**COMPUTER NETWORKS LAB**

**(ETCS – 354)**

**Faculty name:** Mrs. SAURABH RASTOGI **Student name:** Ayush Pandey

**Roll No.:** 45014802718

**Semester:** 6th Semester

**Group:** 6-C-9



Maharaja Agrasen Institute of Technology

PSP Area, Sector – 22, Rohini, New Delhi – 110085



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**VISION**

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

**MISSION**

**The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:**

**Engineering Hardware – Software Symbiosis**

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

**Life – Long Learning**

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

**Liberalization and Globalization**

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

**Diversification**

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

**Entrepreneurship**

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**

**VISION**

To produce “Critical Thinkers of Innovative Technology”.

**MISSION**

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real-life problems and fulfil societal needs.

**PRACTICAL RECORD**

**PAPER CODE : ETCS-354**

**Name of the student : Ayush Pandey**

**University Roll No. : 45014802718**

**Branch : CSE**

**Group : C-9**

**PRACTICAL DETAILS**

A) Experiments according to CN lab syllabus prescribed by GGSIPU

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp. No. | Experiment Name | Date of performance | Date of checking | R1 | R2 | R3 | R4 | R5 | TOTAL |
| 1.(a) | Discuss about event discrete simulation and its tools. | 16/03/2021 |  |  |  |  |  |  |  |
| 1.(b) | Installation of NS3 simulator and discuss all its steps. | 16/03/2021 |  |  |  |  |  |  |  |
| 2. | Introduction to NS3 and its comparison with NS2. | 23/03/2021 |  |  |  |  |  |  |  |
| 3. | Write a program in NS3 to connect 2 nodes. | 08/04/2021 |  |  |  |  |  |  |  |
| 4. | Write a program in NS3 to connect 3 nodes. | 15/04/2021 |  |  |  |  |  |  |  |
| 5. | Write a program in NS3 to implement Bus Topology. | 22/04/2021 |  |  |  |  |  |  |  |
| 6. | Write a program in NS3 to implement Star Topology. | 29/04/2021 |  |  |  |  |  |  |  |
| 7. | Write a program in NS3 to implement hybrid topology. | 20/05/2021 |  |  |  |  |  |  |  |
| 8. | Installation of NetAnim and discuss all its steps. | 27/05/2021 |  |  |  |  |  |  |  |

**Date: 16/03/2021**

**EXPERIMENT-1(A)**

**AIM: Discuss about event discrete simulation and its tools.**

**SYSTEM:**

A system is a group of interacting or interrelated entities that form a unified whole. [1] A system, surrounded and influenced by its environment, is described by its boundaries, structure and purpose and expressed in its functioning.

**DISCRETE AND CONTINUOUS**

**Discrete model:** the state variables change only at a countable number of points in time. These points in time are the ones at which the event occurs/change in state.

**Continuous:** the state variables change in a continuous way, and not abruptly from one state to another (infinite number of states).

**WHAT IS SIMULATION AND WHY DO WE NEED IT?**

In [computer network](https://en.wikipedia.org/wiki/Computer_network) research, network simulation is a technique whereby a software program models the behavior of a network by calculating the interaction between the different network entities (routers, switches, nodes, access points, links etc.). Most simulators use discrete event simulation - the modeling of systems in which state variables change at discrete points in time. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network / protocols would behave under different conditions.

A simulator is a collection of hardware and software systems which are used to mimic the behavior of some entity or phenomenon. Typically, the entity or phenomenon being simulated is from the domain of the tangible -- ranging from the operation of integrated circuits to behavior of a light aircraft during wind shear. Simulators may also be used to analyze and verify theoretical models which may be too difficult to grasp from a purely conceptual level. Such phenomenon range from examination of black holes to the study of highly abstract models of computation. As such, simulators provide a crucial role in both industry and academia.

Despite the increasing recognition of simulators as a viable and necessary research tool, one must constantly be aware of the potential problems which simulators may introduce. Many of the problems are related to the computational limitations of existing hardware platforms but are quickly being overcome as more powerful platforms are introduced. Other problems, unfortunately, are inherent within simulators and are related to the complexity associated with the systems being simulated. This section highlights some of the major advantages and disadvantages posed by modern day simulators.

