NetSim®

Accelerate Network R & D

Experiments Manual

A Network Simulation & Emulation Software

By

TETCOS
LLP

Contact us at
TETCOS LLP
214, 39th A Cross, 7th Main, 5th Block Jayanagar,
Bangalore - 560 041, Karnataka, INDIA. Phone: +91 80 26630624
E-Mail: sales@tetcos.com
Visit: www.tetcos.com

TABLE OF CONTENTS

1 Introduction to Network simulation and NetSim

1.1 Introduction to NetSim (Level 1)

1.1.1 Simulation environment and workflow

NetSim is a network simulation tool that allows you to create network scenarios, model traffic, design protocols and analyze network performance. Users can study the behavior of a network by test combinations of network parameters. The various network technologies covered in NetSim include:

- Internetworks - Ethernet, WLAN, IP, TCP
- Legacy Networks - Aloha, Slotted Aloha
- Cellular Networks - GSM, CDMA
- Mobile Adhoc Networks - DSR, AODV, OLSR, ZRP
- Wireless Sensor Networks - 802.15.4
- Internet of Things - 6LoWPAN gateway, 802.15.4 MAC / PHY, RPL
- Cognitive Radio Networks - 802.22
- Long-Term Evolution Networks – LTE
- Software Defined Networking
- Advanced Routing and Switching - VLAN, IGMP, PIM, L3 Switch, ACL and NAT
- 5G NR - LTE NR
- UWAN – 802.11

The NetSim home screen is as shown below see Figure 1-1. Click on the network type you wish to simulate.
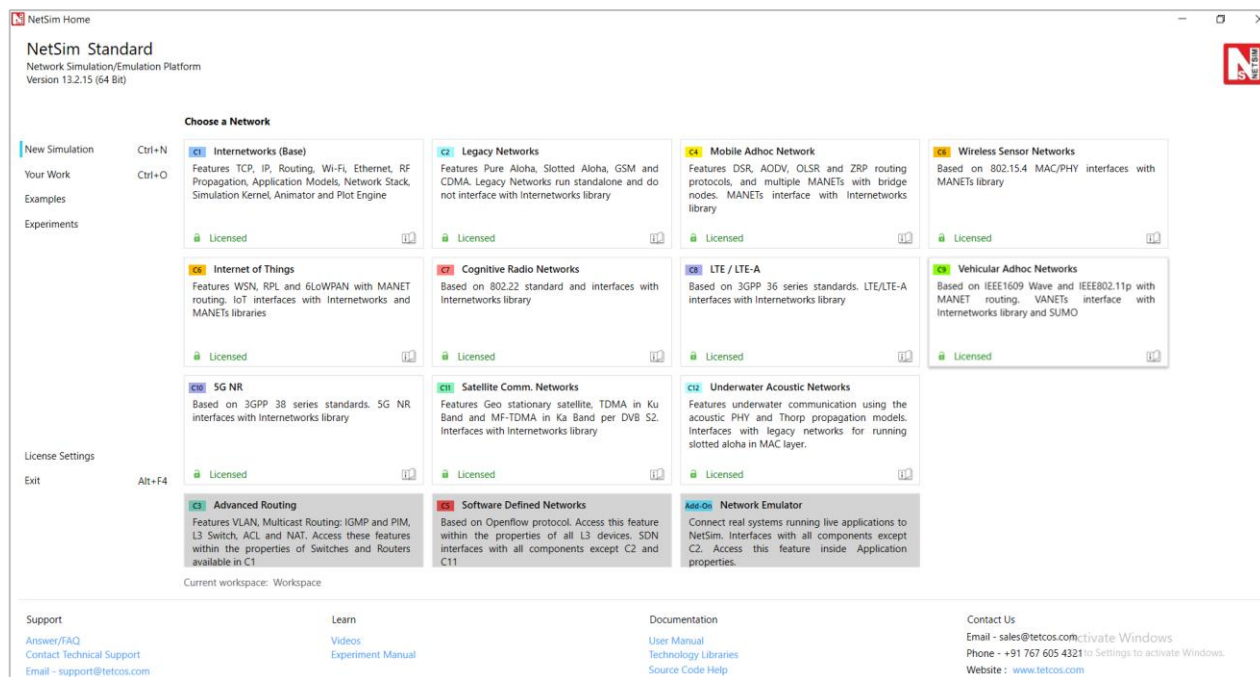
Figure 1-1: NetSim Home Screen

Network Design Window: A user would enter the design window upon selecting a network type in the home screen. The NetSim design window GUI see Figure 1-2. It enables users to model a network comprising of network devices like switches, routers, nodes, etc., connect them through links, and model application traffic to flow through the network. The network devices shown in the palette are specific to the network technologies chosen by the user.
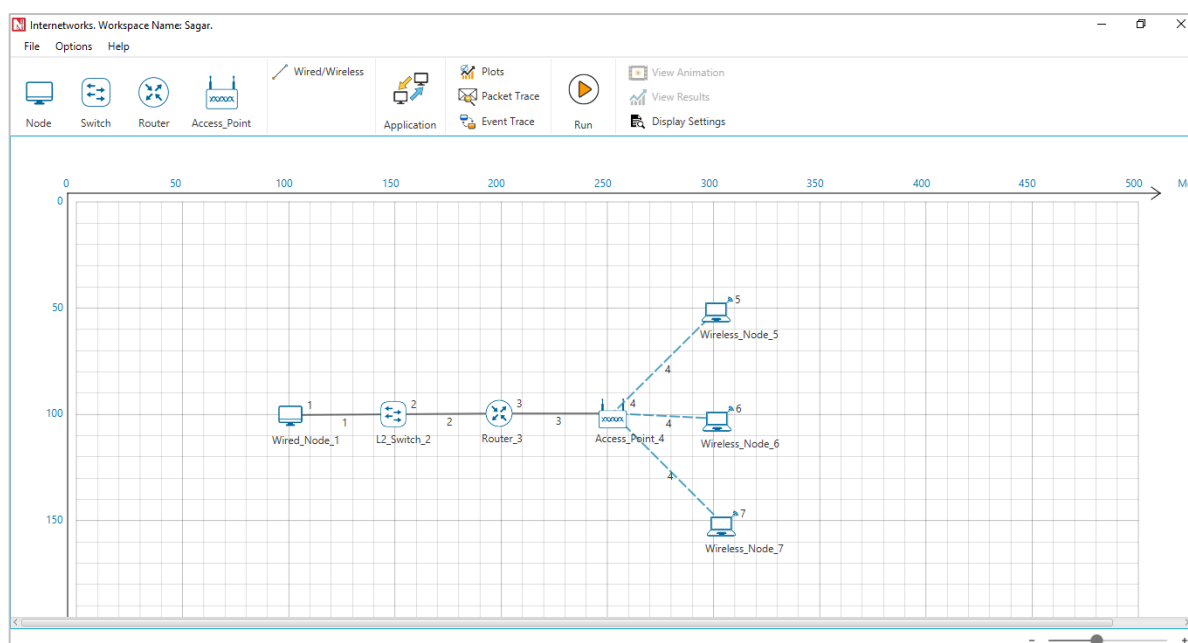
Figure 1-2: Network Design Window

Description:

1. **File** - In order to save the network scenario before or after running the simulation into the current workspace,

- Click on File → Save to save the simulation inside the current workspace. Users can specify their own Experiment Name and Description (Optional).
- Click on File → Save As to save an already saved simulation in a different name after performing required modifications to it.
- Click on Close, to close the design window or GUI. It will take you to the home screen of NetSim.

2. **Options** - Go to Options → Grid/Map Settings and choose the type of environment. Here we have chosen the Grid/Map in the form of a Grid. Map option can be used for specific cases like while designing VANET scenarios.

3. **Help** - Help option allows the users to access all the help features.

   - **Video Tutorials** – Assists the users by directing them to our dedicated YouTube Channel **"TETCOS"**, where we have lots of video presentations ranging from short to long, covering different versions of NetSim up to the latest release.

   - **Answers/FAQ** – Assists the user by directing them to our **"NetSim Support Portal"**, where one can find a well-structured **"Knowledge Base"**, consisting of answers or solutions to all the commonest queries which a new user can go through.

   - **Raise a Support Ticket** – Assists the user by directing them to our **"NetSim Support Portal"**, where one can **"Submit a ticket"** or in other words raise his/her query, which reaches our dedicated Helpdesk and due support will be provided to the user.

   - **User Manual** – Assists the user with the usability of the entire tool and its features. It highly facilitates a new user with lots of key information about NetSim.

   - **Source Code Help** – Assists the user with a structured documentation for **"NetSim Source Code Help"**, which helps the users who are doing their R&D using NetSim with a structured code documentation consisting of more than 5000 pages with very much ease of navigation from one part of the document to another.

   - **Open-Source Code** – Assists the user to open the entire source codes of NetSim protocol libraries in Visual Studio, where one can start initiating the debugging process or performing modifications to existing code or adding new lines of code. Visual Studio Community Edition is a highly recommended IDE to our users who are using the R&D Version of NetSim.

   - **Experiments** – Assists the user with separate links provided for 30+ different experiments covering almost all the network technologies present in NetSim.

   - **Technology Libraries** – Assists the user by directing them to a folder comprising of individual technology library files comprising all the components present in NetSim.

Below the menu options, the entire region constitutes the Ribbon/Toolbar using which the following actions can be performed:

- Click and drop network devices and right click to edit properties.

- Click on Wired/Wireless links to connect the devices to one another. It automatically detects whether to use a Wired/Wireless link based on the devices we are trying to connect.
- Click on Application to configure different types of applications and generate traffic.
- Click on Plots, Packet Trace, and Event Trace and click on the enable check box option which appears in their respective windows to generate additional metrics to further analyze the network performance.
- Click on Run to perform the simulation and specify the simulation time in seconds.
- Next to Run, we have View Animation and View Results options. Both the options remain hidden before we run the simulation or if the respective windows are already open.
- Display Settings option is mainly used to display various parameters like Device Name, IP, etc., to provide a better understanding especially during the design and animation.

Results Window: Upon completion of simulation, Network statistics or network performance metrics reported in the form of graphs and tables. The report includes metrics like throughput, simulation time, packets generated, packets dropped, collision counts etc. see Figure 1-3 and Figure 1-4.
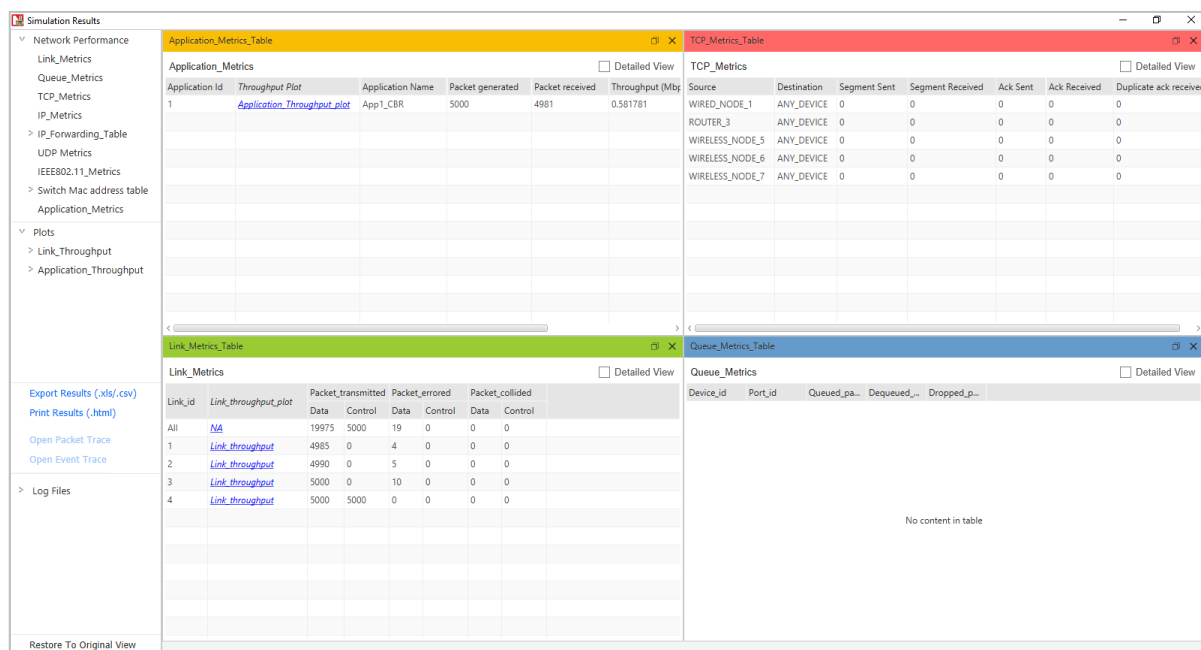
Figure 1-3: Results Window
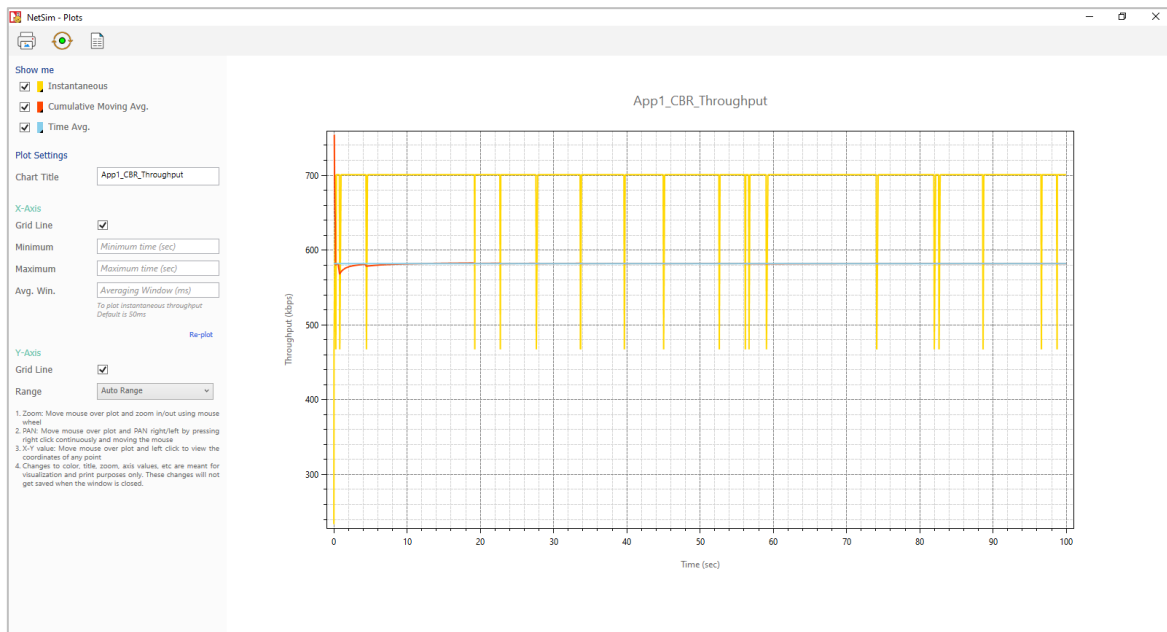
Figure 1-4: Application Throughput Plot

Description:

1. Below Simulation Results, clicking on a particular metrics will display the respective metrics window.

2. Clicking on links in a particular metrics will display the plot in a separate window.

3. Enabling Detailed View by clicking on it will display the remaining properties.

4. Clicking on Restore to Original View will get back to the original view.

5. Click on Open Packet Trace / Open Event Trace to open the additional metrics which provide in depth analysis on each Packets / Events.

Packet Animation Window: When we click on run simulation, we have the option to record / play & record animation. If this is enabled, users can view the animation during the run time or upon completion of the simulation see Figure 1-5, users can see the flow of packets through the network. Along with this, more than 25+ fields of packet information is available as a table at the bottom. This table contains all the fields recorded in the packet trace. In addition, animation options are available for viewing different graphs, IP Addresses, Node movement etc.
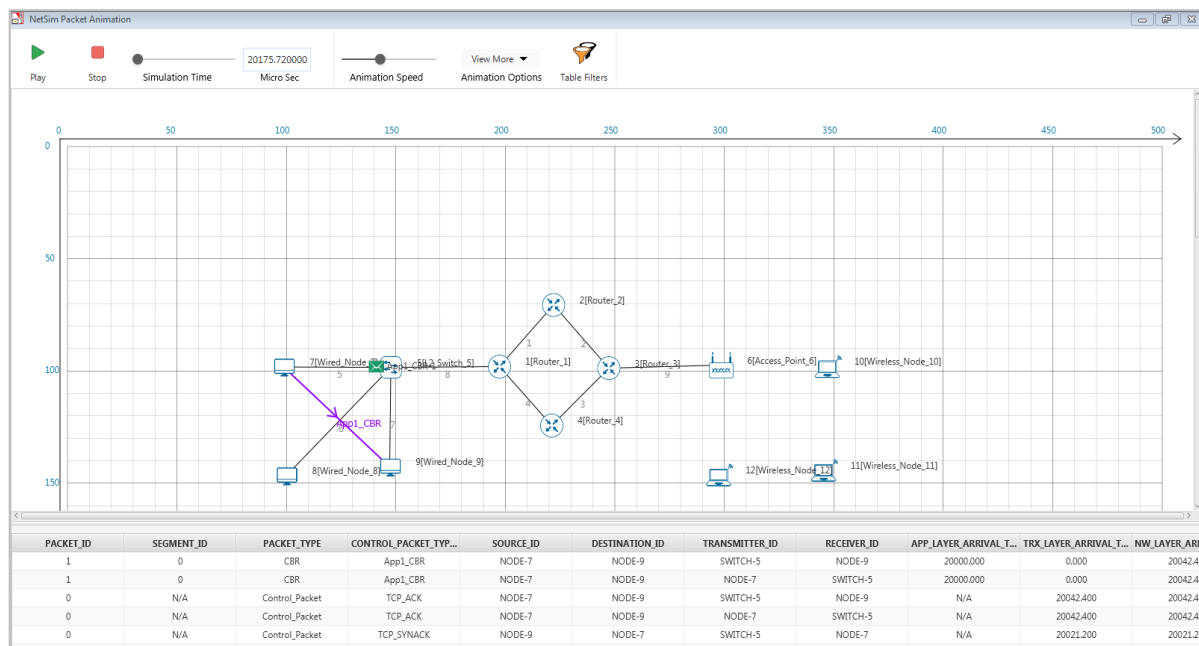
Figure 1-5: Packet Animation Window

1. Click on Play to view the animation. You can Pause the animation at any interval and play again.
2. Click on Stop to stop the animation. Now click on Play to start the animation from the beginning.
3. Next to that we also have speed controllers to increase/decrease Simulation Time and Animation Speed
4. View More option enables the user to view Plots, Throughputs, and IP Tables during the animation
5. Table Filters are used to filter the packet information's shown in the below table during simulation as per user requirement.
6. While setting more than one application, it is differentiated using different color indications.
7. Packets are indicated using different color combinations say, blue color indicates control packets, green color indicates data packets and red color indicates error packets.

1.1.2 How does a user create and save an experiment in workspace?

To create an experiment, select New Simulation-> <Any Network> in the NetSim Home Screen Figure 1-6.
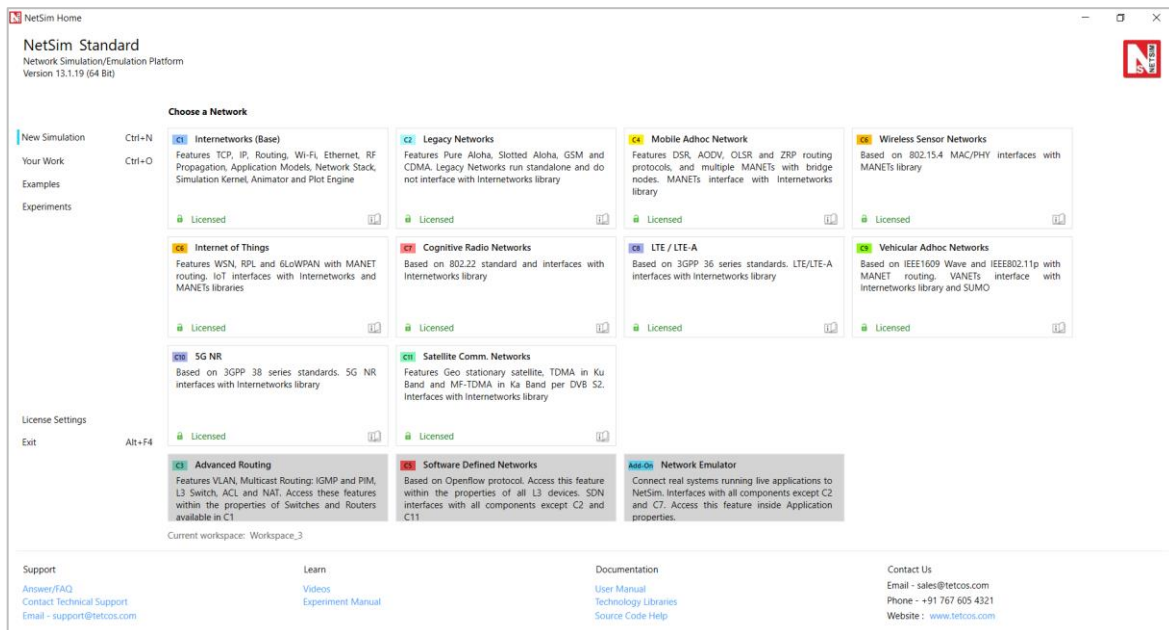
Figure 1-6: NetSim Home Screen

Create a network and save the experiment by clicking on File->Save button on the top left.
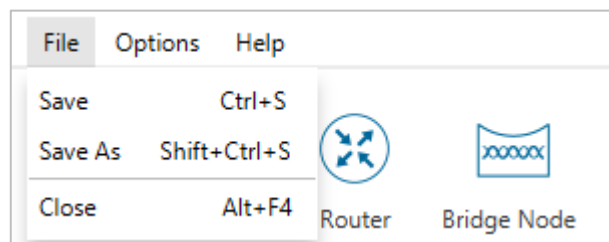
Figure 1-7: Save the network using file option

A save popup window appears which contains Experiment Name, Workspace path and Description see Figure 1-8.
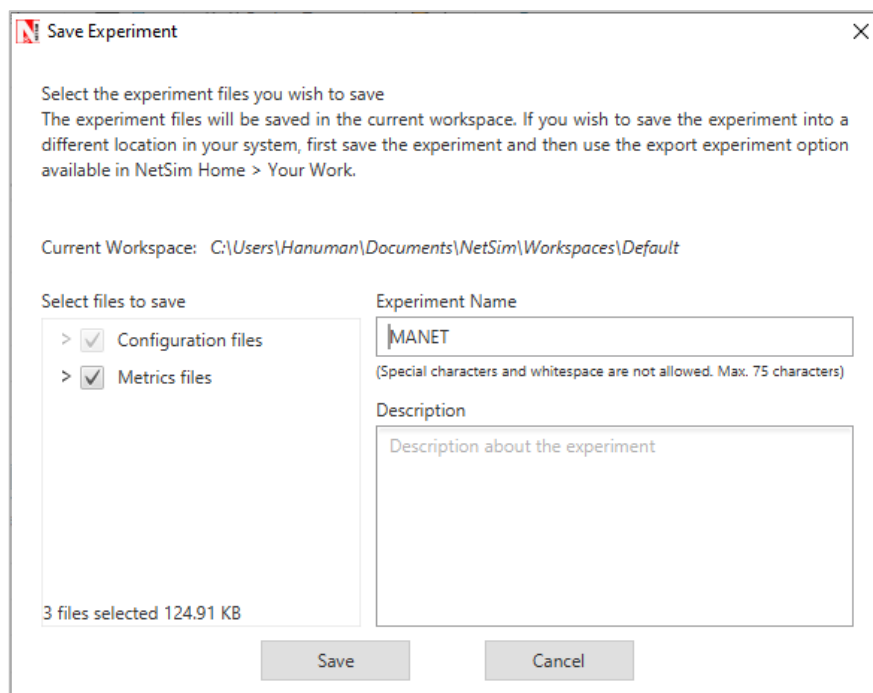
Figure 1-8: NetSim Save Window

Specify the Experiment Name and Description (Optional) and then click on Save. The workspace path is non-editable. Hence all the experiments will be saved in the default workspace path. After specifying the Experiment Name click on Save.

In our example we saved with the name MANET and this experiment can be found in the default workspace path see below Figure 1-9.
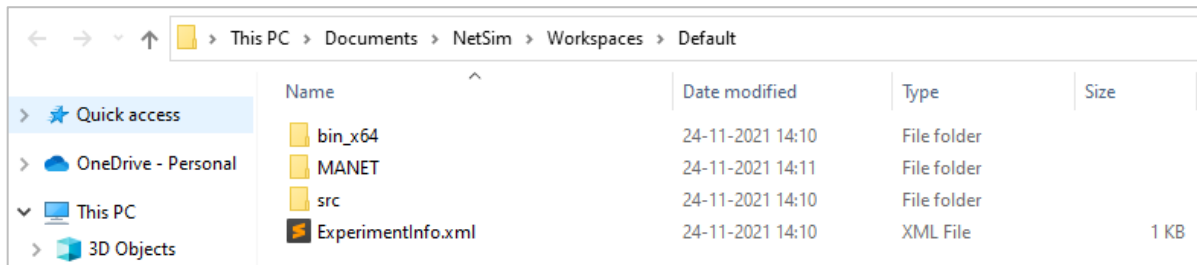
Figure 1-9: NetSim Default Workspace Path

Users can also see the saved experiments in Your work menu shown below Figure 1-10.
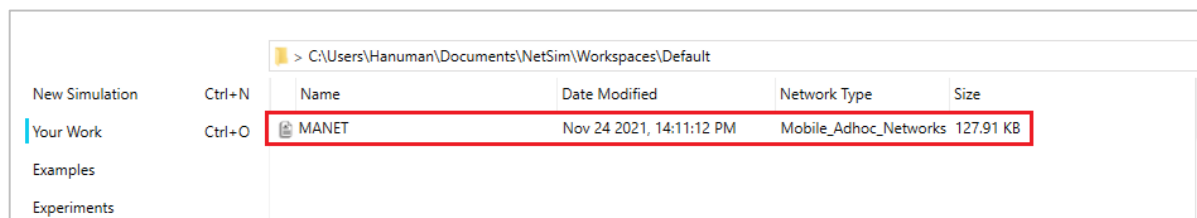
Figure 1-10: Your Work Menu

"Save As" option is also available to save the current experiment with a different name.

1.1.3 Typical sequence of steps to perform the experiments provided in this manual

The typical steps involved in doing experiments in NetSim are:

- **Network Set up:** Drag and drop devices and connect them using wired or wireless links.
- **Configure Properties:** Configure device, protocol, or link properties by right clicking on the device or link and modifying parameters in the properties window.
- **Model Traffic:** Click on the Application icon present on the ribbon and set traffic flows.
- **Enable Trace/Plots (optional):** Click on packet trace, event trace and Plots to enable. Packet trace logs packet flow, event trace logs each event (NetSim is a discrete event simulator) and the Plots button enables charting of various throughputs over time.
- **Save/Save As/Open/Edit:** Click on File → Save / File → Save As to save the experiments in the current workspace. Saved experiments can then opened from NetSim home screen to run the simulation or to modify the parameters and again run the simulation.
- **View Animation/View Results:** Visualize through the animator to understand working and to analyze results and draw inferences.

NOTE: Example Configuration files for all experiments would available where NetSim has been installed. This directory is

(<NetSim_Install_Directory>\Docs\Sample_Configuration\NetSim_Experiment_Manual)

1.2 Understand the working of basic networking commands - Ping, Route Add/Delete/Print, ACL (Level 1)

1.2.1 Theory

NetSim allows users to interact with the simulation at runtime via a socket or through a file. User Interactions make simulation more realistic by allowing command execution to view/modify certain device parameters during runtime.

1.2.1.1 Ping Command

- The ping command is one of the most often used networking utilities for troubleshooting network problems.
- You can use the ping command to test the availability of a networking device (usually a computer) on a network.
- When you ping a device, you send that device a short message, which it then sends back (the echo)
- If you receive a reply then the device is in the Network, if you do not, then the device is faulty, disconnected, switched off, or incorrectly configured.

1.2.1.2 Route Commands

You can use the route commands to view, add and delete routes in IP routing tables.

- **route print:** In order to view the entire contents of the IP routing table.
- **route delete:** In order to delete all routes in the IP routing table.
- **route add:** In order to add a static TCP/IP route to the IP routing table.

1.2.1.3 ACL Configuration

Routers provide basic traffic filtering capabilities, such as blocking the Internet traffic with access control lists (ACLs). An ACL is a sequential list of **Permit** or **Deny** statements that apply to addresses or upper-layer protocols. These lists tell the router what types of packets to: **PERMIT** or **DENY**. When using an access-list to filter traffic, a PERMIT statement is used to **"allow"** traffic, while a DENY statement is used to **"block"** traffic.

1.2.2 Network setup

Open NetSim and click on **Experiments> Advanced Routing> Basic networking commands Ping Route Add/Delete/Print and ACL** then click on the tile in the middle panel to load the example as shown in below Figure 1-11.
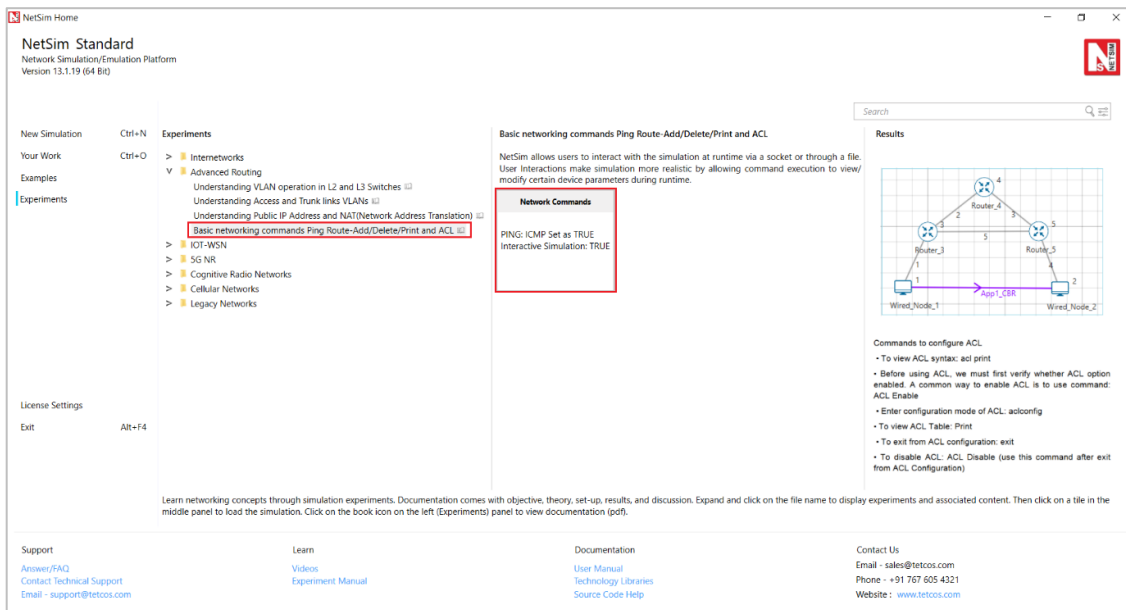
Figure 1-11: List of scenarios for the example of Basic networking commands Ping Route Add/Delete/Print and ACL

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 1-12.
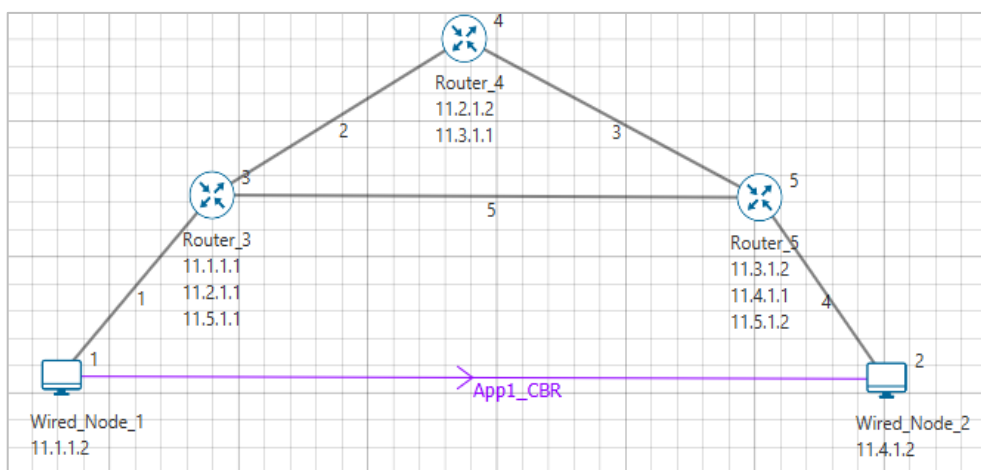
Figure 1-12: Network set up for studying the Basic networking commands Ping Route Add/Delete/Print and ACL

1.2.3 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 3 Routers in the **"Internetworks"** Network Library.

Step 2: In the Network Layer properties of Wired Node 1, **"ICMP Status"** is set as TRUE.

Similarly, ICMP Status is set as TRUE for all the devices as shown Figure 1-13.
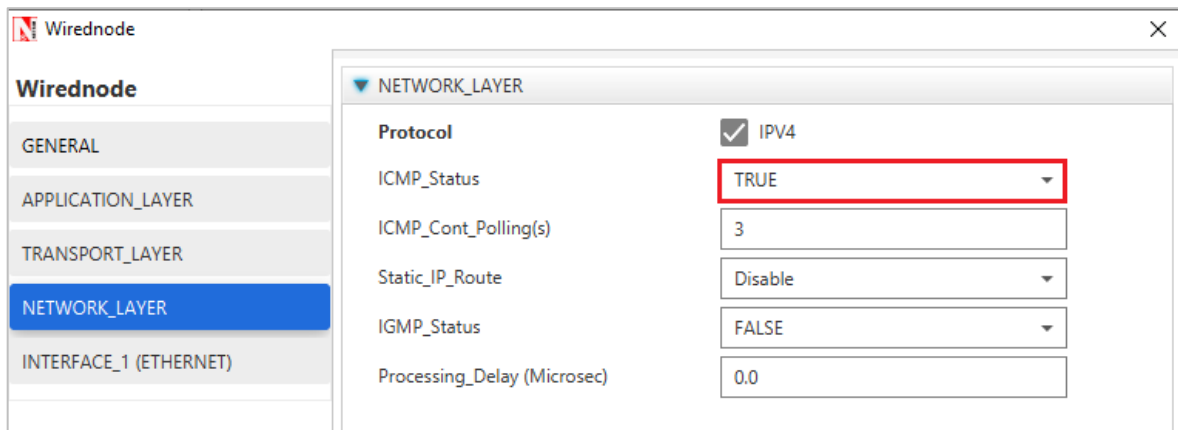
Figure 1-13: Network Layer properties of Wired Node 1

Step 3: In the General properties of Wired Node 1, **Wireshark Capture** is set as Online.

Step 4: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 1 i.e., Source to Wired Node 2 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 233.6µs. Transport Protocol is set to **UDP**.

Additionally, the **"Start Time(s)"** parameter is set to 30, while configuring the application. This time is usually set to be greater than the time taken for OSPF Convergence (i.e., Exchange of OSPF information between all the routers), and it increases as the size of the network increases.

Step 5: Packet Trace is enabled in NetSim GUI. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis. Plots are enabled in NetSim GUI.

Step 6: Click on Run Simulation. Simulation Time is set to 300 Seconds and in the **Runtime Interaction** tab Figure 1-14, Interactive Simulation is set to True.
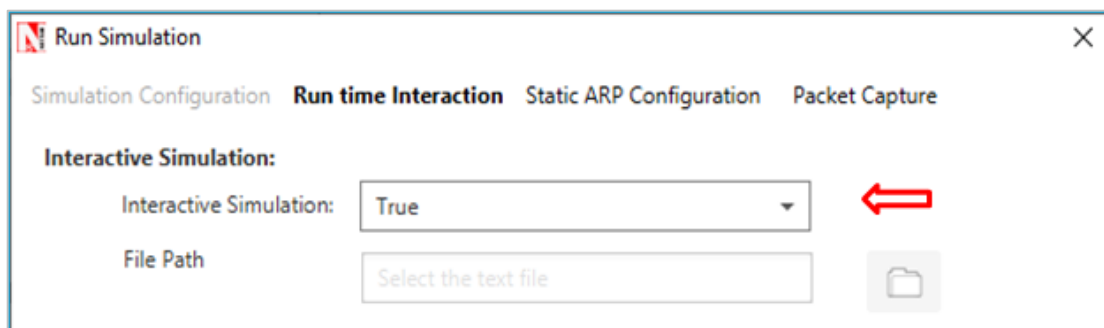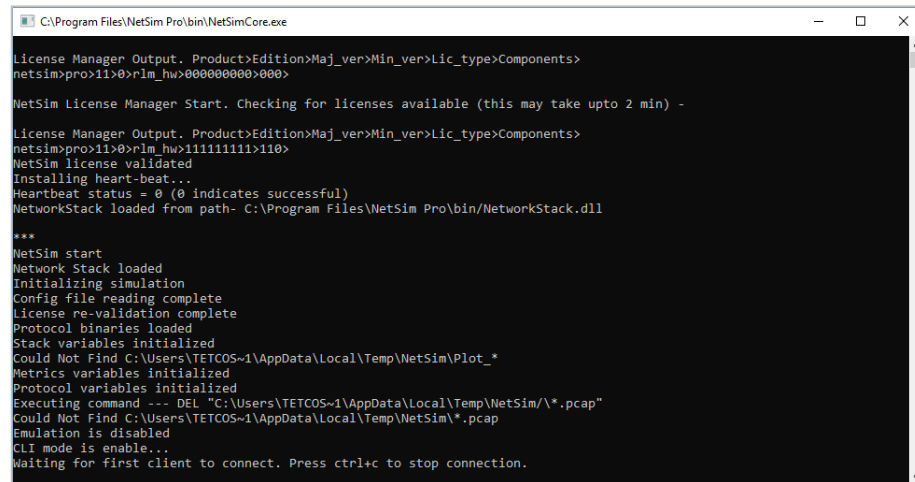
Figure 1-14: Runtime Interaction window

NOTE: It is recommended to specify a longer simulation time to ensure that there is sufficient time for the user to execute the various commands and see the effect of that before the Simulation ends.

Click on **Accept** and then click on **OK**.

- Simulation (NetSimCore.exe) will start running and will display a message **"waiting for first client to connect"** as shown below Figure 1-15.

Figure 1-15: Waiting for first client to connect

- Go back to the network scenario. Click on **"Display Settings"** in the top ribbon/toolbar and select the **"Device IP"** checkbox inorder to display the IP address of all the devices. Now, Right click on Router 3 or any other Router and select **"NetSim Console"** option as shown Figure 1-16.
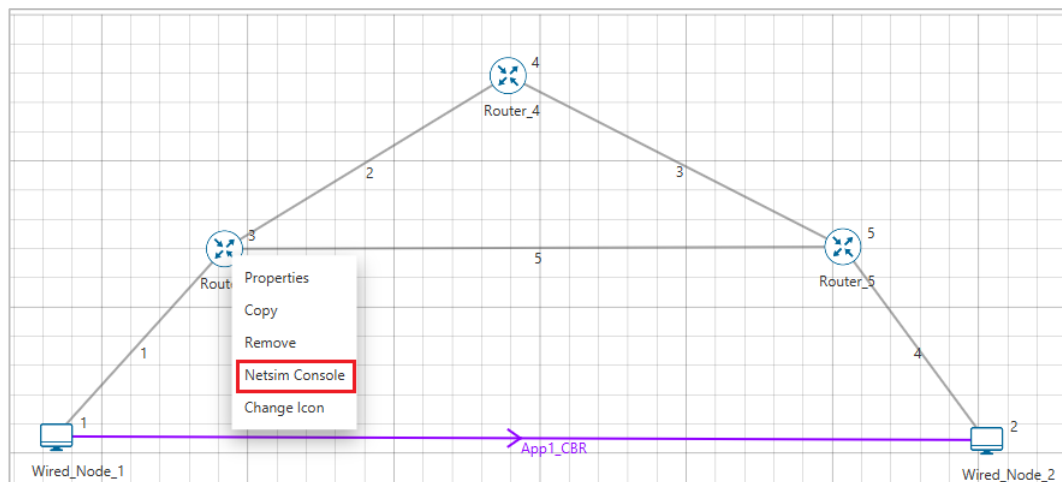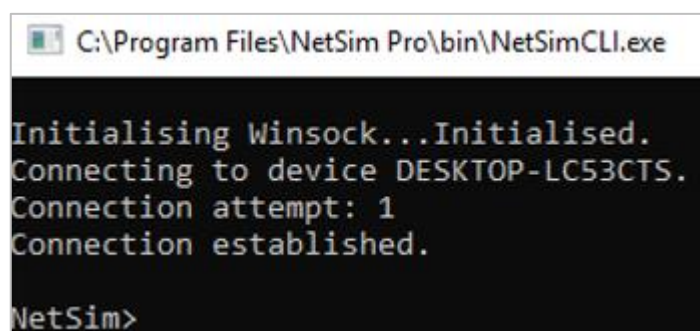
Figure 1-16: Select NetSim Console

- Now Client (NetSimCLI.exe) will start running and it will try to establish a connection with NetSimCore.exe. After the connection is established, the following will be displayed Figure 1-17.

Figure 1-17: Connection established

▪ After this the command line interface can be used to execute all the supported commands.
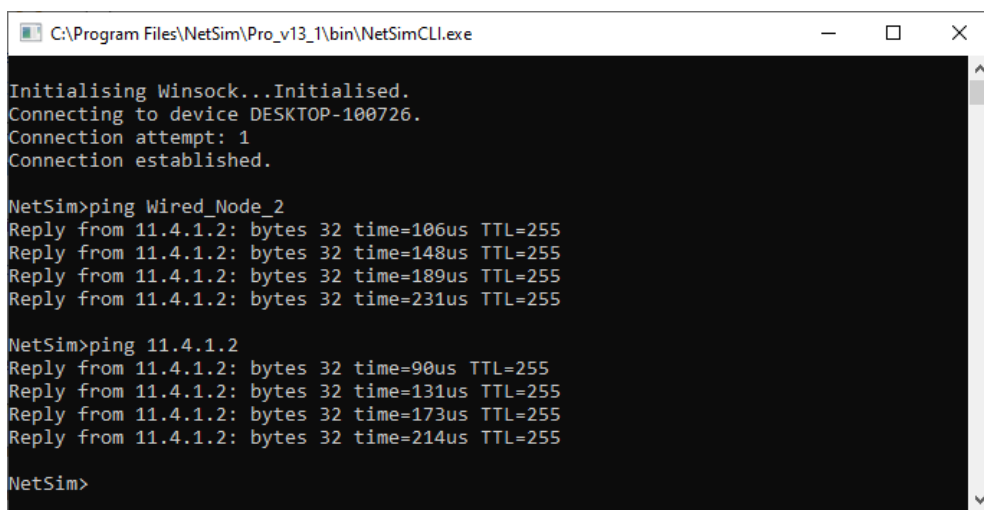
1.2.4 Network Commands

1.2.4.1 Ping Command

▪ You can use the *ping* command with an IP address or Device name.

▪ ICMP_Status should be set as True in all nodes for ping to work.

> **Ping <IP address> e.g. ping 11.4.1.2**
>
> **Ping <Node Name> e.g. ping Wired_Node_2**
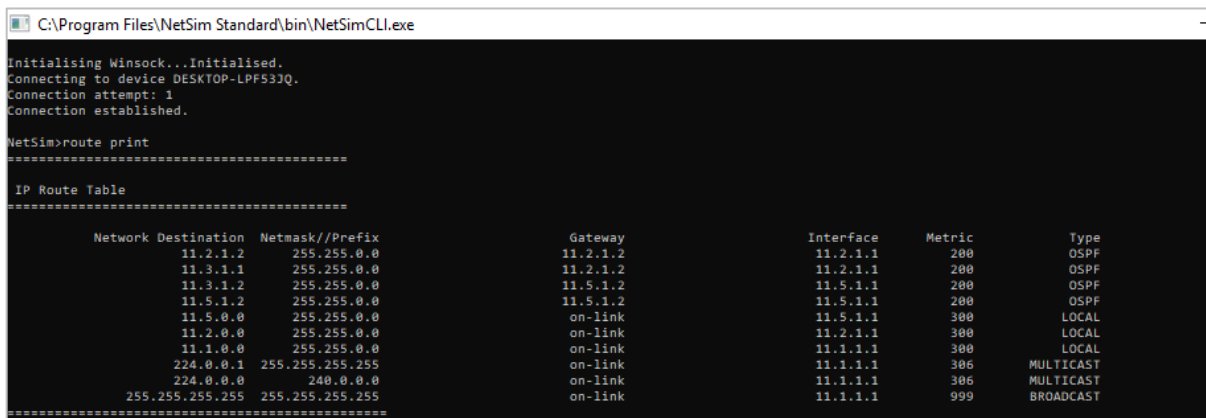
Figure 1-18: Pinging Wired_Node_2

1.2.4.2 Route Commands

▪ To view the entire contents of the IP routing table, use following command **route print**

> **route print**

Figure 1-19: IP routing table

- You'll see the routing table entries with network destinations and the gateways to which packets are forwarded when they are headed to that destination. Unless you've already added static routes to the table, everything you see here is dynamically generated.

- To delete a route in the IP routing table you'll type a command using the following syntax

```
route delete destination_network
```

- So, to delete the route with destination network 11.5.1.2, all we'd have to do is type this command

```
route delete 11.5.1.2
```

- To check whether route has been deleted or not check again using **route print** command.

- To add a static route to the table, you'll type a command using the following syntax.

```
route ADD destination_network MASK subnet_mask  gateway_ip metric_cost interface
```

- So, for example, if you wanted to add a route specifying that all traffic bound for the 11.5.1.2 subnet went to a gateway at 11.5.1.1

```
route ADD 11.5.1.2 MASK 255.255.0.0 11.5.1.1 METRIC 100 IF 2
```

- If you were to use the route print command to look at the table now, you'd see your new static route.

Figure 1-20: Route delete/ Route add

NOTE: Entry added in IP table by routing protocol continuously gets updated. If a user tries to remove a route via route delete command, there is always a chance that routing protocol will re-enter this entry again. Users can use ACL / Static route to override the routing protocol entry if required.

1.2.4.3   ACL Configuration

Commands to configure ACL

- To view ACL syntax: ***acl print***
- Before using ACL, we must first verify whether ACL option enabled. A common way to enable ACL is to use command: ***ACL Enable***
- Enter configuration mode of ACL: ***aclconfig***
- To view ACL Table: ***Print***
- To exit from ACL configuration: ***exit***
- To disable ACL: ***ACL Disable*** (use this command after **exit** from ACL Configuration)

To view ACL usage syntax use: **acl print**

[PERMIT, DENY] [INBOUND, OUTBOUND, BOTH] PROTO SRC DEST SPORT DPORT IFID

1.2.4.4   Step to Configure ACL

- To create a new rule in the ACL, use command as shown below to block UDP packet in Interface 2 and Interface 3 of Router 3.
- Application properties → Transport Protocol → **UDP** as shown Figure 1-21
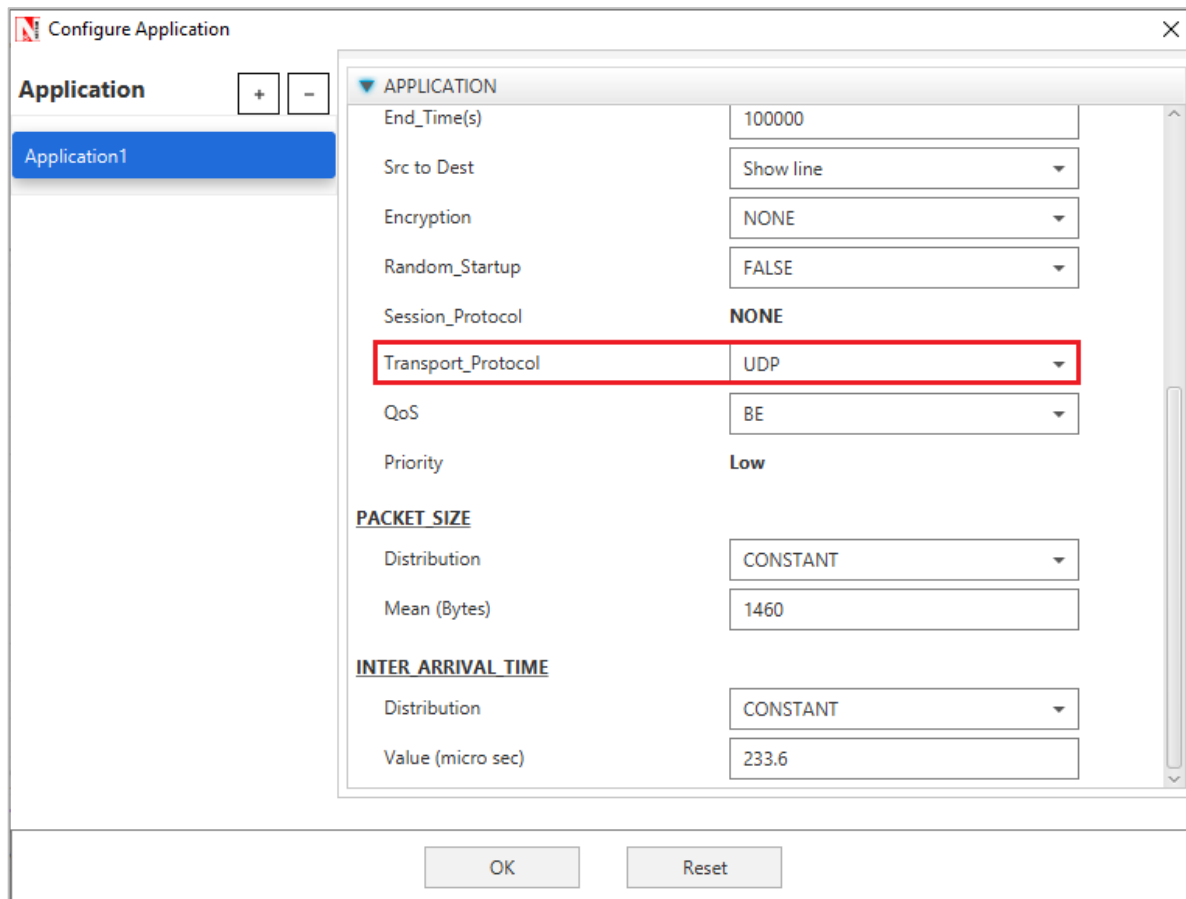
Figure 1-21: Application properties window

▪ Use the command as follows Figure 1-22.

*NetSim>**acl enable***

ACL is enable

*NetSim>**aclconfig***

*ROUTER_3/ACLCONFIG>**acl print***

Usage: [PERMIT, DENY] [INBOUND, OUTBOUND, BOTH] PROTO SRC DEST SPORT DPORT IFID

*ROUTER_3/ACLCONFIG>**DENY BOTH UDP ANY ANY 0 0 2***

OK!

*ROUTER_3/ACLCONFIG>**DENY BOTH UDP ANY ANY 0 0 3***

OK!

*ROUTER_3/ACLCONFIG>**print***

DENY BOTH UDP ANY/0 ANY/0 0 0 2

DENY BOTH UDP ANY/0 ANY/0 0 0 3

*ROUTER_3/ACLCONFIG>**exit***

*NetSim>**acl disable***

ACL is disable

NetSim>

Figure 1-22: ACL Configuration command

1.2.4.5 Ping Command Results

Go to the Results Dashboard and click on **"Open Packet Trace"** option present in the Left-Hand-Side of the window and do the following:

Filter Control Packet Type/App Name to **ICMP EchoRequest** and **ICMP EchoReply** as shown Figure 1-23.

Figure 1-23: Packet Trace - ICMP Control Packets

In Wireshark, apply filter as ICMP. we can see the ping request and reply packets in Wireshark as shown Figure 1-24.

Figure 1-24: ICMP Control packets in Wireshark

1.2.4.6 ACL Results

The impact of ACL rule applied over the simulation traffic can be observed in the IP Metrics Table in the simulation results window. In Router 3, the number of packets blocked by firewall has been shown below Figure 1-25.

Device Id	Packet sent	Packet forwarded	Packet received	Packet discarded	TTL expired	Firewall blocked
1	13599	0	0	0	0	0
2	99	0	3826	0	0	0
3	4007	13484	72	0	0	9651
4	74	0	74	0	0	0
5	4002	3832	74	0	0	0

Figure 1-25: IP Metrics Table from result window

NOTE: Number of packets blocked may vary based on the time at which ACL is configured.

Users can also observe this in Packet Animation before and after the Packets are blocked as shown below Figure 1-26/Figure 1-27.
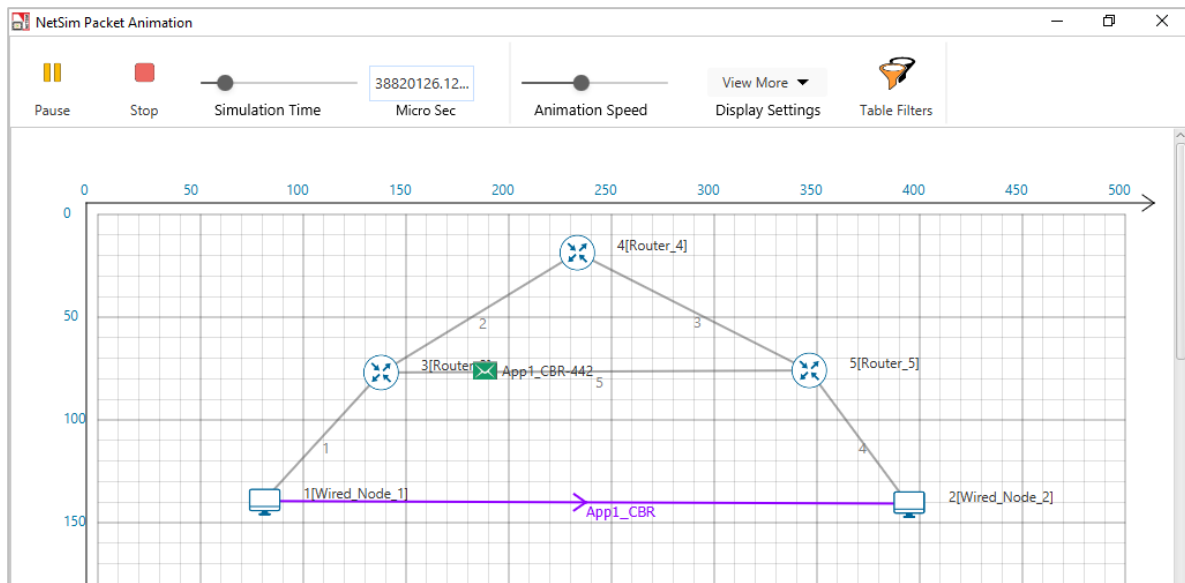
Figure 1-26: In Animation Window before applying ACL rules see the packet flow
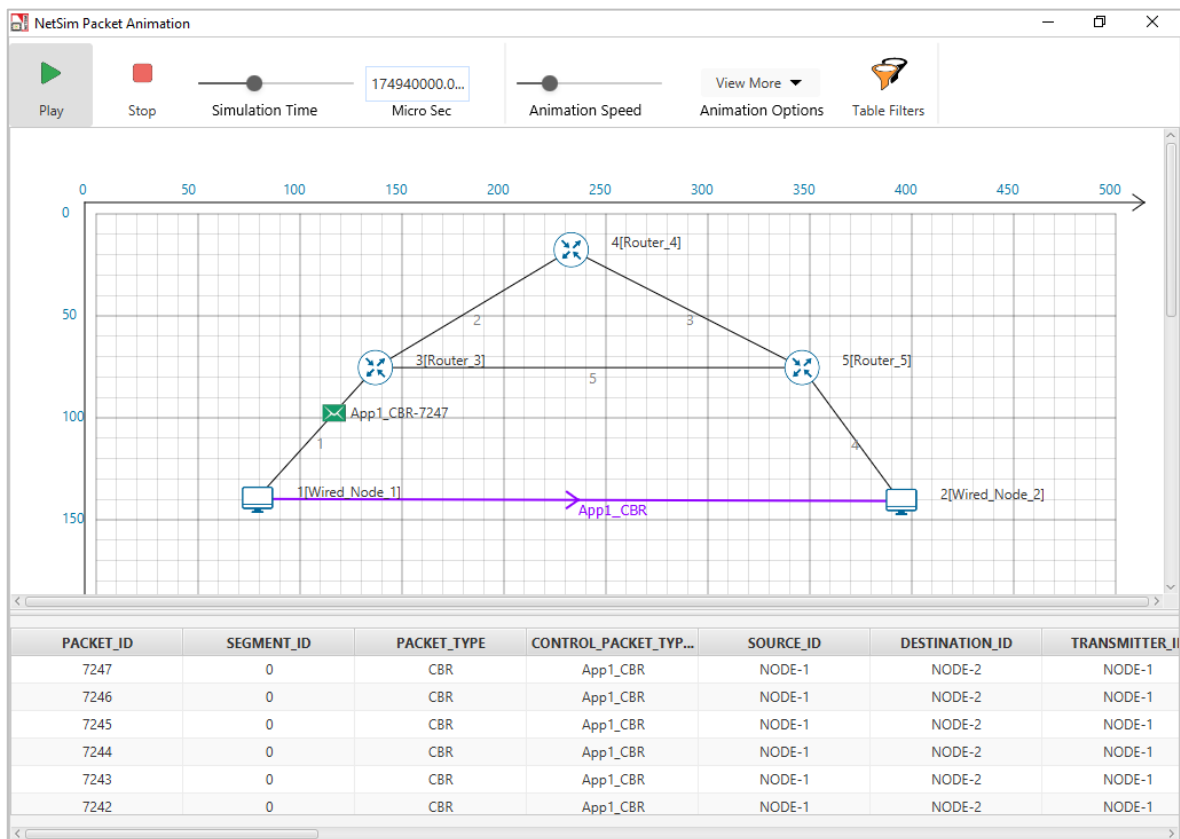
Figure 1-27: In Animation Window after applying ACL rules see the packet flow

- Check Packet animation window whether packets has been blocked in Router_3 or not after entering ACL command to deny UDP traffic.
- Before applying ACL rule there is packet flow from Wired_Node_1 to Wired_Node_2
- After applying ACL rule Packet flows up to Router_3 only.

1.3 Understand the events involved in NetSim DES (Discrete Event Simulator) in simulating flow of one packet from a Wired node to a Wireless node (Level 2)

1.3.1 Theory

NetSim's Network Stack forms the core of NetSim, and its architectural aspects are diagrammatically explained below. Network Stack accepts inputs from the end-user in the form of Configuration file and the data flows as packets from one layer to another layer in the Network Stack. All packets, when transferred between devices move up and down the stack, and all events in NetSim fall under one of these ten categories of events, namely, **Physical IN, Data Link IN, Network IN, Transport IN, Application IN, Application Out, Transport OUT, Network OUT, Data Link OUT** and **Physical OUT**. The IN events occur when the packets are entering a device while the OUT events occur while the packet is leaving a device.
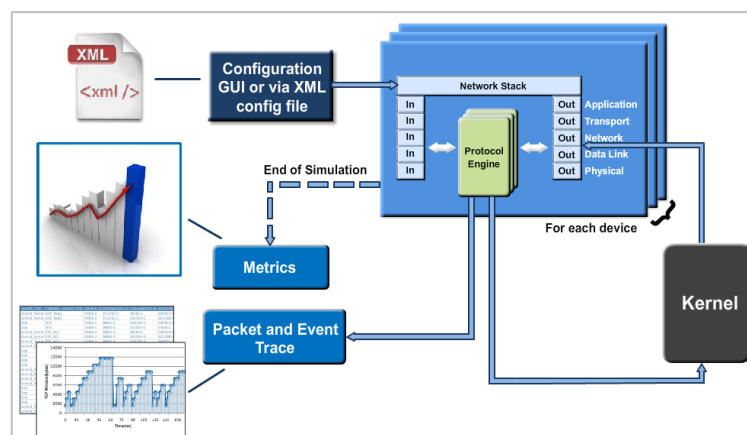
Figure 1-28: Flow of one packet from a Wired node to a Wireless node

Every device in NetSim has an instance of the Network Stack shown above. Switches & Access points have a 2-layer stack, while routers have a 3-layer stack. End-nodes have a 5-layer stack.

The protocol engines are called based on the layer at which the protocols operate. For example, TCP is called during execution of Transport IN or Transport OUT events, while 802.11b WLAN is called during execution of MAC IN, MAC OUT, PHY IN and PHY OUT events.

When these protocols are in operation, they in turn generate events for NetSim's discrete event engine to process. These are known as SUB EVENTS. All SUB EVENTS, fall into one of the above 10 types of EVENTS.

Each event gets added in the Simulation kernel by the protocol operating at the particular layer of the Network Stack. The required sub events are passed into the Simulation kernel. These sub events are then fetched by the Network Stack in order to execute the functionality of each protocol. At the end of Simulation, Network Stack writes trace files and the Metrics files that assist the user in analyzing the performance metrics and statistical analysis.

Event Trace

The event trace records every single event along with associated information such as time stamp, event ID, event type etc. in a text file or .csv file which can be stored at a user defined location.

1.3.2 Network Setup

Open NetSim and click on **Experiments> Internetworks> Network Performance> Advanced Simulation events in NetSim for transmitting one packet** then click on the tile in the middle panel to load the example as shown in below Figure 1-29.
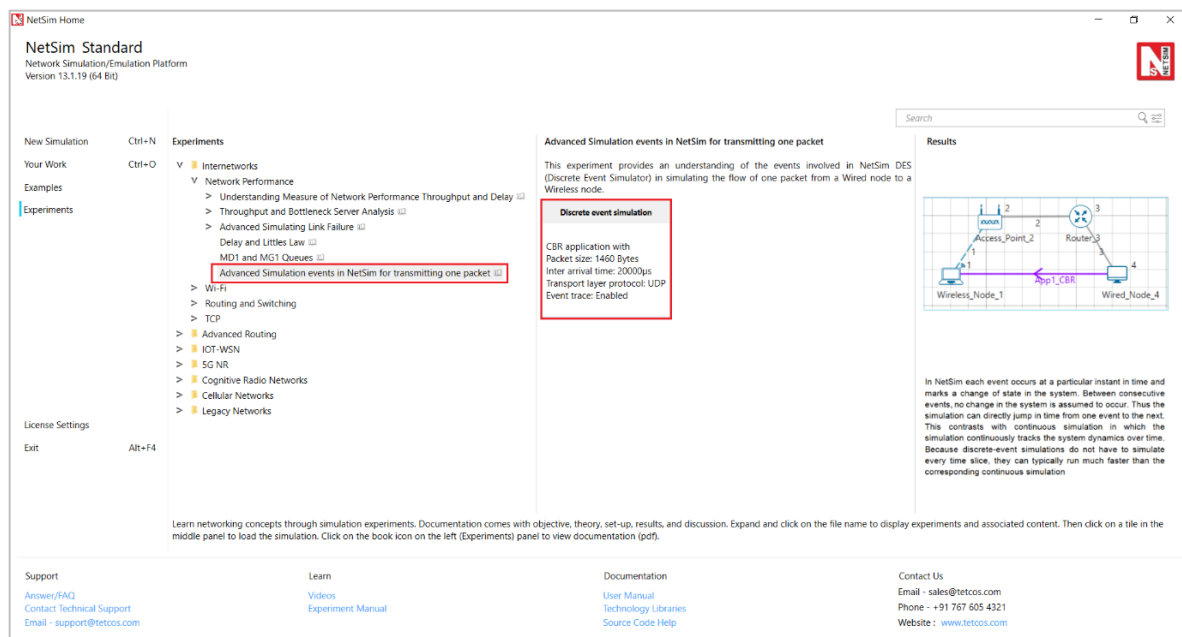
Figure 1-29: List of scenarios for the example of Advanced Simulation events in NetSim for transmitting one packet

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 1-30.
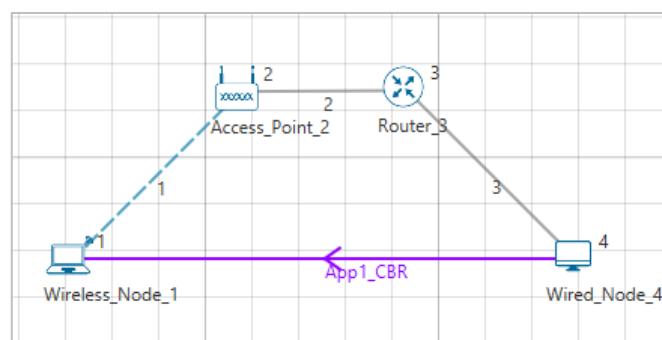
Figure 1-30: Network set up for studying the Advanced Simulation events in NetSim for transmitting one packet

1.3.3 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 1 Wireless Node, 1 Router, and 1 Access Point in the **"Internetworks"** Network Library.

Step 2: The device positions are set as per the below Table 1-1.

Device Positions				
	Access Point 2	**Wired Node 4**	**Wireless Node 1**	**Router 3**
X / Lon	150	250	100	200
Y / Lat	50	100	100	50

Table 1-1: Devices Positions

Step 3: Right-click the link ID (of the wireless link) and select Properties to access the link's properties. The **"Channel Characteristics"** is set to NO PATHLOSS.

Step 4: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 4 i.e., Source to Wireless Node 1 i.e., Destination with Packet Size remaining 1460 Bytes and Inter Arrival Time remaining 20000µs. Transport Protocol is set to UDP instead of TCP.

Step 5: Event Trace is enabled in NetSim GUI.

Step 6: Run the simulation for 10 secs. At the end of the simulation, a very large .csv file is containing all the UDP IN and OUT EVENTS is available for the users. Plots are enabled in NetSim GUI.

Note: Event trace is available only in NetSim Standard and Pro versions.

1.3.4 Output

Once the simulation is complete, go to the Results Dashboard and in the left-hand-side of the window, click on the **"Open Event Trace"** Option. An Event trace file like the following opens in Excel as shown below:

Figure 1-31: Event trace

We start from the **APPLICATION_OUT** event of the first packet, which happens in the Wired Node and end with the **MAC_IN** event of the **WLAN_ACK** packet which reaches the Wired Node. Events in the event trace are logged with respect to the time of occurrence due to which, event id may not be in order.

1.3.4.1 Events Involved

Events are listed in the following format:

[EVENT_TYPE,   EVENT_TIME, PROTOCOL, EVENT_NO, SUBEVENT_TYPE]

[APP_OUT,   20000,   APP,   6,   -]

[TRNS_OUT,   20000,   UDP,   7,   -]

[NW_OUT,   20000,   IPV4,   9,   -]

[MAC_OUT,   20000,   ETH,   10,   -]

[MAC_OUT,   20000,   ETH,   11,   CS]

[MAC_OUT,   20000.96,   ETH,   12,   IFG]

[PHY_OUT,   20000.96,   ETH,   13,   -]

[PHY_OUT,   20122.08,   ETH,   14,   PHY_SENSE]

[PHY_IN,   20127.08,   ETH,   15,   -]

[MAC_IN,   20127.08,   ETH,   16,   -]

[NW_IN,   20127.08,   IPV4,   17,   -]

[NW_OUT,   20127.08,   IPV4,   18,   -]

[MAC_OUT,   20127.08,   ETH,   19,   -]

[MAC_OUT,   20127.08,   ETH,   20,   CS]

[MAC_OUT,   20128.04,   ETH,   21,   IFG]

[PHY_OUT,   20128.04,   ETH,   22,   -]

[PHY_OUT,   20249.16,   ETH,   23,   PHY_SENSE]

[PHY_IN,   20254.16,   ETH,   24,   -]

[MAC_IN,   20254.16,   ETH,   25,   -]

[MAC_OUT,   20254.16,   WLAN,   26,   -]

[MAC_OUT,   20254.16,   WLAN,   27,   DIFS_END]

[MAC_OUT,   20304.16,   WLAN,   28,   BACKOFF]

[MAC_OUT,	20324.16,	WLAN,	29,	BACKOFF]
[MAC_OUT,	20344.16,	WLAN,	30,	BACKOFF]
[MAC_OUT,	20364.16,	WLAN,	31,	BACKOFF]
[PHY_OUT,	20364.16,	WLAN,	32,	-]
[TIMER_EVENT,	21668.16,	WLAN,	35,	UPDATE_DEVICE_STATUS]
[PHY_IN,	21668.4,	WLAN,	33,	-]
[MAC_IN,	21668.4,	WLAN,	36,	RECEIVE_MPDU]
[NW_IN,	21668.4,	IPV4,	37,	-]
[MAC_OUT,	21668.4,	WLAN,	38,	SEND_ACK]
[TRNS_IN,	21668.4,	UDP,	39,	-]
[APP_IN,	21668.4,	APP,	41,	-]
[PHY_OUT,	21678.4,	WLAN,	40,	-]
[TIMER_EVENT,	21982.4,	WLAN,	43,	UPDATE_DEVICE]
[PHY_IN,	21982.63,	WLAN,	42,	-]
[MAC_IN,	21982.63,	WLAN,	44,	RECEIVE_ACK]
[TIMER_EVENT,	21985,	WLAN,	34,	ACK_TIMEOUT]

Event Flow Diagram for one packet from Wired Node to Wireless Node
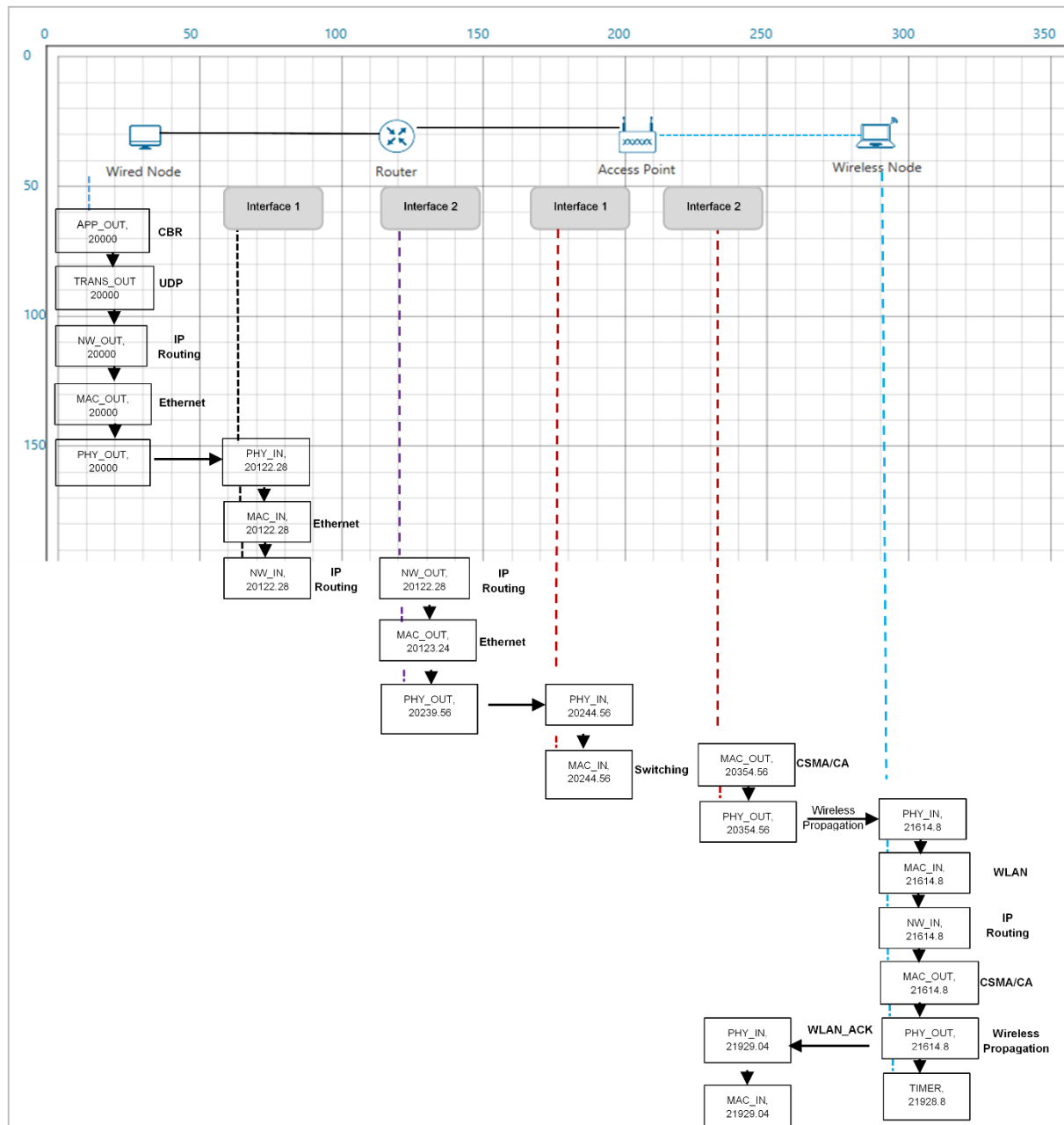
Figure 1-32: Event Flow Diagram for one packet from Wired Node to Wireless Node

For Example:

MAC_OUT in the Access Point involves sub events like CS, IEEE802.11_EVENT_DIFS_END and IEEE802.11_EVENT_BACKOFF. As you can see in the trace file shown below, CS happens at event time 20252.24. Adding DIFS time of 50µs to this will give DIFS_END sub event at 20302.24. Further it is followed by three backoffs each of 20 µs, at event time 20322.24, 20342.24, 020362.24 respectively.

Event_Id	Event_Type	Event_Time(US)	Device_Type	Device_Id	Interface_Id	Application	Packet_Id	Segment_Id	Protocol_Name	Subevent_Type	Packet_Size	Prev_Event_Id
53	TIMER_EVENT	20126.12	ROUTER	3	1	1	2	0	IPV4	IP_PROCESSING_DELAY		1488	52
54	MAC_OUT	20126.12	ROUTER	3	1	1	2	0	ETHERNET		0	1488	53
55	PHYSICAL_OUT	20126.12	ROUTER	3	1	1	2	0	ETHERNET		0	1514	54
56	PHYSICAL_IN	20252.24	ACCESSPOINT	2	2	1	2	0	ETHERNET		0	1514	55
57	MAC_IN	20252.24	ACCESSPOINT	2	2	1	2	0	ETHERNET		0	1514	56
58	MAC_OUT	20252.24	ACCESSPOINT	2	1	1	2	0	WLAN		0	1488	57
59	MAC_OUT	20252.24	ACCESSPOINT	2	1	1	2	0	WLAN	CS		1488	58
60	MAC_OUT	20302.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_DIFS_END		1488	59
61	MAC_OUT	20322.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	60
62	MAC_OUT	20342.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	61
63	MAC_OUT	20362.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	62
64	MAC_OUT	20382.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	63
65	MAC_OUT	20402.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	64
66	MAC_OUT	20422.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	65
67	MAC_OUT	20442.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	66
68	MAC_OUT	20462.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	67
69	MAC_OUT	20482.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	68
70	MAC_OUT	20502.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	69
71	MAC_OUT	20522.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	70
72	MAC_OUT	20542.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	71
73	MAC_OUT	20562.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	72
74	MAC_OUT	20582.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_BACKOFF		1488	73
75	PHYSICAL_OUT	20582.24	ACCESSPOINT	2	1	1	2	0	WLAN	IEEE802_11_EVENT_PHY_TXSTART_REQUEST		1528	74
76	PHYSICAL_IN	21886.25	NODE	1	1	1	2	0	WLAN		0	1528	75
78	MAC_IN	21886.25	NODE	1	1	1	2	0	WLAN	RECEIVE_MPDU		1528	76
79	NETWORK_IN	21886.25	NODE	1	1	1	2	0	IPV4		0	1528	78
80	MAC_OUT	21886.25	NODE	1	1	1	2	0	WLAN	SEND_ACK		1528	78
81	TRANSPORT_IN	21886.25	NODE	1	1	1	2	0	UDP		0	1468	79
83	APPLICATION_IN	21886.25	NODE	1	1	1	2	0	APPLICATION		0	1460	81

Figure 1-33: Sub events like CS, IEEE802.11_EVENT_DIFS_END and

IEEE802.11_EVENT_BACKOFF event times

In this manner the event trace can be used to understand the flow of events in NetSim Discrete Event Simulator.

1.3.5 Discussion

In NetSim each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur. Thus the simulation can directly jump in time from one event to the next.

This contrasts with continuous simulation in which the simulation continuously tracks the system dynamics over time. Because discrete-event simulations do not have to simulate every time slice, they can typically run much faster than the corresponding continuous simulation.

Understanding NetSim's Event trace and its flow is very much helpful especially when customizing existing code and debugging to verify the correctness the modified code. The event IDs provided in the event trace can be used to go to a specific event while debugging.

1.4 Plot the characteristic curve of throughput versus offered traffic for a Pure and Slotted ALOHA system (Level 2)

NOTE: NetSim Academic supports a maximum of 100 nodes and hence this experiment can only be done partially with NetSim Academic. NetSim Standard/Pro would be required to simulate all the configurations.

1.4.1 Theory

ALOHA provides a wireless data network. It is a multiple access protocol (this protocol is for allocating a multiple access channel). There are two main versions of ALOHA: pure and slotted. They differ with respect to whether time is divided up into discrete slots into which all frames must fit.

1.4.1.1 Pure Aloha

In Pure Aloha, users transmit whenever they have data to be sent. There will be collisions and the colliding frames will then be retransmitted. In NetSim's Aloha library, the sender waits a random amount of time per the exponential back-off algorithm and sends it again. The frame is discarded when the number of collisions a packet experiences crosses the the "Retry Limit" - a user settable parameter in the GUI.

Let "frame time" denotes the amount of time needed to transmit the standard, fixed-length frame. In this experiment point, we assume that the new frames generated by the stations are modeled by a Poisson distribution with a mean of N frames per frame time. If N > 1, the nodes are generating frames at a higher rate than the channel can handle, and nearly every frame will suffer a collision. For reasonable throughput, we would expect 0 < N < 1. In addition to the new frames, the stations also generate retransmissions of frames that previously suffered collisions.

The probability of no other traffic being initiated during the entire vulnerable period is given by e^{-2G} which leads to $S = G \times e^{-2G}$ where, S is the throughput and G is the offered load. The units of S and G is frames per frame time.

G is the mean of the Poisson distribution followed by the transmission attempts per frame time, old and new combined. Old frames mean those frames that have previously suffered collisions.

The maximum throughput occurs at $G = 0.5$, with $S = \frac{1}{2e}$, which is about 0.184. In other words, the best we can hope for is a channel utilization of 18%. This result is not very encouraging, but with everyone transmitting at will, we could hardly have expected a 100% success rate.

1.4.1.2 Slotted Aloha

In slotted Aloha, time is divided up into discrete intervals, each interval corresponding to one frame. In Slotted Aloha, a node is required to wait for the beginning of the next slot in order to

send the next packet. The probability of no other traffic being initiated during the entire vulnerable period is given by e^{-G} which leads to $S = G \times e^{-G}$. It is easy to compute that Slotted Aloha peaks at G = 1, with a throughput of $s = \frac{1}{e}$ or about 0.368.

1.4.2 Offered load and throughput calculations[1]

Using NetSim, the attempts per packet time (G) can be calculated as follows.

$$G = \frac{Number\ of\ packets\ transmitted \times PacketTime(s)}{SimulationTime\ (s)}$$

where, G is Attempts per packet time. We derive the above formula keeping in mind that (i) NetSim's output metric, the number of packets transmitted, is nothing but the number of attempts, and (ii) since packets transmitted is computed over the entire simulation time, the number of "packet times" would be $\frac{SimulationTime(s)}{PacketTransmissionTime(s)}$, which is in the denominator. Note that in NetSim the output metric Packets transmitted is counted at link (PHY layer) level. Hence MAC layer re-tries are factored into this metric.

The throughput (in Mbps) per packet time can be obtained as follows.

$$S = \frac{Number\ of\ packets\ successful \times PacketTime(s)}{SimulationTime\ (s)}$$

where, S = Throughput per packet time. In case of slotted aloha packet (transmission) time is equal to slot length (time). The packet transmission time is the PHY layer packet size in bits divided by the PHY rate in bits/s. Considering the PHY layer packet size as 1500B, and the PHY rate as 10 Mbps, the packet transmission time (or packet time) would be $\frac{1500 \times 8}{10 \times 10^6} = 1200\ \mu s$.

In the following experiment, we have taken packet size as 1460 B (Data Size) plus 28 B (Overheads) which equals 1488 B. The PHY data rate is 10 Mbps and hence packet time is equal to 1.2 milliseconds.

1.4.3 Network Set Up

Open NetSim and click on **Experiments> Legacy Networks> Throughput versus load for Pure and Slotted Aloha> Pure Aloha** then click on the tile in the middle panel to load the example as shown in below Figure 1-34.

---

[1] A good reference for this topic is Section 4.2.1: ALOHA, of the book, Computer Networking, 5th Edition by Tanenbaum and Wetherall
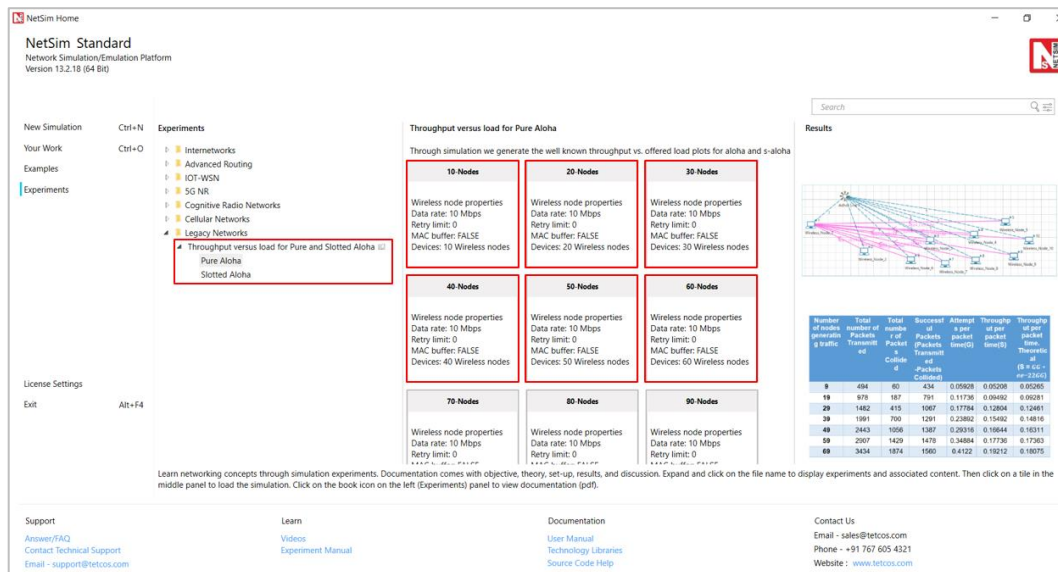
Figure 1-34: List of scenarios for the example of Throughput versus load for Pure and Slotted Aloha

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 1-35.
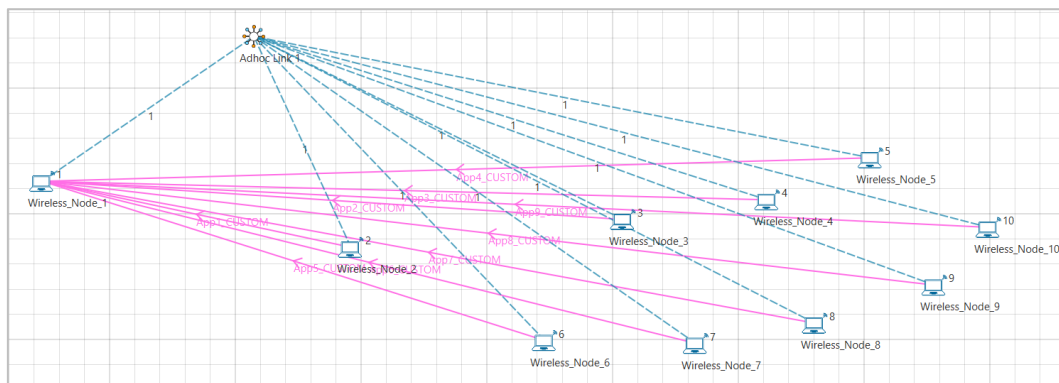
Figure 1-35: Network set up for studying the Pure aloha

Pure Aloha: Input for 10-Nodes sample

Step 1: Drop 10 nodes (i.e., 9 Nodes are generating traffic.)

Node 2, 3, 4, 5, 6, 7, 8, 9, and 10 generates traffic. The properties of Nodes 2, 3, 4, 5, 6, 7, 8, 9, and 10 which transmits data to Node 1 are given in the below table.

Step 2: Wireless Node Properties:

Wireless Node Properties	
Interface1_Wireless (PHYSICAL_LAYER)	
Data Rate (Mbps)	10
Interface1_Wireless (DATALINK_LAYER)	
Retry_Limit	0
MAC_Buffer	FALSE
Slot Length(µs)	1200

Table 1-2: Wireless Node Properties

*(**Note:** Slot Length(µs) parameter present only in Slotted Aloha → Wireless Node Properties → Interface_1 (Wireless))*

Step 3: In Adhoc Link Properties, channel characteristics is set as **No Path Loss**.

Step 4: Application Properties:

- Right click on the Application Flow "App1 CUSTOM" and select Properties or click on the Application icon present in the top ribbon/toolbar. The properties are set according to the values given in the below Table 1-3.

Application_1 Properties		
Application Method	Unicast	
Application Type	Custom	
Source_Id	2	
Destination_Id	1	
Transport Protocol	UDP	
Packet Size	Distribution	Constant
	Value (Bytes)	1460
Inter Arrival Time	Distribution	Exponential
	Packet Inter Arrival Time (µs)	200000

Table 1-3: For Application_1 Properties

- Similarly create 8 more application, i.e., Source_Id as 3, 4, 5, 6, 7, 8, 9, 10 and Destination_Id as 1, set Packet Size and Inter Arrival Time as shown in above table.

Step 5: Plots are enabled in NetSim GUI.

Step 6: Simulation Time- 10 Seconds

Note: Obtain the values of Total Number of Packets Transmitted and Collided from the results window of NetSim.

Input for 20-Nodes sample

Step 1: Drop 20 nodes (i.e., 19 Nodes are generating traffic.)

Nodes 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20 transmit data to Node 1.

Continue the experiment by increasing the number of nodes generating traffic as 29, 39, 49, 59, 69, 79, 89, 99, 109, 119, 129, 139, 149, 159, 169, 179, 189 and 199 nodes.

Slotted ALOHA: Input for 10-Nodes sample

Step 1: Drop 20 nodes (i.e., 19 Nodes are generating traffic.)

Nodes 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20 transmit data to Node 1 and set properties for nodes and application as mentioned above.

Continue the experiment by increasing the number of nodes generating traffic as 39, 59, 79, 99, 119, 139, 159, 179, 199, 219, 239, 259, 279, 299, 319, 339, 359, 379, and 399 nodes.

1.4.4 Output

Comparison Table: The values of Total Number of Packets Transmitted and Collided obtained from the network statistics after running NetSim simulation are provided in the table below along with Throughput per packet time& Number of Packets Transmitted per packet time.

Pure Aloha:

Number of nodes generating traffic	Total number of Packets Transmitted	Total number of Packets Collided	Successful Packets (Packets Transmitted-Packets Collided)	Attempts per packet time(G)	Throughput per packet time(S)	Throughput per packet time. Theoretical ($S = G * e^{-2G}$)
9	494	60	434	0.05928	0.05208	0.05265
19	978	187	791	0.11736	0.09492	0.09281
29	1482	415	1067	0.17784	0.12804	0.12461
39	1991	700	1291	0.23892	0.15492	0.14816
49	2443	1056	1387	0.29316	0.16644	0.16311
59	2907	1429	1478	0.34884	0.17736	0.17363
69	3434	1874	1560	0.4122	0.19212	0.18075
79	3964	2377	1587	0.47568	0.19044	0.18371
89	4468	2909	1559	0.53616	0.18792	0.18348
99	4998	3468	1530	0.59976	0.1836	0.18073
109	5538	4073	1465	0.66456	0.1758	0.17592
119	6023	4574	1449	0.72276	0.17388	0.1703
129	6503	5102	1401	0.78036	0.16812	0.16386
139	6992	5650	1342	0.83904	0.16104	0.15668
149	7481	6208	1273	0.89772	0.15276	0.14907
159	7998	6787	1211	0.95976	0.14532	0.14078
169	8507	7341	1166	1.02084	0.13992	0.13252
179	9008	7924	1084	1.08096	0.13008	0.12442
189	9486	8483	1003	1.13832	0.12036	0.11682
199	10025	9093	932	1.203	0.11184	0.10848

Table 1-4: Total No. of Packets Transmitted, Collided, Attempts per packet time and throughput per packet time for Pure Aloha.

