


<b>Name of Student: Sunny Satish Halkatti</b>		
<b>Roll Number: 17 (B)</b>		<b>Lab Assignment Number: 3</b>
<b>Title of Lab Assignment:</b> <b>Program to simulate star topology</b>		
<b>DOP: 13/04/2023</b>		<b>DOS: 20/04/2023</b>
<b>CO:</b> CO2	<b>PO:</b> PO1, PO2, PO3, PO5,PO7, PSO1	<b>Signature:</b> 

### NWL - Practical-3

**Aim:** Program to simulate star topology

**Theory:**

#### **Network Topology**

Network topology is the arrangement of the elements of a communication network, such as nodes, links, routers, switches, etc. It determines how data is transmitted and received among different devices in the network.

The most basic components of a network topology are nodes, which are devices that are connected to the network, and links, which are the communication channels between these devices. Nodes can include computers, servers, printers, routers, switches, and other devices.

Network topology can be divided into two main categories: physical and logical. Physical topology is the placement of the various components of a network (e.g., device location and cable installation), while logical topology illustrates how data flows within a network. Distances between nodes, physical interconnections, transmission rates, or signal types may differ between two different networks, yet their logical topologies may be identical. A network's physical topology is a particular concern of the physical layer of the OSI model.

There are different types of physical network topologies, such as bus, ring, mesh, tree, etc., each with its own advantages and disadvantages.

#### **Star Topology**

A star topology is a network topology in which all devices are connected to a central hub or switch, forming a star-like structure. Data transmitted from one device passes through the hub/switch and is sent to the intended device. This topology is commonly used in Ethernet LANs, wireless networks, and telephone networks.

#### **Advantages of the star topology:**

1. **Easy to Install and Manage:** The star topology is easy to install and manage, making it ideal for small to medium-sized networks. This is because new devices can be easily added to the network by connecting them to the central hub/switch.
2. **Scalable:** The star topology is highly scalable, allowing new devices to be added to the network without affecting the rest of the network. This makes it an ideal choice for networks that are expected to grow over time.
3. **Easy Fault Detection and Isolation:** If a device in the network fails, it is easy to isolate and replace the faulty device without affecting the rest of the network. This is

because each device is connected to the central hub/switch, which allows for easy fault detection and isolation.

4. **High Reliability:** The star topology is highly reliable because each device is connected to the central hub/switch. This means that if one device fails, the rest of the network is not affected.

**Disadvantages of the star topology:**

1. **More Expensive to Implement:** The star topology is more expensive to implement than other topologies, such as the bus topology, because it requires more cabling and a central hub/switch.

2. **Single Point of Failure:** The central hub/switch is a single point of failure in the network. If the hub/switch fails, the entire network goes down.

3. **Requires More Cabling:** The star topology requires more cabling than other topologies, such as the bus topology. This can make it more difficult and expensive to install and manage.

4. **Network Congestion:** The star topology can suffer from network congestion if a large amount of data is being transmitted to multiple devices at the same time. This is because all data must pass through the central hub/switch, which can become a bottleneck.

Despite its disadvantages, the star topology is widely used in many modern networks because it is highly reliable and can be easily expanded or modified without affecting the rest of the network.

**There are two main types of star topology:**

1. **Physical Star Topology:** In this type of star topology, each device is physically connected to the central hub/switch using a cable.

2. **Logical Star Topology:** In this type of star topology, devices are connected to the network wirelessly, but the network is still organized as a star topology with a central hub/switch. The star topology is a solid choice for smaller networks that require a high level of reliability and ease of management.

However, it is more expensive to implement than other topologies and can suffer from network congestion if a large amount of data is being transmitted. Despite these drawbacks, the star topology remains a popular choice for many modern networks due to its scalability and easy fault detection and isolation.

**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 ("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;
```

```
std::string animfile="stanim.xml";

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

//cmd.AddValue ("animFile", "File Name for Animation Output", animFile);
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 ("10.1.0.0", "255.255.0.0"));
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");

// Set the bounding box for animation
star.BoundingBox (1, 1, 100, 100);

// Create the animation object and configure for specified output
AnimationInterface anim (animfile);