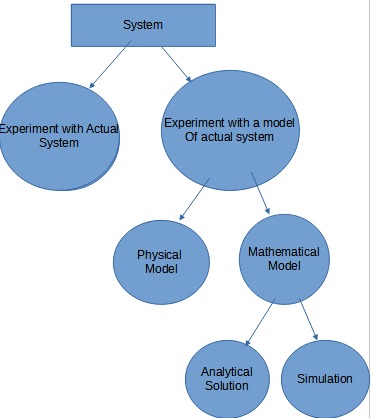
**DISCRETE EVENT SIMULATION**

Discrete event simulation (DES) is a method of simulating the behavior and performance of a real-life process, facility or system. DES is being used increasingly in health-care services24–26 and the increasing speed and memory of computers has allowed the technique to be applied to problems of increasing size and complexity. DES models the system as a series of ‘events’ [e.g. a birth, a stay in an intensive care unit (ICU), a transfer or a discharge] that occur over time. DES assumes no change in the system between events. In DES, patients are modelled as independent entities each of which can be given associated attribute information. In the case of neonatal simulation this may include parameters such as gestational age or weight at birth, hospital of birth, singleton/twin and current location.

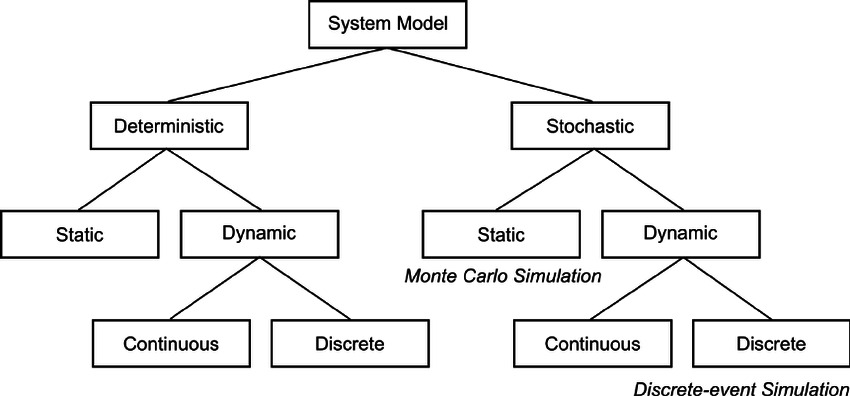
The information may be modified as time runs in the simulation model (e.g. the location will be changed depending on the status of the units in the network, and the level of care being received will be modified as the infant progresses). The simulation also accounts for resources. In the neonatal model the key resources are cots (with the highest level of care for each cot specified) and nurses. In order to care for an infant a unit must have the necessary cot and the necessary nursing staff (applying appropriate guidelines). The model allows each unit to work to a specified level of overcapacity regarding nursing, but will monitor the time each unit is undergoing overcapacity. DES models also allow for complex rules specifying where infants may be accepted; for example, there may be two ICUs, but with different facilities (e.g. surgery) or with different limits on gestational ages. DES thus allows complex decision logic to be incorporated that is not as readily possible in other types of modeling.

