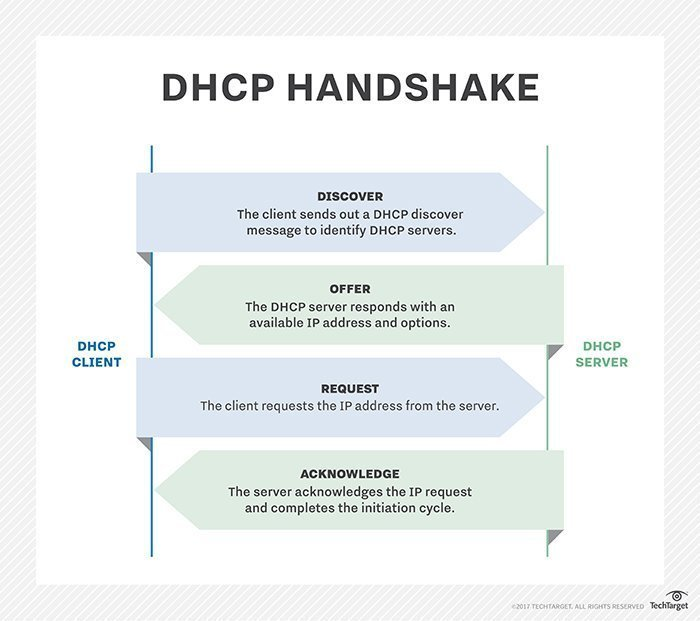
**Practical No 10**

**Aim: -** WAP to simulate DHCP with n clients.

**Objective: -**

* Understanding of DHCP Protocol
* Implementation of DHCP server & n clients network using NS3.
* Visualization of above network using Netanim.

**Theory: -**

DHCP (Dynamic Host Configuration Protocol) is a network management protocol used to dynamically assign an IP address to any device, or node, on a network so it can communicate using IP. DHCP automates and centrally manages these configurations rather than requiring network administrators to manually assign IP addresses to all network devices. DHCP can be implemented on small local networks, as well as large enterprise networks.

**Advantages of DHCP Protocol:**

* Centralized management of IP addresses.
* Centralized and automated TCP/IP configuration.
* Ease of adding new clients to a network.
* Reuse of IP addresses reduces the total number of IP addresses that are required.
* The efficient handling of IP address changes for clients that must be updated frequently, such as those for portable devices that move to different locations on a wireless network.

**Disadvantages of DHCP Protocol:**

* IP conflict can occur.
* The client is not able to access the network in absence of a DHCP Server.
* The name of the machine will not be changed in a case when a new IP Address is assigned.
* Installation is a difficult process.

**Program: -**

#include "ns3/core-module.h"

#include "ns3/internet-apps-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/netanim-module.h"

using namespace ns3;

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

int

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

{

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

bool verbose = false;

bool tracing = false;

std::string animFile = "dhcp-server-client-animation.xml";

cmd.AddValue ("verbose", "turn on the logs", verbose);

cmd.AddValue ("tracing", "turn on the tracing", tracing);

cmd.Parse (argc, argv);

// GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

if (verbose)

{

LogComponentEnable ("DhcpServer", LOG\_LEVEL\_ALL);

LogComponentEnable ("DhcpClient", LOG\_LEVEL\_ALL);

LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_INFO);

LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_INFO);

}

Time stopTime = Seconds (20);

NS\_LOG\_INFO ("Create nodes.");

NodeContainer nodes;

NodeContainer router;

nodes.Create (3);

router.Create (2);

NodeContainer net (nodes, router);

NS\_LOG\_INFO ("Create channels.");

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", StringValue ("5Mbps"));

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

csma.SetDeviceAttribute ("Mtu", UintegerValue (1500));

NetDeviceContainer devNet = csma.Install (net);

NodeContainer p2pNodes;

p2pNodes.Add (net.Get (4));

p2pNodes.Create (1);

PointToPointHelper pointToPoint;

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

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

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

InternetStackHelper tcpip;

tcpip.Install (nodes);

tcpip.Install (router);

tcpip.Install (p2pNodes.Get (1));

Ipv4AddressHelper address;

address.SetBase ("172.30.1.0", "255.255.255.0");

