**NETWORKING TECHNOLOGIES**

**CP3154**

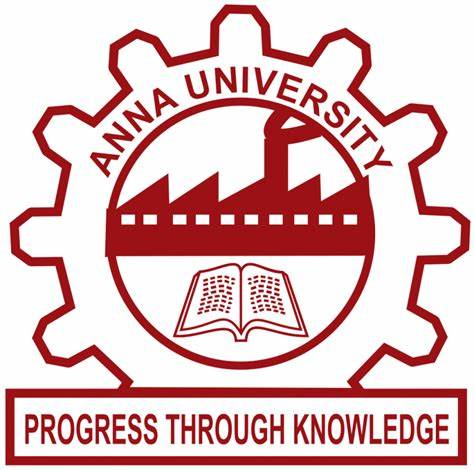
**LAB RECORD**

***Submitted by***

**JOBIN MARK D**

**2023184028**

**MASTER OF ENGINEERING IN   
CSE (SPLN. IN OPERATIONS RESEARCH)**

****

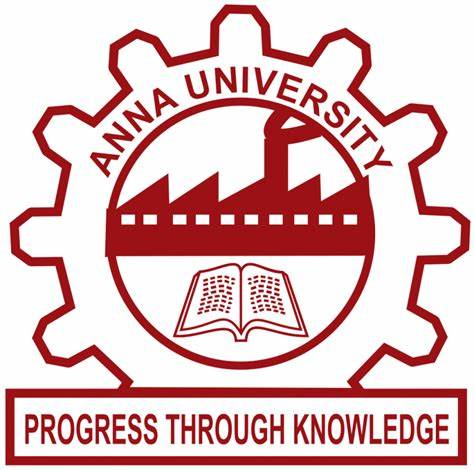
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**COLLEGE OF ENGINEERING GUINDY  
ANNA UNIVERSITY, CHENNAI.  
DECEMBER – 2023.**

**COLLEGE OF ENGINEERING GUINDY**

**ANNA UNIVERSITY, CHENNAI.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**CERTIFICATE**

This is to certify that bonafide record work done by **Mr/Ms** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Register Number**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**CP3154 NETWORKING TECHNOLOGIES** during the Academic year of 2023-2024 (ODD SEMESTER).

**Staff In – Charge Head of Department**

Submitted for the University Partical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER**  **EXTERNAL EXAMINER**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EX.NO** | **DATE** | **LIST OF EXPERIMENTS** | **STAFF SIGNATURE** | **REMARKS** |
| 1 |  | Installation steps for NS-3. |  |  |
| 2 |  | Implementation of Point-to-Point network between two Nodes. |  |  |
| 3 |  | Implementation of Wired Network Topology Using LAN Connectivity. |  |  |
| 4 |  | Implementation of Wireless Network topology Using Wi-Fi Connectivity. |  |  |
| 5 |  | Analysing TCP Congestion Window and Packet Tracing.in Point-to-Point Network . |  |  |
| 6 |  | Implementation of OLSR Routing Protocol in a Point-to-Point Network Using NS-3. |  |  |

|  |  |
| --- | --- |
| **Ex. No: 1** | **Installation steps for NS-3.** |
| **Date: 24.08.23** |

**Aim:**

To provide step-by-step instructions for installing ns-3.36.1 on an Ubuntu-based system and executing both C++ and Python programs using NS-3. The objective is to ensure a successful installation and the ability to run network simulations in NS-3.

**Description:** NS-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free, open-source software, licensed under the GNU GPLv2 license, and maintained by a worldwide community.

**Procedure:**

**Step 1:** Update Package Repositories

* Open a terminal and update the package repositories by running the following command

$ sudo apt update

**Step 2:** Install Prerequisites

* Install the necessary packages by running the following command

$ sudo apt install g++ python3 python3-dev pkg-config sqlite3 cmake python3-setuptools git qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 openmpi-bin openmpi-common openmpi-doc libopenmpi-dev autoconf cvs bzr unrar gsl-bin libgsl-dev libgslcblas0 wireshark tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libc6-dev libc6-dev-i386 libclang-dev llvm-dev automake python3-pip libxml2 libxml2-dev libboost-all-dev

**Step 3:** Download NS-3

* Download the NS-3 package, ns-allinone-3.36.1.tar.bz2, from nsnam.org by running the following command

$ wget https://www.nsnam.org/releases/ns-allinone-3.36.1.tar.bz2

**Step 4:** Extract NS-3

* Navigate to your home folder and extract the downloaded file. You can do this using the GUI by right-clicking and selecting "Extract" or using the following command:

$ tar jxvf ns-allinone-3.36.1.tar.bz2

**Step 5:** Build NS-3

* Change to the extracted directory

$ cd ~/ns-allinone-3.36.1/

* Run the build script with options to enable examples and tests

$ ./build.py --enable-examples --enable-tests

**Step 6:** Executing C++ Program:

* Navigate to the NS-3 Directory:

$ cd ns-3.36.1

* To execute a C++ program, use the following command:

$ ./ns3 run scratch/filename

* To execute a Python program, use the following command:

$ ./ns3 run scratch/filename.py

**Step 7:** For Animation Interface:

* Use following module for NetAnim.

