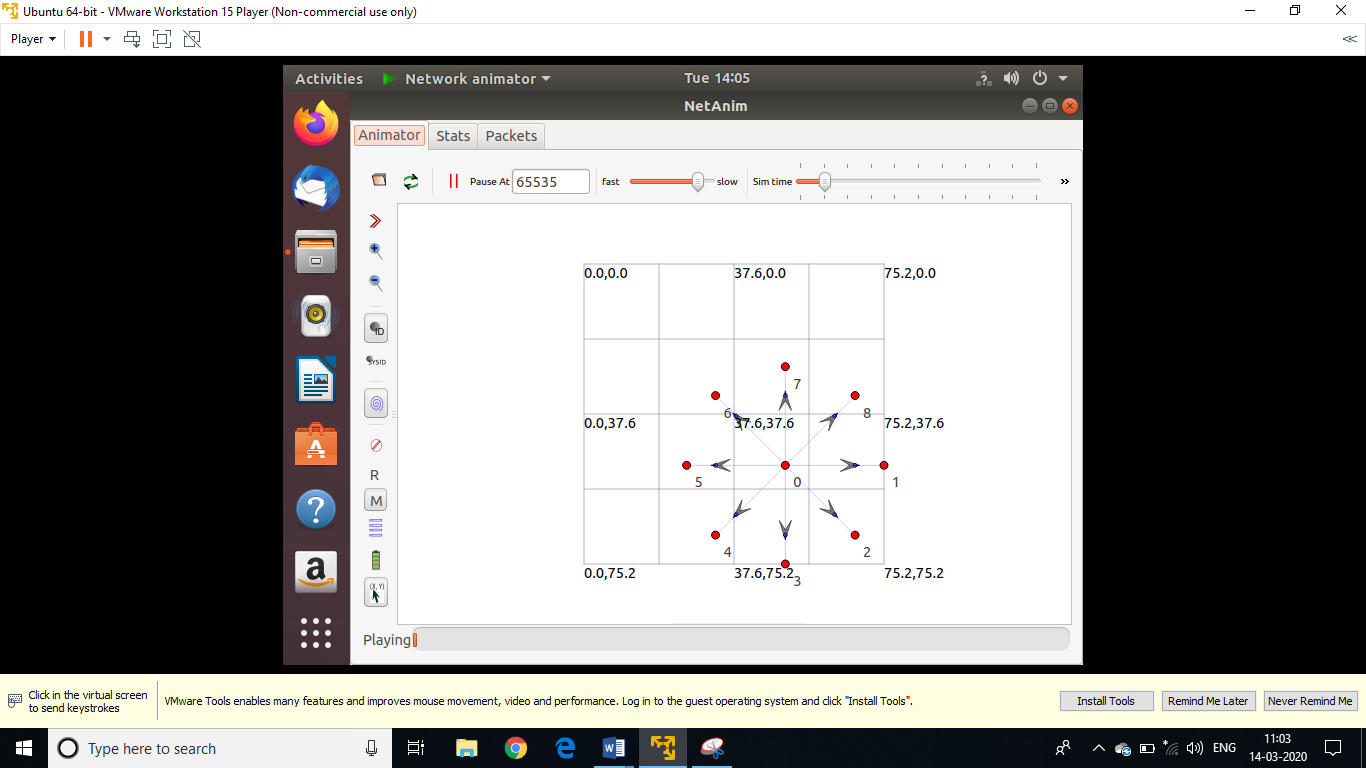
return 0;

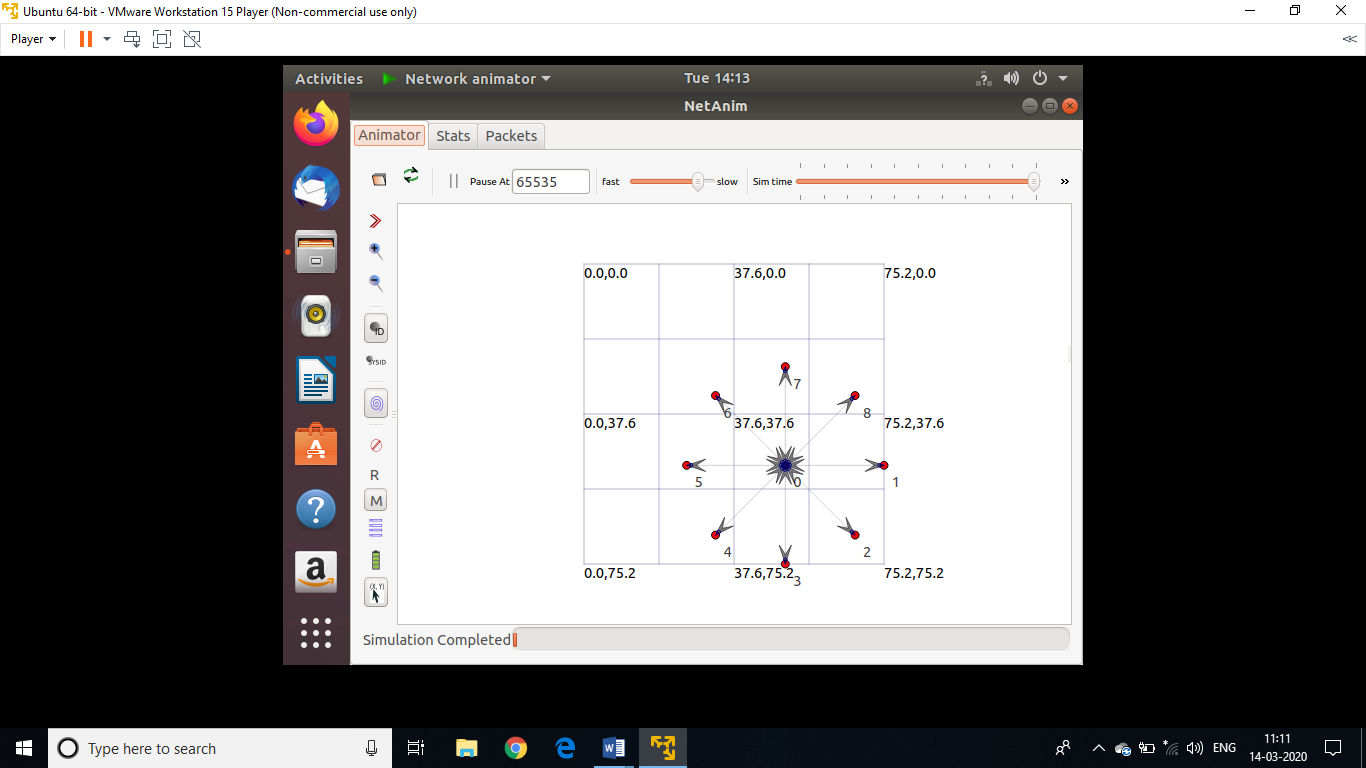
}

**Output :**

****







**EXPERIMENT-8**

**Aim:** Using ns3, design and implement a bus topology using CSMA.

**Theory:**

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.