Ipv4InterfaceContainer p2pInterfaces;

p2pInterfaces = address.Assign (p2pDevices);

// manually add a routing entry because we don't want to add a dynamic routing

Ipv4StaticRoutingHelper ipv4RoutingHelper;

Ptr<Ipv4> ipv4Ptr = p2pNodes.Get (1)->GetObject<Ipv4> ();

Ptr<Ipv4StaticRouting> staticRoutingA = ipv4RoutingHelper.GetStaticRouting

(ipv4Ptr);

staticRoutingA->AddNetworkRouteTo (Ipv4Address ("172.30.0.0"), Ipv4Mask

("/24"),

Ipv4Address ("172.30.1.1"), 1);

NS\_LOG\_INFO ("Setup the IP addresses and create DHCP applications.");

DhcpHelper dhcpHelper;

// The router must have a fixed IP.

Ipv4InterfaceContainer fixedNodes = dhcpHelper.InstallFixedAddress

(devNet.Get (4), Ipv4Address ("172.30.0.17"), Ipv4Mask ("/24"));

// Not really necessary, IP forwarding is enabled by default in IPv4.

fixedNodes.Get (0).first->SetAttribute ("IpForward", BooleanValue (true));

// DHCP server

ApplicationContainer dhcpServerApp = dhcpHelper.InstallDhcpServer

(devNet.Get (3), Ipv4Address ("172.30.0.12"),

Ipv4Address ("172.30.0.0"), Ipv4Mask ("/24"),

Ipv4Address ("172.30.0.10"), Ipv4Address ("172.30.0.15"),

Ipv4Address ("172.30.0.17"));

// This is just to show how it can be done.

DynamicCast<DhcpServer> (dhcpServerApp.Get (0))->AddStaticDhcpEntry

(devNet.Get (2)->GetAddress (), Ipv4Address ("172.30.0.14"));

dhcpServerApp.Start (Seconds (0.0));

dhcpServerApp.Stop (stopTime);

// DHCP clients

NetDeviceContainer dhcpClientNetDevs;

dhcpClientNetDevs.Add (devNet.Get (0));

dhcpClientNetDevs.Add (devNet.Get (1));

dhcpClientNetDevs.Add (devNet.Get (2));

ApplicationContainer dhcpClients = dhcpHelper.InstallDhcpClient

(dhcpClientNetDevs);

dhcpClients.Start (Seconds (1.0));

dhcpClients.Stop (stopTime);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (p2pNodes.Get (1));

serverApps.Start (Seconds (0.0));

serverApps.Stop (stopTime);

UdpEchoClientHelper echoClient (p2pInterfaces.GetAddress (1), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (100));

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

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

ApplicationContainer clientApps = echoClient.Install (nodes.Get (1));

clientApps.Start (Seconds (10.0));

clientApps.Stop (stopTime);

Simulator::Stop (stopTime + Seconds (10.0));

// Create the animation object and configure for specified output

AnimationInterface anim ("dhcp.xml");

if (tracing)

{

csma.EnablePcapAll ("dhcp-csma");

pointToPoint.EnablePcapAll ("dhcp-p2p");

