VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-560018, Karnataka, India

Report On

"MINI-PROJECT"

Submitted in partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering in Computer Science and Engineering

Submitted by

J.YESHAS [1VE21CS064]
JAGADABI SHANMUKHA [1VE21CS065]

Under the Guidance of

Dr. KARTHIK BU
Assistant Professor
Department of CSE

SVCE BENGALURU
SRI VENKATESHWARA COLLEGE OF ENGINEERING
— Affiliated to VTU, Approved by AICTE, Recognised by UGC u/s 2(f) & 12(B)—

DEPARTMENT OF COMUTER SCIENCE AND ENGINEERING

SRI VENKATESHWARA COLLEGE OF ENGINEERING

Vidyanagar, Bengaluru-562157

2023-2024

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without complementing those who made it possible, whose guidance and encouragement made our efforts successful.

My sincere thanks to highly esteemed institution **SRI VENKATESHWARA COLLEGEOF ENGINEERING** for grooming me to be a Software Engineer.

I express our sincere gratitude to **Dr. NAGESWARA GUPTHA M**, Principal, SVCE,Bengaluru for providing the required facility.

I would like to extend our sincere thanks to **Dr. HEMA MS, HOD**, Dept. of CSE, SVCE,Bengaluru, for her suggestions which helped us to complete the Project.

I would also like to express our sincere thanks to **Dr. KARTHIK BU** Asst. Prof, Dept. of CSE, SVCE Bengaluru, for guidance and support in bringing this project to completion.

I would like to express my gratitude to all the education platforms which helped me gain knowledge about many fields related to my interests.

I am thankful to one and all who have been involved in this work directly or indirectly for the successful completion of this project.

J.Yeshas [1VE21CS064]

Jagadabi Shanmukkha [1VE21CS065]

ABSTRACT

This project presents a simulation-based exploration of various network topologies. Specifically focusing on star, ring, and bus configurations, the study aims to analyze their structural attributes, functional characteristics, and performance metrics. Leveraging simulation tools, virtual networks are constructed to emulate these topologies, enabling the examination of factors such as scalability, fault tolerance, and data transmission efficiency. Through systematic experimentation and analysis, insights into the strengths and limitations of each topology are gleaned, offering valuable perspectives for network design and optimization. The findings contribute to a deeper understanding of network topology dynamics and aid in informed decision-making for real-world network-deployments.

TABLE OF CONTENTS

Introduction

In today's interconnected world, networks are the backbone of communication, allowing us to stay connected and share information seamlessly. But have you ever wondered how these networks are structured? That's where network topologies come into play.

Imagine a network as a web of connections between devices like computers, printers, and smartphones. Now, picture three common ways these devices can be connected: the star, the ring, and the bus.

Star Topology: A star topology, also known as a star network, is a common network configuration where each device connects to a central hub. This arrangement resembles a star, with peripheral devices branching out from the central hub. Unlike mesh topology, direct communication between devices isn't possible; all communication occurs through the central hub. Devices typically act as clients, while the central hub serves as a server. Star topology is widely used due to its simplicity in setup, utilizing either RJ-45 or coaxial cables for connections. Examples of star topology can be found in various real-life settings such as airports, hospitals, banks, and educational institutes.

Ring Topology: A ring topology is a network structure where devices are connected in a circular manner, passing information to neighboring nodes. Unlike bus topology, it efficiently handles heavy loads and operates as a one-way unidirectional ring network. Two types exist: bidirectional and unidirectional. Various setups work differently based on the devices connected. It's applicable in LANs or WANs, using RJ-45 or coaxial cables. Advantages include no need for a central hub, simplifying installation and troubleshooting compared to other networks.

Bus Topology: The bus topology connects all stations through a single backbone cable, with nodes linked via drop cables or directly to the backbone. Messages sent by any node are broadcast to all stations, regardless of addressing. Commonly used in Ethernet (802.3) and 802.4 standard networks, it offers simplicity in configuration compared to other topologies. The backbone cable acts as a "single lane" for message transmission, utilizing the CSMA (Carrier Sense Multiple Access) access method.

