

NETWORK PROTOCOL SIMULATION

MINI PROJECT REPORT

By

RA2211003010089 SREYA SUSAN ROY

RA2211003010098 PARVATHY ULLAS

RA2211003010126 AARTHI N

Under the guidance of

Dr C JOTHI KUMAR

ASSISTANT PROFESSOR

In partial fulfilment for the Course

of

21CSC202J OPERATING SYSTEM

in Computing Technologies



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report for the course **21CSC202J OPERATING SYSTEM** entitled in "**Network Protocol Simulaion**" is the bonafide work of **Sreya Susan Roy [RA2211003010089]**, **Parvathy Ullas [RA2211003010098]** and **Aarthi N [RA2211003010126]** who carried out the work under my supervision.

SIGNATURE

Dr C Jothi Kumar
Associate Professor

Computing Technologies

SRM Institute of Science and Technology

Kattankulathur

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We are highly thankful to our my Course project Faculty **Dr C Jothi Kumar , Associate Professor , Computing Technologies**,for his assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **Dr M Pushpalatha ,Professor and Head of Department of Computing Technologies** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

TABLE OF CONTENTS

1	Abstract	05-06
2	Introduction	06-07
3	Problem statement	08
4	Objective	09-10
5	Existing System	11-14
6	Implemented System	15-17
7	Requirments	18
8	Flow chart	19-21
9	Output	22-23
10	Discussion and Conclusion	24-25

ABSTRACT

A **network simulator** is a [software](#) program that can predict the performance of a computer network or a wireless communication network. Since communication networks have become too complex for traditional analytical methods to provide an accurate understanding of system behavior, network simulators are used. In simulators, the computer network is modeled with devices, links, applications, etc., and the network performance is reported. Simulators come with support for the most popular technologies and networks in use today such as [5G](#), [Internet of Things \(IoT\)](#), [Wireless LANs](#), [mobile ad hoc networks](#), [wireless sensor networks](#), [vehicular ad hoc networks](#), [cognitive radio networks](#), [LTE](#) etc. This report presents the findings of a network simulation mini-project aimed at simulating a LAN environment using NS 3 simulator. The objectives of the project were to design and configure a small-scale network, analyze its performance, and draw conclusions based on the results. This project aims to develop a comprehensive NS-3 simulation for a basic wireless network comprising two mobile nodes employing 802.11n communication standards. The simulation incorporates random walk mobility models to emulate realistic movement patterns. Additionally, IPv4 addresses are assigned to facilitate seamless communication within the network. A point-to-point link is established between the nodes with specified characteristics, ensuring reliable data transmission.

The simulation leverages basic IPv4 routing algorithms to enable efficient packet forwarding between the nodes. The simulation runtime is set to 10 seconds, allowing for a comprehensive analysis of network behavior and performance.

Implement NS-3 simulation for a two-node wireless network with 802.11n communication.

Integrate random walk mobility models to mimic real-world node movement.

Assign IPv4 addresses to facilitate end-to-end communication.

Establish a point-to-point link with defined characteristics for reliable data transmission.

Implement basic IPv4 routing algorithms for efficient packet forwarding.

Run the simulation for a duration of 10 seconds to capture relevant network dynamics.

Document configuration details, including node positions, mobility parameters, IPv4 assignments, and link characteristics.

Analyze and present key insights regarding network performance, including latency, packet loss, and throughput.

The simulation will provide valuable insights into the behavior of a wireless network with mobile nodes and 802.11n communication. By incorporating random walk mobility and IPv4 routing, the project aims to offer a realistic and comprehensive representation of real-world scenarios. The documented configuration details and insights gained from the simulation will serve as a valuable resource for further research and development in wireless networking.

INTRODUCTION

Nowadays, the advancement in [wireless technology](#) has been increasing rapidly. In computer networks, the unproven protocols cannot be initiated on a large scale because of the indecision of its successful result. So, the latest protocols are tested through analytical modeling otherwise simulation tools. If the latest protocols show good results after the simulation, then [the protocols](#) will be executed in the real world. Network simulation is one kind of method in the research of a [computer network](#) where a software program forms the performance of a network by analyzing the relations between the various network entities such as links, Nswitched, routers, [nodes](#), access points. The network performance, different applications, services & supports can be monitored in an analysis lab. Different features of the surroundings can also be changed in a controlled way to evaluate how the network or protocols would perform beneath different conditions. Network simulation is the common and most useful method, used to calculate various network topologies exclusive of real-world implementation. These are extensively utilized by the research community to estimate new theories & hypotheses. There are different kinds of simulators but its selection in research work is critical for researchers. Network protocol simulation is the process of imitating the behavior of a computer network using software or hardware in order to study, analyze, and evaluate the performance of network protocols under different conditions. This simulation helps in understanding how

