CS1111: Computer Networks and Distributed System

NS3 Project on Implementing CSMA Protocol in Bus Topology

PREPARED BY

Garv Baheti (2020BTechCSE031)

Shubham Sharma (2020BTechCSE072)

Siddharth Jangid (2020BTechCSE074)

FACULTY GUIDE

Dr. S. Taruna

Department of Computer Science and Engineering

Institute of Engineering and Technology

JK Lakshmipat University Jaipur

April 2023

CERTIFICATE

This is to certify that the project work entitled "**NS3 Project on Implementing CSMA Protocol in Bus Topology**" submitted by **Garv Baheti, Shubham Sharma, and Siddharth Jangid** towards the partial fulfilment of the requirements for the degree of **Bachelor of Technology in Computer Science Engineering** of JK Lakshmipat University, Jaipur is the record of work carried out by them under our supervision and guidance. In our opinion, the submitted work has reached the level required for being accepted for the final submission of the project.

- -

Date of Submission: April 2023

Dr. S. Taruna
Professor
Department of Computer Science Engineering

Institute of Engineering & Technology
JK Lakshmipat University, Jaipur

ACKNOWLEDGEMENTS

We have completed this project under the guidance and supervision of **Dr. S. Taruna** Associate Professor, IET, JK Lakshmipat University, I will be failed in my duty if I do not acknowledge the esteemed scholarly guidance, assistance, feedback, and knowledge I have received from them towards the fruitful and timely completion of this work.

We express our deepest thanks to **Dr Dheeraj Sanghi**, Vice Chancellor, JK Lakshmipat University, for their constant support, encouragement, and guidance.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents for their direct or indirect support during the entire course of this project.

Thanking You

Sincerely Yours,

Garv Baheti
Shubham Sharma
Siddharth Jangid

SUMMARY

This project implements a simulation of a CSMA network in a bus topology. The simulation includes one point-to-point link and multiple CSMA nodes connected to it. The code sets up the network topology, assigns IP addresses to the nodes, installs the Internet stack, and configures the CSMA channel and devices. The simulation also includes an echo server application running on the last CSMA node and an echo client application running on the first point-to-point node.

The code uses the NS-3 network simulator and includes several NS-3 module headers, such as core, network, internet, point-to-point, applications, csma, ipv4-global-routing-helper, and netanim. The code uses the CommandLine class to parse command-line arguments and sets the simulation time resolution to nanoseconds.

The simulation is visualized using the NetAnim module, which creates an animation file of the network topology. The code uses the AnimationInterface class to set the positions of the CSMA nodes in the animation.

Finally, the code runs the simulation using the NS-3 Simulator class, which executes the network simulation and outputs the simulation results to a pcap file. The simulation runs for 10 seconds, during which the echo client sends one packet to the echo server every second. The simulation also logs the output of the echo server and client applications if the verbose flag is set to true. The code populates the routing tables using the Ipv4GlobalRoutingHelper class before running the simulation.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 ABOUT NS3

NS-3 (Network Simulator 3) is an open-source discrete-event network simulator designed to study and evaluate the performance of communication networks. It is written in C++ and provides a set of libraries and modules to simulate different types of networks such as wired, wireless, and cellular networks. NS-3 is used extensively in academic and industrial research to evaluate network protocols, study network behaviour, and analyse network performance.

NS-3 provides a rich set of features and modules that can be used to simulate various network topologies and protocols. These modules include point-to-point, Wi-Fi, LTE, and Bluetooth modules, as well as modules for routing, traffic control, and mobility. Users can create custom network topologies, traffic models, and node behaviour using the provided modules.

NS-3 also provides a comprehensive set of statistical tools to analyse simulation results. These tools can be used to generate graphs, calculate metrics such as throughput, delay, and packet loss, and perform statistical analysis on simulation results.

NS-3 is a powerful tool for network simulation and has been used in many research areas such as ad-hoc networks, vehicular networks, and IoT networks. Its popularity stems from its open-source nature, comprehensive set of features, and ease of use.

Overall, NS-3 is a valuable tool for network researchers and engineers alike, as it allows them to study and evaluate network behaviour in a simulated environment, without the need for expensive and time-consuming real-world testing.

1.2 About NetAnim

NetAnim is a network animator and visualizer that is used with the NS-3 network simulator. It is a graphical user interface tool that allows users to visualize the simulation of network traffic and behaviour. NetAnim is written in C++ and utilizes Qt, a cross-platform application framework, to provide a user-friendly interface.

NetAnim provides a variety of features to help users visualize and analyse network simulations. Users can view node and link animation in real-time, as well as view network traffic and packet flow. NetAnim also provides tools for network visualization, such as the ability to change the layout of the network topology, adjust the colours of nodes and links, and add labels and annotations.

NetAnim supports a wide range of animation types, including point-to-point, Wi-Fi, LTE, and Bluetooth. It also supports different traffic types, such as constant bit rate (CBR), variable bit rate (VBR), and on-off traffic.

One of the main benefits of using NetAnim is the ability to quickly and easily analyse simulation results. Users can generate graphs and charts to analyse network metrics such as throughput, packet loss, and delay. NetAnim also provides a comprehensive statistics viewer that allows users to view detailed statistics about the simulation results.

Overall, NetAnim is a powerful tool for visualizing and analyzing network simulations. Its user-friendly interface and comprehensive set of features make it an essential tool for network researchers and engineers. By providing real-time visualization of network behaviour and the ability to analyse simulation results, NetAnim helps users better understand network performance and behaviour.

1.3 About CSMA (Carrier Sense Multiple Access)

CSMA (Carrier Sense Multiple Access) is a network access method used in wired and wireless networks. It is a contention-based protocol that is used to avoid data collisions when multiple devices are trying to access the network simultaneously. CSMA works by detecting the presence of a carrier signal on the network before transmitting data.

In CSMA, each device on the network listens for a carrier signal before transmitting data. If a carrier signal is detected, the device waits for the signal to stop before transmitting. This is known as carrier sense. After the carrier signal stops, the device waits for a random amount of time before attempting to transmit. This helps to reduce the likelihood of multiple devices trying to transmit at the same time.

If multiple devices transmit at the same time, a collision occurs, and the data is lost. When a collision occurs, the devices involved wait for a random amount of time before attempting to transmit again. This random backoff time helps to reduce the likelihood of another collision.

CSMA is used in a variety of network technologies, including Ethernet and Wi-Fi. In Ethernet networks, CSMA/CD (Carrier Sense Multiple Access with Collision Detection) is used, where devices listen for a carrier signal and also monitor the network for collisions. If a collision is detected, the devices involved stop transmitting and wait for a random amount of time before attempting to transmit again.