Software requirements and specification

SOFTWARE REQUIREMENTS SPECIFICATION

- Operating System: Windows 10/11
- Virtualization platform: Virtual Box / VMware WorkStation

HARDWARE REQUIREMENTS:

- Processor (CPU): A 64-bit x86-compatible CPU with virtualization support ,Multi-core processors are beneficial for running multiple virtual machines simultaneously.
- Memory (RAM): Minimum of 4 GB RAM
- Storage: Minimum of 50GB
- Graphics capable of supporting the VMware

Implementation

An "implementation" of Nam and Tcl should be understood as a software environment that facilitates the execution of network simulations written using Tcl scripts, particularly in conjunction with the Network Animator (Nam) tool. While there are various software packages that offer support for network simulation and visualization, it's important to note that some of these packages serve as distributions or extensions rather than entirely new implementations of the Tcl scripting language or Nam tool. These implementations adhere to the specifications outlined by the Tcl language and the Nam tool, ensuring compatibility and consistency with their respective standards for network simulation and visualization.

Nam

Nam (Network Animator): Nam is a visualization tool for network simulations, commonly used with NS-2. It offers a graphical interface to observe and analyze network behavior in real-time. With Nam, users can create animated representations of network topologies, track data packet movement, and debug network protocols efficiently. It's essential for understanding network dynamics and evaluating performance visually.

Tcl

Tcl, short for Tool Command Language, is a versatile scripting language commonly used for controlling and automating various software applications and systems. With its simple syntax and powerful features, Tcl is widely employed in a variety of fields, including network simulation, software testing, embedded systems, and rapid prototyping. Tcl's flexibility and ease of use make it a popular choice for creating scripts to define network topologies, configure simulation parameters, and automate tasks in network simulation software like NS-2. Overall, Tcl plays a crucial role in streamlining development workflows and enhancing productivity in diverse domains.

CODE FOR STAR TOPOLOGY

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Executenam on the trace file
exec namout.nam&
exit 0
}
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Change the shape of center node in a star topology
$n0 shape square
#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 1Mb 10ms DropTail
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
```

```
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

CODE FOR RING TOPOLOGY

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Executenam on the trace file
exec namout.nam&
exit 0
}
#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
```