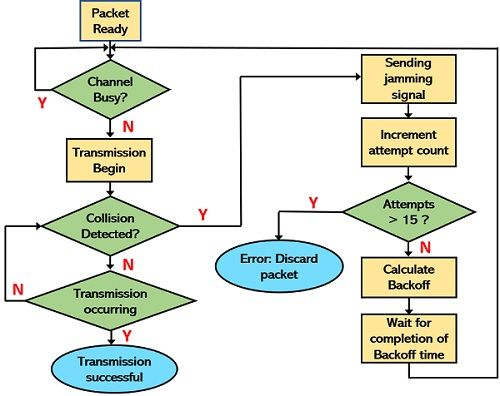


Figure 1: Flowchart of CSMA/CD PROTOCOL

**Code :**

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

#include "ns3/ipv4-global-routing-helper.h"

// Default Network Topology

// 10.1.1.0

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

// point-to-point | | | |

// ================

// LAN 10.1.2.0

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE("SecondScriptExample");

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

{

bool verbose = true;

uint32\_t nCsma = 3;

CommandLine cmd;

cmd.AddValue("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);

cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);

cmd.Parse(argc, argv);

if (verbose)

{

LogComponentEnable("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

LogComponentEnable("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

}

nCsma = nCsma == 0 ? 1 : nCsma;

NodeContainer p2pNodes;

p2pNodes.Create(2);

NodeContainer csmaNodes;

csmaNodes.Add(p2pNodes.Get(1));

csmaNodes.Create(nCsma);

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install(p2pNodes);

CsmaHelper csma;

csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));

csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install(csmaNodes);

InternetStackHelper stack;

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

stack.Install(csmaNodes);

Ipv4AddressHelper address;

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

Ipv4InterfaceContainer p2pInterfaces;

p2pInterfaces = address.Assign(p2pDevices);

address.SetBase("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer csmaInterfaces;

csmaInterfaces = address.Assign(csmaDevices);

UdpEchoServerHelper echoServer(9);

ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));

serverApps.Start(Seconds(1.0));

serverApps.Stop(Seconds(10.0));

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));

clientApps.Start(Seconds(2.0));

clientApps.Stop(Seconds(10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables();

pointToPoint.EnablePcapAll("second");

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

Simulator::Run();

Simulator::Destroy();

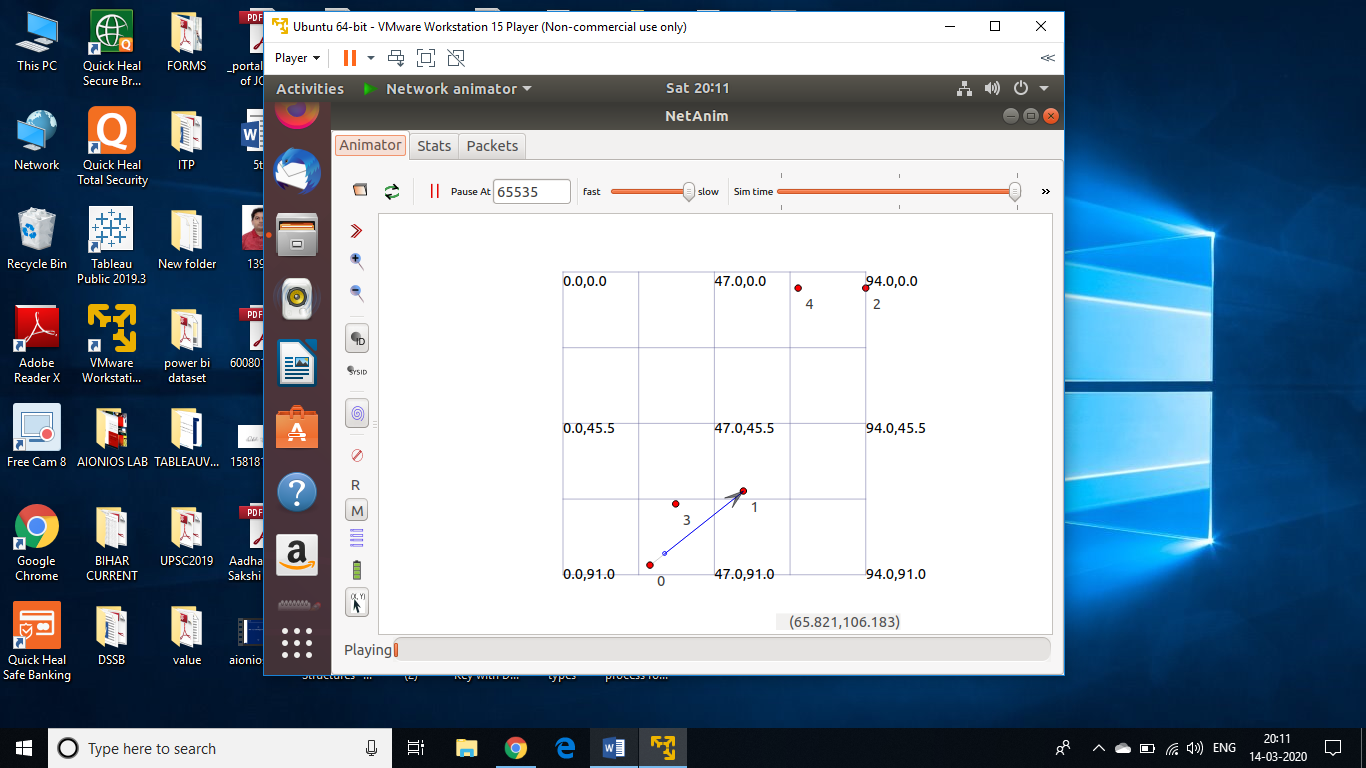
return 0;

