

# ***Computer Networks Lab***

## ***ETMA 354***

Faculty: **Dr. Pooja Gupta**

Name: **Syeda Reeha Quasar**

Roll No.: **14114802719**

Semester: **6**



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 fulfilling societal needs.

# Computer Networks Lab (ETCS -354) Lab Assessment Sheet

Student Enrollment No:

Student Name:

S.No	Experiment	Mar ks					Total marks	Signa ture with date
		R1	R2	R3	R4	R5		
1	Discuss simulation and its various tools.							
2	Introduction to NS3 and its comparison with NS2.							
3	Installation of NS3 simulator and it's steps.							
4	Write a program in NS3 to connect 2 nodes.							
5	Write a program in NS3 to connect 3 nodes.							
6								
7								
8								
9								
10								

Overall Comments:

Faculty Name:

Signature:



# EXPERIMENT - 1

## Computer Networks Lab

### Aim

Discuss simulation and its various tools.

Syeda Reeha Quasar

14114802719

4C7

# EXPERIMENT – 1

## Aim:

Discuss simulation and its various tools.

## Theory:

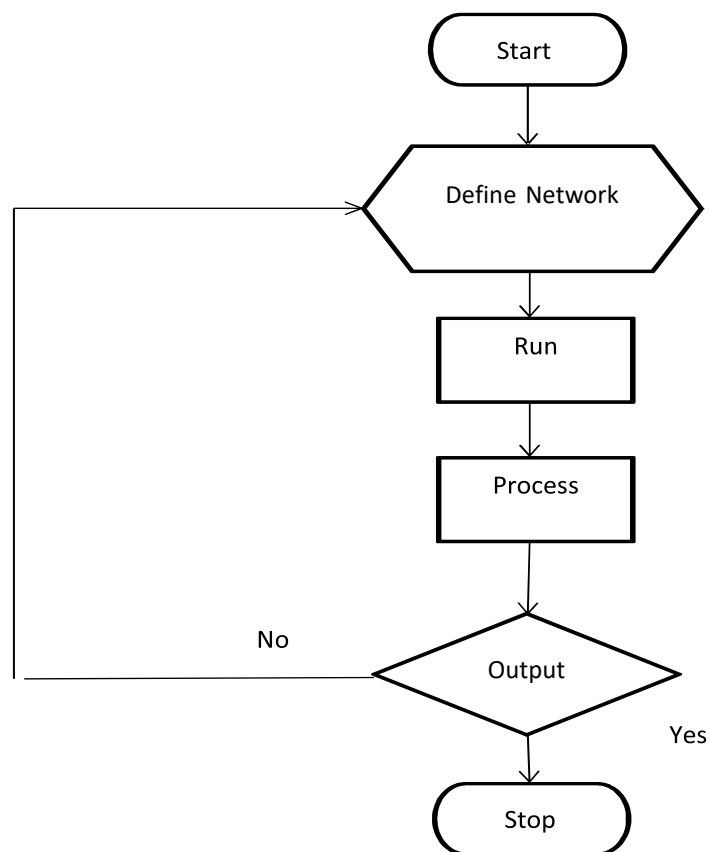
### SYSTEM:

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/changes 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 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 services<sup>24–26</sup> 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.