**TOOLS OF EVENT DISCRETE EVENT SIMULATION**

1. **NS2** is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks.
2. **Ns-3** is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. Ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use. Docs. App Store.
3. **OMNeT++** is a modular, component-based C++ simulation library and framework, primarily for building network simulators. OMNeT++ can be used for free for non-commercial simulations like at academic institutions and for teaching. OMNEST is an extended version of OMNeT++ for commercial use cases.
4. **ns-1**: The first version of ns, known as ns-1, was developed at Lawrence Berkeley National Laboratory (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived in 1989 from an earlier simulator known as REAL by S. Keshav.
5. **NetSim** is an end-to-end, full stack, packet level network simulator and emulator. It provides network engineers with a technology development environment for protocol modeling, network R&D and military communications. The behavior and performance of new protocols and devices can be investigated in a virtual network within NetSim at significantly lower cost and in less time than with hardware prototypes.
6. **The QualNet** network simulation software (QualNet) is a planning, testing, and training tool that “mimics” the behavior of real communication networks. Network simulation is a cost-effective method for developing, deploying, and managing network-centric systems throughout their entire lifecycle.
7. **SIM.JS** is an event-based discrete-event simulation library based on standard JavaScript. The library has been written in order to enable simulation within standard browsers by utilizing web technology. SIM.JS supports entities, resources (Facility, Buffers and Stores), communication (via Timers, Events and Messages) and statistics (with Data Series, Time Series and Population statistics).The SIM.JS distribution contains tutorials, in-depth documentation, and a large number of examples.SIM.JS is released as open source software under the LGPL license. The first version was released in January 2011.



**SYSTEM IMPLEMENTATION AND STUDY**



**MODEL TAXTONOMY**

**Date: 16/03/2021**

**EXPERIMENT-1B**

**AIM: Installation of NS3 simulator and discuss all its steps.**

**Installation of ns3 dependencies**

Ns3 needs so many dependencies, developmental libraries, drivers, etc. so install all those

$] sudo apt update

$] sudo apt upgrade

$] sudo apt-get install build-essential autoconf automake libxmu-dev python-pygoocanvas python-pygraphviz cvs mercurial bzr git cmake p7zip-full python-matplotlib python-tk python-dev python-kiwi python-gnome2 python-gnome2-desktop-dev python-rsvg qt4-dev-tools qt4-qmake qt4-qmake qt4-default gnuplot-x11 wireshark

**Installing ns3**

Go to the location of the download folder and copy the file to the home folder and open the terminal and give the command

$]  tar jxvf ns-allione-3.27.tar.bz2

$] cd ns-allinone-3.27/

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

This will take some time for getting compiled and build. Once the installation is successful, you will get a screen like given below.

|  |
| --- |
| [ns3 installation](https://4.bp.blogspot.com/-bn2memgI8dI/WrZG1ZQ5ivI/AAAAAAABcrg/NHswT7qjtR8FVoNsv0meda1jaoh-oIGEgCLcBGAs/s1600/Screenshot+from+2018-03-24+17-43-46.png) |
|  |

This indicates that ns3 is built successfully.

To check any application is running. Do the following steps

$] cd ns-3.27/

$] ./waf --run hello-simulator

This will print the hello Simulator which indicates that ns3 is installed successfully.

**Date: 23/03/2021**

**EXPERIMENT 2**

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

**THEORY:**

The ns-3 simulator is a discrete-event network simulator targeted primarily for research and educational use. The ns-3 project, started in 2006, is an open-source project developing ns-3.