#include ”ns3/netanim– module.h”

* Before run the ./NetAnim command, open the “filename.cc” in text editor
* and add line before simulation.run()

AnimationInterface anim (“filename.xml”);

anim.SetConstantPosition(nodes.Get(0), 1.0, 2.0);

* For Run NetAnim

$ ./NetAnim

**Result:** Network simulator (ns3) is installed and built successfully in Ubuntu 22.04 OS.

|  |  |
| --- | --- |
| **Ex. No: 2** | **Implementation of Point-to-Point network between two Nodes.** |
| **Date: 31. 08. 23** |

**Aim:**

To create a point-to-point connection between two nodes in ns-3.

**Network Topology:**

10.1.1.0

n0 ------------------ n1

point-to-point

**Algorithm:**

Step 1: Start

Step 2: Initialize ns-3 environment:

Step 3: Setup parse-command line arguments.

Step 4: Setup time resolution in nanoseconds and enable logging for

UdpEchoClient and UdpEchoServer application.

Step 5: Create a container for two network nodes(n0, n1):

Step 6: Setup a Point-to-Point communication channel with a datarate and delay.

Step 7: Install the Internet stack on network nodes.

Step 8: Assign IPv4 addresses to the network interface.

Step 9: UdpEchoServer application:

Step 10: Install and configure the UdpEchoServer application.

Step 11: UdpEchoClient application:

Step 12: Install and configure the UdpEchoClient application.

Step 13: Enable Animation:

Step 14: Create an AnimationInterface and get output in XML file name.

Step 15: Set the initial positions of nodes using SetConstantPosition.

Step 14: Run the Simulation

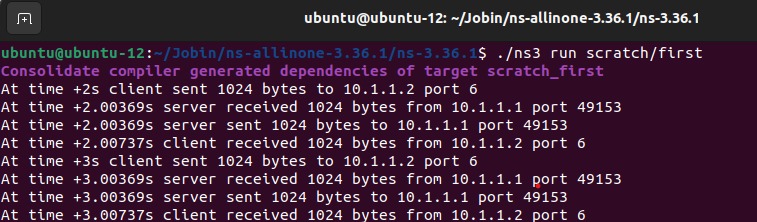
Step 15:  Destroy the Simulator

Step 16: Stop

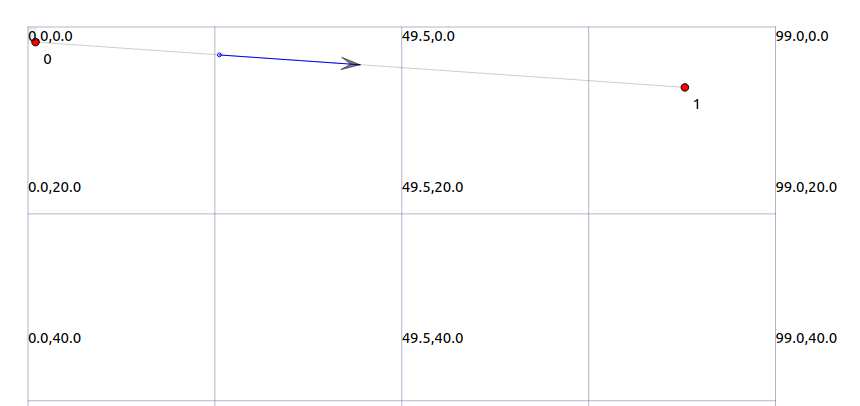
**Table 2.1 - Simulation Parameters:**

|  |  |
| --- | --- |
| Network Topology | Point – to – Point |
| Number of Nodes | 2 |
| Data Rate | 10Mbps |
| Delay | 2ms |
| Transport Protocol | UDP |

**Output:**

****

**Figure 2.1**

**Animation Interface:**

**Figure 2.2**

**Result:**

The successful creation of a point-to-point connection with specified data rate and delay.

|  |  |
| --- | --- |
| **Ex. No: 3** | **Implementation of Wired Network Topology Using**  **LAN Connectivity.** |
| **Date: 07.09.23** |

**Aim:**

To implement and evaluate a point-to-point and CSMA-based wired network simulation using NS-3, assessing the performance and communication between nodes.

**Network Topology:**

10.1.1.0

n0 --------------------- n1 n2 n3 n4

point-to-point | | | |

==============

LAN 10.1.2.0

**Algorithm:**

Step 1: Start

Step 2: Import necessary NS-3 libraries.

Step 3: Command line configuration:

Step 4: Define and give values for command line parameters (nCsma ,verbose).

Step 5: Set up command line arguments parsing.

Step 6: If (verbose is true):

Enable logging for UdpEchoClient and UdpEchoServerApplication.

Step 7: Configure the number of extra CSMA nodes to be at least 1.

Step 8: Create a nodes for point-to-point and csma network:

Step 9: Setup a p2p and csma channel with a appropriate datarate and delay.

Step 10: Install the Internet stack on the first point-to-point node and

set CSMA nodes.

Step 11: Assign IPv4 addresses to the network interface.