}

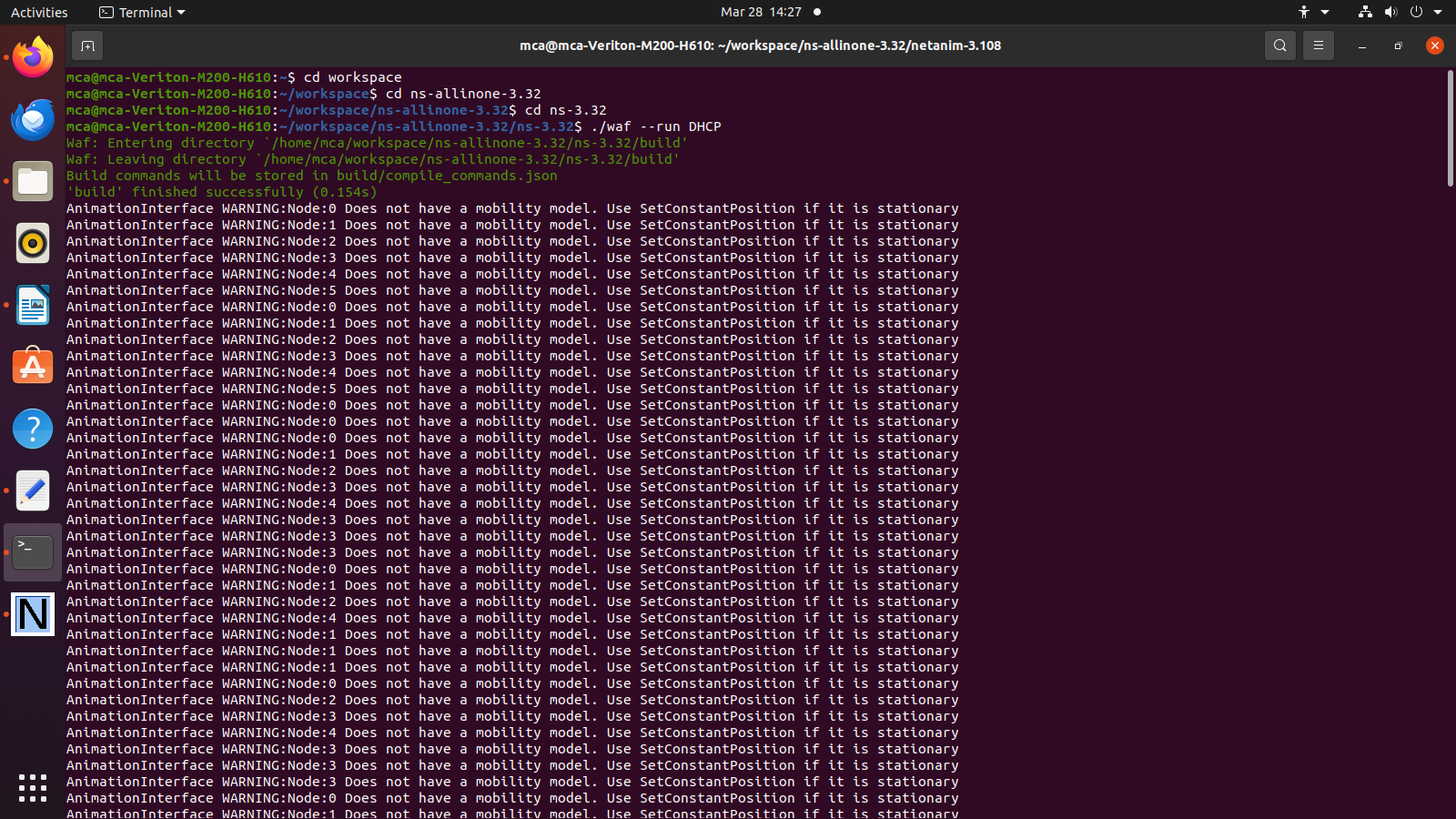
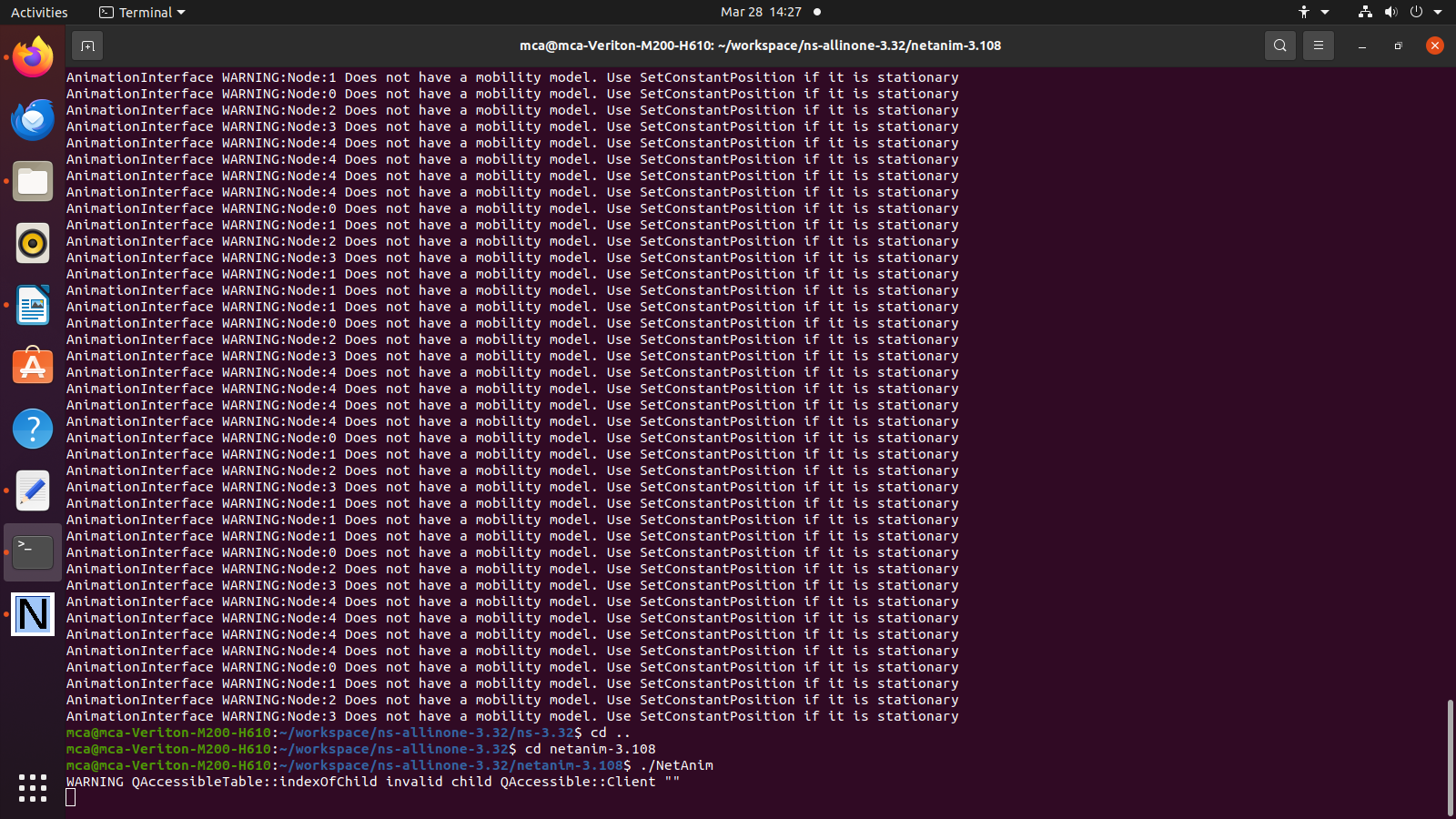
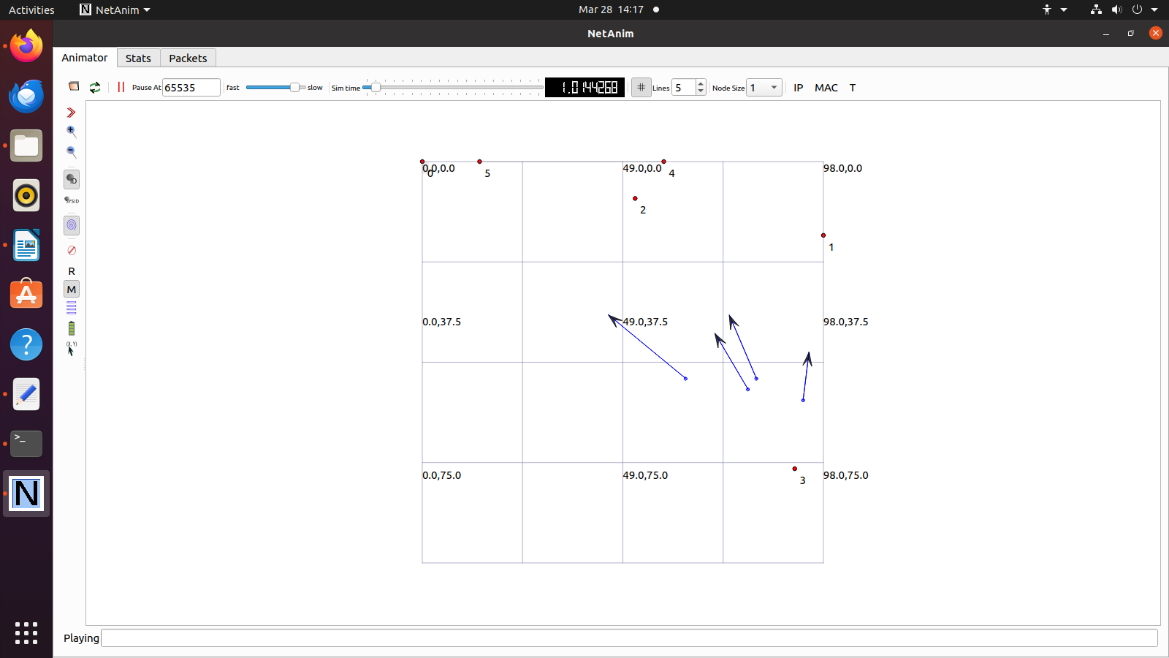
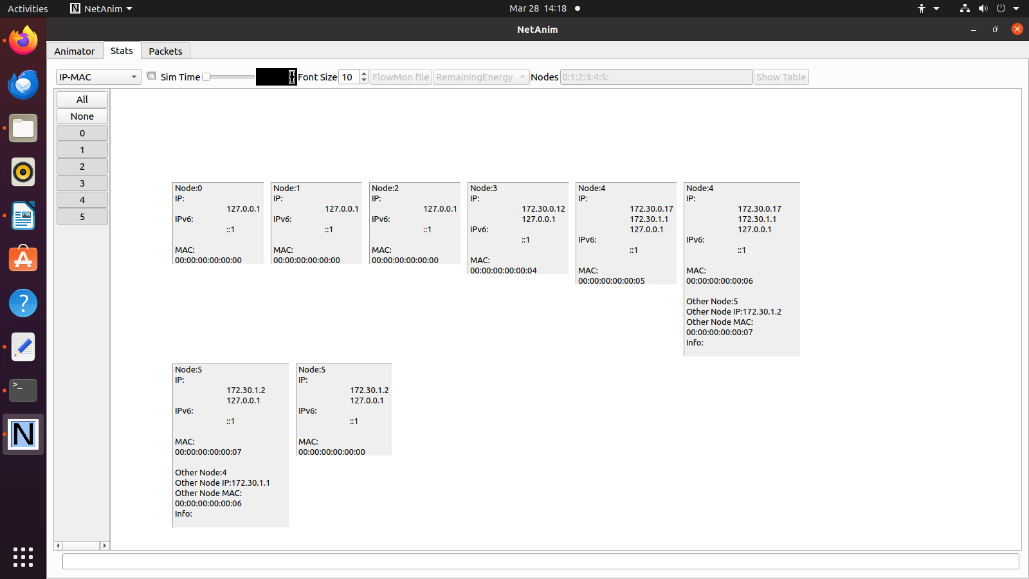