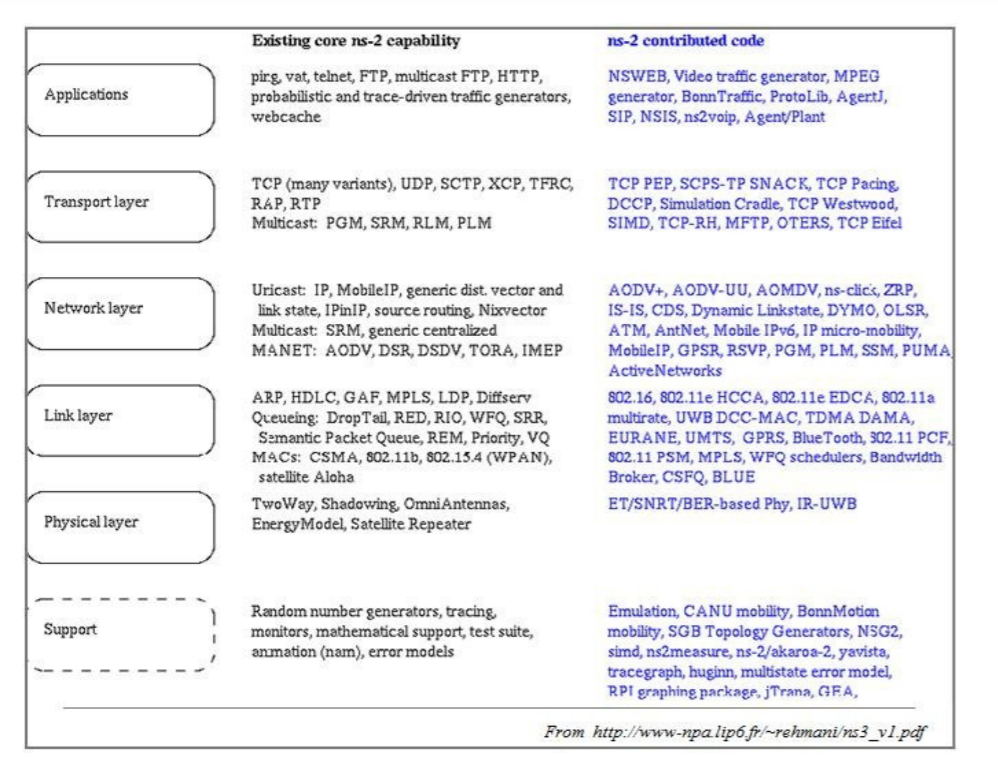
Ns-3 has been developed to provide an open, extensible network simulation platform, for networking research and education. In brief, ns-3 provides models of how packet data networks work and perform, and provides a simulation engine for users to conduct simulation experiments. Some of the reasons to 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.

A few key points are worth noting at the onset:

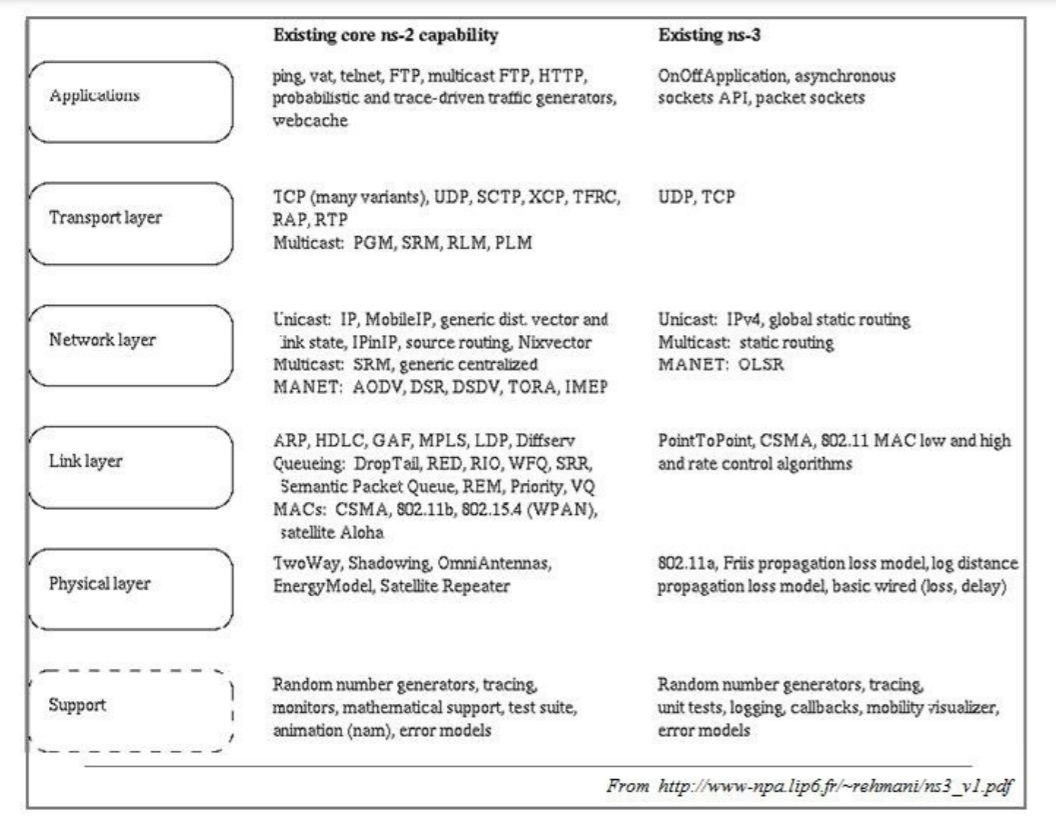
* Ns-3 is open-source, and the project strives to maintain an open environment for researchers to contribute and share their software.
* Ns-3 is not a backwards-compatible extension of ns-2; it is a new simulator. The two simulators are both written in C++ but ns-3 is a new simulator that does not support the ns-2 APIs.