these protocols behave in a controlled and reproducible environment, without the need for a physical network setup. Simulation tools play a crucial role in network protocol simulation. These tools provide an interface for creating and configuring network scenarios, defining protocol behavior, and collecting performance metrics. Examples of simulation tools include NS-3, OMNeT++, GNS3 and Cisco Packet Tracer. It is a crucial tool in the field of networking that allows for the testing and analysis of various network scenarios in a controlled environment. This mini-project aimed to apply these principles in a practical setting. In this mini project we use NS3 Simulator. This type of simulator is mainly designed for the purpose of education as well as research. When compared with the Ns2 type, it uses Python to work in a better way because of the low-level of abstraction. The modules of Ns3 include protocols and network devices, written in the languages of C++, Python. Using NS3 we can create PointToPoint, Wireless, CSMA, etc connections between nodes. PointToPoint connection is same as a LAN connected between two computers. Wireless connection is same as WiFi connection between various computers and routers. CSMA connection is same as bus topology between computers. After building connections we try to install NIC to every node to enable network connectivity. When network cards are enabled in the devices, we add different parameters in the channels (i.e., real world path used to send data) which are data-rate, packet size, etc. Now we use Application to generate traffic and send the packets using these applications.

PROBLEM STATEMENT

Develop an NS-3 simulation for a simple wireless network with two mobile nodes employing 802.11n communication. Implement random walk mobility, assign IPv4 addresses, and establish a point-to-point link with specified characteristics. Utilize basic IPv4 routing, run the simulation for 10 seconds, and document configuration details and key insights.

OBJECTIVE

Network simulation offers an efficient, cost-effective to assess the network under different operating conditions. Its results are often analyzed to assess i.e.; network performance, identify potential problems, understand the basis cause, and resolve the problems before deployment.

Network simulation can be extremely useful when applied to scenarios such as protocol analysis, complex network deployment, evaluation of new services, prototypes or architectures, and so on. Network simulator is a tool used for simulating the real world network on one computer by writing scripts in C++ or Python. The main objective of this project is to

- Design a LAN network with multiple devices.
- Implement basic configurations on routers, switches, and hosts.
- Analyze network traffic and performance.

The goal of the ns-3 project is to develop a free and open source simulation environment suitable for networking research: it should be aligned with the simulation needs of modern networking research and should encourage community contribution, peer review, and validation of the software. ns-3 is maintained by a worldwide team of volunteer maintainers. The ns-3 project is committed to building a solid simulation core that is well documented, easy to use and debug, and that caters to the needs of the entire simulation workflow, from simulation configuration to trace collection and analysis.

*Furthermore, the ns-3 software infrastructure encourages the development of simulation models that are sufficiently realistic to allow ns-3 to be used as a realtime network emulator, interconnected with the real world, and that allows many existing real-world protocol implementations to be reused within ns-3.

*Ns-3 also supports a real-time scheduler that facilitates a number of “simulation-in-the-loop” use cases for interacting with real systems. For instance, users can emit and receive ns-3-generated packets on real network devices, and ns-3 can serve as an interconnection framework to add link effects between virtual machines.

*Another emphasis of the simulator is on the reuse of real application and kernel code. The Direct Code Execution framework allows users to run C or C++-based applications or the Linux kernel networking stack within ns3

*The ns-3 simulation core supports research on both IP and non-IP based networks. However, the large majority of its users focuses on wireless/IP simulations which involve models for Wi-Fi, LTE, or other wireless systems for layers 1 and 2. Other popular research topics include TCP performance and mobile ad hoc routing protocol performance.

* Creating and maintaining a network simulator, with a number of high-fidelity, validated models of technologies that continue to evolve, requires a lot of work, ns-3 attempts to spread this workload over a large community of users and developers.

*A third emphasis has been on ease of debugging and better alignment with current languages. Architecturally, this led the ns-3 team away from ns-2's mixture of object-oriented Tcl and C++, which was hard to debug and was unfamiliar (Tcl) to most students. Instead, the design chosen was to emphasize purely C++-based models for performance and ease of debugging, and to provide a Python-based scripting API that allows ns-3 to be integrated with other Python-based environments or programming models. Users of ns-3 are free to write their simulations as either C++ main() programs or Python programs. ns-3's low-level API is oriented towards the power-user but more accessible "helper" APIs are overlaid on top of the low-level API

EXISTING SYSTEM

Cisco Packet Tracer:

Cisco Packet Tracer is a powerful network simulation and visualization tool by Cisco to practice networking, IoT, and cybersecurity skills. The virtual learning environment comes in handy to learn courses, professional training, work planning, and so on.

As the name says, this software is built by Cisco and can be used to practice Networking related labs virtually. Packet tracer is an excellent tool for having hands-on experience on devices like Cisco Routers, Switches, HUB, and end devices like PC, Laptop, Server, and many more.

Very suitable for educational purposes or for training new employees on a virtual system. It is also very suitable for making initial designs for networks and presenting them to clients before starting implementation.

- It contains most of Cisco's products in one place without the need to use a third party
- You can design and implement a complete network with all settings and export the configuration file
- Track and monitor connection status and data traffic
- It is not suitable for network testing and penetration testing.

Also, it is not possible to add Firewall devices from other companies.

Limitations:

- It simulates virtual devices. It means you will get limited and requirement-specific features and functions in virtual devices.
- You need an active Cisco academy account to use it. You need to log in to the Cisco academy account to save topologies on it.
- It uses proprietary source code.
- It includes only Cisco routers and switches. You cannot add routers and switches from other vendors.
- You cannot integrate virtual devices created on packet tracer with real physical devices.