Slotted Aloha

Number of nodes generating traffic	Total number of Packets Transmitted	Total number of Packets	Successful Packets (Packets Transmitted-	Attempts per packet time(G)	Throughput per packet time(S)	Throughput per packet time. Theoretical al

		Collided	Packets Collided)			$(S = G * e^{-G})$
19	974	111	863	0.11688	0.10356	0.10399
39	1981	407	1574	0.23772	0.18888	0.18742
59	2893	891	2002	0.34716	0.24024	0.24534
79	3946	1504	2442	0.47352	0.29304	0.29491
99	4976	2286	2690	0.59712	0.3228	0.32865
119	5996	3144	2852	0.71952	0.34224	0.3504
139	6961	3999	2962	0.83532	0.35544	0.36231
159	7967	4974	2993	0.95652	0.35904	0.36752
179	8969	5994	2975	1.07628	0.357	0.36686
199	9983	7042	2941	1.19796	0.35292	0.36156
219	10926	8011	2915	1.31112	0.3498	0.35337
239	11928	9073	2855	1.43136	0.3426	0.34207
259	12969	10224	2745	1.55628	0.3294	0.32825
279	13916	11266	2650	1.66992	0.318	0.31438
299	14945	12430	2515	1.7934	0.3018	0.29841
319	15967	13592	2375	1.91604	0.285	0.28202
339	17011	14765	2246	2.04132	0.26952	0.26508
359	17977	15895	2082	2.15724	0.24984	0.24947
379	18983	17010	1973	2.27796	0.23676	0.23348
399	19987	18146	1841	2.39844	0.22092	0.21792

Table 1-5: Total No. of Packets Transmitted, Collided, Throughput per packet time and throughput per packet time for Slotted Aloha

Thus, the following characteristic plot for the Pure ALOHA and Slotted ALOHA is obtained, which matches the theoretical results.
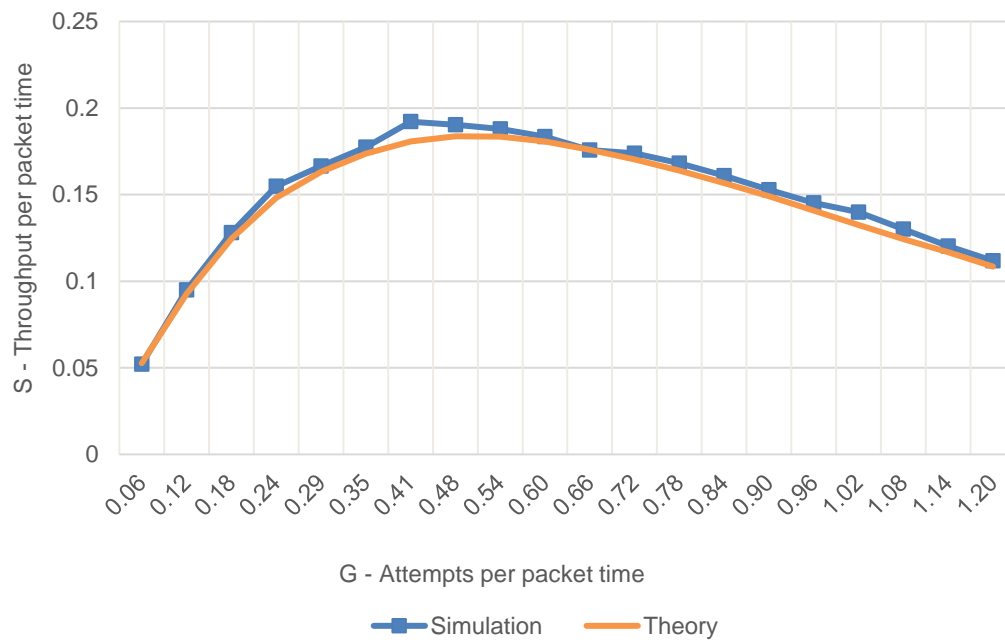
Figure 1-36: Throughput vs offered load for Pure Aloha
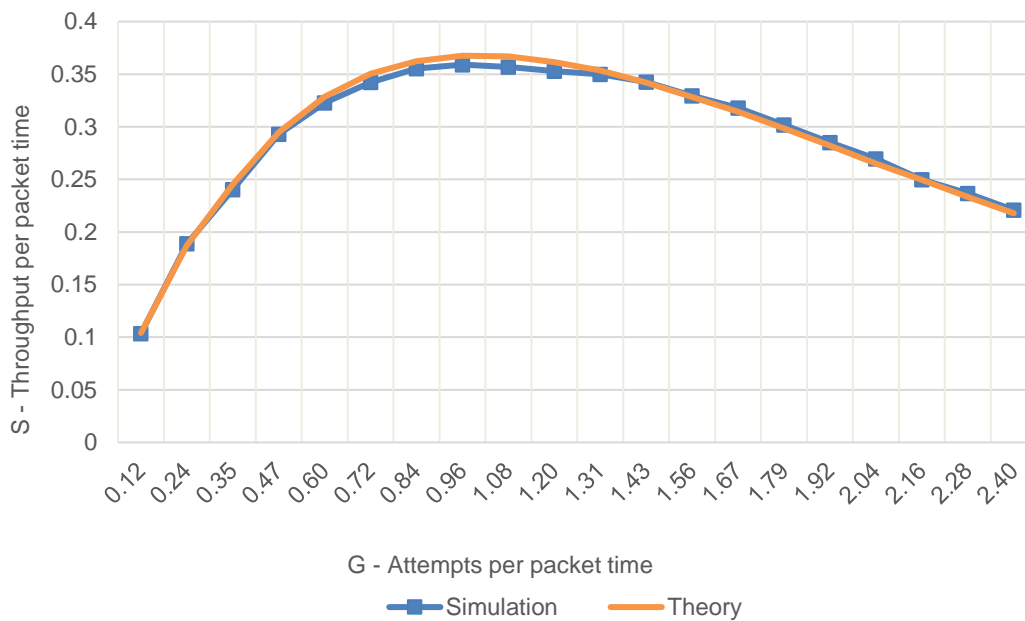
Figure 1-37: Throughput vs. Offered load for Slotted Aloha

2 Network Performance

2.1 Understand Measures of Network Performance: Throughput and Delay (Level 1)

2.1.1 Introduction

The two main performance measures of a network are:

- **Throughput:** how many bits per second are going through the network
- **Delay:** how long does it take a bit from one end to the other

These are two orthogonal concepts, and one could think of it as width of a pipe and length of a pipe through with data flows.

2.1.1.1 Throughput

In general terms, throughput is the rate of production or the rate at which something is processed. When used in the context of communication networks, such as Ethernet or packet. radio, throughput or network throughput is the rate of successful message delivery over a communication channel.

Throughput is related to other quantities like bandwidth or data-rate of a link. A link can have a certain "nominal" bandwidth or data-rate to send data at, however, all of it may not be. used all the time to send useful bits. You may also have packet losses and retransmissions. Throughput measures the number of useful bits delivered at the receiver and is different. from but related to the individual link data rates.

The throughput of a network is limited by the link with the slowest throughput along the path, the bottleneck link. You cannot pump data faster than the rate of the slowest link. Note that the bottleneck link need not always be the link with the slowest nominal data-rate. Sometimes a high-speed link may be shared by several flows, causing each flow to receive a small share, thus becoming the bottleneck. In other cases, you may not always be able to send at the bottleneck rate, because your protocol may have other delays, like waiting for ACKs. So, while instantaneous throughput can be the bottleneck link rate, average throughput may be lower. The way to compute average throughput is always: see the data sent. over a period of time and get the ratio. A file of size F takes T units of time to be transferred. Average throughput is F/T.

2.1.1.2 Delay

The end-to-end delay in a path is sum of delays on all links and intermediate nodes. There are components to delay.

When a packet leaves a node, it first experiences transmission delay. That is, all the bits of a packet that have to be put out on the link. If a link can transmit data at R bits/s, a packet of size B bits will require B/R seconds to be just put out there.

Next is propagation delay. That is, the bits have to propagate at the speed of waves in the transmission medium to reach the other end. This delay depends on the length of the wire and is usually only significant for long distance links. If d is the distance the wave has to travel is s is the speed in the medium, the propagation delay is d/s. The speed of light is 3×10^8 m/s in free space and hence a radio wave takes 1 microsec to traverse a distance of 300 metres. The speed of light in copper is around $2 \times 10^8 \, m/s$, and it would take about 10 ns to travel a 2-meter-long wire.

If propagation delay is less than the transmission delay, then the first bit of the packet would have reached the other end point before the sender finishes putting all bits on the wire. Hence the limiting factor is how fast the link is. On the other hand, if propagation delay is greater than transmission delay, as is the case for long distance links, then the first bit reaches the other end point much after the last bit has been sent.

Next, once the packet arrives at the other end point, it must be processed by the switch or router. This processing delay could involve looking up routing tables, computations of header checksums etc. Again, this is usually not a significant component with today's high-speed hardware.

Once an intermediate point processes the packet and decides which link to send it on, the packet may potentially be queued until the next link becomes free. This delay is called the queueing delay. This is the most unpredictable part of the delay, as it depends on traffic sent by other nodes. A large branch of study called *Queueing theory* is devoted to modelling and understanding this delay under various conditions. Internet traffic is often bursty, and hence queueing delays occur even if the aggregate traffic is less than the capacity of the links on an average. That is, suppose incoming packets arrive at an aggregate rate of L bits/s and link rate is R bits/s, then as long as $L < R$, it appears that there should be no queueing. However, packets don't arrive in an equally spaced fashion, and the arrival pattern is often random. In such cases, the queueing delay maybe high even if $\frac{L}{R} < .1$ In fact, queueing delay increases quite steeply as L/R approaches 1. It is approximately equal to $\frac{1}{(R-L)}$. Usually, network designers try to keep this ratio well below 1.

Once the packet gets out of the queue and gets ready for transmission, the cycle begins again with the transmission delay on the next link. So, we add one of each of the 4 delays for every link traversed. Some switches can also start transmission even before reception fully completes. But most often, switches today are store-and-forward. That is, they wait for entire packet to arrive, then start forwarding. Once a queue is full, it may also drop packets, leading

to losses. Losses can also occur due to transmission errors on the wire. This is more common in wireless links; wired links are pretty reliable.

2.1.2 NetSim Simulation Setup

Open NetSim and click on **Experiments> Internetworks> Network Performance> Understanding Measure of Network Performance Throughput and Delay** then click on the tile in the middle panel to load the example as shown in below Figure 2-1.
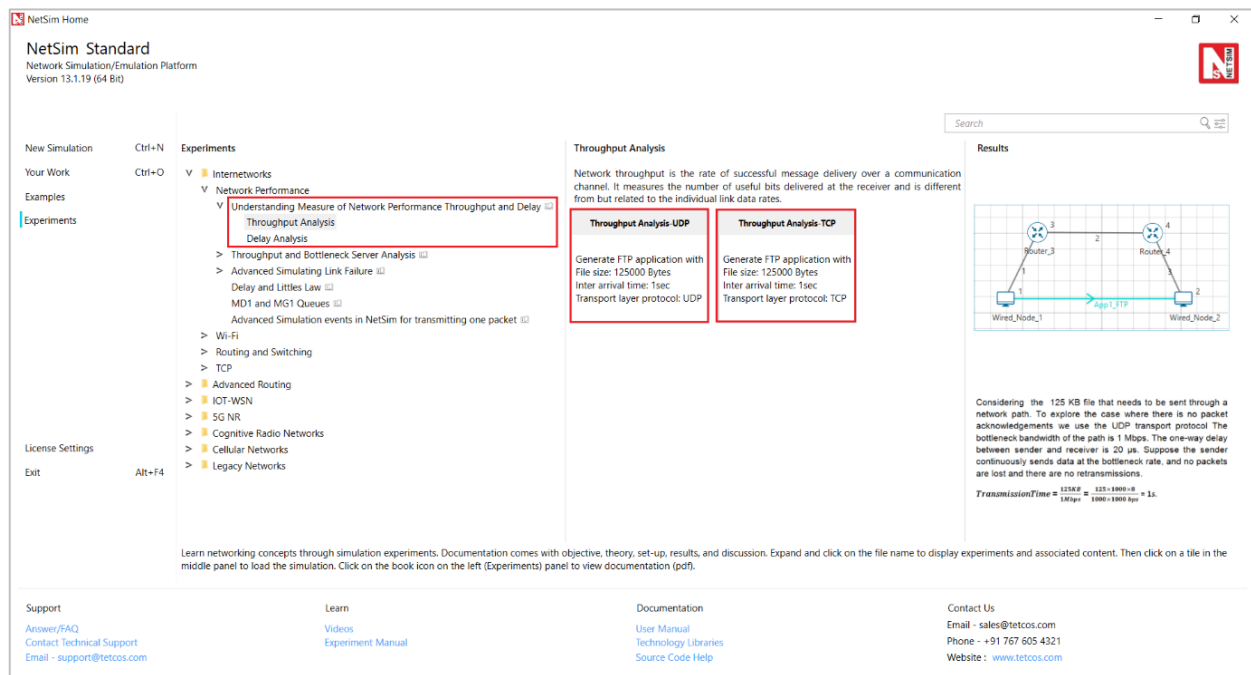
Figure 2-1: List of scenarios for the example of Understanding Measure of Network Performance Throughput and Delay

2.1.3 Part-1: Throughput Analysis

2.1.3.1 Without packet acknowledgement (UDP)

Throughput Analysis-UDP: Consider a 125 KB file that needs to be sent through a network path. To explore the case where there is no packet acknowledgements we use the UDP transport protocol The bottleneck bandwidth of the path is 1 Mbps. The one-way delay between sender and receiver is 20 µs. Suppose the sender continuously sends data at the bottleneck rate, and no packets are lost and there are no retransmissions.

$$TransmissionTime = \frac{125KB}{1\ Mbps} = \frac{125 \times 1000 \times 8}{1000 \times 1000\ bps} = 1s$$

It will take 1 second to send the file and average throughput is 1 Mbps, which is the bottleneck bandwidth.

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 2-2.
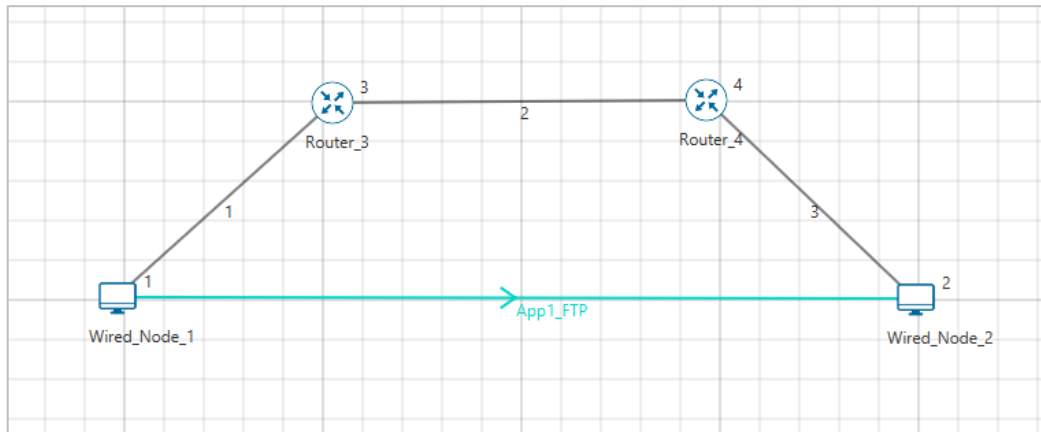
Figure 2-2: Network set up for studying the Throughput Analysis

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Router, and 2 Wired Node in the **"Internetworks"** Network Library.

Step 2: Right click on Wired link and select Properties, BER is set to 0, and Propagation Delay is set to 20µs. For link id 2 Link Speed is set to 1 Mbps.

Step 3: Right click on the Application Flow **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A FTP Application is generated from Wired Node 1 i.e. Source to Wired Node 2 i.e. Destination with File Size remaining 125000Bytes and Inter Arrival Time remaining 1s.

Transport Protocol is set to **UDP** instead of TCP.

Step 4: Enable the plots and click on Run simulation. The simulation time is set to 100 seconds.

2.1.3.2 With Packet Acknowledgement (TCP)

Throughput Analysis-TCP: Suppose the sender needs to wait for an ACK after sending every TCP traffic of 1 KB packet. Assume ACK also takes 20 ms to come back. Now, the sender can send 1 KB in $20 + 20 = 40 \ ms$. Thus, the average throughput (θ) is

$$\theta = \frac{1 \times 8 \times 1000 \ bits}{40 \ ms} = 200 \ kbps$$

Notice that the average throughput is one-fifth of what it was before, with the new ACK requirement. And the time taken to send the file will be 5 times larger, i.e 5 seconds. You can also compute 5 seconds as follows: 1 KB takes 40 ms, so 125 KB takes.

$$= 125 \times 40 \ ms = 5 \ s$$

The following set of procedures were done to generate this sample:

Step 1: Right click on Wired link and select Properties, BER is set to 0, and Propagation Delay is set to 40µs. For link id 2 Link Speed is set to 1 Mbps.

Step 2: Right click on the Application Flow **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A FTP Application is generated from Wired Node 1 i.e. Source to Wired Node 2 i.e. Destination with File Size remaining 125000Bytes and Inter Arrival Time remaining 5s.

Transport Protocol is set to **TCP**.

Step 3: Enable the plots and click on Run simulation. The simulation time is set to 100 seconds.

2.1.3.3 Output

Throughput Analysis-UDP

Application_Metrics_Table					
Application_Metrics					Detailed View
Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbp
1	Application_Throughput_plot	App1_FTP	8500	8340	0.981184

Figure 2-3: Application Throughput for Throughput Analysis-UDP

Throughput Analysis-TCP

Application_Metrics_Table					
Application_Metrics					Detailed View
Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbp
1	Application_Throughput_plot	App1_FTP	1720	1720	0.200000

Figure 2-4: Application Throughput for Throughput Analysis-TCP

2.1.4 Part - 2: Delay Analysis

2.1.4.1 Procedure

Delay Analysis-UDP: Consider the above A--S--B problem. Suppose A wants to send a 1MB file to B. A will divide the 1MB file into 1480-byte (standard UDP packet size) packets.

$$\text{Number of packets} = \frac{1000000}{1480} = 675 \text{ packets of size } 1480 + \text{last packet of size } 1054$$

To these packets a 54-byte header is added. This makes the total packet size as 1534B or $1534 \times 8 = 12{,}272 \, bits$. A packet of size 12,272 bits would take 12,272 μs of time to be transmitted over a 1Mbps (mega bit per second) link. Next, let us compute end-to-end delays. For now, let us ignore propagation and processing delays, as they are small.

A sends first 1534-byte packet in 12.27 ms and 1054-bytes packet in 8.43ms. While S forwards this packet to B, A can send the next packet to S (switches can send packets on one port while receiving them on another port.).

$$\text{File transmission time per link} = 675 \times 12272 + 1 \times 1054 \times 8 = 8.29 \text{ sec}$$

Thus, A takes 8.29 second to send all the packets to S. And a similar amount of time is taken by S to send the file to B. Therefore, the total time to send the file from A to B would be

$$T_{total} = Transmission\ Time\ per\ link \times 2 = 8.29 \times 2 = 16.58\ \mu s$$

This is now simulated in NetSim. The GUI open the configuration file corresponding to this experiment as shown below Figure 2-5
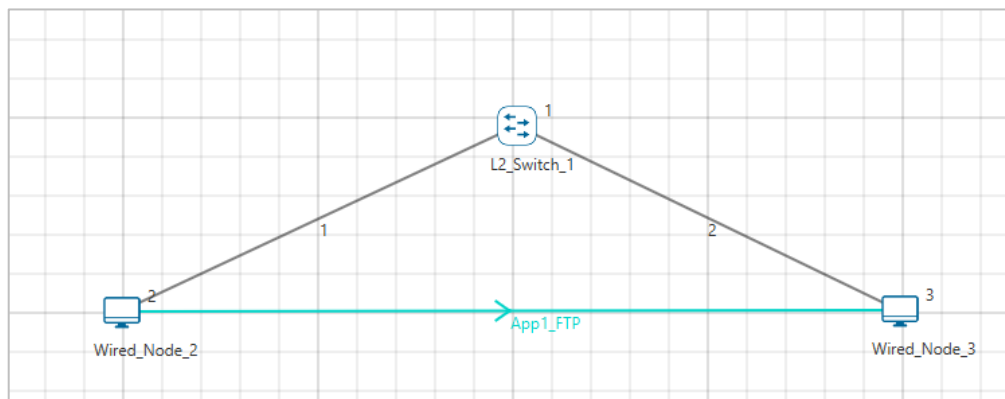
Figure 2-5: Network set up for studying the Delay Analysis

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 1 L2 Switch, and 2 Wired Node in the **"Internetworks"** Network Library.

Step 2: Right click on Wired link and select Properties, Link Speed is set to 1 Mbps, BER is set to 0, and Propagation Delay is set to 0µs.

Step 3: Right click on the Application Flow **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A FTP Application is generated from Wired Node 1 i.e. Source to Wired Node 2 i.e. Destination with File Size remaining 1000000Bytes and Inter Arrival Time remaining 100s.

Transport Protocol is set to **UDP** instead of TCP.

Step 4: Enable the packet trace and plots. Click on Run simulation. The simulation time is set to 100 seconds.

2.1.4.2 Output

Delay Analysis-UDP: In packet trace we can see only one file is generated from source to Destination, the file is divided into packets. Filter the packet type as FTP to calculate.

End to end delay = PHY_LAYER_END_TIME - PHY_LAYER_ARRIVAL_TIME

Sending 1 MB file on 1 Mbps link should take 8.29s and the same is seen in the packet trace. Then it takes another 8.29s to go from the switch to then node, or 16.58s total see Figure 2-6.

TRX_LAYER_ARRIVAL_TIME(US)	NW_LAYER_ARRIVAL_TIME(US)	MAC_LAYER_ARRIVAL_TIME(US)	PHY_LAYER_ARRIVAL_TIME(US)	PHY_LAYER_START_TIME(US)	PHY_LAYER_END_TIME(US)	End to End Delay	APP_LAYER_PAY
0	0	0	0.96	12272.96	12273.96	12273	
0	0	0	12273.92	24545.92	24546.92	12273	
0	0	0	12273.96	24545.96	24546.96	12273	
0	0	0	24546.88	36818.88	36819.88	12273	
0	0	0	24546.92	36818.92	36819.92	12273	
0	0	0	36819.84	49091.84	49092.84	12273	
0	0	0	36819.88	49091.88	49092.88	12273	
0	0	0	49092.8	61364.8	61365.8	12273	
0	0	0	49092.84	61364.84	61365.84	12273	
0	0	0	61365.76	73637.76	73638.76	12273	
0	0	0	61365.8	73637.8	73638.8	12273	
0	0	0	73638.72	85910.72	85911.72	12273	
0	0	0	73638.76	85910.76	85911.76	12273	
0	0	0	85911.68	98183.68	98184.68	12273	
0	0	0	85911.72	98183.72	98184.72	12273	
0	0	0	98184.64	110456.64	110457.64	12273	
0	0	0	98184.68	110456.68	110457.68	12273	
0	0	0	110457.6	122729.6	122730.6	12273	
0	0	0	110457.64	122729.64	122730.64	12273	
0	0	0	122730.56	135002.56	135003.56	12273	
0	0	0	122730.6	135002.6	135003.6	12273	
0	0	0	135003.52	147275.52	147276.52	12273	
0	0	0	135003.56	147275.56	147276.56	12273	
0	0	0	147276.48	159548.48	159549.48	12273	
0	0	0	147276.52	159548.52	159549.52	12273	

Packet Trace | Pivot Table(TX-RX) | Pivot Table(Custom)

Ready   Average: 12267.31953   Count: 1353   Sum: 16585416   100%

Figure 2-6: End to End Delay from Packet Trace

2.2 Simulating Link Failure (Level 1)
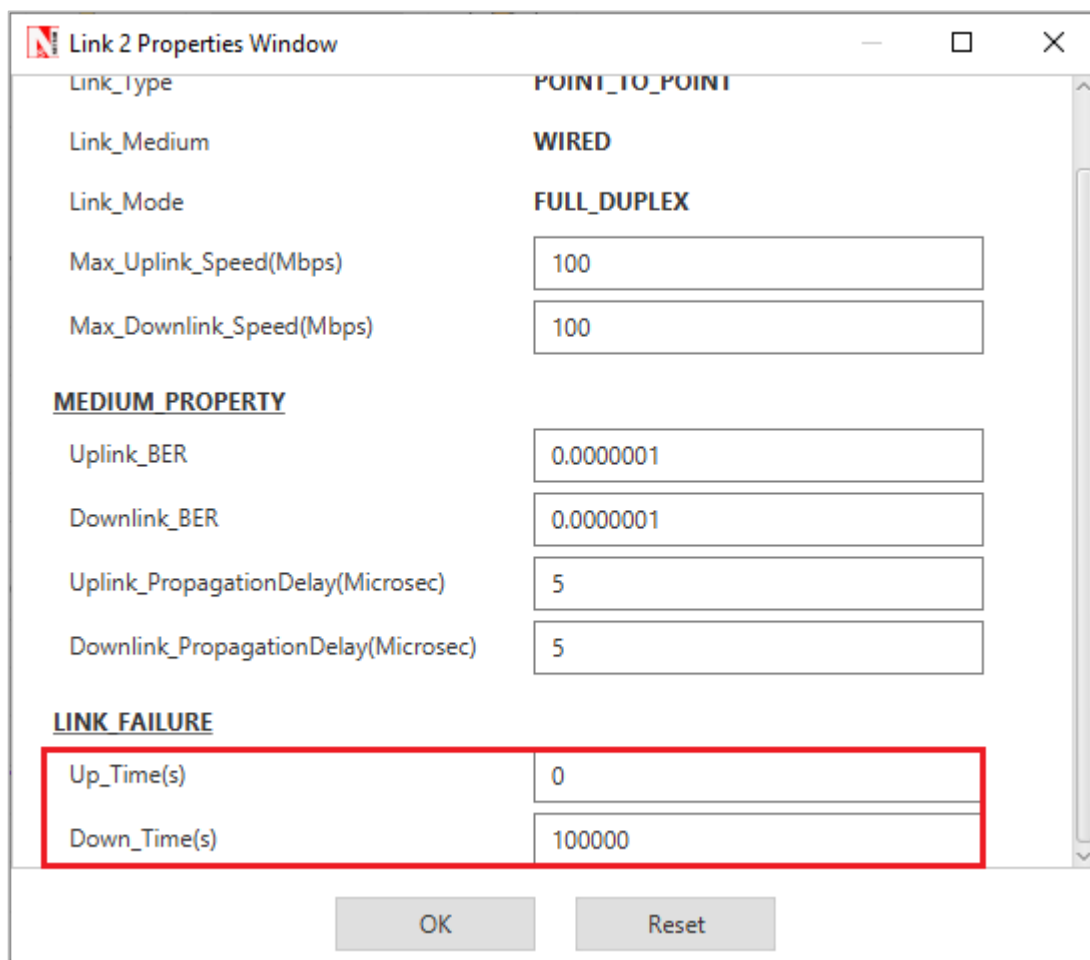
2.2.1 Objective

To model link failure, understand its impact on network performance

2.2.2 Theory

A link failure can occur due to a) faults in the physical link and b) failure of the connected port. When a link fails, packets cannot be transported. This also means that established routes to destinations may become unavailable. In such cases, the routing protocol must recompute an alternate path around the failure.

In NetSim, only WAN links (connecting two routers) can be failed. Right click on a WAN link between two routers and the Link Properties Window is as shown below Figure 2-7.

Figure 2-7: Wired Link Properties Window

Link Up Time refers to the time(s) at which the link is functional and Link Down Time refers to the time (s) at which a link fails. Click on Up_Time or Down_Time to understand the configuration options.

NOTE: Link failure can be set only for "WAN Interfaces".

2.2.3 Network Setup:

Open NetSim and click on **Experiments> Internetworks> Network Performance> Advanced Simulating Link Failure** then click on the tile in the middle panel to load the example as shown in below Figure 2-8.
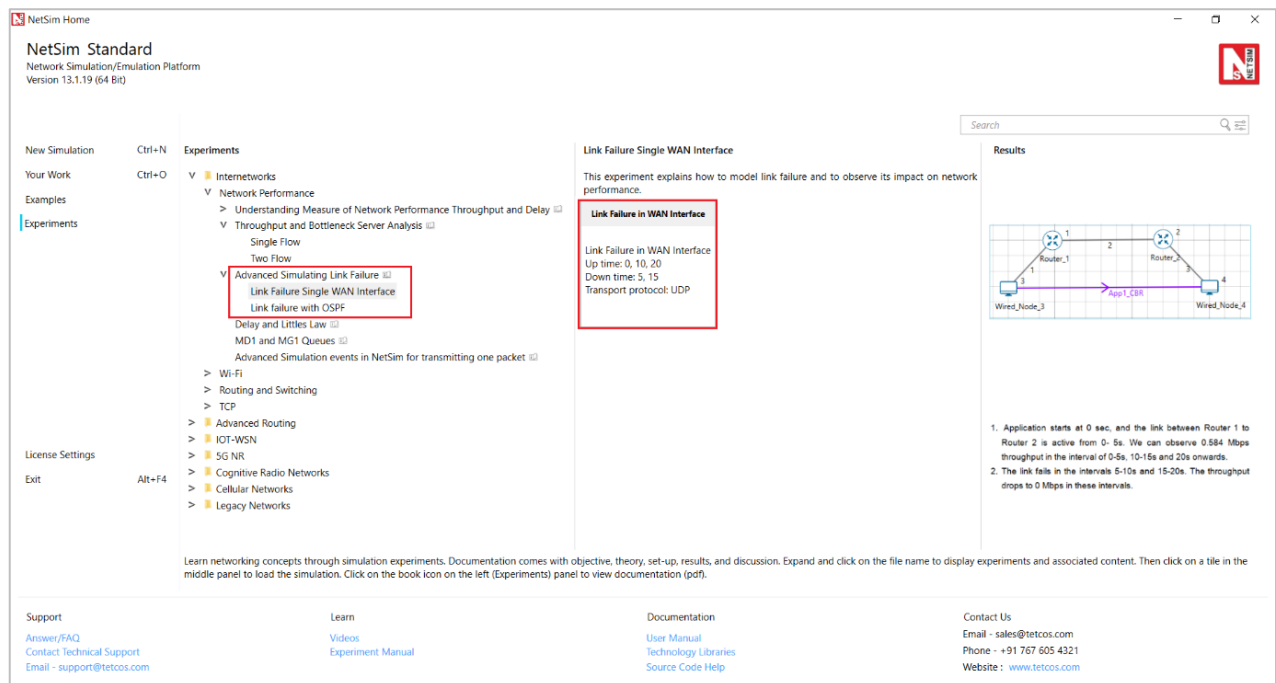
Figure 2-8: List of scenarios for the example of Advanced Simulating Link Failure

2.2.3.1 Link Failure Single WAN Interface

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 2-9.
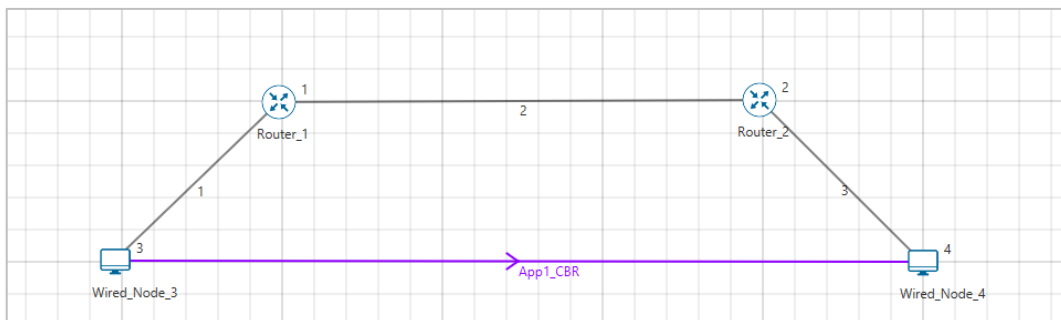
Figure 2-9: Network set up for studying the Link Failure Single WAN Interface

2.2.3.1.1 Procedure

The following set of procedures were done to generate this sample:

Step 1: In the **"Internetworks"** library, and a network scenario is designed in NetSim comprising of 2 Wired Nodes and 2 Routers.

Step 2: By default, Link Failure **Up Time** is set to 0,10,20 and **Down Time** is set to 5,15. This means the link is up 0-5s, 10-15s and 20s onwards, and it is down 5-10s and 15-20s.

Step 3: Packet Trace is enabled in NetSim GUI. At the end of the simulation, a .csv file containing all the packet information is available for performing packet level analysis.

Step 4: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 3 i.e., Source to Wired Node 4 i.e., Destination with Packet Size remaining 1460 Bytes and Inter Arrival Time remaining 20000µs.

Step 5: Transport protocol set as UDP.

Step 6: Enable the plots and run the simulation for 50 Seconds.

2.2.3.1.2 Output

Go to NetSim Simulation Result Window and open the Application Throughput plot. We can notice the following:

Figure 2-10: Application Throughput plot for APP1_CBR

1. Application starts at 0 sec, and the link between Router 1 to Router 2 is active from 0-5s. We can observe 0.584 Mbps throughput in the interval of 0-5s, 10-15s and 20s onwards.
2. The link fails in the intervals 5-10s and 15-20s. The throughput drops to 0 Mbps in these intervals.

2.2.3.2 Link Failure with OSPF

NetSim UI displays the configuration file corresponding to this experiment as shown Figure 2-11.
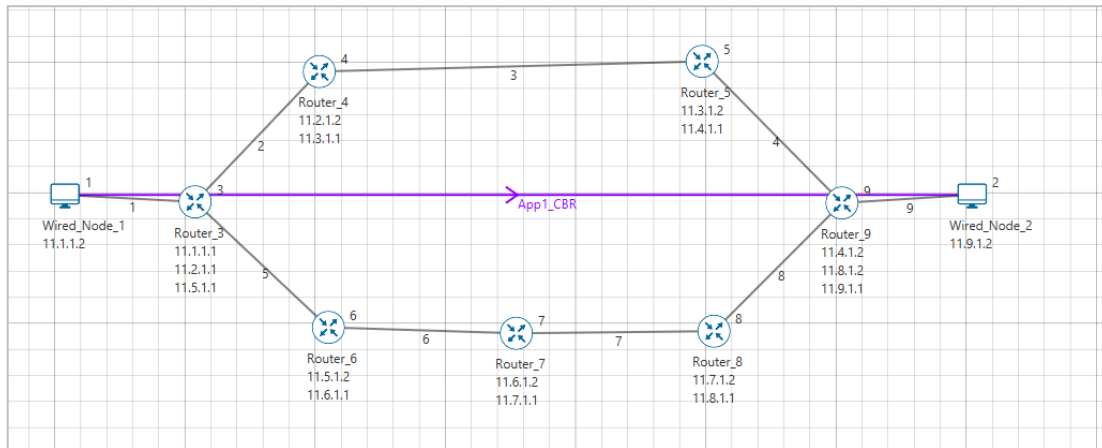
Figure 2-11: Network set up for studying the Link Failure with OSPF

2.2.3.2.1 Procedure

Without link failure: The following set of procedures were done to generate this sample:

Step 1: In the **"Internetworks"** library, and a network scenario is designed in NetSim comprising of 2 Wired Nodes and 7 Routers.

Step 2: By default, Link Failure **Up Time** is set to 0 and **Down Time** is set to 100000.

Step 3: Packet Trace is enabled in NetSim GUI. At the end of the simulation, a .csv file containing all the packet information is available for performing packet level analysis.

Step 4: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 1 i.e., Source to Wired Node 2 i.e., Destination with Packet Size remaining 1460 Bytes and Inter Arrival Time remaining 20000µs.

Additionally, the **"Start Time(s)"** parameter is set to 30, while configuring the application. This time is usually set to be greater than the time taken for OSPF Convergence (i.e., exchange of OSPF information between all the routers), and it increases as the size of the network increases.

Step 5: Transport protocol set as TCP.

Step 6: Enable the plots and run the simulation for 80 Seconds.

With link failure: The following changes in settings are done from the previous sample:

Step 1: In Link 3 Properties, Link Failure **Up Time** is set to 0 and **Down Time** is set to 50. This means that the link would fail at 50 Seconds.

Step 2: Enable the plots and run the simulation for 80 Seconds.

2.2.3.2.2 Output

Go to NetSim Packet Animation Window, click on Play button. We can notice the following:

- Initially OSPF Control Packets are exchanged between all the routers.
- Once after the exchange of control packets, the data packets are sent from the source to the destination.
- The packets are routed to the Destination via, **N1 > R3 > R4 > R5 > R9 > N2** as shown below Figure 2-12.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKE...	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	LOCAL_ADDRESS	REMOTE_ADDRESS	SEGMENT_LEN
42	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-9	NODE-2	11.1.1.2:82	11.9.1.2:36934	1460
42	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-9	11.1.1.2:82	11.4.1.2:36934	1460
42	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	ROUTER-5	11.1.1.2:82	11.4.1.2:36934	1460
42	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	11.1.1.2:82	11.4.1.2:36934	1460
42	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	11.1.1.2:82	11.1.1.1:36934	1460

Figure 2-12: Animation window for without link failure

With link failure

- We create a Link Failure in Link 3, between Router 4 and Router 5 at 50s.
- Since the packets are not able to reach the destination, the routing protocol recomputes an alternate path to the Destination.
- This can be observed in the Packet Trace.
- Go to the Results Dashboard and click on Open Packet Trace option present in the Left-Hand-Side of the window and do the following:
- Filter Control Packet Type/App Name to APP1 CBR and Transmitter ID to Router 3.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
994	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
995	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
996	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
997	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
998	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
999	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
1000	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
1001	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4
1001	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1002	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1003	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1004	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1005	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1006	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1007	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1008	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6
1009	0 CBR		App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-6

Figure 2-13: Packet Trace

▪ We can notice that packets are changing its route from,

N1 > R3 > R4 > R5 > R9 > N2 to **N1 > R3 > R6 > R7 > R8 > R9 > N2** at 50 s of simulation time, since the link between R4 and R5 fails at 50 s.

Users can also observe this in Packet animation before and after the Link Failure as shown below Figure 2-14/Figure 2-15.
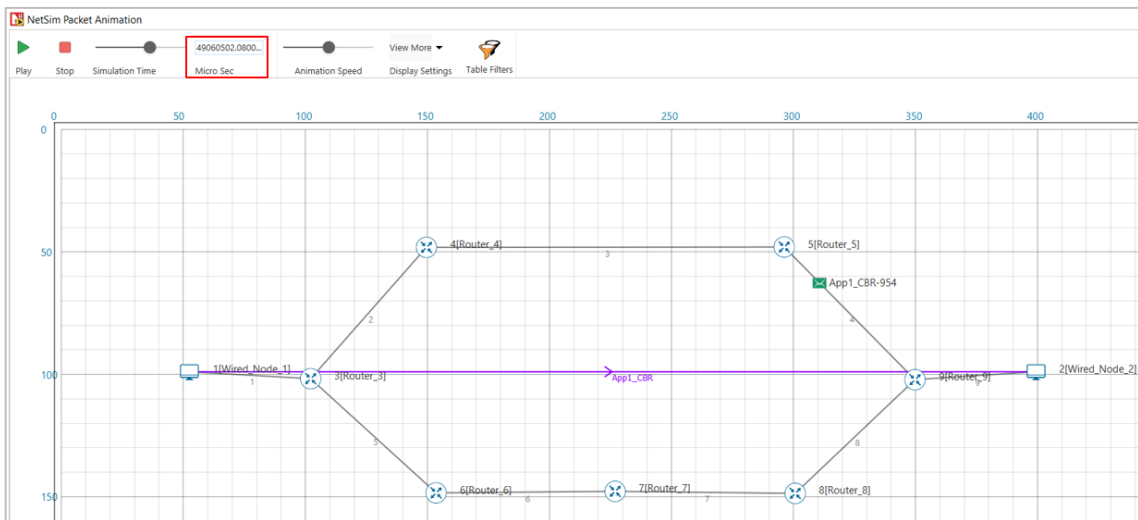
Figure 2-14: Packet animation window before Link Failure showing packet flow for with link failure
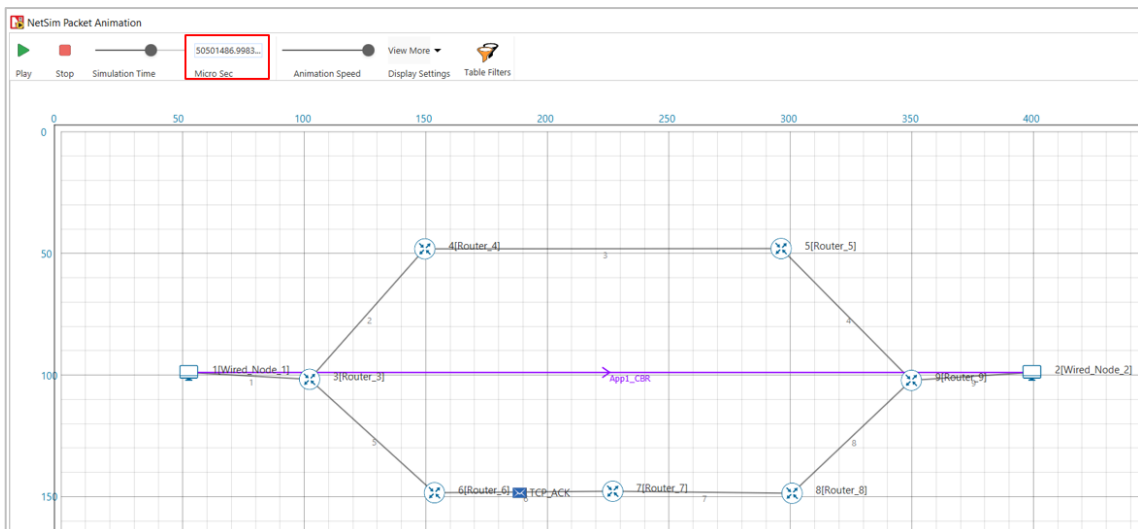
Figure 2-15: Packet animation window after Link Failure showing packet flow for with link failure

2.3 Delay and Little's Law (Level 2)

2.3.1 Introduction

Delay is another important measure of quality of a network, very relevant for real-time applications. The application processes concern over different types of delay - connection establishment delay, session response delay, end-to-end packet delay, jitter, etc. In this experiment, we will review the most basic and fundamental measure of delay, known as end-to-end packet delay in the network. The **end-to-end packet delay** denotes the sojourn time of a packet in the network and is computed as follows. Let a_i and d_i denote the time of arrival of packet i into the network (into the transport layer at the source node) and time of departure of the packet i from the network (from the transport layer at the destination node), respectively. Then, the sojourn time of the packet i is computed as $(d_i - a_i)$ seconds. A useful measure of delay of a flow is the average end-to-end delay of all the packets in the flow, and is computed as

$$\text{average packet delay} = \frac{1}{N} \sum_{i=1}^{N} (d_i - a_i) \text{ secs}$$

where, N is the count of packets in the flow.

A packet may encounter delay at different layers (and nodes) in the network. The transport layer at the end hosts may delay packets to control flow rate and congestion in the network. At the network layer (at the end hosts and at the intermediate routers), the packets may be delayed due to queues in the buffers. In every link (along the route), the packets see channel access delay and switching/forwarding delay at the data link layer, and packet transmission delay and propagation delay at the physical layer. In addition, the packets may encounter processing delay (due to hardware restrictions). It is a common practice to group the various components of the delay under the following four categories: **queueing delay** (caused due to congestion in the network), **transmission delay** (caused due to channel access and transmission over the channel), **propagation delay** and **processing delay**. We will assume zero processing delay and define packet delay as

$$\text{end to end packet delay} = \text{queueing delay} + \text{transmission delay} + \text{propagation delay}$$

We would like to note that, in many scenarios, the propagation delay and transmission delay are relatively constant in comparison with the queueing delay. This permits us (including applications and algorithms) to use packet delay to estimate congestion (indicated by the queueing delay) in the network.

2.3.1.1 Little's Law

The average end-to-end packet delay in the network is related to the average number of packets in the network. **Little's law** states that the average number of packets in the network is equal to the average arrival rate of packets into the network multiplied by the average end-to-end delay in the network, i.e.,

$$\text{average number of packets in the network}$$
$$= \text{average arrival rate into the network}$$
$$\times \text{average end to end delay in the network}$$

Likewise, the average queueing delay in a buffer is also related to the average number of packets in the queue via Little's law.

$$\text{average number of packets in queue}$$
$$= \text{average arrival rate into the queue} \times \text{average delay in the queue}$$

The following figure illustrates the basic idea behind Little's law. In Figure 2-16a, we plot the arrival process $a(t)$ (thick black line) and the departure process $d(t)$ (thick red line) of a queue as a function of time. We have also indicated the time of arrivals (a_i) and time of departures (d_i) of the four packets in Figure 2-16a. In Figure 2-16b, we plot the queue process $q(t) = a(t) - d(t)$ as a function of time, and in Figure 2-16c, we plot the waiting time $(d_{i-} a_i)$ of the four packets in the network. From the figures, we can note that the area under the queue process is the same as the sum of the waiting time of the four packets. Now, the average number of packets in the queue ($\frac{14}{10}$), if we consider a duration of ten seconds for the experiment) is equal to the product of the average arrival rate of packets $\left(\frac{4}{10}\right)$ and the average delay in the queue $\left(\frac{14}{4}\right)$.

(a) Arrival and Departure Processes   (b) Queue Process   (c) Waiting Time
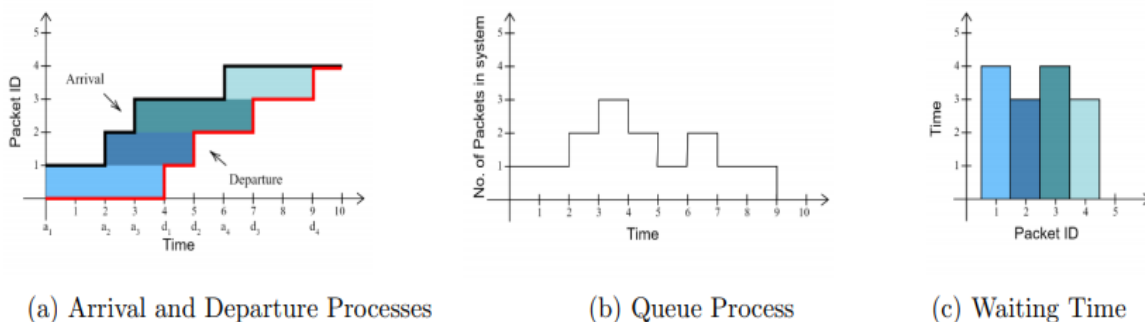
Figure 2-16: Illustration of Little's law in a queue.

In Experiment 3 (Throughput and Bottleneck Server Analysis), we noted that bottleneck server analysis can provide tremendous insights on the flow and network performance. Using M/G/1 analysis of the bottleneck server and Little's law, we can analyze queueing delay at the

bottleneck server and predict end-to-end packet delay as well (assuming constant transmission times and propagation delays).

2.3.2 NetSim Simulation Setup

Open NetSim and click on **Experiments> Internetworks> Network Performance> Delay and Littles Law** then click on the tile in the middle panel to load the example as shown below in Figure 2-17.

Figure 2-17: List of scenarios for the example of Delay and Littles Law

2.3.3 Part-1: A Single Flow Scenario

We will study a simple network setup with a single flow illustrated in Figure 2-18 to review end to end packet delay in a network as a function of network traffic. An application process at Wired_Node_1 seeks to transfer data to an application process at Wired_Node_2. We will consider a custom traffic generation process (at the application) that generates data packets of constant length (say, L bits) with i.i.d. inter-arrival times (say, with average inter-arrival time v seconds). The application traffic generation rate in this setup is $\frac{L}{v}$ bits per second. We prefer to minimize communication overheads (including delay at the transport layer) and hence, will use UDP for data transfer between the application processes.

In this setup, we will vary the traffic generation rate $\left(\frac{L}{v}\right)$ by varying the average inter-arrival time (v), and review the average queue at the different links, average queueing delay at the different links and end-to-end packet delay.

2.3.3.1 Procedure

We will simulate the network setup illustrated in Figure 2-18 with the configuration parameters listed in detail in Table 2-1 to study the single flow scenario.

NetSim UI displays the configuration file corresponding to this experiment as shown below:
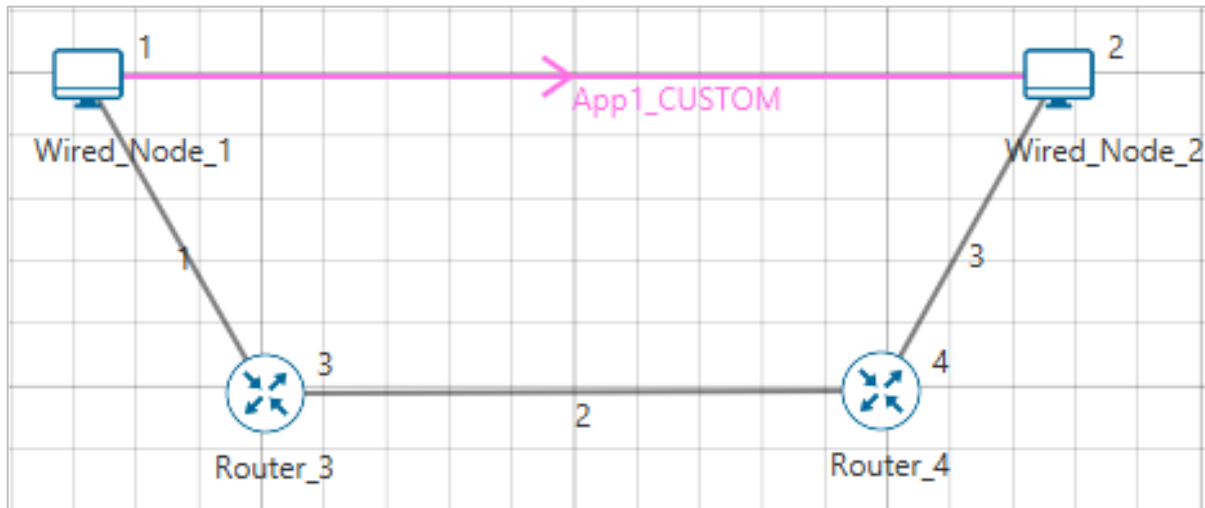
Figure 2-18: Network set up for studying a single flow

The following set of procedures were done to generate this sample:

Step 1: Drop two wired nodes and two routers onto the simulation environment. The wired nodes and the routers are connected with wired links as shown in (See Figure 2-18).

Step 2: Click the **Application** icon to configure a custom application between the two wired nodes. In the **Application** configuration dialog box (see Figure 2-19), select **Application Type** as CUSTOM, **Source ID** as 1 (to indicate Wired_Node_1), **Destination ID** as 2 (to indicate Wired_Node_2) and **Transport Protocol** as UDP. In the PACKET SIZE tab, select **Distribution** as CONSTANT and **Value** as 1460 bytes. In the INTER ARRIVAL TIME tab, select **Distribution** as EXPONENTIAL and **Mean** as 11680 microseconds.
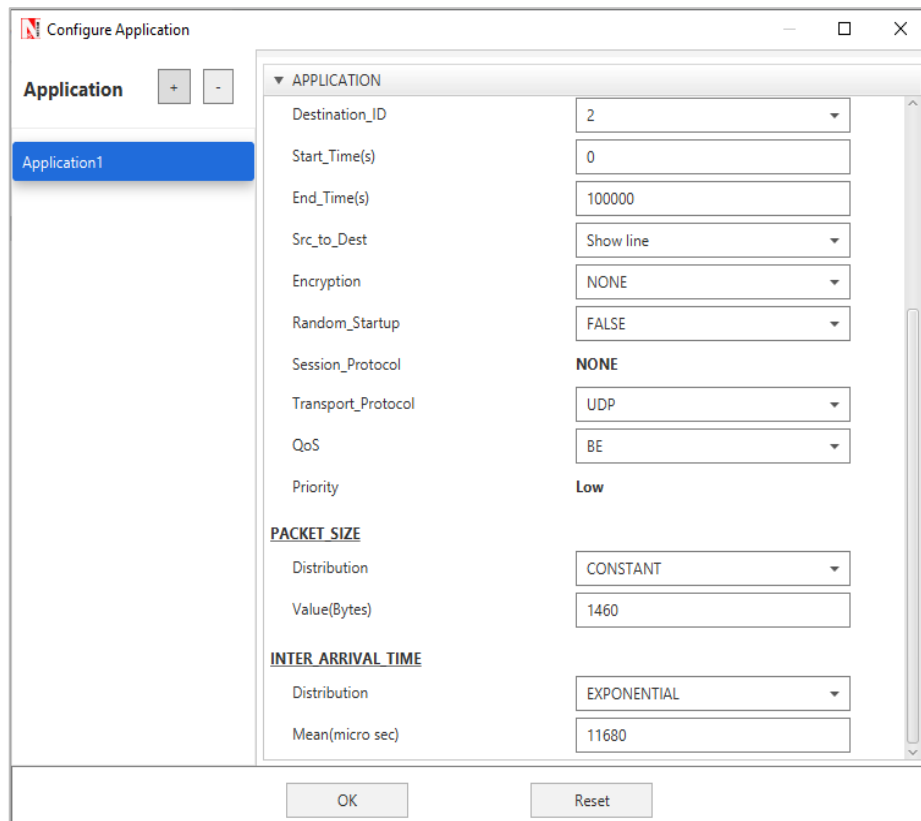
Figure 2-19: Application configuration dialog box

Step 3: The properties of the wired nodes are left to the default values.

Step 4: Right-click the link ID (of a wired link) and select **Properties** to access the link's properties dialog box (see Figure 2-20). Set **Max Uplink Speed** and **Max Downlink Speed** to 10 Mbps for link 2 (the backbone link connecting the routers) and 1000 Mbps for links 1 and 3 (the access link connecting the Wired_Nodes and the routers). Set **Uplink BER** and **Downlink BER** as 0 for links 1, 2 and 3. Set **Uplink_Propagation Delay** and **Downlink_Propagation_Delay** as 0 microseconds for the two-access links 1 and 3 and 10 milliseconds for the backbone link 2.

Figure 2-20: Link_ID_2 Properties dialog box

Step 5: Right-click **Router 3** icon and select **Properties** to access the link's properties dialog box (see Figure 2-21). In the **INTERFACE 2 (WAN)** tab, select the **NETWORK LAYER** properties, set **Buffer size (MB)** to **8.**
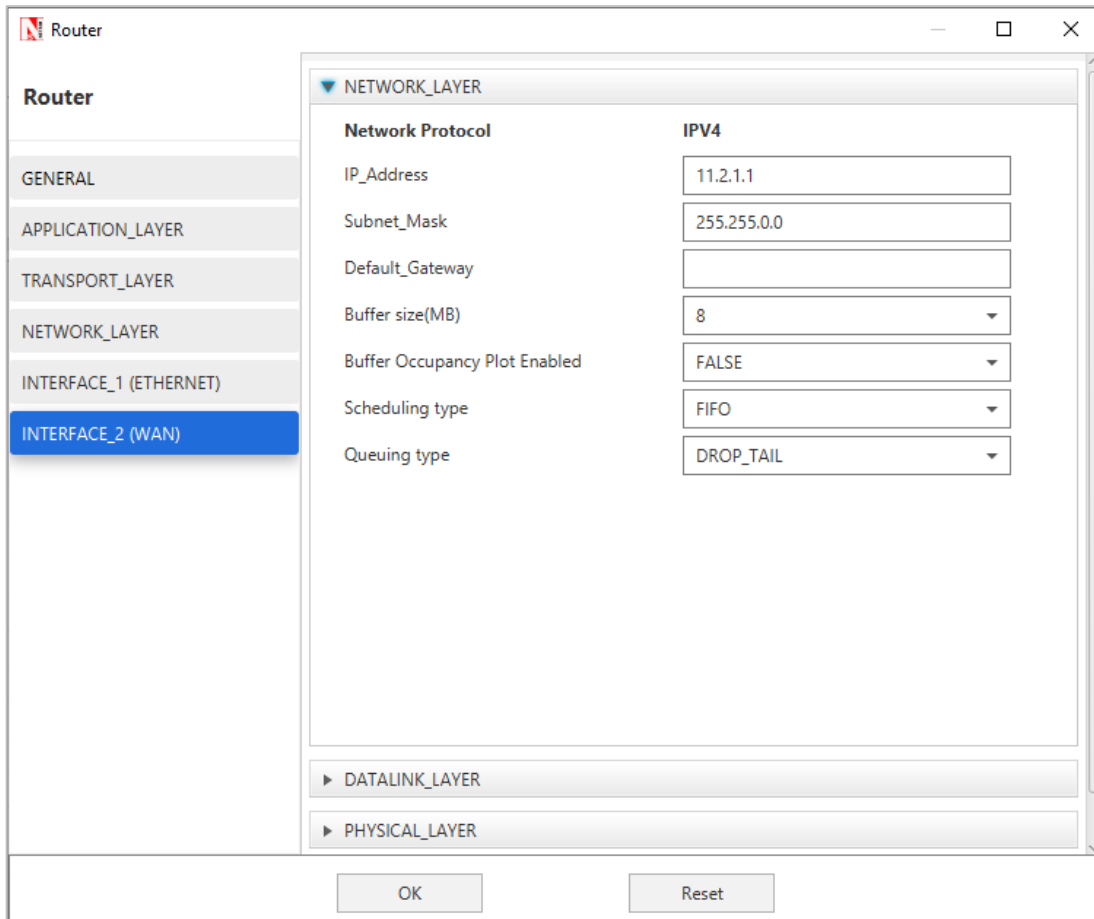
Figure 2-21: Router Properties dialog box

Step 6: Enable Plots and Packet Trace check box. Packet Trace can be used for packet level analysis.

Step 7: Click on **Run** icon to access the Run Simulation dialog box (see Figure 2-22) and set the **Simulation Time** to 100 seconds in the **Simulation Configuration** tab. Now, run the simulation.
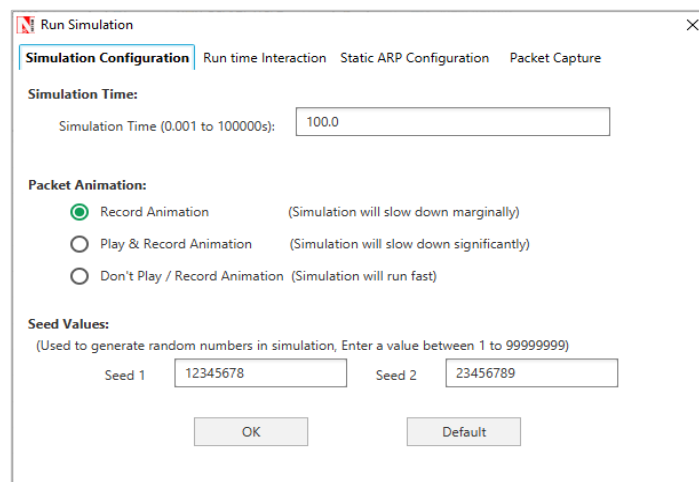
Figure 2-22: Run Simulation dialog box

Step 8: Now, repeat the simulation with different average inter-arrival times (such as 5840 μs, 3893 μs, 2920 μs, 2336 μs and so on). We vary the input flow rate by varying the average inter-

arrival time. This should permit us to identify the bottleneck link and the maximum achievable throughput.

The detailed list of network configuration parameters is presented in (See Table 2-1).

Parameter	Value
LINK PARAMETERS	
Wired Link Speed (access link)	1000 Mbps
Wired Link Speed (backbone link)	10 Mbps
Wired Link BER	0
Wired Link Propagation Delay (access link)	0
Wired Link Propagation Delay (backbone link)	10 milliseconds
APPLICATION PARAMETERS	
Application	Custom
Source ID	1
Destination ID	2
Transport Protocol	UDP
Packet Size – Value	1460 bytes
Packet Size – Distribution	Constant
Inter Arrival Time – Mean	AIAT (µs) **Table 2-2**
Inter Arrival Time – Distribution	Exponential
ROUTER PARAMETERS	
Buffer Size	8
MISCELLANEOUS	
Simulation Time	100 Sec
Packet Trace	Enabled
Plots	Enabled

Table 2-1: Detailed Network Parameters

2.3.3.2 Performance Measure

In Table 2-2 and Table 2-3, we report the flow average inter-arrival time v and the corresponding application traffic generation rate, input flow rate (at the physical layer), average queue and delay of packets in the network and in the buffers, and packet loss rate.

Given the average inter-arrival time v and the application payload size L bits (here, 1460×8 = 11680 bits), we have,

$$\text{Traffic generation rate} = \frac{L}{v} = \frac{11680}{v} \, bps$$

$$\text{input flow rate} = \frac{11680 + 54 \times 8}{v} = \frac{12112}{v} \, bps$$

where the packet overheads of 54 bytes is computed as $54 = 8(UDP\ header) + 20(IP\ header) + 26(MAC + PHY\ header)\ bytes$.

Let $Q_l(u)$ as denote the instantaneous queue at link l at time u. Then, the average queue at link l is computed as

$$average\ queue\ at\ link\ l = \frac{1}{T} \int_0^T Q_l\ (u)\ \ du\ bits$$

where, T is the simulation time. And, let $N(u)$ denote the instantaneous number of packets in the network at time u. Then, the average number of packets in the network is computed as

$$average\ number\ of\ packet\ in\ the\ network\ = \frac{1}{T}\int_0^T N(u)\ du\ bits$$

Let $a_{i,l}$ and $d_{i,l}$ denote the time of arrival of a packet i into the link l (the corresponding router) and the time of departure of the packet i from the link l (the corresponding router), respectively. Then, the average queueing delay at the link l (the corresponding router) is computed as

$$average\ queueing\ delay\ at\ link\ l = \frac{1}{N}\sum_{i=1}^{N}(d_{i,l-}\ a_{i,l})$$

where N is the count of packets in the flow. Let ai and di denote the time of arrival of a packet i into the network (into the transport layer at the source node) and time of departure of the packet i from the network (from the transport layer at the destination node), respectively. Then, the end-to-end delay of the packet i is computed as $(d_{i-}\ a_i)$ seconds, and the average end to end delay of the packets in the flow is computed as

$$average\ end\ to\ end\ packet\ delay = \frac{1}{N}\sum_{i=1}^{N}(d_{i-}\ a_i)$$

2.3.3.2.1 Average Queue Computation from Packet Trace

- Open Packet Trace file using the **Open Packet Trace** option available in the Simulation Results window.
- In the Packet Trace, filter the data packets using the column **CONTROL PACKET TYPE/APP NAME** and the option **App1 CUSTOM** (see Figure 2-23).
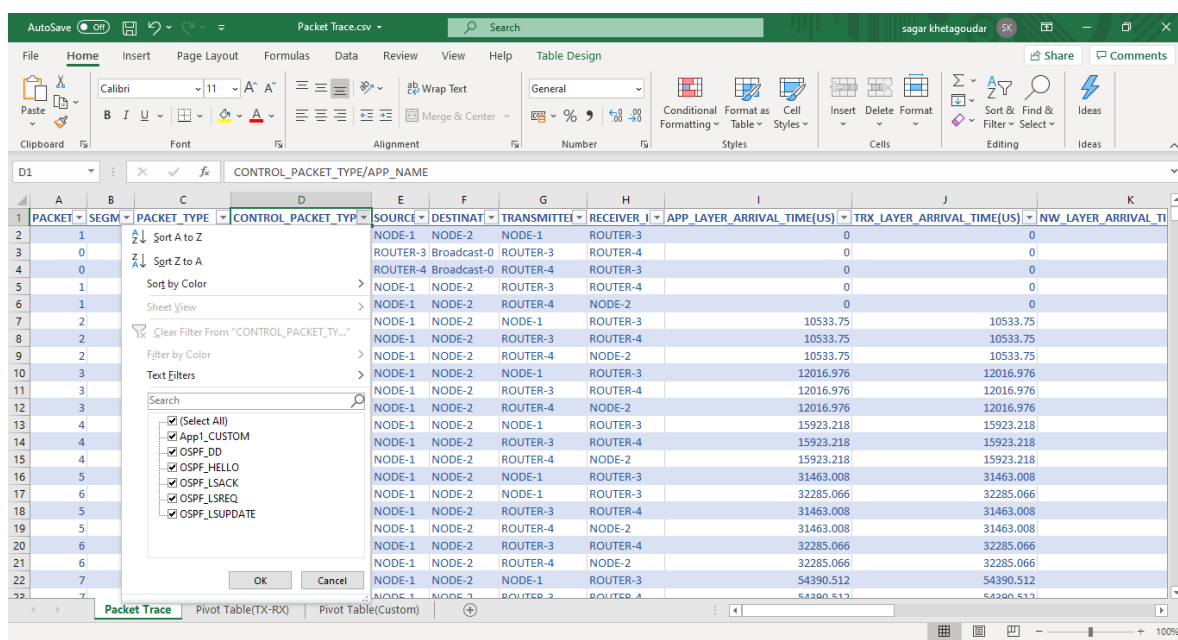
Figure 2-23: Filter the data packets in Packet Trace by selecting App1 CUSTOM.

- Now, to compute the average queue in Link 2, we will select **TRANSMITTER_ID** as **ROUTER-3** and **RECEIVER_ID** as **ROUTER-4**. This filters all the successful packets from Router 3 to Router 4.
- The columns **NW_LAYER_ARRIVAL_TIME(US)** and **PHY_LAYER_ARRIVAL TIME(US)** correspond to the arrival time and departure time of the packets in the buffer at Link 2, respectively (see Figure 2-24).
- You may now count the number of packets arrivals (departures) into (from) the buffer upto time t using the NW_LAYER_ARRIVAL_TIME(US) (PHY_LAYER_ARRIVAL TIME(US)) column. The difference between the number of arrivals and the number of departures gives us the number of packets in the queue at any time.

Figure 2-24: Packet arrival and departure times in the link buffer

- Calculate the average queue by taking the mean of the number of packets in queue at every time interval during the simulation.
- The difference between the PHY LAYER ARRIVAL TIME(US) and the NW LAYER ARRIVAL TIME(US) will give us the delay of a packet in the link (see Figure 2-25).

$$\text{Queuing Delay} = \text{PHY LAYER ARRIVAL TIME(US)} - \text{NW LAYER ARRIVAL TIME(US)}$$

Figure 2-25: Queuing Delay

- Now, calculate the average queuing delay by taking the mean of the queueing delay of all the packets (see Figure 2-25)

2.3.3.2.2 Network Delay Computation from Packet Trace

- Open Packet Trace file using the **Open Packet Trace** option available in the Simulation Results window
- In the Packet Trace, filter the data packets using the column **CONTROL PACKET TYPE/APP NAME** and the option **App1 CUSTOM** (see Figure 2-23**)**.
- Now, we will select the **RECEIVER ID** as **NODE-2**. This filters all the successful packets in the network that reached Wired Node 2
- The columns **APP LAYER ARRIVAL TIME(US)** and **PHY LAYER END TIME(US)** correspond to the arrival time and departure time of the packets in the network respectively.
- You may now count the number of arrivals (departures) into (from) the network upto time t using the APP LAYER ARRIVAL TIME(US) (PHY LAYER END TIME(US)) column. The difference between the number of arrivals and the number of departures gives us the number of packets in the network at any time.
- Calculate the average number of packets in the network by taking the mean of the number of packets in network at every time interval during the simulation.
- Packet Delay at a per packet level can be calculated using the columns **Application Layer Arrival Time** and **Physical Layer End Time** in the packet trace as:
    End-to-End Delay = PHY LAYER END TIME(US) – APP LAYER ARRIVAL TIME(US)
- Calculate the average end-to-end packet delay by taking the mean of the difference between Phy Layer End Time and App Layer Arrival Time columns.

Note: To calculate average number of packets in queue refer the experiment on Throughput and Bottleneck Server Analysis.

2.3.3.3 Results

In Table 2-2, we report the flow average inter-arrival time (AIAT) and the corresponding application traffic generation rate (TGR), input flow rate (at the physical layer), average number of packets in the system, end-to-end packet delay in the network and packet loss rate.

AIAT v (in μs)	TGR $\frac{L}{v}$ (in Mbps)	Input Flow Rate (in Mbps)	Arrival Rate (in Pkts/sec)	Avg no of packets in system	End-to-End Packet Delay (in μs)	Packet Loss Rate (in percent)
11680	1	1.037	86	0.97	11282.188	0.01
5840	2	2.074	171	1.94	11367.905	0.01
3893	3.0003	3.1112	257	2.94	11474.118	0.01
2920	4	4.1479	342	3.98	11621.000	0.02
2336	5	5.1849	428	5.06	11833.877	0.01

1947	5.999	6.2209	514	6.24	12142.376	0.01
1669	6.9982	7.257	599	7.58	12664.759	0.01
1460	8	8.2959	685	9.48	13846.543	0.01
1298	8.9985	9.3313	770	13.73	17840.278	0.02
1284	9.0966	9.433	779	14.73	18917.465	0.02
1270	9.1969	9.537	787	15.98	20318.735	0.02
1256	9.2994	9.6433	796	17.74	22299.341	0.01
1243	9.3966	9.7442	805	20.31	25243.577	0.01
1229	9.5037	9.8552	814	25.77	31677.196	0.03
1217	9.5974	9.9523	822	35.06	42660.631	0.02
1204	9.701	10.0598	831	51.87	62466.981	0.06
1192	9.7987	10.1611	839	101.21	120958.109	0.268
1180	9.8983	10.2644	847	442.71	528771.961	1.152
1168	10	10.3699	856	856.98	1022677.359	2.105
1062	10.9981	11.4049	942	3876.87	4624821.867	11.011
973	12.0041	12.4481	1028	4588.84	5479885.160	18.541
898	13.0067	13.4878	1114	4859.68	5797795.877	24.758
834	14.0048	14.5228	1199	4998.91	5964568.493	30.100
779	14.9936	15.5481	1284	5081.93	6066291.390	34.756

Table 2-2: Packet arrival rate, average number of packets in the system, end-to-end delay and packet loss rate.

Calculation

Calculation done for AIAT 11680µs Sample:

$$\text{Arrival Rate} = \frac{1sec}{IAT} = \frac{1000000}{11680} = 86 \text{ pkts/Sec}$$

$$\text{Packet loss} = \frac{Packet\ genereted - packet\ received}{Packet\ generated} = \frac{8562 - 8561}{8562} \times 100 = 0.01\%$$

$$\text{average number of packets in the system} = \text{Arrival rate} \times \text{Dealy} \times (1 - \text{Packet loss})$$

$$Packet\ loss = \frac{packet\ loss(\%)}{100} = \frac{0.01}{100} = 0.0001$$

End to End packet delay in µs, Convert µs into sec

$$= 11282.188 \text{ µs} = 0.011282188 \text{ Sec}$$

Therefore

$$\text{average number of packets in the system} = \text{Arrival rate} \times \text{Dealy} \times (1 - \text{Packet loss})$$

$$= 86 \times 0.011282188 \times (1 - 0.0001)$$

$$= 86 \times 0.011282188 \times 0.9999$$

$$= 0.97$$

We can infer the following from Table 2-2.

- The average end-to-end packet delay (between the source and the destination) is bounded below by the sum of the packet transmission durations and the propagation delays of the constituent links (2 × 12 + 1211 + 10000 microseconds).
- As the input flow rate increases, the packet delay increases as well (due to congestion and queueing in the intermediate routers). As the input flow rate matches or exceeds the bottleneck link capacity, the end-to-end packet delay increases unbounded (limited by the buffer size).
- The average number of packets in the network can be found to be equal to the product of the average end-to-end packet delay and the average input flow rate into the network. This is a validation of the Little's law. In cases where the packet loss rate is positive, the arrival rate is to be multiplied by (1 - packet loss rate).

In Table 2-3, we report the average queue and average queueing delay at the intermediate routers (Wired Node 1, Router 3 and Router 4) and the average end-to-end packet delay as a function of the input flow rate.

Input Flow Rate (in Mbps)	Arrival Rate (in Pkts/sec)	Avg no of packets in Queue			Average Queueing Delay (in µs)			End-to-End Packet Delay (in µs)
		N	Router	Router 4	N	Router 3	Rout	
1.037	86	0	0	0	0	67.55	0	11282.18
2.074	171	0	0	0	0	153.26	0	11367.90
3.1112	257	0	0.08	0	0	259.47	0	11474.11
4.1479	342	0	0.13	0	0	406.35	0	11621.00
5.1849	428	0	0.26	0	0	619.21	0	11833.87
6.2209	514	0	0.45	0	0	927.70	0	12142.37
7.257	599	0	0.92	0	0	1450.08	0	12664.75
8.2959	685	0	1.82	0	0	2631.85	0	13846.54
9.3313	770	0	5.14	0	0	6625.58	0	17840.27
9.433	779	0	6.86	0	0	7702.77	0	18917.46
9.537	787	0	7.98	0	0	9104.04	0	20318.73
9.6433	796	0	7.82	0	0	11084.6	0	22299.34
9.7442	805	0	10.96	0	0	14028.8	0	25243.57
9.8552	814	0	16.12	0	0	20462.4	0	31677.19
9.9523	822	0	25.73	0	0	31445.9	0	42660.63
10.059	831	0	42.86	0	0	51252.2	0	62466.98
10.161	839	0	91.08	0	0	109743.	0	120958.1
10.264	847	0	434	0	0	517557.	0	528771.9
10.369	856	0	849.1	0	0	101146	0	1022677.
11.404	942	0	3873.	0	0	461360	0	4624821.
12.448	1028	0	4593.	0	0	546867	0	5479885.
13.487	1114	0	4859.	0	0	578658	0	5797795.
14.522	1199	0	5000.	0	0	595335	0	5964568.
15.548	1284	0	5084.	0	0	605507	0	6066291.

Table 2-3: Average queue and average queueing delay in the intermediate buffers and end-to-end packet delay

We can infer the following from Table 2-3.

- There is queue buildup as well as queueing delay at Router_3 (Link 2) as the input flow rate increases. Clearly, link 2 is the bottleneck link where the packets see large queueing delay.

- As the input flow rate matches or exceeds the bottleneck link capacity, the average queueing delay at the (bottleneck) server increases unbounded. Here, we note that the maximum queueing delay is limited by the buffer size (8 MB) and link capacity (10 Mbps), and an upper bounded is $8 \times 1024 \times 1024 \times 8 \ 107 = 6.7 \ seconds$.

- The average number of packets in a queue can be found to be equal to the product of the average queueing delay and the average input flow rate into the network. This is again a validation of the Little's law. In cases where the packet loss rate is positive, the arrival rate is to be multiplied by $(1 - packet \ loss \ rate)$.

- The average end-to-end packet delay can be found to be equal to the sum of the packet transmission delays (12.112µs (link 1), 1211µs (link 2), 12.112 µs (link3)), propagation delay (10000 µs) and the average queueing delay in the three links.

For the sake of the readers, we have made the following plots for clarity. In Figure 2-26, we plot the average end-to-end packet delay as a function of the traffic generation rate. We note that the average packet delay increases unbounded as the traffic generation rate matches or exceeds the bottleneck link capacity.

a) Linear Scale   b) Log Scale
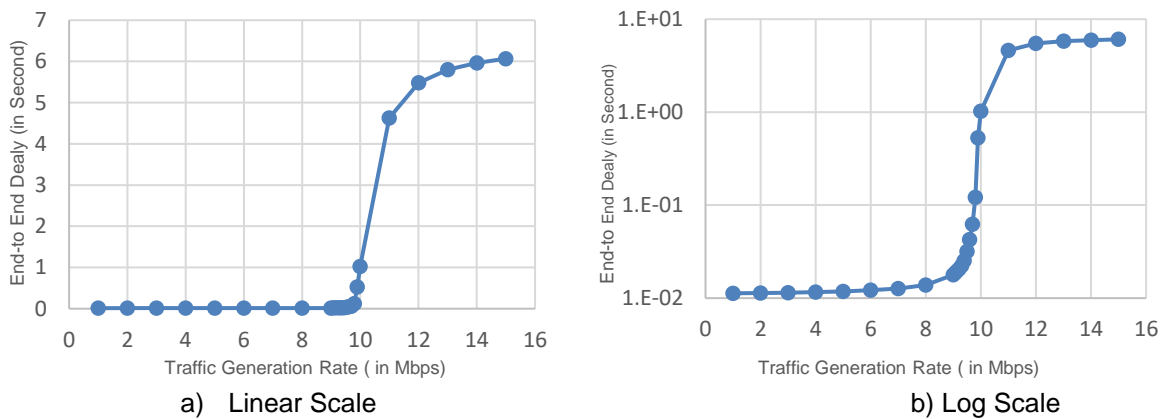
Figure 2-26: Average end-to-end packet delay as a function of the traffic generation rate.

In Figure 2-27, we plot the queueing delay experienced by few packets at the buffers of Links 1 and 2 for two different input flow rates. We note that the packet delay is a stochastic process and is a function of the input flow rate and the link capacity as well.
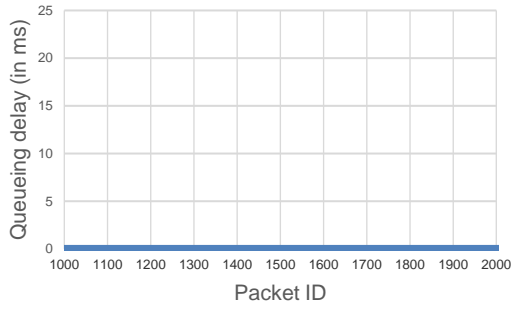
a) At Wired Node 1 for TGR = 8 Mbps
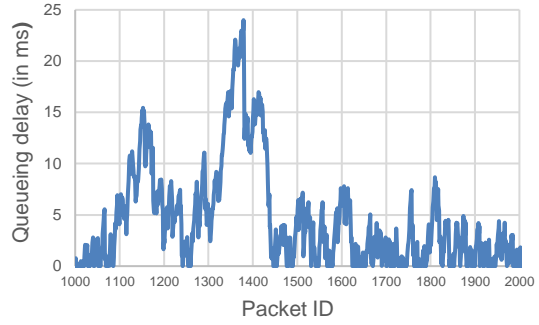
b) At Router 3 for TGR = 8 Mbps
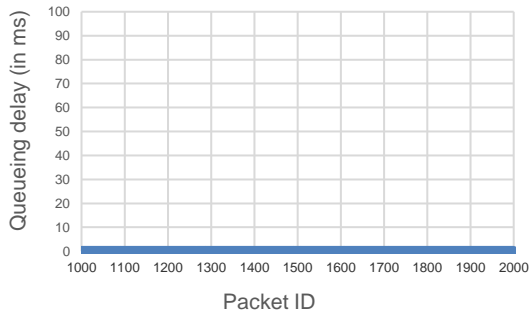
c) At Wired Node 1 for TGR = 9.5037 Mbps
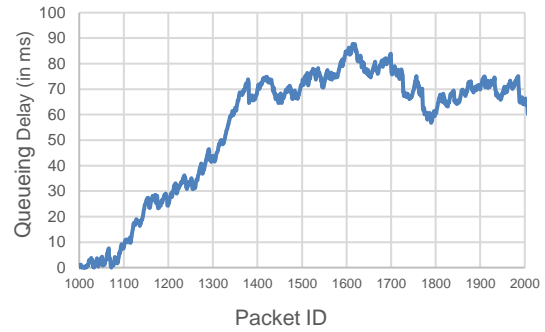
d) At Router 3 for TGR = 9.5037 Mbps

Figure 2-27: Queueing Delay of packets at Wired_Node_1 (Link 1) and Router_3 (Link 2) for two different traffic generation rates

2.3.3.3.1 Bottleneck Server Analysis as M/G/1 Queue

Suppose that the application packet inter-arrival time is i.i.d. with exponential distribution. From the M/G/1 queue analysis (in fact, M/D/1 queue analysis), we know that the average queueing delay at the link buffer (assuming large buffer size) must be.

$$\text{average queueing delay} \; = \; \frac{1}{\mu} + \frac{1}{2\mu}\frac{\rho}{1-\rho} = \; \lambda \times \text{average queue}$$

where ρ is the offered load to the link, λ is the input flow rate in packet arrivals per second and μ is the service rate of the link in packets served per second. Notice that the average queueing delay increases unbounded as $\rho \to 1$.
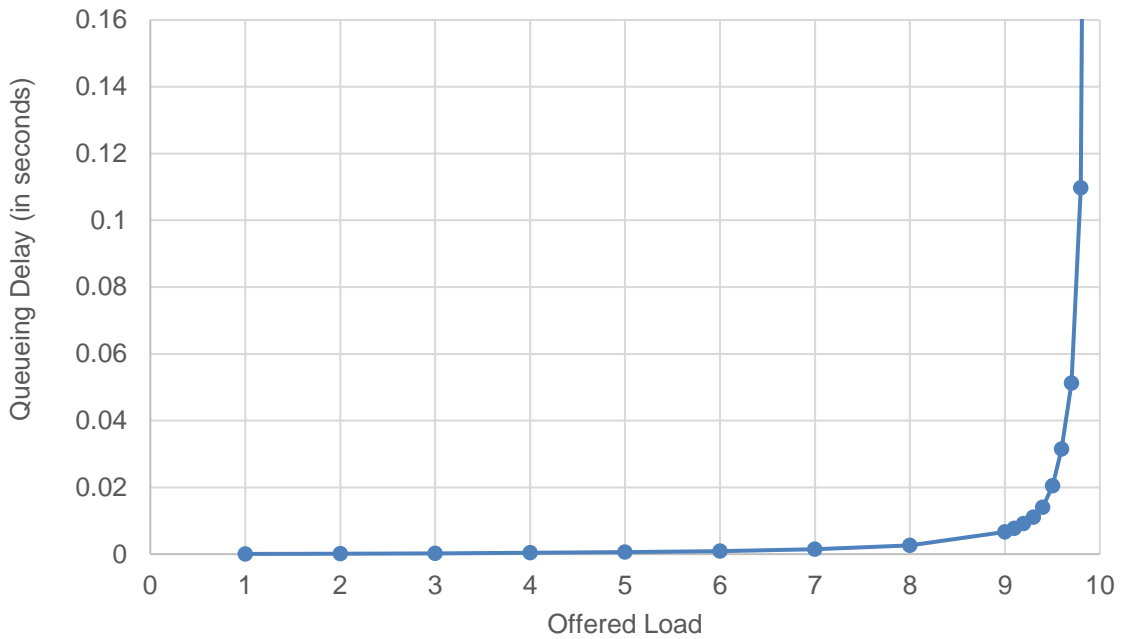
Figure 2-28: Average queueing delay (in seconds) at the bottleneck link 2 (at Router 3) Average queueing delay (in seconds) at the bottleneck link 2 (at Router 3) as a function of the offered load.

In Figure 2-28, we plot the average queueing delay (from simulation) and from (1) (from the bottleneck analysis) as a function of offered load ρ. Clearly, the bottleneck link analysis predicts the average queue (from simulation) very well. Also, we note from (1) that the network performance depends on λ and μ as $\frac{\lambda}{\mu} = \rho$ only.

2.4 Throughput and Bottleneck Server Analysis (Level 2)

2.4.1 Introduction

An important measure of quality of a network is the maximum throughput available to an application process (we will also call it a flow) in the network. **Throughput** is commonly defined as the rate of transfer of application payload through the network, and is often computed as

$$\text{Throughput} = \frac{\text{application bytes transferred}}{\text{Transferred duration}} \text{ bps}$$

2.4.1.1 A Single Flow Scenario
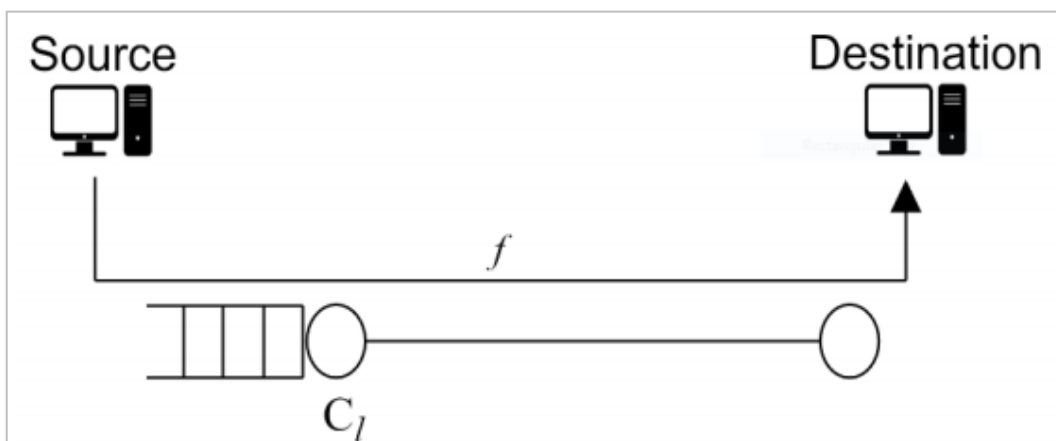
Figure 2-29: A flow f passing through a link l of fixed capacity C_l.

Application throughput depends on a lot of factors including the nature of the application, transport protocol, queueing and scheduling policies at the intermediate routers, MAC protocol and PHY parameters of the links along the route, as well as the dynamic link and traffic profile in the network. A key and a fundamental aspect of the network that limits or determines application throughput is the capacity of the constituent links (capacity may be defined at MAC/PHY layer). Consider a flow f passing through a link l with fixed capacity C_l bps. Trivially, the amount of application bytes transferred via the link over a duration of T seconds is upper bounded by $C_l \times T$ bits. Hence,

$$\text{Throughput} = \frac{\text{application bytes transferred}}{\text{Transferred duration}} \leq C_l \ bps$$

The upper bound is nearly achievable if the flow can generate sufficient input traffic to the link. Here, we would like to note that the actual throughput may be slightly less than the link capacity due to overheads in the communication protocols.
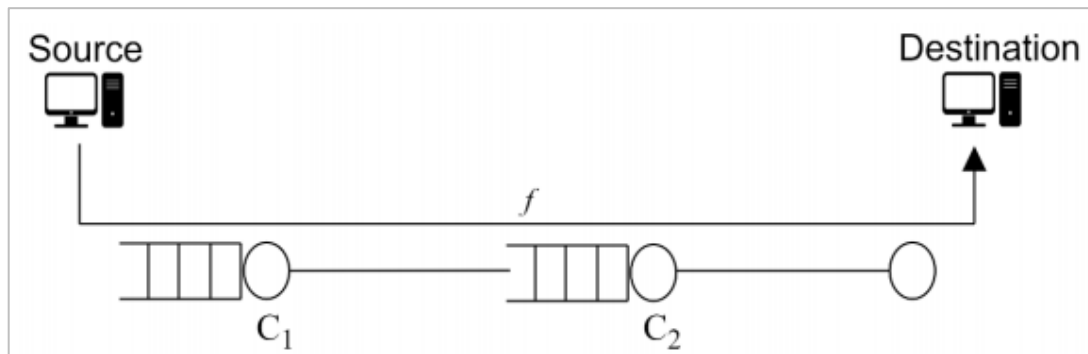
Figure 2-30: A single flow f passing through a series of links. The link with the least capacity will be identified as the bottleneck link for the flow f

If a flow f passes through multiple links $l \in L_f$ (in series), then, the application throughput will be limited by the link with the least capacity among them, i.e.,

$$\text{throughput} \leq \{ \min_{l \in L_f} C_l \} \text{ bps}$$

The link $l_f^* = \arg\min_{l \in L_f} C_l$ may be identified as the bottleneck link for the flow f. Typically, a server or a link that determines the performance of a flow is called as the bottleneck server or bottleneck link for the flow. In the case where a single flow f passes through multiple links (L_f) in series, the link l_f^* will limit the maximum throughput achievable and is the bottleneck link for the flow f. A noticeable characteristic of the bottleneck link is queue (of packets of the flow) build-up at the bottleneck server. The queue tends to increase with the input flow rate and is known to grow unbounded as the input flow rate matches or exceeds the bottleneck link capacity.
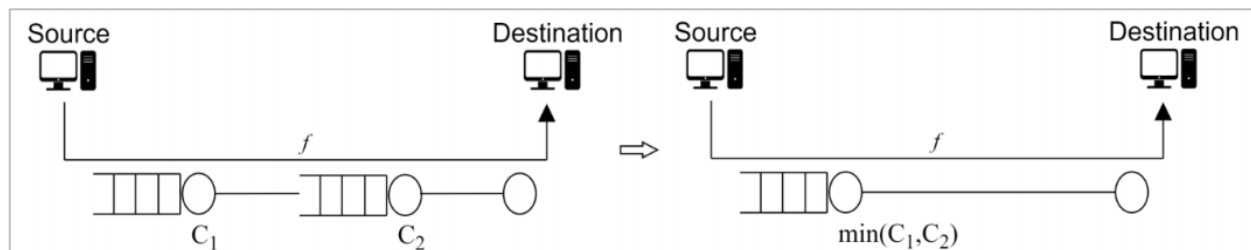
Figure 2-31: Approximation of a network using bottleneck server technique

It is a common and a useful technique to reduce a network into a bottleneck link (from the perspective of a flow(s)) to study throughput and queue buildup. For example, a network with two links (in series) can be approximated by a single link of capacity $\min(C1, C2)$ as illustrated in Figure 2-31. Such analysis is commonly known as bottleneck server analysis. Single server queueing models such as M/M/1, M/G/1, etc. can provide tremendous insights on the flow and network performance with the bottleneck server analysis.