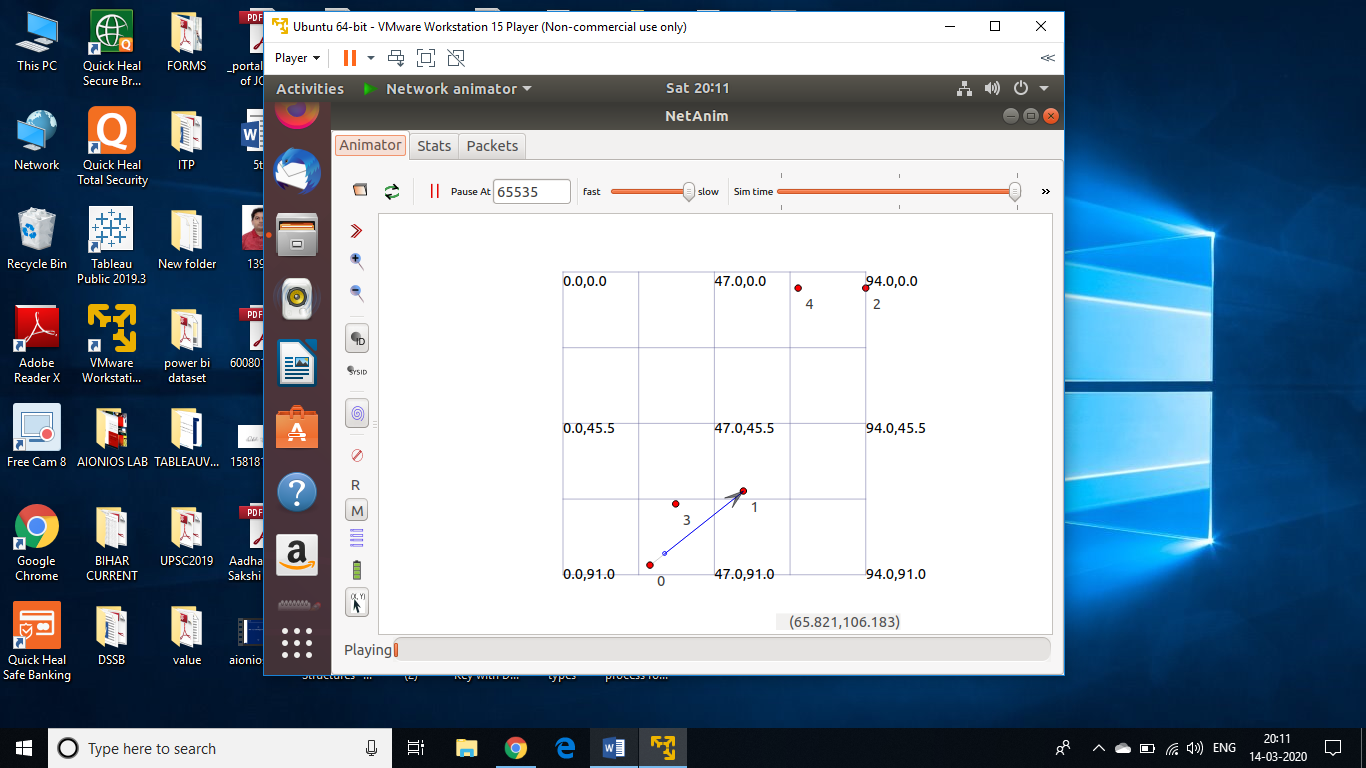
}

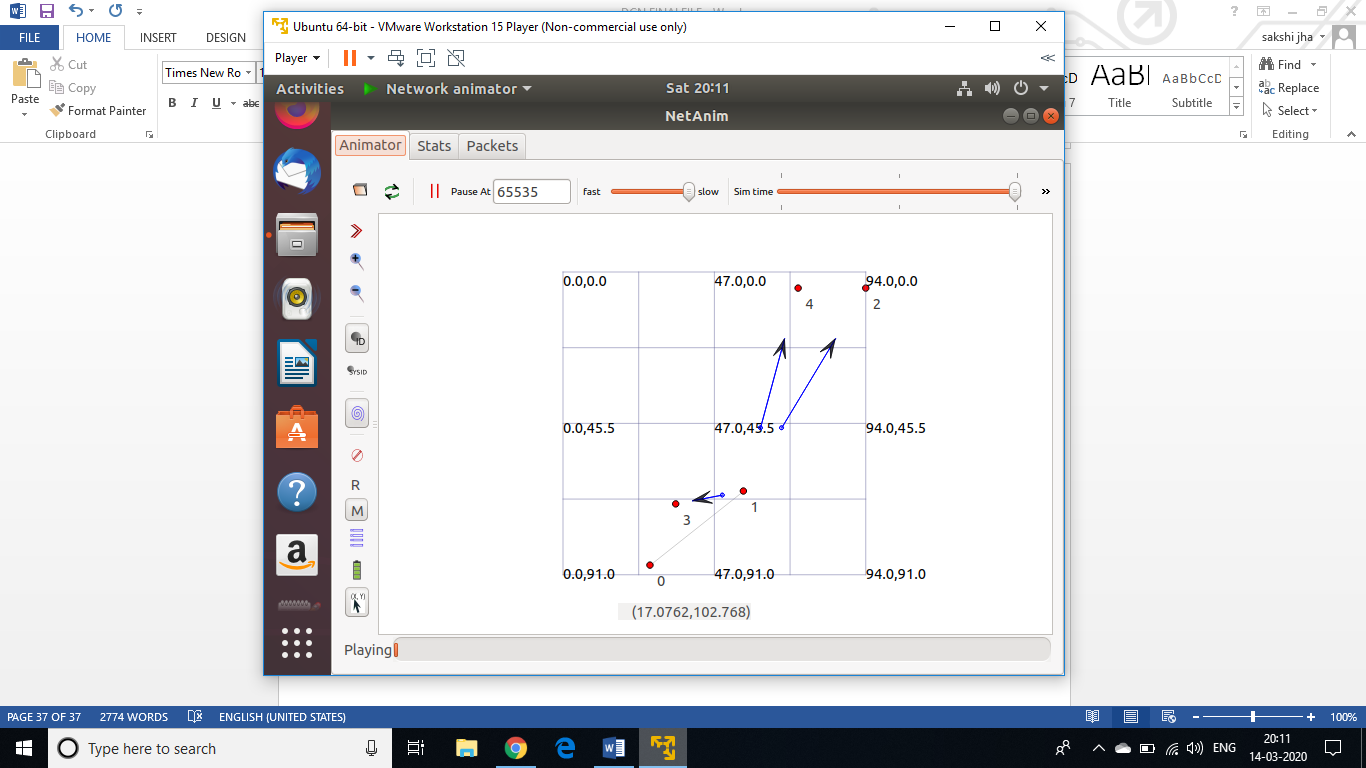
**Output :**

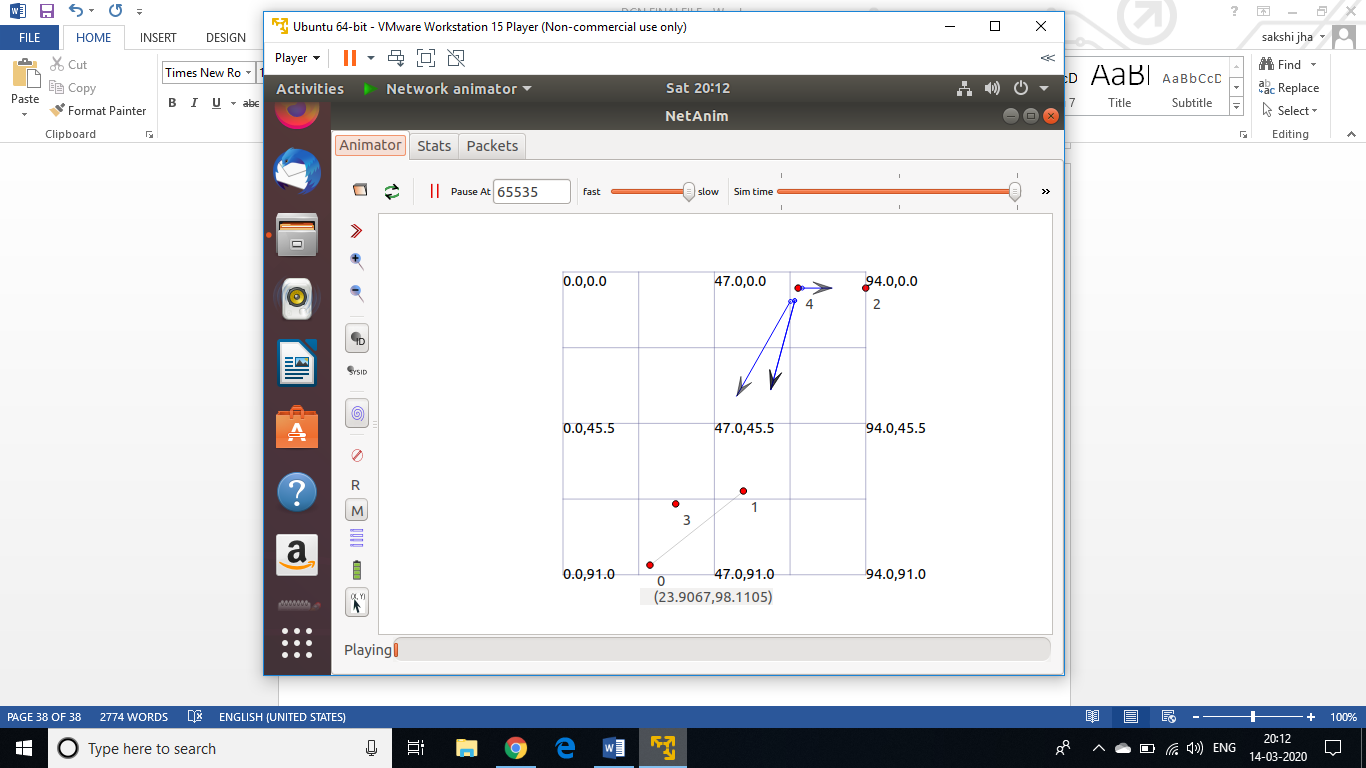