Step 12: UdpEchoServer application:

Step 13: Install and configure the UdpEchoServer application.

Step 14: UdpEchoClient application:

Step 15: Install and configure the UdpEchoClient application.

Step 16: Populate global routing tables.

Step 17: Enable packet capture for point-to-point and csma devices.

Step 18: Animation Interface:

Step 19: Create an AnimationInterface and get output in XML file name.

Step 20: Set the initial positions of nodes using SetConstantPosition.

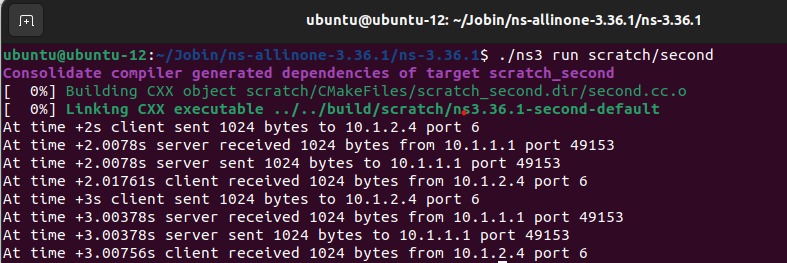
Step 21: Run the Simulation.

Step 22: Destroy the Simulator

Step 23: Stop

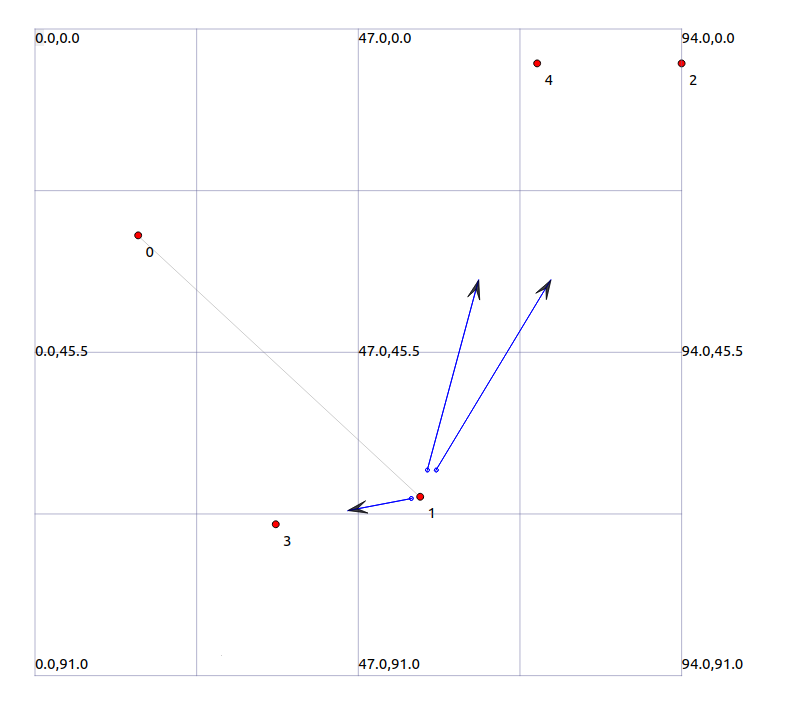
**Table:3.1 - Simulation Parameters:**

|  |  |
| --- | --- |
| Network Topology | combination of a point-to-point link and a CSMA (Ethernet) LAN |
| Number of Nodes | 5 (2 nodes for point-to-point  3 nodes for CSMA) |
| Data Rate | 20Mbps for point-to-point  100Mbps for CSMA |
| Delay | 2ms for point-to-point  6560ns for CSMA |
| UDP Echo Client | |
| Max Packets | 2 |
| Interval | 0.5s |
| Packet Size | 1024 |
| Simulation Time | 10.0s |
| UDP Echo Server Port | 6 |
| Pcap filename | Second-0-0 |