2.4.1.2 Multiple Flow Scenario
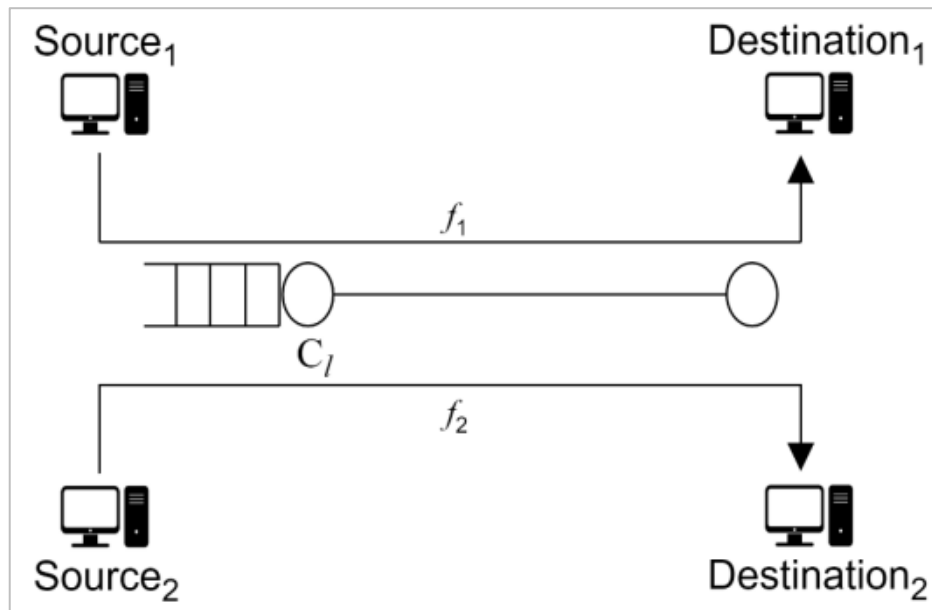
Figure 2-32: Two flows f_1 and f_2 passing through a link l of capacity C_l

Consider a scenario where multiple flows compete for the network resources. Suppose that the flows interact at some link buffer/server, say $\hat{l}$, and compete for capacity. In such scenarios, the link capacity $C_{\hat{l}}$ is shared among the competing flows and it is quite possible that the link can become the bottleneck link for the flows (limiting throughput). Here again, the queue tends to increase with the combined input flow rate and will grow unbounded as the combined input flow rate matches or exceeds the bottleneck link capacity. A plausible bound of throughput in this case is (under nicer assumptions on the competing flows)

$$\text{throughput} = \frac{C_{\hat{l}}}{\text{number of flows competing for capacity at link } \hat{l}} \ bps$$

2.4.2 NetSim Simulation Setup

Open NetSim and click on **Experiments> Internetworks> Network Performance> Throughput and Bottleneck Server Analysis** then click on the tile in the middle panel to load the example as shown in below Figure 2-33.
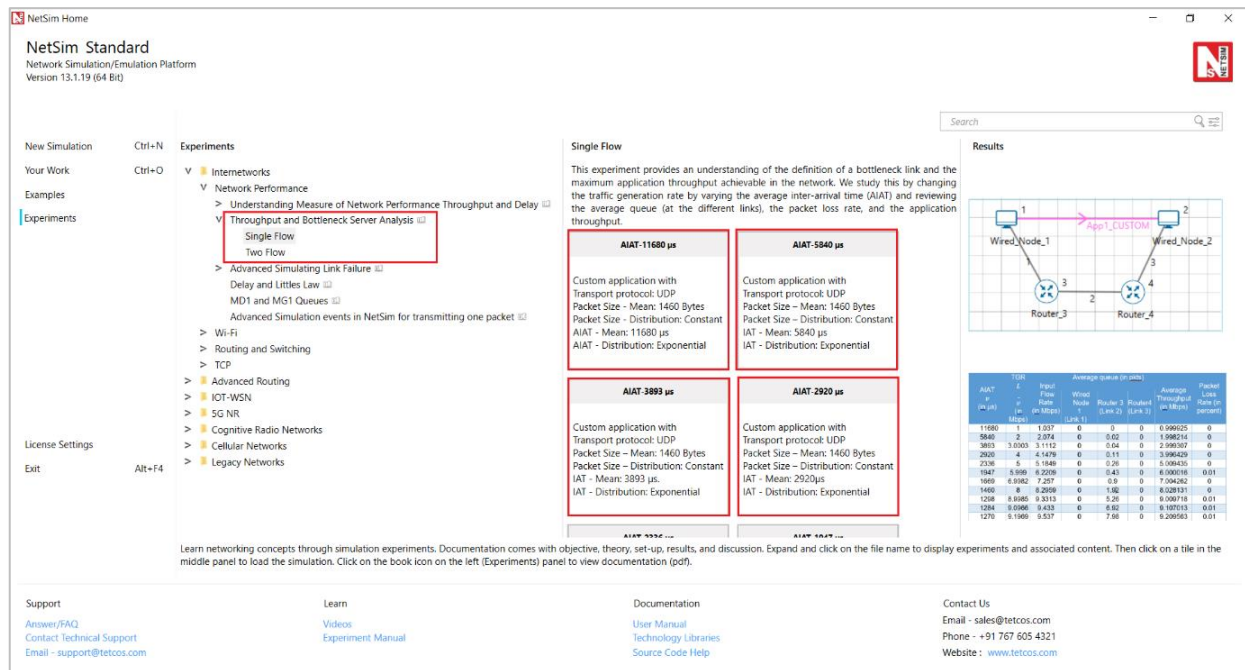
Figure 2-33: List of scenarios for the example of Throughput and Bottleneck Server Analysis

2.4.3 Part-1: A Single Flow Scenarios

We will study a simple network setup with a single flow illustrated in Figure 2-34 to review the definition of a bottleneck link and the maximum application throughput achievable in the network. An application process at Wired_Node_1 seeks to transfer data to an application process at Wired_Node_2. We consider a custom traffic generation process (at the application) that generates data packets of constant length (say, L bits) with i,i,d. inter-arrival times (say, with average inter-arrival time v seconds). The application traffic generation rate in this setup is $\frac{L}{v}$ bits per second. We prefer to minimize the communication overheads and hence, will use UDP for data transfer between the application processes.

In this setup, we will vary the traffic generation rate by varying the average inter-arrival time v and review the average queue at the different links, packet loss rate and the application throughput.

2.4.3.1 Procedure

We will simulate the network setup illustrated in Figure 2-34 with the configuration parameters listed in detail in Table 2-4 to study the single flow scenario.

NetSim UI displays the configuration file corresponding to this experiment as shown below:
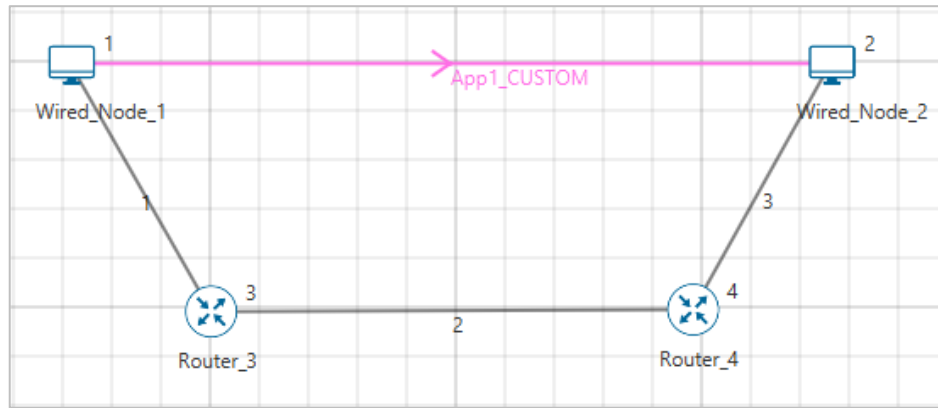
Figure 2-34: Network set up for studying a single flow

The following set of procedures were done to generate this sample.

Step 1: Drop two wired nodes and two routers onto the simulation environment. The wired nodes and the routers are connected with wired links as shown in (See Figure 2-34).

Step 2: Click the Application icon to configure a custom application between the two wired nodes. In the Application configuration dialog box (see Figure 2-35), select Application Type as **CUSTOM**, **Source ID** as **1** (to indicate Wired_Node_1), **Destination ID** as **2** (to indicate Wired_Node_2) and **Transport Protocol** as **UDP**. In the PACKET SIZE tab, select **Distribution** as **CONSTANT** and Value as **1460 bytes**. In the INTER ARRIVAL TIME tab, select **Distribution** as **EXPONENTIAL** and **Mean** as **11680** microseconds.
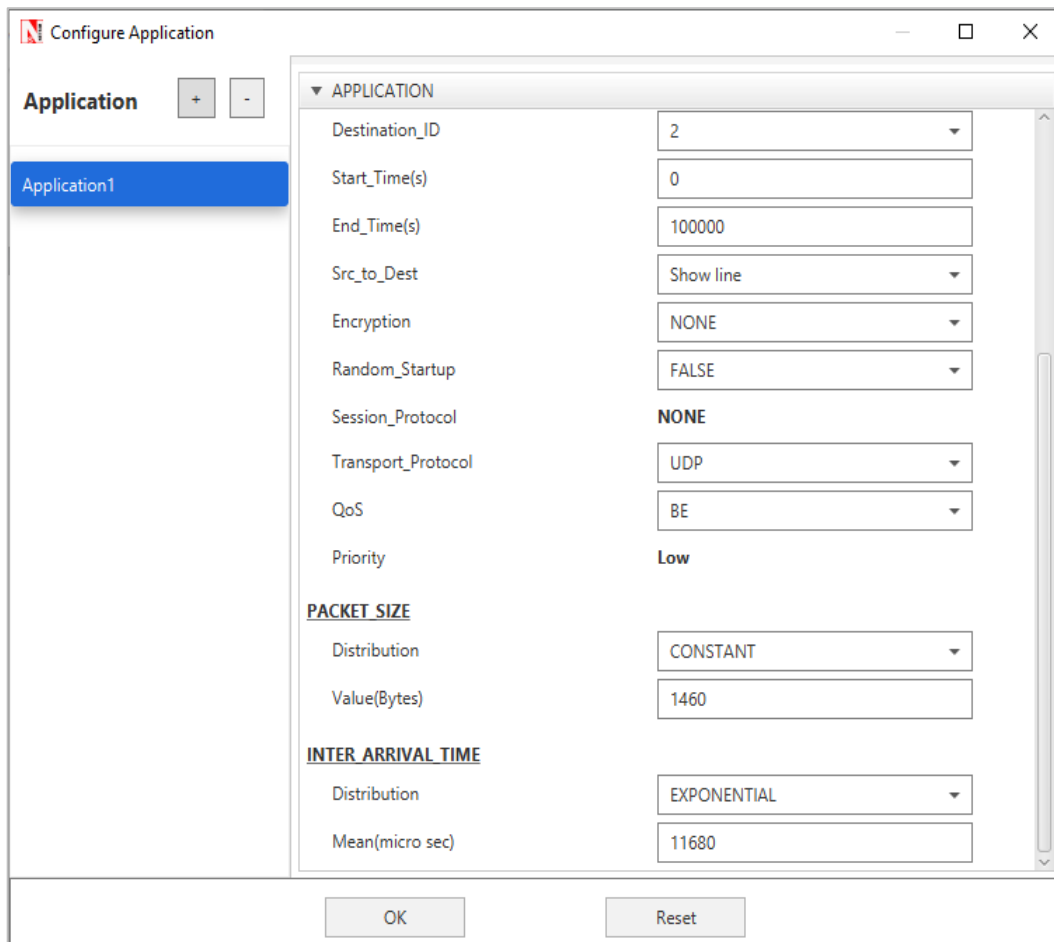
Figure 2-35: Application configuration dialog box

Step 3: The properties of the wired nodes are left to the default values.

Step 4: Right-click the link ID (of a wired link) and select **Properties** to access the link's properties dialog box (see Figure 2-36). Set **Max Uplink Speed** and **Max Downlink Speed** to 10 Mbps for link 2 (the backbone link connecting the routers) and 1000 Mbps for links 1 and 3 (the access link connecting the Wired_Nodes and the routers). Set **Uplink BER** and **Downlink BER** as 0 for links 1, 2 and 3. Set **Uplink_Propagation Delay** and **Downlink_Propagation_Delay** as 0 microseconds for the two-access links 1 and 3 and 100 microseconds for the backbone link 2.

Figure 2-36: Link Properties dialog box

Step 5: Right-click **Router 3** icon and select **Properties** to access the link's properties dialog box (see Figure 2-37). In the **INTERFACE 2 (WAN)** tab, select the **NETWORK LAYER** properties, set **Buffer size (MB)** to **8.**
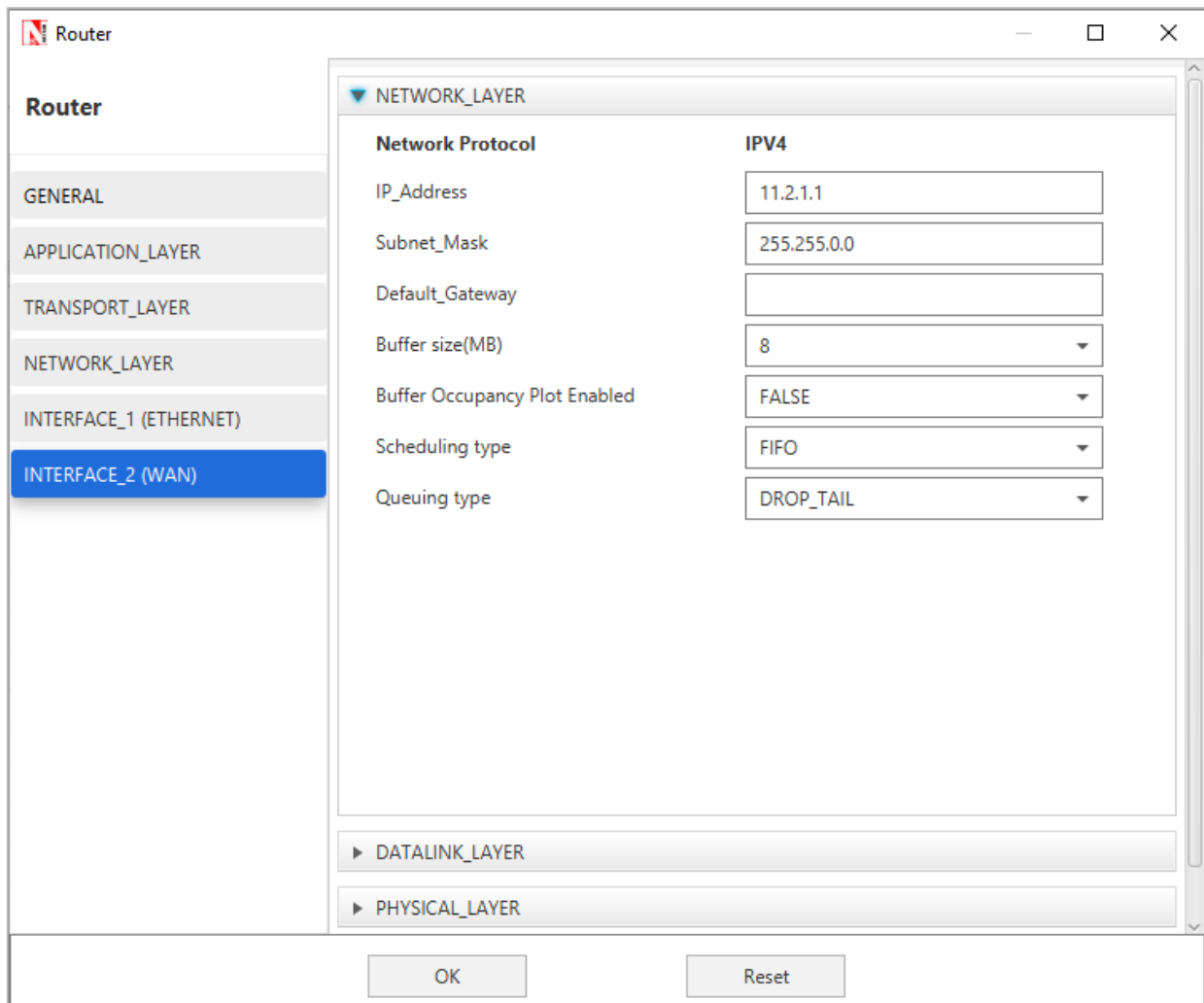
Figure 2-37: Router Properties dialog box

Step 6: Click on Packet Trace option and select the **Enable Packet Trace** check box. Packet Trace can be used for packet level analysis and Enable Plots in GUI.

Step 7: Click on **Run** icon to access the Run Simulation dialog box (see Figure 2-38) and set the **Simulation Time** to 100 seconds in the **Simulation Configuration** tab. Now, run the simulation.

Figure 2-38: Run Simulation dialog box

Step 8: Now, repeat the simulation with different average inter-arrival times (such as 5840 µs, 3893 µs, 2920 µs, 2336 µs and so on). We vary the input flow rate by varying the average inter-arrival time. This should permit us to identify the bottleneck link and the maximum achievable throughput.

The detailed list of network configuration parameters is presented in (See Table 2-4).

Parameter	Value
LINK PARAMETERS	
Wired Link Speed (access link)	1000 Mbps
Wired Link Speed (backbone link)	10 Mbps
Wired Link BER	0
Wired Link Propagation Delay (access link)	0
Wired Link Propagation Delay (backbone link)	100 µs
APPLICATION PARAMETERS	
Application	Custom
Source ID	1
Destination ID	2
Transport Protocol	UDP
Packet Size – Value	1460 bytes
Packet Size – Distribution	Constant
Inter Arrival Time – Mean	AIAT (µs) Table 2-5
Inter Arrival Time – Distribution	Exponential
ROUTER PARAMETERS	
Buffer Size	8
MISCELLANEOUS	
Simulation Time	100 Sec
Packet Trace	Enabled
Plots	Enabled

Table 2-4: Detailed Network Parameters

2.4.3.2 Performance Measure

In Table 2-5, we report the flow average inter-arrival time v and the corresponding application traffic generation rate, input flow rate (at the physical layer), average queue at the three buffers

(of Wired_Node_1, Router_3 and Router_4), average throughput (over the simulation time) and packet loss rate (computed at the destination).

Given the average inter-arrival time v and the application payload size L bits (here, 1460×8 = 11680 bits), we have,

$$\text{Traffic generation rate} = \frac{L}{v} = \frac{11680}{v} \, bps$$

$$\text{input flow rate} = \frac{11680 + 54 * 8}{v} = \frac{12112}{v} \, bps$$

where the packet overheads of 54 bytes is computed as $54 = 8(UDP\ header) + 20(IP\ header) + 26(MAC + PHY\ header)\ bytes$. Let $Q_l(u)$ as denote the instantaneous queue at link l at time u . Then, the average queue at link l is computed as

$$\text{average queue at link } l = \frac{1}{T} \int_0^T Q_l\ (u)\ \ du\ bits$$

where, T is the simulation time. The average throughput of the flow is computed as

$$\text{throughput} = \frac{\text{application byte transferred}}{T} \, bps$$

The packet loss rate is defined as the fraction of application data lost (here, due to buffer overflow at the bottleneck server).

$$\text{packet loss rate} = \frac{\text{application bytes not received at destination}}{\text{application bytes transmitted at source}}$$

2.4.3.2.1 Average Queue Computation from Packet Trace

- Open Packet Trace file using the **Open Packet Trace** option available in the Simulation Results window.
- Click on below highlighted icon to create new Pivot Table.

Figure 2-39: Packet Trace

- Click on Insert on Top ribbon → Select Pivot Table.
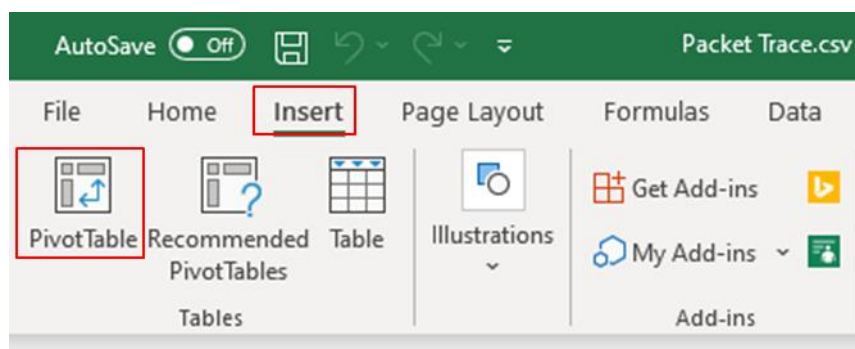
Figure 2-40: Top Ribbon

- Then select packet trace and press Ctrl + A → Select ok

Figure 2-41: Packet Trace Pivot Table

- Then we will get blank Pivot table.
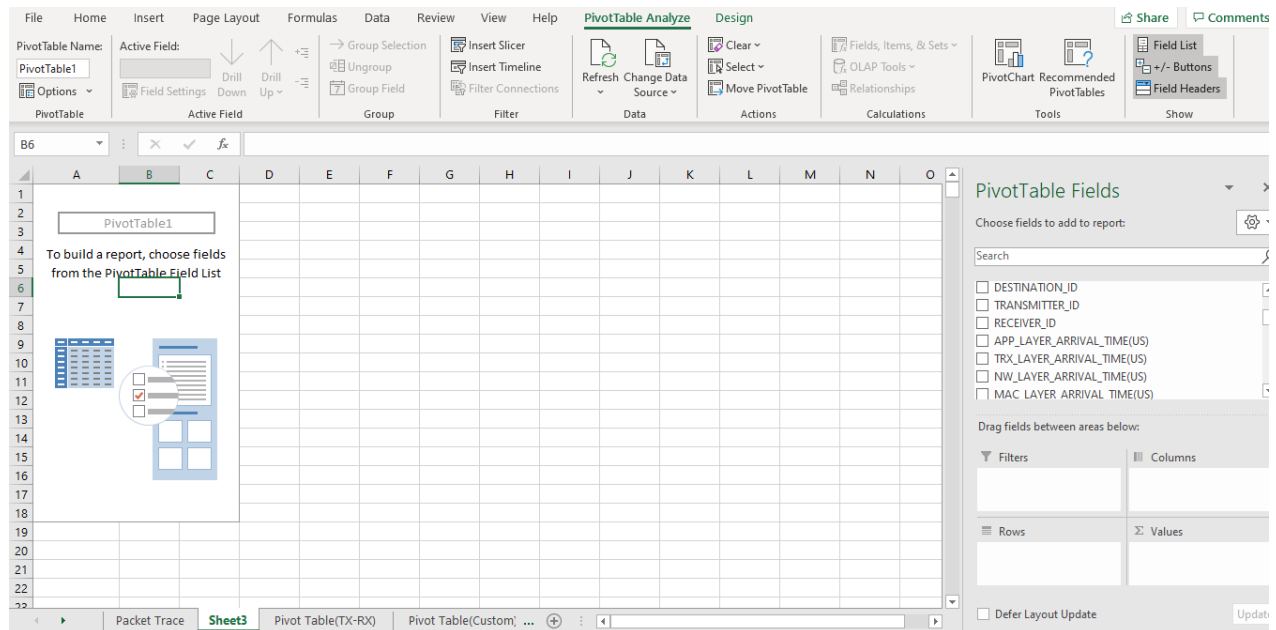
Figure 2-42: Blank Pivot Table

- **Packet ID** drag and drop into **Values field** for 2 times, **CONTROL PACKET TYPE/APP NAME, TRANSMITTER ID, RECEIVER ID** into **Filter field, NW_LAYER_ARRIVAL_TIME (US)** to **Rows field** see Figure 2-43.

- Change **Sum of PACKET ID** -> Values Field Settings **->**Select **Count -> ok** for both Values field**, CONTROL PACKET TYPE** to **APP1 CUSTOM, TRANSMITTER ID** to **Router_3 and RECEIVER ID to Router 4**
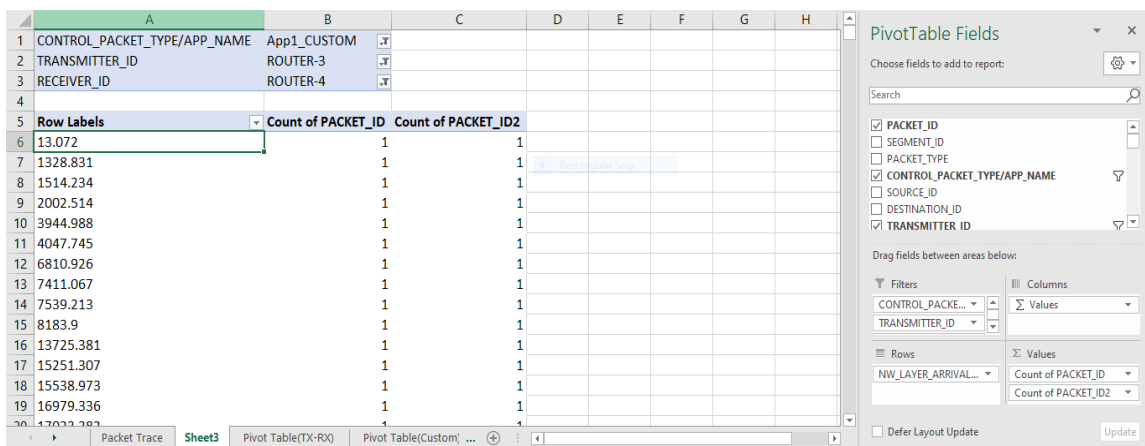
Figure 2-43: Adding fields into Filter, Columns, Rows and Values

- Right click on first value of Row Labels ->Group ->Select **By** value as 1000000.

- Go to Values field under left click on Count of PACKET ID2 ->Values Field Settings-> click on show values as -> **Running total in**-> click on OK.

- Again, create one more Pivot Table, Click on Insert on Top ribbon → Select Pivot Table.

- Then select packet trace and press Ctrl + A → Select ok

- Then we will get blank Pivot table see Figure 2-44.

- **Packet ID** drag and drop into **Values field** for 2 times, **CONTROL PACKET TYPE/APP NAME,** **TRANSMITTER** **ID,** **RECEIVER** **ID** into **Filter** **field,** PHY_LAYER_ARRIVAL_TIME

    (US) to **Rows field** see Figure 2-44,

- Change **Sum of PACKET ID** -> Values Field Settings **->**Select **Count -> ok** for both Values field, **CONTROL PACKET TYPE** to **APP1 CUSTOM, TRANSMITTER ID** to **Router_3 and RECEIVER ID to Router 4**

- Right click on first value of Row Labels for second Pivot Table->Group ->Select **by** value as 1000000.

	A	B	C	D	E	F
1	CONTROL_PACKET_TYPE/APP_NAME	App1_CUSTOM			CONTROL_PACKET_TYPE/APP_NAME	App1_CUSTOM
2	TRANSMITTER_ID	ROUTER-3			TRANSMITTER_ID	ROUTER-3
3	RECEIVER_ID	ROUTER-4			RECEIVER_ID	ROUTER-4
4						
5	Row Labels		Count of PACKET_ID	Count of PACKET_ID2	Row Labels	Count of PACKET_ID Cou
6	0-1000000		164	164	51.2	1
7	1000000-2000000		188	352	5278.987	1
8	2000000-3000000		179	531	6469.387	1
9	3000000-4000000		147	678	7973.721	1
10	4000000-5000000		162	840	15743.616	1
11	5000000-6000000		155	995	16934.016	1
12	6000000-7000000		186	1181	27207.368	1
13	7000000-8000000		211	1392	29607.933	1
14	8000000-9000000		184	1576	30798.333	1
15	9000000-10000000		171	1747	32699.265	1
16	10000000-11000000		178	1925	54865.187	1
17	11000000-12000000		173	2098	60968.894	1
18	12000000-13000000		179	2277	62159.294	1
19	13000000-14000000		161	2438	67881.008	1
20	14000000-15000000		172	2610	69071.408	1
21	15000000-16000000		153	2763	84140.499	1
22	16000000-17000000		172	2935	86615.803	1
23	17000000-18000000		143	3078	87806.202	1

Figure 2-44: Create one more Pivot Table and Add All Fields

- Go to Values field under left click on Count of PACKET ID ->Values Field Settings-> click on show values as -> **Running total in**-> click on OK.

- Calculate the average queue by taking the mean of the number of packets in queue at every time interval during the simulation.

- The difference between the **count of PACKET ID2 (Column C)** and **count of PACKET ID2 (Column G),** Note down the average value for difference see Figure 2-45

	A	B	C	D	E	F	G	H	I	J
1	CONTROL_PACKET_TYPE/APP_NAME	App1_CUSTOM			CONTROL_PACKET_TYPE/APP_NAME	App1_CUSTOM				
2	TRANSMITTER_ID	ROUTER-3			TRANSMITTER_ID	ROUTER-3				
3	RECEIVER_ID	ROUTER-4			RECEIVER_ID	ROUTER-4				
4										
5	Row Labels		Count of PACKET_ID	Count of PACKET_ID2	Row Labels		Count of PACKET_ID	Count of PACKET_ID2	#VALUE!	
6	0-1000000		164	164	0-1000000		164	164	0	
7	1000000-2000000		188	352	1000000-2000000		188	352	0	
8	2000000-3000000		179	531	2000000-3000000		179	531	0	
9	3000000-4000000		147	678	3000000-4000000		147	678	0	
10	4000000-5000000		162	840	4000000-5000000		162	840	0	
11	5000000-6000000		155	995	5000000-6000000		155	995	0	
12	6000000-7000000		186	1181	6000000-7000000		186	1181	0	
13	7000000-8000000		211	1392	7000000-8000000		211	1392	0	
14	8000000-9000000		184	1576	8000000-9000000		184	1576	0	
15	9000000-10000000		171	1747	9000000-10000000		171	1747	0	
16	10000000-11000000		178	1925	10000000-11000000		178	1925	0	
17	11000000-12000000		173	2098	11000000-12000000		173	2098	0	
18	12000000-13000000		179	2277	12000000-13000000		179	2277	0	
19	13000000-14000000		161	2438	13000000-14000000		161	2438	0	
20	14000000-15000000		172	2610	14000000-15000000		172	2610	0	
21	15000000-16000000		153	2763	15000000-16000000		153	2763	0	
22	16000000-17000000		172	2935	16000000-17000000		172	2935	0	
23	17000000-18000000		143	3078	17000000-18000000		143	3078	0	

Packet Trace | **Sheet3** | Pivot Table(TX-RX) | Pivot Table(Custom)

Average: 0.02   Count: 100   Sum: 2

Figure 2-45: Average Packets in Queue

$$\textbf{Packet Loss Rate (in percent)} = \frac{\textbf{Packet Generated} - \textbf{Packet Received}}{\textbf{Packet Generated}} \times 100$$

2.4.3.3 Results

In Table 2-5, we report the flow average inter-arrival time (AIAT) and the corresponding application traffic generation rate (TGR), input flow rate, average queue at the three buffers (of Wired_Node_1, Router_3 and Router_4), average throughput and packet loss rate.

AIAT v (in µs)	TGR $\frac{L}{v}$ (in Mbps)	Input Flow Rate (in Mbps)	Average queue (in pkts)			Average Throughput (in Mbps)	Packet Loss Rate (in percent)
			Wired Node 1 (Link 1)	Router 3 (Link 2)	Router 4 (Link 3)		
11680	1	1.037	0	0	0	0.999925	0
5840	2	2.074	0	0.02	0	1.998214	0
3893	3.0003	3.1112	0	0.04	0	2.999307	0
2920	4	4.1479	0	0.11	0	3.996429	0
2336	5	5.1849	0	0.26	0	5.009435	0
1947	5.999	6.2209	0	0.43	0	6.000016	0.01
1669	6.9982	7.257	0	0.9	0	7.004262	0
1460	8	8.2959	0	1.92	0	8.028131	0
1298	8.9985	9.3313	0	5.26	0	9.009718	0.01
1284	9.0966	9.433	0	6.92	0	9.107013	0.01
1270	9.1969	9.537	0	7.98	0	9.209563	0.01
1256	9.2994	9.6433	0	7.88	0	9.314683	0
1243	9.3966	9.7442	0	11.48	0	9.416182	0.01
1229	9.5037	9.8552	0	16.26	0	9.520718	0.02
1217	9.5974	9.9523	0	25.64	0	9.616027	0.01
1204	9.701	10.0598	0	42.88	0	9.717994	0.05
1192	9.7987	10.1611	0	90.86	0	9.796133	0.26
1180	9.8983	10.2644	0	436.41	0	9.807696	1.15
1168	10	10.3699	0	847.65	0	9.808981	2.09
1062	10.9981	11.4049	0	3876.87	0	9.811667	11.00
973	12.0041	12.4481	0	4593.67	0	9.811667	18.53
898	13.0067	13.4878	0	4859.68	0	9.811667	24.75
834	14.0048	14.5228	0	5000.57	0	9.811667	30.09
779	14.9936	15.5481	0	5085.05	0	9.811667	34.75

Table 2-5: Average queue, throughput and loss rate as a function of traffic generation rate

We can infer the following from Table 2-5.

- The input flow rate is slightly larger than the application traffic generation rate. This is due to the overheads in communication.

- There is queue buildup at Router 3 (Link 2) as the input flow rate increases. So, Link 2 is the bottleneck link for the flow.
- As the input flow rate increases, the average queue increases at the (bottleneck) server at Router 3. The traffic generation rate matches the application throughput (with nearly zero packet loss rate) when the input flow rate is less than the capacity of the link.
- As the input flow rate reaches or exceeds the link capacity, the average queue at the (bottleneck) server at Router 3 increases unbounded (limited by the buffer size) and the packet loss rate increases as well.

For the sake of the readers, we have made the following plots for clarity. In Figure 2-46, we plot application throughput as a function of the traffic generation rate. We note that the application throughput saturates as the traffic generate rate (in fact, the input flow rate) gets closer to the link capacity. The maximum application throughput achievable in the setup is 9.81 Mbps (for a bottleneck link with capacity 10 Mbps).
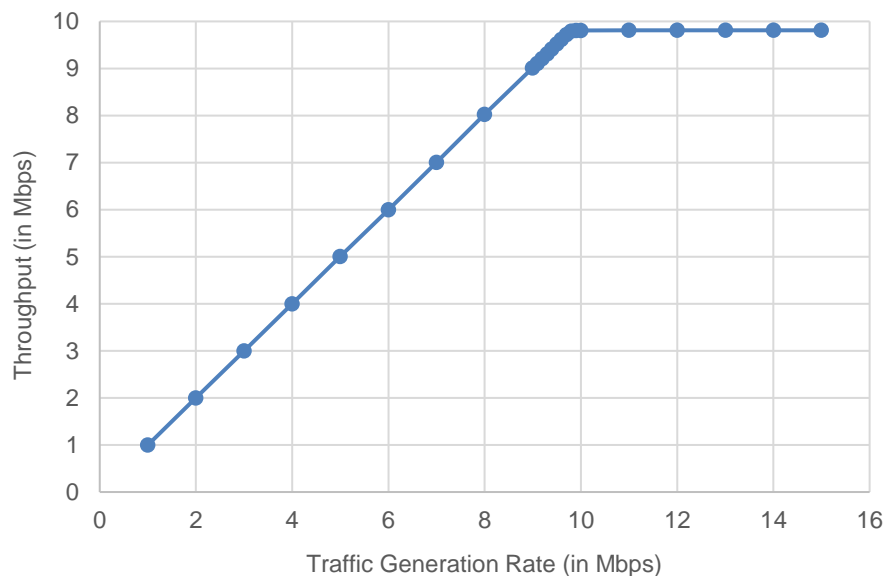
Figure 2-46: Application throughput as a function of the traffic generation rate

Figure 2-47, we plot the queue evolution at the buffers of Links 1 and 2 for two different input flow rates. We note that the buffer occupancy is a stochastic process and is a function of the input flow rate and the link capacity as well.
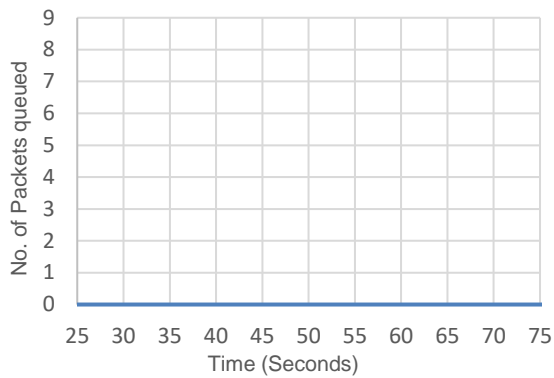
a) At Wired Node 1 for TGR = 8 Mbps
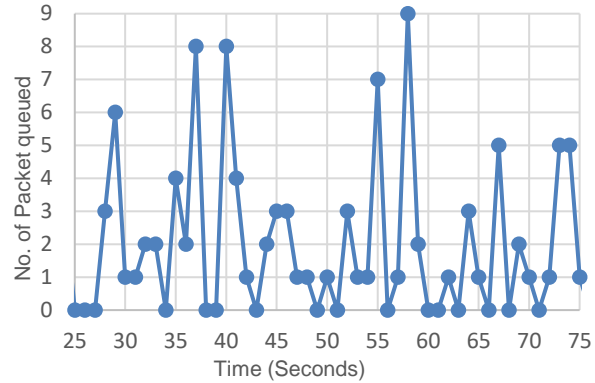
b) At Router 3 for TGR = 8 Mbps
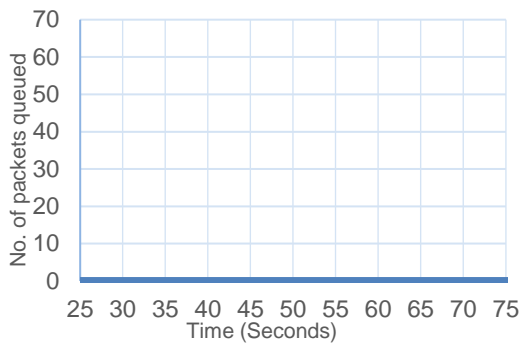
c) At Wired Node 1 for TGR = 9.5037 Mbps
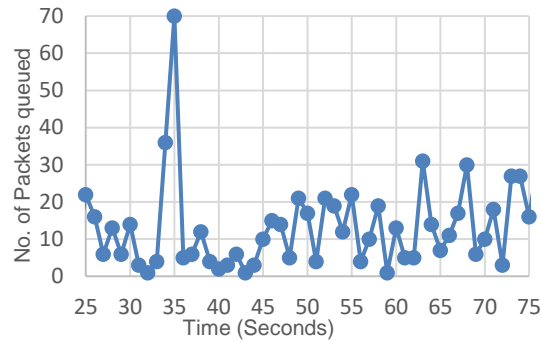
d) At Router 3 for TGR = 9.5037 Mbps

Figure 2-47: Queue evolution at Wired Node 1 (Link 1) and Router 3 (Link 2) for two different traffic generation rates

In Figure 2-48, we plot the average queue at the bottleneck link 2 (at Router 3) as a function of the traffic generation rate. We note that the average queue increases gradually before it increases unboundedly near the link capacity.
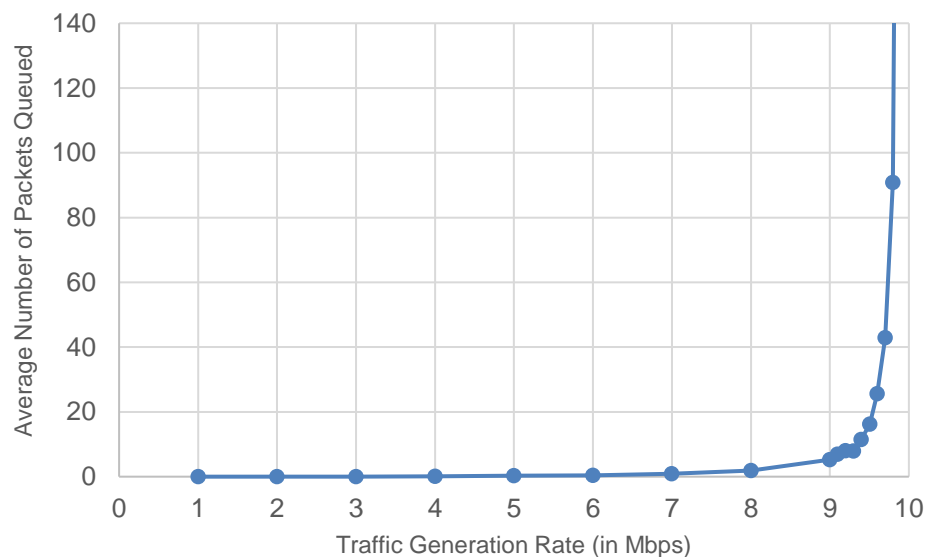
Figure 2-48: Average queue (in packets) at the bottleneck link 2 (at Router 3) as a function of the traffic generation rate

2.4.3.3.1 Bottleneck Server Analysis as M/G/1 Queue

Let us now analyze the network by focusing on the flow and the bottleneck link (Link 2). Consider a single flow (with average inter-arrival time v) into a bottleneck link (with capacity C). Let us the denote the input flow rate in packet arrivals per second as λ, where $\lambda = 1/v$. Let us also denote the service rate of the bottleneck server in packets served per second as μ, where $\mu = \frac{C}{L+54\times8}$. Then,

$$\rho = \lambda \times \frac{1}{\mu} = \frac{\lambda}{\mu}$$

denotes the offered load to the server. When $\rho < 1$, ρ also denotes (approximately) the fraction of time the server is busy serving packets (i.e., ρ denotes link utilization). When $\rho \ll 1$, then the link is barely utilized. When $\rho > 1$, then the link is said to be overloaded or saturated (and the buffer will grow unbounded). The interesting regime is when $0 < \rho < 1$.

Suppose that the application packet inter-arrival time is i.i.d. with exponential distribution. From the M/G/1 queue analysis (in fact, M/D/1 queue analysis), we know that the average queue at the link buffer (assuming large buffer size) must be.

$$\text{average queue} = \rho \times \frac{1}{2}\left(\frac{\rho^2}{1-\rho}\right), \qquad 0 < \rho < 1$$

where, ρ is the offered load. In Figure 2-48, we also plot the average queue from (1) (from the bottleneck analysis) and compare it with the average queue from the simulation. You will notice that the bottleneck link analysis predicts the average queue (from simulation) very well.

An interesting fact is that the average queue depends on λ and μ only as $\rho = \frac{\lambda}{\mu}$.

2.4.4 Part - 2: Two Flow Scenario

We will consider a simple network setup with two flows illustrated in Figure 2-49 to review the definition of a bottleneck link and the maximum application throughput achievable in the network. An application process at Wired_Node_1 seeks to transfer data to an application process at Wired_Node_2. Also, an application process at Wired_Node_3 seeks to transfer data to an application process at Wired_Node_4. The two flows interact at the buffer of Router_ 5 (Link 3) and compete for link capacity. We will again consider custom traffic generation process (at the application processes) that generates data packets of constant length (L bits) with i.i.d. inter-arrival times (with average inter-arrival time v seconds) with a common distribution. The application traffic generation rate in this setup is $\frac{L}{v}$ bits per second (for either application).

In this setup, we will vary the traffic generation rate of the two sources (by identically varying the average inter-arrival time v) and review the average queue at the different links, application throughput (s) and packet loss rate (s).

2.4.4.1 Procedure

We will simulate the network setup illustrated in Figure 2-49 with the configuration parameters listed in detail in Table 2-4 to study the two-flow scenario. We will assume identical configuration parameters for the access links and the two application processes.
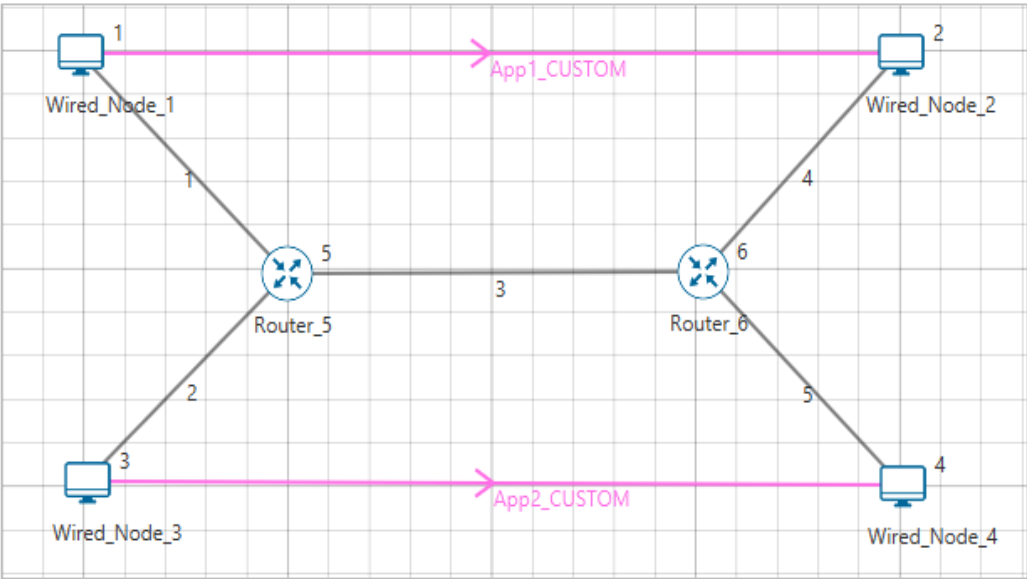
Figure 2-49: Network set up for studying two flows

Step 1: Right-click the link ID (of a wired link) and select Properties to access the link's properties dialog box. Set Max Uplink Speed and Max Downlink Speed to 10 Mbps for link 3 (the backbone link connecting the routers) and 1000 Mbps for links 1,2,4, 5 (the access link connecting the Wired Nodes and the routers). Set Uplink BER and Downlink BER as 0 for all links. Set Uplink Propagation Delay and Downlink Propagation Delay as 0 microseconds for links 1,2,4 and 5 and 100 microseconds for the backbone link 3.

Step 2: Enable Plots and Packet trace in NetSim GUI.

Step 3: Simulation time is 100 sec for all samples.

2.4.4.2 Results

In Table 2-6, we report the common flow average inter-arrival time (AIAT) and the corresponding application traffic generation rate (TGR), input flow rate, combined input flow rate, average queue at the buffers (of Wired_Node_1, Wired_Node_3 and Router_5), average throughput(s) and packet loss rate(s).

AIAT	TGR	Input Flo	Combine d	Average queue (In pkts)		Average Throughput (in Mbps)	Packet Loss Rate (In percent)

v (in µs)	$\frac{L}{v}$ (in Mbps)	w Rate (in Mbps)	Input Flow Rate (in Mbps)	Wired Node 1	Wired Node 3	Router 5	App 1 Custom	App 2 Custom	App1 Custom	App2 Custom
11680	1	1.037	2.074	0	0	0.03	0.999925	1.002728	0	0
5840	2	2.074	4.148	0	0	0.16	1.998214	2.006624	0	0
3893	3.0003	3.1112	6.2224	0	0	0.32	2.9993007	3.0014100	0	0
2920	4	4.1479	8.2958	0	0	1.99	3.9963102	4.0185094	0	0
2336	5	5.1849	10.3698	0	0	847.19	4.9036314	4.9074699	2.12	2.10
1947	5.999	6.2209	12.4418	0	0	4607.12	4.896606	4.915061	18.38	18.38
1669	6.9982	7.257	14.514	0	0	5009.33	4.896373	4.915294	30.10	30.00
1460	8	8.2959	16.5918	0	0	5150.91	4.9066418	4.9055250	38.88	38.78
1298	8.9985	9.3313	18.6626	0	0	5222.86	4.9047802	4.9066885	45.56	45.52
1168	10	10.3699	20.7398	0	0	5265.95	4.9203177	4.8913500	50.88	51.16

Table 2-6: Average queue, throughput(s) packet loss rate(s) as a function of the traffic generation

We can infer the following from Table 2-6.

1. There is queue buildup at Router_5 (Link 3) as the combined input flow rate increases. So, link 3 is the bottleneck link for the two flows.

2. The traffic generation rate matches the application throughput(s) (with nearly zero packet loss rate) when the combined input flow rate is less than the capacity of the bottleneck link.

3. As the combined input flow rate reaches or exceeds the bottleneck link capacity, the average queue at the (bottleneck) server at Router 5 increases unbounded (limited by the buffer size) and the packet loss rate increases as well.

4. The two flows share the available link capacity and see a maximum application throughput of 4.9 Mbps (half of bottleneck link capacity 10 Mbps).

2.4.5 Useful Exercises

1. Redo the single flow experiment with constant inter-arrival time for the application process. Comment on average queue evolution and maximum throughput at the links.

2. Redo the single flow experiment with small buffer size (8 KBytes) at the bottleneck link 2. Compute the average queue evolution at the bottleneck link and the average throughput of the flow as a function of traffic generation rate. Compare it with the case when the buffer size in 8 MB.

3. Redo the single flow experiment with a bottleneck link capacity of 100 Mbps. Evaluate the average queue as a function of the traffic generation rate. Now, plot the average queue as a function of the offered load and compare it with the case of bottleneck link with 10 Mbps capacity (studied in the report). Comment.

3 Routing & Switching

3.1 Study the working and routing table formation of Interior routing protocols, i.e., Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) (Level 1)

3.1.1 Introduction

3.1.1.1 RIP

RIP is intended to allow hosts and gateways to exchange information for computing routes through an IP-based network. RIP is a distance vector protocol which is based on Bellman-Ford algorithm. This algorithm has been used for routing computation in the network.

Distance vector algorithms are based on the exchange of only a small amount of information using RIP messages.

Each entity (router or host) that participates in the routing protocol is assumed to keep information about all of the destinations within the system. Generally, information about all entities connected to one network is summarized by a single entry, which describes the route to all destinations on that network. This summarization is possible because as far as IP is concerned, routing within a network is invisible. Each entry in this routing database includes the next router to which datagram's destined for the entity should be sent. In addition, it includes a "metric" measuring the total distance to the entity.

Distance is a somewhat generalized concept, which may cover the time delay in getting messages to the entity, the dollar cost of sending messages to it, etc. Distance vector algorithms get their name from the fact that it is possible to compute optimal routes when the only information exchanged is the list of these distances. Furthermore, information is only exchanged among entities that are adjacent, that is, entities that share a common network.

3.1.1.2 OSPF

In OSPF, the Packets are transmitted through the shortest path between the source and destination.

OSPF allows administrator to assign a cost for passing through a link. The total cost of a particular route is equal to the sum of the costs of all links that comprise the route. A router chooses the route with the shortest (smallest) cost.

In OSPF, each router has a link state database which is tabular representation of the topology of the network (including cost). Using Dijkstra algorithm each router finds the shortest path between source and destination.

3.1.1.3 Formation of OSPF Routing Table

1. OSPF-speaking routers send Hello packets out all OSPF-enabled interfaces. If two routers sharing a common data link agree on certain parameters specified in their respective Hello packets, they will become neighbors.

2. Adjacencies, which can be thought of as virtual point-to-point links, are formed between some neighbors. OSPF defines several network types and several router types. The establishment of an adjacency is determined by the types of routers exchanging Hellos and the type of network over which the Hellos are exchanged.

3. Each router sends link-state advertisements (LSAs) over all adjacencies. The LSAs describe all of the router's links, or interfaces, the router's neighbors, and the state of the links. These links might be to stub networks (networks with no other router attached), to other OSPF routers, or to external networks (networks learned from another routing process). Because of the varying types of link-state information, OSPF defines multiple LSA types.

4. Each router receiving an LSA from a neighbor records the LSA in its link-state database and sends a copy of the LSA to all its other neighbors.

5. By flooding LSAs throughout an area, all routers will build identical link-state databases.

6. When the databases are complete, each router uses the SPF algorithm to calculate a loop-free graph describing the shortest (lowest cost) path to every known destination, with itself as the root. This graph is the SPF tree.

7. Each router builds its route table from its SPF tree.

3.1.2 Network Setup

Open NetSim and click on **Experiments> Internetworks> Routing and Switching> Route table formation in RIP and OSPF** then click on the tile in the middle panel to load the example as shown in below Figure 3-1.
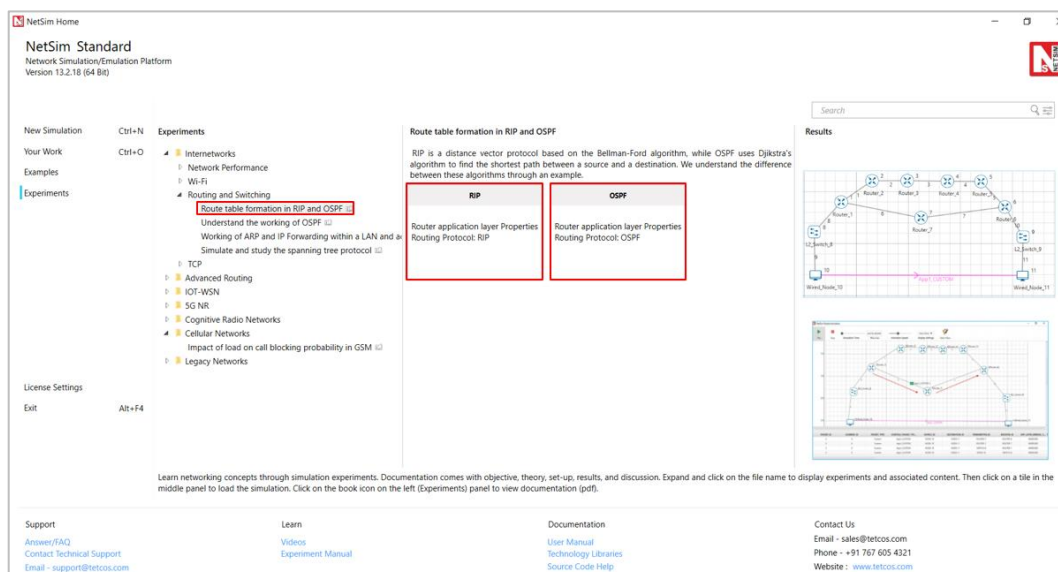
Figure 3-1: List of scenarios for the example of Route table formation in RIP and OSPF

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-2.
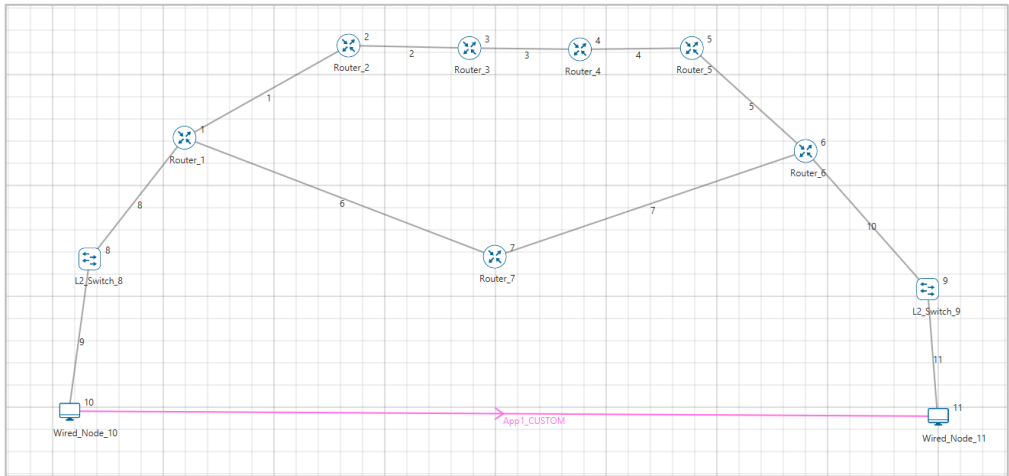
Figure 3-2: Network set up for studying the RIP/OSPF

3.1.3 Procedure

RIP

The following are the set of procedures were done to generate this sample.

Step 1: A network scenario is designed in the NetSim GUI comprising of 2 Wired Nodes, 2 L2 Switches, and 7 Routers.

Step 2: Go to Router 1 Properties. In the Application Layer, Routing Protocol is set as RIP Figure 3-3.
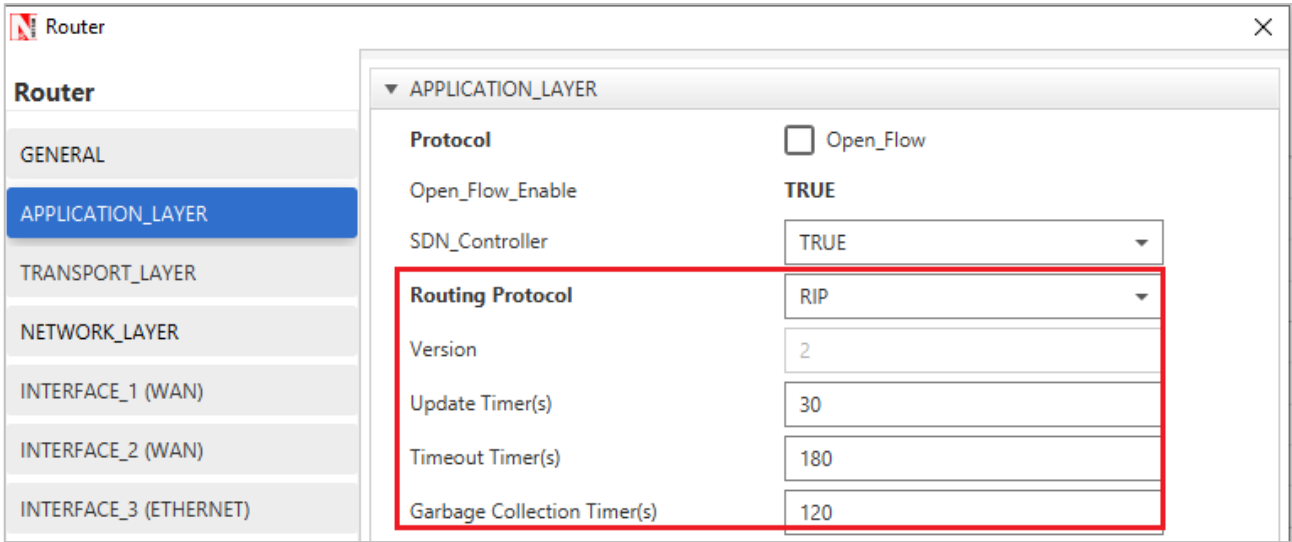
Figure 3-3: Application Layer Window - Routing Protocol is set as RIP

The Router Configuration Window shown above, indicates the Routing Protocol set as RIP along with its associated parameters. The "**Routing Protocol**" parameter is Global. i.e., changing in Router 1 will affect all the other Routers. So, in all the Routers, the Routing Protocol is now set as RIP.

Step 3: Right click on App1 CUSTOM and select Properties or click on the Application icon present in the top ribbon/toolbar. Transport Protocol is set to **UDP.**

A **CUSTOM** Application is generated from Wired Node 10 i.e., Source to Wired Node 11 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000µs.

Step 4: Packet Trace is enabled, and hence we are able to track the route which the packets have chosen to reach the destination based on the Routing Information Protocol that is set.

Step 5: Enable the plots and run the Simulation for 100 Seconds.

OSPF

The following are the set of procedures that are followed to carry out this experiment.

Step 1: A network scenario is designed in the NetSim GUI comprising of 2 Wired Nodes, 2 L2 Switches, and 7 Routers.

Step 2: Go to Router 1 Properties. In the Application Layer, Routing Protocol is set as OSPF Figure 3-4.
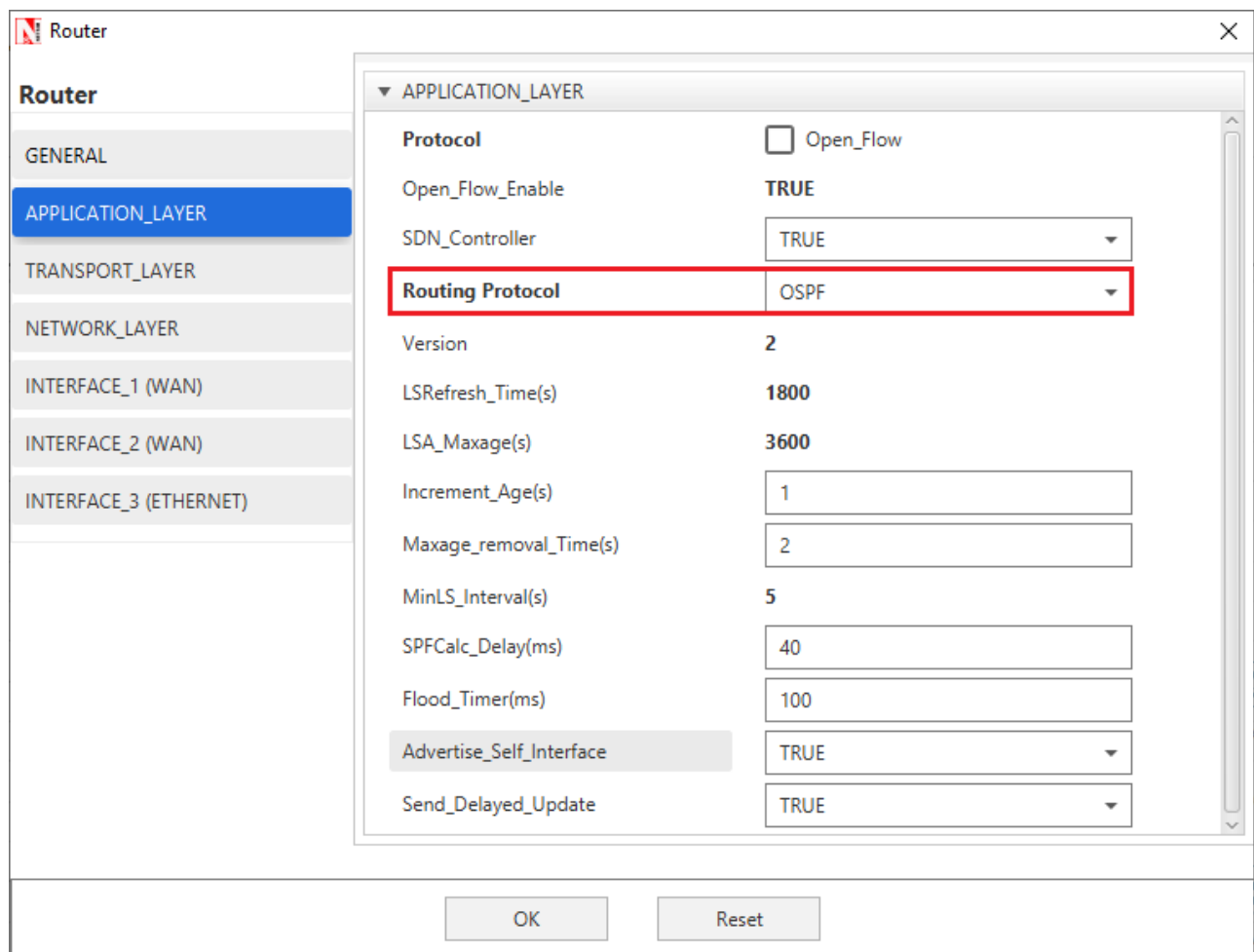
Figure 3-4: Application Layer Window - Routing Protocol is set as OSPF

The Router Configuration Window shown above, indicates the Routing Protocol set as OSPF along with its associated parameters. The "**Routing Protocol**" parameter is Global. i.e.,

changing in Router 1 will affect all the other Routers. So, in all the Routers, the Routing Protocol is now set as OSPF.

Step 3: Go to Router 7 Properties. In both the WAN Interfaces, the Output Cost is set to 2000 Figure 3-5.
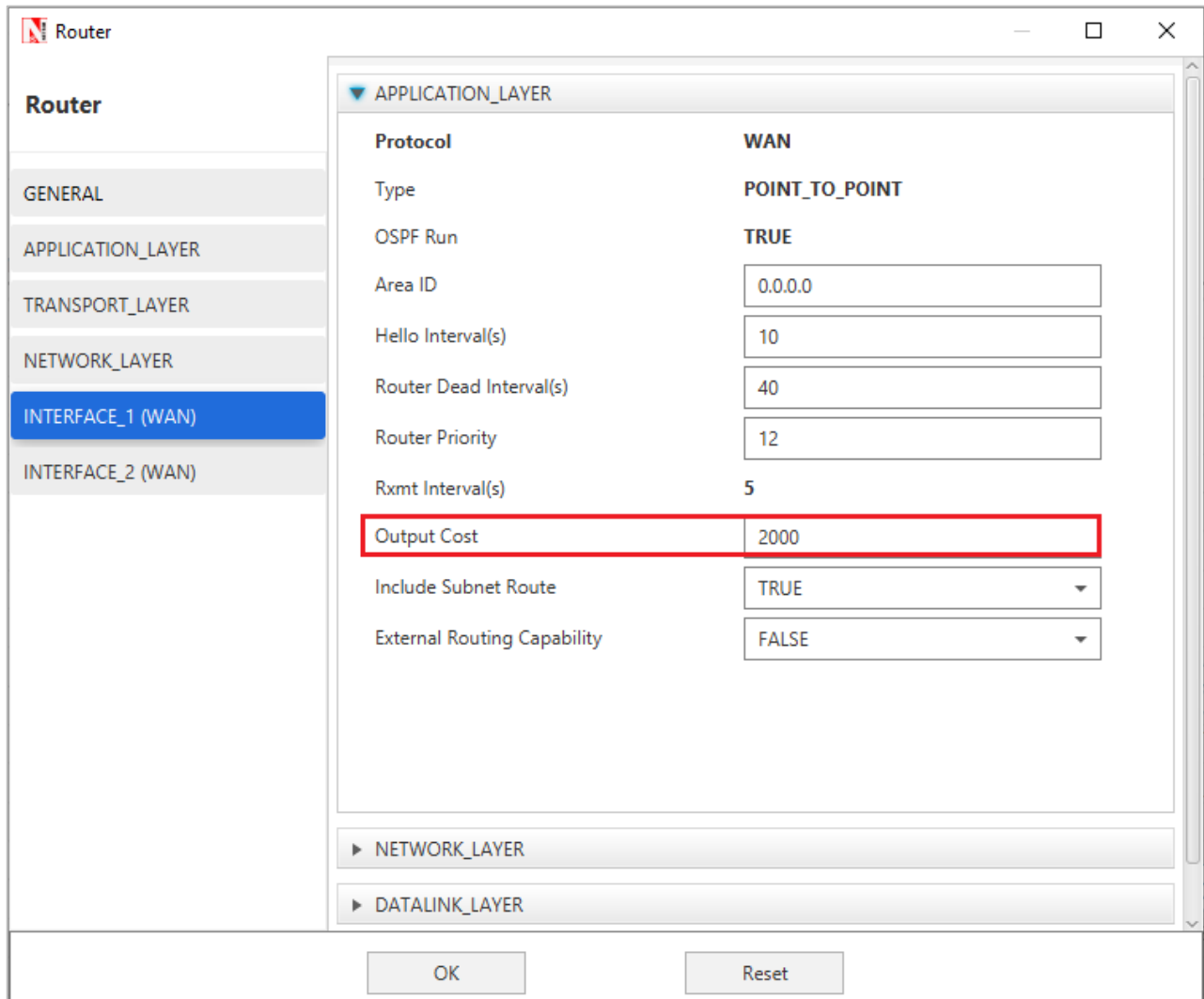
Figure 3-5: WAN Interfaces - Output Cost is set to 2000

The "**Output Cost**" parameter in the **WAN Interface > Application Layer** of a router indicates the cost of sending a data packet on that interface and is expressed in the link state metric.

Step 4: Right click on App1 CUSTOM and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wired Node 10 i.e., Source to Wired Node 11 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000μs.

Additionally, the "**Start Time (s)**" parameter is set to 40, while configuring the application. This time is usually set to be greater than the time taken for OSPF Convergence (i.e., Exchange of OSPF information between all the routers), and it increases as the size of the network increases.

Step 5: Packet Trace is enabled, and hence we are able to track the route which the packets have chosen to reach the destination based on the Open Shortest Path First Routing Protocol that is set.

Step 6: Enable the plots and run the Simulation for 100 Seconds.

3.1.4 Output for RIP

Go to NetSim Packet Animation window and play the animation. The route taken by the packets to reach the destination can be seen in the animation as well as in the below table containing various fields of packet information as shown below Figure 3-6.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYP...	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_T...	T
3	0	Custom	App1_CUSTOM	NODE-10	NODE-11	ROUTER-7	ROUTER-6	40000.000	
3	0	Custom	App1_CUSTOM	NODE-10	NODE-11	ROUTER-1	ROUTER-7	40000.000	
3	0	Custom	App1_CUSTOM	NODE-10	NODE-11	SWITCH-8	ROUTER-1	40000.000	
3	0	Custom	App1_CUSTOM	NODE-10	NODE-11	NODE-10	SWITCH-8	40000.000	

Figure 3-6: Animation window for RIP

Users can view the same in Packet Trace.

Shortest Path from Wired Node 10 to Wired Node 11 in RIP is **Wired Node 10->L2 Switch 8->Router 1->Router 7->Router 6->L2 Switch 9->Wired Node 11**. RIP chooses the lower path (number of hops is less) to forward packets from source to destination, since it is based on hop count.

3.1.5 Output for OSPF

Go to NetSim Packet Animation window and play the animation. The route taken by the packets to reach the destination can be seen in the animation as well as in the below table containing various fields of packet information as shown below Figure 3-7.
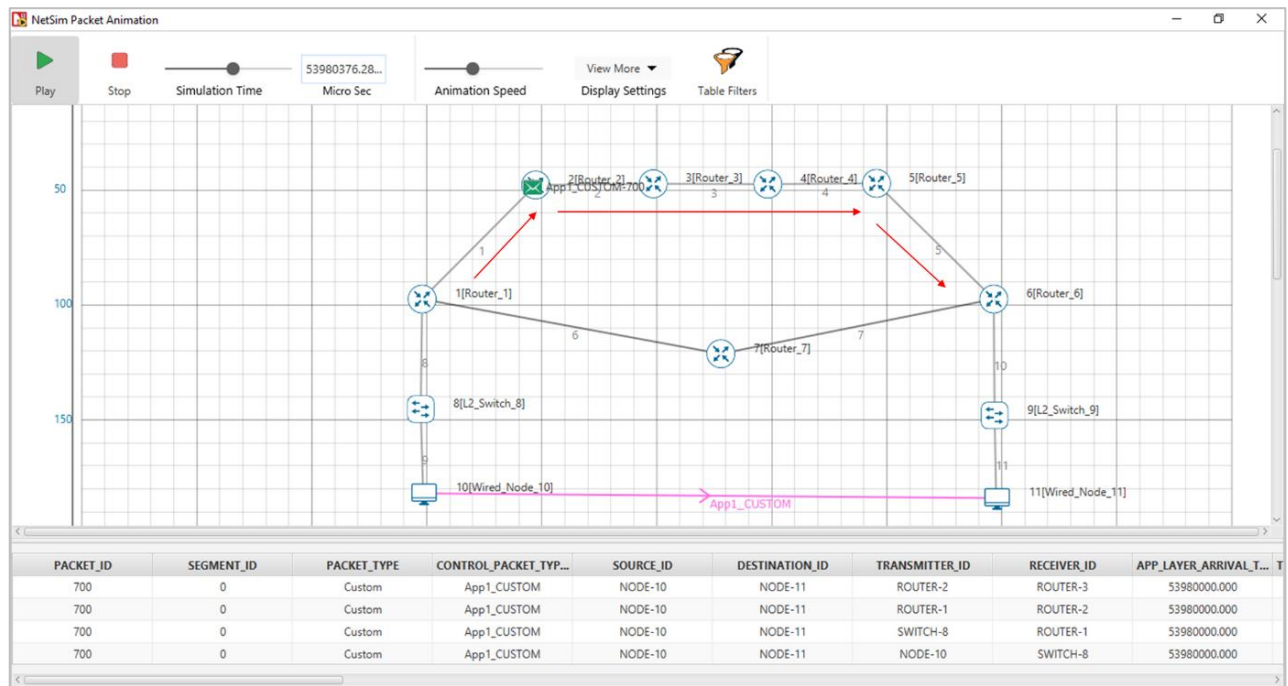
Figure 3-7: Animation window for OSPF

Users can view the same in Packet Trace.

Shortest Path from Wired Node 10 to Wired Node 11 in OSPF (Use Packet Animation to view)
Wired Node 10->L2 Switch 8->Router 1->Router 2->Router 3->Router 4->Router 5->Router 6->L2 Switch 9->Wired Node 11. OSPF chooses the above path (cost is less-5) since OSPF is based on cost.

3.1.6 Inference

3.1.6.1 RIP

In Distance vector routing, each router periodically shares its knowledge about the entire network with its neighbors. The three keys for understanding the algorithm,

1. **Knowledge About The Whole Network -** Router sends all of its collected knowledge about the network to its neighbors.
2. **Routing Only To Neighbors -** Each router periodically sends its knowledge about the network only to those routers to which it has direct links. It sends whatever knowledge it has about the whole network through all of its ports. This information is received and kept by each neighboring router and used to update it's own information about the network.
3. **Information Sharing At Regular Intervals -** For example, every 30 seconds, each router sends its information about the whole network to its neighbors. This sharing occurs whether or not the network has changed since the last time, information was exchanged

In NetSim the Routing Table Formation has 3 stages,

1. **Initial Table:** The Initial Table will show the direct connections made by each Router.

2. **Intermediate Table:** The Intermediate Table will have the updates of the Network in every 30 seconds.
3. **Final Table:** The Final Table is formed when there is no update in the Network.

The data should be forwarded using Routing Table with the shortest distance.

3.1.6.2 OSPF

The main operation of the OSPF protocol occurs in the following consecutive stages, and leads to the convergence of the internetworks:

1. Compiling the LSDB.
2. Calculating the Shortest Path First (SPF) Tree.
3. Creating the routing table entries.

Compiling the LSDB

The LSDB is a database of all OSPF router LSAs. The LSDB is compiled by an ongoing exchange of LSAs between neighboring routers so that each router is synchronized with its neighbor. When the Network converged, all routers have the appropriate entries in their LSDB.

Calculating the SPF Tree Using Dijkstra's Algorithm

Once the LSDB is compiled, each OSPF router performs a least cost path calculation called the Dijkstra algorithm on the information in the LSDB and creates a tree of shortest paths to each other router and network with themselves as the root. This tree is known as the SPF Tree and contains a single, least cost path to each router and in the Network. The least cost path calculation is performed by each router with itself as the root of the tree.

Calculating the Routing Table Entries from the SPF Tree

The OSPF routing table entries are created from the SPF tree and a single entry for each network in the AS is produced. The metric for the routing table entry is the OSPF-calculated cost, not a hop count.

If the application start time isn't changed then,

1. Packets generated before OSPF table convergence may be dropped at the gateway router.
2. The application may also stop if ICMP is enabled in the router.
3. If TCP is enabled TCP may stop after the re-try limit is reached (since the SYN packets would not reach the destination)

NOTE: The device / link numbering and IP Address setting in NetSim is based on order in which in the devices are dragged & dropped, and the order in which links are connected. Hence if the order in which a user executes these tasks is different from what is shown in the screen shots, users would notice different tables from what is shown in the screen shots.

3.2 Understand working of ARP and IP Forwarding within a LAN and across a router (Level 1)

3.2.1 Theory

In network architecture different layers have their own addressing scheme. This helps the different layers in being largely independent. Application layer uses host names, network layer uses IP addresses, and the link layer uses MAC addresses. Whenever a source node wants to send an IP datagram to a destination node, it needs to know the address of the destination. Since there are both IP addresses and MAC addresses, there needs to be a translation between them. This translation is handled by the Address Resolution Protocol (ARP). In IP network, IP routing involves the determination of suitable path for a network packet from a source to its destination. If the destination address is not on the local network, routers forward the packets to the next adjacent network.

(**Reference:** *A good reference for this topic is Section 5.4.1: Link Layer Addressing and ARP, of the book, Computer Networking, A Top-Down Approach, 6th Edition by Kurose and Ross)*

3.2.2 ARP protocol Description

1. ARP module in the sending host takes any IP address as input and returns the corresponding MAC address.
2. First the sender constructs a special packet called an ARP packet, which contains several fields including the sending and receiving IP and MAC addresses.
3. Both ARP request and response packets have the same format.
4. The purpose of the ARP request packet is to query all the other hosts and routers on the subnet to determine the MAC address corresponding to the IP address that is being resolved.
5. The sender broadcasts the ARP request packet, which is received by all the hosts in the subnet.
6. Each node checks if its IP address matches the destination IP address in the ARP packet.
7. The one with the match sends back to the querying host a response ARP packet with the desired mapping.
8. Each host and router have an ARP table in its memory, which contains mapping of IP addresses to MAC addresses.
9. The ARP table also contains a Time-to-live (TTL) value, which indicates when each mapping will be deleted from the table.

3.2.3 ARP Frame Format

Hardware Type		Protocol Type
Hardware Address Length	Protocol address length	Opcode
Sender Hardware Address		
Sender Protocol Address (1-2)		Sender Protocol Address (3-4)
Target hardware Address		
Target Protocol Address		

Figure 3-8: ARP Frame Format

The ARP message format is designed to accommodate layer two and layer three addresses of various sizes. This diagram shows the most common implementation, which uses 32 bits for the layer three ("Protocol") addresses, and 48 bits for the layer two hardware addresses.

3.2.4 IP Forwarding Description

1. Every router has a forwarding table that maps the destination addresses (or portions of the destination addresses) to that router's outbound links.
2. A router forwards a packet by examining the value of a field in the arriving packet's header, and then using this header value to index into the router's forwarding table.
3. The value stored in the forwarding table entry for that header indicates the router's outgoing link interface to which that packet is to be forwarded.
4. Depending on the network-layer protocol, the header value could be the destination address of the packet or an indication of the connection to which the packet belongs.
5. ARP operates when a host wants to send a datagram to another host on the same subnet.
6. When sending a Datagram off the subnet, the datagram must first be sent to the first-hop router on the path to the final destination. The MAC address of the router interface is acquired using ARP.
7. The router determines the interface on which the datagram is to be forwarded by consulting its forwarding table.
8. Router obtains the MAC address of the destination node using ARP.
9. The router sends the packet into the respective subnet from the interface that was identified using the forwarding table.

3.2.5 Network Set up

Open NetSim and click on **Experiments> Internetworks> Routing and Switching > Working of ARP and IP Forwarding within a LAN and across a router** then click on the tile in the middle panel to load the as shown in example see Figure 3-9.
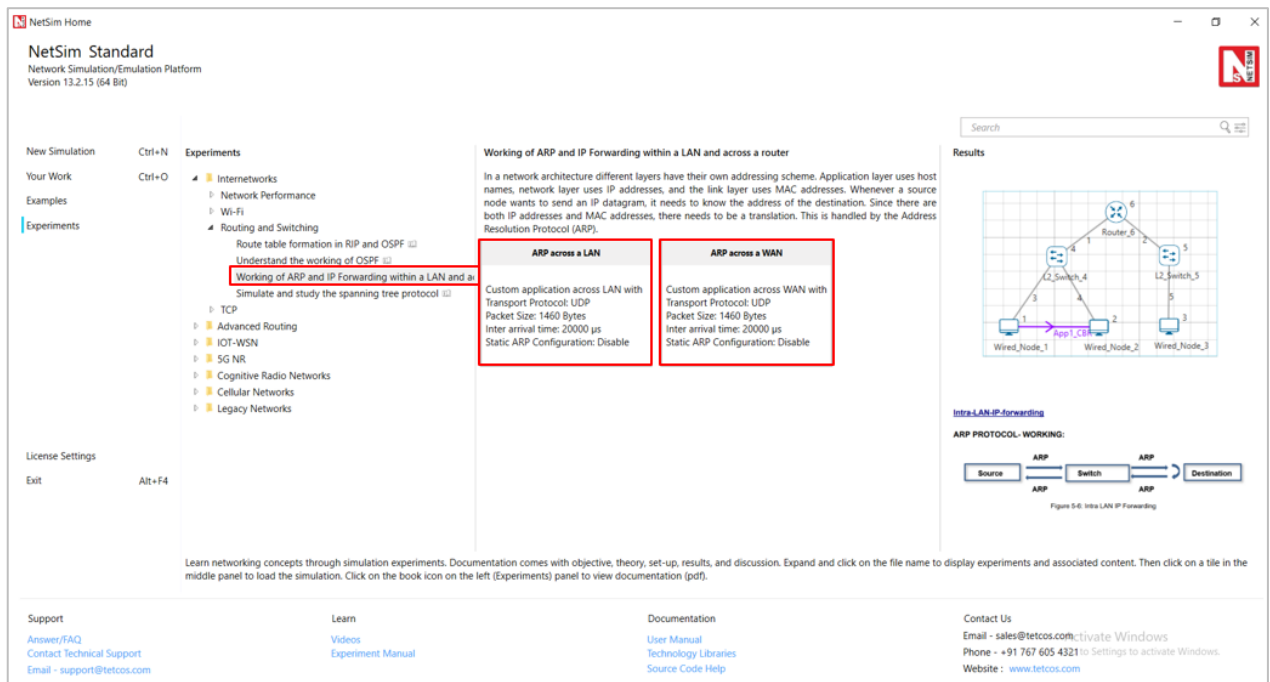
Figure 3-9: List of scenarios for the example of Working of ARP and IP Forwarding within a LAN and across a router

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-10.
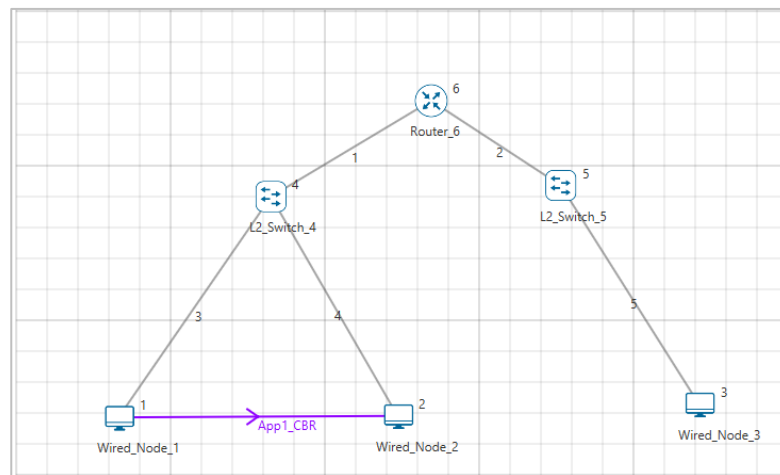
Figure 3-10: Network set up for studying the ARP across a LAN

3.2.6 Procedure

ARP across a LAN

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 3 Wired Nodes, 2 L2 Switches, and 1 Router in the **"Internetworks"** Network Library.

Step 2: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 1 i.e., Source to Wired Node 2 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000μs.

Transport Protocol is set to **UDP** instead of TCP. If set to TCP, the ARP table will get updated due to the transmission of TCP control packets thereby eliminating the need for ARP to resolve addresses.

Step 3: Packet Trace is enabled in the NetSim GUI, and hence we can view the ARP Request and ARP Reply packets exchanged initially, before transmission of the data packets.

Step 4: Enable the plots and click on Run simulation. The simulation time is set to 10 seconds. In the **"Static ARP Configuration"** tab, Static ARP is set to disable see Figure 3-11.
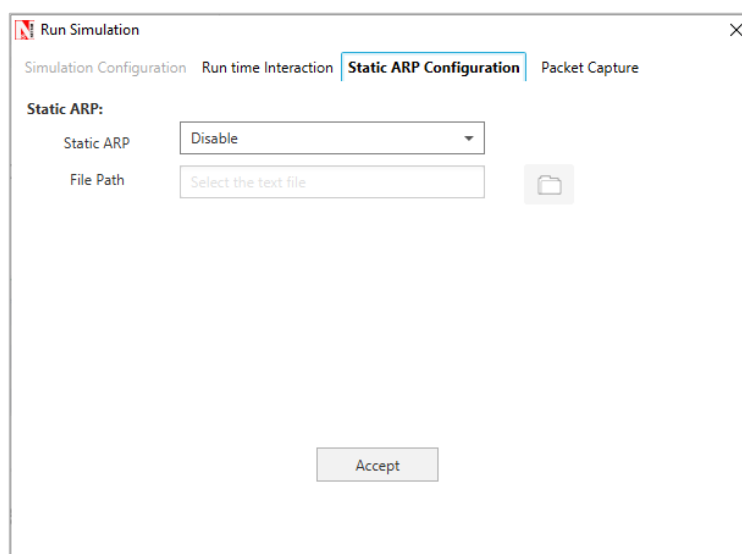
Figure 3-11: Static ARP Configuration Window

Click on Accept and then click on OK.

If Static ARP is enabled, then NetSim will automatically create an ARP table for each node. To see the working of the ARP protocol users should disable Static ARP.

By doing so, ARP request would be sent to the destination to find out the destinations MAC Address.

3.2.7 Output – ARP across a LAN

Once the simulation is complete, to view the packet trace file, click on **"Open Packet Trace"** option present in the left-hand-side of the Results Dashboard.

PACKET_ID ▼	SEGMENT_ID ▼	PACKET_TYPE ▼	CONTROL_PACKET_TYPE/APP_NAME ▼	SOURCE_ID ▼	DESTINATION_ID ▼	TRANSMITTER_ID ▼	RECEIVER_ID ▼
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	NODE-1	SWITCH-4
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	SWITCH-4	ROUTER-6
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	SWITCH-4	NODE-2
0	N/A	Control_Packet	ARP_Reply	NODE-2	NODE-1	NODE-2	SWITCH-4
0	N/A	Control_Packet	ARP_Reply	NODE-2	NODE-1	SWITCH-4	NODE-1

Figure 3-12: Open Packet Trace

NODE 1 will send ARP_REQUEST to SWITCH-4, SWITCH-4 sends this to ROUTER-6, and SWITCH-4 also sends this to NODE-2. ARP-REPLY is sent by the NODE-2 to SWITCH -4, and in-turn SWITCH-4 sends it to NODE-1.

3.2.8 Discussion – ARP across a LAN

Intra-LAN-IP-forwarding:

ARP PROTOCOL- WORKING:

Figure 3-13: Intra LAN IP Forwarding

NODE-1 broadcasts ARP_Request, which is then broadcasted by SWITCH-4. NODE-2 sends the ARP_Reply to NODE-1 via SWITCH-4. After this step, datagrams are transmitted from NODE-1 to NODE-2. Notice the DESTINATION_ID column for ARP_Request type packets, which indicates Broadcast-0.

ARP across a WAN

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-14.
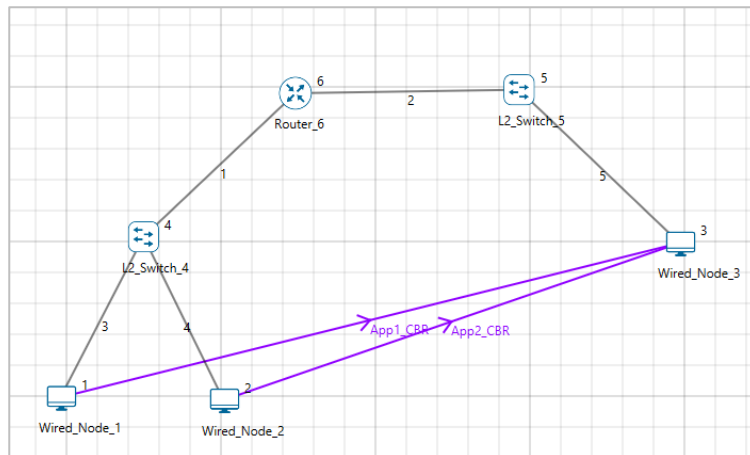
Figure 3-14: Network set up for studying the ARP across a WAN

3.2.9 Procedure

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in the NetSim GUI comprising of 3 Wired Nodes, 2 L2 Switches, and 1 Router.

Step 2: Right click on the Application tool bar and select Properties or click on the Application icon present in the top ribbon/toolbar.

APP 1 CBR is created from Wired Node 1 to Wired Node 3, Packet size set as 1460 bytes and Inter arrival time as 20000 Micro sec and Transport layer protocol to UDP.

APP 2 CBR is created from Wired Node 2 to Wired Node 3, Packet size set as 1460 bytes and Inter arrival time as 20000 Micro sec and Transport layer protocol to UDP. Additionally, the start time is set to 1 second and end time to 3 second.

Transport Protocol is set to UDP instead of TCP. If set to TCP, the ARP table will get updated during transmission of TCP control packets thereby eliminating the need for ARP to resolve addresses.

Step 3: Packet Trace is enabled in the NetSim GUI, and hence we can view the ARP Request and ARP Reply packets exchanged initially, before transmission of the data packets.

Step 4: Click on Run simulation. The simulation time is set to 10 seconds. In the **"Static ARP Configuration"** tab, Static ARP is set to disable.