```
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

CODE FOR BUS TOPOLOGY

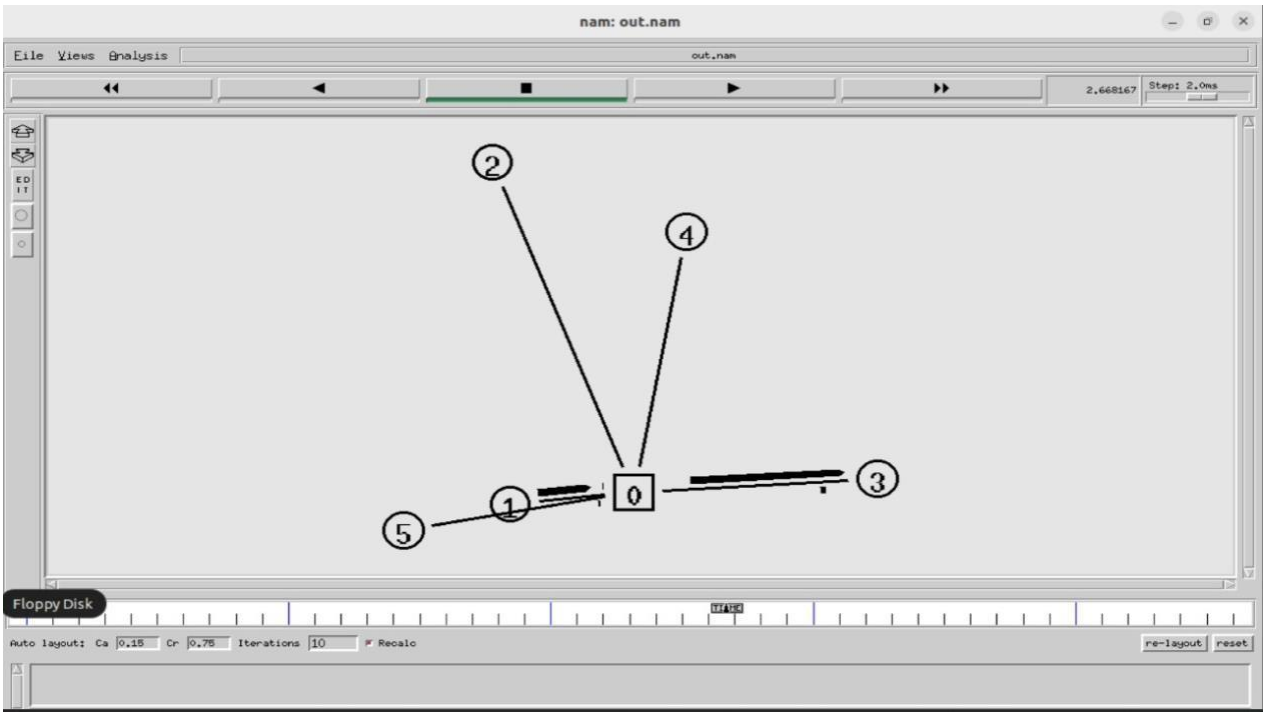
```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Executenam on the trace file
exec namout.nam&
exit 0
}
#Create five nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#Create Lan between the nodes
set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd
Channel]
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
```

```
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

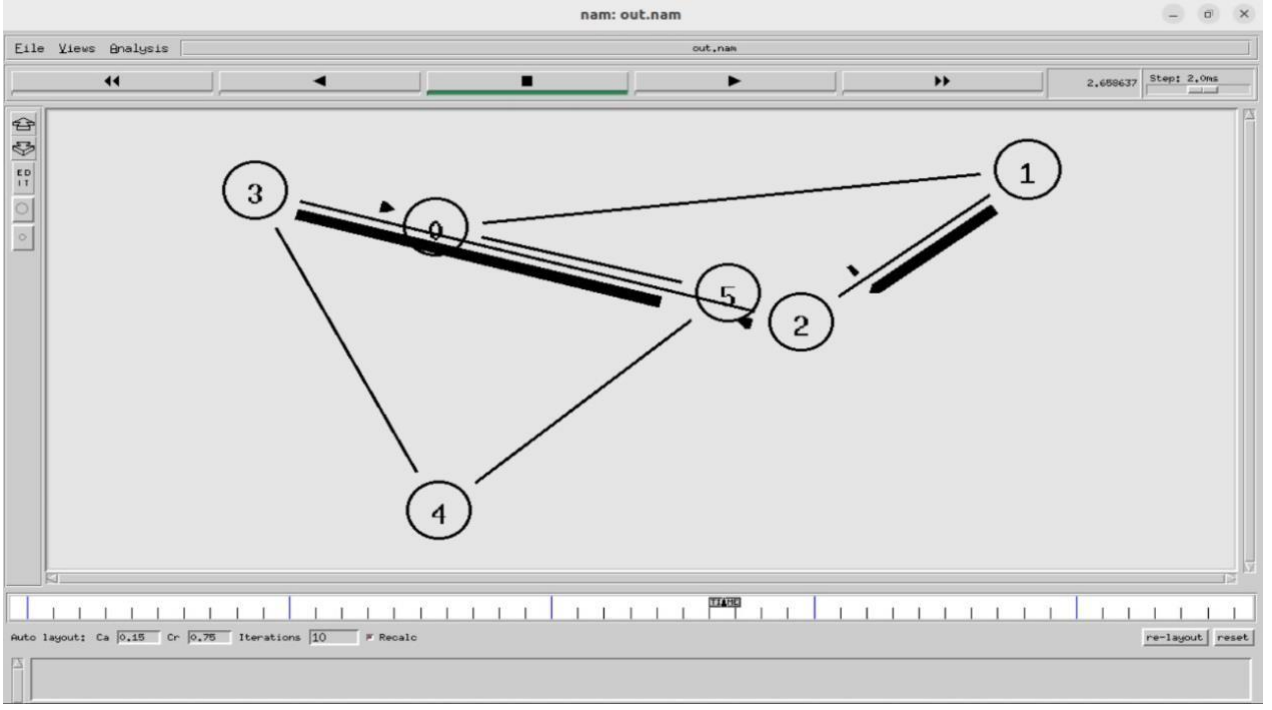
User Interface

SCREENSHOTS:

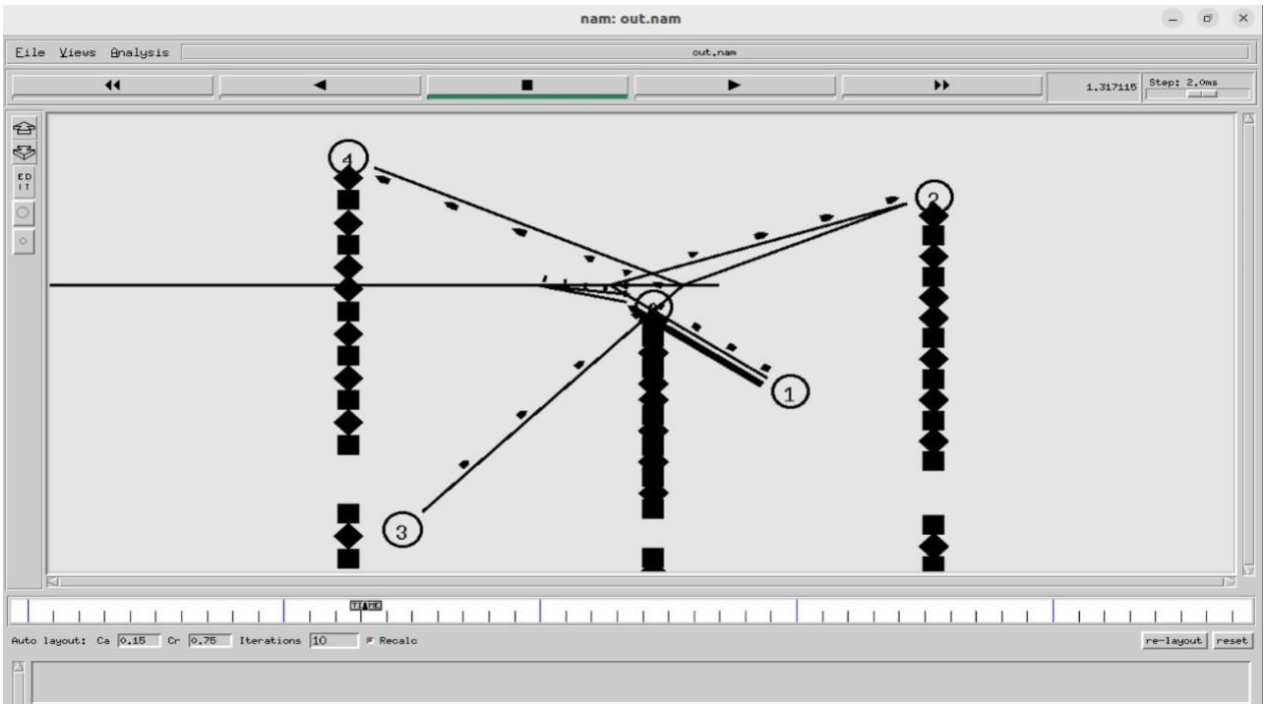
Star topology



Ring topology



Bus topology



CONCLUSION

In conclusion, our exploration into network topology simulation has been a fascinating journey of discovery. We've delved into the intricacies of star, ring, and bus topologies, unraveling their unique characteristics and practical applications in the real world. Through the lens of simulation, we've essentially been architects of virtual networks, tinkering with configurations and observing how they respond to different scenarios. Tools like Tcl and Nam have been our trusty companions on this voyage, providing a user-friendly interface to navigate the complexities of network dynamics. Visualizing the flow of data in