In Wi-Fi networks, CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is used, where devices listen for a carrier signal and wait for a clear channel before transmitting. If a clear channel is not detected, the device waits for a random amount of time before attempting to transmit again.

Overall, CSMA is an effective way to manage access to a shared network medium and avoid data collisions. It is widely used in wired and wireless networks to ensure efficient and reliable data transmission.

1.4 About Point to Point Network

Point-to-point (P2P) is a type of network architecture in which two devices are directly connected to each other to exchange data. In a P2P network, there is no central server, and each device in the network can communicate directly with another device.

Point-to-point connections can be implemented using a variety of technologies, including wired and wireless connections. In a wired P2P network, devices can be connected directly using cables, while in a wireless P2P network, devices can communicate with each other over a shared wireless channel.

One of the primary advantages of a P2P network is its scalability. As each device in the network can communicate directly with another device, additional devices can be added to the network without the need for a centralized server. This makes P2P networks ideal for applications that require large-scale connectivity, such as file sharing, media streaming, and online gaming.

Another advantage of P2P networks is their resiliency. Because there is no central server, P2P networks are less susceptible to single points of failure. If one device in the network fails, other devices can continue to communicate with each other, ensuring that the network remains functional.

However, P2P networks can also have some drawbacks. One issue with P2P networks is security. As devices in the network communicate directly with each other, it can be challenging to ensure the security of data transmitted between devices. Additionally, P2P networks can be vulnerable to attacks, such as distributed denial of service (DDoS) attacks, which can overwhelm the network with traffic.

Overall, P2P networks are a useful type of network architecture for applications that require large-scale connectivity and resiliency. By allowing devices to communicate directly with each other, P2P networks can provide efficient and reliable communication without the need for a centralized server.

1.5 IMPLEMENTING CSMA IN BUS TOPOLOGY

Implementing CSMA (Carrier Sense Multiple Access) in a Bus topology involves ensuring that multiple devices can access the shared communication channel without causing collisions. In a Bus topology, all devices are connected to a single communication line, and any device can transmit data at any time. Here are the steps to implement CSMA in a Bus topology:

1. Carrier Sense: Each device listens to the communication line to detect whether it is busy or idle. If the communication line is idle, the device can start transmitting data. If the line is busy, the device must wait until the line is free.
2. Collision Detection: Each device also listens to the communication line while transmitting data to detect whether a collision has occurred. A collision occurs when two or more devices transmit data at the same time, causing interference on the communication line. If a collision is detected, all devices that were transmitting data stop transmitting and wait for a random amount of time before attempting to transmit again.
3. Backoff Algorithm: To avoid repeated collisions, each device uses a backoff algorithm that randomizes the amount of time a device waits before attempting to transmit again. This algorithm ensures that multiple devices do not try to transmit at the same time after a collision.
4. Jam Signal: If a collision is detected, each device sends a jam signal to indicate that a collision has occurred. The jam signal is sent for a specified duration to ensure that all devices on the network receive it.
5. Priority Access: In some cases, certain devices may have higher priority to access the communication line than others. To ensure that these devices have priority access, they can be given a shorter wait time before attempting to transmit data.

Overall, implementing CSMA in a Bus topology is a useful way to ensure that multiple devices can access the communication line without causing collisions. By using carrier sense, collision detection, a backoff algorithm, and priority access, devices can efficiently share the communication line and avoid interference.

1.6 About UDP Echo Server & Client

1.6.1 UDP ECHO SERVER

A UDP (User Datagram Protocol) echo server is a type of server that listens for incoming UDP packets on a specific port and then echoes back the received data to the client that sent it. This type of server is commonly used for testing purposes, as it provides a simple way to verify that network connectivity is working correctly.

The UDP echo server operates by listening for incoming packets on a designated port, typically port 7 or 9. When a packet is received, the server simply echoes back the received data to the sender without any modification. The server does not maintain any state information about the connection, and each packet is treated independently.

To implement a UDP echo server, the following steps can be taken:

- Create a socket: The server creates a UDP socket and binds it to a specific port. This allows the server to listen for incoming packets on that port.
- Listen for incoming packets: The server waits for incoming packets to arrive on the socket. When a packet is received, the server reads the data from the packet and stores it in a buffer.
- Echo back the data: After the data has been received, the server simply echoes back the data to the client by sending a new UDP packet with the same data back to the client's IP address and port number.
- Repeat: The server continues to listen for incoming packets and echo back the data to the client until it is terminated.

It is important to note that the UDP echo server is a very basic type of server and does not provide any reliability or error handling mechanisms. As UDP is an unreliable protocol, packets can be lost or delivered out of order, and the server does not have any way of detecting or correcting these errors.

1.6.2 UDP ECHO CLIENT

A UDP (User Datagram Protocol) echo client is a type of client application that sends UDP packets to a UDP echo server and waits for the server to echo back the same data. This type of client is commonly used for testing purposes, as it provides a simple way to verify that network connectivity is working correctly.

The UDP echo client operates by creating a UDP packet containing some data and sending it to a UDP echo server on a specific port, typically port 7 or 9. The client then waits for the server to echo back the same data. Once the data has been received, the client can verify that the data matches the original data sent and that the network connection is working correctly.

To implement a UDP echo client, the following steps can be taken:

1. Create a socket: The client creates a UDP socket and binds it to a specific port. This allows the client to send UDP packets from that port.
2. Send data to the server: The client creates a UDP packet containing some data and sends it to the server's IP address and port number.
3. Wait for data: After sending the packet, the client waits for a response from the server. When the response arrives, the client reads the data from the packet and stores it in a buffer.
4. Verify data: The client compares the data received from the server with the original data sent. If the data matches, the client can assume that the network connection is working correctly.
5. Repeat: The client can continue to send data to the server and wait for a response until it is terminated.

It is important to note that the UDP echo client is a very basic type of client and does not provide any reliability or error handling mechanisms. As UDP is an unreliable protocol, packets can be lost or delivered out of order, and the client does not have any way of detecting or correcting these errors.

In summary, a UDP echo client is a simple type of client application that sends UDP packets to a UDP echo server and waits for the server to echo back the same data. While it is not suitable for production use, it is useful for testing network connectivity and troubleshooting network issues.

1.7 BUS TOPOLOGY

A bus topology is a network topology in which all the nodes are connected to a single communication channel, called a bus. In a bus topology, all the nodes share the same communication medium, and data is transmitted by broadcasting it on the bus. Each node on the bus receives the broadcast signal and checks if the data is intended for it. If so, it processes the data; otherwise, it ignores it.

The advantages of a bus topology include simplicity, low cost, and ease of installation. Bus topologies are easy to set up and require only a single cable to connect all the nodes. This makes them ideal for small networks and for applications where cost is a significant factor.