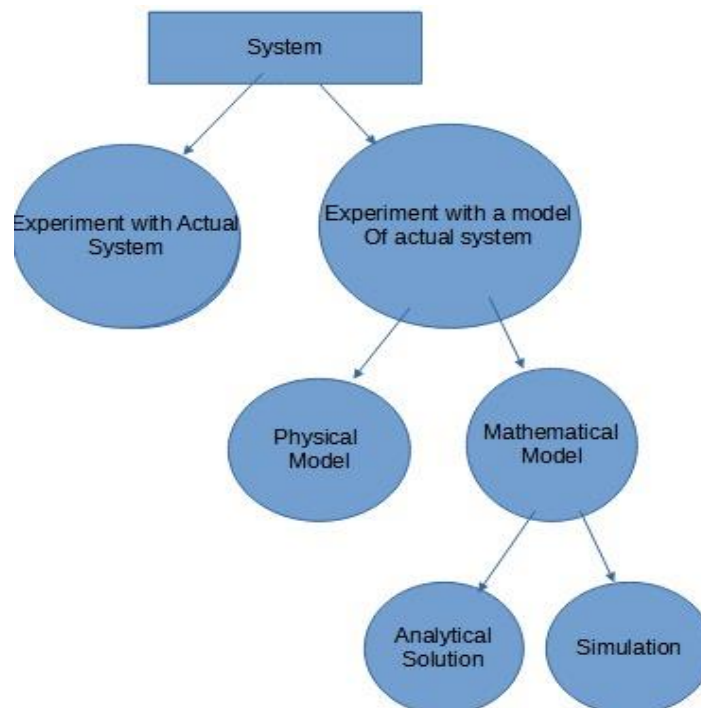
## TOOLS OF EVENT DISCRETE EVENT SIMULATION

1. **NS2** is an open-source simulation tool that runs on Linux. It is a discrete 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. OMNeT 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

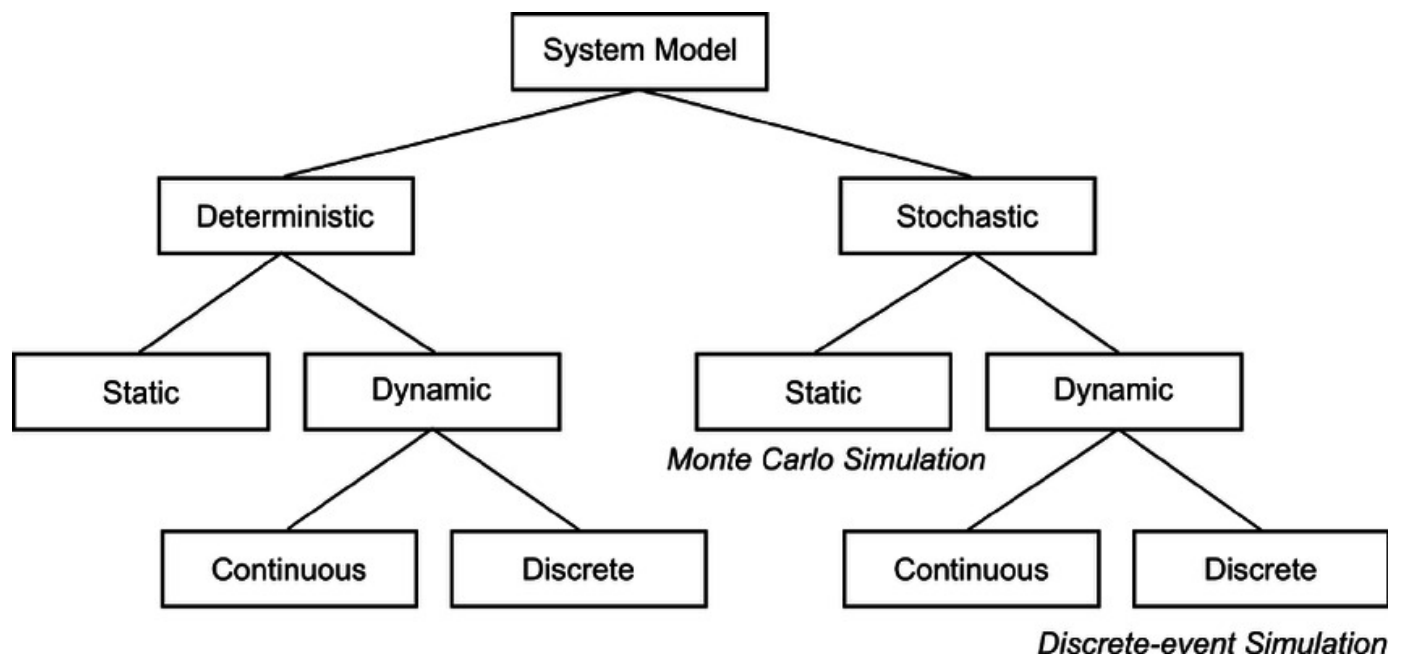


performance of new protocols and devices can be investigated in a virtual network within NetSim at significantly lower cost and in less time than with hardware prototypes.

