NetSim®

Accelerate Network R & D

User Manual

A Network Simulation & Emulation Software

By

The information contained in this document represents the current view of TETCOS LLP on the issues discussed as of the date of publication. Because TETCOS LLP must respond to changing market conditions, it should not be interpreted to be a commitment on the part of TETCOS LLP, and TETCOS LLP cannot guarantee the accuracy of any information presented after the date of publication.

This manual is for informational purposes only.

The publisher has taken care in the preparation of this document but makes no expressed or implied warranty of any kind and assumes no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained herein.

Warning! DO NOT COPY

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 NetSim – Introduction

1.1 Introduction to modeling and simulation of networks

A network simulator[1] enables users to virtually create a network comprising of devices, links, applications, etc., and study the behavior and performance of the Network.

Some example applications of network simulators are:

- Protocol performance analysis
- Application modeling and analysis
- Network design and planning
- Research and development of new networking technologies
- Test and Verification

The typical steps followed when simulating any network are:

- **Building the model**: Create a network with devices, links, applications, etc.
- **Running the simulation**: Run the discrete event simulation (DES) and log different performance metrics.
- **Analyzing the results**: Examine output performance metrics such as throughput, delay, loss etc. at multiple levels - network, link, queue, application etc.
- **Developing your own protocol / algorithm**: Extend existing algorithms by modifying the simulator's source C code.

1.2 Versions of NetSim – Academic, Standard & Pro

NetSim is used by people from different areas such as industry, defense, and academics to design, simulate, analyze and verify the performance of different networks.

NetSim is available in three versions: **Academic, Standard** and **Pro**. The academic version is used for lab experimentation and teaching. The standard version is used for R&D at educational institutions while, NetSim Pro version addresses the needs of defense and industry. The Standard and Pro versions are available as components in NetSim v14.2, which users can select and assemble. A comparison of the features in the three versions are tabulated below Table 1-1.

[1] To be technically precise, NetSim is an end-to-end, full-stack, packet level, continuous time, discrete event network simulator.

Features	Academic	Standard	Pro
Technology Coverage			
Internetworks	Yes	Yes	Yes
Legacy & Cellular Networks	Yes	Yes	Yes
Advanced Routing	Yes	Yes	Yes
Mobile Adhoc networks	Yes	Yes	Yes
Software Defined Networks	Yes	Yes	Yes
Wireless Sensor Networks	Yes	Yes	Yes
Internet of Things	Yes	Yes	Yes
Cognitive Radio Networks	Yes	Yes	Yes
LTE Networks	Yes	Yes	Yes
VANET	Yes	Yes	Yes
5G /6G	No	Yes	Yes
Satellite Communication Networks	No	Yes	Yes
Underwater Acoustic Networks	No	Yes	Yes
TDMA	No	No	Yes
5G NTN	No	Yes	Yes
Performance Reporting Performance metrics available for Network and Sub-networks	Yes	Yes	Yes
Packet Trace Available in tab ordered .csv format for easy post processing	Yes	Yes	Yes
Event Trace Available in tab ordered .csv format for easy post processing	No	Yes	Yes
Protocol Library Source Codes with Documentation Protocol C source codes and appropriate header files with extensive documentation	No	Yes	Yes
External Interfacing Interfacing with SUMO	Yes	Yes	Yes
MATLAB	No		
Wireshark	Yes		
Integrated debugging Users can write their own code, link their code to NetSim and debug using Visual Studio	No	Yes	Yes
Plots Allows users to plot the value of a parameter over simulation time	Yes	Yes	Yes
NetSim Animation Allows user to record and play the animation	Yes	Yes	Yes
Simulation Scale	100 Nodes	500 Nodes	2500 Nodes
Custom Coding and Modeling Support	No	Yes	Yes
Emulator (Add on) Connect to real hardware running live application	No	Yes	Yes
Target Users and Segment	Educational (Lab Experimentation)	Educational (Research)	Commercial (Industrial and Defense)

Table 1-1: A comparison of the features of NetSim Academic, Standard and Pro versions

1.3 Components (Technology Libraries) in Pro and Standard versions

Users can choose and assemble components (technology libraries) in NetSim Standard and Pro versions as shown Table 1-2.

Component No	Networks / Protocols Supported	Reference International Standards
Component 1 (Base: Required for all components)	**Internetworks** Ethernet - Fast & Gigabit, ARP, Routing - RIP, OSPF, WLAN - 802.11 a / b / g /p / n / ac & e, Propagation models - HATA Urban / Suburban, COST 231 HATA urban / Suburban, Indoor Home / Office / Factory, Free Space, Log Distance. Shadowing - Constant, Lognormal. Fading - Rayleigh, Nakagami IPv4, Firewalls, Queuing - Round Robin, FIFO, Priority, WFQ, TCP - Old Tahoe, Tahoe, Reno, New Reno, BIC, CUBIC, Window Scaling, SACK UDP **Common Modules** Traffic Generator: Voice, Video, FTP, Database, HTTP, Email, Custom, CBR, Interactive Gaming. Virtual Network Stack, Simulation Kernel, Command Line Interface Command Line Interpreter Metrics Engine with packet and event trace Plot Generator Packet Encryption External Interfaces: MATLAB, Wireshark Network Logs	IEEE 802.3 IEEE 802.11 a/b/g/n/ac/p/e RFCs 2453, 2328, 826, 793, 2001 and 768
Component 2	**Legacy & Cellular Networks** Aloha – (Pure & Slotted) GSM CDMA	3GPP, ETSI, IMT-MC, IS-95 A/B, IxRTT, 1x-EV-Do, 3xRTT
Component 3	**Advanced Routing** Access Control Lists, Detailed Layer 3 switch mode, Virtual LAN (VLAN), Public IP, Network Address Translation (NAT)	IETF RFC's 1771 & 3121
Component 4	**Mobile Adhoc Networks** Standard MANET, Interconnected MANETs, MANET - DSR, AODV, OLSR, ZRP	IETF RFC 4728, 3561, 3626
Component 5	**Software Defined Network (SDN)**	Based on Open Flow v1.3
Component 6 (Requires C4)	**Internet of things (IOT)** with RPL protocol Wireless Sensor Networks (WSN)	IEEE 802.15.4 MAC, MANET in L3 RFC 6550
Component 7	**Cognitive Radio Networks** WRAN	IEEE 802.22
Component 8	**Long-Term Evolution Networks:** LTE	3GPP
Component 9 (Requires C4)	**VANETs:** IEEE 1609 WAVE, Basic Safety Message (BSM) protocol per J2735 DSRC, Interface with SUMO for road traffic simulation	IEEE 1609

Component 10 (Requires C3, C8)	**5G NR :**3GPP 38 Series. Full Stack covering SDAP, PDCP, RLC – UM, TM, MAC, PHY – FR1 and FR2, mmWave propagation.	3GPP 38.xxx
Component 11 (Requires C3)	**Satellite Communication Networks:** Geo Stationary Satellite. Forward link TDMA in Ku Band and Return link MF-TDMA in Ka band per DVB S2. Markov Loo Fading model. Device models for Satellite, Satellite Gateway, and Satellite User Terminals	DVB S2
Component 12 (Requires C2, C3)	**Underwater Acoustic Network:** Acoustic PHY Model, Propagation based on speed of sound, Thorp propagation (temperature, depth, salinity), Slotted ALOHA in MAC, UDP in L4, Underwater sensor application	----
Component 13 (Requires C10)	**5G NTN:** LEO/MEO/GEO single satellite simulation, Downlink transmission, Link budget calculations, Interference models, Feeder link and service link, Antenna Configuration, Frequence reuse: FR1 and FR3 Frequency bands: S-band and Ka-band	TR 38.821 and TR 38.811
TDMA Radio Networks Add on (Pro version only)	**TDMA Radio Networks:** Standard DTDMA, Interconnected DTDMA TDMA Link 16, Dynamic TDMA, Frequencies – HF, VHF, UHF Bands, Frequency Hopping	----
Network Emulator Add On (Interfaces with all components except C2)	**Network Emulator** Connect real hardware running live applications to NetSim Simulator. IP based, data plane, flow through emulator.	----

Table 1-2: Different Components (Technology Libraries) in Pro and Standard versions of NetSim

2 Installation and License Server Set-up

2.1 System Requirements

2.1.1 NetSim Client (installs locally)

- Hardware: i5 equivalent or above, RAM: 8 GB (Min). 16GB Recommended.
- Monitor Resolution: Minimum – 1366x768, Maximum – 1920x1028. Optional Scale and layout setting: 100%
- Operating System: 64 bit. Win 10, Win 11, Language English
- Software: MS Office, Adobe Reader
- Development Tools: Visual Studio
    - NetSim v8 / v8.1 / v8.3 / v9 / v9.1: Microsoft Visual Studio 2010 (or higher)
    - NetSim v10 / v11 / v11.1: Microsoft Visual Studio 2015 (or higher)
    - NetSim v12 / v12.1 / v12.2: Microsoft Visual Studio 2019 (or higher)
    - NetSim v13 / v13.1 / v13.2 / v13.3: Microsoft Visual Studio 2022 (or higher)
    - NetSim v14.0 / v14.1 / v14.2 / v14.3/v14.4: Microsoft Visual Studio 2022

Visual Studio Community edition (or higher) is required for writing and debugging custom code.

2.1.2 License Server

This is applicable when running Host-ID/Dongle-locked floating licenses and is not applicable for node-locked licenses.

Any one system will have to be made as the license server, and it is to this PC that the license is locked, either via its MAC ID or via a dongle. The dongle is a USB device which controls the licensing. The system(hardware/OS) requirements are same as that applicable for NetSim clients. USB Port is required for connecting and running the dongle. Client systems should be able to communicate with license servers through the network.

2.2 Installing NetSim

Install 64-bit build of NetSim. The start window will show (i) Version type (Pro, Standard, Academic), (ii) Version Number and build number (Eg: 14.4) followed by (iii) Currently supports 64bit in v14.4.

For example, you will see **Network Simulator** for a Standard version to install. Note that the installation must be performed in administrator mode. Right click on the setup file and click on the **Yes** to proceed with the installation.

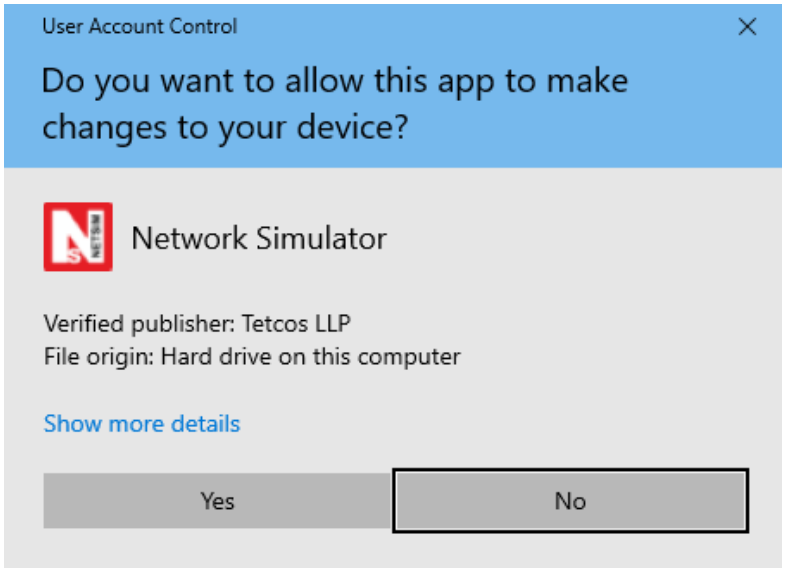


Figure 2-1: User Account Control message window appears and select Yes.

Setup prepares the installation wizard and software installation begins with a **Welcome Screen**. Click on **Next** to continue with the installation.

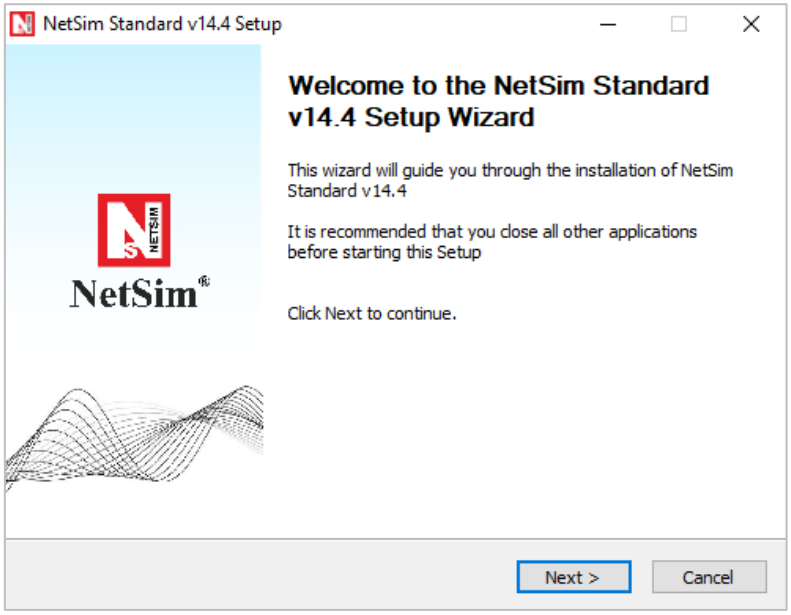


Figure 2-2: Select Next to continue with the installation.

License agreement will be displayed. Read the agreement carefully, scroll down to read the complete license agreement. Click on **I Agree** else quit the setup by clicking **Cancel**.

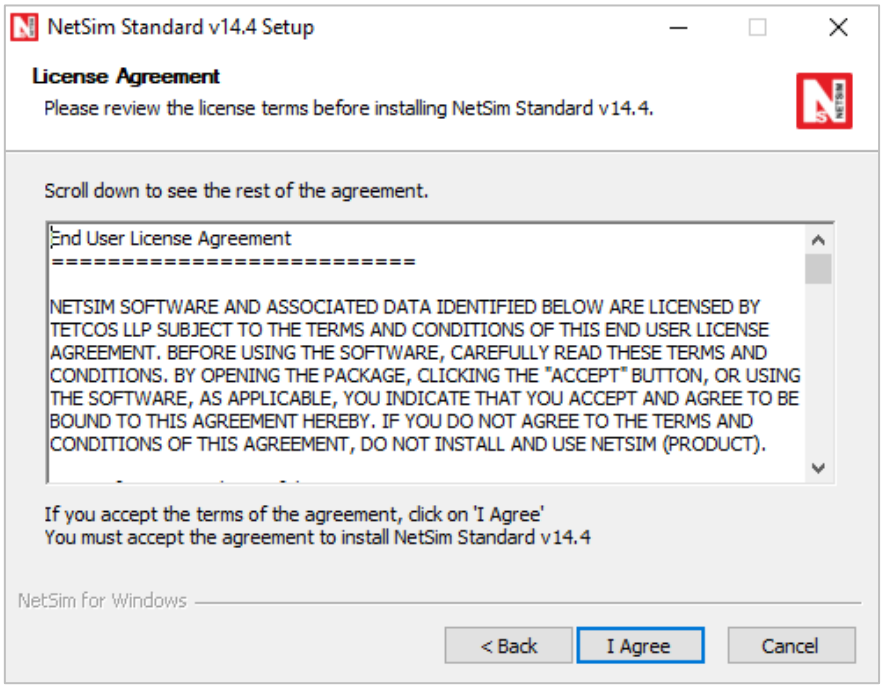


Figure 2-3: Select I Agree in NetSim License Agreement window

If you agree with the license agreement, you will be prompted to select either one of the installation options, Express (Single-click installation) or Custom (Step-by-Step installation).

Express Installation will install the third-party tools silently along with NetSim without displaying any prompts for the user.

Custom Installation is a step-by-step approach in which a user will be prompted to carry out the installation process and the same applies to the installation of the third-party tools which happens alongside NetSim. Both the installation methods are explained below:

2.2.1 Express Installation

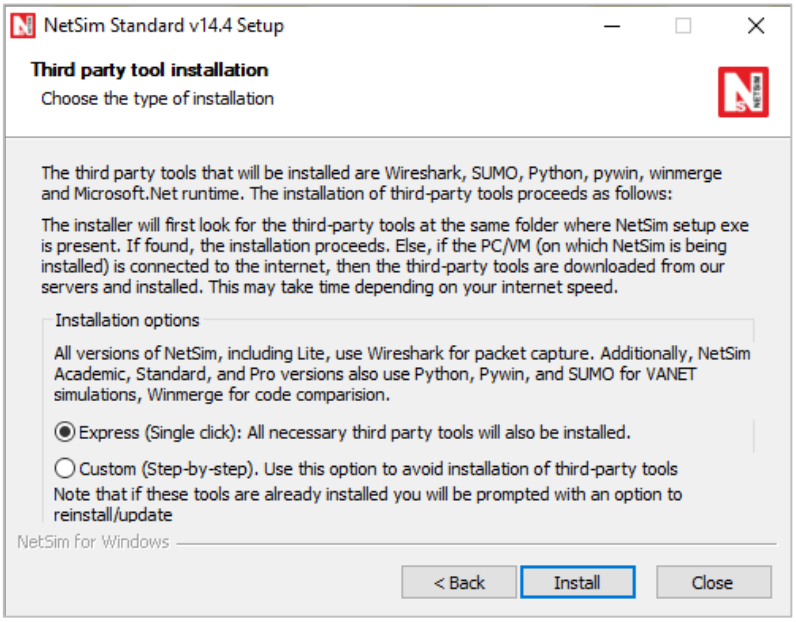


Figure 2-4: Select Express (Single click) radio and click on install

NetSim installation starts, and users can see that the third-party tools download information window click on OK to proceed with the installation.

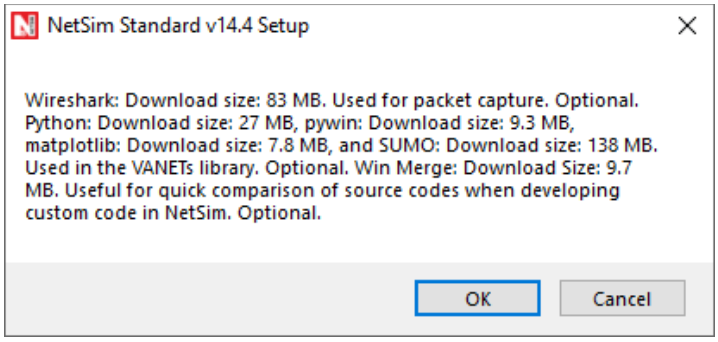


Figure 2-5: Click on the OK to proceed installation process of NetSim

Third-party tools, including Wireshark, SUMO, Python, WinMerge, PyWin, and Microsoft .NET, will begin installation .Before that, the installer will look for the third-party tools at the same folder where NetSim.exe is present if found, the next step of installation proceeds.

Else, the third-party tools will get downloaded from our NetSim servers and installed if the PC/VM is connected to the Internet

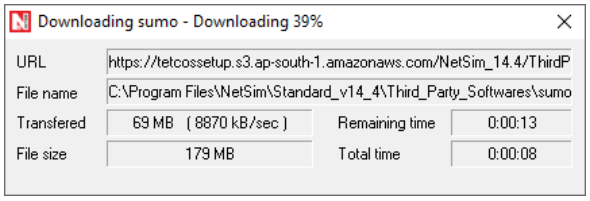


Figure 2-6: Sumo is being downloaded.

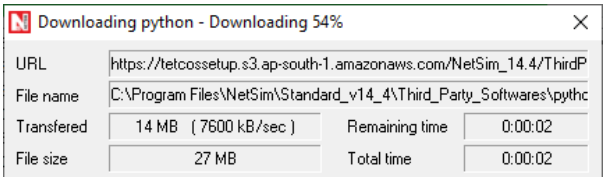


Figure 2-7: Python is being downloaded

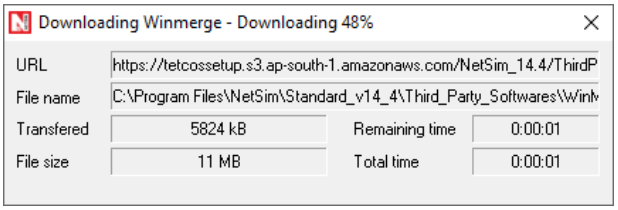


Figure 2-8: Winmerge is being downloaded

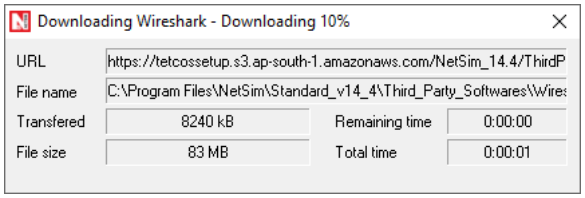


Figure 2-9: Wireshark is being downloaded

NetSim installation starts, and users can see that the third-party tools get installed one by one.

Pywin and matplotlib will be installed through command line.

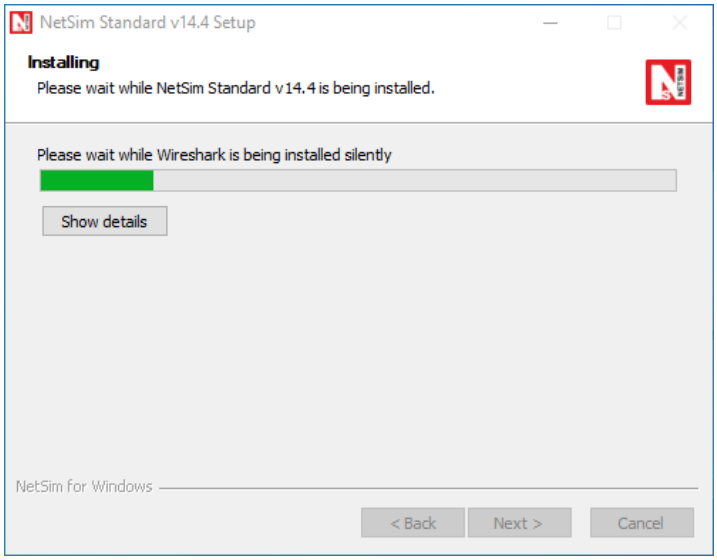


Figure 2-10: Wireshark gets installed silently

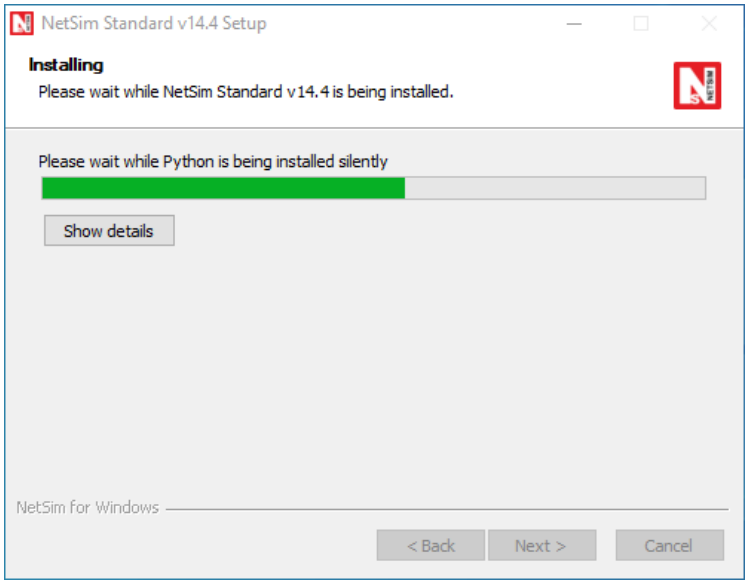


Figure 2-11: Python gets installed silently

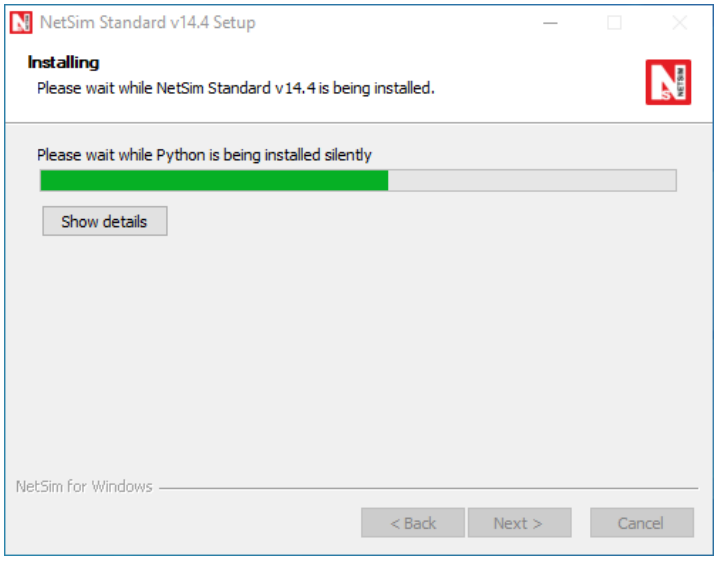


Figure 2-12: Sumo gets installed silently

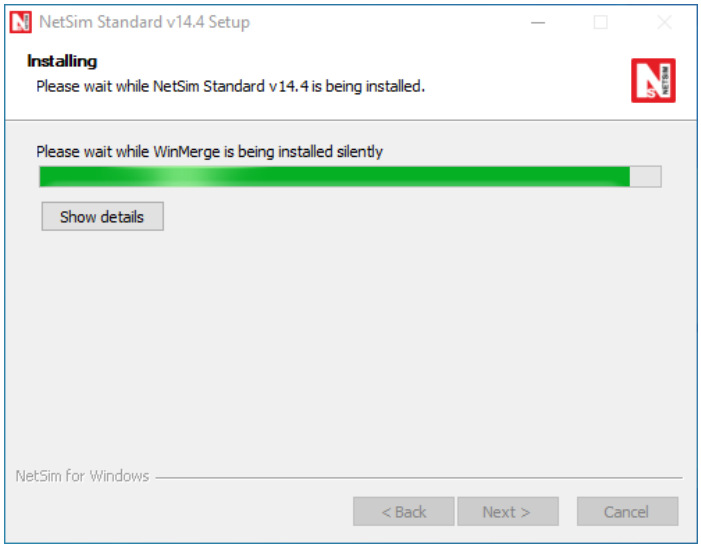


Figure 2-13: Winmerge gets installed silently

After the third-party installations, NetSim installation proceeds. Once it is completed, NetSim-complete setup wizard appears as shown below. Click on the **Finish** to complete the installation process of NetSim.

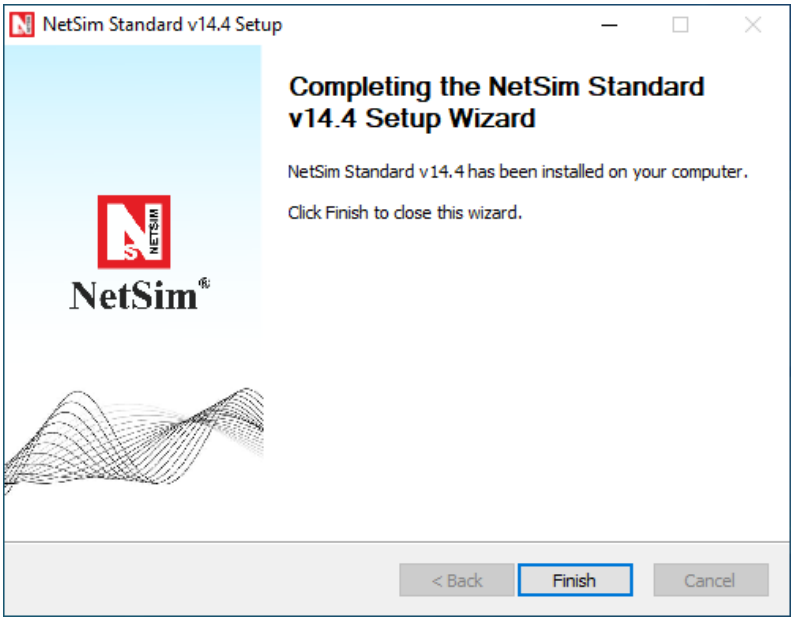


Figure 2-14: Select Finish to complete the installation process of NetSim.

2.2.2 Custom (Step-by-step) installation

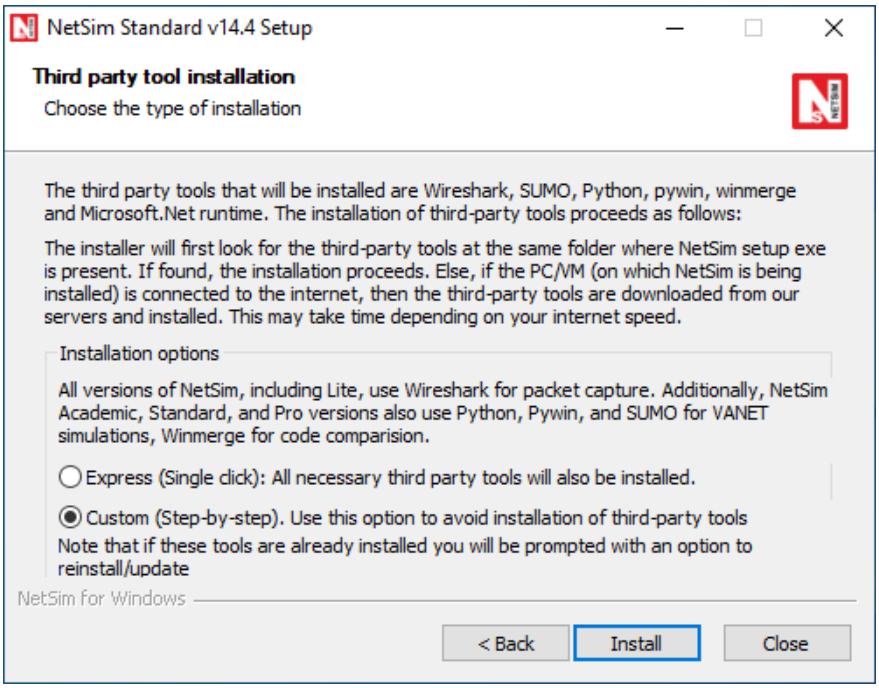


Figure 2-15: Select Custom Radio

Now the user will be prompted to select the components to be installed. The list of components is available for selection and assembly only in the Standard and Pro versions of NetSim. NetSim Academic version is available as a single package.

Note: In Standard and Pro Versions of NetSim, the Choose Components screen will display only those components for which the licenses are obtained by the user. Network Emulator is available as add-on.

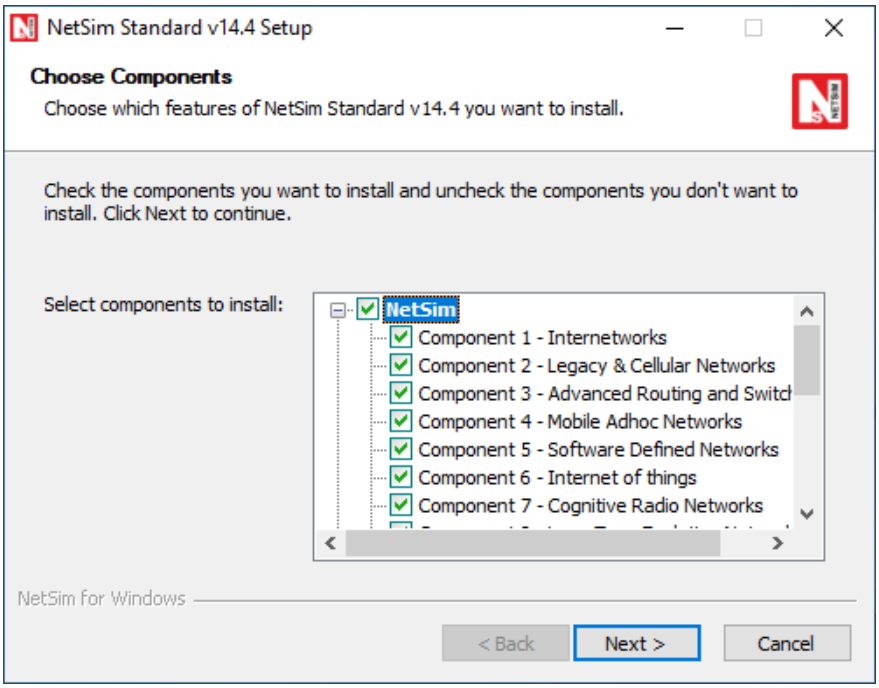


Figure 2-16: list of components is available for selection and assembly only in the Standard and Pro versions

Note: Select all the supporting applications for complete installation of the software as shown below:

Click on the **Next**.

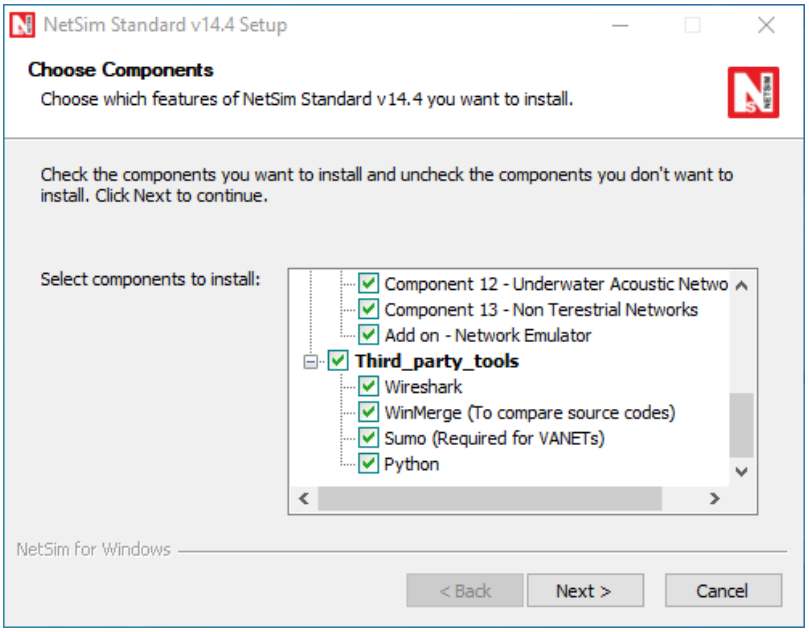


Figure 2-17: list of third-party tools

Note: Winmerge comes only as a part of Standard and Pro Version Install.

In the next screen, you will be requested to enter the installation path. Select the path in which the software needs to be installed and click on **Next** .

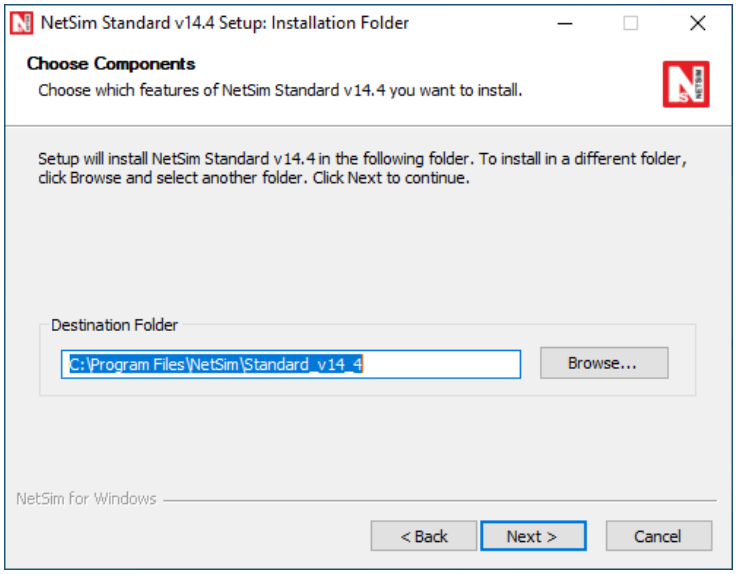


Figure 2-18: NetSim installation directory path

In the next screen, you will be requested to enter the Start Menu folder name. By default, it shows **NetSim Standard** for Standard version install of NetSim. Click on the **Install** to start the installation.

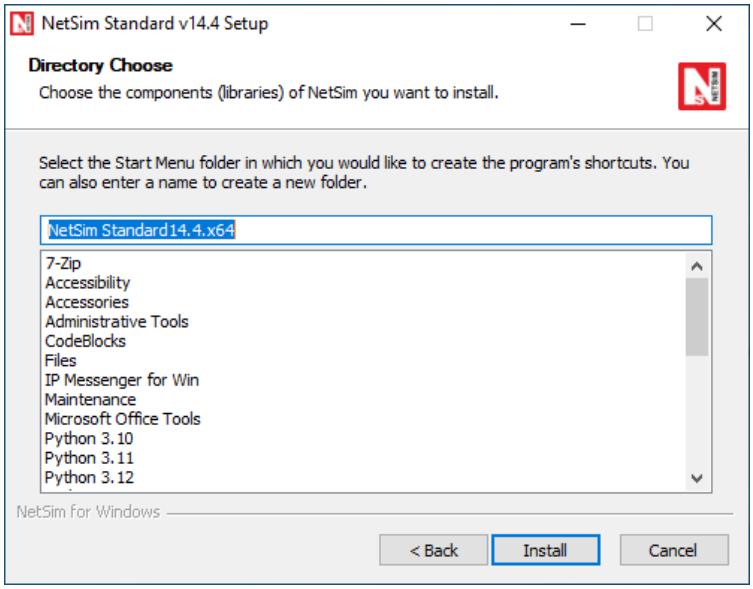


Figure 2-19: Start Menu folder name

The installation process begins.

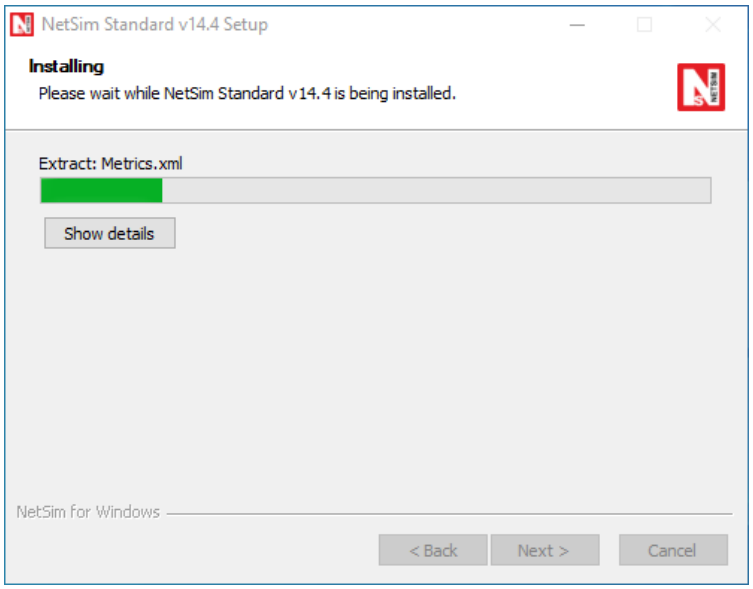


Figure 2-20: NetSim std v14.4 being installed.

After the installation of required NetSim files, the installation of **third-party tools** begins.

For NetSim Academic Version, Npcap and Wireshark will be installed.

For NetSim Standard and Pro Versions, along with WinPcap and Wireshark installation, Dot net, Sumo, Python installation will start automatically. (If not deselected during 3rd party software selection)

If the PC/VM is connected to the Internet third party tools will get downloaded from our NetSim servers (If the third-party tools are not found in folder where NetSim.exe is present) and proceeds with installation.

Click on **Install** to start Dot NET (.NET) installation

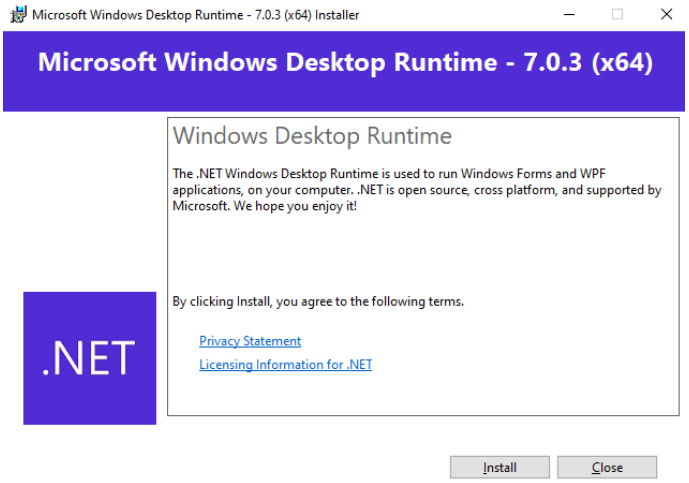


Figure 2-21: Select install to install Dot NET (.NET)

Installation process begins.

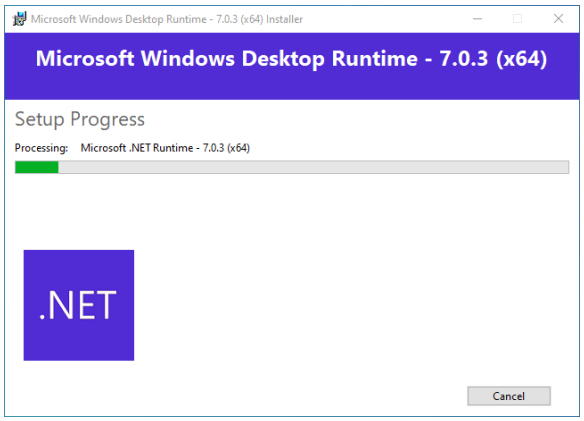


Figure 2-22: Dot NET (.NET) installation begins

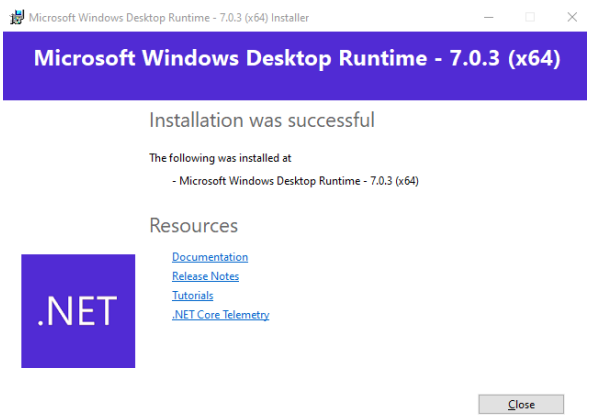


Figure 2-23: Dot NET (.NET) installation successfully completed

After the successful installation of Dot NET (.NET) and click on close then Wireshark installation window appears. Click on **Next** to begin

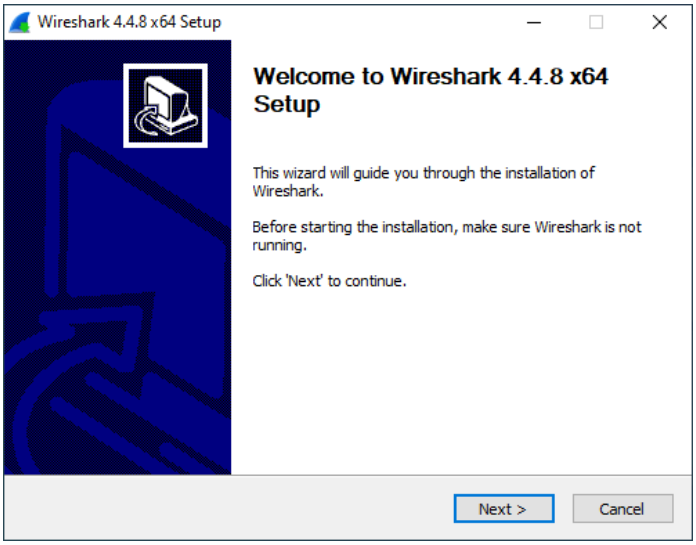


Figure 2-24: Select Next to start Wireshark installation

Wireshark **License Agreement** appears. Click on the **I Agree**.

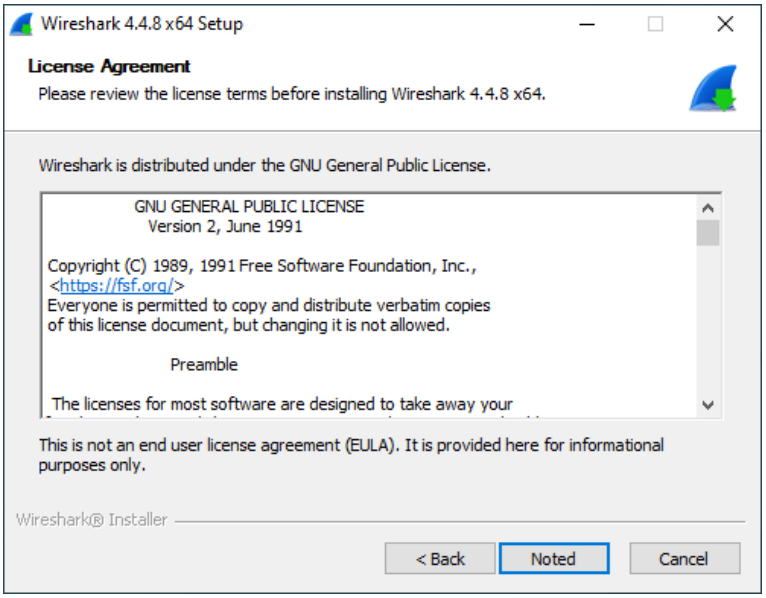


Figure 2-25: Wireshark License Agreement window

Make sure that all the components are selected and click on **Next**.

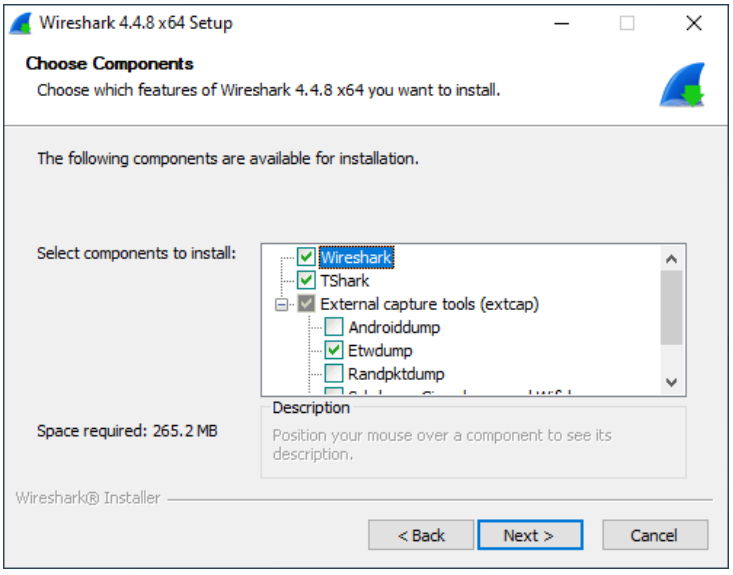


Figure 2-26: Choose Wireshark features

Click on **Next**.

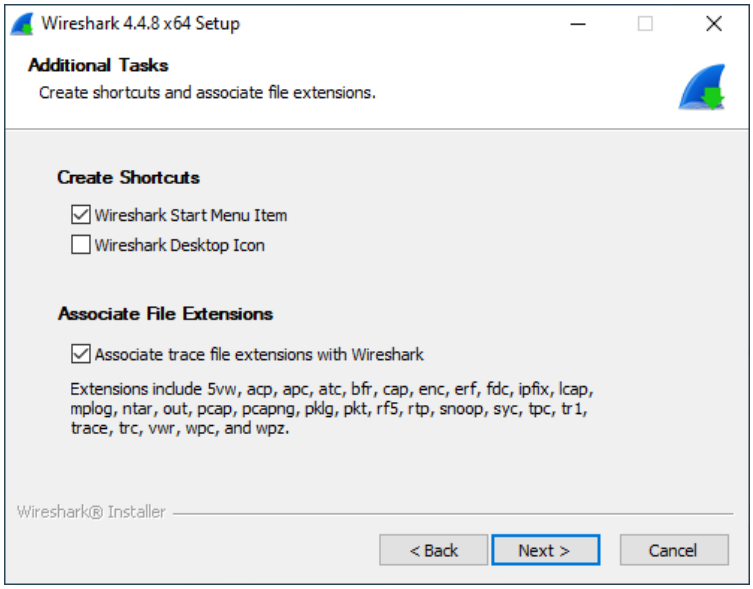


Figure 2-27: Select Next

Select the path in which Wireshark needs to be installed and click on **Next**.

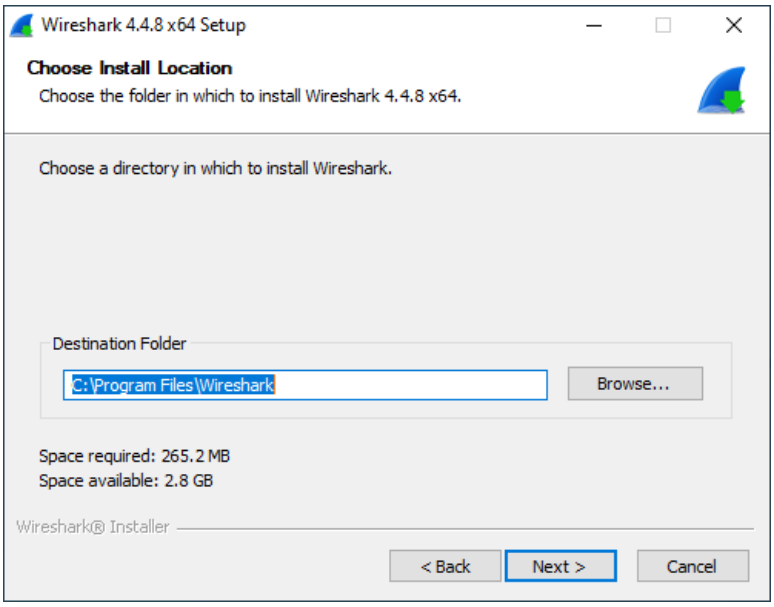


Figure 2-28: Wireshark installation directory path

Select **Install Npcap 1.80** and click on **Next**.

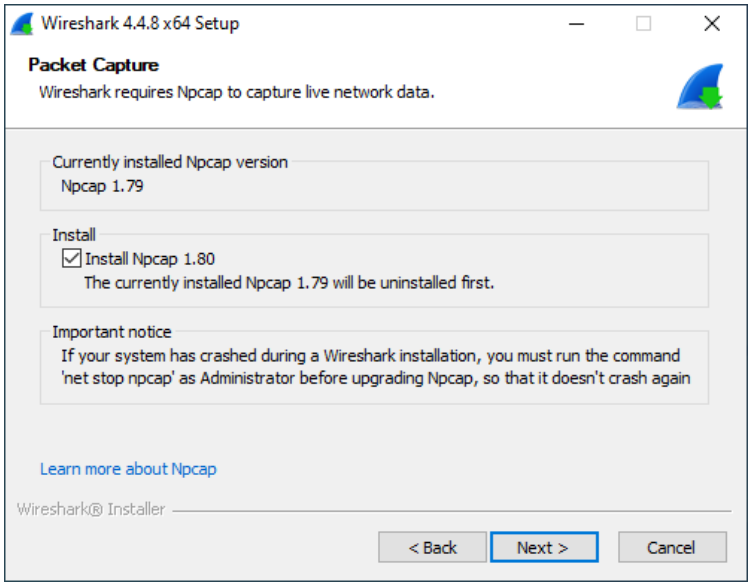


Figure 2-29: Select Install Npcap 1.80 in Wireshark window

Select **Install USBPcap 1.5.4.0** and click on **Install**.

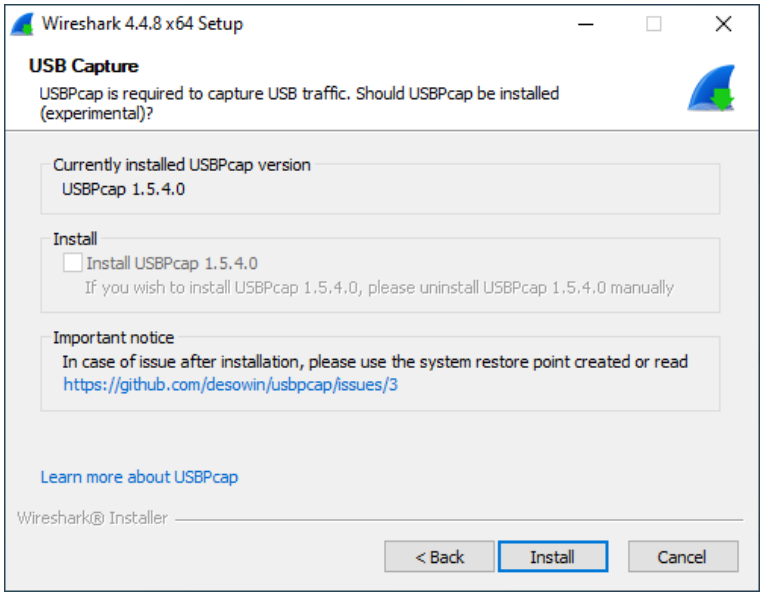


Figure 2-30: Select Install USBPcap 1.5.4.0 in Wireshark window

The installation process begins.

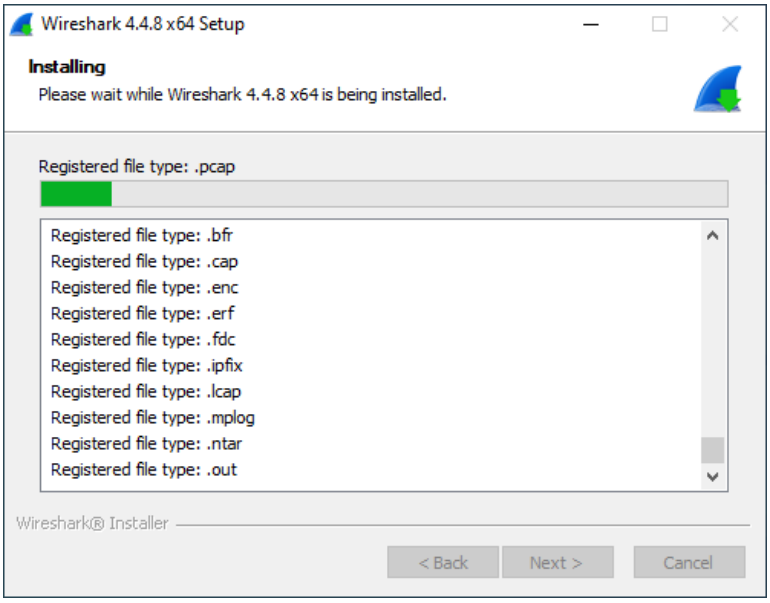


Figure 2-31: Wireshark installation process begins

Npcap License Agreement window appears. Click on **I Agree** and proceed with the installation.

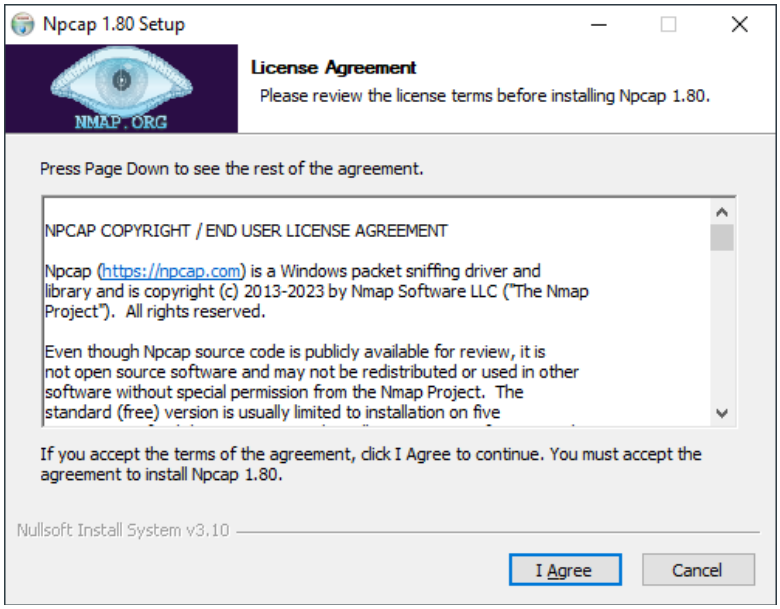


Figure 2-32: Npcap License Agreement window

Installation Options window appears. Simply click on **Install**.

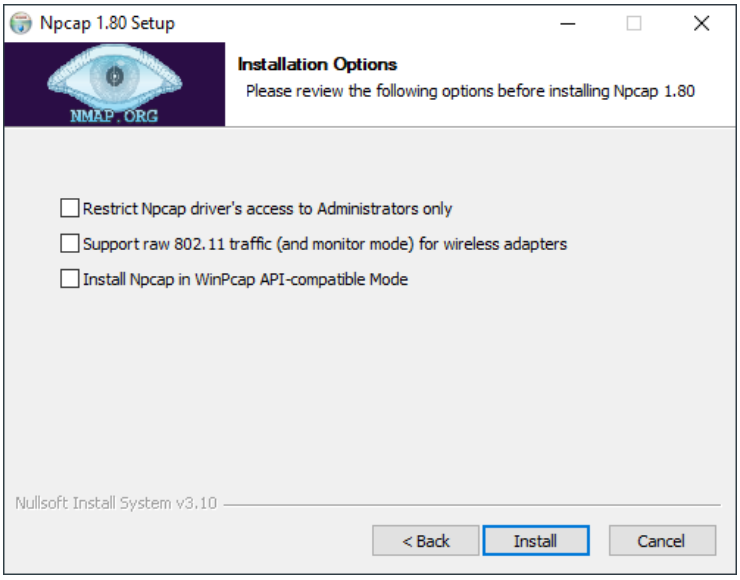


Figure 2-33: Npcap Installation Options window

Npcap Installation begins.

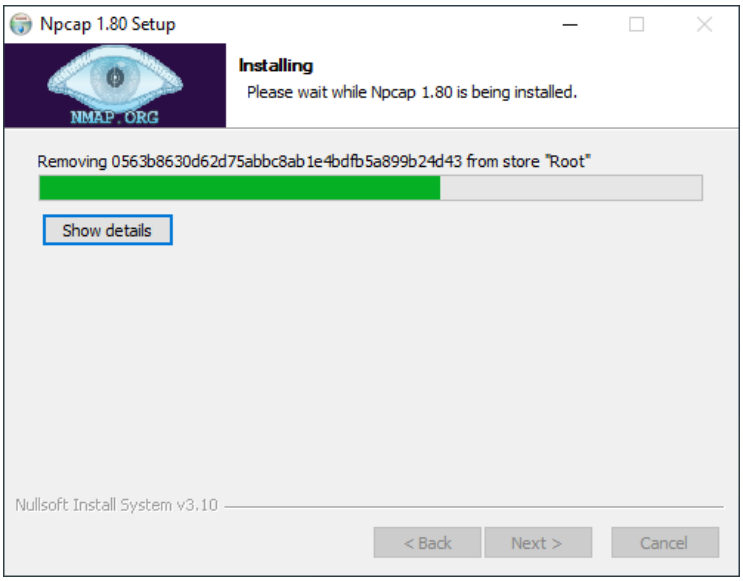


Figure 2-34: Npcap installation

The Installation Complete window appears. Click on **Next** to proceed.

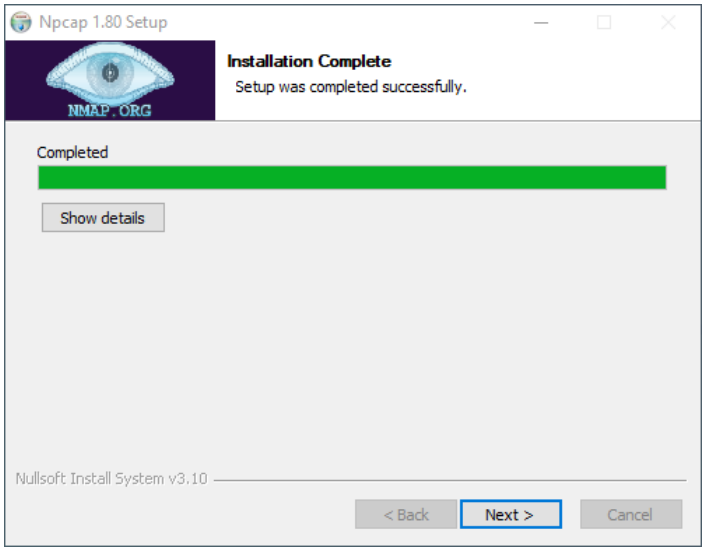


Figure 2-35: Npcap installation is complete

USBPcap Driver License Agreement window appears. Click on I accept the terms of the License Agreement check box and click on **Next**.

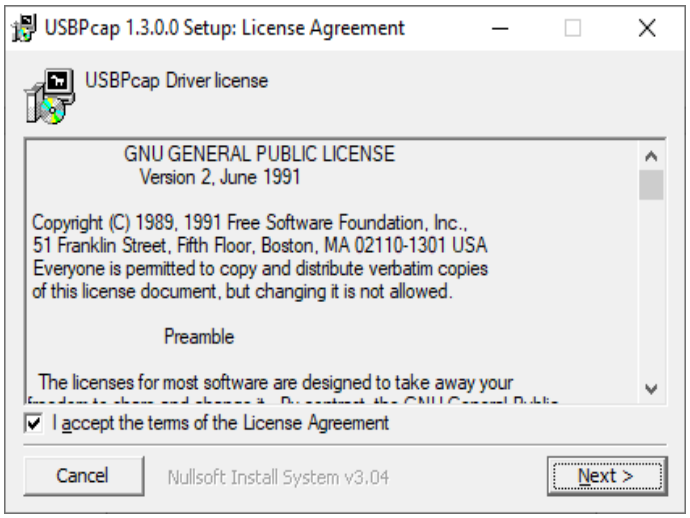


Figure 2-36: USBPcap Driver License Agreement window

USBPcap CMD License Agreement window appears. Click on I accept the terms of the License Agreement check box and click on **Next .**

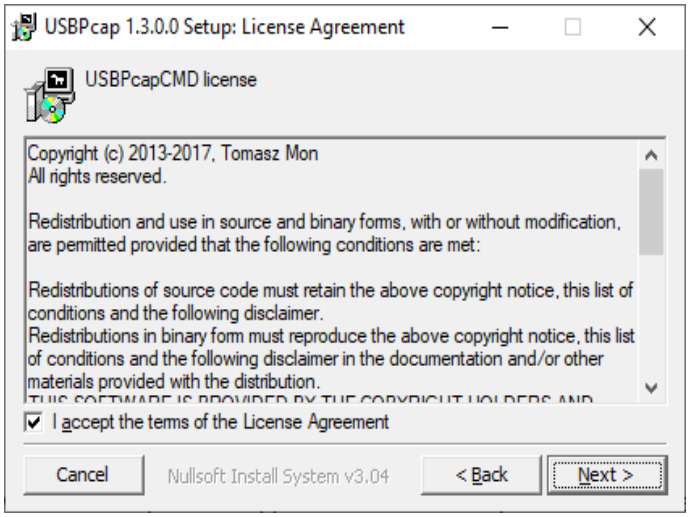


Figure 2-37: USBPcap CMD License Agreement window

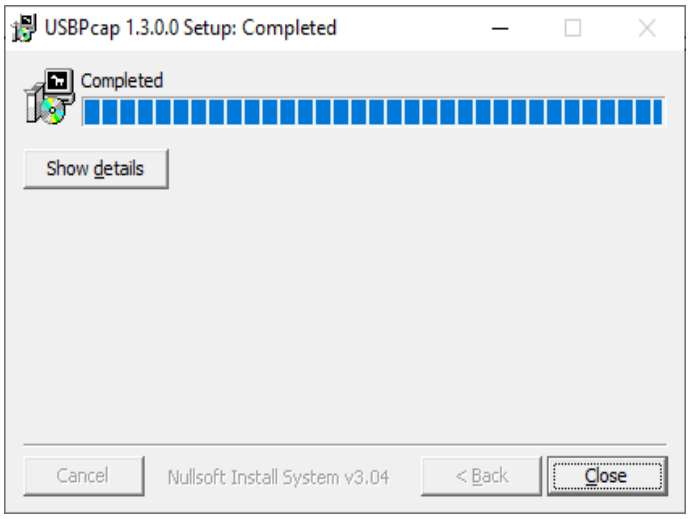


Figure 2-38: USBPcap installation is completed

The Installation Complete dialog box appears once the installation process is completed successfully. Click on the **Next** .

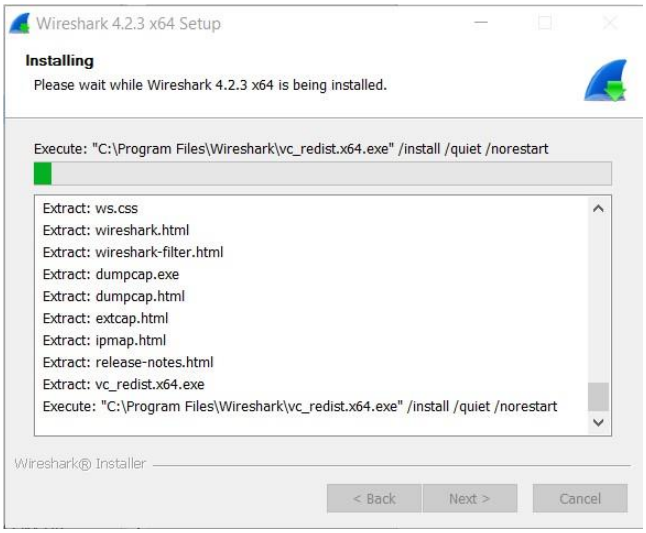


Figure 2-39: Installation Complete dialog box and select next

You will get the Wireshark Completing Setup window. Select the option **I want to manually reboot later**.

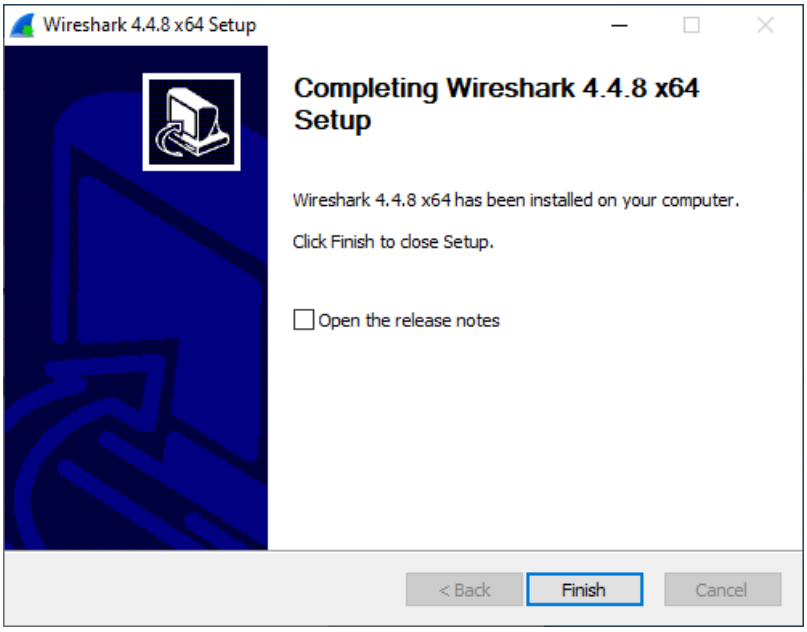


Figure 2-40: Select the option I want to manually reboot later and Click on Finish

This completes the Installation of Wireshark software. NetSim complete Setup wizard appears as shown above. After click on **Finish** to begin with WinMerge installation.

Next the WinMerge **License Agreement** appears. Click on **Next.**

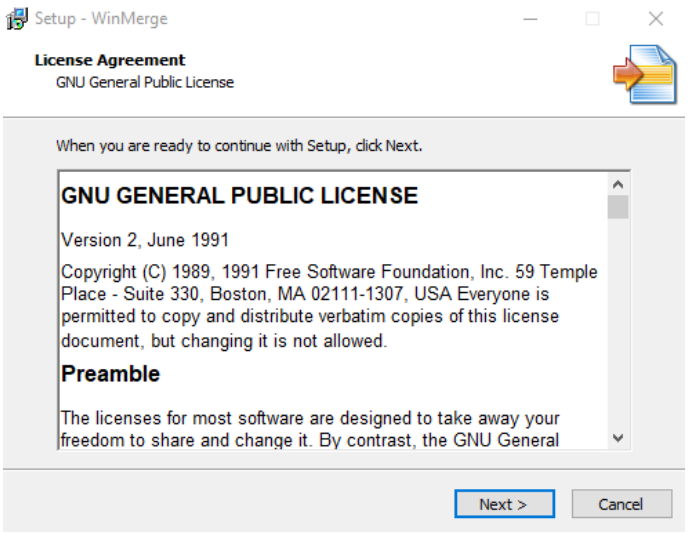


Figure 2-41: WinMerge License agreement window

Select the path in which WinMerge needs to be installed and click on **Next** .

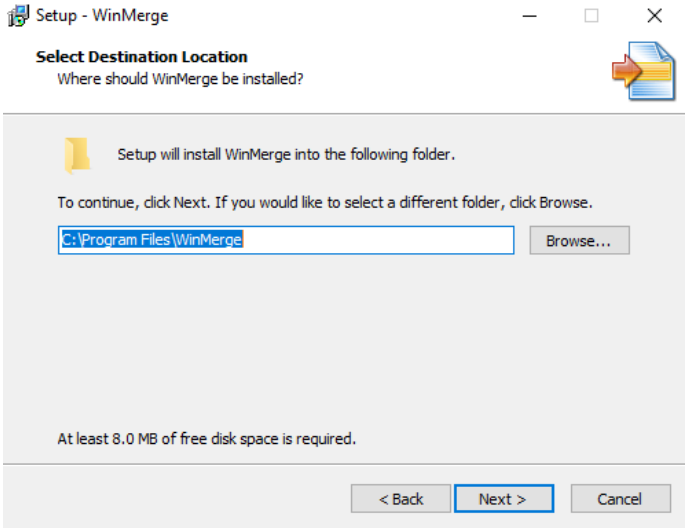


Figure 2-42: Select the location where should WinMerge be installed

Once WinMerge installation completes, click on **Finish** .

Figure 2-43: Click on Finish to completes WinMerge installation

Click on **Next** to start SUMO installation.

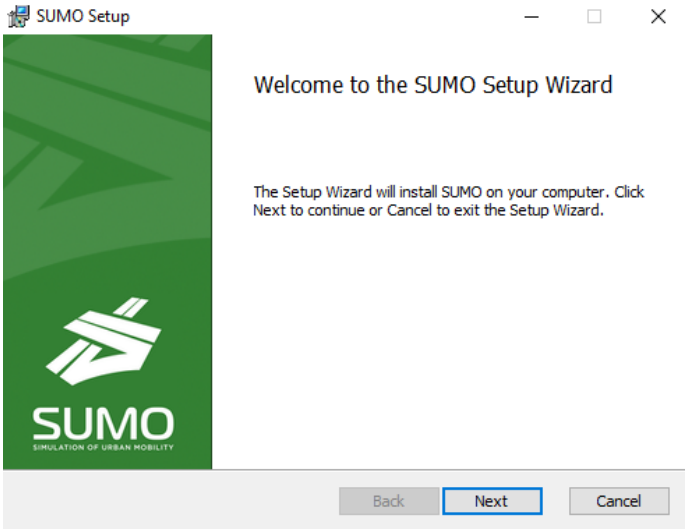


Figure 2-44: Sumo Installation starts

SUMO **License Agreement** appears. Accept the terms in license agreement and click on Next to proceed installation

Figure 2-45: SUMO License Agreement window

Once SUMO installation completes, click on **Finish**.

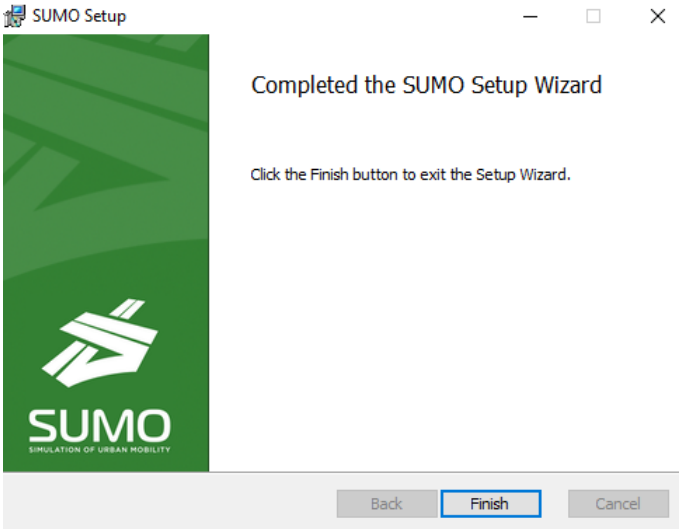


Figure 2-46: Complete SUMO Installation

Click on **Next** to start with Python 3.12.2 installation.

Figure 2-47: Select “install Now” option to install Python

The installation begins once you click on the **Install** option.

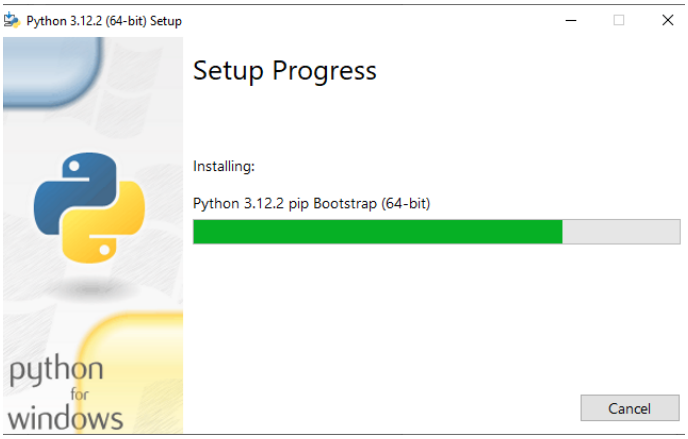


Figure 2-48: Python installation begins.

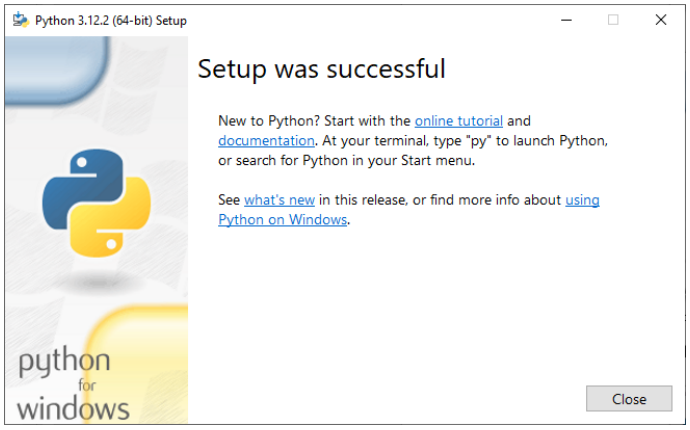


Figure 2-49: Python installation successfully completed.

Once the installation is finished, click on **Close** to start the installation pywin 32

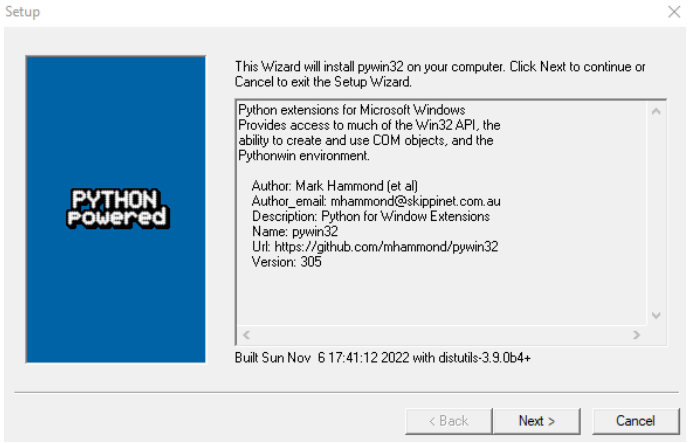


Figure 2-50: pywin 32-224 installation wizard window

Click on **Next** to select the directory to be used.

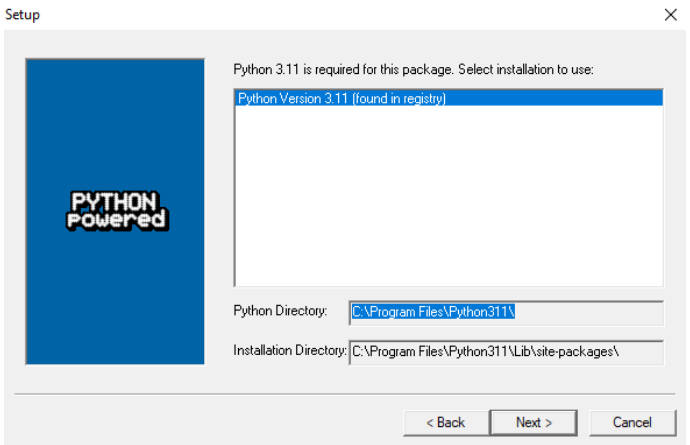


Figure 2-51: Python directory path

Click on **Next** to start the installation.