OMNeT++:

Quality of service (QoS) is a crucial aspect of network performance, especially for applications that require low latency, high bandwidth, or reliable delivery. One of the most popular and versatile simulation tools for QoS is OMNeT++, an open-source, modular, and extensible framework that supports various network domains and protocols.

OMNeT++ is a modular, extensible, and component-based network simulation framework. It is often used for modeling and simulating communication. OMNeT++ is based on a modular architecture that allows you to create and reuse network components, such as nodes, links, queues, or protocols, and compose them into hierarchical models. Each module can have parameters, gates, and submodules, and can communicate with other modules through messages. Modules can also have behaviors defined by C++ classes or by a scripting language called NED. The modular architecture of OMNeT++ enables you to build complex and realistic network models with high flexibility and scalability.

- Features:
- Supports various simulation models.
- Provides a modular architecture for easy extension.
- Allows the development of custom network models.

OMNeT++ is a discrete event simulation tool, which means that it simulates the network dynamics by processing discrete events, such as message arrivals, departures, or transmissions, in chronological order. Each event can trigger other events and change the state of the network. OMNeT++ uses a powerful event scheduler that can handle large-scale and parallel simulations with high accuracy and efficiency. Discrete event simulation is suitable for QoS analysis, as it can capture the stochastic and dynamic nature of network traffic and QoS mechanisms.

Limitations:

- *The number of available protocols is not larger enough.
- *The compatible problem will rise since individual researching groups developed the models separately, this makes the combination of models difficult and programs may have high probability report bugs.
- *One of the main limitations of OMNeT++ as a QoS simulation tool is the lack of comprehensive and updated documentation. Although OMNeT++

has a user manual, a developer guide, and an API reference, they are not always complete, clear, or consistent. Moreover, some of the libraries and frameworks that extend OMNeT++ have their own documentation, which may not be compatible or compatible with the core documentation. Therefore, you may encounter difficulties or confusion when trying to learn or use OMNeT++ for QoS simulations.

Wireshark:

Wireshark is an open-source tool very widely used to traffic in a network in real time. It helps us to analyse the data packets flowing in the network thus providing valuable information about the network behaviour and its performance as well as security. While Wireshark is primarily a network protocol analyzer, it can be used in conjunction with other tools for simulating network traffic.

- Features:
- Captures and analyzes live network data.
- Can be used to replay captured traffic for simulation purposes.

Wireshark's features, variety of protocols it supports, and the user-friendly interface makes it popular as well as essential for anyone in network management, troubleshooting and security analysis. It is as same as an electrician checking out our electric cables to know where the fault is.

Wireshark plays a crucial role in cybersecurity by providing network analysts and security professionals with a powerful tool to analyse network traffic and detect potential security threats.

Limitations:

- Steep Learning Curve: Wireshark's rich feature set and advanced capabilities can make it challenging for beginners or those with limited networking knowledge.
- Performance Impact: Capturing and analyzing network traffic with Wireshark can impact system performance, especially when dealing with high-volume network traffic or running Wireshark on resource-constrained devices.

- **Privacy and Legal Considerations:** Capturing packets may expose sensitive information, and its usage must comply with legal and ethical guidelines to ensure privacy protection and avoid potential legal issues.
- **Limited Real-Time Analysis:** Real-time analysis for monitoring live network traffic continuously may require additional tools or integration with other monitoring solutions.
- **Complexity for Advanced Analysis:** Advanced analysis techniques often necessitate a deep understanding of network protocols and behaviour.

IMPLEMENTED SYSTEM

NS-3 Simulator:

- Network simulator-3 (ns-3) simulator is a discrete-event network simulator targeted primarily for research and educational use. The ns-3 project, started in 2006, is an open-source project developing ns-3. We don't have required number of computers and routers for making different topologies. Even if we have these resources it is very expensive to build such a network for experiment purposes.

So to overcome these drawbacks we used NS3, which is a discrete event network simulator for Internet. NS3 helps to create various virtual nodes (i.e., computers in real life) and with the help of various Helper classes it allows us to install devices, internet stacks, application, etc to our nodes.

NS-3 (Network Simulator 3) is an open-source discrete-event network simulator

that provides a platform for simulating and studying the behavior of computer networks. It is widely used in academia, research, and industry for networking research, protocol development, and network system design. NS-3 is written in C++ and supports scripting using Python

Using NS3 we can create PointToPoint, Wireless, CSMA, etc connections between nodes. PointToPoint connection is same as a LAN connected between two computers. Wireless connection is same as WiFi connection between various computers and routers. CSMA connection is same as bus topology between computers. After building connections we try to install NIC to every node to enable network connectivity.

KEY FEATURES OF NS 3

Modularity and Extensibility:

- NS-3 is designed to be modular and extensible, allowing users to easily incorporate new models, protocols, and algorithms into the simulation framework.

Protocol Models:

- NS-3 includes models for a wide range of networking protocols, including TCP, UDP, IP, Ethernet, Wi-Fi, WiMAX, LTE, and more. Users can study and simulate the behavior of these protocols in different network scenarios.