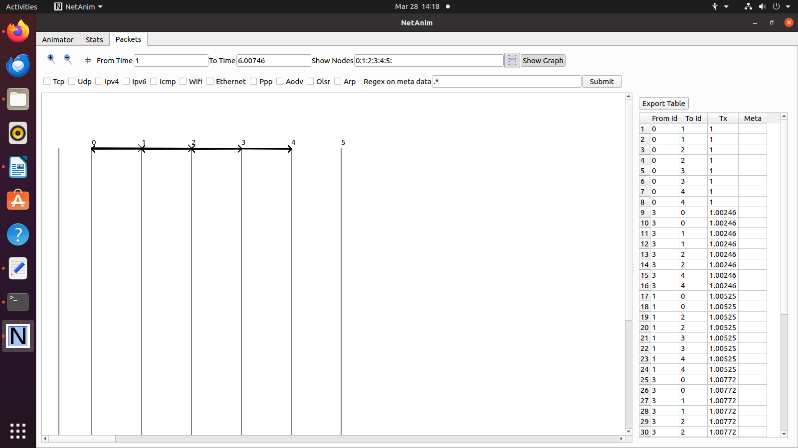
NS\_LOG\_INFO ("Run Simulation.");

Simulator::Run ();

Simulator::Destroy ();

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

}

**Output: -**

**Conclusion: -**

Successfully implemented DHCP server & n clients network using NS3.