

**Computer Networks Lab (B18CS5080)**

**SCHOOL OF COMPUTING**

**AND**

**INFORMATION TECHNOLOGY**

**Submitted to: - prof. Farooque Azam sir and prof. Nikhil Tengli Sir**

**NAME: Manasa. M. A**

**SRN: R18CS212**

**SEM: 5th**

**SECTION: ‘D’**

**Program 1**

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

1. **Introduction to Discrete 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

## 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.

1. **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.**

1. **NS 3 Installation**

**Download the ns3 package (ns-allinone-3.30) from**

**https://www.nsnam.org and follow following steps:  
1. $cd Desktop  
2. $cd ns-allinone-3.28  
3. $./build.py --enable-examples --enable-test  
4. $cd ns-3.28  
5. $./waf --build-profile=debug --enable-examples --enable-test configure  
  
once all the configuration is done NS-3 inastallation is completed.**

**To check whether NS-3 is successfully installed verify with this:  
$./waf --run hello-simulator  
  
The output will be shown as hello-simulator**

1. **Introduction to NetAnim:**

**$ Sudo apt install qt4-default qt4-make**

**$ cd NetAnim**

**$ qmake NetAnim.pro**

**$ make**

**Execution of NetAnim**

**$ Cd to NetAnim from ns3.28...**

**$ ./NetAnim**

**Program 2**

**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.**

**CODE:**

**#include "ns3/core-module.h"**

**#include "ns3/network-module.h"**

**#include "ns3/internet-module.h"**

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