3.2.10 Output I – ARP across a WAN

Once the simulation is complete, to view the packet trace file, click on **"Open Packet Trace"** option present in the left-hand-side of the Results Dashboard.
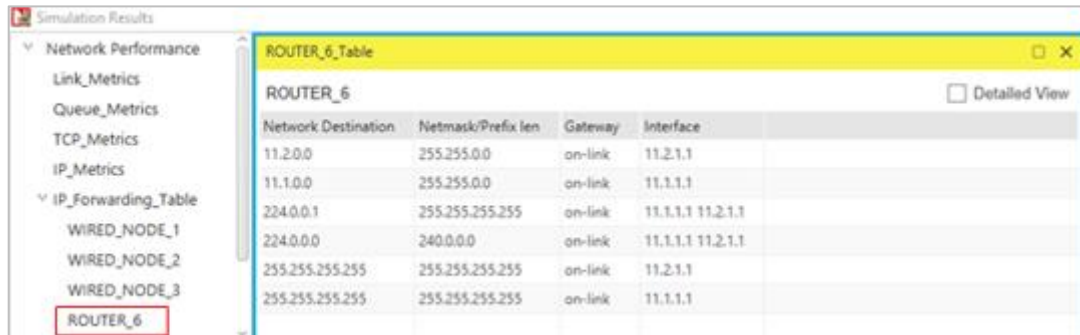
In packet trace, filter the CONTROL PACKET TYPE/ APP NAME filed to view APP 1CBR, ARP_REQUEST, ARP_REPLY.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	NODE-1	SWITCH-4
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	SWITCH-4	ROUTER-6
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	SWITCH-4	NODE-2
0	N/A	Control_Packet	ARP_Reply	ROUTER-6	NODE-1	ROUTER-6	SWITCH-4
0	N/A	Control_Packet	ARP_Reply	ROUTER-6	NODE-1	SWITCH-4	NODE-1
1	0	CBR	App1_CBR	NODE-1	NODE-3	NODE-1	SWITCH-4
1	0	CBR	App1_CBR	NODE-1	NODE-3	SWITCH-4	ROUTER-6
0	N/A	Control_Packet	ARP_Request	ROUTER-6	Broadcast-0	ROUTER-6	SWITCH-5
0	N/A	Control_Packet	ARP_Request	ROUTER-6	Broadcast-0	SWITCH-5	NODE-3
0	N/A	Control_Packet	ARP_Reply	NODE-3	ROUTER-6	NODE-3	SWITCH-5
0	N/A	Control_Packet	ARP_Reply	NODE-3	ROUTER-6	SWITCH-5	ROUTER-6
1	0	CBR	App1_CBR	NODE-1	NODE-3	ROUTER-6	SWITCH-5
1	0	CBR	App1_CBR	NODE-1	NODE-3	SWITCH-5	NODE-3
2	0	CBR	App1_CBR	NODE-1	NODE-3	NODE-1	SWITCH-4
2	0	CBR	App1_CBR	NODE-1	NODE-3	SWITCH-4	ROUTER-6
2	0	CBR	App1_CBR	NODE-1	NODE-3	ROUTER-6	SWITCH-5
2	0	CBR	App1_CBR	NODE-1	NODE-3	SWITCH-5	NODE-3
3	0	CBR	App1_CBR	NODE-1	NODE-3	NODE-1	SWITCH-4
3	0	CBR	App1_CBR	NODE-1	NODE-3	SWITCH-4	ROUTER-6
3	0	CBR	App1_CBR	NODE-1	NODE-3	ROUTER-6	SWITCH-5
3	0	CBR	App1_CBR	NODE-1	NODE-3	SWITCH-5	NODE-3
4	0	CBR	App1_CBR	NODE-1	NODE-3	NODE-1	SWITCH-4

Figure 3-15: Open Packet Trace

NODE 1 will send ARP_REQUEST to SWITCH-4, SWITCH-4 sends this to ROUTER-6, and SWITCH-4 also sends this to NODE-2. ARP-REPLY is sent by the ROUTER-6 to SWITCH -4, and in-turn SWITCH-4 sends it to NODE-1. Again ROUTER-6 will send ARP_REQUEST to SWITCH-5, SWITCH-5 sends this to NODE-3. ARP_REPLY is sent by NODE-3 to SWITCH-5 and in-turn SWITCH-5 sends it to ROUTER-6.

The IP forwarding table formed in the router can be accessed from the IP_Forwarding_Table list present in the Simulation Results window as shown below Figure 3-16.

Figure 3-16: IP Forwarding Table

Click on Detailed View checkbox to view the additional fields as indicated above.

Router forwards packets intended to the subnet 11.2.0.0 to the interface with the IP 11.2.1.1 based on the first entry in its routing table.

3.2.11 Discussion I – ARP across a WAN

From the above case we can understand that, since Router_6 did not know the destination address, the Application packets reach only till Router_6, and ARP mechanism continues with Router_6 re-broadcasting the ARP_REQUEST, finding the destination address and the datagram is getting transferred to Wired node 3 (destination).

3.2.12 Output II – ARP across a WAN

In same packet trace, filter the CONTROL PACKET TYPE/ APP NAME column to view APP 2 CBR, ARP_REQUEST, ARP_REPLY only.

In the below figure you can observer that ARP_REQUEST is broadcasted from Wired Node 2, the ARP Reply is sent from the Router 6, upon receiving the ARP_REPLY. Router 6 directly starts sending the data packet to the Wired Node 3 unlike the previous sample.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME(US)	TRX_LAYER_ARRIVAL_TIME(US)
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	NODE-1	SWITCH-4	N/A	N/A
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	SWITCH-4	ROUTER-6	N/A	N/A
0	N/A	Control_Packet	ARP_Request	NODE-1	Broadcast-0	SWITCH-4	NODE-2	N/A	N/A
0	N/A	Control_Packet	ARP_Reply	ROUTER-6	NODE-1	ROUTER-6	SWITCH-4	N/A	N/A
0	N/A	Control_Packet	ARP_Reply	ROUTER-6	NODE-1	SWITCH-4	NODE-1	N/A	N/A
0	N/A	Control_Packet	ARP_Request	ROUTER-6	Broadcast-0	ROUTER-6	SWITCH-5	N/A	N/A
0	N/A	Control_Packet	ARP_Request	ROUTER-6	Broadcast-0	SWITCH-5	NODE-3	N/A	N/A
0	N/A	Control_Packet	ARP_Reply	NODE-3	ROUTER-6	NODE-3	SWITCH-5	N/A	N/A
0	N/A	Control_Packet	ARP_Reply	NODE-3	ROUTER-6	SWITCH-5	ROUTER-6	N/A	N/A
0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	SWITCH-4	N/A	N/A
0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	SWITCH-4	ROUTER-6	N/A	N/A
0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	SWITCH-4	NODE-1	N/A	N/A
0	N/A	Control_Packet	ARP_Reply	ROUTER-6	NODE-2	ROUTER-6	SWITCH-4	N/A	N/A
0	N/A	Control_Packet	ARP_Reply	ROUTER-6	NODE-2	SWITCH-4	NODE-2	N/A	N/A
1	0	CBR	App2_CBR	NODE-2	NODE-3	NODE-2	SWITCH-4	1000000	1000000
1	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-4	ROUTER-6	1000000	1000000
1	0	CBR	App2_CBR	NODE-2	NODE-3	ROUTER-6	SWITCH-5	1000000	1000000
1	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-5	NODE-3	1000000	1000000
2	0	CBR	App2_CBR	NODE-2	NODE-3	NODE-2	SWITCH-4	1020000	1020000
2	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-4	ROUTER-6	1020000	1020000
2	0	CBR	App2_CBR	NODE-2	NODE-3	ROUTER-6	SWITCH-5	1020000	1020000
2	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-5	NODE-3	1020000	1020000
3	0	CBR	App2_CBR	NODE-2	NODE-3	NODE-2	SWITCH-4	1040000	1040000
3	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-4	ROUTER-6	1040000	1040000
3	0	CBR	App2_CBR	NODE-2	NODE-3	ROUTER-6	SWITCH-5	1040000	1040000
3	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-5	NODE-3	1040000	1040000
4	0	CBR	App2_CBR	NODE-2	NODE-3	NODE-2	SWITCH-4	1060000	1060000
4	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-4	ROUTER-6	1060000	1060000
4	0	CBR	App2_CBR	NODE-2	NODE-3	ROUTER-6	SWITCH-5	1060000	1060000
4	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-5	NODE-3	1060000	1060000
5	0	CBR	App2_CBR	NODE-2	NODE-3	NODE-2	SWITCH-4	1080000	1080000
5	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-4	ROUTER-6	1080000	1080000
5	0	CBR	App2_CBR	NODE-2	NODE-3	SWITCH-5		1080000	1080000

Figure 3-17: Open Packet Trace

3.2.13 Discussion II – ARP across a WAN

Across-Router-IP-forwarding

ARP PROTOCOL- WORKING
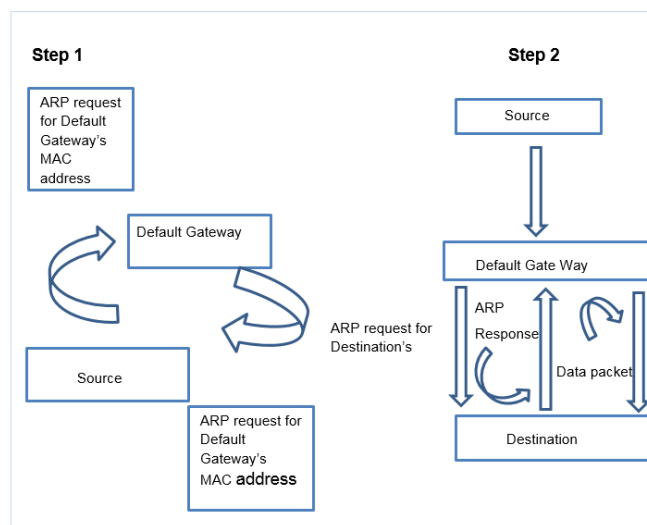
Figure 3-18: Across Router IP Forwarding

NODE-2 transmits ARP_Request which is further broadcasted by SWITCH-4. ROUTER-6 sends ARP_Reply to NODE-2 which goes through SWITCH-4. Then NODE-2 starts sending datagrams to NODE-3. If router has the MAC address of NODE-3 in its ARP table, then ARP ends here, and router starts forwarding the datagrams to NODE-3 by consulting its forwarding table. Router 6, has this information updated during transmission of APP1 packets and hence ARP request for identifying the MAC address of NODE-3, need not be sent again. In the other case (Output -I), Router sends ARP_Request to appropriate subnet and after getting the MAC address of NODE-3, it then forwards the datagrams to NODE-3 using its forwarding table.

3.3 Simulate and study the spanning tree protocol (Level 1)

3.3.1 Introduction

Spanning Tree Protocol (STP) is a link management protocol. Using the spanning tree algorithm, STP provides path redundancy while preventing undesirable loops in a network that are created by multiple active paths between stations. Loops occur when there are alternate routes between hosts. To establish path redundancy, STP creates a tree that spans all of the switches in an extended network, forcing redundant paths into a standby, or blocked state. STP allows only one active path at a time between any two network devices (this prevents the loops) but establishes the redundant links as a backup if the initial link should fail. Without spanning tree in place, it is possible that both connections may simultaneously live, which could result in an endless loop of traffic on the LAN.

(**Reference:** *A good reference for this topic is Section 3.1.4: Bridges and LAN switches, of the book, Computer Networks, 5th Edition by Peterson and Davie*)

3.3.2 Network Setup

Open NetSim and click on **Experiments> Internetworks> Routing and Switching> Simulate and study the spanning tree protocol** then click on the tile in the middle panel to load the example as shown in below Figure 3-19.
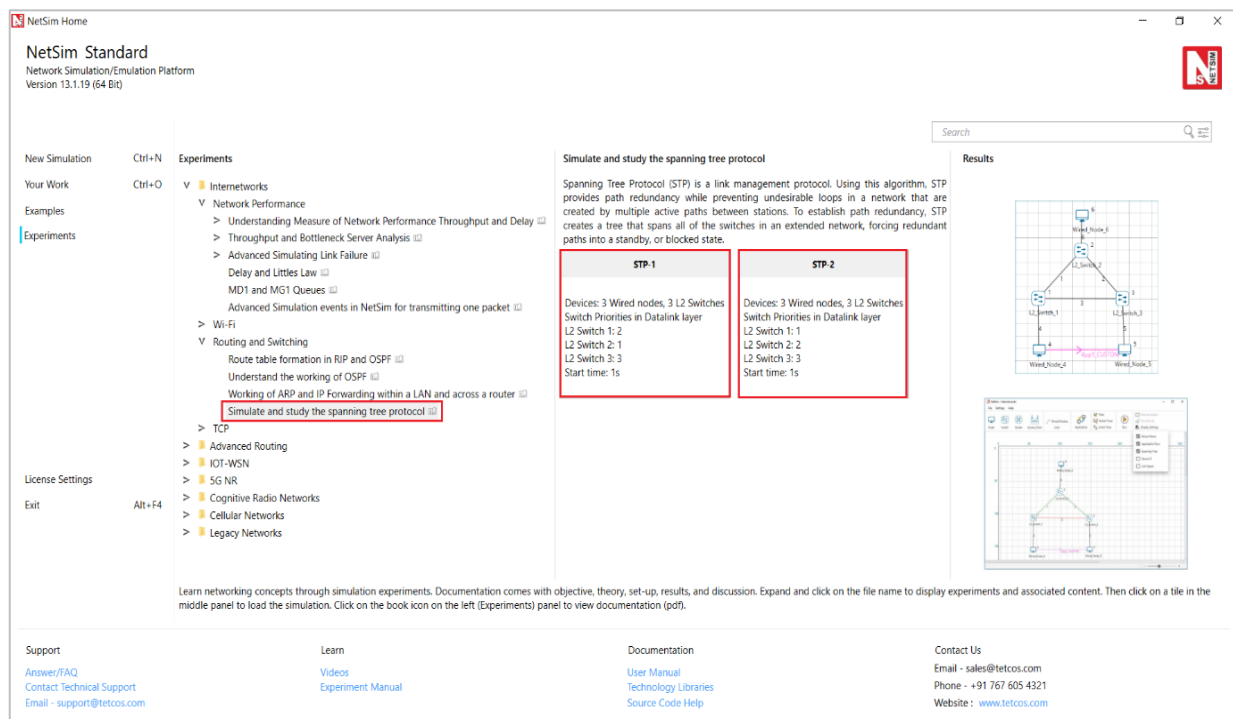
Figure 3-19: List of scenarios for the example of Simulate and study the spanning tree protocol

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-20.
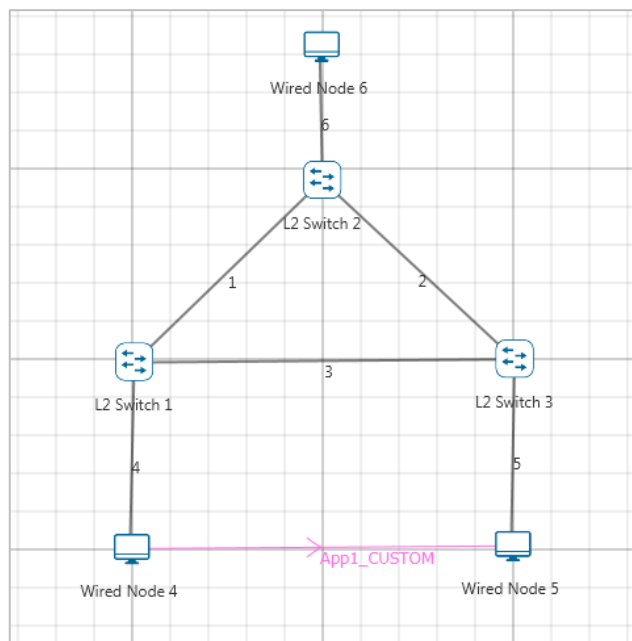
Figure 3-20: Network set up for studying the STP 1

Note: At least three L2 Switches are required in the network to analyze the spanning tree formation.

3.3.3 Procedure

STP-1

Step 1: A network scenario is designed in the NetSim GUI comprising of 3 Wired Nodes and 3 L2 Switches in the **"Internetworks"** Network Library.

Step 2: Go to L2 Switch 1 Properties. In the Interface 1 (ETHERNET) > Datalink Layer, **"Switch Priority"** is set to 2. Similarly, for the other interfaces of L2 Switch 1, Switch Priority is set to 2.

Step 3: Go to L2 Switch 2 Properties. In the Interface 1 (ETHERNET) > Datalink Layer, **"Switch Priority"** is set to 1. Similarly, for the other interfaces of L2 Switch 2, Switch Priority is set to 1.

Step 4: Go to L2 Switch 3 Properties. In the Interface 1 (ETHERNET) > Datalink Layer, **"Switch Priority"** is set to 3. Similarly, for the other interfaces of L2 Switch 3, Switch Priority is set to 3.

L2_Switch Properties	L2_Switch 1	L2_Switch 2	L2_Switch 3
Switch Priority	2	1	3

Table 3-1: Switch Priorities for STP-1

NOTE: Switch Priority is set to all the 3 L2 Switches and Switch Priority has to be changed for all the interfaces of L2 Switch.

Switch Priority is interpreted as the weights associated with each interface of a L2 Switch. A higher value indicates a higher priority.

Step 5: Right click on the Application Flow **"App1 CUSTOM"** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wired Node 4 i.e., Source to Wired Node 5 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000µs. Additionally, the **"Start Time"** parameter is set to 1 second while configuring the application see Figure 3-21.

Figure 3-21: Application Configuring Window

Note: Wired Node 6 is not generating traffic to any other nodes.

Here, Wired Node 4 is sending data to Wired Node 5 and the node properties are set to default.

Step 6: Enable the plots and click on Run simulation. The simulation time is set to 10 seconds

STP-2

The following changes in settings are done from the previous Sample:

In **STP 2**, the **"Switch Priority"** of all the 3 L2 Switches are changed as follows Table 3-2:

L2_Switch Properties	L2_Switch 1	L2_Switch 2	L2_Switch 3
Switch Priority	1	2	3

Table 3-2: Switch Priorities for STP 2

3.3.4 Output

In the NetSim Design Window, click on **Display Settings > Spanning Tree** check box see Figure 3-22.

STP-1
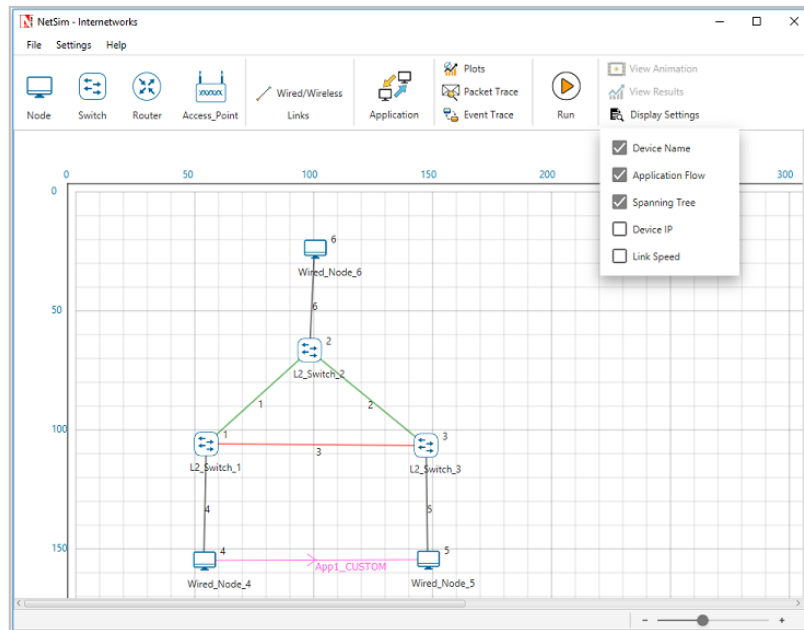
Figure 3-22: NetSim Design Window - Display Setting for STP-1

Go to NetSim Packet Animation Window and click on **Play** button. We can notice that, after the exchange of control packets, the data packets take the following path. **Wired Node 4 > L2 Switch 1 > L2 Switch 2 > L2 Switch 3 > Wired Node 5.**

STP-2
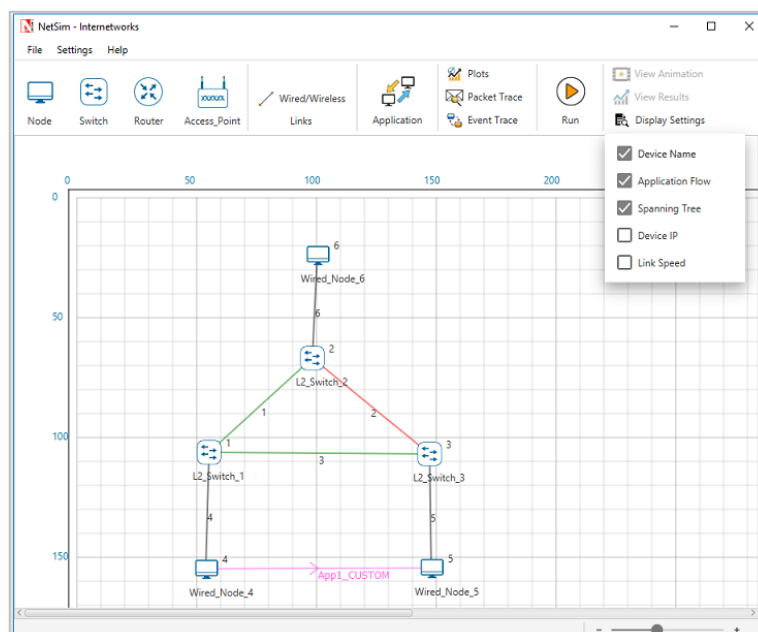
Figure 3-23: NetSim Design Window - Display Setting for STP-2

Go to NetSim Packet Animation window and click on **Play** button. We can notice that, after the exchange of control packets, the data packets take the following path. **Wired Node 4 > L2 Switch 1 > L2 Switch 3 > Wired Node 5.**

Go to Simulation Results window, In the left-hand-side of the Results Dashboard, click on the arrow pointer of **Switch MAC address table** to obtain the Switch MAC address table list of all the L2 Switches.

For each L2 Switch, a Switch MAC Address Table containing the MAC address entries see **Figure 3-24**, the port that is used for reaching it, along with the type of entry can be obtained at the end of Simulation.

Figure 3-24: STP 2 MAC Address table

3.3.5 Inference

Each L2 Switch has an ID which is a combination of its Lowest MAC address and priority. The Spanning tree algorithm selects the L2 Switch with the smallest ID as the root node of the Spanning Tree. The root node forward frames out over all its ports. In the other L2 Switches, the ports that have the least cost of reaching the root switch are set as **Forward Ports** and the remaining are set as **Blocked Ports**. In the STP-1, L2_Switch 2 was assigned least priority and was selected as a Root Switch. The green line indicates the forward path, and the red line indicates the blocked path. The frame from Wired Node 4 should take the path through the L2_Switch 1, 2 and 3 to reach the Wired Node 5. In the STP-2, L2_Switch 1 was assigned least priority and selected as a Root switch. In this case, the frame from Wired Node 4 takes the path through the L2_Switch 1 and 3 to reach the destination Wired Node 5.

3.4 Understanding VLAN operation in L2 and L3 Switches (Level 2)

3.4.1 Introduction to VLAN

VLAN is called as virtual local area network, used in Switches and it operates at Layer 2 and Layer 3. A VLAN is a group of hosts which communicate as if they were attached to the same broadcast domain, regardless of their physical Location.

For example, all workstations and servers used by a particular workgroup team can be connected to the same VLAN, regardless of their physical connections to the network or the fact that they might be intermingled with other teams. VLANs have the same attributes as physical LANs, but you can group end stations even if they are not physically located on the same LAN Segment.
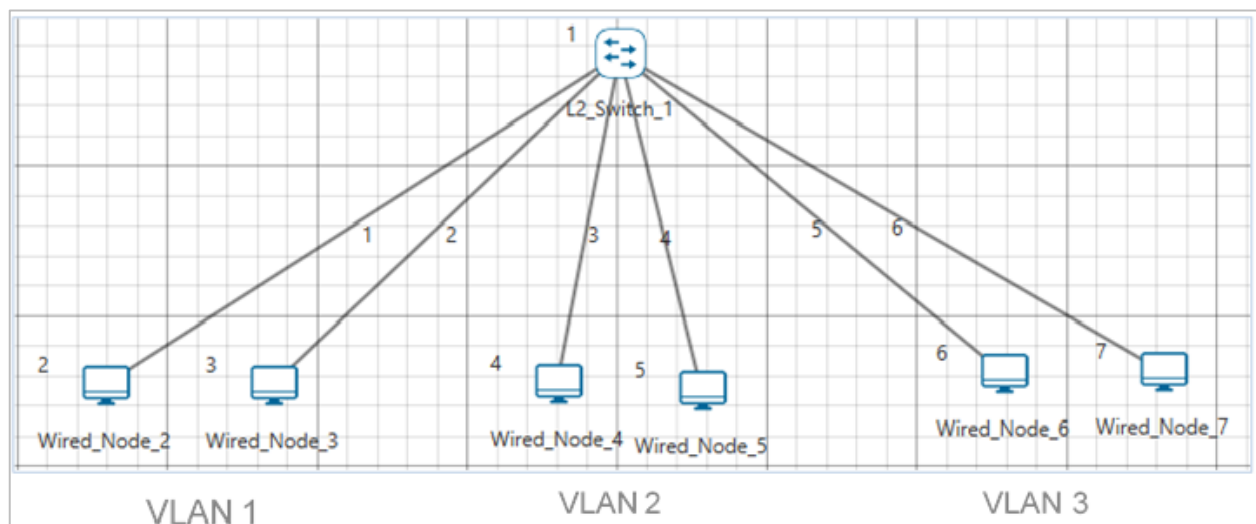
Figure 3-25: Virtual local area network (VLAN)

A VLAN behaves just like a LAN in all respects but with additional flexibility. By using VLAN technology, it is possible to subdivide a single physical switch into several logical switches. VLANs are implemented by using the appropriate switch configuration commands to create the VLANs and assign specific switch interfaces to the desired VLAN.

Switches implement VLANs by adding a VLAN tag to the Ethernet frames as they enter the switch. The VLAN tag contains the VLAN ID and other information, which is determined by the interface from which the frame enters the switch. The switch uses VLAN tags to ensure that each Ethernet frame is confined to the VLAN to which it belongs based on the VLAN ID contained in the VLAN tag. The VLAN tags are removed as the frames exit the switch on the way to their destination.

Any port can belong to a VLAN, and unicast, broadcast, and multicast packets are forwarded and flooded only to end stations in that VLAN. Each VLAN is considered a logical network.

Packets destined for stations that do not belong to the VLAN must be forwarded through a router.

In the below screenshot, the stations in the development department are assigned to one VLAN, the stations in the marketing department are assigned to another VLAN, and the stations in the testing department are assigned to another VLAN.
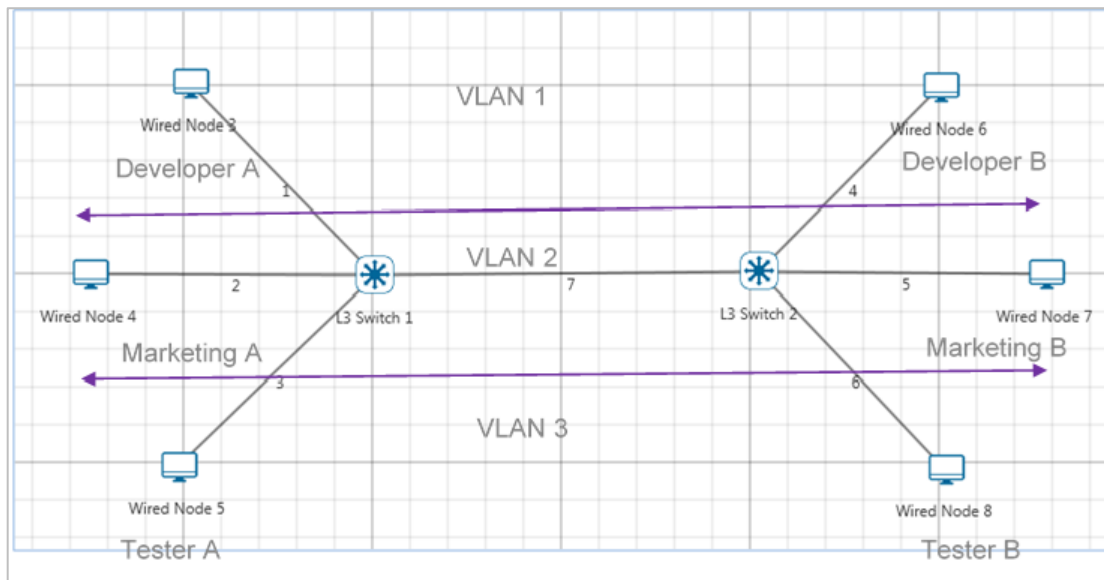
Figure 3-26: Hosts in one VLAN need to communicate with hosts in another VLAN This is known as Inter-VLAN routing.

VLANs divide broadcast domains in a LAN environment. Whenever hosts in one VLAN need to communicate with hosts in another VLAN, the traffic must be routed between them. This is known as Inter-VLAN routing. This can be possible by using L3 Switch.

What is a layer 3 switch?

Layer 3 switch (also known as a multi-layer switch) is a multi-functional device that have the same functionality like a layer 2 switch, but behaves like a router when necessary. It's generally faster than a router due to its hardware-based routing functions, but it's also more expensive than a normal switch.

3.4.2 Network setup

Open NetSim and click on **Experiments > Advanced Routing> Understanding VLAN operation in L2 and L3 Switches** then click on the tile in the middle panel to load the example as shown in below in Figure 3-27.
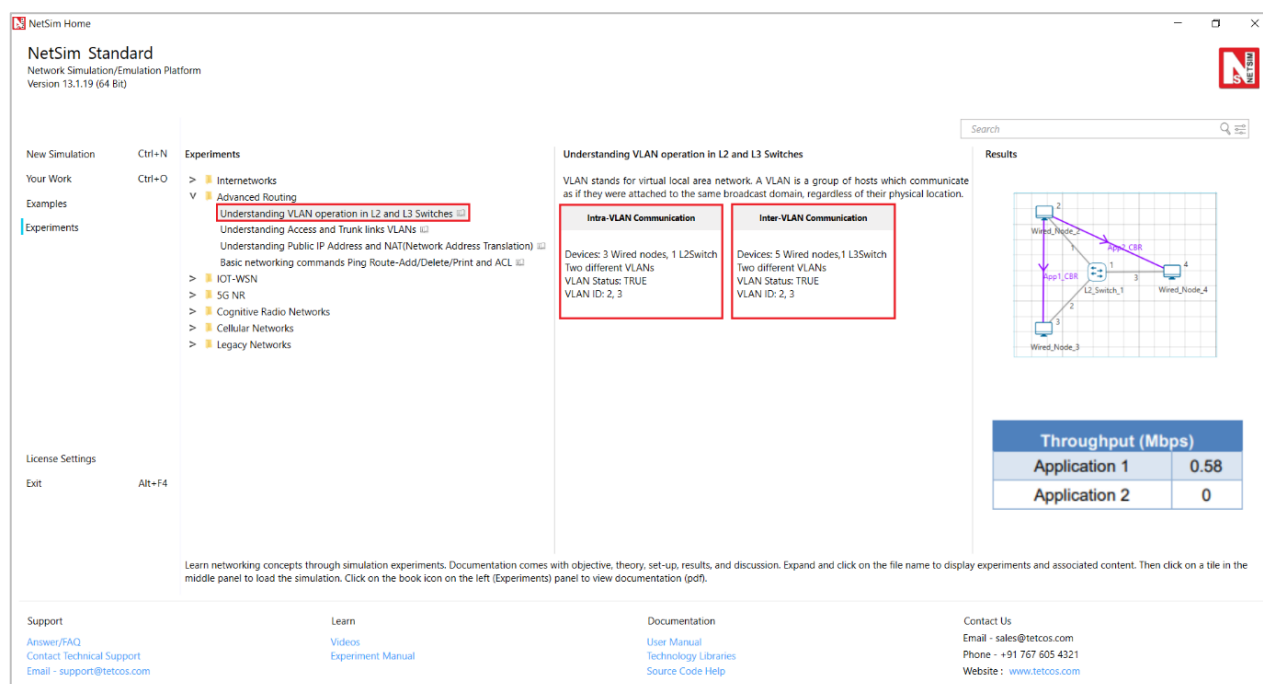
Figure 3-27: List of scenarios for the example of Understanding VLAN operation in L2 and L3 Switches

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-28.
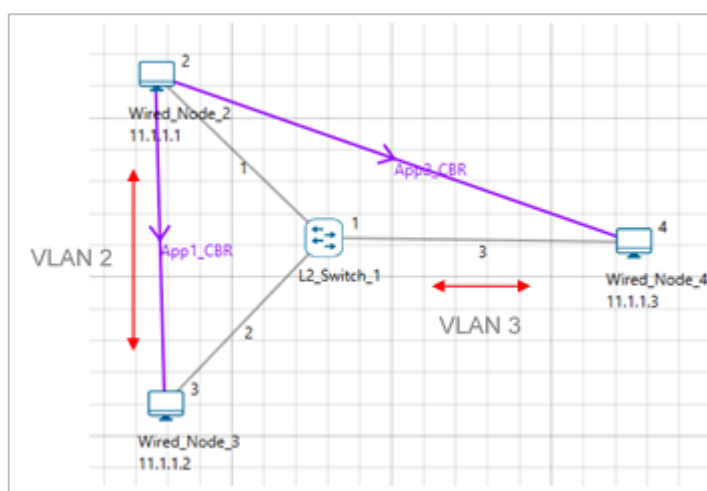
Figure 3-28: Network set up for studying the Intra-VLAN

3.4.3 Procedure

Intra-VLAN

Intra-VLAN is a mechanism in which hosts in same VLAN can communicate to each other.

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 3 Wired Nodes and 1 L2 Switch in the **"Internetworks"** Network Library.

Step 2: L2 Switch 1 Properties are configured as follows Table 3-3.

L2 Switch 1

Interface ID	VLAN Status	VLAN ID	VLAN Port Type
Interface_1	TRUE	2	Access _Port
Interface_2	TRUE	2	Access _Port
Interface_3	TRUE	3	Access _Port

Table 3-3: L2 Switch 1 Properties

In all the INTERFACE (ETHERNET) > DATALINK LAYER Properties of L2 Switch 1, **"VLAN Status"** is set to TRUE.

Figure 3-29: DATALINK LAYER Properties of L2 Switch 1

Now click on **"Configure VLAN"** option and the VLAN 2 fields are entered as shown below Figure 3-30.

Figure 3-30: VLAN Configure window

To add a new entry after entering the required fields, click on the ADD button.

Figure 3-31: Configuring VLAN Properties in VLAN 2

To configure another VLAN, click on the **"+"** symbol located in the top.

Figure 3-32: Configuring VLAN Properties inS VLAN 3

And then we can add the entry to it.

Step 3: Enable the plots and run simulation for 10 Seconds and observe the throughputs.

Inter-VLAN

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-33.
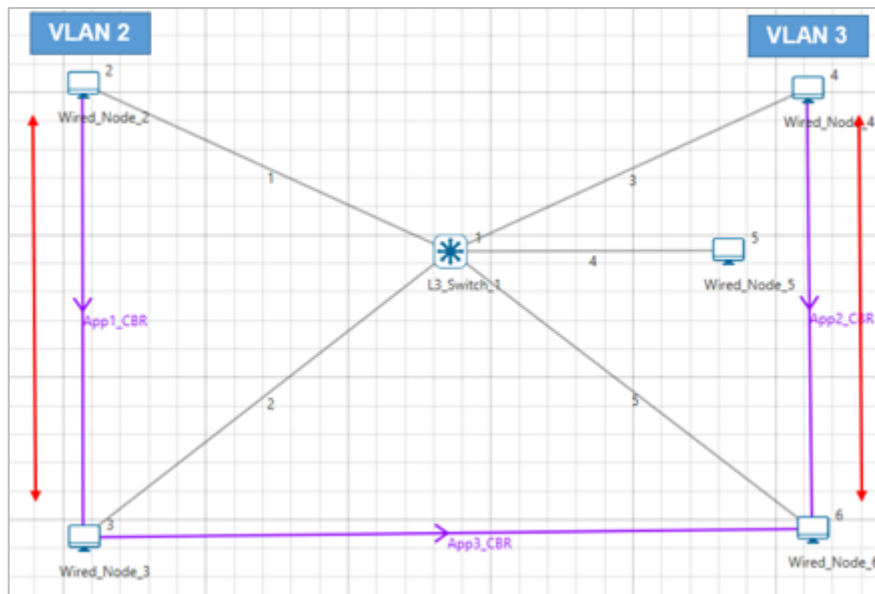
Figure 3-33: Network set up for studying the Inter-VLAN

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 5 Wired Nodes and 1 L3 Switch in the **"Internetworks"** Network Library.

Step 2: The Wired Node properties are set as per the below Table 3-4.

Node	Wired Node2 I/f1_Ethernet	Wired Node3 I/f1_Ethernet	Wired Node4 I/f1_Ethernet	Wired Node5 I/f1_Ethernet	Wired Node6 I/f1_Ethernet
IP Address	10.0.0.4	10.1.0.4	11.2.0.4	11.3.0.4	11.4.0.4
Default Gateway	10.0.0.3	10.1.0.3	11.2.0.3	11.3.0.3	11.4.0.3

Table 3-4: Wired Node properties

Step 3: The L3 Switch 1 Properties are set as per the below table:

L3 Switch	I/f1_Ethernet IP Address	I/f2_Ethernet IP Address	I/f3_Ethernet IP Address	I/f4_Ethernet IP Address	I/f5_Ethernet IP Address
L3 Switch 1	10.0.0.3	10.1.0.3	11.2.0.3	11.3.0.3	11.4.0.3

Table 3-5: L3 Switch 1 Properties

L3 Switch 1			
Interface ID	VLAN Status	VLAN ID	VLAN Port Type
Interface_1	TRUE	2	Access _Port
Interface_2	TRUE	2	Access _Port
Interface_3	TRUE	3	Access _Port

Interface_4	TRUE	3	Access _Port
Interface_5	TRUE	3	Access _Port

Table 3-6: VLAN configurations Properties

The VLAN configurations done are shown as follows:

Figure 3-34: Configuring VLAN Properties in VLAN 2

Figure 3-35: Configuring VLAN Properties in VLAN 3

Step 4: Plots are enabled in NetSim GUI. Run simulation for 10 seconds and observe the throughputs.

3.4.4 Output and Inference for Intra-VLAN

Throughput (Mbps)	
Application 1	0.58
Application 2	0

Table 3-7: Results Comparison

The throughput for 2nd application is zero because the source and destination is in different VLANs, thereby traffic flow or communication between 2 VLANs using Layer 2 switch is not possible. To overcome this problem, an L3 switch is used.

3.4.5 Output and Inference for Inter-VLAN

Throughput (Mbps)	
Application 1	0.58
Application 2	0.58
Application 3	0.58

Table 3-8: Results Comparison

In this case, application1 is in VLAN2, application2 is in VLAN3 and application 3 is in between VLAN2 and VLAN3. From the above results, the throughput for application 3 (different VLANs) is nonzero, because of using L3 switch. So, communication between 2 VLANs is possible using L3 Switch.

3.5 Understanding Access and Trunk Links in VLANs (Level 2)

3.5.1 Theory

An access link is a link that is part of only one VLAN, and normally access links are for end devices. An access-link connection can understand only standard Ethernet frames. Switches remove any VLAN information from the frame before it is sent to an access-link device.

A Trunk link can carry multiple VLAN traffic and normally a trunk link is used to connect switches to other switches or to routers. A trunk link is not assigned to a specific VLAN. Multiple VLAN traffic can be transported between switches using a single physical trunk link.
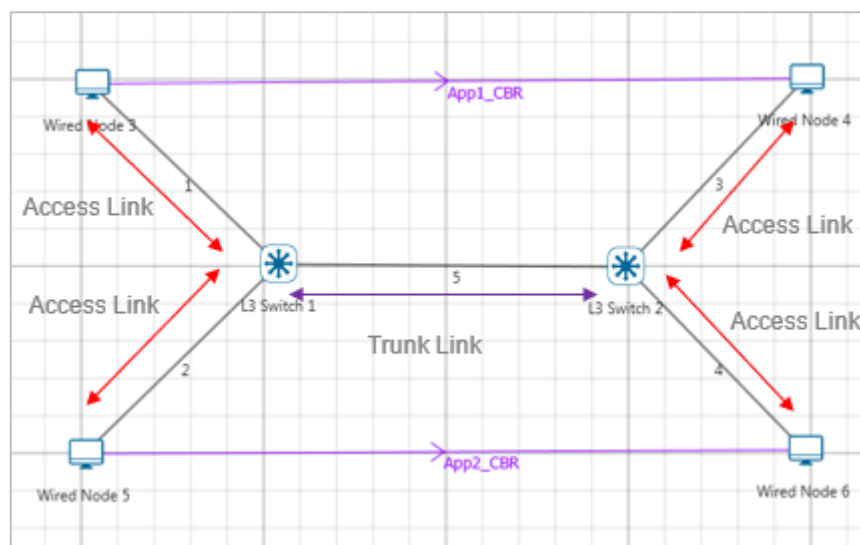
Figure 3-36: Understanding Access and Trunk Links in VLANs

Access link

Access link connection is the connection where switch port is connected with a device that has a standardized Ethernet NIC. Standard NIC only understand IEEE 802.3 or Ethernet II frames. Access link connection can only be assigned with single VLAN. That means all devices connected to this port will be in same broadcast domain.

For example, twenty users are connected to a hub, and we connect that hub with an access link port on switch, then all of these users belong to same VLAN. If we want to keep ten users in another VLAN, then we need to plug in those ten users to another hub and then connect it with another access link port on switch.

Trunk link

Trunk link connection is the connection where switch port relates to a device that is capable to understand multiple VLANs. Usually, trunk link connection is used to connect two switches. A VLAN can span anywhere in network, and that can happen due to trunk link connection.

Trunking allows us to send or receive VLAN information across the network. To support trunking, original Ethernet frame is modified to carry VLAN information.

3.5.2 Network Setup

Open NetSim and click on **Experiments> Advanced Routing> Understanding Access and Trunk Links in** VLANs then click on the tile in the middle panel to load the example as shown in below in Figure 3-37.
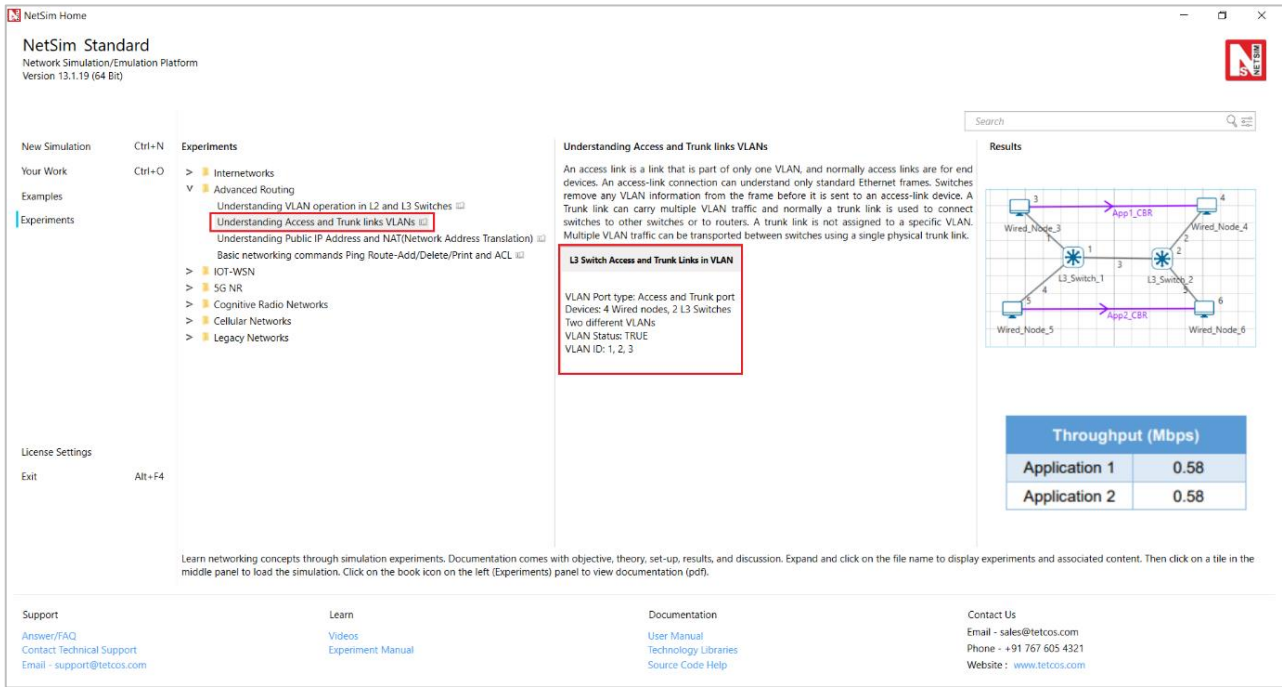
Figure 3-37: List of scenarios for the example of Understanding Access and Trunk Links in VLANs

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-38.
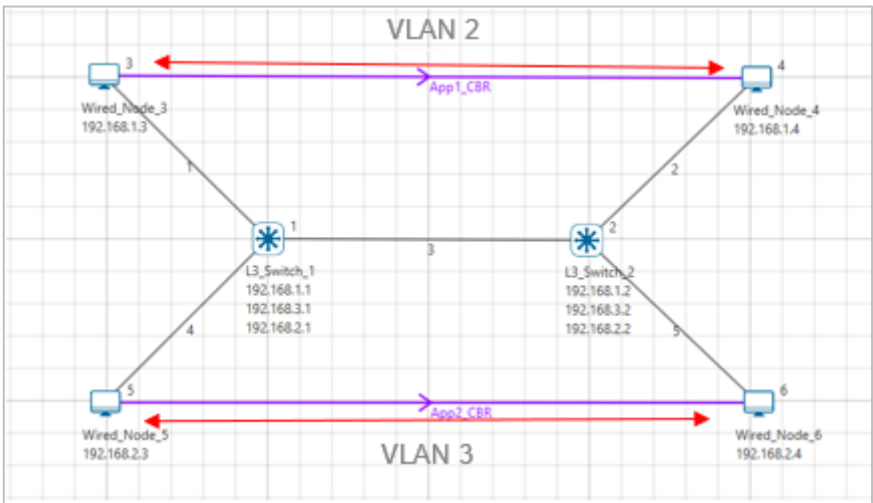
Figure 3-38: Network set up for studying the L3 Switch Access and Trunk Links in VLANs

3.5.3 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 4 Wired Nodes and 2 L3 Switches in the **"Internetworks"** Network Library.

Step 2: In the INTERFACE (ETHERNET) > NETWORK LAYER Properties, set the following **Table** 3-9.

Node	Wired Node 3	Wired Node 4	Wired Node 5	Wired Node 6
	I/f1_Ethernet	I/f1_Ethernet	I/f1_Ethernet	I/f1_Ethernet
IP Address	192.168.1.3	192.168.1.4	192.168.2.3	192.168.2.4
Default Gateway	192.168.1.1	192.168.1.2	192.168.2.1	192.168.2.2
Subnet Mask	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

Table 3-9: Network Layer Properties

NOTE: The subnet mask of all L3 Switch interfaces is set to 255.255.255.0

Step 3: L3 Switch 1 and L3 Switch 2 properties are set as follows:

Switch	I/f1_Ethernet	I/f2_Ethernet	I/f3_Ethernet
	IP Address	IP Address	IP Address
L3 Switch 1	192.168.1.1	192.168.3.1	192.168.2.1
L3 Switch 2	192.168.1.2	192.168.3.2	192.168.2.2

Table 3-10: L3 Switch 1 and L3 Switch 2 properties

Figure 3-39: Datalink layer properties window

Step 4: In the INTERFACE (ETHERNET) > DATALINK LAYER Properties of L3 Switch 1, Click on **"Configure VLAN"** to view the properties for VLAN 2 set as per the screenshot shown below Figure 3-40.

Figure 3-40: Configuring VLAN Properties in VLAN 2

Properties for VLAN 3 is set as per the below screenshot Figure 3-41.

Figure 3-41: Configuring VLAN Properties in VLAN 3

After setting the properties of VLAN2 and VLAN3 click on OK.

Step 5: In the NETWORK LAYER Properties of L3 Switch 1, Enable - Static IP Route ->Click on **"Configure Static Route IP"** to set static route as per the screenshot shown below.
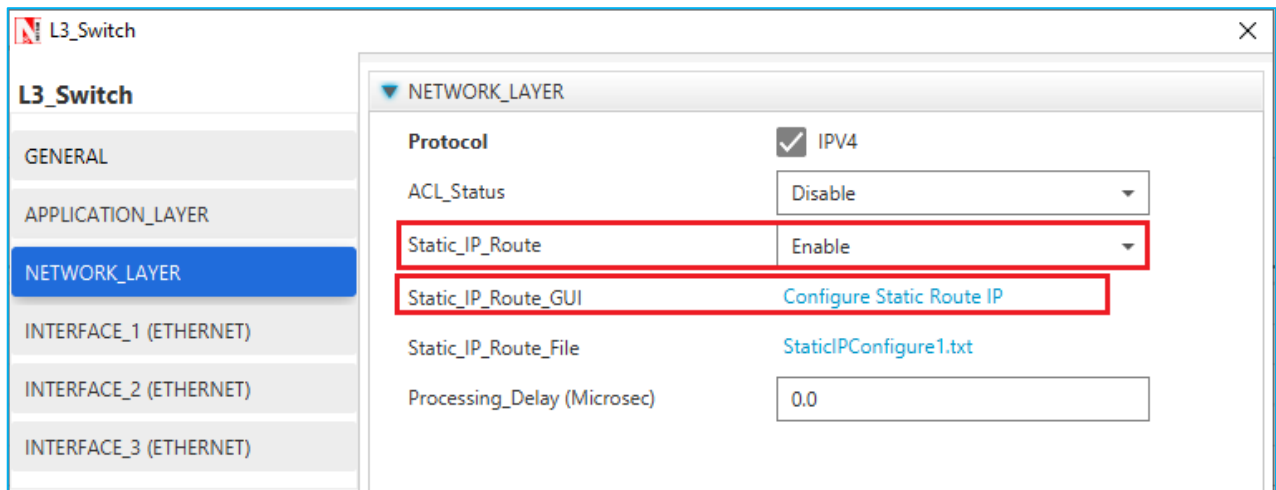
Figure 3-42: Select Configure Static Route IP

Set the properties in Static Route IP window as per the screenshot below and click on **Add**.

Click on **OK**
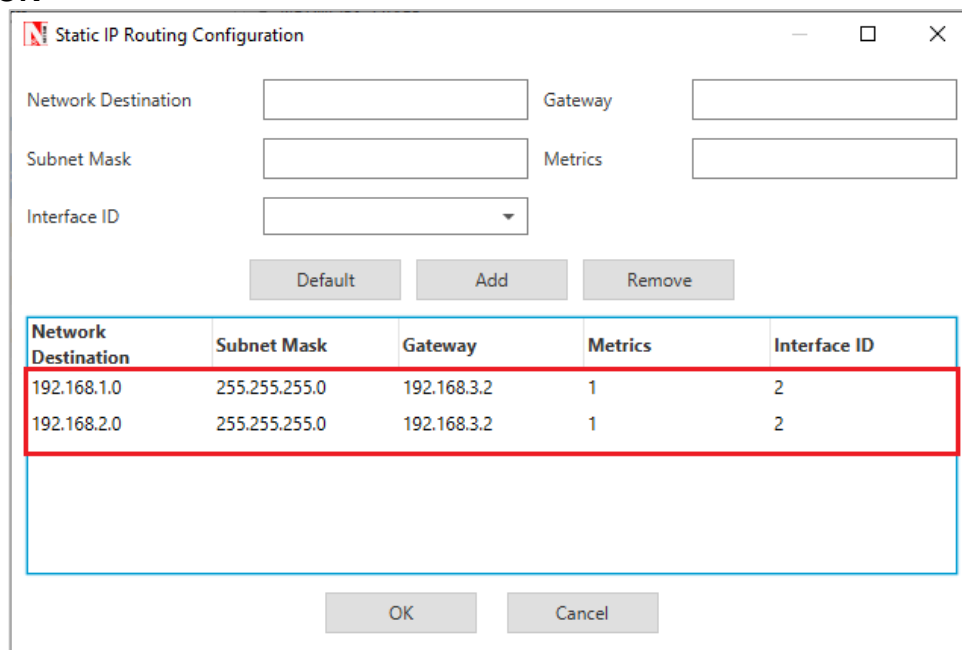
Figure 3-43: Configure Static route in Static Route IP window

NOTE: *Transport Protocol is set to* **UDP** *in Application properties.*

Step 6: Enable the plots and run simulation for 10 seconds and observe the throughput.

3.5.4 Output

Throughput (Mbps)	
Application 1	0.58
Application 2	0.58

Table 3-11: Results Comparison

The above results conclude that trunking allows us to send or receive any VLAN information across the network.

3.6 Understanding Public IP Address & NAT (Network Address Translation) (Level 2)

3.6.1 Theory

3.6.1.1 Public Address

A public IP address is assigned to every computer that connects to the Internet where each IP is unique. Hence there cannot exist two computers with the same public IP address all over the Internet. This addressing scheme makes it possible for the computers to "find each other" online and exchange information. User has no control over the IP address (public) that is assigned to the computer. The public IP address is assigned to the computer by the Internet Service Provider as soon as the computer is connected to the Internet gateway.

3.6.1.2 Private Address

An IP address is considered private if the IP number falls within one of the IP address ranges reserved for private networks such as a Local Area Network (LAN). The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private networks (local networks):

Class	Starting IP address	Ending IP address	No. of hosts
A	10.0.0.0	10.255.255.255	16,777,216
B	172.16.0.0	172.31.255.255	1,048,576
C	192.168.0.0	192.168.255.255	65,536

Table 3-12: Private IP address table

Private IP addresses are used for numbering the computers in a private network including home, school and business LANs in airports and hotels which makes it possible for the computers in the network to communicate with each other. For example, if a network A consists of 30 computers each of them can be given an IP starting from **192.168.0.1 to 192.168.0.30.**

Devices with private IP addresses cannot connect directly to the Internet. Likewise, computers outside the local network cannot connect directly to a device with a private IP. It is possible to interconnect two private networks with the help of a router or a similar device that supports Network Address Translation.

If the private network is connected to the Internet (through an Internet connection via ISP) then each computer will have a private IP as well as a public IP. Private IP is used for communication within the network whereas the public IP is used for communication over the Internet.

3.6.1.3 Network address translation (NAT)

A NAT (Network Address Translation or Network Address Translator) is the virtualization of Internet Protocol (IP) addresses. NAT helps to improve security and decrease the number of IP addresses an organization needs.

A device that is configured with NAT will have at least one interface to the inside network and one to the outside network. In a typical environment, NAT is configured at the exit device between a stub domain (inside network) and the backbone. When a packet leaves the domain, NAT translates the locally significant source address into a globally unique address. When a packet enters the domain, NAT translates the globally unique destination address into a local address. If more than one exit point exists, each NAT must have the same translation table. NAT can be configured to advertise to the outside world only one address for the entire network. This ability provides additional security by effectively hiding the entire internal network behind that one address. If NAT cannot allocate an address because it has run out of addresses, it drops the packet and sends an Internet Control Message Protocol (ICMP) host unreachable packet to the destination.
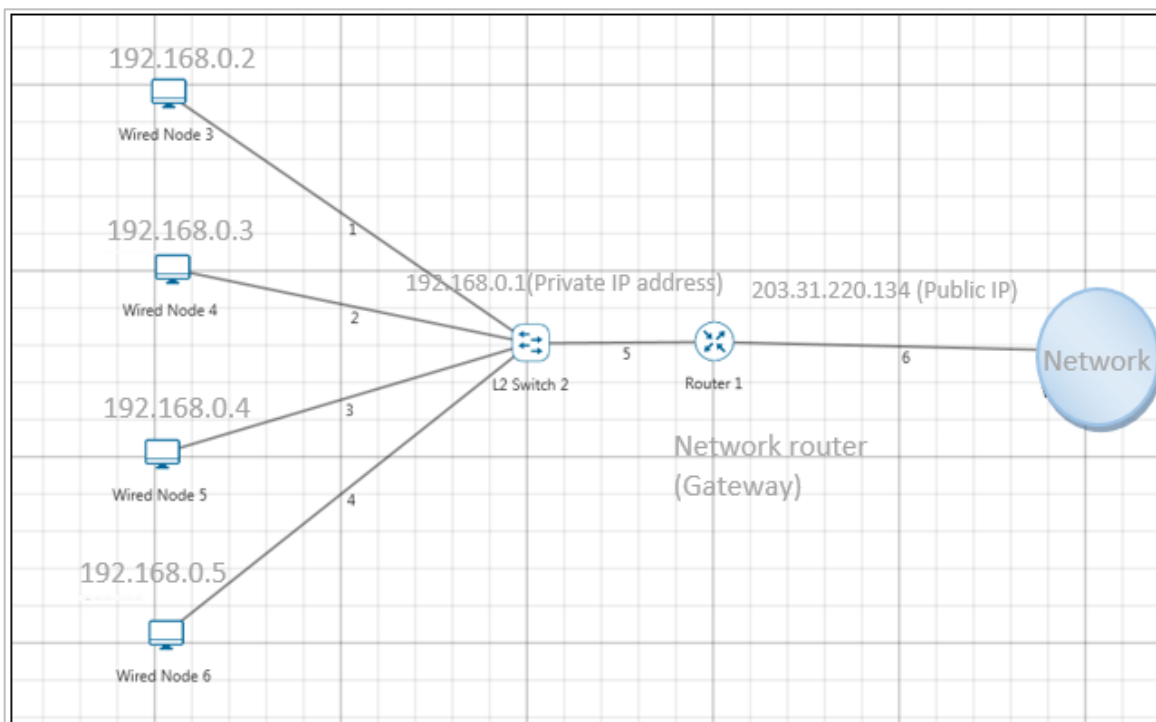
Figure 3-44: NAT implementation

NAT is secure since it hides network from the Internet. All communications from internal private network are handled by the NAT device, which will ensure all the appropriate translations are performed and provide a flawless connection between internal devices and the Internet.

In the above figure, a simple network of 4 hosts and one router that connects this network to the Internet. All hosts in the network have a private Class C IP Address, including the router's private interface (192.168.0.1), while the public interface that's connected to the Internet has a

real IP Address (203.31.220.134). This is the IP address the Internet sees as all internal IP addresses are hidden.

3.6.2 Network Setup

Open NetSim and click on **Experiments> Advanced Routing> Understanding Public IP Address and NAT (Network Address Translation)** then click on the tile in the middle panel to load the example as shown in below Figure 3-45.
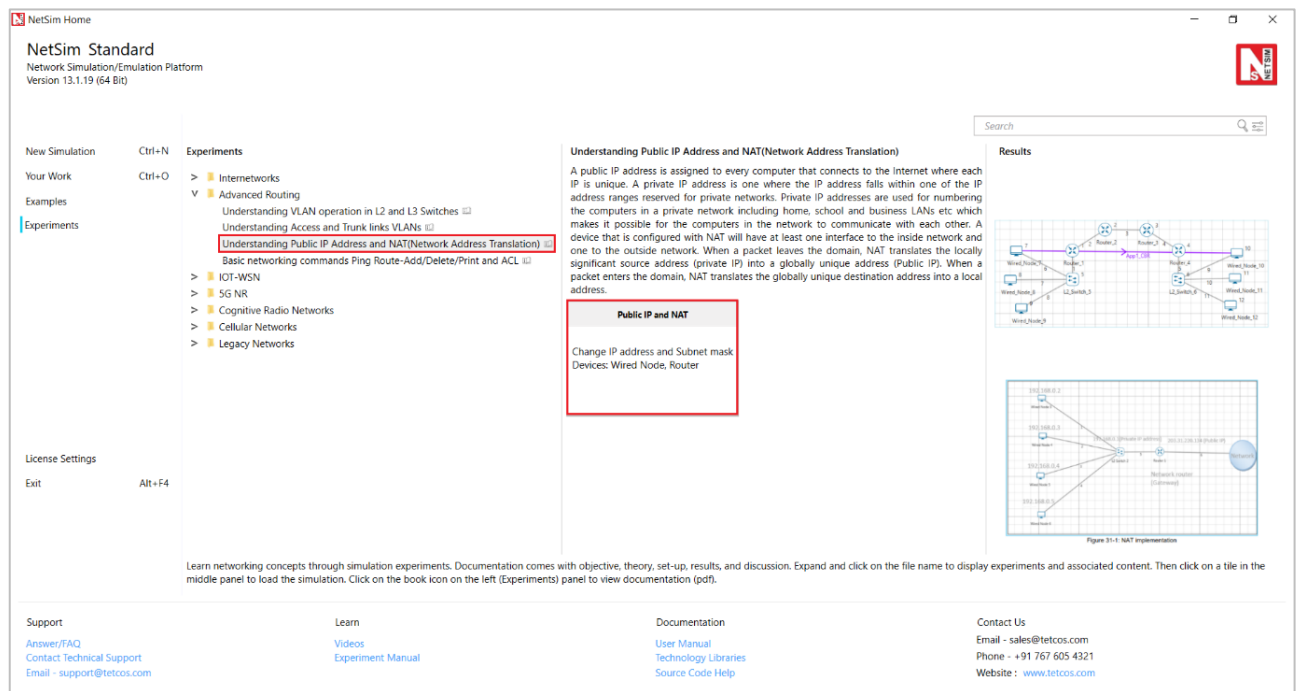
Figure 3-45: List of scenarios for the example of Understanding Public IP Address and NAT (Network Address Translation)

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-46.
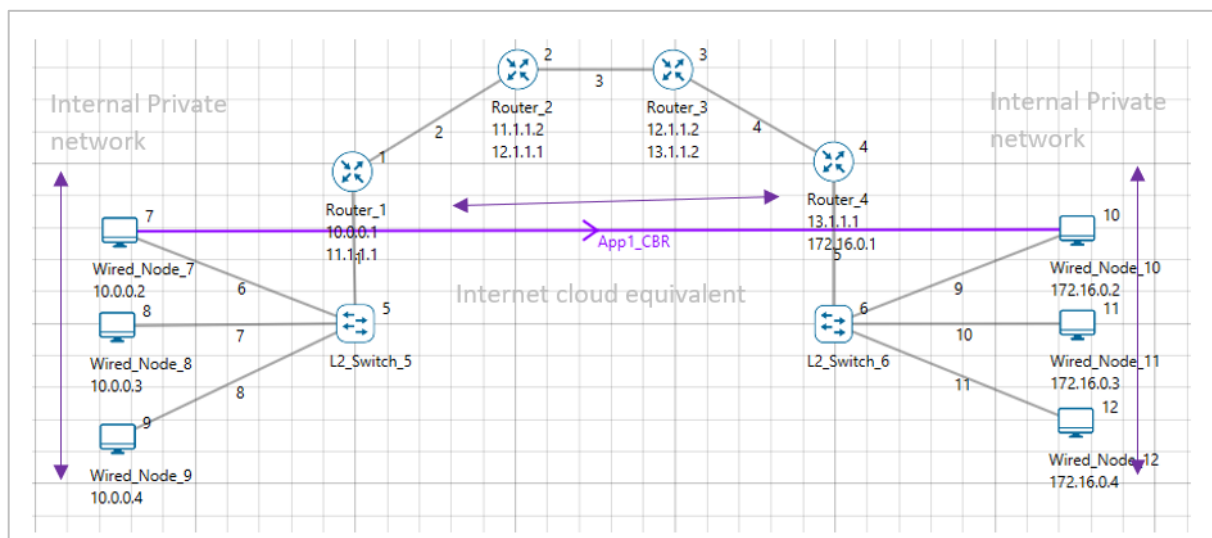
Figure 3-46: Network set up for studying the Understanding Public IP Address and NAT (Network Address Translation)

3.6.3 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 6 Wired Nodes, 2 L2 Switches, and 4 Routers in the **"Internetworks"** Network Library.

Step 2: In the INTERFACE (ETHERNET) > NETWORK LAYER of the Wired Nodes, the IP Address and the Subnet Mask are set as per the table given below Table 3-13.

Wired Node	IP address	Subnet mask
7	10.0.0.2	255.0.0.0
8	10.0.0.3	255.0.0.0
9	10.0.0.4	255.0.0.0
10	172.16.0.2	255.255.0.0
11	172.16.0.3	255.255.0.0
12	172.16.0.4	255.255.0.0

Table 3-13: IP Address and the Subnet mask for Wired nodes

Step 3: The IP Address and the Subnet Mask in Routers are set as per the table given below Table 3-14.

Router	Interface	IP address	Subnet mask
Router 1	Interface_2(WAN)	11.1.1.1	255.0.0.0
	Interface_1(Ethernet)	10.0.0.1	255.0.0.0
Router 2	Interface_1(WAN)	11.1.1.2	255.0.0.0
	Interface_2(WAN)	12.1.1.1	255.0.0.0
Router 3	Interface_1(WAN)	12.1.1.2	255.0.0.0
	Interface_2(WAN)	13.1.1.2	255.0.0.0
Router 4	Interface_1(WAN)	13.1.1.1	255.0.0.0
	Interface_2(Ethernet)	172.16.0.1	255.255.0.0

Table 3-14: IP Address and the Subnet Mask for Routers

Step 4: Right click on the Application Flow App1 CBR and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 7 i.e., Source to Wired Node 10 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000µs.

Additionally, the "Start Time(s)" parameter is set to 50(Figure 3-47), while configuring the application. This time is usually set to be greater than the time taken for OSPF Convergence (i.e., Exchange of OSPF information between all the routers), and it increases as the size of the network increases.

Figure 3-47: Application Properties Window

Step 5: Packet Trace is enabled, and hence we are able to track the route which the packets have chosen to reach the destination.

Step 6: Enable the plots and run the Simulation for 100 Seconds.

3.6.4 Output

After simulation open Packet Trace and filter Packet ID to 1.

PACKET_I	SEGMENT	PACKET_T	CONTROL	SOURCE_I	DESTINATION_I	SOURCE_IP	DESTINATION_IP	GATEWAY_IP	NEXT_HOP_IP
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	10.0.0.1	10.0.0.2	10.0.0.1
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	10.0.0.1	10.0.0.2	10.0.0.1
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	13.1.1.1	11.1.1.1	11.1.1.2
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	13.1.1.1	12.1.1.1	12.1.1.2
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	13.1.1.1	13.1.1.2	13.1.1.1
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	172.16.0.2	172.16.0.1	172.16.0.2
1	0	CBR	App1_CBR	NODE-7	NODE-10	10.0.0.2	172.16.0.2	172.16.0.1	172.16.0.2

Figure 3-48: Packet Trace

SOURCE_IP – source node IP (Node)

DESTINATION_IP – gateway IP/ destination IP (Router/ Node)

GATEWAY_IP – IP of the device which is transmitting a packet (Router/ Node)

NEXT_HOP_IP – IP of the next hop (Router/ Node)

Source node 7 (10.0.0.2) wouldn't know how to route to the destination and hence its default gateway is Router 1 with interface IP (10.0.0.1). So, the first line in the above screenshot specifies packet flow from Source Node 7 to L2 Switch 6 with SOURCE_IP (10.0.0.2), DESTINATION_IP (10.0.0.1), GATEWAY_IP (10.0.0.2) and NEXT_HOP_IP (10.0.0.1). Since Switch is Layer2 device there is no change in the IPs in second line. Third line specifies the packet flow from Router 1 to Router 2 with SOURCE_IP (10.0.0.2), DESTINATION_IP (13.1.1.1- IP of the router connected to destination. Since OSPF is running, the router is looks up the route to its destination from routing table), GATEWAY_IP (11.1.1.1) and NEXT_HOP_IP (11.1.1.2) and so on.

3.7 M/D/1 and M/G/1 Queues (Level 3)

3.7.1 Motivation

In this simulation experiment, we will study a model that is important to understand the queuing and delay phenomena in packet communication links. Let us consider the network shown in Figure 3-50. Wired_Node_1 is transmitting UDP packets to Wired_Node_2 through a router. Link 1 and Link 2 are of speed 10 Mbps. The packet lengths are 1250 bytes plus a 54-byte header, so that the time taken to transmit a packet on each 10 Mbps link is $\frac{1304 \times 8}{10} \mu$sec =1043.2 μsec. In this setting, we would like answers to the following questions:

1. We notice that the maximum rate at which these packets can be carried on a 10 Mbps link is $\frac{10^6}{1043.2} = 958.59$ packets per second. Can the UDP application send packets at this rate?

2. The time taken for a UDP packet to traverse the two links is $2 \times 1043.2 = 2086.4$ μsec. Is this the time it actually takes for a UDP packet generated at Wired_Node_1 to reach Wired_Node_2.

The answer to these questions depends on the manner in which the UDP packets are being generated at Wired_Node_1. If the UDP packets are generated at intervals of 1043.2 μsec then successive packets will enter the Link 1, just when the previous packet departs. In practice, however, the UDP packets will be generated by a live voice or video source. Depending on the voice activity, the activity in the video scene, and the coding being used for the voice and the video, the rate of generation of UDP packets will vary with time. Suppose two packets were generated during the time that one packet is sent out on Link 1, then one will have to wait, giving rise to queue formation. This also underlines the need for a buffer to be placed before each link; a buffer is just some dynamic random-access memory in the link interface card into which packets can be stored while waiting for the link to free up.

Queuing models permit us to understand the phenomenon of mismatch between the service rate (e.g., the rate at which the link can send out packets) and the rate at which packets arrive. In the network in Figure 3-50, looking at the UDP flow from Wired_Node_1 to Wired_Node_2, via Router 3, there are two places at which queueing can occur. At the interface between Wired_Node_1 and Link 1, and at the interface between Router 3 and Link 2. Since the only flow of packets is from Wired_Node_1 to Wired_Node_2, all the packets entering Link 2 are from Link 1, and these are both of the same bit rate. Link 2, therefore, cannot receive packets faster than it can serve them and, at any time, only the packet currently in transmission will be at Link 2. On the other hand at the Wired_Node_1 to Link 1 interface, the packets are generated directly by the application, which can be at arbitrary rates, or inter-packet times.

Suppose that, at Wired_Node_1, the application generates the successive packets such that the time intervals between the successive packets being generated are statistically independent, and the probability distribution of the time intervals has a negative exponential density, i.e., of the form $\lambda\,e^{-\lambda x}$, where λ (packets per second) is a parameter, called the *rate* parameter, and x (seconds) is the argument of the density. The application generates the entire packet *instantaneously,* i.e., all the bits of the packet arrive from the application together, and enter the buffer at Link 1, to wait behind the other packets, in a first-in-first-out manner. The resulting random process of the points at which packets enter the buffer of Link 1 is called a Poisson Process of rate λ packets per second. The buffer queues the packets while Link 1 serves them with *service time* $b = 1043.2\,\mu$sec. Such a queue is called an M/D/1 queue, where the notation is to be read as follows.

- The M before the first slash (denoting "Markov") denotes the Poisson Process of instants at which packets enter the buffer.
- The D between the two slashes (denoting "Deterministic") denotes the fixed time taken to serve each queued packet.
- The 1 after the second slash denotes that there is just a single server (Link 1 in our example)

This way of describing a single server queueing system is called Kendall's Notation.

In this experiment, we will understand the M/D/1 model by simulating the above-described network on NetSim. The M/D/1 queueing model, however, is simple enough that it can be mathematically analyzed in substantial detail. We will summarize the results of this analysis in the next section. The simulation results from NetSim will be compared with the analytical results.

3.7.2 Mathematical Analysis of the M/D/1 Queue

The M/D/1 queueing system has a random number of arrivals during any time interval. Therefore, the number of packets waiting at the buffer is also random. It is possible to mathematically analyze the *random process* of the number of waiting packets. The procedure for carrying out such analysis is, however, beyond the scope of this document. We provide the final formulas so that the simulation results from NetSim can be compared with those provided by these formulas.

As described earlier, in this chapter, the M/D/1 queue is characterized by two parameters: λ (packets per second), which is the arrival rate of packets into the buffer, and μ (packets per second), which is the rate at which packets are removed from a nonempty queue. Note that $1/\mu$ is the service time of each packet.

Define $\rho = \lambda \times \frac{1}{\mu} = \lambda/\mu$. We note that ρ is the average number of packets that arrive during the service time of a packet. Intuitively, it can be expected that if $\rho > 1$ then packets arrive faster than the rate at which they can be served, and the queue of packets can be expected grow without bound. When $\rho < 1$ we can expect the queue to be "stable." When $\rho = 1$, the service rate is exactly matched with the arrival rate; due to the randomness, however, the queue can still grown without bound. The details of this case are beyond the scope of this document.

For the kth arriving packet, denote the instant of arrival by a_k, the instant at which service for this packet starts as s_k, and the instant at which the packet leaves the system as d_k. Clearly, for all $k, d_k - s_k = \frac{1}{\mu}$, the deterministic service time. Further define, for each k,

$$W_k = s_k - a_k$$

$$T_k = d_k - a_k$$

i.e., W_k is called the *queuing delay,* i.e., time from the arrival of the kth packet until it starts getting transmitted, whereas T_k is called the *total delay,* i.e., the time from the arrival of the kth packet until its transmission is completed. Considering a large number of packets, we are interested in the average of the values $W_1, W_2, W_3, \cdots$, i.e., the *average queueing time* of the packets. Denote this average by W. By mathematical analysis of the packet queue process, it can be shown that for an M/D/1 queueing system,

$$W = \frac{1}{2\mu} \times \frac{\rho}{1 - \rho}$$

Denoting by T, the average total time in the system (i.e., the average of $T_1, T_2, T_3, \cdots$), clearly

$$T = W + \frac{1}{\mu}.$$

Observe the following from the above formula:

1. As ρ approaches 0, W becomes 0. This is clear, since, when the arrival rate becomes very small, and arriving packet sees a very small queue. For arrival rate approaching 0, packets get served immediately on arrival.
2. As ρ increases, W inreases.
3. As ρ approaches 1 (from values smaller than 1), the mean delay goes to ∞.

We will verify these observations in the NetSim simulation.

3.7.3 The Experimental Setup

Open NetSim and click on **Experiments> Internetworks> Network Performance> MD1 and MG1 Queues** then click on the tile in the middle panel to load the example as shown in below Figure 3-49.
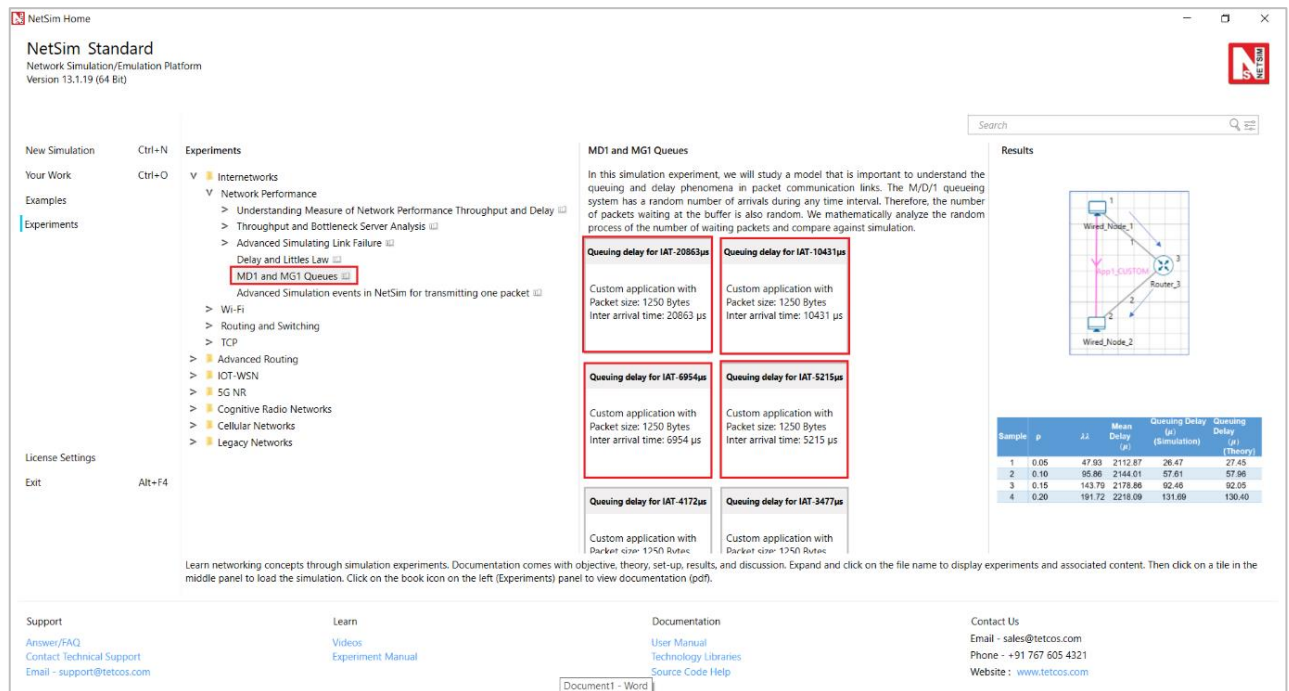
Figure 3-49: List of scenarios for the example of MD1 and MG1 Queues

NetSim UI displays the configuration file corresponding to this experiment as shown above:

The model described at the beginning of this chapter is shown in Figure 3-50.
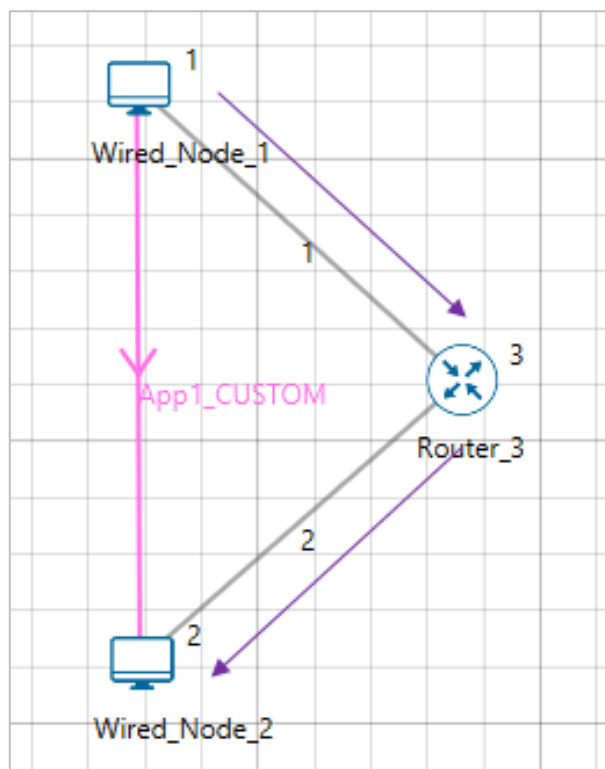
Figure 3-50: A single wired node (Wired_Node_1) sending UDP packets to another wired node (Wired_Node_2) through a router (Router 3). The packet interarrival times at Wired_Node_1 are exponentially distributed, and packets are all of the same length, i.e., 1250 bytes plus UDP/IP header.

3.7.4 Procedure

Queuing delay for IAT-20863 (μs) Sample:

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 1 Router in the **"Internetworks"** Network Library.

Step 2: Link Properties are set as per the table given below Table 3-15.

Link Properties	Link 1	Link 2
Uplink Speed (Mbps)	10	10
Downlink Speed (Mbps)	10	10
Uplink BER	0	0
Downlink BER	0	0
Uplink Propagation Delay (µs)	0	0
Downlink Propagation Delay (µs)	0	0

Table 3-15: Wired link properties

Step 3: Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wired Node 1 i.e. Source to Wired Node 2 i.e. Destination. Transport Protocol is set to **UDP** with Packet Size set to 1250 Bytes and Inter Arrival Time set to 20863 µs and distribution to Exponential.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 0.096 Mbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps) = Packet\ Size\ (Bytes) * 8/Interarrival\ time\ (µs)$$

Step 4: Packet Trace is enabled in NetSim GUI. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis.

Step 5: Plots are enabled in NetSim GUI. Run the Simulation for 100 Seconds.

Similarly, the other samples are created by changing the Inter Arrival Time per the formula

$$IAT = \frac{10^6}{958.59 * \rho}$$

as per the table given below Table 3-16.

P	IAT (µs)
0.05	20863
0.1	10431
0.15	6954
0.2	5215
0.25	4172
0.3	3477
0.35	2980

0.4	2607
0.45	2318
0.5	2086
0.55	1896
0.6	1738
0.65	1604
0.7	1490
0.75	1390
0.8	1303
0.85	1227
0.9	1159
0.95	1098

Table 3-16: Inter Arrival Time Settings

Even though the packet size at the application layer is 1250 bytes, as the packet moves down the layers, overhead is added. The overheads added in different layers are shown in the below table and can be obtained from the packet trace:

Layer	Overhead (Bytes)
Transport Layer	8
Network Layer	20
MAC layer	26
Physical Layer	0
Total	54

Table 3-17: Overheads added to a packet as it flows down the network stack

3.7.5 Obtaining the Mean Queuing delay from the Simulation Output

After running the simulation, note down the **"Mean Delay"** in the Application Metrics within the Results Dashboard. This is the average time between the arrival of packets into the buffer at Wired_Node_1, and their reception at Wired_Node_2.

As explained in the beginning of this chapter, for the network shown in Figure 3-50, the end-to-end delay of a packet is the sum of the queueing delay at the buffer between the wired-node and Link_1, the transmission time on Link_1, and the transmission time on Link_2 (there being no queueing delay between the Router and Link_2). It follows that.

$$Mean\ Delay = \left(\frac{1}{2\mu} \times \frac{\rho}{1-\rho}\right) + \frac{1}{\mu} + \frac{1}{\mu}$$

3.7.6 Output Table

Sample	ρ	λ	Mean Delay (μs)	Queuing Delay (μs) (Simulation)	Queuing Delay (μs) (Theory)
1	0.05	47.93	2112.87	26.47	27.45
2	0.10	95.86	2144.01	57.61	57.96
3	0.15	143.79	2178.86	92.46	92.05
4	0.20	191.72	2218.09	131.69	130.40
5	0.25	239.65	2259.11	172.71	173.87

6	0.30	287.58	2309.49	223.09	223.54
7	0.35	335.51	2365.74	279.34	280.86
8	0.40	383.44	2435.65	349.25	347.73
9	0.45	431.37	2513.79	427.39	426.76
10	0.50	479.30	2608.38	521.98	521.60
11	0.55	527.22	2721.59	635.19	637.51
12	0.60	575.15	2864.88	778.48	782.40
13	0.65	623.08	3052.84	966.44	968.68
14	0.70	671.01	3304.58	1218.18	1217.07
15	0.75	718.94	3633.66	1547.26	1564.80
16	0.80	766.87	4160.39	2073.99	2086.40
17	0.85	814.80	5115.95	3029.55	2955.73
18	0.90	862.73	6967.16	4880.76	4694.39
19	0.95	910.66	12382.98	10296.58	9910.39

Table 3-18: Mean Delay, Queueing delay from Simulation and Queueing delay from analysis

Comparison Chart
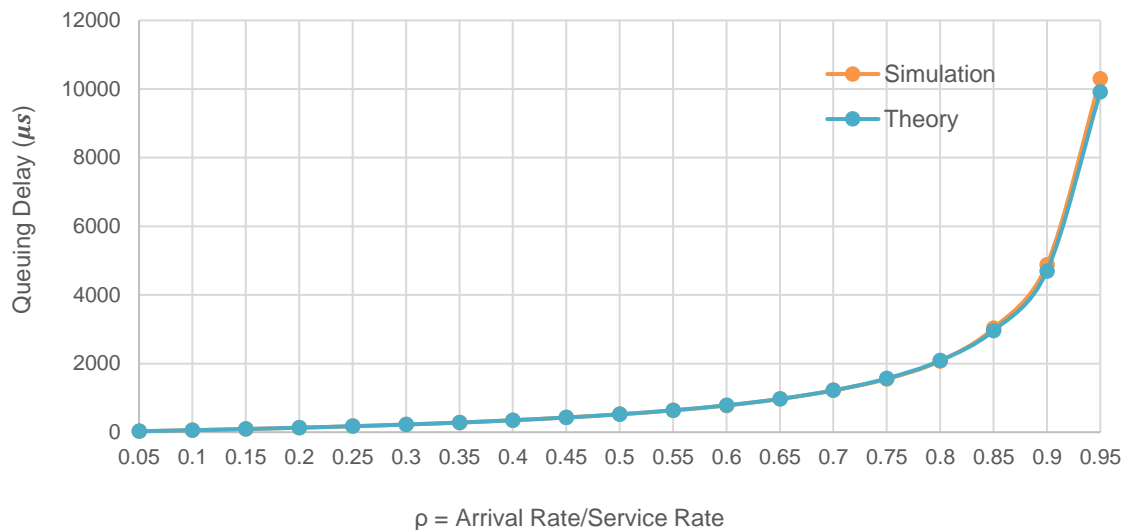
Figure 3-51: Comparison of queueing delay from simulation and analysis

3.7.7 Advanced Topic: The M/G/1 Queue

In Section 3.7.1, we introduced the M/D/1 queue. Successive packets were generated instantly at exponentially distributed time intervals (i.e., at the points of a Poisson process); this gave the "M" in the notation. The packets were all of fixed length; this gave the "D" in the notation. Such a model was motivated by the transmission of packetized voice over a fixed bit rate wireline link. The voice samples are packetized into constant length UDP packets. For example, typically, 20ms of voice samples would make up a packet, which would be emitted at the instant that the 20ms worth of voice samples are collected. A voice source that is a part of a conversation would have listening periods, and "silence" periods between words and sentences. Thus, the intervals between emission instants of successive UDP packets would be random. A simple model for these random intervals is that they are exponentially distributed, and independent from packet to packet. This, formally, is called the Poisson point process. With

exponentially distributed (and independent) inter-arrival times, and fixed length packets we obtain the M/D/1 model. On the other hand, some applications, such as video, generate unequal length packets. Video frames could be encoded into packets. To reduce the number of bits being transmitted, if there is not much change in a frame, as compared to the previous one, then the frame is encoded into a small number of bits; on the other hand if there is a large change then a large number of bits would need to be used to encode the new information in the frame. This motivates variable packet sizes. Let us suppose that, from such an application, the packets arrive at the points of a Poisson process of rate λ, and that the randomly varying packet transmission times can be modelled as independent and identically distributed random variables, $B_1, B_2, B_3, \cdots$, with mean b and second moment $b^{(2)}$, i.e., variance $b^{(2)} - b^2$. Such a model is denoted by M/G/1, where M denotes the Poisson arrival process, and G ("general") the "generally" distributed service times. Recall the notation M/D/1 (from earlier in this section), where the D denoted fixed (or "deterministic") service times. Evidently, the M/D/1 model is a special case of the M/G/1 model.

Again, as defined earlier in this section, let W denote the mean queueing delay in the M/G/1 system. Mathematical analysis of the M/G/1 queue yields the following formula for W

$$W = \frac{\rho}{1-\rho} \frac{b^{(2)}}{2b}$$

where, as before, $\rho = \lambda b$. This formula is called the Pollacek-Khinchine formula or P-K formula, after the researchers who first obtained it. Denoting the variance of the service time by $Var(B)$, the P-K formula can also be written as

$$W = \frac{\rho b}{2(1-\rho)} \left(\frac{Var(B)}{b^2} + 1\right)$$

Applying this formula to the M/D/1 queue, we have $Var(B) = 0$. Substituting this in the M/G/1 formula, we obtain.

$$W = \frac{\rho}{1-\rho} \frac{b}{2}$$

which, with $b = 1/\mu$, is exactly the M/D/1 mean queuing delay formula displayed earlier in this section.

3.7.8 A NetSim Exercise Utilising the M/G/1 Queue

In this section we demonstrate the use of the M/G/1 queueing model in the context of the network setup shown in Figure 3-50. The application generates exponentially distributed data segment with mean d bits, i.e., successive data segment lengths are sampled independently from an exponential distribution with rate parameter $\frac{1}{d}$. Note that, since packets are integer multiples of bits, the exponential distribution will only serve as an approximation. These data

segments are then packetised by adding a constant length header of length h bits. The packet generation instants form a Poisson process of rate λ. Let us denote the link speed by c. Let us denote the random data segment length by X and the packet transmission time by B, so that

$$B = \frac{X + h}{c}$$

Denoting the mean of B by b, we have

$$b = \frac{d + h}{c}$$

Further, since h is a constant,

$$Var(B) = Var(X)/c^2$$

These can now be substituted in the P-K formula to obtain the mean delay in the buffer between Node 1 and Link 1.

We set the mean packet size to 100B or 800 bits, the header length h = 54B or 432 bits and $\lambda = 5000$

For a 10Mbps link, the service rate $\mu = \frac{10*10^6}{154 \times 8} = 8116.8$

Using the Pollaczek–Khinchine (PK) formula, the waiting time for a M/G/1 queuing system is

$$w = \frac{\rho + \lambda \times \mu \times Var(s)}{2(\mu - \lambda)}$$

Where $var(s)$ is the variance of the service time distribution S. Note that

$var(s) = \frac{1}{(\mu')^2}$ where μ' is the mean service time of the exponential random variable (100B packets and not 154B)

$$\mu' = \frac{10 \times 10^6}{100 * 8} = 12500$$

Hence substituting into the PK formula, one gets

$$w = \frac{0.4 + \dfrac{(3467.7 \times 8116.8)}{12500^2}}{2\,(8116.8 - 3246.7)} = 59.5\ \mu s$$

By simulation the queuing delay is 60.5 μs.

