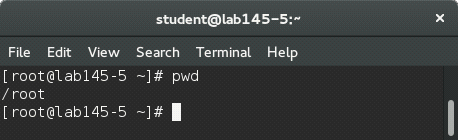
DATE: 18-01-17

**Experiment -1**

**AIM**: Linux commands

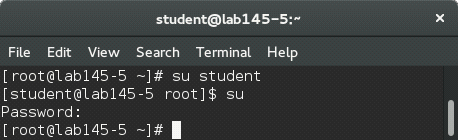
1-**pwd**

**pwd** prints the full pathname of the current working directory.



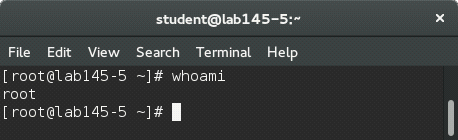
2-**su**

The **su** command makes it possible to log in under a different user name from a running session. When using the command without specifying a user name, you will be prompted for the root password. Specify a user name and the corresponding password to use the environment of the respective user. The password is not required from [root](https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/go01.html#root), as root is authorized to assume the identity of any user.



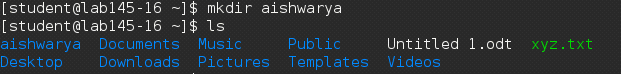
3-**whoami**

The whoami command writes the user name (i.e., **login** name) of the owner of the current **login** session to standard output.



4-**mkdir**

Creates a new directory

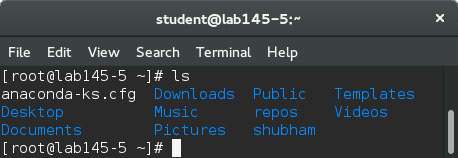


5-**ls**

If you run **ls** without any additional parameters, the program will list the contents of the current directory in short form.

**-l**: detailed list

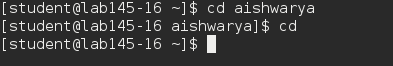
**-a:** displays hidden files



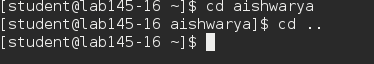
6-**cd**

Changes the current directory.

**cd** without any parameters changes to the user's home directory.

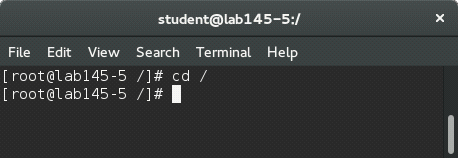


7-**cd..**  
The command cd .. tells your system to go up to the directory immediately above the one in which you are currently working.



8-**cd /**

Returns to the root directory



9-**cp**

Copies sourcefile to targetfile.

**-i**: Waits for confirmation, if necessary, before an existing targetfile is overwritten

**-r**: Copies recursively (includes subdirectories)

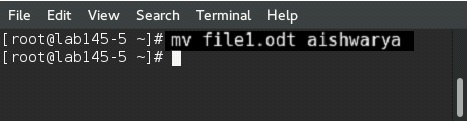
C:\Users\Vishnu\Desktop\3.jpg

10-**mv**

Copies sourcefile to targetfile then deletes the original sourcefile.

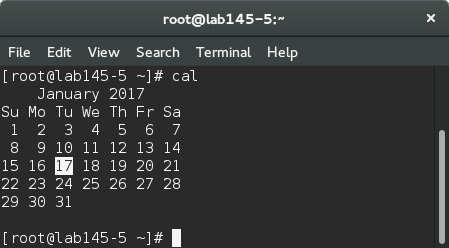
**-b**: Creates a backup copy of the sourcefile before moving

**-i**: Waits for confirmation, if necessary, before an existing targetfile is overwritten



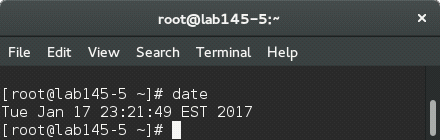
11-**cal**

**cal** displays a simple calendar. If no arguments are specified, the current month is displayed. The *month* may be specified as a number (1-12), as a month name or as an abbreviated month name according to the current locales.



