Computer Networks Lab Manual

(Course Code: B19CS5080)

3rd Year—5thSem

Batch 2019-2023

SCHOOL OF COMPUTING
AND
INFORMATION TECHNOLOGY

CONTENTS

I. **Lab DESCRIPTION**

This laboratory course supplements the material taught in the theory course Computer Networks. The objective of this lab is to get hands-on experience in Computer Networks concept using simulation tools Viz. NS3, NetAnim, Wireshark, Packet Tracer and Nmap. Laboratory exercises will normally be conducted on UNIX Operating system. The students will be exposed to simulating and analyzing concepts.

II. LAB OBJECTIVES

The objectives of this lab course are to make students to:

- Understand Computer Network Concepts.
- Install simulation tools like NS3,NetAnim,Wireshark,Packet Tracer and Nmap
- Analyze the Network performance concepts and Traffic flow analysis for wired and wireless protocols.
- Build virtual LAN's using NS3 and Packet Tracer.

III. LAB OUTCOMES

On successful completion of this course; the student will be able to:

CO1: Describe the knowledge of the basic protocols involved in wired/wireless communication process.

CO2: Identify the key issues for the realization of errors detection and correction in Packets.

CO3: Analyse different network performance concepts and traffic issues for Quality of Service (QoS) in communication.

CO4: Implement concepts of classical computer and network security paradigms for Real world Applications.

IV. LAB REQUIREMENTS

Followingarethe requiredhardwareandsoftwareforthislab, which is availableinthelaboratory.

- ✓ **Hardware:** Desktop system or Virtualmachine in a cloud with OS installed.Presently in the Lab, i3Process or having 4GBRAM and 250GB HardDisk is available.Desktop systems are dual boot having Windows as well as LinuxOS installed on them.
- ✓ **Software:** NS3 compiler with other tools viz. NS3, NetAnim, Wireshark, Packet Tracer and Nmap.

Program 1

1 Problem Statement

Introduction to: (a) discrete event simulation, (b) NS3, (c) NS 3 Installation, (d) NetAnim.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to

- Understand the meaning of dicrete event simulatior like NS3
- Gain knowledge to install the ns3 software in Ubuntu.
- Understand NetAnim and its installation.

(a) Introduction to dicrete event simulation

System: A collection of entities that act and interact together toward the accomplishment of some logical end.

Discrete system: State variables change instantaneously at separated point in time, e.g., a bank, since state variables - number of customers, change only when a customer arrives or when a customer finishes being served and departs

Continuous system: State variable change continuously with respect to time, e.g., airplane moving through the air, since state variables - position and velocity change continuously with respect to time

Why Simulation?

- Many systems are highly complex, precluding the possibility of analytical solution
- The analytical solutions are extraordinarily complex, requiring vast computing resources
- Thus, such systems should be studied by means of simulation

Numerically exercising the model for inputs in question to see how they affect the output measures of performance

“Simulationis the process of designinga model of a real system and conductingexperimentswith this model for the purpose eitherof understandingthe behavior of the system or of evaluatingvarious strategies (within the limits imposed by a criterion or set of criteria) for theoperationof a system.”

What is Discrete-Event Simulation (DES)?

A discrete-event simulationmodels a system whose state may change only at discrete pointmodels a system whose state may change only at discrete pointin time.

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

State:a collection of attributes or state variables that represent theentities of the system.

Event:an instantaneous occurrence in time that may alter the state ofthe system

(b) Introduction to ns3

ns-3 is a discrete-event network simulator, 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.
The goal of the ns-3 project is to develop a preferred, open simulation environment for networking research: it should be aligned with the simulation needs of modern networking research and should encourage community contribution, peer review, and validation of the software.

(c) ns3 installation

Download the ns3 package (ns-allinone-3.28 or any other version) from https://www.nsnam.org and follow following steps:

```
$cd Desktop

$cd ns-allinone-3.28

$./build.py --enable-examples --enable-test

$cd ns-3.28

$./waf --build-profile=debug --enable-examples --enable-test configure
```

once all the configuration is done NS-3 is inastallation is completed.

To ckeck whether NS-3 is successfully installed verify with this:

```
$./waf --run hello-simulator
```

U will see the output as hello-simulator

(d) NetAnim Installation

```
$ Sudo apt install qt4-default qt4-make
$ cd NetAnim
$ qmake NetAnim.pro
$ make
```

<u>Execution of NetAnim</u>

```
$ Cd to NetAnim from ns3.28...
$ ./NetAnim
```

Program 2

1 Problem Statement

Write a NS3 program to connect two nodes with a point to pint link, which have unique interface. Analyze the network performance using UDP client server.

2 Student Learning Outcomes

After successful completion of this lab, the student's shall be able to

- Understand to establish point to point network.
- Create an interface to point to point link
- Anylize the Network performance usibg client server

3 Design of the Program

3.1 Theory

Point to Point Network

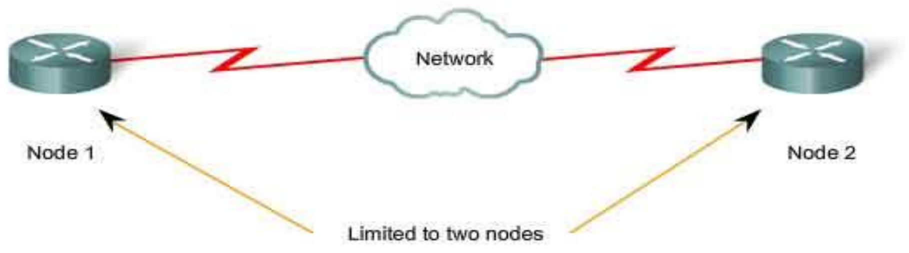


Fig:Point to Point Network

This type of network consists of many connections between individual pairs of machines. To go from the source to the destination, a packet of information on this type of network may have to first visit one or more intermediate machines. Often multiple routes, of different length are possible, so routing algorithms play an important role in point-to-point networks.

Network performance

Network Performance refers to measures of service quality of a network

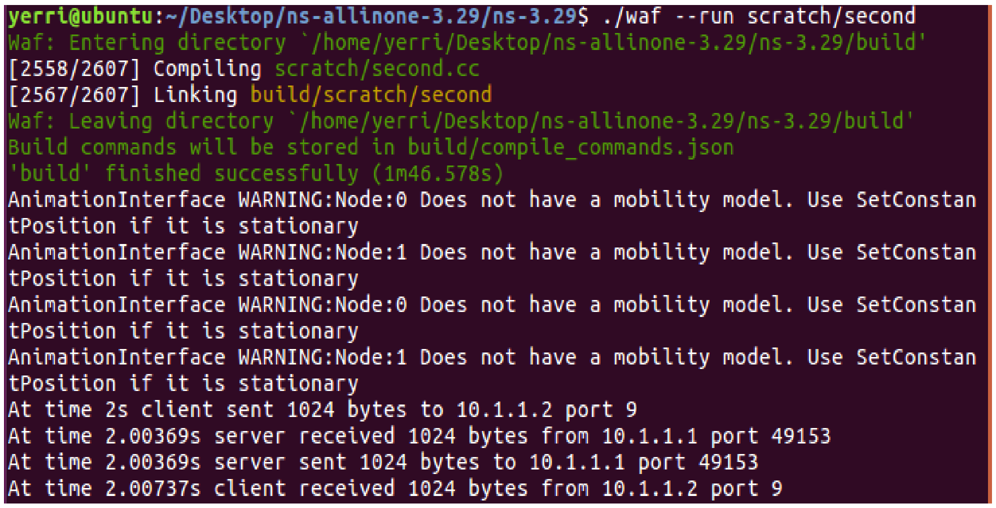
UDP: User Datagram Protocol

The User Datagram Protocol (UDP) is the simpler of the two standard TCP/IP transport protocols. It is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from the upper layer.