real-time has been akin to watching a mesmerizing dance, each step revealing insights into network behavior. In the grand scheme of things, our journey has illuminated the importance of thoughtful network design and management. It's akin to weaving a tapestry, each thread carefully chosen to create a robust and resilient network infrastructure that connects us all in this digital age. As we bid adieu to our simulation adventure, let us carry forward the lessons learned and continue to chart new paths in the ever-evolving land of networking.

Reference Links

1. https://www.javatpoint.com/computer-network-topologies
2. https://youtube.com/playlist?list=PLlXl4m5lGmLXXckrYU9Lw0sAAON_4vSzg&si=Z16IdkBzBA8JFx7Y
3. https://www.geeksforgeeks.org/types-of-network-topology/
4. https://www.tutorialspoint.com/data_communication_computer_network/computer_network_topologies.htm
5. https://www.studytonight.com/computer-networks/network-topology-types
6. https://www.tutorialsweb.com/ns2/NS2-1.htm
7. William Stallings: Cryptography and Network Security, Pearson 6th edition.
8. V. K Pachghare: Cryptography and Information Security, PHI 2nd Edition
9. BehrouzA.Foruzan, Cryptography and Network Security, Tata McGraw Hill 2007
10. https://www.researchgate.net/figure/Network-topology-in-NS2-simulation-platform_fig3_339751094