**#include "ns3/applications-module.h"**

**#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));**

**pointToPoint.EnablePcapAll ("first");**

**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);**

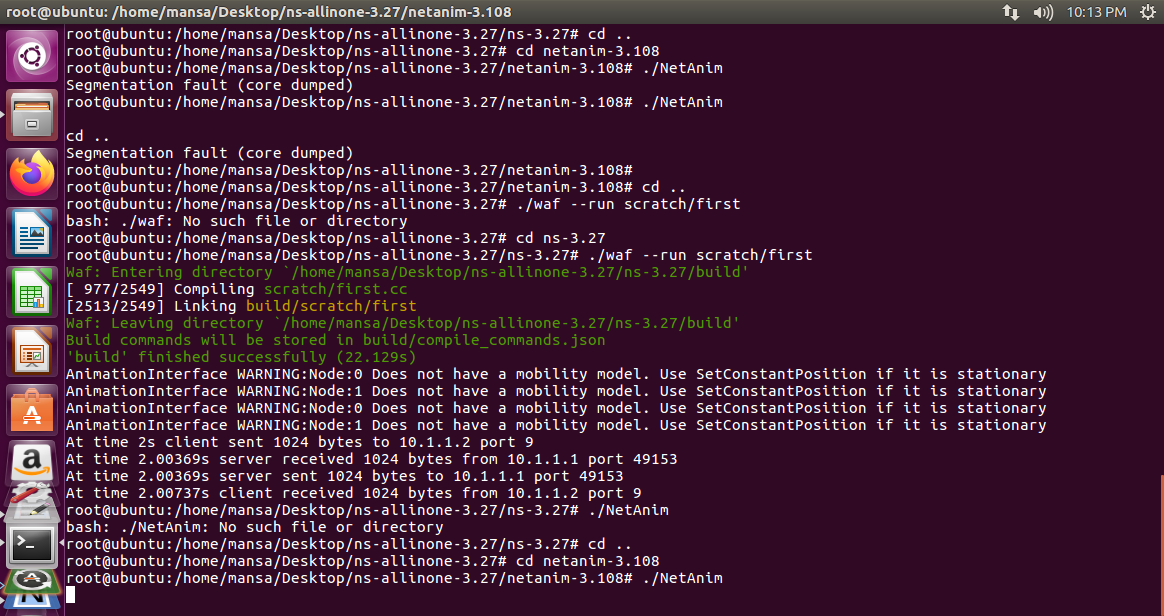
**Simulator::Run ();**

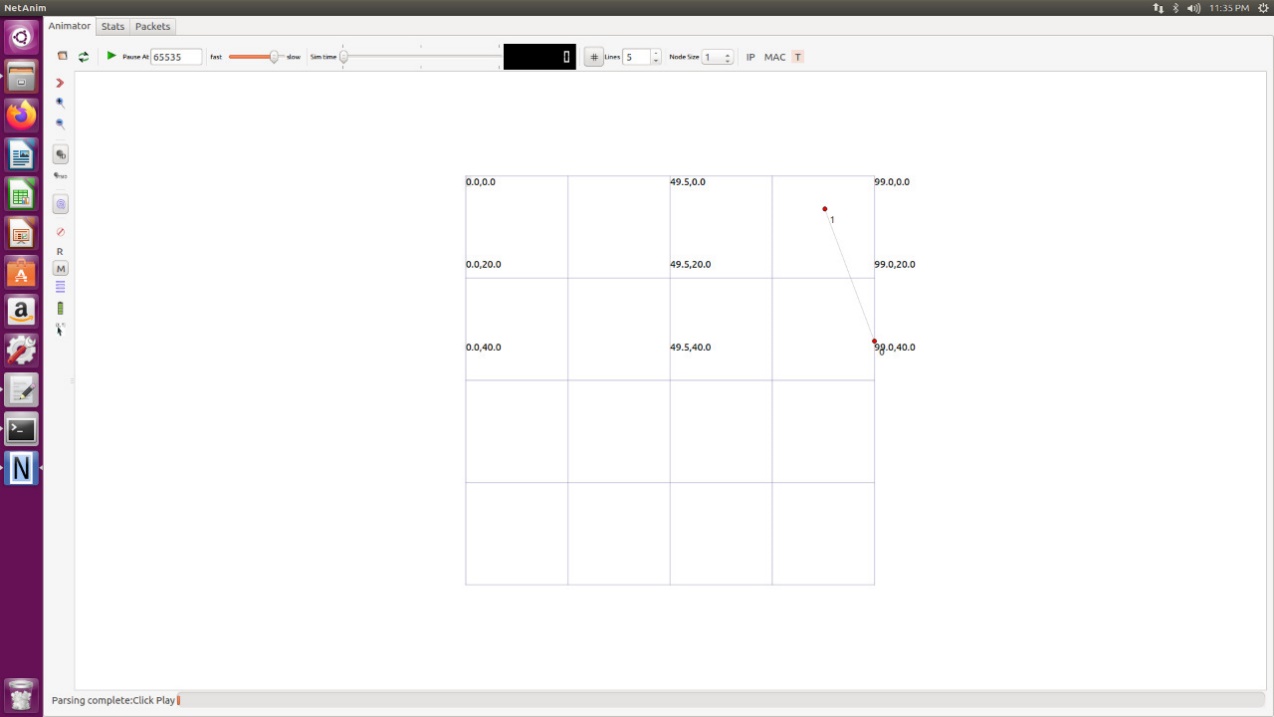
**Simulator::Destroy ();**

**return 0;**

**}**

**OUTPUT:**





**Program 3**

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

**CODE:**

**#include "ns3/core-module.h"**

**#include "ns3/network-module.h"**

**#include "ns3/csma-module.h"**

**#include "ns3/internet-module.h"**

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

**#include "ns3/applications-module.h"**

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

**#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**

**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);**

**AnimationInterface anim ("second.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);**

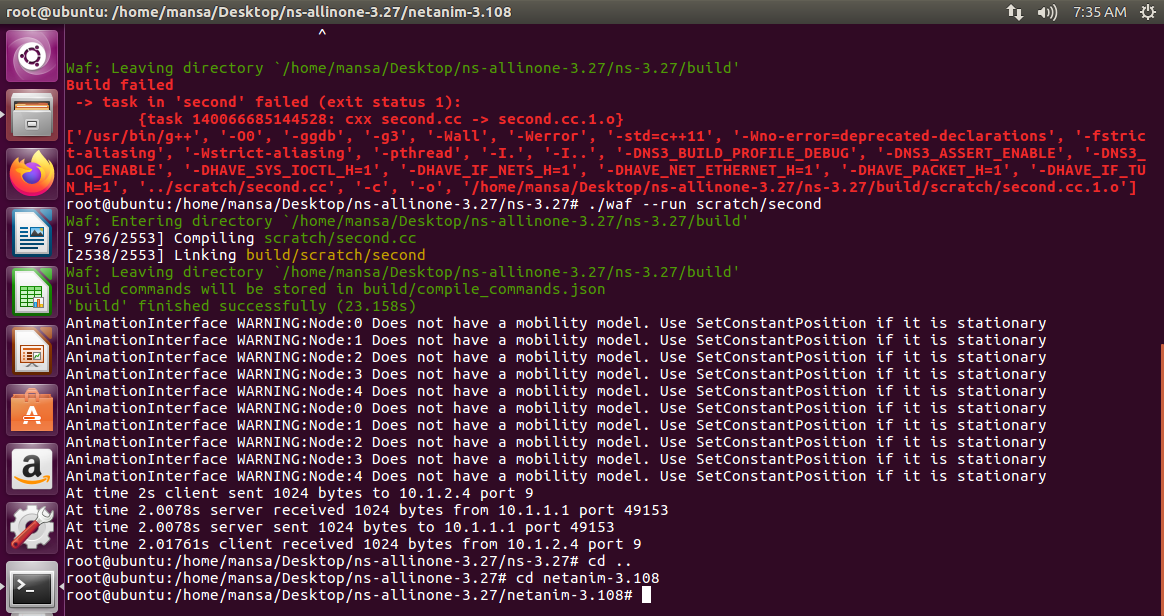
**Simulator::Run ();**

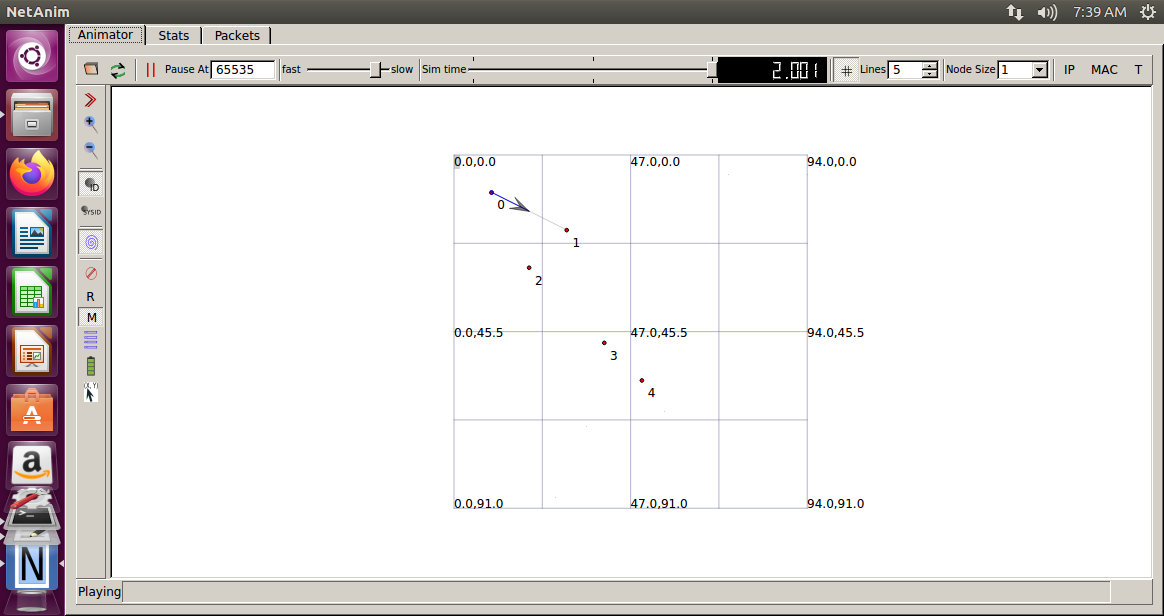
**Simulator::Destroy ();**

**return 0;**

**}**

**Output:**

****

****

**Program 4**

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