3.2 Coding using C++ Language

```
#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"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");


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

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

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

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));
  AnimationInterface anim ("first.xml");
  anim.SetConstantPosition(nodes.Get (0), 10.0, 10.0);
  anim.SetConstantPosition(nodes.Get (1), 20.0, 30.0);
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

3.3 Expected Results



```
yerri@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29$ ./waf --run scratch/second
Waf: Entering directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
[2558/2607] Compiling scratch/second.cc
[2567/2607] Linking build/scratch/second
Waf: Leaving directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1m46.578s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstan
tPosition if it is stationary
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
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 9
```

Program 3

1 Problem Statement

Write a NS 3 program to demonstrate bus topology. Analyze the performance using UDP based applications.

2 Student Learning Outcomes

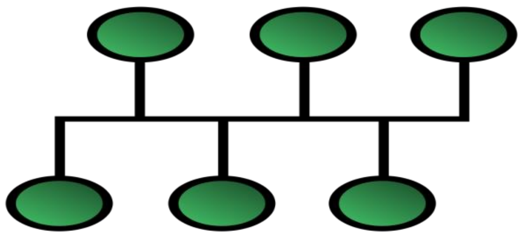
After successful completion of this lab, the student will able to

- Create Bus Topology
- Analyze the performance using UDP based applications.

3 Design of the Program

3.1 Theory

Bus Topology



A **bus network** is a network topology in which nodes are directly connected to a common half-duplex link called a bus

A host on a bus network is called a station. In a bus network, every station will receive all network traffic, and the traffic generated by each station has equal transmission priority.[3] A bus network forms a single network segment and collision domain. In order for nodes to share the bus, they use a medium access control technology such as carrier-sense multiple access (CSMA) or a bus master.

etwork performance

Network Performance refers to measures of service quality of a network

UDP: User Datagram Protocol

The User Datagram Protocol (UDP) is the simpler of the two standard TCP/IP transport protocols. It is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from the upper layer.

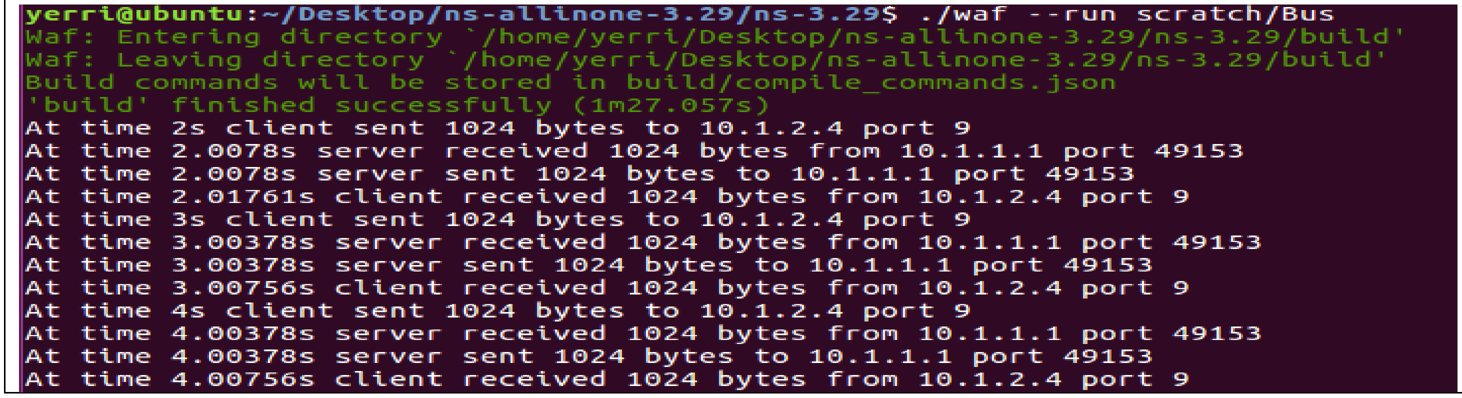
3.2 Coding using C++ Language

```
#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"
#include "ns3/netanim-module.h"
// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                   ================
//                    LAN 10.1.2.0-csma
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 (3));
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 ("p2p");
csma.EnablePcap ("csma1", csmaDevices.Get (1), true);
csma.EnablePcap ("csma2", csmaDevices.Get (2), true);
csma.EnablePcap ("csma3", csmaDevices.Get (3), true);
  AnimationInterface anim("bus.xml");
  anim.SetConstantPosition(p2pNodes.Get(0),10.0,10.0);
  anim.SetConstantPosition(csmaNodes.Get(0),20.0,20.0);
  anim.SetConstantPosition(csmaNodes.Get(1),30.0,30.0);
  anim.SetConstantPosition(csmaNodes.Get(2),40.0,40.0);
  anim.SetConstantPosition(csmaNodes.Get(3),50.0,50.0);
Simulator::Run ();
Simulator::Destroy ();
return 0;
```

```
}
```

3.3	**Expected Results**



```
yerri@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29$ ./waf --run scratch/Bus
Waf: Entering directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1m27.057s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 9
At time 3s client sent 1024 bytes to 10.1.2.4 port 9
At time 3.00378s server received 1024 bytes from 10.1.1.1 port 49153
At time 3.00378s server sent 1024 bytes to 10.1.1.1 port 49153
At time 3.00756s client received 1024 bytes from 10.1.2.4 port 9
At time 4s client sent 1024 bytes to 10.1.2.4 port 9
At time 4.00378s server received 1024 bytes from 10.1.1.1 port 49153
At time 4.00378s server sent 1024 bytes to 10.1.1.1 port 49153
At time 4.00756s client received 1024 bytes from 10.1.2.4 port 9
```

Program.4

1	**Problem Statement**

Write a NS 3 program to demonstrate star topology. Analyze the performance using UDP based applications.

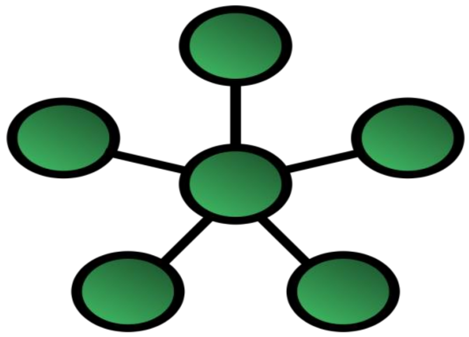
2	**Student Learning Outcomes**

After successful completion of this lab, the student will able to

- Create Star Topology
- Analyze the performance using UDP based applications

3	**Design of the Program**
3.1	**Theory**

Star Topology



A star network is an implementation of a spoke–hub distribution paradigm in computer networks. In a star network, every host is connected to a central hub. In its simplest form, one central hub acts as a conduit to transmit messages.[1] The star network is one of the most common computer network topologies.

The hub and hosts, and the transmission lines between them, form a graph with the topology of a star. Data on a star network passes through the hub before continuing to its destination. The hub manages and controls all functions of the network. It also acts as a repeater for the data flow.

The star topology reduces the impact of a transmission line failure by independently connecting each host to the hub. Each host may thus communicate with all others by transmitting to, and receiving from, the hub. The failure of a transmission line linking any host to the hub will result in the isolation of that host from all others, but the rest of the network will be unaffected.

Network performance

Network Performance refers to measures of service quality of a network

UDP: User Datagram Protocol

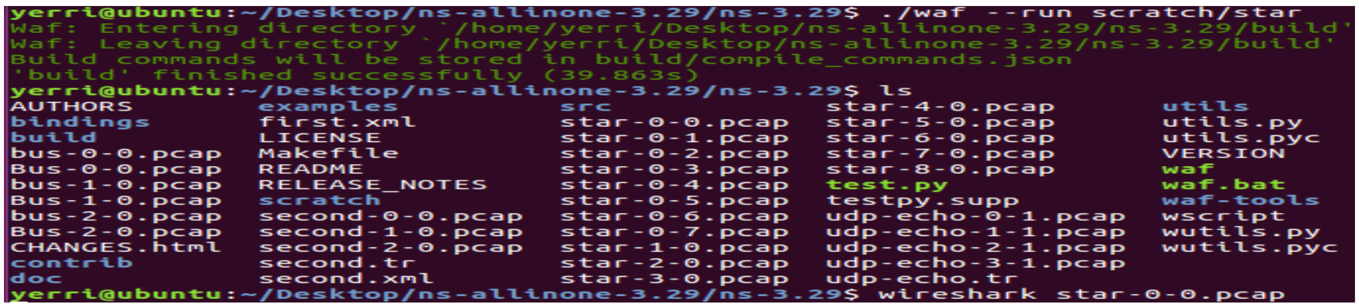
The User Datagram Protocol (UDP) is the simpler of the two standard TCP/IP transport protocols. It is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from the upper layer.

3.2 Coding using C++ Language

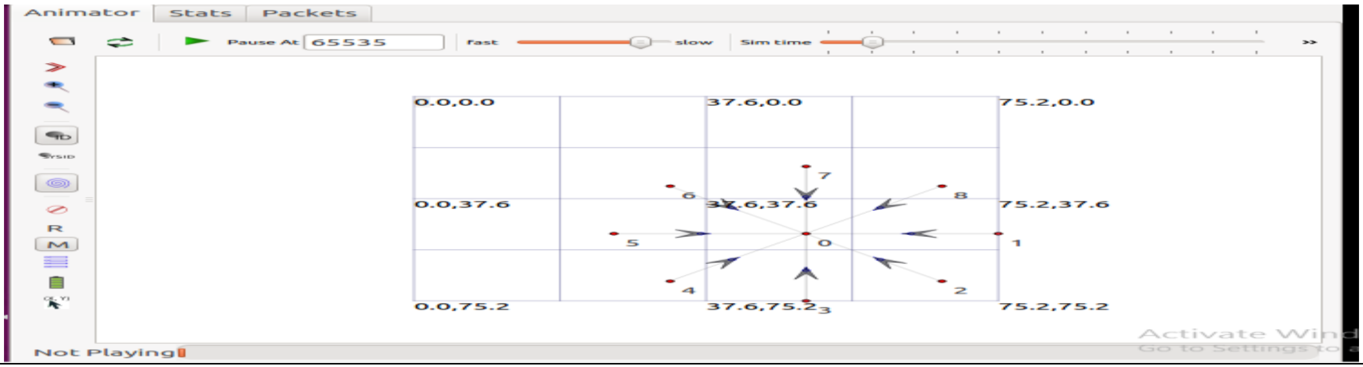
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/netanim-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-layout-module.h"
```

```
// Network topology (default)
//
//       n2 n3 n4              .
//        \ | /                .
//         \|/                 .
//    n1--- n0---n5            .
//         /|\                 .
//        / | \                .
//       n8 n7 n6              .
//
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("Star123");
int
main (int argc, char *argv[])
{
  //
  // Set up some default values for the simulation.
  //
  Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (137));
  // ??? try and stick 15kb/s into the data rate
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("14kb/s"));
  //
  // Default number of nodes in the star.  Overridable by command line argument.
  //
  uint32_t nSpokes = 8;
        std::string animFile1 = "star-animation1.xml";
  CommandLine cmd;
  cmd.AddValue ("nSpokes", "Number of nodes to place in the star", nSpokes);
cmd.AddValue ("animFile1",  "File Name for Animation Output", animFile1);
  cmd.Parse (argc, argv);
  NS_LOG_INFO ("Build star topology.");
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  PointToPointStarHelper star (nSpokes, pointToPoint);
  NS_LOG_INFO ("Install internet stack on all nodes.");
  InternetStackHelper internet;
  star.InstallStack (internet);
  NS_LOG_INFO ("Assign IP Addresses.");
  star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));
  NS_LOG_INFO ("Create applications.");
  //
  // Create a packet sink on the star "hub" to receive packets.
  //
  uint16_t port = 50000;
  Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", hubLocalAddress);
  ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());
  hubApp.Start (Seconds (1.0));
  hubApp.Stop (Seconds (10.0));
  //
  // Create OnOff applications to send TCP to the hub, one on each spoke node.
  //
  OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
  onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  ApplicationContainer spokeApps;
  for (uint32_t i = 0; i < star.SpokeCount (); ++i)
    {
      AddressValue remoteAddress (InetSocketAddress (star.GetHubIpv4Address (i), port));
      onOffHelper.SetAttribute ("Remote", remoteAddress);
      spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
    }
  spokeApps.Start (Seconds (1.0));
  spokeApps.Stop (Seconds (10.0));
  NS_LOG_INFO ("Enable static global routing.");
  //
  // Turn on global static routing so we can actually be routed across the star.
  //
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  NS_LOG_INFO ("Enable pcap tracing.");
  //
  // Do pcap tracing on all point-to-point devices on all nodes.
  //
  pointToPoint.EnablePcapAll ("star123");
// Set the bounding box for animation
  star.BoundingBox (1, 1, 100, 100);
  // Create the animation object and configure for specified output
  AnimationInterface anim (animFile1);
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
  return 0;
}
```

