| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Assignment Number:** 3 | |
| **Aim of Assignment:**  Write a program to simulate star topology | | | |
| **DOP:** 17.4.23 | | **DOS:** 29.4.23 | |
| **CO Mapped:**  CO2 | **PO Mapped:**  PO1, PO2, PO3, PO5, PO7, PSO1 | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 3

**Aim:** Write a program to simulate star topology

**Theory:**

* Star topology is a type of network topology in which every device in the network is individually connected to a central node, known as the switch or hub. When represented visually, this topology resembles a star which gives it its name.
* Star topologies are often combined with bus topologies, resulting in what’s called a tree. This occurs when the switch of the star topology is connected to the backbone of the bus topology.
* Working of Star topology
  + The working principle of star topology is; it doesn’t allow communication directly between different connected devices like in a mesh. But the communication is possible by using a central device like a hub available within the network. This central device/hub might be an active hub, passive hub, or switch which is responsible for both transmitting & receiving messages from the sender.
* PointToPointStarHelper
  + A helper to make it easier to create a star topology with PointToPoint links
* PacketSinkHelper
  + A helper to make it easier to instantiate an ns3::PacketSinkApplication on a set of nodes.
* OnOffHelper
  + A helper to make it easier to instantiate an ns3::OnOffApplication on a set of nodes.
* Ipv4GlobalRoutingHelper
  + Helper class that adds ns3::Ipv4GlobalRouting objects

**Code:**

#include "ns3/applications-module.h"

#include "ns3/core-module.h"

#include "ns3/internet-module.h"

#include "ns3/mobility-module.h"

#include "ns3/netanim-module.h"

#include "ns3/network-module.h"

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

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

#include <bits/stdint-uintn.h>

// Network topology (default)

//

// n2 n3 n4 .

// \ | / .

// \|/ .

// n1--- n0---n5 .

// /|\ .

// / | \ .

// n8 n7 n6 .

//

using namespace ns3;

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

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;

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

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

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("star");

// create trace file

MobilityHelper mobility;

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

mobility.SetPositionAllocator(

"ns3::GridPositionAllocator", "MinX", DoubleValue(0.0), "MinY",

DoubleValue(0.0), "DeltaX", DoubleValue(45.0), "DeltaY",

DoubleValue(80.0), "GridWidth", UintegerValue(5), "LayoutType",

StringValue("RowFirst"));

mobility.Install(star.GetHub());

for (uint32\_t i{0}; i < star.SpokeCount(); ++i) {

mobility.Install(star.GetSpokeNode(i));

}

AnimationInterface{"test.xml"};

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

Simulator::Run();

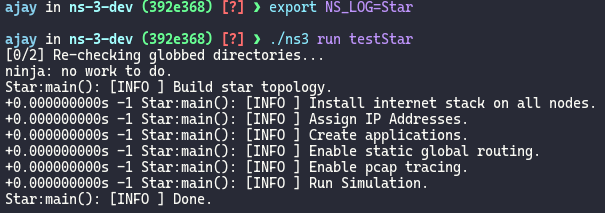
Simulator::Destroy();

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

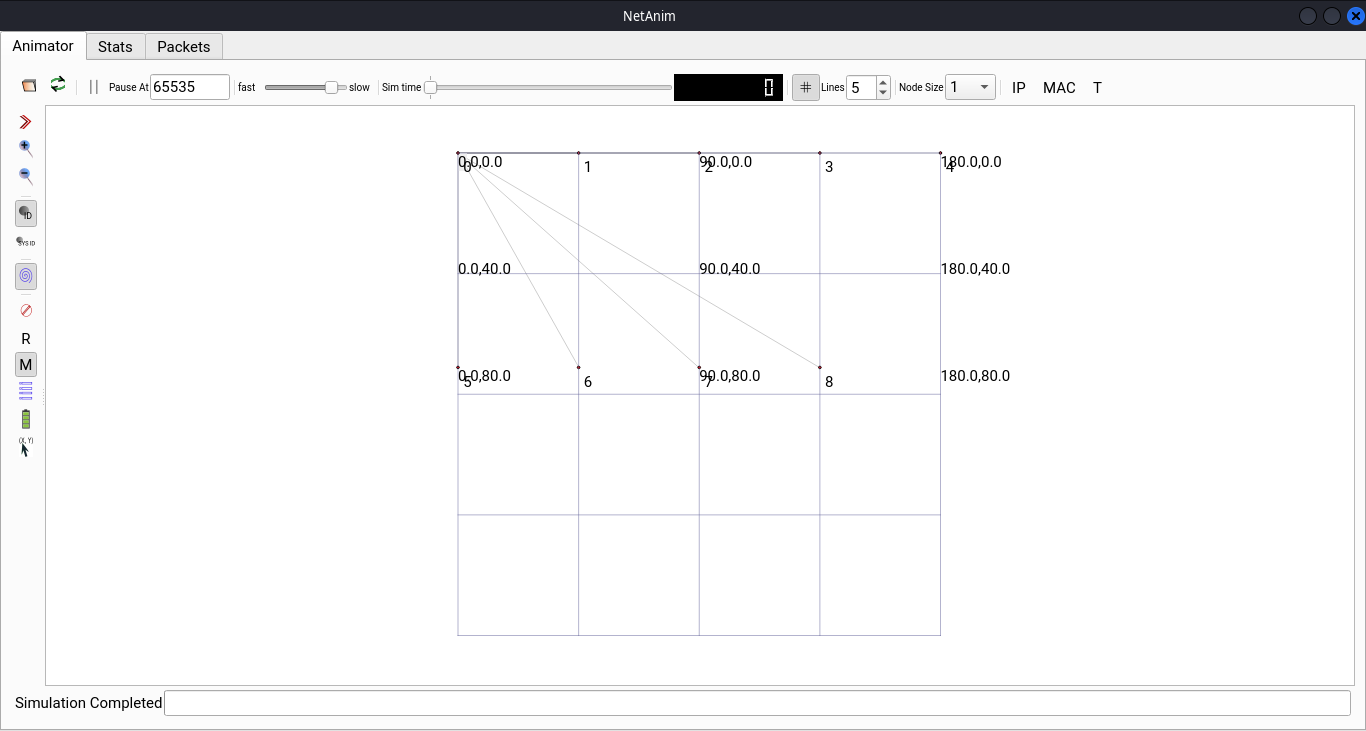
return 0;