However, bus topologies also have some limitations. First, they can be prone to signal interference and noise, which can lead to data errors and network downtime. Second, as the number of nodes on the bus increases, the network performance can decrease, leading to network congestion and slow data transmission speeds.

Despite these limitations, bus topologies are still used in some industrial and automotive applications, where their simplicity and low cost make them an attractive option. They are also commonly used in legacy computer networks and in some educational settings as a way to introduce students to network topologies and basic network concepts.

CHAPTER 2: NS3 CODE WITH EXPLANATION

2.1 STEP BY STEP EXPLANATION

The code implements a network topology consisting of a point-to-point (P2P) link and a CSMA (Carrier Sense Multiple Access) network with UDP echo client and server applications. The code is written in C++ and uses the ns-3 network simulation software.

The code starts by including the required ns-3 modules, such as the core module, network module, internet module, point-to-point module, CSMA module, applications module, IPv4 global routing helper, and netanim module.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/csma-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;
```

The **NS_LOG_COMPONENT_DEFINE** macro defines a logging component for the code.

```
NS_LOG_COMPONENT_DEFINE ("P2P-CSMA");
```

The **main** function is the entry point of the code. It starts by defining some variables and parsing command-line arguments.

```
int main (int argc, char *argv[]){

       bool verbose = true;
       uint32_t nCsma = 19;

       CommandLine cmd;
       cmd.AddValue("nCsna","Number of CSMA nodes", nCsma);
       cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);
       cmd.Parse (argc, argv);
       Time::SetResolution(Time::NS);

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

The **NodeContainer** class is used to create the nodes of the network topology. The code creates two nodes for the P2P link and **nCsma + 1** nodes for the CSMA network.

```
NodeContainer p2pNodes;
 p2pNodes.Create(2);

 NodeContainer csmaNodes;
 csmaNodes.Add(p2pNodes.Get(1));
 csmaNodes.Create(nCsma);
```

The **PointToPointHelper** class is used to configure the P2P link with a data rate of 100 Mbps and a delay of 2 ms.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate",StringValue("100Mbps"));
pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));
```

The **NetDeviceContainer** class is used to store the P2P link devices and the CSMA network devices.

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

CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", StringValue("2ms"));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install(csmaNodes);
```

The **InternetStackHelper** class is used to install the internet stack on the nodes. The P2P node is given a higher priority, so it is installed first.

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

The **Ipv4AddressHelper** class is used to assign IP addresses to the interfaces of the point-to-point and CSMA/CD links. It assigns the IP address **192.168.1.x** to the point-to-point link and **192.168.2.x** to the CSMA/CD network, where **x** is the host number.

```
Ipv4AddressHelper address;
address.SetBase("192.168.1.0","255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces = address.Assign(p2pDevices);

address.SetBase("192.168.2.0","255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces = address.Assign(csmaDevices);
```

The code then creates a UDP echo server and client using the **UdpEchoServerHelper** and **UdpEchoClientHelper** classes, respectively. The echo server is installed on the last node in the **csmaNodes** container and the echo client is installed on the first node of **p2pNodes**. The echo client sends a single packet of size 1024 bytes to the echo server every second, starting from 2 seconds into the simulation and ending at 10 seconds.

```
UdpEchoServerHelper echoServer(4445);

ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma),4445);
echoClient.SetAttribute("MaxPackets",UintegerValue(1));
echoClient.SetAttribute("Interval",TimeValue(Seconds(1.0)));
echoClient.SetAttribute("PacketSize",UintegerValue(1024));
```

Here, we are creating an **ApplicationContainer** named **clientApps** to hold the **UdpEchoClient** application that will send data to the **UdpEchoServer** application running on the last CSMA node.

echoClient.Install(p2pNodes.Get(0)) installs the **UdpEchoClient** application on the first point-to-point node (**p2pNodes.Get(0)**).

clientApps.Start(Seconds(2.0)) starts the **UdpEchoClient** application after 2.0 seconds.

clientApps.Stop(Seconds(10.0)) stops the **UdpEchoClient** application after 10.0 seconds.

```
ApplicationContainer clientApps = echoClient.Install(p2pNodes.Get(0));
clientApps.Start(Seconds(2.0));
clientApps.Stop(Seconds(10.0));
```

The next section of code creates an animation interface object named **anim** for network visualization using NetAnim. The argument "CSMA.xml" is the name of the output file where the animation data will be written.

The **for** loop sets the position of each CSMA node along the x-axis. The **xPos** variable is calculated as the index of the node multiplied by 50.0. **yPos** is set to 0.0 since we are only visualizing the network in two dimensions. The **SetConstantPosition** method of the **AnimationInterface** object is used to set the position of each CSMA node using the **csmaNodes.Get(i)** method.

```
    AnimationInterface anim ("CSMA.xml");

    for (uint32_t i = 0; i < nCsma + 1; i++) {
        double xPos = i * 50.0;
        double yPos = 0.0;
        anim.SetConstantPosition(csmaNodes.Get(i), xPos, yPos);
    }
```

This line of code is used to populate the routing tables of the nodes in the network.

```
    Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

This line of code enables packet capture (pcap) for all point-to-point links in the simulation and writes the captured packets to files with the prefix "p2p-csma".

```
    pointToPoint.EnablePcapAll("p2p-csma");
```

Finally, these two lines start the simulation and run it until it completes. Once the simulation is complete, the **Destroy()** method is called on the **Simulator** object to release any resources used during the simulation.

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

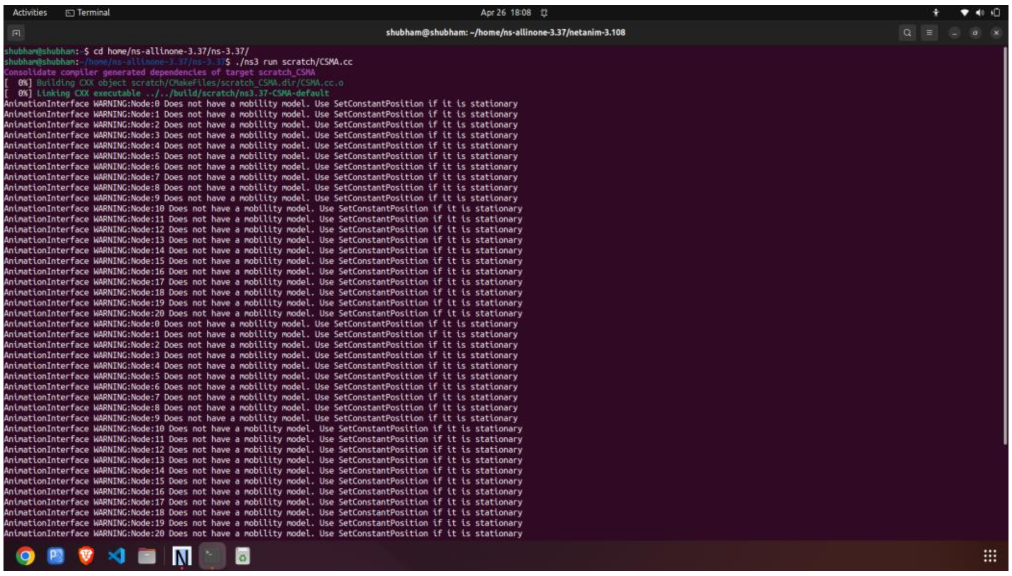
2.2 Final Code

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/csma-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"


using namespace ns3;


NS_LOG_COMPONENT_DEFINE ("P2P-CSMA");

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

       bool verbose = true;
       uint32_t nCsma = 19;

       CommandLine cmd;
       cmd.AddValue("nCsna","Number of CSMA nodes", nCsma);
       cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);
       cmd.Parse (argc, argv);
       Time::SetResolution(Time::NS);

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

       NodeContainer p2pNodes;
       p2pNodes.Create(2);

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


       PointToPointHelper pointToPoint;
       pointToPoint.SetDeviceAttribute("DataRate",StringValue("100Mbps"));
       pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));

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

       CsmaHelper csma;
       csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
       csma.SetChannelAttribute("Delay", StringValue("2ms"));

       NetDeviceContainer csmaDevices;
       csmaDevices = csma.Install(csmaNodes);

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