Text

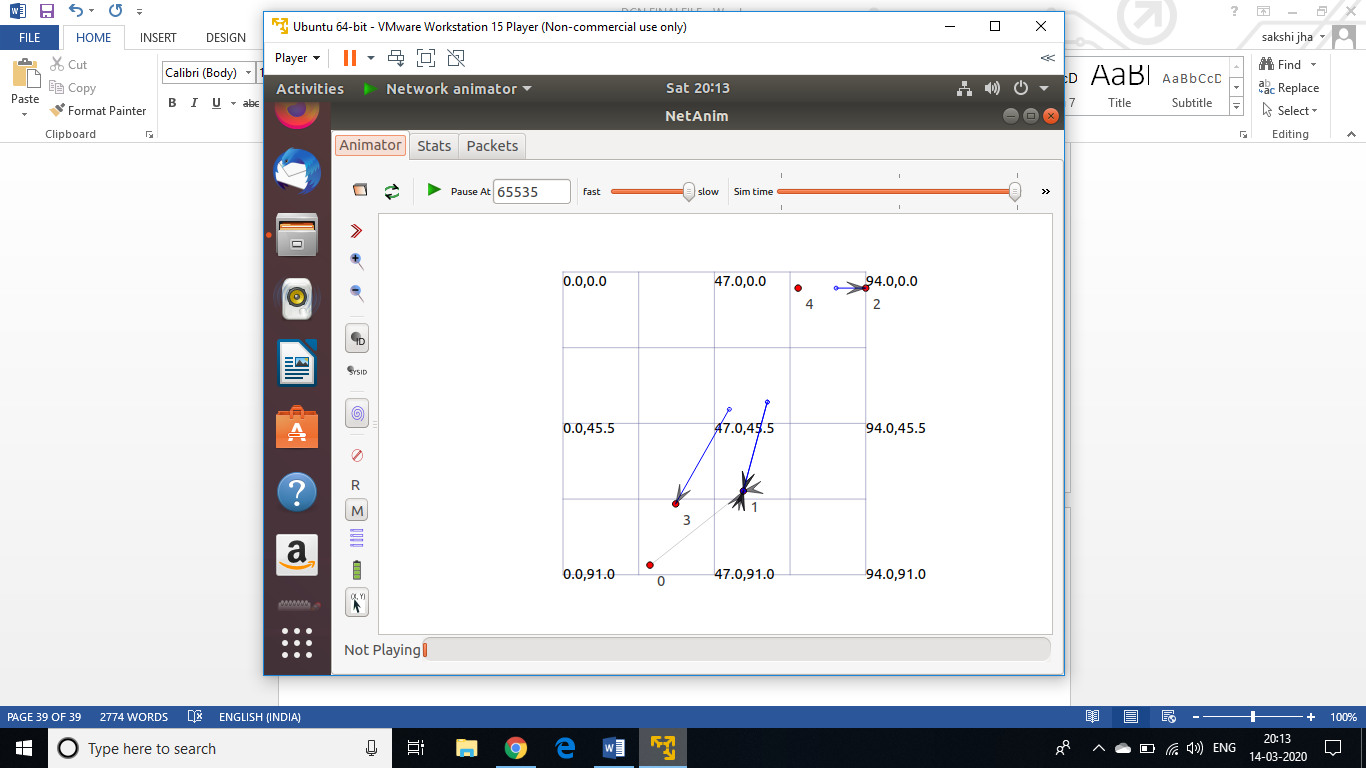
Description automatically generated with medium confidence