3.3 Expected Results



```
yerri@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29$ ./waf --run scratch/star
Waf: Entering directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (39.863s)
yerri@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29$ ls
AUTHORS          examples            src              star-4-0.pcap        utils
bindings         first.xml           star-0-0.pcap    star-5-0.pcap        utils.py
build            LICENSE             star-0-1.pcap    star-6-0.pcap        utils.pyc
bus-0-0.pcap     Makefile            star-0-2.pcap    star-7-0.pcap        VERSION
Bus-0-0.pcap     README              star-0-3.pcap    star-8-0.pcap        waf
bus-1-0.pcap     RELEASE_NOTES       star-0-4.pcap    test.py              waf.bat
Bus-1-0.pcap     scratch             star-0-5.pcap    testpy.supp          waf-tools
bus-2-0.pcap     second-0-0.pcap     star-0-6.pcap    udp-echo-0-1.pcap    wscript
Bus-2-0.pcap     second-1-0.pcap     star-0-7.pcap    udp-echo-1-1.pcap    wutils.py
CHANGES.html     second-2-0.pcap     star-1-0.pcap    udp-echo-2-1.pcap    wutils.pyc
contrib          second.tr           star-2-0.pcap    udp-echo-3-1.pcap
doc              second.xml          star-3-0.pcap    udp-echo.tr
yerri@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29$ wireshark star-0-0.pcap
```



Program 5

1 Problem Statement

Write a NS3 program to implement FTP using TCP bulk transfer, Analyze the performance

2 Student Learning Outcomes

After successful completion of this lab, the student will able to

- Impliment FTP protocol to analyze the flow of bulk transfer.
- Anlyze the performance of Application Protocols.

3 Design of the Program

3.1 Theory

File Transfer Protocol

The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

FTP is built on a client-server model architecture using separate control and data connections between the client and the server.[1] FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP)

TCP Bulk Data Flow

The sender of a data block required an acknowledgment for that block before the next block was sent. In this chapter we'll see that TCP uses a different form of flow control called a sliding window protocol. It allows the sender to transmit multiple packets before it stops and waits for an acknowledgment. This leads to faster data transfer, since the sender doesn't have to stop and wait for an acknowledgment each time a packet is sent.

Network performance

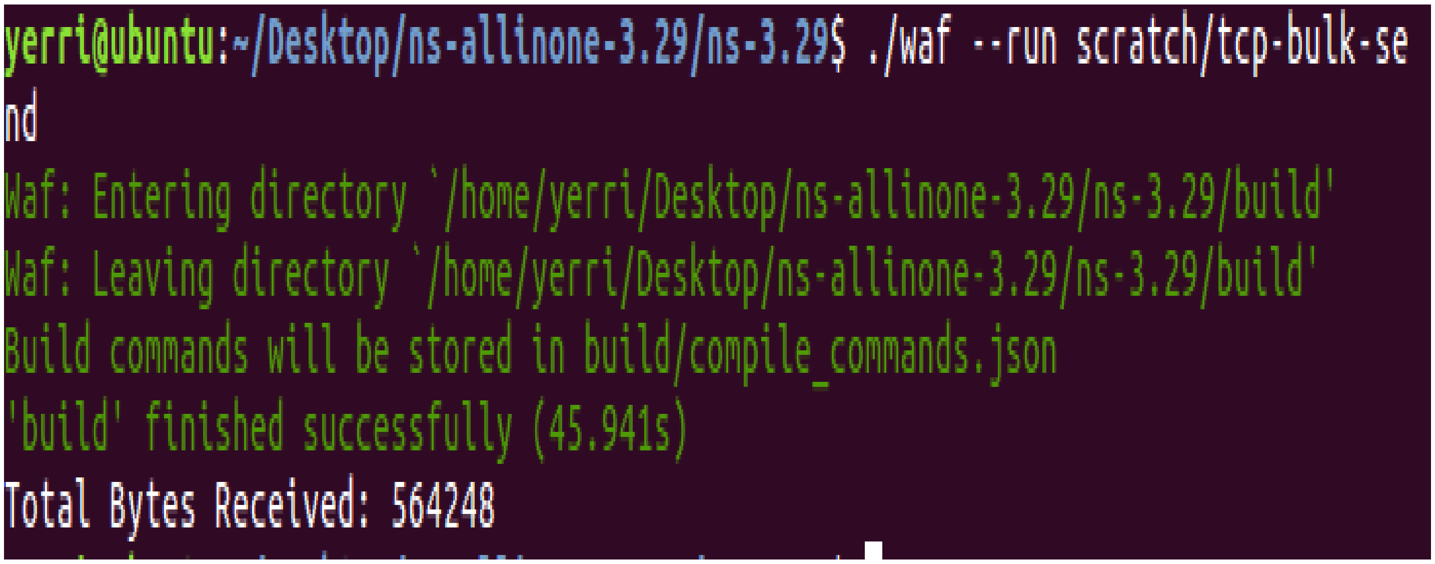
Network Performance refers to measures of service quality of a network