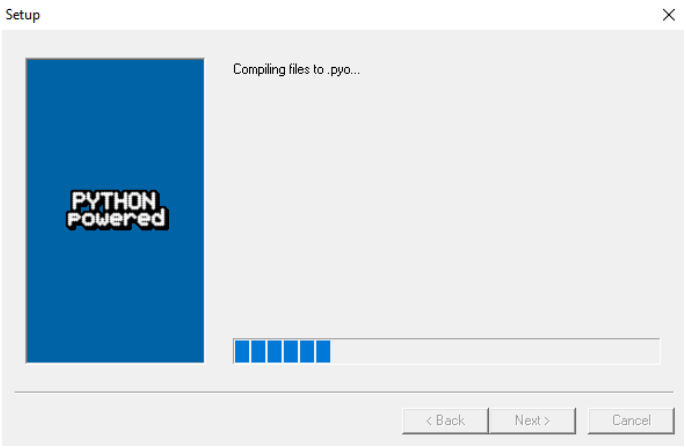


Figure 2-52: Select Next to install of pywin32

Once the installation is finished, click on **Finish**.

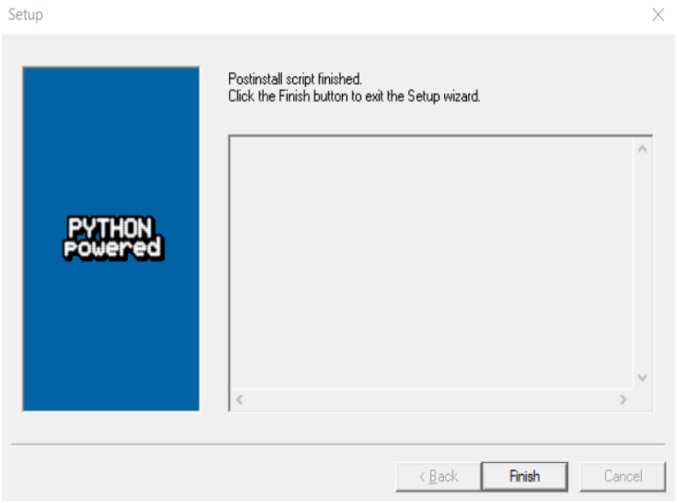


Figure 2-53: Select Finish to complete pywin installation

This completes the Installation of pywin software. NetSim complete Setup wizard appears as shown below. Click on **Finish** to complete the installation process of NetSim.

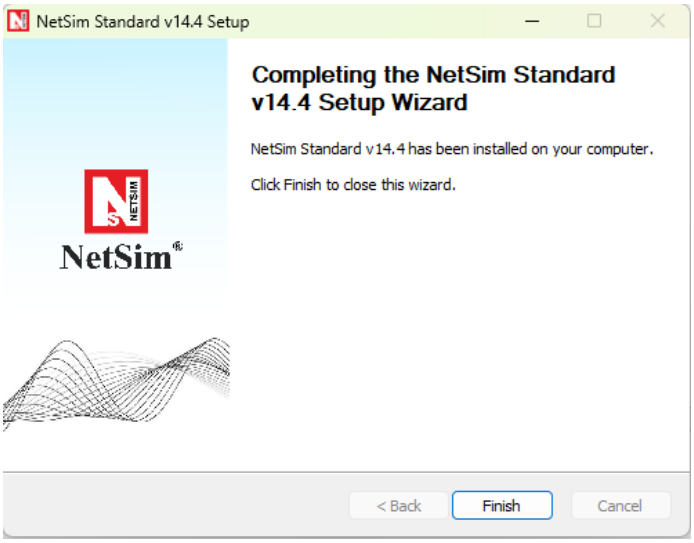


Figure 2-54: NetSim complete Setup wizard

After this, to run NetSim, double click on the NetSim icon present in the desktop or right click and choose **Run as administrator** option. A **NetSim License Server Information** screen appears to start with NetSim.

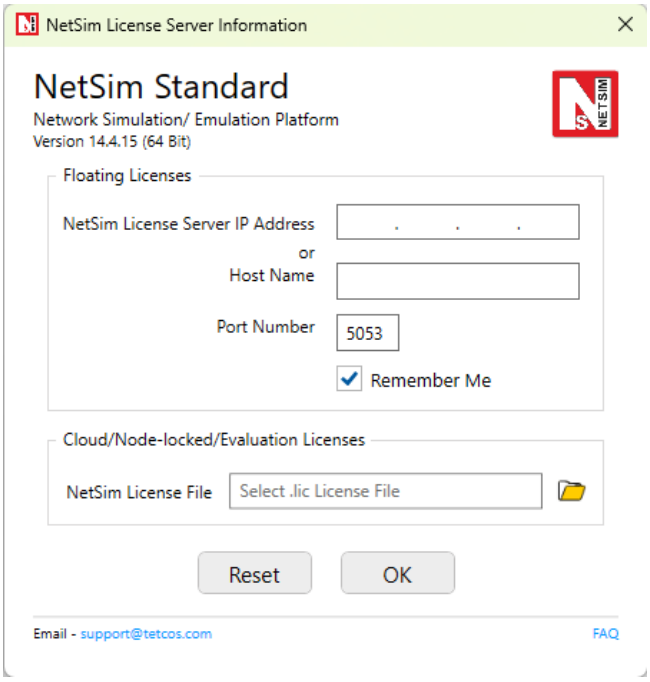


Figure 2-55: Enter NetSim License Server IP Address/Host name/Select NetSim License file

Enter the **NetSim License Server IP Address**, i.e. the system in which the License files are present and the rlm.exe file is running (Refer **Section 2.3** to set up NetSim License Server).

In the case of Cloud/Node-locked/Evaluation license browse the provided LIC file and click on **OK**. Once this is done, NetSim Home screen will appear.

2.2.3 Silent installation

Steps for silent installation in NetSim are as follows.

1. For example, let us take the **NetSim_Standard_14_4_15.exe** setup. Right click on NetSim Standard 64-bit setup → Go to properties and copy the **Location** as shown below.

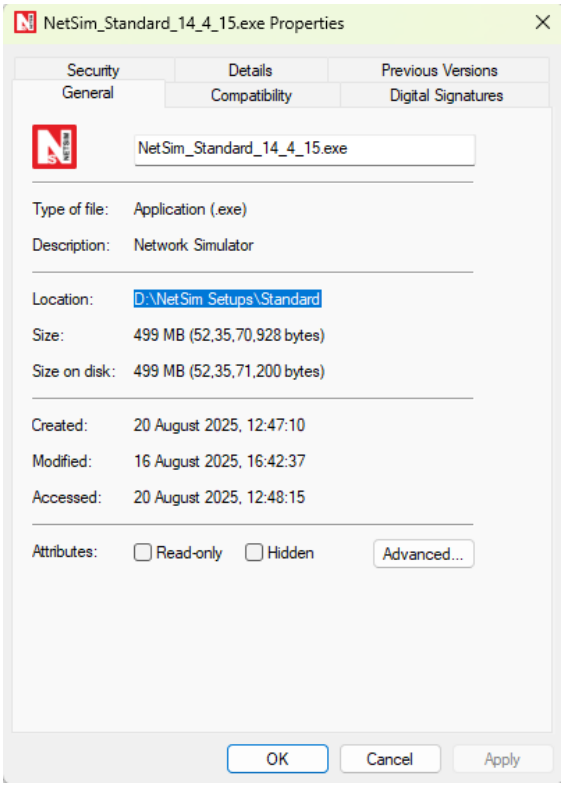


Figure 2-56: NetSim Standard 64-bit setup location

2. Open command prompt and paste the copied location as shown below.

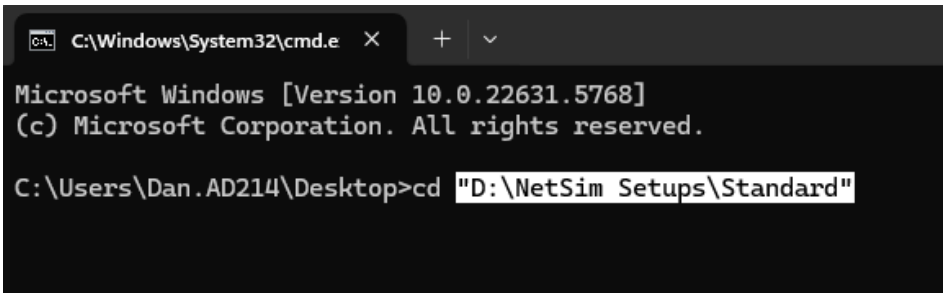


Figure 2-57: Enter setup location in command prompt.

3. Run/Execute Command with the following parameters:

   To install NetSim along with all third-party tools:

"NetSim_Standard_14_4_15.exe"/S /silent=1

- /S: Runs the NetSim installer silently.
- silent=1: Installs NetSim and all third-party tools (Wireshark, Npcap, SUMO, and Python).

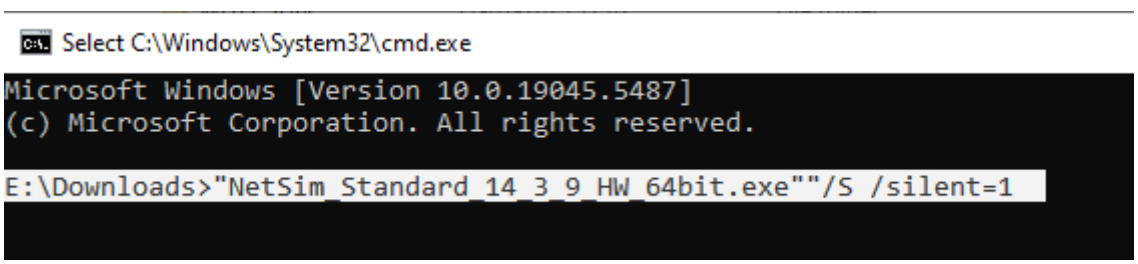


Figure 2-58: Silent installation command in command prompt

To install only NetSim and skip third-party tools:

"NetSim_Standard_14_4_15.exe"/S /silent=0

- silent=0: Installs only **NetSim,** skipping third-party tools.
- Useful for environments where third-party tools are installed separately or not required.

4. Press the **Enter** key. The following User Account Control message window appears. Click on **Yes** to begin silent installation of NetSim.

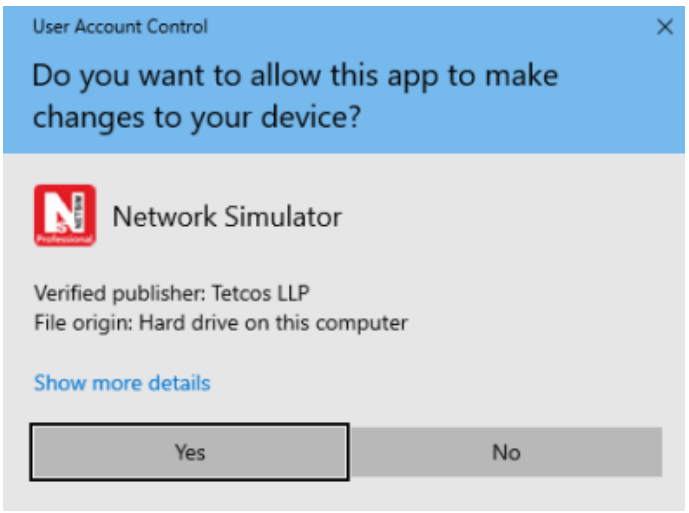


Figure 2-59: User Account Control message window appears and select Yes.

Note: Complete installation of NetSim may take up to 2 or 3 minutes.

2.3 Setting up License Server

2.3.1 Installing NetSim RLM Dongle Driver Software (Dongle Based Licenses)

This section guides you to install the **RLMDongle Driver** software from the CD-ROM.

1. Insert the CD-ROM disc in the CD drive.
2. Double click on My Computer and access the CD Drive.
3. Double click on Driver Software folder.
4. Double click on HASPUserSetup.exe

Each prompt displayed during the process tells you what it is about to do and prompts to either **continue** or exit.

Setup prepares the installation wizard and the driver software installation begins with a Welcome Screen. Click on **Next.**

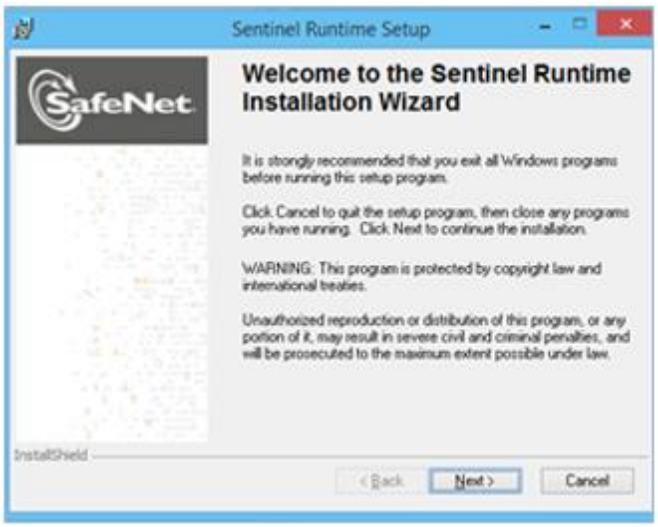


Figure 2-60: Sentinel Runtime Setup window and select Next

Note: Any other program running during the installation of the Dongle will affect the proper installation of the software.

Sentinel Runtime Setup License Agreement appears. Read the license agreement carefully, scroll down to read the complete license agreement. If the requirement of the license agreement is accepted, Click on I accept the license agreement and click on Next else quit the setup by clicking Cancel .

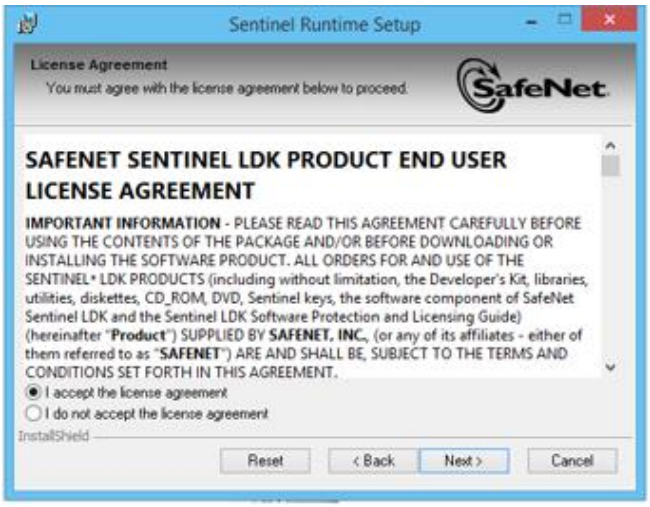


Figure 2-61: Sentinel Runtime Setup License Agreement window appears and select Next

The installation process begins.

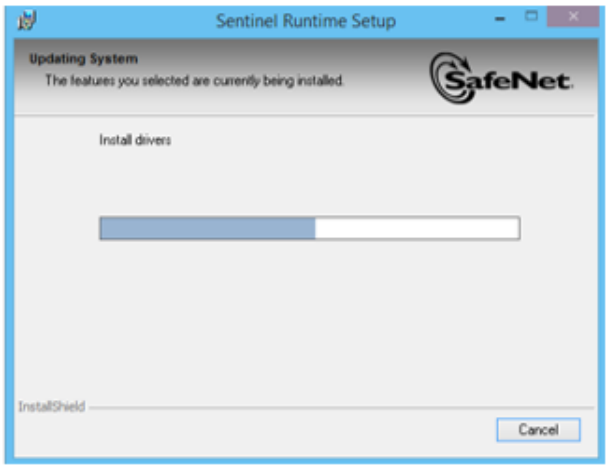


Figure 2-62: Installation process begins

Once the Sentinel Runtime is installed successfully, click on **Finish**.

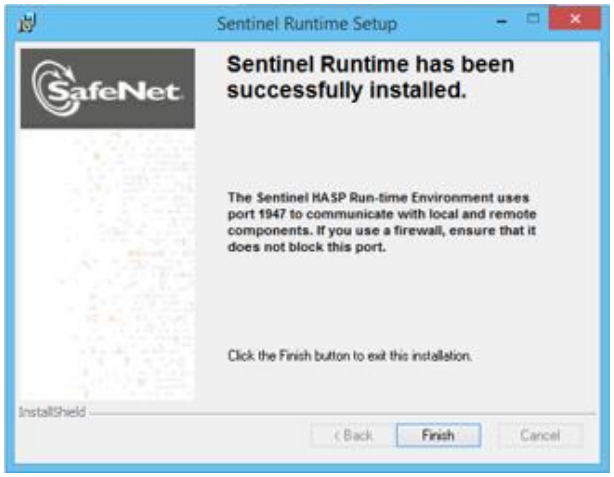


Figure 2-63: Sentinel Runtime is installed successfully and select Finish

Now the RLM driver software is installed successfully. If the driver has been successfully installed, then upon connecting the Dongle in the USB port, a red light will glow (Refer picture below Figure 2-64). If the driver is not properly installed, this light will not glow when the dongle is connected to the USB Port.

Figure 2-64: Connecting the Dongle in the USB Port

2.3.2 Running NetSim License Server

- Copy the **NetSim License Server** folder and paste it onto the **desktop**. Check that it has the license file. If not copy the paste the license file into the **License server folder**
- Double click on **NetSim License Server** folder from Desktop.
- Double click on **rlm.exe**
- For hardware dongle-based users: After the Driver Software installation, connect the **RLM dongle** to the **system USB port**. Double click on My Computer and access the CD Drive. This CD contents will have the NetSim License server folder.

Note: For running NetSim, rlm.exe must be running in the server (license server) system and the server system IP address must be entered correctly. Without running rlm.exe, NetSim won't run.

While running rlm.exe, the screen will appear as shown below Figure 2-65.

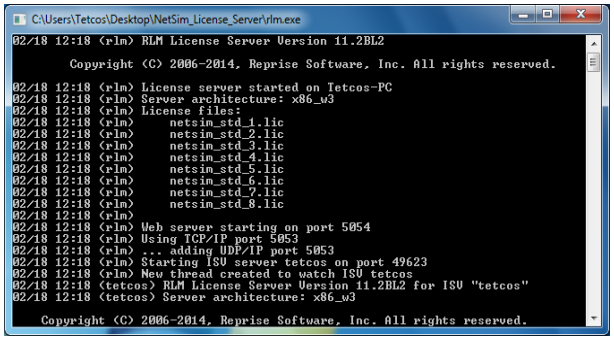


Figure 2-65: When NetSim license server system running, window appears

2.3.3 Running NetSim Software

After running rlm.exe, double click the NetSim icon in the Desktop. The screen given below will be obtained. Enter the Server IP address where the rlm.exe is running and click OK.

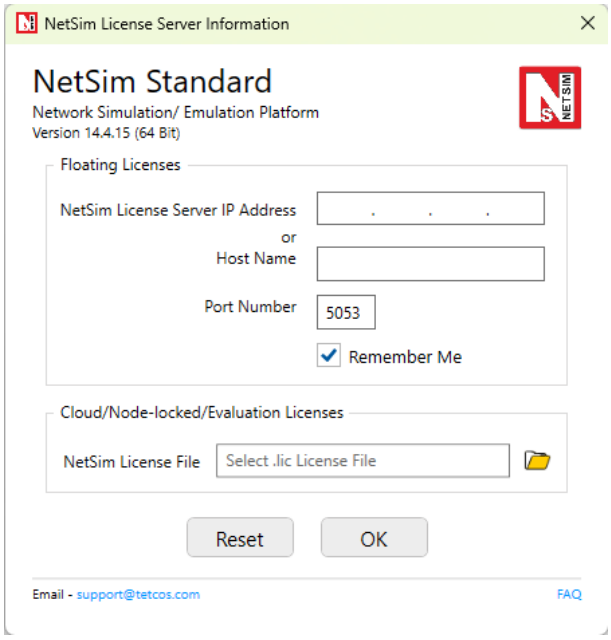


Figure 2-66: Enter NetSim License Server IP Address.

3 NetSim GUI

The graphical user interface (GUI) allows users to interact with the simulator for creating, modifying, and saving, simulation experiments and workspaces. This is much easier to use when compared to command line or text-based simulator interfaces. NetSim GUI comprises of the Home Screen, Design Window, Results Window and Plots Window.

3.1 NetSim Home Screen

The home screen appears after successfully obtaining a license. It's the gateway to all the features and functionality in NetSim and is shown in the image below.

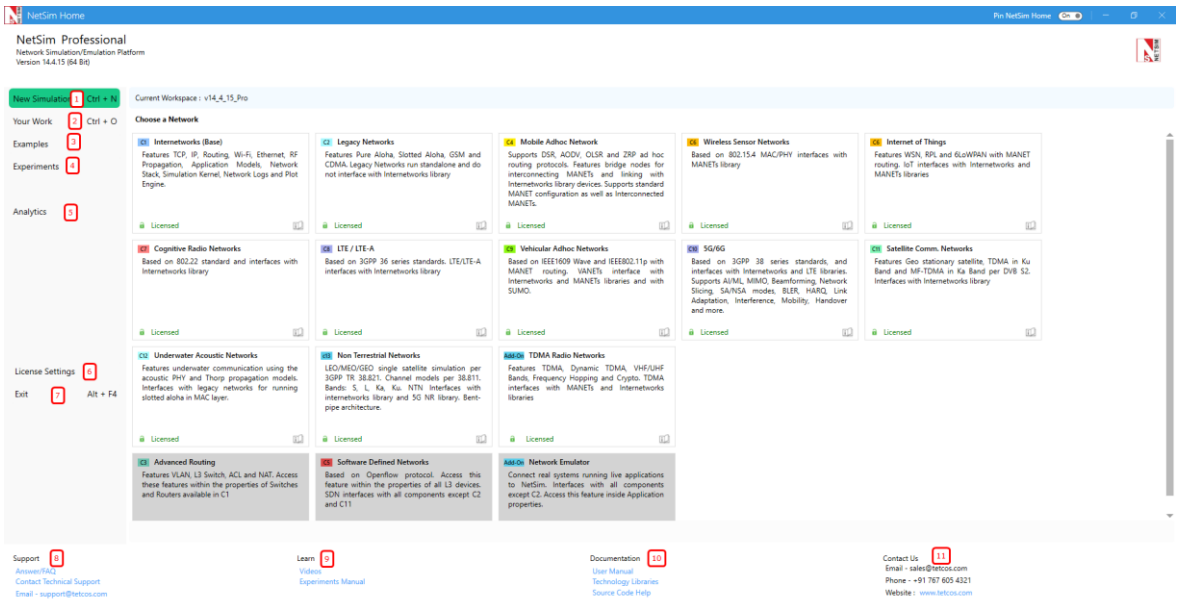


Figure 3-1: NetSim Home screen. Labels 1 through 10 have been added for explanation purposes.

On the NetSim Home Screen, users will find the following items:

1. **New Simulation:** This menu in NetSim allows users to design and simulate various networks. These include Internetworks, Legacy Networks, Mobile Ad hoc networks, Cellular Networks, Wireless Sensor Networks, Internet of Things, Cognitive Radio Networks, LTE/LTE-A Networks, 5G NR, VANETs, Satellite Communication, and Underwater Acoustic Networks, TDMA Radio Networks. Licensed components will have a green open lock, Unlicensed components will show a red closed lock, and those with a grey background at the bottom are inaccessible directly. Users can access these greyed out components through other components as described in the tiles.
2. **Your work:** This menu allows users to access saved experiments within the current workspace. This menu allows users to view, edit, or rerun existing simulations. Additionally, users have the option to export saved files from the current workspace.

3. **Examples:** This menu allows users to conduct simulations categorized by network type. Users can select the network, expand, and click on file names to view the simulation examples. Loading a simulation involves clicking on a tile in the middle panel. Then users can run and analyze results. The book icon on the right side of each network opens corresponding PDF documentation.

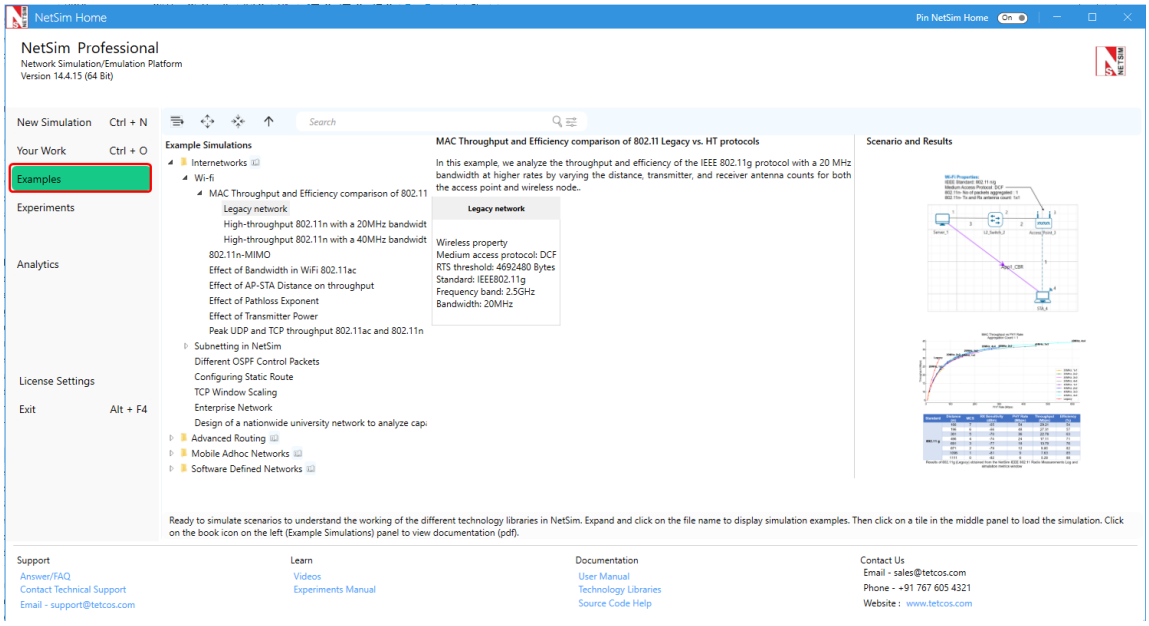


Figure 3-2: Featured Examples

4. **Experiments:** This menu allows users to access the experiments section, covering various experiments across all NetSim technologies. Users can select an experiment by expanding and clicking on the file name, then loading the simulation by clicking on a tile in the middle panel. For all experiments the network and parameter settings are pre-configured. Clicking on the book icon on the right side of each experiment opens the corresponding PDF file that provides a detailed description of the experiment.

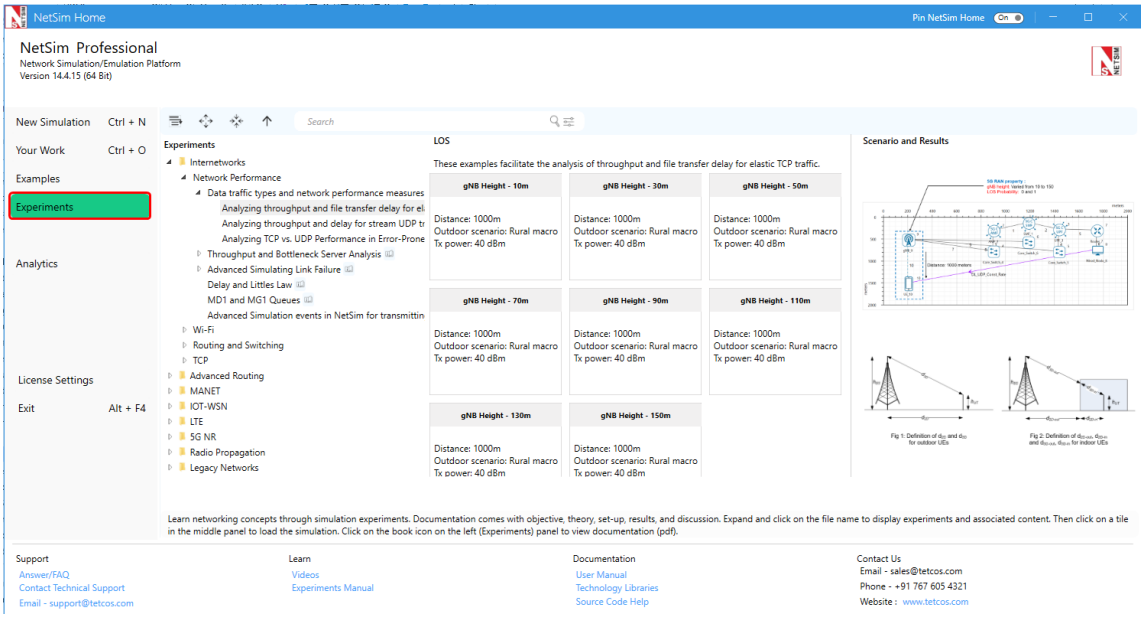


Figure 3-3: Experiments List Window

5. **Analytics:** NetSim supports multi-experiment analytics, enabling users to compare, analyse, and visualize results across different experiments. It presents data through graphical plots for better decision-making and comprehensive analysis.
    - Data can be visualized using bar, scatter, line, or pie charts.
    - The results can be compared for different performance metrics: Application metrics, Link metrics, queue metrics and Battery model metrics
        - Application Metrics: Throughput, delay, and jitter.
        - Link Metrics: Packets transmitted, errored, and collided.
        - Queue Metrics: Number of queued, dequeued, and dropped packets.
        - Battery Model Metrics: Transmit energy, received energy, consumed energy, sleep energy, idle energy, and more.
6. **License Settings:** Clicking on "License Settings" presents users with three sub-menus containing information related to licenses.
    - **License Server Information**: Use this menu to access details about the NetSim License Server, which is the source for checking out licenses by the client.

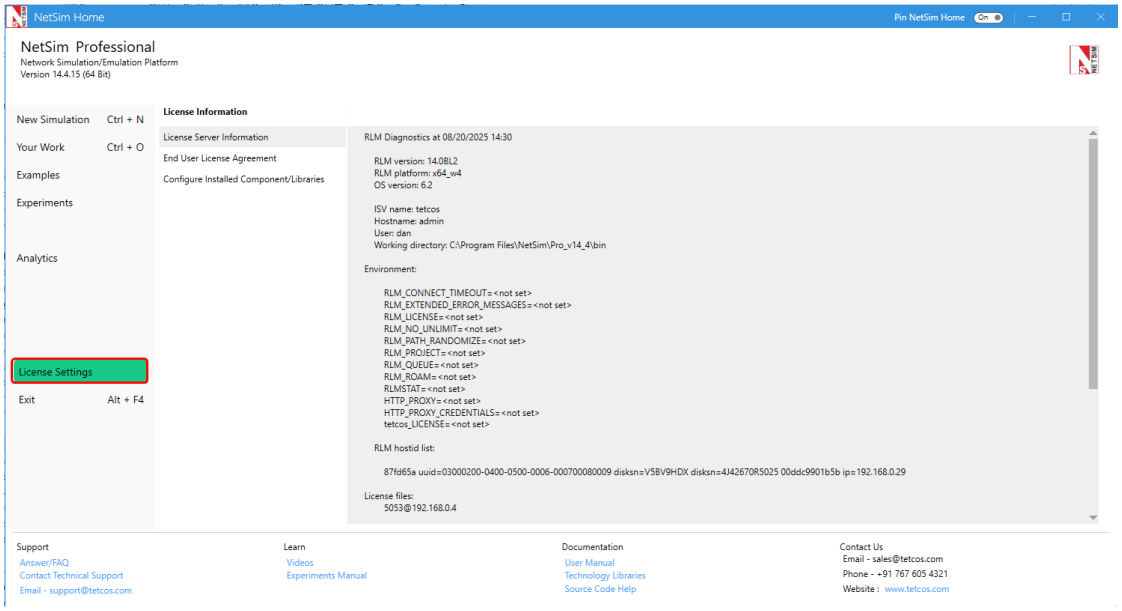


Figure 3-4: NetSim License Server Information window

Upon selecting the "License Server Info" menu item on the NetSim Home screen, user will be presented with details including the platform type running NetSim, the RLM version, Dongle RLM ID, the IP address of the NetSim License Server, and the pathway to the license files on the server hosting NetSim License Server.

- **End User License Agreement:** Use this menu to view the end user license agreement. Upon clicking the "End User License Agreement" menu item on the NetSim Home screen, user will find details such as the Grant of License and Use of the Services, License Restrictions, License Duration, Upgrade and Support Service, and more.
- **Configure Installed Components/Libraries:** Use this menu to enable NetSim users to simulate specific network types based on associated licenses and libraries. You manage network access by selecting libraries for a particular network type, which the NetSim License Server checks out when users initiate NetSim.

On the NetSim Home screen, users can view libraries for components for which they have purchased licenses. Users can select or clear libraries and control access to NetSim, only if they are using floating licenses.

The image below shows the display when a user clicks on the "Configure Installed Components/Libraries" menu item.

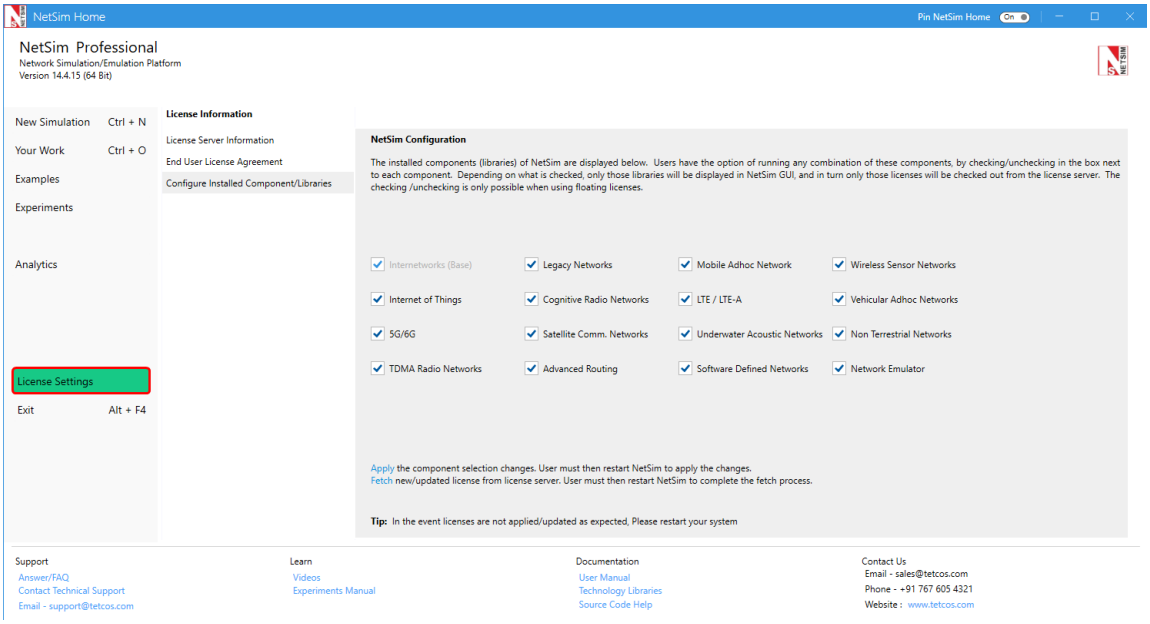


Figure 3-5: The Installed Components (Libraries) of NetSim

The Internetwork component is greyed out and cannot be cleared. This is because Internetworks is a base component essential for the functionality of all other components.

7. **Exit:** Use this option to close NetSim
8. **Support:** Use this section to connect with NetSim support. The **"Contact Technical Support"** link allows users to submit **Support tickets**, while communication via email is possible to support@tetcos.com. The "**Answers/FAQ**" link provides access to our **Knowledge Base**, offering a repository of information categorized into FAQs, Technologies/Protocols, Modelling/UI/Results, and Writing your own code in NetSim. Users can leverage this wealth of information to find answers to their queries.
9. **Learn:** Use this section to learn more about the software which includes the following: **"Videos"** section provides access to NetSim-related videos on TETCOS LLP's YouTube channel. This resource keeps users informed about the latest in NetSim, covers topics related to various network technologies across different NetSim versions, and includes monthly webinars. The **Experiments Manual** features 30+ inbuilt labs, per the curricula of top universities, across a range of networking technologies.
10. **Documentation:** Use this section to open the following NetSim help documents: These include the **User Manual** which explains the various features in NetSim, the **Technology Libraries** which provides users with technical details of various Network Technologies in NetSim through individual PDF files, and **Source code help** which comes along with Standard and Pro Versions of NetSim, enables users to gain a deeper understanding of the underlying code structure for in-depth analysis.
11. **Contact Us:** Use this section to contact us and get in touch and gather more information about our product. Users can write to us via **Email** to **sales@tetcos.com** or contact us

through **Phone** to our official number **+91 76760 54321**. **Website:** Visit our website at www.tetcos.com.

3.1.1 Creating "New" Simulations

The Simulation window appears when users choose the preferred network technology from the New Menu. Click on New Simulation and choose the specific type of network to simulate.

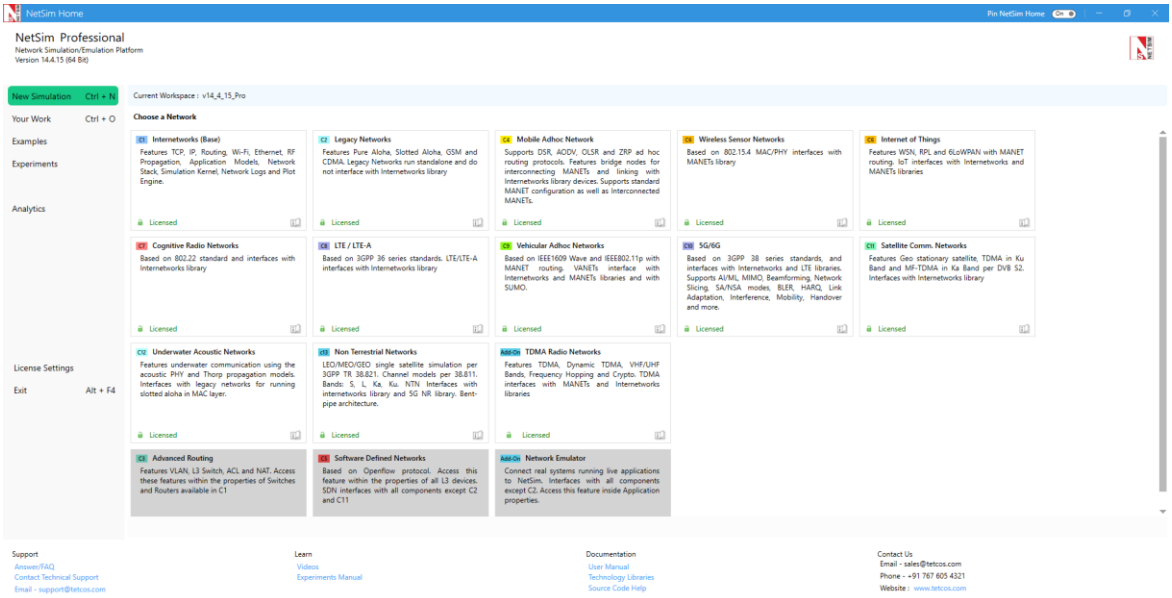


Figure 3-6: NetSim Home Screen

Save

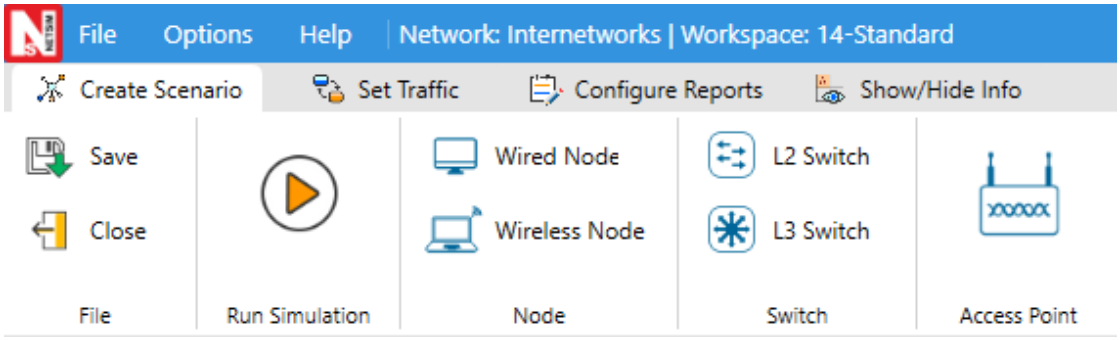


Figure 3-7: To save experiment, Select File >Save.

To save an experiment, users can click on Save, then provide the **Experiment Name** and a Description (optional) and click **Save**. The short cut for the same is Ctrl + S.

Save as

To save a previously saved experiment with a different name after making necessary modifications (without replacing the existing copy), users can utilize the Save As option. Select File > Save As, then input the Experiment Name, Description (Optional) and click Save. The short cut for the same is Shift + Ctrl + S.

3.1.2 Grid: The Working Environment

NetSim allows users to customize the working environment (which we term as the “Grid”) per their preference. Grid values can be adjusted both before and after simulations, taking into consideration the positions of the devices.

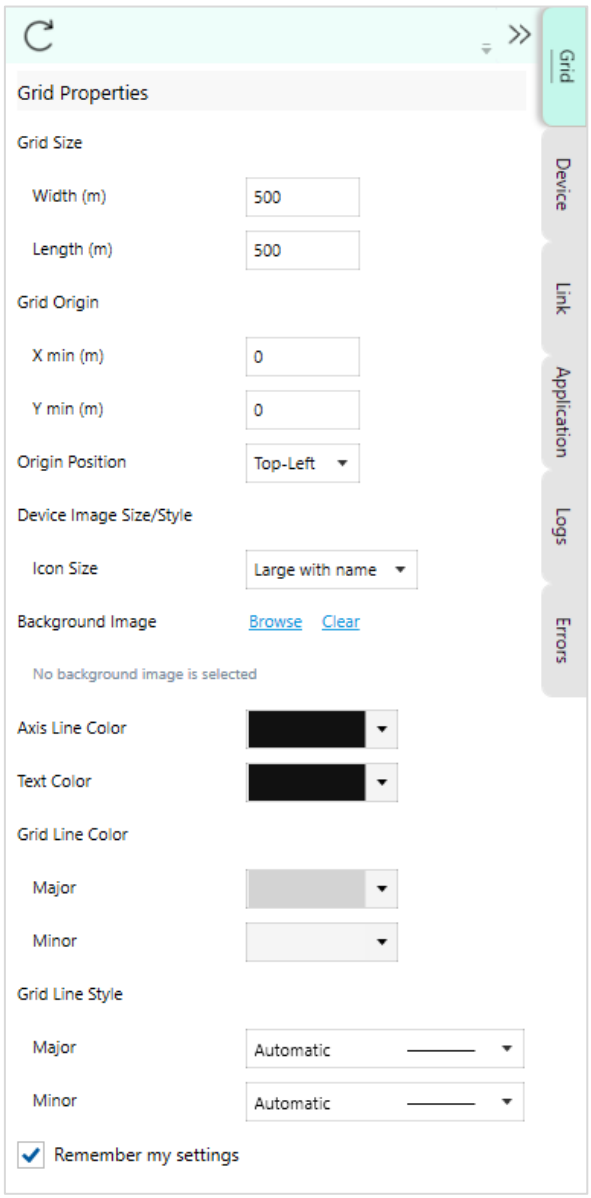


Figure 3-8: Grid Properties Window.

1. **Grid Size:** In NetSim, users are provided with the option to set the grid dimensions both before and after adding devices. The grid width and length need not be equal.

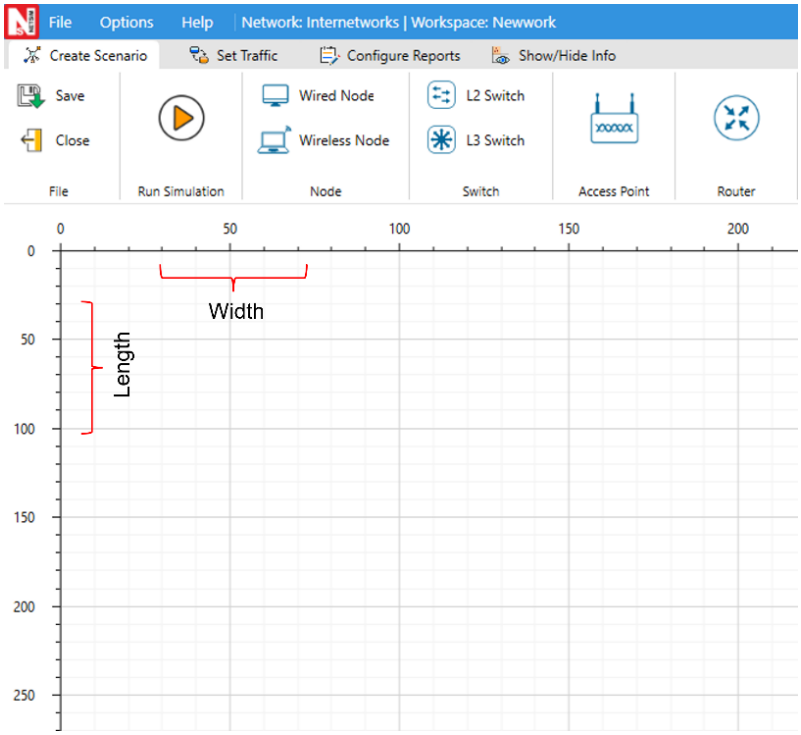


Figure 3-9: NetSim Design window highlighting the grid width and grid length.

2. **Grid Origin:** Grid origin is the starting (X, Y) point for grid lines and serves as the reference for grid coordinate positioning. Note that the origin need not be (0, 0), the (X, Y) origin point can be any positive or negative number.

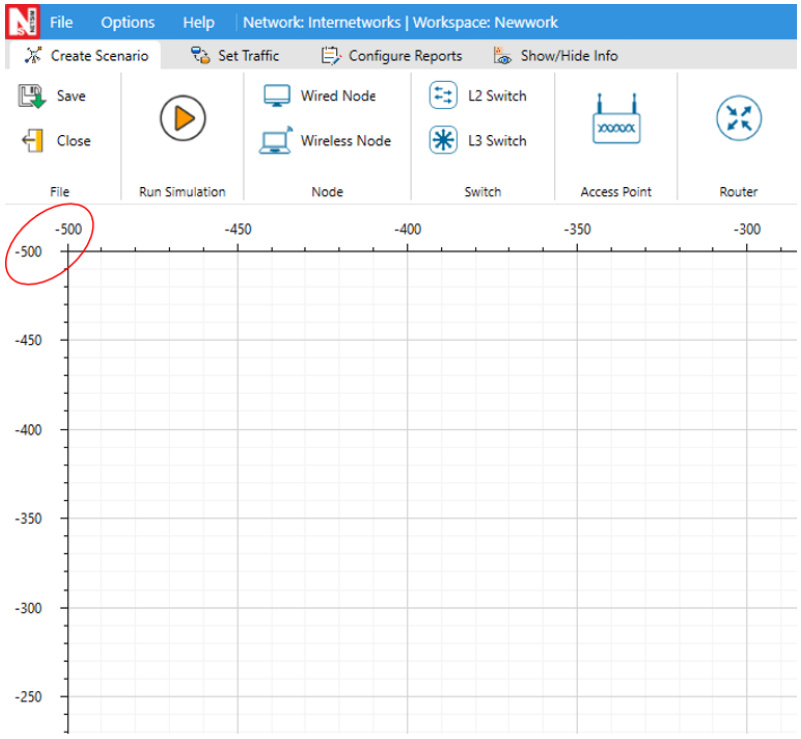


Figure 3-10: Grid with origin at the top right, with the origin set as (-500, -500)

3. **Origin Position:** In NetSim, it's the initial position of the grid origin where users choose between the Top Left or Bottom Left options.

- **Top Left** sets the grid origin at the top-left of the grid, with positive Y going downward and positive X extending to the right.
- **Bottom Left** would set the grid origin at the bottom-left of the grid, with positive Y going upward and positive X extending to the right.

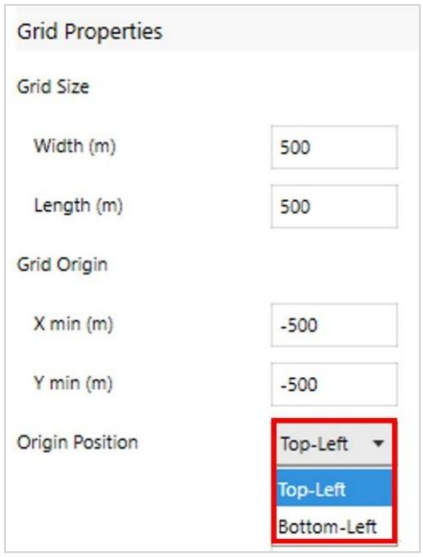


Figure 3-11: Options available in Origin Position.

Additionally, users have the option to modify the **Axis Line color**, representing the lines for the X and Y axes. Moreover, users can also alter the **Text color** of numerical values displayed on the X and Y axes.

Figure 3-12: Color options available for Axis Line Color and Text Color

4. **Grid Line Color:** Grid line color options provide users with the ability to customize the color for both major and minor grid lines.

- **Major Grid Lines:** These are the prominent grid lines that typically correspond to major divisions or values.
- **Minor Grid Lines:** These are the finer grid lines that are often used to represent smaller divisions or values between the major grid lines.

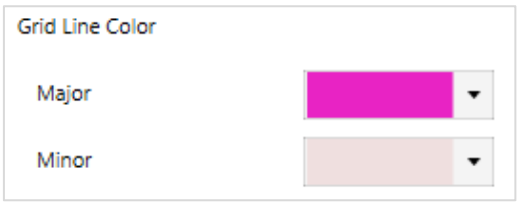


Figure 3-13: Color options available for Grid Line.

5. **Grid Line Style**: Grid line style options comprise major and minor styles, providing users to define the appearance of both prominent and finer grid lines. Within the major and minor grid line style options, users can choose from various line styles like dash, dash-dot, long dash, and more, allowing for customization of the grid lines' appearance.

Figure 3-14: Line style options available for Grid Line

6. **Device Icon Size/Style:** Device icon style lets users personalize the size and style of the icons.

Figure 3-15: Options available for Devie Image Size/Style

7. **Background Image:** When users use the **'Background image'** option in NetSim it substitutes the blank grid environment background with the selected image.

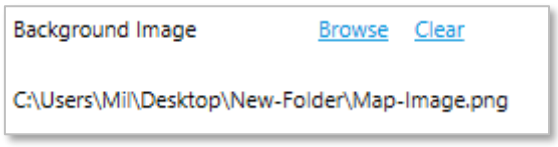


Figure 3-16: Option to change Background Image.

Users have the option to choose any image as the background in the current grid environment by browsing and importing it.

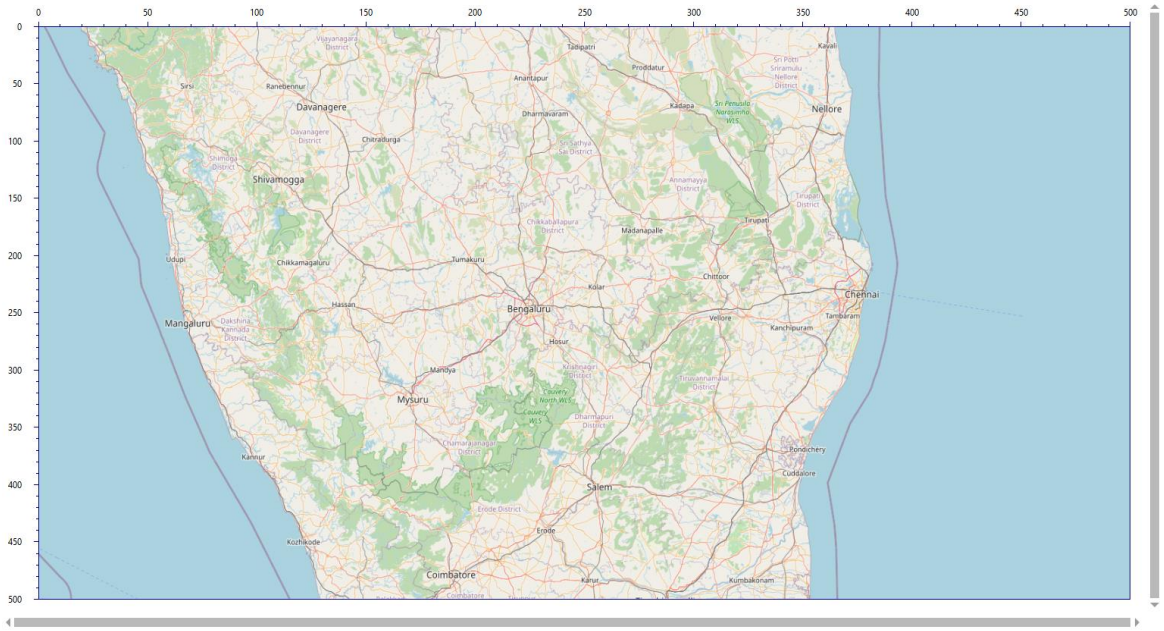


Figure 3-17: A map is added as a background image on the grid.

8. **Remember my settings:** The last option in the Grid Properties window is "Remember my settings." If users check this option, their grid settings will be saved and preserved even after closing and reopening the application.

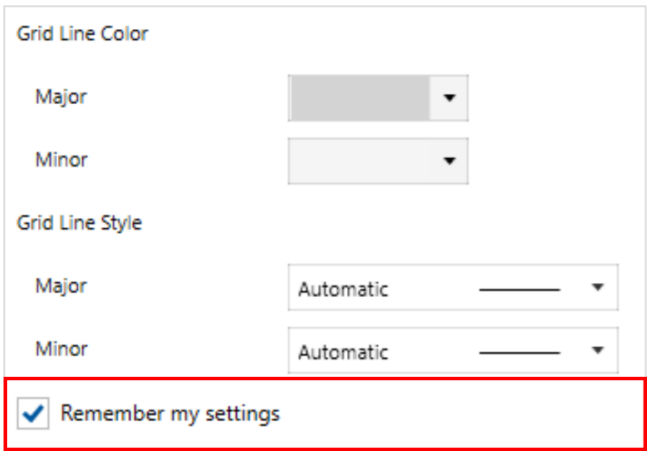


Figure 3-18: Option available to save the modified grid settings.

3.2 Create Scenario

This section will guide the user on creating a basic network scenario and analyzing the results. Let us consider Internetworks. To create a new scenario, go to **New Simulation → Internetworks** as shown in the image below.

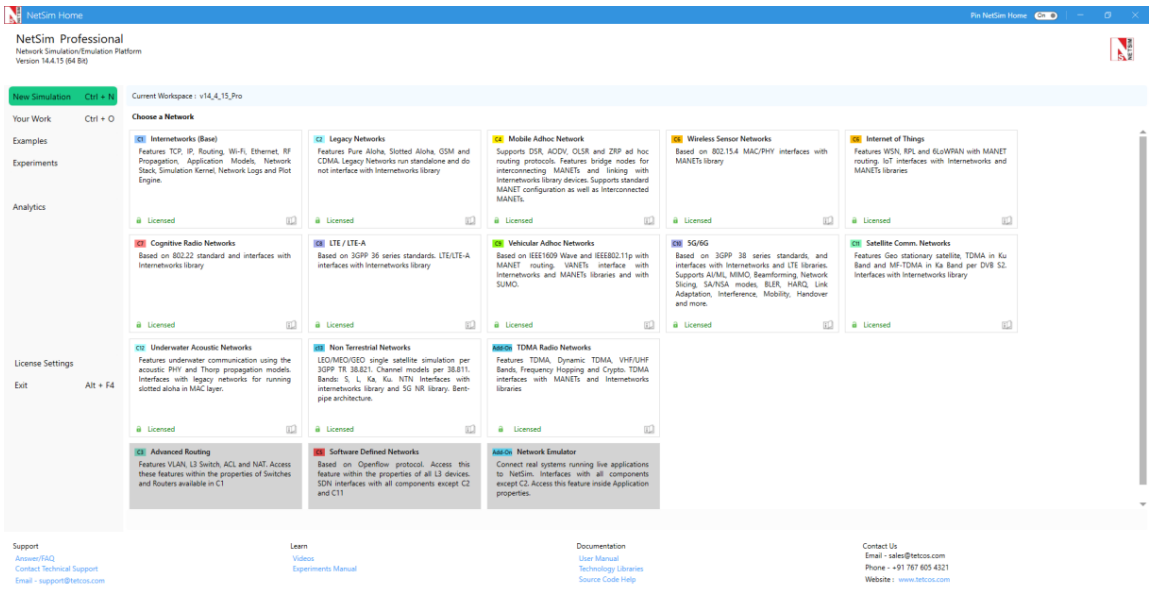


Figure 3-19: NetSim Home Screen

In this example, we design a network with two subnets. Let's assume subnet 1 comprises two wired nodes connected via a switch, while the other subnet consists of one wired node. Both subnets are connected using a router. The flow of traffic in the network goes from a wired node in subnet 1 to the wired node in subnet 2.

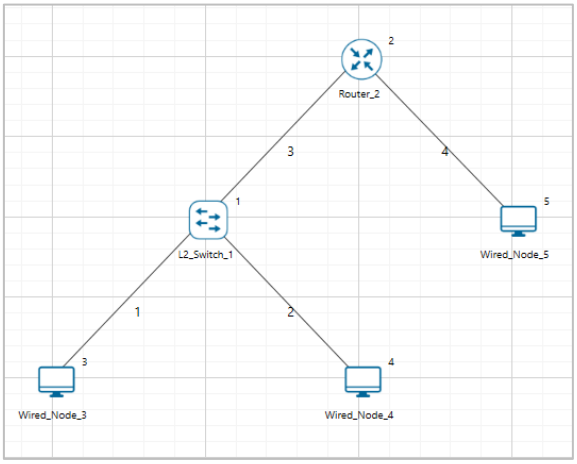


Figure 3-20: Network Topology in this experiment

Perform the following steps to create this network design.

Step 1: Drop the devices. Click on Create Scenario Tab in Top left ribbon and select → Wired Node.

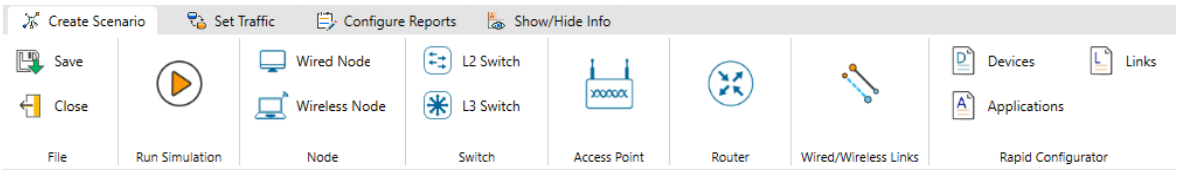


Figure 3-21: Internetworks Device Palette in GUI

Click on the environment (the grid) where users want the Wired Node to be placed. In this way, place two more wired nodes. Similarly, to position a Switch and a Router, click on the respective device and click on the environment at the desired location.

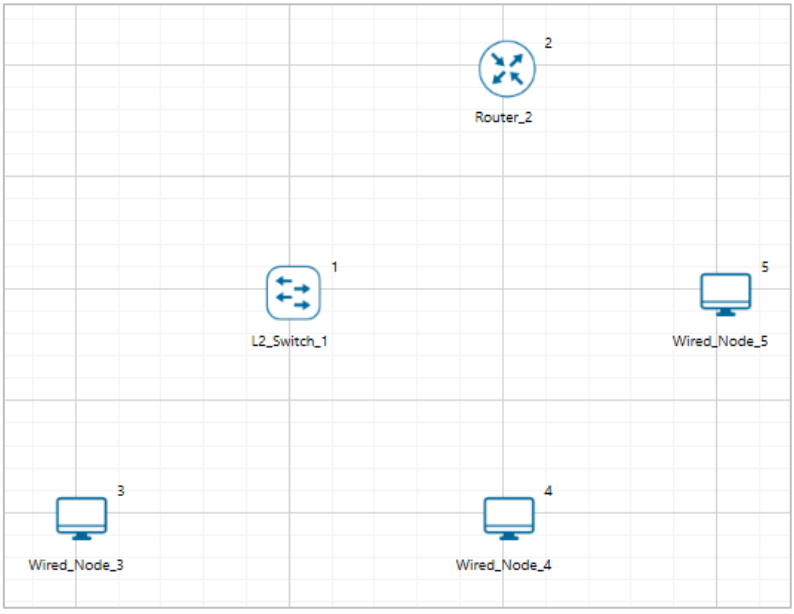


Figure 3-22: Dropped Devices on GUI

Note that in NetSim, the (x, y) position of any device on the grid represents the position of the top-left corner of the icon and not the center of the icon.

Step 2: Connecting devices involves selecting the link, left clicking on one device, releasing the mouse , then clicking on the second device and releasing the mouse . If you right-click anywhere in the environment, the wired links may disappear. Be cautious not to displace the device in the environment by clicking and dragging without releasing the mouse pointer.

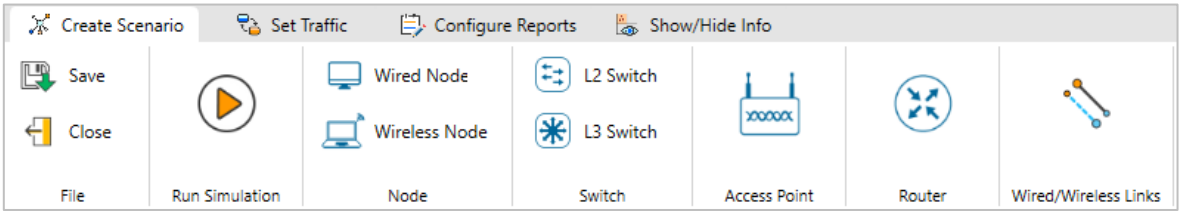


Figure 3-23: To Connect devices select wired/wireless links.

For instance, choose the link, then click on the switch followed by the router to connect them. In this way, proceed to link all devices.

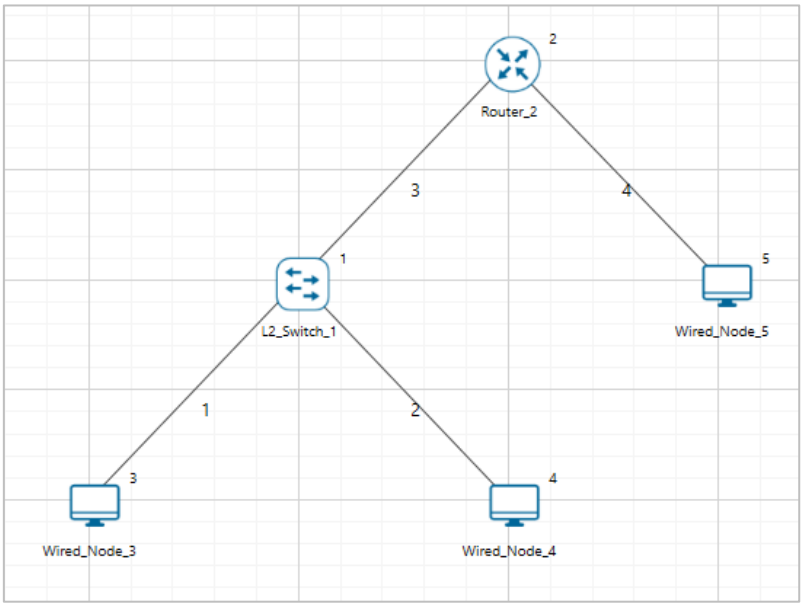


Figure 3-24: Network Topology

3.2.1 Configuring devices and links

Step 1: To configure any device, click on the device which will open the right property panel as shown below.

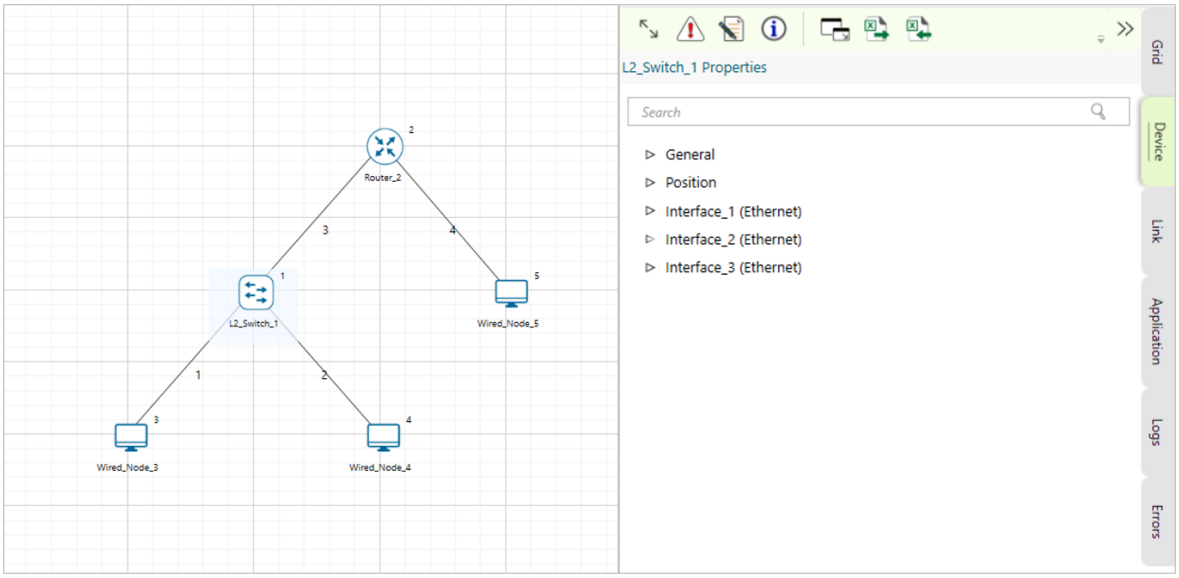


Figure 3-25: Click on the device to select properties.

Users can modify the default properties of any device as per their requirements. Then click “Enter”.

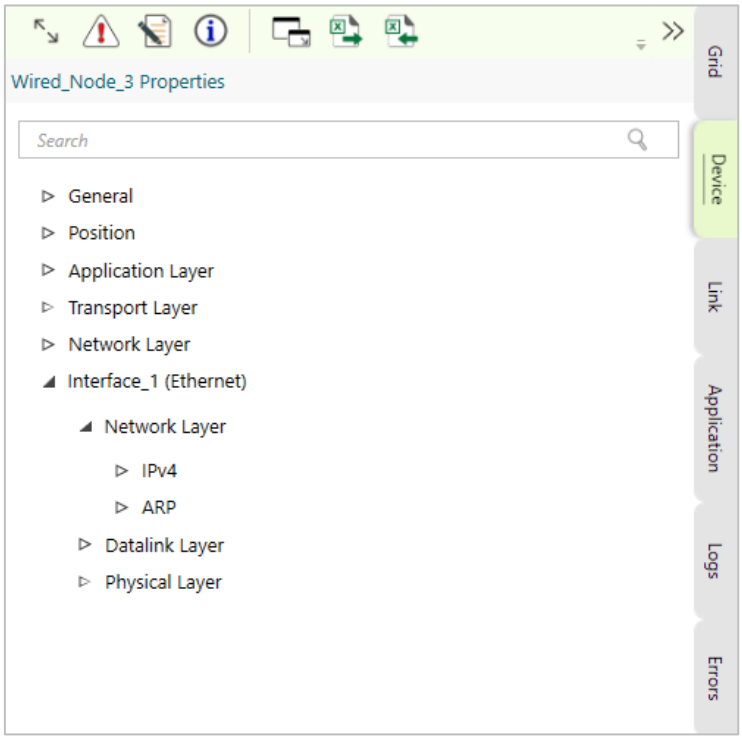


Figure 3-26: Network layer Properties window for wired node

Step 2: To configure the links, click on any Link ID and select Properties as shown below.

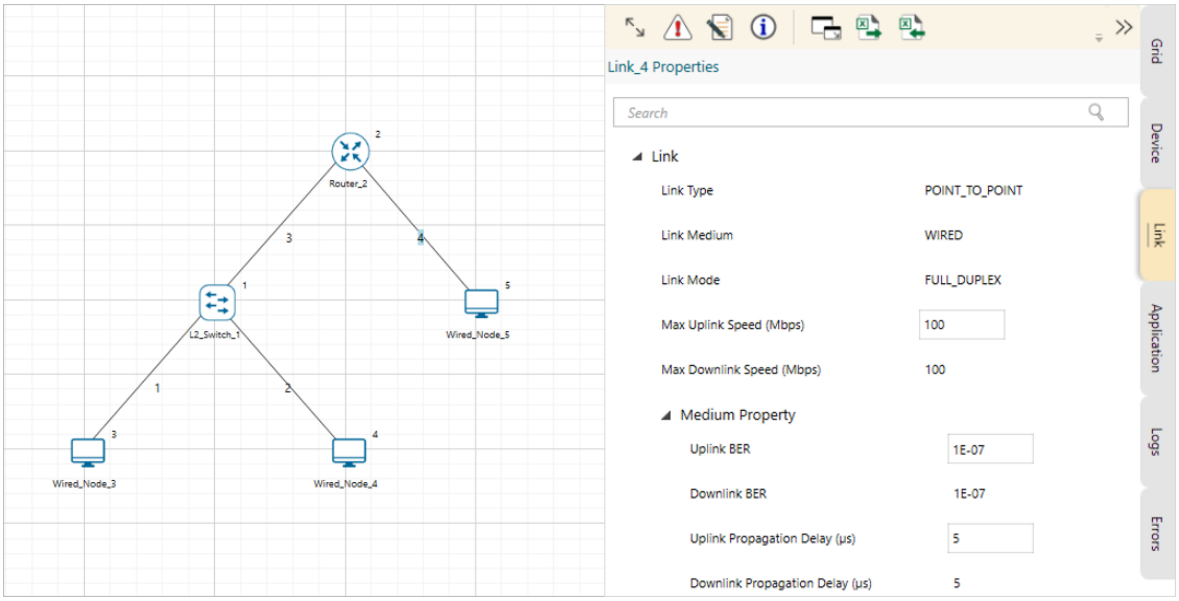


Figure 3-27: Wired Link properties window for links

In NetSim, the properties of devices are categorized as follows:

1. Local: Changes made in one device do not affect any other device in the network scenario.
2. Global: Changes made in one device affect all the other devices of the same type in the network scenario.
3. Link: Changes made in one device affect other devices of the same type connected to the same Wireless/Ad-hoc link.

4. Fixed: These parameters are auto populated by NetSim and cannot be edited by users.

3.2.2 Undo/Redo

In NetSim, users can find the Undo and Redo features in the status bar. With these s, users can revert their actions, such as dropping devices and creating connections, and they can also redo them if necessary.

Consider a scenario with two wired nodes and one router.

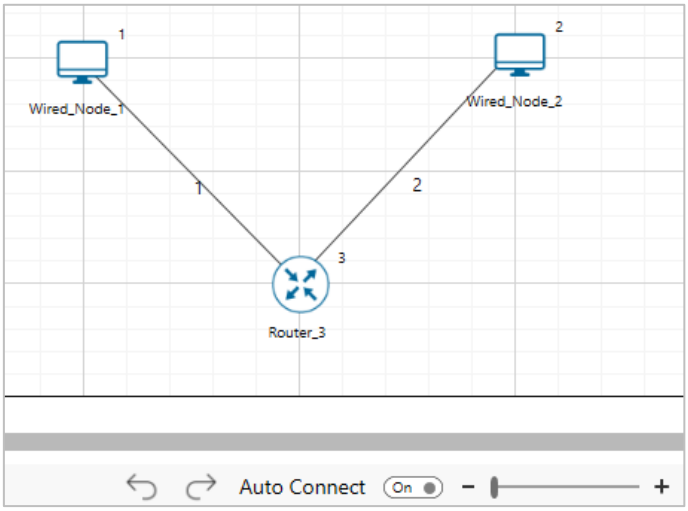


Figure 3-28: Basic scenario

Users can now utilize the Undo feature to reverse any action they've performed. Clicking Undo once removes the most recently created link or node.

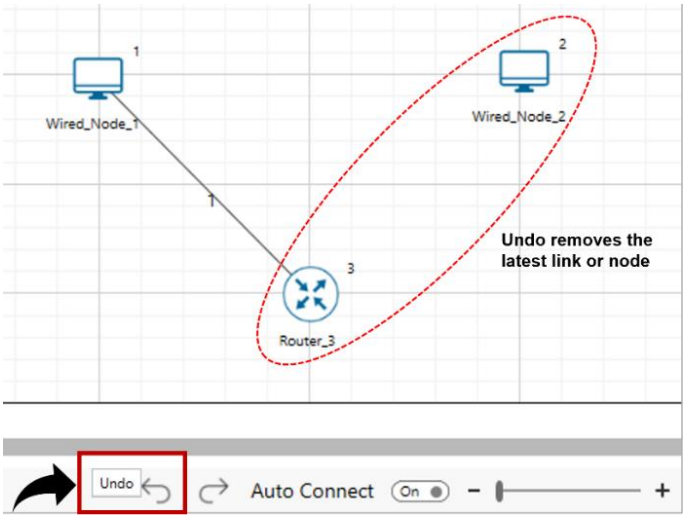


Figure 3-29: Undo option

The Redo feature is also available. When clicked, it restores the previously removed link or node, enabling users to bring back what they've undone with a simple action.

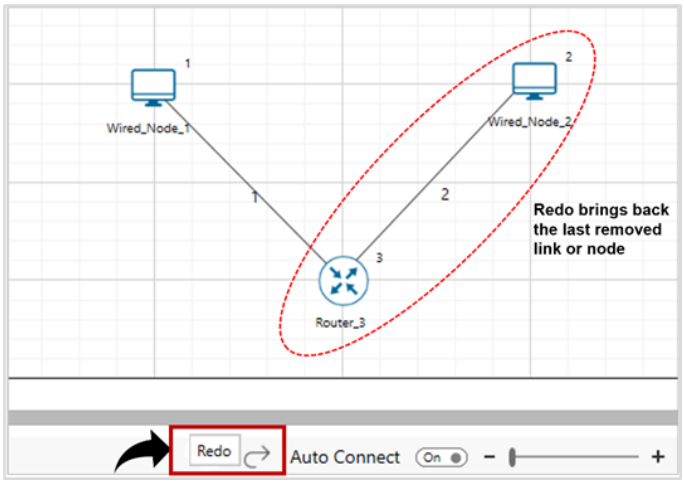


Figure 3-30: Redo Option

3.2.3 Auto Connect

The Auto Connect feature in the status bar can be turned on or off. This functionality is particularly useful for users who want to establish connections among wireless devices automatically.

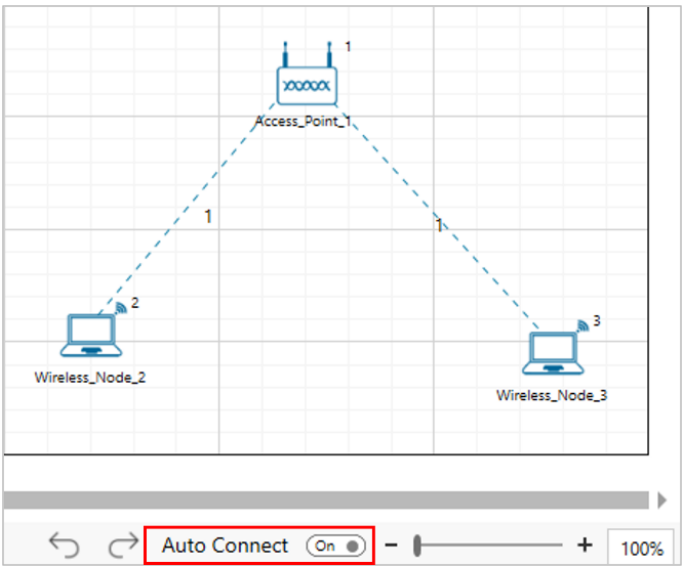


Figure 3-31: Auto Connect option in GUI

3.3 Set Traffic

Users must model data traffic (or application, to be more precise) using the Set Traffic tab. Select the type of application present in the ribbon. Refer the below image.

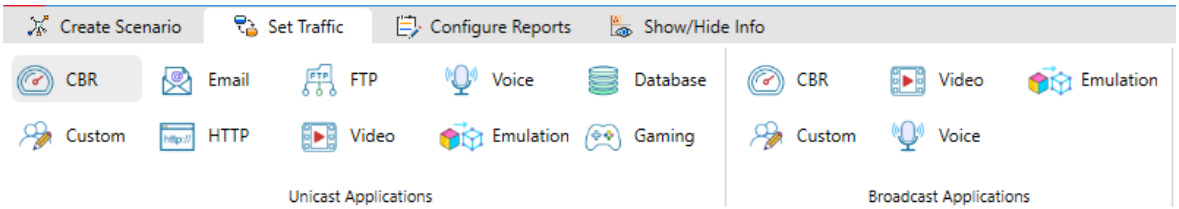


Figure 3-32: Set traffic by configuring applications shown on the ribbon.