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 or macOS systems, although support exists for BSD systems and also for Windows frameworks that can build Linux code, such as Windows Subsystem for Linux, or Cygwin. Native Windows Visual Studio is not presently supported although a developer is working on future support. Windows users may also use a Linux virtual machine.



**Ns2 contributed code**



**NS2 and NS3 existing core capabilities**

**COMPARISON BETWEEN NS3 AND NS2**

**Application level difference between NS3 and NS2**

|  |  |
| --- | --- |
| **NS3** | **NS2** |
| NS3 can be act as the emulator that it can connect to the real world. | NS2 cannot be act as the emulator. |
| Some of the NS2 models can be imported to NS3. | NS3 scripts cannot run in NS2 environment |

**Programming Language level Difference between NS2 and NS3:**

|  |  |
| --- | --- |
| **NS3** | **NS2** |
| NS3 is written using C++ | NS2 is written with the help of TCL and C++ |
| Compilation time is not a matter | C++ recompilation takes more time more than TCL so most of the scripts are written using TCL |
| A Simulation script can be written in ns3 | Simulation script is not possible with NS2 |
| Python is available for the scripting language | Only TCL can be used as the scripting language |

**Packets difference in NS2 and NS3:**

|  |  |
| --- | --- |
| **NS3** | **NS2** |
| Information needed to send through the packet can be added at the header,trailer, buffer ,etc. | The header part of the NS2 includes all the information of header parts in the specified protocol |
| NS3 frees the memory that used to store the packets | NS2 never reuse or re allocate the memory until it gets terminated. |

**File Format Difference between NS2 and NS3:**

|  |  |
| --- | --- |
| **NS3** | **NS2** |
| .tr-> files used for trace analysis | . tr-> files used for trace parameters |
| .XML->files are used for network Animation | .nam -> files used for Network Animation |
| .csv-> files used for gnu plot | .xg -> files used for graph |

**Visualization Difference between NS2 and NS3:**

|  |  |
| --- | --- |
| **NS3** | **NS2** |
| Python visualizer , Network Animator visualization is available | Nam animator is available for visualization |

**Performance level difference between ns2 and ns3:**

|  |  |
| --- | --- |
| **NS3** | **NS2** |
| Memory allocation is good | Memory allocation is not good as NS3 |
| System prevents unnecessary parameters to be stored. | Unnecessary parameters cannot be prevented. |
| Total computation is less when compared to NS2 | Total Computation time is high when compared to NS3 |

**Date: 08/04/2021**

**EXPERIMENT – 3**

**AIM** – Write a program in NS3 to connect 2 nodes

**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[]){

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 ("10Mbps"));

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));

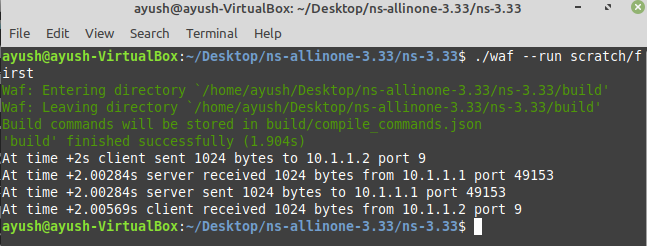
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT –**



**Date: 15/04/2021**

**EXPERIMENT – 4**

**AIM** – Write a program in NS3 to connect 3 nodes

**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 ("3NodeConnection");

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 ("10Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));