**CODE:**

**#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 ("StarAnimation");**

**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 animFile = "star-animation.xml";**

**uint8\_t useIpv6 = 0;**

**Ipv6Address ipv6AddressBase = Ipv6Address("2001::");**

**Ipv6Prefix ipv6AddressPrefix = Ipv6Prefix(64);**

**CommandLine cmd (\_\_FILE\_\_);**

**cmd.AddValue ("nSpokes", "Number of spoke nodes to place in the star", nSpokes);**

**cmd.AddValue ("animFile", "File Name for Animation Output", animFile);**

**cmd.AddValue ("useIpv6", "use Ipv6", useIpv6);**

**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.");**

**if (useIpv6 == 0)**

**{**

**star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));**

**}**

**else**

**{**

**star.AssignIpv6Addresses (ipv6AddressBase, ipv6AddressPrefix);**

**}**

**NS\_LOG\_INFO ("Create applications.");**

**//**

**// Create a packet sink on the star "hub" to receive packets.**

**//**

**uint16\_t port = 50000;**

**Address hubLocalAddress;**

**if (useIpv6 == 0)**

**{**

**hubLocalAddress = InetSocketAddress (Ipv4Address::GetAny (), port);**

**}**

**else**

**{**

**hubLocalAddress = Inet6SocketAddress (Ipv6Address::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;**

**if (useIpv6 == 0)**

**{**

**remoteAddress = AddressValue(InetSocketAddress (star.GetHubIpv4Address (i), port));**

**}**

**else**

**{**

**remoteAddress = AddressValue(Inet6SocketAddress (star.GetHubIpv6Address (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.**