12-**date**

The date command displays the current date and time. It can also be used to display a date in a format you specify. The super-user (root) can use it to set the system clock.

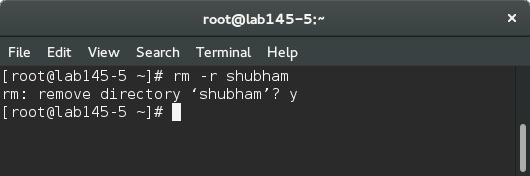


13-**rm**

Removes the specified files from the file system. Directories are not removed by **rm** unless the option -r is used.

**-r:** Deletes any existing subdirectories

**-i:** Waits for confirmation before deleting each file.



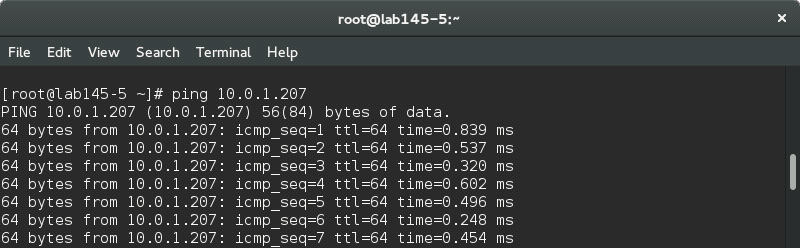
14-**ping**

The ping command is the standard tool for testing the basic functionality of TCP/IP networks. It sends a small data packet to the destination host, requesting an immediate reply. If this works, ping displays a message to that effect, which indicates that the network link is basically functioning.

**-c**: number Determines the total number of packages to send and ends after they have been dispatched. By default, there is no limitation set.

**-f**: *flood ping*: sends as many data packages as possible. A popular means, reserved to root, to test networks.

**-i**: value Specifies the interval between two data packages in seconds. Default: one second



DATE: 18-01-17

**EXPERIMENT NO-2**

**AIM**: Introduction to Discrete Event Simulation and Discrete Event Simulation Tools-ns2/ns3, Omnet++.

**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](https://en.wikipedia.org/wiki/Simulation) can directly jump in time from one event to the next.

**System**

A system is a set of [interacting](https://en.wikipedia.org/wiki/Interaction) or interdependent component parts forming a complex or intricate whole. Every system is delineated by its spatial and temporal boundaries, surrounded and influenced by its environment, described by its structure and purpose and expressed in its functioning.

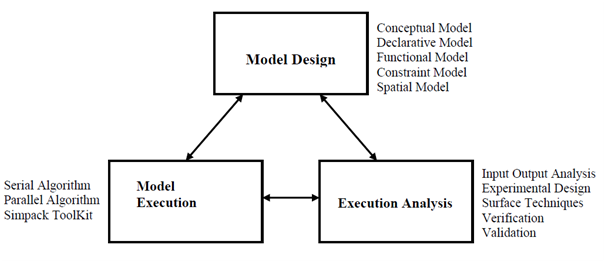
Alternatively, and usually in the context of complex social systems, the term is used to describe the set of rules that govern structure or behavior.

**Simulation**

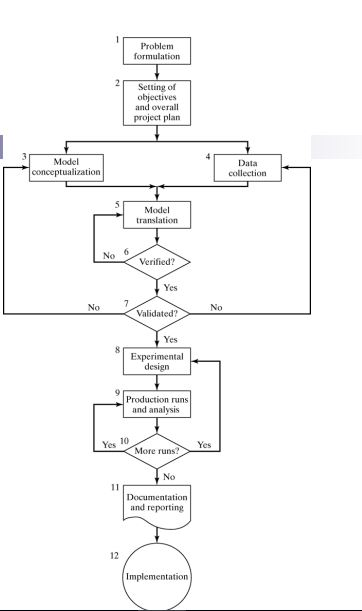
Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output. Simulation embodies the principle of ``learning by doing'' --- to learn about the system we must first build a model of some sort and then operate the model. The use of simulation is an activity that is as natural as a child who role plays. Children understand the world around them by simulating (with toys and figurines) most of their interactions with other people, animals and objects. As adults, we lose some of this childlike behavior but recapture it later on through computer simulation. To understand reality and all of its complexity, we must build artificial objects and dynamically act out roles with them. Computer simulation is the electronic equivalent of this type of role playing and it serves to drive synthetic environments and virtual worlds. Within the overall task of simulation, there are three primary sub-fields: model design, model execution and model analysis (see Fig. 1).