PointToPointHelper pointToPoint1;

pointToPoint1.SetDeviceAttribute ("DataRate", StringValue ("20Mbps"));

pointToPoint1.SetChannelAttribute ("Delay", StringValue ("10ms"));

NetDeviceContainer devices;

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

NetDeviceContainer devices1;

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");

Ipv4AddressHelper address1;

address1.SetBase ("176.0.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);

Ipv4InterfaceContainer interfaces1 = address1.Assign (devices1);

UdpEchoServerHelper echoServer (45);

UdpEchoServerHelper echoServer1 (50);

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

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

ApplicationContainer serverApps1 = echoServer1.Install (nodes.Get (1));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 45);

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

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

echoClient.SetAttribute ("PacketSize", UintegerValue (4096));

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

clientApps.Start (Seconds (4.0));

clientApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient1 (interfaces.GetAddress (1), 50);

echoClient1.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient1.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient1.SetAttribute ("PacketSize", UintegerValue (4096));

ApplicationContainer clientApps1 = echoClient1.Install (nodes.Get (0));

clientApps1.Start (Seconds (5.0));

clientApps1.Stop (Seconds (10.0));

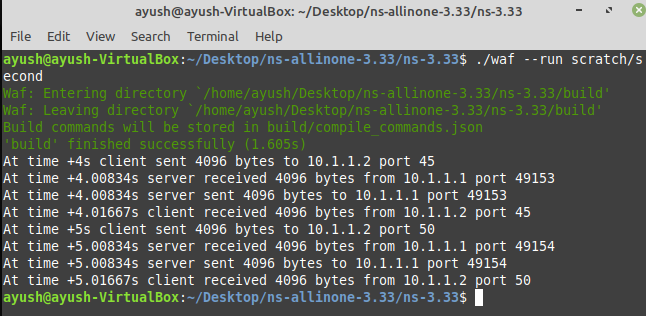
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT –**



**Date: 22/04/2021**

**EXPERIMENT – 5**

**AIM** – Write a program in NS3 to implement Bus Topology.

**SOURCE 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

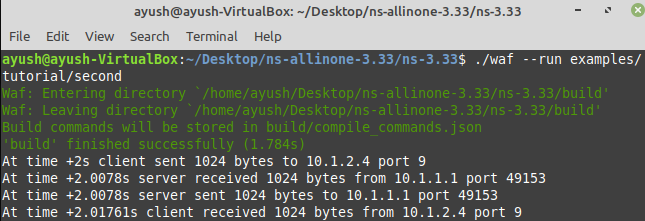
// n0 -------------- n1 n2 n3 n4

// 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 (\_\_FILE\_\_);  
  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 –**



**Date: 29/04/2021**

**EXPERIMENT – 6**

**AIM** – Write a program in NS3 to implement Star Topology.

**SOURCE CODE –**

#include <iostream>

#include <fstream>

#include <string>

#include <cassert>

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

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

#include "ns3/applications-module.h"

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

using namespace ns3;

// Default Network topology, 9 nodes in a star

/\*

n2 n3 n4

\ | /

\|/

n1---n0---n5

/| \

/ | \

n8 n7 n6

\*/

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

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

Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (250));

Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("5kb/s"));

uint32\_t N = 9; //number of nodes in the star

CommandLine cmd ("FILE");

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

cmd.Parse (argc, argv);

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

NodeContainer serverNode;

NodeContainer clientNodes;

serverNode.Create (1);

clientNodes.Create (N-1);

NodeContainer allNodes = NodeContainer (serverNode, clientNodes);

InternetStackHelper internet;

internet.Install (allNodes);

std::vector<NodeContainer> nodeAdjacencyList (N-1);

for(uint32\_t i=0; i<nodeAdjacencyList.size (); ++i){

nodeAdjacencyList[i] = NodeContainer (serverNode, clientNodes.Get (i));

}

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

PointToPointHelper p2p;

p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));

std::vector<NetDeviceContainer> deviceAdjacencyList (N-1);

for(uint32\_t i=0; i<deviceAdjacencyList.size (); ++i){

deviceAdjacencyList[i] = p2p.Install (nodeAdjacencyList[i]);