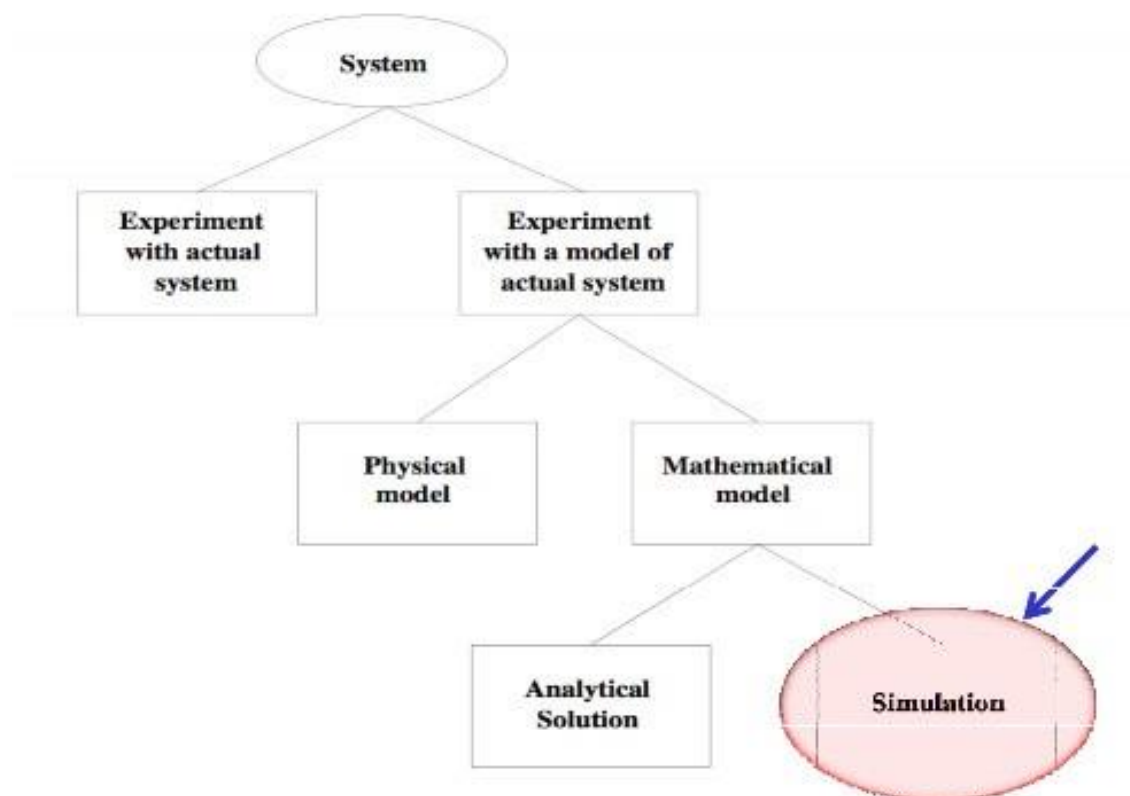
6. **The QualNet** network simulation software (QualNet) is a planning, testing, and training tool that “mimics” the behavior of real communication networks. Network simulation is a cost-effective method for developing, deploying, and managing network-centric systems throughout their entire lifecycle.
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





# EXPERIMENT - 2

## Computer Networks Lab

### Aim

Introduction to NS3 and its comparison with NS2.

Syeda Reeha Quasar

14114802719

4C7

## 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 performing studies that are more difficult or not possible to perform with real systems, studying system behavior in a highly controlled, reproducible environment, and learning 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.

