Practical - 02

Title: - Program to simulate traffic between two nodes

Aim: - To simulate traffic between two nodes

Lab Objectives: -

To get familiarize with a Network Simulation Tool

Description: -

A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another.

There are two possible types of connections:

1. point-to-point
2. multipoint.

Point-to-point connection

Point-to-point connection provides a dedicated link between two devices.

The entire capacity of the link is reserved for transmission between those two devices.

Use an actual length of wire or cable to connect the two ends, but other options, such as microwave or satellite links, are also possible.

For example, When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

Multipoint

A multipoint (also called multidrop) connection is one in which more than two specific devices share a single link

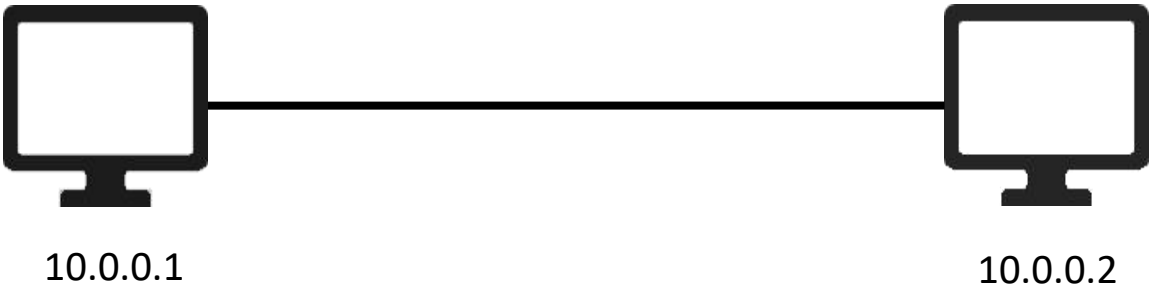
In a multipoint environment, the capacity of the channel is shared, either spatially or temporally.

If several devices can use the link simultaneously, it is a spatially shared connection.

If users must take turns, it is a timeshared connection.

Exercises

1. Write a program to implement the given point to point topology and simulate traffic between two nodes.



Code:

```
/*-*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*-*/
// add required header files
#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/netanim-module.h"
#include "ns3/csma-module.h"
#include "ns3/ipv4-global-routing-helper.h"
// Adding namespace declaration
using namespace ns3;
//Define log component where log msgs will be saved
NS_LOG_COMPONENT_DEFINE("p2pExample");
// Main function
int main(int argc, char *argv[]){
        // read the command line arguments
        CommandLine cmd(__FILE__);
```

```
// Process the command line arguments
cmd.Parse(argc, argv);
// Set time Resolution to 1 nano second
Time::SetResolution(Time::NS);
// Logging
LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication",LOG_LEVEL_INFO);
// Create NodeContainer object to store our nodes
NodeContainer nodes;
// Create 2 nodes
nodes.Create(2);
// create object of the point-to-point helper object class to configure net device and the channels
PointToPointHelper pointToPoint;
// Configure the net Device
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps")); // Set Data Rate
// Configure the Channel
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms")); // Set Delay Attribute
// Install net devices on nodes
NetDeviceContainer devices;
devices=pointToPoint.Install(nodes);
// install netdevices on node and connect with the Channels
// Configure and Install protocol suits on nodes
InternetStackHelper stack;
stack.Install (nodes);
// configure network IP address and subnet mask for network
Ipv4AddressHelper address;
// set data
```