3.2 Coding using C++ Language

```
// Network topology
//
//      n0 ----------- n1
//           500 Kbps
//            5 ms
//
// - Flow from n0 to n1 using BulkSendApplication.
// - Tracing of queues and packet receptions to file "tcp-bulk-send.tr"
//   and pcap tracing available when tracing is turned on.

#include <string>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/network-module.h"
#include "ns3/packet-sink.h"

using namespace ns3;
```

```
NS_LOG_COMPONENT_DEFINE ("TcpBulkSendExample");

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

  bool tracing = false;
  uint32_t maxBytes = 0;

// Allow the user to override any of the defaults at
// run-time, via command-line arguments
  CommandLine cmd;
  cmd.AddValue ("tracing", "Flag to enable/disable tracing", tracing);
  cmd.AddValue ("maxBytes",
                "Total number of bytes for application to send", maxBytes);
  cmd.Parse (argc, argv);

// Explicitly create the nodes required by the topology (shown above).
  NS_LOG_INFO ("Create nodes.");
  NodeContainer nodes;
  nodes.Create (2);

  NS_LOG_INFO ("Create channels.");

// Explicitly create the point-to-point link required by the topology (shown above).
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
// Install the internet stack on the nodes
  InternetStackHelper internet;
  internet.Install (nodes);

// We've got the "hardware" in place.  Now we need to add IP addresses.

  NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);

  NS_LOG_INFO ("Create Applications.");

// Create a BulkSendApplication and install it on node 0
  uint16_t port = 9;  // well-known echo port number


  BulkSendHelper source ("ns3::TcpSocketFactory",
                         InetSocketAddress (i.GetAddress (1), port));
  // Set the amount of data to send in bytes.  Zero is unlimited.
  source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
  ApplicationContainer sourceApps = source.Install (nodes.Get (0));
  sourceApps.Start (Seconds (0.0));
  sourceApps.Stop (Seconds (10.0));

// Create a PacketSinkApplication and install it on node 1
  PacketSinkHelper sink ("ns3::TcpSocketFactory",
                         InetSocketAddress (Ipv4Address::GetAny (), port));
  ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
  sinkApps.Start (Seconds (0.0));
  sinkApps.Stop (Seconds (10.0));

// Set up tracing if enabled
  if (tracing)
    {
      AsciiTraceHelper ascii;
      pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("tcp-bulk-send.tr"));
      pointToPoint.EnablePcapAll ("tcp-bulk-send", false);
    }

// Now, do the actual simulation.
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds (10.0));
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");

  Ptr<PacketSink> sink1 = DynamicCast<PacketSink> (sinkApps.Get (0));
  std::cout << "Total Bytes Received: " << sink1->GetTotalRx () << std::endl;
```

3.3 Expected Results



```
yerri@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29$ ./waf --run scratch/tcp-bulk-se
nd
Waf: Entering directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (45.941s)
Total Bytes Received: 564248
```

Program 6

1 Problem Statement

Write a NS3 program to connect two nodes with a point to point link, which have unique interface. Analyse the traffic control using TCP by changing suitable parameters.

2 Student Learning Outcomes

After successful completion of this lab, the student will able to

- Create Point to point link with unique interface
- Analyze the flow of traffic control of Application Protocols.

3 Design of the Program

3.1 Theory

Network Traffic Control

In computer networking, network traffic control is the process of managing, controlling or reducing the network traffic, particularly Internet bandwidth, e.g. by the network scheduler.[1] It is used by network administrators, to reduce congestion, latency and packet loss. This is part of bandwidth management. In order to use these tools effectively, it is necessary to measure the network traffic to determine the causes of network congestion and attack those problems specifically.

Network traffic control is an important subject in datacenters as it is necessary for efficient use of datacenter network bandwidth and for maintaining service level agreements

TCP: Transmission Control Protocol

The Transmission Control Protocol (TCP) provides full transport-layer services to applications.TCP is a reliable stream transport protocol. The term stream, in this context, means connection- oriented: A connection must be established between both ends of a transmission before either can transmit data.

At the sending end of each transmission, TCP divides a stream of data into smaller units called segments. Each segment includes a sequence number for reordering after receipt, together with an acknowledgment number for the segments received. Segments are carried across the internet inside of IP datagrams. At the receiving end, TCP collects each datagram as it comes in and reorders the transmission based on sequence numbers

3.2 Coding using C++ Language

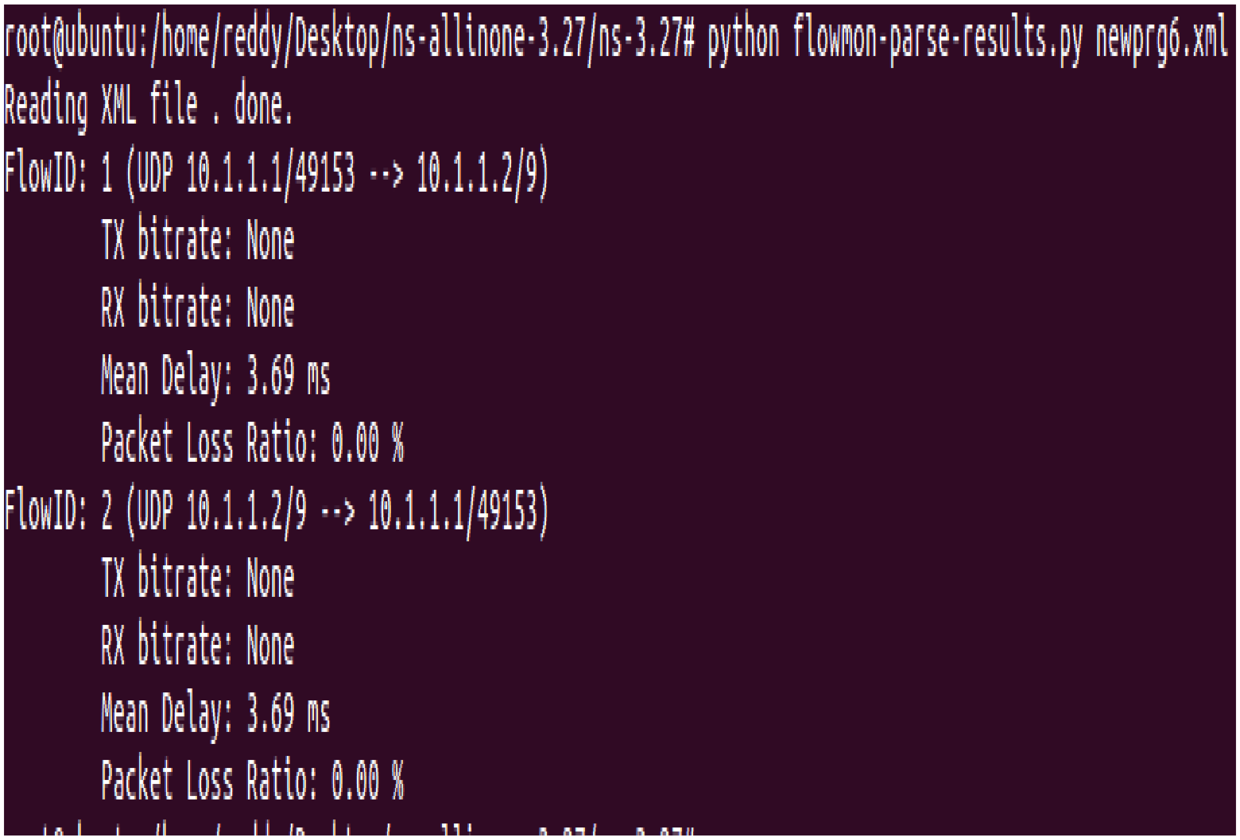
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"


//step1: add the following header files
#include "ns3/flow-monitor.h"

#include "ns3/flow-monitor-helper.h"
#include "ns3/traffic-control-module.h"



using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

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

//step2: declare the variable tracing
bool tracing = false;

 CommandLine cmd;
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

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

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

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

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (6));
  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));




//step3: add the following code for  Flow monitor
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll();

Simulator::Stop(Seconds(10.0));

 if (tracing==true)
    {

pointToPoint.EnablePcapAll ("p2p");

}
```

```
  Simulator::Run ();

//step 4: add the following statement for xml file
flowMonitor->SerializeToXmlFile("newprg6.xml", true, true);
  Simulator::Destroy ();
  return 0;
```

3.3 Expected Results

Steps for Exexution:

run the program: ./waf --run scratch/newprg6 to genetrate xml file.
check the xml file
copy the file "flowmon-parse-results.py" from src/flow monitor/examples to /ns-3.27 folder
use the following command to read the xml file
python flowmon-parse-results.py newprg6.xml



