**Practical No 13**

**Aim: -** WAP to assign IPV4 address on NS3.

**Objective: -**

* Understanding of IPV4 address
* Assignment of IPV4 using NS3

**Theory: -**

The IPv4 address is a 32-bit number that uniquely identifies a network interface on a machine. An IPv4 address is typically written in decimal digits, formatted as four 8-bit fields that are separated by periods. Each 8-bit field represents a byte of the IPv4 address.

Example:

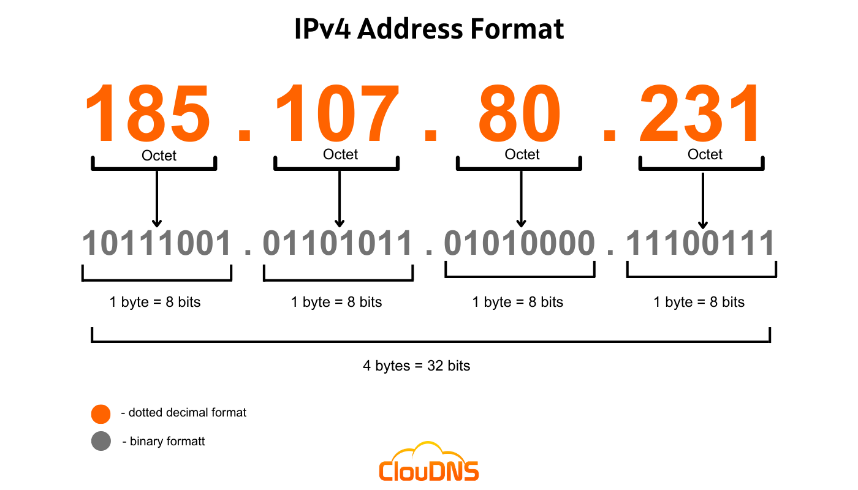
IPV address

192.168.0.1

192->First octet->0....255

168->Second octet->0....255

0->Third octet->0...255

1->Fourth octet->0...255

**Advantages of IPV4:**

* IPv4 security permits encryption to keep up privacy and security.
* IPV4 network allocation is significant and presently has quite 85000 practical routers.
* It becomes easy to attach multiple devices across an outsized network while not NAT.
* This is a model of communication so provides quality service also as economical knowledge transfer.
* IPV4 addresses are redefined and permit flawless encoding.

**Disadvantages of IPV4:**

* IP relies on network layer addresses to identify end-points on network, and each network has a unique IP address.
* The world’s supply of unique IP addresses is dwindling, and they might eventually run out theoretically.
* If there are multiple hosts, we need IP addresses of next class.

**Program: -**

#include <fstream>

#include "ns3/core-module.h"

#include "ns3/csma-module.h"

#include "ns3/applications-module.h"

#include "ns3/internet-module.h"

#include "ns3/network-module.h"

#include "ns3/netanim-module.h"

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("SocketOptionsIpv4");

void ReceivePacket (Ptr<Socket> socket)

{

NS\_LOG\_INFO ("Received one packet!");

Ptr<Packet> packet = socket->Recv ();

SocketIpTosTag tosTag;

if (packet->RemovePacketTag (tosTag))

{

NS\_LOG\_INFO (" TOS = " << (uint32\_t)tosTag.GetTos ());

}

SocketIpTtlTag ttlTag;

if (packet->RemovePacketTag (ttlTag))

{

NS\_LOG\_INFO (" TTL = " << (uint32\_t)ttlTag.GetTtl ());

}

}

static void SendPacket (Ptr<Socket> socket, uint32\_t pktSize,

uint32\_t pktCount, Time pktInterval )

{

if (pktCount > 0)

{

socket->Send (Create<Packet> (pktSize));

Simulator::Schedule (pktInterval, &SendPacket,

socket, pktSize,pktCount - 1, pktInterval);

}

else

{

socket->Close ();

}

}

int

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

{

//

// Allow the user to override any of the defaults and the above Bind() at

// run-time, via command-line arguments

//

uint32\_t packetSize = 1024;

uint32\_t packetCount = 10;

double packetInterval = 1.0;

//Socket options for IPv4, currently TOS, TTL, RECVTOS, and RECVTTL

uint32\_t ipTos = 0;

bool ipRecvTos = true;

uint32\_t ipTtl = 0;

bool ipRecvTtl = true;

CommandLine cmd;

cmd.AddValue ("PacketSize", "Packet size in bytes", packetSize);

cmd.AddValue ("PacketCount", "Number of packets to send", packetCount);

cmd.AddValue ("Interval", "Interval between packets", packetInterval);

cmd.AddValue ("IP\_TOS", "IP\_TOS", ipTos);

cmd.AddValue ("IP\_RECVTOS", "IP\_RECVTOS", ipRecvTos);

cmd.AddValue ("IP\_TTL", "IP\_TTL", ipTtl);

cmd.AddValue ("IP\_RECVTTL", "IP\_RECVTTL", ipRecvTtl);

cmd.Parse (argc, argv);

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

NodeContainer n;

n.Create (2);

InternetStackHelper internet;

internet.Install (n);