**//**

**if (useIpv6 == 0)**

**{**

**Ipv4GlobalRoutingHelper::PopulateRoutingTables ();**

**}**

**// Set the bounding box for animation**

**star.BoundingBox (1, 1, 100, 100);**

**// Create the animation object and configure for specified output**

**AnimationInterface anim (animFile);**

**pointToPoint.EnablePcapAll ("star-animation");**

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

**Simulator::Run ();**

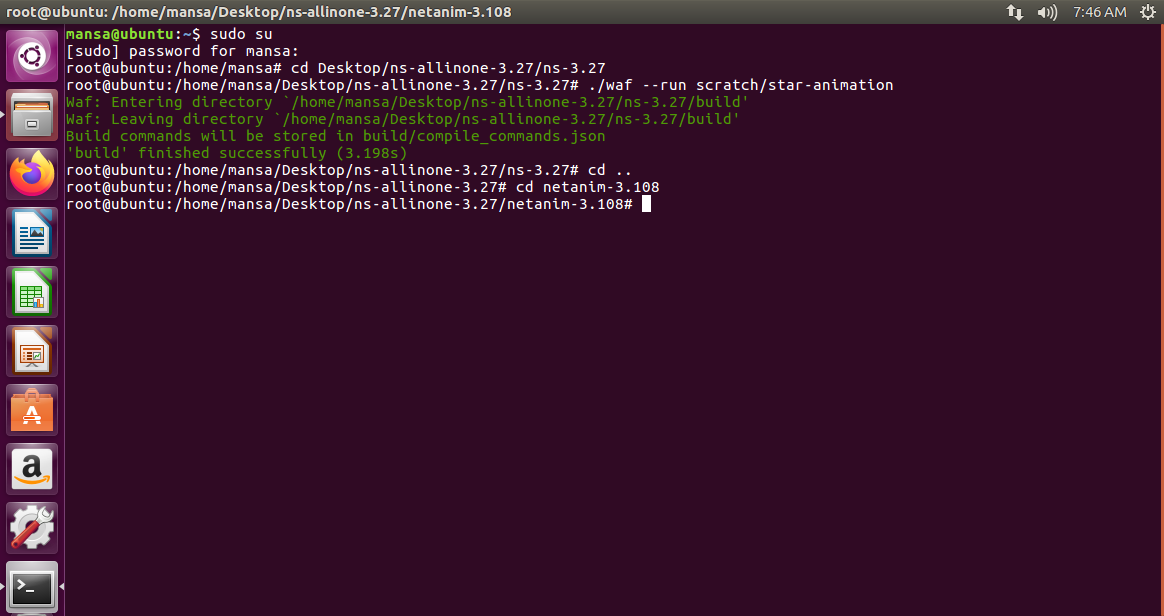
**Simulator::Destroy ();**

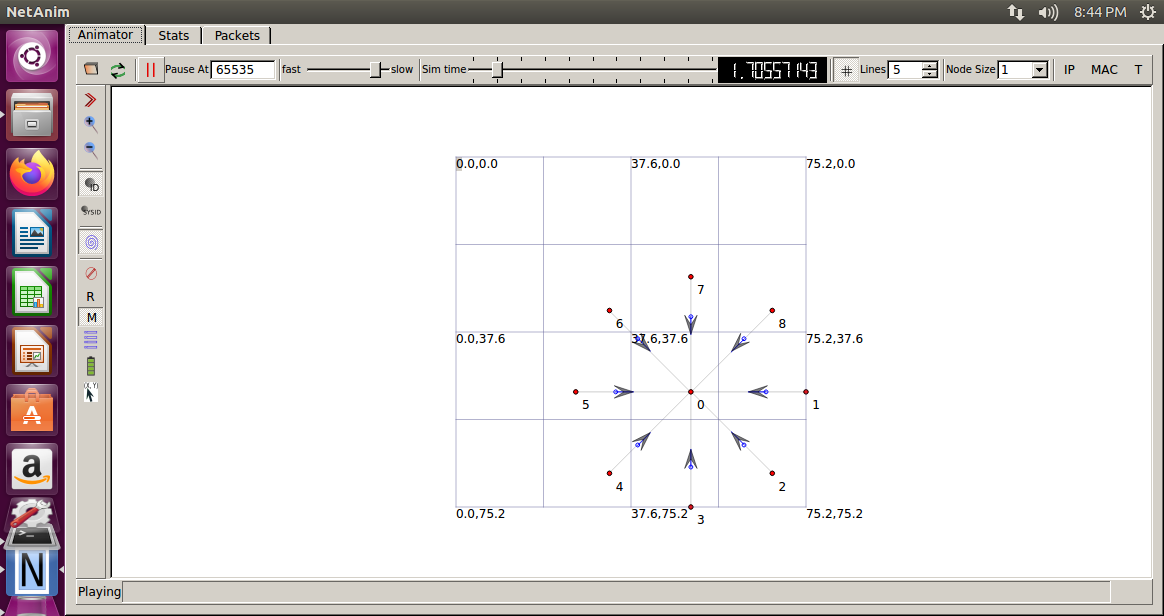