Select, then left click on one device, release the mouse , then click on the second device and release the mouse . In the screenshot shown below, set the Application type to CBR, Source ID to 2, and Destination ID to 3. Click on OK.

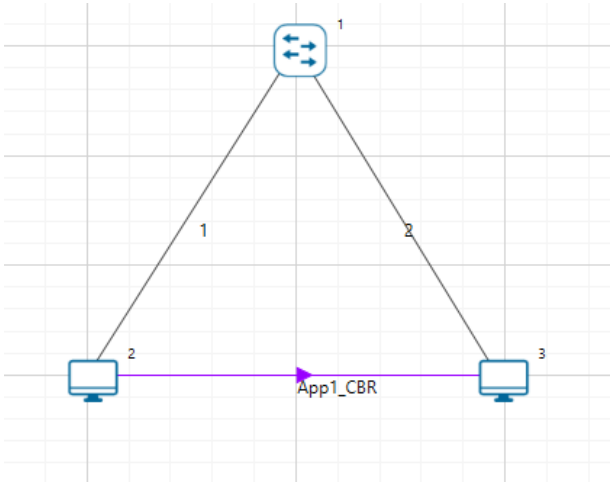


Figure 3-33: Application Configuration window

3.4 Configure Reports

3.4.1 Enable logs (E.g.: Radio measurements, Radio resource allocation etc.)

Users can enable Network log files such as the Radio Measurement log, Radio Resource Allocation log by Clicking on Configure reports tab > Plots icon in the top ribbon as shown below.

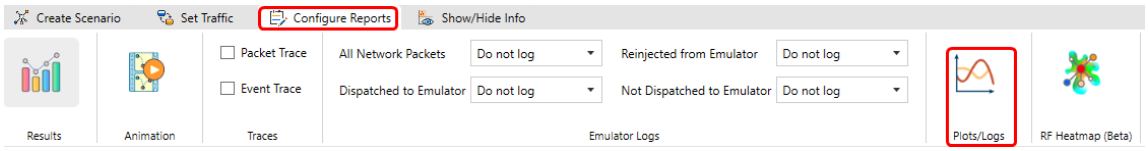


Figure 3-34: Plots option in GUI.

The window that appears will display a list of Network logs that are applicable for the current network. Users can enable checkboxes to generate the respective logs.

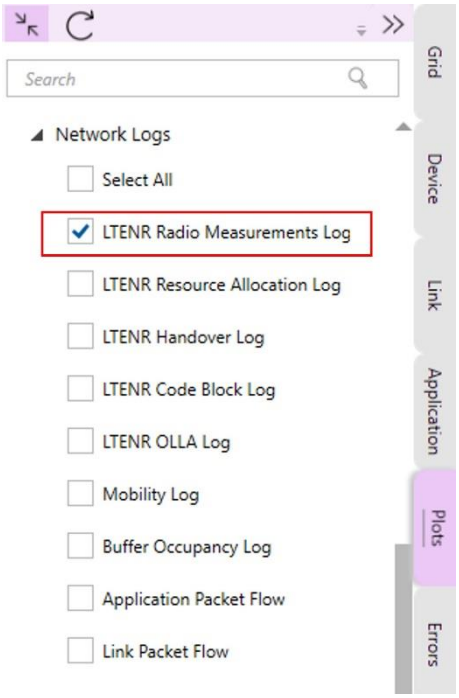


Figure 3-35: Enabling the LTENR Radio Measurements log.

3.4.2 View results

The View Results option enables users to view and review the outcomes of their previous runs.

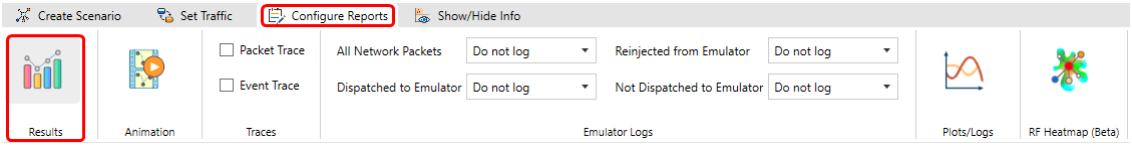


Figure 3-36: View Results in GUI

3.4.3 Packet Trace and Event Trace

Packet and Event Trace files are valuable for in-depth simulation analysis. By default, these are not enabled to prevent slowing down the simulation. To activate Packet Trace & Event Trace, users can click on the Configure Reports tab in the top ribbon and check both packet trace and event trace.

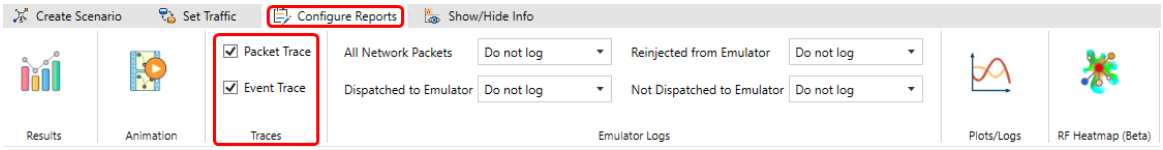


Figure 3-37: Packet Trace and Event trace options present in Configure reports tab on top ribbon.

3.4.4 Packet Animation

NetSim provides the feature to play and record animations to the user. Packet animation allows users to visualize and analyze traffic flow through the network in detail. Users can choose to enable or disable this feature before running a simulation.

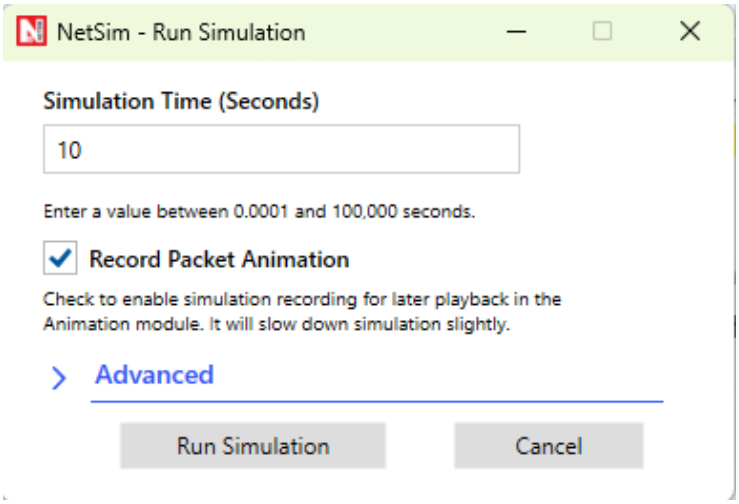


Figure 3-38: Run Simulation window

The packet animation would then be recorded, and the user can view the animation from the NetSim Packet Animation window as shown below

Figure 3-39: Packet Animation window

While viewing packet animation, user can see the flow of packets as well as the type of packet. Blue color packet denotes control packet, green color is used for data packet and red color is error/collided packet.

3.4.5 Heatmap

3.4.5.1 Introduction

The NetSim Heatmap Generator is a tool used to visualize wireless network performance metrics such as Received Power, SINR, and SNR over a specified area.

This tool generates heatmaps for Received Power, SINR, and SNR. For each network type, specific devices are defined as transmitters and receivers. The table below shows the transmitter and receiver device types, and the channels supported, for different network types.

Network	Transmitter	Receiver	Channels
Internetworks	Access Point	Wireless Node	DL
Internetworks	Wireless Node	Access Point	UL
Mobile Ad Hoc Networks	Wireless Node (Omni/Sector)	Wireless Node (Omni)	DL
TDMA Radio Networks	Wireless Node	Wireless Node	DL
5G NR	gNB (Omni/Sector)	UE	DL
5G NR	UE	gNB (Omni/Sector)	UL
LTE	eNB (Omni/Sector)	UE	DL
LTE	UE	eNB (Omni/Sector)	UL

Table 3-1: Transmitter and Receiver nodes based on signal direction in different networks.

The set of assumptions made when plotting heatmaps are listed below.

- Heatmap calculations are performed at time, $t = 0$, without considering mobility or fading effects.
- In Heatmap calculations:
    - The downlink heatmap is generated by placing a virtual receiver at each grid point and assuming that the transmitter sends packets to all the points. For example, in 5G, the received power at virtual UEs located at various points in the grid, is calculated from the gNB(s).
    - The Uplink heatmaps use virtual transmitters at grid points to send packets to fixed receivers. For example, in 5G, the received power at the gNB is calculated assuming virtual UEs at various points on the grid.
- The heatmap plotted with signal direction as uplink takes into account only the transmit power of the first receiver device dropped in the scenario.
- 5G NR and LTE heatmap calculates the signal quality metrics mentioned for the first carrier (in case of carrier aggregation) and the first layer (in case of MIMO).
- For MANETs and internetwork scenarios, the heatmap shows the maximum possible interference, and might differ from Radio measurement logs. This is because the heatmap assumes all transmitters are always transmitting and therefore causing interference. On the other hand, the radio measurement log computes interference only when other transmitters are transmitting.
- For the single transmitter heatmap, the MCS and CQI tables are chosen based on that device's properties. However, for the best transmitter heatmap, the CQI table is chosen based on the properties of the first gNB/eNB dropped into the network.

3.4.5.2 Data Points Selection

Data point selection varies across different networks. The selection method for each network type is given below

- For Internetworks, MANETs, and 5G, which use tables for color selection, small discrepancies may occur because data intervals are rounded off during the writing process.
- Lower cut-offs represent the data points where NetSim considers no successful transmission. Grid points below this threshold are shown as transparent on the heatmap.
- The data range points for 5G color schemes are selected based on AMC tables following 3GPP standards.
- The data range points for MANETs and Internetworks heatmaps (Rx Power and SNR) are based on IEEE 802.11 standards. For SINR, we set a lower transmission limit and divide the SINR range into 10 equal divisions.
- For TDMA and DTDMA heatmaps, the receiver sensitivity is used as the lower transmission limit, and the heatmap points are divided into 10 equal divisions.

3.4.5.3 Heatmap Options

To generate a heatmap, go to the “Configure Reports” section and click on the RF Heatmap (Beta) . This will bring up the heatmap generation window as shown in Figure 3-41 .

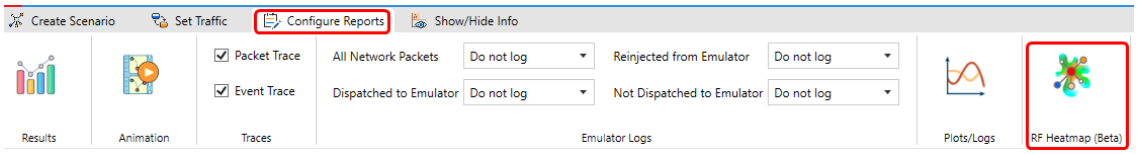


Figure 3-40: Heatmap option present under configure reports tab in GUI.

- Users can plot heatmaps for Receive Power, SNR, and SINR, with options for both uplink and downlink signal directions.
- However, there are some limitations:
    - For MANETs, TDMA, and DTDMA, only downlink signal direction is available.
    - SINR is not supported for TDMA and DTDMA due to network property constraints.
- After plotting a heatmap, you can re-plot it by changing the transmitter position or adjusting device properties.
- NetSim provides a color interpolation option for heatmaps, allowing smooth transitions between colors.
- By disabling the checkbox, "Display Wired/Other Devices", NetSim displays only the transmitting and receiving devices and removes all the other devices from Heatmap.

- NetSim also offers an option to save the plotted heatmap. To do this, click the “Export image” located at the bottom of the Create Heatmap window after plotting. Users can also click the “Open Data Source” located at the bottom of the Create Heatmap window, to obtain the data in CSV format.

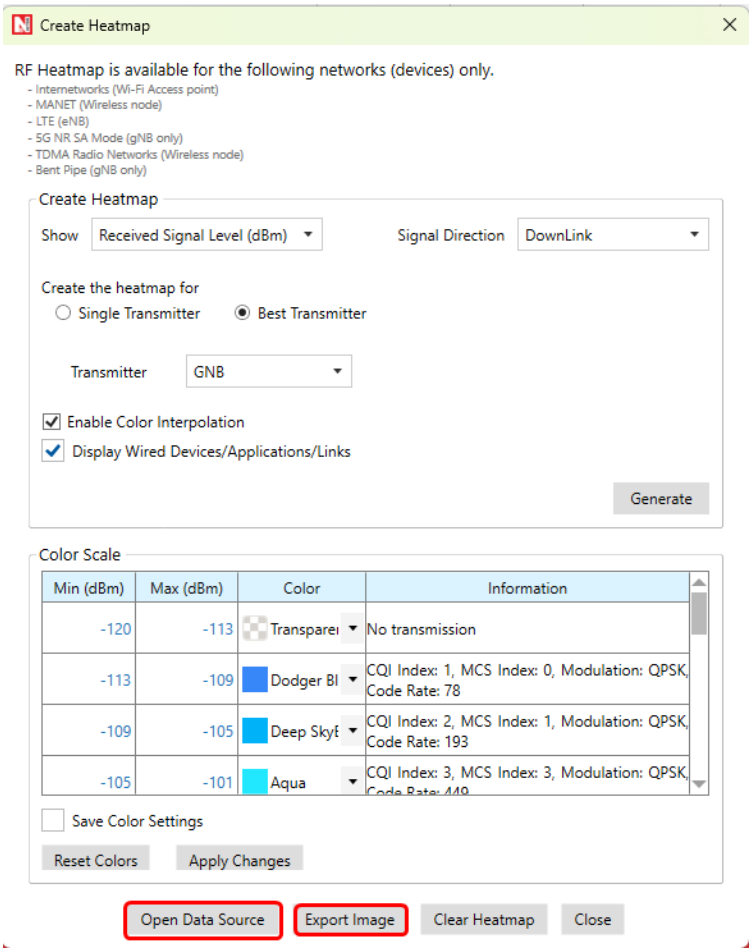


Figure 3-41: Heatmap creation window for RF signal levels in a 5G network, showing received power.

- NetSim allows users to check heatmap data at any point by clicking on the grid.
- NetSim provides a default color scheme for each type of heatmap. Users can customize the colors and re-plot the heatmap as needed.

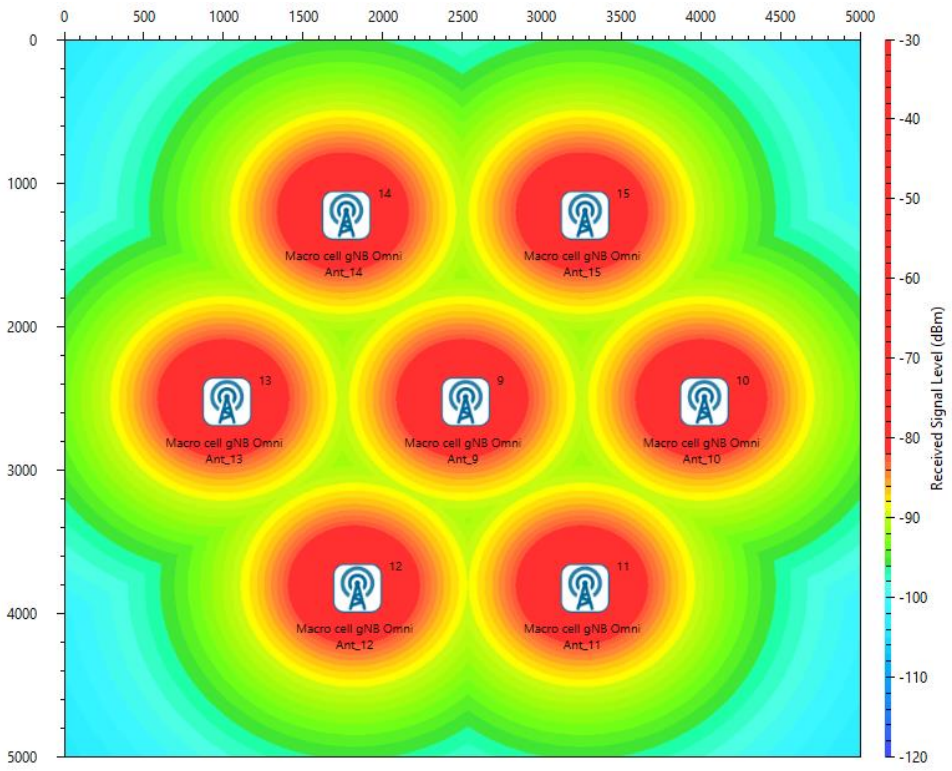


Figure 3-42: Received power Heatmap for 7 cell scenario with Omni-directional antenna.

3.5 Run Simulation

To simulate the created network scenario, users can click on Run Simulation in the ribbon at the top ,the shortcut for same is Ctrl+R.

Figure 3-43: Run Simulation icon in the Ribbon.

Set the Simulation Time to 10 seconds. Click on OK.

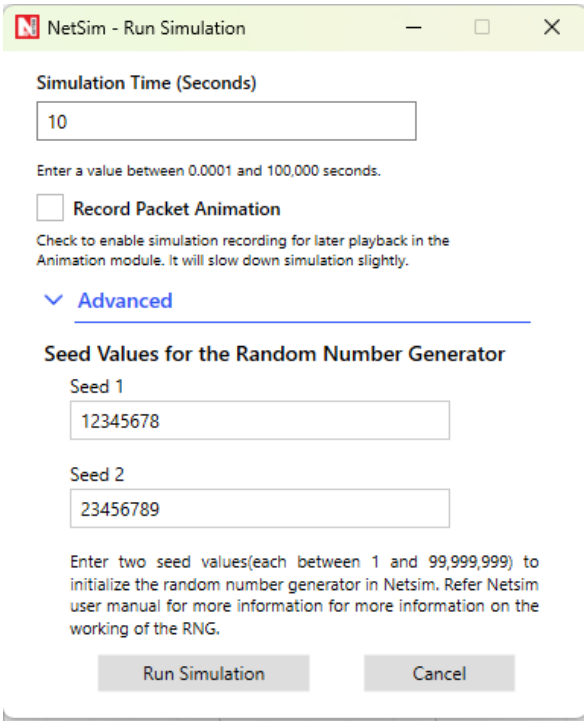


Figure 3-44: Run Simulation window.

3.6 Show/Hide Info

In NetSim, users can choose to display or hide information such as the IP address of devices, device names, distances between devices, link speed, and more. To do this, click on the Show/Hide Info tab in the top ribbon, as shown below

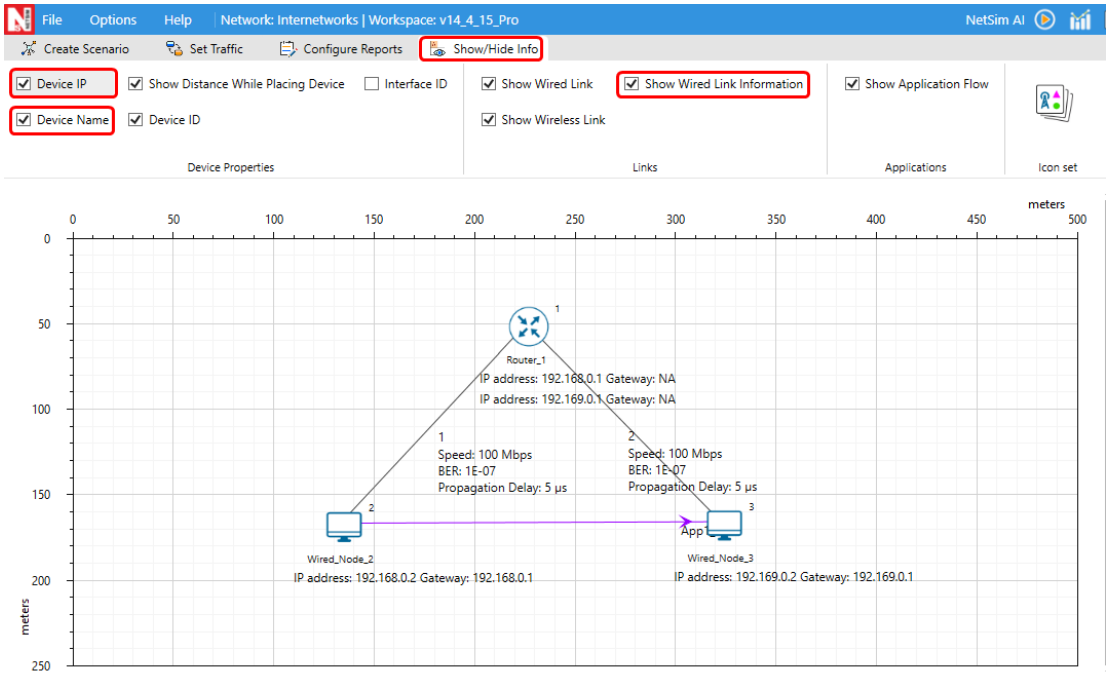


Figure 3-45: Show or Hide display information such as IP Address of the devices, link speed etc.

In NetSim, the device ID acts as a “device identifier,” whereas the IP Address serves as an “Interface identifier”.

Similarly, users can enable or disable the display of wired and wireless links, as well as application flows. This is useful for better visibility, especially in large scenarios.

Figure 3-46: Disabling the wireless link and application flow from Show/ Hide Info tab

NetSim automatically displays the distance between nodes, such as Access Points and Wireless Nodes, eNB/gNB and UE, Base Stations and CR CPE, as well as Base Stations and Mobile Stations. This option is enabled by default

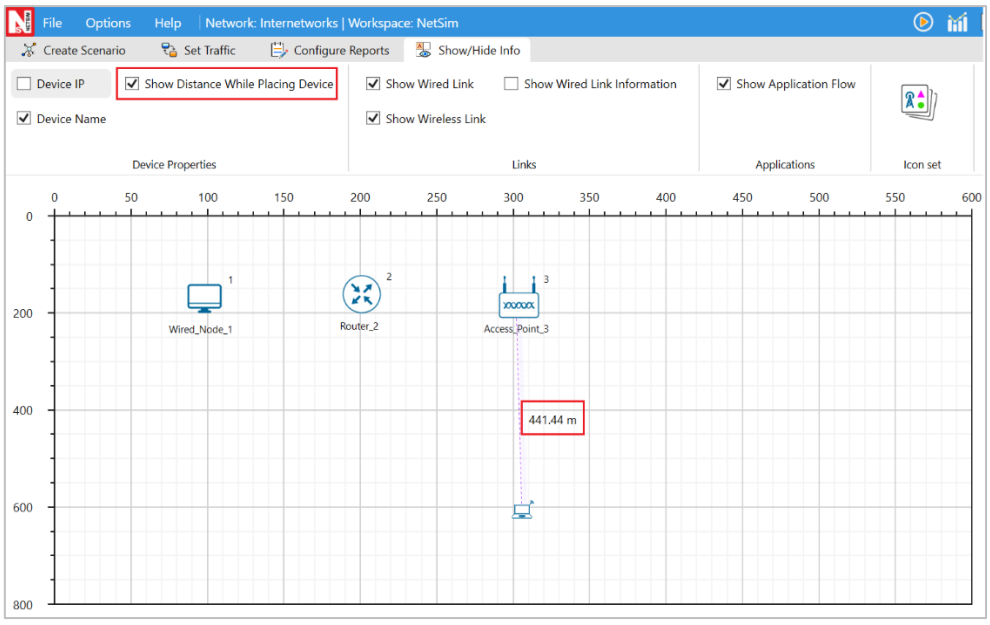


Figure 3-47: Distances displayed while placing devices on the grid.

Icon set in the Show/Hide Info panel will allow the user to change the icons for created scenario.

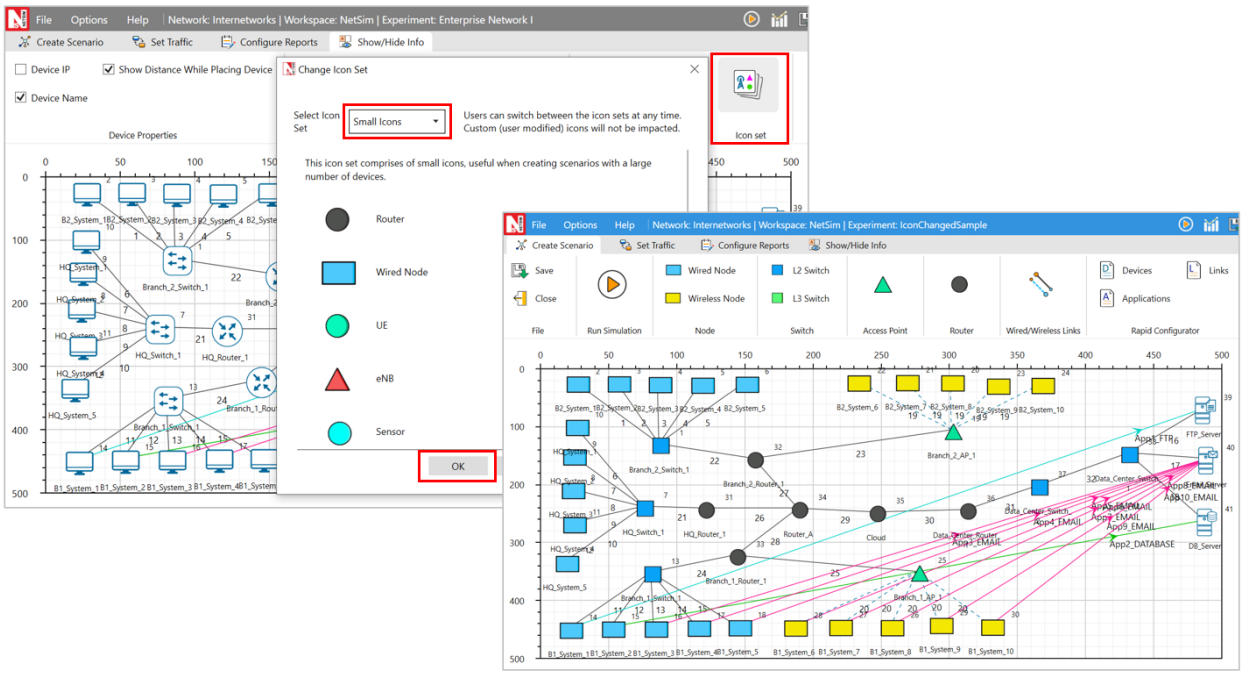


Figure 3-48: Switching icons from default to small icons

3.7 Auto save

Auto Save automatically saves your scenarios in the case of application hang or crash, power outage, or accidental closure of the program. Users can open the autosaved scenarios from the Your Work > Auto Save folder.

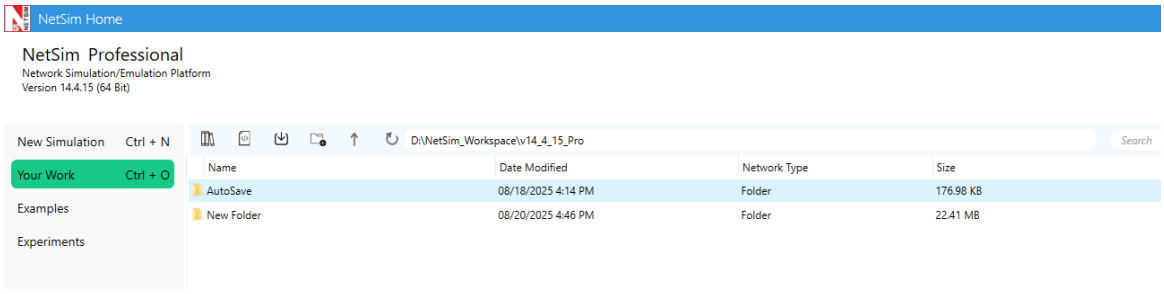


Figure 3-49: AutoSave folder

3.8 Property Panel Headers

The property panel headers for Devices, Links, and Application window can be found on the right-hand side of the Grid.

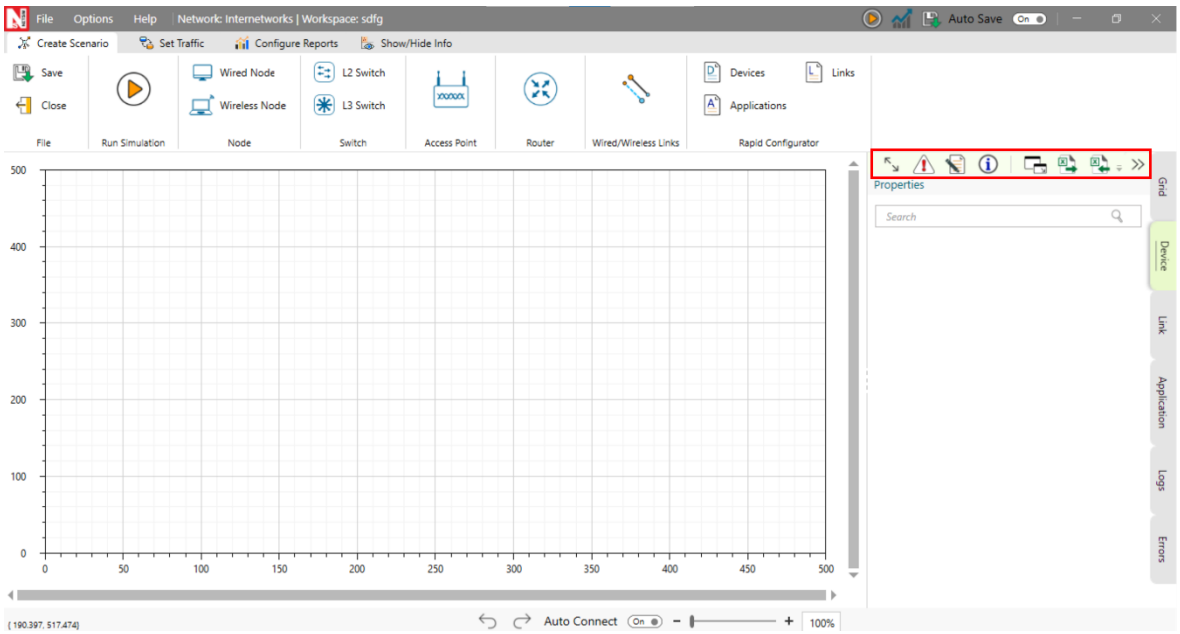


Figure 3-50: Property panel headers available in GUI.

Property Panel Headers have many different features as explained below.

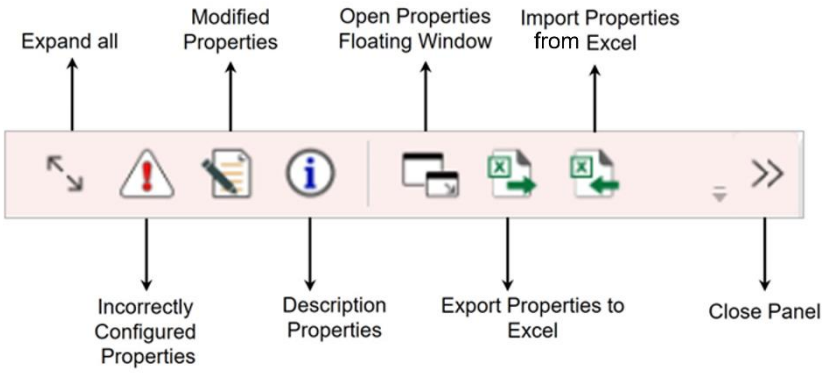


Figure 3-51: Features available in property panel header.

Expand All: Users can utilize this option to expand all the properties present in the configuration panel.

Incorrectly Configured Properties: This feature aids users in identifying specific parameters that have been configured incorrectly.

Modified Properties: The "Modified Properties" feature highlights parameters that have been recently configured or adjusted, making it easy for users to track changes.

Description Properties: This option displays information about the parameter.

Open Properties in Floating Window: By selecting this option, a floating window will open, allowing users to simultaneously edit the device properties.

Export Properties to Excel: This NetSim feature enables users to save device, link, or application configurations into an Excel spreadsheet, facilitating convenient editing and reuse.

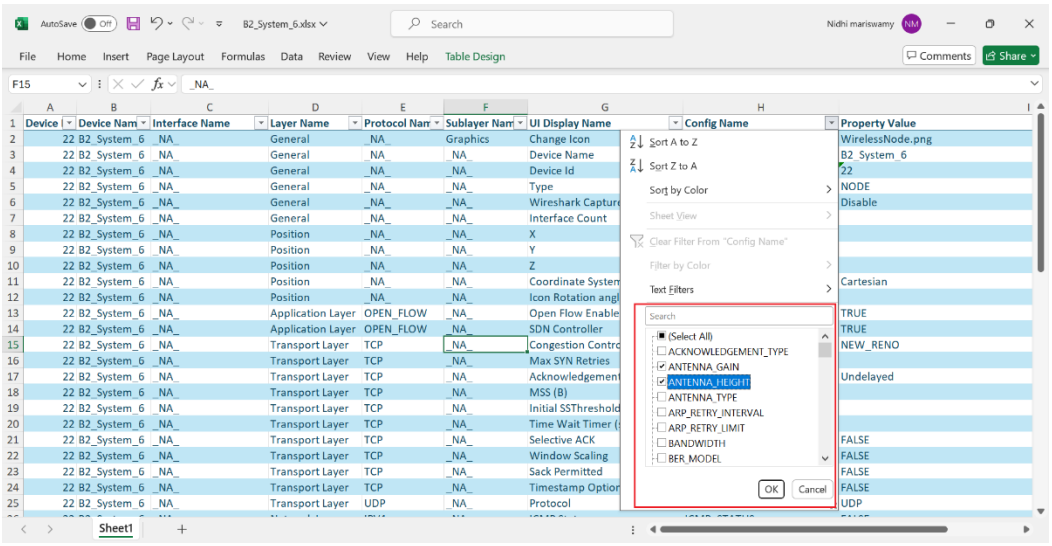


Figure 3-52: Editing properties after clicking on “Export Properties to Excel”.

Import Properties from Excel: In NetSim, "Import Properties from Excel" enables users to effortlessly apply configurations that have been previously saved in an Excel spreadsheet to devices, links, or applications within their network simulations.

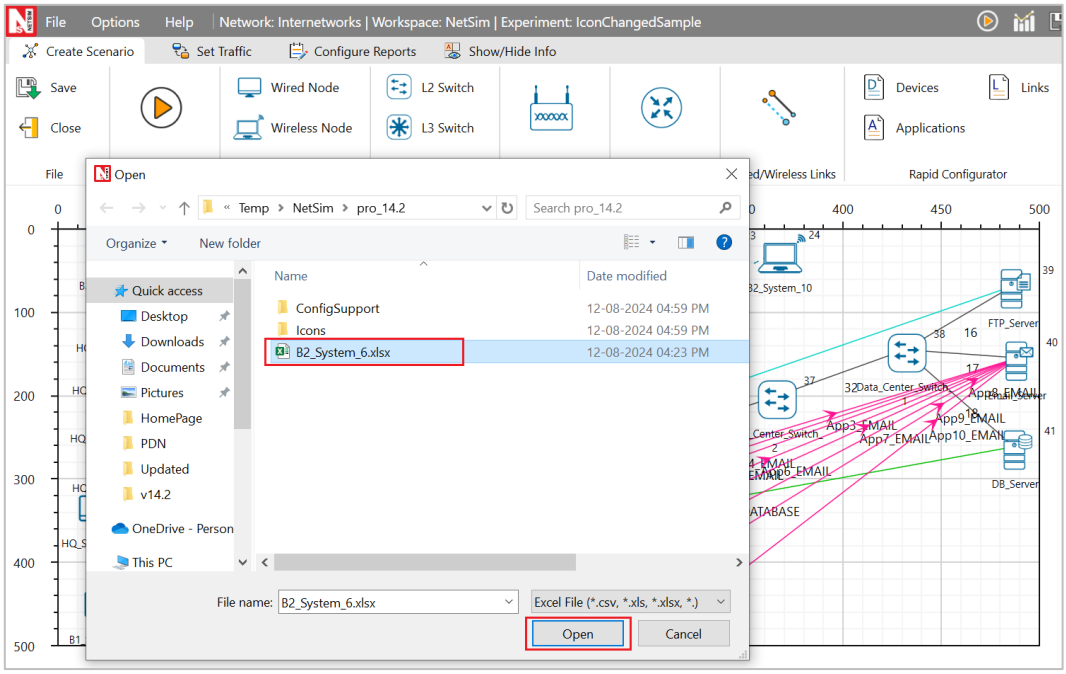


Figure 3-53: Importing edited properties after clicking on " Import Properties from Excel".

Close Panel: This option allows users to close the currently active panel or window quickly and easily.

3.9 Rapid configurator

NetSim v14 has a new feature called the Rapid Configuration window, designed to significantly speed-up the process of creating applications, devices, and links.

3.9.1 Rapid Device configurator

Users can find the Rapid Device Configurator functionality in the top ribbon on the design window.

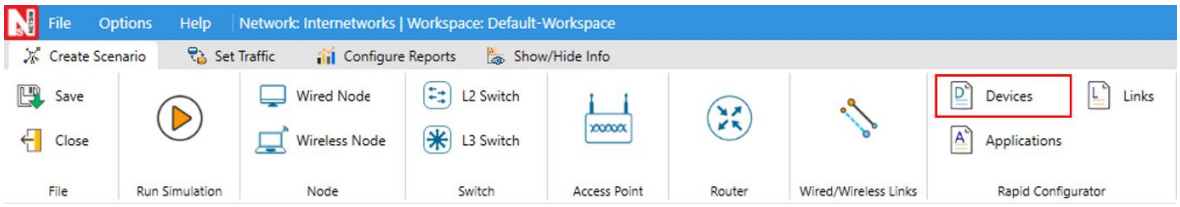


Figure 3-54: Location of the Rapid Device Configurator in design window

When you click the Rapid Device Configurator option, the following window opens

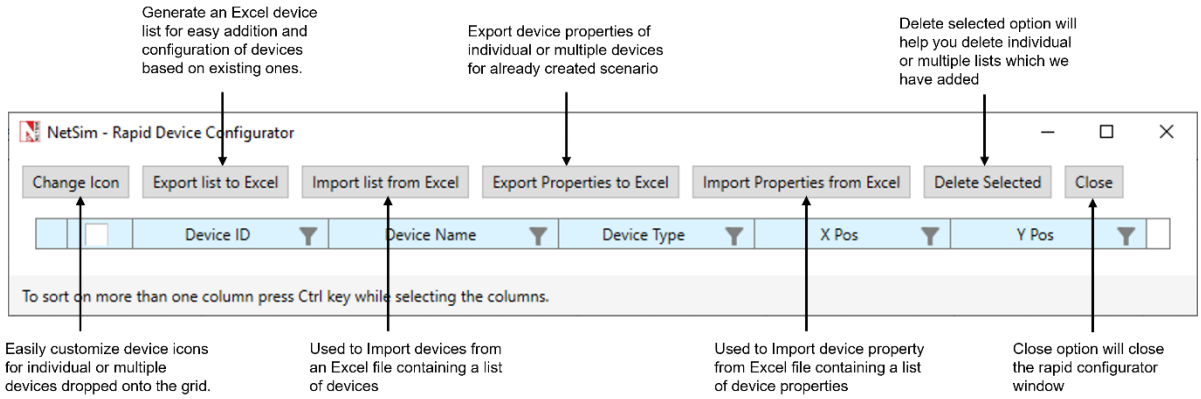


Figure 3-55: Different options within the Rapid Device Configurator

3.9.1.1 Purpose

1. Create large scale scenarios using Excel.
2. Configure the selected devices' (or all devices') properties through the NetSim UI
3. Delete selected/all devices from the scenario.
4. Backup the device properties to excel and use it as template for future scenarios.

3.9.1.2 Steps to Configure Devices

We present below a simple example from the Internetworks library involving an access point and wireless nodes.

Step 1 - Click on the **Create Scenario Tab** and select the **devices** required to create the scenario.

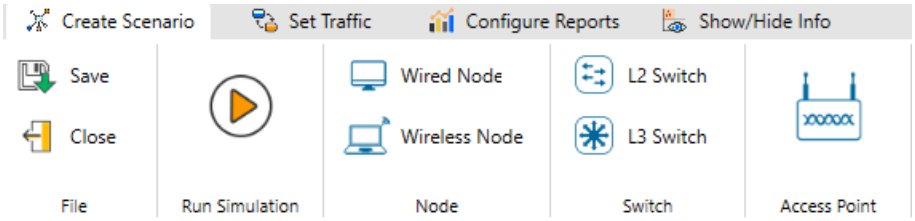


Figure 3-56: Create Scenario Tab

Click on the environment (the grid) where the user wants to drop the devices.

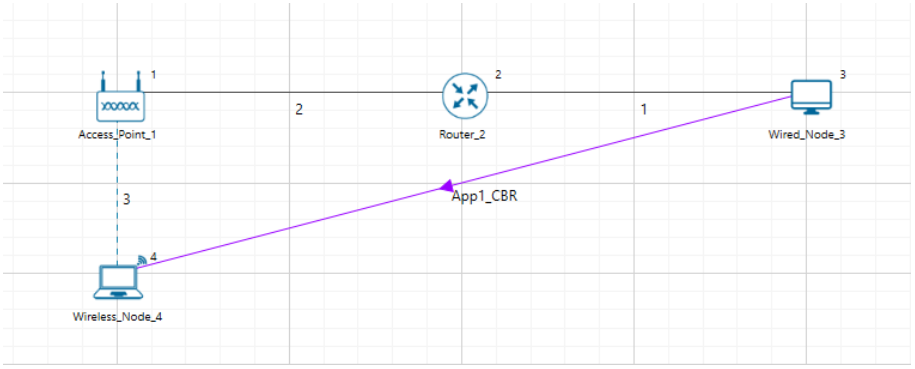


Figure 3-57: A basic Scenario

In the screenshot above, a simple scenario has been created with a Wired node, Router, Access-Point, and Wireless node, with an application configured from Wired node-3 to Wireless node-4. However, if the user needs to connect a large number of Wireless Nodes to the Access-Point, manually dropping and configuring each device can be time-consuming. The Rapid Configurator addresses this challenge by enabling users to speed up adding multiple devices, links, and applications to their network.

Step 2: Go to the top ribbon in **Rapid configurator** and select **Devices** which will open a new Rapid Device Configurator window as shown below.

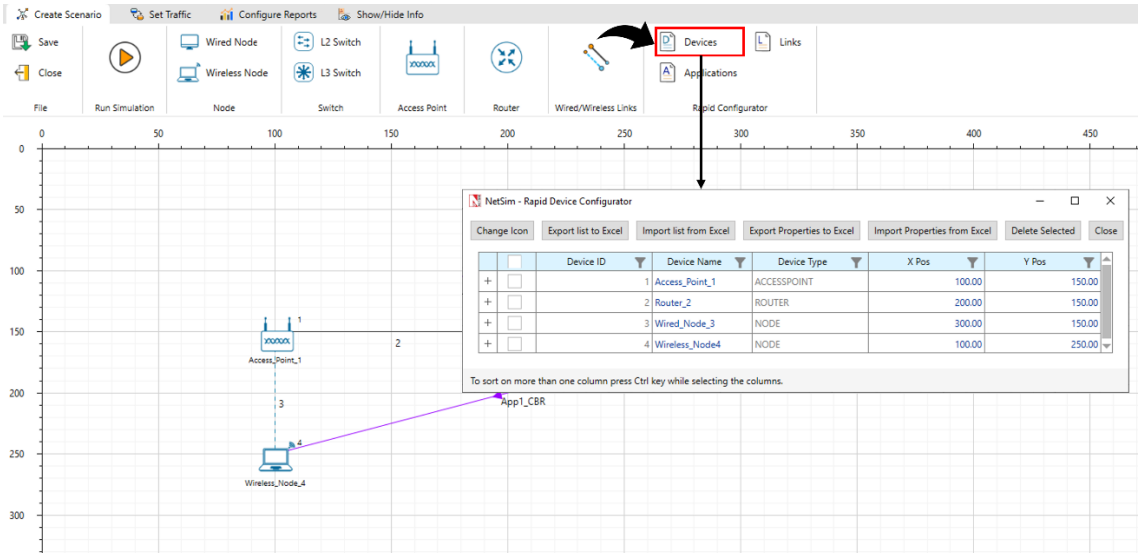


Figure 3-58: Rapid Device Configurator window in GUI

Step 3: In the Rapid Device Configurator window click on **Export list to Excel** option.

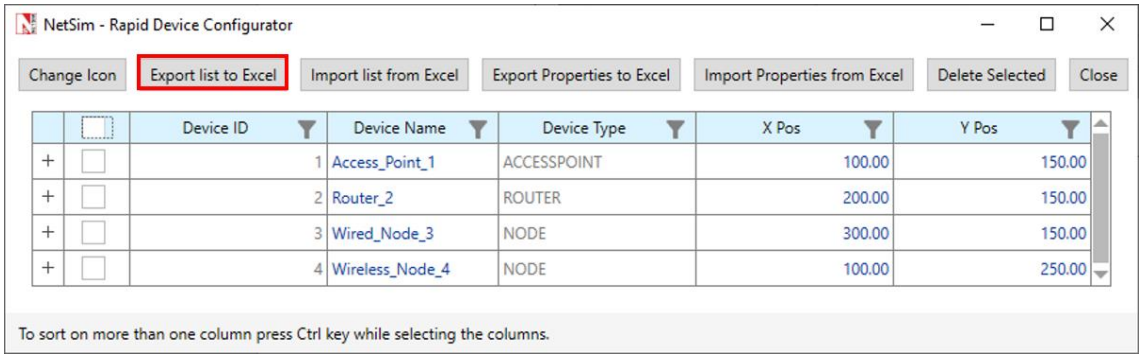


Figure 3-59: Export list to Excel option in Rapid Device Configurator

Step 4: This action will open an **Excel spreadsheet** where users can observe that the device list has been exported from the created scenario.

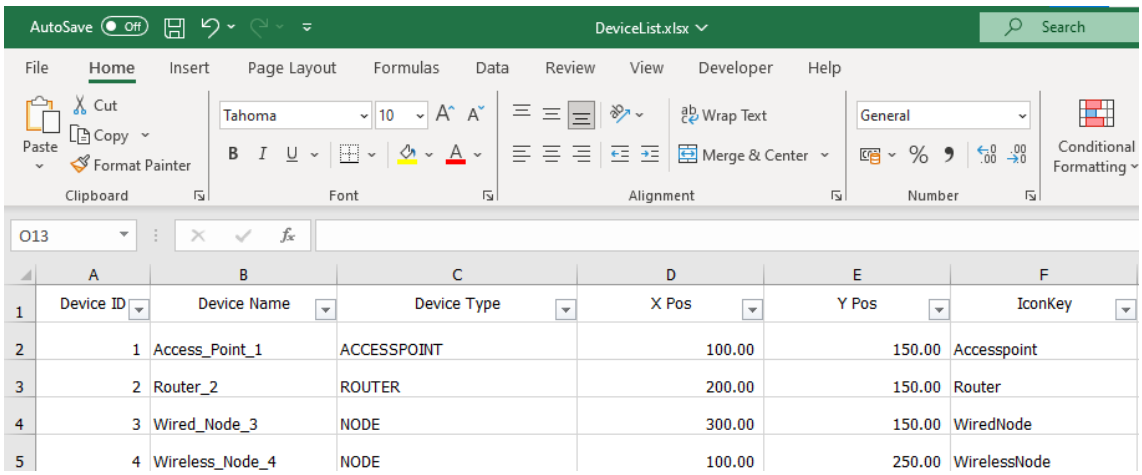


Figure 3-60: Device List in Excel sheet

Step 5: Users can use Excel's features like Autofill to quickly add a number of devices along with their respective coordinates. Then click on save.

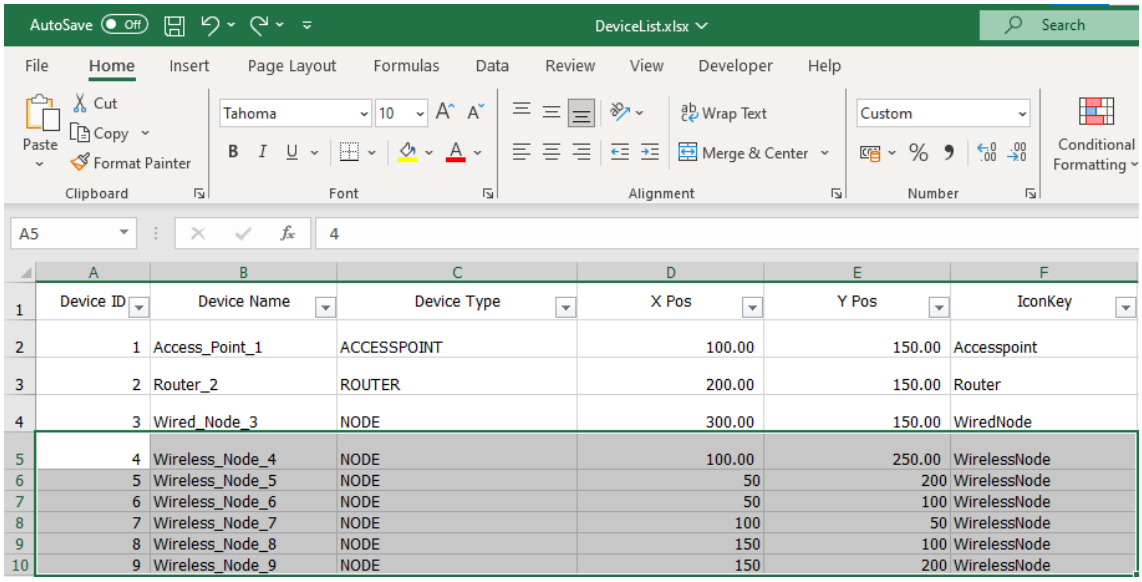


Figure 3-61: Adding new devices into the current sheet.

Step 6: Once it is saved go back to the rapid device configuration window and select **Import list from Excel .**

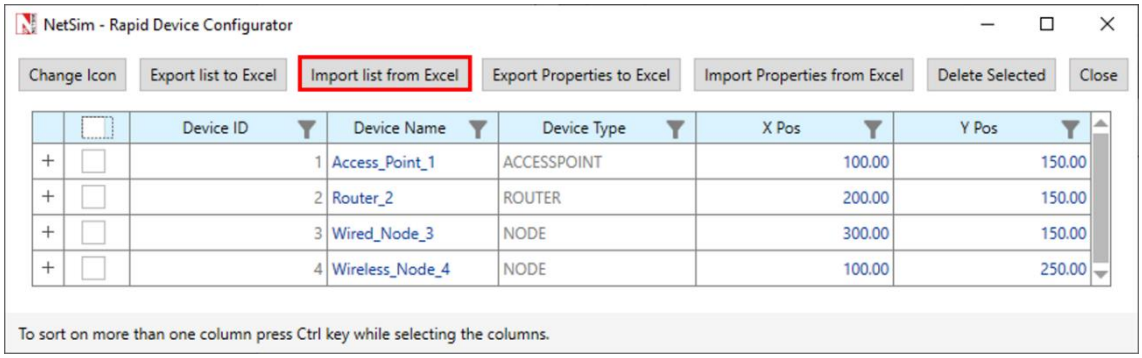


Figure 3-62: Import the updated list from Excel in Rapid Device Configurator

Step 7: This will navigate the user to the folder where **DeviceList.xlsx** is saved. Just click on the file, and it will be imported.

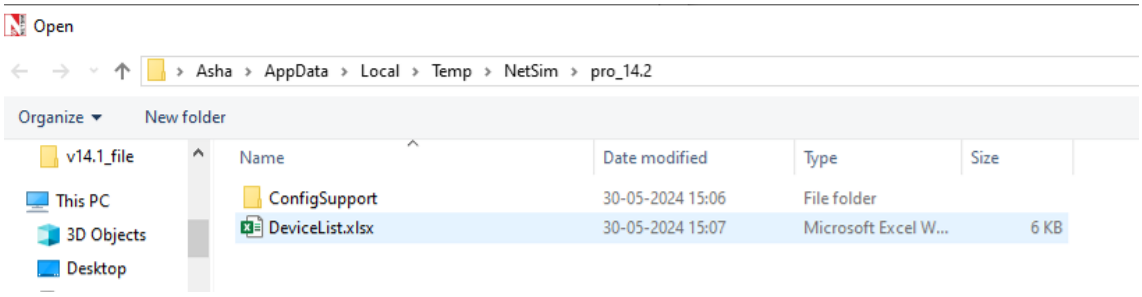


Figure 3-63: DeviceList.xlsx file saved after export.

Step 8: Once the **Import** is successful, users can observe the **newly added devices** successfully placed on the grid.

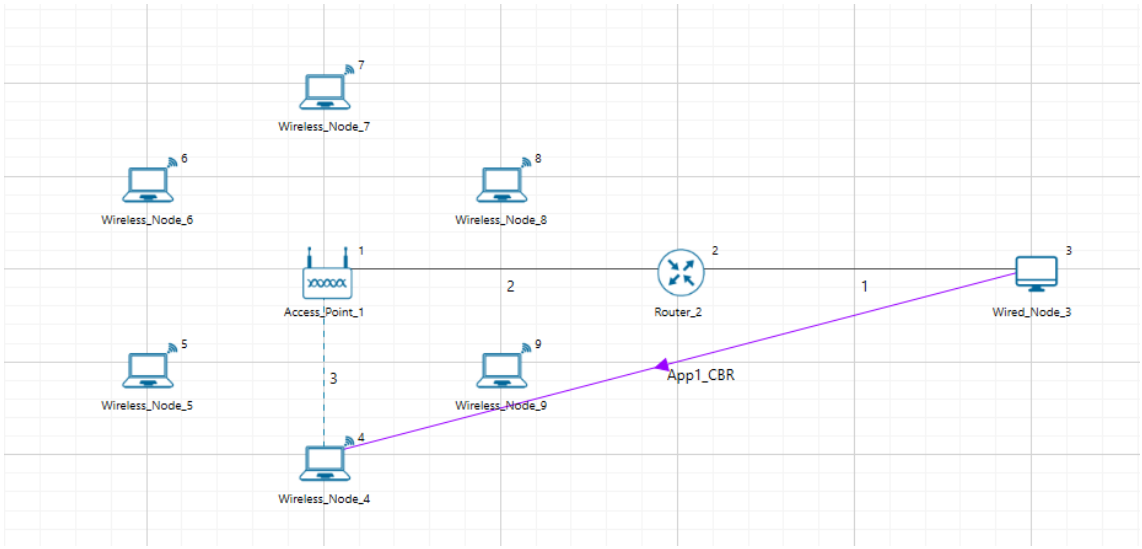


Figure 3-64: New devices added to the current scenario.

Now that users can quickly add new devices to their network using the Rapid Device Configurator, let's explore how to connect them using the Rapid Link Configurator.

3.9.2 Rapid Link Configurator

Users can find the Rapid Link Configurator window in the top ribbon in the Device panel.

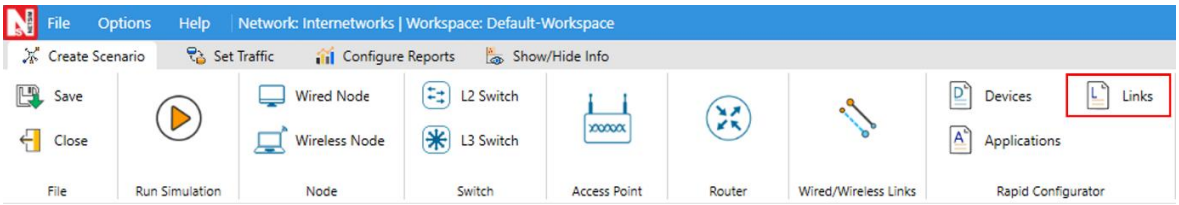


Figure 3-65: Rapid Link Configurator in GUI

Upon clicking the Rapid Link Configurator option, a window will open, containing the features of the Rapid Configurator as shown below.

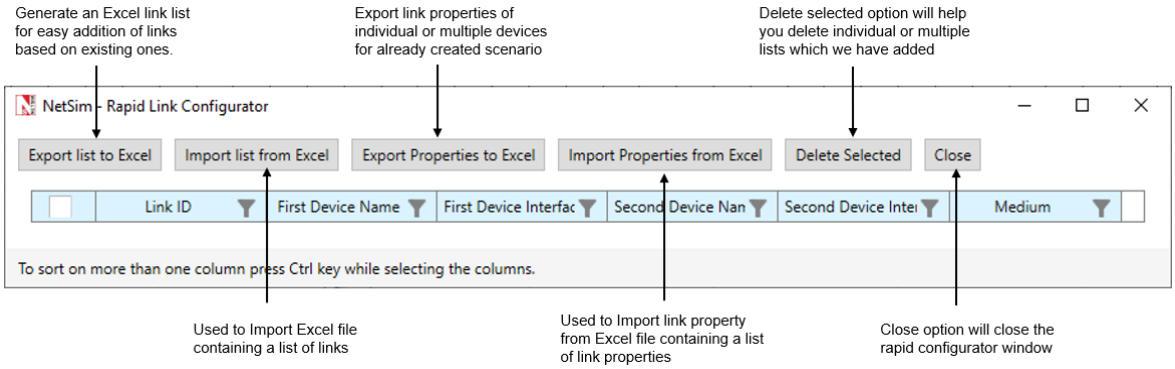


Figure 3-66: Features of Rapid Link Configurator

3.9.2.1 Purpose

1. Links to the devices which are not connected can be added rapidly.
2. Configure selected/all links properties using rapid window or using excel.
3. Delete selected/all links properties.

3.9.2.2 Steps to Configure Links

Step 1: Go to the top ribbon in **Rapid configurator** and select **Links** which will open a new Rapid Link Configurator window as shown below.

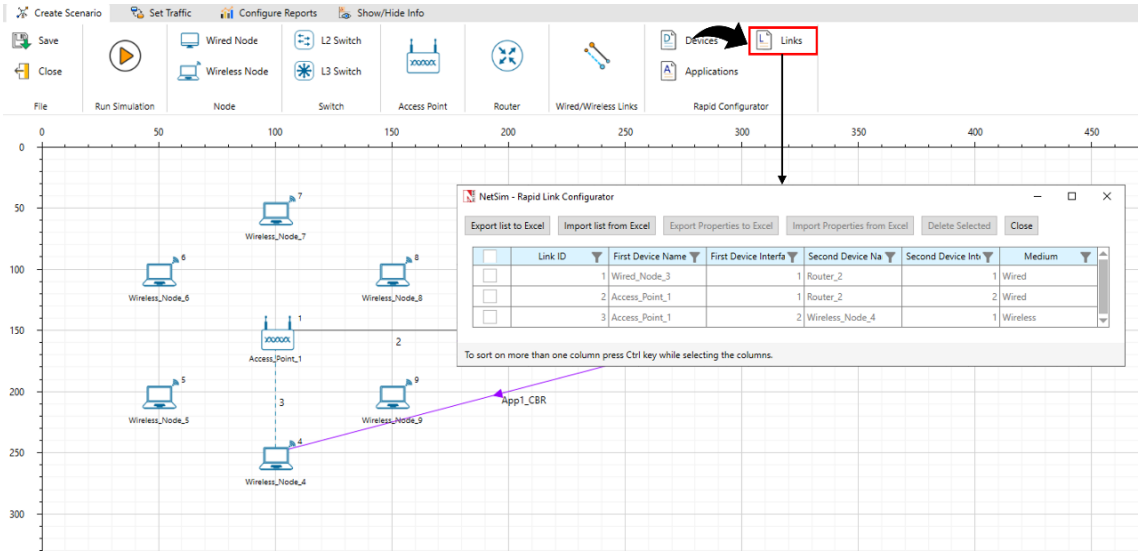


Figure 3-67: Rapid Link Configurator window in GUI

Step 2: In the Rapid Link Configurator window, click on **Export list to Excel** option.

Step 3: This will open an **Excel spreadsheet** where users can observe that the **Connection list** is exported from the created scenario.

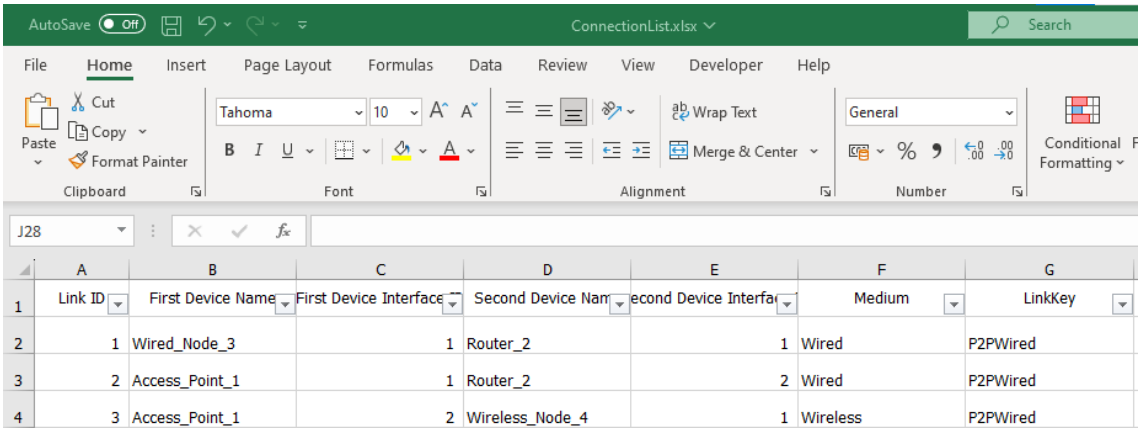
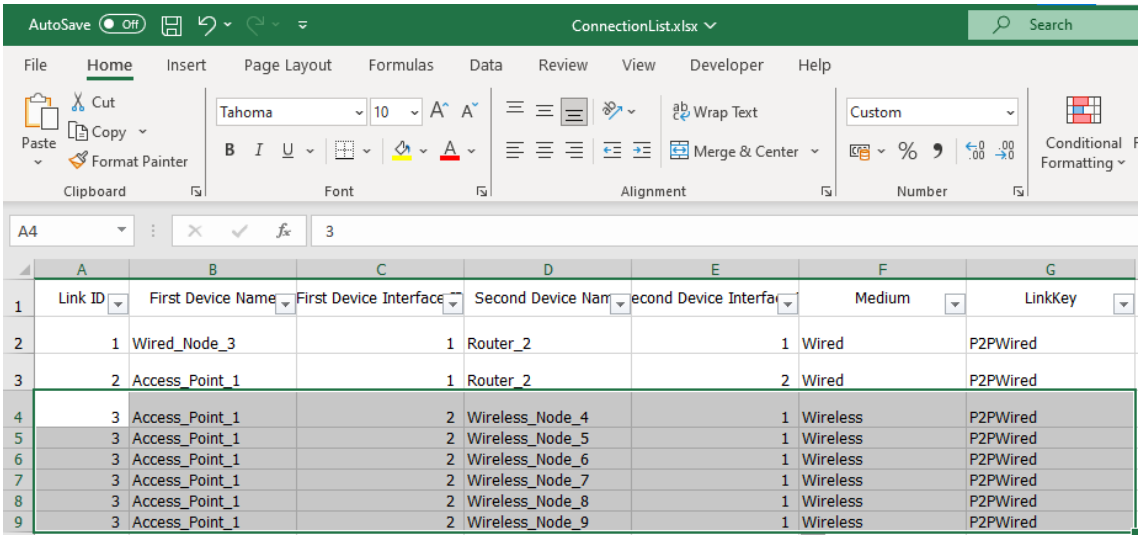


Figure 3-68: Connection List in excel sheet.

Step 4: In Excel, users can add more links with their respective placement coordinates and click on **Save**.



	A	B	C	D	E	F	G
1	Link ID	First Device Name	First Device Interface	Second Device Nam	econd Device Interfa	Medium	LinkKey
2	1	Wired_Node_3	1	Router_2	1	Wired	P2PWired
3	2	Access_Point_1	1	Router_2	2	Wired	P2PWired
4	3	Access_Point_1	2	Wireless_Node_4	1	Wireless	P2PWired
5	3	Access_Point_1	2	Wireless_Node_5	1	Wireless	P2PWired
6	3	Access_Point_1	2	Wireless_Node_6	1	Wireless	P2PWired
7	3	Access_Point_1	2	Wireless_Node_7	1	Wireless	P2PWired
8	3	Access_Point_1	2	Wireless_Node_8	1	Wireless	P2PWired
9	3	Access_Point_1	2	Wireless_Node_9	1	Wireless	P2PWired

Figure 3-69: Adding new links to the current list.

Step 5: Once saved, return to the Rapid Link Configuration window, and select the **Import List from Excel** option.

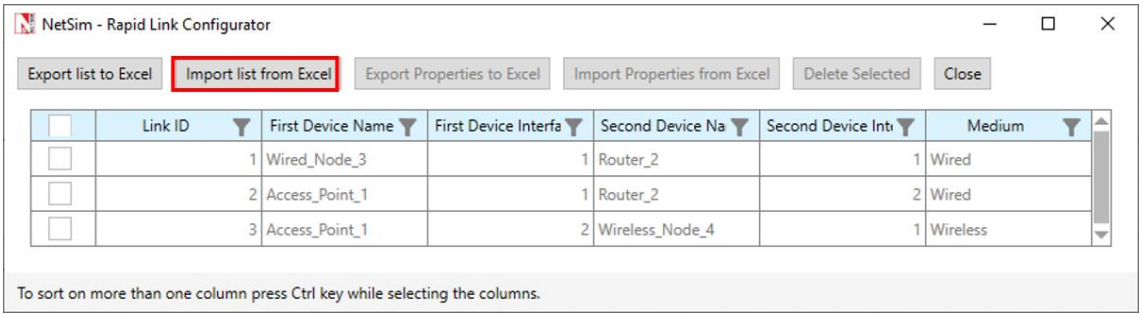


Figure 3-70: Import list from Excel option in Rapid Link Configurator

Step 6: This will direct users to the folder where **ConnectionList.xlsx** is saved. Just click on the file, and it will be imported.

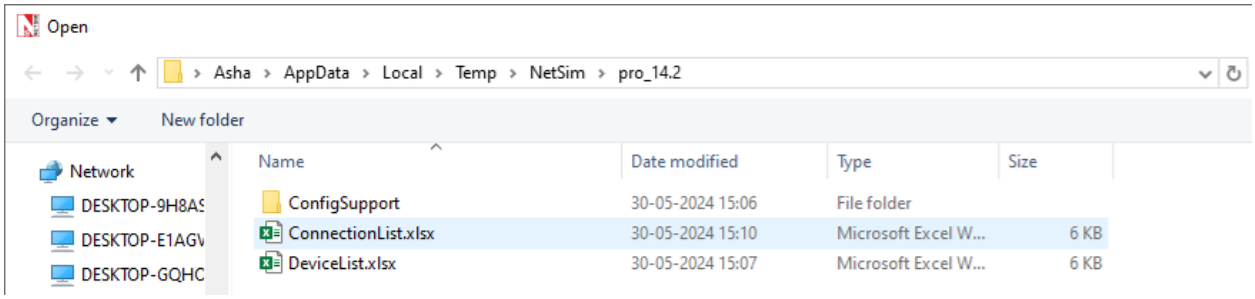


Figure 3-71: ConnectionList.xlsx file saved after export.

Step 7: Upon successful import, users can see that links are automatically connected to the devices previously added with the help of the Rapid Device Configurator.

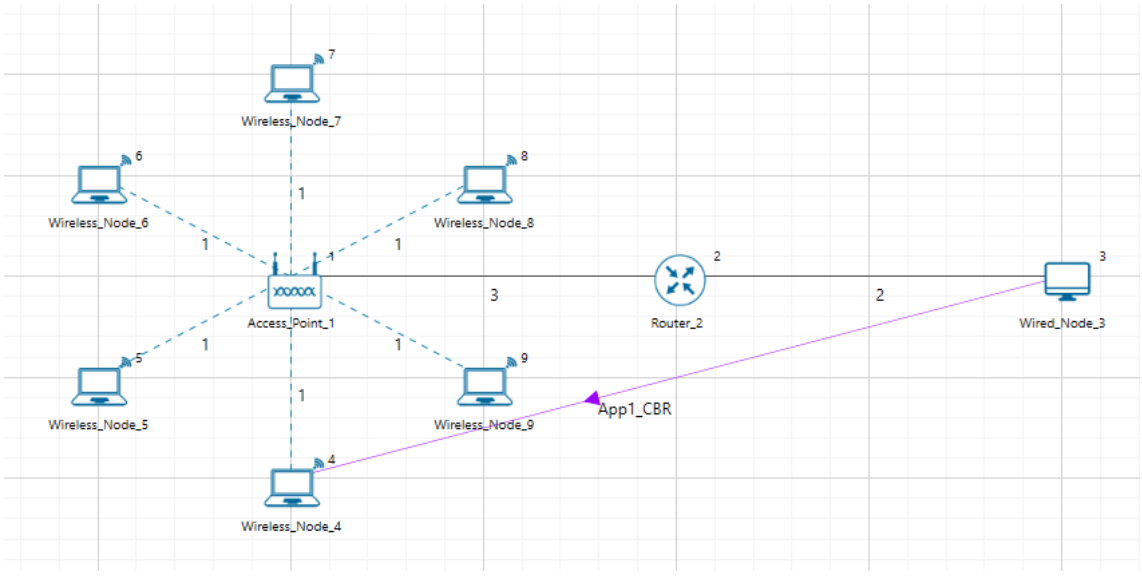


Figure 3-72: Links have now configured between the wireless nodes and the access point.

Now that users can effortlessly connect links to devices using the Rapid Link Configurator, let's explore how to configure applications for the remaining devices using the Rapid Application Configurator.

3.9.3 Rapid Application Configurator

Users can find the Rapid Application Configurator window in the top ribbon in the Device panel.

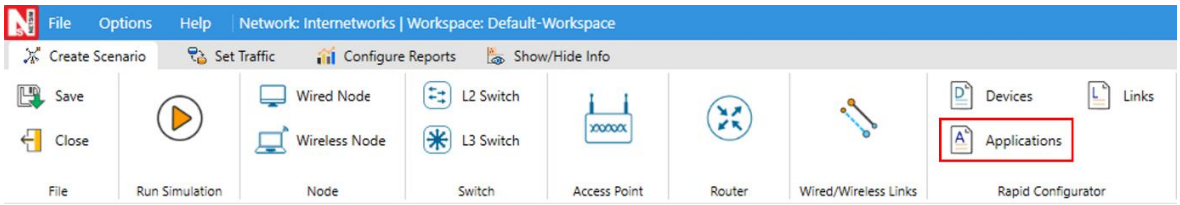


Figure 3-73: Rapid Application Configurator in GUI

Upon clicking the Rapid Application Configurator option, a window will open, containing the features of the Rapid Configurator as shown below.

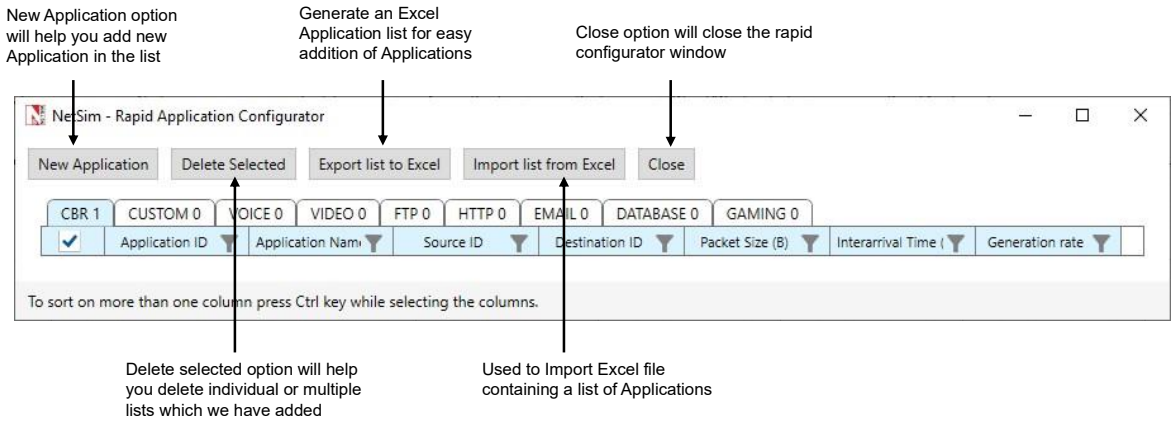


Figure 3-74: Features of Rapid Application Configurator

3.9.3.1 Purpose

1. Create large scale application using Excel or simply using add application option.
2. Delete selected or all applications using delete option.
3. Configure important applications properties using rapid window or using excel.

3.9.3.2 Steps to Configure Applications

Step 1: Go to the top ribbon in **Rapid configurator,** select **Applications** which will open our new Rapid Application Configurator window.

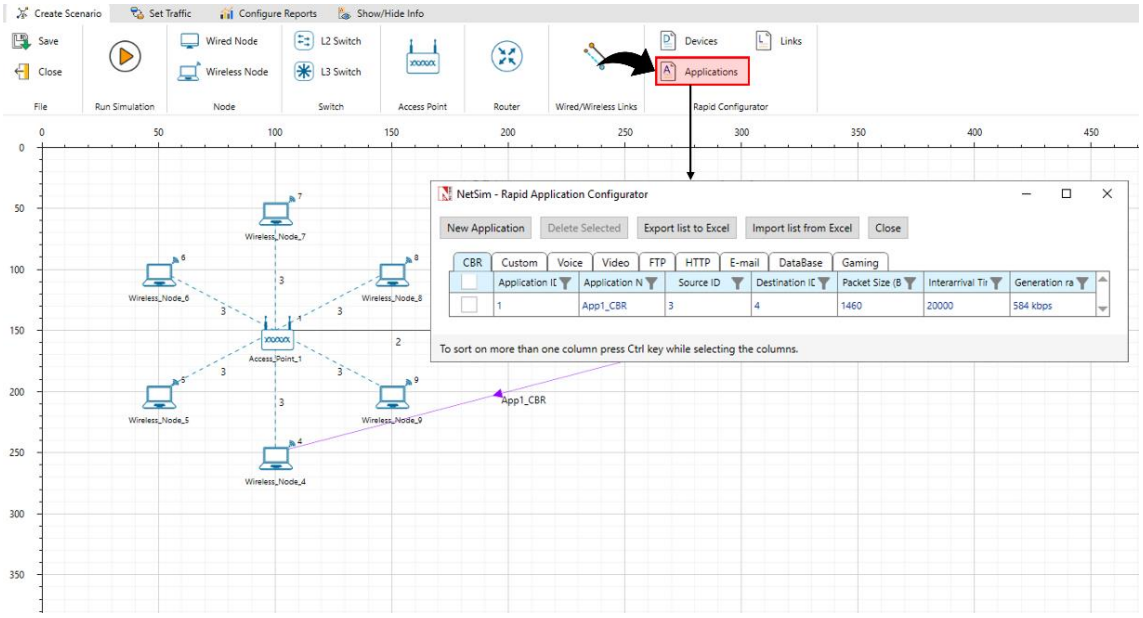


Figure 3-75: Rapid Application Configurator window in GUI

Step 2: In the Rapid Application Configurator window, click on **Export list to Excel** option.

Step 3: This will open an **Excel spreadsheet** where users can observe that the **Application list** is Exported from the created scenario.

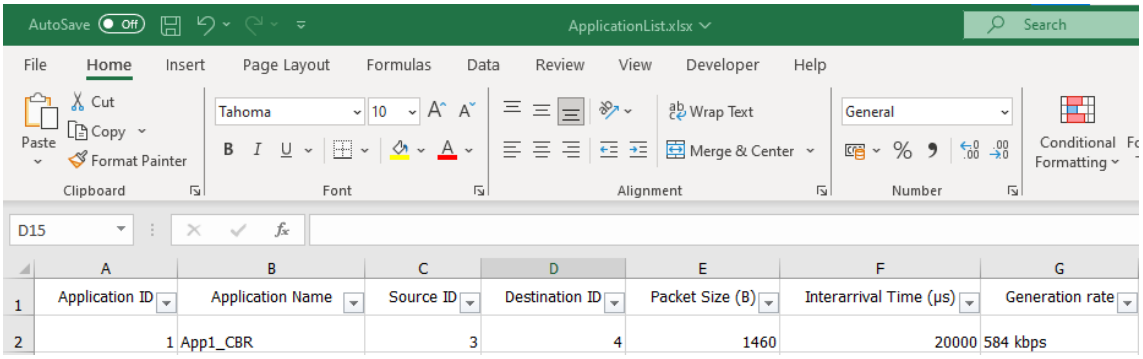
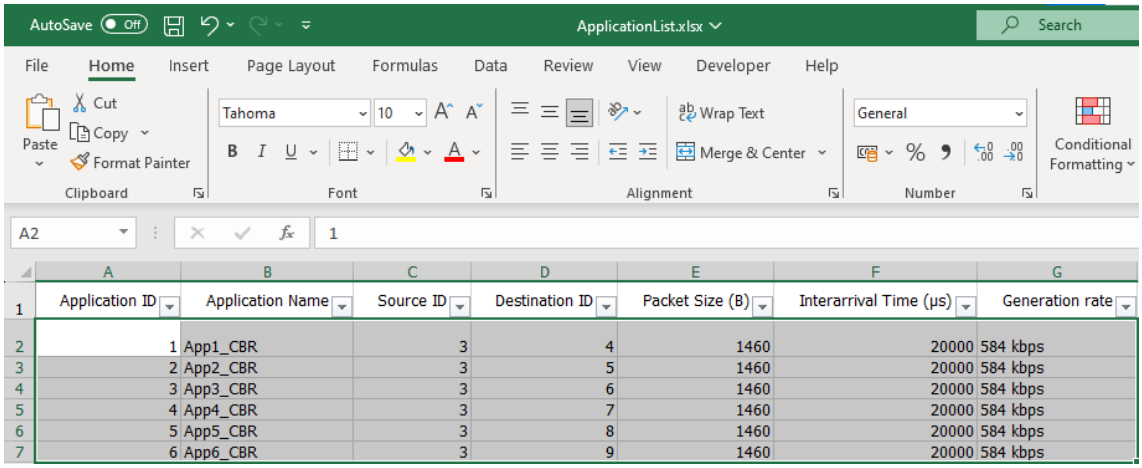


Figure 3-76: Application list in Excel

Step 4: Excel will allow users to add more number of devices with their respective placement co-ordinates and click on **Save**.