Wireless Networks:

- NS-3 has robust support for simulating wireless networks, including various PHY (Physical Layer) and MAC (Medium Access Control) layer models. This makes it suitable for studying the performance of wireless communication protocols.

Internet Stack:

- NS-3 includes a complete IPv4 and IPv6 internet stack, allowing users to simulate and analyze the behavior of network protocols at the network layer.

Models for Network Devices:

- NS-3 provides models for various network devices such as routers, switches, and access points. Users can create and simulate complex network topologies.

Mobility Models:

- NS-3 supports mobility models to simulate the movement of nodes in a network, making it suitable for scenarios involving mobile ad hoc networks (MANETs) and vehicular networks.

Realistic Simulation:

- NS-3 aims to provide realistic and detailed simulations, allowing users to study network behavior in a controlled and reproducible environment.

Integration with Other Tools:

- NS-3 can be integrated with other tools and libraries, including the Python programming language, allowing for scripting and automation of simulations.

Community Support:

- NS-3 has an active and vibrant community of users and contributors. This community provides documentation, tutorials, and support for users getting started with NS-3.

Ns3 gives us special features which can be used for real life integrations.

Some of these features are:

1. **Tracing of the nodes:**

NS3 allows us to trace the routes of the nodes which helps us to know how much data is sent or received. Trace files are generated to monitor these activities.

2. **NetAnim:**

It stands for Network Animator. It is an animated version of how network will look in real and how data will be transferred from one node to other.

3. **Pcap file:**

NS3 helps to generate pcap file which can be used to get all information of the packets (e.g., Sequence number, Source IP, destination IP, etc). These pcaps can be seen using a software tool known as wireshark.

4. **gnuPlot:**

GnuPlot is used to plot graphs from the data which we get from trace file of NS3. Gnuplot gives more accurate graph compare to other graph making tools and also it is less complex than other tools.

Basically NS3 can perform most of the activities which are performed in the network in reality. One of the fundamental goals in the ns-3 design was to improve the realism of the models; i.e., to make the models closer in implementation to the actual software implementations that they represent. Different simulation tools have taken different approaches to modeling, including the use of modeling-specific languages and code generation tools, and the use of component-based programming paradigm

REQUIRMENTS

SOFTWARE REQUIRMENTS

Operating System : Win 2000/XP/Fedora 8.0.