	Existing core ns-2 capability	ns-2 contributed code
Applications	ping, vat, telnet, FTP, multicast FTP, HTTP, probabilistic and trace-driven traffic generators, webcache	NSWEB, Video traffic generator, MPEG generator, BonnTraffic, ProtoLib, AgentJ, SIP, NSIS, ns2voip, Agent/Plant
Transport layer	TCP (many variants), UDP, SCTP, XCP, TFRC, RAP, RTP Multicast: PGM, SRM, RLM, PLM	TCP PEP, SCPS-TP SNACK, TCP Pacing, DCCP, Simulation Cradle, TCP Westwood, SIMD, TCP-RH, MFTP, OTSRS, TCP Eiffel
Network layer	Unicast: IP, MobileIP, generic dist. vector and link state, IPinIP, source routing, Nixvector Multicast: SRM, generic centralized MANET: AODV, DSR, DSDV, TORA, IMEP	AODV+, AODV-UU, AOMDV, ns-cllc, ZRP, IS-IS, CDS, Dynamic Linkstate, DYMO, OLSR, ATM, AntNet, Mobile IPv6, IP micro-mobility, MobileIP, GPRS, RSVP, PGM, PLM, SSM, PUMA, ActiveNetworks
Link layer	ARP, HDLC, GAF, MPLS, LDP, Diffserv Queueing: DropTail, RED, RIO, WFQ, SRR, Semantic Packet Queue, REM, Priority, VQ MACs: CSMA, 802.11b, 802.15.4 (WPAN), satellite Aloha	802.16, 802.11e HCCA, 802.11e EDCA, 802.11a multirate, UWB DCC-MAC, TDMA DAMA, EURANE, UMTS, GPRS, BlueTooth, 802.11 PCF, 802.11 PSM, MPLS, WFQ schedulers, Bandwidth Broker, CSFQ, BLUE
Physical layer	TwoWay, Shadowing, OmniAntennas, EnergyModel, Satellite Repeater	ET/SNRT/BER-based Phy, IR-UWB
Support	Random number generators, tracing, monitors, mathematical support, test suite, animation (nam), error models	Emulation, CANU mobility, BonnMotion mobility, SGB Topology Generators, NSG2, simd, ns2measure, ns-2/akara-2, yavista, tracegraph, huginn, multistate error model, RPI graphing package, jTrans, GFA,

From [http://www.npa.lip6.fr/~rehmani/ns3\\_v1.pdf](http://www.npa.lip6.fr/~rehmani/ns3_v1.pdf)

## Ns2 contributed code

	Existing core ns-2 capability	Existing ns-3
Applications	ping, vat, telnet, FTP, multicast FTP, HTTP, probabilistic and trace-driven traffic generators, webcache	OnOffApplication, asynchronous sockets API, packet sockets
Transport layer	TCP (many variants), UDP, SCTP, XCP, TFRC, RAP, RTP Multicast: PGM, SRM, RLM, PLM	UDP, TCP
Network layer	Unicast: IP, MobileIP, generic dist. vector and link state, IPinIP, source routing, Nixvector Multicast: SRM, generic centralized MANET: AODV, DSR, DSDV, TORA, IMEP	Unicast: IPv4, global static routing Multicast: static routing MANET: OLSR
Link layer	ARP, HDLC, GAF, MPLS, LDP, Diffserv Queueing: DropTail, RED, RIO, WFQ, SRR, Semantic Packet Queue, REM, Priority, VQ MACs: CSMA, 802.11b, 802.15.4 (WPAN), satellite Aloha	PointToPoint, CSMA, 802.11 MAC low and high and rate control algorithms
Physical layer	TwoWay, Shadowing, OmniAntennas, EnergyModel, Satellite Repeater	802.11a, Friis propagation loss model, log distance propagation loss model, basic wired (loss, delay)
Support	Random number generators, tracing, monitors, mathematical support, test suite, animation (nam), error models	Random number generators, tracing, unit tests, logging, callbacks, mobility visualizer, error models

From [http://www.npa.lip6.fr/~rehmani/ns3\\_v1.pdf](http://www.npa.lip6.fr/~rehmani/ns3_v1.pdf)

