<table>
<tr><td colspan="3">Name of Student: Sunny Satish Halkatti</td></tr>
<tr><td colspan="2">Roll Number: 17 (B)</td><td>Lab Assignment Number: 4</td></tr>
<tr><td colspan="3">Title of Lab Assignment:
To write a program to simulate bus topology.</td></tr>
<tr><td colspan="2">DOP: 13/04/2023</td><td>DOS: 20/04/2023</td></tr>
<tr><td>CO:
CO2</td><td>PO:
PO1, PO2, PO3, P05,P07, PSO1</td><td>Signature:</td></tr>
</table>

NWL - Practical 4

Aim: To write a program to simulate bus 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.

Bus Topology

*A bus topology is a type of network topology in which all devices are connected to a common transmission medium known as a **bus***. The bus is typically a single cable that runs the length of the network, connecting all devices to each other. In this topology, data is transmitted along the bus, and all devices receive the transmission, regardless of whether or not the data is intended for them.

Advantages of Bus Topology

Simplicity and Cost-effectiveness: Bus topology is one of the simplest and most cost-effective network topologies to implement. The transmission medium (i.e. the bus) is a single cable, which makes it easy to install and manage. This makes it a popular choice for small to medium-sized networks.

Easy to Expand: Adding new devices to a bus network is easy as all that is required is to connect the device to the bus. This makes the bus topology a scalable and flexible option for networks that are expected to grow over time.

Easy to Troubleshoot: Troubleshooting problems in a bus network is relatively easy since there are only a few components involved. The transmission medium, devices, and connectors are the primary components of a bus network. If a device fails or the cable is damaged, it is easy to identify the problem and fix it.

Efficient for Low Traffic Networks: Bus topology is efficient for networks with low traffic because all devices receive the transmission. However, in high traffic networks, this can lead to congestion and slow down the network.

Disadvantages of Bus Topology

Single Point of Failure: A significant disadvantage of the bus topology is that it has a single point of failure. If the cable is damaged, disconnected, or cut in any way, the entire network will go down.

Limited Scalability: Although bus topology is scalable, it is limited in terms of the number of devices that can be added to the network. As the number of devices increases, the performance of the network decreases, leading to network congestion and a decrease in data transfer rates.

Security Issues: Since all devices receive the transmission in a bus network, there are security issues associated with this topology. If an unauthorized user taps into the bus, they can intercept and view all data transmitted across the network.

Difficult to Isolate Faults: In a bus topology, it can be challenging to isolate faults when they occur. If there is a problem with the transmission medium or a device, it can be difficult to identify the source of the problem.

Code:

```
#include "ns3/core-module.h"
#include "ns3/network-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/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

// Default Network Topology
//
//      172.16.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 176.16.2.0


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int main (int argc, char *argv[])
{
  bool verbose = true; //to keep log information
  uint32_t nCsma = 3;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

  cmd.Parse (argc,argv);
```