**Ou****tput:**

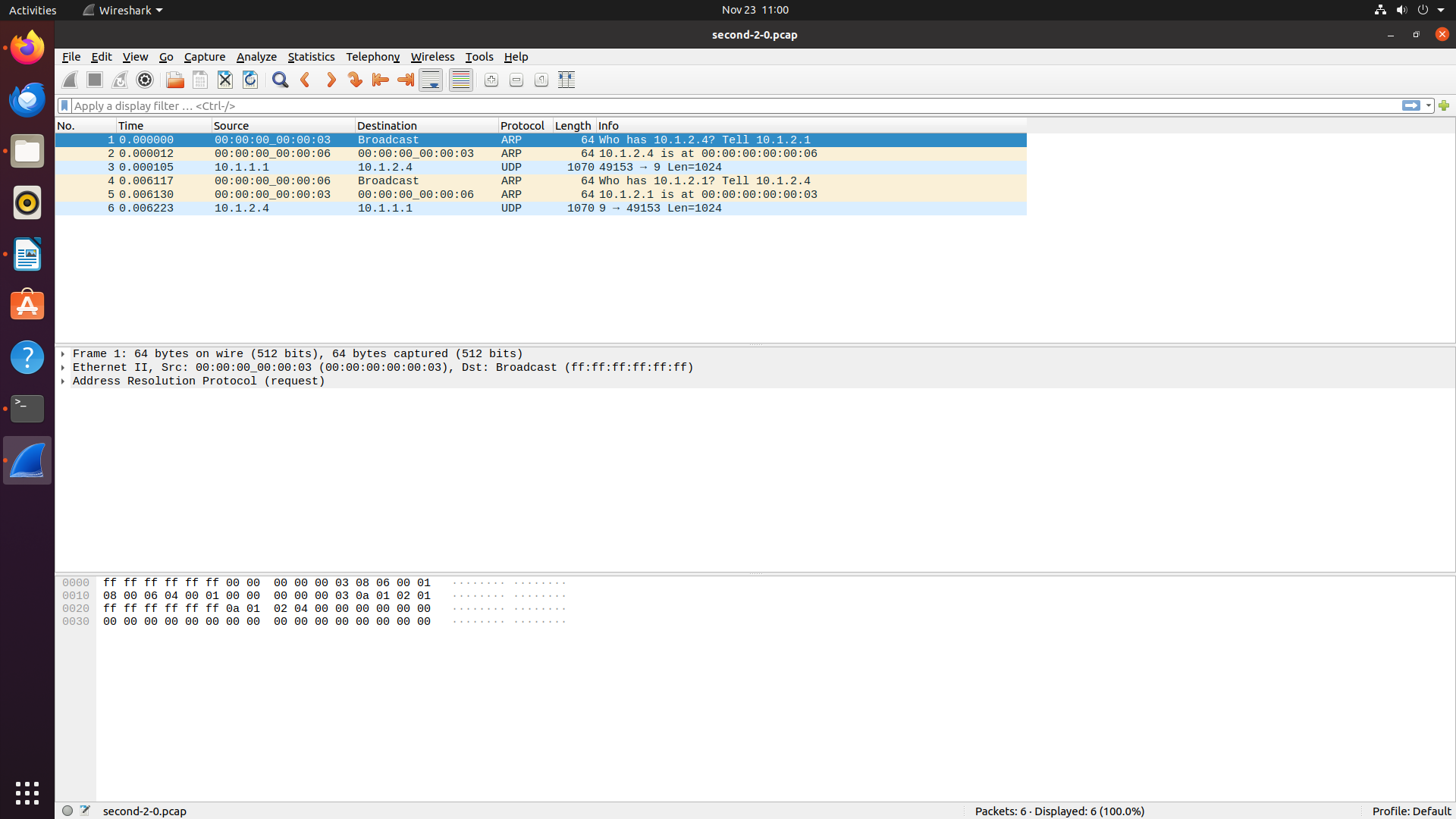
**Figure3.1**

**Animation Interface:**



**Figure 3.2**

**Pcap:**

****

**Figure3.3**

**Result:**

The successful creation of a simulated wired network with point-to-point and CSMA links, demonstrating reliable communication and capturing key performance metrics in a controlled environment.

|  |  |
| --- | --- |
| **Ex. No: 4** | **Implementation of Wireless Network topology Using**  **Wi-Fi Connectivity.** |
| **Date:14.09.23** |

**Aim:**

To implement a wireless network topology using Wi-Fi connectivity, simulating a scenario where multiple wireless devices communicate with each other through an Access Point (AP) and to evaluate the performance of the Wi-Fi network in terms of data transfer, signal strength, and overall network efficiency.

**Network Topology:**

Wifi 10.1.3.0

AP

\* \* \* \*

| | | | 10.1.1.0

n5 n6 n7 n0 -------------------- n1 n2 n3 n4

point-to-point | | | |

============

LAN 10.1.2.0

**Algorithm:**

Step 1: Start

Step 2: Import necessary NS-3 libraries.

Step 3: Command line configuration:

Step 4: Define and give values for command line parameters for

(nCsma, nWifi, verbose).

Step 5: Set up command line argument parsing.

Step 6: If (nWifi>18):

It show between the limit,otherwise it exceeds mobility boundary.

Step 7: If (verbose is true):

Enable logging for UdpEchoClient and UdpEchoServerApplication.

Step 8: Configure the number of extra CSMA nodes (nCsma) to be at least 1.

Step 9: Create nodes for point-to-point and CSMA network:

Step 10: Set up a Point-to-Point and CSMA communication channel with

appropriate datarate and delay.

Step 11: Install the Internet stack on the first p2p node and the set of CSMA nodes.

Step 12: Assign IPv4 addresses to the network interface.

Step 13: Setup WiFi Nodes:

Step 14: Create 3 WiFi STA nodes and an AP node.

Step 15: Configure WiFi channel and physical layer settings

Step 16: Use grid and random walk mobility models for STAs.

Step 17: Install applications:

Step 18: Install UdpEchoServer on the CSMA node with ID nCsma.

Step 19: Install UdpEchoClient on the last WiFi STA node with ID nWifi - 1

Step 20: Configure UDP Echo application parameters for the client

(like : MaxPackets, Interval, PacketSize).