## NS2 and NS3 existing core capabilities

## COMPARISON BETWEEN NS3 AND NS2

### Application-level difference between NS3 and NS2

NS3	NS2
NS3 can act as the emulator that it can connect to the real world.	NS2 cannot act as the emulator.
Some of the NS2 models can be imported to NS3.	NS3 scripts cannot run in an 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:

<b>NS3</b>	<b>NS2</b>
.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:**

<b>NS3</b>	<b>NS2</b>
Python visualizer , Network Animator visualization is available	Nam animator is available for visualization

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

<b>NS3</b>	<b>NS2</b>
Memory allocation is good	Memory allocation is not good as NS3
The 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



# EXPERIMENT - 3

## Computer Networks Lab

### Aim

Installation of NS3 simulator and its steps.

Syeda Reeha Quasar

14114802719

4C7



## EXPERIMENT – 3

### Aim:

Installation of NS3 simulator and its steps.

### 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, 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 bzip2 gdb valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick texlive texlive-latex-extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlive-latex-extra texlive-extra-utils texlive-generic-recommended texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev python-pygccxml`
6. After downloading NS3 on the drive, extract all the files in the NS3 folder, which you have created.
7. Then you can find build.py along with other files in NS3 folder. Then to build the examples in ns-3 run :

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

If the build is successful then it will give output "Build finished successfully".

8. Now run the following command on the terminal window to configure with waf (build tool)

```
$/waf -d debug --enable-examples --enable-tests configure To build with waf (optional)
```

```
$/waf
```

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

```
$/test.py
```

If the tests are ok the installation is done

10. Now after installing ns3 and testing it run some programs first to be ns3 user: make sure you are in directory where waf script is available then run

### 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 bzip2 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

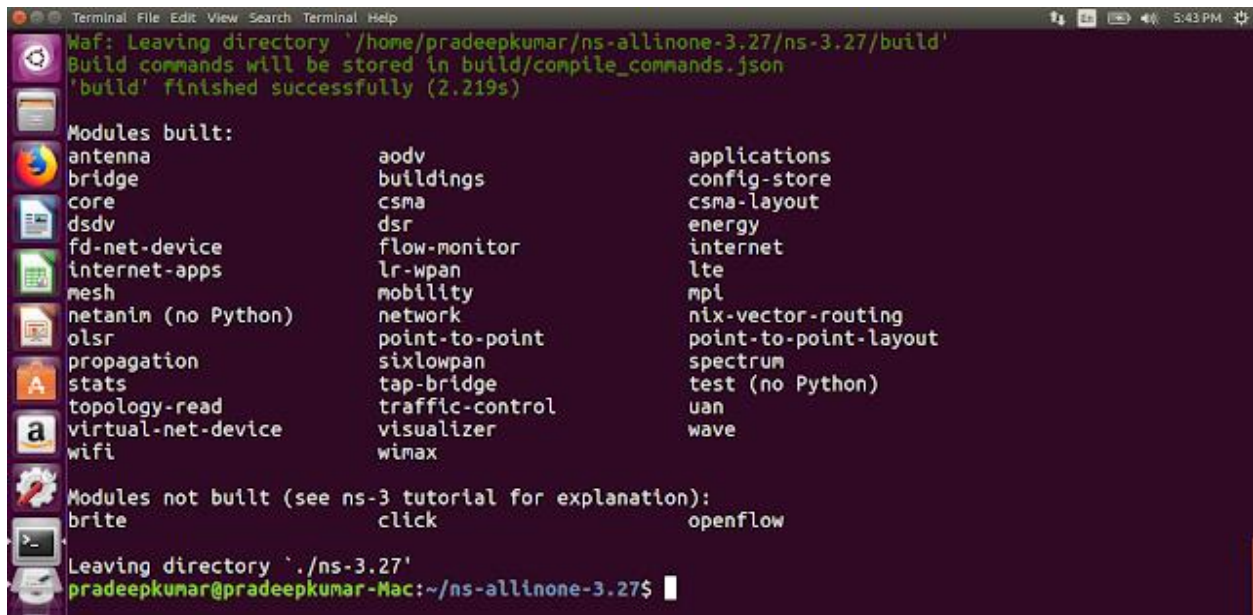
Go to the location of the download folder and copy the file to the home folder and open the terminal and give the command

```
$] tar jxvf ns-allinone-3.27.tar.bz2
```

```
$] cd ns-allinone-3.27/
```