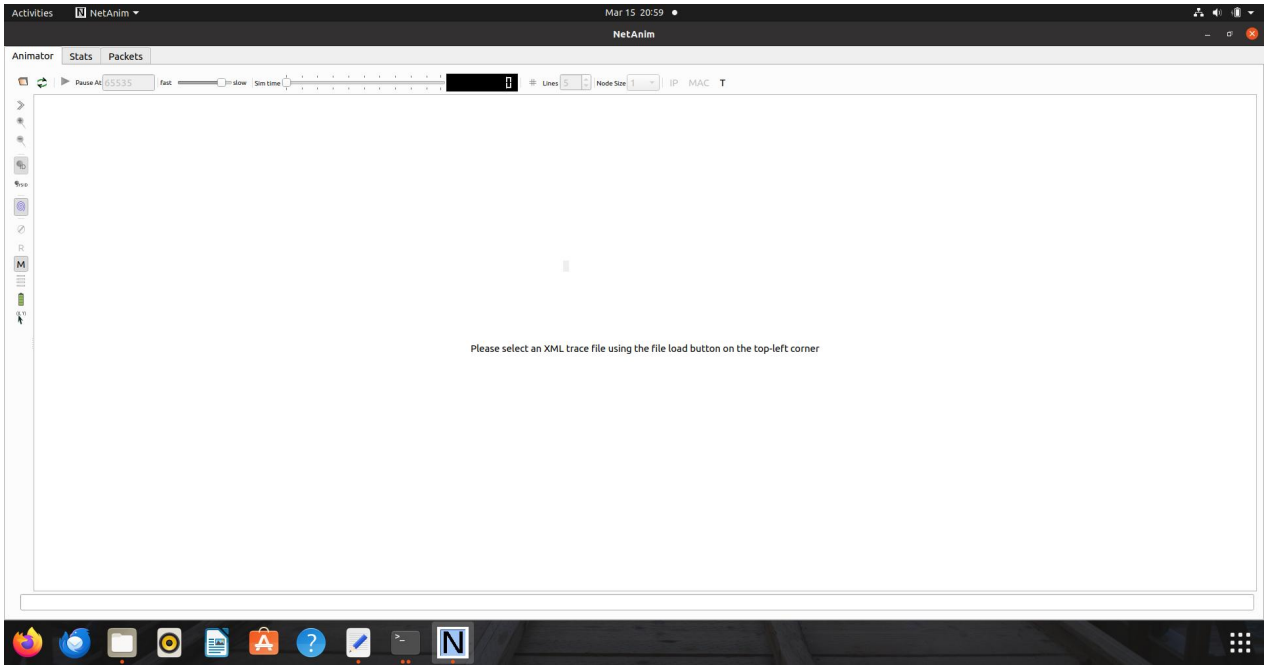
```
address.SetBase("10.0.0.0","255.0.0.0");
// Assign IP addresses to the interfaces of netDevices
Ipv4InterfaceContainer interfaces = address.Assign(devices);
// Configure our Applications
// Configure UDPEchoServerApplication
UdpEchoServerHelper echoServer(9); // Setting port number of server application
// Application Container create object to store server application and install on node(1)
ApplicationContainer serverApp = echoServer.Install(nodes.Get(1)); // indexed 1 server
// Configure start and stop time of server Application
serverApp.Start(Seconds(1.0)); // server app should start first
serverApp.Stop(Seconds(10.0)); // server app should stop
// Configure UdpEchoClientApplication
UdpEchoClientHelper echoClient(interfaces.GetAddress(1),9);
// Configure the attribute of client Application
echoClient.SetAttribute("MaxPackets", UintegerValue (1));
echoClient.SetAttribute("Interval", TimeValue (Seconds(1.0)));
echoClient.SetAttribute("PacketSize", UintegerValue (1024));
// Install Client Application on Node 0
ApplicationContainer clientApp = echoClient.Install(nodes.Get(0));
// Enables Routing between two networks 10.0.0.0 and 20.0.0.0
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
// for Running the code
AnimationInterface anim("p2pAniExcer.xml");
anim.SetConstantPosition(nodes.Get(0),20.0,30.0);
anim.SetConstantPosition(nodes.Get(1),40.0,30.0);
// Configure Start and Stop Time
clientApp.Start(Seconds(2.0));
```

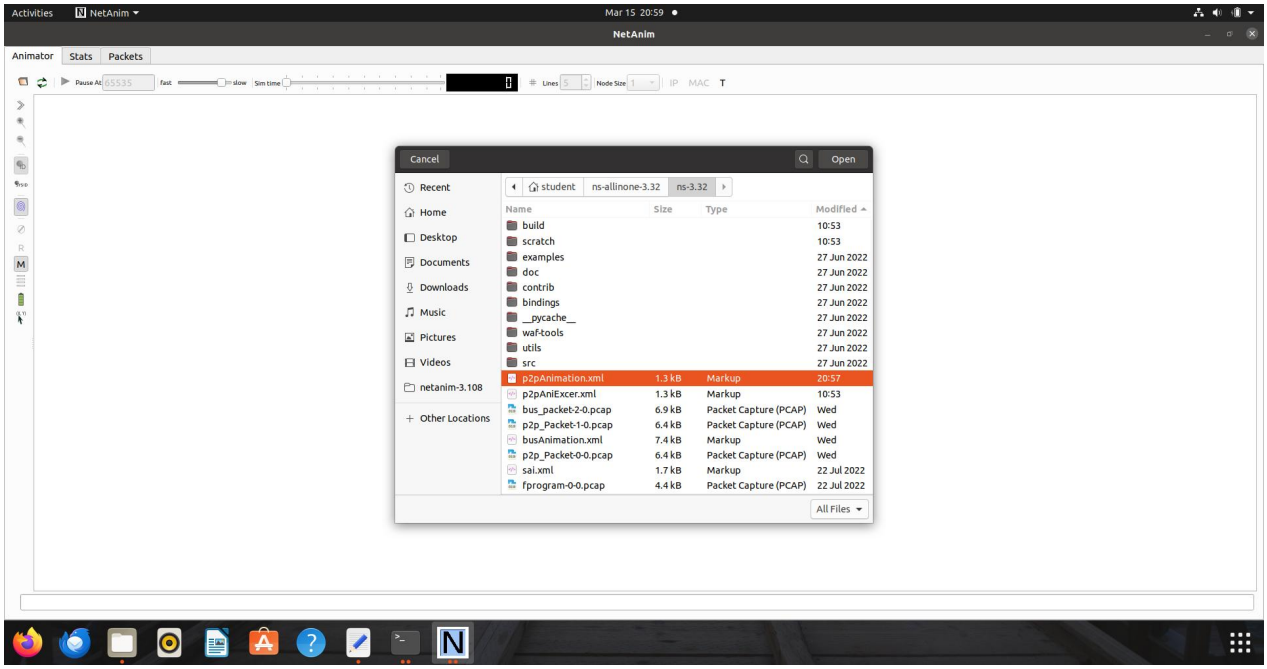
```
    clientApp.Stop(Seconds(10.0));
    // Simulation on Run and start
    Simulator::Run();
    // Destory this Resourses
    Simulator::Destroy();
    return 0;
}
```