Conducting Experiment with THIS Model

Complex Real World System



**Fig.1-Simulation**

****

**Fig.2-Steps in simulation**

**Deterministic System**

A system is deterministic if its outputs are certain. This means that the relationships between its components are fully known and certain. Hence, when an input is given the output is fully predictable. An example of a deterministic system is the common entrance examination for entry into IIM. All the entities in the system and their interrelationships are well known and given an input the output can be determined with certainty.

**Stochastic System**

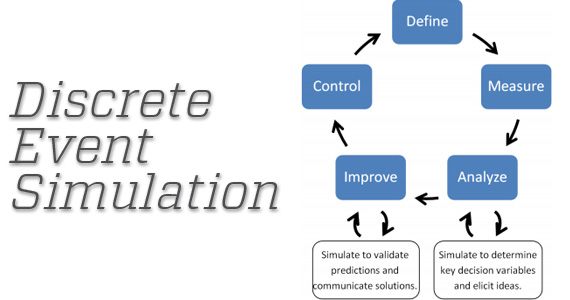
The word "stochastic" means "pertaining to chance" (Greek roots), and is thus used to describe subjects that contain some element of random or stochastic behavior. For a system to be *stochastic*, one or more parts of the system has randomness associated with it. Unlike a *deterministic* system, for example, a stochastic system does not always produce the same output for a given input. A few components of systems that can be stochastic in nature include stochastic inputs, random time-delays, noisy (modelled as random) disturbances, and even stochastic dynamic processes.

**Discrete System**

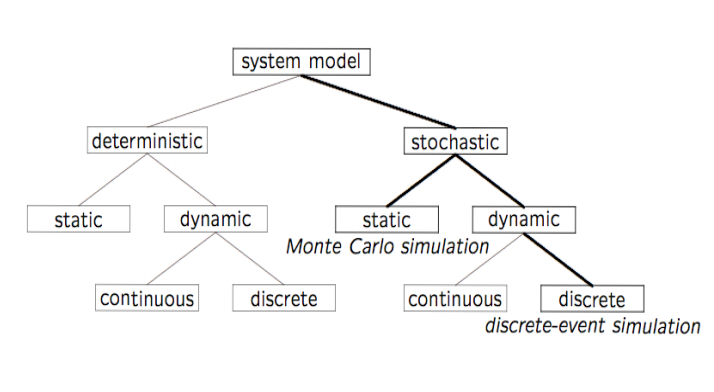
A **discrete system** is a system with a countable number of states. Discrete systems may be contrasted with continuous systems, which may also be called analog systems. A final discrete system is often modeled with a directed [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) and is analyzed for correctness and complexity according to [computational theory](https://en.wikipedia.org/wiki/Computational_theory). Because discrete systems have a countable number of states, they may be described in precise [mathematical models](https://en.wikipedia.org/wiki/Mathematical_models).

A [computer](https://en.wikipedia.org/wiki/Computer) is a [finite state machine](https://en.wikipedia.org/wiki/Finite_state_machine) that may be viewed as a discrete system. Because computers are often used to model not only other discrete systems but continuous systems as well, methods have been developed to represent real-world continuous systems as discrete systems. One such method involves sampling a continuous signal at [discrete time](https://en.wikipedia.org/wiki/Discrete_time) intervals

**Discrete Event Simulation**



**Fig.3**

**Fig.4**

Discrete event simulation (DES) is the process of codifying the behavior of a complex system as an ordered sequence of well-defined events. In this context, an event comprises a specific change in the system's state at a specific point in time.

Common applications of DES include stress testing, evaluating potential financial investments, and modeling procedures and processes in various industries, such as manufacturing and healthcare.