The queuing delay is not available in the NetSim results dashboard. It can be got from the packet trace. It is the average of (PHY_layer_Arrival_time - APP_layer_arrival time) for packets being sent from Node_1.

3.8 Understand the working of OSPF (Level 3)

3.8.1 Objective

To understand the working of OSPF and Shortest Path First (SPF) tree creation.

3.8.2 Theory

OSPF

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) standardized by the Internet Engineering Task Force (IETF) and commonly used in large Enterprise networks. OSPF is a link-state routing protocol providing fast convergence and excellent scalability. Like all link-state protocols, OSPF is very efficient in its use of network bandwidth.

Shortest path First Algorithm

OSPF uses a shorted path first algorithm to build and calculate the shortest path to all known destinations. The shortest path is calculated with the use of the Dijkstra algorithm. The algorithm by itself is quite complicated. This is a very high level, simplified way of looking at the various steps of the algorithm:

- Upon initialization or due to any change in routing information, a router generates a link-state advertisement. This advertisement represents the collection of all link-states on that router.
- All routers exchange link-states by means of flooding. Each router that receives a link-state update should store a copy in its link-state database and then propagate the update to other routers.
- After the database of each router is completed, the router calculates a Shortest Path Tree to all destinations. The router uses the Dijkstra algorithm in order to calculate the shortest path tree. The destinations, the associated cost and the next hop to reach those destinations form the IP routing table.
- In case no changes in the OSPF network occur, such as cost of a link or a network being added or deleted, OSPF should be very quiet. Any changes that occur are communicated through link-state packets, and the Dijkstra algorithm is recalculated in order to find the shortest path.

The algorithm places each router at the root of a tree and calculates the shortest path to each destination based on the cumulative cost required to reach that destination. Each router will have its own view of the topology even though all the routers will build a shortest path tree using the same link-state database.

Example

Refer Pg. no.18 from OSPF RFC 2328 (https://tools.ietf.org/html/rfc2328#section-2.3)

The below network shows a sample map of an Autonomous System.
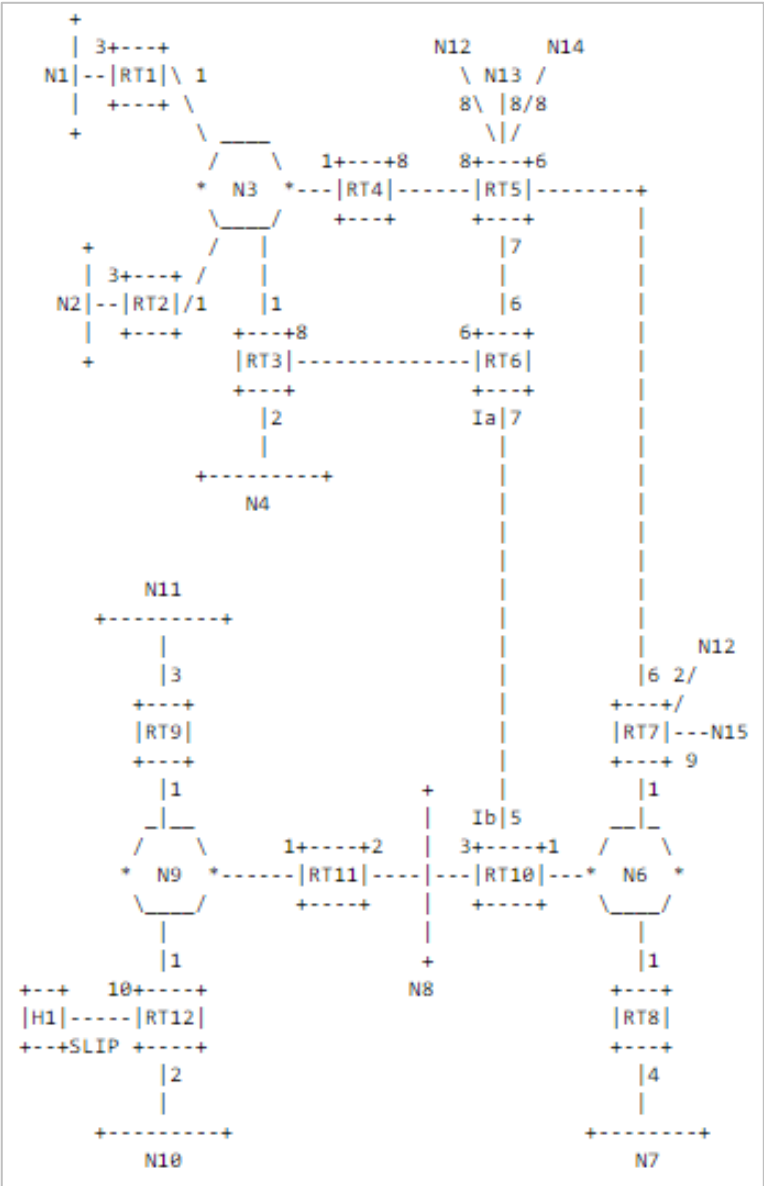

```
        +
        | 3+---+                        N12      N14
     N1|--|RT1|\ 1                        \ N13 /
        |  +---+ \                        8\ |8/8
        +         \                         \|/
                   \ ___      1+---+8     8+---+6
                   /    \    *---|RT4|------|RT5|---------+
                 *   N3  *---|RT4|------|RT5|---------+
                   \___/      +---+        +---+       |
           +        /   |                  |7          |
           | 3+---+ /   |                  |           |
        N2|--|RT2|/1    |1                 |6          |
           |  +---+    +---+8              6+---+       |
           +          |RT3|--------------|RT6|          |
                      +---+              +---+          |
                       |2                Ia|7           |
                       |                   |            |
                +---------+                |            |
                    N4                     |            |
                                           |            |
                                           |            |
                                           |            |
                                           |            |
                                           |            |
           N11                             |            |
        +---------+                        |            |
            |                              |            |  N12
            |3                             |            |6 2/
         +---+                             |          +---+/
         |RT9|                             |          |RT7|---N15
         +---+                             |          +---+ 9
          |1                               |           |1
         _|__                   +          | Ib|5     _|_
         /   \       1+----+2   |    3+----+1  /   \
        *  N9  *------|RT11|----|---|RT10|---*  N6  *
         \___/       +----+     |    +----+   \___/
           |                    |     |          |
           |1                   +     |          |1
      +--+   10+----+           N8    |        +---+
      |H1|------|RT12|                |        |RT8|
      +--+SLIP +----+                 +        +---+
             |2                                  |4
             |                                   |
        +---------+                         +--------+
             N10                                N7
```
Figure 3-52: Sample maps of an Autonomous system

A cost is associated with the output side of each router interface. This cost is configurable by the system administrator. The lower the cost, the more likely the interface is to be used to forward data traffic. Costs are also associated with the externally derived routing data (e.g., the BGP-learned routes).

The directed graph resulting from the above network is depicted in the following table. Arcs are labelled with the cost of the corresponding router output interface. Arcs having no labelled cost have a cost of 0. Note that arcs leading from networks to routers always have cost 0.

	FROM															
		RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	N3	N6	N8	N9

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T O RT1													0		
RT2													0		
RT3						6							0		
RT4					8								0		
RT5				8		6	6								
RT6			8		7										
RT7					6									0	
RT8														0	
RT9															0
RT10						7								0	0
RT11														0	0
RT12															0
N1	3														
N2		3													
N3	1	1	1	1											
N4			2												
N5															
N6							1	1		1					
N7								4							
N8										3	2				
N9									1		1	1			

N 1 0									2			
N 1 1						3						
N 1 2			8		2							
N 1 3			8									
N 1 4			8									
N 1 5					9							
H 1									1 0			

Table 3-19: Directed graph

A router generates its routing table from the above directed graph by calculating a tree of shortest paths with the router itself as root. Obviously, the shortest-path tree depends on the router doing the calculation. The shortest-path tree for Router RT6 in our example is depicted in the following figure.


```
                    RT6(origin)
        RT5 o-------------o-----------o Ib
           /|\     6        |\      7
         8/8|8\             | \
         /  |  \          6|  \
        o   |   o          |   \7
       N12  o  N14         |    \
           N13        2    |     \
              N4 o-----o RT3 \
                 /        \   5
               1/        RT10 o-------o Ia
               /             |\
          RT4 o-----o N3     3| \1
              /|            |  \ N6     RT7
             / |          N8 o   o---------o
            /  |            |   |        /|
        RT2 o   o RT1        |   |      2/ |9
           /    |           |   |RT8  /  |
          /3   |3       RT11 o   o     o   o
         /     |            |   |    N12 N15
      N2 o     o N1         1|  |4
                            |   |
                          N9 o   o N7
                            /|
                           / |
        N11     RT9       /  |RT12
         o--------o-------o   o--------o H1
              3                |   10
                              |2
                              |
                              o N10
```
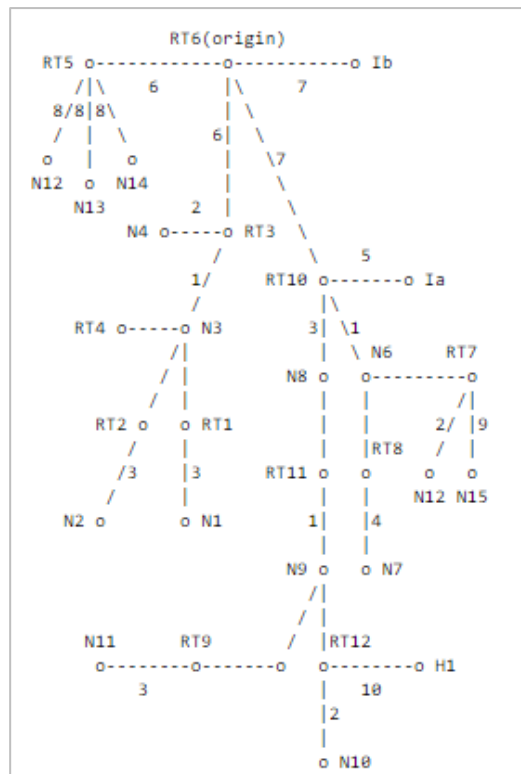
Figure 3-53: SPF tree for Router 6

Routing Table

The IP forwarding table formed in the routers and nodes can be accessed from the IP_Forwarding_Table list present in the Simulation Results window as shown below:

Node-8: As shown in the below screenshot, Node-8 has only one interface with IP 11.7.1.2 and its network address are 11.7.0.0 since its network mask is 255.255.0.0. The first entry represents the router forwards packets intended to the subnet 11.7.0.0 to the interface with the IP 11.7.1.2. 239.12.14.5 is the multicast Group address and 224.0.0.1 is the address for all multicast routers The IP forwarding table formed in the routers and nodes can be accessed from the IP_Forwarding_Table list present in the Simulation Results window as shown below:

The tree gives the entire path to any destination network or host. However, only the next hop to the destination is used in the forwarding process. Note also that the best route to any router has also been calculated. For the processing of external data, we note the next hop and distance to any router advertising external routes. The resulting routing table for Router RT6 is shown in the following table.

Destination	Next hop	Distance
N1	RT3	10
N2	RT3	10
N3	RT3	7
N4	RT3	8
N6	RT10	8
N7	RT10	12
N8	RT10	10
N9	RT10	11
N10	RT10	13
N11	RT10	14
H1	RT10	21
RT5	RT5	6
RT7	RT10	8
N12	RT10	10
N13	RT5	14
N14	RT5	14
N15	RT10	17

Table 3-20: Routing Table for RT6

Distance calculation

Router6 has 3 interfaces i.e., RT3, RT5 and RT10. The distance obtained is 10 for destination N1 via RT3 interface. The packets from Router6 would reach N1 via RT3, N3 and RT1. The cost assigned to routers in this path is 6+1+3 = 10 (cost can be seen in SPF tree for Router6). This is how distance is calculated.

3.8.3 Network Setup

Open NetSim and click on **Experiments> Internetworks> Routing and Switching> Understand the working of OSPF** then click on the tile in the middle panel to load the example as shown in below Figure 3-54.
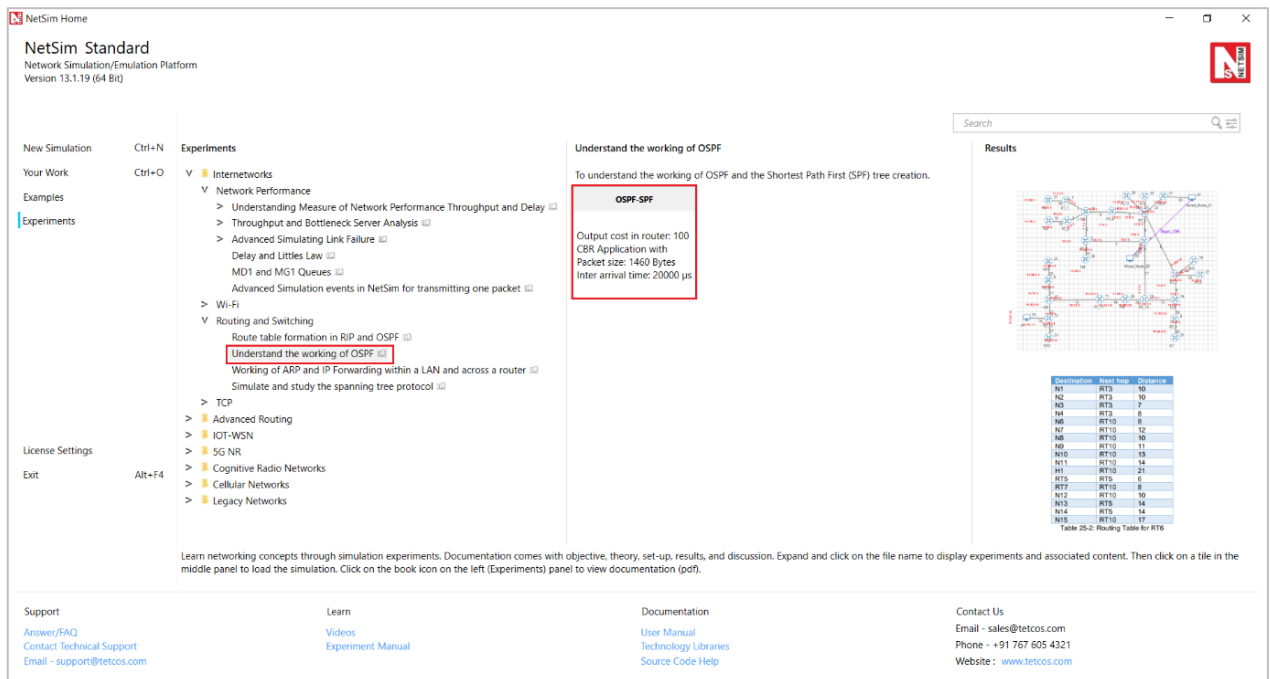
Figure 3-54: List of scenarios for the example of Understand the working of OSPF

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 3-55.
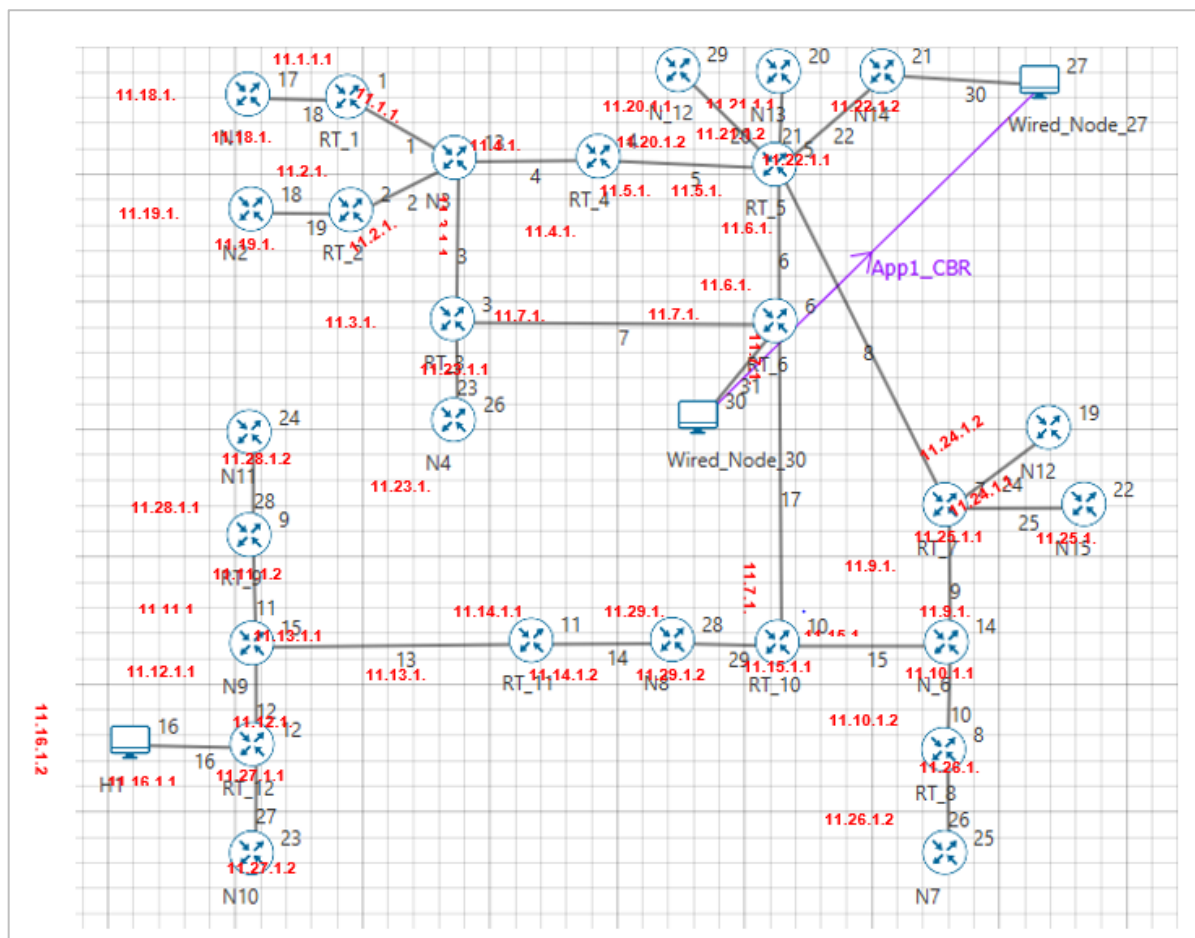
Figure 3-55: Network topology showing IP Addresses in each Router interface and in end nodes

The above network was created in NetSim and it is similar to the network as per the OSPF RFC 2328 (Refer Pg. no. 19 - https://tools.ietf.org/html/rfc2328#page-23)

3.8.4 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 3 Wired Nodes and 27 Routers in the **"Internetworks"** Network Library.

Step 2: The Output Cost for all the Routers in the network is set as per the network shown in Figure 3-56.
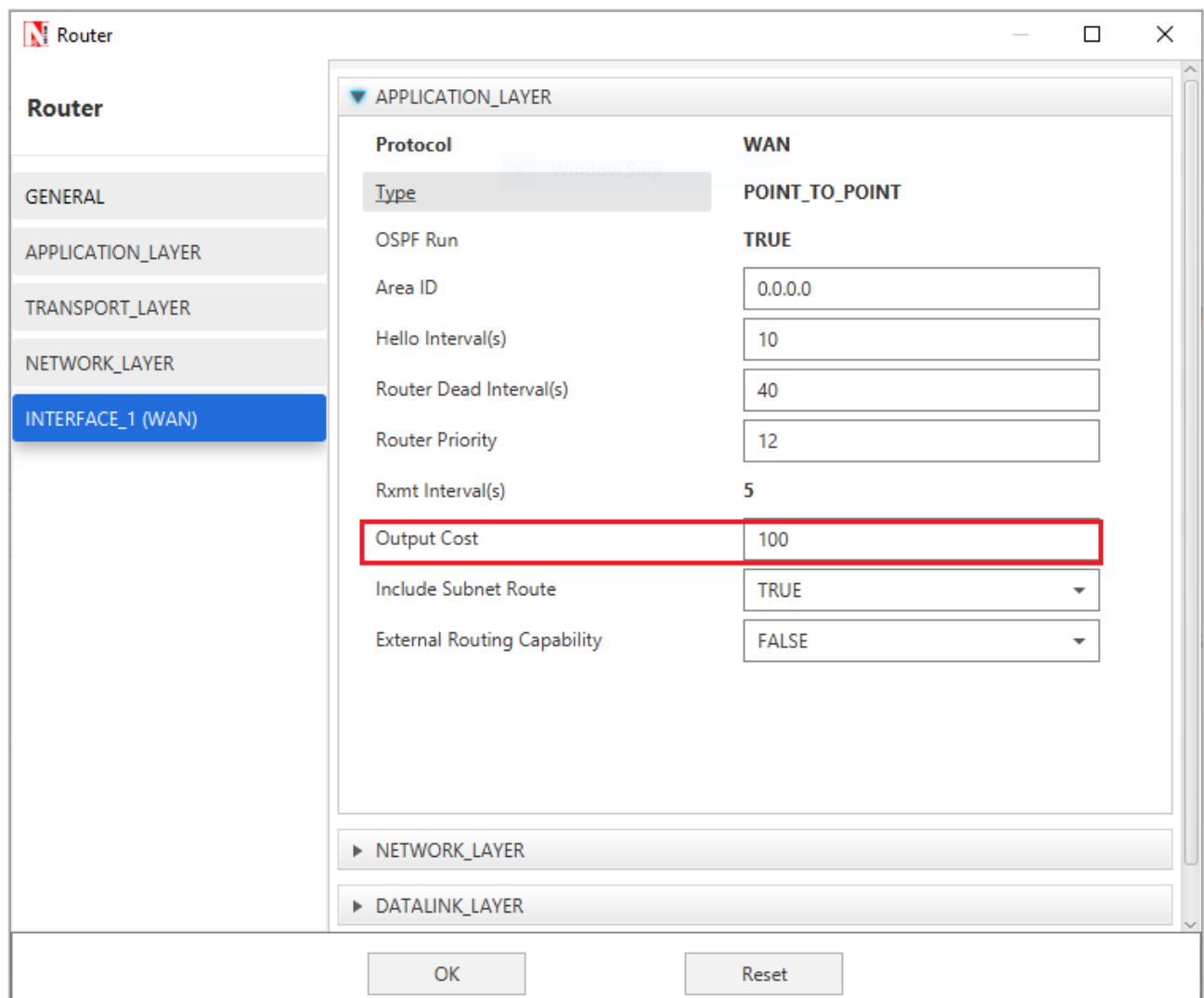
Figure 3-56: Output Cost is set to 100 in WAN interface

Step 3: Packet Trace is enabled in the NetSim GUI, and hence we are able to track the route which the packets have chosen to reach the destination based on the Output Cost that is set.

Step 4: Right click on the Application Flow App1 CBR and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 30 i.e. Source to Wired Node 27 i.e. Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000µs. Additionally, the "Start Time(s)" parameter is set to 30, while configuring the application. This time is usually set to be greater than the time taken for OSPF Convergence (i.e. Exchange of OSPF information between all the routers), and it increases as the size of the network increases.

3.8.5 Output

The following image is a depiction of the shortest path first tree created in NetSim. This is for representational purposes and cannot be opened in NetSim. The blue color numbers are the "Output Cost" parameter of the link and is set by the user in Router > WAN Interface > Application layer. The red numbers are the IP addresses of the interfaces of the Routers.
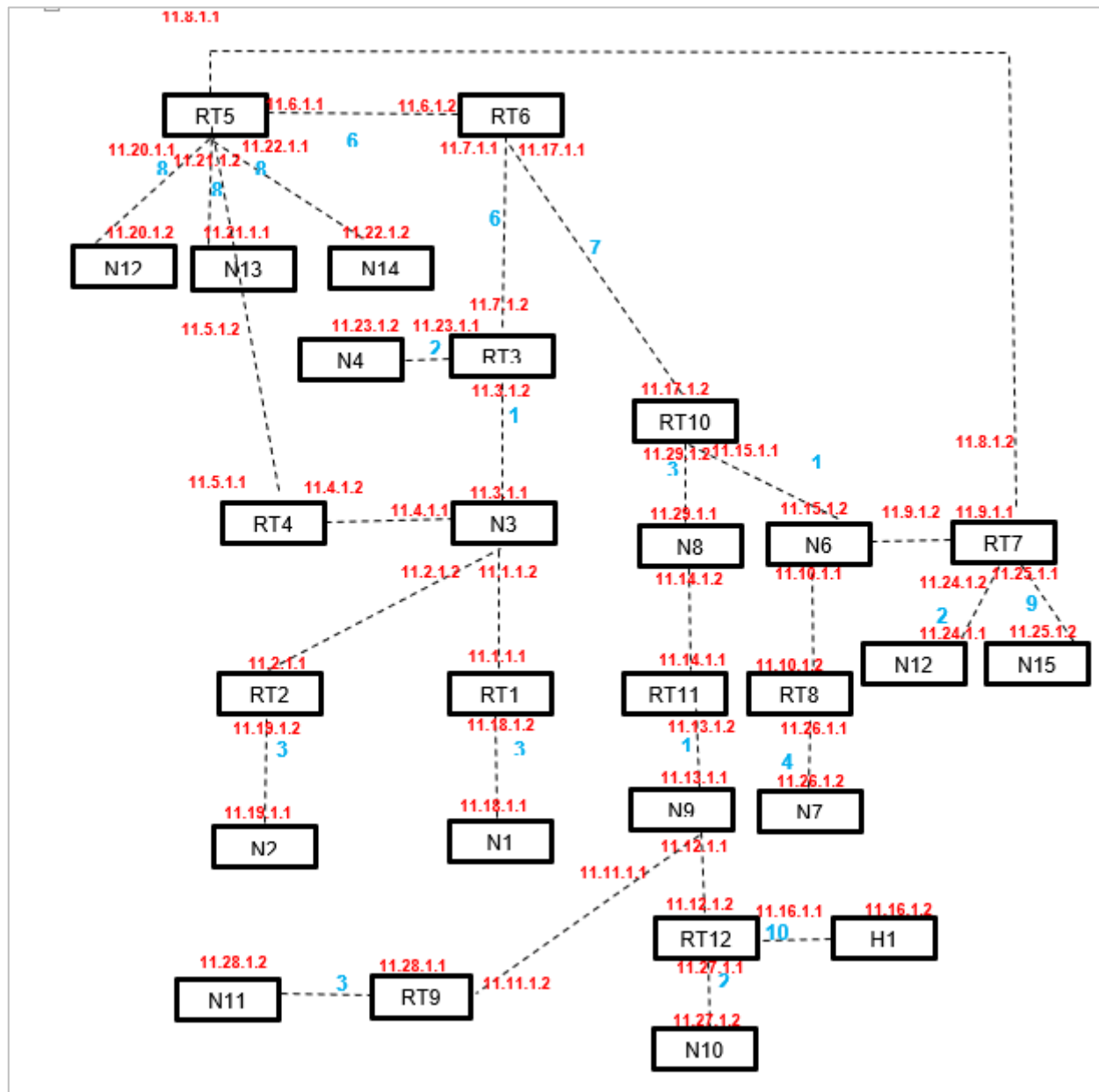
Figure 3-57: SPF tree for Router 6

NOTE: **NetSim, does not implement Link type3 (Link to Stub Network). Hence users would notice a slight difference between the SPF trees of RFC and NetSim.**

The IP forwarding table formed in the routers can be accessed from the IP_Forwarding_Table list present in the Simulation Results window as shown below Figure 3-58.

RT_6_Table

a) RT_6

Network Destination	Netmask/Prefix len	Gateway	Interface	Metrics	Type
11.3.1.2	255.255.255.255	11.7.1.2	11.7.1.1	100	OSPF
11.7.1.2	255.255.255.255	11.7.1.2	11.7.1.1	100	OSPF
11.23.1.1	255.255.255.255	11.7.1.2	11.7.1.1	100	OSPF
11.5.1.2	255.255.255.255	11.6.1.1	11.6.1.2	100	OSPF
11.6.1.1	255.255.255.255	11.6.1.1	11.6.1.2	100	OSPF
11.8.1.1	255.255.255.255	11.6.1.1	11.6.1.2	100	OSPF
11.20.1.2	255.255.255.255	11.6.1.1	11.6.1.2	100	OSPF
11.21.1.2	255.255.255.255	11.6.1.1	11.6.1.2	100	OSPF
11.22.1.1	255.255.255.255	11.6.1.1	11.6.1.2	100	OSPF
11.29.1.2	255.255.255.255	11.17.1.2	11.17.1.1	100	OSPF
11.15.1.1	255.255.255.255	11.17.1.2	11.17.1.1	100	OSPF
11.17.1.2	255.255.255.255	11.17.1.2	11.17.1.1	100	OSPF
11.4.1.2	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.5.1.1	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.8.1.2	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF

a)

RT_6_Table

b) RT_6

| Network Destination	Netmask/Prefix len	Gateway	Interface	Metrics	Type
11.6.1.2	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.9.1.1	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.24.1.2	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.25.1.1	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.1.1.2	255.255.255.255	11.7.1.2	11.7.1.1	200	OSPF
11.2.1.2	255.255.255.255	11.7.1.2	11.7.1.1	200	OSPF
11.3.1.1	255.255.255.255	11.7.1.2	11.7.1.1	200	OSPF
11.4.1.1	255.255.255.255	11.7.1.2	11.7.1.1	200	OSPF
11.9.1.2	255.255.255.255	11.17.1.2	11.17.1.1	200	OSPF
11.10.1.1	255.255.255.255	11.17.1.2	11.17.1.1	200	OSPF
11.15.1.2	255.255.255.255	11.17.1.2	11.17.1.1	200	OSPF
11.21.1.1	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.22.1.2	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.23.1.2	255.255.255.255	11.7.1.2	11.7.1.1	200	OSPF
11.14.1.2	255.255.255.255	11.17.1.2	11.17.1.1	200	OSPF
11.29.1.1	255.255.255.255	11.17.1.2	11.17.1.1	200	OSPF

b)

RT_6_Table

c) RT_6

Network Destination	Netmask/Prefix len	Gateway	Interface	Metrics	Type
11.20.1.1	255.255.255.255	11.6.1.1	11.6.1.2	200	OSPF
11.31.0.0	255.255.0.0	on-link	11.31.1.1	300	LOCAL
11.17.0.0	255.255.0.0	on-link	11.17.1.1	300	LOCAL
11.7.0.0	255.255.0.0	on-link	11.7.1.1	300	LOCAL
11.6.0.0	255.255.0.0	on-link	11.6.1.2	300	LOCAL
11.1.1.1	255.255.255.255	11.7.1.2	11.7.1.1	300	OSPF
11.18.1.2	255.255.255.255	11.7.1.2	11.7.1.1	300	OSPF
11.2.1.1	255.255.255.255	11.7.1.2	11.7.1.1	300	OSPF
11.19.1.2	255.255.255.255	11.7.1.2	11.7.1.1	300	OSPF
11.10.1.2	255.255.255.255	11.17.1.2	11.17.1.1	300	OSPF
11.26.1.1	255.255.255.255	11.17.1.2	11.17.1.1	300	OSPF
11.13.1.2	255.255.255.255	11.17.1.2	11.17.1.1	300	OSPF
11.14.1.1	255.255.255.255	11.17.1.2	11.17.1.1	300	NSPF
11.24.1.1	255.255.255.255	11.6.1.1	11.6.1.2	300	OSPF
11.25.1.2	255.255.255.255	11.6.1.1	11.6.1.2	300	OSPF

c)

RT_6_Table

d) RT_6

Network Destination	Netmask/Prefix len	Gateway	Interface	Metrics	Type
224.0.0.1	255.255.255.255	on-link	11.6.1.2 1...	306	MUL...
224.0.0.0	240.0.0.0	on-link	11.6.1.2 1...	306	MUL...
11.11.1.1	255.255.255.255	11.17.1.2	11.17.1.1	400	OSPF
11.12.1.1	255.255.255.255	11.17.1.2	11.17.1.1	400	OSPF
11.13.1.1	255.255.255.255	11.17.1.2	11.17.1.1	400	OSPF
11.18.1.1	255.255.255.255	11.7.1.2	11.7.1.1	400	OSPF
11.19.1.1	255.255.255.255	11.7.1.2	11.7.1.1	400	OSPF
11.26.1.2	255.255.255.255	11.17.1.2	11.17.1.1	400	OSPF
11.11.1.2	255.255.255.255	11.17.1.2	11.17.1.1	500	OSPF
11.28.1.1	255.255.255.255	11.17.1.2	11.17.1.1	500	OSPF
11.12.1.2	255.255.255.255	11.17.1.2	11.17.1.1	500	OSPF
11.27.1.1	255.255.255.255	11.17.1.2	11.17.1.1	500	OSPF
11.27.1.2	255.255.255.255	11.17.1.2	11.17.1.1	600	OSPF
11.28.1.2	255.255.255.255	11.17.1.2	11.17.1.1	600	OSPF
255.255.255.255	255.255.255.255	on-link	11.31.1.1	999	BRO...

d)

Figure 3-58: IP Forwarding Tables in Result Dashboard for RT 6

In this network, Router6 has 3 interfaces with IP's 11.7.1.1, 11.6.1.2 and 11.17.1.1 and its network addresses are 11.7.0.0, 11.6.0.0 and 11.17.0.0 since its network mask is 255.255.0.0

From the above screenshot, the router forwards packets intended to the subnet:

- 11.1.1.2, 11.2.1.2, 11.3.1.2, 11.3.1.1, 11.4.1.1 via interface 11.7.1.1 with cost 7 (6+1)
- Similarly 11.23.1.1, 11.4.1.2, 11.1.1.1, 11.2.1.1, 11.23.1.2 via interface 11.7.1.1 with cost 8 (6+1+1)
- 11.15.1.1, 11.9.1.2, 11.10.1.1, 11.15.1.2 via interface 11.17.1.1 with cost 8 (7+1)
- 11.9.1.1, 11.10.1.2 via interface 11.17.1.1 with cost 9 (7+1+1)
- 11.29.1.2, 11.29.1.1 and 11.14.1.2 via interface 11.17.1.1 with cost 10 (7+3)
- 11.24.1.2, 11.24.1.1 via interface 11.17.1.1 with cost 10 (7+1+2)

- 11.18.1.2, 11.18.1.1, 11.19.1.2 and 11.19.1.1 via interface 11.7.1.1 with cost 10 (6+1+3)
- 11.13.1.2, 11.11.1.1, 11.12.1.1, and 11.13.1.1 via interface 11.17.1.1 with cost 11 (7+3+1)
- 11.8.1.1 via interface 11.6.1.2 with cost 12 (6+6)
- 11.11.1.2 and 11.12.1.2 via interface 11.17.1.1 with cost 12 (7+3+1+1)
- 11.17.1.2 via interface 11.17.1.1 with cost 12 (7+5)
- 11.26.1.1 and 11.26.1.2 via interface 11.17.1.1 with cost 12 (7+1+4)
- 11.14.1.1 via interface 11.17.1.1 with cost 12 (7+3+2)
- 11.6.1.1 via interface 11.6.1.2 with cost 13 (7+6)
- 11.27.1.1 and 11.27.1.2 via interface 11.17.1.1 with cost 13 (7+3+1+2)
- 11.7.1.2 via interface 11.7.1.1 with cost 14 (8+6)
- 11.5.1.2 via interface 11.6.1.2 with cost 14 (6+8)
- 11.20.1.2, 11.20.1.1, 11.21.1.1, 11.21.1.2, 11.22.1.1, 11.22.1.2 via interface 11.6.1.2 with cost 14 (8+6)
- 11.28.1.2 via interface 11.17.1.1 with cost 14 (7+1+6)
- 11.28.1.1 via interface 11.17.1.1 with cost 14 (7+3+1+3)
- 11.25.1.1, 11.25.1.2 via interface 11.17.1.1 with cost 17 (7+1+9)
- 11.5.1.1 via interface 11.7.1.1 with cost 15 (6+1+8)

We are thus able to simulate the exact example as provided in the RFC and report that SPF Tree obtained, and the routing costs match the analysis provided in the RFC.

4 Transmission control protocol (TCP)

4.1 Introduction to TCP connection management (Level 1)

4.1.1 Introduction

When an application process in a client host seeks a reliable data connection with a process in another host (say, server), the client-side TCP then proceeds to establish a TCP connection with the TCP at the server side. A TCP connection is a point-to-point, full-duplex logical connection with resources allocated only in the end hosts. The TCP connection between the client and the server is established in the following manner and is illustrated in Figure 4-1.

1. The TCP at the client side first sends a special TCP segment, called the SYN packet, to the TCP at the server side.
2. Upon receiving the SYN packet, the server allocates TCP buffer and variables to the connection. Also, the server sends a connection-granted segment, called the SYN-ACK packet, to the TCP at the client side.
3. Upon receiving the SYN-ACK segment, the client also allocates buffers and variables to the connection. The client then acknowledges the server's connection granted segment with an ACK of its own.

This connection establishment procedure is often referred to as the three-way handshake. The special TCP segments can be identified by the values in the fields SYN, ACK and FIN in the TCP header (see Figure 4-2). We also note that the TCP connection is uniquely identified by the source and destination port numbers (see Figure 4-2) exchanged during TCP connection establishment and the source and destination IP addresses.

Once a TCP connection is established, the application processes can send data to each other. The TCP connection can be terminated by either of the two processes. Suppose that the client application process seeks to terminate the connection. Then, the following handshake ensures that the TCP connection is torn down.

1. The TCP at the client side sends a special TCP segment, called the FIN packet, to the TCP at the server side.
2. When the server receives the FIN segment, it sends the client an acknowledgement segment in return and its own FIN segment to terminate the full-duplex connection.
3. Finally, the client acknowledges the FIN-ACK segment (from the server) with an ACK of its own. At this point, all the resources (i.e., buffers and variables) in the two hosts are deallocated.

During the life of a TCP connection, the TCP protocol running in each host makes transitions through various TCP states. Figure 4-1 illustrates the typical TCP states visited by the client and the server during connection establishment and data communication.

TCP is defined in RFCs 793, 1122, 7323 and, 2018. A recommended textbook reference for TCP is Chapter 3: Transport layer, of Computer Networking: A top-down approach, by James Kurose and Keith Ross (Pearson).
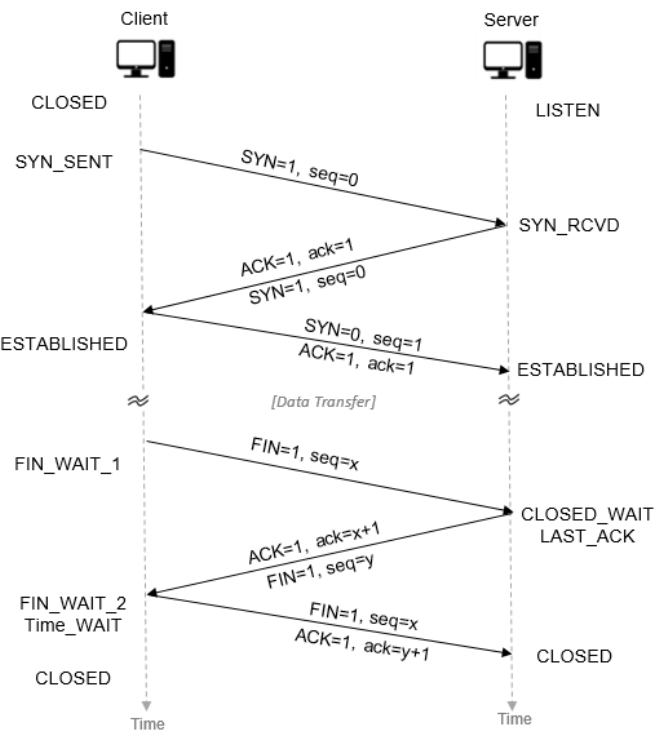
Figure 4-1: TCP connection establishment between a client and a server
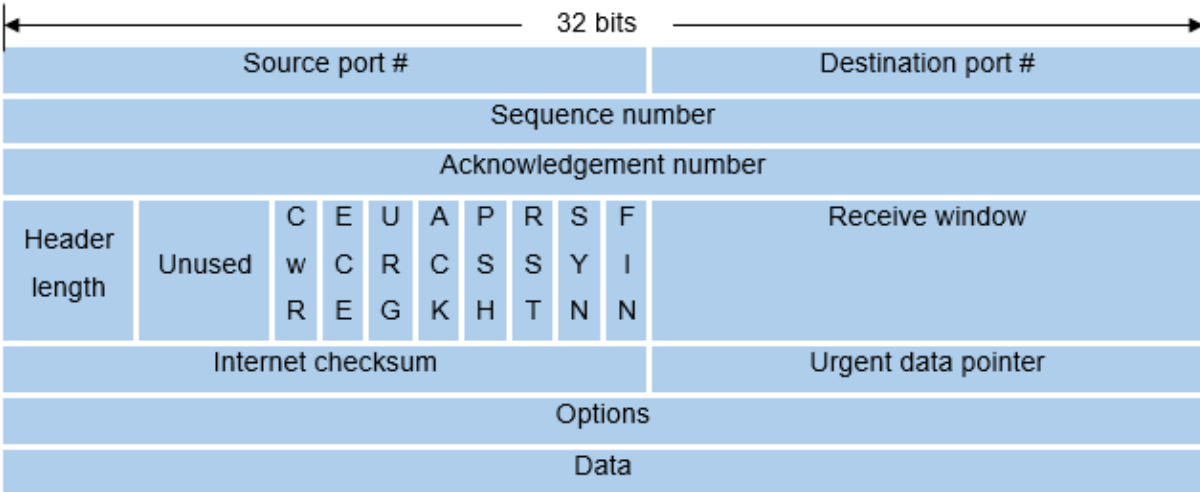
Figure 4-2: TCP Header

4.1.2 Network Setup

Open NetSim and click on **Experiments >Internetworks> TCP> Introduction to TCP connection management** then click on the tile in the middle panel to load the example as shown in below Figure 4-3.
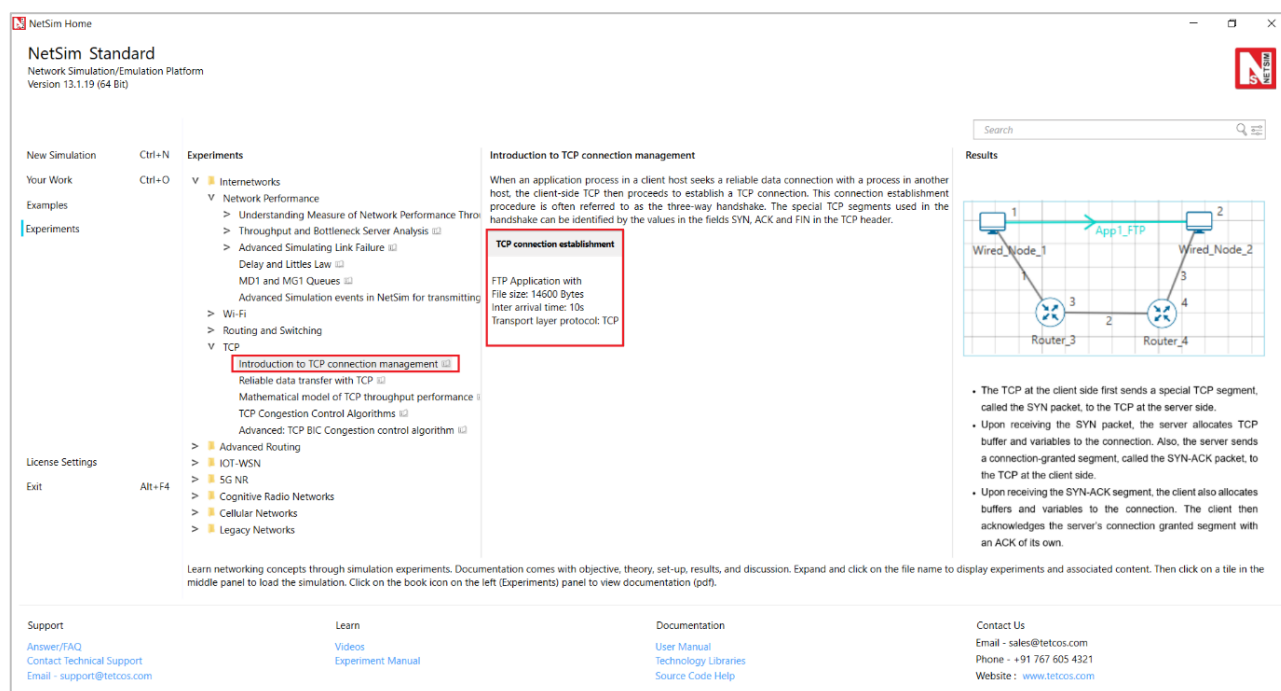
Figure 4-3: List of scenarios for the example of Introduction to TCP connection management

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 4-4
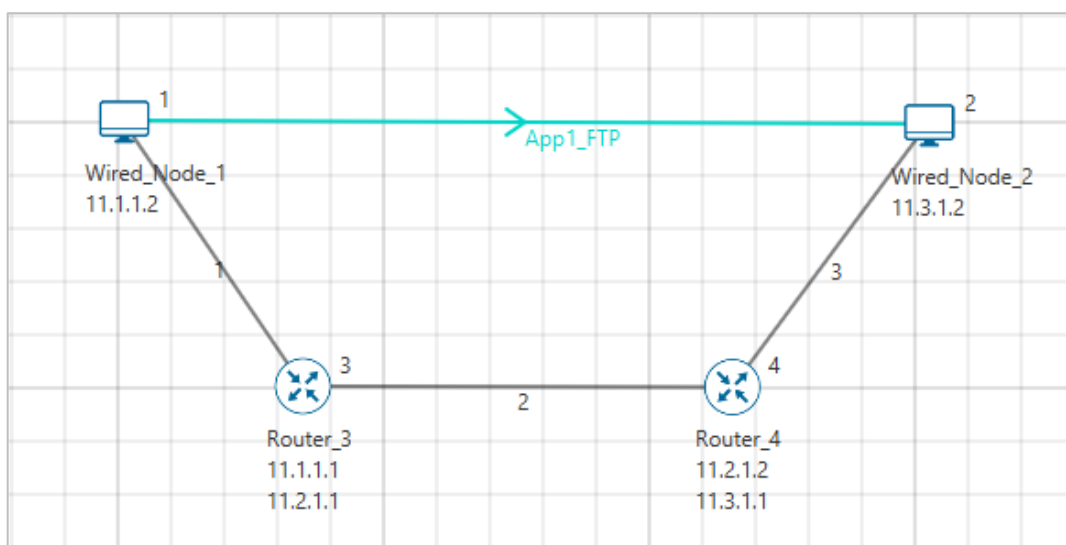
Figure 4-4: Network set up for studying the Introduction to TCP connection management

4.1.3 Procedure

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 2 Routers in the **"Internetworks"** Network Library.

Step 2: In the General Properties of Wired Node 1 i.e., Source, Wireshark Capture is set to Online. Transport Layer properties Congestion plot is set to true.

Note: *Accept default properties for Routers as well as the Links.*

Step 3: Right-click the link ID (of a wired link) and select Properties to access the link's properties. Set Max Uplink Speed and Max Downlink Speed to **10** Mbps. Set Uplink BER and Downlink BER to **0**. Set Uplink Propagation Delay and Downlink Propagation Delay as **100** microseconds for the links 1 and 3 (between the Wired Node's and the routers). Set Uplink Propagation Delay and Downlink Propagation Delay as **50000** microseconds for the backbone link connecting the routers, i.e., 2.

Step 4: Right click on the Application Flow **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

An FTP Application is generated from Wired Node 1 i.e., Source to Wired Node 2 i.e., Destination with File Size set to 14600 Bytes and File Inter Arrival Time set to 10 Seconds.

Step 5: Click on Display Settings > Device IP check box in the NetSim GUI to view the network topology along with the IP address.

Step 6: Enable the plots and click on Run simulation. The simulation time is set to 10 seconds.

4.1.4 Output

We have enabled Wireshark capture in Wired Node 1. The PCAP file is generated at the end of the simulation and is shown in Figure 4-5.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	0.0.0.0	IPv4	20	
2	0.000000	11.1.1.2	11.1.1.1	TCP	44	82 → 36934 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
3	0.100335	11.3.1.2	11.1.1.2	TCP	44	36934 → 82 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
4	0.100335	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [ACK] Seq=1 Ack=1 Win=4380 Len=0
5	0.100335	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=1 Win=4380 Len=1460
6	0.100335	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=1461 Win=4380 Len=1460
7	0.100335	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=2921 Win=4380 Len=1460
8	0.204208	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=1461 Win=4381 Len=0
9	0.204208	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=4381 Win=5840 Len=1460
10	0.204208	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=5841 Win=5840 Len=1460
11	0.205430	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=2921 Win=4381 Len=0
12	0.205430	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=7301 Win=7300 Len=1460
13	0.205430	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=8761 Win=7300 Len=1460
14	0.206651	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=4381 Win=4381 Len=0
15	0.206651	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=10221 Win=8760 Len=1460
16	0.206651	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=11681 Win=8760 Len=1460
17	0.308027	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=5841 Win=4381 Len=0
18	0.308027	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=13141 Win=10220 Len=1460
19	0.309249	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=7301 Win=4381 Len=0
20	0.310471	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=8761 Win=4381 Len=0
21	0.311692	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=10221 Win=4381 Len=0
22	0.312914	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=11681 Win=4381 Len=0
23	0.314136	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=13141 Win=4381 Len=0
24	0.411846	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=14601 Win=4381 Len=0
25	0.411846	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [FIN] Seq=14601 Win=18980 Len=0
26	0.512161	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [FIN, ACK] Seq=1 Ack=14601 Win=4381 Len=0
27	0.512215	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [FIN] Seq=2 Win=4381 Len=0
28	0.512215	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [ACK] Seq=14602 Ack=3 Win=18980 Len=0

Figure 4-5: Wireshark Packet capture at Wired_Node_1

1. The 3-way handshake of TCP connection establishment and TCP connection termination is observed in the packet capture (Figure 4-5).

2. Data is transferred only after the TCP connection is established.

3. We can access the packet header details of the TCP segments (SYN, SYN-ACK, FIN, FINACK) in Wireshark.

4.2 Reliable data transfer with TCP (Level 1)

4.2.1 Introduction

TCP provides reliable data transfer service to the application processes even when the underlying network service (IP service) is unreliable (loses, corrupts, garbles or duplicates packets). TCP uses checksum, sequence numbers, acknowledgements, timers and retransmission to ensure correct and in order delivery of data to the application processes.

TCP views the data stream from the client application process as an ordered stream of bytes. TCP will grab chunks of this data (stored temporarily in the TCP send buffer), add its own header and pass it on to the network layer. A key field of the TCP header is the sequence number which indicates the position of the first byte of the TCP data segment in the data stream. The sequence number will allow the TCP receiver to identify segment losses, duplicate packets and to ensure correct delivery of the data stream to the server application process.

When a server receives a TCP segment, it acknowledges the same with an ACK segment (the segment carrying the acknowledgement has the ACK bit set to 1) and also conveys the sequence number of the first missing byte in the application data stream, in the acknowledgement number field of the TCP header. All acknowledgements are cumulative; hence, all missing, and out-of-order TCP segments will result in duplicate acknowledgements for the corresponding TCP segments.

TCP sender relies on sequence numbering and acknowledgements to ensure reliable transfer of the data stream. In the event of a timeout (no acknowledgement is received before the timer expires) or triple duplicate acknowledgements (multiple ACK segments indicate a lost or missing TCP segment) for a TCP segment, the TCP sender will retransmit the segment until the TCP segment is acknowledged (at least cumulatively). In Figure 4-6, we illustrate retransmission by the TCP sender after a timeout for acknowledgement.
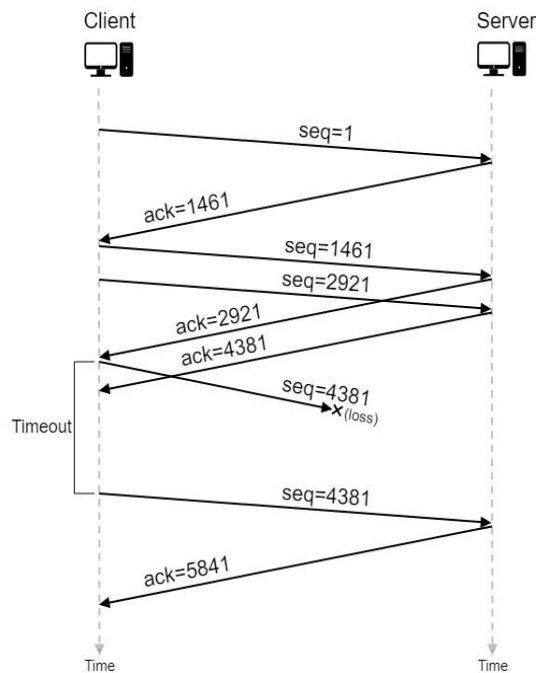
Figure 4-6: An illustration of TCP retransmission with timeout. The segment with sequence number 4381 is lost in the network. The TCP client retransmits the segment after a timeout event.

4.2.2 Network Setup

Open NetSim and click on **Experiments > Internetworks> TCP> Reliable data transfer with TCP** then click on the tile in the middle panel to load the example as shown in below Figure 4-7.

Figure 4-7: List of scenarios for the example of Reliable data transfer with TCP

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 4-8.

Figure 4-8: Network set up for studying the Reliable data transfer with TCP
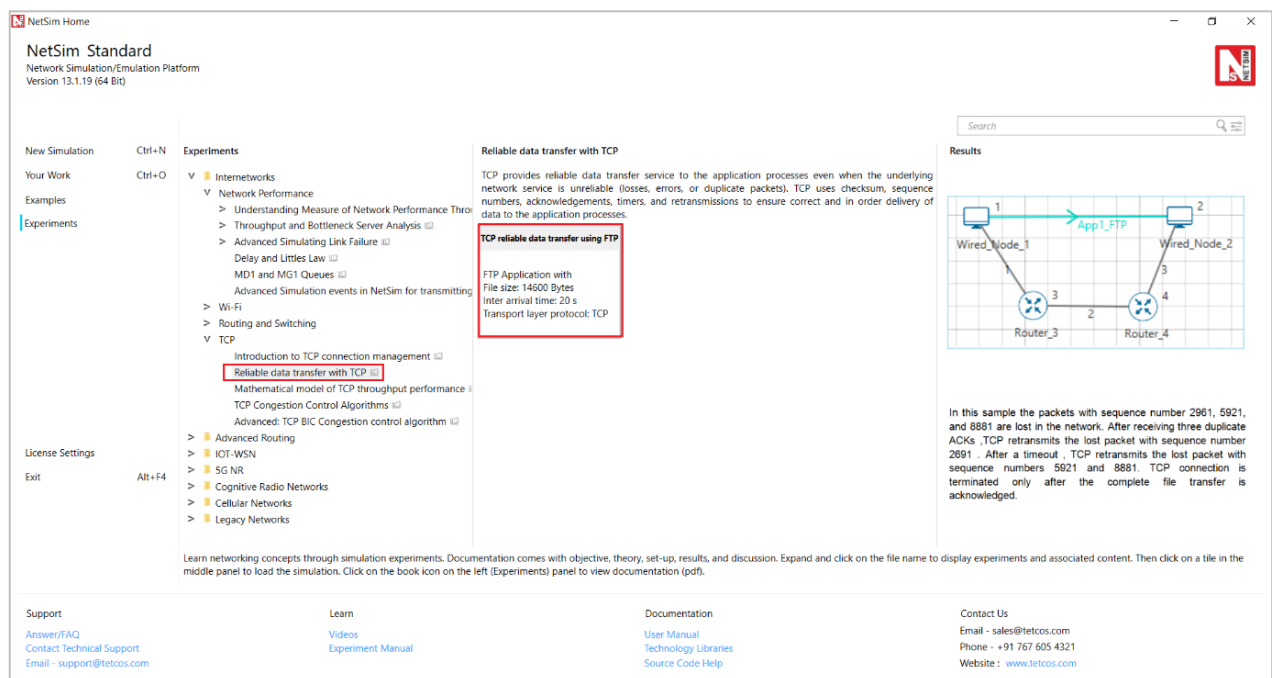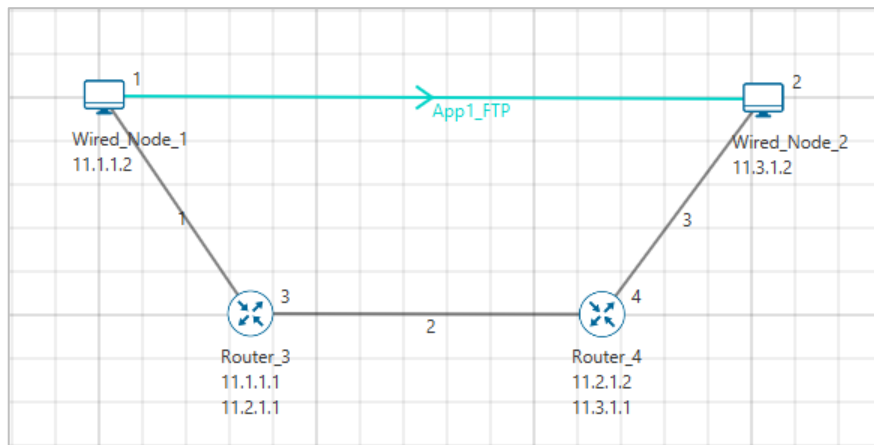
We will seek a simple file transfer with TCP over a lossy link to study reliable data transfer with TCP. We will simulate the network setup illustrated in Figure 4-8 with the configuration parameters listed in detail to study reliable data transfer with TCP connection_

4.2.3 Procedure

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 2 Routers in the **"Internetworks"** Network Library.

Step 2: In the General Properties of Wired Node 1 i.e., Source and Wired Node 2 i.e., Destination, Wireshark Capture is set to Online. In Transport Layer ->**Congestion plot enabled** is set to **True** for **Wired_Node_1** and **False** for **Wired Node_2**.

Transport Layer properties Congestion plot is set to true for Wired_Node_1.

Note: Accept default properties for Routers as well as the Links.

Step 3: Right-click on the link ID (of a wired link) and select Properties to access the link's properties. Set Max Uplink Speed and Max Downlink Speed to **10** Mbps. Set Uplink BER and Downlink BER to **0**. Set Uplink Propagation Delay and Downlink Propagation Delay as **100** microseconds for the links 1 and 3 (between the Wired Node's and the routers). Set Uplink Propagation Delay and Downlink Propagation Delay as **50000** microseconds and Uplink BER and Downlink BER to **0.00001** for the backbone link connecting the routers, i.e., 2.

Step 4: Right click on the Application Flow **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

An FTP Application is generated from Wired_Node_1 i.e., Source to Wired_Node_2 i.e., Destination with File Size set to 14600 Bytes and File Inter Arrival Time set to 20 Seconds.

Step 5: Click on Display Settings > Device IP check box in the NetSim GUI to view the network topology along with the IP address.

Step 6: Enable the plots and click on Run simulation. The simulation time is set to 20 seconds.

4.2.4 Output

We aimed to transfer a file of size 14600 bytes (i.e., 10 packets, each of size 1460 bytes) with TCP over a lossy link. In Figure 4-9, we report the application metrics data for FTP which indicates that the complete file was transferred.

Throughput Plot	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (microsec)	Jitter (microsec)
_Application_Throughput_plot_	App1_FTP	10	10	0.005840	2503061.760000	380974.542222

Figure 4-9: Application Metrics table for FTP

We have enabled Wireshark Capture in Wired_Node 1 and Wired Node_2. The PCAP files are generated at the end of the simulation and are shown in Figure 4-10 and Figure 4-11.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	0.0.0.0	IPv4	20	
2	0.000000	11.1.1.2	11.1.1.1	TCP	44	82 → 36934 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
3	0.100695	11.3.1.2	11.1.1.2	TCP	44	36934 → 82 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
4	0.100695	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [ACK] Seq=1 Ack=1 Win=4380 Len=0
5	0.100695	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=1 Win=4380 Len=1460
6	0.100695	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=1461 Win=4380 Len=1460
7	0.100695	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=2921 Win=4380 Len=1460
8	0.204928	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=1461 Win=4381 Len=0
9	0.204928	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=4381 Win=5840 Len=1460
10	0.204928	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=5841 Win=5840 Len=1460
11	0.309107	11.3.1.2	11.1.1.2	TCP	40	[TCP Dup ACK 8#1] 36934 → 82 [ACK] Seq=1 Ack=1461 Win=4381 Len=0
12	0.809161	11.1.1.2	11.1.1.1	TCP	1500	[TCP Retransmission] 82 → 36934 [<None>] Seq=1461 Win=4380 Len=1460
13	0.913340	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=2921 Win=4381 Len=0
14	1.517566	11.1.1.2	11.1.1.1	TCP	1500	[TCP Retransmission] 82 → 36934 [<None>] Seq=2921 Win=4380 Len=1460
15	1.621745	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=5841 Win=4381 Len=0
16	1.621745	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=7301 Win=2920 Len=1460
17	1.725925	11.3.1.2	11.1.1.2	TCP	40	[TCP Dup ACK 15#1] 36934 → 82 [ACK] Seq=1 Ack=5841 Win=4381 Len=0
18	3.319939	11.1.1.2	11.1.1.1	TCP	1500	[TCP Retransmission] 82 → 36934 [<None>] Seq=5841 Win=5840 Len=1460
19	3.424119	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=8761 Win=4381 Len=0
20	3.424119	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=8761 Win=2920 Len=1460
21	3.424119	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=10221 Win=2920 Len=1460
22	3.528298	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=10221 Win=4381 Len=0
23	3.528298	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=11681 Win=4380 Len=1460
24	3.528298	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=13141 Win=4380 Len=1460
25	3.529519	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=11681 Win=4381 Len=0
26	3.632477	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=13141 Win=4381 Len=0

Figure 4-10: PCAP file at Wired Node 1. TCP ensures reliable data transfer using timeout, duplicate ACKs and retransmissions.

Figure 4-11: PCAP file at Wired Node 2

4.2.5 Inference

1. From Figure 4-10 and Figure 4-11, we note that the packets with sequence number 1461 and 5841 are errored in the network, which can also observe in Packet Trace.

2. After receiving three duplicate ACKs (in lines 13, 14 of Figure 4-10), TCP retransmits the errored packet. (In line 15 of Figure 4-10).

3. TCP connection is terminated only after the complete file transfer is acknowledged which can be observed in Figure 4-5 (Line 25 and 26).

4.3 Mathematical Modelling of TCP Throughput Performance (Level 2)

4.3.1 Introduction

The average throughput performance of additive-increase multiplicative-decrease TCP congestion control algorithms have been studied in a variety of network scenarios. In the regime of large RTT, the average throughput performance of the TCP congestion control algorithms can be approximated by the ratio of the average congestion window *cwnd* and RTT.

4.3.1.1 Loss-less Network

In a loss-less network, we can expect the TCP congestion window *cwnd* to quickly increase to the maximum value of 64 KB (without TCP scaling). In such a case, the long-term average throughput of TCP can be approximated as

$$Throughput \approx \frac{64 \times 1024 \ (bits)}{RTT \ (in \ secs)}$$

4.3.1.2 Lossy Network

We refer to an exercise in Chapter 3 of Computer Networking: A top-down approach, by Kurose and Ross for the setup. Consider a TCP connection over a lossy link with packet error rate p. In a period of time between two packet losses, the congestion window may be approximated to increase from an average value of W/2 to W (see **Figure 4-19** for motivation). In such a scenario, the throughput can be approximated to vary from W/2/RTT to W/RTT (in the cycle between two packet losses). Under such assumptions, we can then show that the loss rate (fraction of packets lost) must be equal to

$$p = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

and the average throughput is then approximately,

$$Throughput \approx \sqrt{\frac{3}{2p}} \times \frac{MSS\ (in\ bits)}{RTT\ (in\ secs)}$$

4.3.2 Network Setup

Open NetSim and click on **Experiments> Internetworks> TCP> Mathematical model of TCP throughput performance** then click on the tile in the middle panel to load the example as shown below in Figure 4-12.
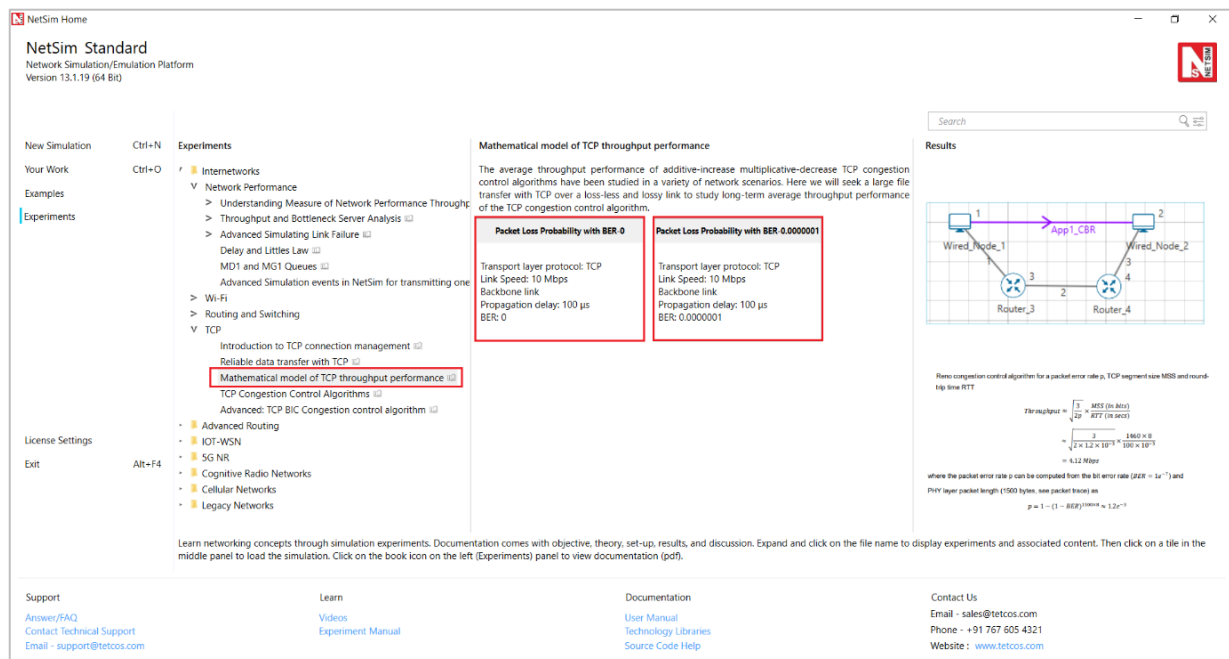

Figure 4-12: List of scenarios for the example of Mathematical model of TCP throughput performance

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 4-13.
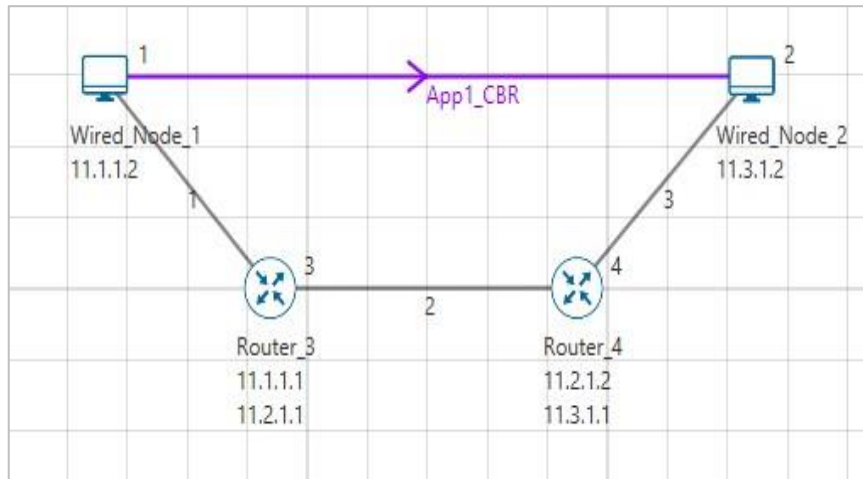
Figure 4-13: Network set up for studying the Mathematical model of TCP throughput performance

We will seek a large file transfer with TCP over a loss-less and lossy link to study long-term average throughput performance of the TCP congestion control algorithm. We will simulate the network setup illustrated in Figure 4-13 with the two (loss-less and lossy) configuration parameters listed in detail to study the throughput performance of TCP New Reno.

4.3.3 Procedure

Packet Loss Probability with BER-0 Sample

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 2 Routers in the **"Internetworks"** Network Library.

Step 2: In the General Properties of Wired Node 1 i.e., Source, Wireshark Capture is set to Online. Transport Layer properties Congestion plot enabled is set to true.

Note: Accept default properties for Routers.

Step 3: Right-click the link ID (of a wired link) and select Properties to access the link's properties. Set Max Uplink Speed and Max Downlink Speed to **10** Mbps. Set Uplink BER and Downlink BER to **0**. Set Uplink Propagation Delay and Downlink Propagation Delay as **100** microseconds for the links 1 and 3 (between the Wired Node's and the routers). Set Uplink Propagation Delay and Downlink Propagation Delay as **50000** microseconds for the backbone link connecting the routers, i.e., 2.

Step 4: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

An CBR Application is generated from Wired Node 1 i.e., Source to Wired Node 2 i.e., Destination with Packet Size set to 1460 Bytes and Inter Arrival Time set to 1168 microseconds.

Step 5: Click on Display Settings > Device IP Enable check box in the NetSim GUI to view the network topology along with the IP address.

Step 6: Click on **Plots** icon and select the **Enable Plots** checkbox. This enables us to view the throughput plot of the application **App1 CBR**.

Step 7: Click on Run simulation. The **simulation time** is set to 100 seconds. In the "**Static ARP Configuration**" tab, Static ARP is set to **disable**.

Packet Loss Probability with BER-0.0000001 Sample

Step 1: Right-click the link ID (of a wired link) and select Properties to access the link's properties. Set Max Uplink Speed and Max Downlink Speed to **10** Mbps. Set Uplink BER and Downlink BER to **0**. Set Uplink Propagation Delay and Downlink Propagation Delay as **100** microseconds for the links 1 and 3 (between the Wired Node's and the routers). Set Uplink Propagation Delay and Downlink Propagation Delay as **50000** microseconds and Uplink BER and Downlink BER to **0.0000001** for the backbone link connecting the routers, i.e., 2.

Step 2: Click on Run simulation. The **simulation time** is set to 100 seconds. In the "**Static ARP Configuration**" tab, Static ARP is set to **disable**.

4.3.4 Output

In Figure 4-14, we report the application metrics data for data transfer over a loss-less link (**Packet Loss Probability with BER-0 sample**).

Application_Metrics_Table								
Application_Metrics								Detailed View
Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)	
1	Application_Throughput_plot	App1_CBR	85617	42055	4.912024	25667072.446895	1206.188143	

Figure 4-14: Application Metrics with BER = 0

In Figure 4-15, we report the plot of long-term average throughput of the TCP connection over the loss-less link.
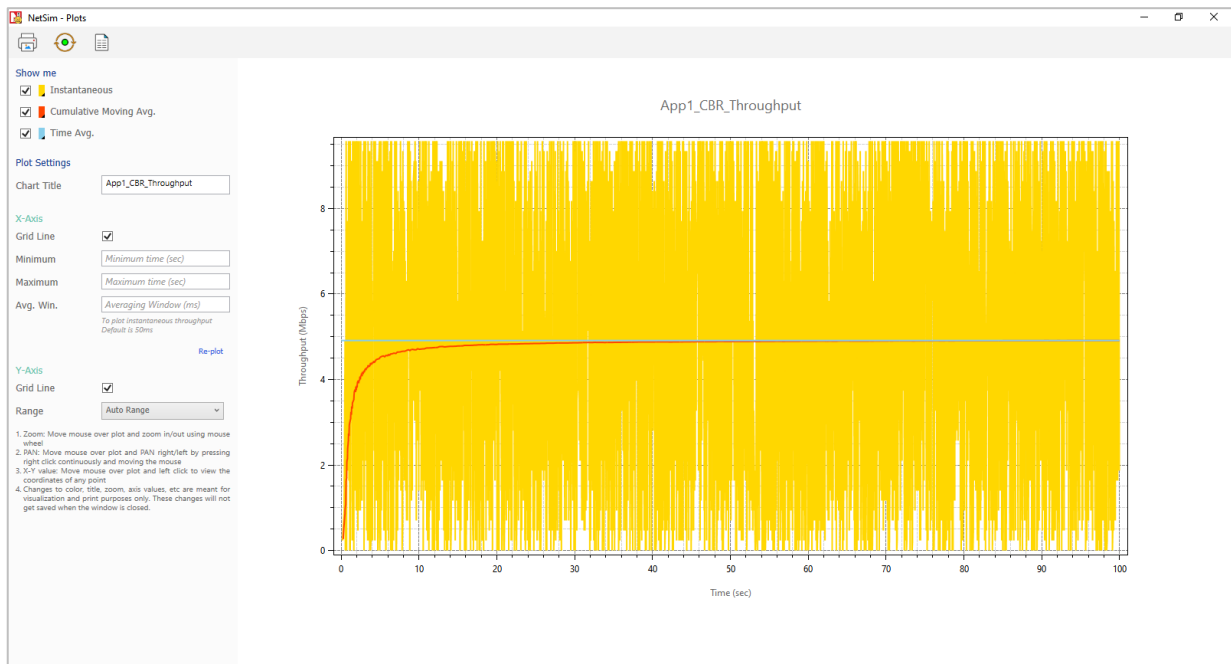
Figure 4-15: Long-term average throughput of TCP New Reno over a loss-less link

We have enabled Wireshark Capture in the Wired Node 1. The PCAP file is generated at the end of the simulation. From the PCAP file, the congestion window evolution graph can be obtained as follows. In Wireshark, select any data packet with a left click, then, go to **Statistics** > **TCP Stream Graphs** > **Window Scaling** >Select **Switch Direction**. In Figure 4-16, we report the congestion window evolution of TCP New Reno over the loss-less link.
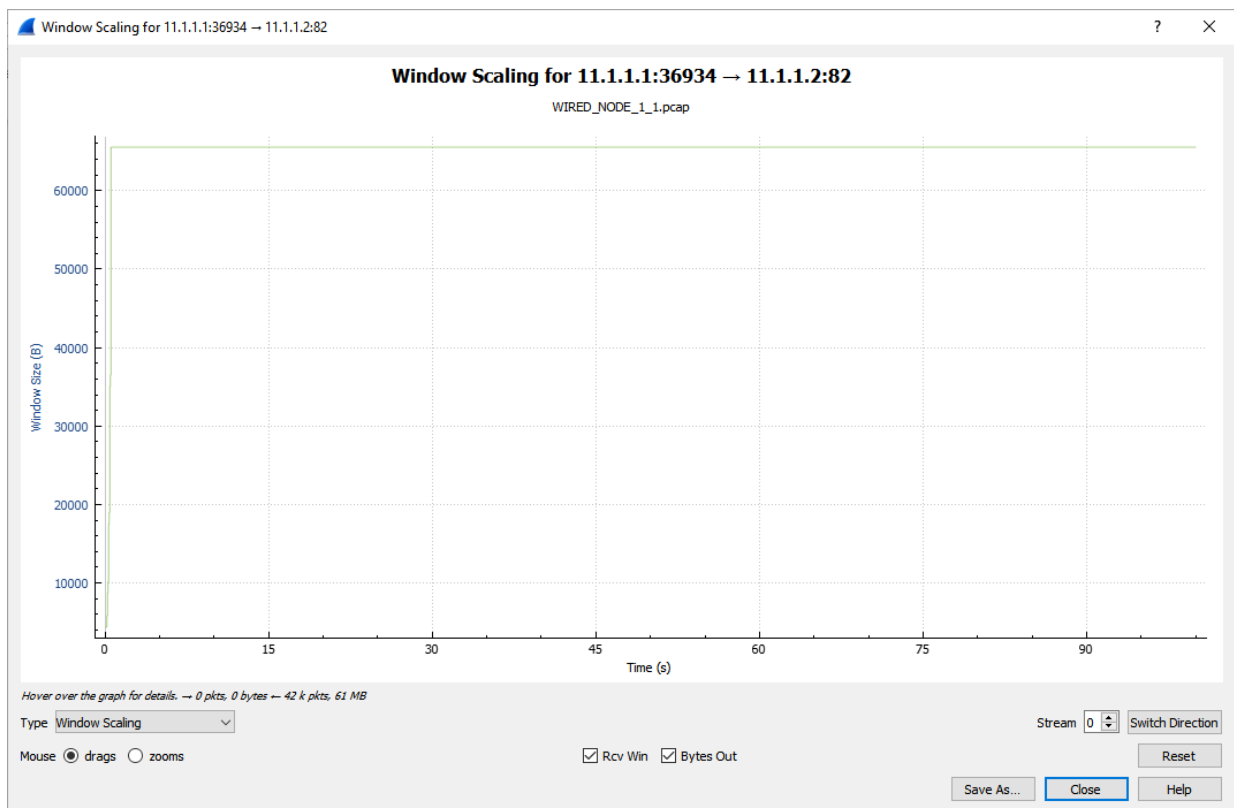
Figure 4-16: Congestion window evolution with TCP New Reno over a loss-less link.

In Figure 4-17, we report the application metrics data for data transfer over a lossy link (**Packet Loss Probability with BER-0.0000001 sample**).

Application_Metrics_Table							□ X
Application_Metrics							☐ Detailed View
Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)
1	Application_Throughput_plot	App1_CBR	85617	32589	3.806395	31884960.015729	1976.200182

Figure 4-17: Application Metrics when $BER = 1 \times 10^{-7}$

In Figure 4-18, we report the plot of long-term average throughput of the TCP connection over the lossy link.
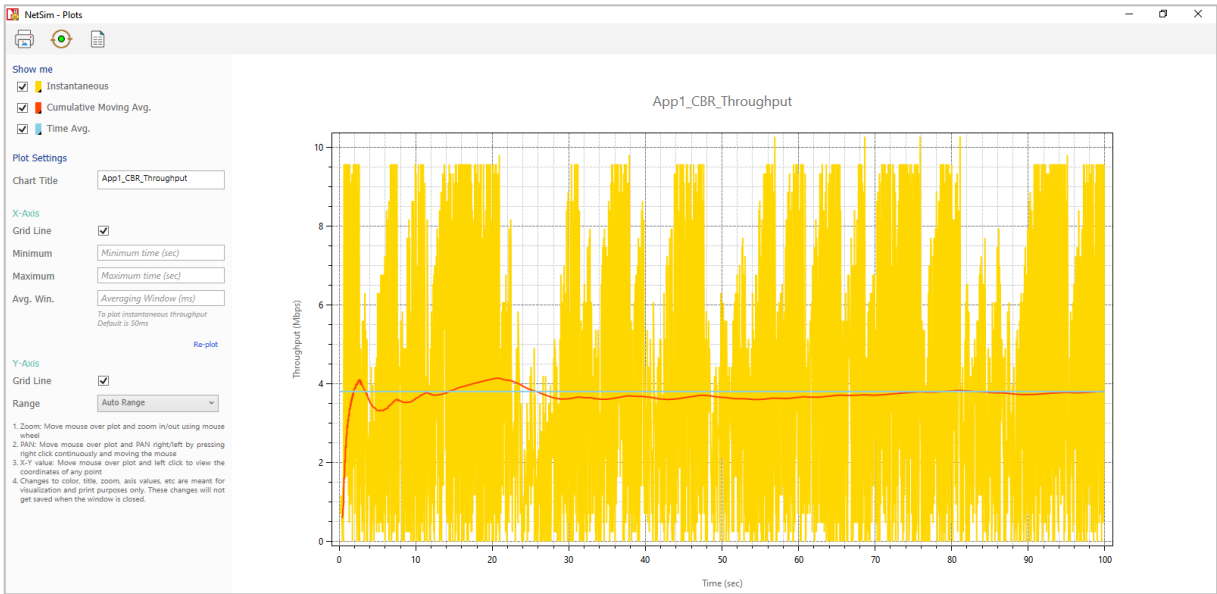
Figure 4-18: Throughput graph

In Figure 4-19, we report the congestion window evolution of TCP New Reno over the lossy link.

Figure 4-19: Congestion window evolution with TCP New Reno over a lossy link

4.3.5 Observations and Inference

1. In Figure 4-16, we notice that the congestion window of TCP (over the loss-less link) increases monotonically to 64 KB and remains there forever. So, a block of 64 KBs of data is transferred over a round-trip time (RTT) of approximately 100 milliseconds. Hence, a good approximation of the TCP throughput over the loss-less link is.

$$Throughput \approx \frac{Window\ Size\ (in\ bits)}{RTT\ (in\ secs)}$$

$$= \frac{65535 \times 8}{100 \times 10^{-3}} = 5.24\ Mbps$$

We note that the observed long-term average throughput (see **Figure 4-14**) is approximately equal to the above computed value.

2. In Figure 4-19, for the lossy link with $BER = 1e^{-7}$, we report the congestion window evolution with New Reno congestion control algorithm. The approximate throughput of the TCP New Reno congestion control algorithm for a packet error rate p, TCP segment size MSS and round-trip time RTT

$$Throughput \approx \sqrt{\frac{3}{2p}} \times \frac{MSS\ (in\ bits)}{RTT\ (in\ secs)}$$

$$\approx \sqrt{\frac{3}{2 \times 1.2 \times 10^{-3}}} \times \frac{1460 \times 8}{100 \times 10^{-3}}$$

$$= 4.12 \, Mbps$$

where the packet error rate p can be computed from the bit error rate ($BER = 1e^{-7}$) and the PHY layer packet length (1500 bytes, see packet trace) as

$$p = 1 - (1 - BER)^{1500 \times 8} \approx 1.2e^{-3}$$

We note that the observed long-term average throughput (see **Figure 4-17**) is approximately equal to the above computed value.

4.4 TCP Congestion Control Algorithms (Level 2)

4.4.1 Introduction

A key component of TCP is end-to-end congestion control algorithm. The TCP congestion control algorithm limits the rate at which the sender sends traffic into the network based on the perceived network congestion. The TCP congestion control algorithm at the sender maintains a variable called congestion window, commonly referred as *cwnd*, that limits the amount of unacknowledged data in the network. The congestion window is adapted based on the network conditions, and this affects the sender's transmission rate. The TCP sender reacts to congestion and other network conditions based on new acknowledgements, duplicate acknowledgements and timeouts. The TCP congestion control algorithms describe the precise manner in which TCP adapts *cwnd* with the different events.

The TCP congestion control algorithm has three major phases (a) slow-start, (b) congestion avoidance, and (c) fast recovery. In slow-start, TCP is aggressive and increases *cwnd* by one MSS with every new acknowledgement. In congestion avoidance, TCP is cautious and increases the *cwnd* by one MSS per round-trip time. Slow-start and congestion avoidance are mandatory components of all TCP congestion control algorithms. In the event of a packet loss (inferred by timeout or triple duplicate acknowledgements), the TCP congestion control algorithm reduces the congestion window to 1 (e.g., Old Tahoe, Tahoe) or by half (e.g., New Reno). In fast recovery, TCP seeks to recover from intermittent packet losses while maintaining a high congestion window. The new versions of TCP, including TCP New Reno, incorporate fast recovery as well. Figure 4-20 presents a simplified view of the TCP New Reno congestion control algorithm highlighting slow-start, congestion avoidance and fast recovery phases.

TCP congestion control algorithm is often referred to as additive-increase multiplicative-decrease (AIMD) form of congestion control. The AIMD congestion control algorithm often leads to a "saw tooth" evolution of the congestion window (with linear increase of the congestion window during bandwidth probing and a multiplicative decrease in the event of packet losses), see Figure 4-25.
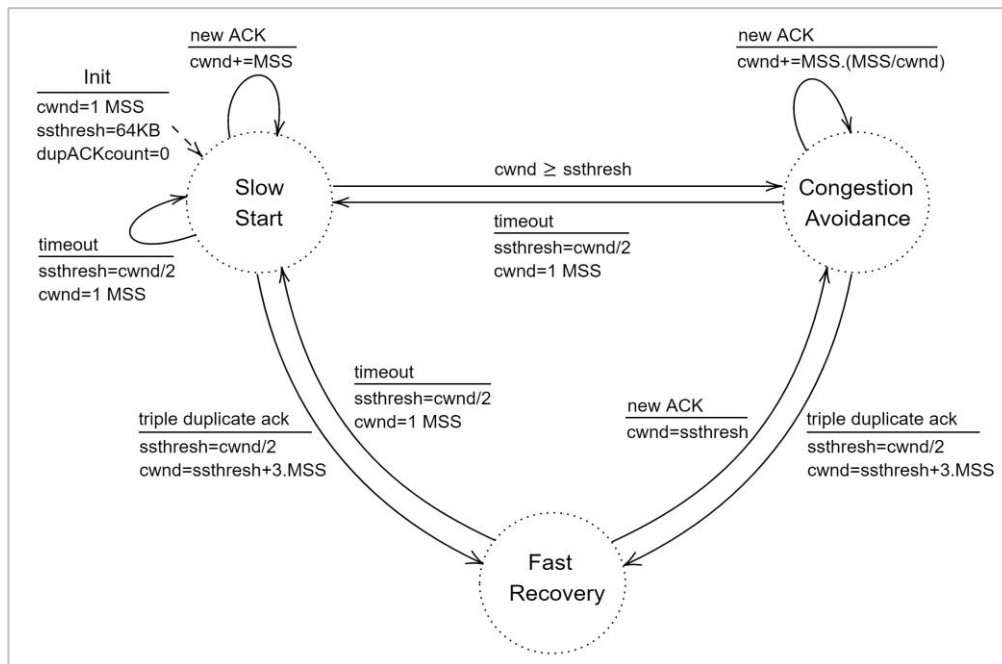
Figure 4-20: A simplified view of FSM of the TCP New Reno congestion control algorithm

4.4.2 Network Setup

We will seek a large file transfer with TCP over a lossy link to study the TCP congestion control algorithms. We will simulate the network setup illustrated in Figure 4-22 with the configuration parameters listed in detail in steps to study the working of TCP congestion control algorithms.

Open NetSim and click on **Experiments> Internetworks>TCP> TCP Congestion Control Algorithms > Old-Tahoe** then click on the tile in the middle panel to load the example as shown in below Figure 4-21.
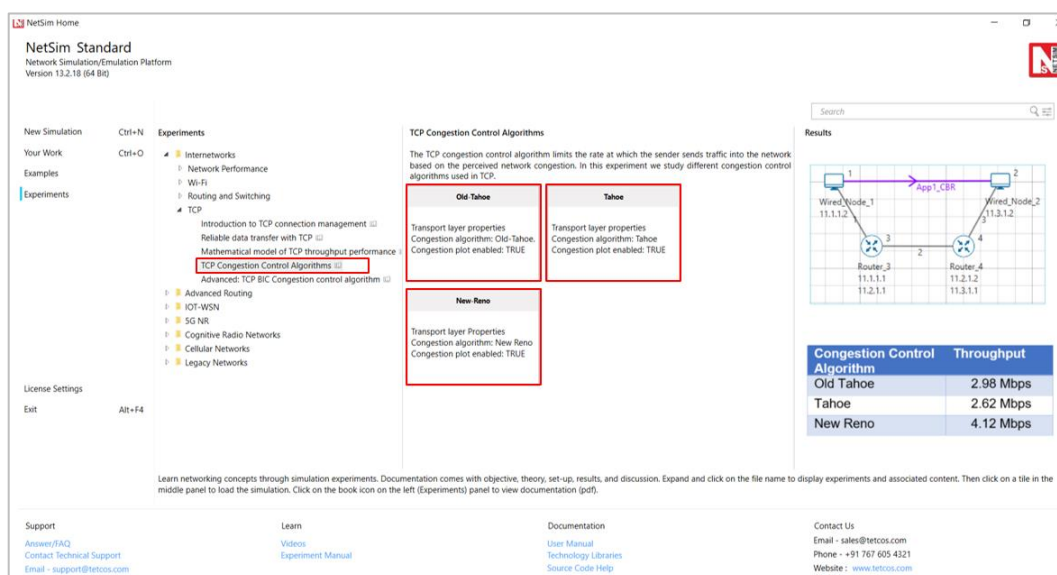
Figure 4-21: List of scenarios for the example of TCP Congestion Control Algorithms

NetSim UI displays the configuration file corresponding to this experiment as shown below:
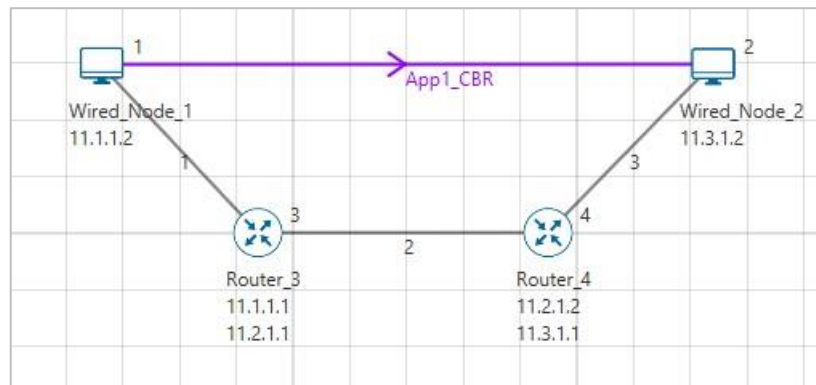
Figure 4-22: Network set up for studying the TCP Congestion Control Algorithms

4.4.3 Procedure

Old -Tahoe

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 2 Routers in the **"Internetworks"** Network Library.