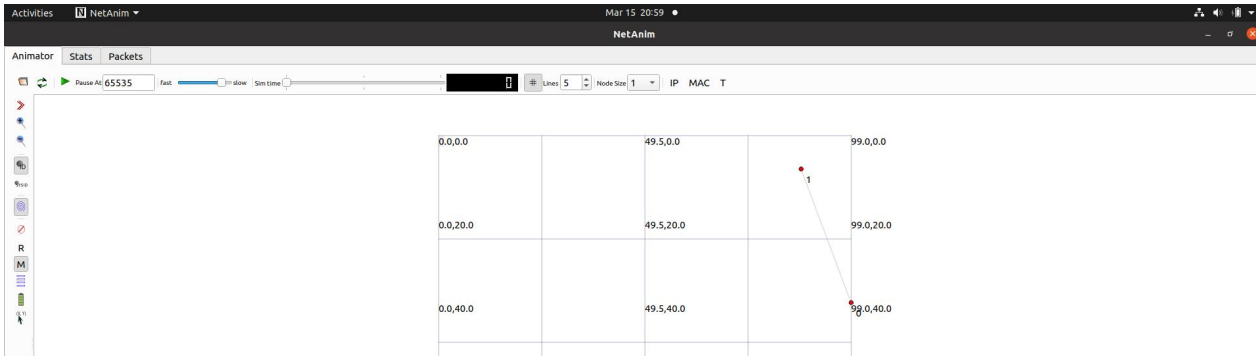
Output

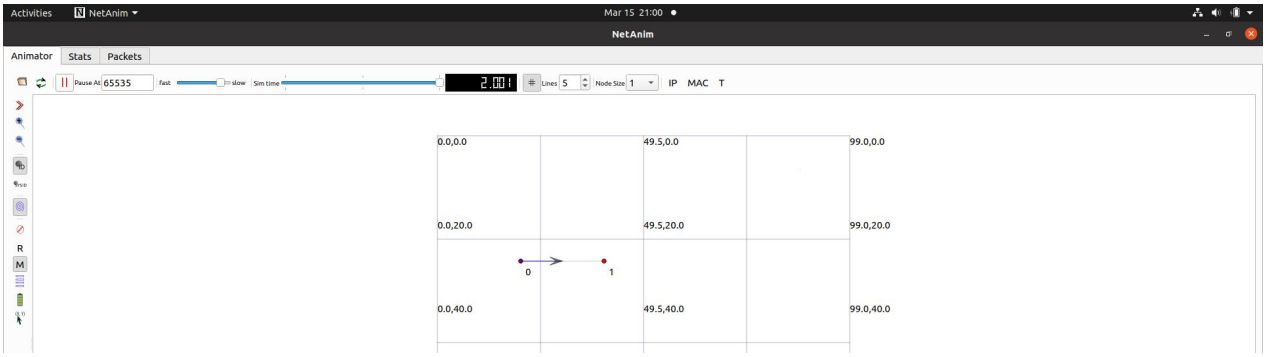
```
Activities   Terminal ▾                                                    Mar 15 20:57
                                                          student@student-VirtualBox: ~/n
student@student-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf
Waf: Entering directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.034s)

Modules built:
antenna                   aodv                      applications
bridge                    buildings                 config-store
core                      csma                      csma-layout
dsdv                      dsr                       energy
fd-net-device             flow-monitor              internet
internet-apps             lr-wpan                   lte
mesh                      mobility                  netanim
network                   nix-vector-routing        olsr
point-to-point            point-to-point-layout     propagation
sixlowpan                 spectrum                  stats
tap-bridge                test (no Python)          topology-read
traffic-control           uan                       virtual-net-device
wave                      wifi                      wimax

Modules not built (see ns-3 tutorial for explanation):
brite                     click                     dpdk-net-device
mpi                       openflow                  visualizer

student@student-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/p2p
Waf: Entering directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.597s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
At time +2s client sent 1024 bytes to 10.0.0.2 port 9
At time +2.00369s server received 1024 bytes from 10.0.0.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.0.0.1 port 49153
At time +2.00737s client received 1024 bytes from 10.0.0.2 port 9
student@student-VirtualBox:~/ns-allinone-3.32/ns-3.32$
```