```
        stack.Install(csmaNodes);

        Ipv4AddressHelper address;
        address.SetBase("192.168.1.0","255.255.255.0");
        Ipv4InterfaceContainer p2pInterfaces = address.Assign(p2pDevices);

        address.SetBase("192.168.2.0","255.255.255.0");
        Ipv4InterfaceContainer csmaInterfaces = address.Assign(csmaDevices);

        UdpEchoServerHelper echoServer(4445);

        ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));
        serverApps.Start(Seconds(1.0));
        serverApps.Stop(Seconds(10.0));
        UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma),4445);
        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));

        AnimationInterface anim ("CSMA.xml");

        for (uint32_t i = 0; i < nCsma + 1; i++) {
                double xPos = i * 50.0;
                double yPos = 0.0;
                anim.SetConstantPosition(csmaNodes.Get(i), xPos, yPos);
        }

        Ipv4GlobalRoutingHelper::PopulateRoutingTables();

        pointToPoint.EnablePcapAll("p2p-csma");

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

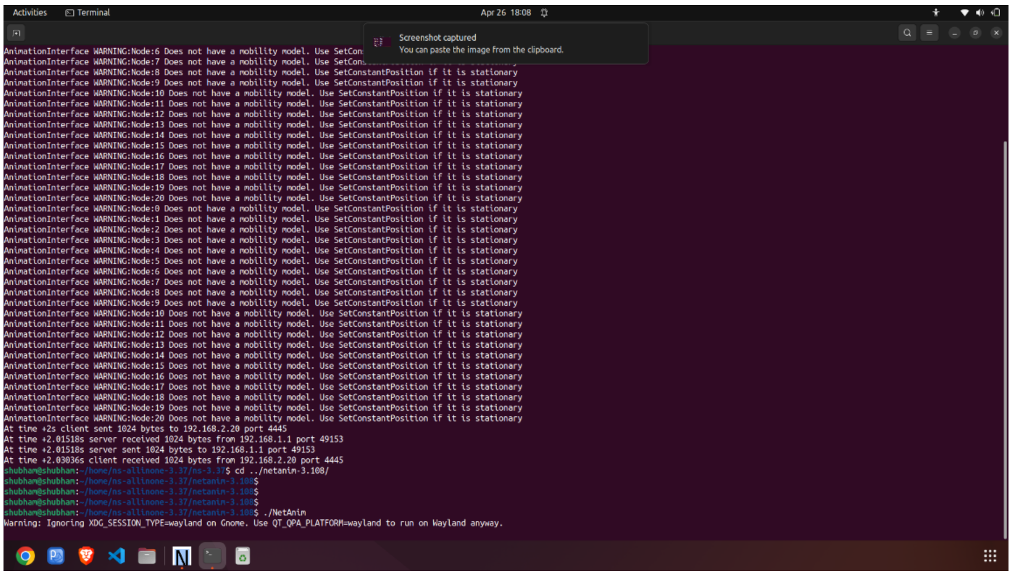
2.3 TO RUN THE CODE

1. Save the code in scratch folder of ns3 with the file name of **CSMA.cc**
2. Open the Terminal and navigate to the ns3 directory.
3. Run the code using **./ns3 run scratch/CSMA.cc**



```
shubham@shubham:~$ cd home/ns-allinone-3.37/ns-3.37/
shubham@shubham:~/home/ns-allinone-3.37/ns-3.37$ ./ns3 run scratch/CSMA.cc
Consolidate compiler generated dependencies of target scratch_CSMA
[  0%] Building CXX object scratch/CMakeFiles/scratch_CSMA.dir/CSMA.cc.o
[  0%] Linking CXX executable ../../build/scratch/ns3.37-CSMA-default
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:6 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:8 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:9 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:10 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:11 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:12 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:13 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:14 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:15 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:16 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:17 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:18 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:19 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:20 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:6 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:8 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:9 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:10 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:11 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:12 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:13 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:14 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:15 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:16 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:17 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:18 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:19 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:20 Does not have a mobility model. Use SetConstantPosition if it is stationary
```

4. This will run the program and generate a file with the name of **CSMA.xml** that will be used to view visual simulation of the program.
5. Navigate to the NetAnim Directory and run the command **./NetAnim**