Step 21: Populate global routing tables and Enable packet Capture:

Step 22: Enable packet capture for point-to-point and CSMA and WiFi devices.

Step 23: Animation Interface:

Step 24: Create an AnimationInterface and get output in XML file name.

Step 25: Set the initial positions of nodes using SetConstantPosition.

Step 26: Run the Simulation

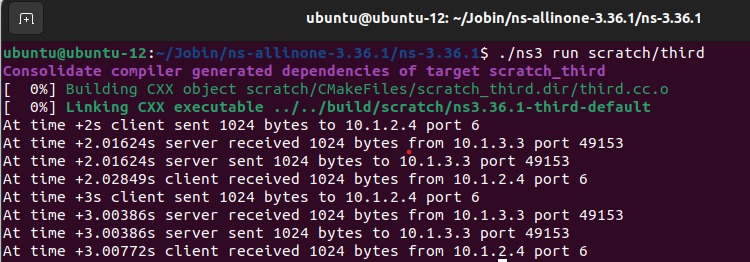
Step 27: Destroy the Simulator.

Step 28: Stop

**Table 4.1 - Simulation Parameters:**

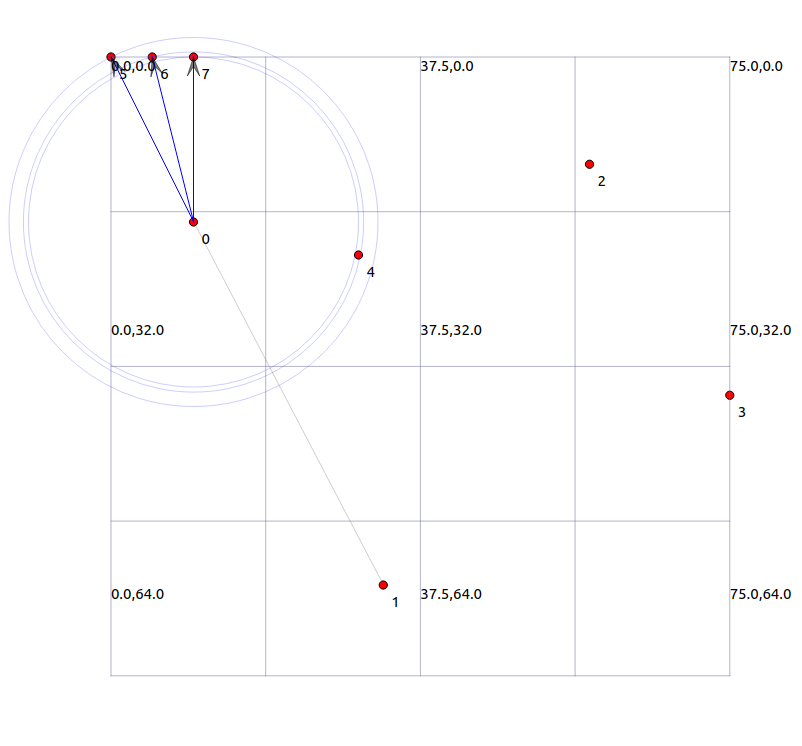
|  |  |
| --- | --- |
| Network Topology | combination of a point-to-point link and a CSMA and Wi-Fi |
| Number of Nodes | 8 (2 nodes for point-to-point, 3 nodes for nCSMA and 3 for nWiFi) |
| Data Rate | 25Mbps for point-to-point  150Mbps for CSMA |
| Delay | 5ms for point-to-point  6560ns for CSMA |
| Transport Protocol | UDP |
| IPv4 Addressing | point-to-point link (10.1.1.0),  CSMA network (10.1.2.0),  WiFi network (10.1.3.0). |
| UDP Echo Client | |
| Max Packets | 2 |
| Interval | 2.0s |
| Packet Size | 1024 |
| Simulation Time | 15.0s |
| UDP Echo Server Port | 6 |

**Output:**

****

**Figure 4.1**

**Animation Interface:**

 **Figure 4. 2**

**Result:**

The simulation successfully created a wireless network with devices communicating through an Access Point, showcasing the robust performance and effectiveness of the simulated Wi-Fi network.

|  |  |
| --- | --- |
| **Ex. No: 5** | **Analysing TCP Congestion Window and Packet Tracing in Point-to-Point Network .** |
| **Date:21.09.23** |

**Aim:** To evaluate changes in the TCP congestion window over time and analyze packet drops in a simulated network..

**Network Topology:**

Point-to-Point

TCP

Node 0 Node 1

(10.1.1.1) (10.1.1.2)

**Algorithm:**

Step 1: Start:

Step 2: Import necessary NS-3 libraries.

Step 3: Command line configuration:

Step 4: Add a command line option for using IPv6 (useIpv6).

Step 5: Parse command line arguments.

Step 6: Create nodes and point-to-point link:

Step 7: Create two nodes and set up a point-to-point link between them with specific a datarate and delay.

Step 8: Install Internet stack on the nodes and configure addresses:

Step 9: If not using IPv6:

Set IPv4 addresses on the devices.

Step10: If using IPv6:

Set IPv6 addresses on the devices.