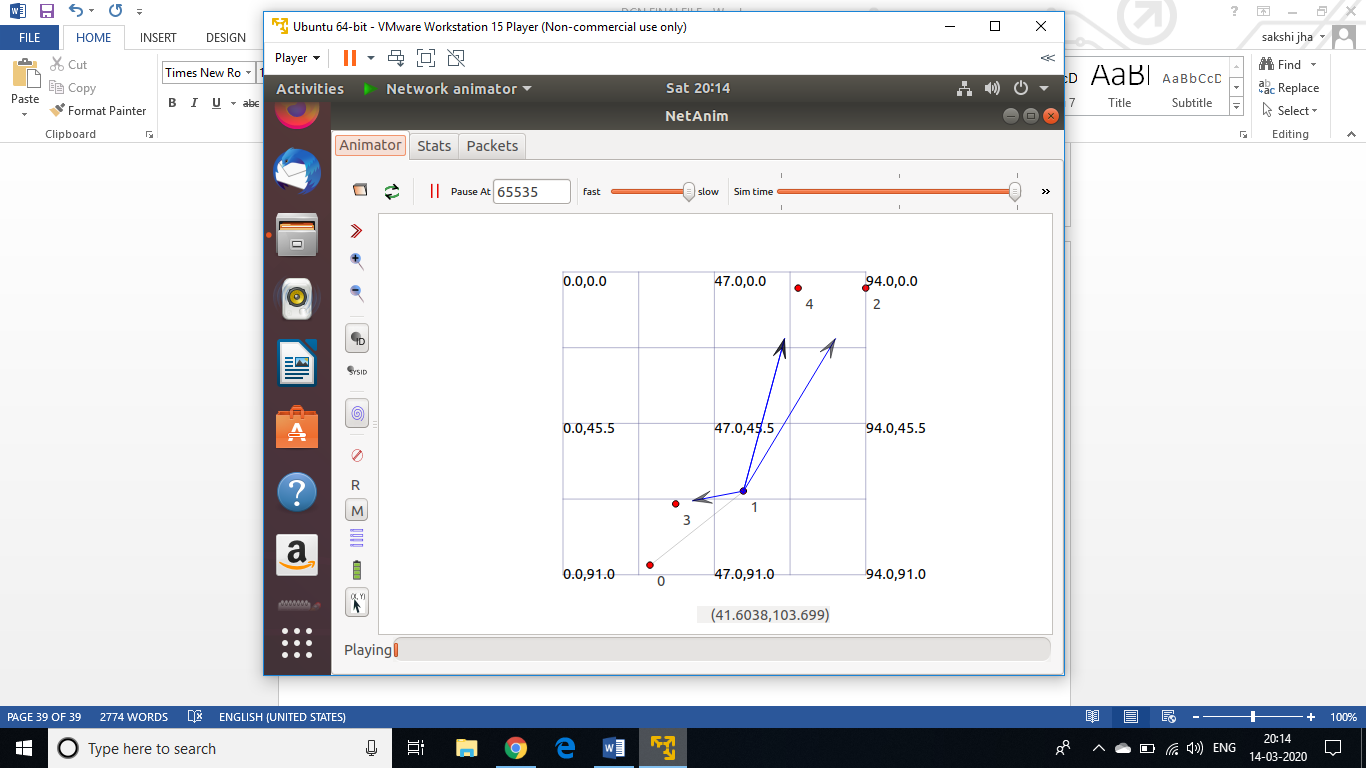


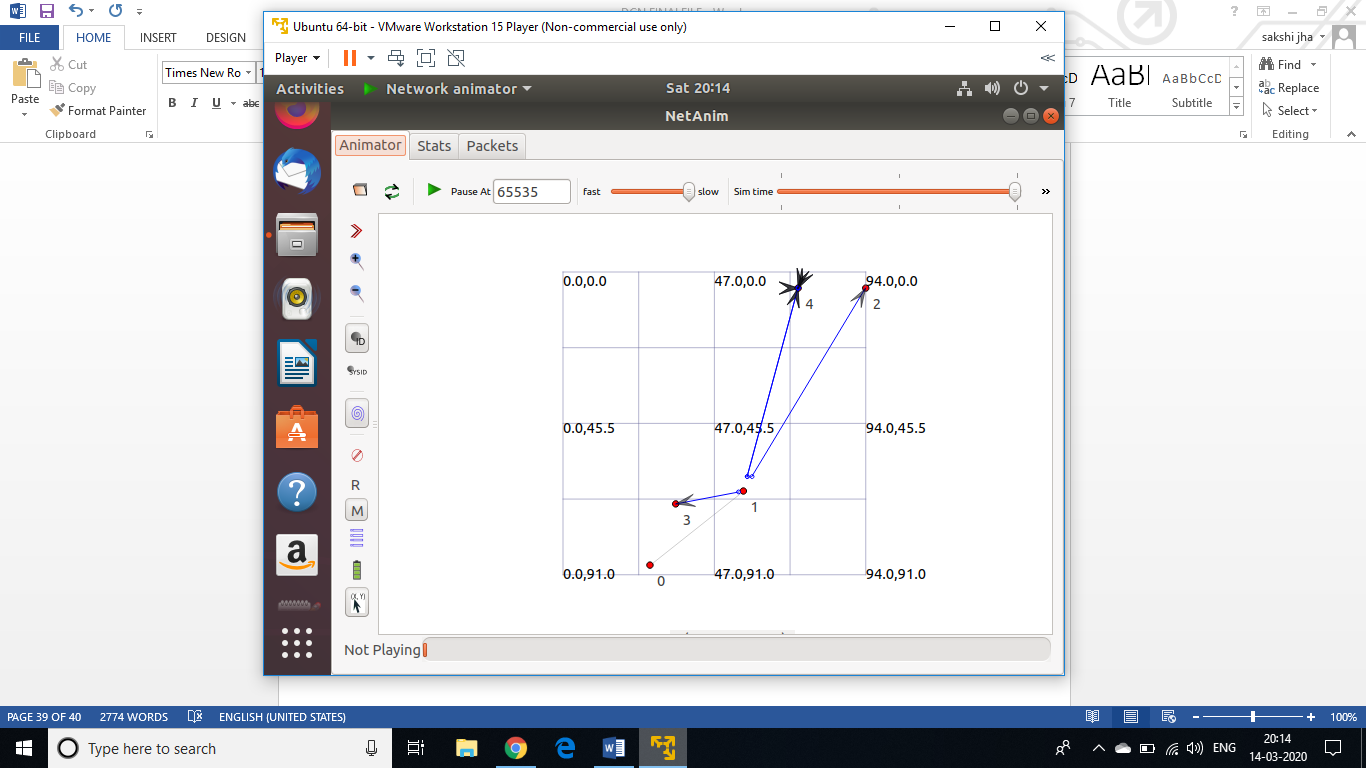


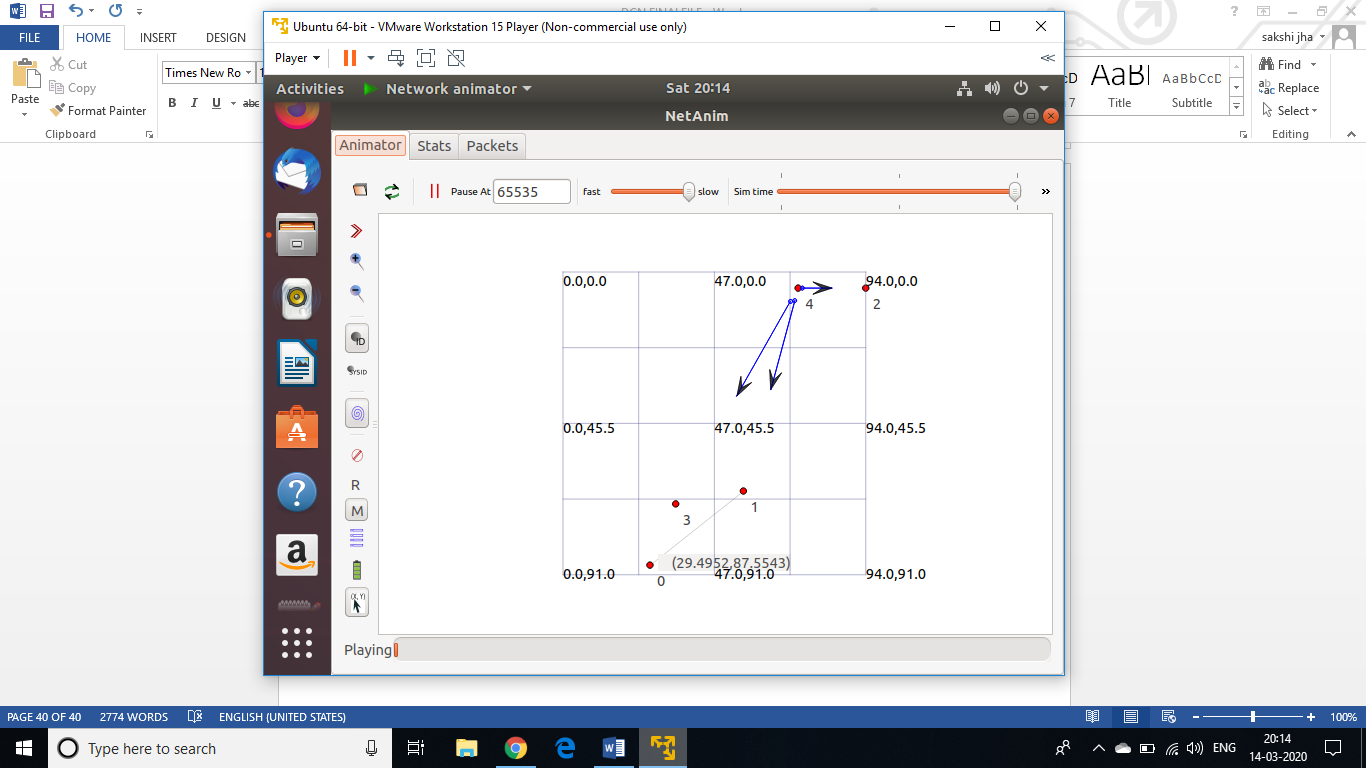


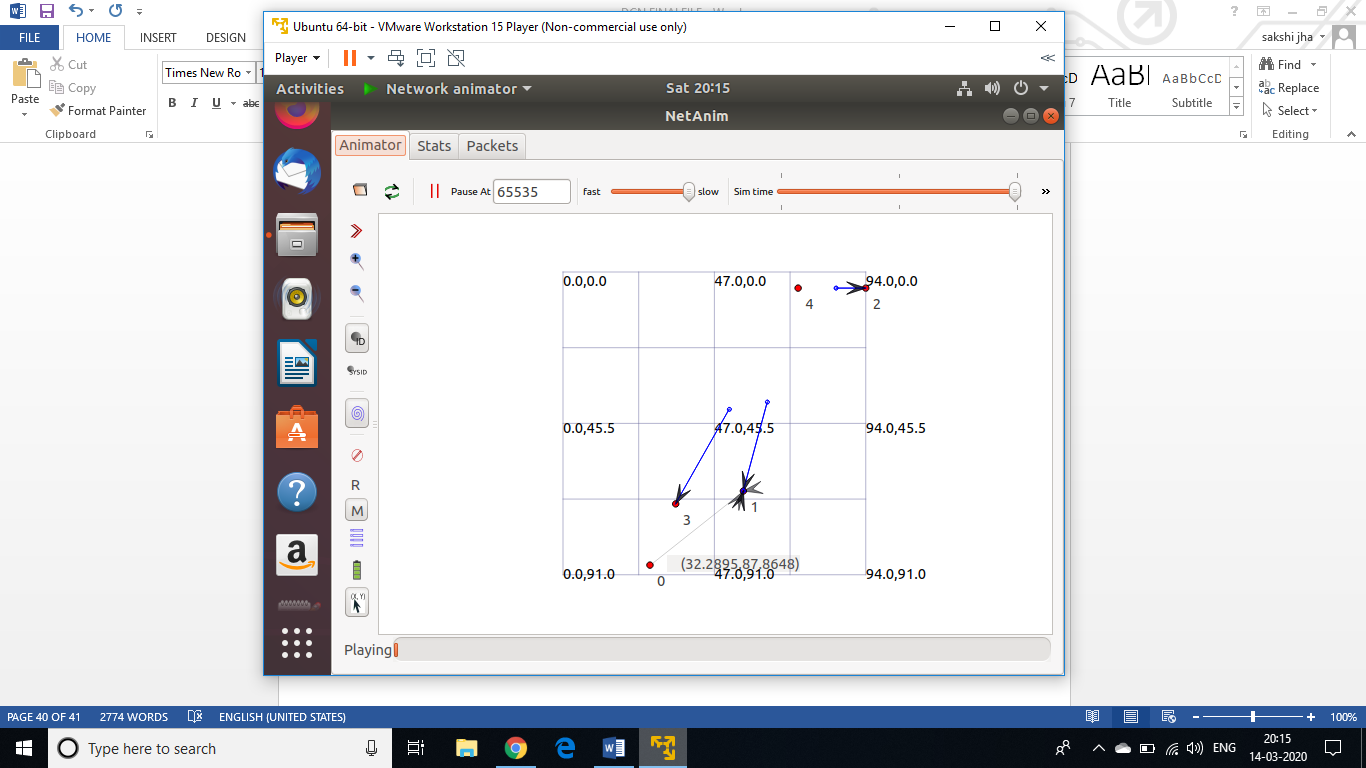


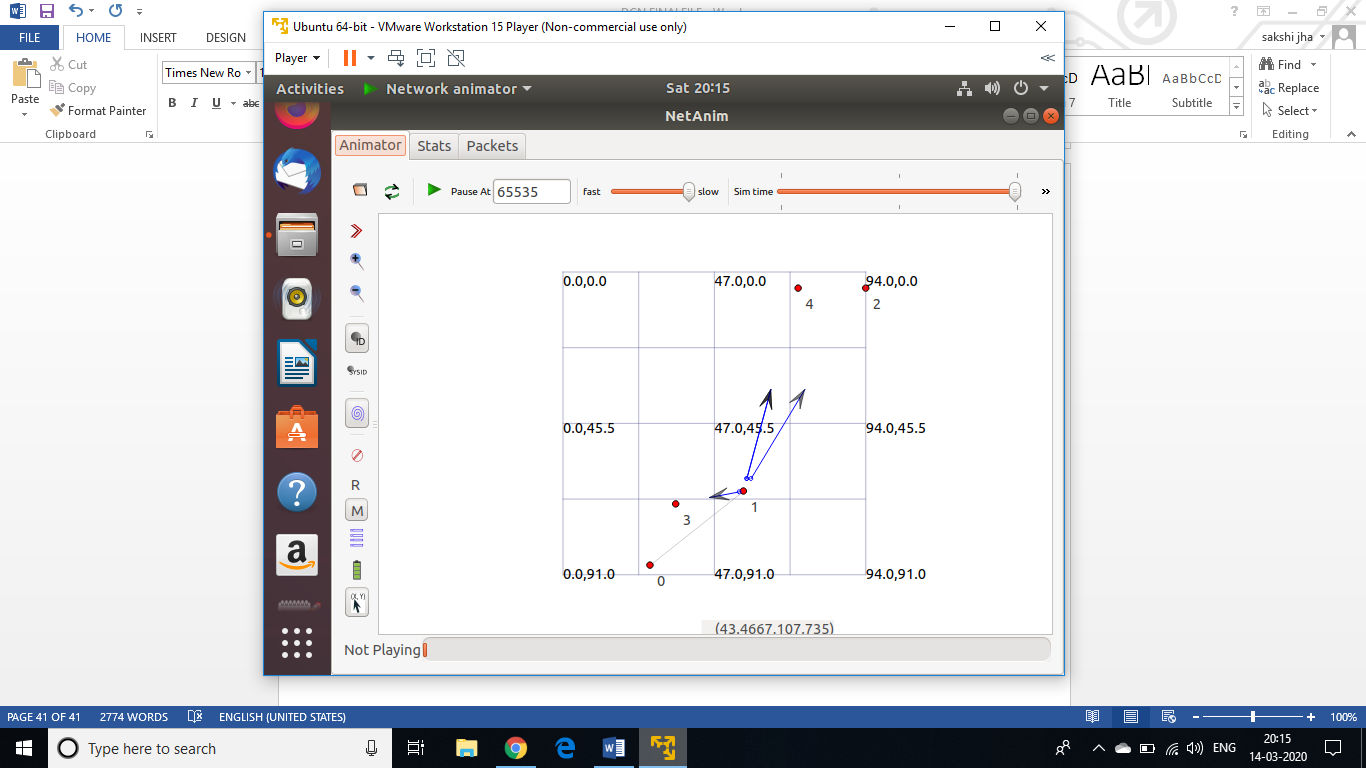


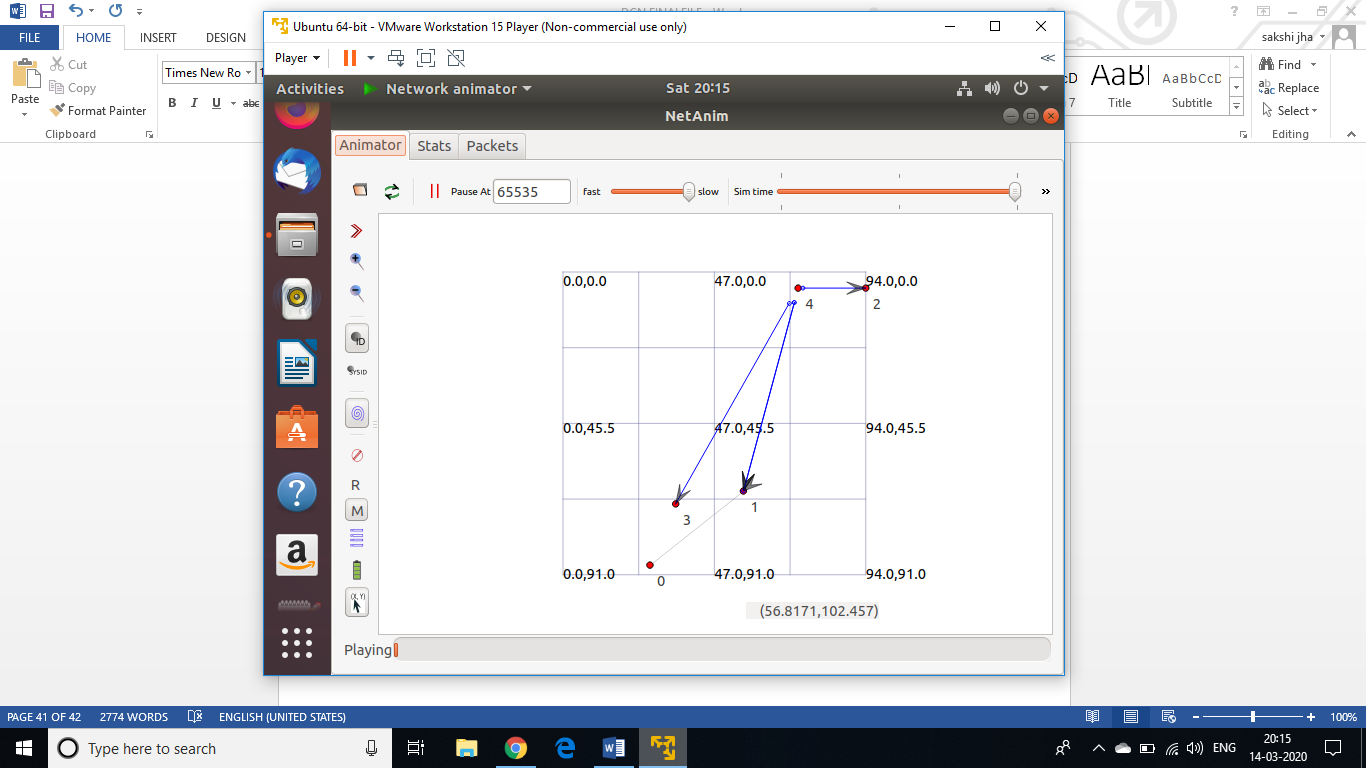


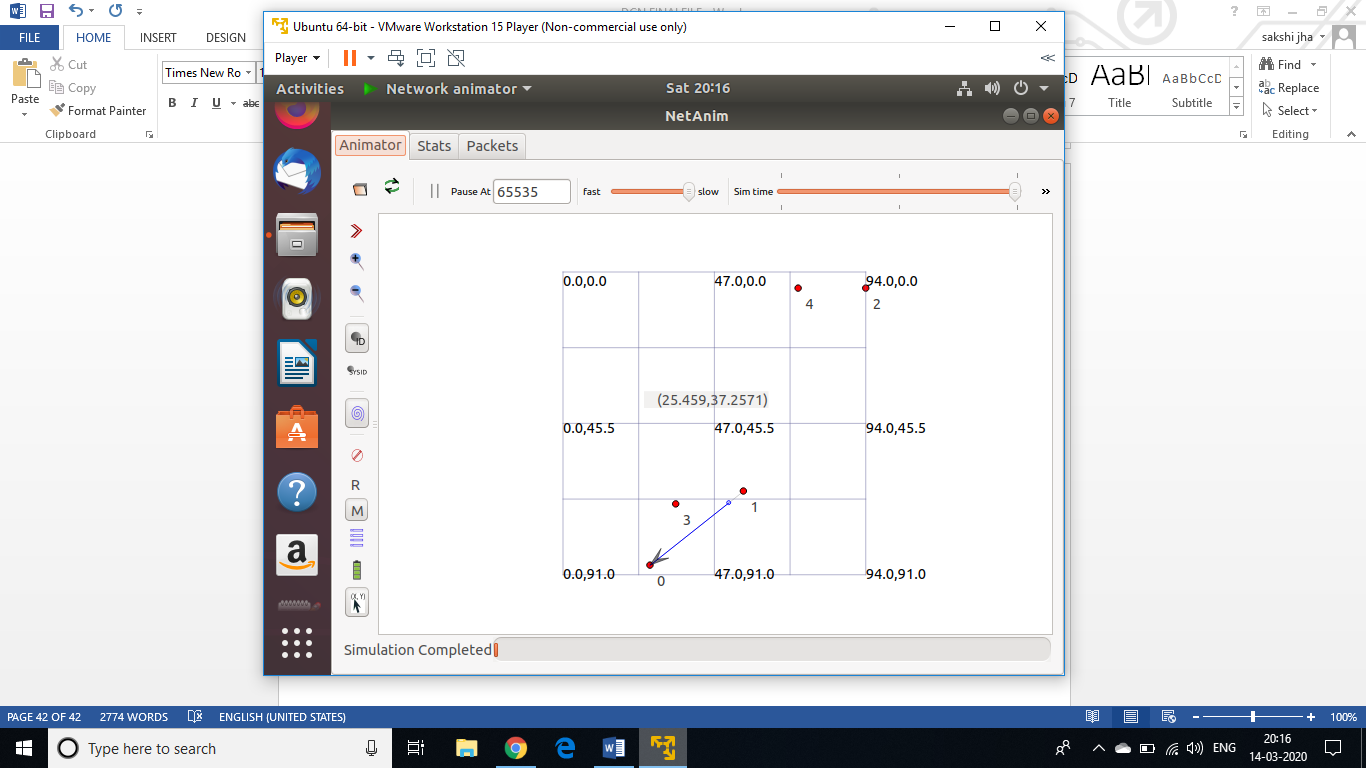












**EXPERIMENT-9**

**Aim:** Using ns3, design and implement hybrid topology connecting multiple routers and nodes.

**Theory :**

**Code :**

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

#include "ns3/ipv4-global-routing-helper.h"

// Default Network Topology 10.1.5.0

// r2---------n1

// / 10.1.3.0

// no----------r0--------r1

// 10.1.1.0 10.1.2.0 \ 10.1.4.0

// r3

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("SecondScriptExample");

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