	A	B	C	D	E	F	G
1	Application ID	Application Name	Source ID	Destination ID	Packet Size (B)	Interarrival Time (µs)	Generation rate
2	1	App1_CBR	3	4	1460	20000	584 kbps
3	2	App2_CBR	3	5	1460	20000	584 kbps
4	3	App3_CBR	3	6	1460	20000	584 kbps
5	4	App4_CBR	3	7	1460	20000	584 kbps
6	5	App5_CBR	3	8	1460	20000	584 kbps
7	6	App6_CBR	3	9	1460	20000	584 kbps

Figure 3-77: Adding new applications to the list

Step 5: Once saved, return to the Rapid Application Configuration window and select the **Import List from Excel** option.

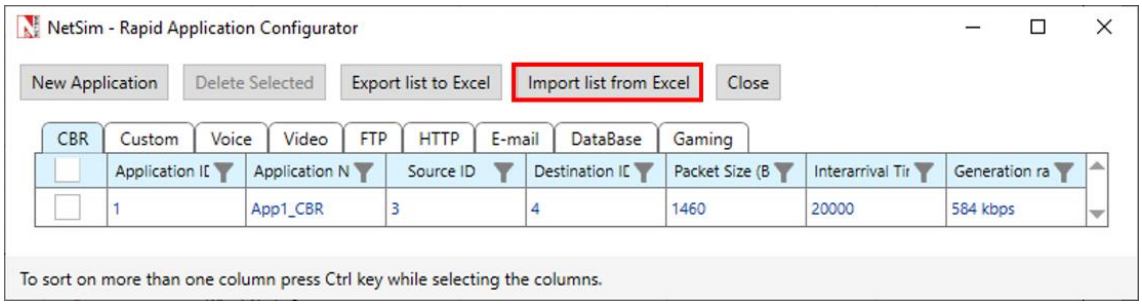


Figure 3-78: Import list from Excel option in Rapid Application Configurator

Step 6: This will direct users to the folder where **ApplicationList.xlsx** is saved. Just click on the file, and it will be imported.

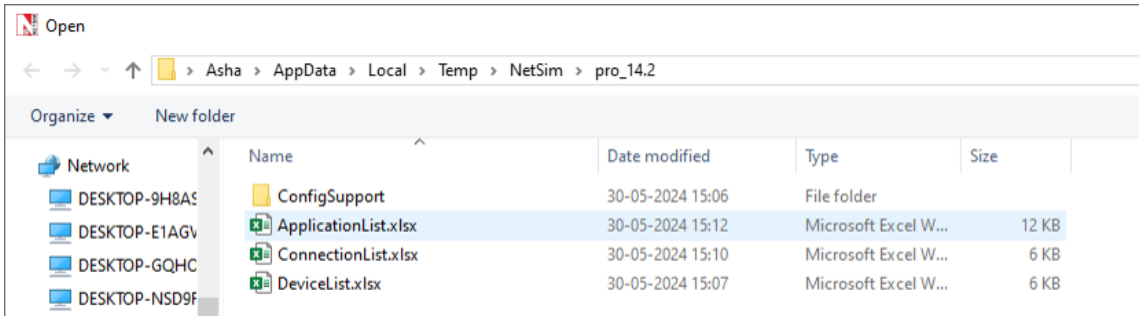


Figure 3-79: ApplicationList.xlsx file saved after exporting.

Step 7: Upon successful import, users can see that Applications are automatically configured between the devices with the help of the Rapid Application Configurator.

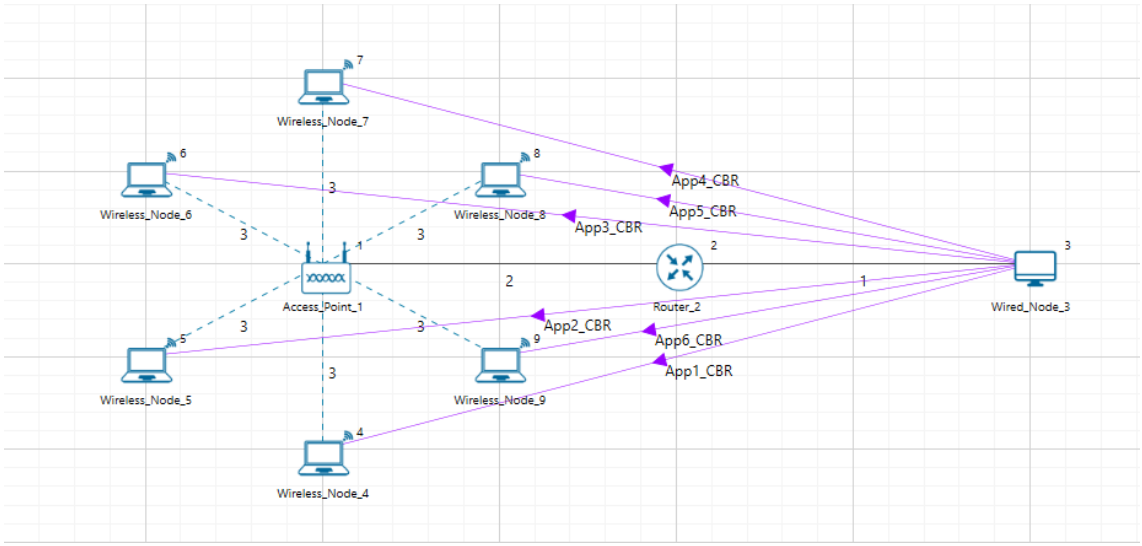


Figure 3-80: New applications added to the current scenario.

Now that users can create scenarios using the Rapid Device, Link, and Application Configurator, let's delve into some more of NetSim's latest features. With the new Excel feature, users can also configure device, link, and application properties.

3.9.4 Limitations of Rapid configurator

- There's currently no validation mechanism in place when importing data from Excel. Importing incorrect parameter settings can potentially lead to an application crash during simulation.
    - Name and ID of devices, applications and links cannot be duplicated.
    - Fields in the configurator do not have any validation based on characters and numbers.
- Device, Link, and Application names should not have special characters.
- The input in Excel should be values only, cells with mathematical formulas are not supported during import.
- Exported properties and device/application/link lists should be manually saved in a suitable directory for the backup purposes. NetSim currently exports these files to the temp folder (%temp%\NetSim\std_14.2); these files will not get saved when saving the experiment.
- Users need to be careful when using the Export option multiple times as it would lead to the overwriting of previously exported files.
- Users should be careful when changing *global* properties. NetSim will set all global properties based on the last device in Excel. Users should refer to NetSim documentation (of the specific network) before modifying global properties using the rapid configurator.

3.10 Configuring Device Properties using the import-from/export-to Excel option

We will now explore a simple example of how users can configure device properties with the export to Excel option.

Step 1: Right-click on Access Point and select Open Properties as New Window, which will open the Access Point Properties window. Here, users can choose the **Export Properties to Excel option.**

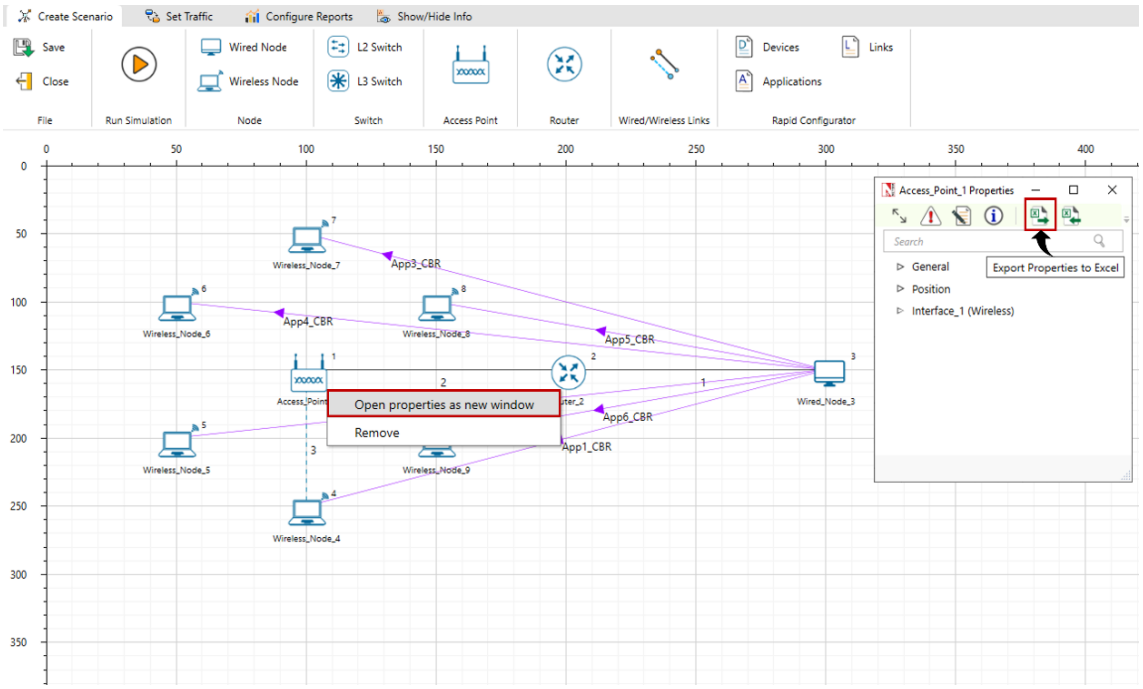
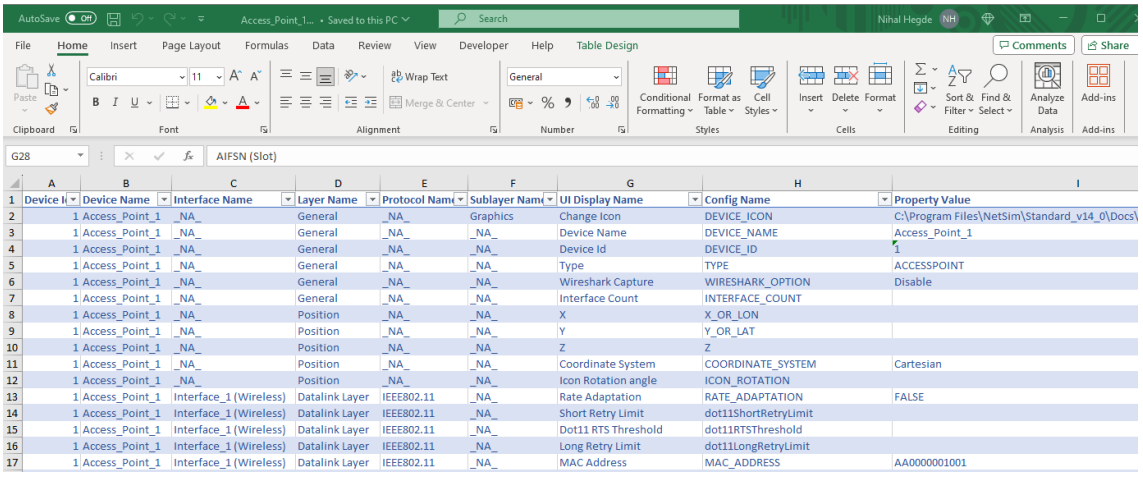


Figure 3-81: Right click on the device to open properties.

Step 2: This will open an Excel sheet where users can view all the Device properties related to the Device that was selected.



Device Id	Device Name	Interface Name	Layer Name	Protocol Name	Sublayer Name	UI Display Name	Config Name	Property Value
1	Access_Point_1	_NA_	General	_NA_	Graphics	Change Icon	DEVICE_ICON	C:\Program Files\NetSim\Standard_v14_0\Docs\
1	Access_Point_1	_NA_	General	_NA_	_NA_	Device Name	DEVICE_NAME	Access_Point_1
1	Access_Point_1	_NA_	General	_NA_	_NA_	Device Id	DEVICE_ID	1
1	Access_Point_1	_NA_	General	_NA_	_NA_	Type	TYPE	ACCESSPOINT
1	Access_Point_1	_NA_	General	_NA_	_NA_	Wireshark Capture	WIRESHARK_OPTION	Disable
1	Access_Point_1	_NA_	General	_NA_	_NA_	Interface Count	INTERFACE_COUNT	
1	Access_Point_1	_NA_	Position	_NA_	_NA_	X	X_OR_LON	
1	Access_Point_1	_NA_	Position	_NA_	_NA_	Y	Y_OR_LAT	
1	Access_Point_1	_NA_	Position	_NA_	_NA_	Z	Z	
1	Access_Point_1	_NA_	Position	_NA_	_NA_	Coordinate System	COORDINATE_SYSTEM	Cartesian
1	Access_Point_1	_NA_	Position	_NA_	_NA_	Icon Rotation angle	ICON_ROTATION	
1	Access_Point_1	Interface_1 (Wireless)	Datalink Layer	IEEE802.11	_NA_	Rate Adaptation	RATE_ADAPTATION	FALSE
1	Access_Point_1	Interface_1 (Wireless)	Datalink Layer	IEEE802.11	_NA_	Short Retry Limit	dot11ShortRetryLimit	
1	Access_Point_1	Interface_1 (Wireless)	Datalink Layer	IEEE802.11	_NA_	Dot11 RTS Threshold	dot11RTSThreshold	
1	Access_Point_1	Interface_1 (Wireless)	Datalink Layer	IEEE802.11	_NA_	Long Retry Limit	dot11LongRetryLimit	
1	Access_Point_1	Interface_1 (Wireless)	Datalink Layer	IEEE802.11	_NA_	MAC Address	MAC_ADDRESS	AA0000001001

Figure 3-82: Properties of Access point

In this example, we will demonstrate how to change Transmitter Power and IEEE standard with the help of Excel input.

Step 3: Select UI Display Name and filter Standard and Transmitter Power.

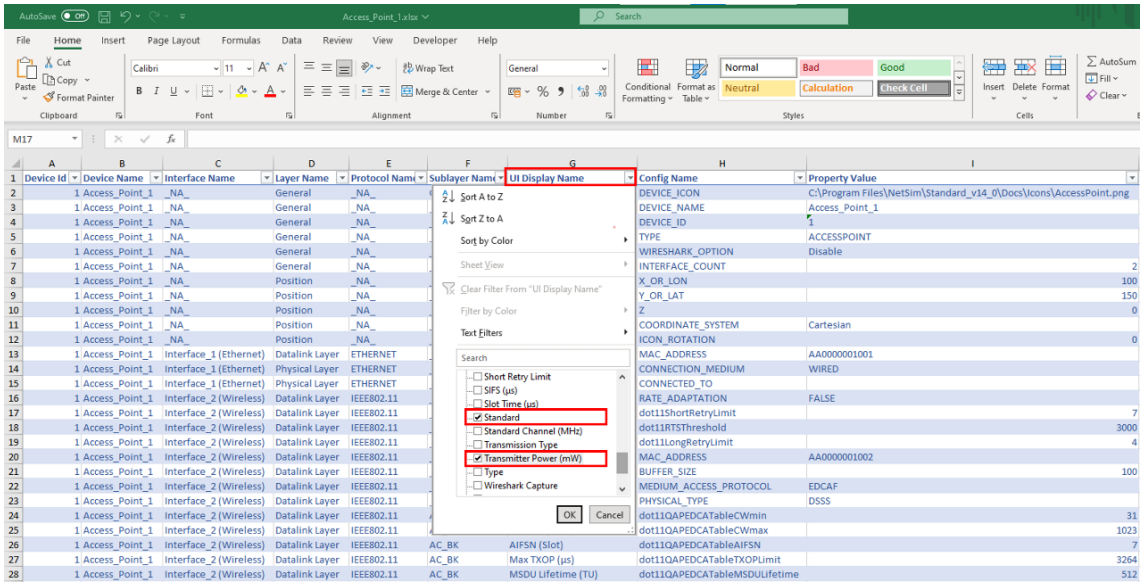


Figure 3-83: Filter Standard and Transmitter Power

Step 4: In the Property Value tab, users can see that Standard is IEEE802.11b, and Transmitter Power is 100.

	A	B	C	D	E	F	G	H	
1	Device Id	Device Name	Interface Name	Layer Name	Protocol Name	Sublayer Name	UI Display Name	Config Name	Property Value
46	1	Access_Point_1	Interface_2 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11b
48	1	Access_Point_1	Interface_2 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	100

Figure 3-84: Values of standard and transmit power before modifying.

Step 5: Now, update Standard to IEEE802.11g and Transmitter Power to 20. After making these changes, save the Excel file.

	A	B	C	D	E	F	G	H	
1	Device Id	Device Name	Interface Name	Layer Name	Protocol Name	Sublayer Name	UI Display Name	Config Name	Property Value
46	1	Access_Point_1	Interface_2 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
48	1	Access_Point_1	Interface_2 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20

Figure 3-85: Values of standard and transmit power after modifying.

Step 6: Once the Excel file is saved, click on Import from Excel option in the Access_Point_1 Properties Window.

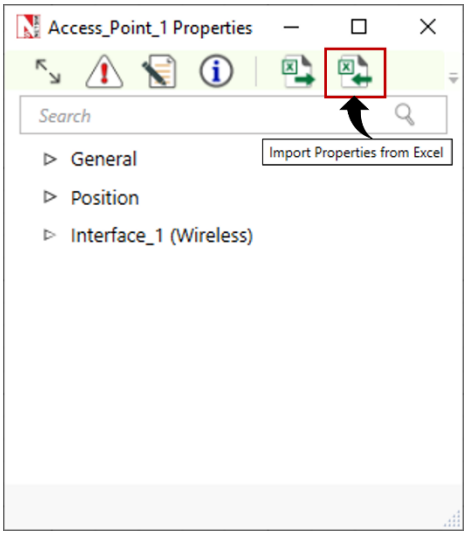


Figure 3-86: Import to excel option in GUI.

Step 7: This will direct users to the folder where **Access_Point_1.xlsx** is saved. Click on the file for it to be imported.

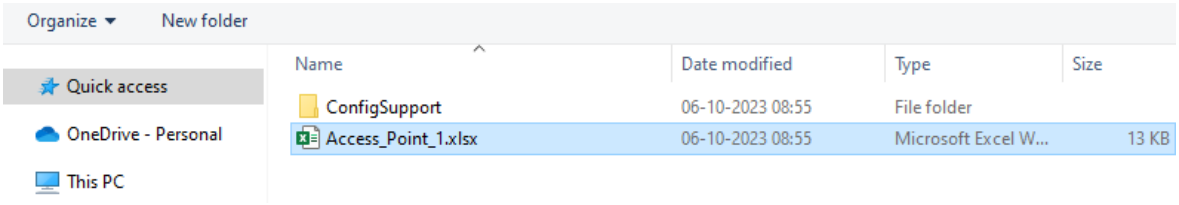


Figure 3-87: Access_Point_1.xlsx file saved after exporting.

Step 8: Upon successful import, right click on Access_Point_1 property and expand Interface_2(Wireless) > Physical layer > IEEE802.11, where users can find the Transmitter Power.

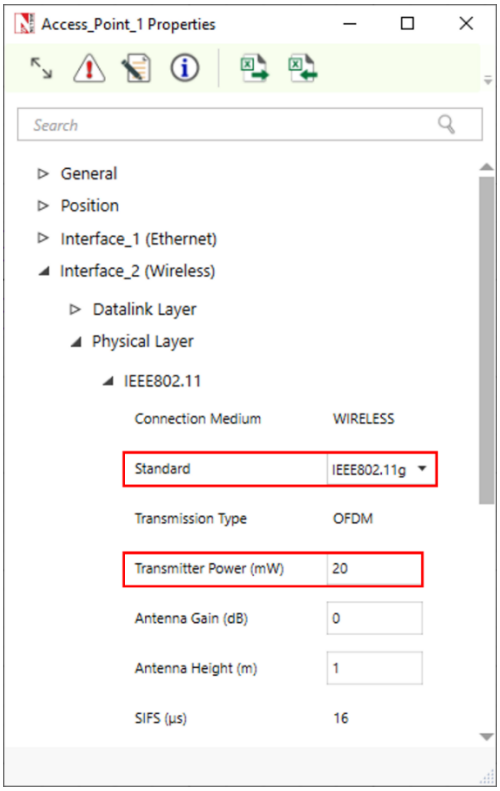


Figure 3-88: Modified Standard and Transmitter Power values.

The example above is just one simple demonstration of how users can utilize the Excel option to configure device properties.

The new Excel feature in NetSim v14 goes beyond configuring device, link, and application properties; it can also be employed to configure multiple device properties simultaneously. For instance, in the Rapid Device Configurator window, users can select multiple devices and change all their properties at once.

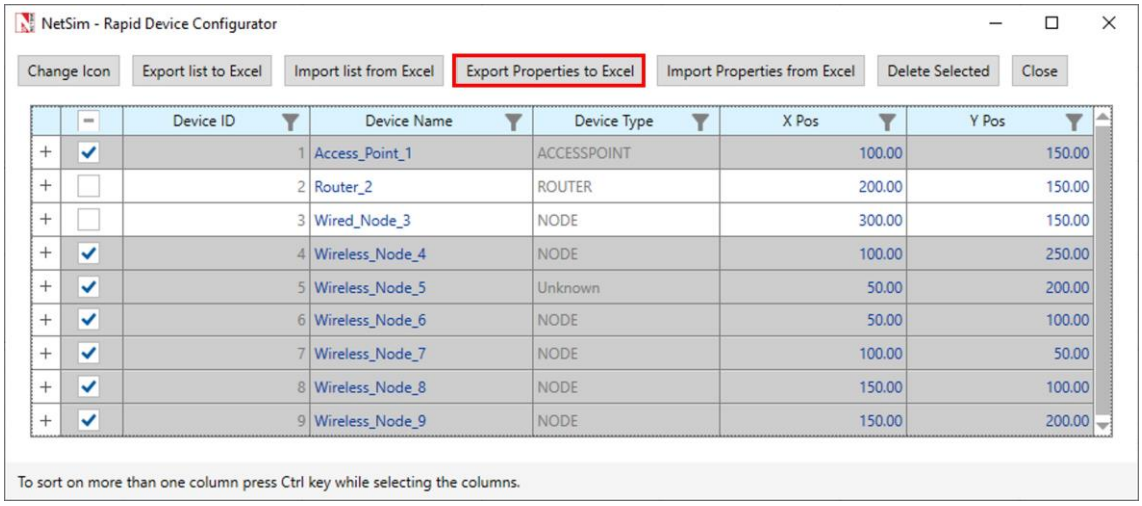


Figure 3-89: Export to excel option in Rapid Device Configurator

After filtering UI Display Name for Transmitter Power and Standard, users have set the property value to 20 and IEEE802.11g for all the devices.

	A	B	C	D	E	F	G	H	
1	Device I	Device Name	Interface Name	Layer Name	Protocol Nam	Sublayer Nam	UI Display Name	Config Name	Property Value
46	1	Access_Point_1	Interface_2 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
48	1	Access_Point_1	Interface_2 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20
123	4	Wireless_Node_4	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
125	4	Wireless_Node_4	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20
200	5	Wireless_Node_5	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
202	5	Wireless_Node_5	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20
277	6	Wireless_Node_6	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
279	6	Wireless_Node_6	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20
354	7	Wireless_Node_7	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
356	7	Wireless_Node_7	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20
431	8	Wireless_Node_8	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
433	8	Wireless_Node_8	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20
508	9	Wireless_Node_9	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Standard	STANDARD	IEEE802.11g
510	9	Wireless_Node_9	Interface_1 (Wireless)	Physical Layer	IEEE802.11	_NA_	Transmitter Power (mW)	TX_POWER	20

Figure 3-90: Modified property values of Standard and Transmit Power

Once users make changes to the Excel sheet, they can simply save it and then click "Import Properties from Excel" in Rapid Device Configurator window. This action will automatically update the Transmitter Power and Standard settings for all the selected devices.

3.11 Configuring Link Properties using the import-from/export-to Excel option

We will now explore a simple example of how users can configure link properties with the export to Excel option.

Step 1: Right-click on Wireless link and select Open Properties as New Window, which will open the link Properties window. Here, users can choose the **Export Properties to Excel** option.

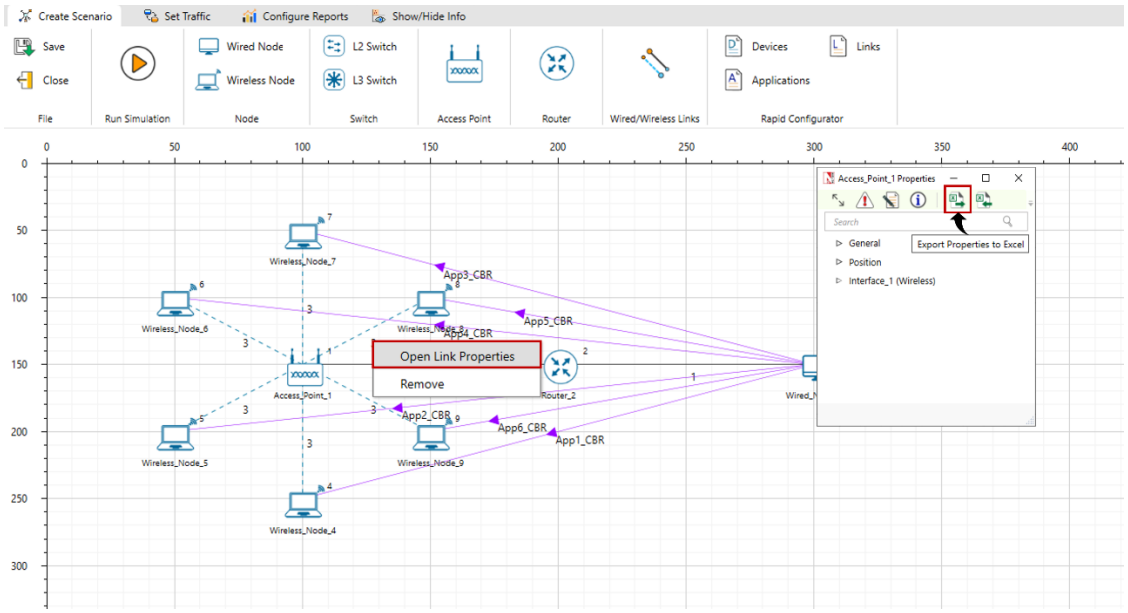
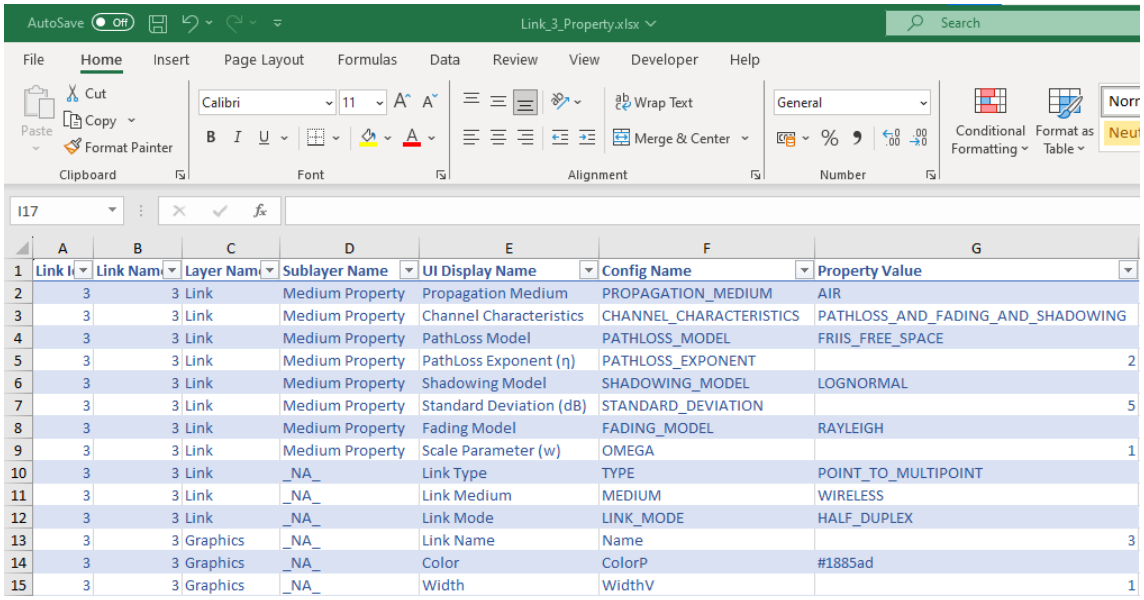


Figure 3-91: Right click on the link to open properties.

Step 2: This will open an Excel sheet where users can view all link properties associated with the wireless link.



	A	B	C	D	E	F	G
1	Link I	Link Nam	Layer Nam	Sublayer Name	UI Display Name	Config Name	Property Value
2	3	3	Link	Medium Property	Propagation Medium	PROPAGATION_MEDIUM	AIR
3	3	3	Link	Medium Property	Channel Characteristics	CHANNEL_CHARACTERISTICS	PATHLOSS_AND_FADING_AND_SHADOWING
4	3	3	Link	Medium Property	PathLoss Model	PATHLOSS_MODEL	FRIIS_FREE_SPACE
5	3	3	Link	Medium Property	PathLoss Exponent (η)	PATHLOSS_EXPONENT	2
6	3	3	Link	Medium Property	Shadowing Model	SHADOWING_MODEL	LOGNORMAL
7	3	3	Link	Medium Property	Standard Deviation (dB)	STANDARD_DEVIATION	5
8	3	3	Link	Medium Property	Fading Model	FADING_MODEL	RAYLEIGH
9	3	3	Link	Medium Property	Scale Parameter (w)	OMEGA	1
10	3	3	Link	_NA_	Link Type	TYPE	POINT_TO_MULTIPOINT
11	3	3	Link	_NA_	Link Medium	MEDIUM	WIRELESS
12	3	3	Link	_NA_	Link Mode	LINK_MODE	HALF_DUPLEX
13	3	3	Graphics	_NA_	Link Name	Name	3
14	3	3	Graphics	_NA_	Color	ColorP	#1885ad
15	3	3	Graphics	_NA_	Width	WidthV	1

Figure 3-92: Properties of wireless link

In this example, users will learn how to modify Channel characteristics, Pathloss Exponent, and Pathloss model using an Excel sheet.

Step 3: Select UI Display Name and filter Channel characteristics, Pathloss Exponent, and Pathloss model in the property value tab.

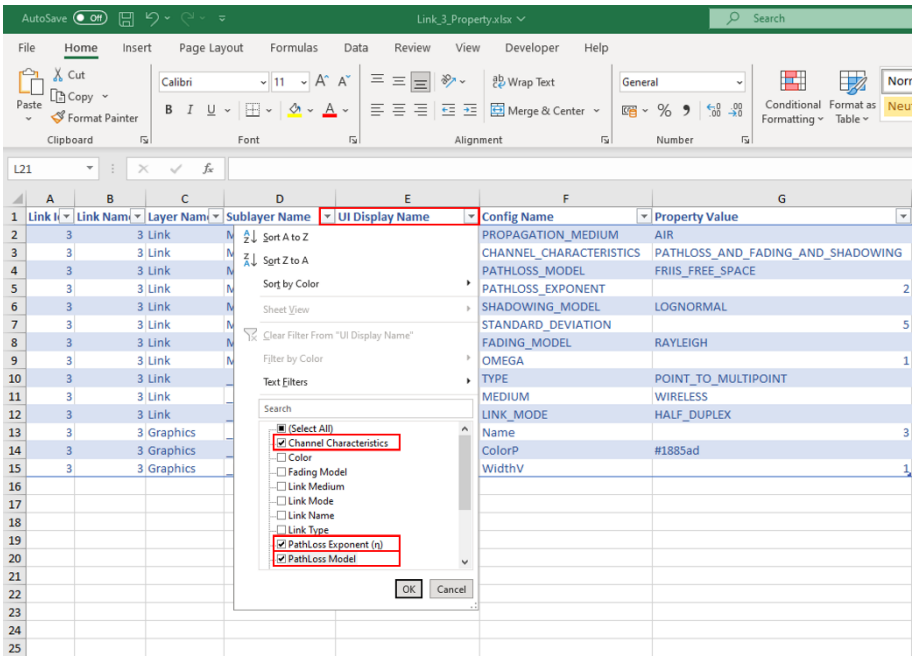


Figure 3-93: Filter Channel characteristics, Pathloss Exponent, and Pathloss model

Step 4: After applying filters to Channel characteristics, Pathloss Exponent, and Pathloss model, proceed to the **property value** tab to make the desired changes.

	A	B	C	D	E	F	G
1	Link Id	Link Nam	Layer Nam	Sublayer Name	UI Display Name	Config Name	Property Value
3	3	3	Link	Medium Property	Channel Characteristics	CHANNEL_CHARACTERISTICS	PATHLOSS_AND_FADING_AND_SHADOWING
4	3	3	Link	Medium Property	PathLoss Model	PATHLOSS_MODEL	FRIIS_FREE_SPACE
5	3	3	Link	Medium Property	PathLoss Exponent (n)	PATHLOSS_EXPONENT	2

Figure 3-94: Values of Channel Characteristics, Pathloss Exponent and Pathloss model before modifying.

Step 5: In the Property Value tab, set the following values:

Channel Characteristics: Pathloss Only

Pathloss Exponent: 3

Pathloss Model: LOG_DISTANCE

	A	B	C	D	E	F	G
1	Link Id	Link Nam	Layer Nam	Sublayer Name	UI Display Name	Config Name	Property Value
3	3	3	Link	Medium Property	Channel Characteristics	CHANNEL_CHARACTERISTICS	PATHLOSS_ONLY
4	3	3	Link	Medium Property	PathLoss Model	PATHLOSS_MODEL	LOG_DISTANCE
5	3	3	Link	Medium Property	PathLoss Exponent (n)	PATHLOSS_EXPONENT	3

Figure 3-95: Values of Channel Characteristics, Pathloss Exponent and Pathloss model after modifying.

Step 6: Once the Excel file is saved, click on the **"Import to Excel"** option in the Link Properties Window.

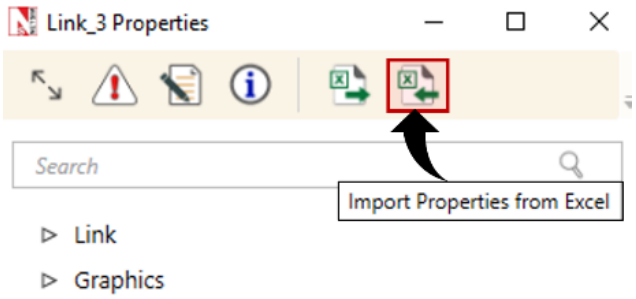


Figure 3-96: Import to excel option in GUI.

Step 7: This will direct users to the folder where **Link_3_Property.xlsx** is saved. Click on the file for it to be imported.

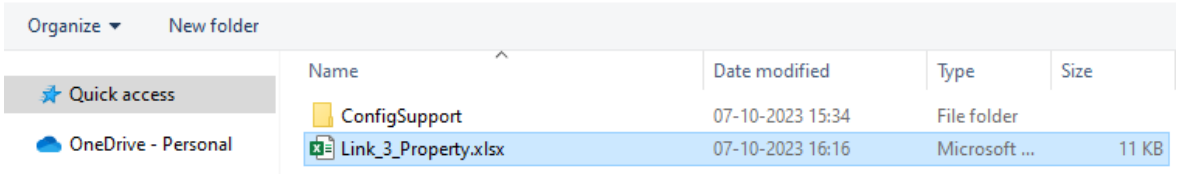


Figure 3-97: Link_3_Property.xlsx file saved after exporting.

Step 8: Upon successful import, right click on Link_3 Properties and expand the Link>Medium Property, where users will observe the updated Channel characteristics, Pathloss Exponent, and Pathloss models.

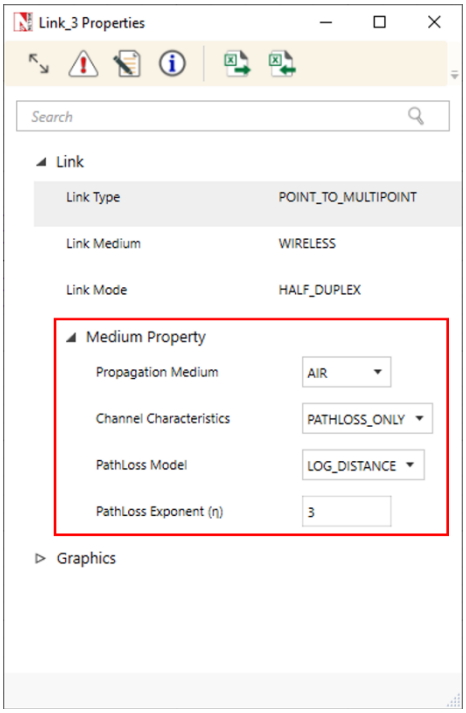


Figure 3-98: Modified Channel Characteristics, Pathloss Exponent and Pathloss model values.

Also, In the Rapid Link Configurator window, users have the option to select multiple links and modify all link properties simultaneously.

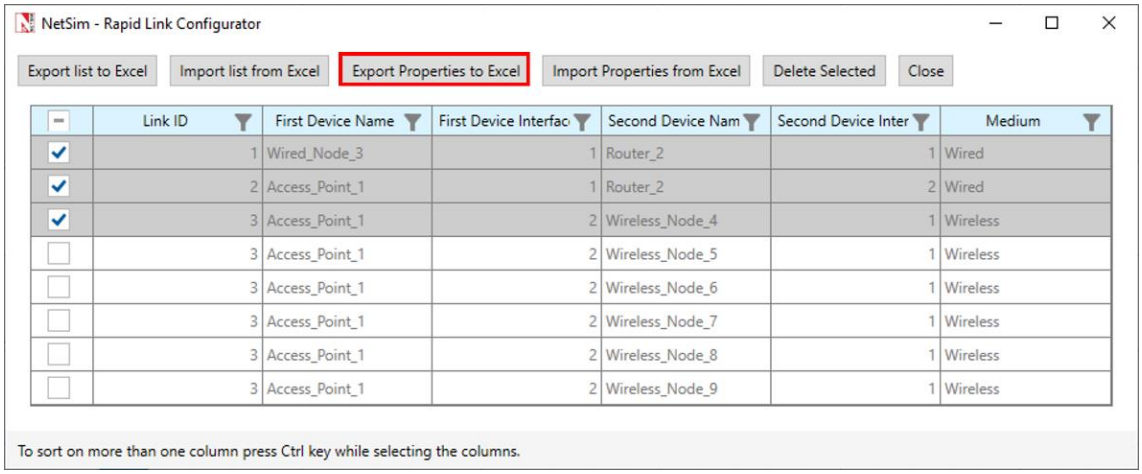


Figure 3-99: Selecting and Exporting multiple links using Export Properties to Excel option.

After filtering UI Display name to Max uplink speed and width, users have set property values to 200 Mbps and 2 m for all the devices.

	A	B	C	D	E	F	
1	Link Id	Link Nam	Layer Nam	Sublayer Name	UI Display Name	Config Name	Property Value
9	1	1	Link	_NA_	Max Uplink Speed (Mbps)	LINK_SPEED_UP	200
13	1	1	Graphics	_NA_	Width	WidthV	2
21	2	2	Link	_NA_	Max Uplink Speed (Mbps)	LINK_SPEED_UP	200
25	2	2	Graphics	_NA_	Width	WidthV	2
39	3	3	Graphics	_NA_	Width	WidthV	2

Figure 3-100: Modified values of Max Uplink Speed and Width

Once changes are made in excel sheet just save it, and the Max Uplink Speed and width will be updated to 200 Mbps and 2m for all the links.

3.12 Configuring Application Properties using the import-from/export-to Excel option

We will now explore a simple example of how users can configure link properties with the export to Excel option.

Step 1: Right-click on App1_CBR and select "**Open Application Properties**." This will open the Application Properties window, where users can choose the "**Export properties to Excel**" option.

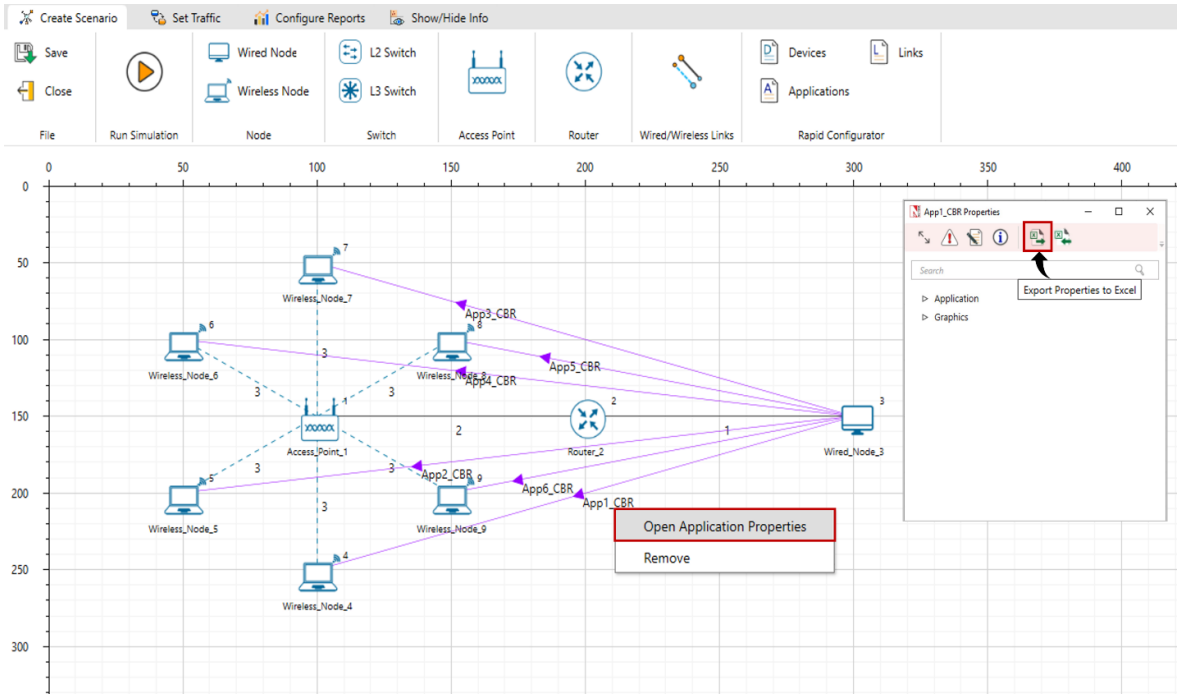
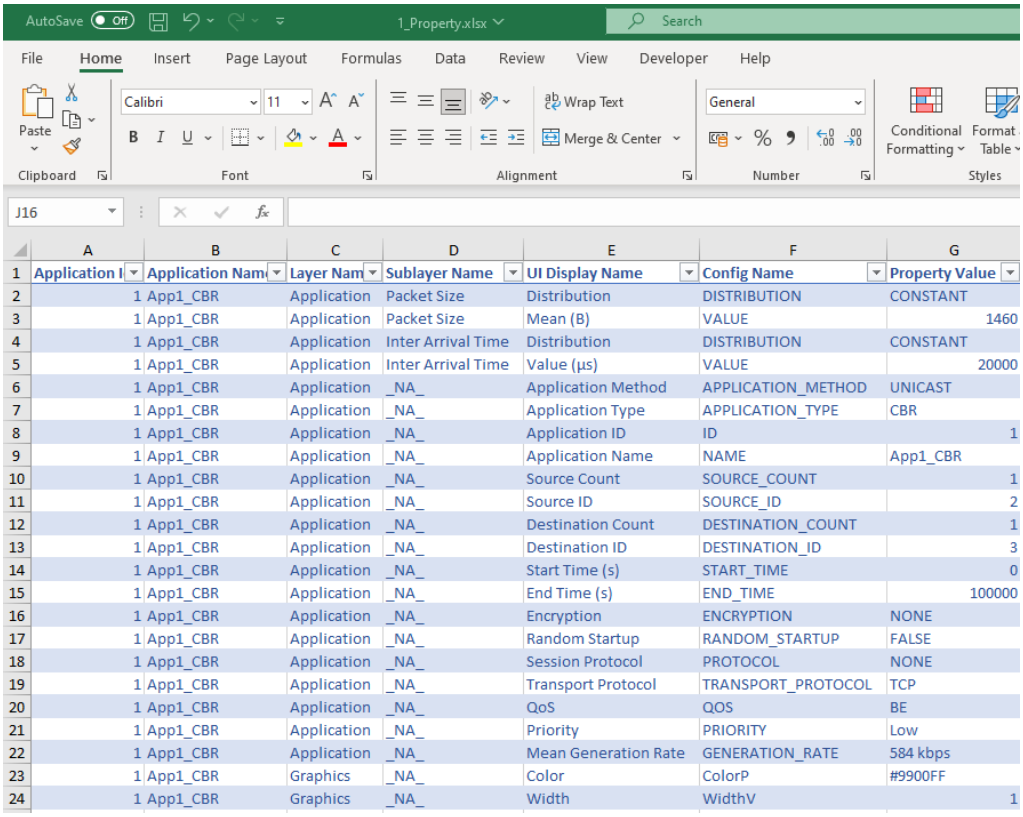


Figure 3-101: Right click on the application to open properties.

Step 2: This will open an **Excel sheet** where users can view all the application properties related to the application that was selected.



	A	B	C	D	E	F	G
1	Application I	Application Nam	Layer Nam	Sublayer Name	UI Display Name	Config Name	Property Value
2	1	App1_CBR	Application	Packet Size	Distribution	DISTRIBUTION	CONSTANT
3	1	App1_CBR	Application	Packet Size	Mean (B)	VALUE	1460
4	1	App1_CBR	Application	Inter Arrival Time	Distribution	DISTRIBUTION	CONSTANT
5	1	App1_CBR	Application	Inter Arrival Time	Value (µs)	VALUE	20000
6	1	App1_CBR	Application	_NA_	Application Method	APPLICATION_METHOD	UNICAST
7	1	App1_CBR	Application	_NA_	Application Type	APPLICATION_TYPE	CBR
8	1	App1_CBR	Application	_NA_	Application ID	ID	1
9	1	App1_CBR	Application	_NA_	Application Name	NAME	App1_CBR
10	1	App1_CBR	Application	_NA_	Source Count	SOURCE_COUNT	1
11	1	App1_CBR	Application	_NA_	Source ID	SOURCE_ID	2
12	1	App1_CBR	Application	_NA_	Destination Count	DESTINATION_COUNT	1
13	1	App1_CBR	Application	_NA_	Destination ID	DESTINATION_ID	3
14	1	App1_CBR	Application	_NA_	Start Time (s)	START_TIME	0
15	1	App1_CBR	Application	_NA_	End Time (s)	END_TIME	100000
16	1	App1_CBR	Application	_NA_	Encryption	ENCRYPTION	NONE
17	1	App1_CBR	Application	_NA_	Random Startup	RANDOM_STARTUP	FALSE
18	1	App1_CBR	Application	_NA_	Session Protocol	PROTOCOL	NONE
19	1	App1_CBR	Application	_NA_	Transport Protocol	TRANSPORT_PROTOCOL	TCP
20	1	App1_CBR	Application	_NA_	QoS	QOS	BE
21	1	App1_CBR	Application	_NA_	Priority	PRIORITY	Low
22	1	App1_CBR	Application	_NA_	Mean Generation Rate	GENERATION_RATE	584 kbps
23	1	App1_CBR	Graphics	_NA_	Color	ColorP	#9900FF
24	1	App1_CBR	Graphics	_NA_	Width	WidthV	1

Figure 3-102: Application Properties

In this example, users can learn how to modify the Start Time and Transport Protocol using the Excel sheet.

Step 3: Select UI Display Name, filter Start Time and Transport Protocol, and in the property value tab, update the Start time to 20 and Transport Protocol to UDP.

1	Application I	Application Nam	Layer Nam	Sublayer Name	UI Display Name	Config Name	Property Valu
14	1	App1_CBR	Application	_NA_	Start Time (s)	START_TIME	20
19	1	App1_CBR	Application	_NA_	Transport Protocol	TRANSPORT_PROTOCOL	UDP

Figure 3-103: Values of Start Time and Transport protocol after modifying.

Step 4: Once the Excel file is saved, click on "**Import to Excel**" option in the Properties Window.

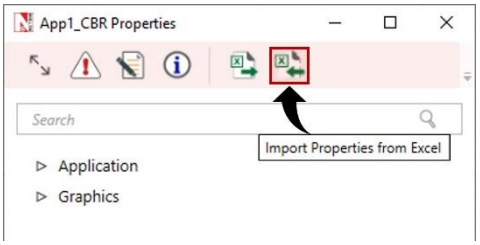


Figure 3-104: Import Properties from Excel option in GUI.

Step 5: This will direct users to the folder where **APP1_CBR_Property.xlsx** is saved. Click on the file for it to be imported.

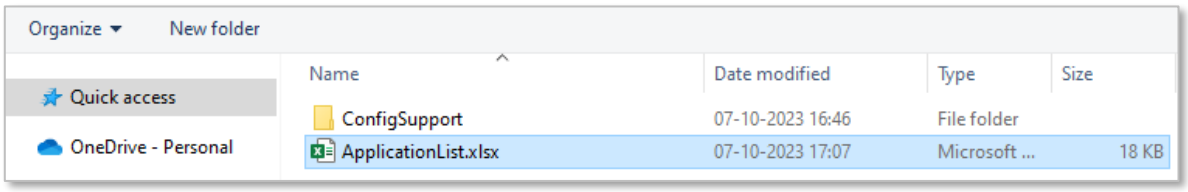


Figure 3-105: ApplicationList.xlsx file saved after exporting.

Step 6: Upon successful import, right click on App1_CBR properties and expand the application where users can see Start Time and Transport Protocol is changed to 20 and UDP by default it will be 0 and TCP.

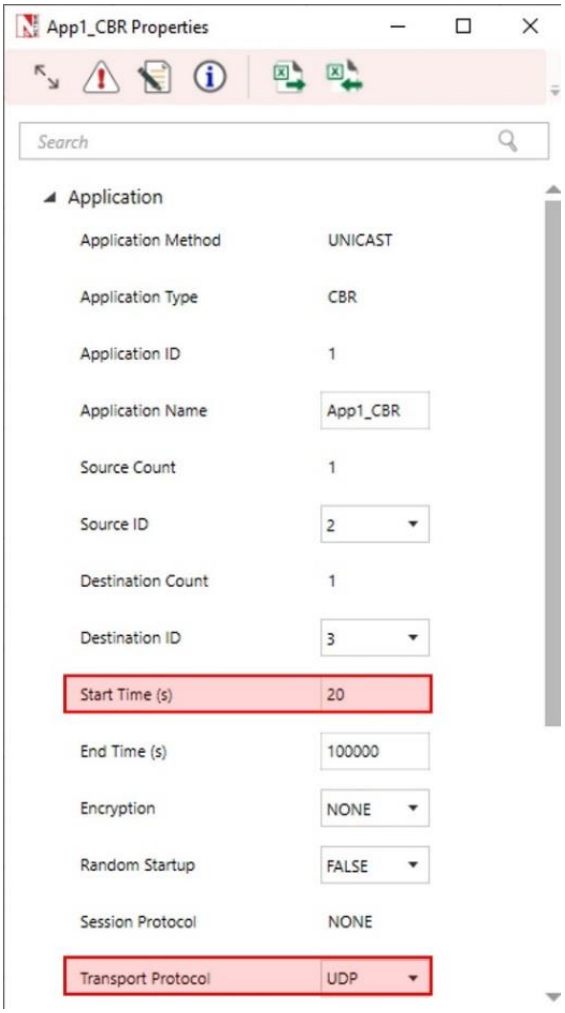


Figure 3-106: Modified Start time and Transport Protocol values.

3.13 NetSim Keyboard Shortcuts

NetSim keyboard shortcuts can be used for frequently performed tasks. The keyboard shortcuts that are currently supported are listed in the table below Table 3-1

Keys	Functions
Zoom -Grid	
Ctrl + / Ctrl -	Zoom in / Zoom out
Up / Down arrow	Moving grid pan
Mouse wheel	Zoom in / Zoom out
Devices	
ESC	Remove focus of the selected device
Global - keys	
Ctrl + S	Save window or save directly
Ctrl + Shift + S	Save As window
Alt + F4	Close design window
F1	Open User manual
Ctrl + R	Run simulation
Ctrl + Z	Undo
Ctrl + Y	Redo
Property panel (Devices / Link / Application / Grid / Log)	
Enter key	Update the current value
Home screen	
Ctrl + N	Opens / Switch to New Simulation tab
Ctrl + O	Opens / Switch to Your Work tab
Alt+ F4	Close Home window
Experiment list	
Shift + Up / Down arrow	For multiple experiment selection and un selection
Delete	Deleting of selected single / multiple experiments
Ctrl + A →	Select all the experiments from list
Simulation Console	
Ctrl + C	Terminates Simulation in Mid way. Results will be calculated till the time at which the simulation is terminated

Table 3-2: NetSim keyboard shortcuts

3.14 NetSim Interactive Simulation

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.

This section will demonstrate how to perform Interactive simulation for a simple network scenario. Let us consider Internetworks. To create a new scenario, go to New Simulation → Internetworks. Click & drop Wired Nodes and Router onto the grid and link them as shown below Figure 3-107.

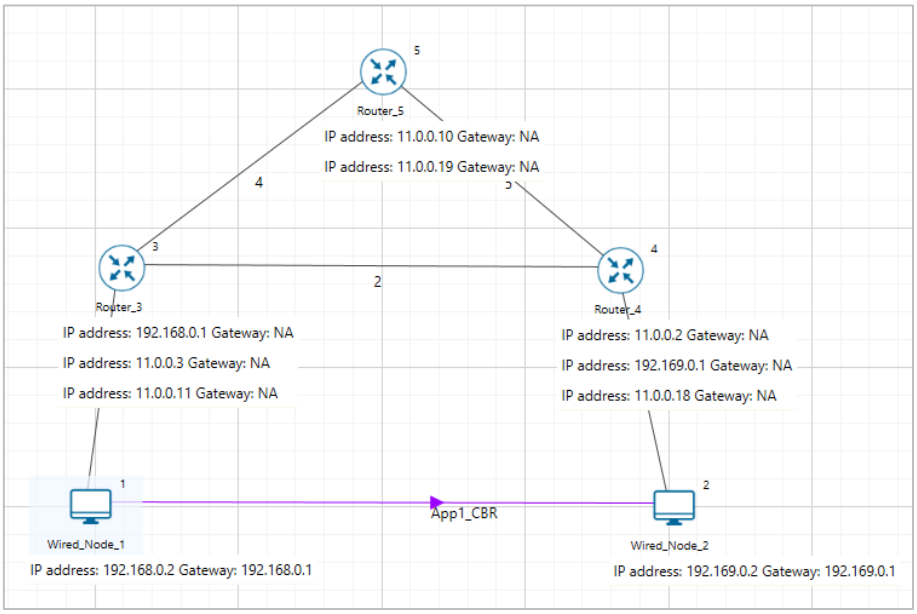


Figure 3-107: Network Topology

- Click on the Application in the ribbon at the top under set traffic tab and configure CBR application from the Source Id 1 to Destination Id 2.
- Set Start Time as 30 Sec
- Enable Plots and Packet trace options
- Before simulating, go to network scenario and right click on Router_3 or any other node and select NetSim Console option as shown below.

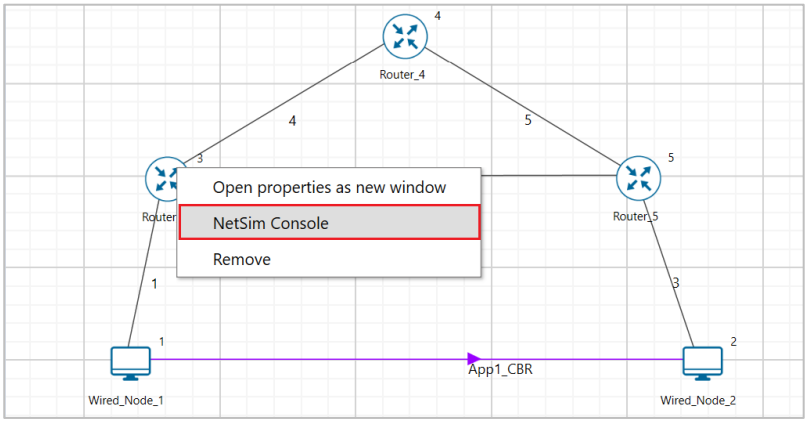


Figure 3-108: Open NetSim console window

- Now client (NetSimCLI.exe) will start running and it will wait to establish connection with NetSimCore.exe. After connection is established, the window will look similar like this shown below

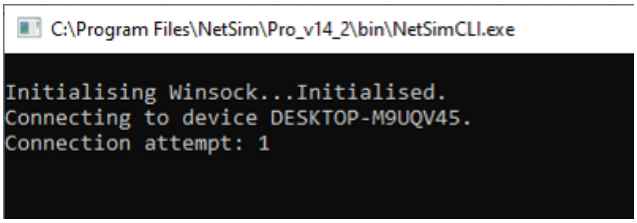


Figure 3-109: Connection is established.

- Click on **Options** tab and select Run Time Interaction option Figure 3-110.

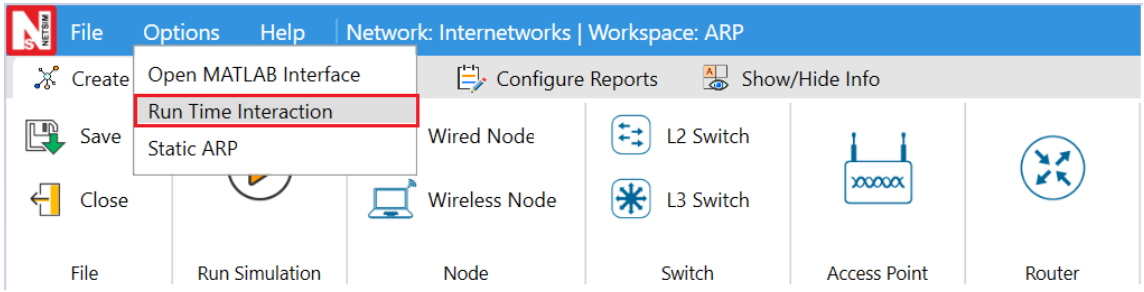


Figure 3-110: Run time Interaction tab set Interactive Simulation as True

- In the Run time Interaction tab, Interactive Simulation option is set to **True** and click on **OK.**

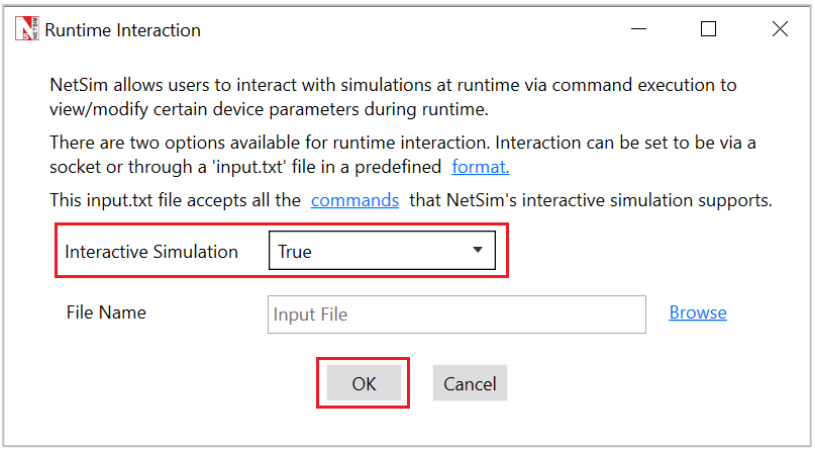


Figure 3-111: Runtime Interaction window

- Click on run simulation and set Simulation Time as 500 sec. (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 Simulation ends) and click Run.
- Simulation (NetSimCore.exe) will start running and will display a message “waiting for first client to connect” as shown below Figure 3-112.

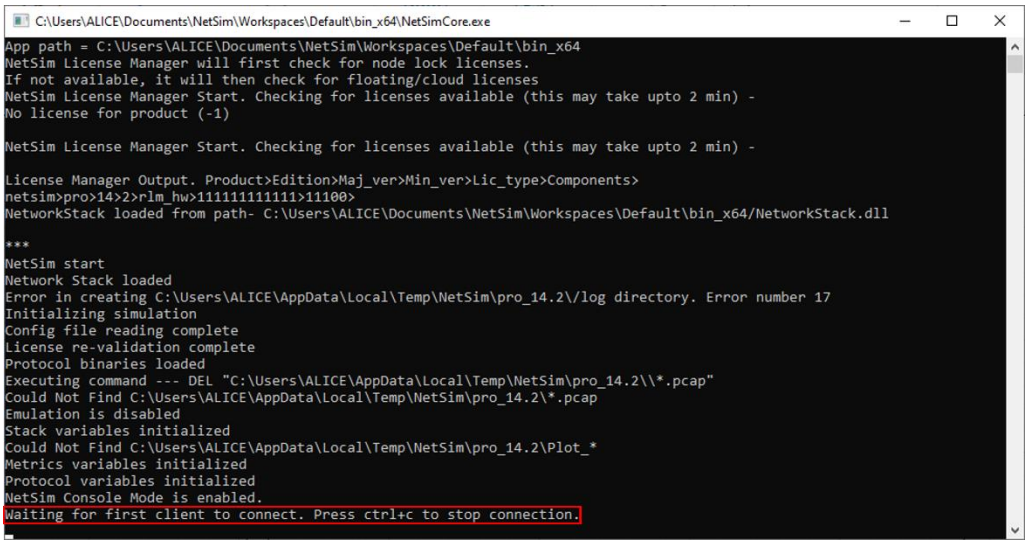


Figure 3-112: Waiting for first client to connect.

- After this, the command line interface can be used to execute the supported commands as shown below. The Supported commands are explained in the below sections.

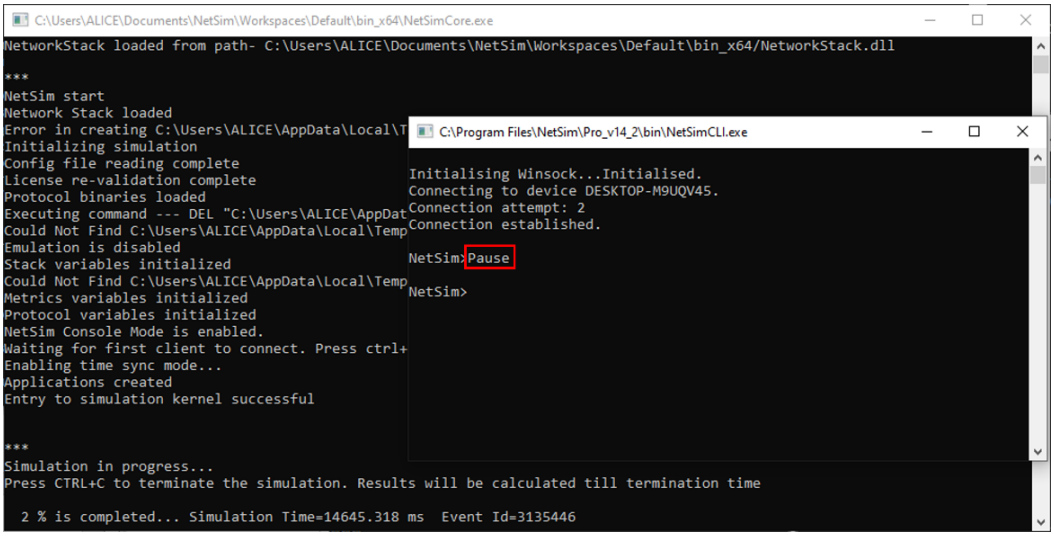


Figure 3-113: Executing NetSim CLI commands.

Note: Commands are not case sensitive

3.14.1 Simulation specific (Not applicable for file based interactive simulation)

- Pause
- Pause At
- Continue
- Stop
- Exit
- Reconnect

Pause: To pause the currently running simulation

PauseAt: To pause the currently running simulation with respect to particular time (Ex: To Pause simulation at 70.2 sec use command as **PauseAt 70.2**)

Continue: To start the currently paused simulation. When a user pauses simulation and then continues using the *pause/continue* commands, it may appear as if simulation was running in the background. This is not true. When interactive simulation is run, the simulation clock in NetSim is Min (Wall Clock, Simulation Clock). Thus, before pausing simulation may have been running at Wall clock (real time) speed even though the simulation could have run much faster. On typing the *continue* command simulation will run at it usual (much faster) speed till it equals Wall clock (Real time). This behaviour sometimes can be confusing to users.

Stop: To stop the currently running simulation (NetSimCore.exe)

Exit: To exit from the client (NetSimCLI.exe)

Reconnect: To reconnect client (NetSimCLI.exe) to simulation (NetSimCore.exe) when we rerun simulation again

3.14.2 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 Network, if you don't then the device is faulty, disconnected, switched off, incorrectly configured.
- You can use the ***ping*** cmd with an IP address or Device name.
- **ICMP_Status** should be set as True in all nodes(Wired_Node and Router)

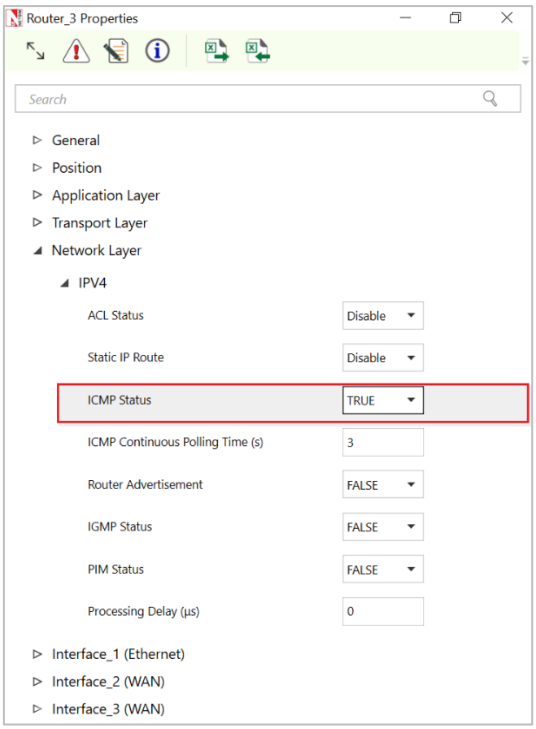


Figure 3-114: Set ICMP_Status to True in Network layer window.

- Click on Wired_Node_1 and expand right hand side properties . Under General properties enable Wireshark Capture option as "Online"

Ping <IP address> e.g. ping 192.168.0.2

Ping <Node Name> e.g. ping Wired_Node_2

3.14.2.1 Ping Command Results

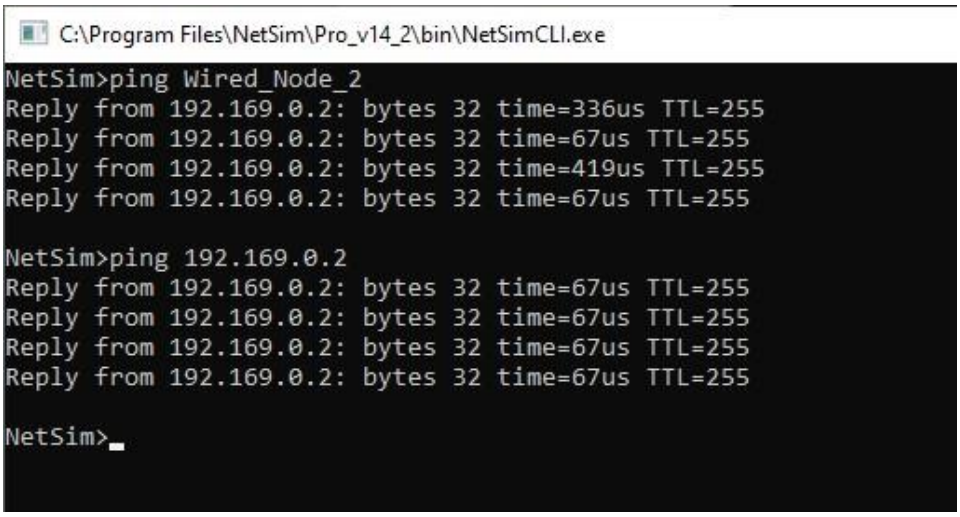
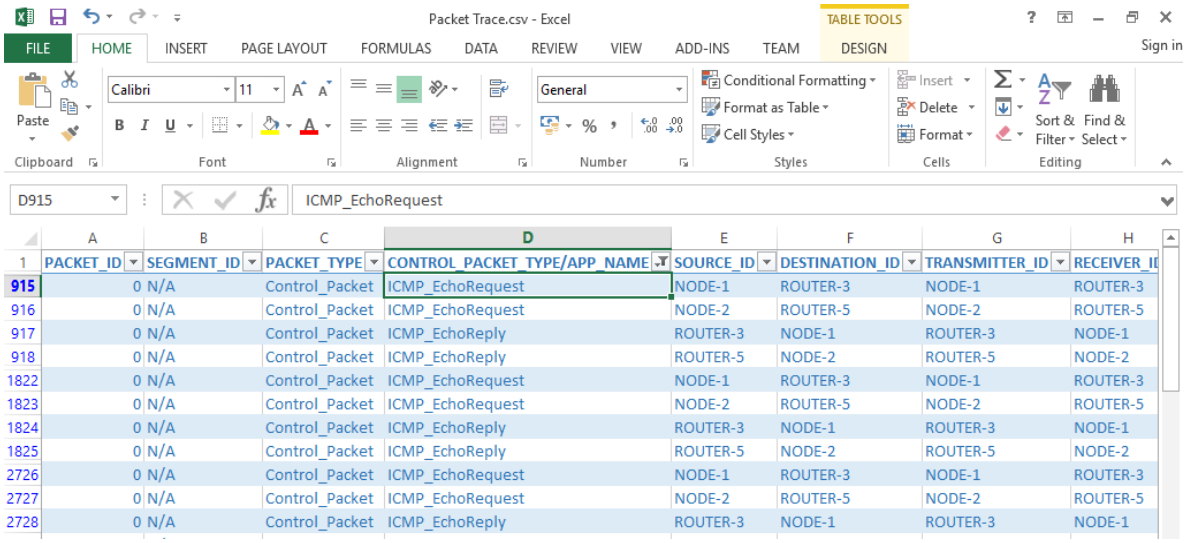


Figure 3-115: Pinging to Wired_Node_2

- After simulation open packet trace and filter ICMP_EchoRequest and ICMP_EchoReply from CONTROL_PACKET_TYPE/APP_NAME column



	A	B	C	D	E	F	G	H
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
915	0	N/A	Control_Packet	ICMP_EchoRequest	NODE-1	ROUTER-3	NODE-1	ROUTER-3
916	0	N/A	Control_Packet	ICMP_EchoRequest	NODE-2	ROUTER-5	NODE-2	ROUTER-5
917	0	N/A	Control_Packet	ICMP_EchoReply	ROUTER-3	NODE-1	ROUTER-3	NODE-1
918	0	N/A	Control_Packet	ICMP_EchoReply	ROUTER-5	NODE-2	ROUTER-5	NODE-2
1822	0	N/A	Control_Packet	ICMP_EchoRequest	NODE-1	ROUTER-3	NODE-1	ROUTER-3
1823	0	N/A	Control_Packet	ICMP_EchoRequest	NODE-2	ROUTER-5	NODE-2	ROUTER-5
1824	0	N/A	Control_Packet	ICMP_EchoReply	ROUTER-3	NODE-1	ROUTER-3	NODE-1
1825	0	N/A	Control_Packet	ICMP_EchoReply	ROUTER-5	NODE-2	ROUTER-5	NODE-2
2726	0	N/A	Control_Packet	ICMP_EchoRequest	NODE-1	ROUTER-3	NODE-1	ROUTER-3
2727	0	N/A	Control_Packet	ICMP_EchoRequest	NODE-2	ROUTER-5	NODE-2	ROUTER-5
2728	0	N/A	Control_Packet	ICMP_EchoReply	ROUTER-3	NODE-1	ROUTER-3	NODE-1

Figure 3-116: ICMP Control Packets in Packet Trace

- Open Wireshark and apply filter ICMP. We can see the ping request and reply packets in Wireshark.

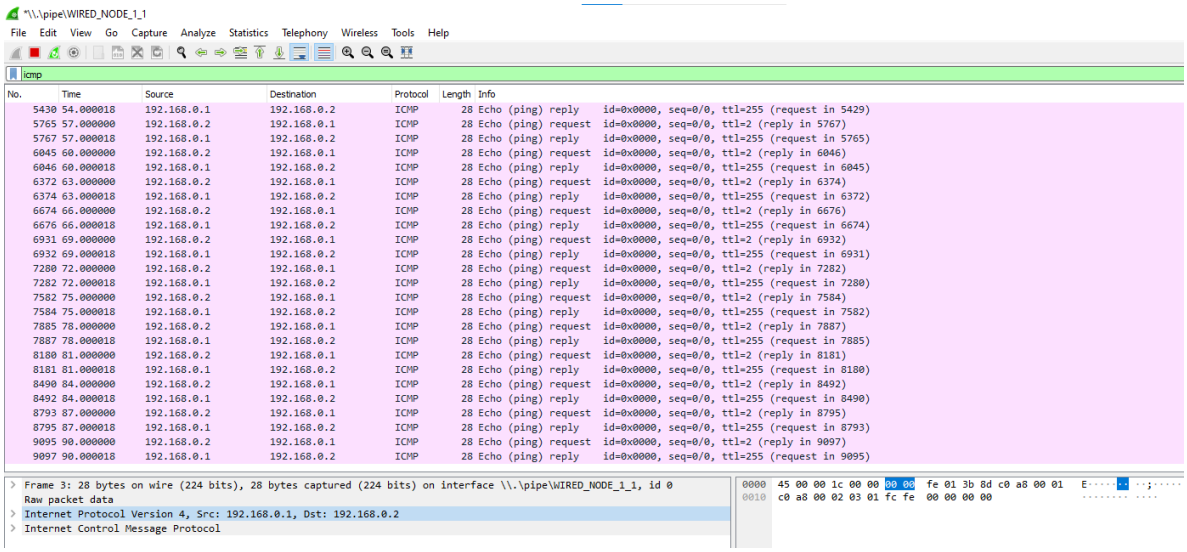


Figure 3-117: ICMP Control Packets in Wireshark

3.14.3 Route Commands

- route print
- route delete
- route add

In order to view the entire contents of the IP routing table, use the following commands **route print.**

route print

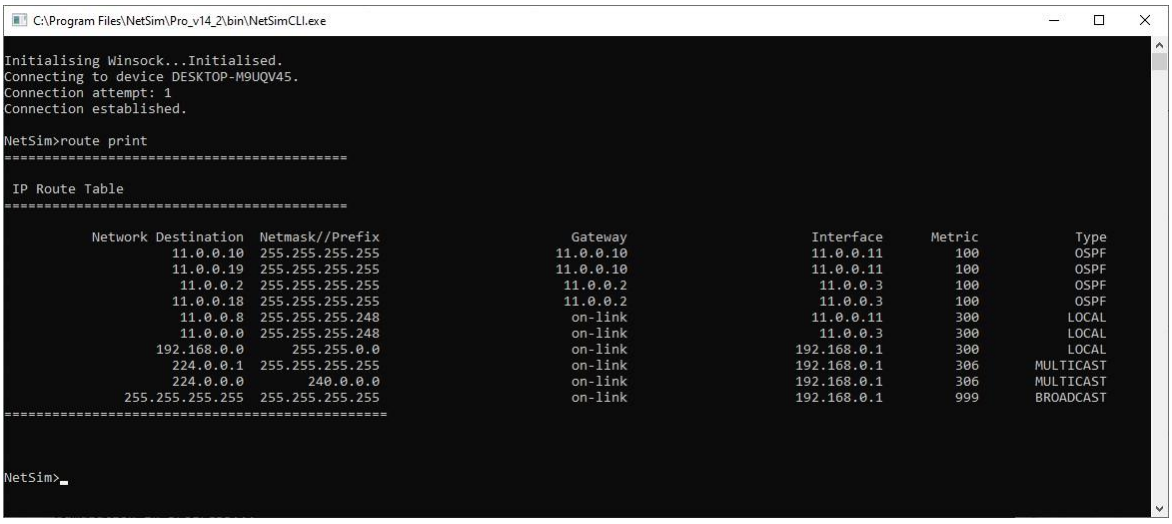


Figure 3-118: Network Route Print

- You will 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 will be dynamically generated.
- In order to delete route in the IP routing table you will type a command using the following syntax

Route delete destination_network

- So, to delete the route with destination network 11.0.0.2, all we'd have to do is type this command.

route delete 11.0.0.2

- To check whether route has been deleted or not check again using **route print** command.
- To add a static route to the table, you will type a command using the following syntax.

route ADD destination_network MASK subnet_mask gateway_ip METRIC metric_cost IF interface_id

- 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 11.0.0.3

route ADD 11.0.0.2 MASK 255.255.0.0 11.0.0.3 METRIC 1 IF 2

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

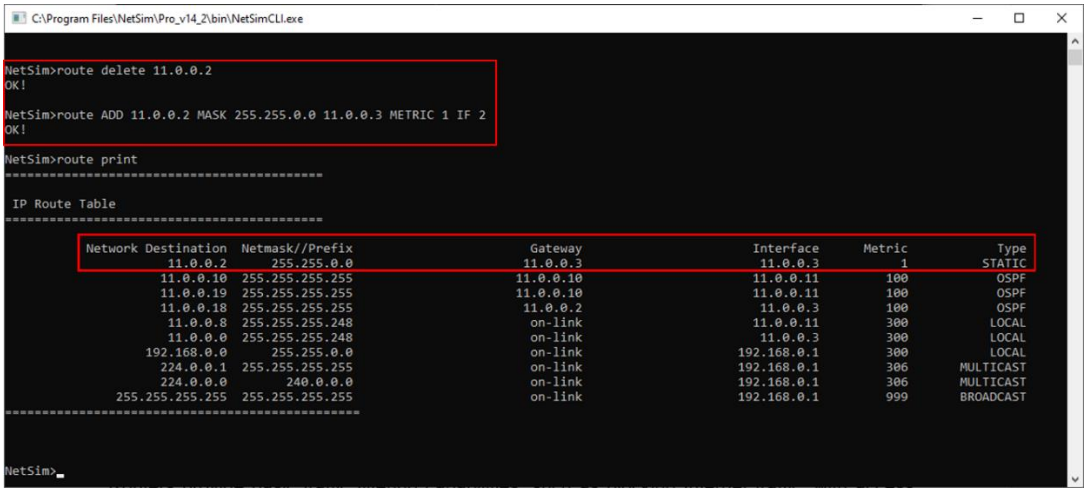


Figure 3-119: Route added into Network.