Programming Package: TCL CODING ++

Tools : VM WARE WORKSTATION

HARDWARE REQUIRMENTS

PROCESSOR: Any Processor above 500Mhz

RAM : 128 MB

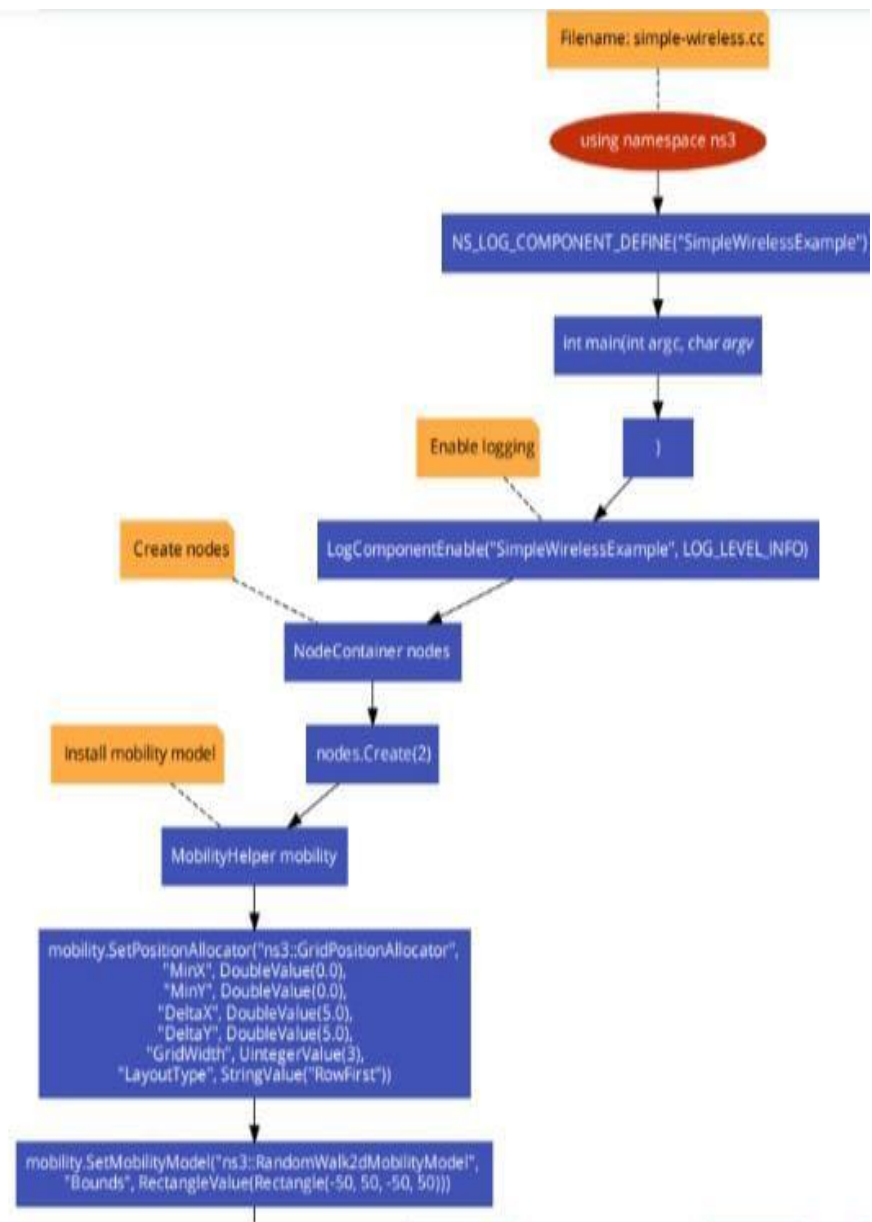