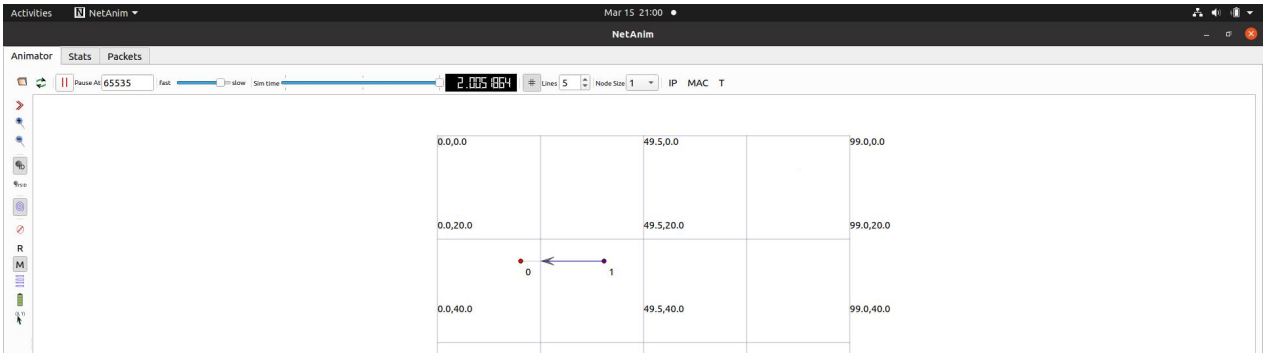
```
Activities   Terminal ▾

student@student-VirtualBox:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim
```


Activities
NetAnim
Mar 15 20:59
NetAnim
Animator
Stats
Packets
Please select an XML trace file using the file load button on the top-left corner


Activities
NetAnim
Mar 15 20:59
NetAnim
Animator
Stats
Packets
Cancel
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
student
ns-allinone-3.32
ns-3.32
Name
Size
Type
Modified
build
scratch
examples
doc
contrib
bindings
__pycache__
waf-tools
utils
src
p2pAnimation.xml
p2pAniExcer.xml
bus_packet-2-0.pcap
p2p_Packet-1-0.pcap
busAnimation.xml
p2p_Packet-0-0.pcap
sai.xml
fprogram-0-0.pcap
All Files


Activities
NetAnim
Mar 15 20:59
NetAnim
Animator
Stats
Packets
Pause At 65535
Lines 5
Node Size 1
IP
MAC
T
0.0,0.0
49.5,0.0
99.0,0.0
0.0,20.0
49.5,20.0
99.0,20.0
0.0,40.0
49.5,40.0
99.0,40.0





Conclusion: Implemented point to point topology and simulate traffic between two nodes

After performing this Practical/lab, students are expected to answer the following questions.

1. What is the Boilerplate?
2. What are the seven levels of log messages?
3. Explain the following classes and methods in those classes which are used in your script.
    a. NodeContainer
    b. PointToPointHelper
    c. NetDeviceContainer
    d. InternetStackHelper
    e. Ipv4AddressHelper
    f. Ipv4InterfaceContainer

Reference

https://www.nsnam.org/docs/release/3.15/doxygen/classns3_1_1_node_container.html#details

https://www.nsnam.org/docs/release/3.15/doxygen/classns3_1_1_point_to_point_helper.html#details

https://www.nsnam.org/docs/release/3.15/doxygen/classns3_1_1_net_device_container.html

https://www.nsnam.org/docs/release/3.15/doxygen/classns3_1_1_internet_stack_helper.html

https://www.nsnam.org/docs/release/3.15/doxygen/classns3_1_1_ipv4_address_helper.html

https://www.nsnam.org/docs/release/3.15/doxygen/classns3_1_1_ipv4_interface_container.html

https://www.nsnam.org/docs/tutorial/html/conceptual-overview.html

https://www.nsnam.org/docs/release/3.7/tutorial/tutorial_21.html