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

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

A terminal window screenshot showing the output of the 'build.py' command. The terminal title is 'Terminal File Edit View Search Terminal Help'. The output shows 'Waf: Leaving directory \'/home/pradeepkumar/ns-allinone-3.27/ns-3.27/build\'', 'Build commands will be stored in build/compile\_commands.json', and 'build finished successfully (2.219s)'. Below this, a list of 'Modules built:' is displayed in three columns: antenna, bridge, core, dsdv, fd-net-device, internet-apps, mesh, netanim (no Python), olsr, propagation, stats, topology-read, virtual-net-device, wifi, aodv, buildings, csma, dsr, flow-monitor, lr-wpan, mobility, network, point-to-point, sixlowpan, tap-bridge, traffic-control, visualizer, wimax, applications, config-store, csma-layout, energy, internet, lte, mpl, nix-vector-routing, point-to-point-layout, spectrum, test (no Python), uan, and wave. A section 'Modules not built (see ns-3 tutorial for explanation):' lists brite, click, and openflow. The terminal ends with 'Leaving directory \'/ns-3.27\'', the prompt 'pradeepkumar@pradeepkumar-Mac:~/ns-allinone-3.27\$', and a cursor.

This indicates that ns3 is built successfully.

To check if 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.



# EXPERIMENT - 4

## Computer Networks Lab

### Aim

Write a program in NS3 to connect 2 nodes.

Syeda Reeha Quasar

14114802719

4C7

## EXPERIMENT – 4

### 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"

// Default Network Topology
//
//      10.1.1.0
// n0 ----- n1
//      point-to-point
//

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
    CommandLine cmd (__FILE__);
    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 (5);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 5);
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:

```

reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ ./waf --run first
Waf: Entering directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Waf: Leaving directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.490s)
At time +2s client sent 1024 bytes to 10.1.1.2 port 5
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 5
reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ █

```



# EXPERIMENT - 5

## Computer Networks Lab

### Aim

Write a program in NS3 to connect 3 nodes.

Syeda Reeha Quasar

14114802719

4C7

## EXPERIMENT – 5

### 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"

// Network Topology
//
// n0(client) ----- n1(server) ----- n2(client)

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

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

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



```

devices.Add(pointToPoint.Install (nodes.Get (1), nodes.Get(2)));

InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);

UdpEchoServerHelper echoServer (5);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient1 (interfaces.GetAddress (1), 5);
echoClient1.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient1.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient1.SetAttribute ("PacketSize", UIntegerValue (1024));

UdpEchoClientHelper echoClient2 (interfaces.GetAddress (1), 5);
echoClient2.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient2.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient2.SetAttribute ("PacketSize", UIntegerValue (5026));

ApplicationContainer clientApps = echoClient1.Install (nodes.Get (0));
ApplicationContainer clientApps1 = echoClient2.Install (nodes.Get (2));

clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

clientApps1.Start (Seconds (2.0));
clientApps1.Stop (Seconds (10.0));

Simulator::Run ();
Simulator::Destroy ();
return 0;
}

```

## Output:

```
reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ ./waf --run three_nodes
Waf: Entering directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
[2946/2999] Compiling scratch/three_nodes.cc
[2960/2999] Linking build/scratch/three_nodes
Waf: Leaving directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.887s)
At time +2s client sent 1024 bytes to 10.1.1.2 port 5
At time +2s client sent 5026 bytes to 10.1.1.2 port 5
At time +2.00584s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00584s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.0091s server received 5026 bytes from 10.1.1.4 port 49153
At time +2.0091s server sent 5026 bytes to 10.1.1.4 port 49153
At time +2.0182s client received 5026 bytes from 10.1.1.3 port 5
reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$
```