**NS\_LOG\_INFO ("Done.");**

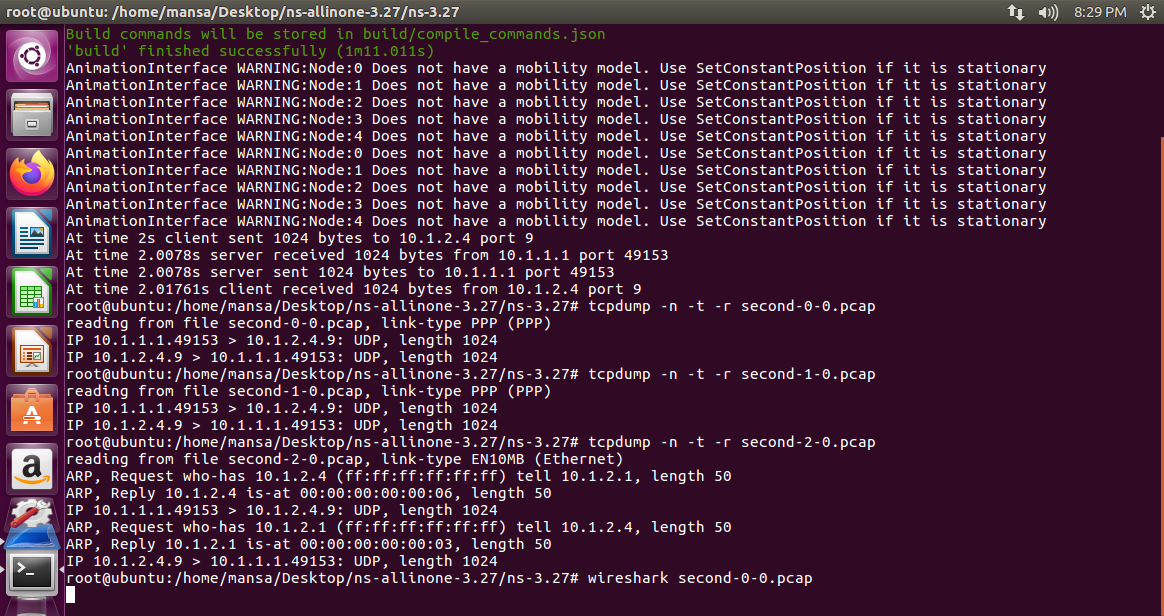
**return 0;**

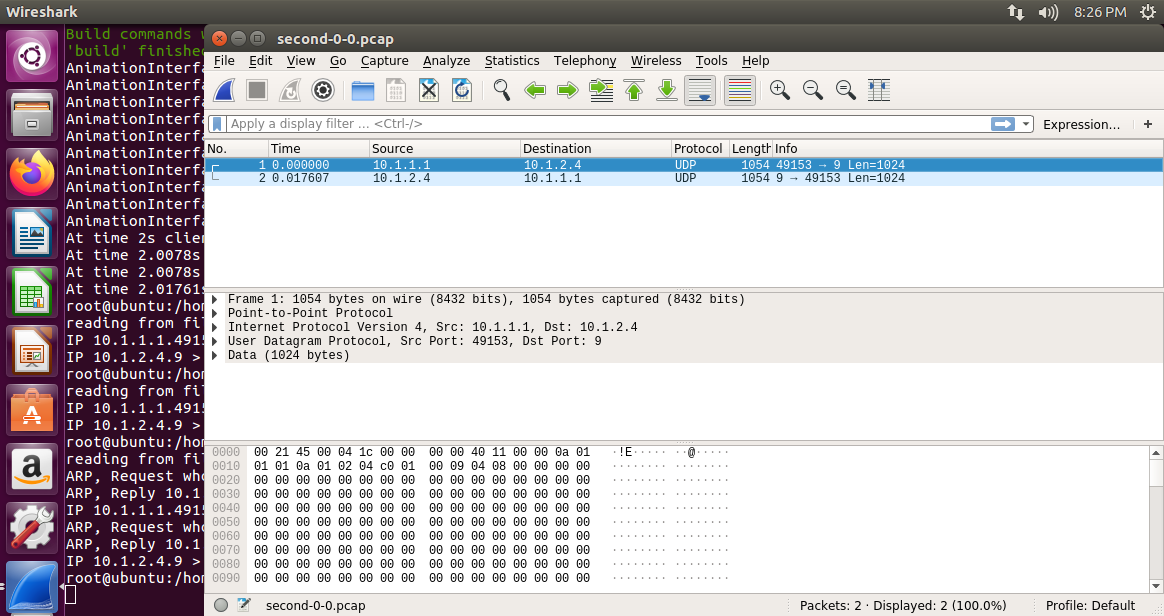
**}**

**OUTPUT:**

****

****

****



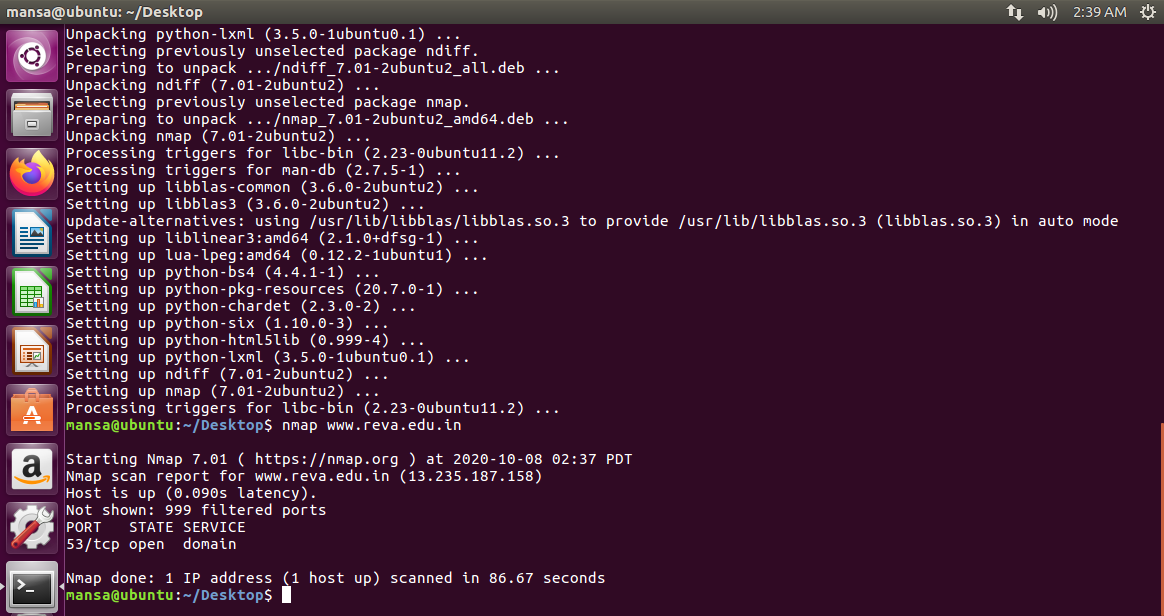
**Program 5**

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