As an example of a situation that lends itself to DES, consider the amount of electrical [power](http://searchcio-midmarket.techtarget.com/definition/power) consumed by a corporation's office building as a function of time. Discrete events affecting this function include [power-up](http://whatis.techtarget.com/definition/power-up-or-power-on) or power-down of any electrical device in the building. Instantaneous changes of state in a device already powered-up are also discrete events; for example, a speed change in a cooling fan or a brightness change in a desk lamp.

An effective DES process must include, at a minimum, the following characteristics:

* Predetermined starting and ending points, which can be discrete events or instants in time.
* A method of keeping track of the time that has elapsed since the process began.
* A list of discrete events that have occurred since the process began.
* A list of discrete events pending or expected (if such events are known) until the process is expected to end.
* A graphical, statistical, or tabular record of the function for which DES is currently engaged.

**Applications of Discrete Event Simulation**

* DES is commonly used to monitor and predict the behavior of investments; the stock market is a classic example. DES can also help [administrators](http://searchnetworking.techtarget.com/definition/system-administrator) predict how a [network](http://searchnetworking.techtarget.com/definition/network) will behave under extraordinary conditions, such as the [Internet](http://searchwindevelopment.techtarget.com/definition/Internet) during a major disaster.
* Manufacturing Applications
* Semiconductor Manufacturing
* Construction Engineering and project management 
* Military application
* Logistics, Supply chain and distribution application
* Transportation modes and Traffic
* Business Process Simulation
* Health Care
* Automated Material Handling System (AMHS)

Test beds for functional testing of control-system software

* Risk analysis -- Insurance
* Computer Simulation
* Network simulation

Internet backbone, LAN (Switch/Router), Wireless, PSTN (call center)

**Tools Used for Discrete Event Simulation**

1. [**CPN Tools**](https://en.wikipedia.org/wiki/CPN_Tools)

A tool to analyse logistics/queuing models in all types of applications.

1. **ns3**

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](http://www.gnu.org/copyleft/gpl.html), and is publicly available for research, development, and use.

1. **omnet++**

**OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators.**

1. **ns2**

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

**Advantages of Discrete Event Simulation**

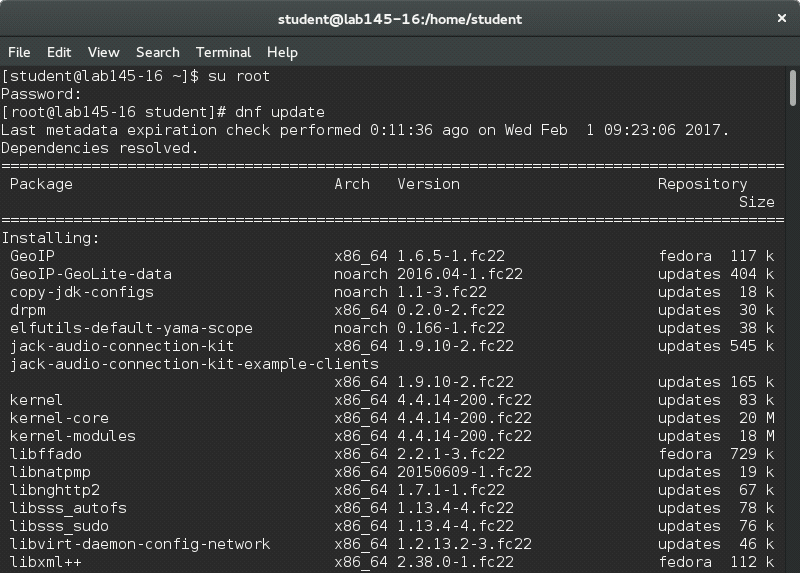
* New policies, operating procedures, information flows and son on can be explored without disrupting ongoing operation of the real system. 
* New hardware designs, physical layouts, transportation systems can be tested without committing resources for their acquisition.
* Time can be compressed or expanded to allow for a speed-up or slow-down of the phenomenon( clock is self-control). 
* Insight can be obtained about interaction of variables and important variables to the performance.  Bottleneck analysis can be performed to discover where work in process, the system is delayed.