Address serverAddress;

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

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));

csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));

NetDeviceContainer d = csma.Install (n);

NS\_LOG\_INFO ("Assign IP Addresses.");

Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (d);

serverAddress = Address(i.GetAddress (1));

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

//Receiver socket on n1

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");

Ptr<Socket> recvSink = Socket::CreateSocket (n.Get (1), tid);

InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 4477);

recvSink->SetIpRecvTos (ipRecvTos);

recvSink->SetIpRecvTtl (ipRecvTtl);

recvSink->Bind (local);

recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

//Sender socket on n0

Ptr<Socket> source = Socket::CreateSocket (n.Get (0), tid);

InetSocketAddress remote = InetSocketAddress (i.GetAddress (1), 4477);

//Set socket options, it is also possible to set the options after the socket has been created/connected.

if (ipTos > 0)

{

source->SetIpTos (ipTos);

}

if (ipTtl > 0)

{

source->SetIpTtl (ipTtl);

}

source->Connect (remote);

AsciiTraceHelper ascii;

csma.EnableAsciiAll (ascii.CreateFileStream ("socket-options-ipv4.tr"));

csma.EnablePcapAll ("socket-options-ipv4", false);

//Schedule SendPacket

Time interPacketInterval = Seconds (packetInterval);

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),

Seconds (1.0), &SendPacket,

source, packetSize, packetCount, interPacketInterval);

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

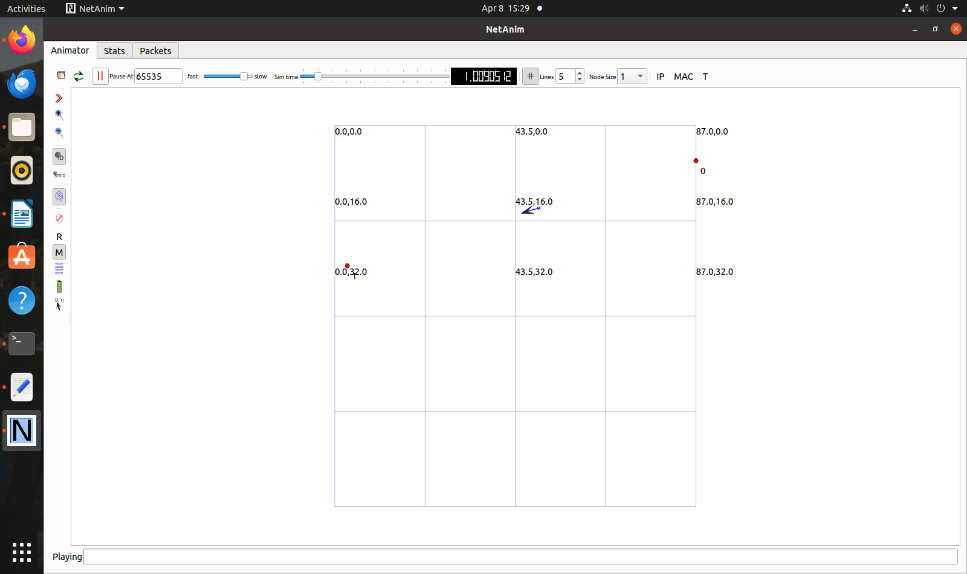
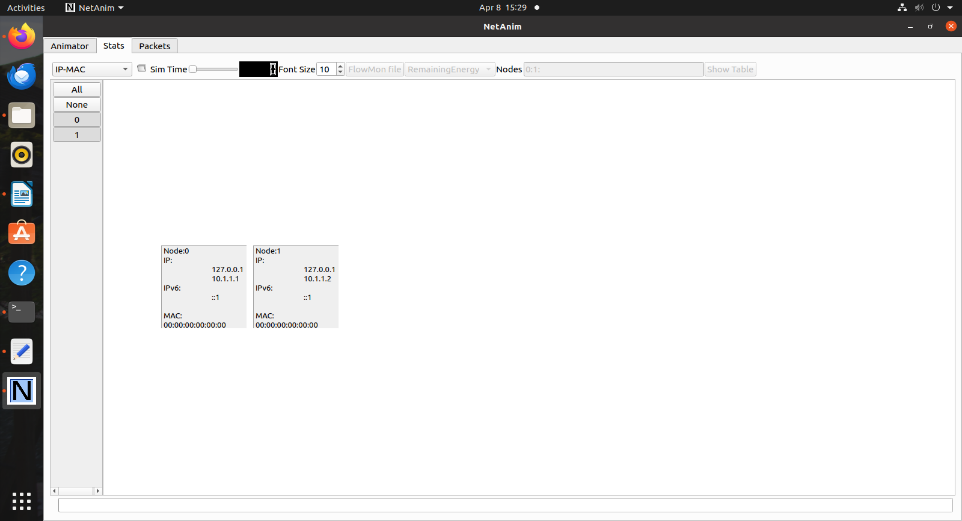
AnimationInterface anim("assign.xml");

Simulator::Run ();

Simulator::Destroy ();

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

}

**Output: -**

**Conclusion: -**

Successful assignment of TPV4 address on NS3.