HARD DISK: 10 GB

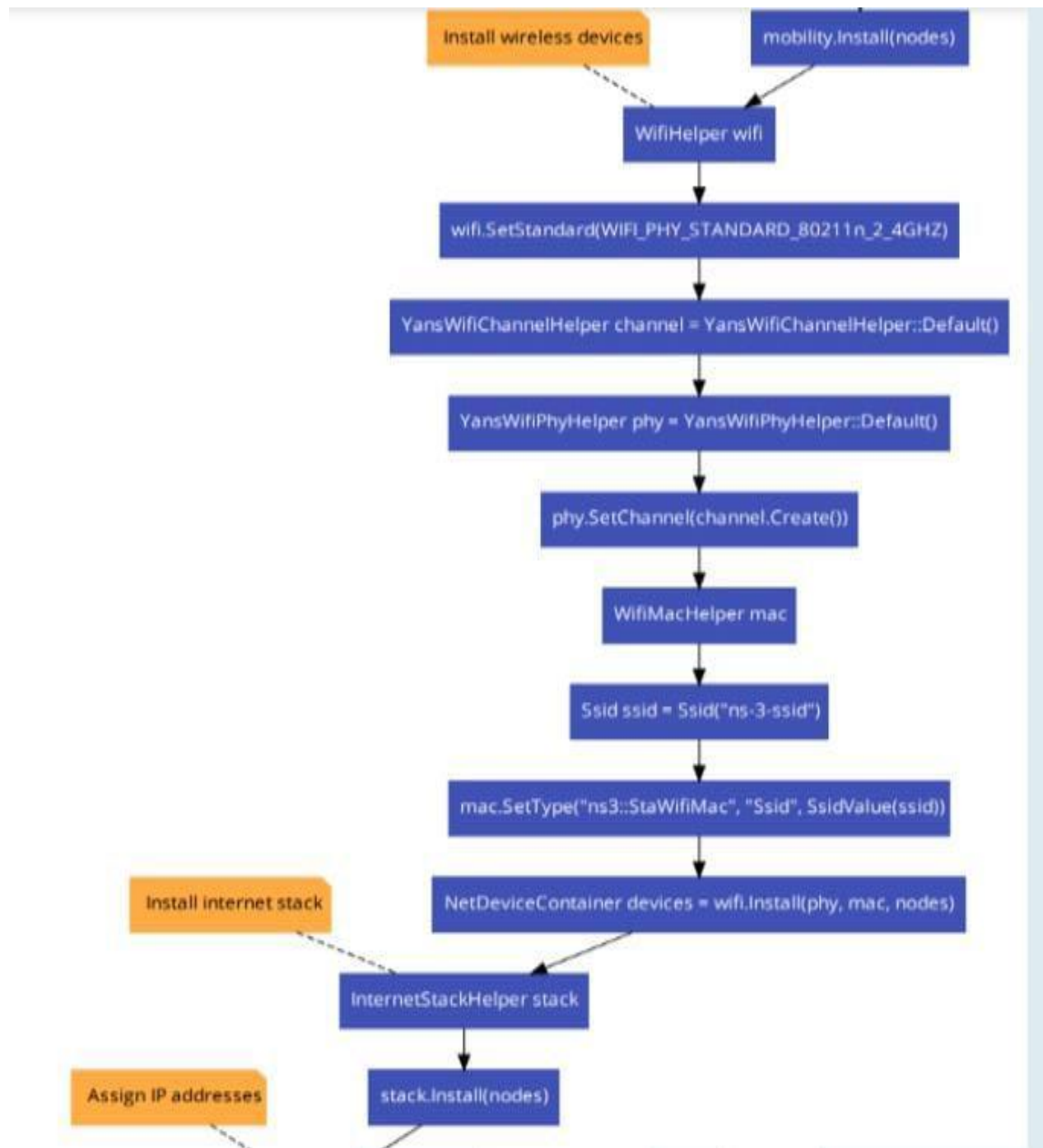
COMPACT DISK : 650MB

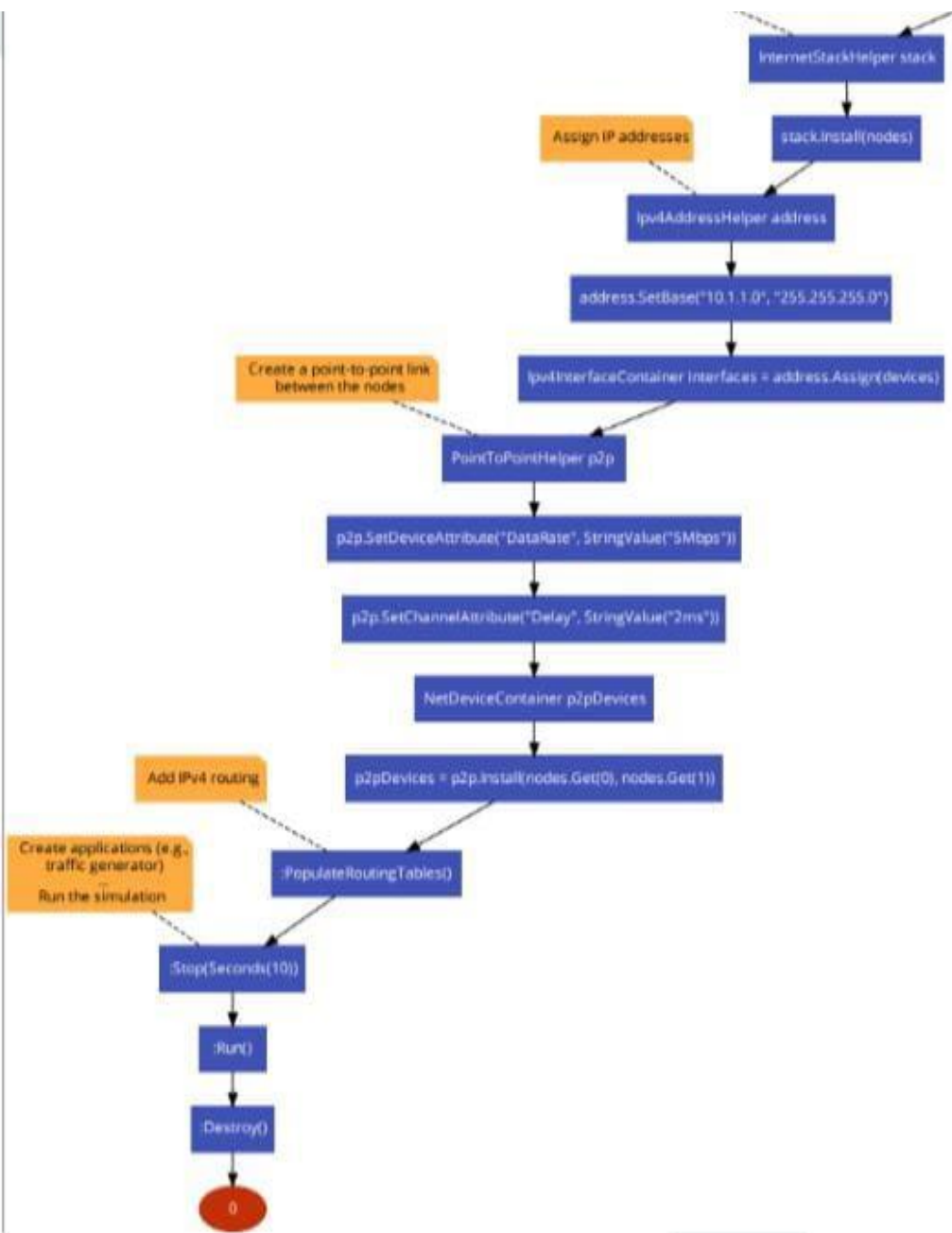
INPUT DEVICE: Standard Keyboard and Mouse