DATE: 01-02-17

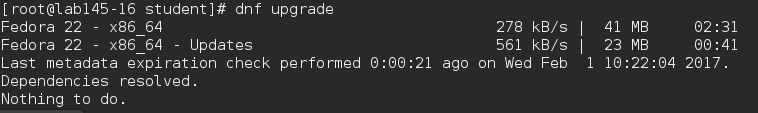
**EXPERIMENT-3**

**AIM**: Installation of ns3

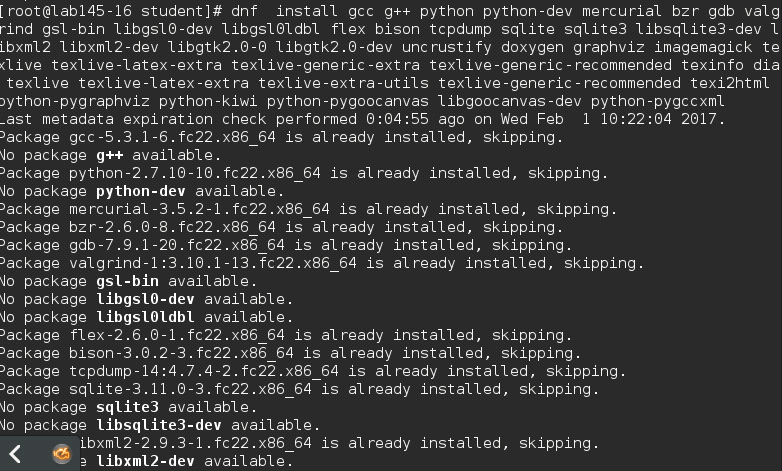
1-**dnf update**



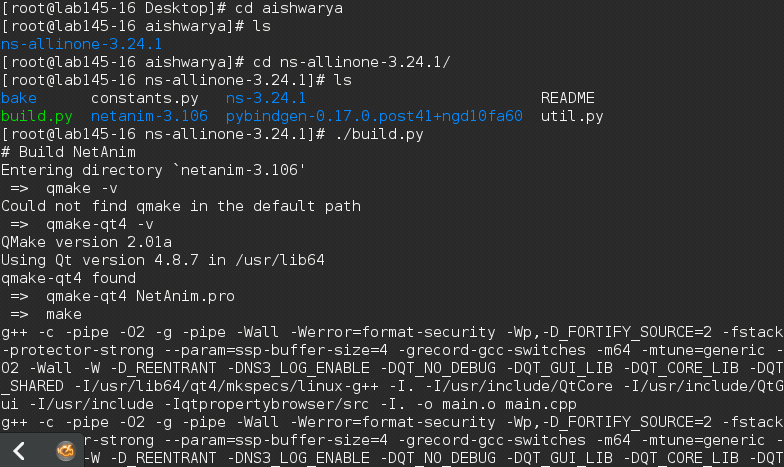
2-**dnf upgrade**



3-**dnf  install gcc g++ python python-dev mercurial bzr gdb valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick texlive texlive-latex-extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlive-latex-extra texlive-extra-utils texlive-generic-recommended texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev python-pygccxml**