Step 2: In the Source Node, i.e., Wired Node 1, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to **OLD TAHOE**. **Congestion plot enabled** is set to **TRUE**.

Step 3: In the General Properties of Wired Node 1 i.e., Source, Wireshark Capture is set to Online.

Note: Accept default properties for Routers as well as the Links Properties should be changed.

Step 4: Right-click the link ID (of a wired link) and select Properties to access the link's properties. Set Max Uplink Speed and Max Downlink Speed to **10** Mbps. Set Uplink BER and Downlink BER to **0**. Set Uplink Propagation Delay and Downlink Propagation Delay as **100** microseconds for the links 1 and 3 (between the Wired Node's and the routers). Set Uplink Propagation Delay and Downlink Propagation Delay as **50000** microseconds and Uplink BER and Downlink BER to **0.0000001** for the backbone link connecting the routers, i.e., 2.

Step 5: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

An CBR Application is generated from Wired Node 1 i.e., Source to Wired Node 2 i.e., Destination with Packet Size set to 1460 Bytes and File Inter Arrival Time set to 1168 microseconds.

Step 6: Click on Display Settings > Device IP check box in the NetSim GUI to view the network topology along with the IP address.

Step 7: Click on **Plots** icon and select the **Enable Plots** checkbox. This enables us to view the throughput plot of the application **App1 CBR**.

Step 8: Click on Run simulation. The **simulation time** is set to 20 seconds. In the "**Static ARP Configuration**" tab, Static ARP is set to **disable**.

Tahoe

Step 1: In the Source Node, i.e., Wired Node 1, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to **TAHOE**. **Congestion plot enabled** is set to **TRUE**.

Step 2: Click on Run simulation. The **simulation time** is set to 20 seconds. In the "**Static ARP Configuration**" tab, Static ARP is set to **disable**.

New Reno

Step 1: In the Source Node, i.e., Wired Node 1, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to **NEW RENO**. **Congestion plot enabled** is set to **TRUE**.

Step 2: Click on Run simulation. The **simulation time** is set to 20 seconds. In the "**Static ARP Configuration**" tab, Static ARP is set to **disable**.

4.4.4 Output

We have enabled Wireshark Capture in the Wired Node 1. The PCAP file is generated at the end of the simulation. From the PCAP file, the congestion window evolution graph can be obtained as follows. In Wireshark, select any data packet with a left click, then, go to **Statistics** > **TCP Stream Graphs** > **Window Scaling >** Select **Switch Direction**.

The congestion window evolution for Old Tahoe, Tahoe and New Reno congestion control algorithms are presented in Figure 4-23, Figure 4-24, and Figure 4-25, respectively.

Table 4-1 shows the throughput values of different congestion control algorithms (obtained from the Application Metrics).

Figure 4-23: Congestion window evolution with TCP Old Tahoe. We note that Old Tahoe infers packet loss only with timeouts, and updates the slow-start threshold ssthresh and congestion window cwnd as ssthresh = cwnd/2 and cwnd = 1
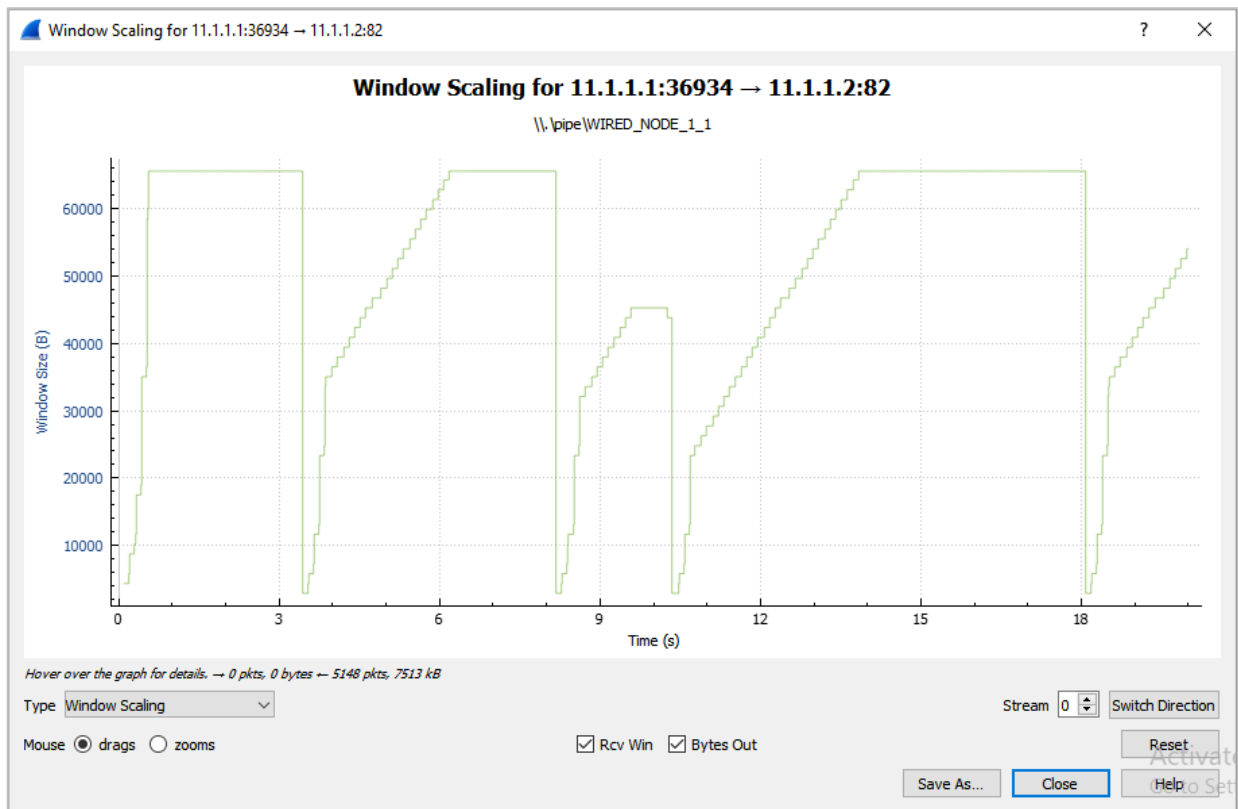

Figure 4-24: Congestion window evolution with TCP Tahoe. We note that Tahoe infers packet loss with timeout and triple duplicate acknowledgements, and updates the slow-start threshold ssthresh and congestion window cwnd as ssthresh = cwnd/2 and cwnd = 1
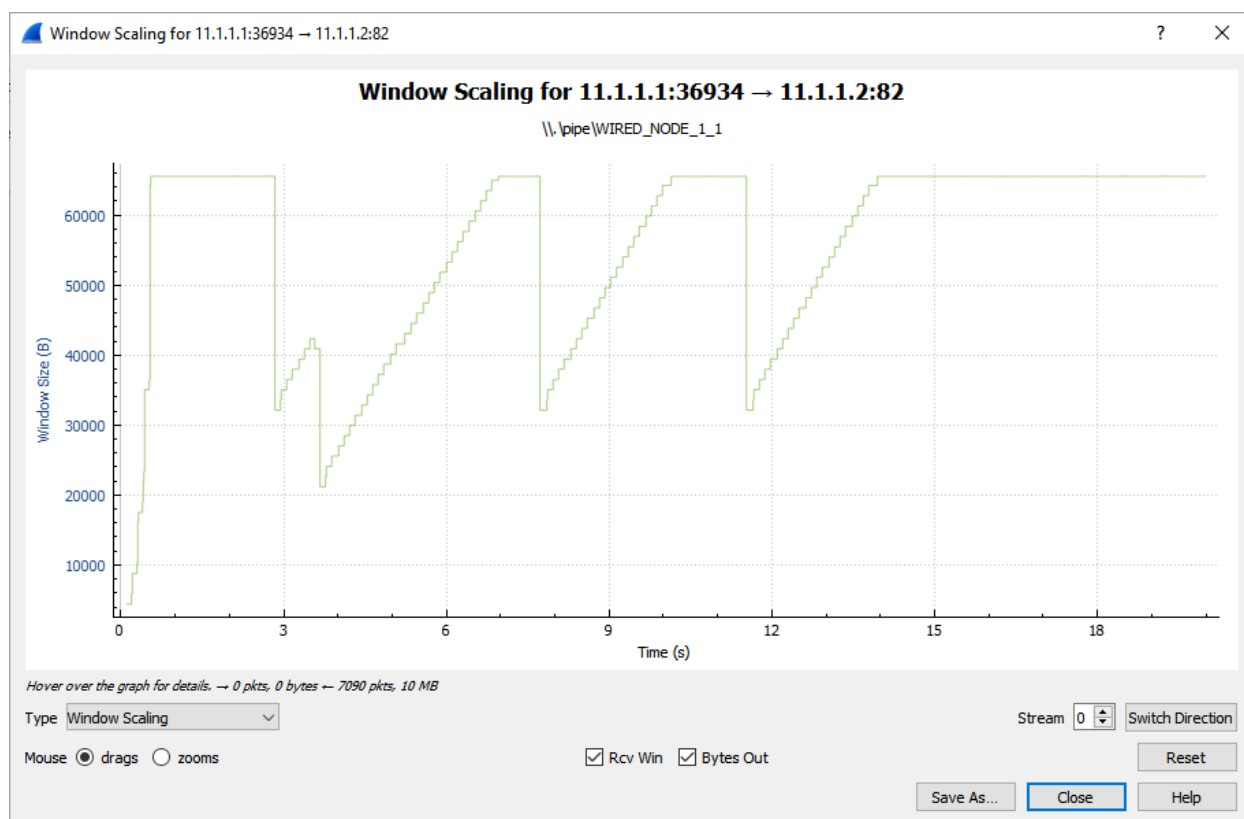
Figure 4-25: Congestion window evolution with TCP New Reno. We note that New Reno infers packet loss with timeout and triple duplicate acknowledgements and updates the slow-start threshold ssthresh and congestion window cwnd as ssthresh = cwnd/2 and cwnd = ssthresh + 3MSS (in the event of triple duplicate acknowledgements).

Congestion Control Algorithm	Throughput
Old Tahoe	2.98 Mbps
Tahoe	2.62 Mbps
New Reno	4.12 Mbps

Table 4-1: Long-term average throughput of the different TCP congestion control algorithms

4.4.5 Observations and Inference

1. We can observe slow start, congestion avoidance, timeout, fast retransmit and recovery phases in the Figure 4-23, Figure 4-24, and Figure 4-25. In Figure 4-23, we note that Old Tahoe employs timeout, slow-start and congestion avoidance for congestion control. In Figure 4-24, we note that Tahoe employs fast retransmit, slow-start and congestion avoidance for congestion control. In Figure 4-25, we note that New Reno employs fast retransmit and recovery, congestion avoidance and slow-start for congestion control.

2. We note that TCP New Reno reports a higher long term average throughput (in comparison with Old Tahoe and Tahoe, see Table 4-1) as it employs fast retransmit and recovery to recover from packet losses.

4.5 Understand the working of TCP BIC Congestion control algorithm, simulate, and plot the TCP congestion window (Level 2)

4.5.1 Theory

In BIC congestion control is viewed as a searching problem in which the system can give yes/no feedback through packet loss as to whether the current sending rate (or window) is larger than the network capacity. The current minimum window can be estimated as the window size at which the flow does not see any packet loss. If the maximum window size is known, we can apply a binary search technique to set the target window size to the midpoint of the maximum and minimum. As increasing to the target, if it gives any packet loss, the current window can be treated as a new maximum and the reduced window size after the packet loss can be the new minimum. The midpoint between these new values becomes a new target. Since the network incurs loss around the new maximum but did not do so around the new minimum, the target window size must be in the middle of the two values. After reaching the target and if it gives no packet loss, then the current window size becomes a new minimum, and a new target is calculated. This process is repeated with the updated minimum and maximum until the difference between the maximum and the minimum falls below a preset threshold, called the minimum increment (S_{min}). This technique is called binary search increase.

Additive Increase

To ensure faster convergence and RTT-fairness, binary search increase is combined with an additive increase strategy. When the distance to the midpoint from the current minimum is too large, increasing the window size directly to that midpoint might add too much stress to the network. When the distance from the current window size to the target in binary search increase is larger than a prescribed maximum step, called the maximum increment (Smax) instead of increasing window directly to that midpoint in the next RTT, we increase it by Smax until the distance becomes less than Smax, at which time window increases directly to the target. Thus, after a large window reduction, the strategy initially increases the window linearly, and then increases logarithmically. This combination of binary search increase and additive increase is called as binary increase. Combined with a multiplicative decrease strategy, binary increase becomes close to pure additive increase under large windows. This is because a larger window results in a larger reduction by multiplicative decrease and therefore, a longer additive increase period. When the window size is small, it becomes close to pure binary search increase – a shorter additive increase period.

Slow Start

After the window grows past the current maximum, the maximum is unknown. At this time, binary search sets its maximum to be a default maximum (a large constant) and the current window size to be the minimum. So, the target midpoint can be very far. According to binary increase, if the target midpoint is very large, it increases linearly by the maximum increment. Instead, run a "slow start" strategy to probe for a new maximum up to Smax. So if cwnd is the current window and the maximum increment is Smax, then it increases in each RTT round in steps cwnd+1, cwnd+2, cwnd+4,,, cwnd+Smax. The rationale is that since it is likely to be at the saturation point and also the maximum is unknown, it probes for available bandwidth in a "slow start" until it is safe to increase the window by Smax. After slow start, it switches to binary increase.

Fast Convergence

It can be shown that under a completely synchronized loss model, binary search increase combined with multiplicative decrease converges to a fair share. Suppose there are two flows with different window sizes, but with the same RTT. Since the larger window reduces more in multiplicative decrease (with a fixed factor β), the time to reach the target is longer for a larger window. However, its convergence time can be very long. In binary search increase, it takes $log(d) - log(Smin)$ RTT rounds to reach the maximum window after a window reduction of d. Since the window increases in a log step, the larger window and smaller window can reach back to their respective maxima very fast almost at the same time (although the smaller window flow gets to its maximum slightly faster). Thus, the smaller window flow ends up taking away only a small amount of bandwidth from the larger flow before the next window reduction. To remedy this behaviour, binary search increase is modified as follows. After a window reduction, new maximum and minimum are set. Suppose these values are max_wini and min_wini for flow i (i =1, 2). If the new maximum is less than the previous, this window is in a downward trend. Then, readjust the new maximum to be the same as the new target window (i.e. max_wini = (max_wini-min_wini)/2), and then readjust the target. After that apply the normal binary increase. This strategy is called fast convergence.

4.5.2 Network setup

Open NetSim and click on **Experiments> Internetworks> TCP> Advanced TCP BIC Congestion control algorithm** then click on the tile in the middle panel to load the example as shown in below Figure 4-26.
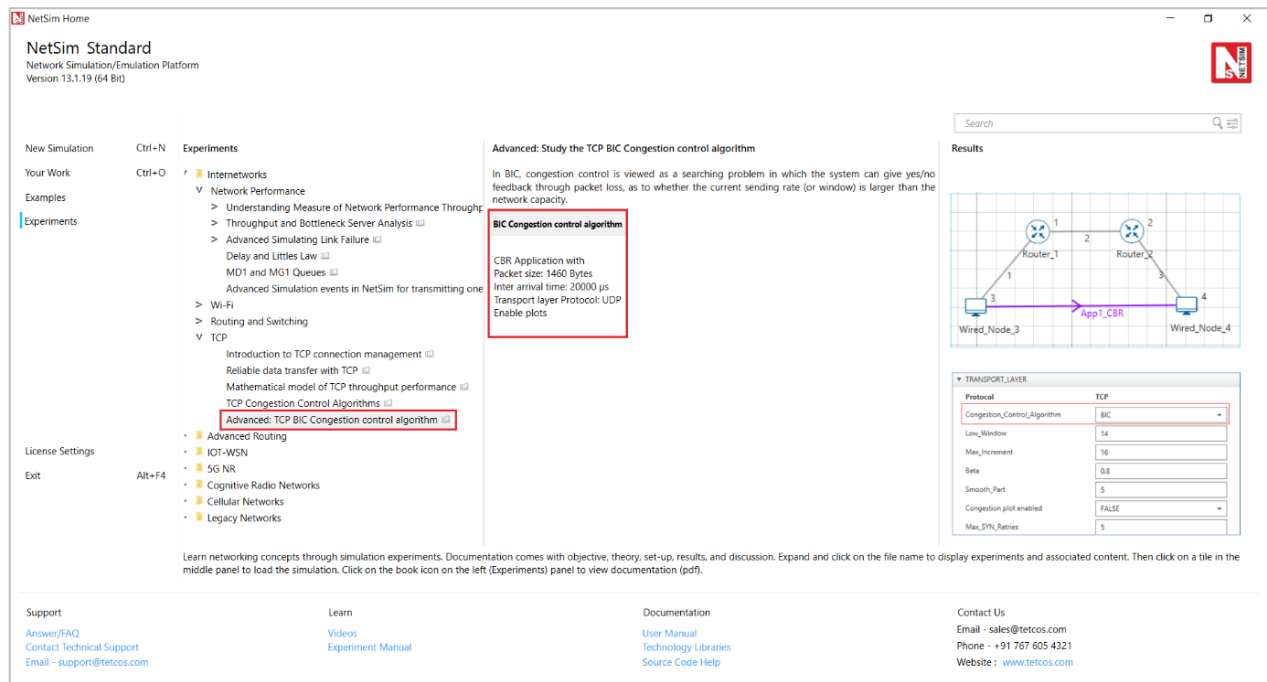
Figure 4-26: List of scenarios for the example of Advanced TCP BIC Congestion control algorithm

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 4-27.
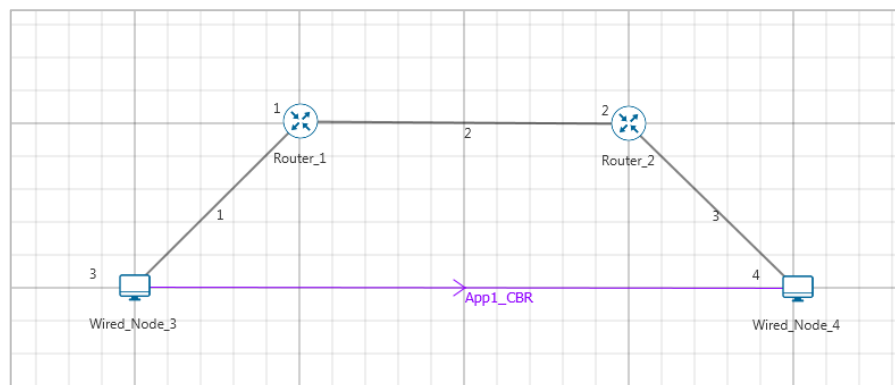
Figure 4-27: Network set up for studying the Advanced TCP BIC Congestion control algorithm

4.5.3 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 2 Routers in the **"Internetworks"** Network Library.

Step 2: In the General Properties of Wired Node 3 i.e., Source, Wireshark Capture is set to Online and in the TRANSPORT LAYER Properties, Window Scaling is set as TRUE.

Step 3: For all the devices, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to BIC. But the congestion plot is set to TRUE only in Wired_Node_3.

Figure 4-28: Transport Layer window

Step 4: The Link Properties are set according to the table given below Table 4-2.

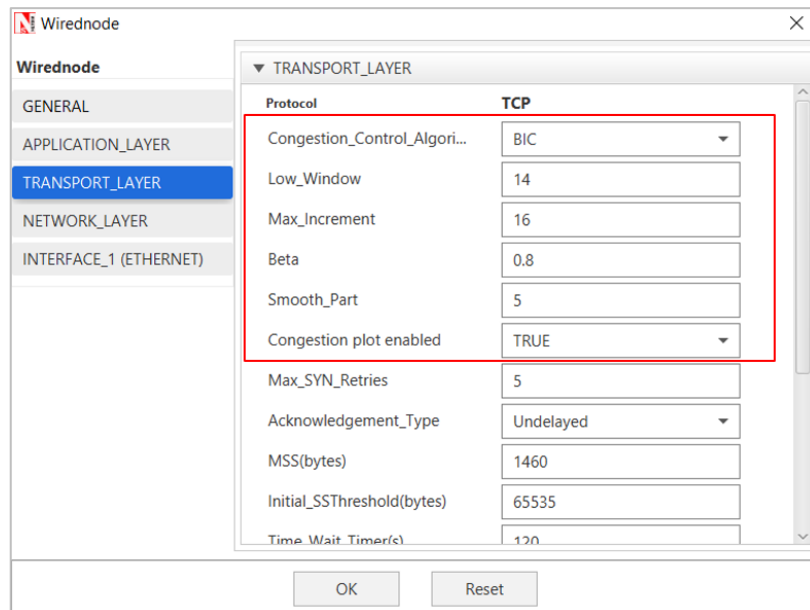| Link Properties	Wired Link 1	Wired Link 2	Wired Link 3
Uplink Speed (Mbps)	100	20	100
Downlink Speed (Mbps)	100	20	100
Uplink propagation delay (μs)	5	1000	5
Downlink propagation delay (μs)	5	1000	5
Uplink BER	0.00000001	0.00000001	0.00000001
Downlink BER	0.00000001	0.00000001	0.00000001

Table 4-2: Wired Link Properties

Step 5: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 3 i.e., Source to Wired Node 4 i.e. Destination with Packet Size set to 1460 Bytes and Inter Arrival Time set to 400 μs. Additionally, the **"Start Time"** parameter is set to 20 Seconds.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 140 Kbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps) = Packet\ Size\ (Bytes) * 8/Interarrival\ time\ (\mu s)$$

Step 6: Enable the plots and click on Run simulation. The simulation time is set to 100 seconds.

4.5.4 Output
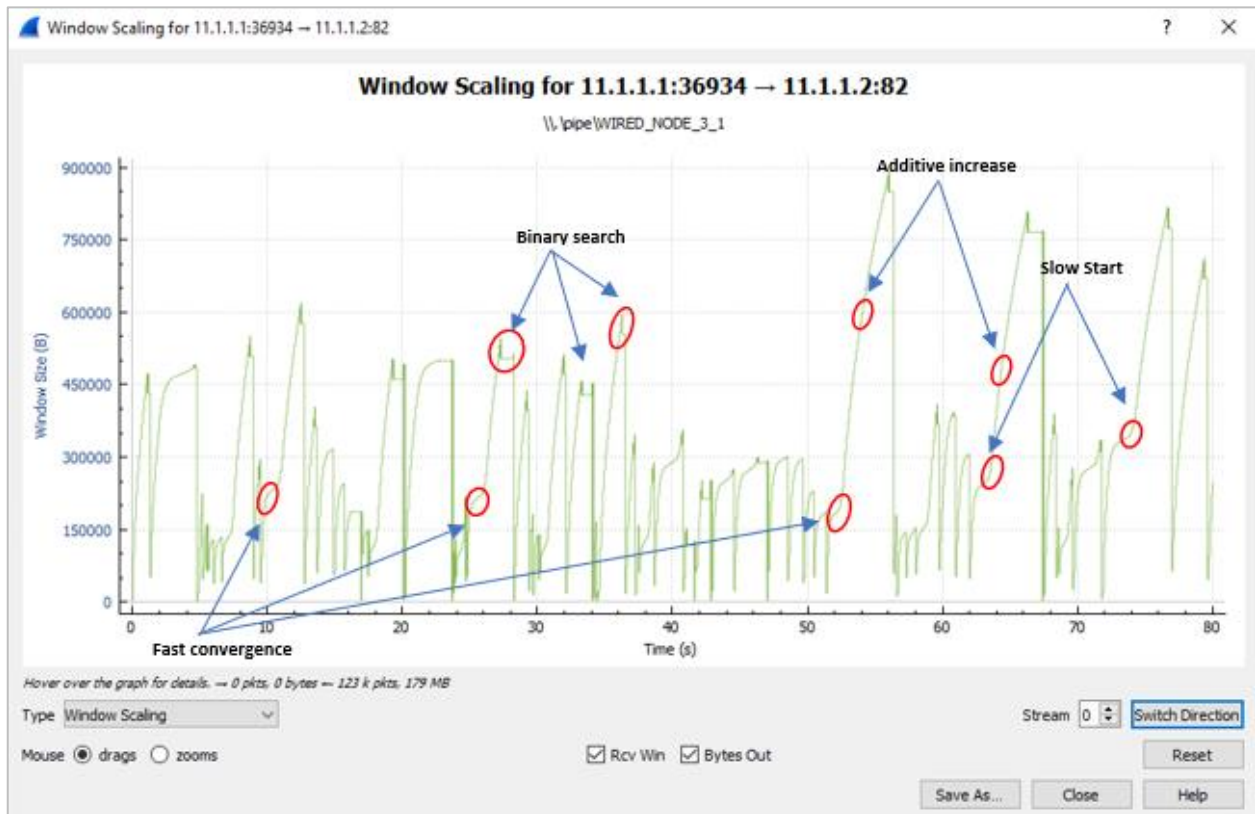
Figure 4-29: Plot of Window Scaling in Wireshark Capture

Note: User need to "zoom in" to get the above plot.

Go to the Wireshark Capture window.

Click on data packet i.e. <None>. **Go to Statistics → TCP Stream Graphs → Window Scaling**.

Click on **Switch Direction** in the window scaling graph window to view the graph.

(For more guidance, refer to section - 8.7.5 Window Scaling" in user manual)

The graph shown above is a plot of Congestion Window vs Time of BIC for the scenario shown above. Each point on the graph represents the congestion window at the time when the packet is sent. You can observe Binary Search, Additive Increase, Fast Convergence, Slow Start phases in the above graph.

5 Wi-Fi: IEEE 802.11

5.1 Wi-Fi: Throughput variation with distance (Level 1)

5.1.1 Introduction

A typical question people ask is "How fast will the WiFi be?". The answer is complicated and depends on many factors, one of which is distance between an access point (AP) and the station (STA), say a laptop or a mobile phone. The term WiFi refers to an umbrella of IEEE standards from legacy 802.11b to the latest 802.11ax. In this experiment, we study the throughput performance of an IEEE 802.11g connection as the AP-STA distance varies.

The AP and STA communicate over a wireless radio channel. This channel is susceptible to (i) attenuation (ii) noise and (iii) interference. Attenuation or propagation losses, in-turn comprise of three components namely pathloss, shadowing and fading. We only analyze pathloss and ignore more complicated phenomena i.e., shadowing, fading and interference[2].

5.1.2 Simplified pathloss model

The complexity of signal propagation makes it difficult to obtain a single model that characterizes path loss accurately across a range of different environments. For general tradeoff analysis of various system designs it is sometimes best to use a simple model that captures the essence of signal propagation without resorting to complicated path loss models, which are only approximations to the real channel anyway. The following simplified model for path loss as a function of distance is commonly used for system design:

$$P_r = P_t \times c_0 \times \left(\frac{d_0}{d}\right)^{\eta}$$

In this approximation, P_r is the received power sometimes called received signal strength (RSS), P_t is the transmit power, c_0 is the path loss at the "reference" distance, d_0 (usually 1m), η is the path-loss exponent and d is the distance between the transmitter and the receiver. The dB attenuation is thus

$$P_r(dBm) = P_t(dBm) + c_0\ (dB) - 10 \times \eta \times \log_{10}\left(\frac{d}{d_0}\right)$$

---

[2] NetSim supports shadowing, fading and interference. Interested users can enable them and study their effects.

As d increases, the received power decreases, e.g., doubling the distance reduces the received power by approximately 3η, since $\log_{10} 2 \approx 0.3$. Typical values of η, indoors, could be 3 to 5, resulting in 9 dB to 15 dB additional path loss for doubling the value of d.

5.1.3 The IEEE 802.11g PHY Rates Table

IEEE 802.11g utilizes OFDM over the entire 20 MHz channel. There are 52 OFDM carriers, of which 48 carriers are used for data transmission and 4 are used for control. The OFDM symbol rate is 250 Ksps. With 48 symbols being sent together (across the 48 carriers), we obtain 12 Msps. In principle, each symbol can be modulated according to any of the modulation and coding schemes (MCS) shown in Table 5-1. The bit rate shown in the last column of the table assumes the situation in which all the symbols are modulated using the same MCS. Thus, for example, the MCS in the row indexed by 4 uses 16 QAM (i.e., 4 bits per symbol) with a coding rate of 1/2 (i.e., half the bits are data bits, and the rest are channel error protection bits), yielding 2 data bits per symbol, and, therefore, 24 Mbps overall bit rate, assuming that all OFDM symbols use the MCS.

MCS stands for modulation and coding scheme. The MCS defines the numbers of useful bits which can carried by one symbol. In WiFi IEEE 802.11g standard, the MCS depends on the received signal strength (RSS). The higher the signal strength the higher the MCS and more useful bits can be transmitted in a symbol. Thus, the PHY bit rate depends on the MCS chosen. IEEE 802.11g devices can transmit at speeds of 6, 9, 12, 18, 24, 36, 48 and 54Mbps as shown in the table below.

Index	Rx Sensitivity (dBm)	Modulation	Code Rate	Bit Rate
0	-82	BPSK	1/2	6 Mbps
1	-81	BPSK	3/4	9 Mbps
2	-79	QPSK	1/2	12 Mbps
3	-77	QPSK	3/4	18 Mbps
4	-74	16 QAM	1/2	24 Mbps
5	-70	16 QAM	3/4	36 Mbps
6	-66	64 QAM	2/3	48 Mbps
7	-65	64 QAM	3/4	54 Mbps

Table 5-1: 802.11g bit rates for different modulation schemes, and the minimum received signal power for achieving each bit rate.

In the above table, Rx Sensitivity is the minimum RSS. A simulation assumption in NetSim is that the transmitter knows the RSS at the receiver. Thus, the transmitter chooses the MCS by comparing the RSS against the Receiver-Sensitivity for different MCSs. The highest possible MCS is then chosen.

5.1.4 Calculating distances at which the PHY rate changes

In this section, we predict the AP-STA distance thresholds for the different PHY rates. We know that

$$P_r = P_t - c_0 - 10\,\eta\,log_{10}\,(d)$$

At 2.4 GHz, c_0 is $40.09\ dB$. For $P_t = 100\ mW\ (20\ dBm)$, $\eta = 3.5$, and setting P_r as equal to the receive sensitivity (Ref: **Error! Reference source not found.**), and we get the following inequality for the 6 Mbps PHY rate

$$-82 \geq 20 - 40.09 - 35 \times \log(d) \geq -81$$

This gives $54.99m \leq d \leq 58.72\ m$. Similarly, we compute the AP-PHY distance for all the rates and arrive at the table below

Rx Sensitivity	Bit Rate	$d_{max}\ (m)$	$d_{min}\ (m$
-82	6 Mbps	58.72	54.99
-81	9 Mbps	54.99	48.21
-79	12 Mbps	48.21	42.27
-77	18 Mbps	42.27	34.68
-74	24 Mbps	34.68	26.67
-70	36 Mbps	26.67	20.50
-66	48 Mbps	20.50	19.19
-65	54 Mbps	19.19	1

Table 5-2: We see the maximum and minimum AP-STA distances for different 801.11g PHY bit rates. The PHY rate is 0 for $d \geq 58.72m$. We have chosen $P_t = 100\ mW$ and $\eta = 3.5$.
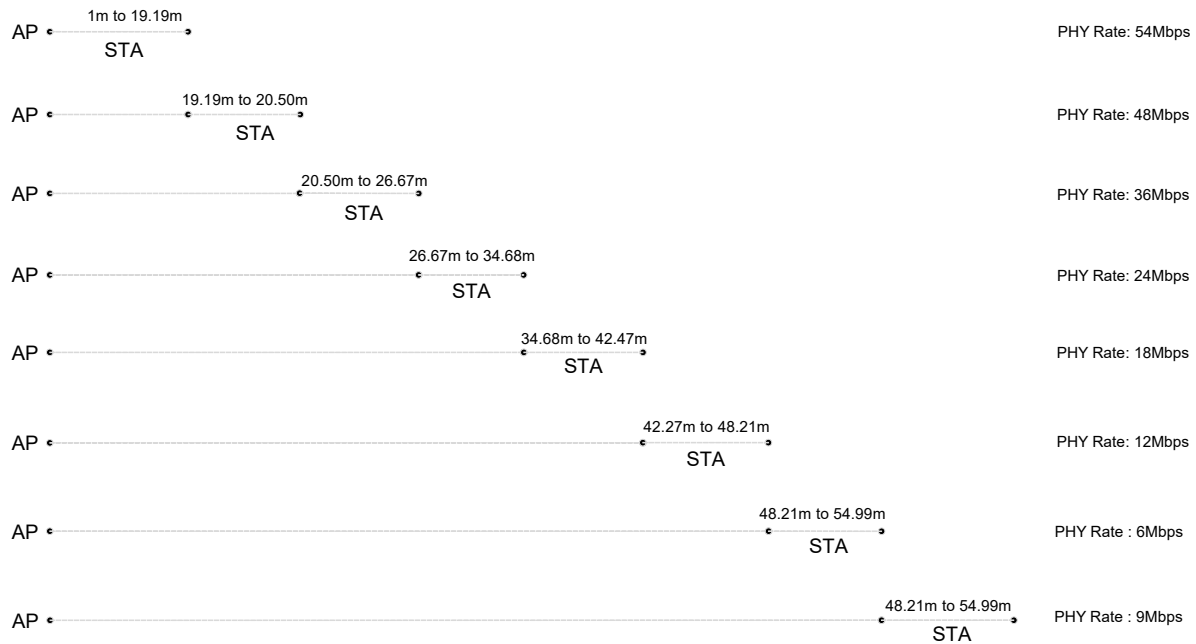
Table **5-2** can be visualized as shown below.

Figure 5-1: Illustration of variation in AP data (PHY) rate vs. distance for $P_t = 100\ mW$ and $\eta = 3.5$

5.1.5 Predicting the throughput

From the experiment Wi-Fi-UDP-Download-Throughput we know that application throughput, θ is

$$\theta = \frac{Application\ Payload\ in\ Packet\ (bits)}{Average\ Time\ per\ Packet(\mu s)}$$

$Average\ time\ per\ packet\ (\mu s)$

$$= DIFS + Average\ Backoff\ time + Packet\ Transmission\ Time + SIFS$$
$$+ ACK\ Transmission\ Time$$

Therefore,

$$\theta = \frac{L_{pkt} \times 8}{T_{DIFS} + \left(\frac{CW_{min}}{2} \times T_{slot}\right) + \left(T_{preamble} + \frac{(L_{pkt} + OH) \times 8}{PHYRate}\right) + T_{SIFS} + \left(T_{preamble} + \frac{L_{ACK} \times 8}{PHYRate_{min}}\right)}$$

In the above formula θ is in Mbps as the time in the denominator is in μs.

The predicted application throughput for a 1460B packet, with 68B overheads, ACK size of 14B, and PHY Rate of 54 Mbps is

$$\theta = \frac{1450 \times 8}{34 + \left(\frac{15}{2} \times 9\right) + \left(20 + \frac{(1450 + 68) \times 8}{54}\right) + 16 + \left(20 + \frac{14 \times 8}{6}\right)} = \frac{11600}{401.04} = 28.92\ Mbps$$

Doing the same computation for the different PHY rates leads to the following application throughput predictions. Users just need to replace 54 in the above equation with the appropriate PHY rate.

PHY rate (Mbps)	Predicted Application Throughput (Mbps)
54	28.92
48	27.02
36	22.59
24	17.00
18	13.63
12	9.76
9	7.60
6	5.27

Table 5-3: Predicted application throughput for various PHY rates

In the following section we create scenarios with varying AP-STA distances in NetSim, model the same pathloss equation and compare simulation results against predictions.

5.1.6 Network Setup

Open NetSim and click on **Experiments> Internetworks> Wi-Fi> Impact of distance on Wifi throughput** then click on the tile in the middle panel to load the exampleas shown in below Figure 5-2.
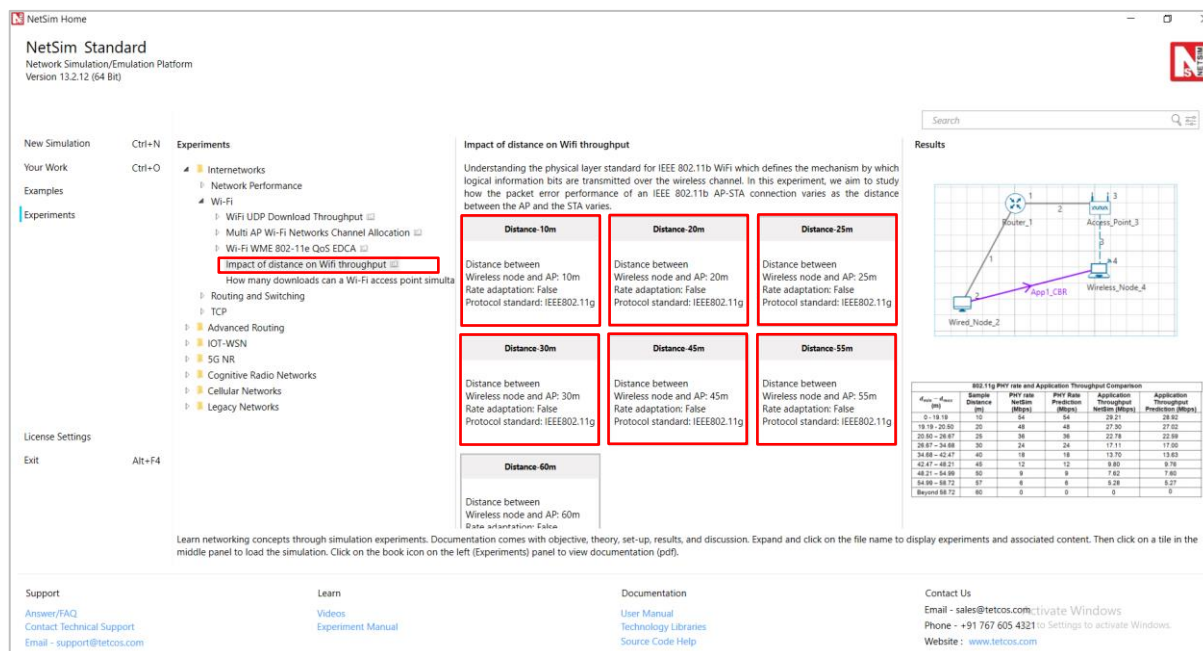
Figure 5-2: List of scenarios for the example of Impact of distance on Wi Fi throughput

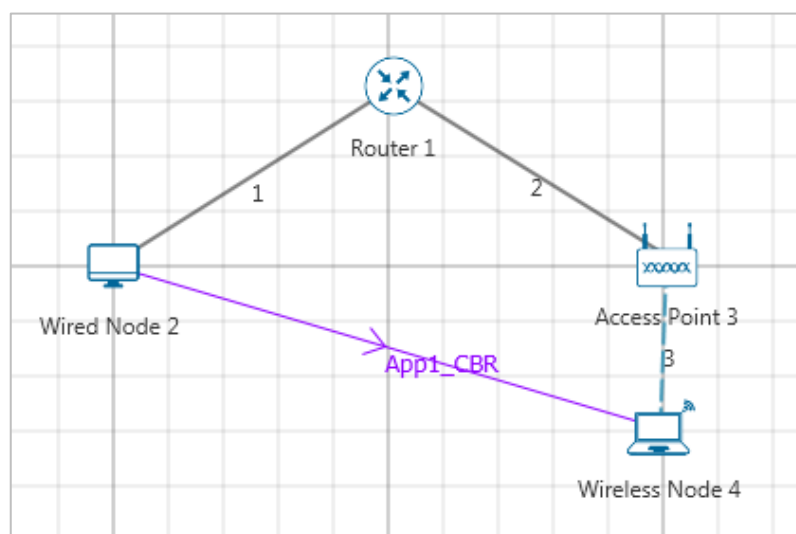NetSim UI displays the configuration file corresponding to this experiment as shown below in Figure 5-3.

Figure 5-3: Network set up for studying the Impact of distance on Wi-Fi throughput

5.1.7 Procedure

The following set of procedures were done to generate this sample.

Step 1: A network scenario is created in NetSim GUI comprising of 1 Wired Node, 1 Router, 1 Access Point and 1 Wireless Node in the **"Internetworks"** Network Library.

Step 2: In the Destination Node, i.e. Wireless Node 4, the Interface 1 (WIRELESS) > Physical Layer, Protocol Standard is set to IEEE802.11g and in the Interface 1 (WIRELESS) > Datalink Layer, Rate Adaptation is set to False.

Step 3: The position of the Wireless Node and the Access Point in the grid environment is set according to the values given in the below table see Table 5-4.

Device Positions		
	Wireless Node 4	**Access Point**
X	250	250
Y	10	0

Table 5-4: Device Positions

Step 4: Right click on **Wireless Node** and **Access Point,** select Properties and select DCF as the Medium Access Protocol in the DATALINK_LAYER of INTERFACE_1.

Step 5: Right-click the link ID (of a wired/wireless link) and select Properties to access the link's properties. The parameters are set according to the values given in the below **Error! Reference s ource not found.**/**Error! Reference source not found.**.

Wired Link Properties	
Max Uplink Speed (Mbps)	100
Max Downlink Speed (Mbps)	100
Uplink BER	0
Downlink BER	0
Uplink Propagation Delay (µs)	0
Downlink Propagation Delay (µs)	0

Table 5-5: Wired Link Properties

Wireless Link Properties	
Channel Characteristics	Pathloss Only
Path Loss Model	Log Distance
Path Loss Exponent	3.5

Table 5-6: Wireless Link Properties

Step 6: Right click on **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 2 i.e., Source to Wireless Node 4 i.e., Destination with Packet Size set to 1450 Bytes and Inter Arrival Time set to 200 µs. It is set such that, the **Generation Rate** equals to 58 Mbps.

Transport Protocol is set to **UDP** instead of TCP.

Step 7: Run the simulation for 10 sec. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis.

Go back to the scenario and change the distance between Access Point and Wireless Node (i.e., Change the Y axis of Wireless Node (STA)) as 10, 20, 30, 40, 45, 50, 57 and 60.

5.1.8 Simulation Output

Data rate can be calculated from packet trace by using the formula given below:

$$PHYRate\ (Mbps) = \frac{PHYLayerPayload\ (B) \times 8}{PHYEndTime\ (\mu s) - PHYArrivalTime(\mu s) - 20(\mu s)}$$

$20\mu s$ is the preamble time for 802.11g.

Given below are the steps to calculate the PHY rate for a 10m AP-STA distance of STA using the packet trace.

Step 1: Make sure that the packet trace is enabled before the simulation, and then run the simulation for a desired time.

Step 2: Click on 'Open Packet Trace'

Step 3: Filter CONTROL_PACKET_TYPE/APP_NAME as App1_CBR (Only data packets) and TRANSMITTER_ID as ACCESSPOINT-1 (Only wireless channel).

Step 3: Add a column 'Phy Rate (Mbps)' after PHY_LAYER_PAYLOAD(Bytes)

Step 4: Add formula in the first cell of newly added column.

=([@[PHY_LAYER_PAYLOAD(Bytes)]]*8)/([@[PHY_LAYER_END_TIME(US)]]-
[@[PHY_LAYER_ARRIVAL_TIME(US)]]-20)

Step 5: We get the Phy Rate as 54Mbps. (The column Phy Rate is formatted to zero decimal places)

Figure 5-4: Calculating the Phy Rate using its formula in packet trace.

5.1.9 Results

Simulation results with PHY rates and application throughputs are tabulated in Table 5-7.

802.11g PHY rate and Application Throughput Comparison					
$d_{min} - d_{max}$ (m)	Sample Distance (m)	PHY rate NetSim (Mbps)	PHY Rate Prediction (Mbps)	Application Throughput NetSim (Mbps)	Application Throughput Prediction (Mbps)
0 - 19.19	10	54	54	29.15	28.92
19.19 - 20.50	20	48	48	27.24	27.02
20.50 – 26.67	25	36	36	22.71	22.59
26.67 – 34.68	30	24	24	17.09	17.00
34.68 – 42.47	40	18	18	13.68	13.63

42.47 – 48.21	45	12	12	9.79	9.76
48.21 – 54.99	50	9	9	7.61	7.60
54.99 – 58.72	57	6	6	5.28	5.27
Beyond 58.72	60	0	0	0	0

Table 5-7: We see how PHY rate, Application Throughput varies with AP-STA distance
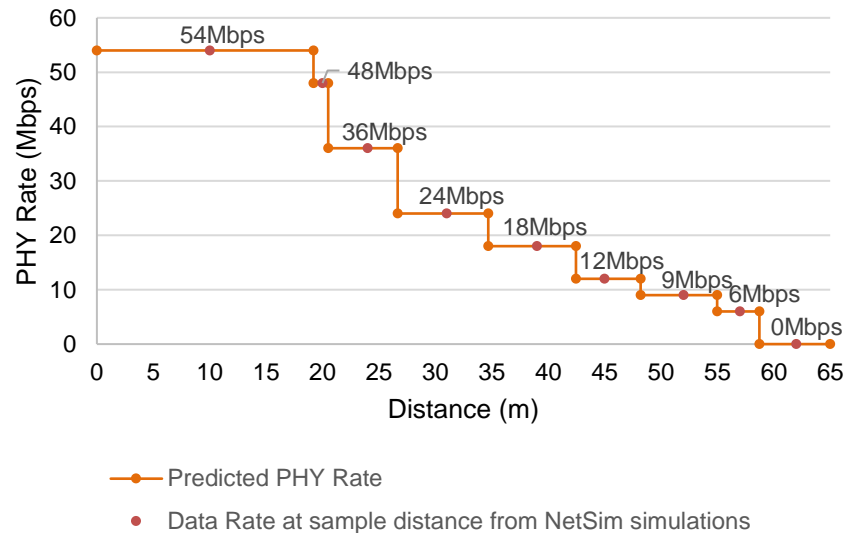

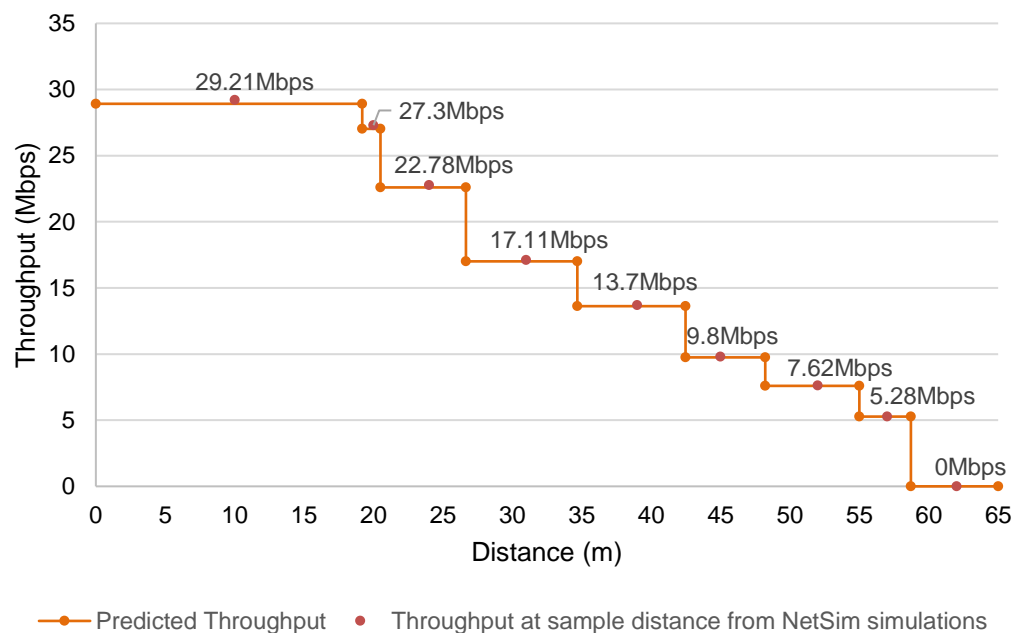Figure 5-5: Data Rate vs. Distance

Figure 5-6: Application Throughput vs. Distance

5.1.10 Discussion

It is interesting to note that for 54Mbps PHY rate the application throughput is 29.21 Mbps, which is about 54.1% of the PHY rate. However, for 6 Mbps PHY rate the application throughput is 5.28 Mbps which is 88% of the PHY rate. Why is this so?

Let us go back to the average time per packet – the denominator of the throughput expression - which is

$$T_{DIFS} + \left(\frac{CW_{min}}{2} \times T_{slot}\right) + \left(T_{preamble} + \frac{(L_{pkt} + OH) \times 8}{PHYRate}\right) + T_{SIFS} + \left(T_{preamble} + \frac{L_{ACK} \times 8}{PHYRate_{min}}\right)$$

Per the 802.11g standards the 14 Byte MAC ACK is always sent at the control rate or $PHYRate_{min}$ which is 6 Mbps. Therefore, all terms in the above expression except $\frac{(L_{pkt}+OH)\times 8}{PHYRate}$ do not vary with the PHY rate. Thus, the average time per packet reduces to $\left(176.18 + \frac{12144}{PHYRate}\right)$. Substituting the different $PHYRates$ in this expression we get

PHY rate (Mbps)	Avg time per packet (in μs)
54	401.05
48	429.17
36	513.5
24	506
18	850.83
12	1188.17
9	1525.5
6	2200.17

At high PHY rates the constant value of 176.18 μs is a significant part of the average transmission time. As the PHY rate reduces the significance of this terms reduces exponentially.

5.1.11 Conclusion

To summarize, we understood the log distance pathloss model and saw how WiFi PHY rates and Application throughputs vary with AP-STA distance. Then we predicted the WiFi PHY rate and application throughputs. Simulation results show excellent agreement with theory.

5.1.12 References

Some of the theoretical content is from the books (i) Wireless Communications by Andrea Goldsmith, and (ii) Wireless Networking by Anurag Kumar.

5.1.13 Exercises

1. Keeping other variables fixed, change the transmit power (P_t) to a different value. Predict d for different PHY rates. Compare against simulation
2. Keeping other variables fixed, change the pathloss exponent η (generally in the range of 2 to 5). Predict d for different PHY rates. Compare against simulation.
3. Keeping other variables fixed, change the packet size L_{pkt} (from between 100B to 1460B). Take care to ensure the generation rate is sufficiently high to ensure full buffers (saturation) at the transmitting node. Predict θ for different PHY rates. Compare against simulation.

5.2 Wi-Fi: UDP Download Throughput (Level 1)

5.2.1 The Setup and Motivation

The most basic packet transfer service offered by the Internet is called the "datagram" service, in which a series of packets are transmitted to a receiver without any packet loss recovery, flow control, or congestion control. The Internet's UDP protocol implements the datagram service. In this experiment, we will study the performance of UDP transfers from a server on a wireline local area network to Wi-Fi Stations (STA), via Wi-Fi Access Points (AP). The schematic of the network that we will be simulating in NetSim is shown in the figure below Figure 5-7.
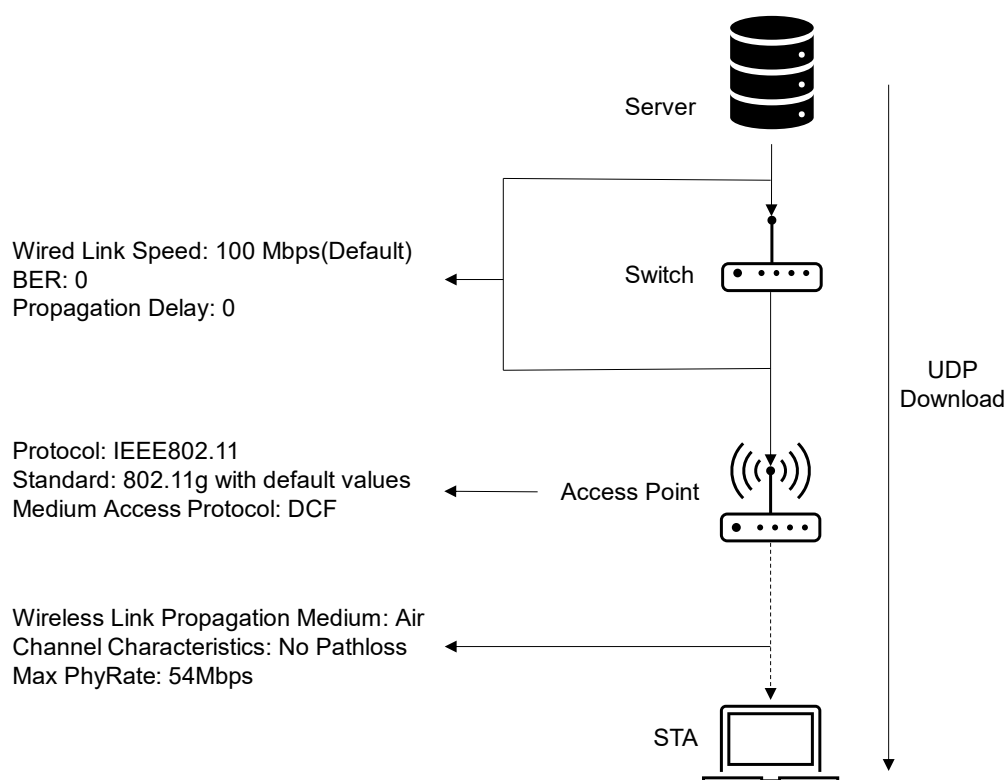

Figure 5-7: Network topology and application flow from server to STA via a single AP

The server, which contains the data that needs to be transferred to the STAs (say, laptops), is connected by a 100 Mbps switched Ethernet link to an Ethernet switch, which is, in turn, connected to the Wi-Fi APs. Each AP is associated (i.e., connected) at a phy rate of 54Mbps (802.11g standard) to a single STA. The objective is to transfer many packets (say, constituting a video) from the server to each of the STAs, the packet stream to each of the STAs being different (e.g., each STA is receiving a different video from the server). In this experiment, we are interested in studying the limitation that the Wi-Fi link places on the data transfers. We assume that the server transmits the packets at a saturation rate of wireless link (i.e., above 54Mbps) so that the queues at the APs fill up, and the rate of the UDP transfers is therefore, governed by the Wi-Fi link. It may be noted that, in practice, there will be a flow control

mechanism between each STA and the server, that will control the rate at which the server releases packets, to prevent buffer overflow at the APs.

In this setting, this experiment will ask one precise question. With the buffers at the AP full, at what rate will the Wi-Fi protocol transfer the packets from the APs to the STAs over the wireless link. We will study two cases:

1. A single AP and a single STA: Since there is only one transmitter in this wireless network (namely, the AP), there is no contention, and the rate of packet transfer over the link will be governed by the basic overheads in the protocol, such as the interframe spacings, packet header overheads, transmit-receive turn-around times, and acknowledgement times. We will begin by a simple calculation (essentially timing book-keeping) that will predict the UDP throughput, and then we will verify our calculation using the NetSim simulator.

2. Multiple APs and one STA for each AP: This is the more common situation (for example neighboring apartments in a building, each with one AP and one laptop, all drawing data from the Internet service provider). The performance of such a system depends on the wireless propagation path-loss between the various APs. A predictive analysis is difficult in the general case. For deriving some insight, we will study the case where all the APs are close to each other (i.e. setting channel characteristics as No Pathloss), and thus exactly one transmission from AP to an STA can be successful at any time. If two or more APs transmit together, then all the transmissions are not successful. Even in this case, the analysis mathematically complex and is available in, Anurag Kumar, D. Manjunath and Joy Kuri. 2008: Wireless Networking. Sec 7.4

5.2.2 Predicting the UDP Throughput

5.2.2.1 One AP and one STA

As stated above, in the setup described, the AP queue is full. Thus, after a packet is completely transmitted over the wireless link, immediately the process for transmitting the next packet starts. This is illustrated by the upper part of the Figure 5-8, where the successive packets from the AP are shown as being sent back-to-back. The time taken to send a packet is, however, not just the time to clock out the physical bits corresponding to the packet over the Wi-Fi medium. After the completion of a packet transfer, the AP's Wi-Fi transmitter waits for a Distributed Coordination Function Inter-Frame Space (DIFS), followed by a backoff that is chosen randomly between 1 and 32 slots. Upon the completion of the backoff, the packet transmission starts. Each packet carries physical layer overheads, MAC layer overheads, and IP overheads. After the transmission of the packet, there is a Short Inter-Frame Space (SIFS), which gives time to the receiver (namely, the STA) to transition from the listening mode to the transmit mode. After the SIFS, the STA sends back a MAC acknowledgement (ACK). This completes the transmission of one UDP packet from the AP to the STA. Immediately, the process for sending the next packet

can start. The details of the various timings involved in sending a single UDP packet are shown in the lower part of the figure below Figure 5-8.
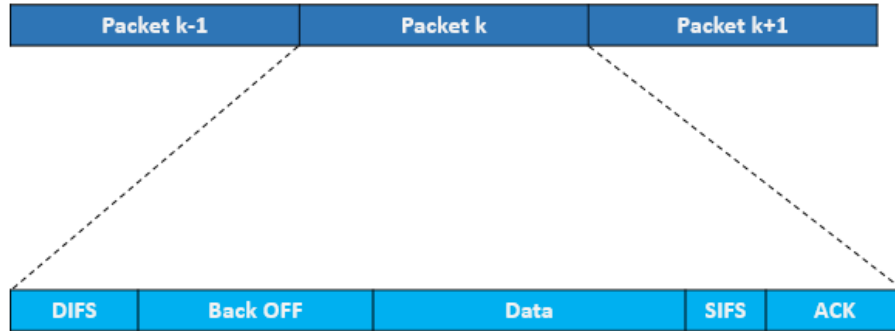

Figure 5-8: Detailed view of transmission of a single packet in WiFi involving single AP-STA.

In this experiment, the payload in each packet is the same (1450 Bytes). Since the packets are sent back-to-back, and the state of the system is identical at the beginning of each packet transmission, the throughput (in Mbps) is computed by the following simple (and intuitive) relation.

5.2.2.1.1 Without RTS-CTS

$$UDP\ Throughput\ (Mbps) = \frac{Application\ Payload\ in\ Packet\ (bits)}{Average\ Time\ per\ Packet(\mu s)}$$

$Average\ time\ per\ packet\ (\mu s)$
$$= DIFS + Average\ Backoff\ time + Packet\ Transmission\ Time + SIFS$$
$$+ Ack\ Transmission\ Time$$

$$Packet\ Transmission\ Time\ (\mu s) = Preamble\ time + \left(\frac{MPDUSize}{PHYrate}\right)$$

$$Average\ Backoff\ time\ (\mu s) = \left(\frac{CWmin}{2}\right) \times Slot\ Time$$

$$Ack\ Transmission\ Time\ (\mu s) = Preamble\ time + \left(\frac{AckPacketSize}{AckPHYRate}\right)$$

$$DIFS\ (\mu s) = SIFS + 2 \times Slot\ Time$$

$$Average\ Backoff\ time\ (\mu s) = \left(\frac{CWmin}{2}\right) \times Slot\ Time$$

$Preamble\ time = 20\ \mu s\ for\ 802.11g$

$MPDU\ Size = 1450 + 8 + 20 + 40 = 1518\ Bytes$

$Application\ Payload = 1450\ Bytes$

$SIFS = 10\ \mu s\ and\ Slot\ Time = 9\ \mu s$

$CW_{min} = 15\ slots\ for\ 802.11g$

$$DIFS = SIFS + 2 \times Slot\ Time = 16\ \mu s + 2 \times 9 \mu s = 34\ \mu s$$

$$Average\ Backoff\ Time = \left(\frac{CWmin}{2}\right) \times Slot\ Time = \left(\frac{15}{2}\right) \times 9 = 67.5\ \mu s$$

$$Packet\ Transmission\ Time = 20\mu s + \frac{1518\ (B) \times 8}{54\ (Mbps)} = 244.88\ \mu s$$

$$Ack\ Transmission\ Time = 20\mu s + \frac{14\ (B) \times 8}{6\ (Mbps)} = 38.66\ \mu s$$

$$Average\ time\ per\ packet = 34 + 67.5 + 244.88 + 16 + 38.66 = 401.04\ \mu s$$

$$UDP\ Throughput = \frac{1450 \times 8}{401.04} = 28.92\ Mbps$$

5.2.2.1.2 With RTS-CTS

$$UDP\ Throughput\ (Mbps) = \frac{Application\ Payload\ in\ Packet\ (bits)}{Average\ Time\ per\ Packet(\mu s)}$$

$$Average\ time\ per\ packet\ \ (\mu s)$$
$$= DIFS +\ RTS\ Packet\ Transmission\ Time + SIFS$$
$$+ CTS\ Packet\ Transmission\ Time + SIFS +\ Average\ Backoff\ time$$
$$+ Packet\ Transmission\ Time\ +\ SIFS\ +\ Ack\ Transmission\ Time$$

$$RTS\ packet\ transmission\ time = Preamble\ time + \left(\frac{RTS\ Packet\ payload}{Data\ Rate}\right) = 20 + \left(\frac{20 \times 8}{6}\right)$$
$$= 46.66\ \mu s$$

$$CTS\ packet\ transmission\ time = Preamble\ time + \left(\frac{CTS\ Packet\ payload}{Data\ Rate}\right) = 20 + \left(\frac{14 \times 8}{6}\right)$$
$$= 38.66\ \mu s$$

$$Average\ time\ per\ packet = 34 + 46.66 + 16 +\ 38.66 + 16 +\ 67.5 + 244.88 + 16 + 38.66$$
$$= 518.36\ \mu s$$

$$UDP\ throughput = \frac{1450 \times 8}{518.36} = 22.37\ Mbps$$

5.2.2.2 Multiple APs (near each other) and one STA per AP

Since the AP queues are full, on the WiFi medium the packet transmission can still be viewed as being back-to-back as shown in the upper part of the figure.

Figure 5-9. However, since there are multiple contending AP-STA links, there are two differences between this figure and the one shown above (for the single AP and single STA case Figure 5-8).
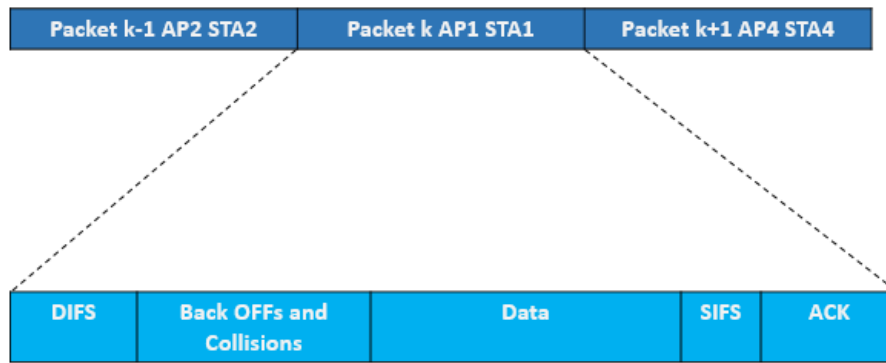
Figure 5-9: Detailed view of transmission of a single packet in WiFi involving Multiple AP-STA

a. Within each transmission period, there is now a "backoffs and collisions" period, where in the figure above we only showed a "backoff" period. Access to the channel is by contention, collision, and backoff, and this "backoffs and collisions" duration is the time taken to select one transmitting AP.

b. The other difference is that, after each "backoffs and collisions" period, any one AP-STA pair "wins" the contention, and the corresponding AP can then send a packet. It turns out that the contention mechanism is such that each of the AP-STA pairs can succeed with equal probability, independent of the pair that has previously been successful. Thus, if there are, say, 5 AP-STA pairs, then each successful packet transmission will be from any of these pairs with a probability of 0.2.

With reference to the figure above, note that, all the APs are contending to send their packets to their respective STAs, and the "Backoffs and Collisions" time is due to all the APs. However, finally, only one packet transmission succeeds. We will attribute all the contention overheads to the successful transmission of this packet. Thus, we will call the time duration from the beginning of a DIFS until the end of the ACK for the transmitted packet as the "effective" time taken to transmit that packet on the wireless medium. The average of these effective packet transmission times can be called the "Average time per Packet."

With this discussion, and the upper part of the figure above, it follows that the following expression still holds.

$$Total\ UDP\ Throughput\ (Mbps) = \frac{Application\ Payload\ in\ Packet\ (bits)}{Average\ Time\ per\ Packet(\mu s)}$$

We observe from the figure that the average time per packet will be larger than when there is a single AP-STA pair. Hence, the total UDP throughput will be smaller when there are multiple AP-STA pairs (since the "Application Payload in the Packet" is the same in both cases.

Having obtained the total throughput over all the AP-STA pairs in this manner, by the fact that each packet transmission is with equal probability from any of the AP-STA pairs, the UDP throughput for each AP-STA pair (for N pairs) is just $\frac{1}{N}$ of the total throughput.

5.2.3 Network Setup

Open NetSim and click on **Experiments> Internetworks> Wi-Fi> WiFi UDP Download Throughput** then click on the tile in the middle panel to load the example as shown in below Figure 5-10.
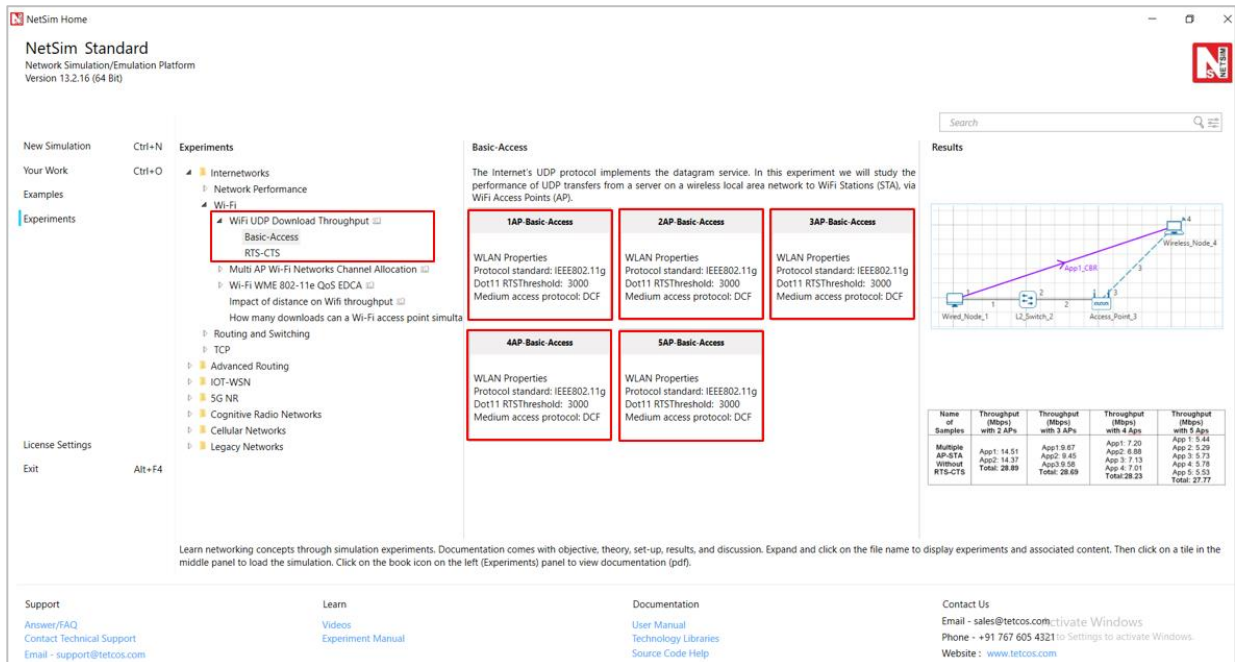

Figure 5-10: List of scenarios for the example of Wi-Fi UDP Download Throughput

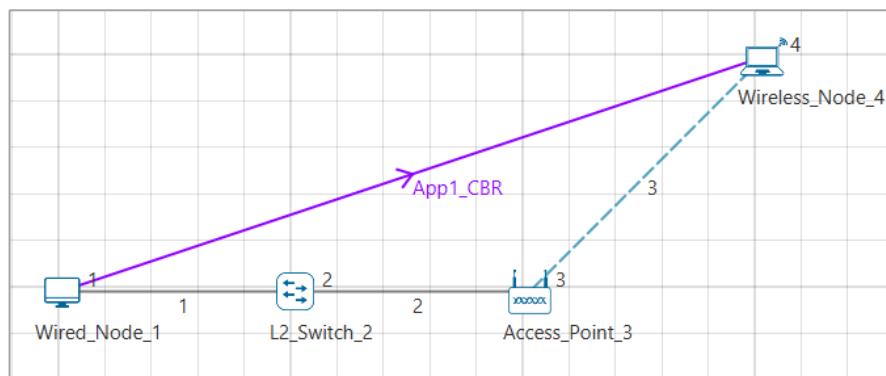NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 5-11.

Figure 5-11: Network set up for studying the A single AP-STA Without RTS-CTS

5.2.4 Procedure

5.2.4.1 Basic Access

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 1 Wireless Node, 1 L2 Switch, and 1 Access Point in the **"Internetworks"** Network Library.

Step 2: In the Interface Wireless > Physical Layer Properties of Wireless Node 5, Protocol Standard is set to IEEE 802.11g. In the Interface Wireless > Data Link Layer Properties of

Wireless Node and Access Point, RTS Threshold is set to 3000. Medium Access Protocol is set to DCF for all nodes.

Step 3: In the Wired Link Properties, Bit Error Rate and Propagation Delay is set to the default value.

Step 4: In the Wireless Link Properties, Channel Characteristics is set to NO PATH LOSS.

Step 5: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar. A CBR Application is generated from Wired Node 4 i.e., Source to Wireless Node 5 i.e., Destination with Packet Size set to 1450 Bytes and Inter Arrival Time set to 200μs. Transport Protocol is set to **UDP.**

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 100 Mbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps)\ =\ Packet\ Size\ (Bytes)\ *\ 8/Interarrival\ time\ (\mu s)$$

Step 6: Run the Simulation for 10 Seconds and note down the throughput.

5.2.4.1.1 With RTS-CTS

The following changes in settings are done from the previous sample:

Step 1: In the Interface Wireless > Data Link Layer Properties of Wireless Node and Access Point, RTS Threshold is set to 1000.

Step 2: Run the Simulation for 10 Seconds and note down the throughput.

5.2.4.2 Multiple AP-STA Without RTS-CTS: 2APs

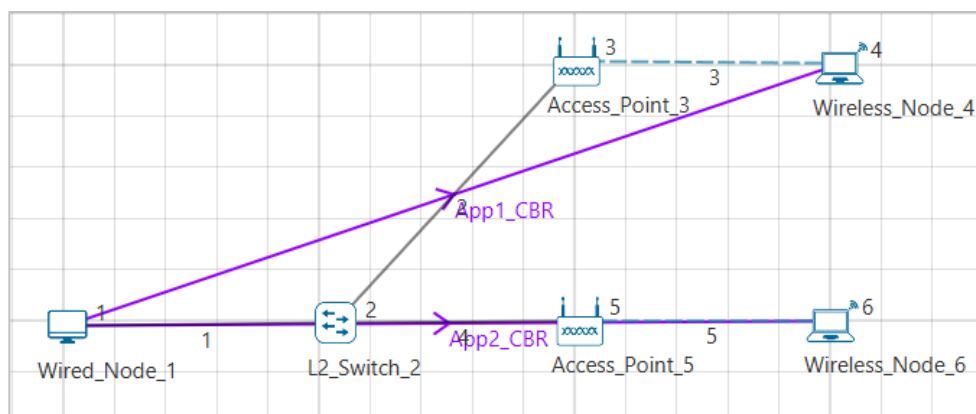The following changes in settings are done from the previous sample Figure 5-12.

Figure 5-12: Network set up for studying the Multiple AP-STA Without RTS-CTS

Step 1: A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 2 Wireless Node, 1 L2 Switch, and 2 Access Points in the **"Internetworks"** Network Library.

Step 2: In the Interface Wireless > Data Link Layer Properties of Wireless Node and Access Point, RTS Threshold is set to 3000. Medium Access Protocol is set to DCF for all nodes.

Step 3: Two CBR applications are generated from Wired Node 1 i.e., Source to Wireless Node 4 and Wireless Node 6 i.e., Destination with a Generation Rate of 58 Mbps.

Step 4: Run the Simulation for 10 Seconds and note down the throughput.

Similarly, the subsequent samples are carried out with 3, 4, and 5 Access Points and Wireless Nodes by increasing the number of AP and STA with same PHY and MAC Layer properties set in step 2 and step 3 with applications configured.

5.2.4.3 Multiple AP-STA With RTS-CTS: 2APs

The following changes in settings are done from the previous sample:

Step 1: In the Interface Wireless > Data Link Layer Properties of Wireless Node and Access Point, RTS Threshold is set to 1000. Medium Access Protocol is set to DCF for all nodes.

Step 2: Run the Simulation for 10 Seconds and note down the throughput.

Similarly, the subsequent samples are carried out with 3, 4, and 5 Access Points and Wireless Nodes.

Note: For the Next sample Newly added devices and links change the properties are per Without RTS/CTS example except RTS Threshold value.

5.2.5 Output Without and with RTS-CTS

After running simulation, check throughput in Application metrics as shown in the below screenshot Figure 5-13.
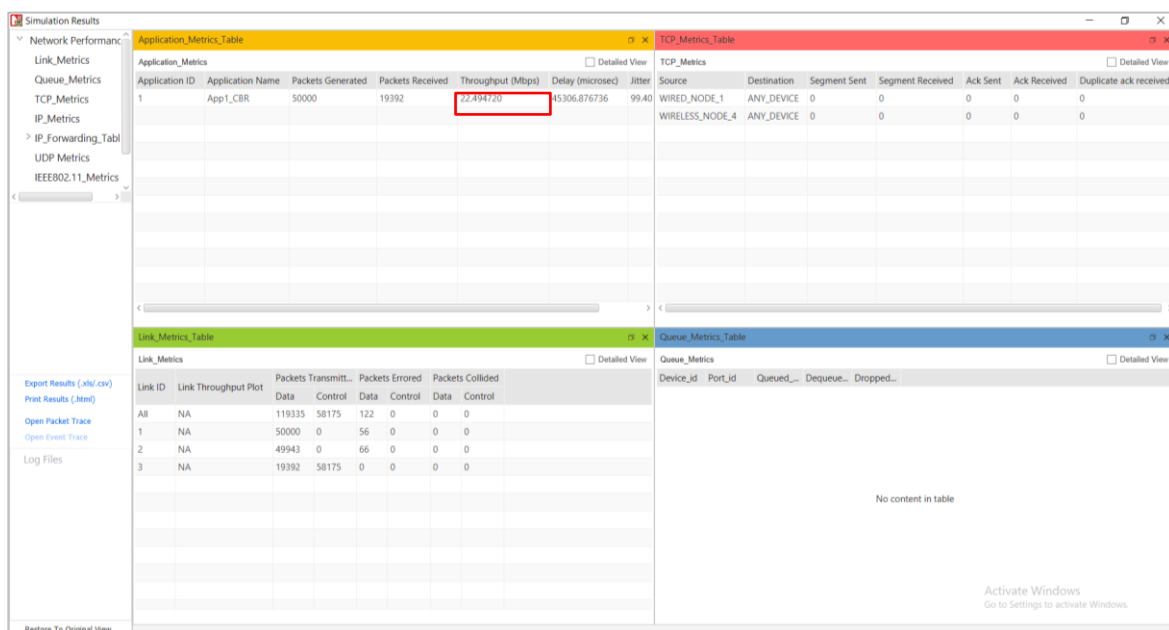

Figure 5-13: Results for Single AP-STA

Name of Samples	Predicted Throughput (Mbps)	Simulated Throughput (Mbps)
Without RTS-CTS	28.92	29.15
With RTS-CTS	22.37	22.49

Table 5-8:UDP throughput for a single AP-STA, with and without RTS-CTS

5.2.6 Output Multiple AP-STA Without and with RTS-CTS

After running simulation, check throughput in Application metrics as shown in the below screenshot Figure 5-14.
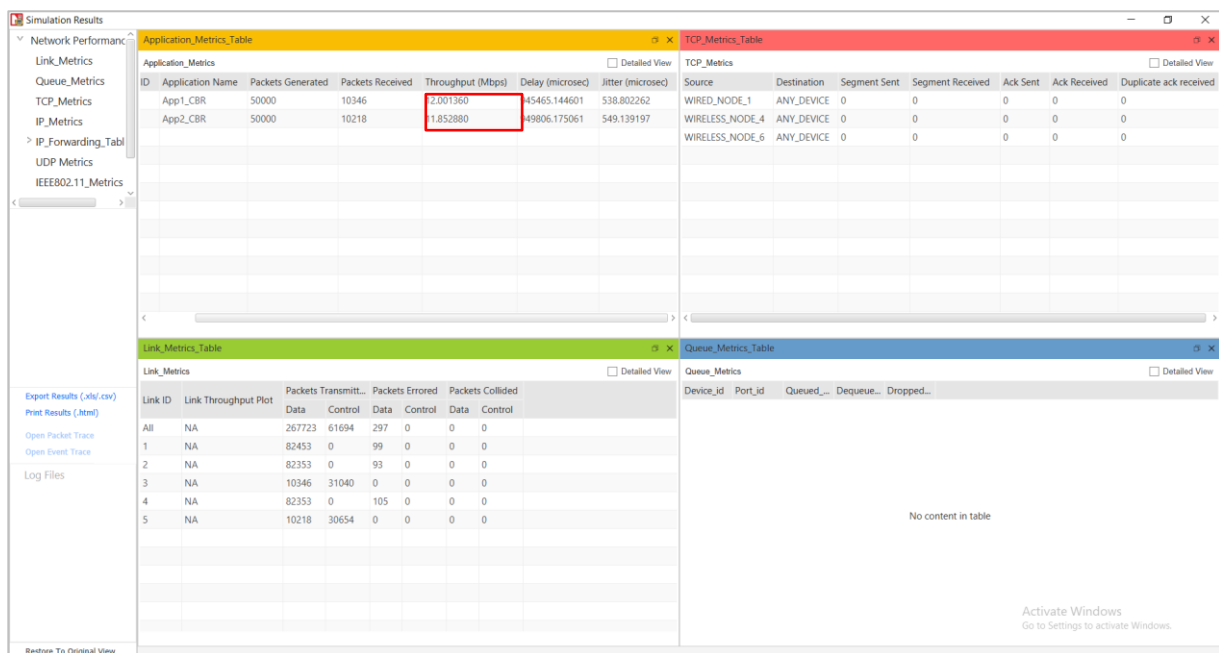

Figure 5-14: Results for Multiple AP-STA

Name of Samples	Throughput (Mbps) with 1 AP	Throughput (Mbps) with 2 APs	Throughput (Mbps) with 3 APs	Throughput (Mbps) with 4 Aps	Throughput (Mbps) with 5 Aps
Multiple AP-STA Basic Access	App 1: 29.15	App 1: 14.48 App 2: 14.36 **Total: 28.84**	App 1: 9.66 App 2: 9.43 App 3: 9.57 **Total: 28.65**	App 1: 7.18 App 2: 6.89 App 3: 7.13 App 4: 6.98 **Total: 28.19**	App 1: 5.44 App 2: 5.28 App 3: 5.71 App 4: 5.76 App 5: 5.53

				Total: 27.73
Multiple AP-STA With RTS-CTS	App 1: 22.49	App 1: 12.00 App 2: 11.85 **Total: 23.85**	App 1: 9.36 App 2: 8.02 App 3: 6.78 **Total: 24.18**	App 1: 8.20 App 2: 6.51 App 3: 4.91 App 4: 4.48 **Total: 24.21**	App 1: 7.40 App 2: 5.32 App 3: 4.54 App 4: 3.59 App 5: 3.33 **Total: 24.18**

Table 5-9: UDP throughput for 2, 3, 4, and 5 AP-STA pairs, with and without RTS-CTS.

5.2.7 Discussion

Table 5-8 shows the AP-STA UDP throughput (predicted and simulated) for a single AP-STA. Table 5-9 shows the UDP throughputs for 2, 3, 4, and 5 AP-STA pairs; the total throughput is shown along with the individual AP-STA throughputs. We can make the following observations, along with explanations (as bulleted comments) for the observations.

1. The UDP throughput with RTS/CTS turned off is larger than when RTS/CTS is used.
   - The reduction in throughput with RTS/CTS is due the RTS/CTS overheads. The RTS/CTS mechanism aims at alerting "hidden" nodes that a transmission is about to start and can reduce collisions if there are hidden nodes. Since in this experiment all nodes can directly hear each other's transmissions, the Basic Access mode suffices, whereas RTS/CTS only adds overhead.
2. In RTS-CTS case, the UDP throughput increases slightly with 2, 3 and 4 AP-STA pairs than with just one.
   - With just one AP-STA pair, there is wastage of time due to backoffs, even when there is no possibility of contention. When one more AP-STA is added some of this wastage is compensated by two APs attempting, with the possibility that one of them might finish its backoff early and grab the channel, thus reducing the backoff overhead. There is, of course, the additional time wasted due to collisions, but the balance between these two opposing phenomena is such that there is a small gain in throughput.
3. Further increase in the number AP-STA pairs leads to a decrease in throughput, but the decrease is small.
   - The IEEE 802.11 Distributed Coordination Function (DCF) manages the sharing of the Wi-Fi channel in a distributed manner. If there was a centralized scheduler than each AP could be scheduled by turn, without any backoff and collision overheads, and the total throughput would have been just that due to sending UDP packets back-to-back: $\frac{1450 \times 8}{244.88} = 47.37$ Mbps.

Thus, the total throughput with DCF is smaller than if the UDP packets were being sent back-to-back, about 28 Mbps rather than 47.37 Mbps. However, DCF implements an adaptive attempt rate mechanism, which causes nodes to attempt less aggressively as the number of contending nodes increases. It is this mechanism that prevents the total throughput from dropping steeply as the number of AP-STA pairs increases.

4. The total throughput is distributed roughly equally between the AP-STA pairs.

   ▪ This is another feature of DCF. The contending nodes obtain fair access at the packet level, i.e., each successful packet is from any of the contending nodes with equal probability. The downside of this feature is that if an AP-STA is using long packets, then that UDP flow will get a larger throughput. In this experiment, all the AP-STA UDP flows are using the same packet lengths.

5.3 How many downloads can a Wi-Fi access point simultaneously handle? (Level 2)

5.3.1 Motivation

Wi-Fi has become the system of choice for access to Internet services inside buildings, offices, malls, airports, etc. In order to obtain access to the Internet over Wi-Fi a user connects his/her mobile device (a laptop or a cellphone, for example) to a nearby Wi-Fi access point (AP). A popular use of such a connection is to download a document, or a music file; in such an application, the user's desire is to download the file as quickly as possible, i.e., to get a high throughput during the download. It is a common experience that as the number of users connected to an AP increase, the throughput obtained by all the users decreases, thereby increasing the time taken to download their files. The following question can be asked in this context.

If during the download, a user expects to get a throughput of at least θ bytes per second, what is the maximum number of users (say, n_θ) up to which the throughput obtained by every user is at least θ. We can say that n_θ is the *capacity* of this simple Wi-Fi network for the *Quality of Service (QoS)* objective θ. [3]

5.3.2 Objective

In this experiment we will learn how to obtain n_θ in a simple WiFi network where the packet loss due to channel errors is 0. In this process we will understand some interesting facts about how WiFi networks perform when doing file transfers.

5.3.3 Theory

In NetSim, we will set up a network comprising a server that carries many large files that the users would like to download into their mobile devices. The server is connected to a Wi-Fi AP, with the IEEE 802.11b version of the protocol, via an Ethernet switch. Several mobile devices (say, N) are associated with the AP, each downloading one of the files in the server. The Ethernet speed is 100Mbps, whereas the mobile devices are connected to the AP at 11Mbps, which is one of the IEEE 802.11b speeds.

We observe, from the above description, that the file transfer throughputs will be limited by the wireless links between the AP and the mobile devices (since the Ethernet speed is much larger than the Wi-Fi channel speed). There are two interacting mechanisms that will govern the throughputs that the individual users will get:

---

[3] It may be noted that the term *capacity* has several connotations in communications. Our use of the word here must not be confused with the notion of *information theoretic capacity* of a communication channel.

1. The Wi-Fi medium access control (MAC) determines how the mobile devices obtain access to the wireless medium. There is one instance of the WiFi MAC at each of the mobile devices.
2. The end-to-end protocol, TCP, controls the sharing of the wireless bandwidth between the ongoing file transfers. In our experiment, there will be one instance of TCP between the server and each of the mobile devices.

For simplicity, the default implementation of TCP in NetSim does not implement the delayed ACK mechanism. This implies that a TCP receiver returns an ACK for every received packet. In the system that we are simulating, the server is the transmitter for all the TCP connections, and each user's mobile device is the corresponding receiver.

Suppose each of the N TCP connection transmits one packet to its corresponding mobile device; then each mobile device will have to return an ACK. For this to happen, the AP must send N packets, and each of the N mobile devices must send back N ACKs. Thus, for the file transfers to progress, the AP needs to N packets for each packet (i.e., ACK) returned by each mobile device. We conclude that, in steady state, the AP must send as many packets as all the mobile devices send, thus requiring equal channel access to the AP as to all the mobile devices together.

At this point, it is important to recall that when several nodes (say, an AP and associated mobile devices) contend for the channel, the WiFi medium access control provides fair access at the packet level, i.e., each contending device has an equal chance of succeeding in transmitting a packet over the channel. Now consider the system that we have set up in this present experiment. There are N mobile devices associated with one AP. Suppose, for example, 10 of them ($N \geq 10$) all have a packet to transmit (and none other has a packet). By the fair access property of the WiFi MAC, each of these 10 nodes, along with the AP, has an equal probability of successfully transmitting. It follows, by the packet level fair access property, that each node will have a probability of $\frac{1}{11}$ of succeeding in transmitting its packet. If this situation continues, the channel access ratio to the AP will be inadequate and the equal channel access argued in the previous paragraph will be violated. It follows from this that, on the average, roughly only one mobile device will have an ACK packet in it; the AP will contend with one other node, thus getting half the packet transmission opportunities.

With the just two nodes contending, the collision probability is small (~ 0.06) and the probability of packet discard is negligibly small. Thus, the TCP window for every transfer will grow to the maximum window size. The entire window worth of TCP data packets for the N sessions will be in the AP buffer, except for a very small number of packets (averaging to about 1) which will appear as ACKs in the mobile devices.

It follows that, in steady state, the system will look like two contending WiFi nodes, one with TCP data packets and the other with TCP ACK packets. This will be the case no matter how many

downloading mobile devices there are. The total throughput can be obtained by setting up the model of two saturated nodes, one with TCP data packets, and the other with TCP ACK packets. The data packets of all the TCP connections will be randomly ordered in the AP buffer, so that the head-of-the-line packet will belong to any particular mobile device with probability $\frac{1}{N}$. This throughput is shared equally between the N mobile devices.

Now suppose that the TCP data packet throughput with the two-node model is Θ. Then

$$n_\theta = \lfloor \frac{\Theta}{\theta} \rfloor$$

where the $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Use NetSim to verify that for an 11Mbps Wi-Fi speed, with RTS/CTS enabled the total TCP throughput is 3.4 Mbps. If $\theta = 0.65\ Mbps$, then $n_\theta = \lfloor \frac{3.4}{0.65} \rfloor = 5$. In this example, if $N = 5$ the download throughput obtained by each of them will be $0.68 Mbps$, but if one more downloading device is added then each will get a throughput less than $\theta = 0.65\ Mbps$. We say that the capacity of this network for a target throughput of $0.65 Mbps$ is 5.

5.3.4 Procedure

Open NetSim and click on **Experiments>Internetworks> Wi-Fi> How many downloads can a Wi-Fi access point simultaneously handle?** then click on the tile in the middle panel to load the example as shown in below Figure 5-15.

Figure 5-15: List of scenarios for the example of How many downloads can a Wi-Fi access point simultaneously handle

NetSim UI displays the configuration file corresponding to this experiment Figure 5-16.
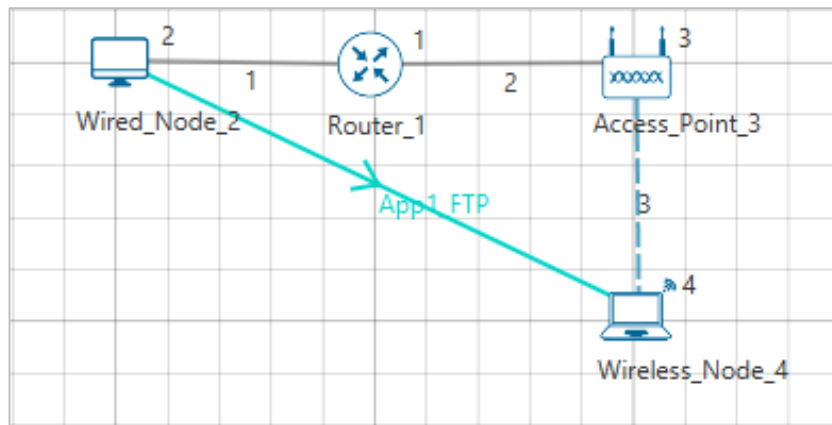
Figure 5-16: Network set up for studying the WiFi Network with Single TCP Download

5.3.5 Procedure

1-WN-1AP

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in the NetSim GUI comprising of 1 Wired Node, 1 Wireless Node, 1 Access Point, and 1 Router in the **"Internetworks"** Network Library.