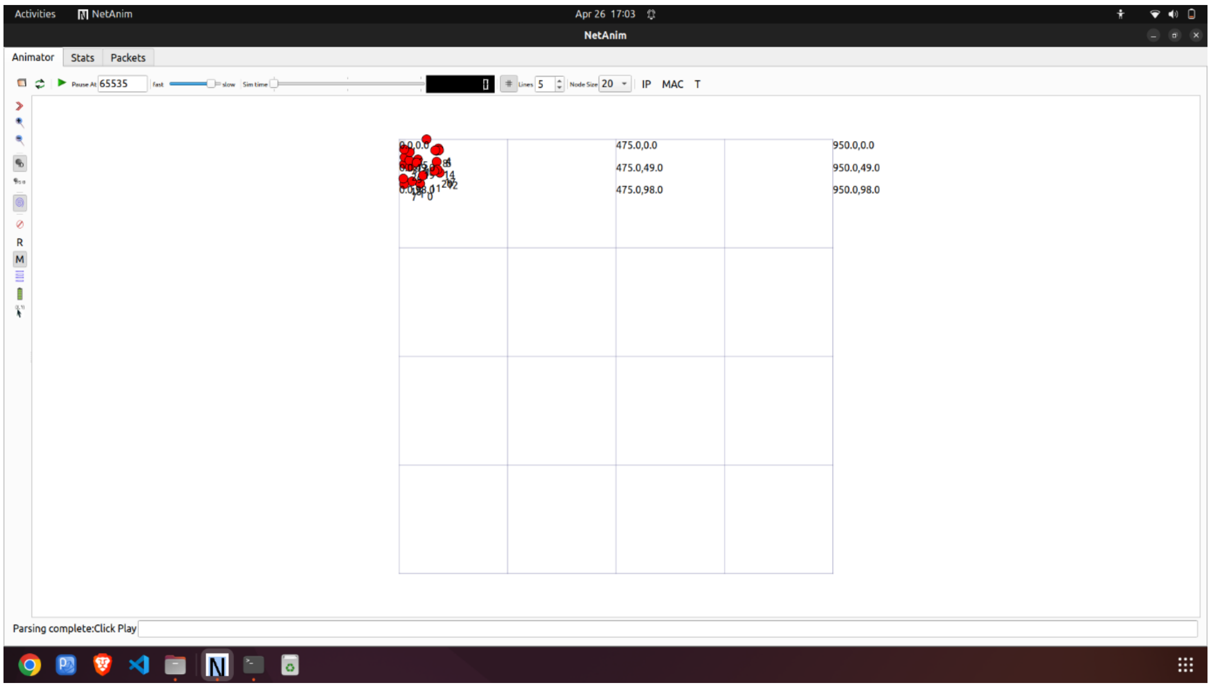


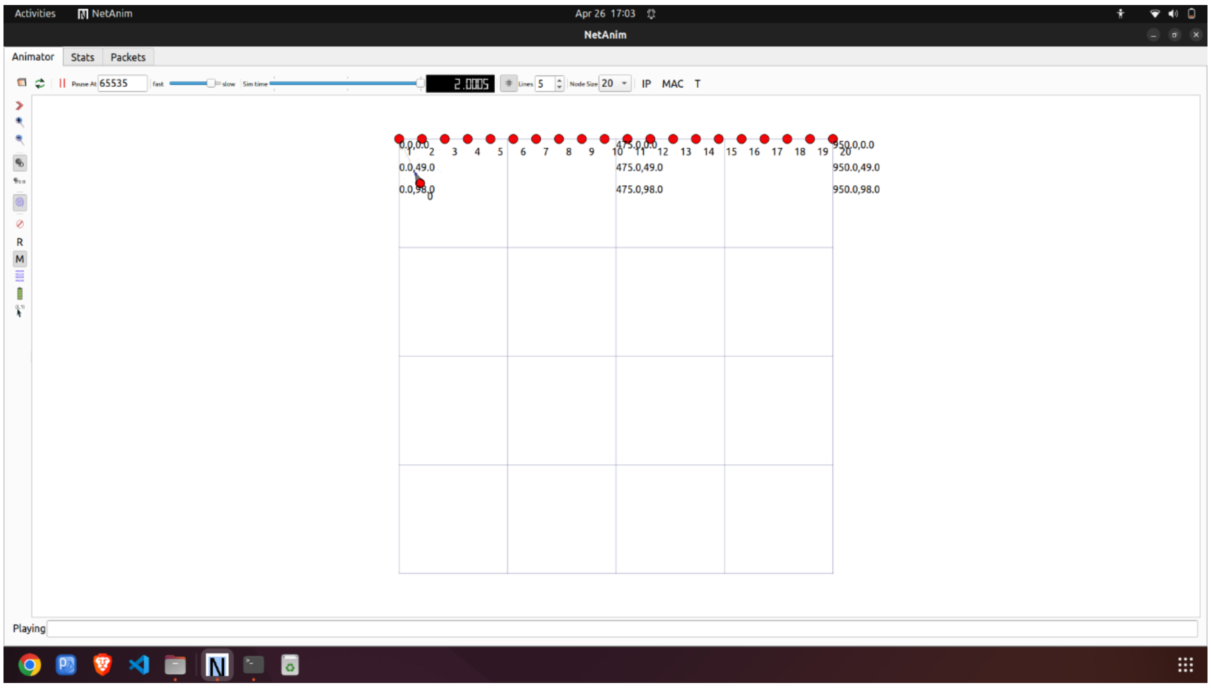
```
AnimationInterface WARNING:Node:6 Does not have a mobility model. Use SetCon
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetCon
AnimationInterface WARNING:Node:8 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:9 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:10 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:11 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:12 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:13 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:14 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:15 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:16 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:17 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:18 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:19 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:20 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:6 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:8 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:9 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:10 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:11 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:12 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:13 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:14 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:15 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:16 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:17 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:18 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:19 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:20 Does not have a mobility model. Use SetConstantPosition if it is stationary
At time +2s client sent 1024 bytes to 192.168.2.20 port 4445
At time +2.01518s server received 1024 bytes from 192.168.1.1 port 49153
At time +2.01518s server sent 1024 bytes to 192.168.1.1 port 49153
At time +2.03036s client received 1024 bytes from 192.168.2.20 port 4445
shubham@shubham:~/home/ns-allinone-3.37/ns-3.37$ cd ../netanim-3.108/
shubham@shubham:~/home/ns-allinone-3.37/netanim-3.108$
shubham@shubham:~/home/ns-allinone-3.37/netanim-3.108$
shubham@shubham:~/home/ns-allinone-3.37/netanim-3.108$
shubham@shubham:~/home/ns-allinone-3.37/netanim-3.108$ ./NetAnim
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
```