}

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

Ipv4AddressHelper ipv4;

std::vector<Ipv4InterfaceContainer> interfaceAdjacencyList (N-1);

for(uint32\_t i=0; i<interfaceAdjacencyList.size (); ++i){

std::ostringstream subnet;

subnet<<"10.1."<<i+1<<".0";

ipv4.SetBase (subnet.str ().c\_str (), "255.255.255.0");

interfaceAdjacencyList[i] = ipv4.Assign (deviceAdjacencyList[i]);

}

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

uint16\_t port = 50000;

Address sinkLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));

PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", sinkLocalAddress);

ApplicationContainer sinkApp = sinkHelper.Install (serverNode);

sinkApp.Start (Seconds (1.0));

sinkApp.Stop (Seconds (10.0));

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

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

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

ApplicationContainer clientApps;

for(uint32\_t i=0; i<clientNodes.GetN (); ++i){

AddressValue remoteAddress

(InetSocketAddress (interfaceAdjacencyList[i].GetAddress (0), port));

clientHelper.SetAttribute ("Remote", remoteAddress);

clientApps.Add (clientHelper.Install (clientNodes.Get (i)));

}

clientApps.Start (Seconds (1.0));

clientApps.Stop (Seconds (10.0));

AsciiTraceHelper ascii;

p2p.EnableAsciiAll (ascii.CreateFileStream ("tcp-star-server.tr"));

p2p.EnablePcapAll ("tcp-star-server");

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

Simulator::Run ();

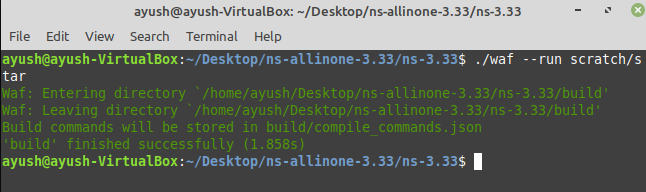
Simulator::Destroy ();

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

return 0;

}

**OUTPUT –**



**Date: 20/05/2021**

**EXPERIMENT – 7**

**AIM** – Write a program in NS3 for connecting multiple routers and nodes and building a hybrid topology.

**SOURCE CODE –**

#include "ns3/core-module.h"

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

#include "ns3/network-module.h"

#include "ns3/applications-module.h"

#include "ns3/mobility-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

#include "ns3/yans-wifi-helper.h"

#include "ns3/ssid.h"

// 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

using namespace ns3;

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

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

bool verbose = true;

uint32\_t nCsma = 3;

uint32\_t nWifi = 3;

bool tracing = false;

CommandLine cmd ("\_FILE\_");

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

cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);

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

cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

cmd.Parse (argc,argv);

if (nWifi > 18) {

std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" << std::endl;

return 1;

}

if (verbose){

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

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

}

NodeContainer p2pNodes;

p2pNodes.Create (2);

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

NodeContainer csmaNodes;

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

csmaNodes.Create (nCsma);

CsmaHelper csma;

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

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

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;

wifiStaNodes.Create (nWifi);

NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();

YansWifiPhyHelper phy;

phy.SetChannel (channel.Create ());

WifiHelper wifi;

wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;

Ssid ssid = Ssid ("ns-3-ssid");

mac.SetType ("ns3::StaWifiMac",

"Ssid", SsidValue (ssid),

"ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;

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

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

NetDeviceContainer apDevices;

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

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",

"MinX", DoubleValue (0.0),

"MinY", DoubleValue (0.0),

"DeltaX", DoubleValue (5.0),

"DeltaY", DoubleValue (10.0),

"GridWidth", UintegerValue (3),

"LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",

"Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));

mobility.Install (wifiStaNodes);

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

mobility.Install (wifiApNode);

InternetStackHelper stack;

stack.Install (csmaNodes);

stack.Install (wifiApNode);

stack.Install (wifiStaNodes);

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);

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

address.Assign (staDevices);

address.Assign (apDevices);