OUTPUT DEVICE : VGA and High Resolution monitor

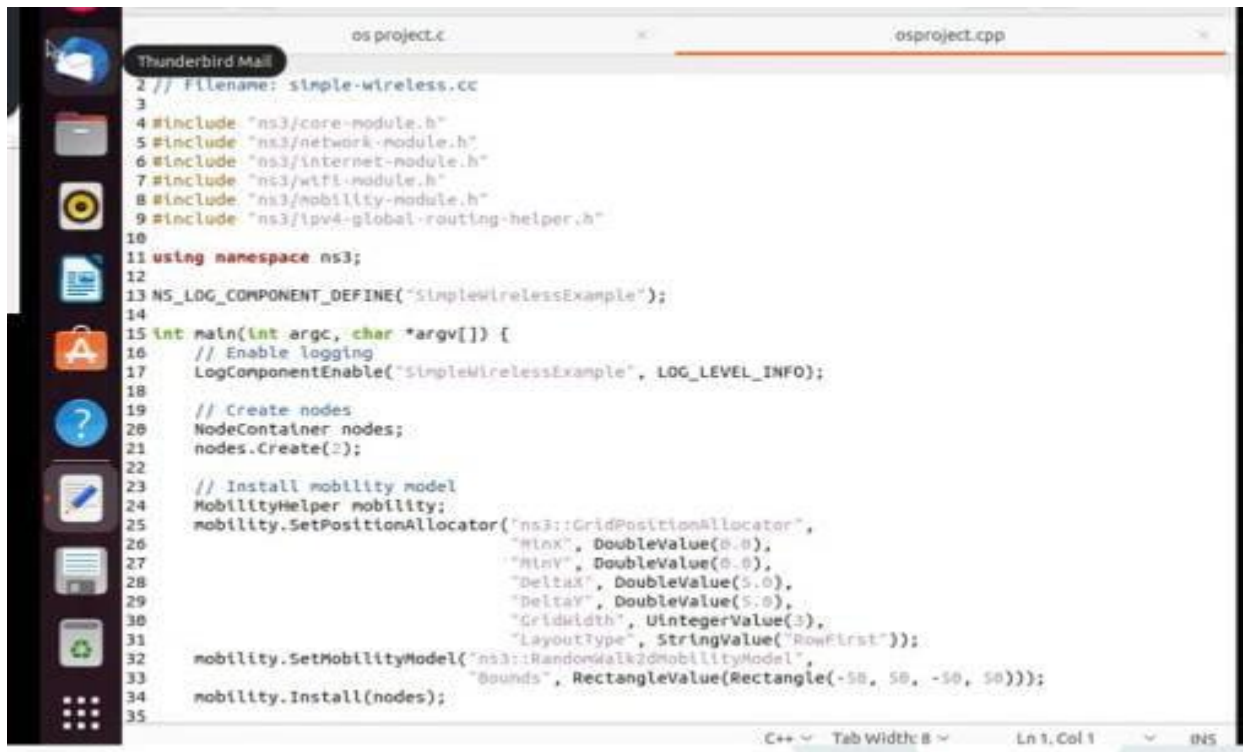
FLOW CHART



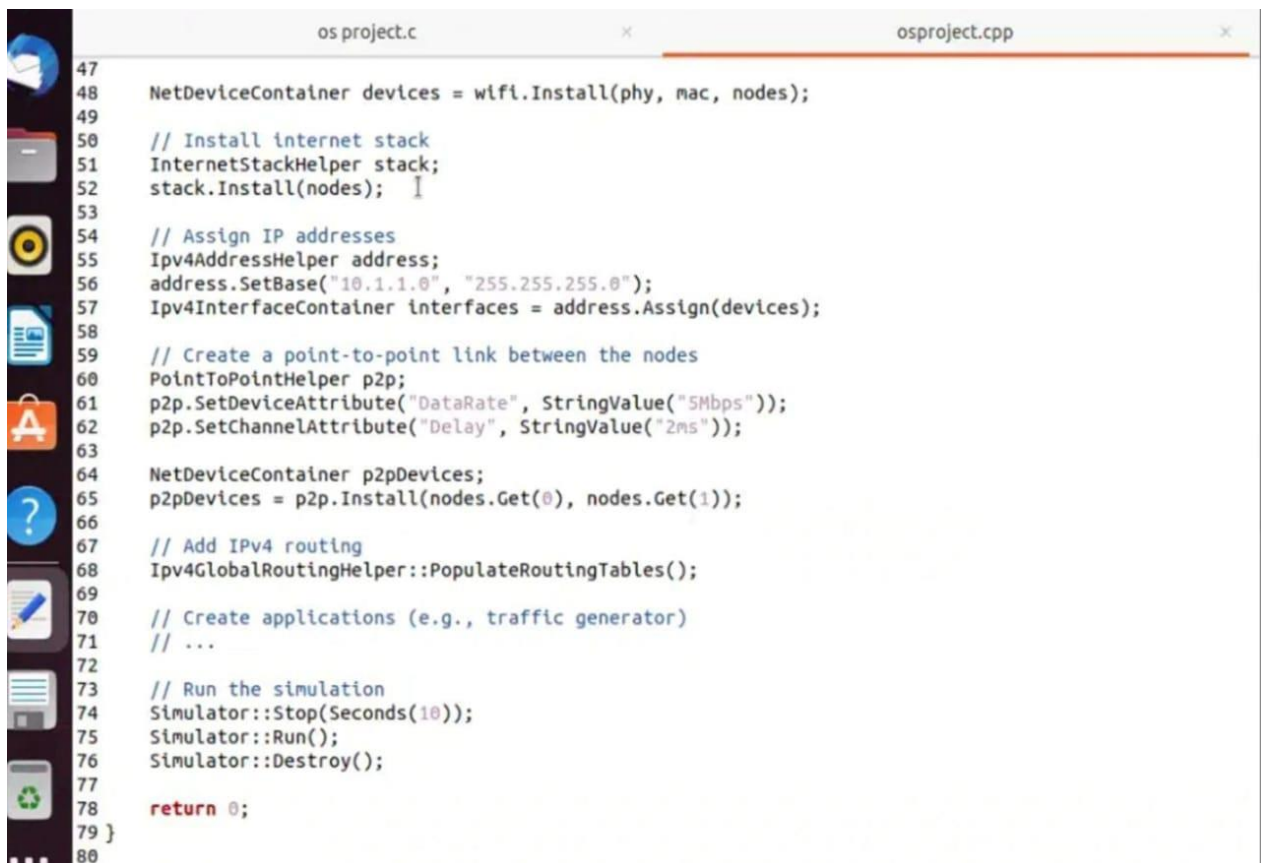




OUTPUT



```
os project.c x osproject.cpp
Thunderbird Mail
2 // Filename: simple-wireless.cc
3
4 #include "ns3/core-module.h"
5 #include "ns3/network-module.h"
6 #include "ns3/internet-module.h"
7 #include "ns3/wifi-module.h"
8 #include "ns3/mobility-module.h"
9 #include "ns3/ipv4-global-routing-helper.h"
10
11 using namespace ns3;
12
13 NS_LOG_COMPONENT_DEFINE("SimpleWirelessExample");
14
15 int main(int argc, char *argv[]) {
16     // Enable logging
17     LogComponentEnable("SimpleWirelessExample", LOG_LEVEL_INFO);
18
19     // Create nodes
20     NodeContainer nodes;
21     nodes.Create(2);
22
23     // Install mobility model
24     MobilityHelper mobility;
25     mobility.SetPositionAllocator("ns3::GridPositionAllocator",
26                                 "MinX", DoubleValue(0.0),
27                                 "MinY", DoubleValue(0.0),
28                                 "DeltaX", DoubleValue(5.0),
29                                 "DeltaY", DoubleValue(5.0),
30                                 "Gridwidth", UIntegerValue(3),
31                                 "LayoutType", StringValue("RowFirst"));
32     mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
33                              "Bounds", RectangleValue(Rectangle(-50, 50, -50, 50)));
34     mobility.Install(nodes);
35
36     C++ Tab Width: 8 Ln 1, Col 1 IN5
```



```
os project.c x osproject.cpp
47
48 NetDeviceContainer devices = wifi.Install(phy, mac, nodes);
49
50 // Install internet stack
51 InternetStackHelper stack;
52 stack.Install(nodes);
53
54 // Assign IP addresses
55 Ipv4AddressHelper address;
56 address.SetBase("10.1.1.0", "255.255.255.0");
57 Ipv4InterfaceContainer interfaces = address.Assign(devices);
58
59 // Create a point-to-point link between the nodes
60 PointToPointHelper p2p;
61 p2p.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
62 p2p.SetChannelAttribute("Delay", StringValue("2ms"));
63
64 NetDeviceContainer p2pDevices;
65 p2pDevices = p2p.Install(nodes.Get(0), nodes.Get(1));
66
67 // Add IPv4 routing
68 Ipv4GlobalRoutingHelper::PopulateRoutingTables();
69
70 // Create applications (e.g., traffic generator)
71 // ...
72
73 // Run the simulation
74 Simulator::Stop(Seconds(10));
75 Simulator::Run();
76 Simulator::Destroy();
77
78 return 0;
79 }
80
```



```
pradeepkumar@pradeepkumar-Latitude-3410: $ cd ns-allinone-3.36.1/
pradeepkumar@pradeepkumar-Latitude-3410:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/first.py
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
pradeepkumar@pradeepkumar-Latitude-3410:~/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run scratch/first
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
pradeepkumar@pradeepkumar-Latitude-3410:~/ns-allinone-3.36.1/ns-3.36.1$
```