{

bool verbose = true;

if (verbose)

{

LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

}

NodeContainer host, router, host1;

host.Create (2);

router.Create (4);

NodeContainer subnet1;

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

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

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer subnet1Devices;

subnet1Devices = pointToPoint.Install (subnet1);

InternetStackHelper stack;

stack.Install (router);

stack.Install (host);

Ipv4AddressHelper address1, address2, address3, address4, address5, address6,;

Address1.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer subnet1Interfaces;

subnet1Interfaces = address1.Assign (subnet1Devices);

NodeContainer subnet2;

subnet2.Add (router.Get (0));

subnet2.Add (router.Get (1));

NetDeviceContainer subnet2Devices;

subnet2Devices = pointToPoint.Install (subnet2);

address2.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer subnet2Interfaces;

subnet2Interfaces = address2.Assign (subnet2Devices);

NodeContainer subnet3;

subnet3.Add (router.Get (1));

subnet3.Add (router.Get (2));

NetDeviceContainer subnet3Devices;

subnet3Devices = pointToPoint.Install (subnet3);

address3.SetBase ("10.1.3.0", "255.255.255.0");

Ipv4InterfaceContainer subnet3Interfaces;

subnet3Interfaces = address3.Assign (subnet3Devices);

NodeContainer subnet4;

subnet4.Add (router.Get (1));

subnet4.Add (router.Get (3));

NetDeviceContainer subnet4Devices;

subnet4Devices = pointToPoint.Install (subnet4);

address4.SetBase ("10.1.4.0", "255.255.255.0");

Ipv4InterfaceContainer subnet4Interfaces;

subnet4Interfaces = address4.Assign (subnet4Devices);

NodeContainer subnet5;

subnet5.Add (router.Get (2));

subnet5.Add (host.Get (1));

NetDeviceContainer subnet5Devices;

subnet5Devices = pointToPoint.Install (subnet5);

address5.SetBase ("10.1.5.0", "255.255.255.0");

Ipv4InterfaceContainer subnet5Interfaces;

subnet5Interfaces = address5.Assign (subnet5Devices);

UdpEchoServerHelper echoServer (9);

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

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (subnet5Interfaces.GetAddress (1), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (3));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (subnet1.Get (0));

clientApps.Start (Seconds (1.0));

clientApps.Stop (Seconds (10.0));

ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**Output :**