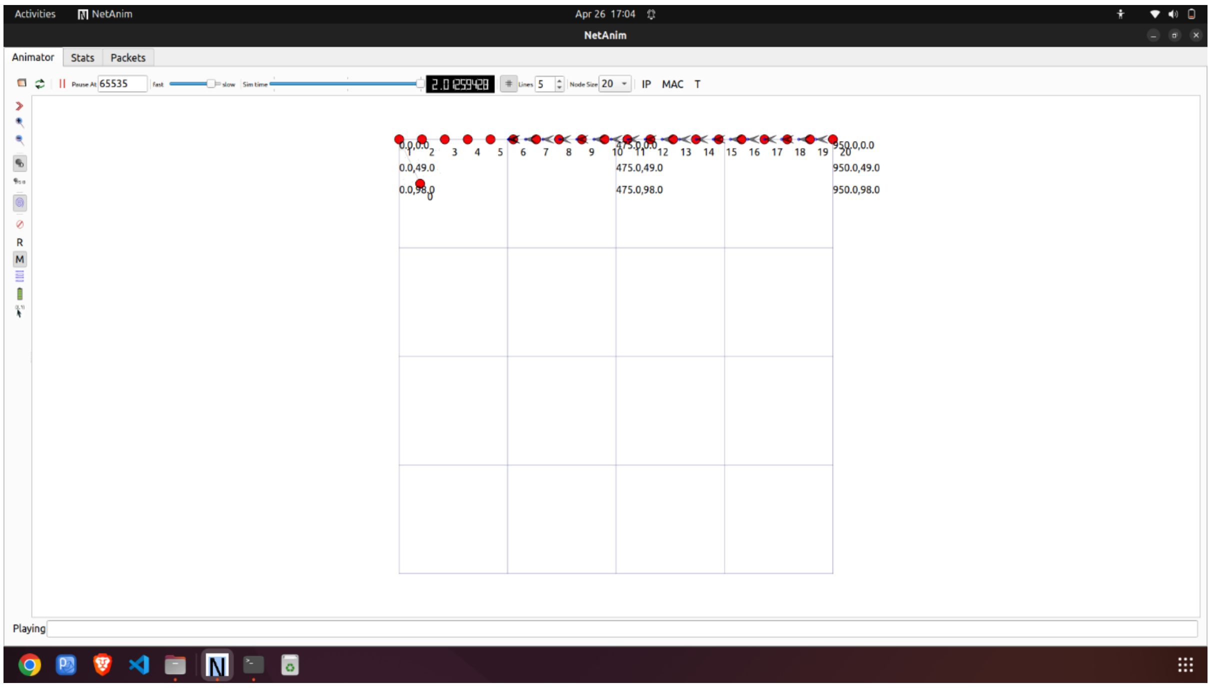
6. This will open the NetAnim and open your CSMA.xml file
7. Now run the simulation.

CHAPTER 3: RESULT

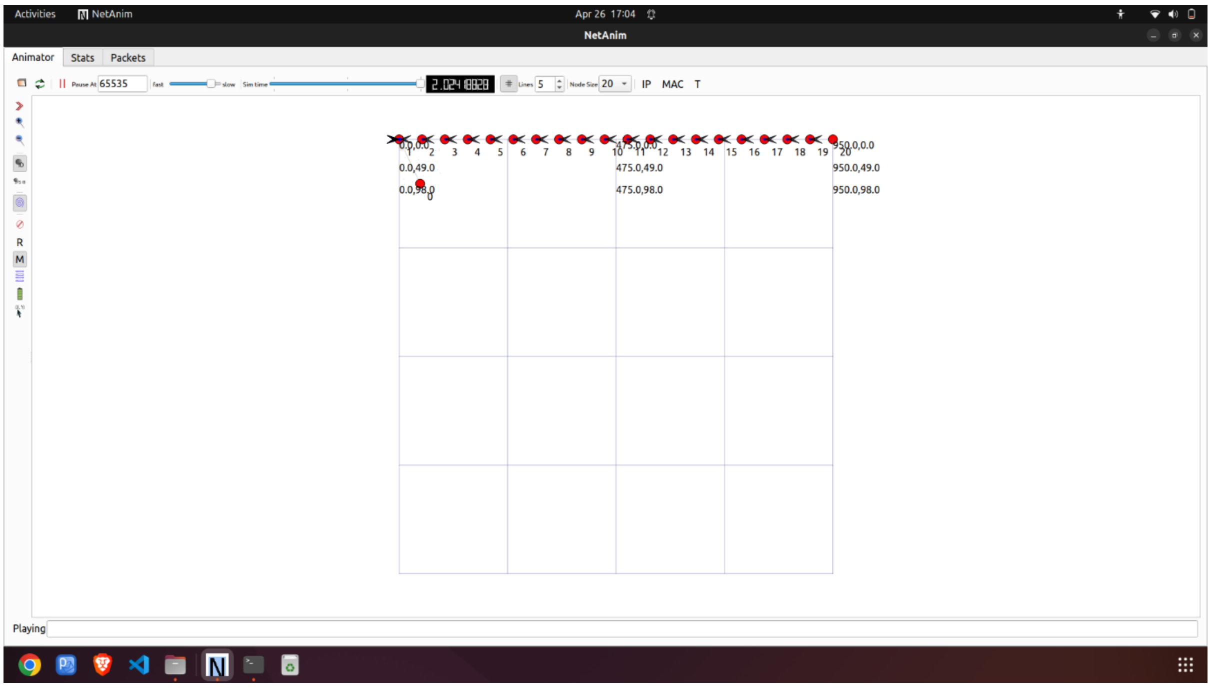
3.1 VISUAL SIMULATION ON NETANIM



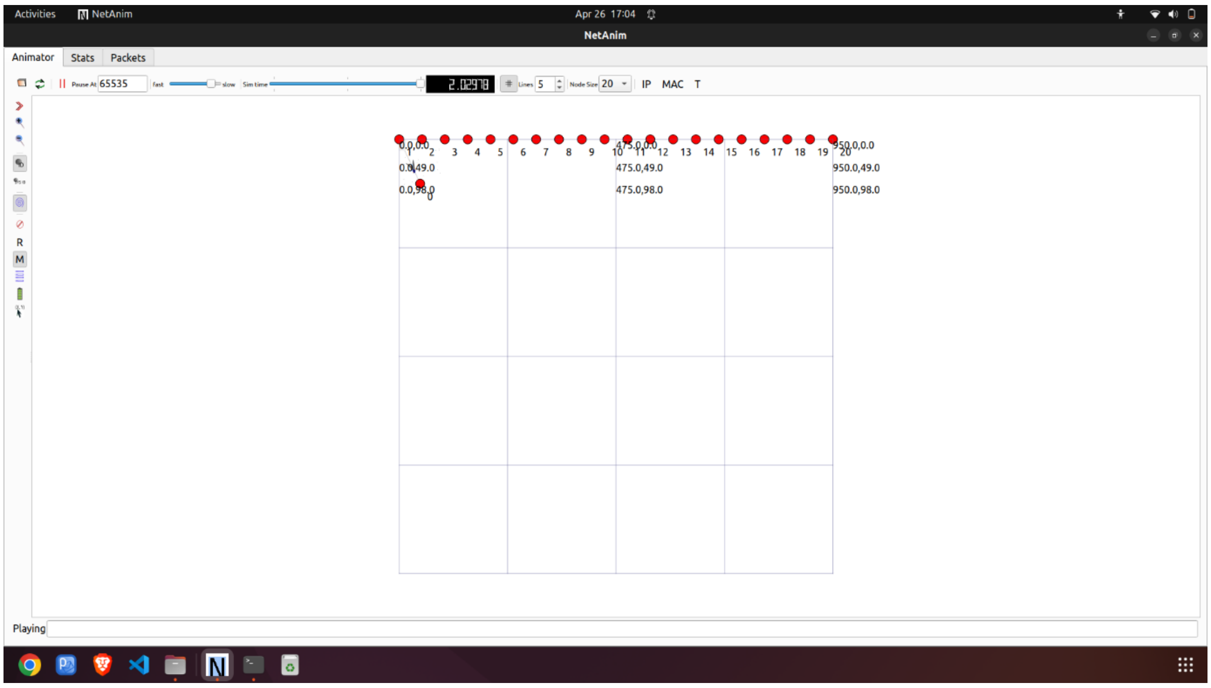




Activities
NetAnim
Apr 26 17:04
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
2.0125942
Lines 5
Node Size 20
IP
MAC
T
0.0,0.0
0.0,49.0
0.0,98.0
475.0,0.0
475.0,49.0
475.0,98.0
950.0,0.0
950.0,49.0
950.0,98.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Playing