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.

3.14.4 ACL Configuration

Routers provide basic traffic filtering capabilities, such as blocking Internet traffic, with access control lists (ACLs). An ACL is a sequential list of permits 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.

ACL Commands

- To view ACL syntax use: ***acl print.***
- Before using ACL’s, we must first verify that acl option enabled. A common way to enable ACL use command: ***acl enable.***
- Enters configuration mode of ACL using: ***aclconfig***
- To view ACL Table: ***Print***
- To exit from ACL configuration use command: ***exit***
- To disable ACL use command: ***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

3.14.4.1 Step to Configure ACL

- Create Network scenario as shown in below figure.

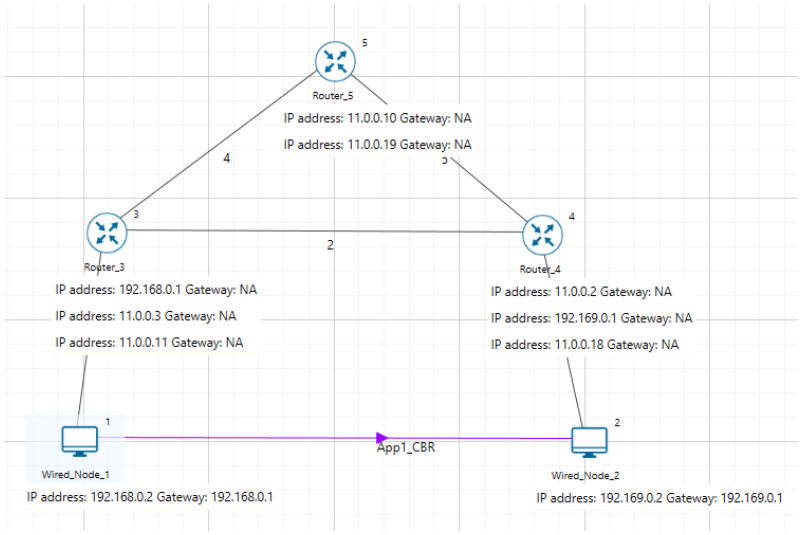


Figure 3-120: Network Scenario

- To create a new rule in the ACL use command as shown below to block UDP packet in Interface_3 of the Router_3.
- Configure an application between any two nodes by selecting CBR application from the Set Traffic tab from the top ribbon.
  - CBR Application from Wired Node 1 to Wired Node 2 with 10 Mbps Generation Rate (Packet Size: 1460, Inter Arrival Time: 1168µs).
  - Set Transport Protocol to UDP.
  - Set Start Time as 30 Sec
- Click on run simulation option and In the Run time Interaction tab set Interactive Simulation as True and click on Accept.
- Set the Simulation Time as 200sec or more. Click Ok.
- Right click on Router_3 and select NetSim Console. Use the command as follows:

*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 3***

OK!

*ROUTER_3/ACLCONFIG>**print***

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

*ROUTER_3/ACLCONFIG>**exit***

*NetSim>**acl disable***

ACL is disable

NetSim>

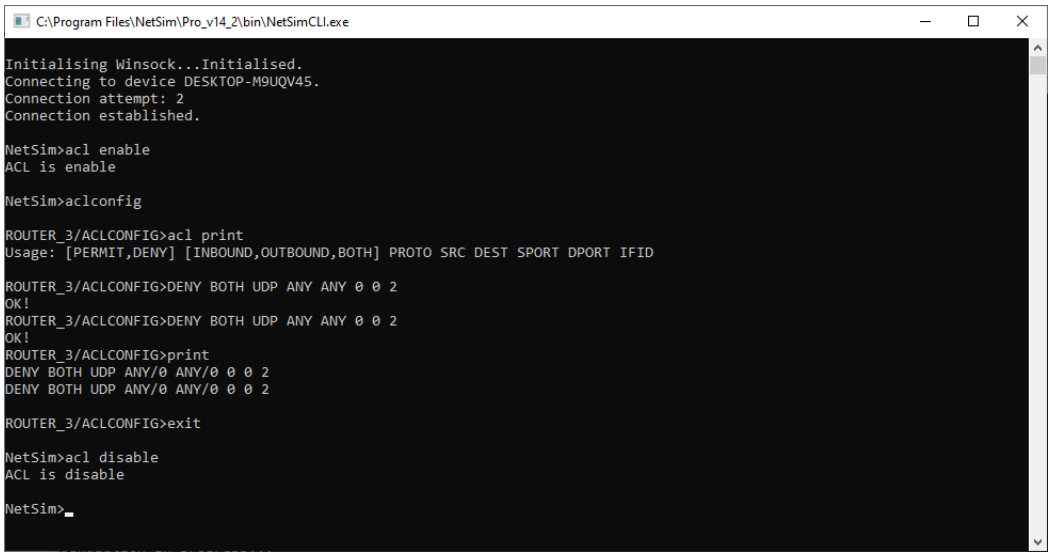


Figure 3-121: ACL Configuration command

3.14.4.2 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 number of packets blocked by firewall has been shown below.

Note: Results will vary based on time of ACL command are executed

IP_Metrics

Device Id	Packet sent	Packet forwarded	Packet received	Packet discarded	TTL expired	Firewall blocked
1	55	0	52	0	0	0
2	97156	97105	53	0	0	0
3	97278	145371	54	0	0	48147
4	145548	0	0	0	0	0
5	0	0	96987	0	0	0

Figure 3-122: IP Metrics Table in result window

The impact of ACL rule applied over the simulation traffic can be observed in the Application throughput plot. Throughput graph will show a drop after ACL is set. If ACL is disabled after a while, application packets will start flowing across the router. The Application throughput plot will show a drop and increase(Moving throughput graph) in throughput after setting ACL and disabling ACL respectively.

Example: ACL rule applied at around 97sec user can see the drop in throughput in the graph, since router blocks UDP packets in the plot. Once ACL has been disabled at around sec router permits packets and hence increase in throughput can be observed in the plot shown below.

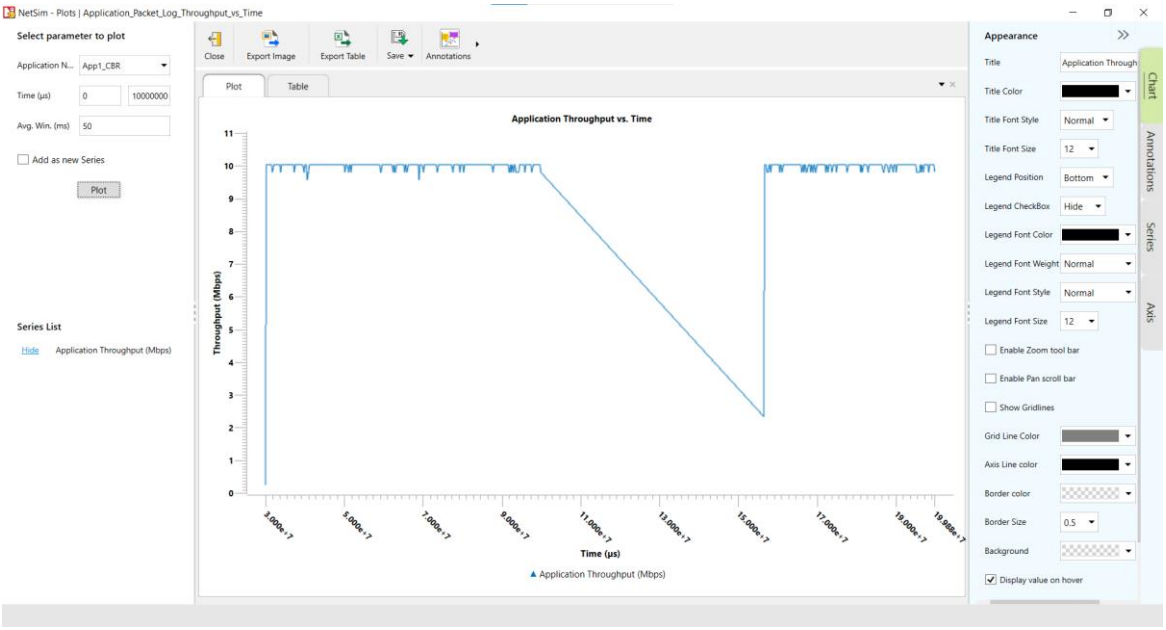


Figure 3-123: CBR Application throughput plot

3.14.5 Interactive Simulation using file

Interactive simulation using file allows users to pass commands as input through a text file. In the text file users can provide the commands along with the device in which it has to be executed by specifying the time stamps. This provides the user to have control over the scenario to execute commands at a specified time.

In the Interactive simulation text file, user should specify the exact time in seconds, along with the name of the device.

Format of the input text file for one device.

TIME=<*SIMULATION TIME IN SECONDS*>
DEVICE=<*DEVICE_NAME*>
<*COMMAND TO BE EXECUTED*>

The below Network scenario explains how to perform the Interactive simulation using file as input.

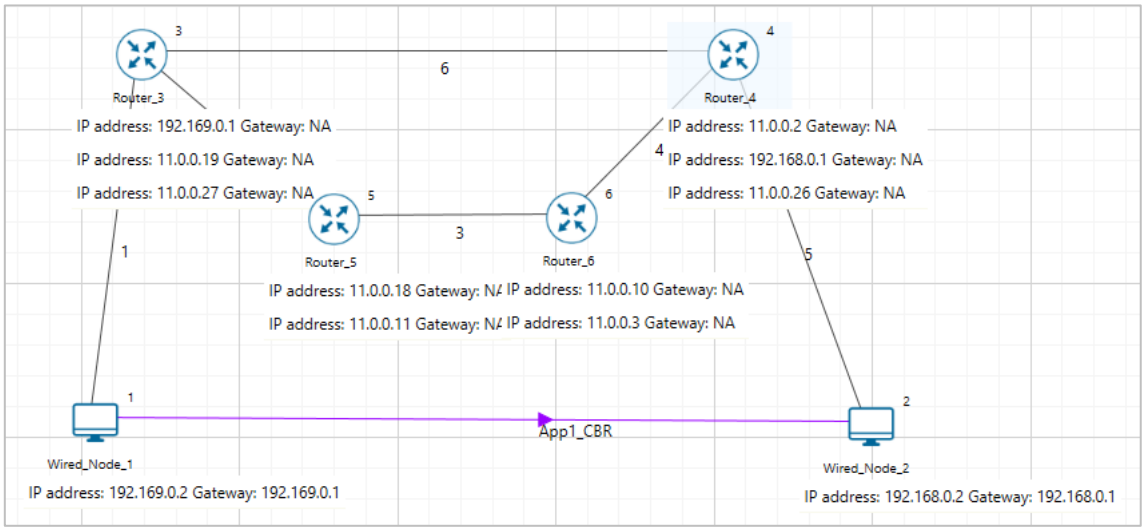


Figure 3-124: Network Topology

- The scenario comprises of 4 Routers, 2 Wired Node
- In the Router Application layer, the routing protocol is set as RIP, since it is global property, routing protocol will be set as RIP in all Routers.
- Configure an application between any two nodes by selecting CBR application from the Set Traffic tab from the top ribbon.
- 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.
- Additionally, Set start time as 1 sec.
- To set static routes to forward packets from Router 3 to the destination Wired Node via Router 5 instead of Router_4, such that the packets flow from Router 3=> Router 5=>Router 6=>Router4 =>Wired node 2 from a specified time say 25 seconds, the following input can be provided:

  TIME= 25
  DEVICE=Router_
  route ADD 192.168.0.9 MASK 255.255.255.0 11.0.0.2 METRIC 1 IF 2

- Create a text file with the above input and save it as input.txt file.
- Click on **the Options** tab and select Run Time Interaction option.

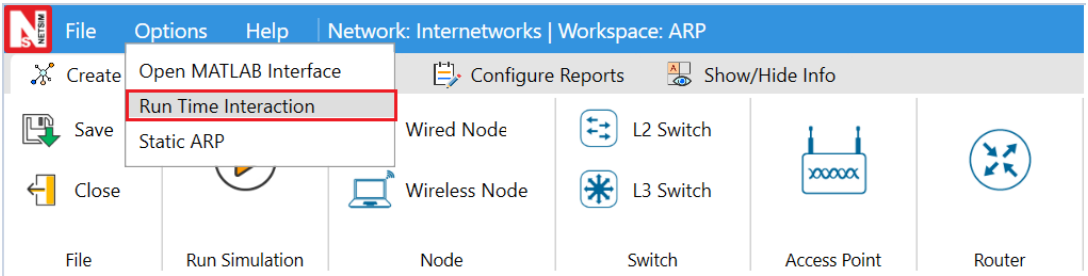


Figure 3-125: Run time Interaction tab set Interactive Simulation as True

- In the Run time Interaction tab, Interactive Simulation option is set to **True (using file)**

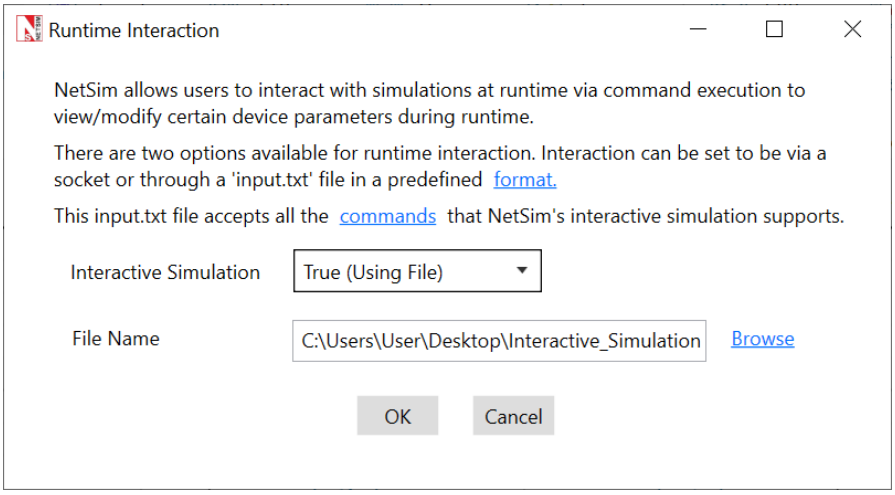


Figure 3-126: Run time Interaction tab with Interactive Simulation option set as True (Using File)

- Browse the Saved input.txt file in file path and click on ok .

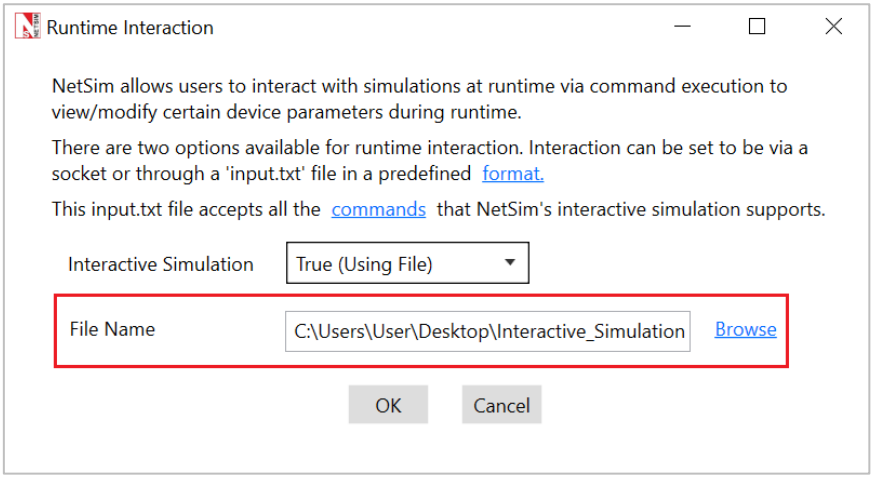


Figure 3-127: Run time Interaction tab with Interactive Simulation File Path set

- Open the packet trace, you can observe that till 25 seconds the data packets are transmitted from Wired Node 1=>Router 3=> Router 4=> Wired Node 2.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME(US)
1196	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24900000
1197	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	24920000
1197	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	24920000
1197	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24920000
1198	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	24940000
1198	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	24940000
1198	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24940000
1199	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	24960000
1199	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	24960000
1199	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24960000
1200	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	24980000
1200	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	24980000
1200	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24980000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-6	ROUTER-4	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	25000000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-6	ROUTER-4	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	25020000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25040000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25040000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25040000

Figure 3-128: Data packets are transmitting from Wired Node 1=>Router 3=> Router 4=> Wired Node 2 in packet trace before 25th seconds

- After 25th second you can observe that, the routing happens according to the command specified in the input file.
- The data (APP 1 CBR) packets are transmitted from Wired Node 1=>Router 3=>Router 5=>Router 6=>Router 4=>Wired Node 2

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)
1198	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24940000	24940000
1199	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	24960000	24960000
1199	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	24960000	24960000
1199	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24960000	24960000
1200	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	24980000	24980000
1200	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-4	24980000	24980000
1200	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	24980000	24980000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25000000	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25000000	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25000000	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-6	ROUTER-4	25000000	25000000
1201	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	25000000	25000000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25020000	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25020000	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25020000	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-6	ROUTER-4	25020000	25020000
1202	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	25020000	25020000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25040000	25040000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25040000	25040000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25040000	25040000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-6	ROUTER-4	25040000	25040000
1203	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	25040000	25040000
1204	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25060000	25060000
1204	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25060000	25060000
1204	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5	ROUTER-6	25060000	25060000
1204	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-6	ROUTER-4	25060000	25060000
1204	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4	NODE-2	25060000	25060000
1205	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	25080000	25080000
1205	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	ROUTER-5	25080000	25080000

Figure 3-129: After 25th sec routing is initiating according to the command mentioned in input file

3.14.6 Interfacing Python with NetSim

NetSim provides run-time interfacing with Python so that users who are familiar with python can implement some parts of their algorithm in Python without having to modify the C source codes of NetSim. A lot of work related to machine learning, artificial intelligence, and specialized mathematical algorithms which can be used for networking research, can be carried out using Python code.

NetSim offers Socket interfacing to interact with Python during runtime.

3.14.6.1 NetSim-Python socket interface

Connections are made using the port and IP address/Loopback address of the system running NetSim.

NetSim provides a TCP socket with which a socket program/application written in any programming language can establish a connection. The conversation can remain until the connection is terminated by either side.

By default, NetSim uses the port 8999 in which it listens for any incoming connections. Client programs can connect to the NetSim socket using the loopback/IP address of the system running NetSim along with the port number 8999.

A client socket program written in Python can connect to the NetSim Server process in a similar way as depicted here using the loopback address 127.0.0.1 and port 8999.

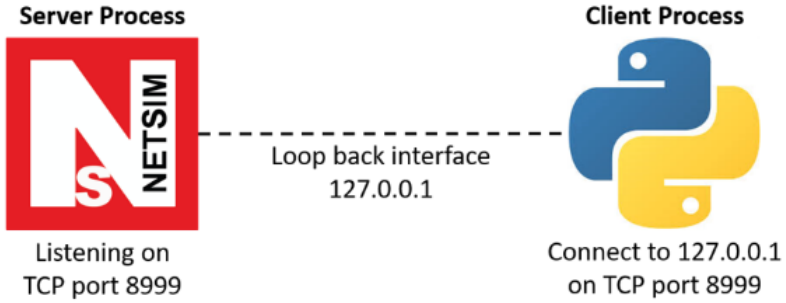


Figure 3-130: NetSim Python Interfacing

3.14.6.2 Prerequisites for Python-NetSim Interfacing

Python 3.7 or above should be installed in the same system where NetSim is installed for proper functioning of the python socket library.

3.14.6.3 Implementing a simple Netsim-python interfacing example

In this example, we will use NetSim interactive simulation to implement a simple internetworking example using python socket interfacing.

Procedure

- Create a new Internetworks (Base) Scenario in NetSim.

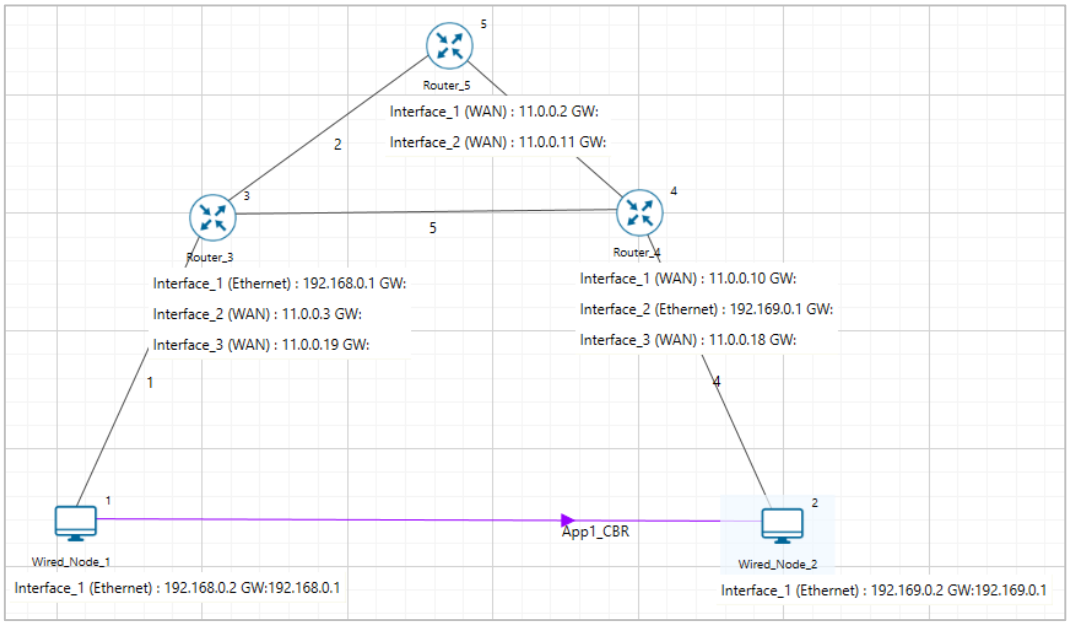


Figure 3-131: Network Topology

- Create a new python file with python code for socket interfacing.

Any specific command can be used. (Refer NetSim user manual v14.2 Section 3.5)

Here in the given code, route command has been used.

Python code

(socketdemo.py)

```
import socket # for socket
import sys
import time

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print ("Socket successfully created.")
except socket.error as err:
    print ("Socket creation failed with error %s" %(err))

    # default port for socket
port = 8999

try:
    host_ip = socket.gethostbyname('127.0.0.1')
except socket.gaierror:
    # this means could not resolve the host
    print ("Error resolving host.")
    sys.exit()

    # connecting to the server
s.connect((host_ip, port))
print ("Connection established to NetSim.")
name='Wired_Node_1'
name = name + '\0'
s.send(name.encode())
```

```
command = 'route print'
command = command + '\0'
s.send(command.encode())

resp = s.recv(1024).decode('utf-8')
cont = '__continue__'
while cont not in resp:
    resp = resp + s.recv(1024).decode('utf-8')

print ("Received:", resp)

s.close()
```

- In the Netsim runtime interaction window, set Interactive simulation to true and then click on ‘OK’ . Run the application in Run simulation window.

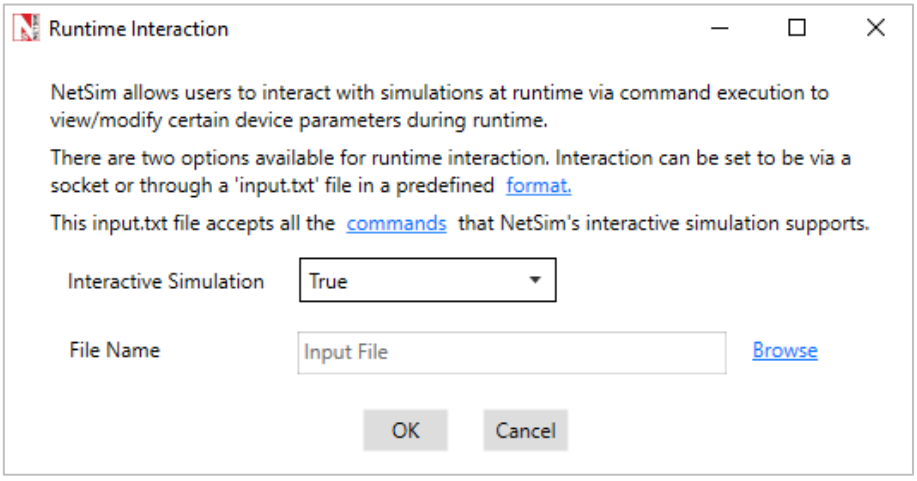


Figure 3-132: Runtime Interactive Simulation tab set Interactive Simulation to true

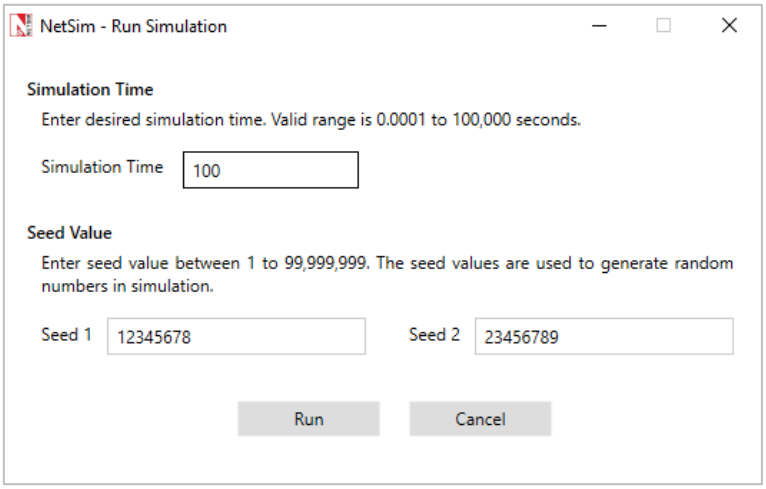


Figure 3-133: Run Simulation Configuration tab set sufficient simulation time

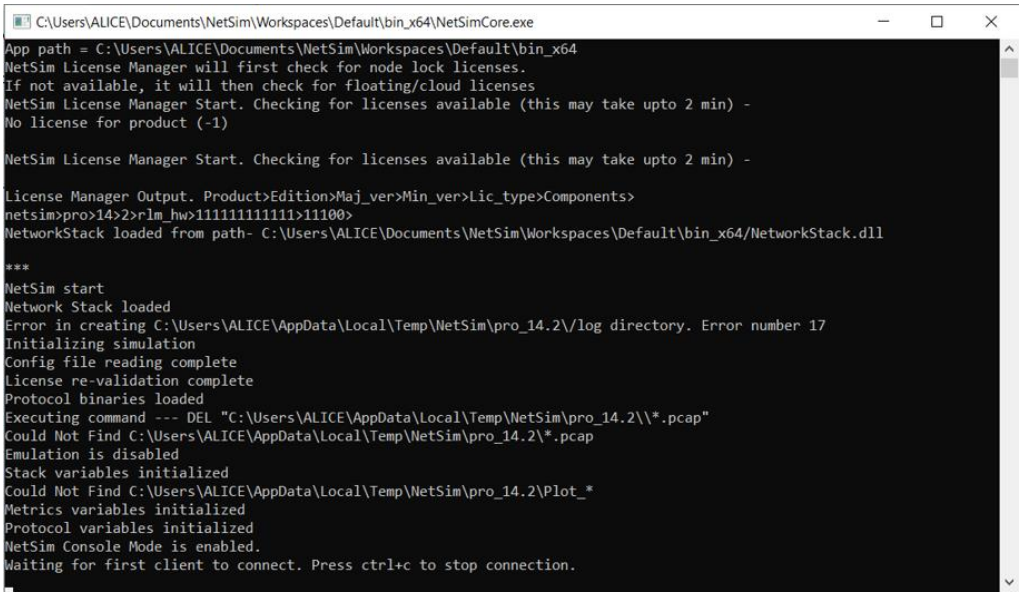


Figure 3-134: Waiting for first client to connect

- Now run the python file created ‘socketdemo.py’ using cmd.

```
C:\Users\KEN\Documents>python socketdemo.py
Socket successfully created.
Connection established to NetSim.
Received: ========================================

  IP Route Table
 ========================================

           Network Destination Netmask//Prefix                   Gateway                   Interface      Metric           Type
                     11.5.0.0      255.255.0.0                         on-link                     11.5.1.2
                300           LOCAL
                     11.2.0.0      255.255.0.0                         on-link                     11.2.1.1
                300           LOCAL
                     11.1.0.0      255.255.0.0                         on-link                     11.1.1.1
                300           LOCAL
                   224.0.0.1  255.255.255.255                         on-link                     11.1.1.1
                306      MULTICAST
                   224.0.0.0        240.0.0.0                         on-link                     11.1.1.1
                306      MULTICAST
             255.255.255.255  255.255.255.255                         on-link                     11.1.1.1
                999      BROADCAST
 ========================================


 __continue__

C:\Users\KEN\Documents>
```

Figure 3-135: Output after running the previously created python file using cmd

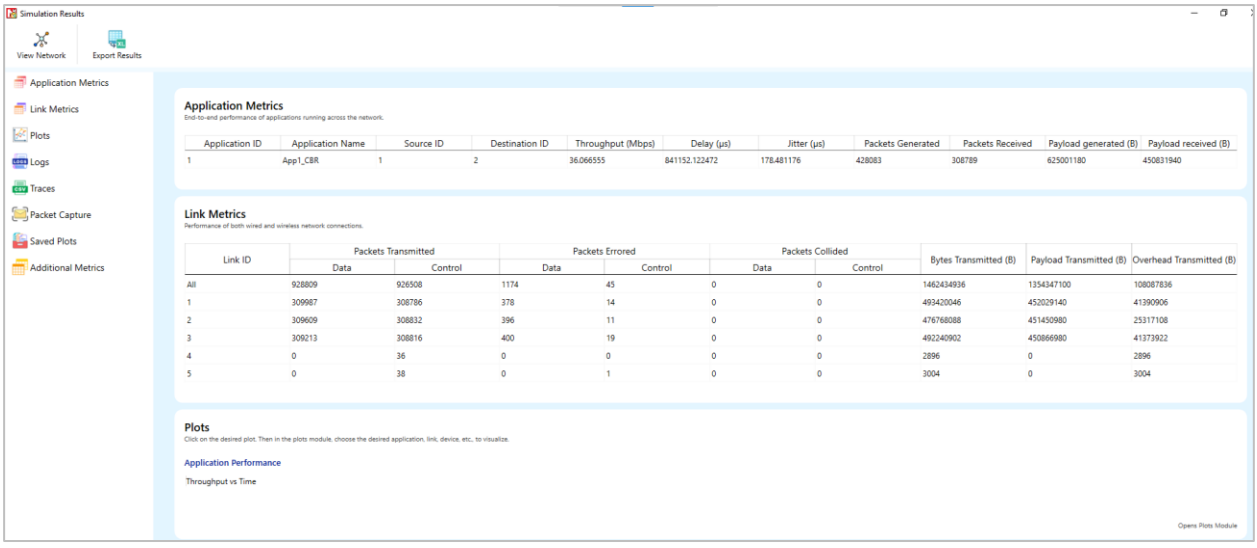


Figure 3-136: Simulation Results tab.

3.14.6.4 Reference

NetSim Python interfacing : NetSim Support Portal (tetcos.com)

3.15 Static ARP configuration in NetSim

In a network architecture, different layers have their own addressing scheme. The application layer uses hostnames, 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).

The static ARP entries are address resolutions that are manually added to the cache table for a device and are retained in the cache on a permanent basis.

Static ARP settings can be configured by selecting the Options tab in the design window and choosing Static ARP option as shown below.

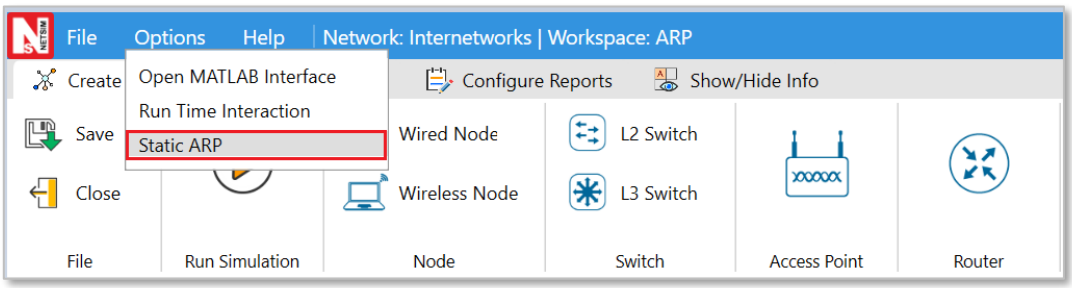


Figure 3-137: Enabling the Static ARP from NetSim UI

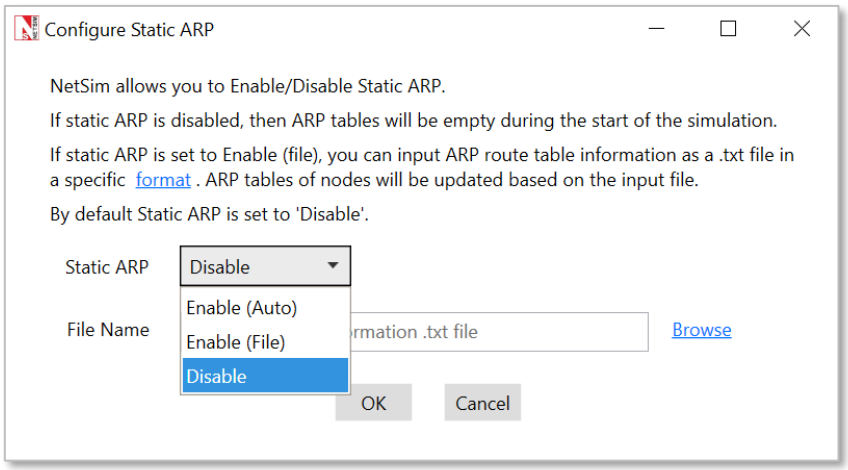


Figure 3-138: Different modes available for setting up Static ARP

Netsim supports three modes for setting up Static ARP

1. Enable (Auto)
2. Enable (Using File)

3. Disable

Enable (Auto): Static ARP enabled mode updates the ARP table before the simulation. As a result, no ARP Request or ARP Reply packets are exchanged at the start of the simulation to resolve addresses in the network. In NetSim, Static ARP mode is enabled by default.

Disable: When the Static ARP is disabled, the model allows the exchange of ARP Request and Reply packets to update the ARP table.

This can be observed in packet trace post simulation by filtering the Control packet type to **ARP Request and 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
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

Figure 3-139: ARP Route Request and Reply packets in packet trace.

Enable (Using file): This option allows the user to pass a file with ARP route table information as an input to the simulation. ARP tables of nodes will be updated based on the input file, preventing ARP messages from being exchanged between the devices.

The standard format for ARP File is as follows:

DEVICE_ID= <Device ID>, <Device ID>,

NUMBER_OF_ENTRIES=<n> // n is the no. of entries say 1,2,3, etc.

<Device IP Address><TAB><MAC ID>

<Device IP Address><TAB><MAC ID>

...

<Device IP Address><TAB><MAC ID>

Example: Consider the below scenario with wired nodes and router connected across the switch, a static ARP file for the below scenario can be written as follows

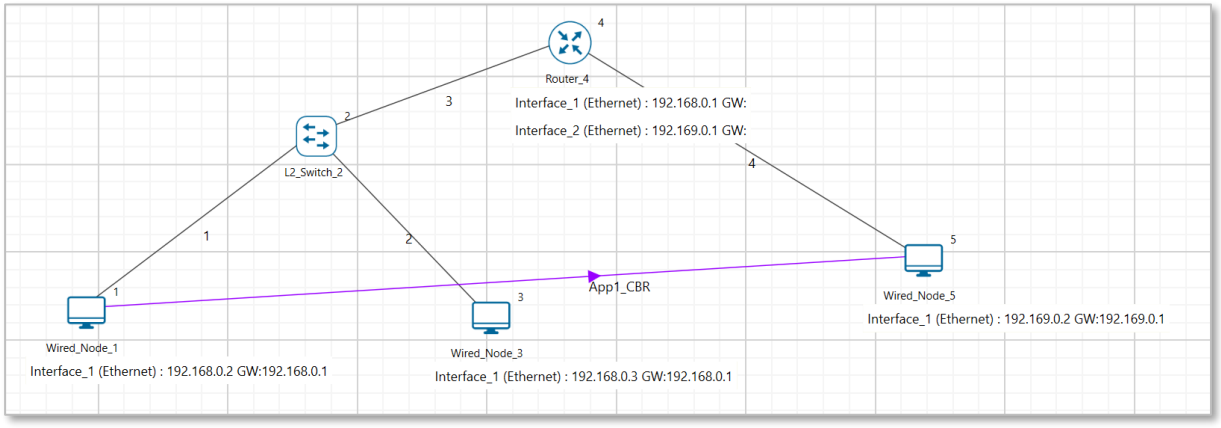


Figure 3-140: NetSim example scenario for configuring Static ARP

Note: The MAC ID can be obtained from the Device properties->Interface properties-> Datalink layer

DEVICE_ID=1,3,4,5,

NUMBER_OF_ENTRIES=5

192.168.0.2 155D00001001

192.168.0.3 155D00003001

192.168.0.1 155D00004001

192.169.0.1 155D00004002

192.169.0.2 155D00005001

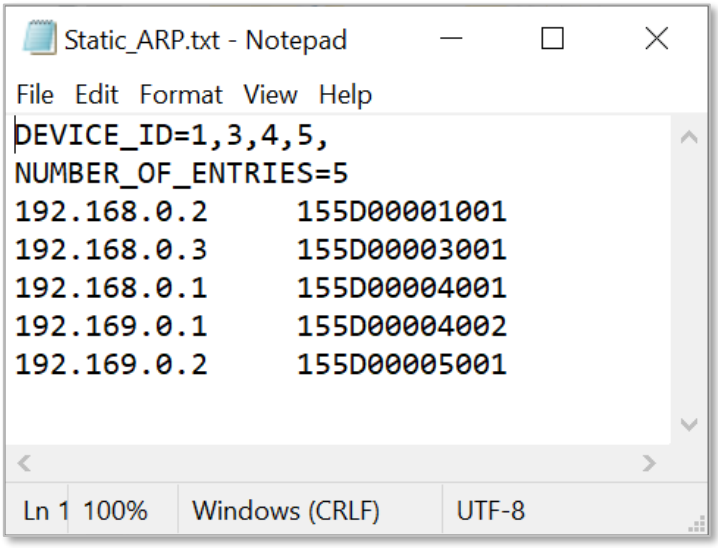


Figure 3-141: Static ARP input file

After providing the input, save the ARP route table information file. Then, select the **Enable (file)** option, browse for the saved file, click 'OK,' and perform the simulation.

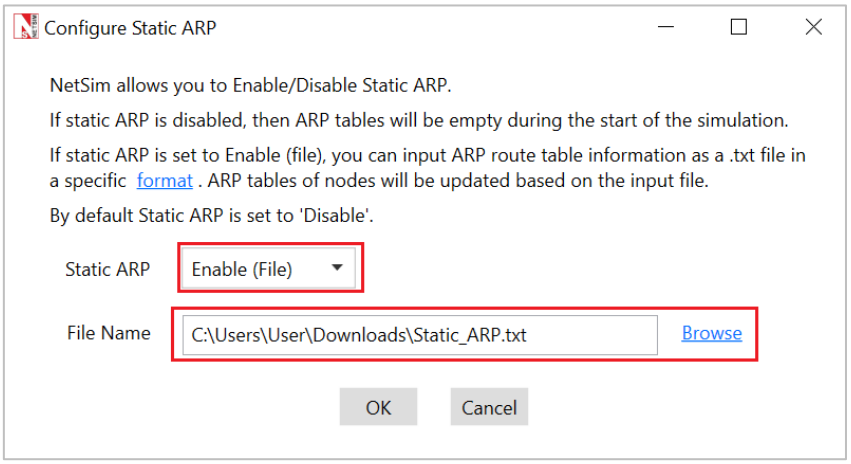


Figure 3-142: Option for enabling Static ARP file

4 Workspaces and Experiments

4.1 What is an Experiment and workspace in NetSim?

After users design and simulate a network in NetSim it can be saved as an *experiment*. This experiment is saved in a Workspace. A workspace also contains the source codes, executable files, icons, data files, etc. A workspace can contain one or more experiments. While NetSim supports multiple workspaces, users generally work in the *default* workspace. The default workspace of NetSim will have the master source code and the master binaries (compiled library, executable and DLL files).

New workspaces need to be created when:

- The user wants to modify the underlying source code of NetSim.
- A user chooses to organize a large number of saved experiments. The experiments can be saved in a different workspace.
- NetSim running on one PC/VM is time-shared between multiple users. Each user has his/her own workspace.

As mentioned earlier, NetSim stores your experiments (projects) in a folder termed as a Workspace. Default workspace is created in a user selected directory when NetSim is run for the first time after installation. Choose the path and enter the workspace name where you want the default workspace to be created.

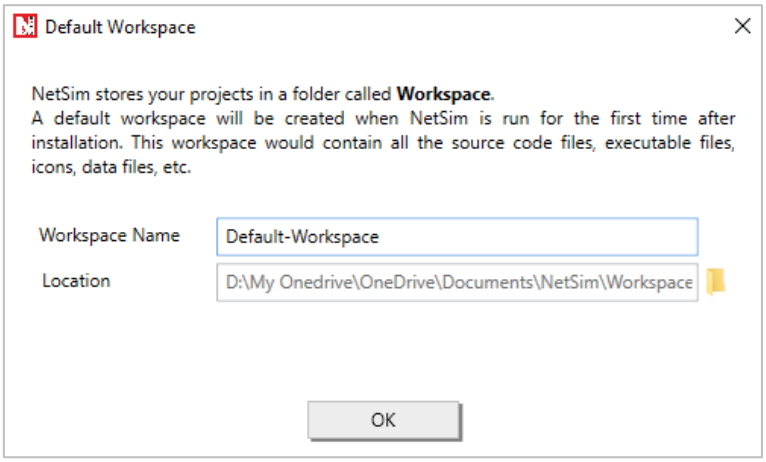


Figure 4-1: Default workspace created in Documents folder

This default workspace contains the following folders:

1. \src - contains protocol source codes.
2. \bin_x64 contains NetSim binaries for 64-bit.

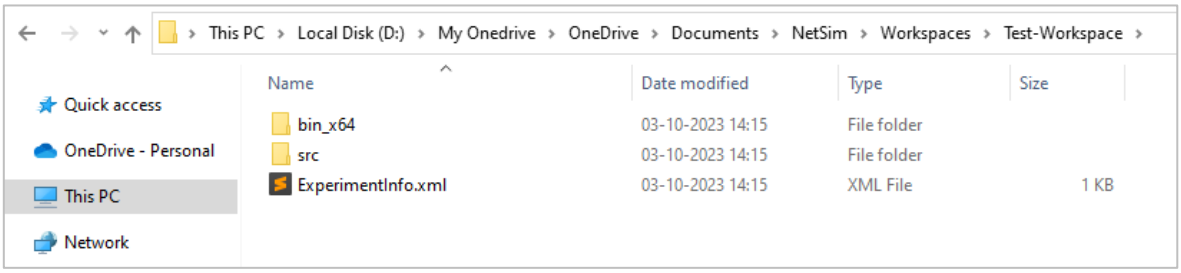


Figure 4-2: Default workspace containing the different folders

Note: In NetSim, the default workspace cannot be renamed.

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

To create an experiment, select **New Simulation - <Any Network>** in the NetSim Home Screen as shown Figure 4-3.

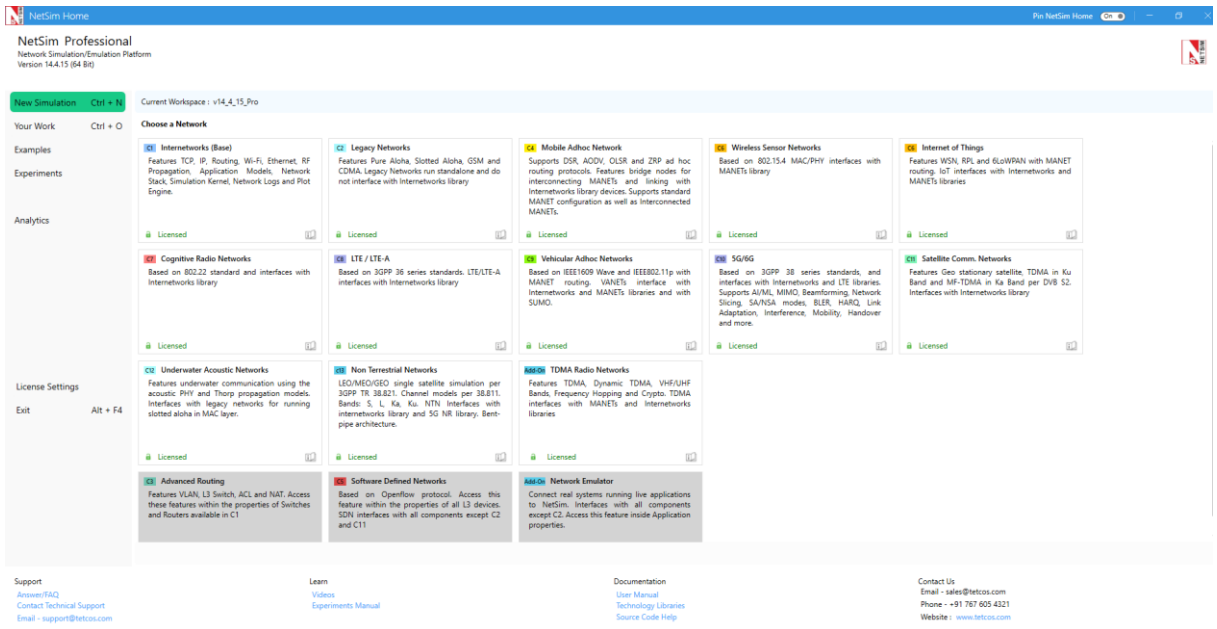


Figure 4-3: NetSim Home Screen

The created experiment can be saved by clicking on **Save** on the top left corner of the design window.

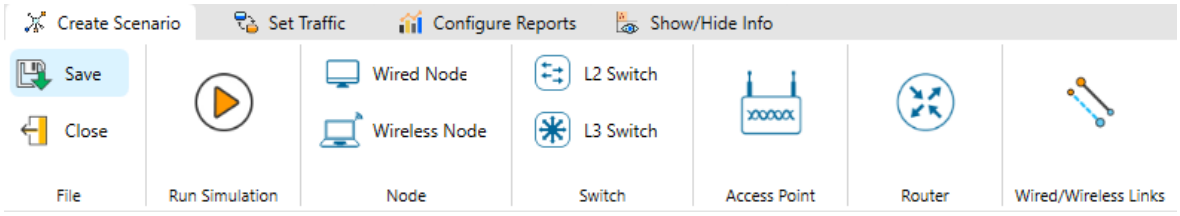


Figure 4-4: Save an experiment by clicking on its icon

A save pop-up window appears which asks the user to input an **Experiment Name and Description** for the experiment. The workspace path is also shown in the window.

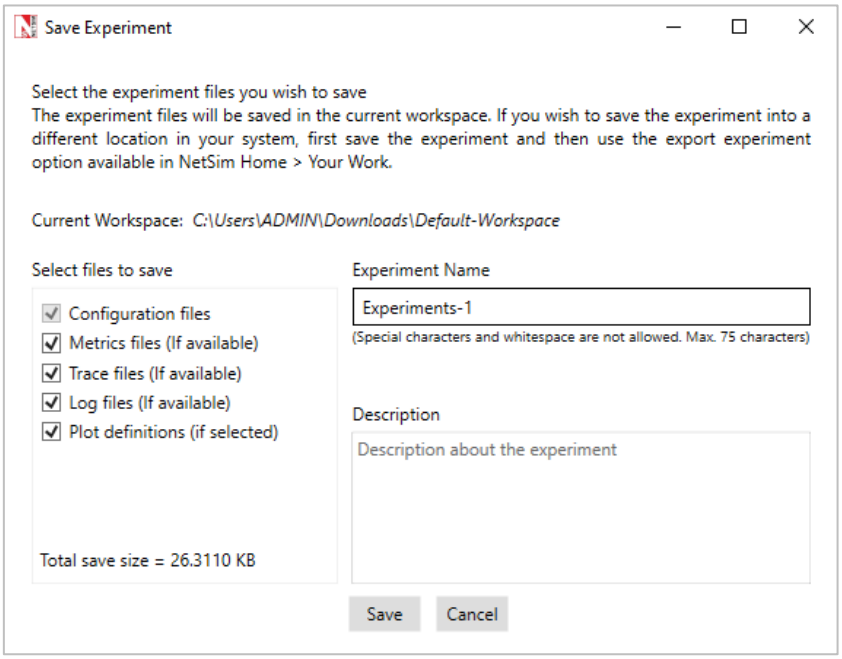


Figure 4-5: Save popup window.

User needs to input the Experiment Name (Description is optional) and then click on **Save**. The workspace path is non-editable. The experiment will be saved in the current workspace directory. Users can also select the files which are to be saved into the experiment folder.

- The Configuration file will be mandatorily saved into the experiment folder.
- Optional: Simulation output files such as Metrics Files, Trace files, Log Files, Plot files will be saved if enabled.

In our example, we saved the experiment with the name **Example-1** and this experiment can be found in the default workspace path as shown below Figure 4-6.

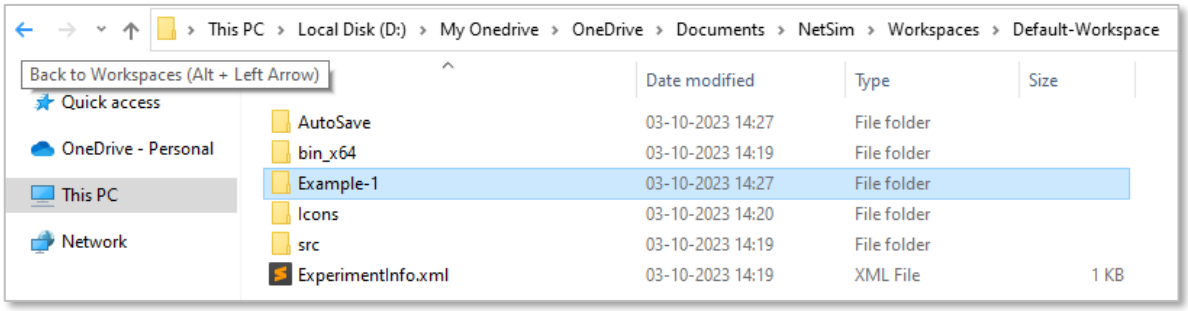


Figure 4-6: Sample Example Saved in Workspace

Users can also see the saved experiment in **Your Work** menu as shown below Figure 4-7.

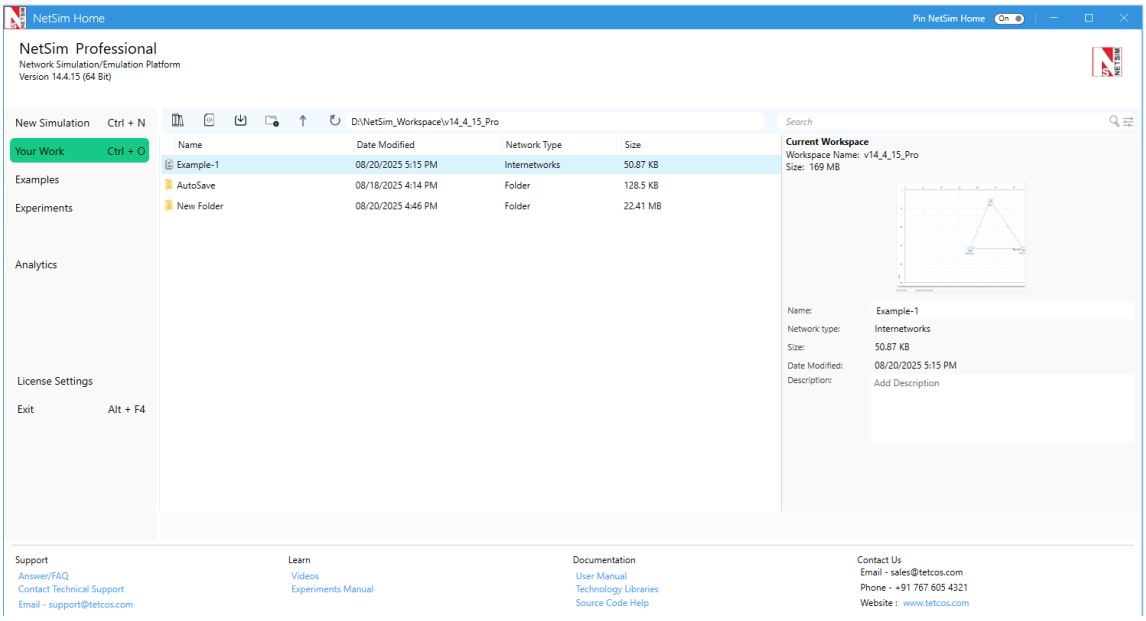


Figure 4-7: Saved experiments in Your work menu

If a Description was provided while saving the experiment, it will be displayed on the Description panel on the right. Users can also edit the description for an experiment in the description panel.

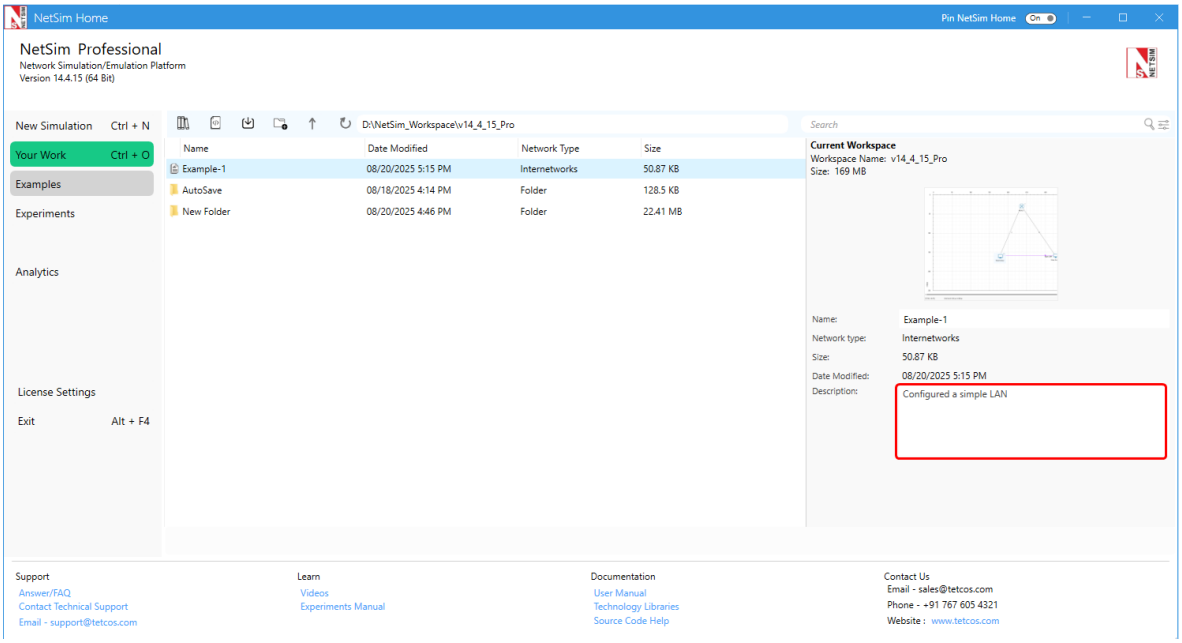


Figure 4-8: Description will be displayed on the right side of the description panel

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

Users can **Open file location** where the experiment is saved, **Free up Space** is used to delete the files that may not be important there by reducing the folder size, **Cut** (and paste inside a different folder), **Export** the experiment or **delete** the experiment, by right-clicking on the experiment in the **Your Work** window as shown below in Figure 4-9.

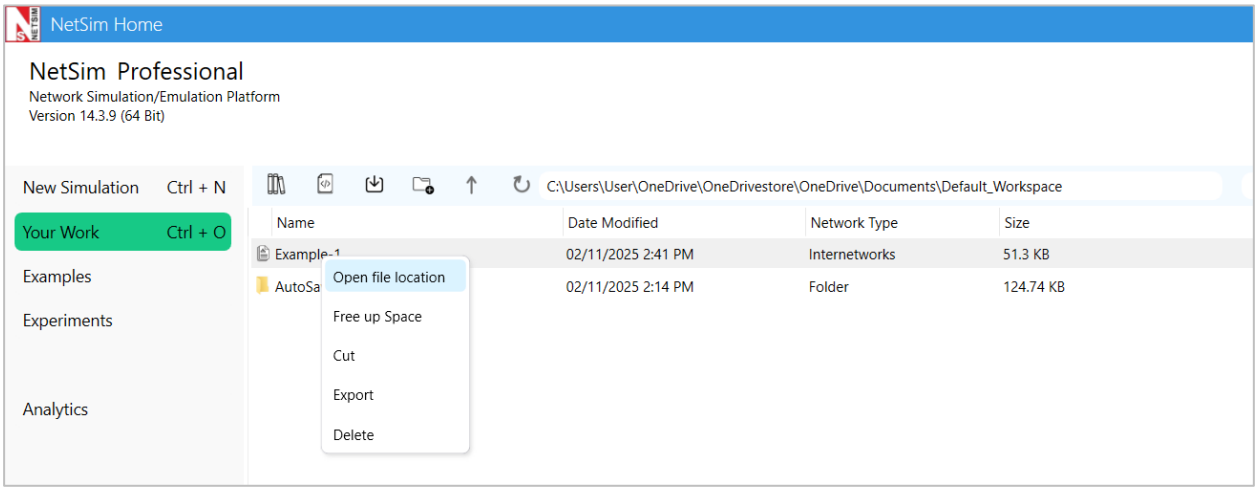


Figure 4-9: Right click on experiment name to view different options like “Open file location, Cut, Delete, Export” and Free up Space etc.

If the user wants to move the experiment into a New folder, create a **New folder** by right clicking on the experiment panel (either in the white space below the experiment list or on the header) or click on the **New folder** icon which is present in **Your Work** as shown below in Figure 4-10.

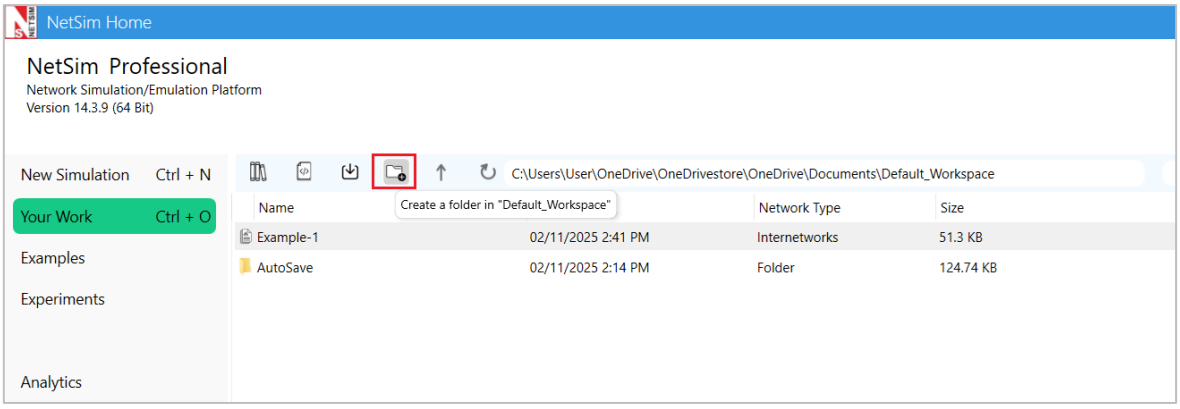


Figure 4-10: Create a “New Folder” using New Folder icon

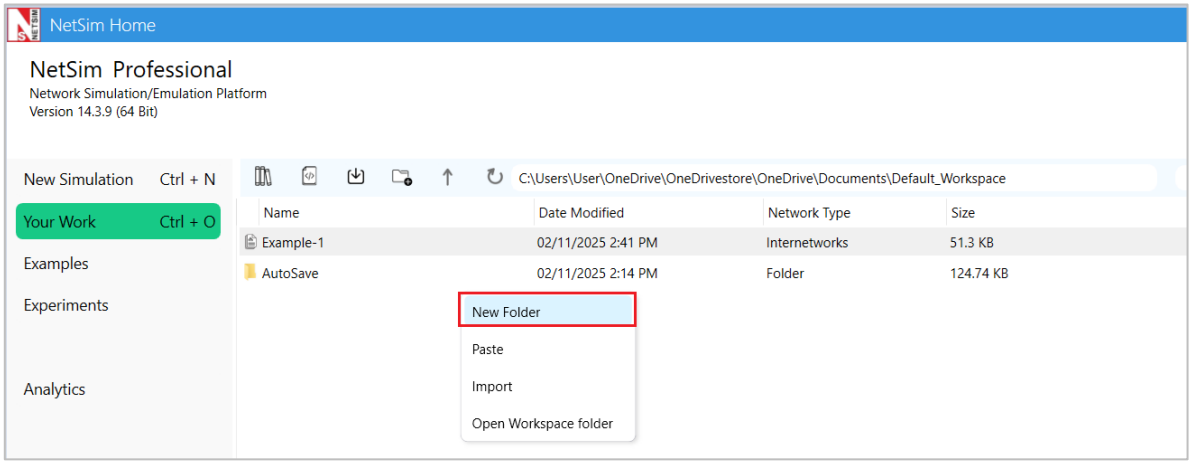


Figure 4-11: Create a “New Folder” by right clicking on the experiment panel

After creating a **New Folder** cut and paste the experiment inside it as shown below in Figure 4-12

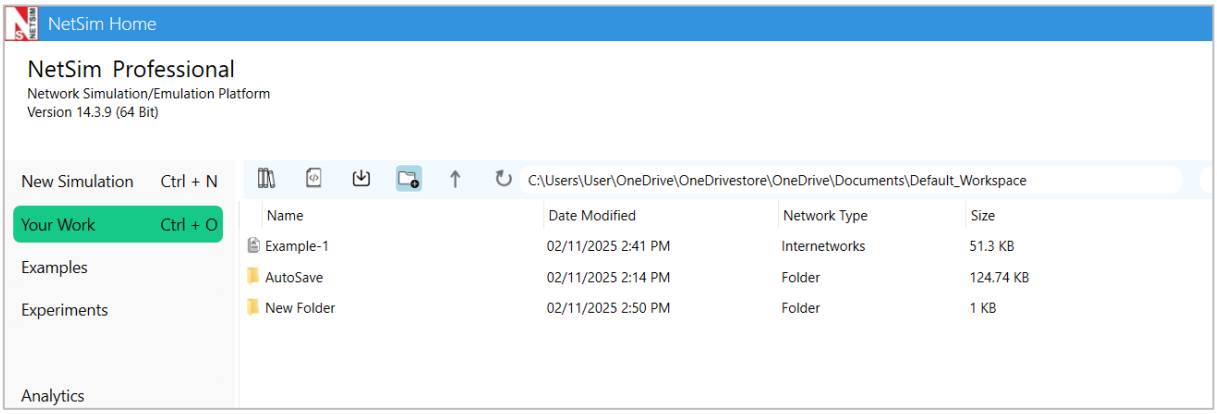


Figure 4-12: Cut and Paste the experiment in New Folder

Users can also **free up space** by deleting files which may not be important. Select the files and click on run. The deleted files can be regenerated by running the simulation again.

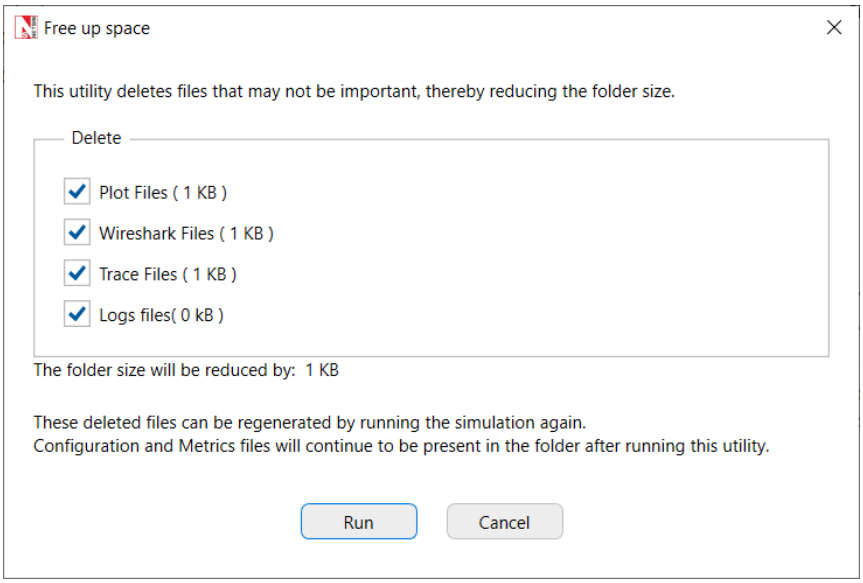


Figure 4-13: Free up space window

In this example, we have saved all the files related to Example-1. You can see the various files saved in the experiment folder in Figure 4-14.

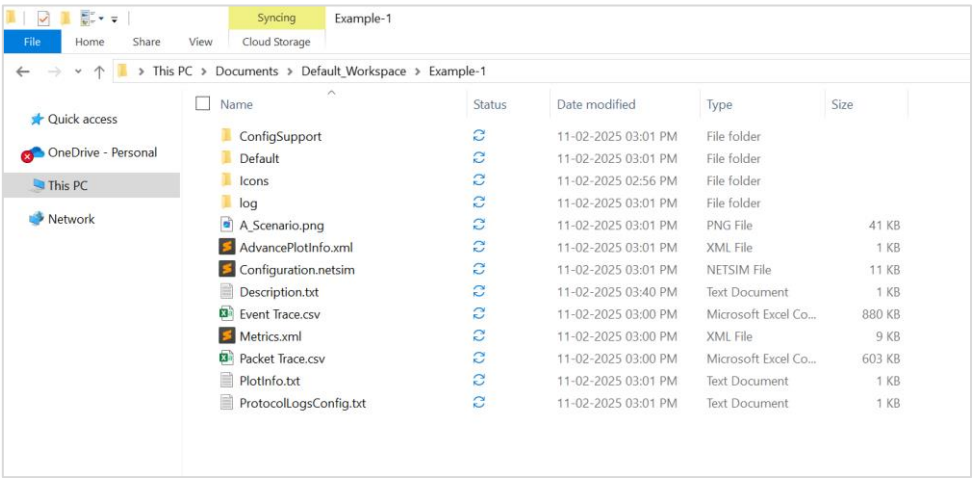


Figure 4-14: Simulation output files in experiment folder

4.3 Should each user have a workspace?

There is no strict association between users and workspaces. A single user can have multiple workspaces (and in turn experiments in each workspace), or multiple users can operate in one workspace.

4.4 How does a user export an experiment?

To save experiments in a different location, you have to first save the experiment in the current workspace and then use the export option present under **Your work** in the NetSim Home Screen as shown in Figure 4-15.

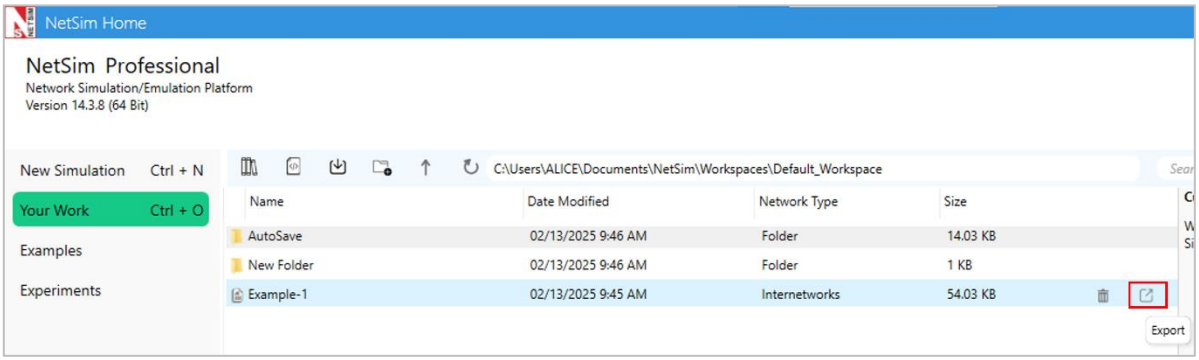


Figure 4-15: Export option present in Your work in NetSim Home Screen

If you click on the Export option, an Export Experiment panel appears where you can select the files to be exported. You can also select the source code and binaries if required. While the Configuration file is mandatory, other files are optional.

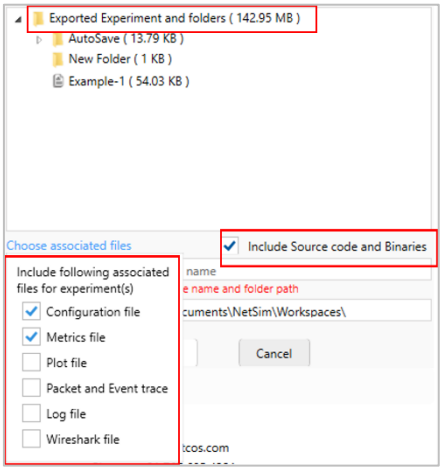


Figure 4-16: Export Experiment pop-up window

You need to give the destination path and name of the experiment while exporting. The exported experiments will be saved with a .netsimexp extension.

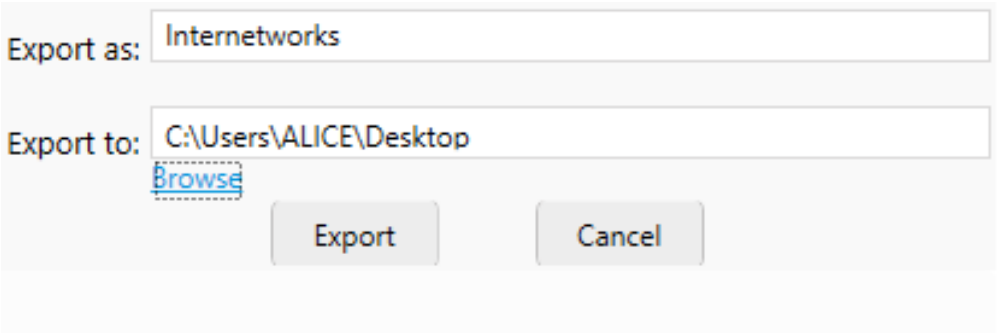


Figure 4-17: Select Export file and Path in export window

4.5 How does a user delete an Experiment in a workspace?

Users can delete experiments by clicking on the delete icon as shown below in Figure 4-18.

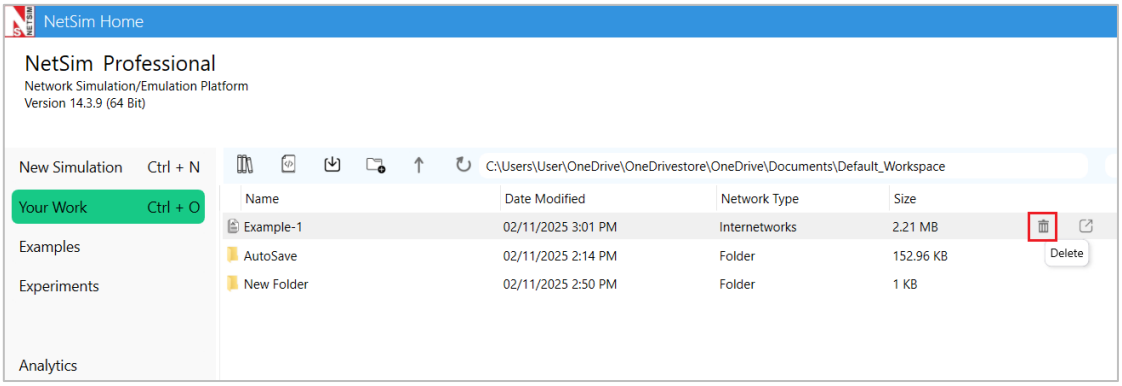


Figure 4-18: Delete an Experiment in a workspace

It displays a popup window as shown in Figure 4-19. Click on YES.

