**Practical No 7**

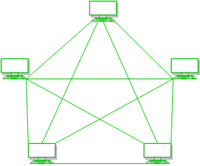
**Aim: -** WAP to simulate Mesh topology.

**Objective: -**

* Understanding of Mesh topology
* Implementation of Mesh topology using NS3

**Theory: -**

In mesh, all the computers are interconnected to every other during a network. Each computer not only sends its own signals but also relays data from other computers. The nodes are connected to every other completely via a dedicated link during which information is travel from nodes to nodes and there are N(N-1)/2 links in mesh if there are N nodes. Every node features a point-to-point connection to the opposite node. The connections within the mesh are often wired or wireless.



**Advantages of Mesh topology:**

* Failure during a single device won’t break the network.
* There is no traffic problem as there is a dedicated point to point links for every computer.
* Fault identification is straightforward.
* This topology provides multiple paths to succeed in the destination and tons of redundancy.
* It provides high privacy and security.
* Data transmission is more consistent because failure doesn’t disrupt its processes.
* Adding new devices won’t disrupt data transmissions.

**Disadvantages of Mesh topology:**

* It’s costly as compared to the opposite network topologies i.e. star, bus, point to point topology.
* Installation is extremely difficult in the mesh.
* Power requirement is higher as all the nodes will need to remain active all the time and share the load.
* The cost to implement mesh is above other selections.
* There is a high risk of redundant connections.
* Each node requires a further utility cost to think about.
* Maintenance needs are challenging with a mesh.

**Program: -**

#include "ns3/core-module.h"

#include "ns3/internet-module.h"

#include "ns3/network-module.h"

#include "ns3/netanim-module.h"

#include "ns3/applications-module.h"

#include "ns3/wifi-module.h"

#include "ns3/mesh-module.h"

#include "ns3/mobility-module.h"

#include "ns3/mesh-helper.h"

#include <iostream>

#include <sstream>

#include <fstream>

using namespace ns3;

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

class MeshTest{

public:

MeshTest ();

void Configure (int argc, char \*\* argv);

int Run ();

private:

int m\_xSize;

int m\_ySize;

double m\_step;

double m\_randomStart;

double m\_totalTime;

double m\_packetInterval;

uint16\_t m\_packetSize;

uint32\_t m\_nIfaces;

bool m\_chan;

bool m\_pcap;

std::string m\_stack;

std::string m\_root;

NodeContainer nodes;

NetDeviceContainer meshDevices;

//Addresses of interfaces:

Ipv4InterfaceContainer interfaces;

// MeshHelper. Report is not static methods

MeshHelper mesh;

private:

void CreateNodes ();

void InstallInternetStack ();

void InstallApplication ();

void Report ();

};

MeshTest::MeshTest () :

m\_xSize (3),

m\_ySize (3),

m\_step (100.0),

m\_randomStart (0.1),

m\_totalTime (100.0),

m\_packetInterval (0.1),

m\_packetSize (1024),

m\_nIfaces (1),

m\_chan (true),

m\_pcap (false),

m\_stack ("ns3::Dot11sStack"),

m\_root ("ff:ff:ff:ff:ff:ff")

{ }

void

MeshTest::Configure (int argc, char \*argv[]){

CommandLine cmd;

cmd.AddValue ("x-size", "Number of nodes in a row grid. [6]", m\_xSize);

cmd.AddValue ("y-size", "Number of rows in a grid. [6]", m\_ySize);

cmd.AddValue ("step", "Size of edge in our grid, meters. [100 m]", m\_step);

cmd.AddValue ("start", "Maximum random start delay, seconds. [0.1 s]", m\_randomStart);

cmd.AddValue ("time", "Simulation time, seconds [100 s]", m\_totalTime);

cmd.AddValue ("packet-interval", "Interval between packets in UDP ping, seconds [0.001 s]", m\_packetInterval);