Step 11: TCP packet sink on node 1:

Step 12: Set up a TCP packet sink on node 1 listening on a specified port (8080).

Step 13: Record congestion window changes of the sender's socket.

Step 14: TCP application on node 0:

Step 15: Create a TCP socket on node 0 and Set up a custom application

Step 16: Configure the application to use the created socket, target address , . packet size and data rate.

Step 17: Trace and record events:

Step 18: Trace congestion window changes of the TCP socket and save to a file

(seventh.cwnd).

Step 19: Trace PHY layer RX drops and save packetsdrops to a pcap file

(seventh.pcap).

Step 20: save to a file packet byte count over time to a formatted file

(seventh-packet-byte-count).

Step 21: Animation Interface:

Step 22: Create an AnimationInterface and get output in XML file name.

Step 23: Run the simulation:

Step 24: Set the simulation duration to 20 seconds.

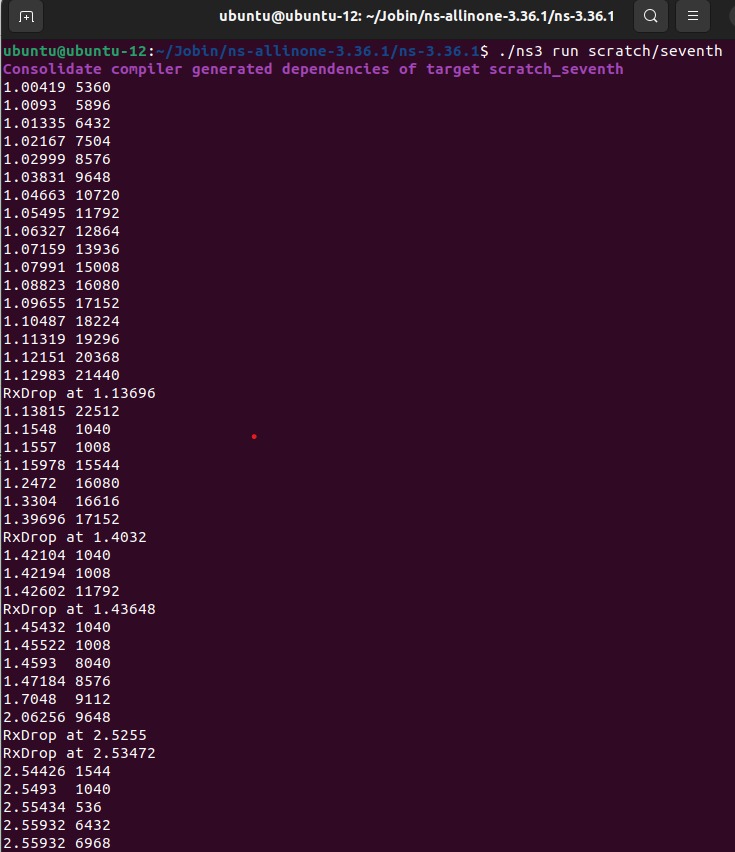
Step 25: Destroy the Simulator.

Step 26: Stop.

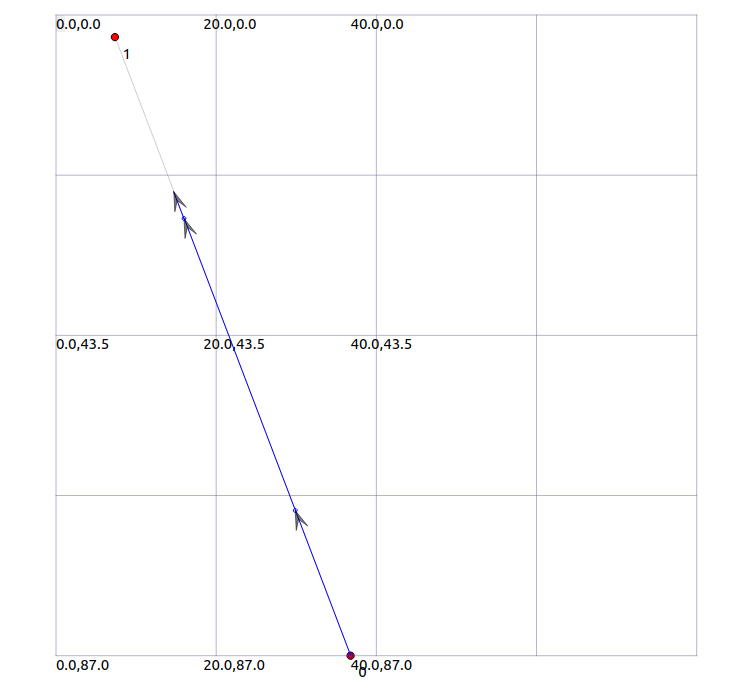
**Table 5.1 - Simulation Parameters:**

|  |  |
| --- | --- |
| Network Topology | Point-to-Point link |
| No. of .Nodes | 2 |
| Data Rate and Delay | 5Mbps and 2ms |
| Error Rate on second device | 0.00001 |
| TCP Application Parameter | |
| Sink port | 8080 |
| Start time | 1 seconds |
| Stop time | 20 seconds |
| Simulation Time | 20.0 seconds |
| Congestion window tracing | seventh.cwnd |
| PHY RX Drop | seventh.pcap |
| Packet Byte count | seventh-packet-byte-count |