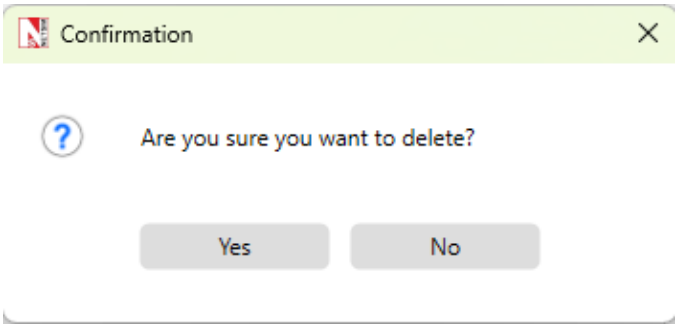


Figure 4-19: Delete experiment confirmation window popup

4.6 How does a user create a new workspace?

To create a new workspace, click on Manage Workspaces present **in Your work >> select List of Workspaces** Options as shown below

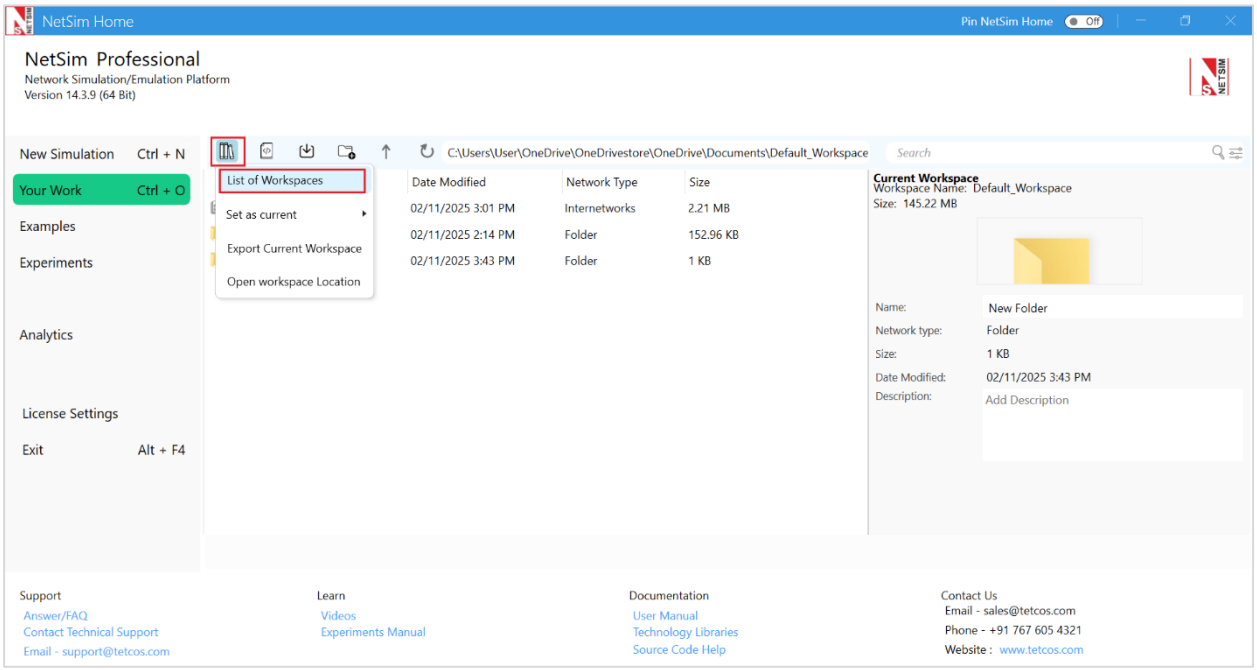


Figure 4-20: Select List of Workspaces option in Your work window

Select **New Workspace** as shown below.

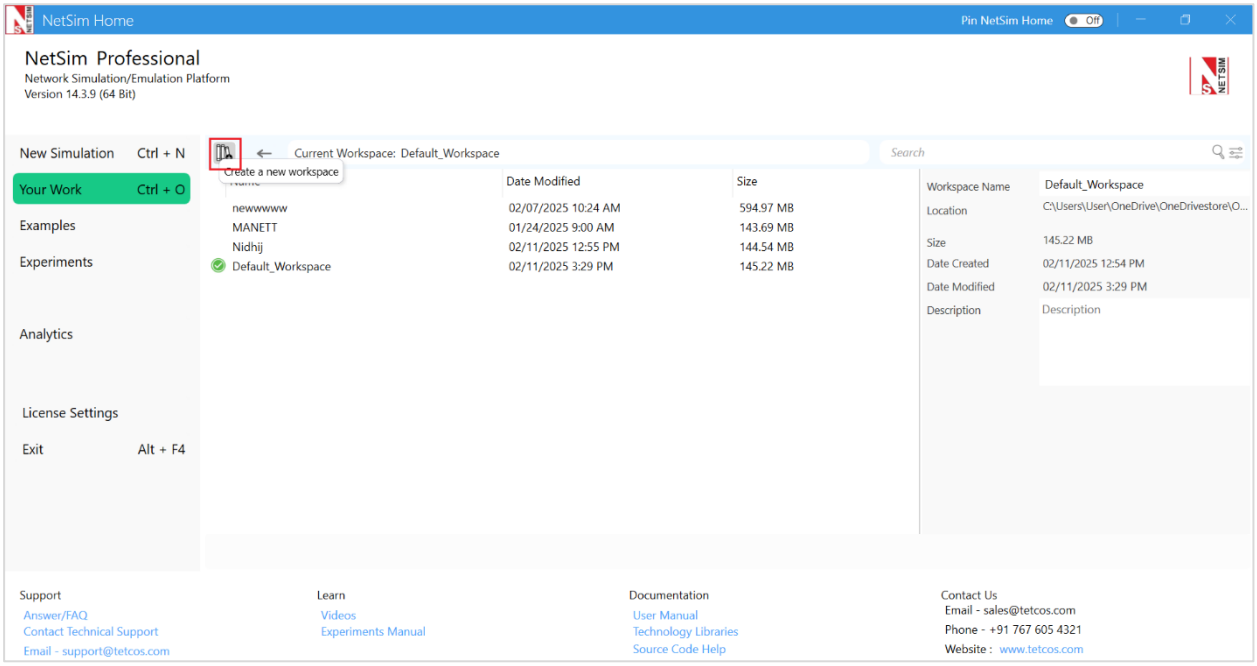


Figure 4-21: Select New Workspace option

A New Workspace pop-up window appears where user can input the Workspace Name, Description and Workspace Path as shown below

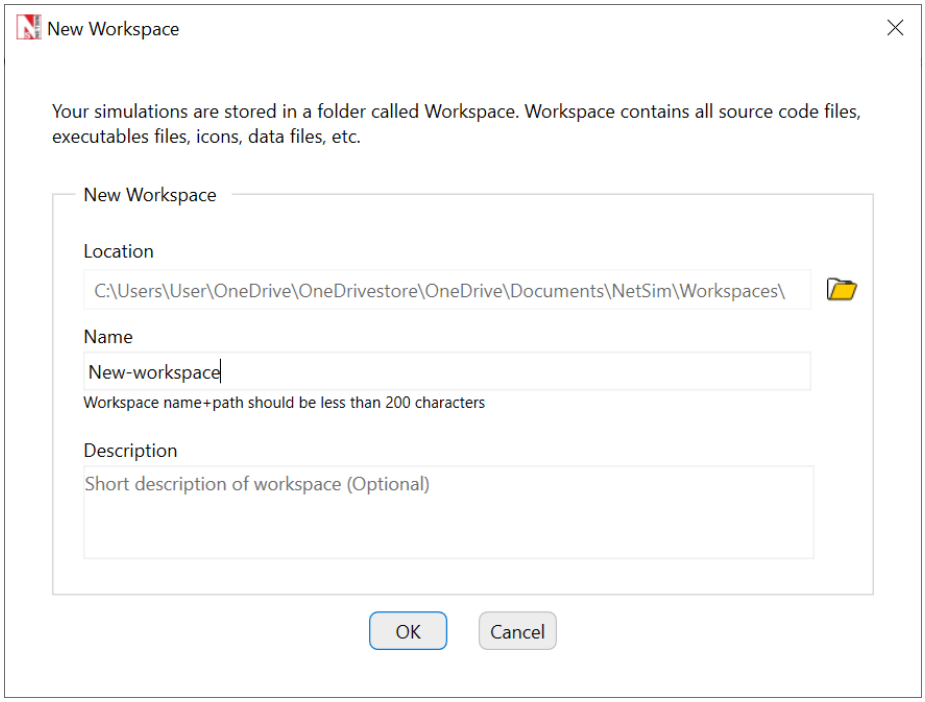


Figure 4-22: New Workspace pop-up window

4.7 How does a user switch between workspaces?

Users can switch from one workspace to another. Select **Your work >> Manage Workspaces >> List of Workspaces** and click on the workspace you want to switch to .

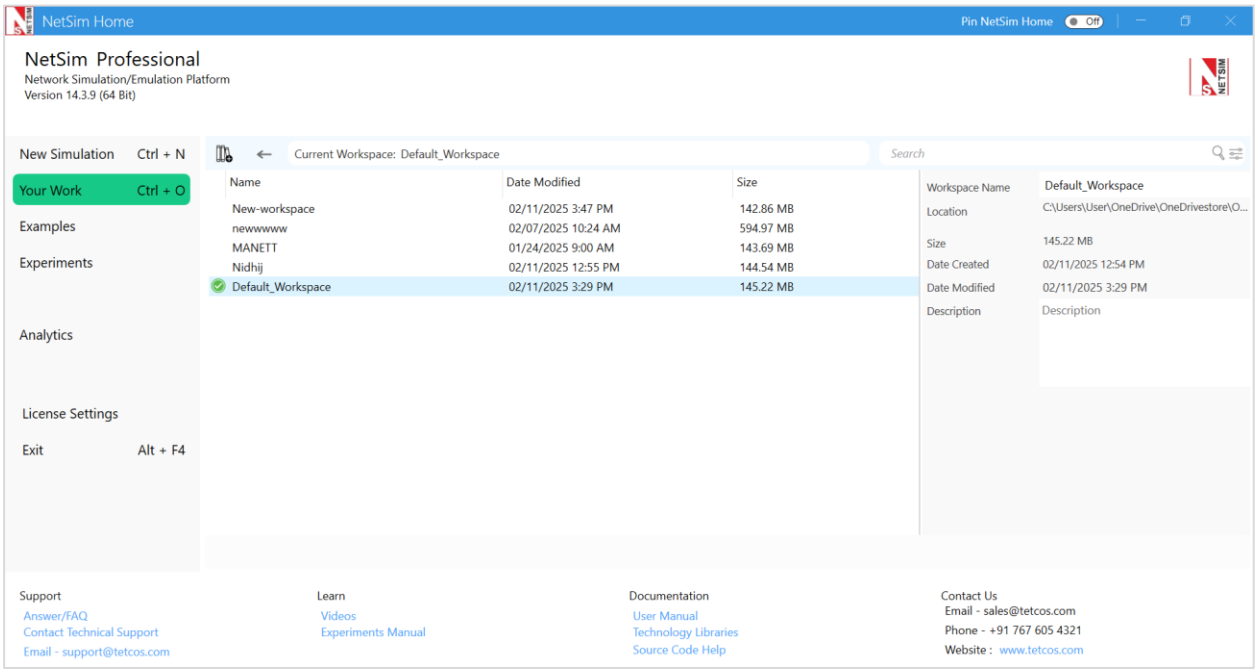


Figure 4-23: Switch between workspaces

And then select Set as Current symbol (the green tick mark) as shown below

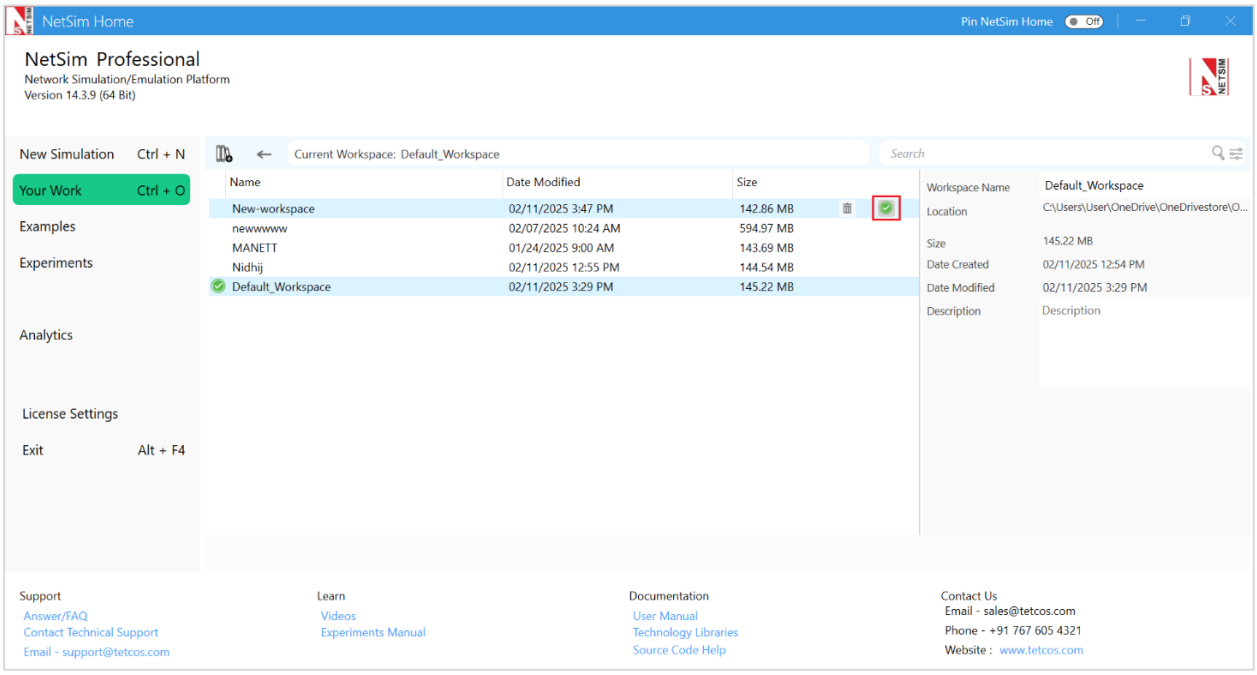


Figure 4-24: Select Green to “Set as Current” workspace

4.8 How does a user export a workspace?

Users can export only the workspace by selecting **Your Work >> Manage Workspaces >> Export Current Workspace** as shown below in Figure 4-25.

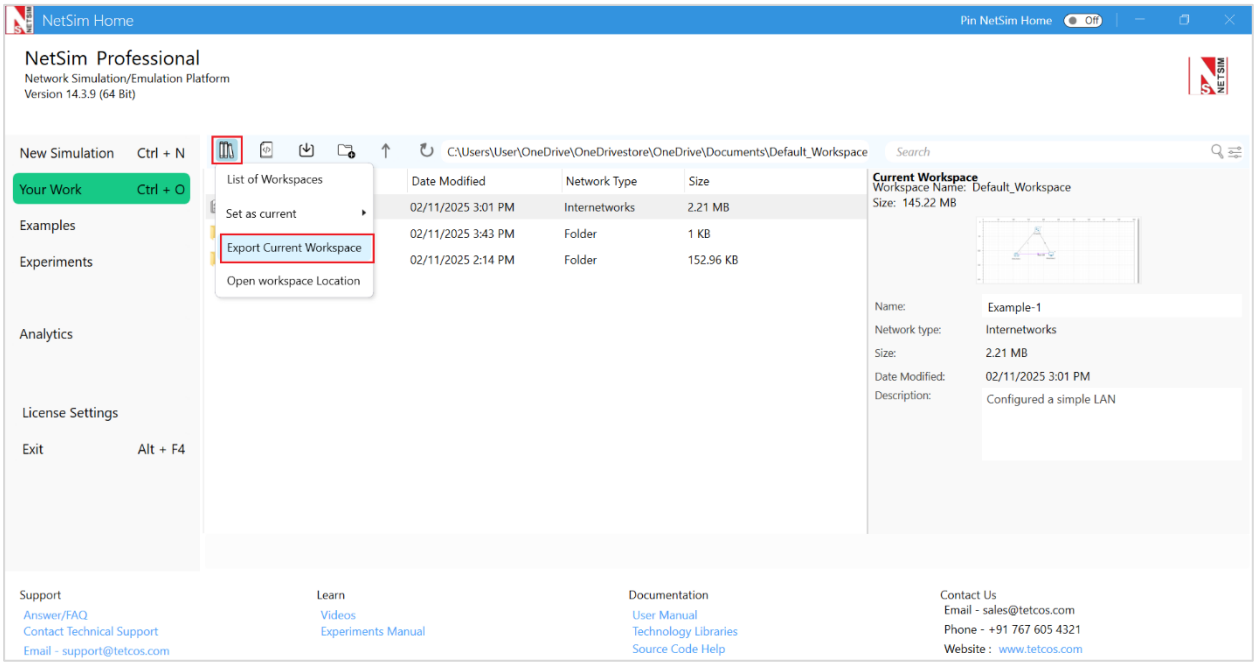


Figure 4-25: Select Export option in Your work window

This will show the export workspace window panel in the right side panel with all the existing experiments in that particular workspace. This option is similar to exporting an experiment. You can select the files which are to be exported as part of the workspace and then can select the

source code and binaries if required. The Configuration file is mandatory and other files are optional.

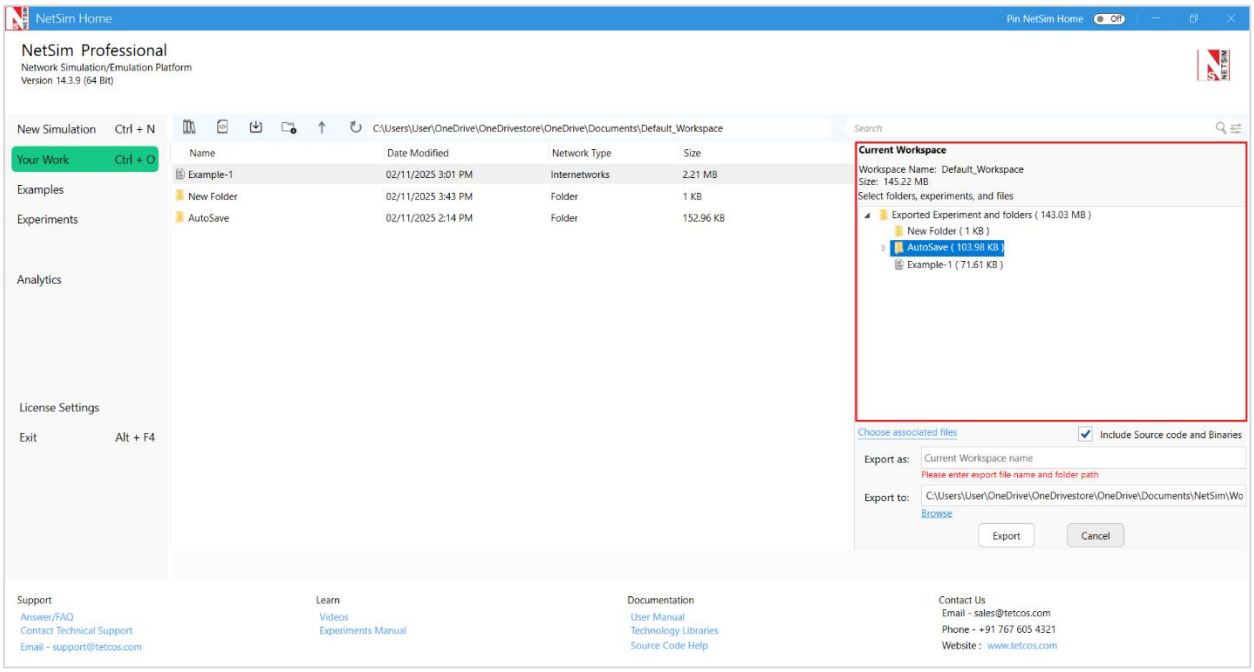


Figure 4-26: Adding default binaries, source code and icons to Selected Experiments list

Users can enter the name and path in which the workspace is to be exported and then click on Export.

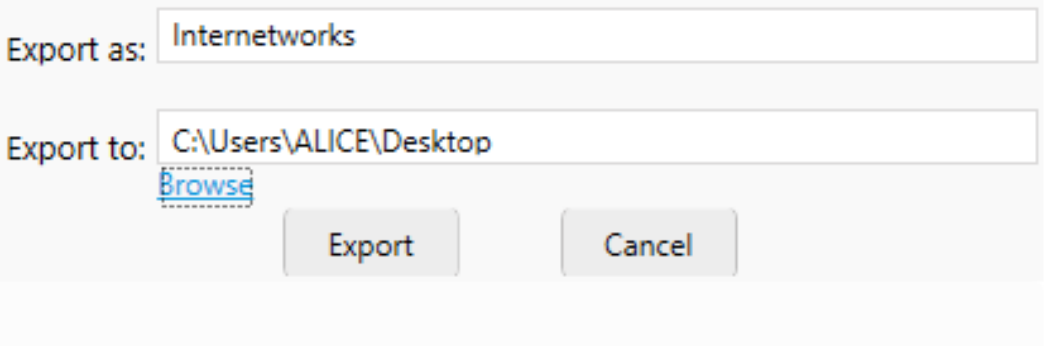


Figure 4-27: Enter the workspace name and select the export location

The workspace will be exported to the path selected. It will have the extension .netsimexp as shown in Figure 4-28.

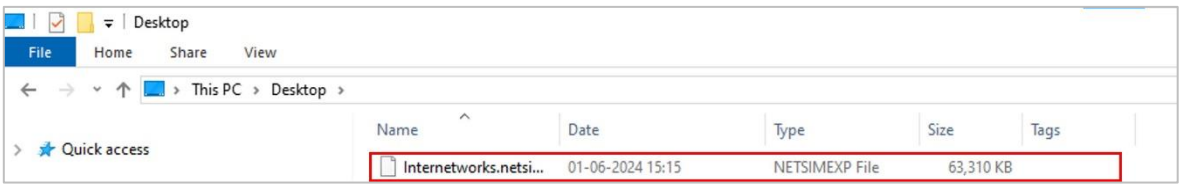


Figure 4-28: Workspace exported to Location.

If you want to remove an experiment from the workspace being exported, click on that experiment, and click on remove as shown in Figure 4-29.

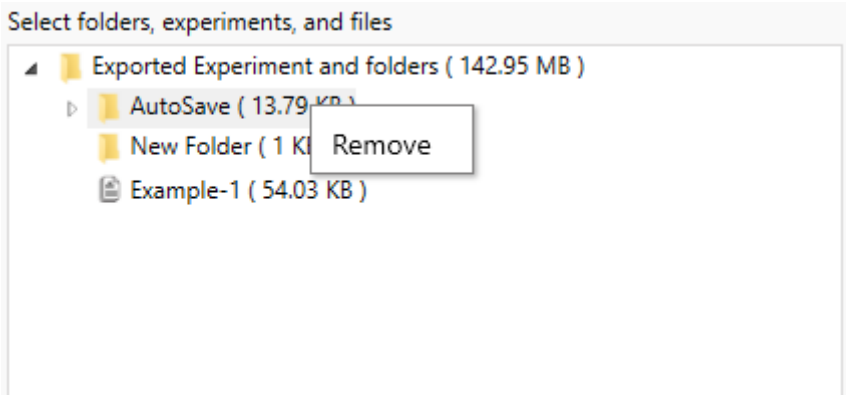


Figure 4-29: In Export list window Right click on experiment and select Remove

4.9 How does a user import experiment and workspace?

You can import only an exported workspace, import experiments and workspaces by first selecting Your work and then selecting the Import option as shown in Figure 4-30.

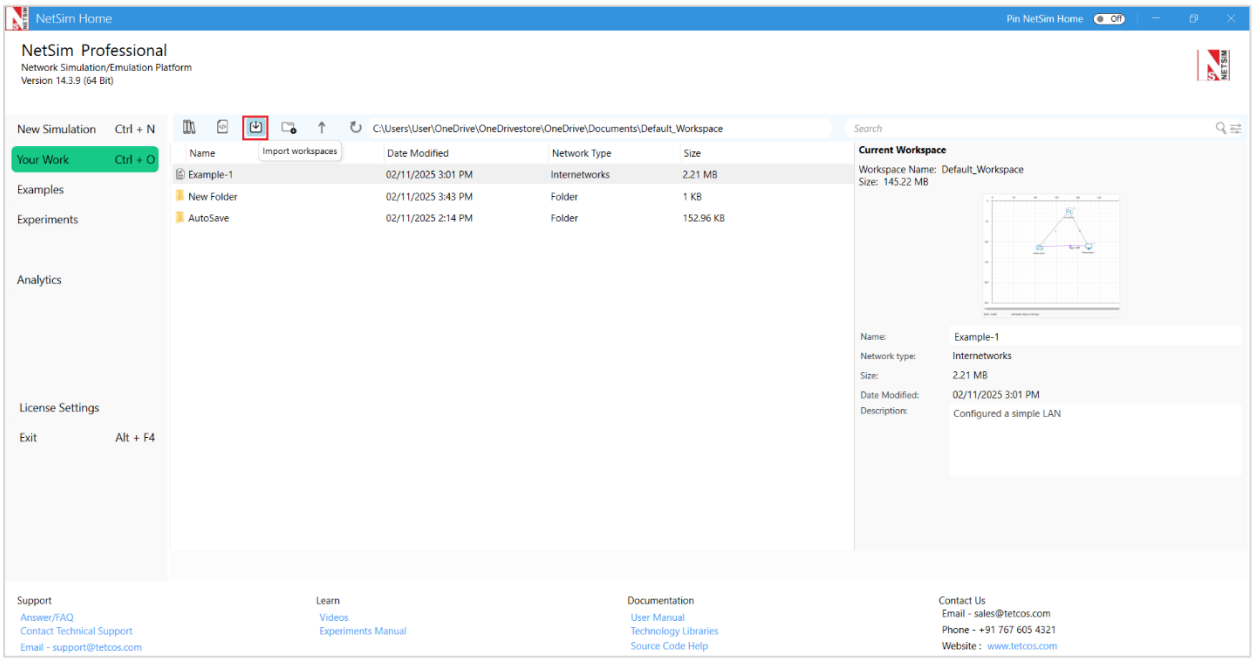


Figure 4-30: Select Import option in Your work window

4.9.1 Importing Configuration.netsim file from experiment folder

Once you click the import option from Your work the following window will open. Click on **Experiment/Workspace file** option and import the Configuration file. Enter the path from where the configuration file has to be imported as shown in Figure 4-31.

Note that

- Only files with "**.netsim"** extension can be imported.
- By Selecting "**Copy all files available in the folder**" option user can import all files present along with Configuration.netsim.

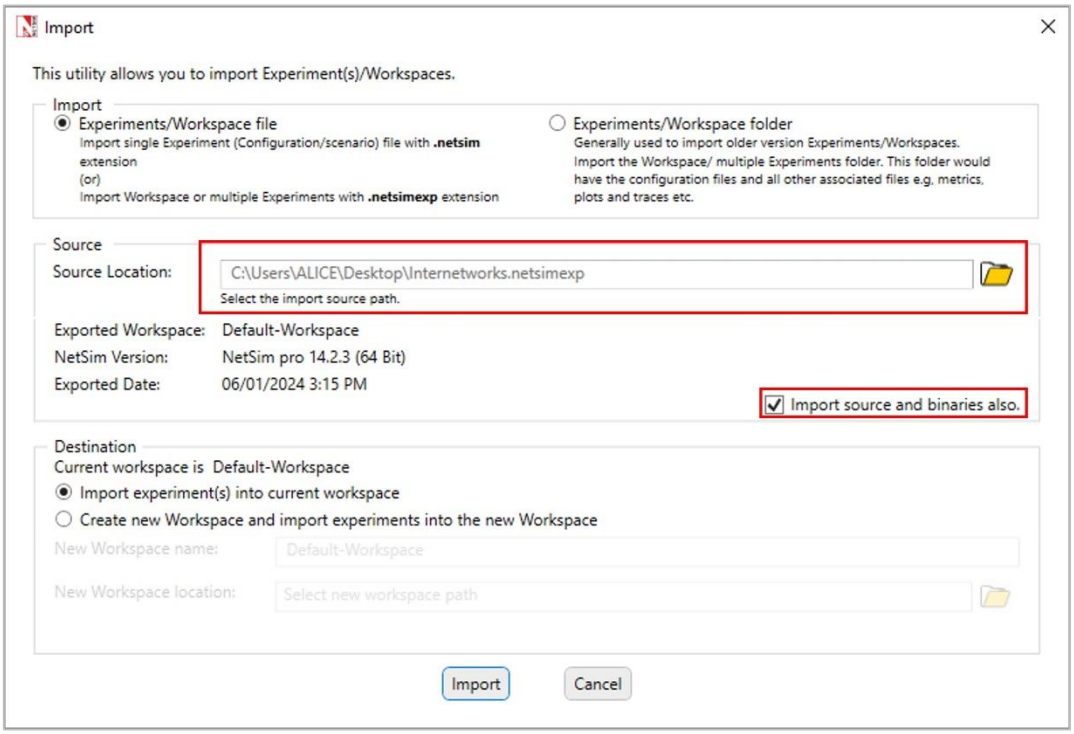


Figure 4-31: Importing configuration files

The imported experiment file will be available in the current workspace. It can be seen by clicking on Your Work as shown below

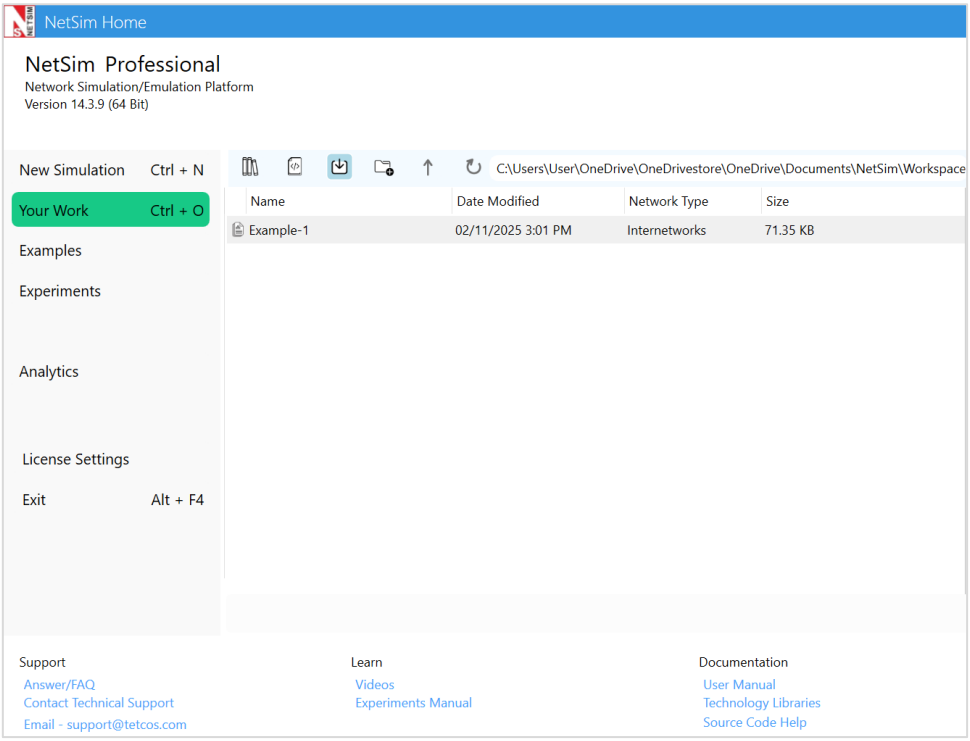


Figure 4-32: Imported experiment file in the current workspace

4.9.2 Import workspace or multiple experiments file

This section explains how (i) Users can import multiple experiments into your current workspace, and (ii) Import a complete workspace. Click Your work and then select Import option as shown below

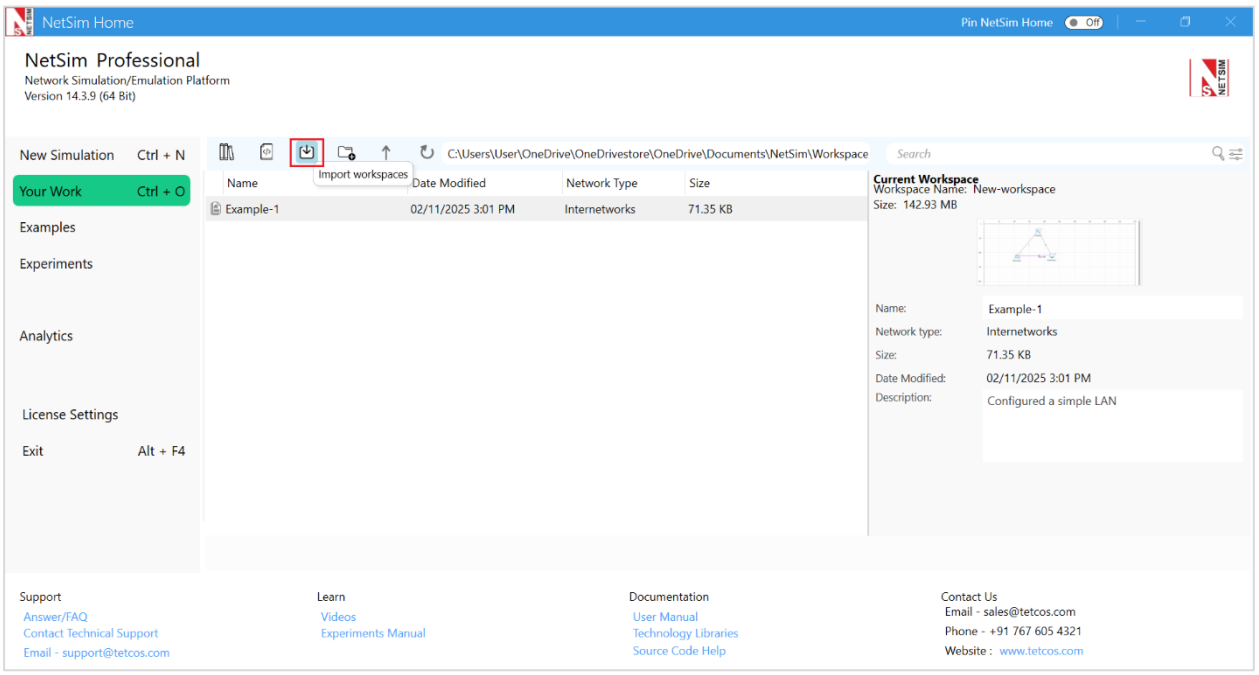


Figure 4-33: Import workspace/multiple experiments option in Your work window

You need to input the path from where (i) the experiments (a single folder) or (ii) the workspace will be imported. To import multiple experiments into the current workspace, click on the option as shown in Figure 4-34.

Note that

- Only previously exported experiment/workspaces with "**.netsimexp**" extension can be imported.
- By Selecting "**Import source and binaries also**" user can import source code and binaries present along with the exported experiments or workspace.

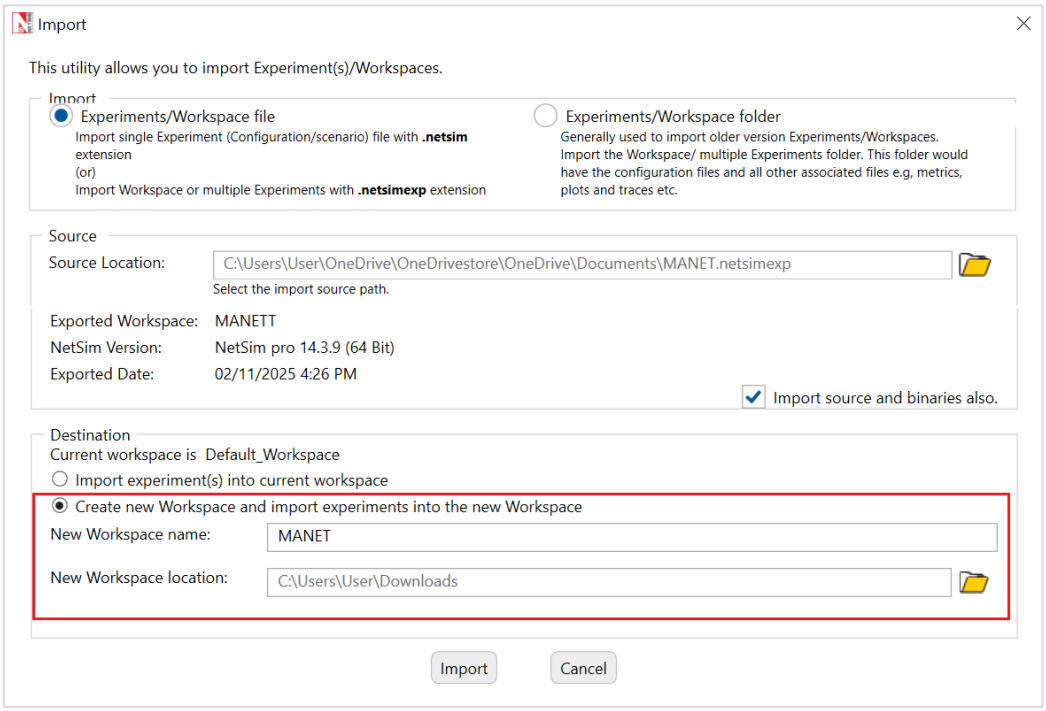


Figure 4-34: Window for importing multiple experiments or workspace into current workspace

If the user wants to import the experiments into the new workspace, select the option as shown in Figure 4-35, and proceed accordingly.

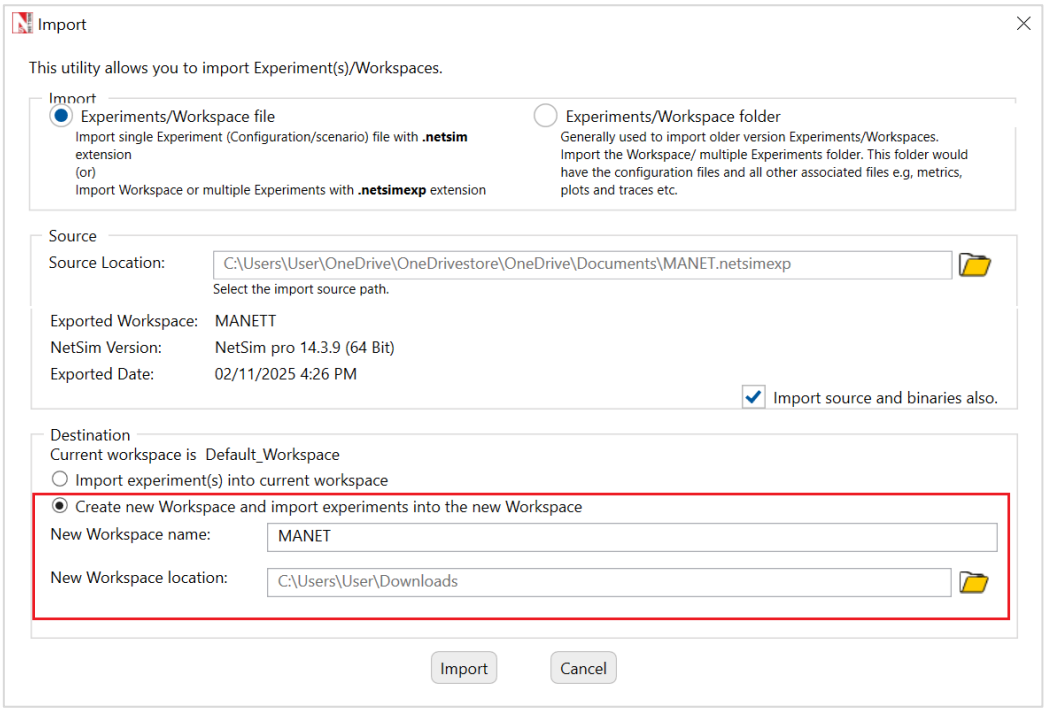


Figure 4-35: Create a new workspace, and import experiments and source code/binaries, into the new workspace

If you import the experiments into the current workspace then experiments will be displayed in the Your Work menu of the current workspace as shown below.

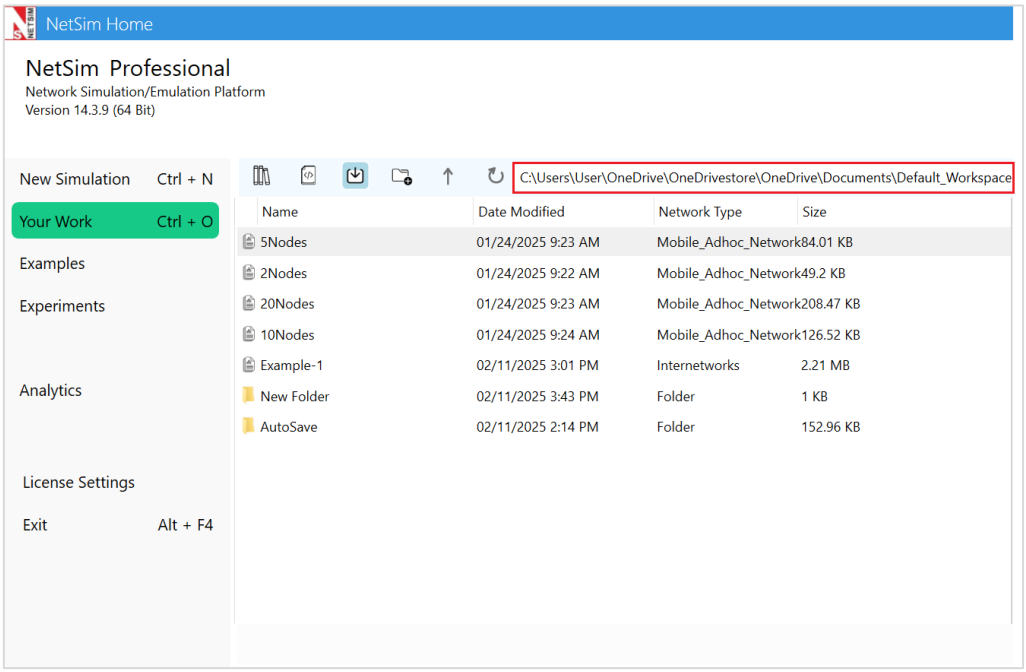


Figure 4-36: Imported experiments shown in Your Work of current workspace

If you create a new workspace and import experiments and binaries/source code then, these experiments will be shown in the Your Work Menu of the new workspace

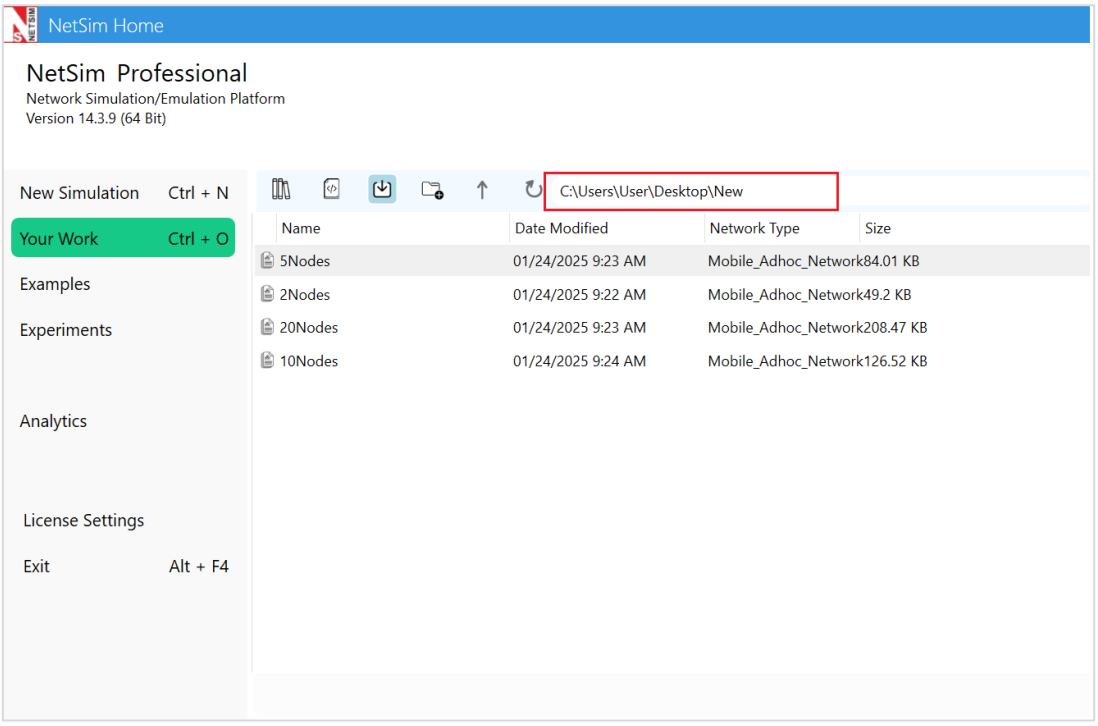


Figure 4-37: Imported experiments shown in Your Work menu of a newly created workspace

4.10 Import Experiments or Workspace folder

User can import an experiment folder or a workspace folder by choosing the following option in the experiment import window as shown below

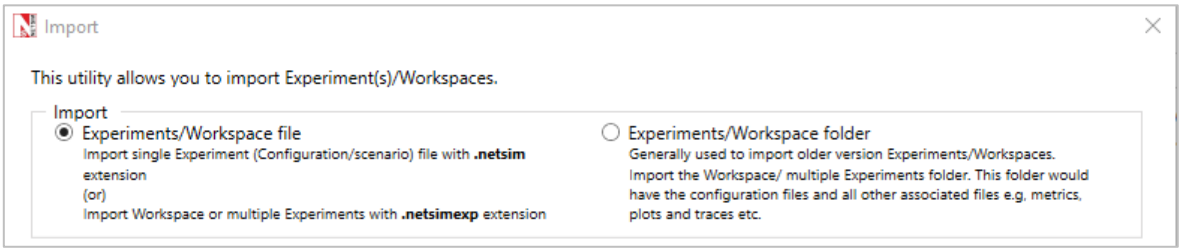


Figure 4-38: Option to import Experiment/Workspace folder

You should give the path from where the workspace/experiment folder must be imported. Then click on import as shown below

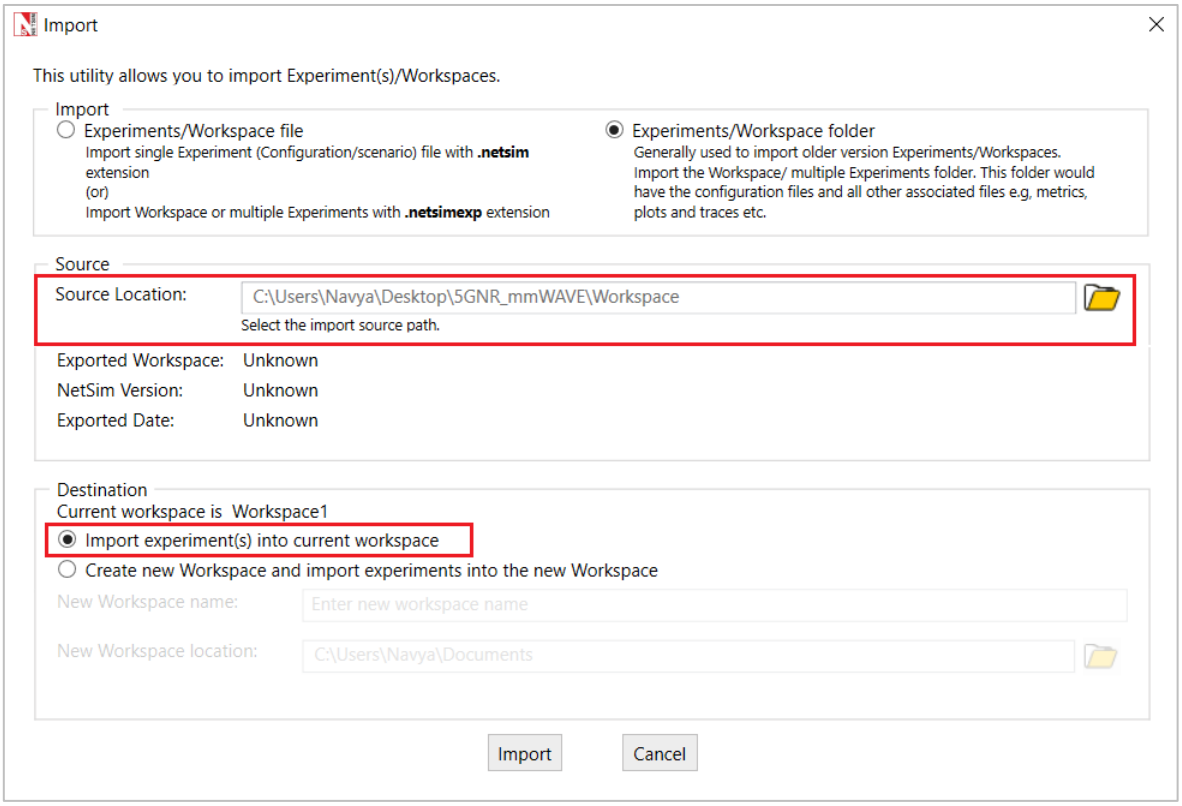


Figure 4-39: Window for navigation of experiments/workspace into current workspace

If you want to import the folder into new workspace, choose the option as shown below

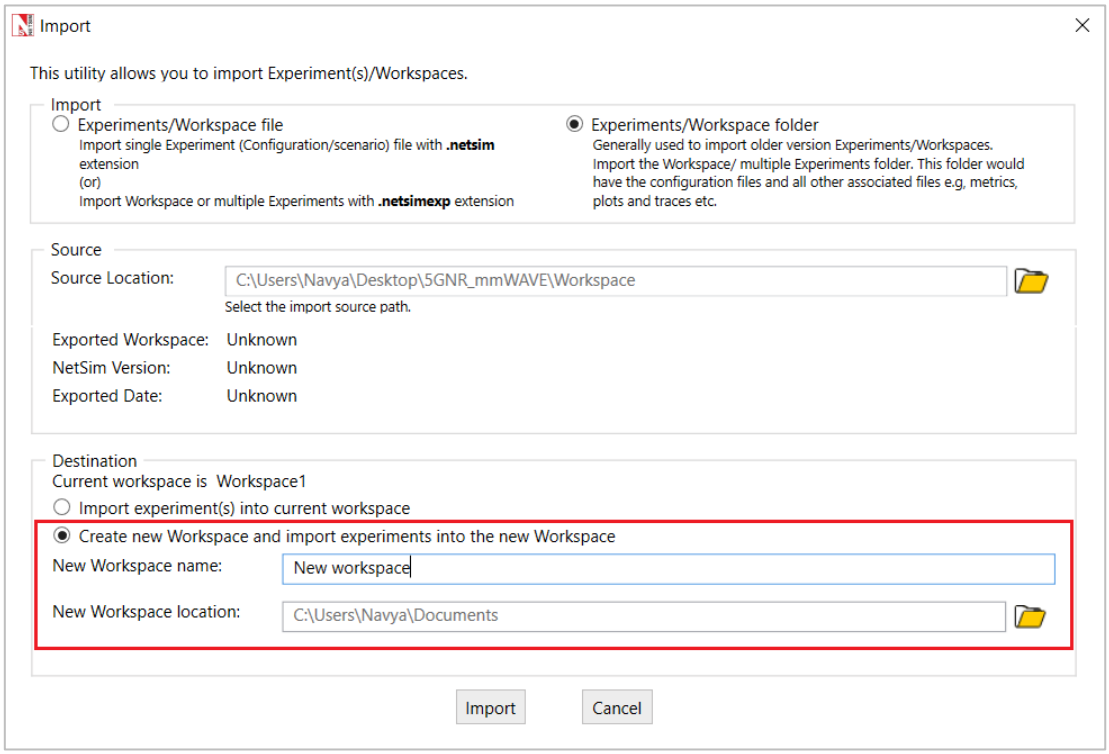


Figure 4-40: Importing the Experiment folder into new workspace

4.11 Import into current workspace vs. creating a new workspace

A new workspace generally needs to be created when the underlying source code is likely to be modified. If you are importing only experiments, then they can be imported into your existing workspace. However, if you wish to import experiments plus the binaries/source codes then we recommend you create a new workspace for the same.

4.12 How does a user delete a workspace?

You can delete a workspace by clicking on the delete icon shown below

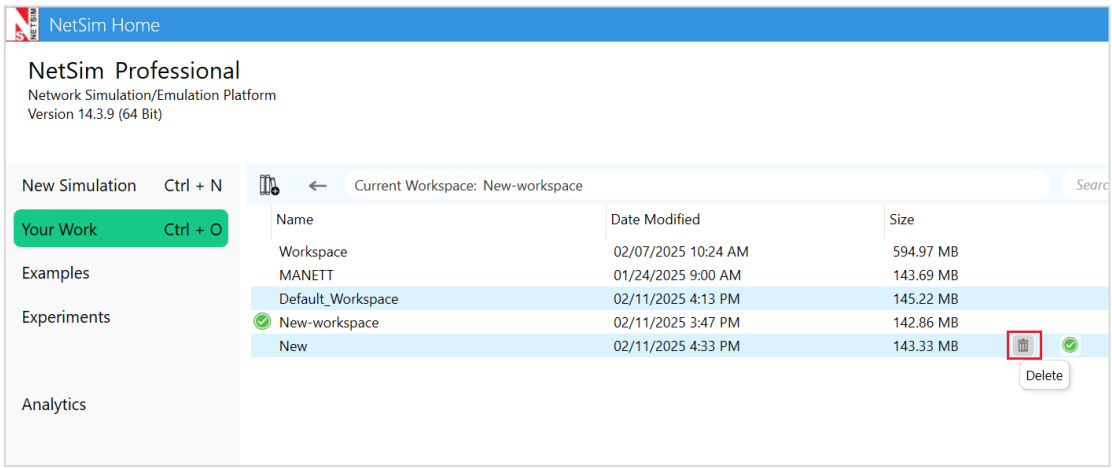


Figure 4-41: Delete a workspace by clicking on the delete icon in Your work window

Deleting a workspace will delete all the saved experiments and code modifications done in that workspace.

The following window appears if the current workspace is deleted or removed after re-installation of NetSim.

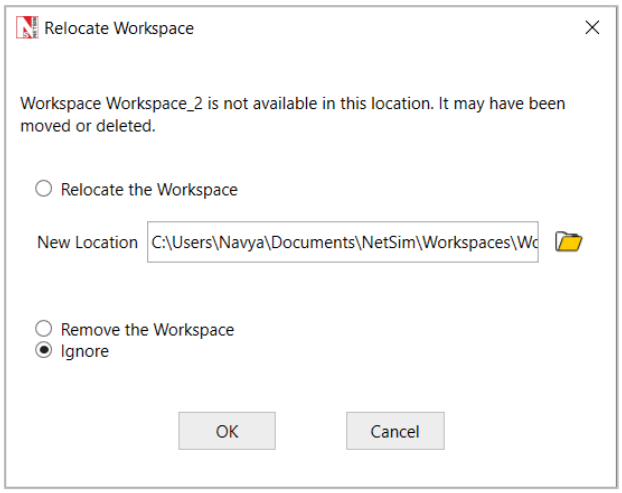


Figure 4-42: Relocation workspace window

The “Relocate the workspace” option will allow the user to select a new location for the workspace which was removed/ deleted. User can also ignore the message by selecting “Ignore” option and clicking on OK .

4.13 How does a user open and modify source codes?

You can modify the source codes within a workspace. For doing this, select Your work -> Open, Reset or Compare code option->Open Code as shown below

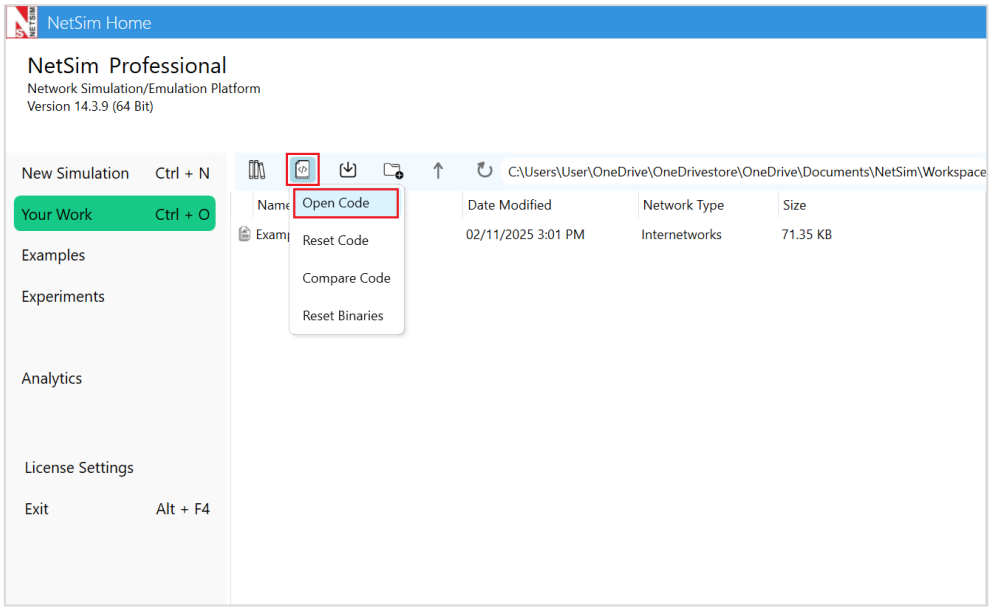


Figure 4-43: Open code option is available in Your work window

This opens the C source codes in MS Visual Studio. You can then modify the protocol codes and compile the codes. Then create a network in NetSim or open the saved experiment which simulates the protocol that has been modified. Run the simulation. This simulation will run per the modified code. Note that the changes in the source codes applies to the current workspace only.

4.14 How do I reset my code changes?

Each workspace has two Reset options. They are reset:

1. **Binaries** (compiled files) to default: If users need to reset the protocol binaries (DLLs) to default/ standard working of the protocol, users can use 'Reset binaries' option. Note that using this option will NOT reset the code modifications. It will reset all the protocol binaries of that workspace.

   Steps to reset: Go to Home screen > Your work > Open, Reset or Compare code option > Reset Binaries.

2. **Code** (source C codes) to default: If users need to reset the protocol source code to the default/ standard protocol source code, users can use 'Reset Code' option. Note that using this option will NOT reset the working of protocol if user has compiled modified code. It will reset only the code to default of that workspace.

   Steps to reset: Go to Home screen > Your work > Open, Reset or Compare code option > Reset Code.

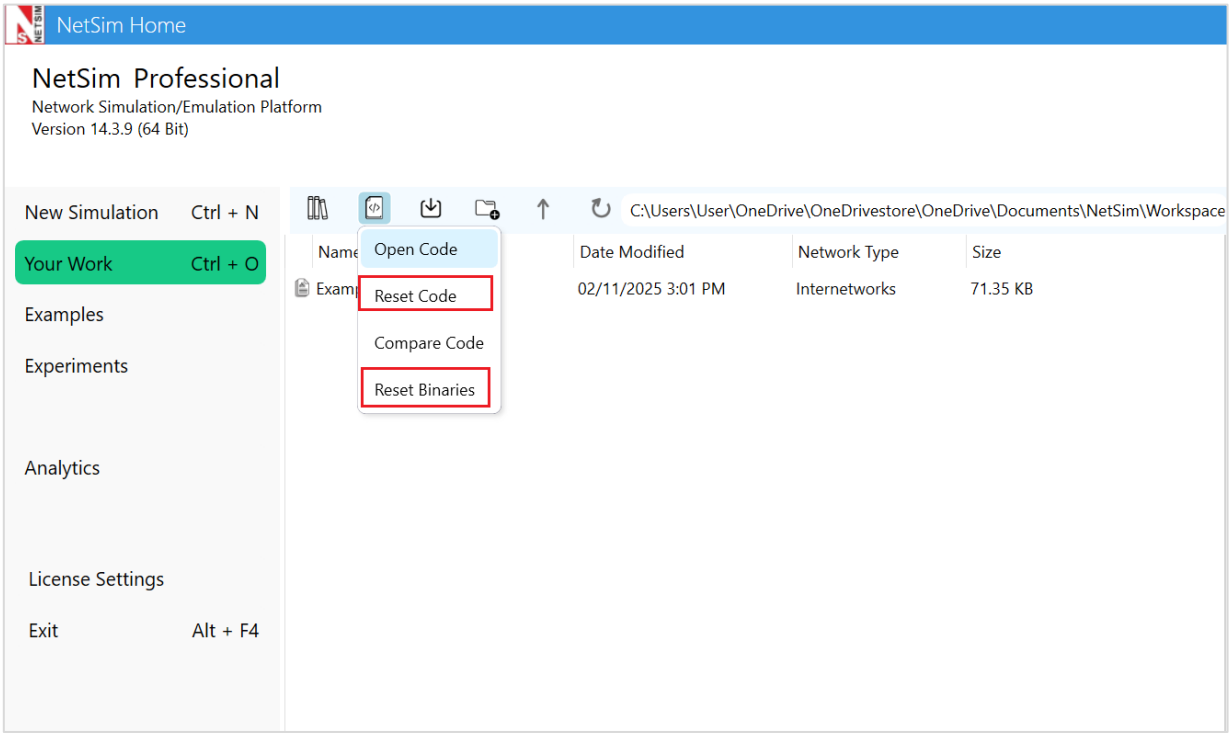


Figure 4-44: Use appropriate option to reset the code and binaries

5 Simulating different networks in NetSim

The following table lists the networking technologies available in the different versions of NetSim.

Type of Network	NetSim Versions
Internetwork	All versions
Legacy Network	All versions
Cellular Network	All versions
Advanced Routing	All versions
MANET	All versions
Wireless Sensor Network	All versions
Software Defined Network	All versions
Internet of Things	All versions
Cognitive Radio	All versions
LTE	All versions
5G NR	Available only with NetSim Standard and NetSim Pro versions
VANET	All versions
Satellite Communication	Available only with NetSim Standard and NetSim Pro versions
Underwater Acoustic Network	Available only with NetSim Standard and NetSim Pro versions
Non Terrestrial Networks	Available only with NetSim Standard and NetSim Pro versions
Network Emulator (Add On)	Available only with NetSim Standard and NetSim Pro versions
TDMA Radio Networks (Add on)	Available only with NetSim Pro version

Table 5-1: Networking technologies available in the different versions of NetSim

NetSim comes with inbuilt examples to help you understand how the different types of networks work.

The devices models in NetSim represent common networking devices in a generic way and do not model any specific vendor's implementation of the device. In real-world networks, each device has specific vendor implementation of networking protocols.

5.1 Internetworks

An Internetwork is a collection of two or more computer networks (typically Local Area Networks or LANs) which are interconnected to form a bigger network.

Internetwork's library in NetSim covers Ethernet, Address Resolution Protocol (ARP), Wireless LAN – 802.11 a / b / g / n / ac / ax / p and e, Internet Protocol (IP), Transmission Control Protocol (TCP), Virtual LAN (VLAN), User Datagram Protocol (UDP), and routing protocols such as Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Internet Group Management Protocol (IGMP), and Protocol Independent Multicast (PIM).

To simulate Internetworks, click on New Simulation and then click on Internetworks.

5.1.1 Internetworks Examples

To simulate the Examples for different types of Internetworks

1. Go to the NetSim UI and click **Examples**.

The Example Simulation pane appears at the right.

2. Click the **Internetwork** example you wish to simulate. NetSim UI loads the example.

5.1.2 Internetwork Documentation

To view help documentation users can either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to Internetworks in examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Internetworks Experiments in NetSim
- Reference Documents
- Latest FAQ available online

5.2 Legacy Networks

Legacy networks cover older generation protocols which are rarely used today and not part of the TCP/IP protocol suite. With the advent of TCP/IP as a common networking platform in the mid-1970s, most legacy networks are no longer used.

NetSim Legacy Network library cover Pure Aloha and Slotted Aloha.

ALOHA is a protocol that was developed at the University of Hawaii and used for satellite communication systems in the Pacific. ALOHA protocol was designed to send and receive messages between multiple stations, on a shared medium. Slotted ALOHA is improvised version of pure ALOHA designed to reduce the chances of collisions when sending data between the sender and the receiver.

To simulate Legacy Networks, click on New Simulation and then under Legacy networks click on either Pure Aloha or Slotted Aloha

5.2.1 Legacy Networks Examples

To simulate Pure ALOHA and Slotted ALOHA Examples:

1. Go to the NetSim UI and click **Experiments**.
   The Experiment Simulation pane appears at the right.
2. Click the **Legacy Network** example you want to simulate. NetSim UI loads the example.

5.2.2 Legacy Network Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to Legacy Networks in experiments. The help documentation explains the following:

- Introduction
- Simulation GUI
- Legacy Networks Experiments in NetSim
- Note: Release on Unsupported Basis
- Latest FAQ available online

5.3 Cellular Networks

A cellular network (also known as a mobile network) is a communication network where the last link is wireless. The network is distributed over land areas called cells. Every cell is served by at least one fixed-location transceiver known as a base station. These cells together provide radio coverage over larger geographical areas. User equipment’s such as mobile phones can communicate even if the user is moving across different cells.

NetSim cellular networks library covers Global System for Mobile communication (GSM) and Code-Division Multiple Access (CDMA).

To simulate Cellular Networks, click on New Simulation and then under Legacy networks click on either GSM or CDMA.

5.3.1 Cellular Networks Examples

To simulate GSM and CDMA Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **Cellular Network** example you want to simulate. NetSim UI loads the example.

5.3.2 Cellular Networks Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next Cellular Networks in examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Featured Examples

- Cellular Networks Experiments in NetSim
- Note: Release on Unsupported Basis
- Latest FAQ available online

5.4 Advanced Routing

NetSim supports the following advanced routing protocols.

- Access Control Lists (ACLs)
- Virtual LAN (VLAN)
- Public IP and Network Address Translation (NAT)

To simulate the above-mentioned routing protocols, click on New Simulation and then Internetworks.

5.4.1 Advanced Routing Examples

To simulate the Examples for different types of Advanced Routing protocols

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **Advanced-Routing** example you wish to simulate. NetSim UI loads the example.

5.4.2 Advanced Routing Documentation

To view help documentation users can either click on “Technology Libraries” under documentation in the home screen or they can click the ‘Book’ link located next to Advanced Routing in examples. The help documentation explains the following:

- Simulation GUI
- Model Features
- Featured Examples
- Advanced Routing Experiments in NetSim
- Reference Documents
- Latest FAQ available online

5.5 MANETs

Mobile Ad-hoc Network (MANET) is an ad hoc network that can change locations and configure itself on the fly. Because MANETS are mobile, they use wireless connections to connect to various networks.

NetSim MANET library covers:

- L3 Routing Protocols – DSR, AODV, OLSR and ZRP
- MAC Layer – IEEE 802.11
- MANET using Bridge_Node (Wired) and Bridge_Node (Wireless)

To simulate MANET, click on New Simulation and then select Mobile Adhoc networks.

5.5.1 MANET Examples

To simulate MANET Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **Mobile-Adhoc-Networks** example you want to simulate. NetSim UI loads the example.

5.5.2 MANET Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next MANET Networks in examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Reference Documents
- Latest FAQ available online

5.6 Wireless Sensor Networks (WSN)

Wireless Sensor Network (WSN) is a group of spatially dispersed sensors that monitor and collect the physical conditions of the environment and transmit the data they collect to a central location. WSNs measure environmental conditions such as temperature, sound, pollution levels, humidity, wind, and so on.

WSN in NetSim is part of NetSim’s IOT library and covers 802.15.4 MAC, PHY with MANET routing protocols.

To simulate WSN, click on New Simulation and then Wireless Sensor Networks.

5.6.1 Wireless Sensor Networks (WSN) Examples

To simulate Wireless Sensor Networks Examples:

1. Go to the NetSim UI and click **Examples**.

The Example Simulation pane appears at the right.

2. Click the **IOT-WSN > Wireless-Sensor-Networks** example you want to simulate. NetSim UI loads the example.

5.6.2 WSN Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to IOT-WSN examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- IOT-WSN Experiments in NetSim
- Reference Documents
- Latest FAQ available online

5.7 Internet of Things

Internet of things (IoT) is a network of object such as vehicles, people, home appliances that contain electronics, software, actuators that are accessible from the public Internet. The objects are embedded with suitable technology and use IP addresses to interact and exchange data without manual assistance or intervention. The objects can also be remotely monitored and controlled.

In NetSim, IOT is modeled as a WSN that connects to the internet via a LowPAN Gateway. WSN for IoT uses the following protocols: AODV and RPL with IPv6 addressing at the L3 layer and 802.15.4 at the MAC & PHY layers. WSN sends data to the LowPAN Gateway which uses a Zigbee (802.15.4) interface and a WAN Interface. The Zigbee interface connects wirelessly to the WSN and the WAN interface connects to the Internet. Additionally, users can also simulate and analyze energy model for IoT.

To simulate IOT, click on New Simulation and then Internet of Things.

5.7.1 Internet of Things (IOT) Examples

To simulate IOT Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **IOT-WSN > Internet-of-Things** example you want to simulate. NetSim UI loads the example.

5.7.2 IOT Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to IOT-WSN examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- IOT-WSN Experiments in NetSim
- Reference Documents
- Latest FAQ available online

5.8 Software Defined Networks (SDN)

Software-defined networking (SDN) is an architecture that makes networks agile and flexible. SDN decouples the network control and forwarding functions. SDN allows you to program your network control and abstracts the physical infrastructure for applications and network services. This approach enables enterprises and service providers to respond quickly to the changing business requirements.

Unlike other technologies, and due to the way SDN works it is not available as a menu item under New Simulation. SDN can be configured when running Internetworks, MANET, IOT, WSN, Cognitive Radio, LTE or VANETs

5.8.1 Software Defined Networks (SDN) Examples

To simulate Software Defined Networks Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **Software-Defined-Networks** example you want to simulate. NetSim UI loads the example.

5.8.2 SDN Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to Software Defined Network examples. The help documentation explains the following:

- About SDN
- SDN in NetSim
- Featured Examples
- Latest FAQ available online

5.9 Cognitive Radio

Cognitive Radio (CR) is an adaptive, intelligent radio and network technology that automatically detects available channels in a wireless spectrum and changes transmission parameters to enable higher levels of communication. Cognitive Radio can be programmed and configured dynamically to use the best wireless channels in its vicinity to avoid user interference and congestion.

NetSim Cognitive Radio module is based on the IEEE 802.22 standard. Additionally, you can connect a Cognitive Radio with Internetwork devices and run all the protocols supported in Internetworks.

To simulate Cognitive Radio, click on New Simulation and then Cognitive Radio Networks

5.9.1 Cognitive Radio Examples

To simulate Cognitive Radio Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **Cognitive-Radio** example you want to simulate. NetSim UI loads the example.

5.9.2 Cognitive Radio Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to Cognitive Radio examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Cognitive Radio Networks Experiments in NetSim
- Reference Documents
- Latest FAQ available online

5.10 LTE/LTE-A

Long Term Evolution (LTE) is a standard for 4G wireless broadband technology that offers increased network capacity and speed to mobile device users. LTE offers higher peak data transfer rates -- up to 100 Mbps downstream and 30 Mbps upstream.

NetSim LTE Library support LTE/LTE-Advanced Networks.

Additionally, you can connect an LTE Network with Internetwork devices and run all the protocols supported in Internetworks.

To simulate LTE/LTE-A networks, click on New Simulation and then select LTE/LTE-A Networks.

5.10.1 LTE Examples

To simulate LTE Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **LTE and LTE-A** example you want to simulate. NetSim UI loads the example.

5.10.2 LTE Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to LTE and LTE-A examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Reference Documents
- Latest FAQ available online

5.11 5G NR

NetSim 5G library features full stack, end-to-end, packet level simulation of 5G NR networks. The 5G library is based on Rel 16/17 of the 3GPP 38.xxx series.

NetSim 5G library models all layers of the protocol stack as well as applications running over the network. This 5G library is architected to connect to the base component of NetSim (and in turn to other components) which provides functionalities such as TCP/IP stack protocols, Wireless protocols, Routing algorithms, Mobility, Output Metrics, Traces etc.

To simulate 5G NR networks, click on New Simulation and then 5G NR.

5.11.1 5G NR Examples

To simulate 5G NR Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **5G NR** example you want to simulate. NetSim UI loads the example.

5.11.2 5G NR Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to 5G NR examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Omitted Features
- 5G NR Experiments in NetSim
- Reference Documents

5.12 VANETs

Vehicular Ad-Hoc Network (VANET) is a subset of a Mobile Ad-Hoc Network or MANET that allows vehicle-to-vehicle and vehicle-to-roadside communications to ensure safe transportation.

To simulate VANET click on New Simulation and then click on VANET.

5.12.1 VANET Examples

To simulate VANET Examples:

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **VANETs** example you want to simulate. NetSim UI loads the example.

5.12.2 VANET Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to VANET examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Reference Documents
- Latest FAQ available online

5.13 Satellite Communication

NetSim satellite library models end-to-end, full stack, packet level communication between terrestrial nodes and Geostationary satellites.

The satellite can be thought of as a relay station. It operates on the bent-pipe (transparent star) principle, sending back to Earth what comes in, with only amplification and a shift from uplink to downlink frequency.

The Satellite MAC layer protocol supported in NetSim is TDMA for forward link and MF-TDMA for return link (based on the DVB S2 standards). The forward link is in the Ku band (12 – 18 GHz) while the return link is in the Ka band (26 – 40 GHz)

To simulate Satellite Communication networks, click on New Simulation and then click on Satellite Comm. Networks

5.13.1 Satellite Communication Examples

To simulate the Examples for different types of Internetworks

1. Go to the NetSim UI and click **Examples**.

   The Example Simulation pane appears at the right.

2. Click the **Satellite-Communication** example you wish to simulate. NetSim UI loads the example.

5.13.2 Satellite Communication Documentation

To view help documentation users can either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to Satellite-Communication in examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Reference Documents

- Latest FAQ available online

5.14 Underwater Acoustic Networks

NetSim's UWAN library enables users to design, simulate and analyze performance of underwater networks that use acoustic communication.

UWAN is architected to interface with NetSim component 2 (Legacy networks) which provides L2 functionality and component 3 (Advanced switching and routing) which provides the L3 static routing functionality.

The UWAN library is available as Component 12 and is currently available only in NetSim standard and NetSim Pro versions. NetSim's protocol source C code shipped along with (standard / pro versions) is modular and customizable to help researchers to design and test their own UWAN protocols.

5.14.1 UWAN Documentation

To view help documentation users can either click on "Technology Libraries" under documentation in the home screen or click the 'Book' link located next to UWAN in examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Reference Documents
- Latest FAQ available online

5.15 TDMA Radio Networks

NetSim TMDA Radio Network module uses TDMA/DTDMA in MAC/PHY along with MANET Routing protocols in Layer 3.

To simulate TDMA Radio Networks, click on **New Simulation → TDMA Radio Networks** and select TDMA/DTDMA in MAC/PHY layer of the devices.

Note: TDMA Radio Network component is available only in NetSim pro version.

5.15.1 TDMA Radio Network Examples

To simulate TDMA Radio Networks Examples:

1. Go to the NetSim UI and click **Examples**. The Example Simulation pane appears at the right.

2. Click the **TDMA Radio Networks** example you want to simulate. NetSim UI loads the example.

5.15.2 TDMA Radio Network Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to TDMA Radio Networks examples. The help documentation explains the following:

- Introduction
- Simulation GUI
- Model Features
- Featured Examples
- Model Limitations
- Latest FAQ available online.

5.16 Non Terrestrial Network Examples

NetSim's NTN Library features standards based simulation of non-terrestrial networks (NTN), allowing users to:

- Model end-to-end, full-stack, packet-level simulation of 5G NTN based LEO/MEO/GEO satellite networks
- Evaluate satellite network performance for various parameter configurations - orbit heights, elevation angle, number of spot beams, frequency reuse factor, etc.

5.16.1 Standards and Architecture

- Based on 3GPP TR 38.821 standards for single satellite simulation
- Supports both transparent (bent pipe) and regenerative (gNB on satellite) architectures
- Configurable orbit heights: LEO, MEO, GEO with adjustable UE elevation angles

5.16.2 Non Terrestrial Network Examples

To view help documentation either click on “Technology Libraries” under documentation in the home screen or click the ‘Book’ link located next to Non Terrestrial Network examples. The help documentation explains the following:

- NetSim 5G NTN Satellite Networks: Overview
- Link budget
- Satellite Antenna Pattern
- Interference Models
- Radio Measurements Log

- Frequency Reuse
- Link Budget with Interference
- Beam Radius Calculations
- Featured Examples
- Limitations and assumptions

5.17 Network Emulator Add On

A network simulator mimics the behavior of networks but cannot connect to real networks. NetSim Emulator enables users to connect NetSim simulator to real hardware and interact with live applications.

- NetSim emulator is an IP based, data plane, flow-through emulator. This means:
- It can interact with IP based devices.
- It can emulate data plane functionality and not control plane functionality.
- The source and destination for traffic should be external. A virtual device within NetSim cannot be a source or sink for traffic.

5.17.1 Emulation Library Documentation

To view help documentation either click on “Technology Libraries” under documentation in the home screen, The Emulation documentation explains the following:

- Introduction
- Emulation Set-up
- Model Features
- Featured Examples
- Trouble shooting
- Latest FAQ available online.

6 Applications (Network Traffic Generator)

Applications are the sources of traffic in the network. This traffic is modeled as individual packets. These packets flow from the source to the destination over the designed network. As it flows through the network, depending on the devices, link bandwidths and networking protocols, the packets would experience network effects such as delay, error, loss etc.

Applications are generally parameterized in terms of packet size, inter-packet arrival time, priority, transport protocol running below etc. Therefore, each application has its own distinctive traffic pattern and creates its own unique load on the network.

Different applications have differing levels of complexity. Some applications are used to quickly model basic requirements while in other cases parameters can be accurately modeled to carefully reproduce real world characteristics. For example, if the goal is to analyze protocol behavior, then using a simple CBR application (that generates a certain number of packets every second of a fixed size) would suffice.

NetSim allows users to model and simulate different types of applications.

1. CBR
2. Custom
3. Database
4. FTP
5. Email
6. HTTP
7. Video
8. Voice
9. Sensor App
10. Erlang Call
11. BSM
12. Interactive Gaming
13. Emulation (available only if Emulator Add-on is licensed)

To set up the application click on **Set Traffic** from the top ribbon as shown below Figure 6-1.