Step 2: In the Interface (WIRELESS) > Data Link Layer Properties of Wireless Node 4 and Access Point, Short Retry Limit was set to 7, Long Retry Limit was set to 4 and RTS Threshold was set to 1000 bytes. Medium-Access-Protocol is set to DCF Figure 5-17.
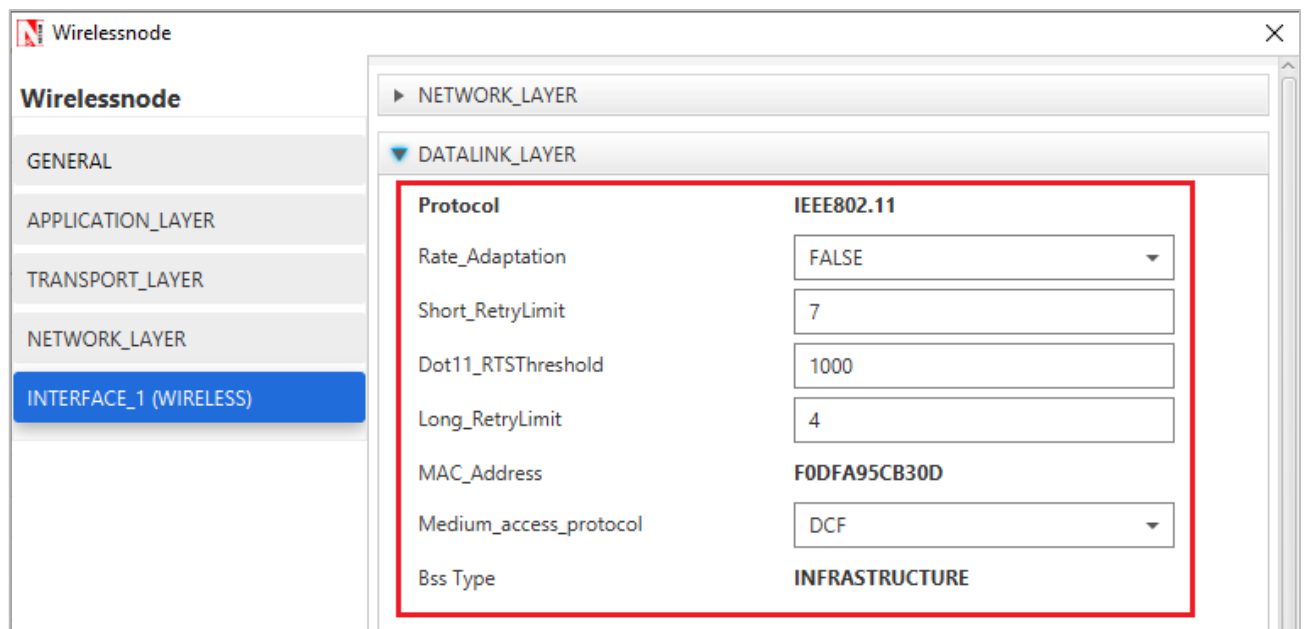

Figure 5-17: Data Link Layer Properties

Step 3: Right-click the link ID (of a wired link) and select Properties to access the link's properties. The Link Properties are set according to the values given in the below.

Wired Link	
Max Uplink Speed (Mbps)	100
Max Downlink Speed (Mbps)	100
Uplink BER	0
Downlink BER	0
Uplink Propagation Delay (µs)	0
Downlink Propagation Delay (µs)	0

Wireless Link	
Channel Characteristics	No path loss

Table 5-10: Detailed Wired/Wireless Links Properties

Step 4: Right click on **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

An FTP Application is generated from Wired Node 2 i.e., Source to Wireless Node 4 i.e. Destination with File Size set to 10,000,000 Bytes and Inter Arrival Time set to 20 s.

Step 5: Enable the plots, run the Simulation for 15 Seconds, and note down the throughput.

5-WN-1AP

The following changes in settings are done from the previous sample:

Step 1: The number of Wireless Nodes is increased to 5 and FTP applications are generated from Wired Node 2 to each of the Wireless Nodes as shown below Figure 5-18.
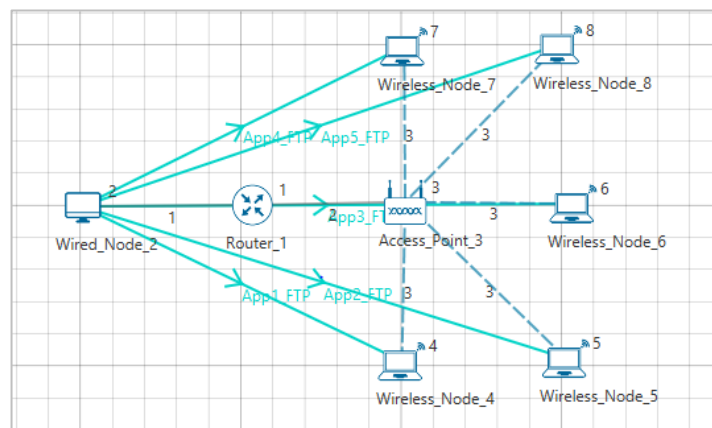
No. of wireless nodes = 5

Figure 5-18: WiFi Network with Multiple TCP Download

Application Properties

Properties	App1	App2	App3	App4	App5
Application Type	FTP	FTP	FTP	FTP	FTP
Source Id	2	2	2	2	2
Destination Id	4	5	6	7	8
File size (Bytes)	10,000,	10,000,	10,000,	10,000,	10,000,
File Inter arrival	20 s	20 s	20 s	20 s	20 s

Table 5-11: Detailed Application properties

Step 2: Enable the plots, run the Simulation for 15 Seconds, and note down the throughput.

5.3.6 Measurements and Output

Aggregated download throughput with different values of N (wireless nodes) is shown below Table 5-12.

$$Throughput\ Per\ Device\ (Mbps) = \frac{Sum\ of\ throughpots\ (Mbps)}{Number\ of\ Devices}$$

Samples Name	Number of Devices	Sum of throughputs (Mbps)	Throughput Per Device (Mbps)
1-WN-1AP	1	3.55	3.55
5-WN-1AP	5	3.55	0.72
10-WN-1AP	10	3.55	0.35
15-WN-1AP	15	3.54	0.23
20-WN-1AP	20	3.53	0.17
25-WN-1AP	25	3.48	0.13

Table 5-12: Aggregated download throughput for different number of wireless nodes
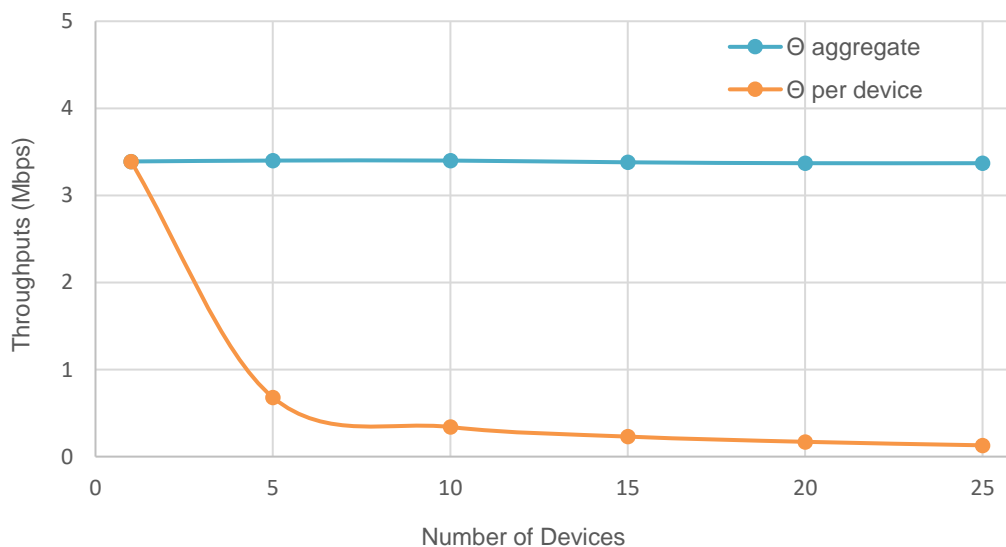
Plot

Figure 5-19: Plot of Number of devices vs. Throughputs (Mbps)

NOTE: In the referred paper we see that, a throughput value for 11 Mbps WLAN is 3.8 Mbps. Please note that this is the aggregate PHY throughput of the AP. However, in NetSim, we are calculating the total Application throughput.

To derive the PHY layer throughput from the APP layer throughput, we need to add overheads of all layers Table 5-13.

Layer	Overhead (Bytes)
Transport Layer	20
Network Layer	20
MAC Layer	40
PHY layer	48µs = (11*48)/8 = 66
Total Overhead	146

Table 5-13: Overhead of different layers

$$PHY_{Throughput} = APP_{Throughput} * \frac{1606}{1460} = \frac{3.42 * 1606}{1460} = 3.76\ Mbps$$

5.3.7 Observations

We see that as the number of devices increase, the aggregate (combined) throughput remains constant, whereas the throughput per user decreases.

As discussed earlier, our goal was to identify that if during the download, a user expects to get a throughput of at least θ bytes per second, what is the maximum number of users (say, n_θ)?

If we set θ to be 650 Kbps, then we see that from the output table that the maximum number of users who can simultaneously download files is 5 (n_θ)

5.3.8 Reference Documents

1. Analytical models for capacity estimation of IEEE 802.11 WLANs using DCF for internet applications. George Kuriakose, Sri Harsha, Anurag Kumar, Vinod Sharma

5.4 Multi-AP Wi-Fi Networks: Channel Allocation (Level 2)

5.4.1 Introduction

A single Wi-Fi Access Point (AP) can connect laptops and other devices that are a few meters distance from the AP, the actual coverage depending on the propagation characteristics of the building in which the Wi-Fi network is deployed. Thus, for large office buildings, apartment complexes, etc., a single AP does not suffice, and multiple APs need to be installed, each covering a part of the building. We will focus on 2.4GHz and 5GHz systems. In each of these systems the available bandwidth is organized into channels, with each AP being assigned to one of the channels. For example, 2.4GHz Wi-Fi systems operate in the band 2401MHz to 2495MHz, which has 14 overlapping channels each of 22MHz. There are 3 nonoverlapping channels, namely, Channels 1, 6, and 11, which are centered at 2412MHz, 2437MHz, and 2462MHz. Evidently, if neighboring APs are assigned to the same channel or overlapping channels they will interfere, thereby leading to poor performance. On the other hand, since there are only three nonoverlapping channels, some care must be taken in assigning channels to APs so that nearby APs have nonoverlapping channels, whereas APs that are far apart can use the same or overlapping channels.

In this experiment we will understand some basic issues that arise in multi-AP networks, particularly with attention to channel allocation to the APs.

5.4.2 Network Setup

Open NetSim and click on **Experiments>Internetworks> Wi-Fi> Multi AP Wi-Fi Networks Channel Allocation> APs on the Same channel** then click on the tile in the middle panel to load the example as shown in below
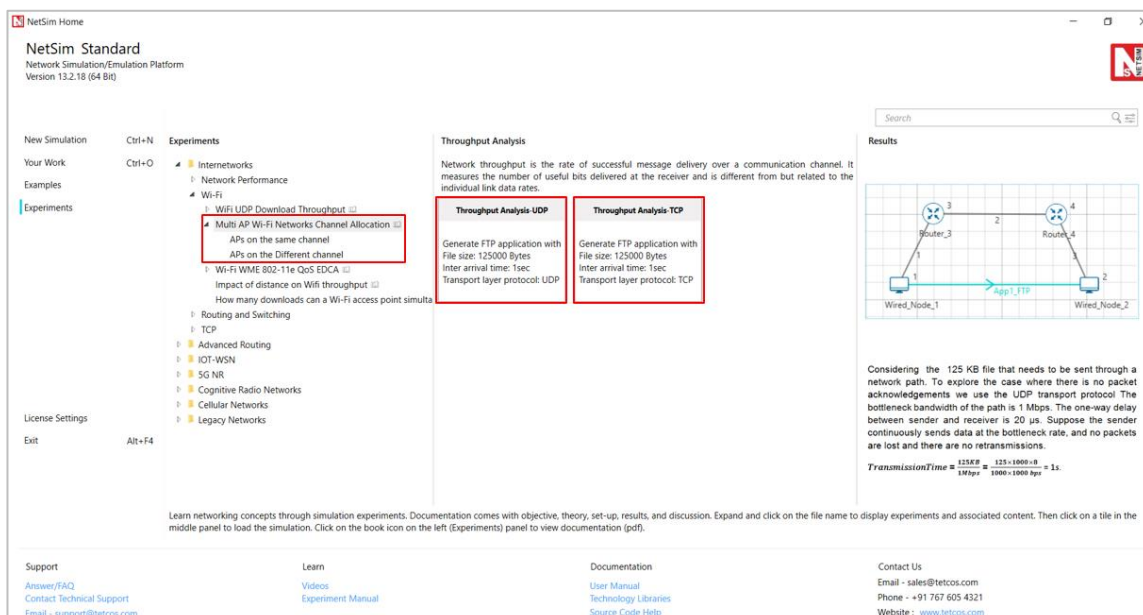

Figure 5-20: List of scenarios for the example of Multi AP Wi-Fi Networks Channel Allocation

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 5-21.
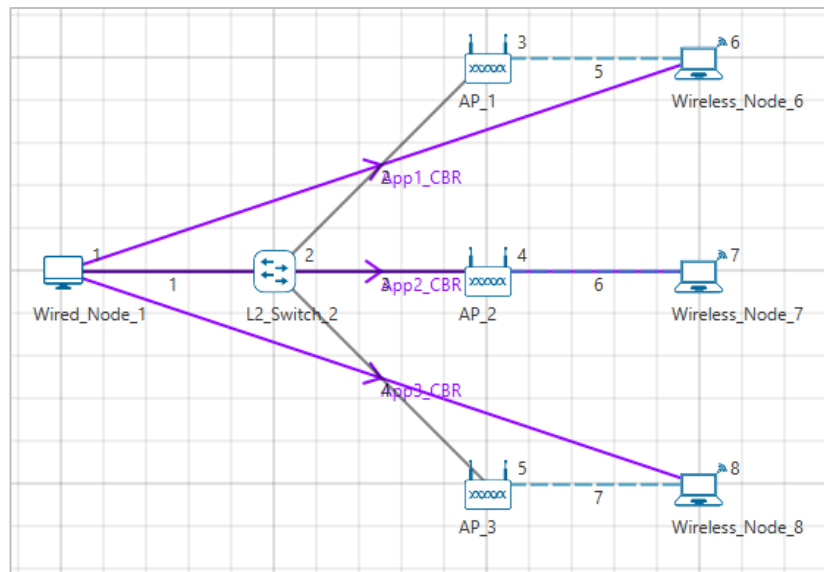
APs in the same channel

Figure 5-21: Network set up for studying the Multiple APs-WiFi Networks with APs in same Channel - Interfering-I

Interfering-I: The following set of procedures were done to generate this sample:

Step 1: Environment Grid length: 50m x 50m.

Step 2: A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 1 L2 Switch, 3 Wireless Nodes and 3 Access Points in the **"Internetworks"** Network Library.

Step 3: The device positions are set as per the table given below Table 5-14.

General Properties		
Device Name	**X / Lon**	**Y / Lat**
AP_1	15	5
AP_2	15	10
AP_3	15	15
Wireless_Node_6	20	5
Wireless_Node_7	20	10
Wireless_Node_8	20	15

Table 5-14: Device positions for APs-STA - Interfering-I

Step 4: In the INTERFACE (WIRELESS) > PHYSICAL LAYER Properties of all the Wireless Nodes and Access Points, the Protocol Standard is set to IEEE 802.11 b. In the INTERFACE (WIRELESS) > DATALINK LAYER Properties of all the Wireless Nodes and Access Points, Medium Access Protocol is set to DCF.

Step 5: Right-click the link ID (of a wired link) and select Properties to access the link's properties. For all the Wired Links, Bit Error Rate and Propagation Delay is set to 0.

Step 6: The Wireless Link Properties are set according to the values given in the below Table 5-15.

Channel Characteristics	PATH LOSS ONLY
Path Loss Model	LOG DISTANCE
Path Loss Exponent	3.5

Table 5-15: Wireless Link Properties

Step 7: Right click on the Application Flow App1 CBR and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 1 i.e., Source to Wireless Node 6 i.e., Destination with Packet Size set to 1460 Bytes and Inter Arrival Time set to 1168μs.

Transport Protocol is set to **UDP** instead of TCP.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 10 Mbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps)\ =\ Packet\ Size\ (Bytes)\ *\ 8/Interarrival\ time\ (μs)$$

Similarly, two more CBR applications are generated from Wired Node 1 to Wireless Node 7 and Wired Node 1 to Wireless Node 8.

Step 8: Plots is enabled in NetSim GUI. Run the Simulation for 10 Seconds and note down the throughput.

No Interfering: The following changes in settings are done from the previous sample:

Step 1: Before we start designing the network scenario, the Grid Length is set to 1000 meters. This can be set by choosing the Menu **Options>Change Grid/Map settings>Grid/Map settings** from the GUI.

Step 2: From the previous sample, we have removed App2 CBR (i.e., from Wired Node1 to Wireless Node7), set distance between the other 2 Access Points (AP 1 and AP 3) as 400m and distance between APs and Wireless nodes as 10m as shown below Figure 5-22.
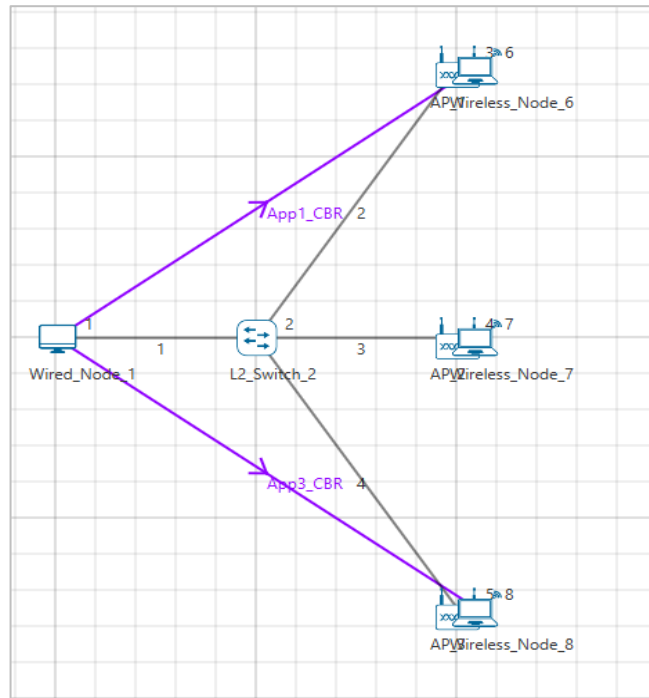
Figure 5-22: Network set up for studying the Multiple APs-WiFi Networks with APs in same Channel No-Interfering

Step 3: The device positions are set according to the table given below Table 5-16.

General Properties		
Device Name	**X / Lon**	**Y / Lat**
AP_1	400	0
AP_2	400	200
AP_3	400	400
Wireless_Node_6	410	0
Wireless_Node_7	410	200
Wireless_Node_8	410	400

Table 5-16: Device positions for APs-STA – No Interfering

Step 4: Plots is enabled in NetSim GUI. Run the Simulation for 10 Seconds and note down the throughput.

Interfering-II: The following changes in settings are done from the previous sample:

Step 1: From the previous sample, Add App2 CBR (i.e., from Wired Node1 to Wireless Node7), The distance between the Access Points (AP 1 and AP 3) is set to 400m and distance between APs and Wireless nodes as 10m as shown below Figure 5-23.
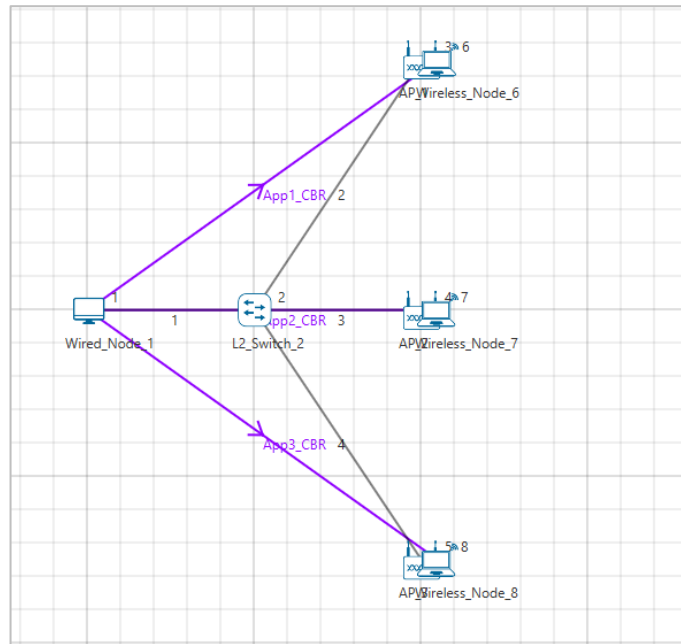
Figure 5-23: Network set up for studying the Multiple APs-WiFi Networks with APs in same Channel Interfering-II

Step 2: The device positions are set according to the table given below Table 5-17.

General Properties		
Device Name	**X / Lon**	**Y / Lat**
AP_1	400	120
AP_2	400	200
AP_3	400	280
Wireless_Node_6	420	120
Wireless_Node_7	420	200
Wireless_Node_8	420	280

Table 5-17: Device positions for APs-STA - Interfering-II

Step 3: Plots is enabled in NetSim GUI. Run the Simulation for 10 Seconds and note down the throughput.

Interfering-III: The following changes in settings are done from the previous sample:

Step 1: From the previous sample, we have removed App1 CBR (i.e., from Wired Node 1 to Wireless Node 6), set distance between the other 2 Access Points (AP 2 and AP 3) as 200m and distance between APs and Wireless nodes as 10m as shown below Figure 5-24.
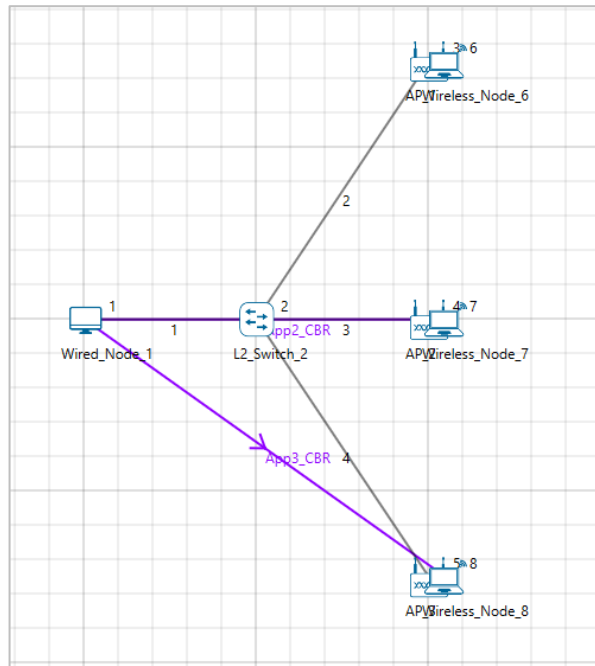
Figure 5-24: Network set up for studying the Multiple APs-WiFi Networks with APs in same Channel Interfering - III

Step 2: The device positions are set according to the table given below Table 5-18.

General Properties		
Device Name	**X / Lon**	**Y / Lat**
AP_1	400	150
AP_2	400	200
AP_3	400	250
Wireless_Node_6	420	150
Wireless_Node_7	420	200
Wireless_Node_8	420	250

Table 5-18: Device positions for APs-STA – Interfering III

Step 3: Plots is enabled in NetSim GUI. Run the Simulation for 10 Seconds and note down the throughput.

No Interfering-I: The following changes in settings are done from the previous sample:

Step 1: From **Interfering-II**, we have removed first, and third applications as shown below Figure 5-25.
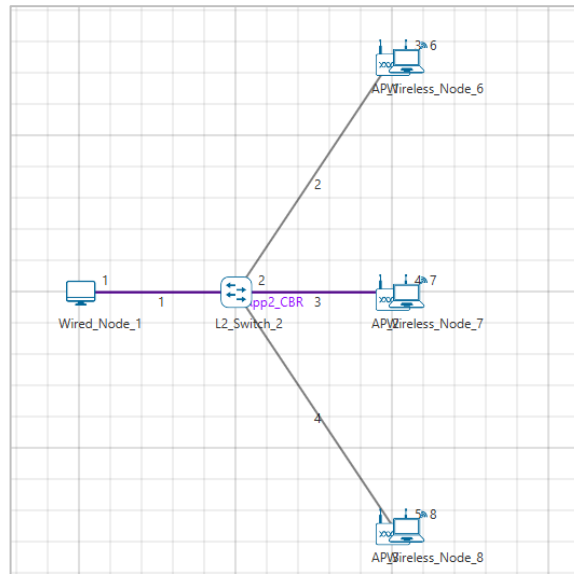
Figure 5-25: Network set up for studying the Multiple APs-WiFi Networks with APs in same Channel No Interfering-I

Step 2: The device positions are set according to the table given below Table 5-19.

General Properties		
Device Name	**X / Lon**	**Y / Lat**
AP_1	400	0
AP_2	400	200
AP_3	400	400
Wireless_Node_6	410	0
Wireless_Node_7	410	200
Wireless_Node_8	410	400

Table 5-19: Device positions for APs-STA - No Interfering-I

Step 3: Plots is enabled in NetSim GUI. Run the Simulation for 10 Seconds and note down the throughput.

APs on the different channel: The following changes in settings are done from the previous sample:

Step 1: From previous sample, we have changed standard channel to **11_2462** under INTERFACE (WIRELESS) > DATALINK LAYER Properties of AP 2.

Step 2: Plots is enabled in NetSim GUI. Run the Simulation for 10 Seconds and note down the throughput.

5.4.3 Output

After running simulation, check throughput in Application metrics as shown in the below screenshot Figure 5-26.

Figure 5-26: Application Metrics Table in Result window

Sample	Throughput (Mbps)		
	AP_1	AP_2	AP_3
All APs in the same channel			
Interfering-I	2.09	2.06	2.00
No Interfering	5.94	N/A	5.92
Interfering-II	4.71	0.13	4.67
Interfering-III	N/A	3.08	3.05
No Interfering-I	N/A	5.92	N/A
Each AP in a different nonoverlapping channel			
Different channel	5.94	5.92	5.92

Table 5-20: Throughput for APs on the Same and different Channels

NOTE: *Please refer "Wi-Fi UDP Download Throughput" experiment for theoretical WLAN throughput calculations in NetSim Experiment Manual.*

5.4.4 Discussion

We recall that each AP is associated with one station (STA; e.g., a laptop). All the APs are connected to the same server which is sending separate UDP packet streams to each of the STAs via the corresponding AP. The packet transmission rate from the server is large enough so that the AP queue in permanently backlogged, i.e., the rate at which the server transmits packets is larger than the rate at which the AP can empty the packet queue.

5.4.4.1 All APs in the same channel

Interfering-I: All the APs and their associated STAs are close together, so that all devices (APs and STAs) can sense every other device.

- The table shows that all the AP-STA links achieve the same UDP throughput. This is because all the AP-STA links are equivalent (since all interfere with each other), and only one can be active at one time. The throughput for this scenario can be predicted from the analysis in Section 7.4 of the book *Wireless Networking* by *Anurag Kumar, D. Manjunath and Joy Kuri.*

No Interfering: AP1 and AP3 are close to their associated STAs but are 400m apart. The link from AP2 to its STA is half-way between the other two APs and is not carrying any traffic.

- The table shows that both the links from AP1 and AP3 to their respective STAs carry the same throughput, of 5.94Mbps and 5.92Mbps. These are also the throughputs that each link would have if the other was not present, indicating that the two links are far enough apart that they do not interfere.

Interfering-II: This is the same scenario as **No Interfering**, but the AP2-STA link is now carrying traffic.

- We find that, in comparison with **No Interfering**, the AP1-STA and AP3-STA carry slightly lower throughputs of about 5.21Mbps, whereas the AP2-STA link carries a small throughput of 0.92Mbps. Comparing **Interfering-I** and **II** we conclude that in these networks there can be severe unfairness depending on the relative placement of the AP-STA links. In **Interfering-I**, all the links could sense each other, and each got a fair chance. In **Interfering-II**, we have what is called the "link-in-the-middle problem." The AP2-STA link is close enough to interfere with the AP1-STA link and the AP3-STA link, whereas the AP1-STA link and the AP3-STA link do not "see" each other. The AP2-STA link competes with the links on either side, whereas the other links compete only with the link in the center, which thereby gets suppressed in favour of the outer links.

Interfering-III: Here we stop the traffic to AP1 but send the traffic to the AP2-STA link and the AP3-STA link.

- The two active links interfere with each other, but the situation is symmetric between them (unlike in **Interfering-II**), and they obtain equal throughput. Again, the throughput obtained by these two links can be predicted by the analysis mentioned earlier in this section.

No Interfering-I: Now we send traffic only to AP2.

- The throughput is now 5.92Mbps, since the AP2-STA link can transmit without interference; there are no collisions. The reason that this throughput is less than the sum of the two throughputs in **Interfering-III** is that the single link acting by itself, with all the attendant overheads, is unable to occupy the channel fully.

5.4.4.2 Each AP in a different nonoverlapping channel

There is only one case here. Having observed the various situations that arose in the previous subsection when all the APs are in the same channel, now we consider the case where all the AP-STA pairs are each on a different nonoverlapping channel. As expected, every AP-STA pair gets the same throughput as when they are alone on the network.

5.5 Wi-Fi Multimedia Extension (IEEE 802.11 EDCA) (Level 3)

5.5.1 Introduction

In this experiment, we will study an enhancement to WiFi that enables APs and STAs to handle various packet flows in such a way so as to provide differentiated quality of service (QoS) to these flows. In the original WiFi standard (IEEE 802.11 DCF), every device in the network has one output buffer (queue) for all packets to be transmitted onto the wireless channel. The consequence of this would be that a packet stream with strict delivery constraints and another will relatively loose delivery objectives are queued in the same output buffer at every device. Each such queue is scheduled by the DCF CSMA/CA (see the experiment "WiFi: UDP Download Throughput"), and when a queue gets its transmission opportunity the first (head-of-the-line (HOL)) packet is transmitted. This might result in packets with strict delivery constraints being kept waiting, while other, less urgent, packets get transmitted.

For example, an interactive voice call might have 200-byte packets being transmitted periodically, with a period of 20 ms. Ideally, for perfect voice playout at the receiver, this voice stream must arrive exactly as it was transmitted: every 200-byte packet must arrive, and with the gaps of 20 ms intact. If the voice packets are delayed excessively, or if the delay is highly variable, the playout is affected, and voice quality (and speaker interaction) is affected. On the other hand, TCP controlled file transfers can adapt to network delay and delay variability. Evidently, the solution is to create multiple output buffers in each device, of different transmit priorities, queue the more urgent packets in higher priority buffers, and create a mechanism for preferential transmission of the packets with tighter QoS requirements.

In this experiment we will study the EDCAF mechanism, an extension to DCF, which implements service differentiation in WiFi.

5.5.2 EDCAF: Access Categories

In the year 2005, the standard IEEE 802.11e (EDCAF-Enhanced Distributed Channel Access Function) was introduced, with the above issues in mind. In EDCAF, there are four Access Categories (AC0, AC1, AC2, and AC3), with AC3 being the highest priority and AC0 being the lowest. The assignment of application usage to these ACs was.

Access Category	Application	Example
AC0	Background	Background print job
AC1	Best Effort	TCP File Transfer
AC2	Video	Video Conference Video

AC3	Voice	Video Conference Voice

Table 5-21: EDCA 4 access categories for 802.11 both AP and STA

Now, if a device (an AP or a STA) is sending interactive voice packets then these packets can be queued in the AC3 buffer of the device, whereas packets of a simultaneous TCP file transfer can be queued in the AC1 buffer. The human brain is less sensitive to video packet losses, and delays in the rendering of video frames (than the human hearing system is of voice corruption), hence video is given a priority between voice and TCP. The lowest category, AC0, can be used for any application whose packets need to just be delivered, without any well-defined quality of service, for example, a low urgency bulk printing service.

Having created buffers into which the various priority packets are queued, a mechanism is needed to schedule transmissions from these buffers so that service differentiation is achieved. Ideally, strict priority service could be enforced, i.e., assuming that there is only AC3 and AC2 traffic in the network, if any device has a nonempty AC3 buffer, all packets from AC3 category should be served before any AC2 traffic is served. Further, ideally, the video and the TCP file transfers could have been assigned a guaranteed service rate, to meet their QoS requirements. Such strict priorities and guaranteed service would belong to the concept of Integrated Services. However, the IEEE 802.11 wireless access mechanism is distributed, and there is no central entity that has the instantaneous state of all the buffers in all the devices. Hence, strict priority or a guaranteed service rate is not possible to. Instead, the IEEE 802.11 series of standards adopted EDCAF (an extension to DCF) for scheduling the service of the access category queues at the contending devices. The EDCAF mechanism achieves Differentiated Services. How does the MAC layer in a device know which access category buffer to queue a packet in? This is achieved by the corresponding application using the DSCP (Differentiated Service Code Point) field in the IPV4 header to indicate the Differentiated Services class of the packet. The MAC layer of the device would have a table that maps the DSCP value to the access category.

5.5.3 EDCAF: Service Differentiation Mechanisms

We begin by recalling how the basic EDCF works, since EDCAF is built as an extension of EDCF. In EDCF, after handling the HOL packet in its (single) buffer (which could result in the packet being transmitted successfully or discarded (due to exceeding the maximum number of reattempts)), a device waits for DIFS, samples an initial back-off for the next packet in the buffer, and begins to count down (at slot boundaries) until the back-off counter reaches zero, at which instant the first attempt for the next packet is made. A collision leads to a new back-off being sampled from a distribution with a larger mean. All nodes behave in exactly the same manner, thus getting opportunities to transmit packets whenever their back-off counters reach zero. Thus,

all devices (STAs and APs) have the same behavior (statistically), and there is no service differentiation.

Now consider an AP with AC3 packets to be transmitted (say, voice), and an STA with AC1 packets (say, TCP). After the AP transmits a voice packet, in EDCAF, the AP's MAC waits for AIFS3 (Arbitration Inter Frame Space for Category 3) which is 2 slots, and samples a back-off from a uniform distribution over 1 slot to 2^7 slots. On the other hand, at this point the STA waits for AIFS1, which is 3 slots. In addition, after a TCP packet has been transmitted (or dropped) the STA samples a back-off for the next packet from a uniform distribution over 1 slot to 2^{31} slots. Thus, the HOL packet waiting in the AC3 buffer has two advantages over the HOL packet waiting in the AC1 buffer:

i.   The back-off counter of the AC3 category starts counting down one slot earlier than the AC1 category.
ii. The back-off counters are smaller for AC3 than for AC1.

These two mechanisms conspire to differentiate the wireless access service in favour of AC3. Note that we do not get strict priority. For example, if a voice packet has been transmitted by the AP, and after AIFS3, the back-off sampled is 3 slots, whereas the residual back-off of the TCP transfer (at the STA) was 2 slots, then the TCP packet will be transmitted next. However, the service differentiation is significant as the simulation results from NetSim will demonstrate later in this chapter. The following is the table of all EDCAF parameters as specified by the standard.

Access Category	CWmin	CWmax	AIFSN	Max TXOP (μs)
Background (AC_BK)	31	1023	7	3264
Best Effort (AC_BE)	31	1023	3	3264
Video (AC_VI)	15	31	2	6016
Voice (AC_VO)	7	15	2	3264

Table 5-22: EDCA access parameters for 802.11 b for both AP and STA

5.5.4 The Experimental Plan

- Voice over DCF: We first want to understand the limitation when carrying interactive voice over DCF.
  - With this in mind, we will set up several full-duplex voice calls between several STAs and the wired network, one such call for each STA. Each full-duplex voice call will be modelled by a periodic stream of 200-byte UDP packets (160B voice plus 40B of UDP/IP headers), generated at 20 ms intervals, from the STA to the wired network, and another such, independent stream from the wired network to the STA. We will increase the number of STAs, thereby increasing the number full-duplex voice calls, and will determine the number of calls that can be handled.

- o Then we will add on TCP controlled file transfer from the wired network to another STA. Due to reasons explained earlier in this chapter, the voice performance should degrade, leading to fewer calls being possible to handle along with the TCP transfer.
  - ▪ Voice over EDCAF: Next we repeat the above two experiments with the EDCAF mechanism enabled. We should find that it is possible to maintain a substantial number of voice conversations even while running the TCP file transfer. Next we will study what happens if the number of TCP file transfers is increased, the question being whether the number of voice conversations that can be handled gets affected.

5.5.5 Simulation Experiments to Study IEEE 802.11 EDCAF

Open NetSim and click on **Experiments> Internetworks> Wi-Fi> Wi Fi WME 802.11e QoS EDCA>DCF Voice Only** then click on the tile in the middle panel to load the example as shown in below Figure 5-27.

Figure 5-27: List of scenarios for the example of Wi Fi WME 802.11e QoS EDCA
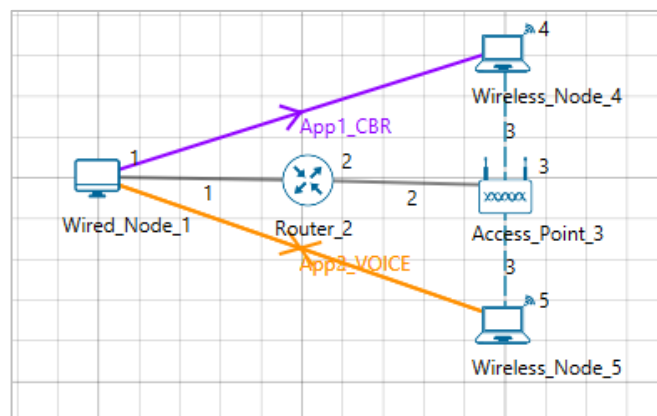
Case 1: DCF with full-duplex voice calls only

Network Scenario
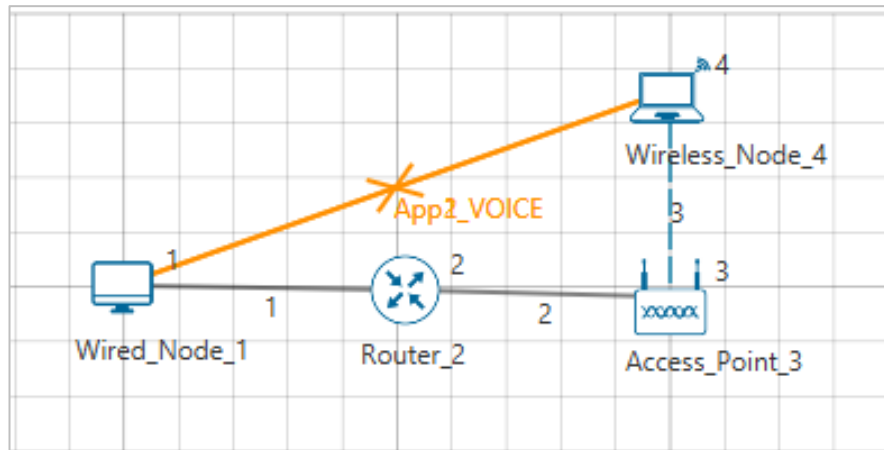
Figure 5-28: Network set up for studying the DCF with full-duplex voice calls only

Network Setup

- AP and STA operate in DCF.
- There is no error in all wired and wireless links.

Applications

- Two-way UDP voice calls from Node to Wireless_Node_i, with i being incremented. Each voice calls in NetSim is modelled as 2 one-way voice applications. The voice modelling option in NetSim UI currently allows transfer in one direction only. Hence, we model a two-way voice call as 2 one-way voice applications.
  o $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
  o Each voice call runs G.711 at 64 Kbps.

Case 2: DCF with full-duplex voice calls and a single TCP download

Network Scenario

Figure 5-29: Network set up for studying the DCF with full-duplex voice calls and a single TCP download

Network Setup

- AP and STA operate in DCF.
- There is no error in all wired and wireless links.

Applications

- TCP full buffer (or saturated case) download from N to WN_1. The saturation is generated by using a CBR application with Packet Size of 1460 Bytes and Inter packet arrival time of 1000 μs. Since this generation rate is higher than the Wi-fi link it rate as a saturated (full buffer) TCP flow is achieved. The reason for emulating a TCP download using a CBR session, is that a TCP file download would take a longer time to simulate.
- Voice calls from N to WN_{i+1} with i being incremented. Two-way UDP voice calls from Node to Wireless_Node_i. Each voice calls in NetSim is modelled as 2 one-way voice applications
  - $WN_{i+1} \rightarrow N \: and \: N \rightarrow WN_{i+1}. WN_i$ represent wireless node i, while N represents the wired node or remote host.
  - Each voice call runs G.711 at 64 Kbps.

Case 3: EDCAF with full-duplex voice calls only

Network Scenario
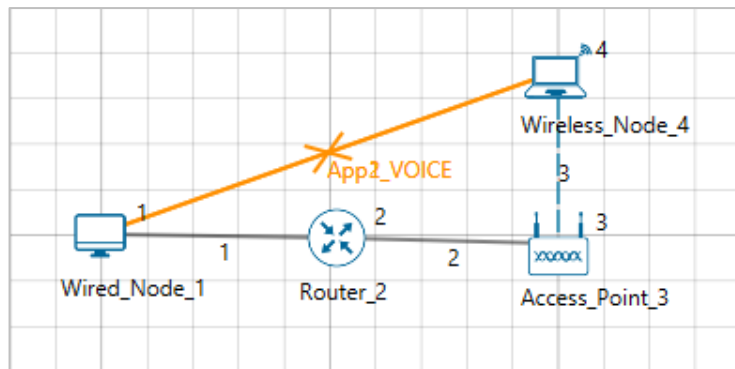
Figure 5-30: Network set up for studying the EDCAF with full-duplex voice calls only

Network Setup

- AP and STA operate in EDCAF, with EDCAF parameters set per reference paper
- There is no error in all wired and wireless links.

Applications

- Two-way UDP voice calls from Node to Wireless_Node_i, with i being incremented. Each voice calls in NetSim is modelled as 2 one-way voice applications. The voice modelling option in NetSim UI currently allows transfer in one direction only. Hence, we model a two-way voice call as 2 one-way voice applications.
  - $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
  - Each voice call runs G.711 at 64 Kbps.

Case 4: EDCAF with full-duplex voice calls and a single TCP download
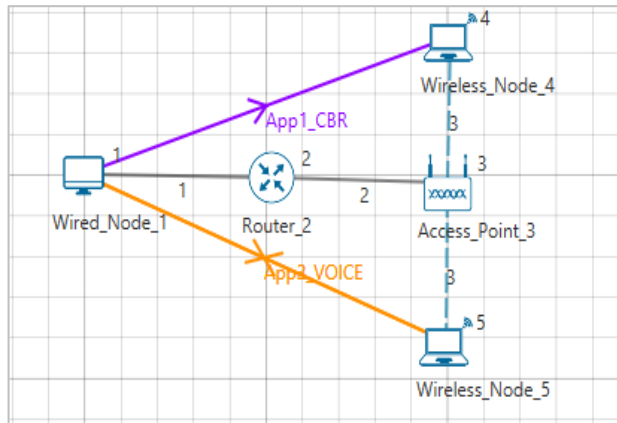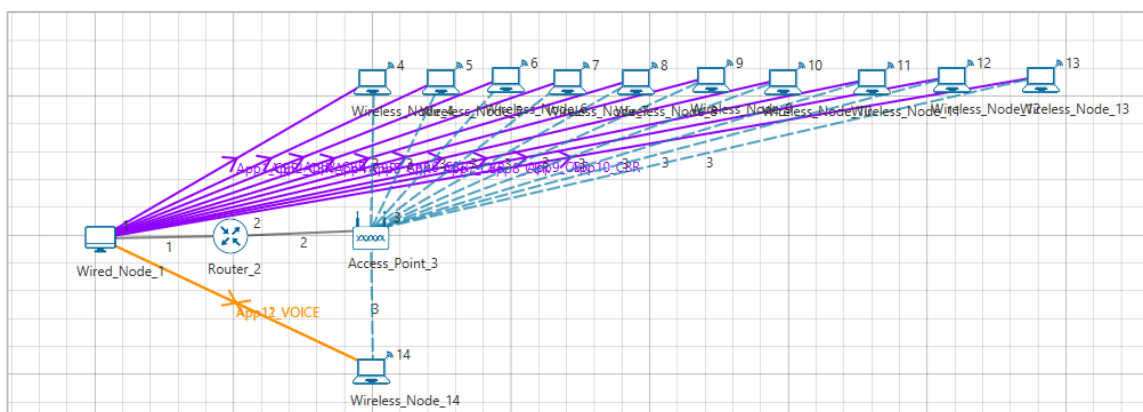
Network Scenario

Figure 5-31: Network set up for studying the EDCAF with full-duplex voice calls and a single TCP download

Network Setup

- AP and STA operate in EDCAF, with EDCAF parameters set per reference paper. In the AP, TCP would be queued in AC_BE while Voice packets would be queued in AC_VO.
- There is no error in all wired and wireless links.

Applications

- TCP full buffer (or saturated case) download from N to WN_1. The saturation is generated by using a CBR application with Packet Size of 1460 Bytes and Inter packet arrival time of 1000 μs. Since this generation rate is higher than the Wi-fi link it rate as a saturated (full buffer) TCP flow is achieved.
- Voice calls from N to WN_{i+1} with i being incremented. Two-way UDP voice calls from Node to Wireless_Node_i. Each voice calls in NetSim is modelled as 2 one-way voice applications
  - $WN_{i+1} \rightarrow N$ and $N \rightarrow WN_{i+1}$. WN_i represent wireless node i, while N represents the wired node or remote host.
  - Each voice call runs G.711 at 64 Kbps.

Case 5: EDCAF with full-duplex voice calls and multiple TCP downloads

Network Scenario

Figure 5-32: Network set up for studying the EDCAF with full-duplex voice calls and multiple TCP downloads

Network Setup

- AP and STA operate in EDCAF mode.
- There is no error in all wired and wireless links.

Applications

- 10 TCP downloads from N to WN_1 through WN_{11}
  - TCP full buffer (or saturated case) download from $N \rightarrow WN$. The saturation is generated by using a CBR application with Packet Size of 1460 B and Inter packet arrival time of 1000 μs. Since this generation rate is higher than the Wi-fi link it rate as a saturated (full buffer) TCP flow is achieved.
- UDP Voice calls from N to WN_{11+i} with i being incremented.
- Each voice calls in NetSim is modelled as 2 one-way voice applications
  - $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
  - Each G.711 at 64 Kbps.

5.5.6 Simulation Results

Case 1: DCF with full-duplex voice calls only

- Tabulated separately for applications going $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
- A mean delay of 20,000 μs is considered as a threshold.
- For the case $N \rightarrow WN_i$ this threshold is crossed at 11 calls
- For the case $WN_i \rightarrow N$ this threshold is crossed at 20 calls

No of voice calls	Mean Delay Voice $N \rightarrow WN_i$	Mean Delay Voice $WN_i \rightarrow N$
1	1181.64	1119.24
2	2417.94	1666.23
3	3737.10	2117.73
4	5089.30	2558.37
5	6433.89	3000.71
6	7808.15	3426.20
7	9159.53	3929.14
8	10544.23	4383.35
9	12026.21	4851.28
10	13879.48	5459.24
11	4571867.59	6308.80
12	12197132.38	6889.13
13	18832275.60	7408.73
14	24246265.61	8200.14
15	29106781.12	9032.72
16	33825893.09	10071.78
17	37129900.58	11548.87
18	40666670.30	14032.98
19	43206845.02	16694.24
20	45890755.08	28593.24
21	45778456.09	74205.13
22	45570258.85	1517955.44

Table 5-23: DCF with full-duplex voice calls only. $N \rightarrow WN_i$ and $WN_i \rightarrow N$, where N represents the wired node and WN_i represents the i-th wireless node. Therefore $N \rightarrow WN_i$ represents the flows from AP to STAs while $WN_i \rightarrow N$ represents the flows from STAs to AP.

* Mean Delay Voice $WN_i \rightarrow N$ = Average of the delay of all the applications flowing $WN_i \rightarrow N$

Case 2: DCF with full-duplex voice calls and single TCP download

- Tabulated separately for applications going $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
- A mean delay of 20,000 μs is considered as a threshold.
- For the case $N \rightarrow WN_i$ this threshold is crossed at 1 call itself

No of TCP Connections	No of voice calls	Mean Delay Voice $N \rightarrow WN_i$	Mean Delay Voice $WN_i \rightarrow N$
1	1	95704.39	2977.85
	2	140275.69	3320.34

Table 5-24: DCF with full-duplex voice calls and single TCP download $N \rightarrow WN_i$ and $WN_i \rightarrow N$

Case 3: EDCAF with full-duplex voice calls

- Tabulated separately for applications going $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
- A mean delay of 20,000 μs is considered as a threshold.
- For the case $N \rightarrow WN_i$ this threshold is crossed at 13 calls, for the case $WN_i \rightarrow N$ this threshold is crossed at 21 calls

No of voice calls	Mean Delay Voice $N \rightarrow WN_i$	Mean Delay Voice $WN_i \rightarrow N$
1	1000.70	704.67
2	1720.33	1575.46
3	2460.83	2464.80
4	3231.18	3356.51
5	4084.78	4274.84
6	5092.33	5068.09
7	6090.03	5872.13
8	7083.65	6648.27
9	8193.41	7373.58
10	9226.28	8115.11
11	10508.87	8671.12
12	11697.42	9477.25
13	460068.57	12297.35
14	495197.84	12778.37
15	498731.77	13294.44
16	499873.05	13932.10
17	498605.48	14600.35
18	497835.90	15401.32
19	494160.35	16329.19
20	488004.13	17798.12
21	492066.02	20362.62

Table 5-25: EDCAF with full-duplex voice calls $N \rightarrow WN_i$ and $WN_i \rightarrow N$

Case 4: EDCAF with full-duplex voice calls and single TCP download

- Tabulated separately for applications going $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
- A mean delay of 20,000 μs is considered as a threshold.
- For the case $N \rightarrow WN_i$ this threshold is crossed at 13 calls

No of TCP Connections	No of Voice calls	Mean Delay Voice $N \rightarrow WN_i$	Mean Delay Voice $WN_i \rightarrow N$
1	1	1643.79	1846.09
	2	2556.46	2877.60
	3	3384.93	3735.75
	4	4245.55	4614.23
	5	5225.55	5463.47
	6	6385.25	6227.89
	7	7495.18	6921.69
	8	8587.71	7605.80
	9	9706.95	8334.68
	10	10954.29	8994.38
	11	11398.15	9255.41
	12	11794.89	9499.69
	13	473299.48	12278.23
	14	495401.02	12840.93

Table 5-26: EDCAF with full-duplex voice calls and single TCP download $N \rightarrow WN_i$ and $WN_i \rightarrow N$

Case 5: EDCAF with full-duplex voice calls and multiple TCP downloads

- Tabulated separately for applications going $WN_i \rightarrow N$ and $N \rightarrow WN_i$. WN_i represent wireless node i, while N represents the wired node or remote host.
- A mean delay of 20,000 μs is considered as a threshold.
- For the case $N \rightarrow WN_i$ this threshold is crossed at 13 calls

No of TCP Connections	No of voice calls	Mean Delay Voice $N \rightarrow WN_i$	Mean delay Voice $WN_i \rightarrow N$
10	1	1667.21	1849.21
	2	2572.35	2901.66
	3	3394.76	3762.22
	4	4249.79	4631.62
	5	5256.49	5465.44
	6	6427.52	6240.32
	7	7498.21	6967.31
	8	8693.33	7646.78
	9	9851.28	8303.38
	10	11018.91	9035.10
	11	12459.80	9775.56
	12	16771.02	10708.19
	13	478500.49	12276.20
	14	496507.21	12833.58

Table 5-27: EDCAF with full-duplex voice calls and multiple TCP downloads $N \rightarrow WN_i$ and $WN_i \rightarrow N$

Comparison Charts
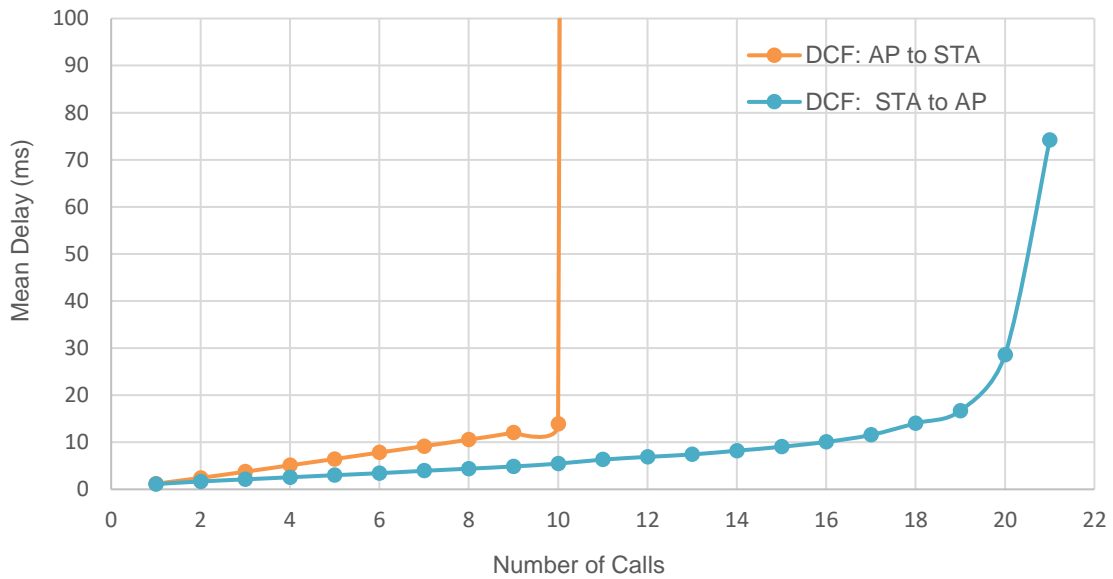
(a)
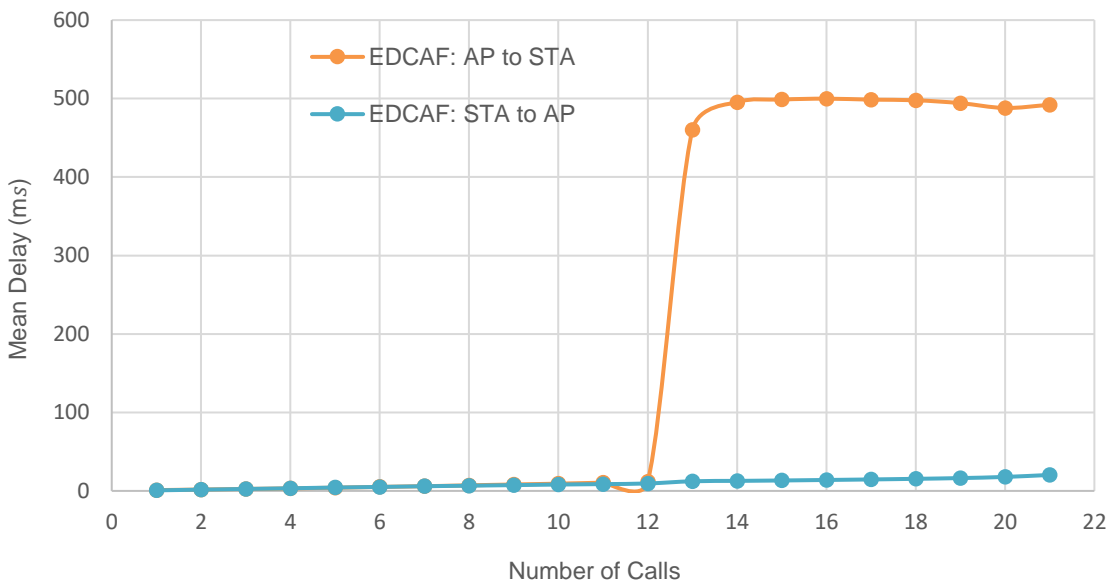
(b)

Figure 5-33: The plot of Mean Delay Vs. Number of Calls for DCF and EDCAF (a & b)

5.5.7 Discussion of the Simulation Results

5.5.7.1 Results for DCF

1. Observations

- Only voice calls: With one voice conversation, the mean packet delay for the wired-to-wireless (downlink) direction (i.e., these packets queue in the AP) is 1.18161 ms, whereas in the wireless-to-wired (uplink) direction (i.e., these packets queue in the STA) the mean packet delay is 1.11924 ms (we will, henceforth, report these delays in milliseconds and round-off to two decimal places). These mean delays increase as the number of voice

conversations increase. We notice that with 10 conversations the downlink mean packet delay is 13.69 ms, whereas the uplink packet delay is 5.47 ms. An increase of one more call result in the downlink mean packet delay becoming 4027.85 ms and the uplink mean packet delay being 6.28 ms.

- One TCP download to one STA from a wired node and increasing number of full-duplex voice calls: With just one voice call the mean delay is 116.00 ms in the downlink and 3.18 ms in the uplink. These values should be compared with 1.18 ms and 1.12 ms, respectively. Thus, with just one TCP connection, a single voice call experiences substantially larger mean delay.

2. Discussion

- With increasing number of voice calls (without any simultaneous TCP download) the dramatic change in the downlink delay, when the number of voice calls is increased from 10 to 11 is due to the downlink queue becoming unstable, i.e., the arrival rate of bits exceeds the rate at which the DCF wireless access mechanism can service the bits queued at the AP. The sudden transition from low delays to very high delays is typical of a queue going a stable regime to an unstable regime.

- It is interesting to note that at this transition point, the uplink delays have not increased as dramatically. In fact, in the uplink direction the transition from stability to instability appears in going from 22 calls to 23 calls. This difference in the downlink and uplink directions is because all the downlink voice packet streams are handled at one queue (the AP's WiFi buffer), with one contention process, whereas each uplink voice packet stream has its own buffer with its own contention process. If all the uplink voice streams had also been from one STA then the same phenomenon would have been observed in both directions.

- Next, we saw that with a single downlink TCP transfer the downlink mean delay of a single voice call is almost 100 times that without TCP. This occurs because the TCP transfer over a the local area network builds up a large window, most of which resides in the AP buffer. The TCP file transfer packets are large (about 1500 bytes). A single voice stream generates 200-byte packets at 20 ms intervals. The downlink voice packets see a very large buffer, due to the TCP packets queued in the AP buffer. It may be noted here, that with this kind of delay, even a single interactive voice call will not be supported.

Results for EDCAF

1. Observations

- With voice calls alone the transition in downlink delay occurs in going from 12 to 13 calls.
- With TCP downloads (1 or 10 downloads) the transition in downlink voice packet delay does not change as compared to without TCP

2. Discussion

- EDCAF creates different buffers for voice and for TCP file transfers (AC3 and AC1, respectively). The service differentiation mechanism between these buffers is described earlier in this chapter. The experimental results show that voices call performance is not seriously affected by the TCP controlled file transfers.
- As before, and for the same reasons, the voice capacity is limited by the service rendered to the AP buffers.

6 Internet of Things (IOT) and Wireless Sensor Networks

6.1 Cyber physical systems (CPS) and IoT – An Introduction (Level 2)

6.1.1 Introduction

Cyber Physical Systems (CPS) are systems that link the physical world (e.g., through sensors or actuators) with the virtual world of information processing. This does not just mean the convergence. Many systems can be categorized as CPS. Let us consider the example of a smart grids. On the demand side the various domestic appliances (of end users) constitute the physical components, and data of demand load are collected by smart meters. These smart meters connect the physical world to cyber space. The demand load data is transferred via two-way communication channels that are used to measure and control the physical components. On the cyber (cloud) side computations are carried out by the objectives of utility maximization and cost minimization. Based on this a suitable real-time electricity price is calculated.

Internet of Things (**IoT**) is a network of physical devices, vehicles, buildings and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. The IoT network allows objects to be sensed and/or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit.

IoT is the platform on which cyber physical systems run.
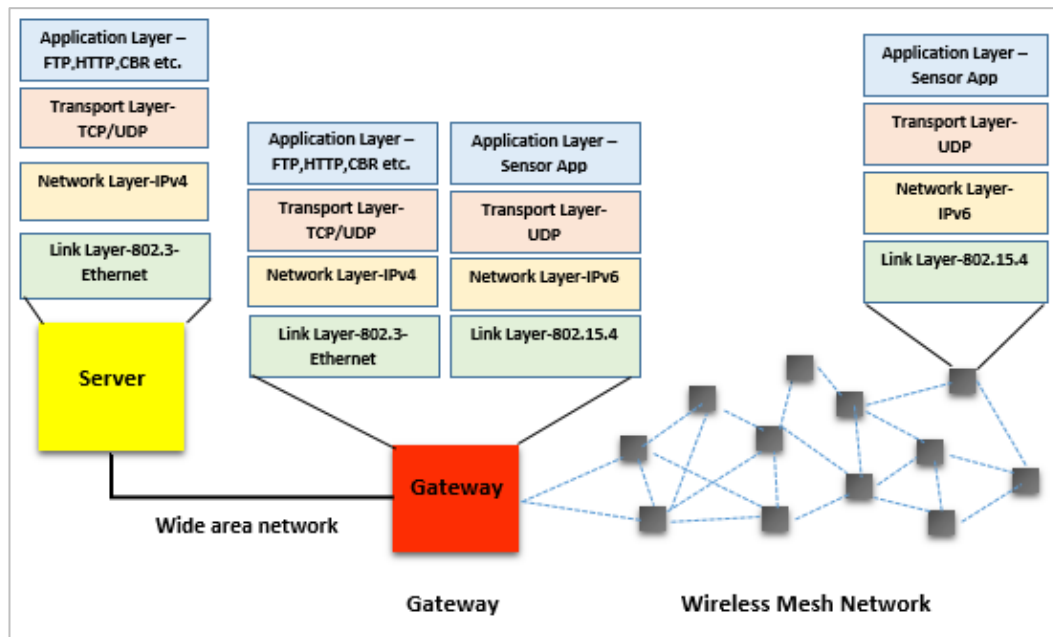
Figure 6-1: IOT Network Components and the TCP/IP stack running in the network devices

6.1.2 Components of IoT

1. **Sensors:** Sensors are used to detect physical phenomena such as light, heat, pressure, temperature, humidity etc. Sensors are regarded as a revolutionary information gathering method to build the information and communication system which will greatly improve the reliability and efficiency of infrastructure systems. It follows IPv6 addressing system. IP addresses are the backbone to the entire IoT ecosystem. IPv6's huge increase in address space is an important factor in the development of the Internet of Things.

2. **LowPAN Gateway:** These are the Gateways to Internet for all the things/devices that we want to interact with. Gateway help to bridge the internal network of sensor nodes with the external Internet i.e., it will collect the data from sensors and transmitting it to the internet infrastructure. A 6LowPAN Gateway will have 2 interfaces, one is Zigbee interface connected to sensors (follows 802.15.4 MAC and PHY) and the other is WAN interface connected to ROUTER.
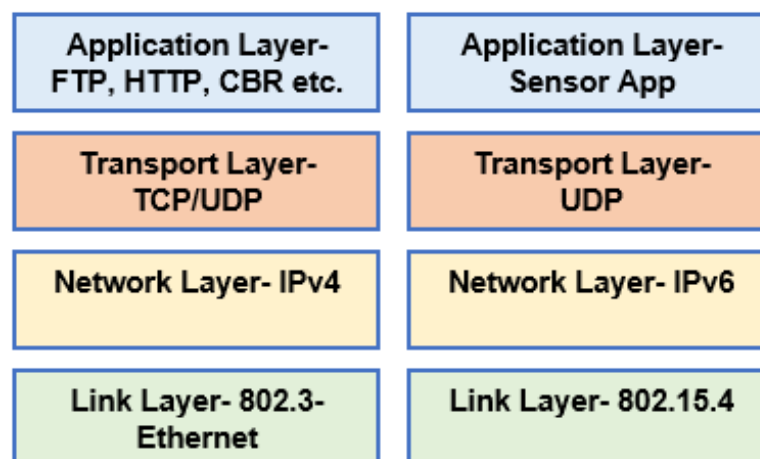
Figure 6-2: The 6LowPAN Gateway's TCP/IP Stack at the wired and wireless Interfaces

6LoWPAN is an acronym of IPv6 over Low Power Wireless Personal Area Network. The 6LoWPAN concept originated from the idea that "the Internet Protocol should be applied even to the smallest devices, and that low-power devices with limited processing capabilities should be able to participate on the Internet of Things.

6.1.3 Network Setup

Open NetSim and click on **Experiments> IOT-WSN> Introduction to cyber physical systems (CPS) and IoT> Multiple Sensor to Wired Node Sample** click on the tile in the middle panel to load the example as shown in below Figure 6-3.
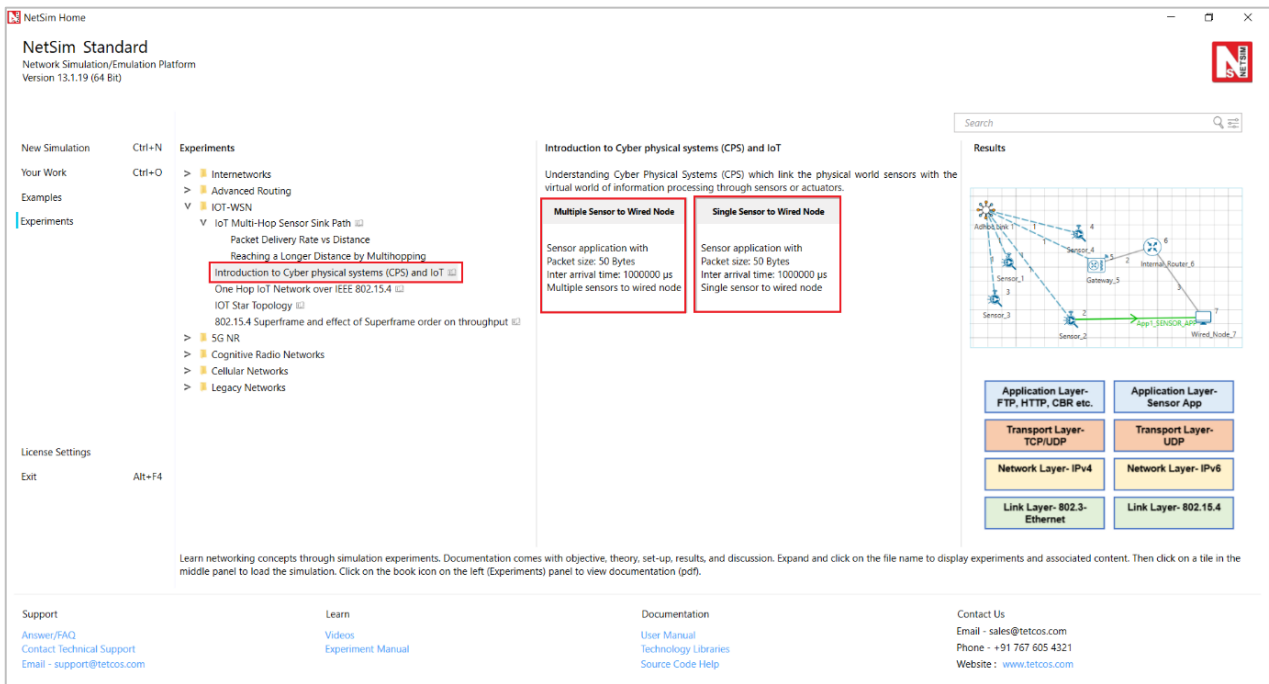
Figure 6-3: List of scenarios for the example of Introduction to cyber physical systems (CPS) and IoT

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-4.

Multiple Sensor to Wired Node Sample
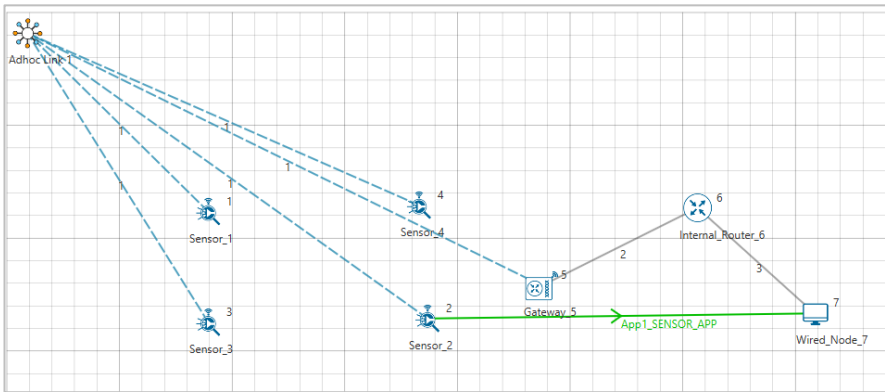
Figure 6-4: Network set up for studying the Multiple Sensor to Wired Node Sample

6.1.4 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 4 Wireless Sensors, 1 Gateway, 1 Router, and 1 Wired Node in the **"Internet of Things"** Network Library.

Step 2: Before we actually designed this network, in the **Fast Config Window** containing inputs for **Grid/ Map Settings and Sensor Placement**, the Grid Length and Side Length were set to 500 and 250 meters respectively, instead of the default 100 and 50 meters and we have chosen **Manually Via Click and Drop** option.

Step 3: The **Ad hoc Link** is used to link all the Sensors and the Gateway in an ad hoc basis.

The Ad hoc link properties is set to **NO PATHLOSS** for the channel characteristics.

Step 4: Right click on the Application Flow **App1 Sensor App** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A Sensor Application is generated from Wireless Sensor 2 i.e., Source to Wired Node 7 i.e., Destination with Packet Size remaining 50 Bytes and Inter Arrival Time remaining 1000000 μs.

Step 5: Enable the packet trace and plots. Run the Simulation for 100 Seconds.

6.1.5 Output for Multiple Sensor to Wired Node Sample
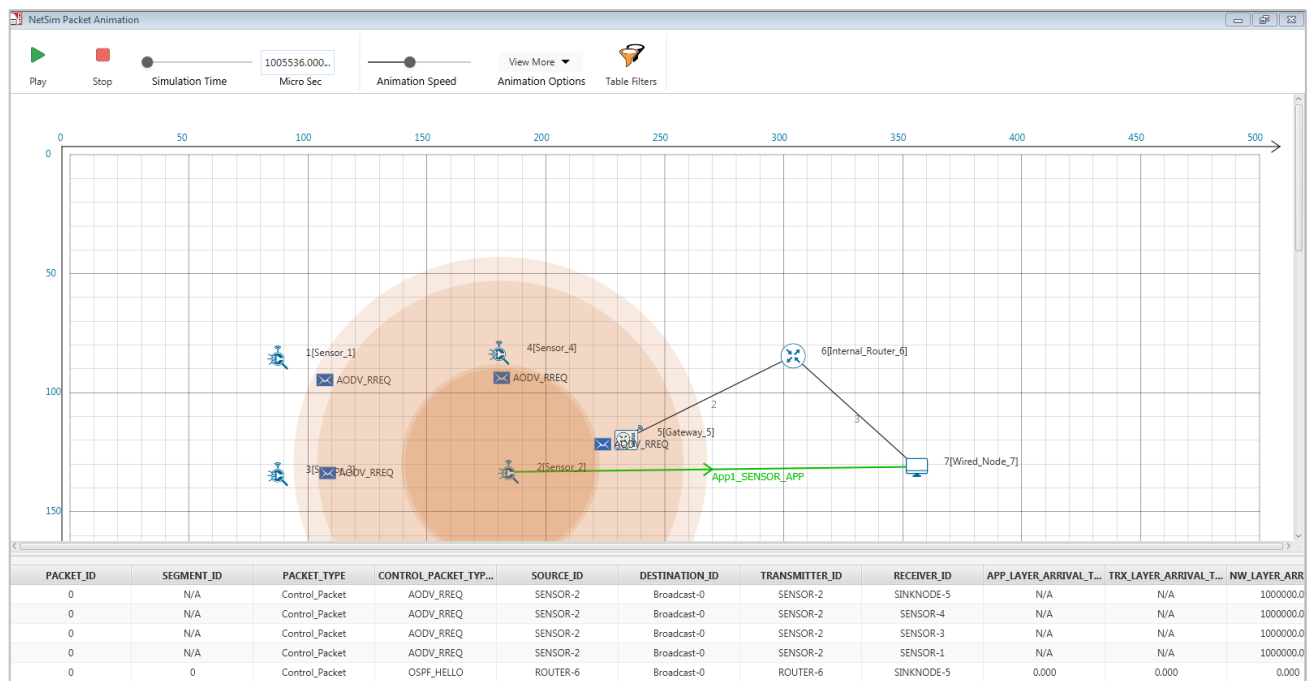
Figure 6-5: Animation Window

Single Sensor to Wired Node Sample
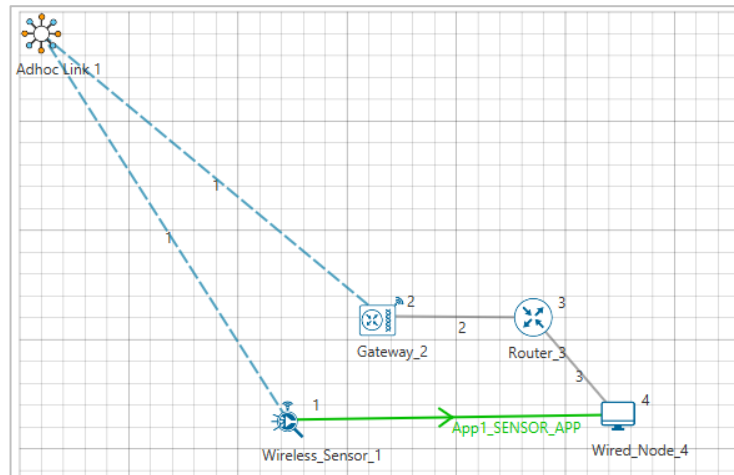
Figure 6-6: Network set up for studying the Single Sensor to Wired Node Sample

The following changes in settings are done from the previous sample:

Step 1: We have only one Sensor and the Sensor Application is generated between that Sensor and the Wired Node.

Step 2: Enable the plots and Packet trace.

Step 3: Run the Simulation for 10 Seconds.

6.1.6 Output for Single Sensor to Wired Node Sample

Users can understand how the IP addresses are changing from IPv6 to IPv4 and vice versa with the help of packet trace file.

After simulation, open packet trace and filter PACKET_TYPE to Sensing and observe the columns SOURCE_IP, DESTINATION_IP, GATEWAY_IP and NEXT_HOP_IP

> **SOURCE_IP** – source node IP

> **DESTINATION_IP** – gateway IP/destination IP

> **GATEWAY_IP** – IP of the device which is transmitting a packet.

> **NEXT_HOP_IP** – IP of the next hop

1. Sensor and 6_LOWPAN_Gateways 1st interface follows IPv6 addressing.
2. 6_LOWPAN_Gateways 2nd interface, Router and Wired Node follows IPv4 addressing.
3. From the screenshot below, users can identify the changing of IP addresses from source to destination.

Figure 6-7: Screenshot of packet trace showing IP addresses are changing from IPv6 to IPv4 and vice versa

6.2 One Hop IoT Network over IEEE 802.15.4 (Level 2)

6.2.1 Introduction

The concept of Cyber Physical Systems (CPS) over the Internet of Things (IoT) was explained in Experiment -19: Cyber physical systems (CPS) and IoT. In most situations, due to the practical difficulty of laying copper or optical cables to connect sensors and actuators, digital wireless communication has to be used. In such applications, since the energy available in the sensor and actuator devices is small, there is a need for keeping costs low, and the communication performance requirement (in terms of throughput and delay is limited) several wireless technologies have been developed, with the IEEE 802.15.4 standard being one of the early ones.

The IEEE Standard 802.15.4 defines PHY and MAC protocols for low-data-rate, low-power, and low-complexity short-range radio frequency (RF) digital transmissions.

In this experiment, we will study the simplest IEEE 802.15.4 network with one wireless node transmitting packets to an IEEE 802.15.4 receiver, from where the packets are carried over a high-speed wireline network to a compute server (where the sensor data would be analyzed).

6.2.2 The IEEE 802.15.4 PHY and MAC

We will study the IEEE 802.15.4 standard that works in the 2.4 GHz ISM band, in which there is an 80 MHz band on which 16 channels are defined, each of 2 MHz, with a channel separation of 5 MHz. Each IEEE 802.15.4 network works in one of these 2 MHz channels, utilizing spread spectrum communication over a chip-stream of 2 million chips per second. In this chip-stream, 32 successive chips constitute one symbol, thereby yielding 62,500 symbols per second (62.5 Ksps; $\frac{(2\times10^6)}{32} = 62,500$). Here, we observe that a symbol duration is $32 \times \frac{1}{2\times10^6} = 16\,\mu sec$. Binary signaling (OQPSK) is used over the chips, yielding 2^{32} possible sequences over a 32 chip symbol. Of these sequences, 16 are selected to encode 4 bits ($2^4 = 16$). The sequences are selected so

as to increase the probability of decoding in spite of symbol error. Thus, with 62.5 Ksps and 4 bits per symbol, the IEEE 802.15.4 PHY provides a raw bit rate of $62.5 \times 4 = 250$ Kbps.

Having described the IEEE 802.15.4 PHY, we now turn to the MAC, i.e., the protocol for sharing the bit rate of an IEEE 802.15.4 shared digital link. A version of the CSMA/CA mechanism is used for multiple access. When a node has a data packet to send, it initiates a random back-off with the first back-off period being sampled uniformly from 0 to ($2^{macminBE} - 1$), where $macminBE$ is a standard parameter. The back-off period is in slots, where a slot equals 20 symbol times, or $20 \times 16 = 320$ μsec. The node then performs a *Clear Channel Assessment* (CCA) to determine whether the channel is idle. A CCA essentially involves the node listening over 8 symbols times and integrating its received power. If the result exceeds a threshold, it is concluded that the channel is busy and CCA fails. If the CCA succeeds, the node does a Rx-to-Tx turn-around, which takes 12 symbol times and starts transmitting on the channel. The failure of the CCA starts a new back-off process with the back-off exponent raised by one, i.e., to macminBE+1, provided it is less than the maximum back-off value, macmaxBE. The maximum number of successive CCA failures for the same packet is governed by macMaxCSMABackoffs; if this limit is exceeded the packet is discarded at the MAC layer. The standard allows the inclusion of acknowledgements (ACKs) which are sent by the intended receivers on a successful packet reception. Once the packet is received, the receiver performs a Rx-to-Tx turnaround, which is again 12 symbol times, and sends a 22-symbol fixed size ACK packet. A successful transmission is followed by an InterFrameSpace (IFS) before sending another packet.

The IEEE 802.15.4 can operate either in a beacon enabled or a nonbeacon enabled mode. In the beacon enabled mode, the PAN coordinator works with time slots defined through a superframe structure (see Figure 6-8). This permits a synchronous operation of the network. Each superframe has active and inactive portions. The PAN Coordinator interacts with the network only during the active portion. The active portion is composed of three parts: a beacon, a contention access period (CAP), and a contention free period (CFP). The active portion starts with the transmission of a beacon and a CAP commences immediately after the beacon. All frames, except acknowledgment frames and any data frame that immediately follows the acknowledgment of a data request command (as would happen following a data request from a node to the PAN coordinator), transmitted in the CAP, must use a slotted CSMA/CA mechanism to access the channel.
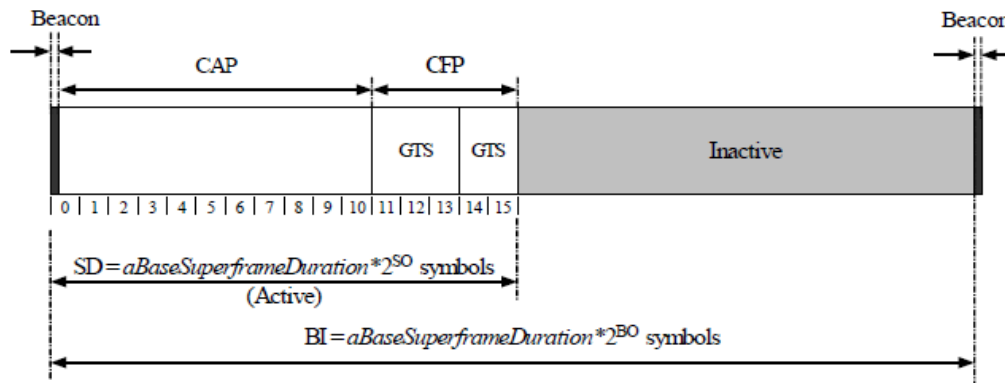
Figure 6-8: The IEEE 802.15.4 Frame Structure

When a transmitted packet collides or is corrupted by the PHY layer noise, the ACK packet is not generated which the transmitter interprets as packet delivery failure. The node reattempts the same packet for a maximum of a Max FrameRetries times before discarding it at the MAC layer. After transmitting a packet, the node turns to Rx-mode and waits for the ACK. The macAckWaitDuration determines the maximum amount of time a node must wait to receive the ACK before declaring that the packet (or the ACK) has collided. The default values of macminBE, macmaxBE, macMaxCSMABackoffs, and a Max FrameRetries are 3, 5, 4, and 3.

6.2.3 Objectives of the Experiment

In Section 6.2.2, we saw that the IEEE 802.15.4 PHY provides a bit rate of 250 Kbps, which has to be shared among the nodes sharing the 2 MHz channel on which this network runs. In the simulation experiment, the packets will have an effective length of 109 bytes (109 B = 872 bits). Thus, over a 250 Kbps link, the maximum packet transmission rate is $\frac{250 \times 1000}{872} = 286.70$ packets per second. We notice, however, from the protocol description in Section 6.2.2, that due to the medium access control, before each packet is transmitted the nodes must contend for the transmission opportunity. This will reduce the actual packet transmission rate well below 286.7.

In this experiment, just one node will send packets to a receiver. Since there is no contention (there being only one transmitter) there is no need for medium access control, and packets could be sent back-to-back. However, the MAC protocol is always present, even with one node, and we would like to study the maximum possible rate at which a node can send back-to-back packets, when it is the only transmitter in the network. Evidently, since there is no uncertainty due to contention from other nodes, the overhead between the packets can be calculated from the protocol description in Section 6.2.2. This has been done in Section 6.2.6.

This analysis will provide the maximum possible rate at which a node can send packets over the IEEE 802.15.4 channel. Then in Section 6.2.7, we compare the throughput obtained from the simulation with that obtained from the analysis. In the simulation, in order to ensure that the node sends at the maximum possible rate, the packet queue at the transmitting node never empties

out. This is ensured by inserting packets into the transmitting node queue at a rate higher than the node can possibly transmit.

6.2.4 NetSim Simulation Setup

Open NetSim and click on **Experiments>IOT-WSN> One Hop IoT Network over IEEE 802.15.4** click on the tile in the middle panel to load the example as shown in below Figure 6-9.
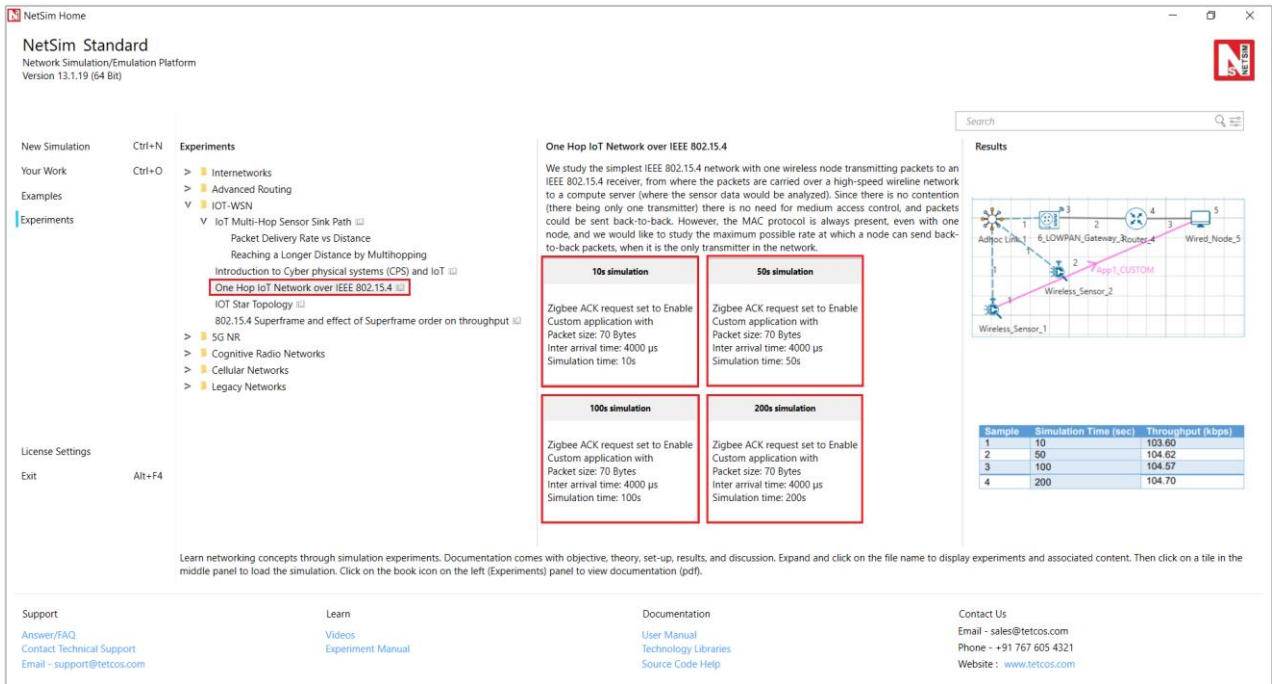

Figure 6-9: List of scenarios for the example of One Hop IoT Network over IEEE 802.15.4

<u>10s Simulation sample</u>

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-10.
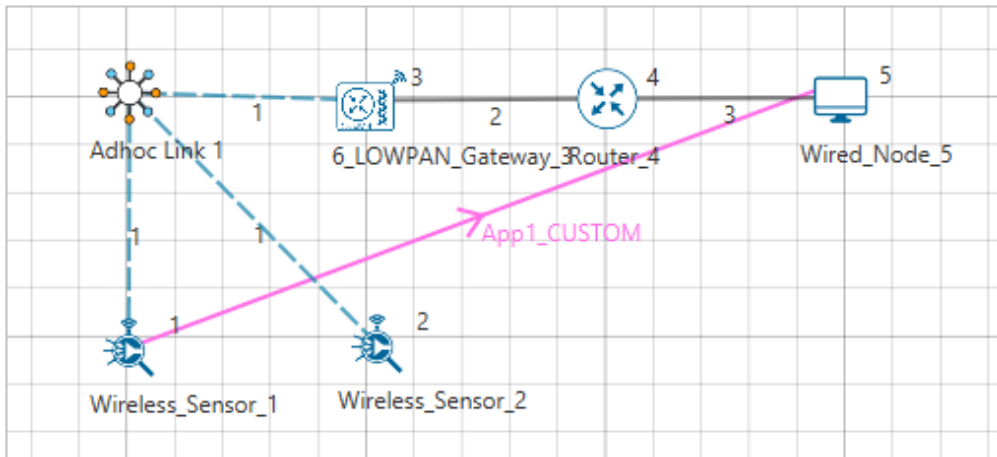

Figure 6-10: Network set up for studying the One Hop IoT Network over IEEE 802.15.4

6.2.5 Simulation Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wireless Sensors, a 6 LOWPAN Gateway, 1 Router, and 1 Wired Node.

Step 2: In the Interface Zigbee > Data Link Layer of Wireless Sensor 1, **Ack Request** is set to Enable and **Max Frame Retries** is set to 7. It will automatically be set for Wireless Sensor 2, since the above parameters are Global.

Step 3: In the Interface Zigbee > Data Link Layer of 6 LOWPAN Gateway, **Beacon Mode** is set to Disable by default.

Step 4: The Ad hoc link properties are set to **NO PATHLOSS** for the channel characteristics.

Step 5: Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A Custom Application is set from Wireless Sensor 1 i.e., Source to Wired Node 5 i.e., Destination. Transport Protocol is set to **UDP** with Packet Size set to 70 Bytes and Inter Arrival Time set to 4000 µs. The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 140 Kbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps) = Packet\ Size\ (Bytes) * 8/Interarrival\ time\ (µs)$$

***NOTE:** If the size of the packet at the Physical layer is greater than 127 bytes, the packet gets fragmented. Taking into account the various overheads added at different layers (which are mentioned below), the packet size at the application layer should be less than 80 bytes.*

Step 6: Plots are enabled in NetSim GUI. Run simulation for 10 Seconds and note down the throughput.

Similarly, do the other samples by increasing the simulation time to 50, 100, and 200 Seconds respectively and note down the throughputs.

6.2.6 Analysis of Maximum Throughput

We have set the Application layer payload as 70 bytes in the Packet Size and when the packet reaches the Physical Layer, various other headers get added like Table 6-1.