**Output:**

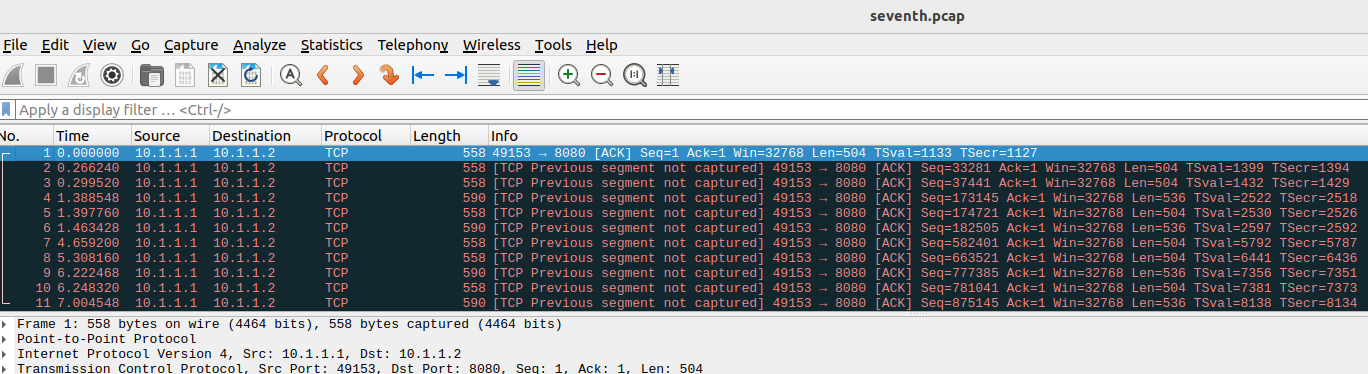
 **Figure 5.1**

**Animation Interface:**

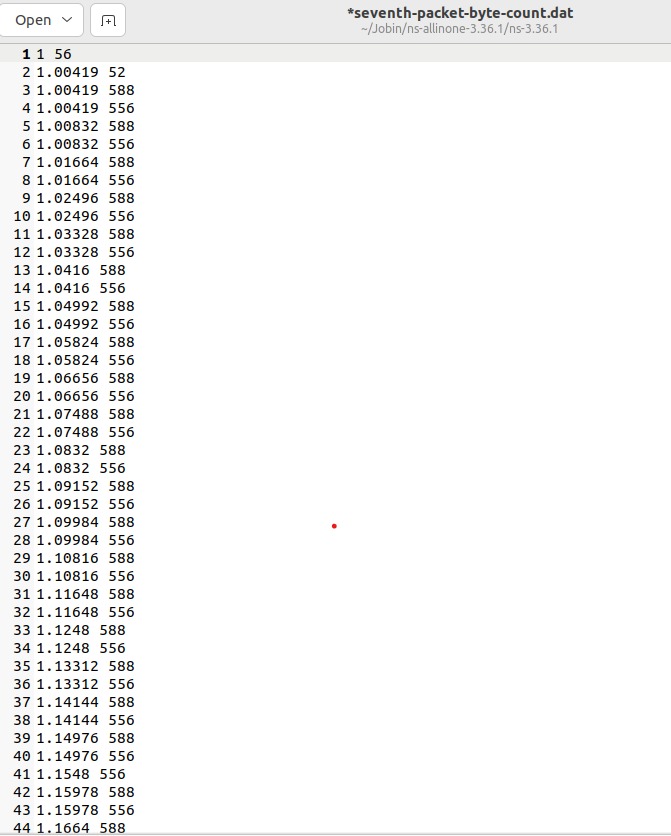
****

**Figure 5.2**

**Pcap:**

**Figure 5.3**

**Packet-byte-count**

 **Figure 5.4**

**Result:**

The simulation successfully provides insights into congestion control mechanisms, visualized through congestion window variations and packet drop analysis in a point-to-point TCP.

|  |  |
| --- | --- |
| **Ex. No: 6** | **Implementation of OLSR Routing Protocol in**  **a Point-to-Point Network Using NS-3.** |
| **Date: 05.10. 23** |

**Aim:** To Investigate the efficiency of the OLSR routing protocol in a basic point-to-point network, measuring packet delivery and routing performance.

**Network Topology:**

N0 10.1.1. 0 Point-to-Point - UDP

\

\ 10.1.3.0

n2 -------------------------n3----------------------n4

/ 10.1.4.0

/

n1 10.1.2.0

**Algorithm:**

Step 1: Start

Step 2: Import necessary NS-3 libraries.

Step 3: Set up default values and allow command-line argument.

Step 4: Create a node container with 5 nodes (Node 0 to Node 4)

Step 5: Use OLSR routing protocol.

Step 6: Create Point-to-Point Channels:

Step 7: Create channels between nodes with specified data rates and delays.

Step 8: Assign IP addresses to each interface on the nodes.