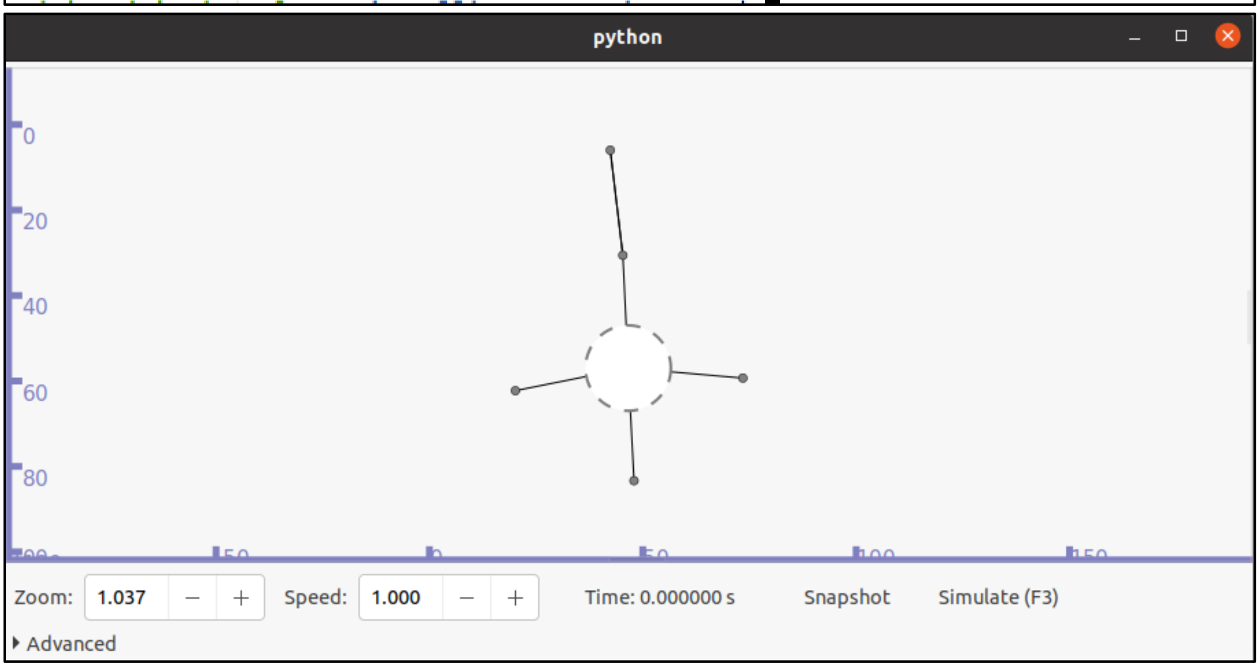
```
if (verbose)
  {
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
  }

nCsma = nCsma == 0 ? 1 : nCsma;

NodeContainer p2pNodes;
p2pNodes.Create (2);

NodeContainer csmaNodes; /*different from first practical*/
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);/*end*/

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

CsmaHelper csma;/*Different from first*/
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));/*end*/

NetDeviceContainer csmaDevices;/*csma is the only change from first*/
csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));
stack.Install (csmaNodes);

Ipv4AddressHelper address;
address.SetBase ("172.16.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
```

```
  address.SetBase ("172.16.2.0", "255.255.255.0");/*Different*/
  Ipv4InterfaceContainer csmaInterfaces;
  csmaInterfaces = address.Assign (csmaDevices);/*end*/

  UdpEchoServerHelper echoServer (2000);

  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));// we
will get the last node
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));


  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 2000);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0)); // installing
at 0th node
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();/*Automatically create the routing
tables*/

  pointToPoint.EnablePcapAll ("p2p");
  csma.EnablePcap ("csma1", csmaDevices.Get (1), true);
  csma.EnablePcap ("csma2", csmaDevices.Get (2), true);

//AnimationInterface anim("bus.xml");/*Net ANIM ABOVE 2 MODULES WE HAVE
TO ADD Part 2*/

//AnimationInterface::SetConstantPosition(p2pNodes.Get(0),10,10);/*Part 3*/
//AnimationInterface::SetConstantPosition(p2pNodes.Get(1),20,20);
//AnimationInterface::SetConstantPosition(csmaNodes.Get(1),30,30);
//AnimationInterface::SetConstantPosition(csmaNodes.Get(2),40,40);
```

