

Aim: Program to simulate Bus topology.

THEORY:

Understanding Network Topology:

Network topology is the arrangement of the elements (links, nodes, etc.) of a communication network. Network topology can be used to define or describe the arrangement of various types of telecommunication networks, including command and control radio networks, industrial field busses and computer networks.

Network topology is the topological structure of a network and may be depicted physically or logically. It is an application of graph theory wherein communicating devices are modelled as nodes and the connections between the devices are modelled as links or lines between the nodes.

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.

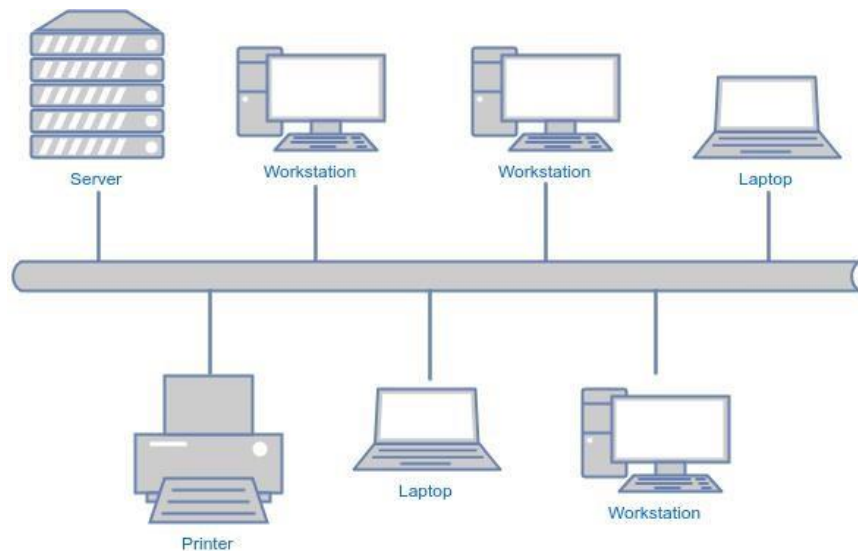
Examples of network topologies are found in local area networks (LAN), a common computer network installation. Any given node in the LAN has one or more physical links to other devices in the network; graphically mapping these links results in a geometric shape that can be used to describe the physical topology of the network.

Bus Topology:

Bus topology is a specific kind of network topology in which all of the various devices in the network are connected to a single cable or line. In general, the term refers to how various devices are set up in a network.

Alternatively mentioned as line topology, bus topology could even be a specific quite topology during which each computer and network device is connected to a minimum of one cable or backbone. In general, term refers to how various devices are acknowledged during a network. counting on sort of network card, a coax or an RJ-45 network cable is employed to attach them together.

Bus topology carries transmitted data through cable. because data reaches each node, node checks destination address (MAC/IP address) to work out if it matches their address. If address does not match with node, node does nothing more. But if addresses of node match to address contained within data, then they process on knowledge. In bus, communication between nodes are done through foremost network cable.



Bus Topology Network

Advantages of Bus Topology:

- It is the easiest network topology for connecting peripherals or computers in a linear fashion.
- It works very efficient well when there is a small network.
- Length of cable required is less than a star topology.
- It is easy to connect or remove devices in this network without affecting any other device.
- Very cost-effective as compared to other network topology i.e. mesh and star
- It is easy to understand topology.
- Easy to expand by joining the two cables together.

Disadvantages of Bus Topology:

- Bus topology is not great for large networks.
- Identification of problem becomes difficult if whole network goes down.
- Troubleshooting of individual device issues is very hard.
- Need of terminators are required at both ends of main cable.
- Additional devices slow network down.
- If a main cable is damaged, whole network fails or splits into two.
- Packet loss is high.
- This network topology is very slow as compared to other topologies.