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 (wifiStaNodes.Get (nWifi - 1));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));

if (tracing == true){

pointToPoint.EnablePcapAll ("third");

phy.EnablePcap ("third", apDevices.Get (0));

csma.EnablePcap ("third", csmaDevices.Get (0), true);

}

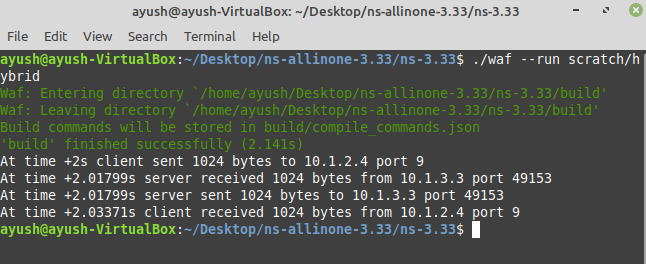
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT –**



**Date: 27/05/2021**

**EXPERIMENT – 8**

**AIM** – Installation of NetAnim and discuss all its steps.

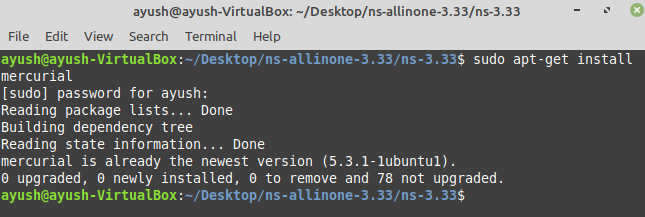
**Installation of NetAnim dependencies**

NetAnim is an offline animator based on the Qt toolkit. It currently animates the simulation using an XML trace file collected during simulation.

If you haven’t installed the NetAnim dependencies while installing NS3, install the dependencies given below

$] apt-get install mercurial

$] apt-get install qttools5-dev-tools



After you’re done installing, change directories to check if NetAnim program is initiating properly

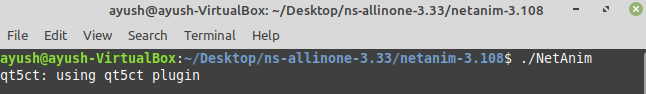
$] cd ns-allinone-3.33

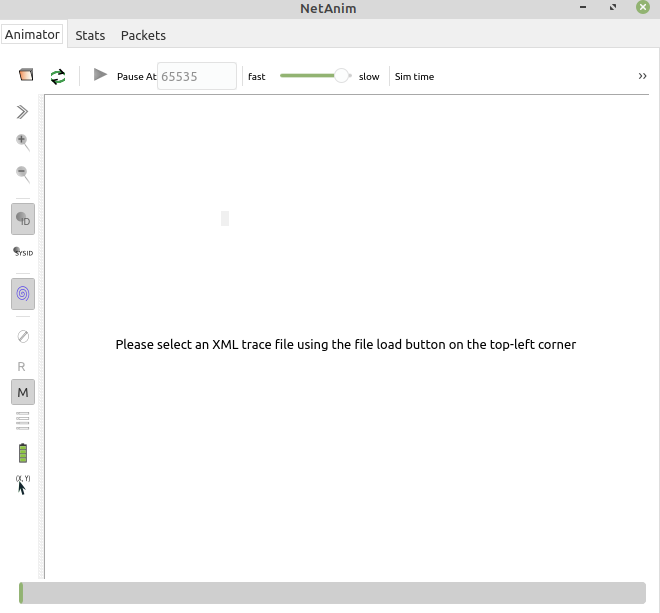
$] cd netanim-3.108

$] make clean

$] make

$] ./NetAnim





After successful initiation, we have to make some changes in our pre-existing .cc file so that it could be run successfully

* Ensure that your wscript includes the "netanim" module. Example as in: src/netanim/examples/wscript.
* Also include the header [#include "ns3/netanim-module.h"] in your test program.
* Add the statement "AnimationInterface anim ("animation.xml");" before Simulator::Run().

Finally you can run any ns3 .cc file after making the mentioned changes by the command

$] Enter the directory you’ve saved the file in

$] ./waf –run file\_name

**Output:**