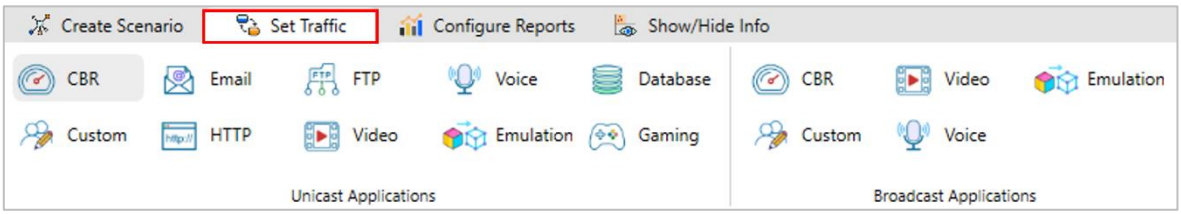


Figure 6-1: Types of applications available in Set Traffic

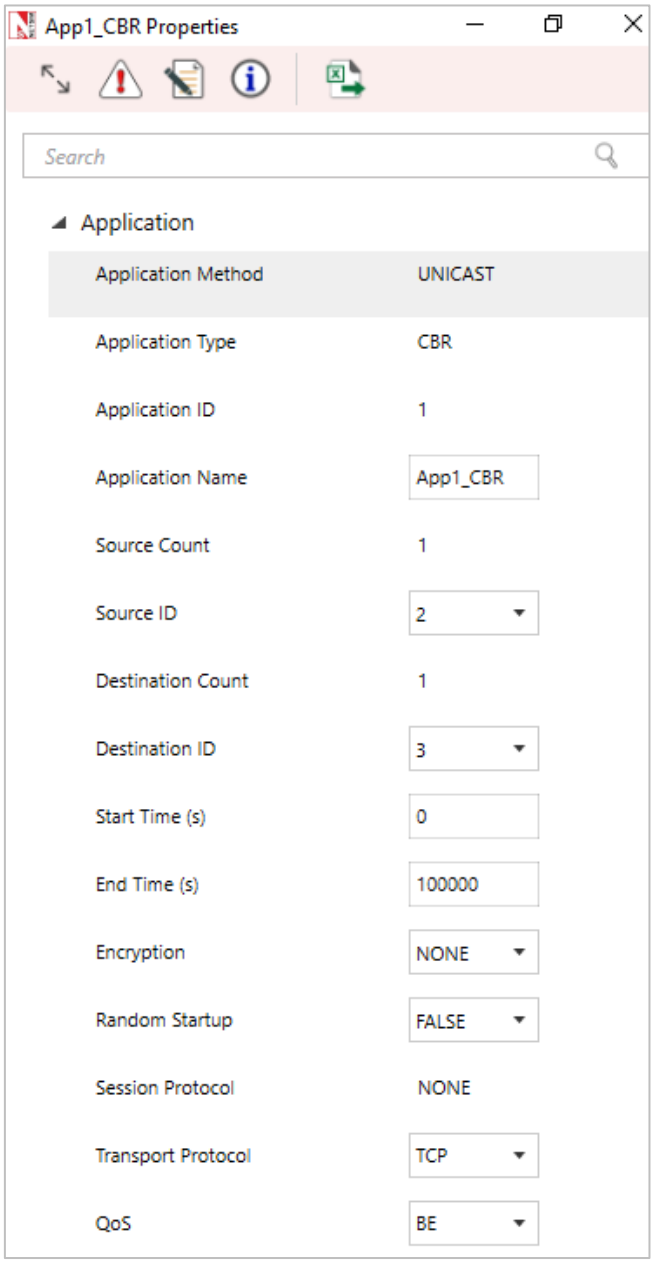


Figure 6-2: Application Configuration Window

These application models have default values set, for the various application properties, to model standard application behavior. Users can modify the parameters to model their own applications.

6.1 Common properties for all applications

Application Method: It specifies the type of Application method Unicast/Multicast/Broadcast.

Application Type: It specifies the type of application such as CBR, Custom, Peer to Peer, Email, HTTP, FTP, Voice, Video, Database, Erlang Call, Sensor App, BSM, and Emulation.

Application ID: This property represents the unique identification number of the application.

Application Name: It specifies the name of the application.

Source Count: This property represents number of sources for the application. Voice, Video, FTP, Database and Custom applications have only one source.

Source ID: This property represents the unique identification number of the source.

Destination Count: This property represents number of destinations for the application. Voice, Video, FTP, Database and Custom applications have only one destination.

Destination ID: This property represents the unique identification numbers of the destination.

- For **Unicast** Applications, users can select the ID of a device in the network as the Destination ID.
- For **Broadcast** Applications, the Destination ID, is set to '0'.

Start time: This property represents the start time of the application in seconds.

End time: This property represents the end time of the application in seconds.

For example, if Start time is 1s and end time is 10s then application starts generating traffic at the 1st second and stops at the 10th second.

Encryption: Encrypts Application packet payload using algorithms such as AES, DES, XOR and TEA. The effect of encryption can be analyzed by enabling Wireshark option in either the source or the destination devices. Refer Section 8.5 on "Packet Capture and Analysis Using Wireshark" for further details.

In NetSim the packet size remains constant when encrypting using these algorithms. Therefore, using different encryption models will not have any impact on the network performance metrics that NetSim outputs. NetSim does not perform decryption of the packet at the receiver end since it does not have any impact on the performance metrics generated.

Random Startup: If random start-up is set to true, the application will start at a random time between the start time and 100 milliseconds after the start time. Having a random start-up time provides more realism to the model, as in the real world, all applications do not necessarily start at time = 0.

QoS: NetSim provides QoS differentiation for the different types of applications through four defined scheduling service types, also called QoS classes as shown below Table 6-1.

QoS Class	Description	Priority
UGS - Unsolicited Grant Service	The UGS scheduling service type is designed to support real-time data streams consisting of fixed-size data packets issued at periodic intervals.	High
rtPS - Real-time Polling Service	The rtPS scheduling service type is designed to support real-time data streams consisting of variable-sized data packets that are issued at periodic intervals. This would be the case,	Medium

	for example, for MPEG (Moving Pictures Experts Group) video transmission.	
ertPS - Extended real-time Polling Service	The ertPS is a scheduling mechanism that builds on the efficiency of both UGS and rtPS. UGS allocations are fixed in size, ertPS allocations are dynamic. The ertPS is suitable for variable rate real-time applications that have data rate and delay requirements.	Normal
nrtPS - Non-real-time Polling Service	The nrtPS is designed to support delay-tolerant data streams consisting of variable-size data packets for which a minimum data rate is required. The standard considers that this would be the case, for example, for an FTP transmission.	Low
BE - Best Effort	The BE service is designed to support data streams for which no minimum service guarantees are required and therefore may be handled on a best basis.	Low

Table 6-1: Different QoS classes with Description and Priority in NetSim

Priority: The priority is automatically set based on the QoS class set by the user. Depending on the scheduling algorithm the router would process packets, with different priorities, differently.

Transport Protocol: This parameter is newly added to the Applications window where by default it selects the Transport Layer Protocol (either TCP or UDP) depending on the application that is set by the user.

Note: Users can also change the value of this parameter according to the transport protocol they intend to run a particular application.

6.2 Application Types

Application Type	Properties	Units	Description
CBR – Constant bit Rate	**Packet size** (Constant distribution) – A fixed packet size	bytes	Packets of constant size are generated at constant inter arrival times. The generation times would be as follows: Packet 1: Application start time. Packet 2: Packet 1 + Interarrival Time Packet 3: Packet 2 + Interarrival Time .... Packet (n+1): Packet n + Interarrival Time Ends at Application end time.
	Inter Arrival Time (Constant distribution) – A fixed time gap between two successive packets	µs	
Custom	**Packet size** (Constant, Exponential, Uniform and Normal distribution) – Packet sizes are drawn from the respective distribution	bytes	It is user defined application where the packet sizes and inter-packet arrival times can be fixed (constant distribution) or can be a random variable (exponential, uniform or normal distribution)
	Inter Arrival Time (Constant, Exponential, Uniform and	µs	

	Normal distribution) – The time gap between two successive packets are drawn from the respective distribution		
Email	**Email send/receive** Represents the rate at which emails are sent/receive	-	Email is a client-server configuration, not a source-destination configuration. Both sides can send and receive. E.g., Outlook, Apple mail, Gmail etc.
	Duration (Constant, Exponential distribution) Time between two successive emails	Seconds	
	Email size (Constant, Exponential distribution) Size of an email	Bytes	
HTTP – Hyper Text Transfer Protocol	**Inter Arrival Time** (Constant, Exponential distribution). It is the time gap between two successive HTTP requests	seconds	HTTP is a request-response application; it uses a client-server configuration, not a source-destination configuration. The client node sends a page request to the server, and the server responds with the pages (whose size is in bytes). HTTP utilizes TCP to transfer its information between computers (usually Web servers and clients). TCP should mandatorily be set as the transport layer protocol. In the application metrics as part of an HTTP application, users can see two rows of metrics, one corresponding to requests from the client to the server and the other corresponding to the replies from the server to the client
	Page size (Constant, Exponential distribution) It is the size of each page	bytes	
	Page count – Represents the number of pages	-	
FTP – File Transfer Protocol	**File size** (Constant, Exponential distribution) – It is the size of the file	bytes	It is a standard network protocol used for the transfer of files between a client and server Note: Devices must have TCP enabled as the transport layer protocol. E.g., FileZilla The generation times would be as follows: File 1: Application start time. File 2: File 1 + Interarrival Time File 3: File 2 + Interarrival Time .... File (n+1): File n + Interarrival Time Ends at Application end time The files are in-turn fragmented into packets during the simulation. Users can generate one file by setting $Application\ End\ Time < (Application\ Start\ Time + File\ Interarrival\ Time)$
	File Inter Arrival Time size (Constant, Exponential distribution) – It is the gap between two files	seconds	
Database	**Transaction size** (Constant, Exponential distribution) - It	bytes	A database application is a computer program whose primary purpose is

	represents the size of each transaction		entering and retrieving information from a computerized database. E.g., MS Excel, MySQL etc.
	Transaction Inter Arrival Time (Constant, Exponential distribution) – It is the time gap between two successful transactions	µs	
Voice	**Packet size** (Constant, Exponential) – It is the size of the packet	bytes	It allows users to configure voice application between client and server. Note: Distribution is constant only for all codec types except custom. E.g., Skype
	Packet Inter Arrival Time (Constant, Exponential distribution) - It is the gap between two successful packets	µs	
	Service type – CBR, VBR	-	
	Suppression models available for VBR – Deterministic, Markov chain	-	
	Success ratio - Sets the ratio of the packets that are not silenced during VBR calls	%	
Video	**Model Type** – Independent Gaussian First order dependent gaussian H_261, H_263, MPEG1_Low_Res, MPEG1_High_Res, MPEG2_Low_Res, MPEG2_High_Res, Buffered video streaming 1 Buffered video streaming 2 Buffered video streaming 3 Buffered video streaming 4 Buffered video streaming 5 Buffered video streaming 6	-	It allows users to configure video application between client and server. E.g., – Skype
Erlang Call	**Packet size** (Constant, Exponential distribution) – It is the size of the packet	bytes	The erlang is a unit of traffic density in a telecommunications system. One erlang is the equivalent of one call The Distribution is constant only for all codec types except custom
	Packet Inter Arrival Time (Constant, Exponential distribution) - It is the gap between two successful packets	µs	
	Call duration (Constant, Exponential distribution) – It is the duration of each call	seconds	
	Call Inter Arrival Time (Constant, Exponential distribution) - It is the gap between two successful calls	seconds	
	Service type – VBR, CBR	-	
	Suppression model available for VBR – Deterministic, Markov chain	-	
	Success ratio - Sets the ratio of the packets that are not silenced during VBR calls	%	

Sensor App	**Packet size** (Constant, Uniform and Normal distribution) – It is the size of the packet	bytes	Used to create application between two sensors. E.g., Smart home, Smart water etc.
	Packet Inter Arrival Time (Constant, Uniform and Normal distribution) - It is the gap between two successful packets	µs	
BSM – Basic safety message	**Packet size** (Constant, Uniform and Normal distribution) – It is the size of the packet	bytes	The BSM Application class sends and receives the IEEE 1609 WAVE (Wireless Access in Vehicular Environments) Basic Safety Messages (BSMs). The BSM is a 20-byte packet that is generally broadcast from every vehicle at a nominal rate of 10 Hz. Note: Available only with VANET component. E.g., Traffic management
	Packet Inter Arrival Time (Constant, Uniform and Normal distribution) - It is the gap between two successful packets	µs	
Interactive Gaming	**Uplink Inter Arrival Time Constant distribution -** A fixed time gap between two successive packets	µs	Interactive Gaming is based on standard 3GPP R1-070674. It is a two-way application and involves UL and DL packet transfers between a client and a server. The transport protocol used in this application is UDP. Uplink Inter Arrival Time (ms) is constant value and is a user input $[1, 100]$. The generation times for Uplink gaming network would therefore be as follows: Packet 1: Application start time. Packet 2: Packet 2 + Interarrival Time Packet 3: Packet 3 + Interarrival Time ... Packet (n+1): Packet n + Interarrival Time Ends at Application end time. Downlink Inter packet arrival times are calculated using Fisher-Tippett distribution whose PDF is: $f_x = \frac{1}{b} e^{-\left(\frac{x-a}{b}\right)} e^{\left(-e^{-\left(\frac{x-a}{b}\right)}\right)}$ where, a and b are constants. $a\ (ms) = [20, 500]$, $b(ms) = [4, 25]$ and $b \leq \frac{a}{3}$. Uplink and Downlink Packet-Size is calculated using Fisher-Tippett distribution whose PDF is:
	Uplink Packet size Fisher-Tippett distribution - The time gap between two successive packets is drawn from the Fisher-Tippett distribution	Bytes	
	Downlink Inter Arrival Time Fisher-Tippett distribution - The time gap between two successive packets are drawn from the Fisher-Tippett distribution	µs	
	Downlink Packet size Fisher-Tippett distribution - The time gap between two successive packets is drawn from the Fisher-Tippett distribution	Bytes	

			$f_x = \frac{1}{b} e^{-\left(\frac{x-a}{b}\right)} e^{\left(-e^{-\left(\frac{x-a}{b}\right)}\right)}$ where, a and b are constants. $a\ (B) = [20, 500],\ b(ms) = [4, 100]$ and $b \leq \frac{a}{3}$
Emulation	**Source Real IP** - Specifies the real IP Address of source device in Emulation **Source Port** - Specifies the Port no used for transmission by Application running in source device **Destination Real IP** - Specifies the real IP Address of destination device in Emulation **Destination Port** - Specifies the Port no used for reception by Application running in destination device	-	NetSim Emulation application enables users to connect NetSim simulator to real devices and interact with live applications. Note: Will be present only when Emulator Add-on is licensed

Table 6-2: Parameters used in modeling different application types.

6.2.1 Voice Models

Codec stands for Coder-decoder. Codecs are devices which encode / decode digital data streams. Codec is the component of any voice system that translates between analog speech and the bits used to transmit them. Every codec transmits a burst of data in a packet that can be reconstructed into voice.

Various voice codecs are available in NetSim to choose from. Packet size and Inter-arrival time value will vary depending on the codec value chosen.

- **G.711:** G.711 is a Pulse code modulation (PCM) of voice frequencies on a 64-kbps channel. G.711 uses a sampling rate of 8,000 samples per second. Non-uniform quantization with 8 bits is used to represent each sample, resulting in a 64-kbps bit rate.
- **G.729:** The G.729 speech codec uses audio data compression algorithm and compress the data at bit rates that vary between 6.4 and 12.4 kbps. Coding of speech at 8 kbps using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP).
- **G.723:** G.723 is an ITU standard for speech codecs that uses the ADPCM method and provides good quality audio at 24 and 40 Kbps.
- **GSM-FR:** GSM–Full Rate (GSM-FR). The codec operates on each 20ms frame of speech signals sampled at 8 KHz and generates compressed bit-streams with an average bit-rate of 13 kbps. The codec uses Regular Pulse Excited – Long Term Prediction – Linear Predictive Coder (RPE-LTP) technique to compress speech.

- **GSM-EFR:** GSM enhanced full rate speech codec is a speech coding standard that was developed in order to improve the quite poor quality of GSM-Full Rate (FR) codec. Working at 12.2 kbps the EFR provides wire like quality in any noise free and background noise conditions. The EFR 12.2 kbps speech coding standard is compatible with the highest AMR mode (both are ACELP).
- **CELP**: Code excitation linear prediction. This model has a packet size of 18 B and inter-packet arrival time of 30 ms.
- **MELP**: Mixed excitation linear prediction. This model has a packet size of 8 B and an inter-packet arrival time of 22.5 ms.
- **CUSTOM**: It is similar to the CUSTOM application type explained in the table above.

Codec	Packet Size (B)	Inter Arrival Time (µs)	Average Bit Rate (kbps)
G.711	160	20000	64
G.729	20	20000	8
G.723	24	30000	6.4
GSM-FR	33	20000	13.2
GSM-EFR	31	20000	12.4
CELP	18	30000	4.8
MELP	8	22500	2.84
CUSTOM	1460	20000	584

Table 6-3: Average bit rate for various voice codecs (CBR Service)

6.2.2 Video Models

Introduction to Video Models

A digital video source (e.g., a movie stored on a computer disk, or a live video from a teleconference) essentially comprises a sequence of images, called frames. Each frame is a digital image comprising an array of pixels. A video source is characterized by the frame rate, expressed in frames per second (fps), and the number of pixels per frame (ppf). Typical values of the frame rate are 30, 50, and 60, whereas typical values of pixels per frame are in the range of 10^5 to 10^6.

Each pixel is encoded into a number of bits to represent the intensity and colour at that point in the image. If the “raw” bits for all the pixels are put together, the number of bits per frame become very large, and it becomes impractical to handle digital video on packet communication networks. Various intraframe and interframe coding techniques are used to reduce the number of bits that need to be sent for each frame.

6.2.2.1 Video Models in NetSim

In NetSim the following video models are available:

1. Basic bits per pixel models:

- Independent Gaussian (termed as continuous normal VBR prior to v13)
- First order dependent Gaussian (termed as Continuous State Autoregressive Markov prior to v13)

2. Video Codec Models: H.261, H.263, MPEG1 Low Res, MPEG1 High Res, MPEG2 Low Res, MPEG2 High Res
3. Buffered Video Streaming Model: Options BV1 through BV6

Note: Quantized State Continuous Time Markov and Simple IPB Composite Model were discontinued from v13 onwards. If older config files with these entries are imported, then NetSim would automatically replace them with the Independent gaussian model with default parameters set.

Basic bits per pixel models

Since, for a given video, the number of pixels per frame remains constant from frame to frame, it is convenient to consider the number of bits per pixel (obtained by dividing the number of bits generated for a frame by the number of pixels). This measure is also illustrative as it permits comparison between the number "raw" bits generated for each pixel, and the average number of bits per pixel after data compression. This results in a sequence of bits per pixel, say b_k, where k is the sequence number of the frame. For simulating a video stream in a communication network simulation, we need a statistical model of the sequence b_k, k=0,1, 2, ⋯ The bits emitted for each frame (obtained by multiplying the generated bits per pixel by the number of pixels per frame; we denote this sequence by B_k) are then packetized, thereby yielding a sequence of packets which are then emitted into the network for transport.

- **Independent Gaussian**: This simple model uses independent samples from a Gaussian (or Normal) distribution to generate the number of bits per pixel generated for each frame. In this simplest model, the number of bits per pixel is not necessarily an integer (hence, the term "continuous"), and the number of bits in successive frames are assumed to be statistically independent. Hence, this model has just two parameters
    - μ (bits): the mean number of bits per pixel. NetSim range [0.01, 1.00]
    - σ (bits): the standard deviation of the number of bits per pixel. NetSim range [0.01, 1.00]

  The data generation rate (in bits per second) for the video application can be calculated by $\bar{B} = fps \times ppf \times \mu$, and $Var(B) = fps^2 \times ppf^2 \times \sigma^2$.
- **First order dependent Gaussian:** This model incorporates the autocorrelation between the frames. The number of bits generated for each frame is not necessarily an integer, but the number of bits per pixel in successive frames are modeled as a first-order autoregressive process driven by an independent Gaussian sequence (which is itself independent from frame to frame). Thus, starting with bits per pixel b_0, the successive b_k are generated as follows:

$$b_k = a\, b_{k-1} + b\, w_k$$

where

- a and b are parameters of the autoregressive process, and $w_k, k \geq 1$, is an independent sequence of independent random variables. It is important to note that the b is positive and $|a| < 1$, and w_k is an independent Gaussian sequence with mean η and variance 1.
- With the above conditions on a and b, the sequence b_k (and, therefore, the sequence B_k) has a steady state, with $\bar{b} = \left(\frac{b}{1-a}\right)\eta$ and $Var(b) = \left(\frac{b^2}{1-a^2}\right)$. The steady state variance and standard deviation of the sequence B_k can be obtained by recalling that $B_k = fps \times ppf \times b_k$; i.e., $\bar{B} = fps \times ppf \times \left(\frac{b}{1-a}\right)\eta$, and $Var(B) = (fps \times ppf)^2 \times \left(\frac{b^2}{1-a^2}\right)$.

Video Codec Models

The list of video codec models and their parameters is provided in table below

Model Type	Packet Size (B) (Exponentially distributed)	Inter packet arrival time (ms)	Frames Per second	Average Bit Rate (Mbps)
H.261	160	20	50	0.064
H.263	160	20	50	0.064
MPEG1_Low_Res	2500	20	50	1
MPEG1_High_Res	7500	20	50	3
MPEG2_Low_Res	7500	20	50	3
MPEG2_High_Res	37500	20	50	15

Table 6-4: The input parameters used in NetSim for the various Video codecs and the resultant average bit rate

Buffered Video Streaming Models

The Buffered Video Streaming Models are based on IEEE 802.11-14/0571r12 standard. The video traffic from Video Source to Video Receiver is generated as follows.

- The packet size (bytes) is generated per the Weibull distribution which has the following formula.

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

- The parameters to use are specified in Table 6-5.

Model_Type	Avg bit rate	λ	k	Frames per second (fps)
Buffered_Video_Streaming_1	2 Mbps	6950	0.8099	31
Buffered_Video_Streaming_2	4 Mbps	13900	0.8099	31
Buffered_Video_Streaming_3	6 Mbps	20850	0.8099	31
Buffered_Video_Streaming_4	8 Mbps	27800	0.8099	31
Buffered_Video_Streaming_5	10 Mbps	34750	0.8099	31
Buffered_Video_Streaming_6	15.6 Mbps	54210	0.8099	31

Table 6-5: Parameters for the Buffered video streaming models BV1 through BV6. $\boldsymbol{\lambda}$ and k parameters are the scale and shape parameters of the Weibull distribution for determining packet size. The average bit rate is the generation rate for the given values of $\boldsymbol{\lambda}, \boldsymbol{k}$ and $\boldsymbol{fps}$.

- The steps pertaining to adding TCP latency in the AP is not required since NetSim models Wi-Fi and TCP protocol operation.
- The standard does not provide the frame per second values. This was derived from the average bit rate, λ and k.

6.3 Network Traffic Generation Rate for Different Applications

This section explains how the traffic generation rate can be calculated for different types of applications:

CBR and Custom application

$$Generation\ Rate\ (Mbps) = \frac{Packet\ size\ (bytes) * 8}{InterArrivalTime(\mu s)}$$

Example: $Packet\ size\ =\ 1460\ Bytes\ and\ Inter\ arrival\ time\ =\ 20000\ \mu s.$

$$Generation\ rate\ (Mbps)\ =\ \frac{1460 \times 8}{20000}\ =\ 0.584 Mbps$$

Video

The Independent Gaussian model is the simplest of all video models in NetSim. It uses the normal distribution for the generation of bits per pixel. In this model, consecutive packet sizes are independent of each other. The generation rate for video application can be calculated by using the formula shown below:

$$Generation\ Rate\ (bits\ per\ second) =\ fps \times ppf \times bpp$$

where,

$fps\ =\ frames\ per\ second,\ ppf\ =\ pixel\ per\ frame\ and\ bpp\ (\mu)\ =\ bits\ per\ pixel\ (mean)$

Users can set the above-mentioned parameters in the Application Properties.

Example: Frames per second = 20, pixels per frame = 10000, bits per pixel = 0.52 then the generation rate would be

$$Generation\ Rate\ (bps) =\ 20 \times 10000 \times 0.52\ =\ 104000\ bits\ per\ second\ =\ 0.1040\ Mbps$$

Voice, CBR Service

$$Generation\ Rate\ (Mbps) = \frac{Packet\ Size(B) \times 8}{Packet\ IAT(\mu s)}$$

Example: Packet Size = 160 Bytes, Packet Inter Arrival Time = 20000 µs,

$$Generation\ Rate\ (Mbps) = \frac{160\ \times 8}{20000} = 0.064\ Mbps$$

Voice, VBR Service, Deterministic

$$Generation\ Rate\ (Mbps) = \frac{Packet\ Size(B) \times 8}{Packet\ IAT(\mu s)}\ \times\ SuccessRatio$$

Example: Packet Size = 160 Bytes, Packet Inter Arrival Time = 20000 µs, Success Ratio = 90 %

$$Generation\ Rate\ (Mbps) = \frac{160\ \times 8}{20000} \times 90\%\ = 0.0576\ Mbps$$

Voice, VBR Service, Markov chain

$$Generation\ Rate\ (Mbps) = \frac{Packet\ Size(B) \times 8}{Packet\ IAT(\mu s)}\ \times\ SAF$$

Example: Packet Size = 160 Bytes, Packet Inter Arrival Time = 20000 µs, Speech Activity Factor = 0.6.

$$Generation\ Rate\ (Mbps) = \frac{160\ \times 8}{20000} \times 0.6\ = 0.0384\ Mbps$$

Email

$$Generation\ Rate\ (Mbps) = \frac{Email\ size\ (bytes) \times 8}{Duration(\mu s)}$$

Example: Email size = 20000bytes, Duration = 1s.

$$Generation\ rate\ (Mbps) = \frac{20000 * 8}{1000000}\ =\ 0.16\ Mbps$$

HTTP

$$Generation\ Rate\ (Mbps) = \frac{Page\ size\ (bytes) \times 8 \times\ Page\ count}{InterArrivalTime(\mu s)}$$

Example: Page size = 20000 Bytes, Page Count = 2, Inter arrival time = 3s

$$Generation\ rate\ (Mbps) = \frac{20000 \times 8 \times 2}{3000000}\ =\ 0.106\ Mbps$$

FTP

$$Generation\ Rate\ (Mbps) = \frac{File\ size\ (bytes) \times 8}{InterArrivalTime(\mu s)}$$

Example: File size = 100000 Bytes, Inter arrival time = 5 µs

$$Generation\ rate\ (Mbps) = \frac{100000 \times 8}{5 \times 1000000} = 0.16\ Mbps$$

Database

$$Generation\ Rate\ (Mbps) = \frac{Packet\ size\ (bytes) \times\ 8}{InterArrivalTime(\mu s)}$$

Example: Packet size = 10000 Bytes, Inter arrival time = 1000000 µs

$$Generation\ rate\ (Mbps) = \frac{(10000 \times 8)}{1000000} = 0.08\ Mbps$$

BSM

$$Generation\ Rate\ (Mbps) = \frac{Packet\ size\ (bytes) \times\ 8}{InterArrivalTime(\mu s)}$$

Example: Packet size = 20Bytes and Inter arrival time = 1000000µs.

$$Generation\ rate\ (Mbps) = \frac{20 \times 8}{1000000} = 0.00016\ Mbps$$

Erlang Call

1. CBR Service

$$Generation\ Rate\ (Mbps) = \frac{Packet\ Size(B) \times 8}{Packet\ IAT(\mu s)} \times \frac{Call\ Duration}{Call\ Duration\ (s) + Call\ IAT\ (s)}$$

Example: Packet Size = 160 Bytes, Packet Inter Arrival Time = 20000 µs, Call Duration = 60 s, Call Inter Arrival Time = 60 s.

$$Generation\ Rate\ (Mbps) = \frac{160\ \times 8}{20000} \times \frac{60}{60 + 60} = 0.032\ Mbps$$

2. VBR Service, Deterministic

$$Generation\ Rate\ (Mbps) = \frac{Packet\ Size(B) \times 8}{Packet\ IAT(\mu s)} \times \frac{Call\ Duration}{Call\ Duration\ (s) + Call\ IAT\ (s)} \times\ SuccessRatio$$

Example: Packet Size = 160 Bytes, Packet Inter Arrival Time = 20000 µs, Call Duration = 60 s, Call Inter Arrival Time = 60 s.

$$Generation\ Rate\ (Mbps) = \frac{160\ \times 8}{20000} \times \frac{60}{60 + 60} \times 100\% = 0.032\ Mbps$$

3. VBR Service, Markov chain

$$Generation\ Rate\ (Mbps) = \frac{Packet\ Size(B) \times 8}{Packet\ IAT(\mu s)} \times \frac{Call\ Duration}{Call\ Duration\ (s) + Call\ IAT\ (s)} \times SAF$$

Example: Packet Size = 160 Bytes, Packet Inter Arrival Time = 20000 µs, Call Duration = 60 s, Call Inter Arrival Time = 60 s, Speech Activity Factor = 0.5.

$$Generation\ Rate\ (Mbps) = \frac{160\ \times 8}{20000} \times \frac{60}{60 + 60} \times 0.5\ = 0.016\ Mbps$$

Sensor

$$Generation\ Rate\ (Mbps) = \frac{Packet\ size\ (bytes) \times\ 8}{InterArrivalTime(\mu s)}$$

Example: Packet size = 50Bytes and Inter arrival time = 1000000µs.

$$Generation\ rate\ (Mbps) = \frac{50 \times 8}{1000000}\ =\ 0.0004\ Mbps$$

Interactive Gaming Model

1. Uplink

$$Generation\ Rate\ (Mbps) = \frac{Packet\ size\ (bytes) \times 8}{InterArrivalTime(\mu s)}$$

Example: a = 45 B, b = 5.7B, Inter arrival time = 40ms. Then the mean packet size is given by

$$\mathbb{E}(x) = \mathbb{E}(a - b \times \ln(-\ln(R))$$

$$\mathbb{E}(x) = a - b \times \ln\left(-\ln\left(\mathbb{E}(R)\right)\right.$$

Since $R = rand(0,1)$, $\mathbb{E}(R) = 0.5$ and hence,

$$\mathbb{E}(x) = round(45 - 5.7\ln(-ln(0.5)))\ =\ 47.089 \approx 47\ B$$

And since interarrival time is 40ms, the mean generation rate is:

$$\frac{47 \times 8}{40 \times 10^3} = 0.0094\ Mbps$$

2. Downlink

$$Generation\ Rate\ (Mbps) = \frac{Packet\ size\ (bytes) \times 8}{InterArrivalTime(\mu s)}$$

Example: $a\ =\ 120\ B, b\ =\ 36\ B, a\ (IAT) =\ 55ms, b\ (IAT)\ =\ 6ms.$

The mean packet size is given by:

$$\mathbb{E}(x) = \mathbb{E}(a - b \times \ln(-\ln(R))$$

$$\mathbb{E}(x) = a - b \times \ln\left(-\ln\left(\mathbb{E}(R)\right)\right.$$

We know the $\mathbb{E}(R) = 0.5$ and hence,

$$\mathbb{E}(x) = round(120 - 36 \times \ln(-ln(0.5))) = 133.194 \approx 133B$$

The mean Inter Arrival Time is given by:

$$\mathbb{E}(t) = \mathbb{E}(a - b \times \ln(-\ln(R))$$

$$\mathbb{E}(t) = a - b \times \ln(-\ln(\mathbb{E}(R))$$

$$\mathbb{E}(t) = 55000 - 6000 \times \ln(-ln(0.5)) = 57199.077\ \mu s$$

The mean generation rate is:

$$\frac{133 \times 8}{57199.077} = 0.0186\ Mbps$$

6.4 Priority and QoS of Applications

The various application traffic generated in NetSim have the following priority and QoS values as shown below.

Application Type	Priority Value	Priority	QoS Class
Voice – One way Voice – Two way	8 8	High High	RTPS UGS
Video	6	Medium	nRTPS
Database	2	Low	BE
Custom	2	Low	BE
FTP	2	Low	BE
Email	2	Low	BE
HTTP	2	Low	BE

Table 6-6: Priority and QoS of Applications

Note: Priority of “Normal” has a Priority Value of 4 and “nRTPS” QoS Class. Ex: Video over TCP.

Priority will have an impact on network performance when multiple applications with different priorities are configured in a network. These packets will be queued and dequeued from the router buffer based on the priority.

6.5 Capture real applications and simulate in NetSim

Users can capture packets from a live network using Wireshark. This can then be used as an input to NetSim as explained in Section 4 of the *Emulator* technology library user guide.

6.6 Modelling Poisson arrivals in NetSim

Any time you have events which occur individually at random moments, but which tend to occur at an average rate when viewed as a group, you have a Poisson process.

For example, we can estimate that a certain node generates 1200 packets per minute. These packets are randomly generated within a minute, but there are on average 1200 packets per minute. If 1200 packets generated per minute that, on average, one packet is generated every $\frac{60}{1200} = 0.05\,s$. So, let's define a variable $\lambda = \frac{1}{0.05} = 20$ and call it the *rate parameter*. The rate parameter λ is a measure of frequency: the average rate of events (packets) per unit of time (in this case, seconds).

Knowing this, we can ask questions like, what is the probability that a packet will be generated within the next second? What's the probability within the next 10 seconds? There's a well-known function to answer such questions. It's called the cumulative distribution function for the exponential distribution, and it looks like this:

$$F(x) = 1 - e^{-\lambda x}$$

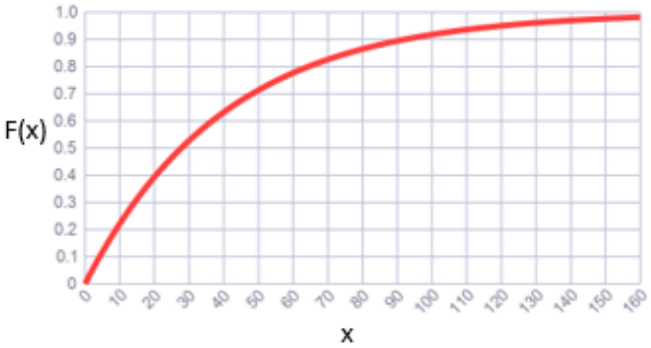


Figure 6-3: Plot for cumulative distribution function for the exponential distribution

Basically, the more time passes, the more likely it is that a packet is generated. The word "exponential", in this context, refers to exponential decay. As time passes, the probability of having *no* packets generated decays towards zero – and correspondingly, the probability of having at least one packet generated increases towards one.

Plugging in a few values, we find that:

- The probability of generating a packet within the next 0.05 seconds is F (0.05) ≈ 0.63
- The probability of generating a packet within 1 second is F (1) ≈ 0.999999998

In particular, note that after 0.05 seconds – the prescribed average time between packets – the probability is F (0.05) ≈ 0.63.

Generating Poisson arrivals in NetSim

We simply write a function to determine the exact amount of time until the next packet. This function should return random numbers, but not the uniform kind of random number produced by most generators. We want to generate random numbers in a way that follows our exponential distribution.

Given that the inverse of the exponential function is ln, it's easy to write this analytically, where R is the random value between 0 and 1:

$$T = -log_e \frac{1-R}{\lambda}$$

Where T is the time at which the next packet is generated.

The simple way of selecting this via the UI is to select exponential distribution for inter-arrival time inside application properties.

6.7 Application Configuration – Special Conditions

1. In a wired network with routers and switches OSPF, spanning tree etc. takes times to converge and hence it is a good practice to set the application start time greater than OSPF convergence time. In general, the applications can start at 20s for smaller networks and should be increased as the size of the network grows.
2. If applications are started before OSPF convergence, then.
   - Packets generated before OSPF table convergence may be dropped at the gateway router.
   - The application may also stop if ICMP is enabled in the router.'
   - If TCP is enabled TCP may stop after the re-try limit is reached (since the SYN packets would not reach the destination)
3. For MANET networks the application start time should be a min of 5s, since the amount of time is required for convergence of OLSR/ZRP.

7 Running Simulation via Command Line Interface

7.1 Running NetSim via CLI

Advanced users can model their simulation via a configuration file (which can be created without the NetSim GUI) and run the simulation from command line. This is typically done in cases where very large networks are to be simulated (it takes too long to create it in the GUI), or to run a series of simulations automatically. The configuration file contains all required information to run the simulation including the network topology, devices, links, traffic, statistics, traces etc. To run Simulation in NetSim through command line interface (CLI), the following steps have to be followed.

Step 1: Note the Application Path

Application path is the current workspace location of the NetSim that you want to run. The default application path will be something like

"C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64" for 64-bit. For more information on NetSim workspace, *Refer Section 4 "Workspaces and Experiments".*

Step 2: Note the IO Path

IO path (Input/output Path) is the directory where the NetSim input files are placed, and output files are written by NetSim during (or after) simulation. In this directory, the input file i.e., the NetSim configuration file (Configuration.netsim), of the simulation scenario should be present. App path and IO path can also be same, i.e., Configuration.netsim can be placed inside the app path (the app path should have write permission). Otherwise, users can create a folder for IO path and Configuration.netsim can be placed inside that folder.

Note: Sample configuration.netsim files are available in the <NetSim installation Directory>/Docs/ Sample_Configurations folder of the NetSim install directory inside the respective protocol folder names.

Step 3: Running NetSim through command line for Simulation.

To run NetSim through command line, copy the app path where NetSimCore.exe is present and paste it in the command prompt.

>cd <app path>

Note: File path should be always added in the command prompt within double quotes. For example,

>cd "C:\Users\PC\Documents\<Your default workspace>\bin_x64"

7.1.1 Running in CLI Mode when using floating licenses

For floating licenses, type the following in the command prompt.

>NetSimCore.exe<space>-apppath<space><app path><space>-iopath<space>

<io path><space>-license<space>5053@<Server IP Address>

Where,

- **<app path>** contains all files of NetSim including NetSimCore.exe. Specifying the app path is optional. NetSim will take the current path as app path if not specified.
- **<iopath>** contains **Configuration.netsim.**
  (**Configuration.xsd** is available in the bin folder of NetSim's current workspace path ***C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64*** for 64-bit (Refer section 7.2.4 to know about configuration.xsd file).
- **5053** is the port number through which the system communicates with the license server i.e. the system in which the dongle is running (for floating license users)
- **<Server IP Address>** is the ip address of the system where NetSim license server is running.

Note: Please contact your network administrator / lab in-charge to know the IP address of the PC where the NetSim license server is running.

The following screenshot is the example of running NetSim through CLI where the ip address of the NetSim license server is 192.168.0.9.

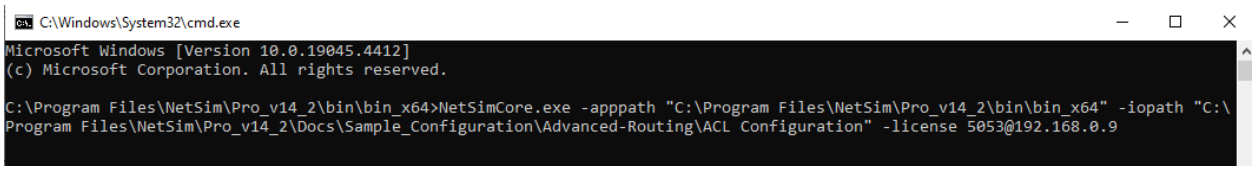


Figure 7-1: Running NetSim through CLI mode for floating license

7.1.2 Running in CLI Mode when using node-locked or cloud licenses

For cloud licenses and node-locked licenses, type the following in the command prompt

>NetSimCore.exe<space>-apppath<space><apppath><space>-iopath<space><io path><space>-license<space><license file path>

Where,

- **<app path>** contains all files of NetSim including NetSimCore.exe
- **<iopath>** contains Configuration.netsim and Configuration.xsd
- **<license file path>** path where the license file is present. This is generally the <NetSim_Installation_Directory>/bin folder.

For E.g. C:\Program Files\NetSim\Pro_v14.4\bin

The following screenshot is the example of running NetSim through CLI for the node locked or cloud license.

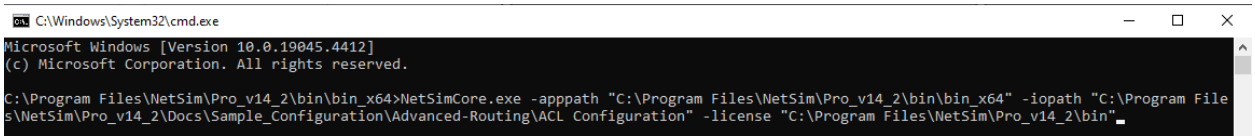


Figure 7-2: Running NetSim through CLI mode for Cloud and Node lock licenses

Once simulation is complete the text files that are requested by the end user in Configuration.netsim will be written in the **<iopath>**.

To know more about the options that are available to run NetSim via CLI, type the following in the command prompt.

>cd <app path>

>NetSimCore.exe –h

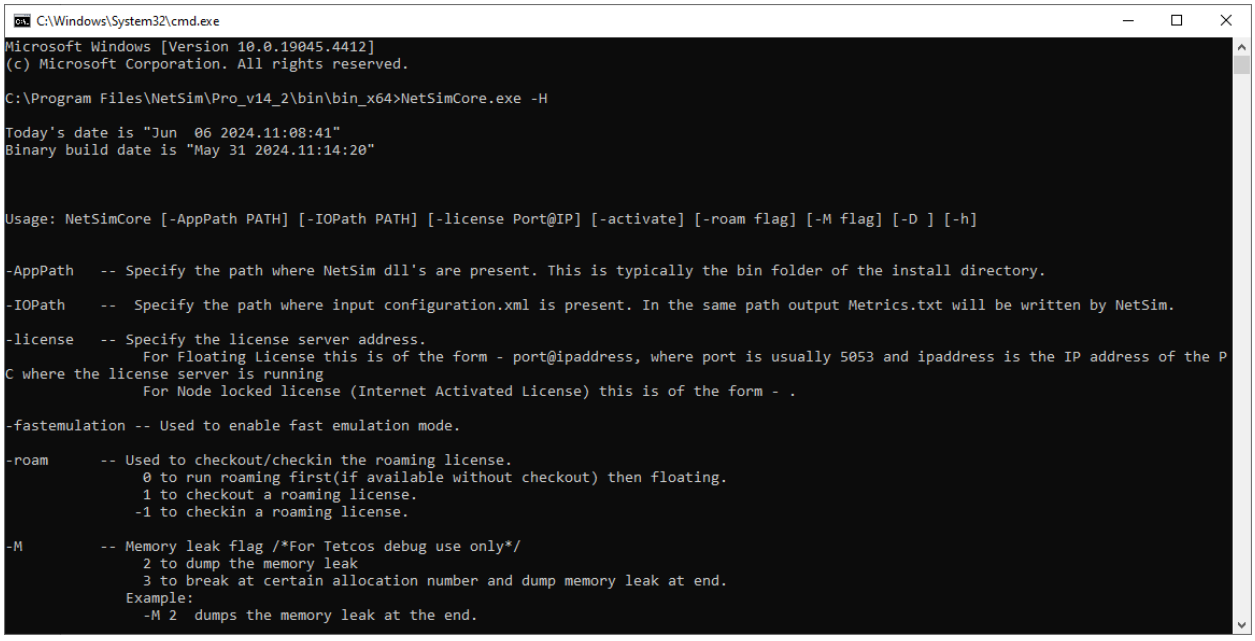


Figure 7-3: More Options available to run NetSim via CLI

7.1.3 Quick edit for copy pastes in CLI mode.

With Quick Edit mode, you can copy text between a command window and Windows-based programs, and you can also paste text into a command window by using a right-click operation. To use Quick edit mode in command prompt users can run the command prompt → **Right Click** the icon in the upper-left corner of the Command Prompt window, and then Click **Properties** →In the options, enable **Quick Edit mode** → and click on **OK**.

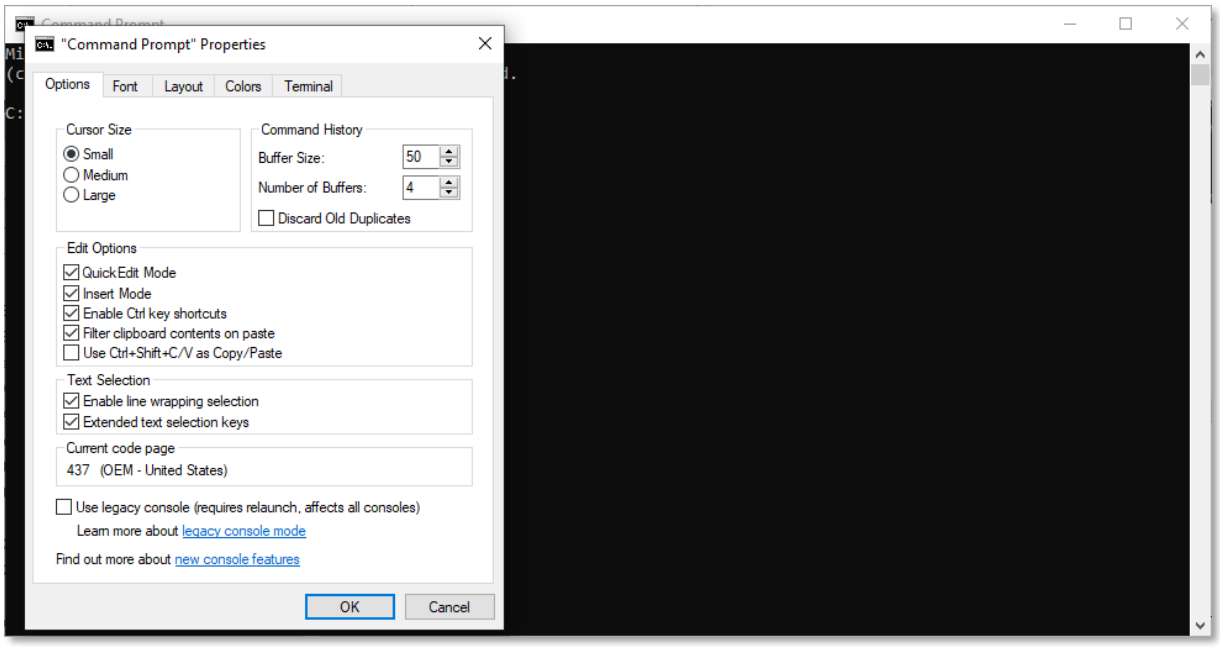


Figure 7-4: Quick edit mode via CLI Running

7.2 Understanding the Configuration.netsim file

When a scenario is created in the GUI, NetSim's UI code write all the details about the devices used and its properties, the links used and their properties, the properties of the environment being used, etc. in the file ***Configuration.netsim***

The simulation engine that contains DLLs and NetSimCore.exe reads this Configuration.netsim, executes the simulation and writes output metrics files. The GUI then displays the metrics based on the text files written by the backend.

In order to run NetSim through command line (CLI), the user must create the Configuration.netsim file furnishing all the details about the devices, links and the environment of the desired scenario.

7.2.1 How to use Visual Studio to edit the Configuration file?

In Visual Studio, XML view provides an editor for editing raw XML and provides *IntelliSense* and *color coding*. After you type the element name and press the CTRL+ SPACE, you will be presented with a list of attributes that the element supports. This is known as "IntelliSense". Using this feature, you can select the options that are required to create the desired scenario.

Color coding is followed to indicate the elements and the attributes in a unique fashion.