SOURCE 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"

// Default Network Topology
//
//      172.16.1.0
// n0 ----- n1  n2  n3  n4
// point-to-point |  |  |  |
//      =====
//      LAN 172.16.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int
main (int argc, char *argv[])
{
    bool verbose = true;
    uint32_t nCsmas = 3;

    CommandLine cmd (__FILE__);
    cmd.AddValue ("nCsmas", "Number of \"extra\" CSMA nodes/devices", nCsmas);
    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);
    }

    nCsmas = nCsmas == 0 ? 1 : nCsmas;

    NodeContainer p2pNodes;
    p2pNodes.Create (2);

    NodeContainer csmaNodes;
    csmaNodes.Add (p2pNodes.Get (1));
    csmaNodes.Create (nCsmas);

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

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

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
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");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsmas));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsmas), 9);
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

pointToPoint.EnablePcapAll ("second2");
csma.EnablePcap ("second2", csmaDevices.Get (2), true);

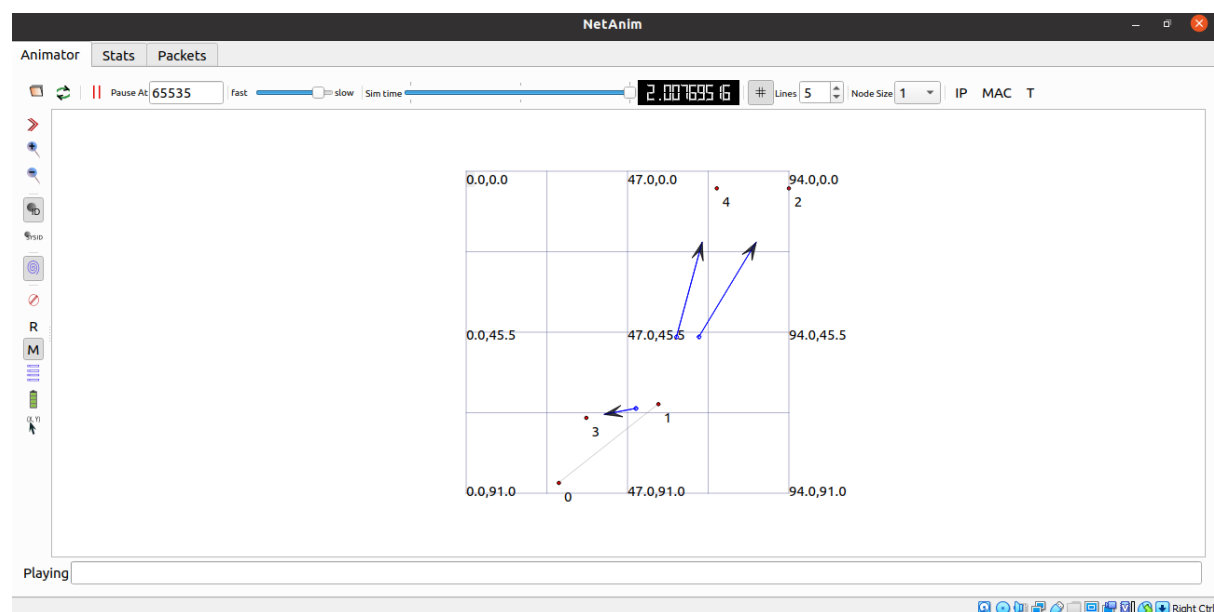
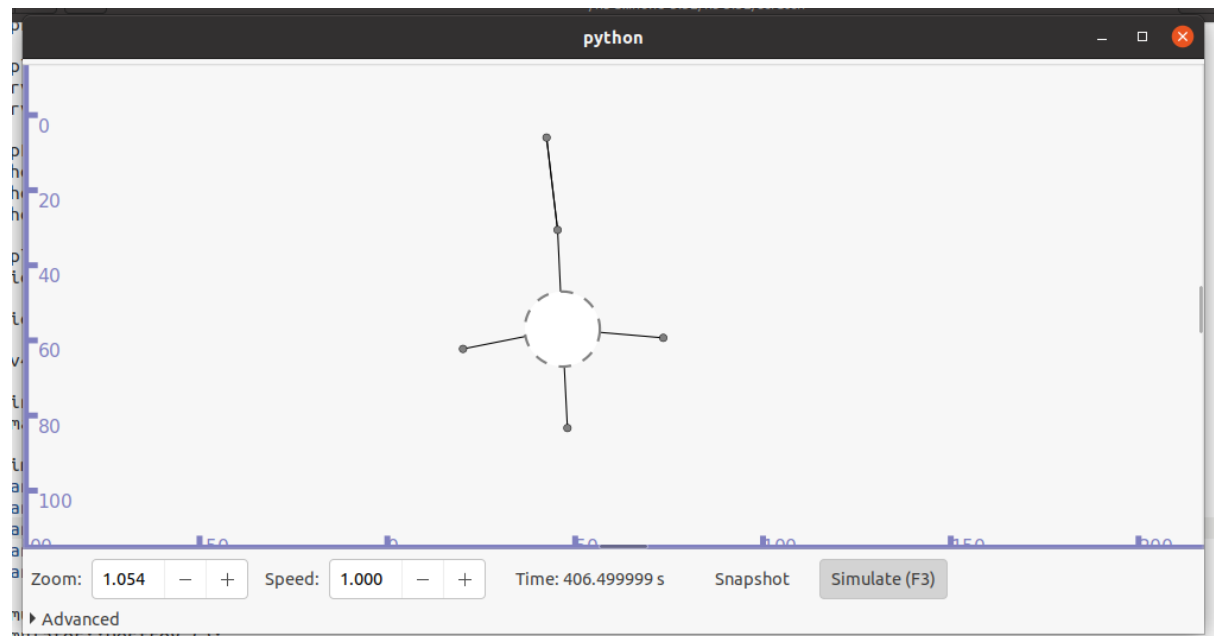
AnimationInterface anim ("anim3.xml");
//anim.SetConstantPosition (p2pNodes.Get(1),10.0,10.0);
//anim.SetConstantPosition (p2pNodes.Get(2),20.0,20.0);
//anim.SetConstantPosition (csmaNodes.Get(1),10.0,10.0);
//anim.SetConstantPosition (csmaNodes.Get(2),10.0,10.0);
//anim.SetConstantPosition (csmaNodes.Get(3),10.0,10.0);
```