```
root@ubuntu:/home/reddy/Desktop/ns-allinone-3.27/ns-3.27# python flowmon-parse-results.py newprg6.xml
Reading XML file . done.
FlowID: 1 (UDP 10.1.1.1/49153 --> 10.1.1.2/9)
	TX bitrate: None
	RX bitrate: None
	Mean Delay: 3.69 ms
	Packet Loss Ratio: 0.00 %
FlowID: 2 (UDP 10.1.1.2/9 --> 10.1.1.1/49153)
	TX bitrate: None
	RX bitrate: None
	Mean Delay: 3.69 ms
	Packet Loss Ratio: 0.00 %
```

Program 7

1 Problem Statement

Write NS 3 Program to configure two nodes on an 802.11b physical layer, with802.11b NICs in adhoc mode, and by default, sends one packet of 1000 (application) bytes to the other node. The physical layer is configured to receive at a fixed RSS (regardless of the distance and transmit power); therefore, changing position of the nodes has no effect. Analyze the performance.

2 Student Learning Outcomes

After successful completion of this lab, the student will able to

- Implement the wireless Technology in Adhoc mode.
- Anayze the performace by changing the position of nodes.

3 Design of the Program

3.1 Theory

802.11ax Wi-Fi

The next-generation 802.11ax Wi-Fi standard, also known as Wi-Fi 6, is the latest step in a journey of nonstop innovation. The standard builds on the strengths of 802.11ac, adding efficiency, flexibility, and scalability that allows new and existing networks increased speed and capacity with next-generation applications.

The Institute of Electrical and Electronics Engineers (IEEE) proposed the 802.11ax standard so that it can couple the freedom and high speed of Gigabit Ethernet wireless with the reliability and predictability found in licensed radio

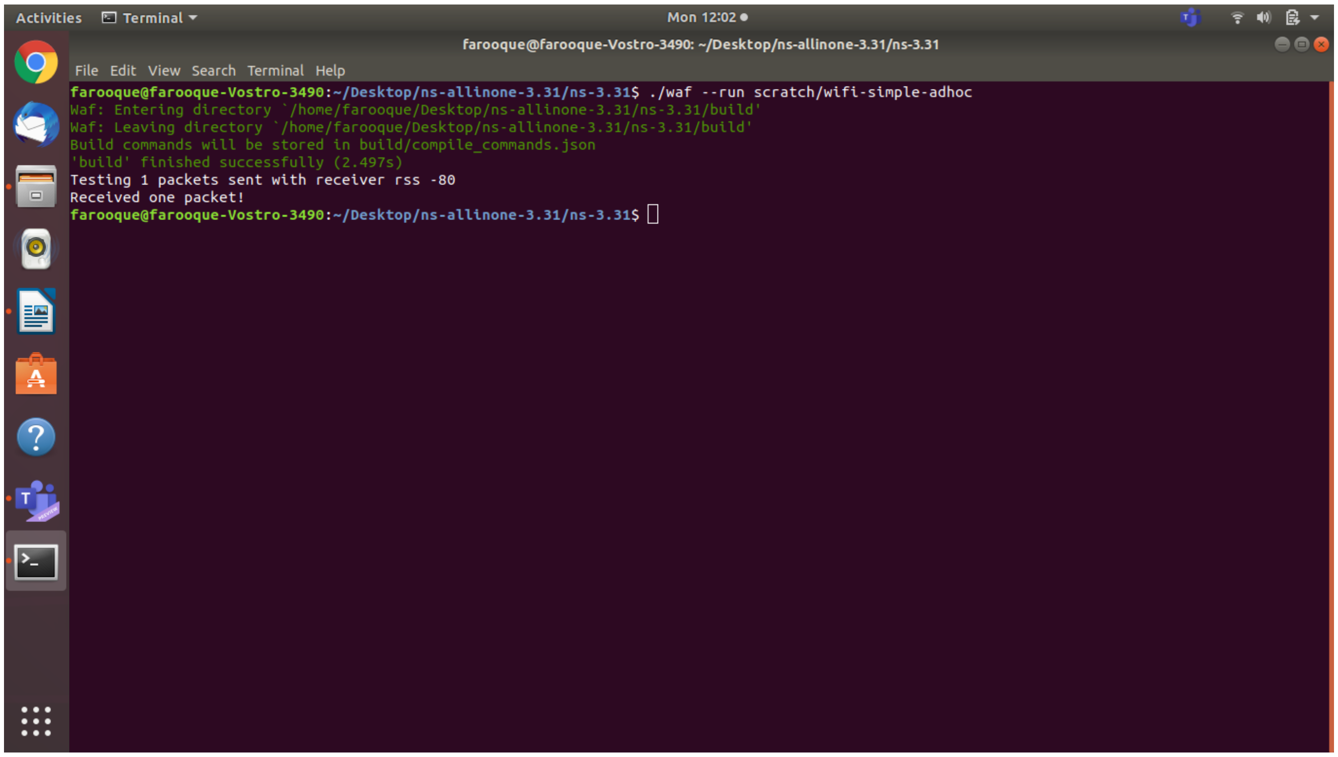
3.2 Coding using C++ Language

```
// This script configures two nodes on an 802.11b physical layer, with
 // 802.11b NICs in adhoc mode, and by default, sends one packet of 1000
 // (application) bytes to the other node.  The physical layer is configured
 // to receive at a fixed RSS (regardless of the distance and transmit
 // power); therefore, changing position of the nodes has no effect.
 //
 // There are a number of command-line options available to control
 // the default behavior.  The list of available command-line options
 // can be listed with the following command:
 // ./waf --run "wifi-simple-adhoc --help"
 //
 // For instance, for this configuration, the physical layer will
 // stop successfully receiving packets when rss drops below -97 dBm.
 // To see this effect, try running:
 //
 // ./waf --run "wifi-simple-adhoc --rss=-97 --numPackets=20"
 // ./waf --run "wifi-simple-adhoc --rss=-98 --numPackets=20"
 // ./waf --run "wifi-simple-adhoc --rss=-99 --numPackets=20"
 //
 // Note that all ns-3 attributes (not just the ones exposed in the below
 // script) can be changed at command line; see the documentation.
 //
 // This script can also be helpful to put the Wifi layer into verbose
 // logging mode; this command will turn on all wifi logging:
 //
 // ./waf --run "wifi-simple-adhoc --verbose=1"
 //
 // When you are done, you will notice two pcap trace files in your directory.
 // If you have tcpdump installed, you can try this:
 //
 // tcpdump -r wifi-simple-adhoc-0-0.pcap -nn -tt
 //

 #include "ns3/command-line.h"
 #include "ns3/config.h"
 #include "ns3/double.h"
 #include "ns3/string.h"
 #include "ns3/log.h"
 #include "ns3/yans-wifi-helper.h"
 #include "ns3/mobility-helper.h"
 #include "ns3/ipv4-address-helper.h"
 #include "ns3/yans-wifi-channel.h"
 #include "ns3/mobility-model.h"
 #include "ns3/internet-stack-helper.h"

 using namespace ns3;

 NS_LOG_COMPONENT_DEFINE ("WifiSimpleAdhoc");

 void ReceivePacket (Ptr<Socket> socket)
 {
   while (socket->Recv ())
     {
       NS_LOG_UNCOND ("Received one packet!");
     }
 }

 static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
                         uint32_t pktCount, Time pktInterval )
 {
   if (pktCount > 0)
     {
       socket->Send (Create<Packet> (pktSize));
       Simulator::Schedule (pktInterval, &GenerateTraffic,
                            socket, pktSize,pktCount - 1, pktInterval);
     }
   else
     {
       socket->Close ();
     }
 }

 int main (int argc, char *argv[])
 {
   std::string phyMode ("DsssRate1Mbps");
   double rss = -80;  // -dBm
   uint32_t packetSize = 1000; // bytes
   uint32_t numPackets = 1;
   double interval = 1.0; // seconds
   bool verbose = false;

   CommandLine cmd (__FILE__);
   cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
   cmd.AddValue ("rss", "received signal strength", rss);
   cmd.AddValue ("packetSize", "size of application packet sent", packetSize);
   cmd.AddValue ("numPackets", "number of packets generated", numPackets);
   cmd.AddValue ("interval", "interval (seconds) between packets", interval);
   cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
   cmd.Parse (argc, argv);
   // Convert to time object
   Time interPacketInterval = Seconds (interval);

   // Fix non-unicast data rate to be the same as that of unicast
   Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
                       StringValue (phyMode));

   NodeContainer c;
   c.Create (2);
```

```
// The below set of helpers will help us to put together the wifi NICs we want
WifiHelper wifi;
if (verbose)
  {
    wifi.EnableLogComponents ();  // Turn on all Wifi logging
  }
wifi.SetStandard (WIFI_STANDARD_80211b);

YansWifiPhyHelper wifiPhy;
// This is one parameter that matters when using FixedRssLossModel
// set it to zero; otherwise, gain will be added
wifiPhy.Set ("RxGain", DoubleValue (0) );
// ns-3 supports RadioTap and Prism tracing extensions for 802.11b
wifiPhy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
// The below FixedRssLossModel will cause the rss to be fixed regardless
// of the distance between the two stations, and the transmit power
wifiChannel.AddPropagationLoss ("ns3::FixedRssLossModel","Rss",DoubleValue (rss));
wifiPhy.SetChannel (wifiChannel.Create ());

// Add a mac and disable rate control
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));
// Set it to adhoc mode
wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, c);

// Note that with FixedRssLossModel, the positions below are not
// used for received signal strength.
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (5.0, 0.0, 0.0));
mobility.SetPositionAllocator (positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (c);
```

```
  InternetStackHelper internet;
  internet.Install (c);

  Ipv4AddressHelper ipv4;
  NS_LOG_INFO ("Assign IP Addresses.");
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);

  TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
  Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);
  InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
  recvSink->Bind (local);
  recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

  Ptr<Socket> source = Socket::CreateSocket (c.Get (1), tid);
  InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"), 80);
  source->SetAllowBroadcast (true);
  source->Connect (remote);

  // Tracing
  wifiPhy.EnablePcap ("wifi-simple-adhoc", devices);

  // Output what we are doing
  NS_LOG_UNCOND ("Testing " << numPackets  << " packets sent with receiver rss " << rss
);

  Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
                                  Seconds (1.0), &GenerateTraffic,
                                  source, packetSize, numPackets, interPacketInterval);