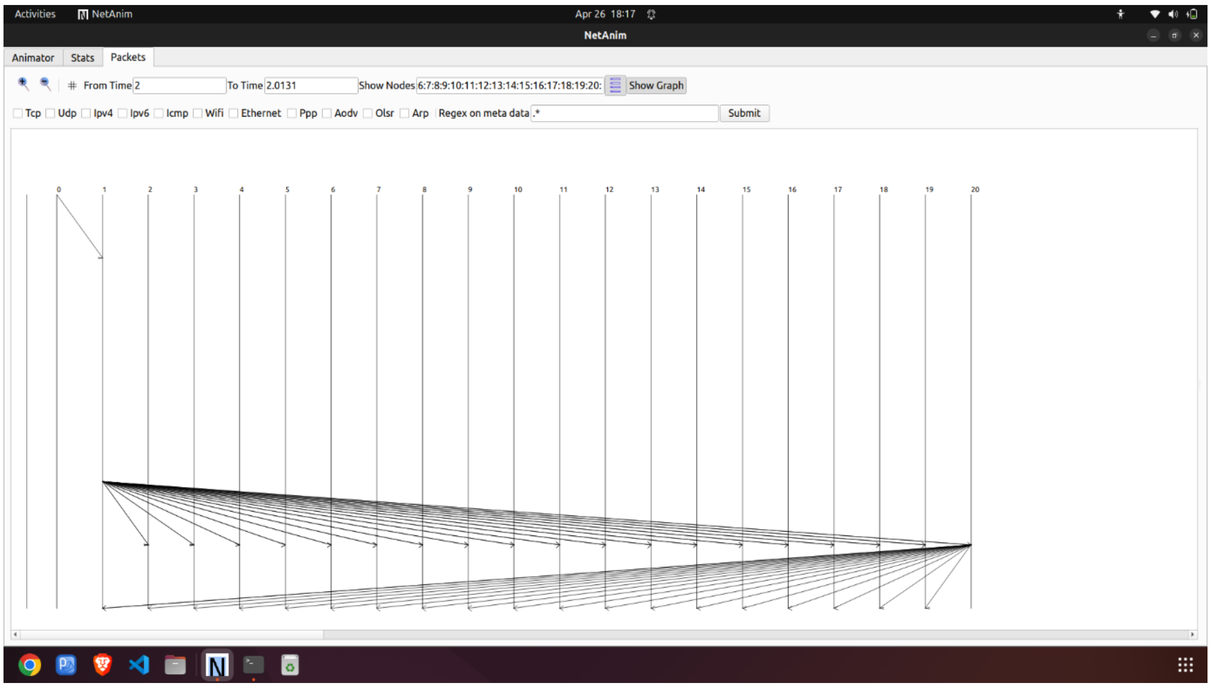


Activities
NetAnim
Apr 26 17:04
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
2.0241882
Lines 5
Node Size 20
IP
MAC
T
0.0,0.0
0.0,49.0
0.0,98.0
475.0,0.0
475.0,49.0
475.0,98.0
950.0,0.0
950.0,49.0
950.0,98.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Playing



Activities
NetAnim
Apr 26 17:04
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
2.02978
Lines 5
Node Size 20
IP
MAC
T
0.0,0.0
475.0,0.0
950.0,0.0
0.0,49.0
475.0,49.0
950.0,49.0
0.0,98.0
475.0,98.0
950.0,98.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Playing

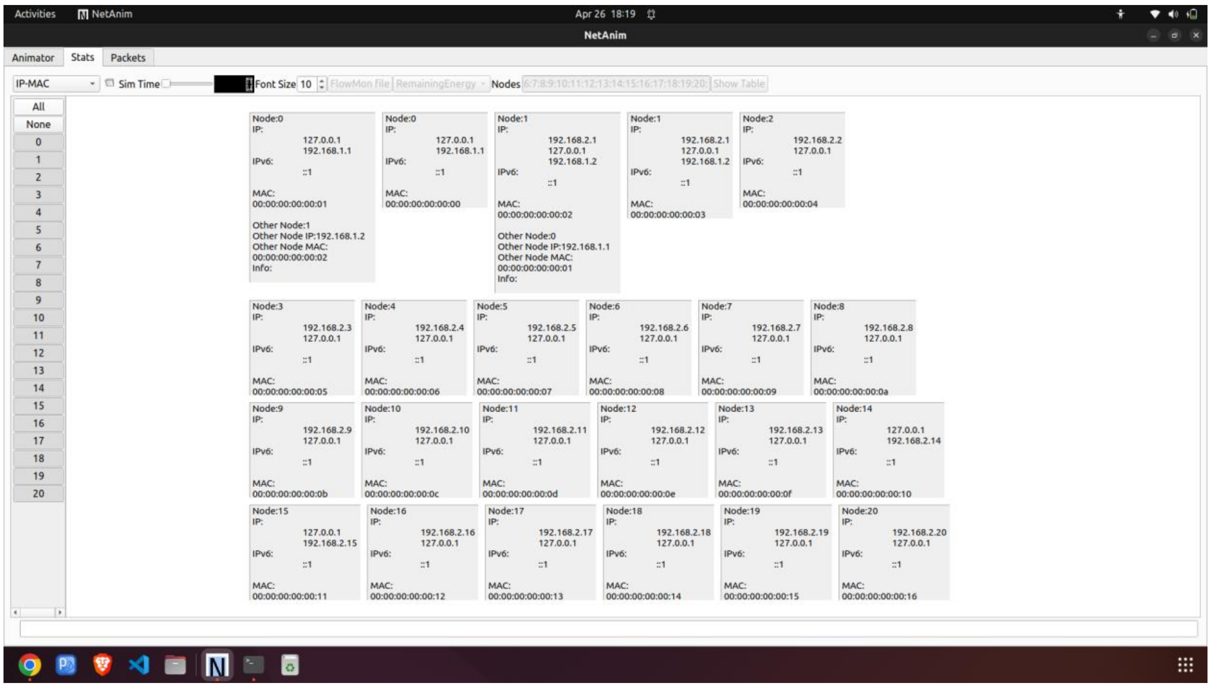
3.2 GRAPH



From Time 2 To Time 2.0131 Show Nodes 6:7:8:9:10:11:12:13:14:15:16:17:18:19:20:

Regex on meta data .*

3.3 STATS



Font Size 10 Nodes 6:7:8:9:10:11:12:13:14:15:16:17:18:19:20:

```
Node:0
IP:
          127.0.0.1
          192.168.1.1
IPv6:
          ::1

MAC:
00:00:00:00:00:01

Other Node:1
Other Node IP:192.168.1.2
Other Node MAC:
00:00:00:00:00:02
Info:

Node:0
IP:
          127.0.0.1
          192.168.1.1
IPv6:
          ::1

MAC:
00:00:00:00:00:00

Node:1
IP:
          192.168.2.1
          127.0.0.1
          192.168.1.2
IPv6:
          ::1

MAC:
00:00:00:00:00:02

Other Node:0
Other Node IP:192.168.1.1
Other Node MAC:
00:00:00:00:00:01
Info:

Node:1
IP:
          192.168.2.1
          127.0.0.1
          192.168.1.2
IPv6:
          ::1

MAC:
00:00:00:00:00:03

Node:2
IP:
          192.168.2.2
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:04

Node:3
IP:
          192.168.2.3
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:05

Node:4
IP:
          192.168.2.4
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:06

Node:5
IP:
          192.168.2.5
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:07

Node:6
IP:
          192.168.2.6
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:08

Node:7
IP:
          192.168.2.7
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:09

Node:8
IP:
          192.168.2.8
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:0a

Node:9
IP:
          192.168.2.9
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:0b

Node:10
IP:
          192.168.2.10
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:0c

Node:11
IP:
          192.168.2.11
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:0d

Node:12
IP:
          192.168.2.12
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:0e

Node:13
IP:
          192.168.2.13
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:0f

Node:14
IP:
          127.0.0.1
          192.168.2.14
IPv6:
          ::1

MAC:
00:00:00:00:00:10

Node:15
IP:
          127.0.0.1
          192.168.2.15
IPv6:
          ::1

MAC:
00:00:00:00:00:11

Node:16
IP:
          192.168.2.16
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:12

Node:17
IP:
          192.168.2.17
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:13

Node:18
IP:
          192.168.2.18
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:14

Node:19
IP:
          192.168.2.19
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:15

Node:20
IP:
          192.168.2.20
          127.0.0.1
IPv6:
          ::1

MAC:
00:00:00:00:00:16
```

3.4 TABLE

From Id	To Id	Tx
0	1	2
1	2	2.00908
1	2	2.00908
1	3	2.00908
1	3	2.00908
1	4	2.00908
1	4	2.00908
1	5	2.00908
1	5	2.00908
1	6	2.00908
1	6	2.00908
1	7	2.00908
1	7	2.00908
1	8	2.00908
1	8	2.00908
1	9	2.00908
1	9	2.00908
1	10	2.00908
1	10	2.00908
1	11	2.00908
1	11	2.00908
1	12	2.00908
1	12	2.00908
1	13	2.00908
1	13	2.00908
1	14	2.00908
1	14	2.00908
1	15	2.00908
1	15	2.00908
1	16	2.00908
1	16	2.00908
1	17	2.00908
1	17	2.00908
1	18	2.00908
1	18	2.00908
1	19	2.00908
1	19	2.00908
1	20	2.00908
1	20	2.00908
20	1	2.01109
20	1	2.01109
20	2	2.01109
20	3	2.01109
20	4	2.01109
20	5	2.01109
20	6	2.01109
20	7	2.01109
20	8	2.01109
20	9	2.01109
20	10	2.01109
20	11	2.01109
20	12	2.01109
20	13	2.01109
20	14	2.01109
20	15	2.01109
20	16	2.01109
20	17	2.01109
20	18	2.01109
20	19	2.01109

CONCLUSION

In conclusion, the code implements a simulation of a CSMA network in a bus topology using the NS-3 network simulator. The code sets up the network topology, assigns IP addresses to the nodes, installs the Internet stack, and configures the CSMA channel and devices. The simulation includes an echo server application running on the last CSMA node and an echo client application running on the first point-to-point node. The simulation is visualized using the NetAnim module, and the simulation results are output to a pcap file. The code populates the routing tables before running the simulation and runs the simulation for 10 seconds, during which the echo client sends one packet to the echo server every second.

Overall, the provided code provides a good starting point for simulating a CSMA network in a bus topology using the NS-3 network simulator. However, further modifications and additions may be required to adapt the code to specific research or industrial requirements. Additionally, the results of the simulation should be analysed and validated to ensure their accuracy and relevance to the intended use case.

LEARNINGS

From this project, there are several key learnings:

1. **NS-3 network simulator:** This project provided an introduction to the NS-3 network simulator, its architecture, and its capabilities. NS-3 is a powerful tool for simulating complex network topologies and protocols, and it can be used for research, testing, and development purposes.
2. **CSMA:** This project provided an overview of the Carrier Sense Multiple Access (CSMA) protocol, which is a widely used protocol for sharing a communication channel among multiple nodes. The code implemented a CSMA network in a bus topology, which is a common configuration in industrial and automotive applications.
3. **Network topology:** The project provided an introduction to network topologies and how they affect network performance and efficiency. The bus topology used in this project is a simple topology that allows easy expansion and modification of the network.
4. **Application layer:** The project included the implementation of echo server and client applications, which are common applications used for testing and measuring network performance. The echo server receives packets and sends them back to the sender, allowing the sender to measure the round-trip time and other performance metrics.
5. **NetAnim:** The project used the NetAnim module to visualize the simulation and monitor the behavior of the network. NetAnim is a powerful tool for visualizing network topologies and analyzing network performance.

Overall, this project provided a hands-on learning experience in network simulation, network topologies, and CSMA protocols, and it introduced useful tools such as the NS-3 network simulator and the NetAnim module.

REFERENCES

- NS-3 Network Simulator: https://www.nsnam.org/
- NS-3 Documentation: https://www.nsnam.org/docs/
- NS-3 Tutorial: https://www.nsnam.org/tutorial/
- Bus Topology Overview: https://www.geeksforgeeks.org/bus-topology-in-network-communication/
- Bus Topology Advantages and Disadvantages: https://www.fomsn.com/networking/topology/bus-topology-advantages-disadvantages/
- Introduction to CSMA/CD: https://www.geeksforgeeks.org/computer-network-csma-cd/
- CSMA/CD Protocol: https://www.tutorialspoint.com/csmacd-protocol-in-computer-networks