**1. Basic Nmap Scan against IP or host**

**nmap www.reva.edu.in**

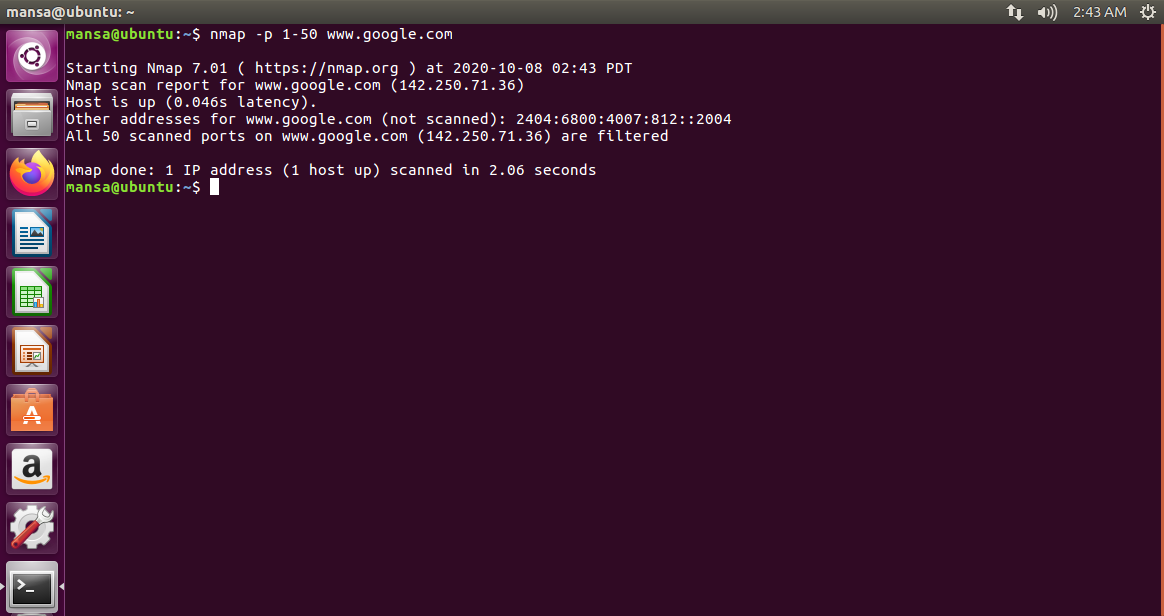
Now, if you want to scan a hostname, simply replace the IP for the host, as you see below: nmap [www.reva.edu.in](http://www.reva.edu.in)