```

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

```

OUTPUT:



```

vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ wireshark second2-0-0.pcap
^C
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ wireshark second2-1-0.pcap
^C
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ wireshark second2-3-0.pcap
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$

```

second2-0-0.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.1.1	172.16.2.4	UDP	1054	49153 → 9 Len=1024
2	0.017607	172.16.2.4	172.16.1.1	UDP	1054	9 → 49153 Len=1024

Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
 Point-to-Point Protocol
 Internet Protocol Version 4, Src: 172.16.1.1, Dst: 172.16.2.4
 User Datagram Protocol, Src Port: 49153, Dst Port: 9
 Data (1024 bytes)

0000 00 21 45 00 04 1c 00 00 00 00 40 11 00 00 ac 10 ..!E.....@.....
 0010 01 01 ac 10 02 04 c0 01 00 09 04 08 00 00 00 00@.....

second2-0-0.pcap Packets: 2 · Displayed: 2 (100.0%) Profile: Default

second2-1-0.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

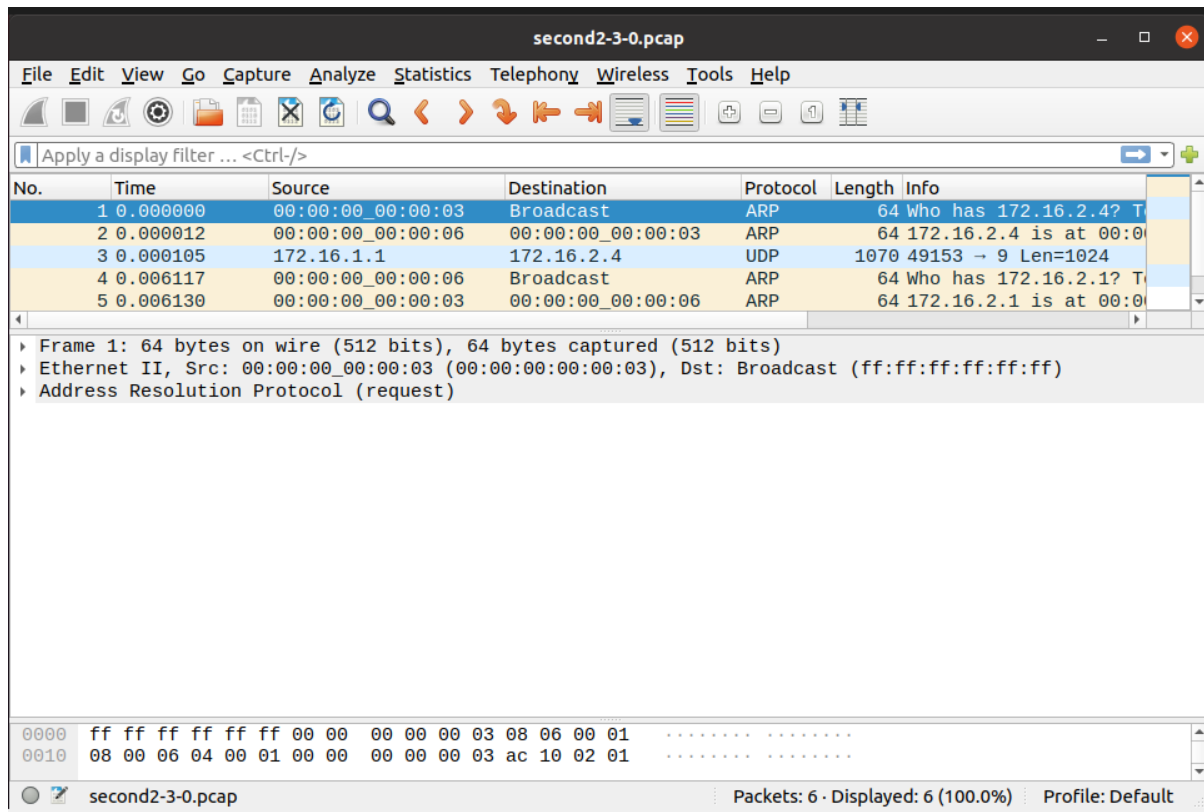
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.1.1	172.16.2.4	UDP	1054	49153 → 9 Len=1024
2	0.010235	172.16.2.4	172.16.1.1	UDP	1054	9 → 49153 Len=1024

Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
 Point-to-Point Protocol
 Internet Protocol Version 4, Src: 172.16.1.1, Dst: 172.16.2.4
 User Datagram Protocol, Src Port: 49153, Dst Port: 9
 Data (1024 bytes)

0000 00 21 45 00 04 1c 00 00 00 00 40 11 00 00 ac 10 ..!E.....@.....
 0010 01 01 ac 10 02 04 c0 01 00 09 04 08 00 00 00 00@.....

second2-1-0.pcap Packets: 2 · Displayed: 2 (100.0%) Profile: Default



The image shows a Wireshark packet capture window titled "second2-3-0.pcap". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help), a toolbar with various icons, and a display filter bar. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:03	Broadcast	ARP	64	Who has 172.16.2.4? T
2	0.000012	00:00:00_00:00:06	00:00:00_00:00:03	ARP	64	172.16.2.4 is at 00:0
3	0.000105	172.16.1.1	172.16.2.4	UDP	1070	49153 → 9 Len=1024
4	0.006117	00:00:00_00:00:06	Broadcast	ARP	64	Who has 172.16.2.1? T
5	0.006130	00:00:00_00:00:03	00:00:00_00:00:06	ARP	64	172.16.2.1 is at 00:0

The details pane for the selected packet (No. 1) shows:

- Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
- Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Address Resolution Protocol (request)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0000  ff ff ff ff ff ff 00 00 00 00 00 03 08 06 00 01  .....
0010  08 00 06 04 00 01 00 00 00 00 03 ac 10 02 01  .....
  
```

The status bar at the bottom indicates: second2-3-0.pcap, Packets: 6 · Displayed: 6 (100.0%), Profile: Default.

CONCLUSION:

From this practical, I have learned about bus topology simulation in ns3.