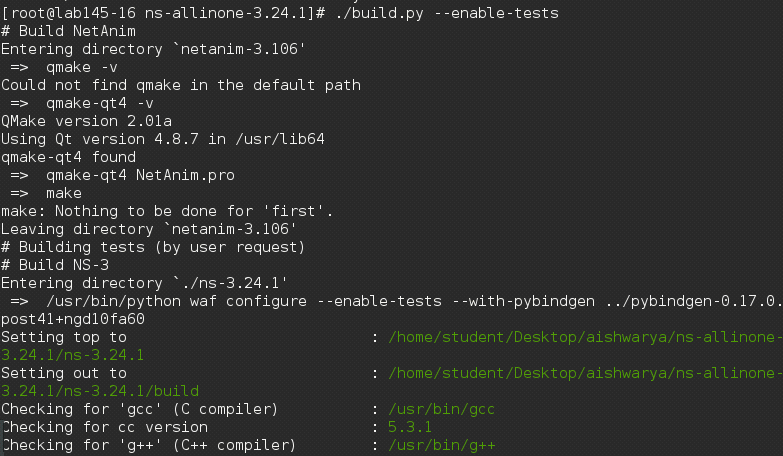


4-**./build.py**

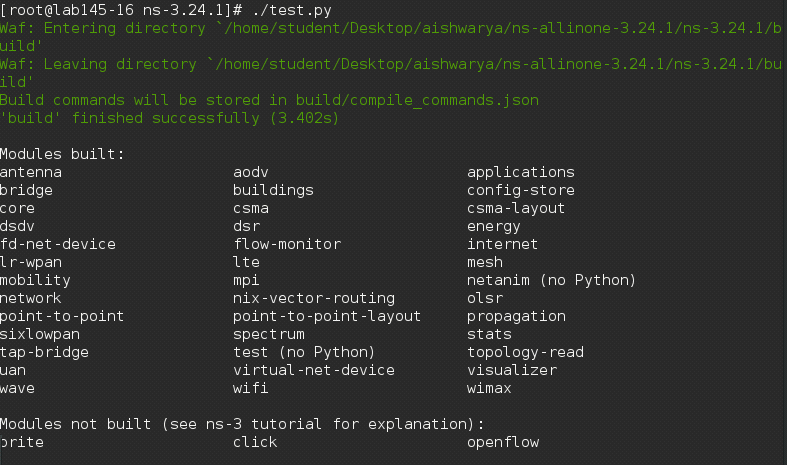


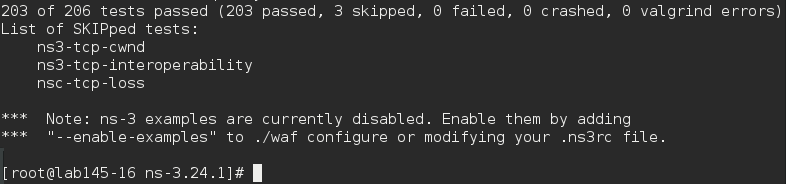


5-**./build.py --enable-tests**



6-**./test.py**





DATE: 08-02-17

**EXPERIMENT NO-4**

**AIM:** Program to connect two different nodes.

#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 ("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));

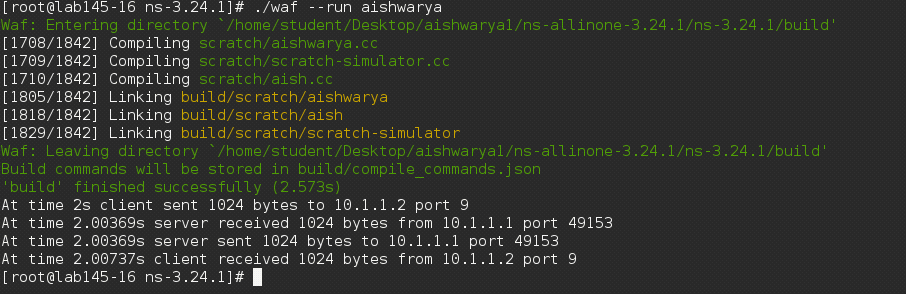
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT-**



DATE: 08-02-17

**Experiment No-5**

**AIM**: Program to connect three different nodes.

#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 (3);

PointToPointHelper pointToPoint;

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

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

PointToPointHelper pointToPoint1;

pointToPoint1.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint1.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer devices;

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

NetDeviceContainer devices1;

devices1 = pointToPoint1.Install (nodes.Get(1),nodes.Get(2));

InternetStackHelper stack;

stack.Install (nodes);

Ipv4AddressHelper address;

address.SetBase ("192.1.1.0", "255.255.255.0");

Ipv4AddressHelper address1;

address1.SetBase ("198.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign (devices);

Ipv4InterfaceContainer interfaces1 = address1.Assign (devices1);

UdpEchoServerHelper echoServer (100);

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

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (20.0));

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 100);

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

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

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

UdpEchoClientHelper echoClient1 (interfaces1.GetAddress (0), 100);

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

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

echoClient.SetAttribute ("PacketSize", UintegerValue (2024));

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

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

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

clientApps1.Start (Seconds (11.0));