**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](http://www.google.com)



**3. Scan multiple IP addresses**

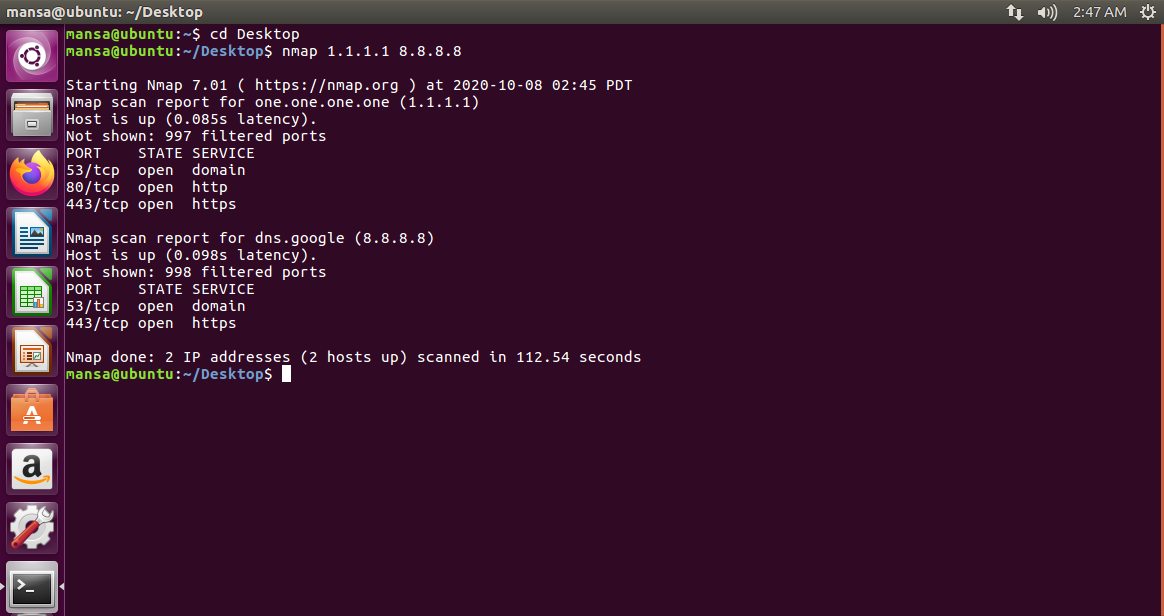
Scan multiple IP addresses. For this you need to use this syntax:

**nmap 1.1.1.1 8.8.8.8**

Scan consecutive IP addresses:

**nmap -p 1.1.1.1,2,3,4**

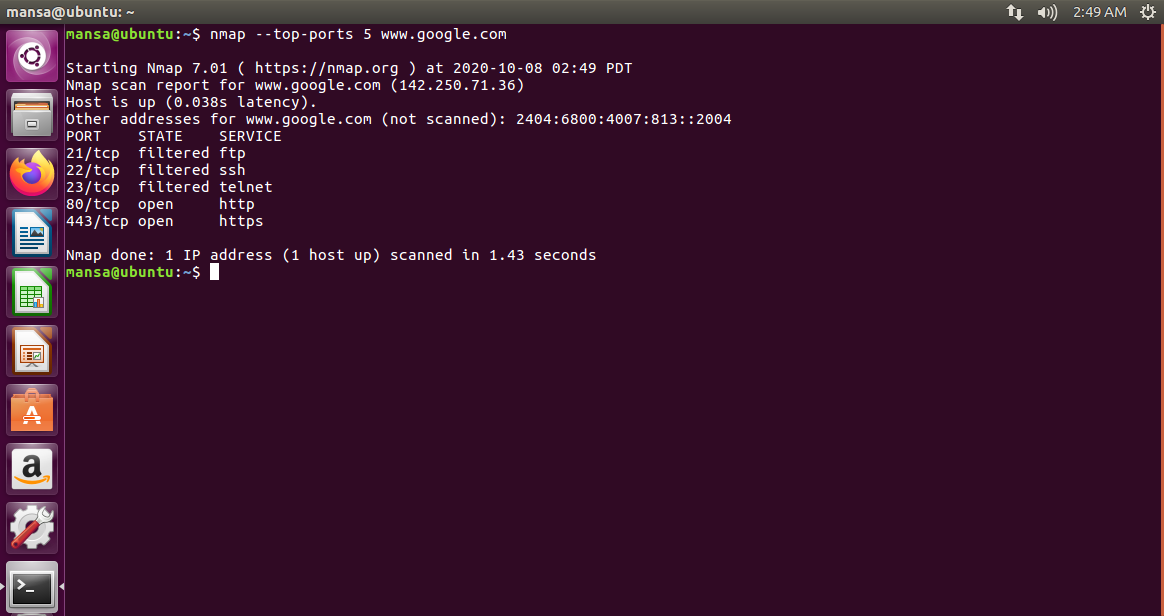
This will scan 1.1.1.1, 1.1.1.2, 1.1.1.3 and 1.1.1.4.



**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 5**[**www.google.com**](http://www.google.com)



**5. Scan using TCP or UDP protocols**

Nmap works for both TCP and UDP protocols. And while most services run on TCP, we 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**