cmd.AddValue ("packet-size", "Size of packets in UDP ping", m\_packetSize);

cmd.AddValue ("interfaces", "Number of radio interfaces used by each mesh point. [1]", m\_nIfaces);

cmd.AddValue ("channels", "Use different frequency channels for different interfaces. [0]", m\_chan);

cmd.AddValue ("pcap", "Enable PCAP traces on interfaces. [0]", m\_pcap);cmd.AddValue ("stack", "Type of protocol stack. ns3::Dot11sStack by default", m\_stack);

cmd.AddValue ("root", "Mac address of root mesh point in HWMP", m\_root);

cmd.Parse (argc, argv);

NS\_LOG\_DEBUG ("Grid:" << m\_xSize << "\*" << m\_ySize);

NS\_LOG\_DEBUG ("Simulation time: " << m\_totalTime << " s"); }

void

MeshTest::CreateNodes ()

{

nodes.Create (m\_ySize\*m\_xSize);

// Configure YansWifiChannel

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();

wifiPhy.SetChannel (wifiChannel.Create ());

mesh = MeshHelper::Default ();

if (!Mac48Address (m\_root.c\_str ()).IsBroadcast ())

{

mesh.SetStackInstaller (m\_stack, "Root", Mac48AddressValue (Mac48Address (m\_root.c\_str ())));}

else{

mesh.SetStackInstaller (m\_stack);

}

if (m\_chan){

mesh.SetSpreadInterfaceChannels (MeshHelper::SPREAD\_CHANNELS);

}

else{

mesh.SetSpreadInterfaceChannels (MeshHelper::ZERO\_CHANNEL);}

mesh.SetMacType ("RandomStart", TimeValue (Seconds (m\_randomStart)));

// Set number of interfaces - default is single-interface mesh point

mesh.SetNumberOfInterfaces (m\_nIfaces);

// Install protocols and return container if MeshPointDevices

meshDevices = mesh.Install (wifiPhy, nodes);

// Setup mobility - static grid topology

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",

"MinX", DoubleValue (0.0),

"MinY", DoubleValue (0.0),

"DeltaX", DoubleValue (m\_step),

"DeltaY", DoubleValue (m\_step),

"GridWidth", UintegerValue (m\_xSize),

"LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility.Install (nodes);

if (m\_pcap)

wifiPhy.EnablePcapAll (std::string ("mp-"));}

void

MeshTest::InstallInternetStack ()

InternetStackHelper internetStack;

internetStack.Install (nodes);

Ipv4AddressHelper address;

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

interfaces = address.Assign (meshDevices);}

void

MeshTest::InstallApplication (){

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));

serverApps.Start (Seconds (0.0));

serverApps.Stop (Seconds (m\_totalTime));

UdpEchoClientHelper echoClient (interfaces.GetAddress (0), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue ((uint32\_t)(m\_totalTime\*(1/m\_packetInterval))));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (m\_packetInterval)));

echoClient.SetAttribute ("PacketSize", UintegerValue (m\_packetSize)); ApplicationContainer clientApps = echoClient.Install (nodes.Get (m\_xSize\*m\_ySize-1));

clientApps.Start (Seconds (0.0));

clientApps.Stop (Seconds (m\_totalTime));

}

int

MeshTest::Run (){

CreateNodes ();

InstallInternetStack ();

InstallApplication ();

Simulator::Schedule (Seconds (m\_totalTime), &MeshTest::Report, this);

Simulator::Stop (Seconds (m\_totalTime));

// Create the animation object and configure for specified output

AnimationInterface anim ("meshT.xml");

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

Simulator::Run ();

Simulator::Destroy ();

return 0;}

void

MeshTest::Report (){

unsigned n (0);

for (NetDeviceContainer::Iterator i = meshDevices.Begin (); i != meshDevices.End (); ++i, ++n)