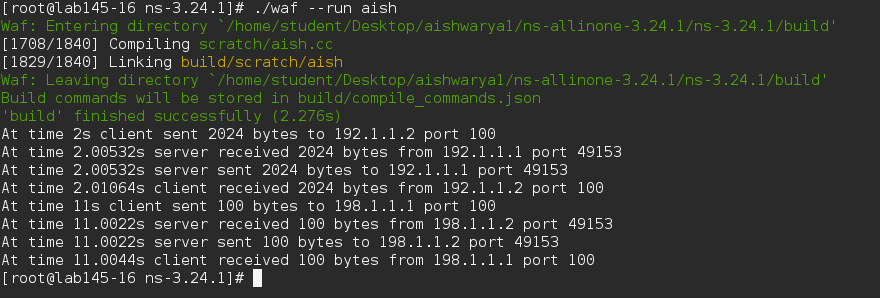
clientApps1.Stop (Seconds (12.0));

Simulator::Run ();

Simulator::Destroy ();

return 0;}

**OUTPUT:**



DATE: 15-02-17

**EXPERIMENT NO-6**

**AIM:** Program to connect point to point connection with bus topology.

#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"

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);

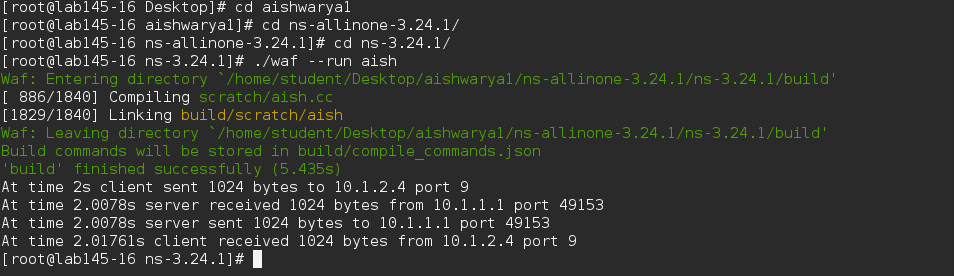
Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT-**



DATE: 15-02-17

**EXPERIMENT NO-7**

**AIM**: Program to connect two different point to point connections with bus topology.

#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"

using namespace ns3;

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

int

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

{

bool verbose = true;

uint32\_t nCsma = 2;

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 p2pNodes1;

p2pNodes1.Create (2);

NodeContainer p2pNodes2;

p2pNodes2.Create (2);

NodeContainer csmaNodes;

csmaNodes.Add (p2pNodes1.Get (1));

csmaNodes.Create (nCsma);

csmaNodes.Add (p2pNodes2.Get (0));

PointToPointHelper pointToPoint1;

pointToPoint1.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint1.SetChannelAttribute ("Delay", StringValue ("2ms"));

PointToPointHelper pointToPoint2;

pointToPoint2.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint2.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices1;

p2pDevices1 = pointToPoint1.Install (p2pNodes1);

NetDeviceContainer p2pDevices2;

p2pDevices2 = pointToPoint2.Install (p2pNodes2);

CsmaHelper csma;

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

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

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;

stack.Install (p2pNodes1.Get (0));

stack.Install (csmaNodes);

stack.Install (p2pNodes2.Get (1));

Ipv4AddressHelper address1;

address1.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer p2pInterfaces1;

p2pInterfaces1 = address1.Assign (p2pDevices1);

Ipv4AddressHelper address2;

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

Ipv4InterfaceContainer csmaInterfaces;

csmaInterfaces = address2.Assign (csmaDevices);

Ipv4AddressHelper address3;

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

Ipv4InterfaceContainer p2pInterfaces2;

p2pInterfaces2= address3.Assign (p2pDevices2);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (p2pNodes2.Get (1));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (p2pInterfaces2.GetAddress (1), 9);

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

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

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

ApplicationContainer clientApps = echoClient.Install (p2pNodes1.Get (0));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

pointToPoint1.EnablePcapAll ("second");

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

Simulator::Run ();

Simulator::Destroy ();

return 0;

}

**OUTPUT-**