Step 9: Create Applications:

Step 10: Create OnOff applications to send UDP datagrams from Node 0 to Node 4

and from Node 3 to Node 1.

Step 11: Set constant data rate and start/stop times.

Step 12: Create Packet Sink:

Step 13: Create packet sinks to receive the UDP packets at the destination nodes

Step 14: Enable Tracing:

Step 15: Enable ASCII and pcap tracing for Point-to-Point channels.

Step 16: Create an animation interface for visualization.

Step 17: Run Simulation:

Step 18: Set simulation time to 30 seconds.

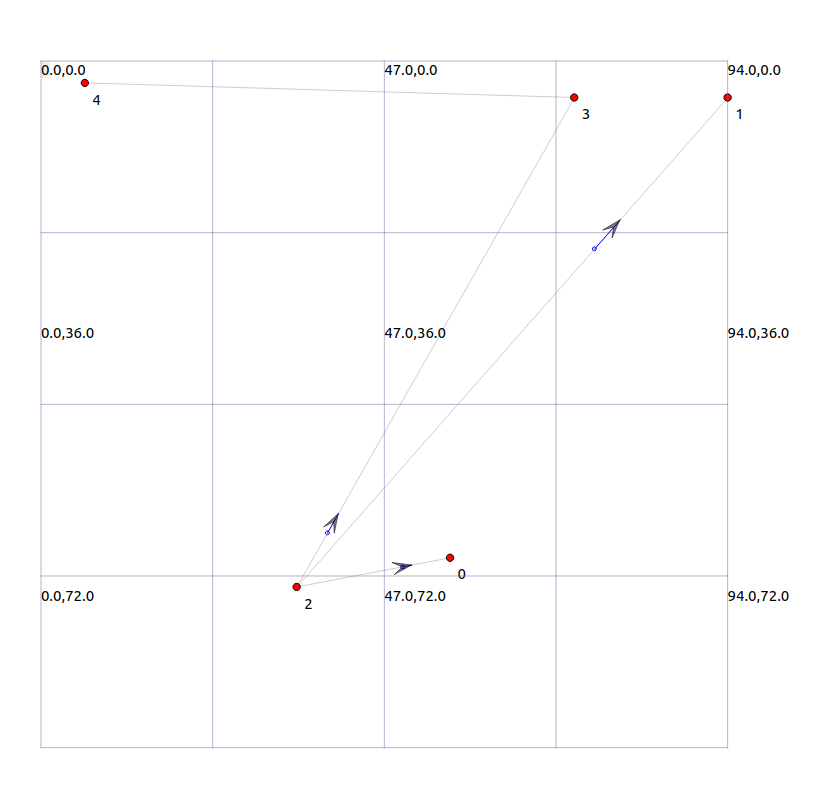
Step 19: Destroy Simulator:

Step 20: Stop

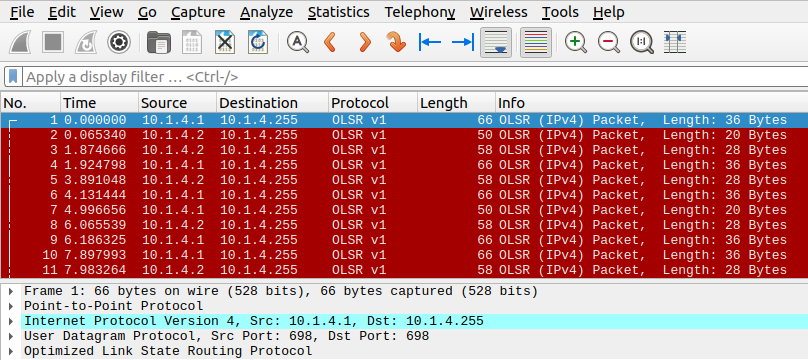
**Table 6.1 - Simulation Parameters:**

|  |  |
| --- | --- |
| Network Topology | Point-to-Point link |
| No. of .Nodes | 5 |
| Data Rate and Delay(node : 0 -2) | 5Mbps and 2ms |
| Data Rate and Delay(node : 3 -2) | 1.5Mbps and 10ms |
| IP Address | |
| Nodes 0-2 | 10.1.1.0 / 24 |
| Nodes 1-2 | 10.1.2.0 / 24 |
| Nodes 3-2 | 10.1.3.0 / 24 |
| Nodes 4-2 | 10.1.4.0 / 24 |
| Off Application Parameter(UDP) | |
| Packet Size | 210 bytes |
| State time | 1 sec |
| Stop time | 10.0 seconds |
| Date Rate | 448 kbps |
| Simulation Time | 30.0 seconds |
| ASCII and Pcap Tracing | |
| Tracing of ques and packet reception | Olsr.tr |
| Pcap tracing | olsr.pcap |

**Animation Interface:**

**Figure 6.1**

**Pcap file:**

****

**Figure 6. 2**

**ASCII Tracing(.tr file):**

 **Figure 6. 3**

**Result:**

The simulation successfully demonstrates the OLSR protocol's effectiveness in optimizing routing and improving packet delivery in the specified network topology, visualized through network animation and trace files.