App layer Payload	70 bytes
Transport Layer Header	8 bytes
Network Layer Header	20 bytes
MAC Header	5 bytes
PHY Header (includes Preamble, and Start Packet Delimiter)	6 bytes
Packet Size	109 bytes

Table 6-1: Overheads added to a packet as it flows down the network stack

By default, NetSim uses Unslotted CSMA/CA and so, the packet transmission happens after a Random Back Off, CCA, and Turn-Around-Time and is followed by Turn-Around-Time and ACK

Packet and each of them occupies specific time set by the IEEE 802.15.4 standard as per the timing diagram shown below Figure 6-11.
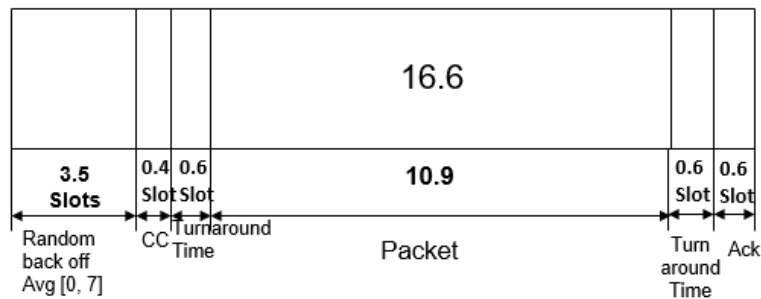

Figure 6-11: Zigbee timing diagram

From IEEE standard, each slot has 20 Symbols in it and each symbol takes 16µs for transmission.

Symbol Time	T_s	16 µs
Slot Time	$20 * T_s$	0.32 ms
Random Backoff Average	3.5 * Slots	1.12 ms
CCA	0.4 * Slots	0.128 ms
Turn-around-Time	0.6 * Slots	0.192 ms
Packet Transmission Time	10.9 * Slots	3.488 ms
Turn-around-Time	0.6 * Slots	0.192 ms
ACK Packet Time	0.6 * Slots	0.192 ms
Total Time	16.6 * Slots	5.312 ms

Table 6-2: Symbol times as per IEEE standard

$$Analytical\ Application\ Throughput = \frac{70(bytes)inApplayer * 8}{5.312\ ms} = 105.42\ kbps$$

6.2.7 Comparison of Simulation and Calculation

Throughput from simulation	104.74 kbps
Throughput from analysis	105.42 kbps

Table 6-3: Results comparison form simulation and theoretical analysis

Throughput from theoretical analysis matches the results of NetSim's discrete event simulation. The slight difference in throughput is due to two facts that

- The average of random numbers generated for backoff need not be exactly 3.5 as the simulation is run for short time.
- In the packet trace one can notice that there are OSPF and AODV control packets (required for the route setup process) that sent over the network. The data transmissions occur only after the control packet transmissions are completed.

As we go on increasing the simulation time, the throughput value obtained from simulation approaches the theoretical value as can be seen from the table below Table 6-4.

Sample	Simulation Time (sec)	Throughput (kbps)
1	10	103.60
2	50	104.62
3	100	104.57
4	200	104.70

Table 6-4: Throughput comparison with different simulation times

6.3 IoT – Multi-Hop Sensor-Sink Path (Level 3)

NOTE: *It is recommended to carry out this experiment in Standard Version of NetSim.*

6.3.1 Introduction

The Internet provides the communication infrastructure for connecting computers, computing devices, and people. The Internet is itself an interconnection of a very large number of interconnected packet networks, all using the same packet networking protocol. The Internet of Things will be an extension of the Internet with sub-networks that will serve to connect "things" among themselves and with the larger Internet. For example, a farmer can deploy moisture sensors around the farm so that irrigation can be done only when necessary, thereby resulting in substantial water savings. Measurements from the sensors have to be communicated to a computer on the Internet, where inference and decision-making algorithms can advise the farmer as to required irrigation actions.

Farms could be very large, from a few acres to hundreds of acres. If the communication is entirely wireless, a moisture sensor might have to communicate with a sink that is 100s of meters away. As the distance between a transmitter and a receiver increase, the power of the signal received at the receiver decreases, eventually making it difficult for the signal processing algorithms at the receiver to decode the transmitted bits in the presence of the ever-present thermal noise. Also, for a large farm there would need to be many moisture sensors; many of them might transmit together, leading to *collisions* and *interference.*

6.3.2 Theory

The problem of increasing distance between the transmitter and the receiver is solved by placing *packet routers* between the sensors and the sink. There could even be multiple routers on the path from the sensor to the sink, the routers being placed so that any intermediate link is short enough to permit reliable communication (at the available power levels). We say that there is a *multi-hop* path from a sensor to the sink.

By introducing routers, we observe that we have a system with sensors, routers, and a sink; in general, there could be multiple sinks interconnected on a separate *edge* network. We note here that a sensor, on the path from another sensor to the sink, can also serve the role of a router. Nodes whose sole purpose is to forward packets might also need to be deployed.

The problem of collision and interference between multiple transmission is solved by overlaying the systems of sensors, routers, and sinks with a *scheduler* which determines (preferably in a distributed manner) which transmitters should transmit their packets to which of their receivers.
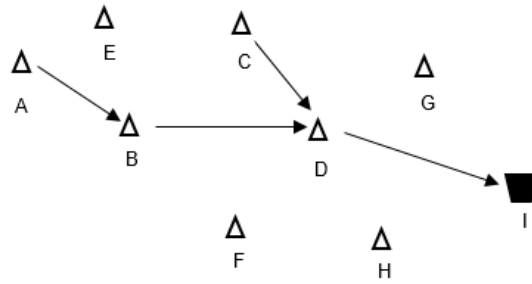
Figure 6-12: Data from Sensor A to Sink I takes the path A-B-D-I while data from sensor C to Sink I takes the path C-D-I

In this experiment, we will use NetSim Simulator to study the motivation for the introduction of packet routers, and to understand the performance issues that arise. We will understand the answers to questions such as:

1. How does packet error rate degrade as the sensor-sink distance increases?
2. How far can a sensor be from a sink before a router needs to be introduced?
3. A router will help to keep the signal-to-noise ratio at the desired levels, but is there any adverse implication of introducing a router?

6.3.3 Network Setup

Open NetSim and click on **Experiments> IOT-WSN> IoT Multi Hop Sensor Sink Path > Packet Delivery Rate and Distance** then click on the tile in the middle panel to load the example as shown in below Figure 6-13.

Figure 6-13: List of scenarios for the example of IoT Multi Hop Sensor Sink Path

6.3.4 Packet Delivery Rate vs. Distance

In this part, we perform a simulation to understand, **"How the distance between the source and sink impacts the received signal strength (at the destination) and in turn the packet error rate?"** We will assume a well-established path-loss model under which, as the distance varies, the received signal strength (in dBm) varies linearly. For a given transmit power (say 0dBm), at a certain reference distance (say 1m) the received power is c_0dBm and decreases beyond this point as $-10\eta \log_{10} d$ for a transmitter-receiver distance of d. This is called a *power-law* path loss model, since in mW the power decreases as the η power of the distance d. The value of η is 2 for free space path loss and varies from 2 to 5 in the case of outdoor or indoor propagation. Values of η are obtained by carrying out experimental propagation studies.

Distance vs BER PER and RSSI sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-14.

Figure 6-14: Network set up for studying the Distance vs BER PER and RSSI sample

6.3.5 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in the NetSim GUI comprising of a WSN Sink and 1 Wireless Sensor in **Wireless Sensor Networks**.
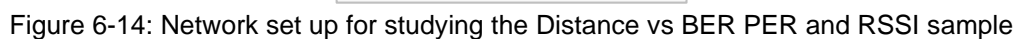
Note: NetSim currently supports a maximum of only one device as WSN Sink.

Step 2: Before we actually designed this network, in the Fast Config Window containing inputs for **Grid Settings and Sensor Placement**, the Grid Length and Side Length were set to 500 meters respectively, instead of the default 50 meters and we have chosen **Manually Via Click and Drop** option.

Step 3: The distance between the WSN Sink and Wireless Sensor is 5 meters.

Step 4: Go to Network Layer properties of Wireless Sensor 2, the Routing Protocol is set as **AODV**.

Note: The Routing Protocol parameter is Global. i.e., It will automatically be set to AODV in WSN Sink.

Step 5: In the Interface Zigbee > Data Link Layer of Wireless Sensor 2, **Ack Request** is set to Enable and **Max Frame Retries** is set to 4. Similarly, it is set for WSN Sink 1.

Step 6: In the Interface Zigbee > Physical Layer of Wireless Sensor 2, **Transmitter Power** is set to 1mW, **Reference Distance** is set to 1m, **Receiver Sensitivity** is set to -105dBm, and **ED Threshold** is set to -115dBm.

Step 7: Channel Characteristics: Path Loss Only, Path Loss Model: Log Distance, Path Loss Exponent: 3.5

Step 8: Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wireless Sensor 2 i.e., Source to WSN Sink 1 i.e., Destination with Transport Protocol set to UDP, Packet Size set to 70 Bytes and Inter Arrival Time set to 4000 μs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 140 Kbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps)\ =\ Packet\ Size\ (Bytes)\ *\ 8/Interarrival\ time\ (μs)$$

Step 9: The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2 Figure 6-15, Enable - Static IP Route ->Click on **Configure Static Route IP**.
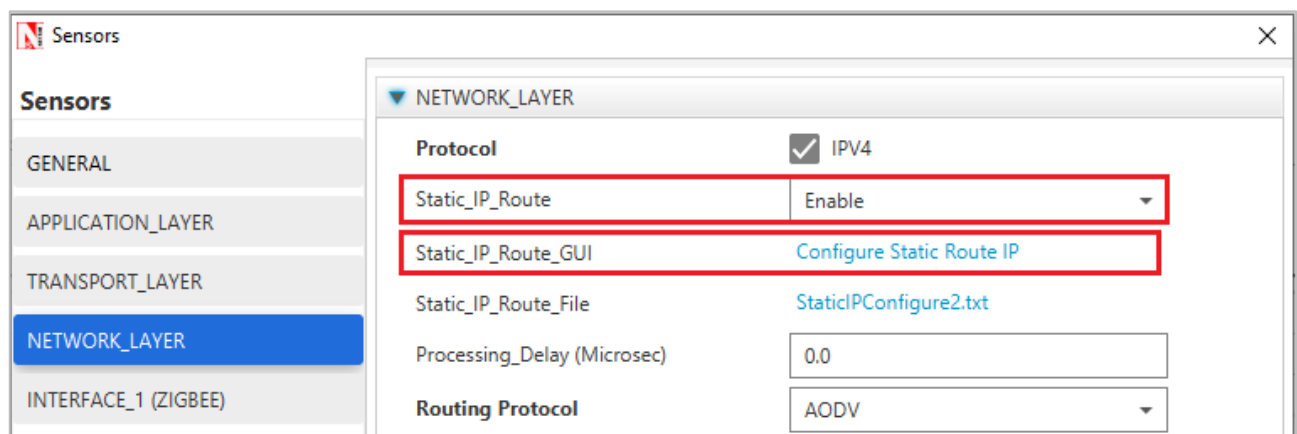

Figure 6-15: Network layer properties window

Static IP Routing Dialogue box gets open.

Enter the Network Destination, Gateway, Subnet Mask, Metrics, and Interface ID. Click on **Add**.

You will find the entry added to the below Static IP Routing Table as shown below.

Click on **OK**.

Figure 6-16: Static Route Configuration Window

Step 10: Packet Trace is enabled in NetSim GUI. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis. Enable the plots from NtSim GUI.

*Note: Before we click on **Run** simulation, user need to modify the code as per the **"Procedure to log RSSI and BER"** given below.*

NOTE: **The following changes need to be done manually by the user inorder to carry out this experiment.**

Procedure to log RSSI and BER (Possible in Standard / Pro Versions only):

RSSI and BER in ZigBee project can be logged into a text file. The following code changes are required to log these parameters into a txt file.

- Go to NetSim Home page and click on **Your work.**
- Click on Workspace Options and then Click on Source code→open code and open the codes in Visual Studio. Set **x64** according to the NetSim build which you are using.

NOTE: **We recommend Visual Studio Community Edition 2017 or Higher.**

- Go to the Zigbee Project in the Solution Explorer. Open 802_15_4.c file and add the follwing lines of code highlighted in red, inside the **fn_NetSim_Zigbee_init()** function as shown below:

```
_declspec (dllexport) int fn_NetSim_Zigbee_Init()
{
FILE* fp;
//RSSI BER SNR LOG
fp = fopen("ZIGBEE_BER_LOG.txt", "w+");
if (fp)
{
```

```
fprintf(fp,"PACKET_ID,\tTRANSMITTER,\t\tRECEIVER,\tRX_POWER(dBm),\tTOTAL
_RX_POWER(dBm), \tBER");
fclose(fp);
}
//RSSI BER SNR LOG
        return fn_NetSim_Zigbee_Init_F();
}
```

- Add the lines of code highlighted in red inside the **fn_NetSim_Zigbee_Run()** function under **PHYSICAL_IN_EVENT** as shown below:

```
case PHYSICAL_IN_EVENT:
{
NetSim_PACKET *pstruPacket;
PACKET_STATUS nPacketStatus;
double SNR;
double dBER;
FILE* fp;

pstruPacket = pstruEventDetails->pPacket;
if (pstruPacket->nReceiverId && pstruPacket->nReceiverId != pstruEventDetails-
>nDeviceId)
{
fnNetSimError("Different device packet received..");
assert(false);
return 0;
}
if (!ZIGBEE_CHANGERADIOSTATE(pstruEventDetails->nDeviceId,
WSN_PHY(pstruEventDetails->nDeviceId)->nRadioState, RX_ON_IDLE))
return 0;
if (WSN_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower -
GET_RX_POWER_mw(pstruPacket->nTransmitterId, pstruPacket->nReceiverId,
pstruEventDetails->dEventTime) >= WSN_PHY(pstruEventDetails->nDeviceId)-
>dReceiverSensivity)
pstruPacket->nPacketStatus = PacketStatus_Collided;
nPacketStatus = pstruPacket->nPacketStatus;
ZIGBEE_SINR(&SNR,
WSN_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower,
GET_RX_POWER_mw(pstruPacket->nTransmitterId, pstruPacket->nReceiverId,
pstruEventDetails->dEventTime));
```

```
dBER = fn_NetSim_Zigbee_CalculateBER(SNR);


//RSSI BER SNR LOG
double rxpwr = MW_TO_DBM(WSN_PHY(pstruEventDetails->nDeviceId)-
>dTotalReceivedPower);
double total_rxpwr = GET_RX_POWER_dbm(pstruPacket->nTransmitterId,
pstruPacket->nReceiverId, pstruEventDetails->dEventTime);
fp = fopen("ZIGBEE_BER_LOG.txt", "a+");
if (fp)
{
fprintf(fp, "\n%lld,\t\t%s,\t%s,\t%lf,\t%lf,\t\t%lf", pstruPacket->nPacketId,
DEVICE_NAME(pstruPacket->nTransmitterId),
DEVICE_NAME(pstruPacket->nReceiverId),
rxpwr, total_rxpwr, dBER);
fclose(fp);
}
//RSSI BER SNR LOG
if (fn_NetSim_Packet_DecideError(dBER, pstruEventDetails->dPacketSize))
```

- Right click on the **ZigBee** project in the solution explorer and click on rebuild.
- After the Zigbee project is rebuild successful, go back to the network scenario.

Step 10: Enable the plots and run the simulation for 10 Seconds. Once the simulation is complete, it will generate a text file named **ZIGBEE_BER_LOG.txt** containing **RSSI** and **BER** in the binary folder of NetSim. i.e., **<NetSim Install Directory>/bin**.

6.3.6 Output for Distance vs BER PER and RSSI sample

RSSI, PER, BER vs. Distance (path-loss: linear in log-distance, with $\eta = 3.5$)				
Distance(m)	RSSI (dBm) (Pathloss model)	BER	PER	PLR (After MAC retransmissions*)
5	-64.51	0.00	0	0
10	-75.04	0.00	0	0
15	-81.20	0.00	0	0
20	-85.58	0.00	0	0
25	-88.97	0.00	0	0
30	-91.74	0.00	0	0
35	-94.08	0.000005	0.0051	0
40	-96.11	0.000229	0.2076	0
45	-97.90	0.002175	0.8905	0.447

50	-99.51	0.008861	0.9999	1
55	-100.95	0.022370	1	1
60	-102.28	0.042390	1	1
65	-103.49	0.067026	1	1
70	-104.62	0.094075	1	1
75	-	-	-	-
80	-	-	-	-

Table 6-5: RSSI, PER, BER from ZIGBEE_BER_LOG.txt vs. Distance

Comparison Charts
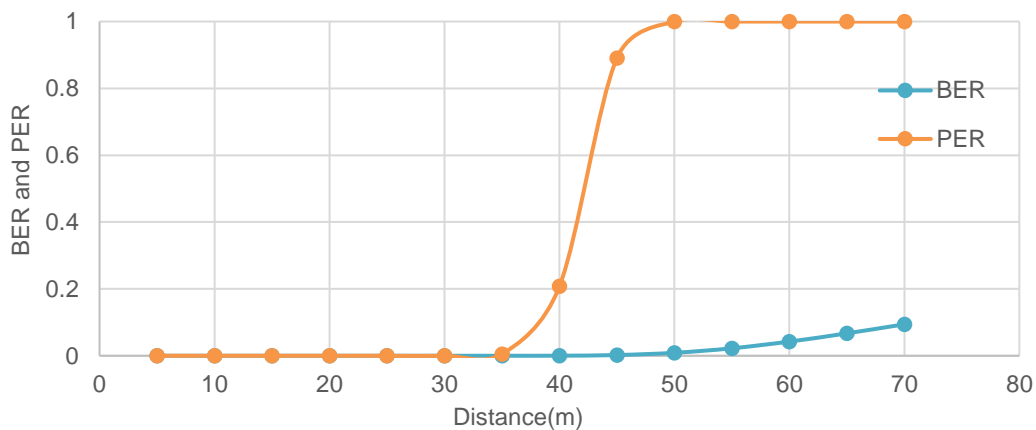
(a)
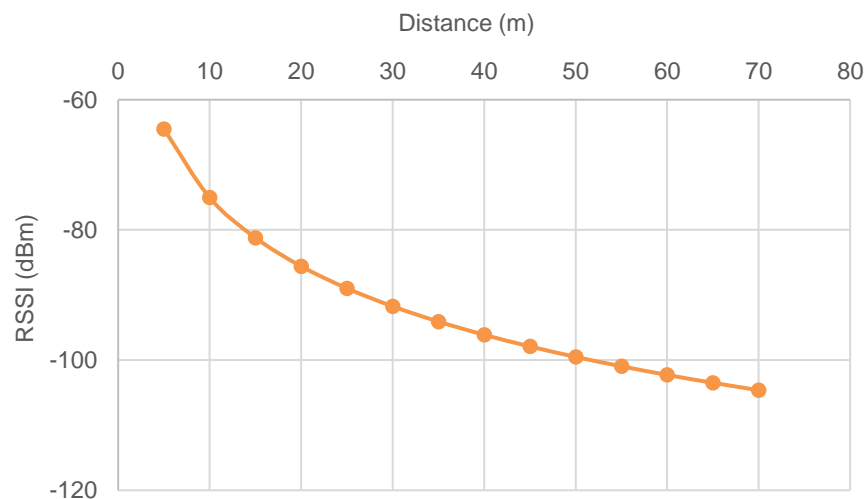
(b)

Figure 6-17: (a) Distance Vs. BER, PER and (b) Distance vs. RSSI

* The IEEE 802.15.4 MAC implements a retransmission scheme that attempts to recover errored packets by retransmission. If all the retransmission attempts are also errored, the packet is lost.

The table above reports the RSSI (Received Signal Strength), BER (Bit Error Rate), and Packet Error Rate (PER), and the Packet Loss Rate (PLR) as the distance between the sensor to the

sink is increased from 5m to 50m with path loss exponent $\eta = 3.5$. We see that the BER is 0 until a received power of about -92dBm. At a distance of 35m the received power is -94 dBm, and we notice a small BER of 5×10^{-6}. As the distance is increased further the BER continues to grow and at 45m the BER is about 0.002175, yielding $PER = 0.89$, and $PLR = 0.44$. Here PER is obtained from the following formula (which assumes independent bit errors across a packet)

$$PER = 1 - (1 - BER)^{PL},$$

Where,

$$PL - \ packet\ length\ in\ bits\ at\ the\ PHY\ layer$$

$$PL\ (bits) = \big(70\ (payload) + 57(overhead)\big) * 8$$

The PLR in the above table has been obtained from NetSim, which implements the details of the IEEE 802.15.4 MAC acknowledgement and reattempt mechanism. This mechanism is complex, involving a MAC acknowledgement, time-outs, and multiple reattempts. Analysis of the PLR, therefore, is not straightforward. Assuming that the probability of MAC acknowledgement error is small (since it is a small packet), the PLR can be approximated as PER^{K+1}, where K is the maximum number of times a packet can be retransmitted.

$$PLR = \frac{Total\ number\ of\ Lost\ Packet}{Total\ number\ of\ Packet\ Sent\ by\ Source\ MAC}$$

$$Total\ number\ of\ Lost\ packets$$
$$= Total\ number\ of\ Packet\ Sent\ by\ SourceMAC$$
$$- Packets\ Received\ at\ Destination\ MAC$$

Steps to calculate Packet Loss Rate

- Open Packet Trace from the Results Dashboard. Filter the PACKET TYPE column as Custom and note down the packet id of the last packet sent from the PACKET ID column.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
1850	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1851	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1852	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1853	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1854	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1855	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1856	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1857	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1858	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1859	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1860	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1861	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1862	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1863	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1864	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1865	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1
1866	0	Custom	App1_CUSTOM	SENSOR-2	SINKNODE-1	SENSOR-2	SINKNODE-1

Figure 6-18: Packet Trace

This represents the total number of packets sent by the source.

- Note down the Packets Received from the Application Metrics in the Results Dashboard Figure 6-19.

Application_Metrics_Table							⊡ ✕
Application_Metrics							☐ Detailed View
Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)
1	_Application_Throughput_plot_	App1_CUSTOM	2500	1866	0.104496	1261780.928725	1358.554960

Figure 6-19: Application metrics table in Result Dashboard

This represents the total number of packets received at the destination.

- Calculate the total number of Lost Packets and PLR as follows:

  For the above case,

$$Total\ number\ of\ Packet\ Sent\ by\ SourceMAC = 463$$

$$Packets\ Received\ at\ Destination\ MAC = 256$$

$$Total\ number\ of\ Lost\ packets = 463 - 256 = 207$$

$$PLR = \frac{207}{463} = 0.447$$

6.3.7 Inference

It is clear that Internet applications, such as banking and reliable file transfer, require that all the transmitted data is received with 100% accuracy. The Internet achieves this, in spite of unreliable communication media (no medium is 100% reliable) by various protocols above the network layer. Many IoT applications, however, can work with less than 100% packet delivery without affecting the application. Take, for example, the farm moisture sensing application mentioned in the introduction. The moisture levels vary slowly; if one measurement is lost, the next received measurement might suffice for the decision-making algorithm. This sort of thinking also permits the IoT applications to utilize cheap, low power devices, making the IoT concept practical and affordable.

With the above discussion in mind, let us say that the application under consideration requires a measurement delivery rate of at least 80%. Examining the table above, we conclude that the sensor-sink distance must not be more than 40 meters. Thus, even a 1 acre farm $(61m \times 61m)$ would require multi-hopping to connect sensors to a sink at the edge of the farm.

In Part 2 of this experiment, we will study the placement of a single router between the sensor and the sink, to increase the sensor-sink distance beyond 40 meters.

6.3.8 Reaching a Longer Distance by Multihopping

Direct sensor sink link sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-20.
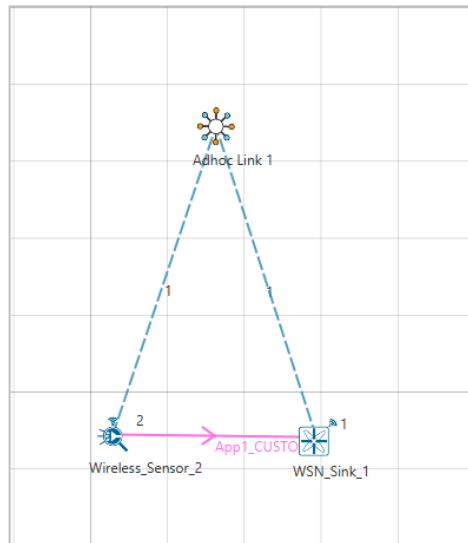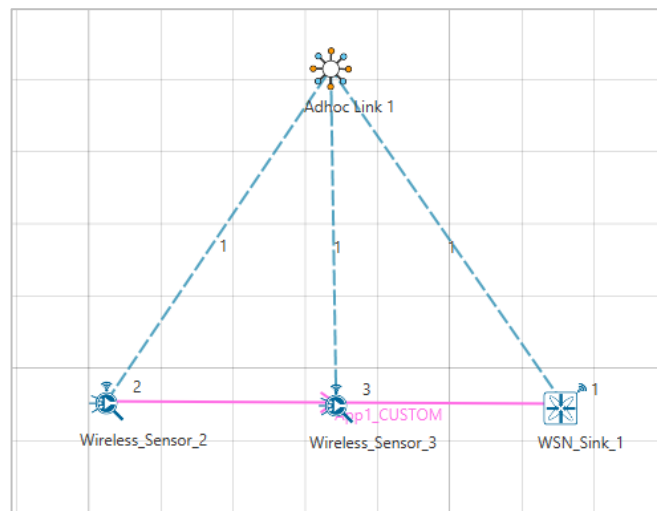

Figure 6-20: Network set up for studying the Direct sensor sink link sample

6.3.9 Procedure

The following changes in settings are done from the previous sample:

Step 1: The distance between the WSN Sink and Wireless Sensor is 40 meters.

Step 2: In the Interface Zigbee > Data Link Layer of Wireless Sensor 2, **Ack Request** is set to Enable and **Max Frame Retries** is set to 3.

Step 3: The Ad hoc Link properties are set as follows Figure 6-21.
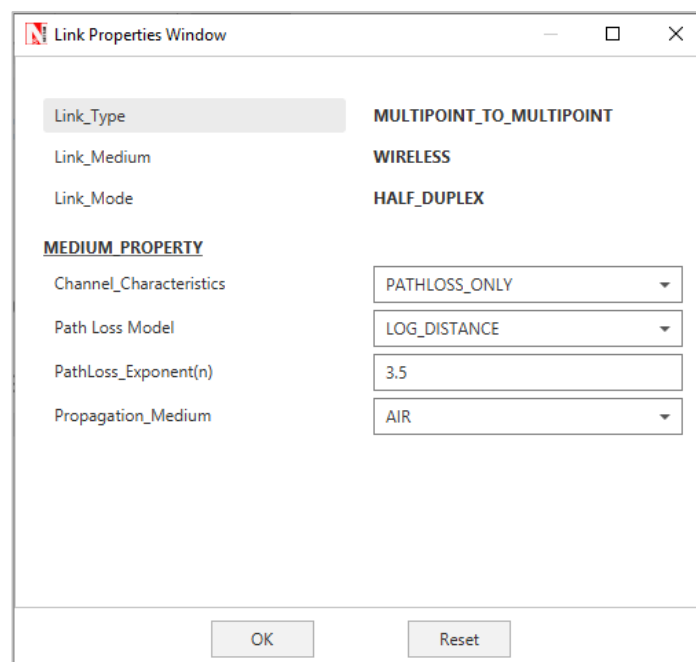
Figure 6-21: Wireless Link properties

Step 4: Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wireless Sensor 2 i.e. Source to WSN Sink 1 i.e. Destination with Packet Size set to 70 Bytes and Inter Arrival Time set to 100000 µs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 5.6 Kbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps)\ =\ Packet\ Size\ (Bytes)\ *\ 8/Interarrival\ time\ (µs)$$

Step 5: Enable the plots and run the Simulation for 100 Seconds. Once the simulation is complete, note down the Packet Generated value and Throughput value from the **Application Metrics**.

Note down the Packet Received, Packet Errored, and Packet Collided from the **Link Metrics**.

Router between sensor and sink sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-22.

Figure 6-22: Network set up for studying the Router between sensor and sink sample

6.3.10 Procedure

The following changes in settings are done from the previous sample:

Step 1: One more Wireless Sensor is added to this network. The distance between Wireless Sensor 2 and Wireless Sensor 3 is 40 meters and the distance between Wireless Sensor 3 and the WSN Sink is 40 meters.

Step 2: The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2 Figure 6-23, Enable - Static IP Route ->Click on **Configure Static Route IP**.
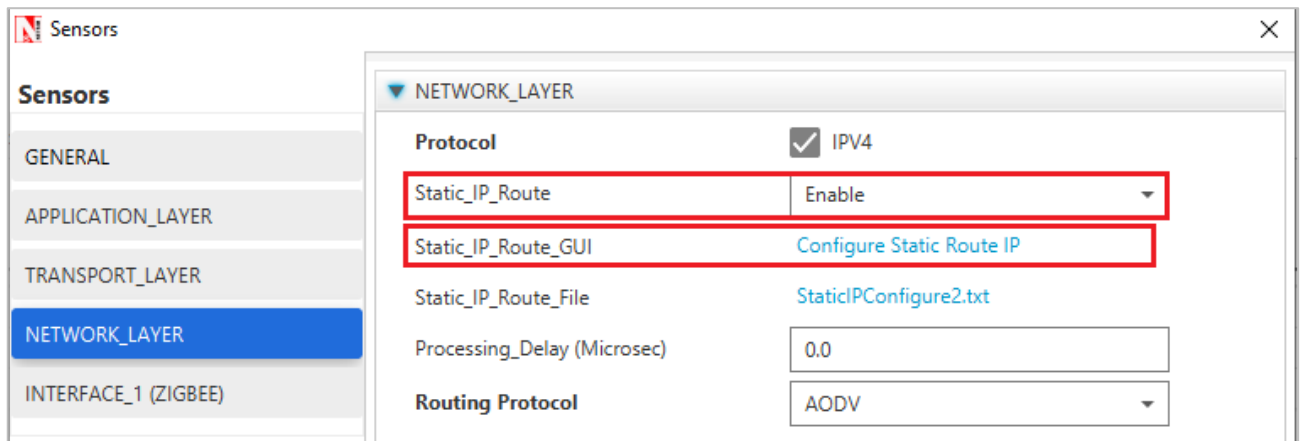
Figure 6-23: Network layer properties window

Static IP Routing Dialogue box gets open.

Enter the Network Destination, Gateway, Subnet Mask, Metrics, and Interface ID. Click on **Add**.

You will find the entry added to the below Static IP Routing Table as shown below:
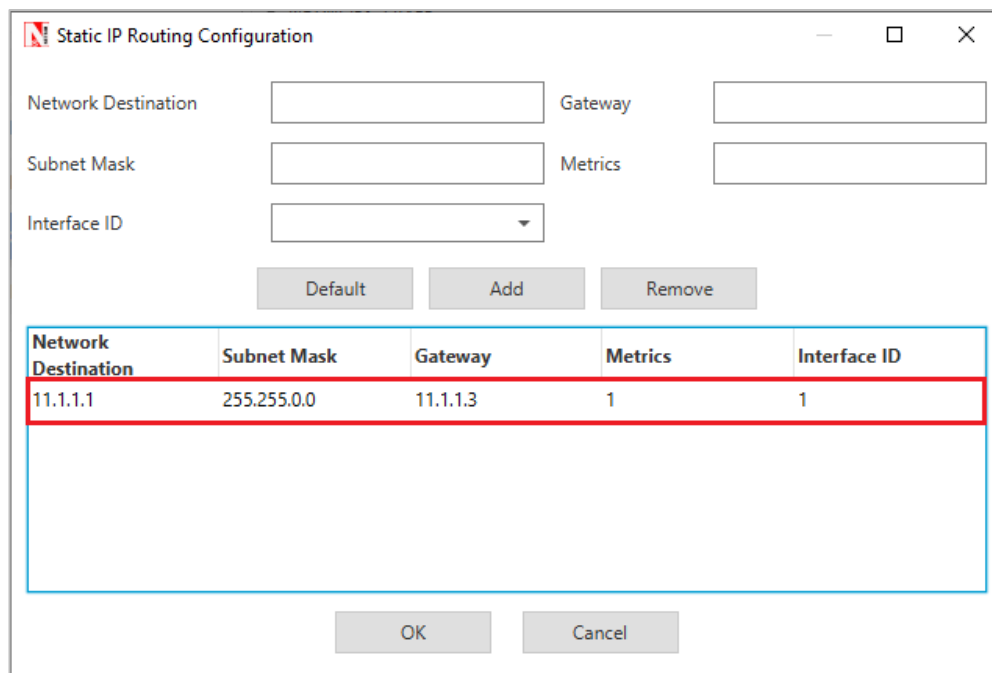
Click on **OK**.

Figure 6-24: Static Route configuration for Wireless Sensor 2

Similarly, Static IP is set for Wireless Sensor 3 as shown below Figure 6-25.

Figure 6-25: Static Route configuration for Wireless Sensor 3

Step 3: Enable the plots and run the Simulation for 100 Seconds. Once the simulation is complete, note down the Packet Generated value and Throughput value from the **Application Metrics**.

Note down the Packet Received, Packet Errored, and Packet Collided from the Link Metrics Table 6-6.

6.3.11 Output for Router between sensor and sink sample

	Source-Sink Distance (m)	Packets Generated	Packets Received	Packets Errored (PHY)	Packets Collided	Packet Loss (MAC)	PLR	Mean Delay (μs)
Direct sensor-sink link	40	1000	1012	244	0	0	0	6514.45
Router between sensor and sink	80 (router at midpoint)	1000	1015	540	0	0	0	14239.94

Table 6-6: Packet Generated/Received/Errored/Collided and Mean delay from result dashboard

NOTE: Packet loss (PHY) is the number of packets that were received in error and then recovered by retransmission. Packets received is slightly higher than packets generated on account of retransmissions of successful packets in case of ACK errors.

6.3.12 Inference

In **Distance vs BER PER and RSSI** sample of this experiment, we learnt that if the sensor device uses a transmit power of 0dBm, then for one-hop communication to the sink, the sensor-sink

distance cannot exceed 40m. If the sensor-sink distance needs to exceed 40m (see the example discussed earlier), there are two options:

1. The transmit power can be increased. There is, however, a maximum transmit power for a given device. Wireless transceivers based on the CC 2420 have a maximum power of 0dBm (i.e., about 1 mW), whereas the CC 2520 IEEE 802.15.4 transceiver provides maximum transmit power of 5dBm (i.e., about 3 mW). Thus, given that there is always a maximum transmit power, there will always be a limit on the maximum sensor-sink distance.

2. Routers can be introduced between the sensor and the sink, so that packets from the sensor to the sink follow a *multihop* path. A router is a device that will have the same transceiver as a sensor, but its microcontroller will run a program that will allow it to forward packets that it receives. Note that a sensor device can also be programmed to serve as a router. Thus, in IOT networks, sensor devices themselves serve as routers.

In this part of the experiment, we study the option of introducing a router between a sensor and the sink to increase the sensor-sink distance. We will compare the performance of two networks, one with the sensor communicating with a sink at the distance of 40m, and another with the sensor-sink distance being 80m, with a sensor at the mid-point between the sensor and the sink.

Direct sensor sink link sample simulates a one hop network with a sensor-sink distance of 40m. We recall from Part 1 that, with the transceiver model implemented in NetSim, 40m is the longest one hop distance possible for 100% packet delivery rate. In **Router between sensor and sink** sample, to study the usefulness of routing we will set up network with a sensor-sink distance of 80m with a packet router at the midpoint between the sensor and the sink.

The measurement process at the sensor is such that one measurement (i.e., one packet) is generated every 100ms. The objective is to deliver these measurements to the sink with 100% delivery probability. From Part 1 of this experiment, we know that a single hop of 80m will not provide the desired packet delivery performance.

The Table at the beginning of this section shows the results. We see that both networks are able to provide a packet delivery probability of 100%. It is clear, however, that since the second network has two hops, each packet needs to be transmitted twice, hence the mean delay between a measurement being generated and it being received at the sink is doubled. Thus, the longer sensor-sink distance is being achieved, for the same delivery rate, at an increased delivery delay.

The following points may be noted from the table:

1. The number of packets lost due to PHY errors. The packet delivery rate is 100% despite these losses since the MAC layer re-transmission mechanism is able to recover all lost packets.

2. There are no collisions. Since both the links (sensor-router and router-sink) use the same channel and there is no co-ordination between them, it is possible, in general for sensor-

router and router-sink transmissions to collide. This is probable when the measurement rate is large, leading to simultaneously nonempty queues at the sensor and router. In this experiment we kept the measurement rate small such that the sensor queue is empty when the router is transmitting and vice versa. This avoids any collisions.

6.4 Performance Evaluation of a Star Topology IoT Network (Level 3)

6.4.1 Introduction

In IOT experiments 20 and 21 we studied a single IoT device connected to a sink (either directly or via IEEE 802.15.4 routers). Such a situation would arise in practice, when, for example, in a large campus, a ground level water reservoir has a water level sensor, and this sensor is connected wirelessly to the campus wireline network. Emerging IoT applications will require several devices, in close proximity, all making measurements on a physical system (e.g., a civil structure, or an industrial machine). All the measurements would need to be sent to a computer connected to the infrastructure for analysis and inferencing. With such a scenario in mind, in this experiment, we will study the performance of several IEEE 802.15.4 devices each connected by a single wireless link to a sink. This would be called a "star topology" as the sensors can be seen as the spikes of a "star."

We will set up the experiment such that every sensor can sense the transmissions from any other sensor to the sink. Since there is only one receiver, only one successful transmission can take place at any time. The IEEE 802.15.4 CSMA/CA multiple access control will take care of the coordination between the sensor transmissions. In this setting, we will conduct a saturation throughput analysis. The IoT communication buffers of the IEEE 802.15.4 devices will always be nonempty, so that as soon as a packet is transmitted, another packet is ready to be sent. This will provide an understanding of how the network performs under very heavy loading. For this scenario we will compare results from NetSim simulations against mathematical analyses.

Details of the IEEE 802.15.4 PHY and MAC have been provided in the earlier IOT experiments 20 and 21, and their understanding must be reviewed before proceeding further with this experiment. In this experiment, all packets are transferred in a single hop from and IoT device to the sink. Hence, there are no routers, and no routing to be defined.

6.4.2 NetSim Simulation Setup

Open NetSim and click on **Experiments> IOT-WSN> IOT Star Topology** then click on the tile in the middle panel to load the example as shown in below Figure 6-26.
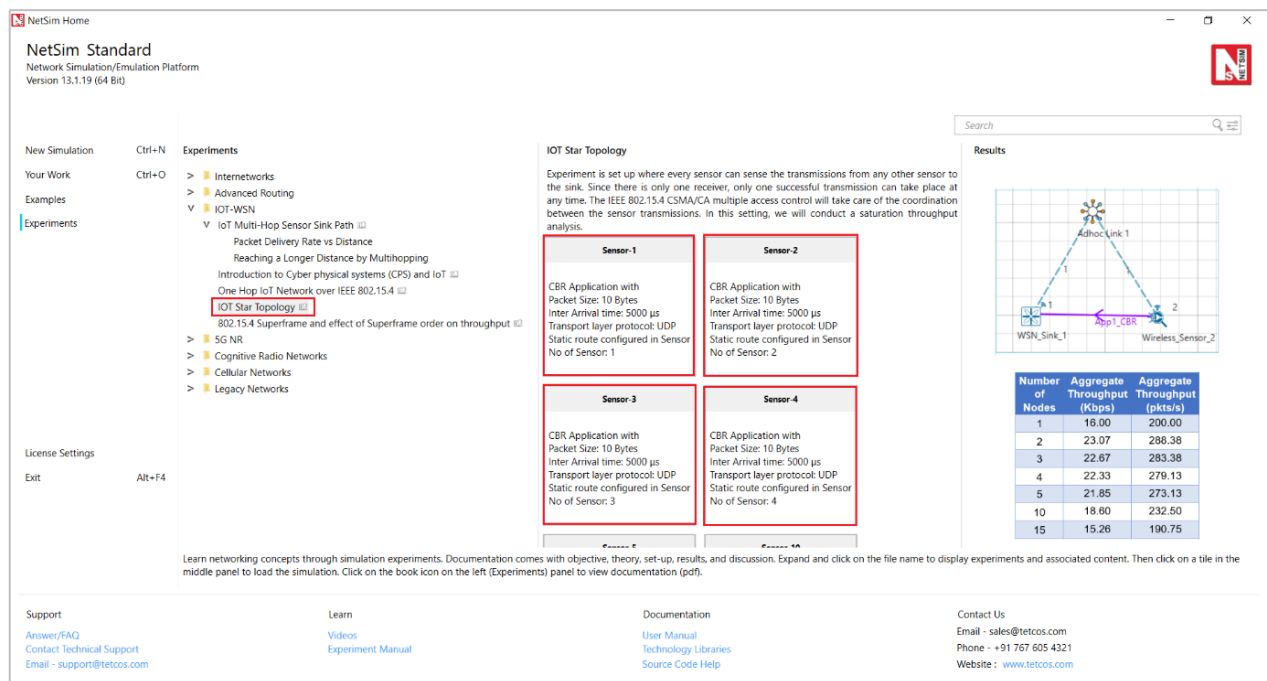
Figure 6-26: List of scenarios for the example of IOT Star Topology

The simulation scenario consists of n nodes distributed uniformly around a sink (PAN coordinator) at the center. In NetSim the nodes associate with the PAN coordinator at the start of the simulation. CBR traffic is initiated simultaneously from all the nodes. The CBR packet size is kept as 10 bytes to which 20 bytes of IP header, 7 bytes of MAC header and 6 bytes of PHY header are added. To ensure saturation, the CBR traffic interval is kept very small; each node's buffer receives packets at intervals of 5 ms

6.4.3 Procedure

Sensor-1

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-27.
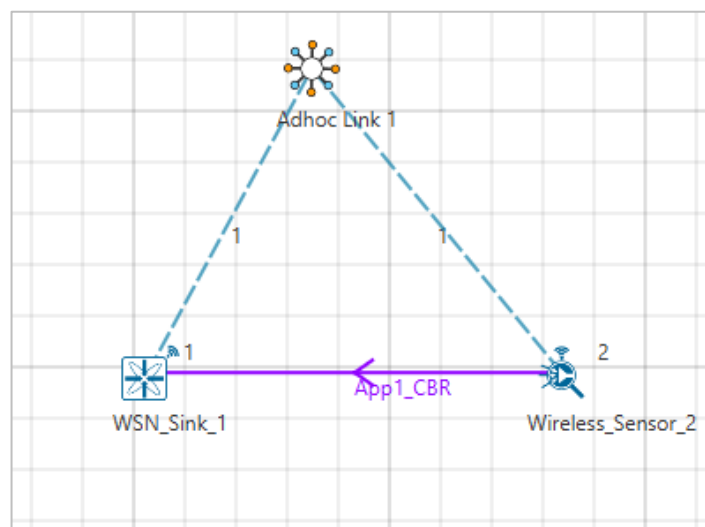
Figure 6-27: Network set up for studying the Sensor 1

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 1 SinkNode, and 1 wireless sensor in the **"WSN"** Network Library. Distance between the SinkNode and Wireless Sensor is set to 8m.

Step 2: Right click on Adhoc link and select Properties, Channel Characteristics is set to No_Pathloss

Step 3: Right click on Wireless sensor properties in the network layer the static route is configured as shown below Table 6-7**.**

Network Destination	Gateway	SubnetMask	Metrics	Interface ID
11.1.1.0	11.1.1.1	255.255.255.0	1	1

Table 6-7: Static route for Wireless sensor

Step 4: Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wireless Sensor 2 i.e., Source to Sink Node 1 i.e., Destination with Packet Size remaining 10Bytes and Inter Arrival Time remaining 5000µs. Start time is set to 5s.

Transport Protocol is set to **UDP** instead of TCP.

Step 5: Plots are enabled in NetSim GUI. Click on Run simulation. The simulation time is set to 100 seconds.

Increase wireless sensor count to 2, 3, and 4 with the same above properties to design Sensor-2, 3, and 4.

Sensor-5: NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-28.
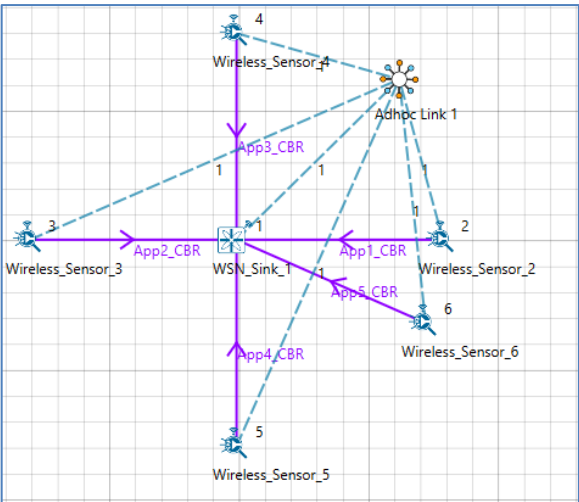
Figure 6-28: Network set up for studying the Sensor-5

The following set of procedures were done to generate this sample:

Step 1: Right click on Wireless sensors properties in the network layer the static route is configured as shown below Table 6-8.

Network Destination	Gateway	SubnetMask	Metrics	Interface ID
11.1.1.0	11.1.1.1	255.255.255.0	1	1

Table 6-8: Static route for Wireless sensors

Step 2: Click on the Application icon present in the top ribbon/toolbar. The following application properties is set shown in below Table 6-9.

	App 1	App 2	App 3	App 4	App 5
Application	CBR	CBR	CBR	CBR	CBR
Source ID	2	3	4	5	6
Destination ID	1	1	1	1	1
Start time	5s	5s	5s	5s	5s
Transport protocol	UDP	UDP	UDP	UDP	UDP
Packet Size	10bytes	10bytes	10bytes	10bytes	10bytes
Inter-Arrival Time	5000 µs	5000 µs	5000 µs	5000 µs	5000 µs

Table 6-9: Detailed Application properties

Step 5: Plots are enabled in NetSim GUI. Click on Run simulation. The simulation time is set to 100 seconds.

Increase wireless sensor count to 10, 15, 20, 25, 30, 35, 40, and 45 with the same above properties to design Sensor-6, 7, 8, 9, 10, 11, 12, and 13.

6.4.4 Output

The aggregate throughput of the system can be got by adding up the individual throughput of the applications. NetSim outputs the results in units of kilobits per second (kbps). Since the packet size is $80\ bits$ we convert per the formula

$$Aggregate\ Throughput\ (pkts\ per\ sec) = \frac{Aggregate\ Throughput\ (kbps) * 1000}{80}$$

Number of Nodes	Aggregate Throughput (Kbps)	Aggregate Throughput (pkts/s)
1	16.00	200.00
2	23.07	288.38
3	22.67	283.38
4	22.33	279.13
5	21.85	273.13
10	18.60	232.50
15	15.26	190.75
20	12.32	154.00
25	9.98	124.75
30	7.84	98.00

35	6.17	77.13
40	4.88	61.00
45	3.72	46.50

Table 6-10: Aggregate throughput for Star topology scenario
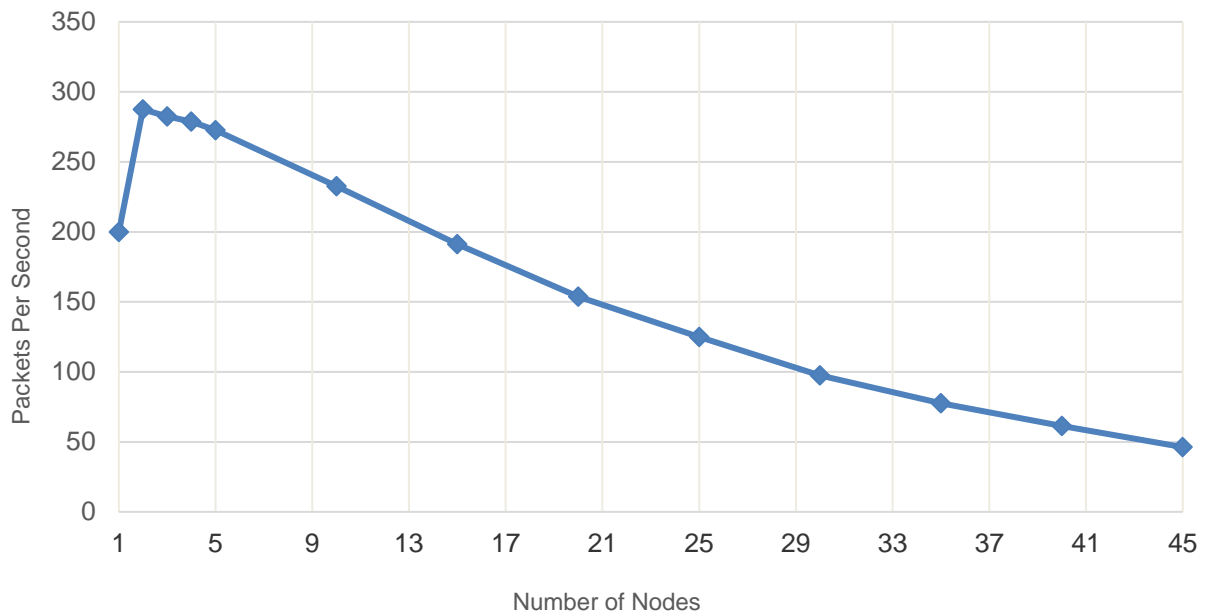
Figure 6-29: Plot of Aggregated throughput (packets/s) vs Number of Nodes

6.4.5 Discussion

In Figure 6-29 we plot the throughput in packets per second versus the number of IoT devices in the star topology network. We make the following observations:

1. Notice that the throughput for a single saturated node is 200 packets per second, which is just the single hop throughput from a single saturated node, when the packet payload is 10 bytes. When a node is by itself, even though there is no contention, still the node goes through a backoff after every transmission. This backoff is a waste and ends up lowering the throughput as compared to if the node knew it was the only one in the network and sent packets back-to-back.

2. When there are two nodes, the total throughput increases to 287.5 packets per second. This might look anomalous. In a contention network, how can the throughput increase with more nodes? The reason can be found in the discussion of the previous observation. Adding another node helps fill up the time wasted due to redundant backoffs, thereby increasing throughput.

3. Adding yet another node results in the throughput dropping to 282.5 packets per second, as the advantage gained with 2 nodes (as compared to 1 node) is lost due to more collisions.

4. From there on as the number of nodes increases the throughput drops rapidly to about 100 packets per second for about 30 nodes.

5. The above behaviour must be compared with a Experiment 12 where several IEEE 802.11 STAs, with saturated queues, were transmitting packets to an AP. The throughput increased from 1 STA to 2 STAs, dropped a little as the number of STAs increased and then flattened

out. On the other hand, in IEEE 802.15.4 the throughput drops rapidly with increasing number of STAs. Both IEEE 802.11 and IEEE 802.15.4 have CSMA/CA MACs. However, the adaptation in IEEE 802.11 results in rapid reduction in per-node attempt rate, thus limiting the drop in throughput due to high collisions. On the other hand, in IEEE 802.15.4, the per-node attempt rate flattens out as the number of nodes is increased, leading to high collisions, and lower throughput. We note, however, that IoT networks essentially gather measurements from the sensor nodes, and the measurement rates in most applications are quite small.

6.4.6 References

1. Chandramani Kishore Singh, Anurag Kumar. P. M. Ameer (2007). Performance evaluation of an IEEE 802.15.4 sensor network with a star topology. *Wireless Netw (2008) 14:543–568.*

6.5 Study the 802.15.4 Superframe Structure and analyze the effect of Superframe order on throughput (Level 3)

6.5.1 Introduction

A coordinator in a PAN can optionally bound its channel time using a Superframe structure which is bound by beacon frames and can have an active portion and an inactive portion. The coordinator enters a low-power (sleep) mode during the inactive portion.

The structure of this Superframe is described by the values of macBeaconOrder and macSuperframeOrder. The MAC PIB attribute macBeaconOrder, describes the interval at which the coordinator shall transmit its beacon frames. The value of macBeaconOrder, BO, and the beacon interval, BI, are related as follows:

For $0 \leq BO \leq 14$, BI = aBaseSuperframeDuration * 2^{BO} symbols.

If BO = 15, the coordinator shall not transmit beacon frames except when requested to do so, such as on receipt of a beacon request command. The value of macSuperframeOrder, SO shall be ignored if BO = 15.

An example of a Superframe structure is shown in following Figure 6-30.
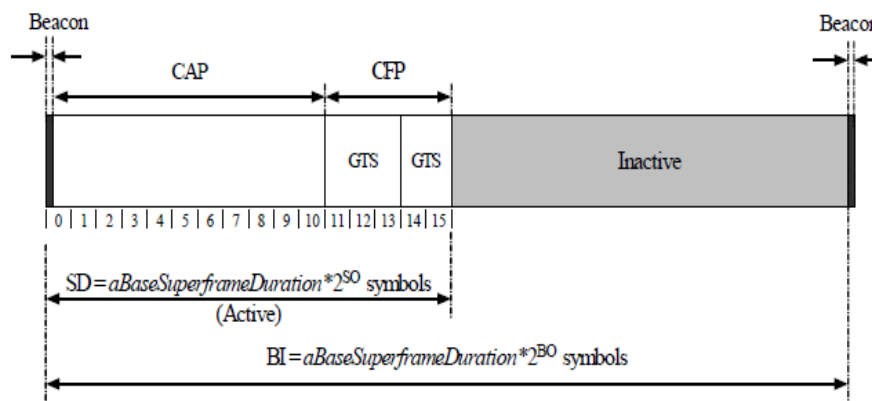
Figure 6-30: An example of the Super Frame structure

Theoretical Analysis

From the above Superframe structure,

$$SuperFrame\ Duration = aBaseSuperframeDuration * 2^{BO}$$

$$Active\ part\ of\ SuperFrame = aBaseSuperframeDuration * 2^{SO}$$

$$Inactive\ part\ of\ SuperFrame = aBaseSuperframeDuration * (2^{BO} - 2^{SO})$$

If Superframe Order (SO) is same as Beacon Order (BO) then there will be no inactive period and the entire Superframe can be used for packet transmissions.

If BO=10, SO=9 half of the Superframe is inactive and so only half of Superframe duration is available for packet transmission. If BO=10, SO=8 then $(3/4)^{th}$ of the Superframe is inactive and so nodes have only $(1/4)^{th}$ of the Superframe time for transmitting packets and so we expect throughput to approximately drop by half of the throughput obtained when SO=9.

Percentage of inactive and active periods in Superframe for different Superframe Orders is given below Table 6-11. This can be understood from Beacon Time Analysis section of the IoT-WSN technology library manual.

Beacon Order (BO)	Super Frame Order (SO)	Active part of Superframe(%)	Inactive part of Superframe (%)	Throughput estimated (%)
10	10	100	0	> 200% of T
10	9	50	50	Say T = 21.07 (Got from simulation)
10	8	25	75	50 % T
10	7	12.5	87.5	25 % T
10	6	6.25	93.75	12.5 % of T
10	5	3.125	96.875	6.25 % of T
10	4	1.5625	98.4375	3.12% of T
10	3	0.78125	99.21875	1.56 % of T

Table 6-11: Inactive and active periods in Superframe for different Superframe

We expect throughput to vary in the active part of the Superframe as sensors can transmit a packet only in the active portion.

6.5.2 Network Setup

Open NetSim and click on **Experiments> IOT-WSN> 802.15.4 Superframe and effect of Superframe order on throughput** then click on the tile in the middle panel to load the example as shown in below Figure 6-31.

Super Frame Order 10 Sample

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 6-32.
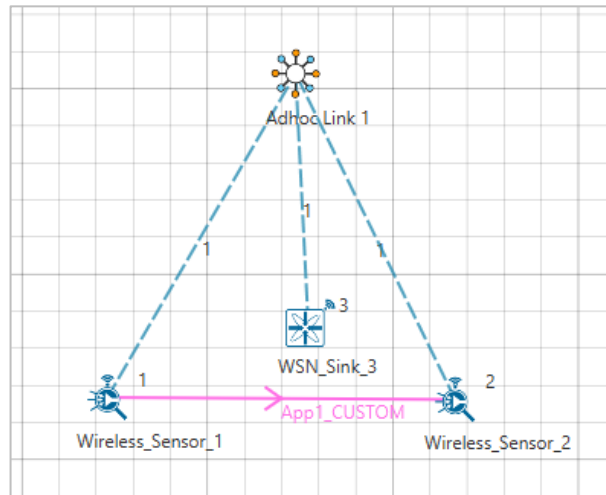

Figure 6-32: Network set up for studying the Super Frame Order 10 Sample

6.5.3 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wireless Sensors and a WSN Sink in the **"Wireless Sensor Networks"** Network Library.

Step 2: Before we designed this network, in the **Fast Config Window** containing inputs for **Grid Settings and Sensor Placement**, the Grid Length and Side Length were set to 500 and 250 meters respectively, instead of the default 50 and 50 meters and we have chosen **Manually Via Click and Drop** option.

Step 3: The **Ad hoc Link** is used to link the Sensors and the WSN Sink in an ad hoc basis.

The Ad hoc link properties is set to **NO PATHLOSS** for the channel characteristics.

Step 4: In the Interface Zigbee > Data Link Layer of WSN Sink, **Beacon Mode** is set to Enable, and **Beacon Order and Super Frame Order** is set to 10 respectively.

Step 5: Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wireless Sensor 1 i.e. Source to Wireless Sensor 2 i.e. Destination with Packet Size set to 25 Bytes and Inter Arrival Time set to 3000 µs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 67 Kbps. Generation Rate can be calculated using the formula:

$$Generation\ Rate\ (Mbps)\ =\ Packet\ Size\ (Bytes)\ *\ 8/Interarrival\ time\ (µs)$$

Step 6: Plots are enabled in NetSim GUI. Run the Simulation for 30 Seconds and note down the **Throughput** value.

Similarly, run the other samples by varying the Super Frame Order to 9, 8, 7, 6, 5, and 4 and note down the throughput values.

6.5.4 Output

The following are the throughputs obtained from the simulation for different Super Frame Orders Table 6-12.

Super Frame Order	Throughput (Kbps)
10	41.68
9	21.07
8	10.5
7	5.25
6	2.63
5	1.30
4	0.62

Table 6-12: Different Super Frame Orders vs. throughputs

To obtain throughput from simulation, payload transmitted values will be obtained from Link metrics and calculated using following formula:

$$Application\ Throughput(in\ Mbps)$$
$$= \frac{Total\ payload\ delivered\ to\ destination\ (bytes) * 8}{Simulation\ Time(Millisecond) - App\ Start\ Time(Millisecond)}$$
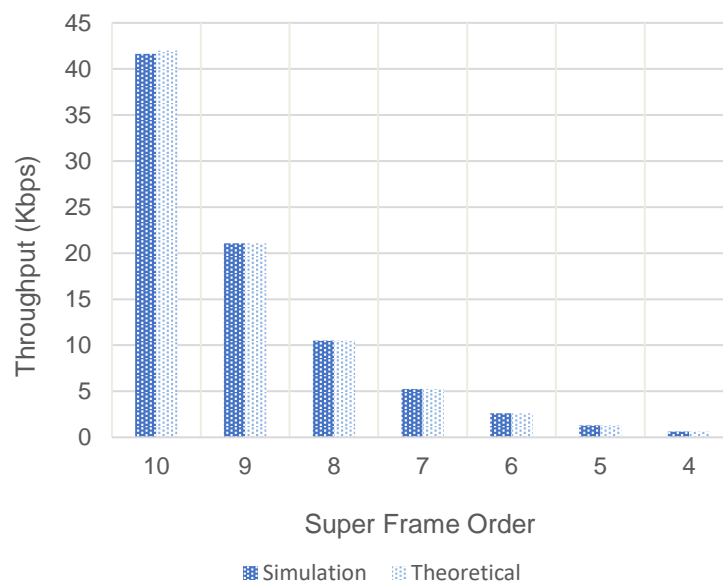
Figure 6-33: Plot of Super Frame Order with Simulated Throughput vs. theoretical analysis

Comparison Chart: All the above plots highly depend upon the placement of Sensor in the simulation environment. So, note that even if the placement is slightly different the same set of values will not be got but one would notice a similar trend.

6.5.5 Inference

From the comparison chart both the simulation and theoretical throughputs match except for the case with no inactive period. A sensor will be idle if the last packet in its queue is transmitted. If a packet is generated in inactive period, then the packet has to wait in the queue till the next Superframe so sensor has packets waiting in its queue and so it cannot be idle in the next Superframe, but if there is no inactive period then there might be no packets waiting in the queue and so sensor can be idle resulting in lesser throughput.

6.5.5 Inference

7 Cognitive Radio

7.1 To analyze how the allocation of frequency spectrum to the Incumbent (Primary), CR CPE (Secondary User) affects throughput (Level 1)

7.1.1 Introduction

An important component of the cognitive radio concept is the ability to measure, sense, learn, and be aware of the parameters related to the radio channel characteristics, availability of spectrum and power, radio's operating environment, user requirements and applications, available networks (infrastructures) and nodes, local policies and other operating restrictions.

NetSim simulator models IEEE 802.22 Cognitive Radio per the theory explained below.

A spectrum hole has been defined as a band of frequencies assigned to a primary user, but at a particular time and specific geographic location, the band is not being utilized by that user. Cognitive Radio was proposed as the means to promote the efficient use of spectrum by exploiting the existence of spectrum holes.
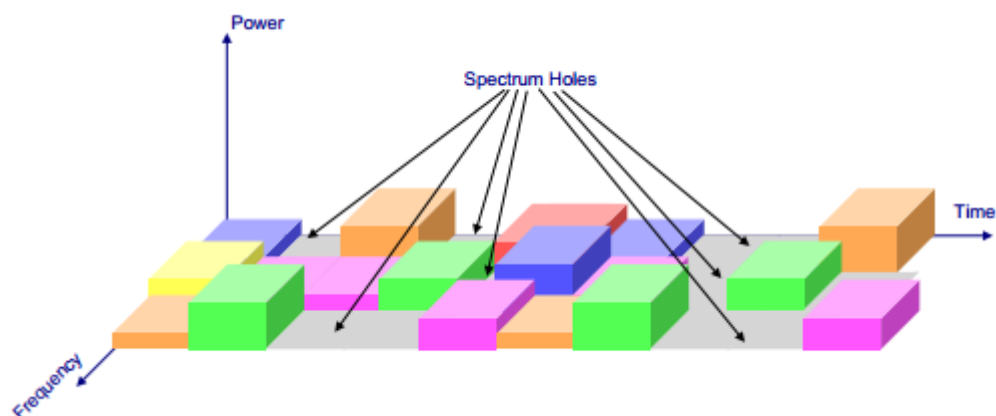
Figure 7-1: Spectrum holes are used by SU for its transmission scheme is often referred to as opportunistic spectrum access (OSA)

These spectrum holes are used by the SU for its transmission. This scheme is often referred to as opportunistic spectrum access (OSA). No concurrent transmission of the PU and the SU is allowed. The SU must vacate the channel as soon as the PU reappears, which leads to the forced termination of the SU connection (if there is no other available channel for the SU). Since the SU has no control over the resource availability, the transmission of the SU is blocked when the channel is occupied by the PU. The forced termination and blocking of a SU connection is shown in the below figure. The forced termination probability and blocking probability are the key parameters which determine the throughput of the SU, and thus its viable existence. The forced

termination depends on the traffic behavior of the PUs and the SUs (e.g. arrival rates, service time etc.). In the case of multiple SU groups with different traffic statistics, the forced termination and blocking probabilities lead to unfairness among the SU groups.
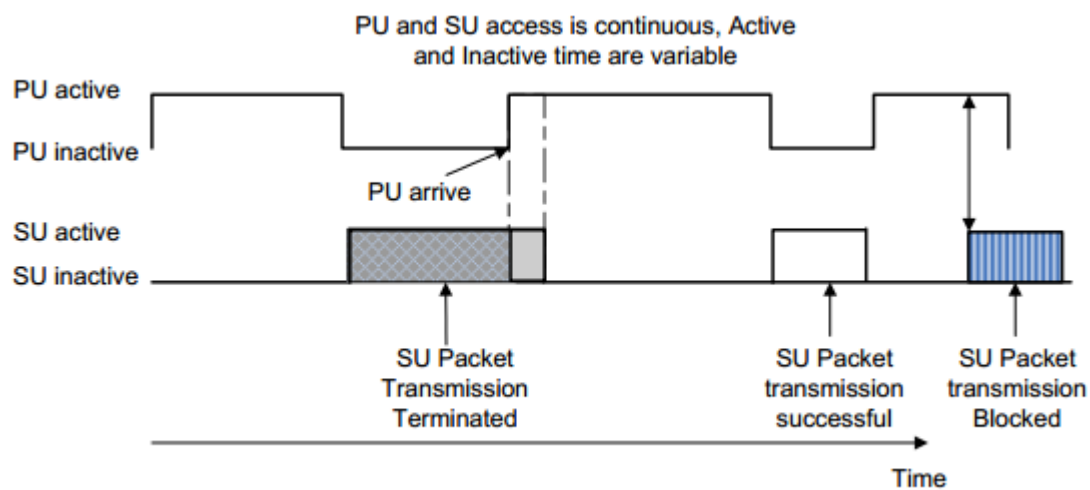

Figure 7-2: Illustration of forced termination and blocking of a SU connection

Performance metrics

The different parameters used to analyze the performance are explained as follows:

- **Throughput**: It is the rate of successfully transmitted data packets in unit time in the network during the simulation.
- **Spectral Efficiency:** It refers to the information rate that can be transmitted over a given bandwidth in a specific communication system. It is a measure of how efficiently a limited frequency spectrum is utilized by the physical layer protocol, and sometimes by the media access control protocol.

7.1.2 Network Setup

Open NetSim and click on **Experiments> Cognitive Radio Networks> Cognitive Radio Impact of frequency allocation to PU and SU on throughput** then click on the tile in the middle panel to load the example as shown in below Figure 7-3.
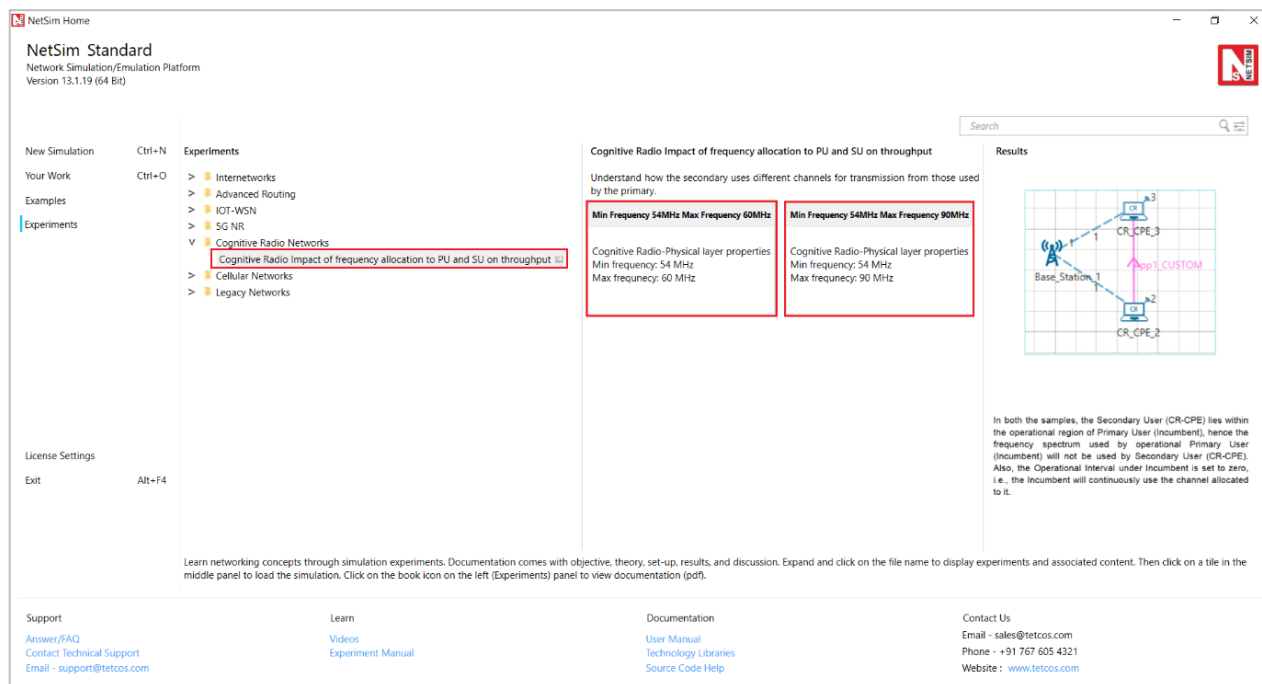
Figure 7-3: List of scenarios for the example of frequency allocation to PU and SU on throughput

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 7-4.
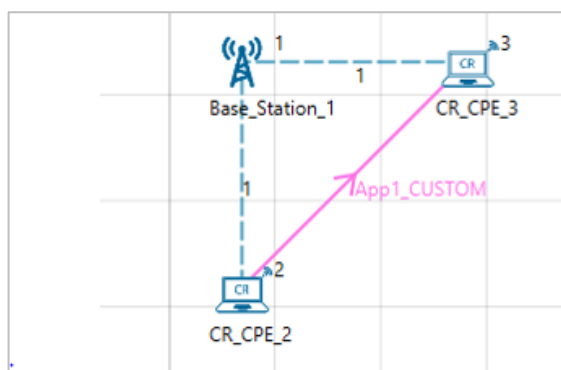

Figure 7-4: Network set up for studying the Frequency allocation to PU and SU

7.1.3 Procedure

Min Frequency 54MHz Max Frequency 60MHz Sample

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 1 Base Station and 2 CR CPE's in the **"Cognitive Radio"** Network Library.

Step 2: The device positions are set as follows Table 7-1.

	Base Station 1	CR CPE 2	CR CPE 3
X/Lat	100	100	120
Y/Lon	100	120	100

Table 7-1: Device positions

Step 3: Go to Base Station Properties **Interface_1(Cognitive Radio)> Datalink Layer > Incumbent 1,** the following are set as shown below Figure 7-5.
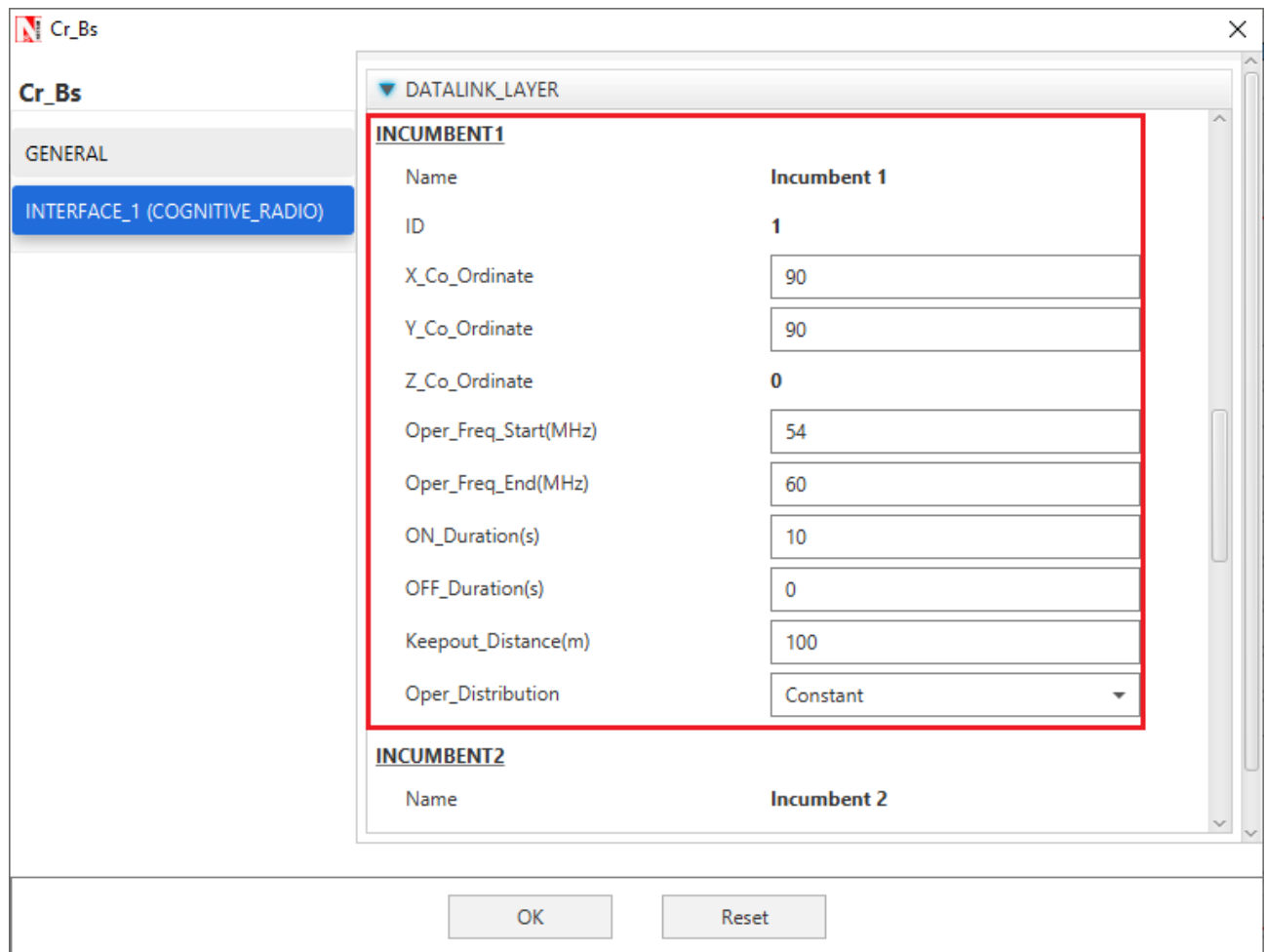
Figure 7-5: Datalink Layer properties window

Step 4: In the **Interface_1(Cognitive Radio)> Physical Layer,** the Min Frequency and Max Frequency parameters are set to 54 and 60 MHz respectively.

Step 5: Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from CR CPE 2 i.e., Source to CR CPE 3 i.e., Destination with Packet Size remaining 1460Bytes and Inter Arrival Time remaining 20000µs.

Step 6: Enable the plots and run the Simulation for 100 Seconds.

Min Frequency 54MHz Max Frequency 90MHz Sample

The following changes in settings are done from the previous sample:

Step 1: In the **Interface_1(Cognitive Radio) > Physical Layer,** the Min Frequency and Max Frequency parameters are set to 54 and 90 MHz respectively.

Step 2: Enable the plots and run the Simulation for 100 Seconds.

7.1.4 Output

Once after the simulation is complete, go to the Results Dashboard and check the **"Application Metrics"** Table. Throughput of the application will be 0.

In the Left-Hand-Side of the Results Dashboard Figure 7-6/Figure 7-7, click on the arrow pointer indicating **"CR Metrics"**, from the drop down select the **"Channel Metrics"** which gives you the Spectral Efficiency.

Min Frequency 54MHz Max Frequency 60MHz Sample

Figure 7-6: Results of Min Frequency 54MHz Max Frequency 60MHz Sample

Min Frequency 54MHz Max Frequency 90MHz Sample

Figure 7-7: Results of Min Frequency 54MHz Max Frequency 90MHz Sample

7.1.5 Inference

In both the samples, the Secondary User (CR-CPE) lies within the operational region of Primary User (Incumbent), hence the frequency spectrum used by operational Primary User (Incumbent) will not be used by Secondary User (CR-CPE). Also, the Operational Interval under Incumbent is set to zero, i.e., the Incumbent will continuously use the channel allocated to it.

In the **Min Frequency 54MHz Max Frequency 60MHz** sample, both the Primary User (Incumbent) and the Secondary User (CR-CPE) has been allocated the same channel (frequency band of 54 - 60 MHz). As Incumbent will continuously use the channel allocated to it, so there will be no Spectrum Hole, hence the secondary user will not be able to transmit any data in an opportunistic manner. Therefore, the throughput of the application in the CR-CPE and the spectral efficiency is almost equal to zero.

In the **Min Frequency 54MHz Max Frequency 90MHz** sample, the Primary User (Incumbent) has been allocated frequency band of 54 - 60 MHz and the Secondary User (CR-CPE) has been allocated the frequency band of 54 - 90 MHz Incumbent will continuously use the channel allocated to it, but the rest channels will remain free i.e. there will be Spectrum Hole, which the CR-CPE will utilize to transmit data.

NOTE: The results are highly dependent on position/velocity/ traffic etc. Any modifications with the above-mentioned input parameters will change the final output result.

8 Cellular Networks

8.1 Study how call blocking probability varies as the load on a GSM network is continuously increased (Level 1)

8.1.1 Network Setup

Open NetSim and click on **Experiments> Cellular Networks> Impact of load on call blocking probability in GSM** then click on the tile in the middle panel to load the example as shown in below Figure 8-1.
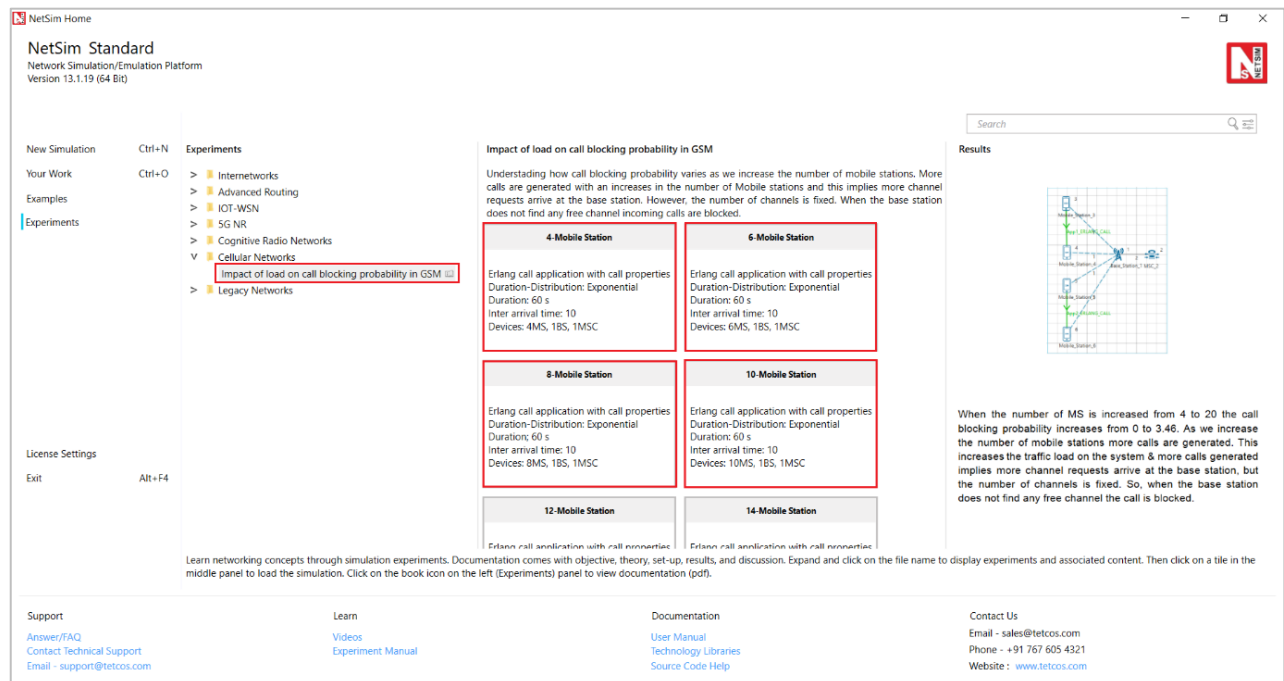

Figure 8-1: List of scenarios for the example of Impact of load on call blocking probability in GSM

NetSim UI displays the configuration file corresponding to this experiment as shown below Figure 8-2.
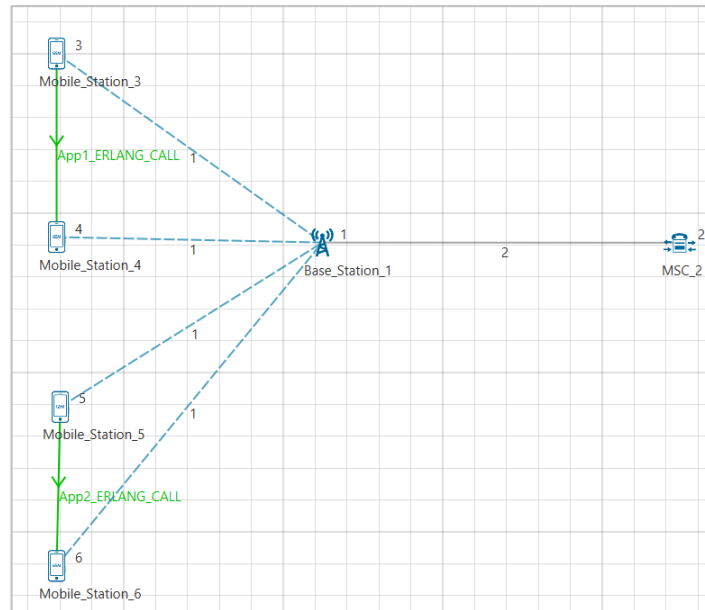
Figure 8-2: Network set up for studying the 4-Mobile Station

8.1.2 Procedure

The following set of procedures were done to generate this sample:

Step 1: A network scenario is designed in NetSim GUI comprising of 4 Mobile Stations, 1 MSC, and 1 Base Station in the **"Cellular Networks"** Network Library.

Step 2: Ensure all the Mobile Stations are placed within the range of Base Station.

Step 3: In the Interface GSM > Data Link Layer Properties of MSC 2, Uplink BW Min and Uplink BW Max are set to 890 MHz and 890.2 MHz respectively Figure 8-3.
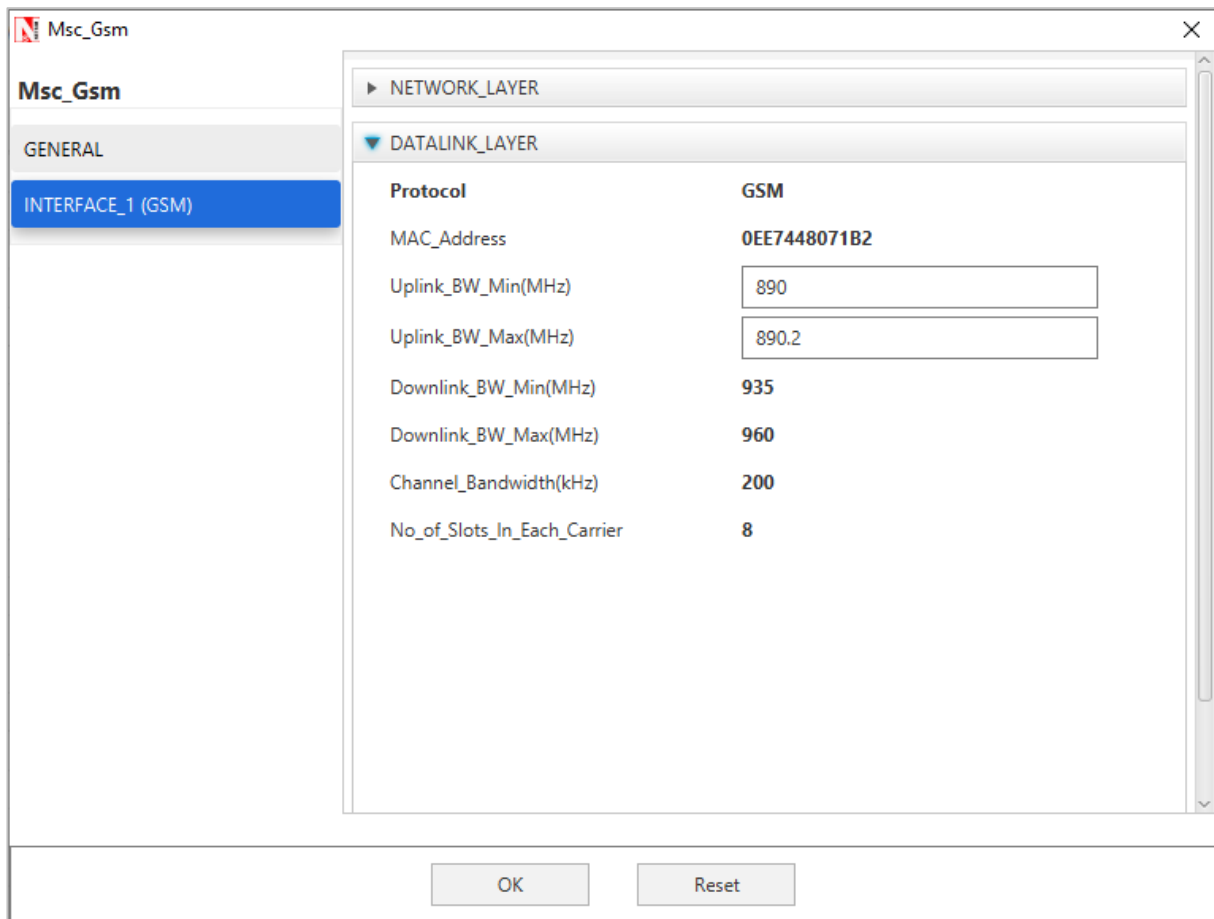
Figure 8-3: Data Link Layer Properties

Step 4: Right click on the Application Flow **App1 ERLANG CALL** and select Properties or click on the Application icon present in the top ribbon/toolbar.

The applications are set as per the below Table 8-1:

Application Properties	Application 1	Application2
Application type	Erlang_call	Erlang_call
Source_Id	3	5
Destination_Id	4	6
Call		
Duration_ Distribution	Exponential	Exponential
Duration(s)	60	60
Inter Arrival Time (sec)	10	10
IAT_ Distribution	Exponential	Exponential
Codec	Custom	Custom
Inter Arrival Time distribution	Constant	Constant
Packet Distribution	Constant	Constant
Service Type	CBR	CBR
Packet Size	33	33
Inter Arrival Time (µs)	20000	20000

Table 8-1: Detailed Application Properties

Step 5: Plots are enabled in NetSim GUI. Run the Simulation for 100 Seconds.

The following changes in settings are done from the previous sample:

Step 1: In the next sample, increase the number of Mobile Stations by 2 and add one more application between them.

Step 2: Plots are enabled in NetSim GUI. Run the Simulation for 100 Seconds.

The following changes in settings are done from the previous sample:

Step 1: Similarly, increase the number of Mobile Stations by 2 up to 20 and set properties for different Samples by adding an application every time and changing Source ID and Destination ID.

Step 2: Plots are enabled in NetSim GUI. Run the Simulation for 100 Seconds.

8.1.3 Output

To view the output, go to the Cellular Metrics. In MS metrics, take sum of call blocking probability (It is the as ratio of Total call blocked to Total call generated).
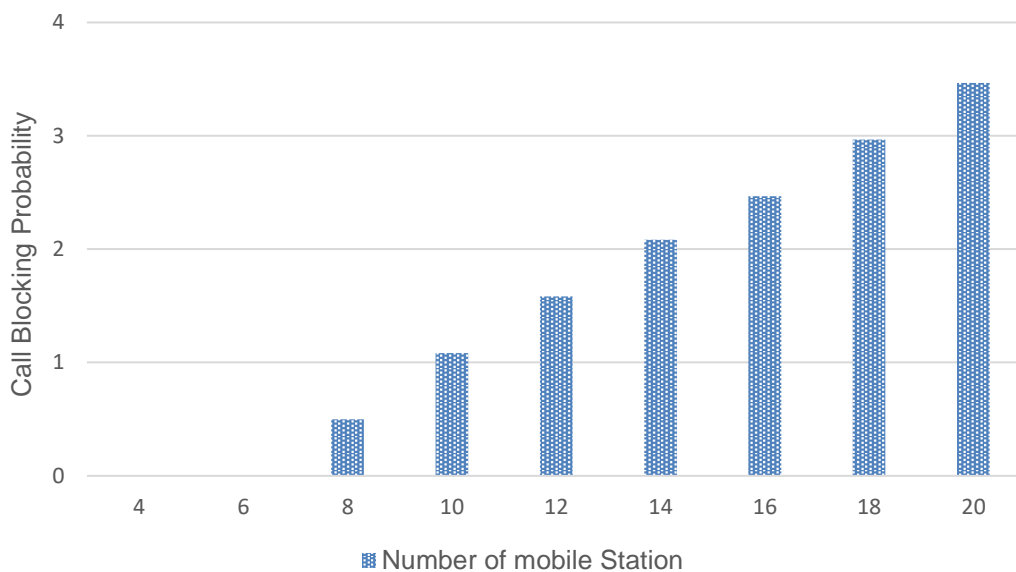
Comparison Charts

Figure 8-4: Plot of Call Blocking Probability vs. Number of Mobile Stations

*** All the above plots highly depend upon the placement of Mobile station in the simulation environment. So, note that even if the placement is slightly different the same set of values will not be got but one would notice a similar trend.

8.1.4 Inference

When the number of MS is increased from 4 to 20 the call blocking probability increases from 0 to 3.46. As we increase the number of mobile stations more calls are generated. This increases the traffic load on the system & more calls generated implies more channel requests arrive at the base station, but the number of channels is fixed. So when the base station does not find any free channel the call is blocked. An additional observation is that the call blocking is zero until 8 MS.

This is because the number of channels is sufficient to handle all call that 6 MS may generate. Only after this the base station does not find free channels and blocks calls.