  Simulator::Run ();
  Simulator::Destroy ();

  return 0;
 }
```

3.3 Expected Results


Activities
Terminal
Mon 12:02
farooque@farooque-Vostro-3490: ~/Desktop/ns-allinone-3.31/ns-3.31
File Edit View Search Terminal Help

```
farooque@farooque-Vostro-3490:~/Desktop/ns-allinone-3.31/ns-3.31$ ./waf --run scratch/wifi-simple-adhoc
Waf: Entering directory `/home/farooque/Desktop/ns-allinone-3.31/ns-3.31/build'
Waf: Leaving directory `/home/farooque/Desktop/ns-allinone-3.31/ns-3.31/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.497s)
Testing 1 packets sent with receiver rss -80
Received one packet!
farooque@farooque-Vostro-3490:~/Desktop/ns-allinone-3.31/ns-3.31$
```

Program 8

1 Problem Statement

Install wireshark, and analyze the packets using it on a selected interface. Apply filters and check the packets.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to

- Install the wireshark tool for packet analysis
- Apply the filters and anlysze the packet parameters.

3 Design of the Program

3.1 Theory

Wireshark

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.[4]

Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License.

Packet

A network packet is a formatted unit of data carried by a packet-switched network. Computer communications links that do not support packets, such as traditional point-to-point telecommunications links, simply transmit data as a bit stream. When data is formatted into packets, the bandwidth of the communication medium can be better shared among users than if the network were circuit switched.

A packet consists of two kinds of data: control information and user data (also known as payload). The control information provides data the network needs to deliver the user data, for example: source and destination network addresses, error detection codes, and sequencing information. Typically, control information is found in packet headers and trailers, with payload data in between.

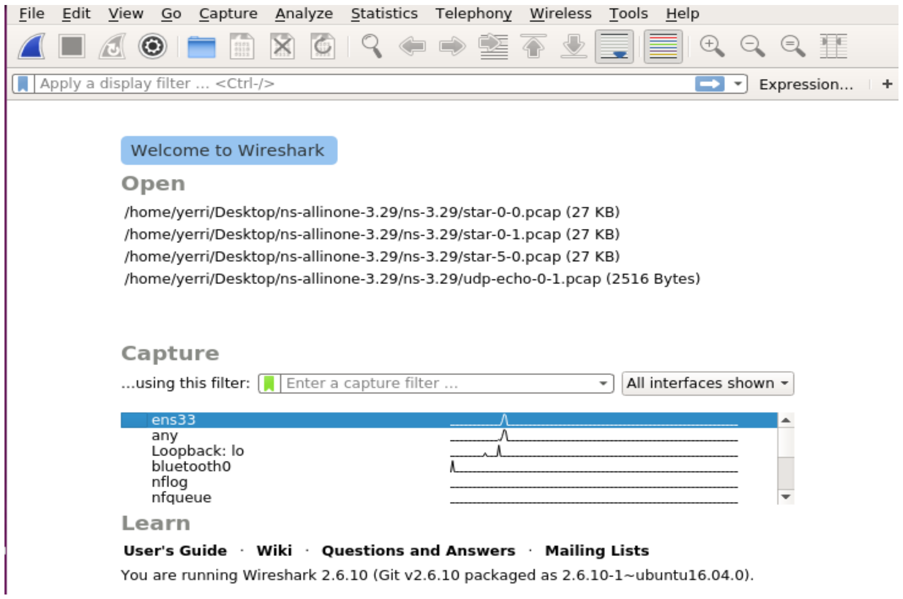
3.2 Intallation steps

Installation of Wireshark

```
$ sudo apt install wireshark
```

Opening wireshark

```
$sudo wireshark
```



Welcome to Wireshark

Open

/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/star-0-0.pcap (27 KB)
/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/star-0-1.pcap (27 KB)
/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/star-5-0.pcap (27 KB)
/home/yerri/Desktop/ns-allinone-3.29/ns-3.29/udp-echo-0-1.pcap (2516 Bytes)

Capture

...using this filter: Enter a capture filter ... All interfaces shown

ens33
any
Loopback: lo
bluetooth0
nflog
nfqueue

Learn

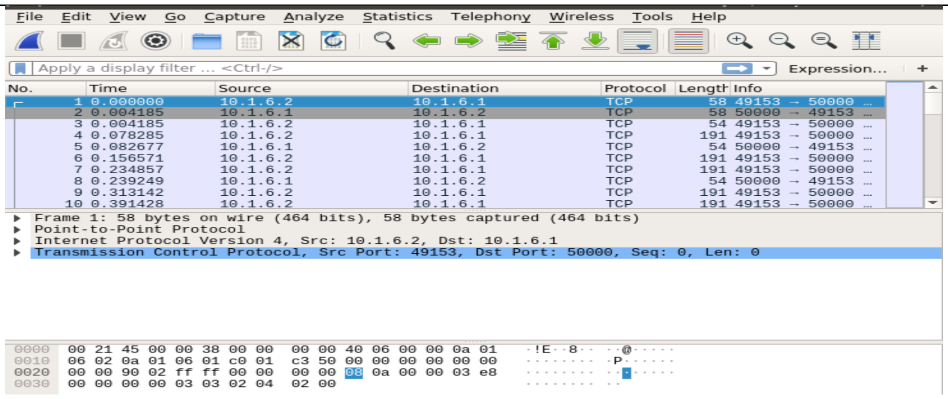
User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.6.10 (Git v2.6.10 packaged as 2.6.10-1~ubuntu16.04.0).

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

Filteration can be applied for a pcap (Packet capturing) file for our code output.
Also, live streaming can be analysed by applying suitable filter
In the output, filtering of UDP packets is recorded for that particular session
Enter the filter in the text area and click enter

3.3 Expected Results



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time       Source      Destination  Protocol Length Info
  1 0.000000   10.1.6.2    10.1.6.1     TCP        58 49153 → 50000 …
  2 0.004185   10.1.6.1    10.1.6.2     TCP        58 50000 → 49153 …
  3 0.004185   10.1.6.2    10.1.6.1     TCP        54 49153 → 50000 …
  4 0.078285   10.1.6.2    10.1.6.1     TCP       191 49153 → 50000 …
  5 0.082677   10.1.6.1    10.1.6.2     TCP        54 50000 → 49153 …
  6 0.156571   10.1.6.2    10.1.6.1     TCP       191 49153 → 50000 …
  7 0.234857   10.1.6.2    10.1.6.1     TCP       191 49153 → 50000 …
  8 0.239249   10.1.6.1    10.1.6.2     TCP        54 50000 → 49153 …
  9 0.313142   10.1.6.2    10.1.6.1     TCP       191 49153 → 50000 …
 10 0.391428   10.1.6.2    10.1.6.1     TCP       191 49153 → 50000 …

▸ Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
▸ Point-to-Point Protocol
▸ Internet Protocol Version 4, Src: 10.1.6.2, Dst: 10.1.6.1
▸ Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0

0000  00 21 45 00 00 38 00 00  00 00 40 06 00 00 0a 01   ·!E··8·· ··@·····
0010  06 02 0a 01 06 01 c0 01  c3 50 00 00 00 00 00 00   ········ ·P······
0020  00 00 90 02 ff ff 00 00  00 00 08 0a 00 00 03 e8   ········ ········
0030  00 00 00 00 03 03 02 04  02 00                     ········ ··
```

Program 9

1 Problem Statement

Install packet tracer, and consider a topology and configure VLAN

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to

- Install the Packet tracer
- Create and Configure a simple Virtual Local Area Network(VLAN)

3 Design of the Program

3.1

Theory

Packet Tracer

Packet Tracer is a cross-platform visual simulation tool designed by Cisco Systems that allows users to create network topologies and imitate modern computer networks. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface. Packet Tracer makes use of a drag and drop user interface, allowing users to add and remove simulated network devices as they see fit. The software is mainly focused towards Certified Cisco Network Associate Academy students as an educational tool for helping them learn fundamental CCNA concepts

VLAN

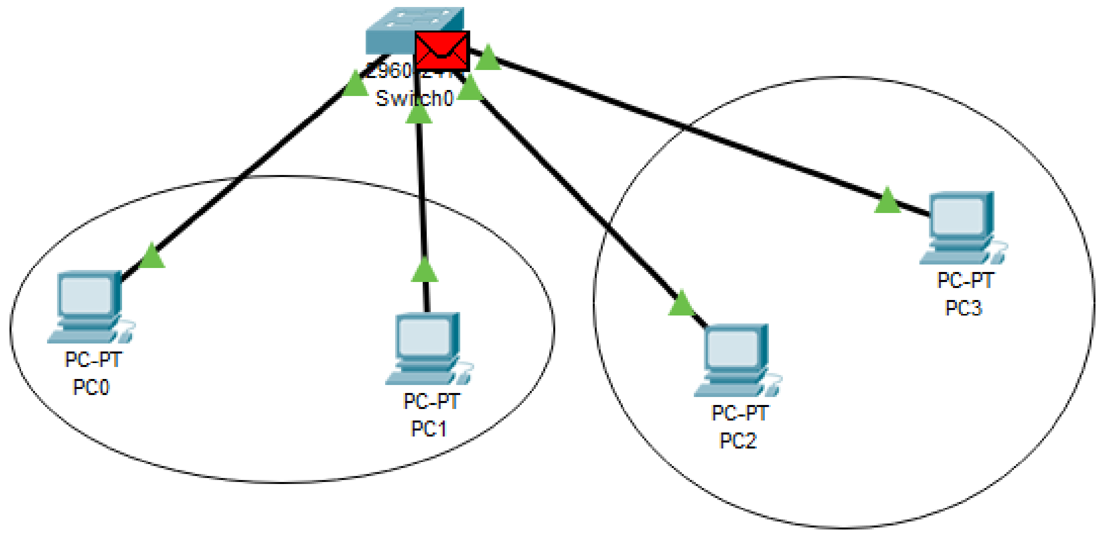
A Virtual LAN (**VLAN)** is simply a logical LAN, just as its name suggests. VLANs have similar characteristics with those of physical LANs, only that with VLANs, you can logically group hosts even if they are physically located on separate LAN segments.