{

std::ostringstream os;

os << "mp-report-" << n << ".xml";

std::cerr << "Printing mesh point device #" << n << " diagnostics to " << os.str () << "\n";

std::ofstream of;

of.open (os.str ().c\_str ());

if (!of.is\_open ())

{

std::cerr << "Error: Can't open file " << os.str () << "\n";

return;

}

mesh.Report (\*i, of);

of.close ();

}

}

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

{

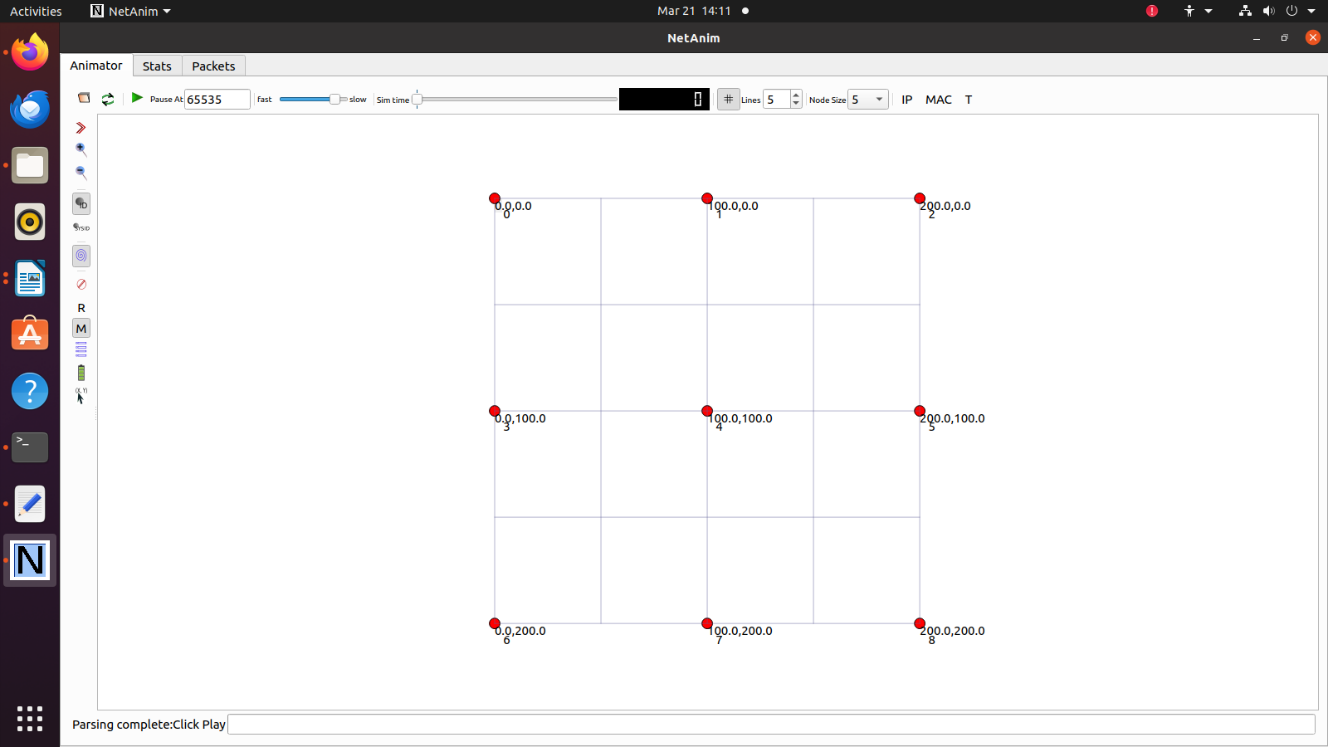
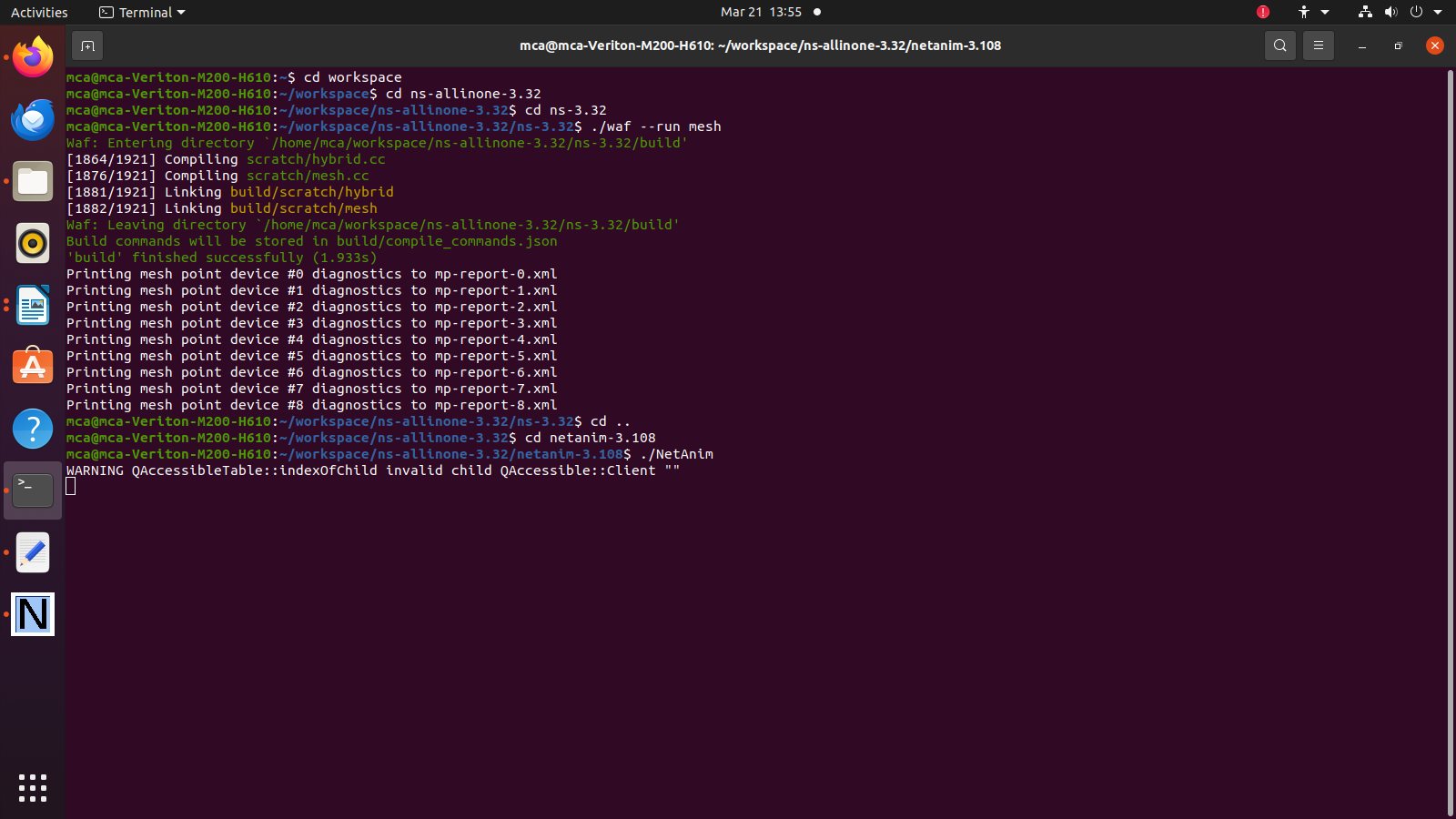
MeshTest t;

t.Configure (argc, argv);

return t.Run ();

}

**Output: -**



**Conclusion: -**

Successfully implemented Mesh topology using NS3.