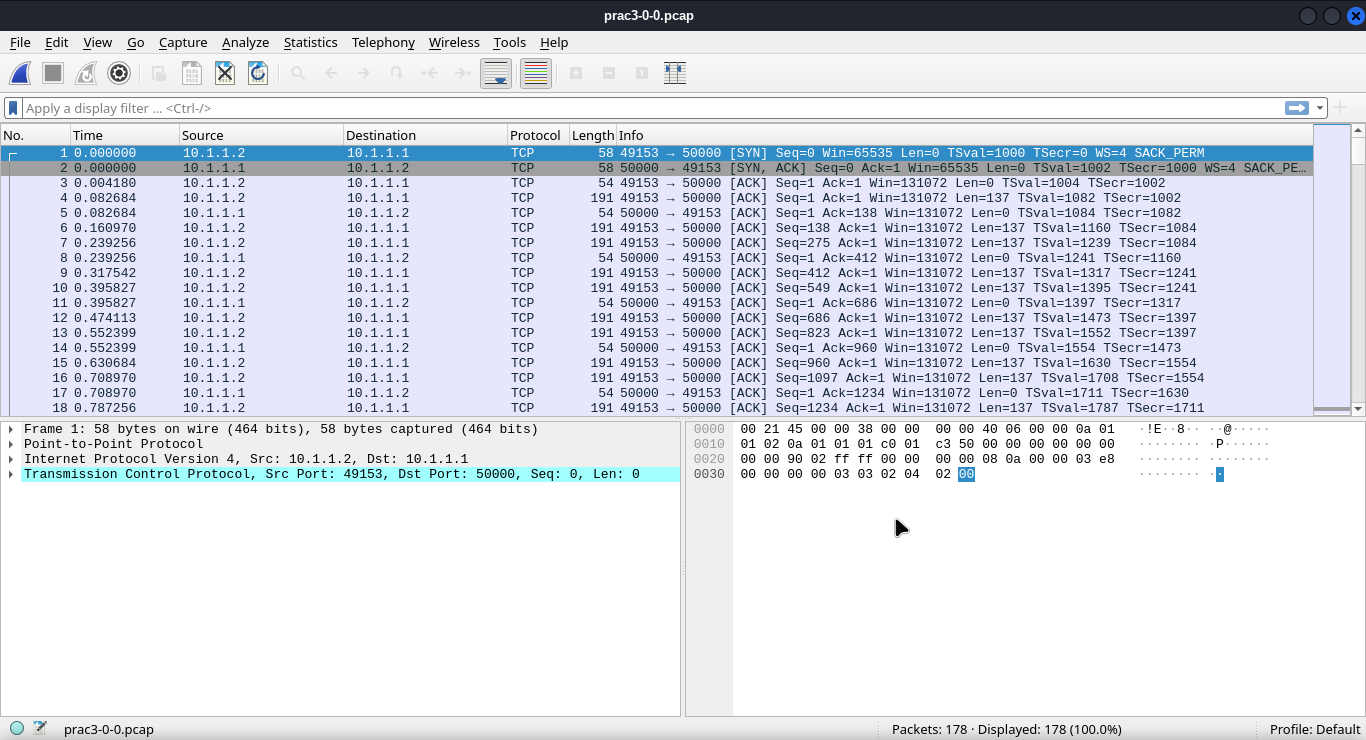
}

**Output:**









**Conclusion:**

I learnt how to write a program to simulate star topology.