We treat each VLAN as a separate subnet or broadcast domain. For this reason, to move packets from one VLAN to another, we have to use a router or a layer 3 switch.

VLANs are configured on switches by placing some interfaces into one broadcast domain and some interfaces into another.

3.2 Coding using C++ Language

1. Go to this web link https://www.netacad.com/courses/packet-tracer and register to download packet tracer and install
2. Steps to create VLAN
   In Cisco Packet Tracer, **create the network topology**with two VLAN viz. VLAN 10 and VLAN 20 as shown below:



Steps

Open and choose switch and four computers
Choose the link and join with fast Ethernet
Select now pc1 by double click->desktop->IP address
192.168.10.2 Keeping subnet mask as 192.168.10.1
Select now pc2 by double click->desktop->IP address
192.168.10.3 Keeping subnet mask as 192.168.10.1
Select now pc3 by double click->desktop->IP address
192.168.20.2 Keeping subnet mask as 192.168.20.1
Select now pc4 and double click->desktop->IP address
192.168.20.3 Keeping subnet mask as 192.168.20.1
After this we need to configure the switch
For this double click on switch
Choose the tab CLI
Type the following command to configure and name of VLAN

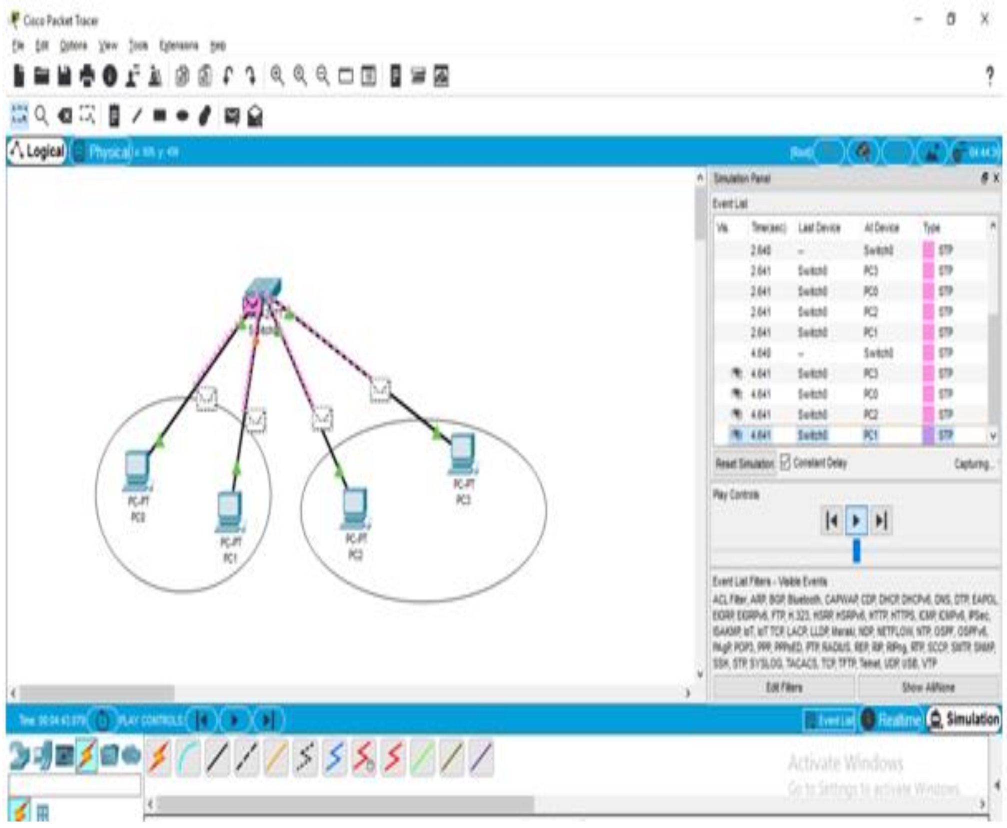
```
$en
$config
$vlan 10
$name Farooque
$exit
$en
$config
$vlan 20
$name Yerri
$exit
```

After this you can type show vlan
Once done, now your task is to map this to the PC from VLAN 10 and VLAN 20

```
$en
```

$interface f0/1 (check it while building the network)
$switchport access vlan 10
$exit
$interface f0/2 (check it while building the network)
$switchport access vlan 10
$exit
$interface f0/3 (check it while building the network)
$switchport access vlan 20
$exit
$interface f0/4 (check it while building the network)
$switchport access vlan 20
$exit
$exit
Then finally simulate topology

3.3 Expected Results



Vis.	Time(sec)	Last Device	At Device	Type
	2.640	--	Switch0	STP
	2.641	Switch0	PC3	STP
	2.641	Switch0	PC0	STP
	2.641	Switch0	PC2	STP
	2.641	Switch0	PC1	STP
	4.640	--	Switch0	STP
	4.641	Switch0	PC3	STP
	4.641	Switch0	PC0	STP
	4.641	Switch0	PC2	STP
	4.641	Switch0	PC1	STP

Program 10

1 Problem Statement

Install NMAP, and execute atleast 5 commands to demonstrate the scanning of networks hosts and ports.

2 Student Learning Outcomes

After successful completion of this lab, the student shall be able to

- Install Nmap in both Windows and Ubantu Platform.
- Demonstrate the scanning of networks hosts and ports using suitable commends.

3 Design of the Program

3.1 Theory

Nmap is a network mapper that has emerged as one of the most popular, free network discovery tools on the market. Nmap is now one of the core tools used by network administrators to map their networks. The program can be used to find live hosts on a network, perform port scanning, ping sweeps, OS detection, and version detection.

3.2 Installation steps and Commands

```
$sudo apt install nmap
```

Commands:

1. Basic Nmap Scan against IP or host

```
nmap www.reva.edu.in
```

Now, if you want to scan a hostname, simply replace the IP for the host, as you see below:

```
nmap www.reva.edu.in
```

2. Scan specific ports or scan entire port ranges on a local or remote server

```
nmap -p 1-50www.google.com
```

In this example, we scanned all 50 ports for www.google.com

3. Scan multiple IP addresses

Let’s try to scan multiple IP addresses. For this you need to use this syntax:

```
nmap 1.1.1.1 8.8.8.8
```

You can also scan consecutive IP addresses:

nmap -p 1.1.1.1,2,3,4
This will scan 1.1.1.1, 1.1.1.2, 1.1.1.3 and 1.1.1.4.

4. Scan the most popular ports

Using "–top-ports" parameter along with a specific number lets you scan the top X most common ports for that host, as we can see:
nmap --top-ports 5www.google.com
Replace "5" with the desired number. Output example:

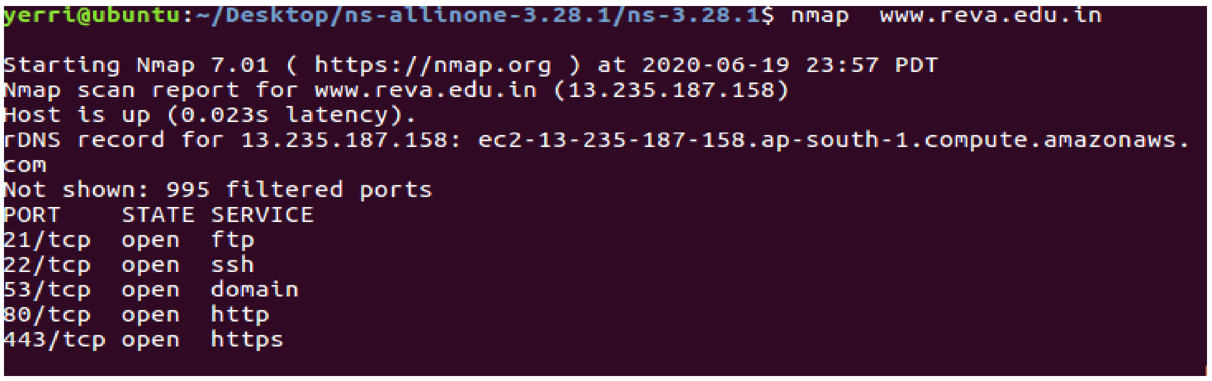
5. Scan using TCP or UDP protocols

One of the things we love most about Nmap is the fact that it works for both TCP and UDP protocols. And while most services run on TCP, you can also get a great advantage by scanning UDP-based services. Let's see some examples.
Standard TCP scanning output:

nmap -sT www.reva.edu.in
nmap -sU 192.168.1.1

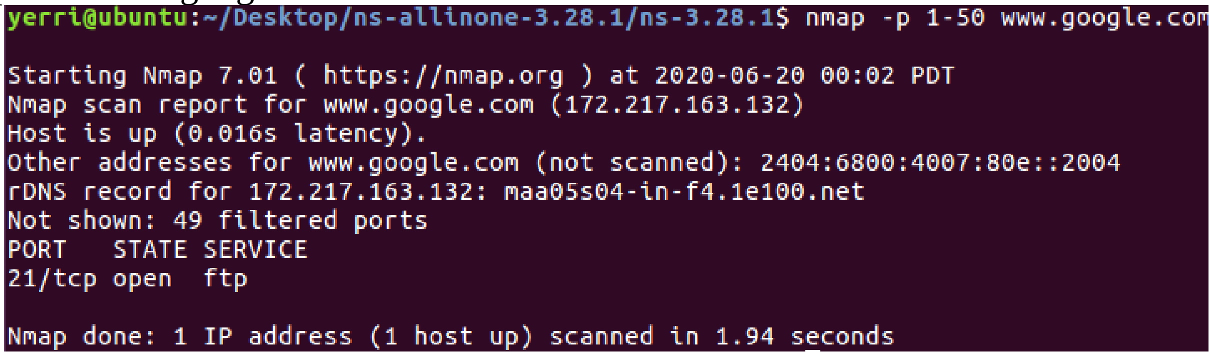
3.3	**Expected Results**

nmap www.reva.edu.in



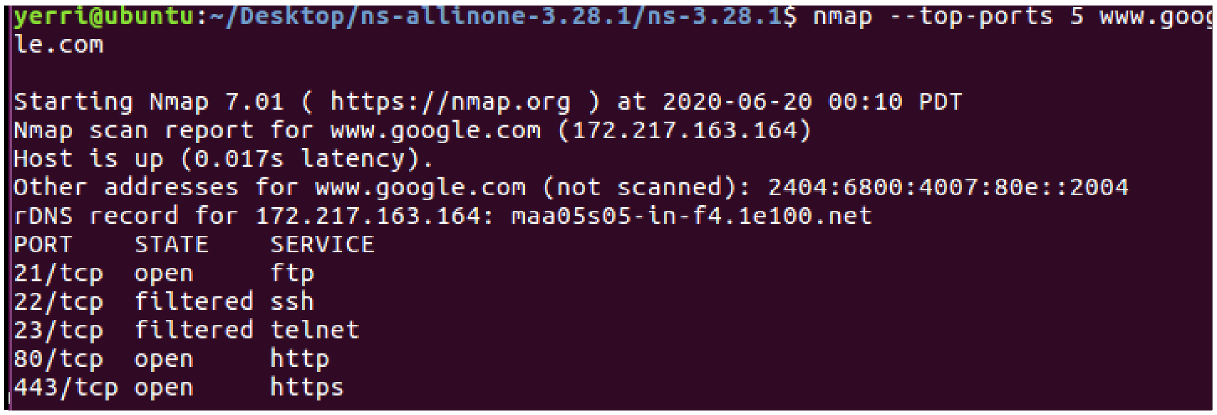
```
yerri@ubuntu:~/Desktop/ns-allinone-3.28.1/ns-3.28.1$ nmap  www.reva.edu.in

Starting Nmap 7.01 ( https://nmap.org ) at 2020-06-19 23:57 PDT
Nmap scan report for www.reva.edu.in (13.235.187.158)
Host is up (0.023s latency).
rDNS record for 13.235.187.158: ec2-13-235-187-158.ap-south-1.compute.amazonaws.
com
Not shown: 995 filtered ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
```

nmap -p 1-50www.google.com



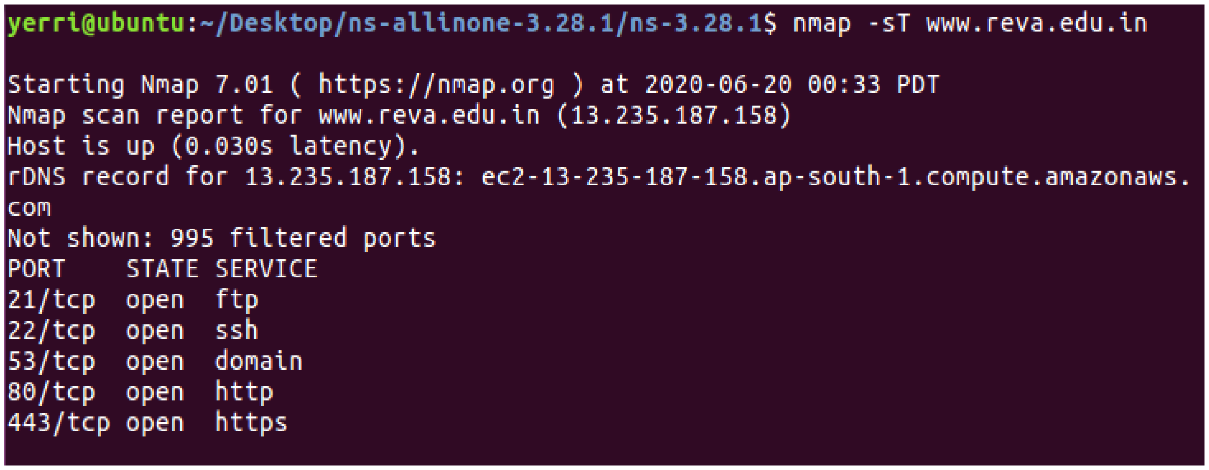
```
yerri@ubuntu:~/Desktop/ns-allinone-3.28.1/ns-3.28.1$ nmap -p 1-50 www.google.com

Starting Nmap 7.01 ( https://nmap.org ) at 2020-06-20 00:02 PDT
Nmap scan report for www.google.com (172.217.163.132)
Host is up (0.016s latency).
Other addresses for www.google.com (not scanned): 2404:6800:4007:80e::2004
rDNS record for 172.217.163.132: maa05s04-in-f4.1e100.net
Not shown: 49 filtered ports
PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 1.94 seconds
```

nmap --top-ports 5www.google.com



```
yerri@ubuntu:~/Desktop/ns-allinone-3.28.1/ns-3.28.1$ nmap --top-ports 5 www.goog
le.com

Starting Nmap 7.01 ( https://nmap.org ) at 2020-06-20 00:10 PDT
Nmap scan report for www.google.com (172.217.163.164)
Host is up (0.017s latency).
Other addresses for www.google.com (not scanned): 2404:6800:4007:80e::2004
rDNS record for 172.217.163.164: maa05s05-in-f4.1e100.net
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   filtered ssh
23/tcp   filtered telnet
80/tcp   open     http
443/tcp  open     https
```

nmap -sT www.reva.edu.in



```
yerri@ubuntu:~/Desktop/ns-allinone-3.28.1/ns-3.28.1$ nmap -sT www.reva.edu.in

Starting Nmap 7.01 ( https://nmap.org ) at 2020-06-20 00:33 PDT
Nmap scan report for www.reva.edu.in (13.235.187.158)
Host is up (0.030s latency).
rDNS record for 13.235.187.158: ec2-13-235-187-158.ap-south-1.compute.amazonaws.
com
Not shown: 995 filtered ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
```

Viva Questions

- What is Simulator?
- What is tha difference between Simulator and Emulator?
- What is NetAnim?
- What is the ise of Waf Tool in NS3?
- What is Packet?
- What is a Link?
- What is a Gateway or Router?
- What is Point-Point Link?
- What is topology?
- List the types of Topologies?
- Name the Factors that affect the performance of Network?
- What is ptotocol?
- Name protocols in Application Layer.
- What is TCP congestion control?
- What is Descrete event simulator?
- What is VLAN?
- What is the use of NMAP?
- What is the protocol number for TCP?
- Which protocol gets a hardware address from a known IP address?
- What Is Wireshark?
- How Would You Setup Wireshark To Monitor Packets Passing Through An Internet Router?
- Can Wireshark Be Setup On A Cisco Router?
- Which Wireshark Filter Can Be Used To Check All Incoming Requests To A Http Web Server?
- What Kind Of Shark Is Wireshark?
- What Do You Think Of Wireshark?
- How To Remove Wireshark Antivirus From My Computer?
- How Do I Use Wireshark To Find A Password In My Network?

- What Exactly Does Wireshark Do?
- Name a popular tool used for network analysis in multiprotocol diverse network?
- List port numbers for wired and wireless protocols?

****