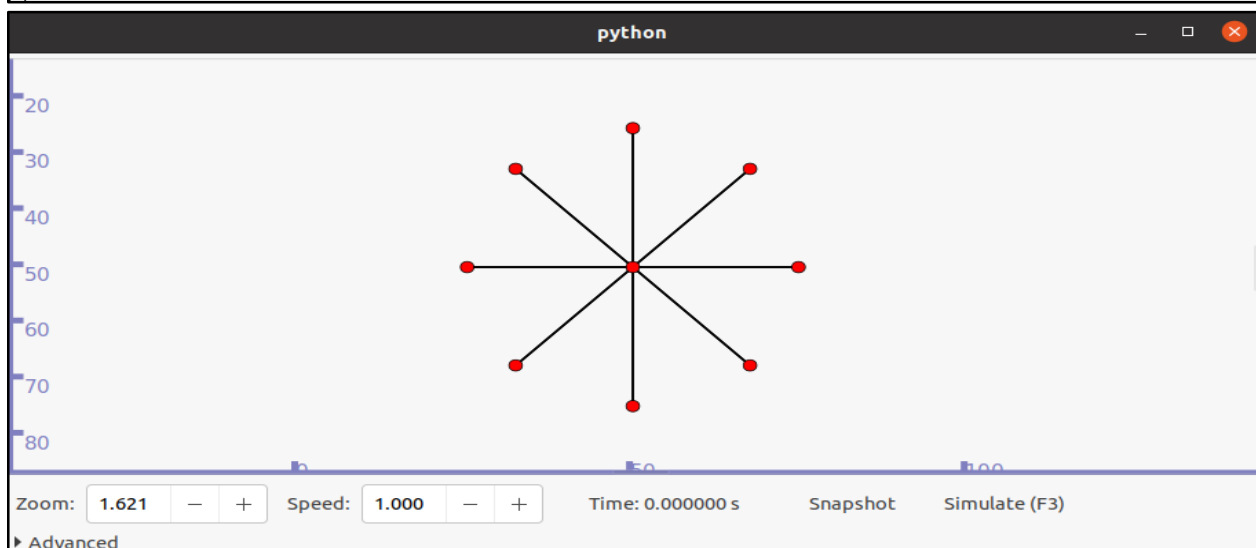
NS_LOG_INFO ("Run Simulation.");
Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");
return 0;
```

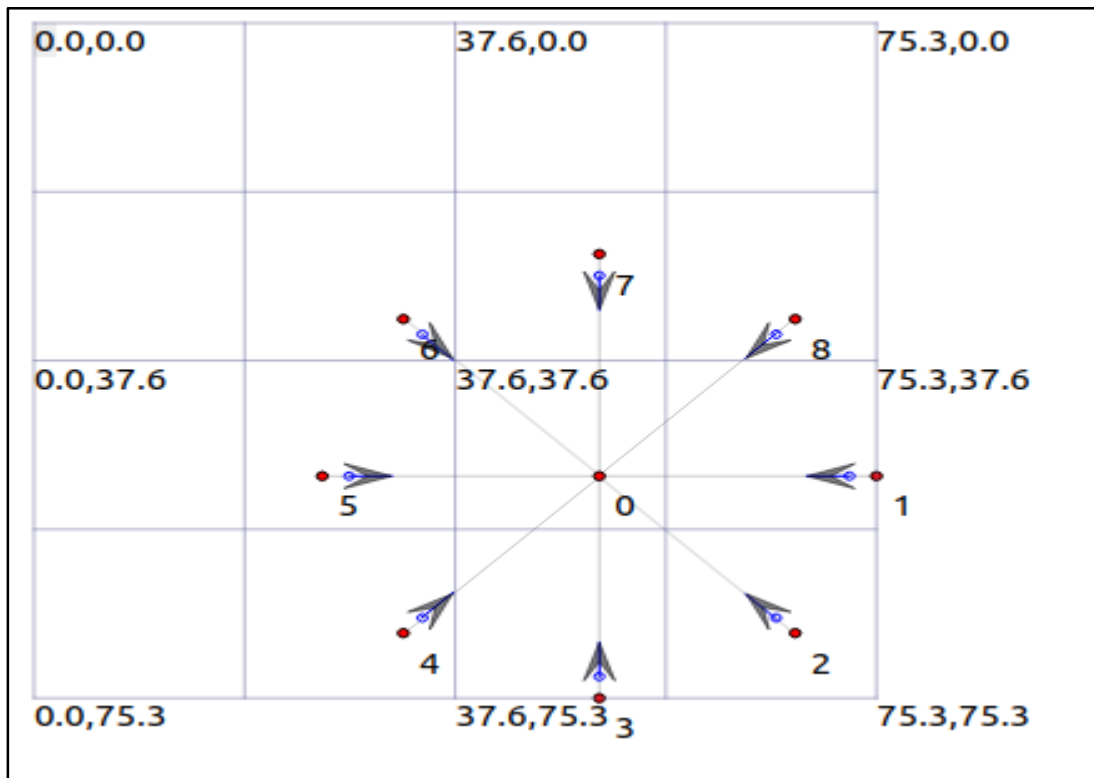
```
}
```

**Output:**

```
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf --run star.cc
Waf: Entering directory `/home/vaish/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vaish/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.282s)
```

```
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf --run star --vis
Waf: Entering directory `/home/vaish/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vaish/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.272s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 9 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
```

**NetAnim**

**Conclusions:**

We have successfully implemented a program to simulate star topology in this practical.