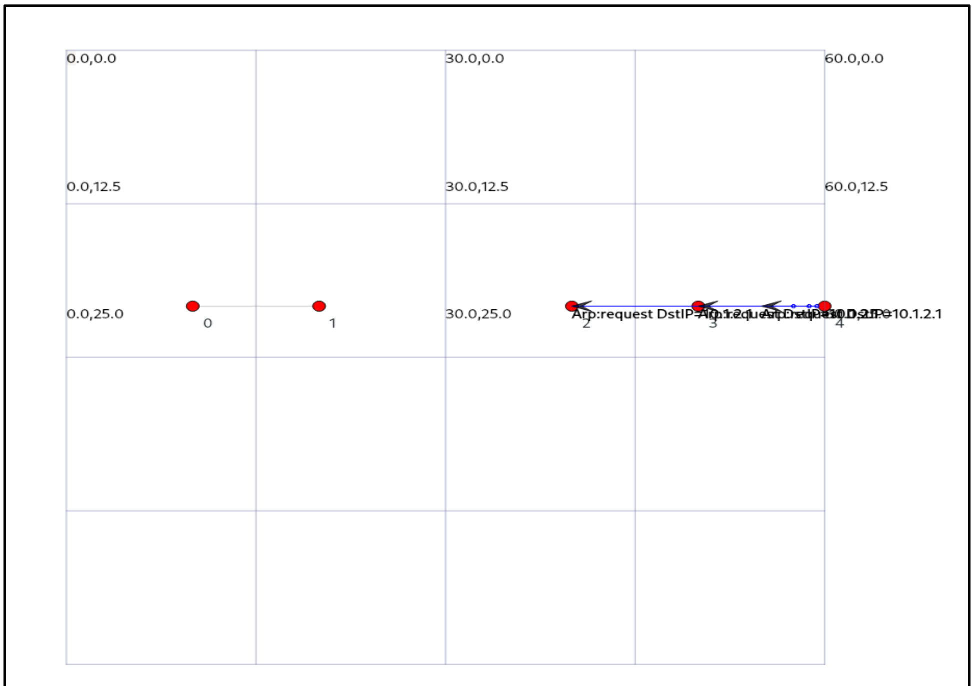
```
//AnimationInterface::SetConstantPosition(csmaNodes.Get(3),50,50);/*Part 3 end*/

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

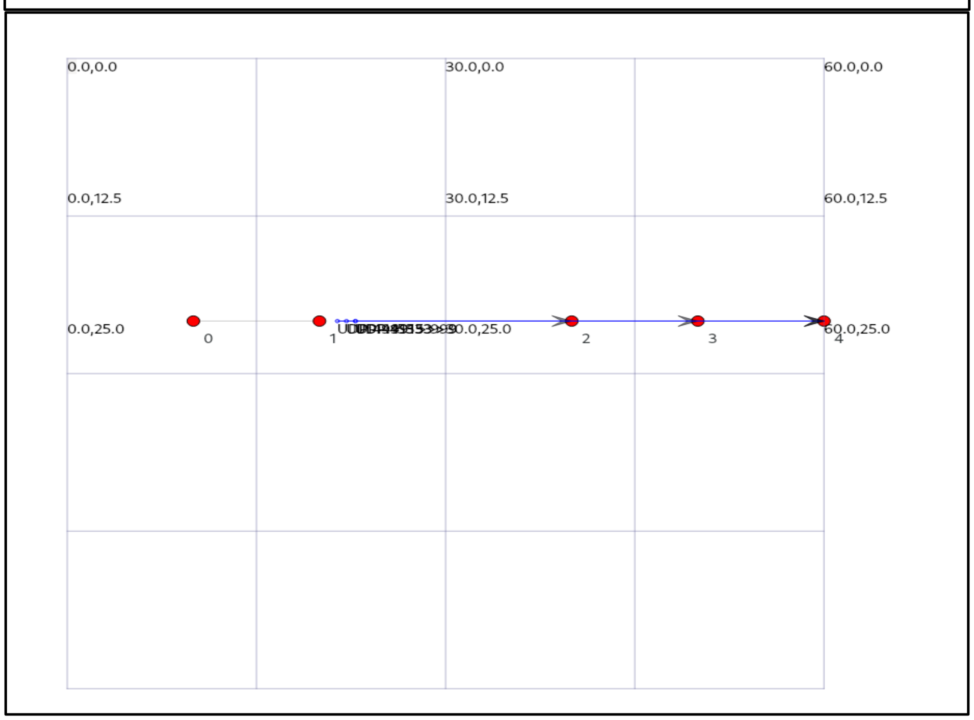
Output:

```
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf --run bus_prog.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 (0.252s)
At time +2s client sent 1024 bytes to 172.16.2.4 port 2000
At time +2.0078s server received 1024 bytes from 172.16.1.1 port 49153
At time +2.0078s server sent 1024 bytes to 172.16.1.1 port 49153
At time +2.01761s client received 1024 bytes from 172.16.2.4 port 2000
```



NetAnim Output:





Conclusions:

We have successfully implemented a program to simulate bus topology.