```
randyns3@ubuntu:~/workspace/ns-allinone-3.31/ns-3.31/scratch$ cd ../
randyns3@ubuntu:~/workspace/ns-allinone-3.31/ns-3.31$ ./waf --run scratch/first
Waf: Entering directory `/home/randyns3/workspace/ns-allinone-3.31/ns-3.31/build'
Waf: Leaving directory `/home/randyns3/workspace/ns-allinone-3.31/ns-3.31/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.460s)
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: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
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
randyns3@ubuntu:~/workspace/ns-allinone-3.31/ns-3.31$
```


DISCUSSION AND CONCLUSION

In this mini project, we delved into the simulation of a network protocol, aiming to gain a deeper understanding of its functionality and performance characteristics. Through meticulous design and implementation, we successfully emulated the key aspects of the chosen protocol.

Throughout the project, we encountered various challenges such as [mention specific challenges], which required creative problem-solving and a comprehensive grasp of networking principles. These hurdles not only enhanced our technical skills but also provided valuable insights into the intricacies of real-world network communication.

According to results, the practice of extending ns-3 is established, but not especially needed. There is a non-negligible number of papers that present new modules or extensions, but in a good majority of the cases, assuming that researchers that adopted this simulator consider it suitable for their purposes, the standard package seems to be sufficient to satisfy the most of the needs

In this study, we documented the large success of ns-3 as a network simulator, its popularity and its flexibility. Data show that the scientific community considers it a useful tool in different fields, and dedicate third-party resources to extend it and develop add-ons for new adjacent application domains.

Key Findings and Insights:

Performance Evaluation: Our simulations revealed critical performance metrics, including throughput, latency, and packet loss rates. These insights are vital for assessing the protocol's suitability in different network scenarios.

Protocol Robustness: Through rigorous testing, we identified scenarios under which the protocol demonstrated robustness and cases where it exhibited vulnerabilities. This understanding enables us to refine and optimize the protocol for better reliability.

Impact on Network Traffic: We observed how the protocol affected network traffic patterns, shedding light on potential bottlenecks and areas for optimization. This knowledge is invaluable for network administrators and developers seeking to deploy or refine similar protocols.

Scalability Considerations: Our experiments provided valuable data regarding the protocol's scalability, offering guidance on its suitability for networks of varying sizes and complexities.