The following screenshot displays the Configuration.netsim which is opened through the Visual Studio as shown below Figure 7-5.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TETCOS_NETSIM xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance" noNamespaceSchemaLocation="Configuration.xsd">
  <EXPERIMENT_INFORMATION>
    <VERSION>std</VERSION>
    <NUMBER>14.2.1.14 (64 Bit)</NUMBER>
    <USER>DESKTOP-O0MSHL3\User</USER>
    <NAME>Experiment</NAME>
    <DATE>03/29/2024 2:20 PM</DATE>
    <COMMENT>-</COMMENT>
  </EXPERIMENT_INFORMATION>
  <GUI_INFORMATION>
```

Figure 7-5: Open Configuration.netsim file via Visual Studio

To reformat click on edit→Advanced→Format Document.

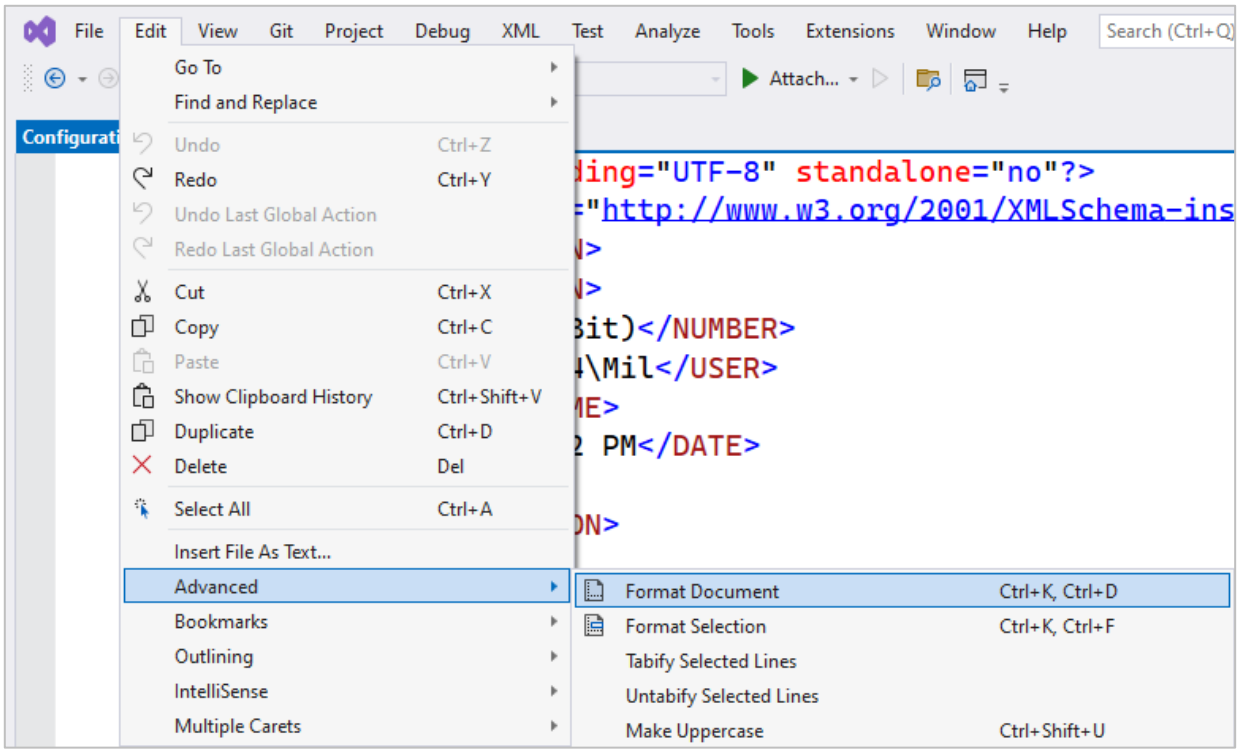


Figure 7-6: Reformat the Configuration.netsim file

7.2.2 Sections of Configuration file

These are the different sections in Configuration.netsim:

- EXPERIMENT_INFORMATION
- GUI_INFORMATION
- NETWORK_CONFIGURATION
- SIMULATION_PARAMETER
- PROTOCOL_CONFIGURATION
- STATISTICS_COLLECTION

EXPERIMENT_INFORMATION:

This section contains the details about the user credentials, such as the user mode (Admin or Exam or Practice), experiment name, date on which the experiment is created and the comments about the experiment. This section plays a significant role while running NetSim through GUI.

GUI_INFORMATION:

This section contains the GUI information like the environment length, view type etc. and the network name which is desired to be run.

NETWORK_CONFIGURATION:

This section is used to configure the devices and the links of the desired network at each layer of the TCP/IP stack. It consists of DEVICE_CONFIGURATION, CONNECTION and APPLICATION_CONFIGURATION. DEVICE_CONFIGURATION configures the devices in the desired network while the CONNECTION configures the links in the desired network and APPLICATION configures the Applications.

SIMULATION_PARAMETER:

Simulation time and seed values are described in this section.

PROTOCOL_CONFIGURATION:

IPV4 and static ARP are enabled or disabled in this section. The text files illustrating the static routing and static ARP can be obtained by enabling the corresponding tags in the Configuration.netsim.

STATISTICS_COLLECTION:

The packet trace and the event trace can be observed in the text files which are created by enabling the tags in this section. The required fields of the packet trace can be enabled in the PACKET_TRACE while the event trace can be enabled in the EVENT_TRACE of this section.

7.2.3 Sample Configuration file

Sample “Configuration.netsim” file will be installed in user system along with the software at <NetSim installed Path>\Docs\ Sample_Configuration\ <Network Technology>.User can open and edit these files using Visual Studio 2015/2017/2019/2022 or any XML editor. The purpose of providing the sample “Configuration.netsim” file is to assist the user in writing a network scenario manually by analyzing the format for that specific network technology.

7.2.4 Configuration.xsd file

Configuration.xsd is an XML schema Definition file which is present in the bin folder of NetSim’s current workspace path

<C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64> for 64-bit.

Configuration.xsd file can be placed inside the <iopath> along with the configuration.netsim file to verify the contents in the configuration.netsim file. This file checks and validates the structure

and vocabulary of a configuration.netsim document against the grammatical rules of the appropriate XML language.

It is not mandatory to place the configuration.xsd file along with the Configuration.netsim file in the iopath. But if it is done, then it will be easier to check & validate changes that are done to the Configuration.netsim file.

7.2.5 NetSim Advanced Plot GUI Without Opening NetSim Application

NetSimAdvancePlot GUI can be launched without the main application by passing necessary arguments to the executable file present in the software installation directory.

The NetSimAdvancePlot binaries can be found in the path

"C:\Program Files\NetSim\Pro_v14_4\bin\net7.0-windows"

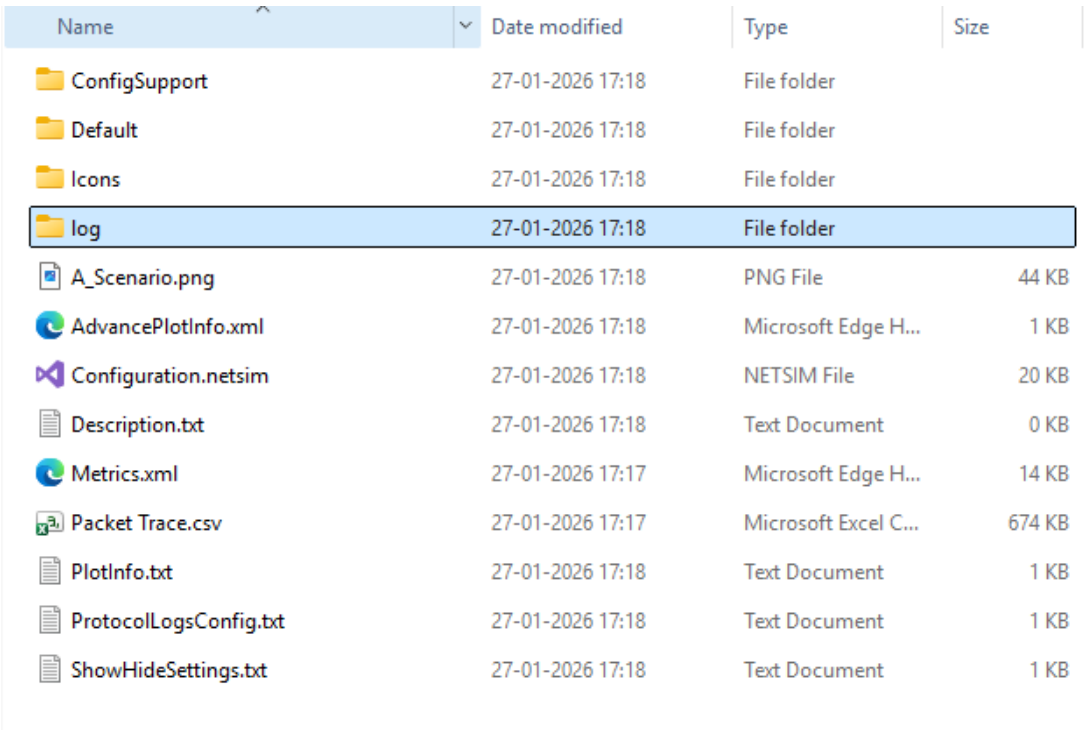
The Plot script files are present in the path

"C:\Program Files\NetSim\Pro_v14_4\Docs\Advanced_PlotScripts"

For example if you want to plot SNR vs Time for a TDMA network from the TDMA Radio Measurements log file,

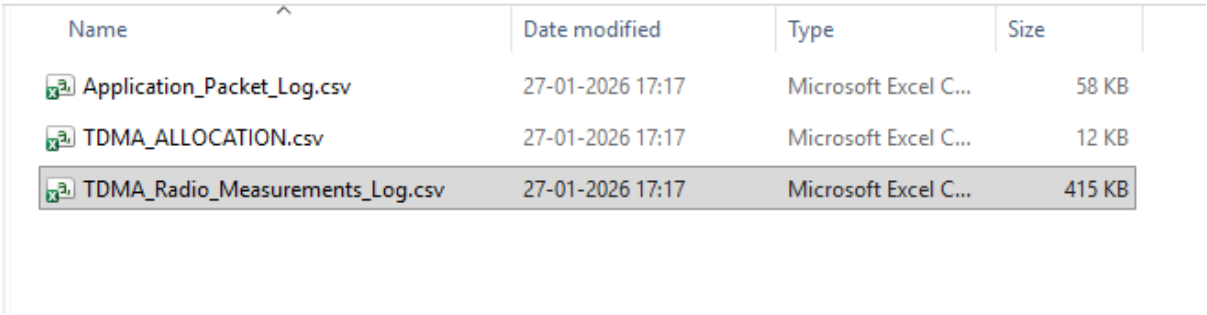
1. Open Command Prompt and call the NetSimAdvancedPlot executable passing two arguments.
   NetSimAdvancedPlot.exe <Log-File> <Plot-Script-File>

The log file can be found in the location where the simulation related files are saved, inside a log folder.



Name	Date modified	Type	Size
ConfigSupport	27-01-2026 17:18	File folder	
Default	27-01-2026 17:18	File folder	
Icons	27-01-2026 17:18	File folder	
log	27-01-2026 17:18	File folder	
A_Scenario.png	27-01-2026 17:18	PNG File	44 KB
AdvancePlotInfo.xml	27-01-2026 17:18	Microsoft Edge H...	1 KB
Configuration.netsim	27-01-2026 17:18	NETSIM File	20 KB
Description.txt	27-01-2026 17:18	Text Document	0 KB
Metrics.xml	27-01-2026 17:17	Microsoft Edge H...	14 KB
Packet Trace.csv	27-01-2026 17:17	Microsoft Excel C...	674 KB
PlotInfo.txt	27-01-2026 17:18	Text Document	1 KB
ProtocolLogsConfig.txt	27-01-2026 17:18	Text Document	1 KB
ShowHideSettings.txt	27-01-2026 17:18	Text Document	1 KB

Figure 7-7: Workspace folder contains Log Folder.



Name	Date modified	Type	Size
Application_Packet_Log.csv	27-01-2026 17:17	Microsoft Excel C...	58 KB
TDMA_ALLOCATION.csv	27-01-2026 17:17	Microsoft Excel C...	12 KB
TDMA_Radio_Measurements_Log.csv	27-01-2026 17:17	Microsoft Excel C...	415 KB

Figure 7-8: Sample Radio measurements.csv file present inside the Log folder.

Eg:

"C:\Program Files\NetSim\Pro_v14_4\bin\net7.0-windows\NetSimAdvancePlot.exe" "C:\Users\leo\Documents\NetSim\Workspaces\Prov14.4.17\TDMA slot allocation\log\TDMA_Radio_Measurements_Log.csv" "C:\Program Files\NetSim\Pro_v14_4\Docs\Advanced_PlotScripts\TDMA Radio Measurements\TDMA_Radio_Measurements_Log_SNR_vs_Time.xml"

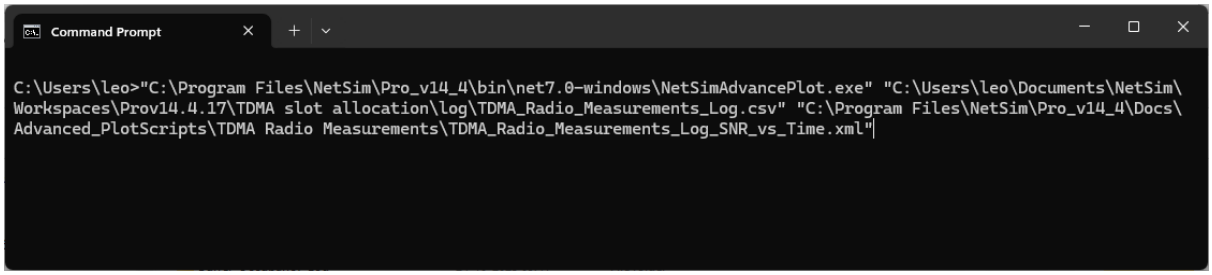


Figure 7-9: Running the ***NetSimAdvancePlot.exe*** through CLI.

2. This will launch the NetSimAdvancedPlots window where you can generate SNR plots for different Tx-Rx pairs as shown below:

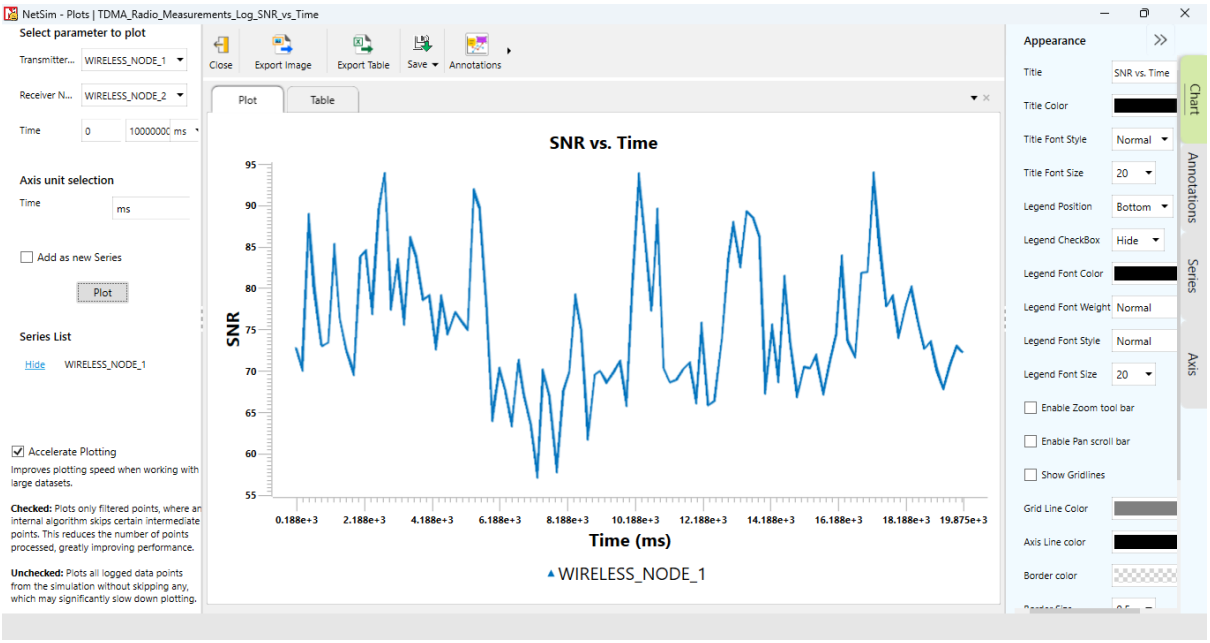


Figure 7-10: SNR Vs. Time plot.

Similarly, for different plots, you can pass the respective file and associated script file as argument to the NetSimAdvancePlot executable and generate plots without launching NetSim GUI.

8 Outputs: Results, Plots and Data Files

8.1 Results Window

The simulation results window in NetSim offers a unified dashboard for convenient analysis. Simulation results include application metrics, link metrics, graphical plots, detailed logs, traces, packet capture for network analysis, and the capability to save plots for future analysis. The tabular presentation includes end-to-end delays, jitter, errors, packets generated/received/collided, route tables, TCP acknowledgments, retransmissions, etc.

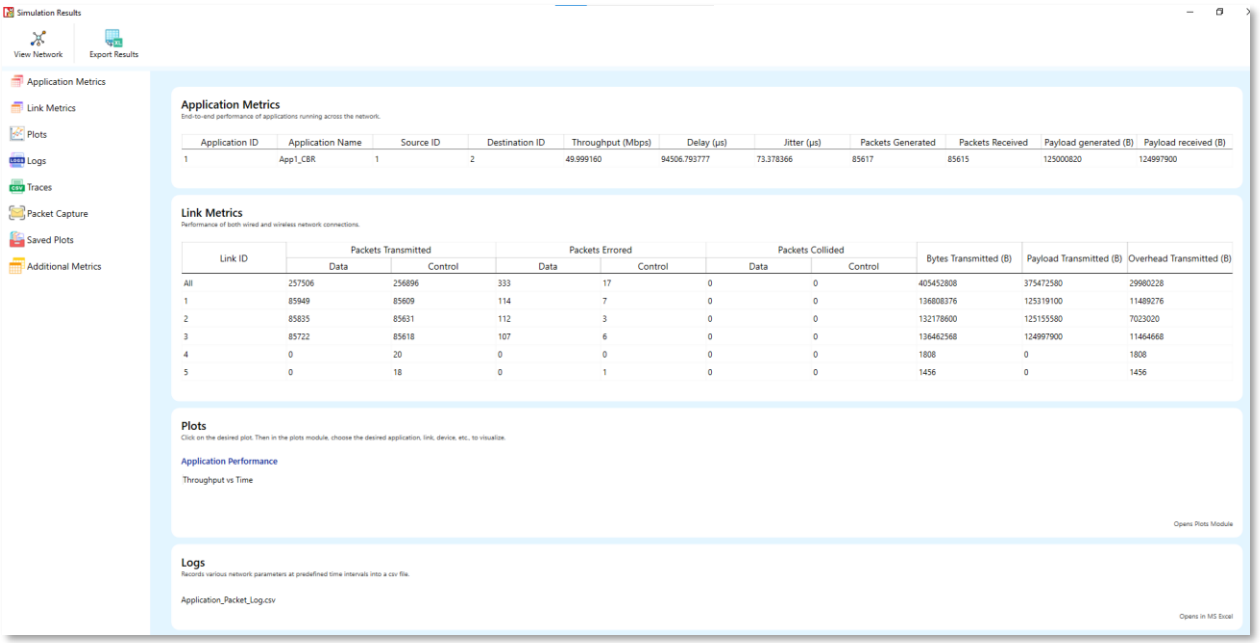


Figure 8-1: Simulation Results Window.

8.1.1 Application Metrics

Displays Application performance metrics.

- **Application Id** - It is the unique Id of the application running at the source.
- **Application Name** - It is unique name of the application running.
- **Source Id** - It is the unique Id of the device running that particular application.
- **Destination Id** - It is the unique Id of the destination device.
- **Packet generated** - It is the total number of packets generated from the source.
- **Packets Transmitted** - It is the total number of packets generated and transmitted from the source.
- **Packet received** - It is the total number of packets received at the destination.
- **Payload Transmitted** - It is the total payload transmitted in bytes. It is equal to the product of 'Packets Transmitted' and 'Packet Size'. This calculation will apply only in case of a

constant packet size (CBR, CUSTOM (constant) etc. In other cases, this should be considered as the sum of the payload of the packets transmitted.

- **Payload Received** - It is the total payload received at the destination in bytes.
- **Throughput** - Total user data (or) payload delivered to their respective destination every second.

  If Simulation Time > Application End Time, then

$$Application\ Throughput(Mbps) = \frac{Total\ payload\ delivered\ to\ destination\ (bytes)\ *\ 8}{Time\ last\ received\ packet\ at\ App\ layer(\mu s) - App\ Start\ Time\ (\mu s)}$$

  If Simulation Time < Application End Time, then

$$Application\ Throughput(in\ Mbps) = \frac{Total\ payload\ delivered\ to\ destination\ (bytes)\ *\ 8}{Simulation\ Time(\mu s)\ - App\ Start\ Time(\mu s)}$$

- **Jitter**

$$Jitter(\mu s) = \frac{TotalPacket\ Jitter\ of\ all\ successfull\ packets}{Total\ Number\ of\ successfully\ recieved\ packets - 1}$$

$$Packet\ Jitter\ (\mu s) = |EndtoEnd\ Delay\ of\ Current\ packet - EndtoEnd\ Delay\ of\ Previous\ Packet|$$

- **Delay** - It is the average amount of time taken (calculated for all successful packets) to reach the destination application layer from when the packet is sent from source's application layer. It is calculated as the APP_IN time at destination minus APP_OUT time at source.

8.1.2 Link metrics

Displays metrics pertaining to each link.

- **Link ID:** It is the unique Id for the link.
- **Link Throughput Graph:** Plots throughput vs. Simulation time.

Formula:

$$Link\ Throughput\ (Mbps) = \frac{Total\ bytes\ transmitted\ over\ the\ link\ (Bytes)\ \times\ 8}{Simulation\ Time\ (\mu s)}$$

The link throughput considers the entire traffic that was sent through a link. It includes data packets and control packets and includes retransmissions, errors, or collisions. This would also include packet flows from multiple applications that may flow through the same link. The calculation is based on the packet size (bytes) at the PHY layer, which would include app layer payload plus the overheads of all layers.

- **Packets Transmitted:** It is the total number of packets transmitted in the link. Along with data packets, it includes protocol control packets like ARP Request, ARP Reply, TCP_ACK,

TCP_SYN, RTS, CTS, WLAN_ACK, OSPF_ HELLO, RIP packets etc. Note that this is a link (PHY layer) level measure, and it is not a MAC layer measure. Therefore, the packets transmitted can be greater than the packets generated when running wireless protocols due to re-tries.

- **Packets Errored:** Total number of packets error in the link inclusive of data and control packets.
- **Packets Collided:** Total number of packets collided in the link including data and control packets. Note: (i) If two packets collide then this counter is incremented by two (once for each packet). (ii) If a single packet collides N times, then this counter is incremented N times
- **Bytes Transmitted:** It is the total number of bytes transmitted in the link. It is equal to the sum of the 'Payload Transmitted' and 'Overhead Transmitted' transmitted in the link.
- **Payload Transmitted:** It is the total payload transmitted in the link.
- **Overhead Transmitted:** It is the total overhead transmitted in the link. It includes the layer-wise overheads and all control packets in the link.

8.1.3 Additional Metrics

Users can observe additional metrics based on the protocols used in the network. Users can click on Additional Metrics icon from the Results dashboard to view the additional metrics table as shown in figure below.

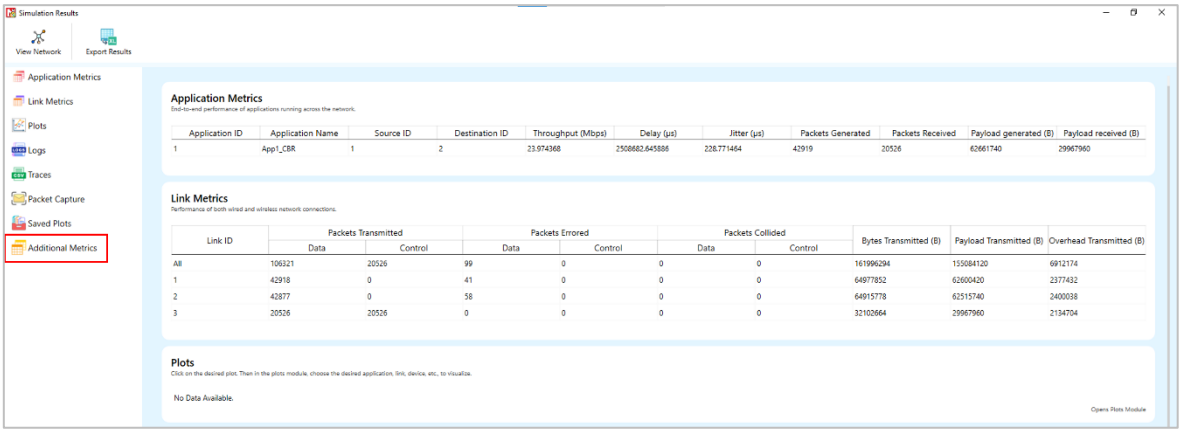


Figure 8-2: Additional Metrics option in Results Dashboard.

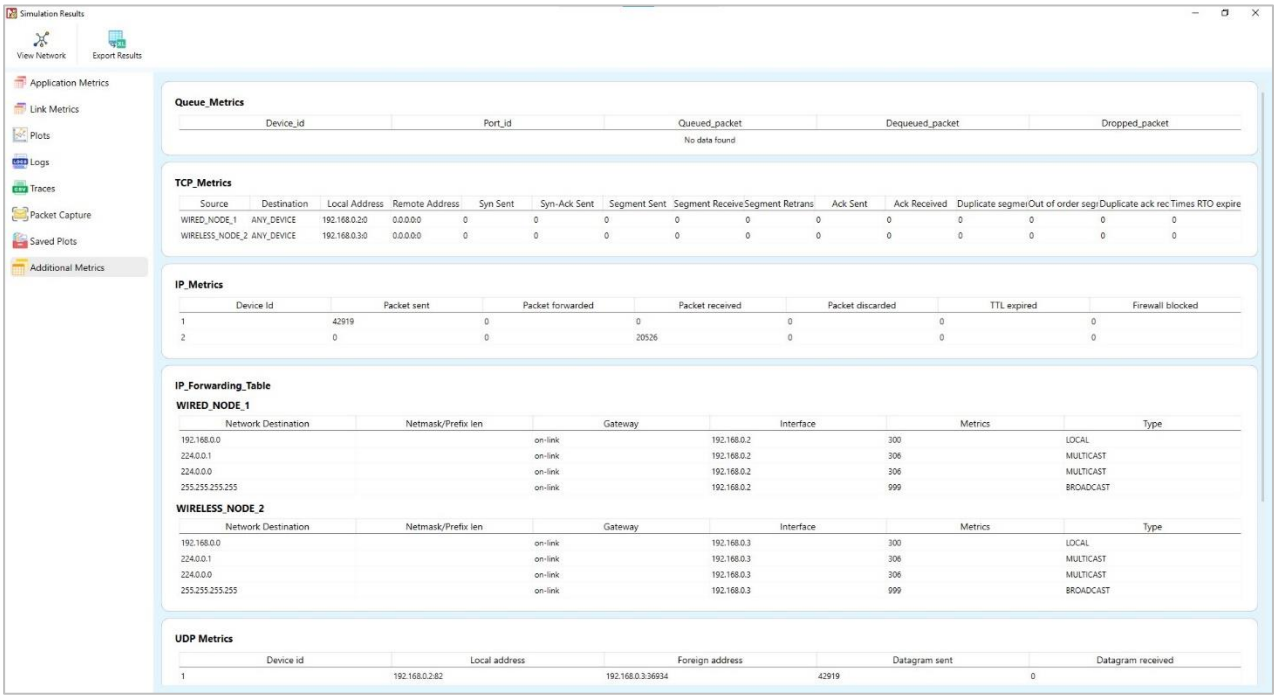


Figure 8-3: Additional Metrics Window

8.1.3.1 Queue Metrics

Displays the values of the queue metrics for the devices containing buffer queue like routers, access points etc.

- **Device Id** - Unique id number of the device.
- **Port Id** - Unique id number of the port of the device. This is also called as interface id.
- **Queued Packet** - Number of packets queued at a particular port of a device.
- **Dequeued Packet** - Number of packets removed from the queue at a particular port of device.
- **Dropped Packet** - Number of packets dropped at a particular port of a device.

8.1.3.2 Protocol Metrics

The Performance metrics tables of protocols such as TCP, UDP, IP, IEEE802.11, LTE, AODV and DSR are provided in the respective technology library documentation.

8.1.3.3 Device Metrics

Displays device related metrics like ARP table, IP forwarding tables. This is also dependent upon the type of network/technology simulated.

IP_Forwarding Table

- **Network Destination** - It represents the Network address of the destination.
- **Netmask/Prefix length** - A 32-bit combination used to describe which portion of an address refers to the subnet and which part refers to the host.

- **Gateway** - It is the IP address of the next-hop router.
- **Interface** - It represents a network connection.
- **Metrics** - It is the value used to choose between two routes.
- **Type** - It represents the type of the network i.e. local/Multicast/Broadcast

Switch MAC Address Table: These metrics will be displayed when we run networks having Switches.

- **MAC Address** - It represents the MAC address of the switch interfaces.
- **Type** - It is the type of the switch.
- **Outport** - It is the output port of the switch.

8.1.3.4 Cellular Metrics

Displayed if GSM or CDMA is running in the network.

GSM/CDMA Metrics. MS Metrics

- **MS Id** - It is the id of the Mobile station.
- **Call Generated** - It is the number of calls generated by a Mobile Station.
- **Call Blocked** - It is the number of calls blocked by a Base station when no channel available.
- **Call Blocking probability** - It is the probability of calls blocked by a base station.
- **Channel request sent** - It is the number of channel requests sent by a mobile station.
- **Call request sent** - It is the number of call requests sent by a mobile station (at source)
- **Call request received** - It is the number of call requests received by a mobile station (at destination)
- **Call accepted** - It represents the number of calls accepted by a mobile station.
- **Call rejected** - It represents the number of calls rejected by a mobile station.
- **Handover request** - It is the number of handover requests sent by a mobile station. Handover refers to the process of transferring an ongoing call or data session from one channel connected to the core network to another channel.
- **Call dropped** - It represents the number of calls dropped by a BS.
- **Call dropping probability** - It represents the probability of number of calls dropped by a BS.

8.1.3.5 Channel metrics

- **BS Id** - It is the Id of a Base Station.
- **Channel Id** - It represents the channel number.
- **Uplink frequency** - It is the uplink frequency of the GSM network to send data from mobile station to base station.

- **Downlink frequency** - It is the downlink frequency of the GSM network to send data from base station to mobile station.
- **Time slot** - It represents the time slot. In GSM network, Frequency band is divided into 200kHz carriers and then each carrier is divided into 8 time slots (0-7).

8.1.3.6 Sensor Metrics (IEEE802.15.4 Metrics)

Displayed if WSN/IOT is running in the network.

- **Device Id** - It represents the Id's of the sensor and LoWPAN Gateway.
- **Packet Transmitted** - It is the number of packets (either data/routing/ZigBee) transmitted by Sensor and LoWPAN gateway
- **Packet Received** - It is the number of packets (either data/routing/ZigBee) received by Sensor and LoWPAN gateway
- **Ack Transmitted** - It is the number of acknowledgements transmitted by a particular device.
- **Ack Received** - It is the number of acknowledgements received by a particular device.
- **CCA Attempt** - It represents the number of Clear channel Assessment attempts at sensors and LoWPAN Gateway used to determine whether the medium is idle or not.
- **Successful CCA Attempt** - It represents the number of successful CCA attempts at sensors and LoWPAN Gateway.
- **Failed CCA** - It represents the number of failed CCA attempts at sensors.
- **Total Backoff Time** - It is the total backoff time obtained. It is the time that sensors have to wait before attempting to access the channel.
- **Average Backoff time** - It is the average backoff time.
- **Beacon Transmitted** - It the total number of beacons transmitted by a LoWPAN Gateway. It transmits network beacons in a beacon enabled mode. If beacon mode is enabled, it follows slotted CSMA/CA algorithm
- **Beacon Received** - It is the total number of beacons received by the sensors.
- **Beacon Forwarded** - It is the total number of beacons forwarded by the sensors.
- **Beacon Time** - It is the total time calculated for beacon transmission at LoWPAN Gateway.
- **CAP Time** - It is the total Contention Access Period obtained during simulation. During this time, sensors compete for channel.
- **CFP Time** - It is the total Contention free period obtained. In CFP, nodes request for guaranteed time slots. If GTS is allocated, nodes can transmit without contention.

8.1.3.7 Battery Model

- **Device Name** - It represents the Name and Id of the Sensor
- **Initial Energy** - It represents the initial energy of the sensors.
- **Consumed Energy** - This is the total energy consumed by the respective sensor.

- **Remaining Energy** - This is the remaining energy of the sensor at the end of the simulation.
- **Harvested Energy** – It's the process of capturing and utilizing ambient energy from the environment to replenish the battery. In NetSim, harvested energy is computed periodically based on user inputs - harvesting current and voltage.

$$Engergy\ Harvested\ =\ Recharging\ current \times\ Voltage \times\ Time$$

- **Transmission Energy** - It is the energy consumed by the respective sensor for transmitting data.

$$TransmissionEnergy\ (mJ) = TransmitCurrent\ (mA) \times Voltage\ (V) \times AmountOfTimeRadioIsInTransmitState(s).$$

- **Receiving Energy** - It is the energy consumed by the respective sensor while receiving data.

$$ReceivingEnergy\ (mJ) = ReceiveCurrent\ (mA) \times Voltage\ (V) \times AmountOfTimeRadioIsInReceiveState(s).$$

- **Idle Energy** - When the sensor is active and ready but not currently receiving or transmitting data packets, it is said to be in an idle state. This metrics calculates the energy consumed by the sensor in idle state.

$$IdleEnergy\ (mJ) = IdleCurrent\ (mA) \times Voltage\ (V) \times AmountOfTimeRadioIsInIdleState(s).$$

- **Sleep Energy** - This is the energy consumed when the respective sensor is in an inactive mode.

$$SleepEnergy\ (mJ) = SleepCurrent\ (mA) \times Voltage\ (V) \times AmountOfTimeRadioIsInSleepState(s).$$

8.1.3.8 CR metrics

Displayed if 802.22 cognitive radio is running in the network.

8.1.3.8.1 Base station Metrics

- **BS Id** - It is the id of a Base Station.
- **Interface Id** - It is the Interface Id of a BS
- **SCH sent** - SCH. It is the number of Superframe Control Headers sent by a BS. SCH carries Base Station's MAC address along with the schedule of quiet periods for sensing, as well as other information about the cell.
- **FCH sent** - It represents the number of Frame Control Headers sent by a BS. It is transmitted as a part of Down Stream (DS) Protocol Data Unit in DS subframe specifies

length of either DS-Map if transmitted or US-Map. It is sent in the first two subchannels of the symbol immediately following the preamble symbol.

- **DSA req received** - It is the number of Dynamic Service Addition requests received by a BS used to create a new service flow.
- **DSA rep sent** - It is the number of DSA replies sent by a BS.
- **DSC req received** - It is the number of Dynamic Service Change requests received by a BS to dynamically change the parameters of an existing service flow.
- **DSC rep sent** - It is the number of DSC replies sent by a BS.
- **DSD req received** - It is the number of Dynamic Service Deletion requests received by a BS to delete an existing service flow.
- **DSD rep sent** - It is the number of DSD replies sent by a BS.
- **CHS req sent** - It is the number of Channel Switch Requests sent by a BS.

8.1.3.8.2 CPE metrics

- **CPE Id** - It represents the Id of Customer Premise Equipment
- **Interface Id** - It represents the Interface Id of the CPE
- **SCH received** - It is the number of Superframe Control Headers received by a CPE.
- **FCH received** - It represents the number of Frame Control Headers received by a CPE.
- **DSA req sent** - It is the number of Dynamic Service Addition requests sent by a CPE.
- **DSA rep received** - It is the number of DSA replies received by a CPE.
- **DSC req sent** - It is the number of Dynamic Service Change requests sent by a CPE.
- **DSC rep received** - It is the number of DSC replies received by a CPE.
- **DSD req sent** - It is the number of Dynamic Service Deletion requests sent by a CPE.
- **DSD rep received** - It is the number of DSD replies received by a CPE.
- **CHS req received** - It is the number of Channel Switch Requests received by a CPE.
- **UCS Sent** - It is the number of Urgent Coexistence Situations sent by a CPE.

8.1.3.8.3 Incumbent Metrics

- **BS Id** - It represents the Id of the Base Station
- **Incumbent Id** - It represents the Id of the Incumbent.
- **Frequency** - It is the frequency at which the incumbent operates.
- **Operational Time** - It is the active period of the incumbent.
- **Idle Time** - It is the inactive period of the incumbent.
- **Interference Time** - It is the time when interference occurs due to CPE.

8.1.3.8.4 Channel Metrics

- **BS Id** - It is the Id of the BS

- **Channel Number** - It represents the channel number at which the BS is operating.
- **Frequency** - It is the frequency of the channel at which the BS is operating.
- **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.

8.1.3.9 LTENR Cell Metrics

Displays LTE/LTENR Metrics associated with each cell. Since NetSim currently uses Omni directional antennas a cell is all carriers in a Macro cell eNB Omni Ant; it is not a "sector carrier".

- **Macro cell eNB Omni Ant Name -** It is the device name of Macro cell eNB Omni Ant.
- **Macro cell eNB Omni Ant Interface ID -** It is unique LTE/LTENR RAN interface ID of the Macro cell eNB Omni Ant.
- **PDSCH Bytes Transmitted (bytes) -** It is the total number of bytes of data traffic transmitted in the downlink (Macro cell gNB/eNB Omni Ant to UEs) in the LTE/LTENR RAN interface of the Macro cell gNB/eNB Omni Ant.
- **PUSCH Bytes Transmitted (bytes) -** It is the total number of bytes of data traffic transmitted in the uplink (UEs to gNB/eNB) in the LTE/LTENR RAN interface of the Macro cell eNB Omni Ant.
- **PDSCH Throughput (Mbps) -** Downlink data delivered to its (respective) destination every second.

$$PDSCH\ Throughput(Mbps) = \frac{PDSCH\ Bytes\ Transmitted\ (bytes)\ *\ 8}{Simulation\ Time\ (\mu s)}$$

- **PUSCH Throughput (Mbps) -** Uplink data delivered to its (respective) destination every second.

$$PUSCH\ Throughput(Mbps) = \frac{PUSCH\ Bytes\ Transmitted\ (bytes)\ *\ 8}{Simulation\ Time\ (\mu s)}$$

The Bytes transmitted and Throughput calculations take into account all UEs associated with the Base station (BS or Macro cell eNB Omni Ant.). During simulations there could be handovers. When this occurs, calculations are carried out considering the BS to which the UE is associated. Recall that in the application throughput expression the denominator is $(ApplicationEndTime - ApplicationStartTime)$. Whereas in the above throughput expressions the denominator is $SimulationTime$. Hence even under cases with no handovers, the Cell throughputs will not reconcile with Application Throughputs.

8.1.3.10 IP Metrics

IP layer metrics calculated for the overall network and displayed for each device.

- **Device Id** - It is the unique ID of the Device.
- **Packet sent** - Specifies the number of packets (L3 and above) sent by the node.
- **Packet forwarded** - Specifies the number of packets (L3 and above) forwarding by an intermediate node(s) to next hop/target node.
- **Packets Received** - Specifies the number of packets (L3 and above) successfully received at the destination, from intermediate node(s) and source node(s).
- **Packets discarded** - Specifies the number of packets (L3 and above) discarded when there is no route available.
- **TTL Expired** - Specifies the number of Data and Control packets (L3 and above) dropped when TTL expires.
- **Firewall block** - Specifies the number of packets (L3 and above) blocked by Firewall for example TCP, UDP and ICMP Packets etc.

8.1.4 Advanced Metrics

In the Application metrics table, in addition to packets generated and packets received, additional information on duplicate packets that were received can be obtained. This is achieved by adding the following environment variable:

PC Settings → Properties → Advance system settings → Environment Variables → User Variables → New

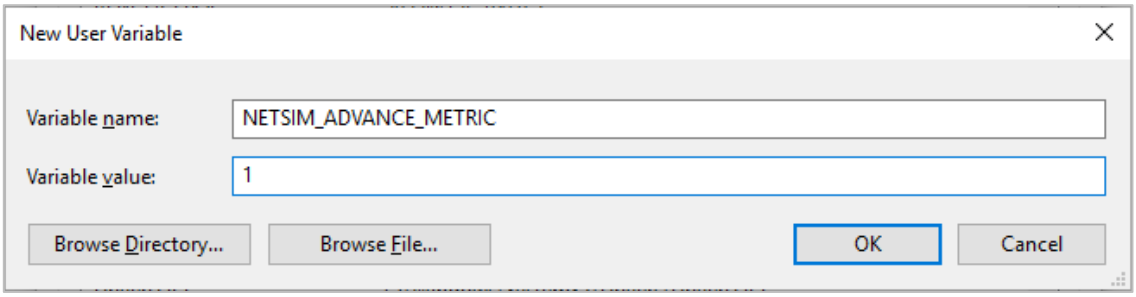


Figure 8-4: Environment Variables window

Note: To effect changes, user must restart NetSim. In the event this doesn't work please restart you system.

The Application metrics table in the results dashboard will display an additional column Duplicate packet received as shown below Figure 8-5.

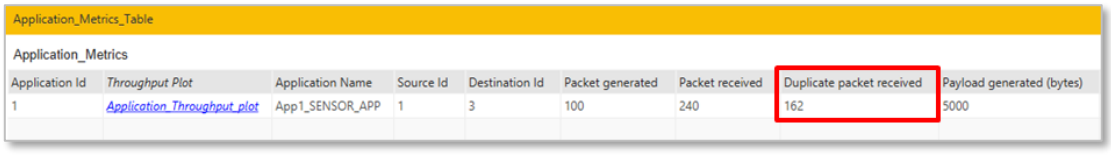


Figure 8-5: Application metrics table in results window

8.1.5 Export to .csv

In NetSim Simulation Result Dashboard, users can use the option **Export Results** to export all the metrics file to XL/CSV file for the further computation or analysis using it.

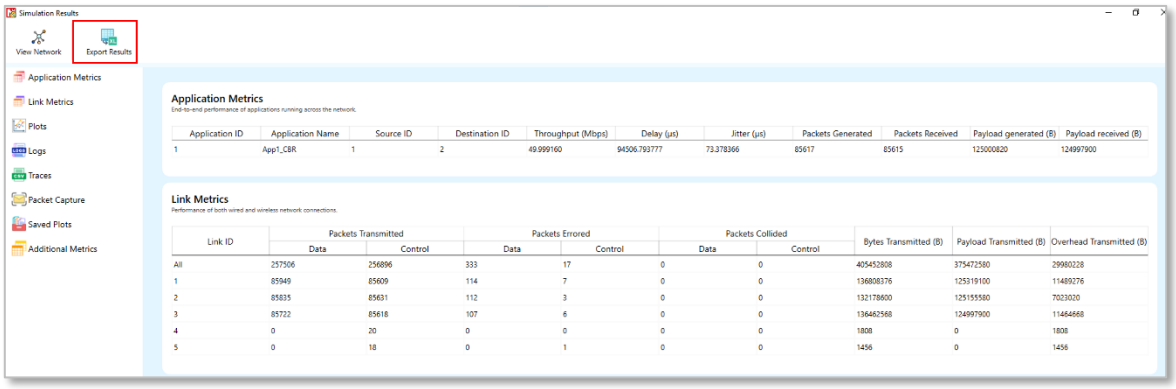
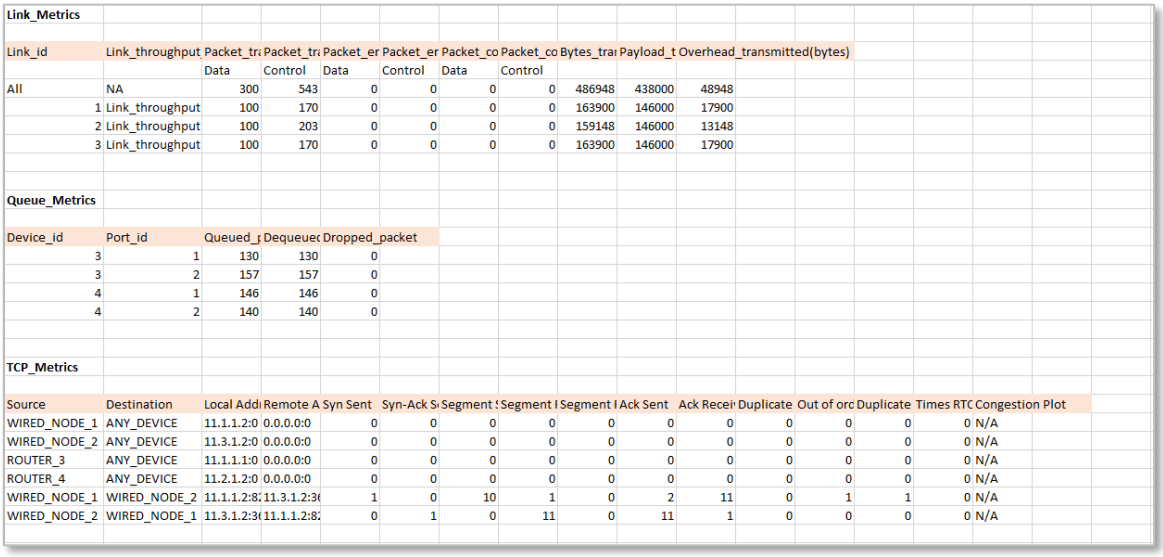


Figure 8-6: Select Option Export Results (.xls/.csv) in Result window.

XL/CSV file:



Link_Metrics

Link_id	Link_throughput	Packet_tr	Packet_tr	Packet_er	Packet_er	Packet_co	Packet_co	Bytes_tra	Payload_t	Overhead_transmitted(bytes)
		Data	Control	Data	Control	Data	Control			
All	NA	300	543	0	0	0	0	486948	438000	48948
1	Link_throughput	100	170	0	0	0	0	163900	146000	17900
2	Link_throughput	100	203	0	0	0	0	159148	146000	13148
3	Link_throughput	100	170	0	0	0	0	163900	146000	17900

Queue_Metrics

Device_id	Port_id	Queued_p	Dequeuec	Dropped_packet
3	1	130	130	0
3	2	157	157	0
4	1	146	146	0
4	2	140	140	0

TCP_Metrics

Source	Destination	Local Addı	Remote A	Syn Sent	Syn-Ack S	Segment S	Segment I	Segment I	Ack Sent	Ack Recei	Duplicate	Out of orc	Duplicate	Times RTC	Congestion	Plot
WIRED_NODE_1	ANY_DEVICE	11.1.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	0	N/A	
WIRED_NODE_2	ANY_DEVICE	11.3.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	0	N/A	
ROUTER_3	ANY_DEVICE	11.1.1.1:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	0	N/A	
ROUTER_4	ANY_DEVICE	11.2.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	0	N/A	
WIRED_NODE_1	WIRED_NODE_2	11.1.1.2:8	11.3.1.2:3	1	0	10	1	0	2	11	0	1	1	0	N/A	
WIRED_NODE_2	WIRED_NODE_1	11.3.1.2:3	11.1.1.2:8	0	1	0	11	0	11	1	0	0	0	0	N/A	

Figure 8-7: Option Export Results (.xls/.csv) to export all the metrics.

8.1.6 Notes on metrics

1. The metrics are calculated at each layer and might not be equivalent to the same metric calculated at a different layer. For exactness and precision, we recommend users also verify the results with the event trace & packet trace generated by NetSim.
2. Broadcast application will have no entries under Application Metrics in Results window if there are zero packets received. In other words, it will not show ‘0’ throughput. Users may notice that ‘0’ throughput is shown for unicast applications, and this is because of the way Broadcast application metrics is architected in NetSim.

8.1.7 Results files written at the end of simulation.

The following table lists the various files that will be written in the NetSim input-output directory on completion of simulation.

S. No	File	Contents
1	Metrics.xml	Contains network metrics
2	ConfigLog.txt	Records errors if any, in the configuration file.
3	LogFile.txt	Logs control flows across various layers in the Network Stack
4	PacketTrace.csv	If enabled, records detailed packet level information
5	EventTrace.csv	If enabled, records information about each event in the discrete event simulation
6	Static ARP.txt	Logs the IP address and MAC address of devices in the scenario

Table 8-1: Different results files written at the end of simulation in the I/O directory

8.2 Plots Window

In NetSim, Plots are the graphical representations of simulation results. Plots visualize various aspects of network performance including Application Performance, Link Performance, Radio Measurements, TCP Congestion and Buffer Occupancy.

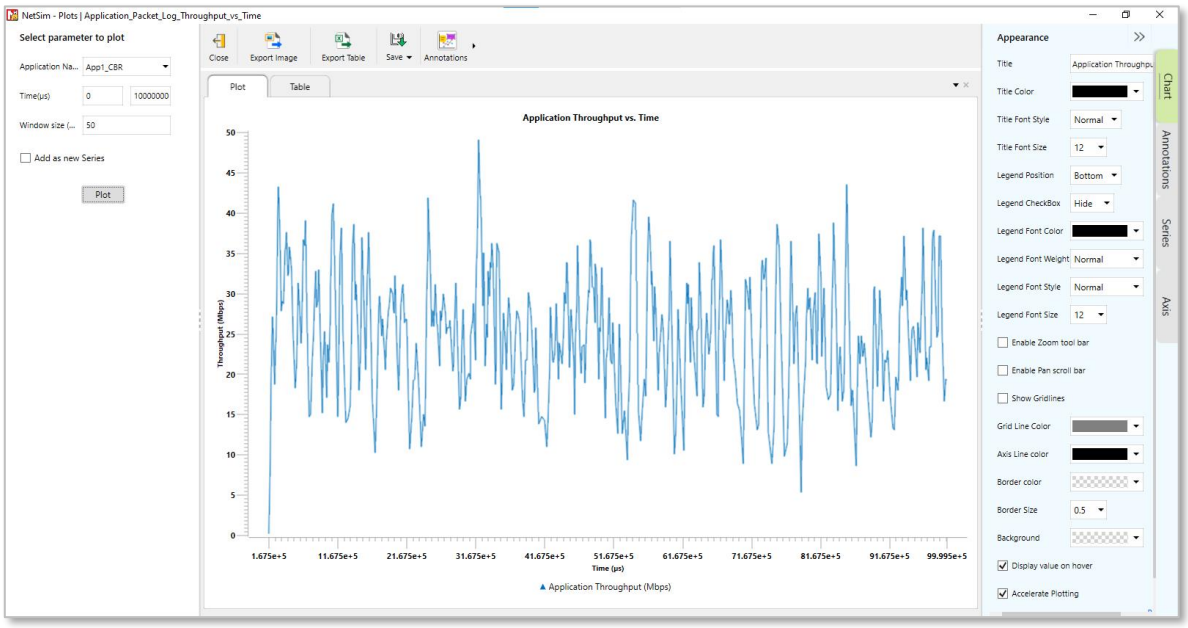


Figure 8-8: NetSim Plots window.

The process of generating plots involves two steps. First, the user selects the Plot(s) from the list available in the design window as show in Figure 8-9. Then the user must choose the specific device, link, application, etc, in the plots window as shown in Figure 8-9. This plots window, as shown in Figure 8-8, becomes accessible only after the simulation concludes.

To enable Plots,

- Users can click on the “Configure Reports” tab on the top.
- Then click on “Plots” icon in the top ribbon, which will then open a right plot panel, containing a list of available plots for the network as chosen by the user.
- Check boxes can be enabled to generate the respective plots, as shown below.

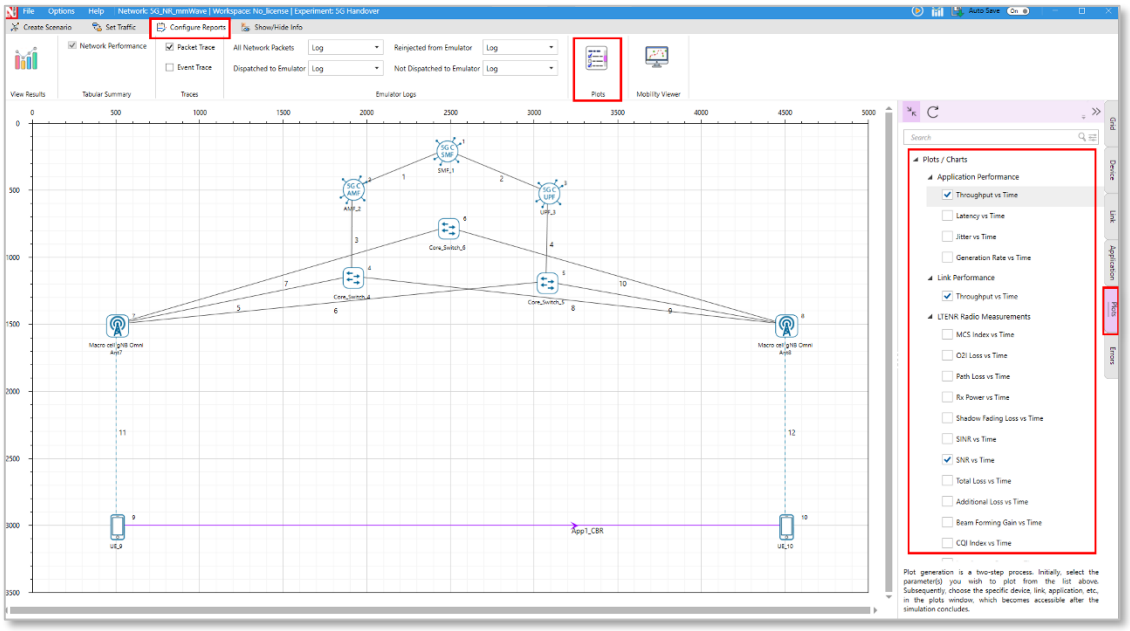


Figure 8-9: Click on “Configure reports” tab on top. Then click on “plots” icon in the top ribbon. You can then see the list of available plots is in the right-hand side properties panel.

Plots can be accessed post simulation from the results dashboard. Click on plots in the Results window to go to the plots section. Then click on the appropriate plot to open the Plots Window. Multiple plots can be opened simultaneously.

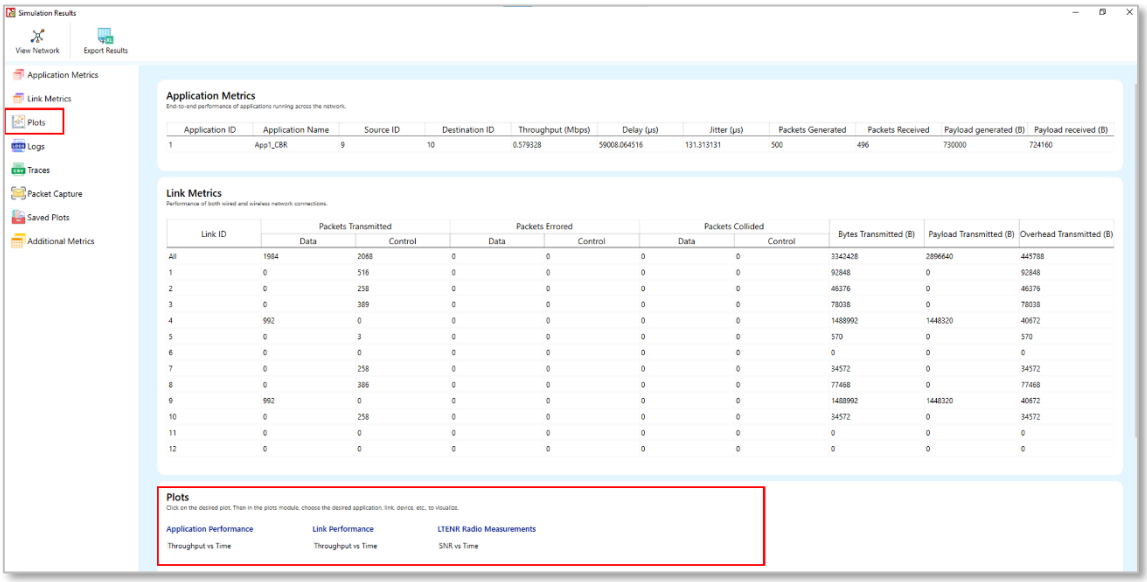


Figure 8-10: Click on plots in the Results window to go to the plots section. Then click on the appropriate plot to open the Plots Window. Note that multiple plots can be opened simultaneously.

The NetSim Plot window consists of plot commands, plot parameters, tabs related to chart, annotation, series, axis, etc.

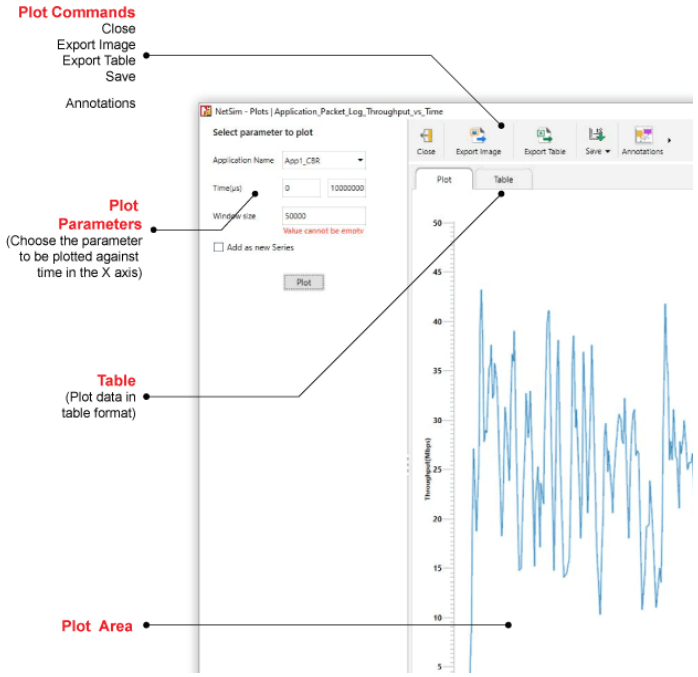


Figure 8-11: Plot commands, Plot parameters, Data Table and Plot area.

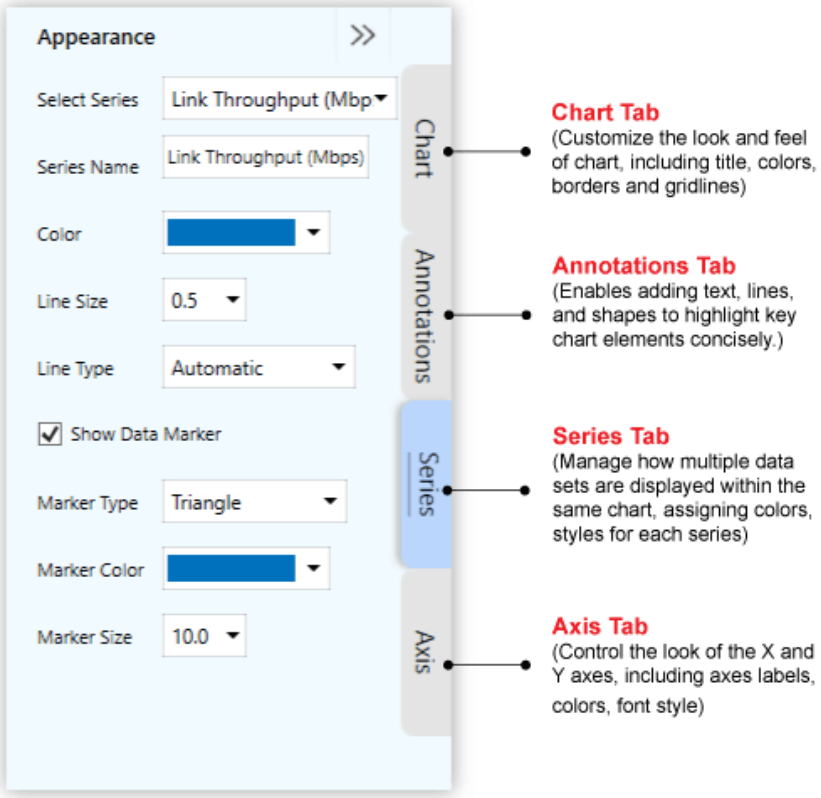


Figure 8-12: The different options on the right-hand side tab of the Plots window.

8.2.1 Application and Link Throughput Plots

If application and link throughput plots are enabled, NetSim plots the instantaneous (50 ms averaging window default) throughput all links and all applications in the network.

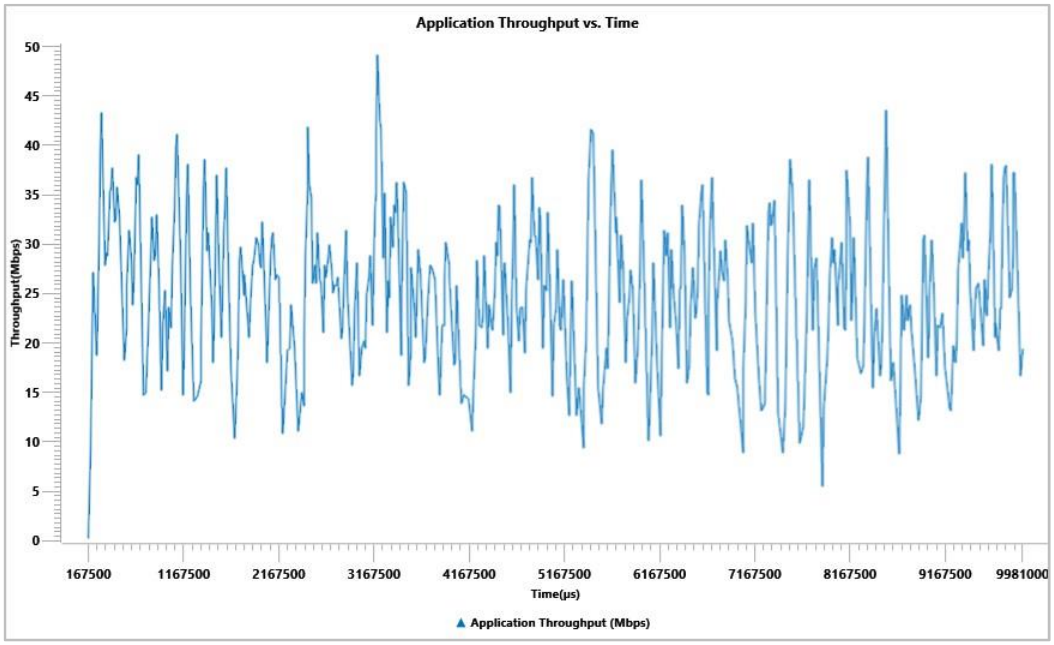


Figure 8-13: Application Throughput Plot

Since NetSim is a packet level simulator, at an instant in time a destination (of a link or application) can either be (i) receiving packets or (ii) not receiving packets. Therefore, the throughput at a point in time is either θ_{max} (the link speed) or 0. It is thus not meaningful to define the instantaneous throughput for a link or application at an instant in time. Hence, NetSim measures instantaneous throughput over an averaging window as explained below.

Instantaneous Throughput is defined as the bits successfully received in the averaging window divided by the averaging window size (in time units). It is measured every averaging window.

$$\theta_{Inst}\ (bits/s) = \frac{B_{Window}^{RxSuccess}\ (bits)}{W_{size}\ (s)}$$

Each value of the computed instantaneous throughput represents one point in the throughput plot. The computation and plotting are done every W seconds of virtual simulation time.

The link throughput statistic counts all traffic that was sent through a link. It includes data packets and control packets and includes retransmissions, errors, or collisions. This would also include packet flows from multiple applications that may flow through the same link. Timestamp for link throughput is the PHY layer end time of each packet passing through that link.

The application throughput only considers those data packets (application layer packet size) that were sent from the source and that were successfully received at the destination.

8.2.2 Buffer Occupancy Plot

The buffer occupancy over time can be plotted by enabling Buffer Occupancy Plot in the GUI.

Users can click on Configure reports tab, then click on plots option from the top ribbon. From the right panel under Buffer Occupancy select the Buffer Size vs Time.

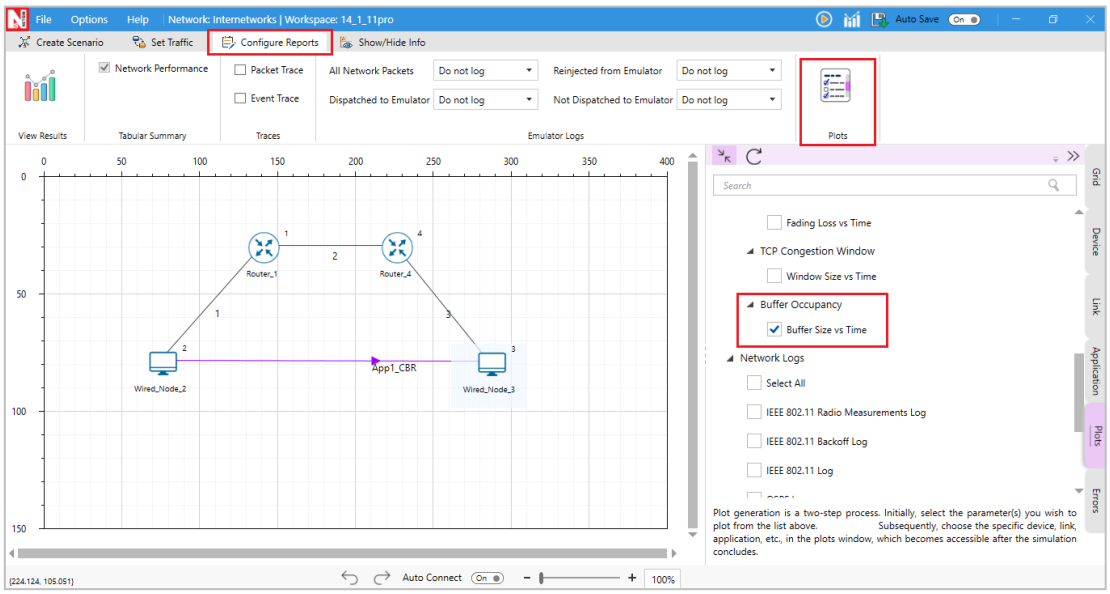


Figure 8-14: Buffer Occupancy Plot Enabled

Post simulation, we can see the Buffer Size vs Time plot in plots section and Buffer_Occupancy_Log under the Logs section in the Results Dashboard.

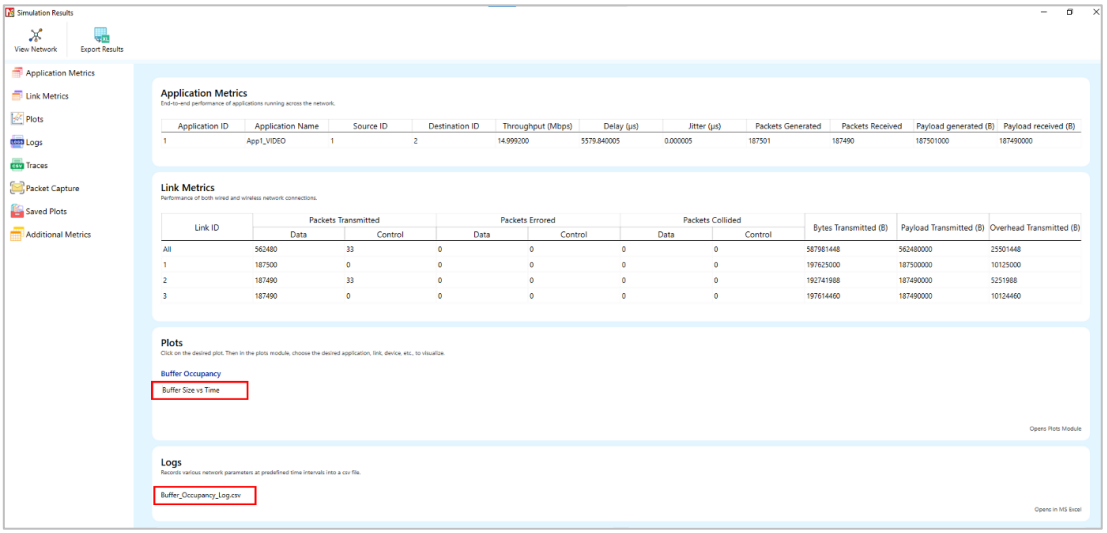


Figure 8-15: Results Dashboard

Buffer_Occupancy_Log records the occupied buffer size, total buffer size of each device over time. Information present in Buffer Occupancy Log is:

- Time in Microseconds
- Device ID

- Device Name
- Interface ID
- Interface Name
- Occupied Buffer Size (Bytes) by each device.
- Total Buffer size (Bytes), devices having buffer configured in GUI will be the total buffer size record in this column. Rest of the devices will have infinite buffers.
- Virtual Interface Name for the Access point running over EDCAF protocol.
- Remarks such as ENQUEUE, DEQUEUE and DROP.

Time (Microseconds)	Device ID	Device Name	Interface ID	Interface Name	Occupied Buffer Size (Bytes)	Total Buffer Size (MB)	Virtual Interface Name	Remarks
0	1	ROUTER_1	2	Interface_2 (WAN)	64	8	N/A	ENQUEUE
0	2	ROUTER_4	1	Interface_1 (WAN)	64	8	N/A	ENQUEUE
0	1	ROUTER_1	2	Interface_2 (WAN)	0	8	N/A	DEQUEUE
0	2	ROUTER_4	1	Interface_1 (WAN)	0	8	N/A	DEQUEUE
0	3	WIRED_NODE_2	1	Interface_1 (ETHER	44	inf	N/A	ENQUEUE
11.56	1	ROUTER_1	2	Interface_2 (WAN)	44	8	N/A	ENQUEUE
11.56	1	ROUTER_1	2	Interface_2 (WAN)	0	8	N/A	DEQUEUE
20.08	2	ROUTER_4	2	Interface_2 (ETHER	44	inf	N/A	ENQUEUE
30.68	4	WIRED_NODE_3	1	Interface_1 (ETHER	44	inf	N/A	ENQUEUE
41.28	2	ROUTER_4	1	Interface_1 (WAN)	44	8	N/A	ENQUEUE
41.28	2	ROUTER_4	1	Interface_1 (WAN)	0	8	N/A	DEQUEUE
49.8	1	ROUTER_1	1	Interface_1 (ETHER	44	inf	N/A	ENQUEUE
60.4	3	WIRED_NODE_2	1	Interface_1 (ETHER	40	inf	N/A	ENQUEUE
60.4	3	WIRED_NODE_2	1	Interface_1 (ETHER	1500	inf	N/A	ENQUEUE
70.68	1	ROUTER_1	2	Interface_2 (WAN)	40	8	N/A	ENQUEUE
70.68	1	ROUTER_1	2	Interface_2 (WAN)	0	8	N/A	DEQUEUE

Buffer_Occupancy_Log | Pivot Table(Custom)

Figure 8-16: Buffer Occupancy Log

Buffer size vs Time plot is plotted for Occupied Buffer size (Bytes) in Y-axis against Time (Microseconds) in X-axis.

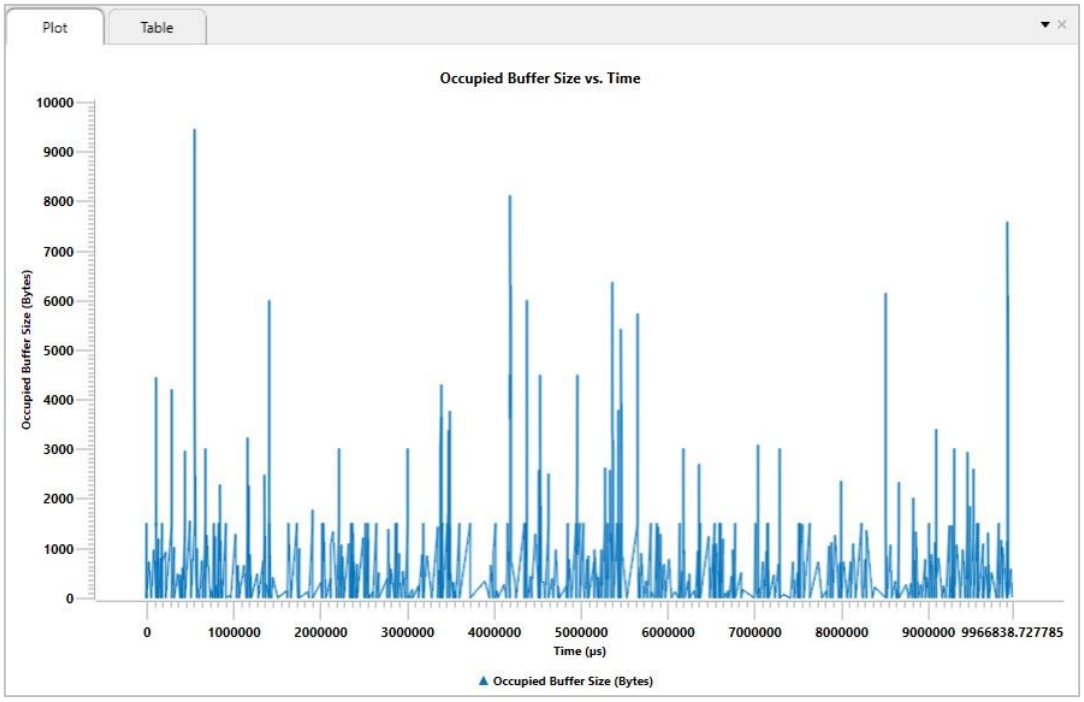


Figure 8-17: Buffer Occupancy Plot

8.2.3 TCP Congestion Window Plot

The TCP Congestion window over time can be plotted by enabling TCP Congestion Window Plot in the GUI.

Users can click on Configure reports tab, then click on plots option from the top ribbon. From the right panel under TCP Congestion Window select the Window Size vs Time.

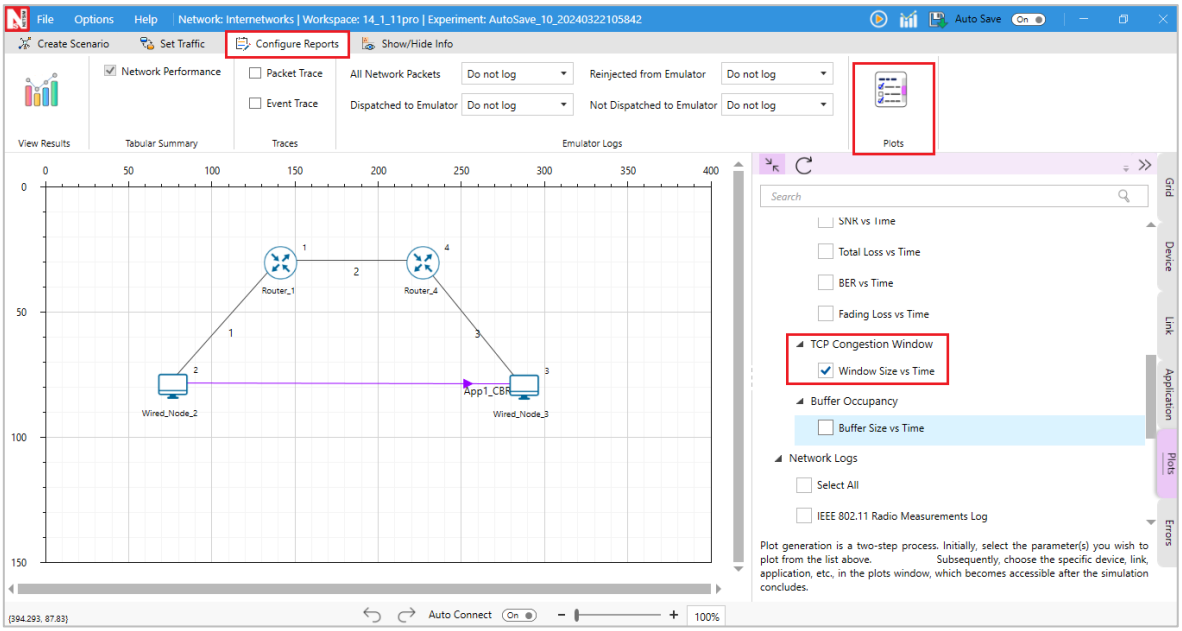


Figure 8-18: TCP Congestion Plot Enabled

Post simulation, we can see the Window Size vs Time plot in plots section and TCP_Congestion_Window_Log under the Logs section in the Results Dashboard.

TCP_Congestion_Window_Log records the sender window size over time. Information present in TCP_Congestion_Window_ Log is:

- Time in Microseconds
- Application ID
- Application Name
- Source Device Name
- Source ID
- Destination Device Name
- Destination ID
- Source Socket Address
- Destination Socket Address
- Sender Window Size (Bytes)

Time (Microseconds)	Application ID	Application Name	Source Device Name	Source ID	Destination Device Name	Destination ID	Source Socket Address	Destination Socket Address	Sender Window Size(Bytes)
163.4	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	0
163.4	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	4380
1628.8	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	0
1628.8	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
20000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	4380
20000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	5840
21459.16	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
21459.16	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
40000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	5840
40000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	7300
41459.16	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
41459.16	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
60000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	7300
60000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	8760
61459.16	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
61459.16	1	App1_CBR	WIRED_NODE_3	4	WIRED_NODE_2	3	192.169.0.2:36934	192.168.0.2:82	4381
80000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	8760
80000	1	App1_CBR	WIRED_NODE_2	3	WIRED_NODE_3	4	192.168.0.2:82	192.169.0.2:36934	10220

TCP_Congestion_Window_Log | Pivot Table(Custom)

Figure 8-19; TCP Congestion Window Log

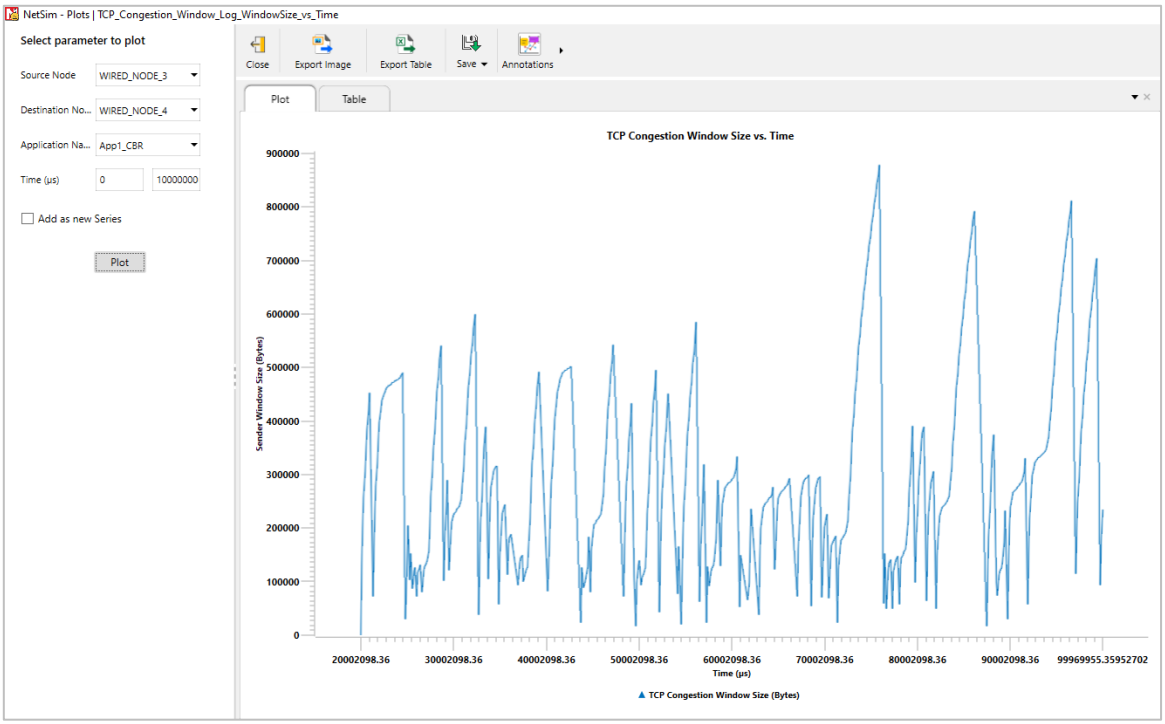


Figure 8-20: TCP Congestion window plot post simulation.

The down sampling algorithm in NetSim's plot engine leads to approximations while plotting, especially in TCP. To obtain a very precise TCP congestion plot window please enable Wireshark interfacing and view the TCP congestion window in Wireshark.

8.2.4 Notes on plots

1. To accelerate plotting, NetSim uses down-sampling/decimation to choose n points from N for plotting. NetSim generates n random numbers from a discrete uniform $U(0, N-1)$ distribution and plots for these n points.
2. To get a more precise plot users can select the min and max values (time) and replot.
3. The link throughput is calculated as the sum of throughputs in both directions for a full duplex link.

4. Application throughput is plotted till the last packet reaches or till end of simulation time, whichever is earlier.
5. Cumulative Moving Average: This is the average of the metric up until the current time and is defined as

$$\bar{\theta}(t) = \frac{1}{t}\int_0^t r(u)du$$

8.3 Network Logs

Users can enable protocol specific log files such as the Radio Measurement log, Radio resource allocation log etc., by clicking on the Configure Reports tab and select Plot icon option present in the toolbar as shown below.

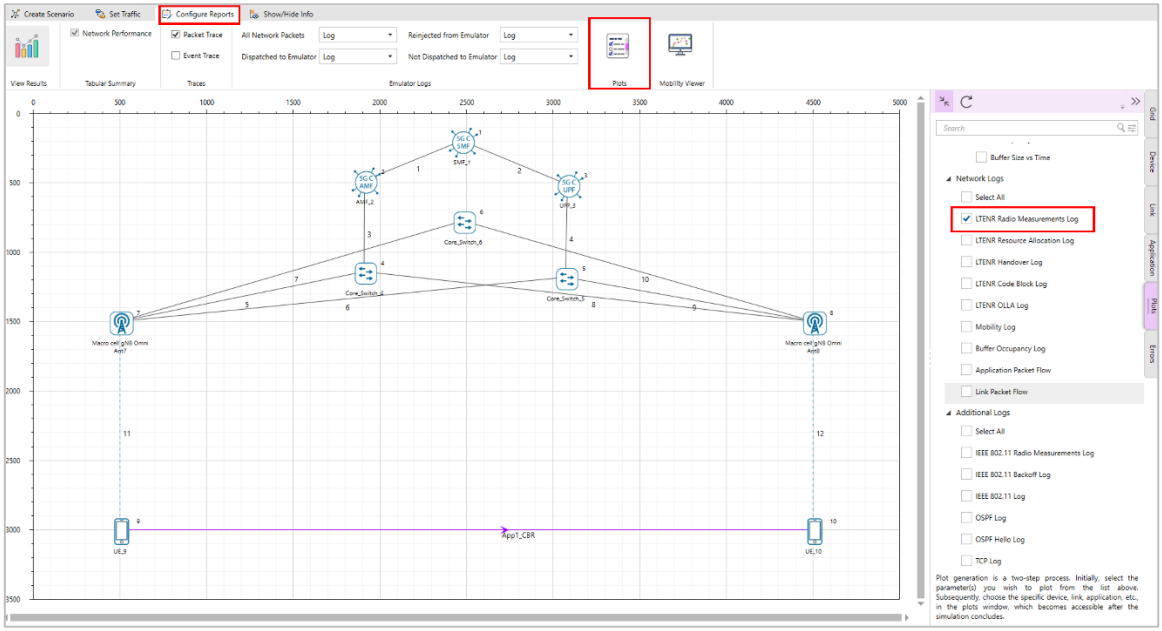


Figure 8-21: Enabling log files in NetSim GUI.

The panel that appears will contain a list of logs that are applicable for the current Network. Check boxes can be enabled to generate the respective logs. Log files can be accessed post simulation from the results dashboard as shown below:

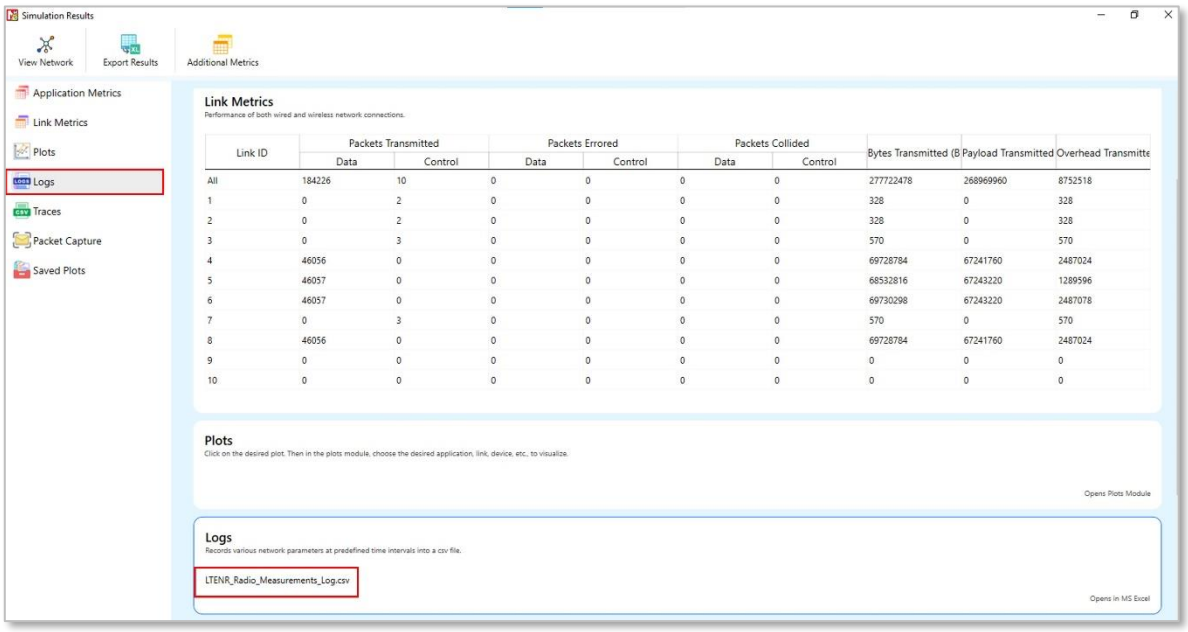


Figure 8-22: Log Files available in Results dashboard.

8.4 Packet Trace

NetSim allows users to generate trace files which provide detailed packet information useful for performance validation, statistical analysis, custom code de-bugging and synthetic data generation for machine learning. Packet Trace logs a set of chosen parameters for every packet as it flows through the network such as arrival times, queuing times, departure times, payload, overhead, errors, collisions etc.

The packet trace is written whenever a packet is received at a device. For example, if we have transmission N1 -> N2 -> N3, then the packet trace is written for every packet being received at N2 and at N3. Note that it is not written for every packet being transmitted by N1 and the subsequently by N2. This means that packets which are transmitted from N1 but which may have been errored or collided before being received by N2 are not written in the packet trace.

By providing a host of information and parameters of every packet that flows through the network, packet trace provides necessary forensics for users to catch logical errors without setting a lot of breakpoints or restarting the program often. Window size variation in TCP, Route Table Formation in OSPF, Medium Access in Wi-fi, etc., are examples of protocol functionalities that can be easily understood from the trace.

Note that by default, packet tracing option is turned off. Turning on Packet Trace will slow down the simulation significantly.

8.4.1 How to Enable Packet trace

Step 1: Consider a scenario comprising of two Wired nodes and Router, create a traffic flow between the two wired nodes.

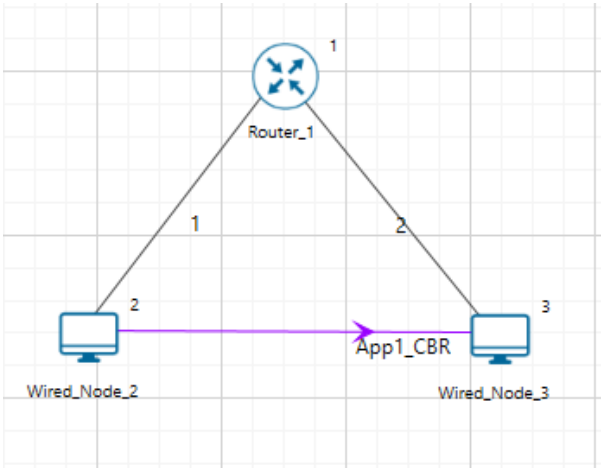


Figure 8-23: Network Scenario

Step 2: By default, Packet trace is disabled, to enable packet trace click on Configure Reports tab and enable Packet trace as shown in the below figure.

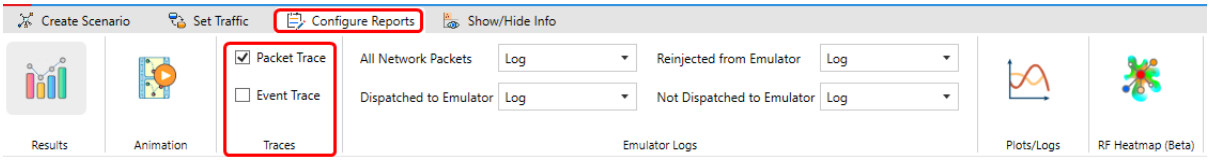


Figure 8-24: Packet trace option in ribbon.

Step 3: Click on the Run and simulate the scenario. After completion of simulation NetSim Results dashboard window appears.

Click on the Open packet trace option from result dashboard Window as shown below:

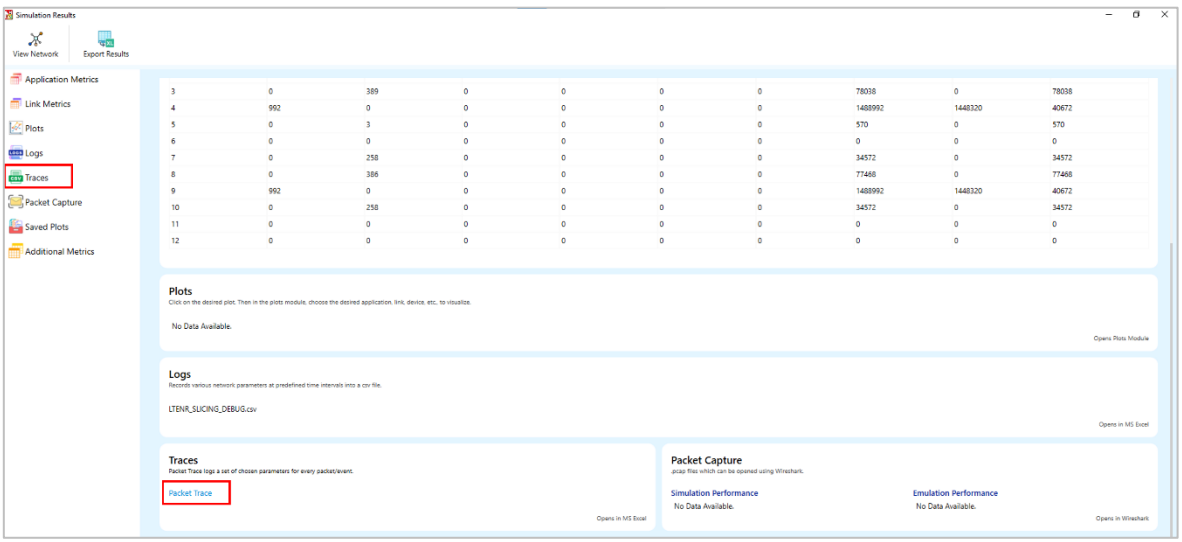


Figure 8-25: Result Dashboard Window

8.4.2 How to set filters to NetSim trace file

Step 1: Open the trace file. (In this example packet trace is opened)

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME(µS)
1	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	SWITCH-3	0
1	0	CBR	App1_CBR	NODE-1	NODE-2	SWITCH-3	ACCESSPOINT-4	0
2	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	SWITCH-3	233
2	0	CBR	App1_CBR	NODE-1	NODE-2	SWITCH-3	ACCESSPOINT-4	233
1	0	CBR	App1_CBR	NODE-1	NODE-2	ACCESSPOINT-4	NODE-2	0
3	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	SWITCH-3	466
0	N/A	Control_Packet	WLAN_BlockACK	NODE-2	ACCESSPOINT-4	NODE-2	ACCESSPOINT-4	N/A
3	0	CBR	App1_CBR	NODE-1	NODE-2	SWITCH-3	ACCESSPOINT-4	466
4	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	SWITCH-3	699
4	0	CBR	App1_CBR	NODE-1	NODE-2	SWITCH-3	ACCESSPOINT-4	699

Figure 8-26: Packet Trace

Step 2: Click the arrow in the header of the column you want to filter. In the list of text or numbers, uncheck the (Select All) box at the top of the list, and then check the boxes of the items you want to show.

For example, click on arrow of SOURCE_ID and uncheck the “Select all” check box and select NODE 2 then click on OK.

All the rows which are having NODE 2 as source id will be shown below Figure 8-27.

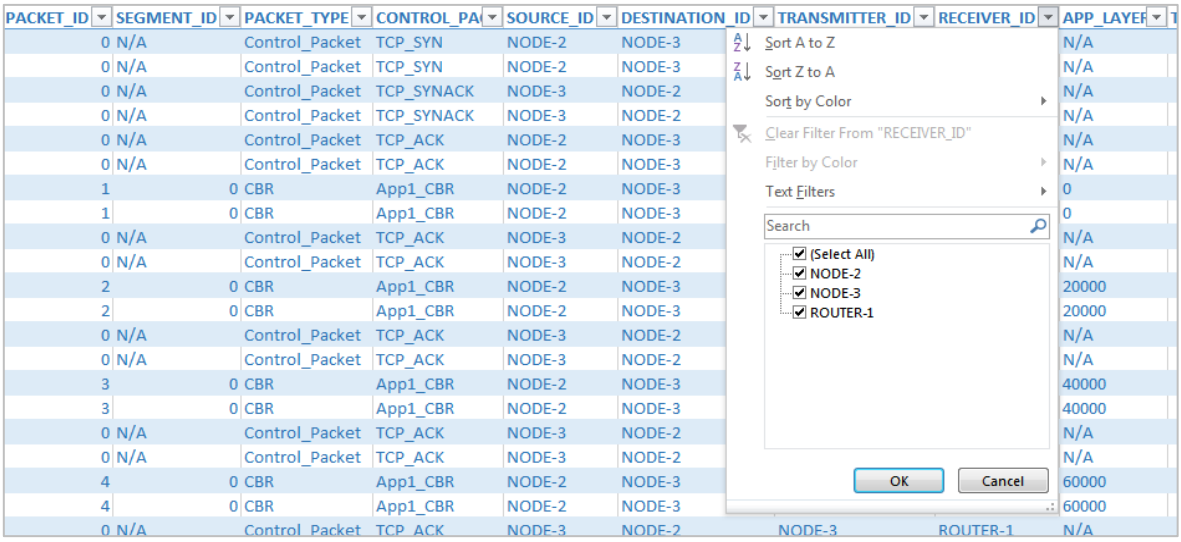


Figure 8-27: Select Transmitter ID arrow mark in the header in packet trace

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PA	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER
0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
1	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	0
2	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	20000
3	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	40000
4	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	60000
5	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	80000
6	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	100000
7	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	120000
8	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	140000
9	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	160000
10	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	180000
11	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	200000
12	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	220000

Figure 8-28: Filter Transmitter ID to NODE 2 in packet trace

Typically, filters can be set to observe **"Errored/Collided/Successful**"packets, packets of destination and packets of source.

8.4.3 Observing packet flow in the Network through packet trace file

Open the packet trace file, Click the arrow in the header of the column PACKET_ID and uncheck the "Select all" check box and select the packet id which you want to observe, for example 1, and then click on OK.

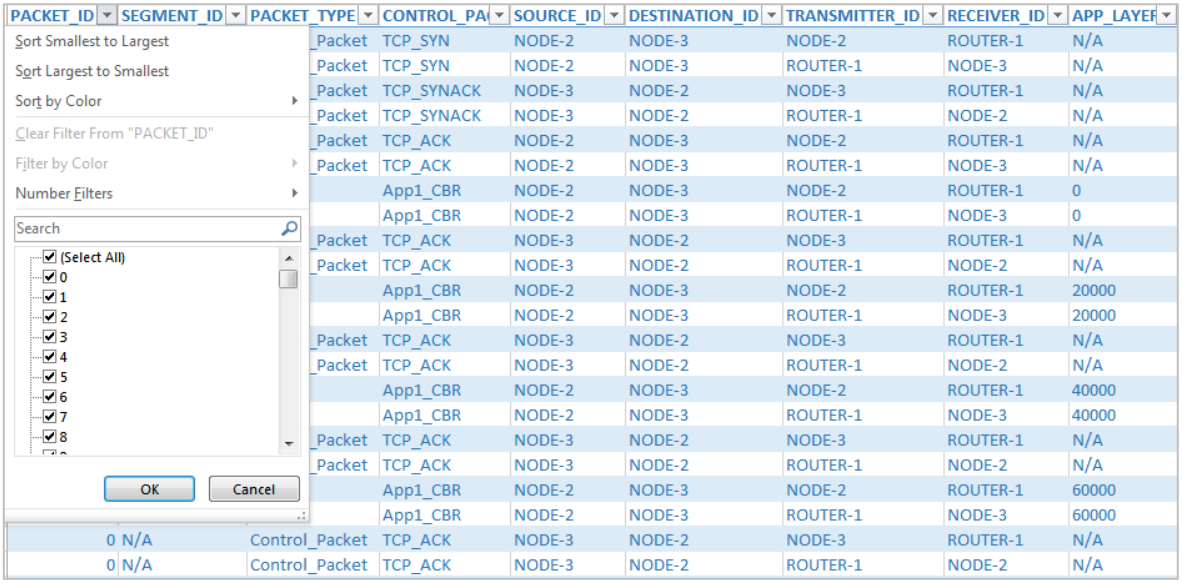


Figure 8-29: Select Packet ID arrow mark in the header in packet trace

Scenario is as shown below Figure 8-30 and traffic flow is from Wired Node 2 to Wired Node 3.

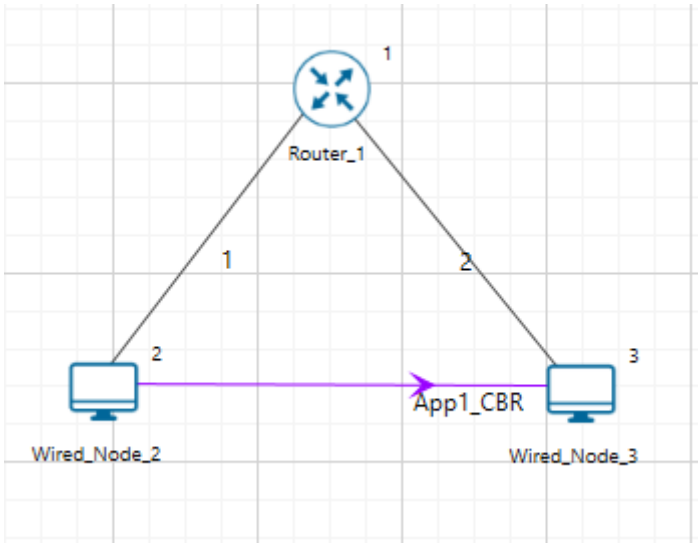


Figure 8-30: Traffic flow is from Wired Node 2 to Wired Node 3

Flow of packet 1 can be observed from the packet trace as shown below Figure 8-31.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PA	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER
0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3	N/A
0	N/A	Control_Packet	TCP_SYNACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
0	N/A	Control_Packet	TCP_SYNACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3	N/A
1	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	0

Figure 8-31: Flow of packet observed in the packet trace

Note: In the trace file device IDs are shown not device names. Wired Node 1's ID is 2 so it is Shown as NODE-2, Wired Node 2's ID is 3 so it is shown as NODE -3, Router-1' ID is 1 so it is shown as ROUTER-1. Device IDs are shown on the top of the device icon in the above scenario.

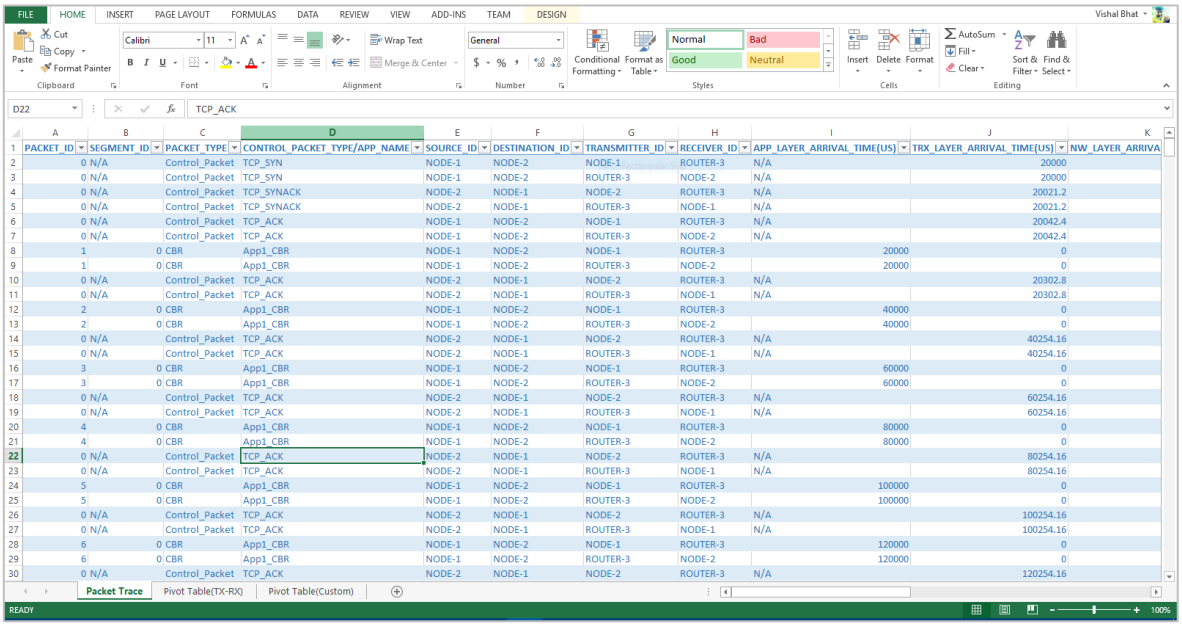
In a scenario source and destinations are fixed but transmitter and receiver are changed. For example, in the above scenario NODE-2 is the source and NODE-3 is the destination, but when NODE- 2 sending the packet to the ROUTER-1 then NODE-2 is the transmitter and ROUTER-1 is the receiver. When ROUTER-1 sending the packet to the NODE-3, ROUTER-1 is the transmitter and NODE-3 is the receiver.

8.4.4 Analysing Packet Trace using Pivot Tables

NetSim Packet trace is saved as a spread sheet. Packet Trace can be converted to an Excel table to make the management and analysis of data easier. A table typically contains related data in a series of worksheet rows and columns that have been formatted as a table. By using the table features, you can then manage the data in the table rows and columns independently from the data in other rows and columns on the worksheet.

PivotTables are a great way to summarize, analyze, explore, and present your data, and you can create them with just a few clicks. PivotTables are highly flexible and can be quickly adjusted depending on how you need to display your results. You can also create Pivot Charts based on PivotTables that will automatically update when your PivotTables do.

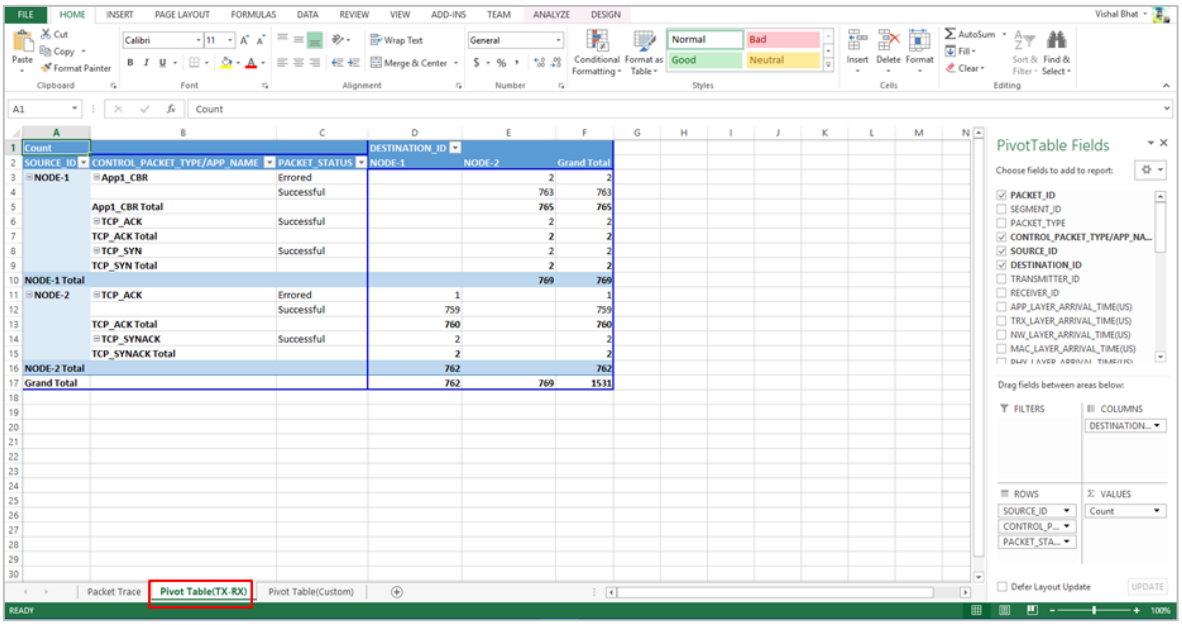
If you enable packet trace, Open Packet Trace link present in the Simulation Results Window can be used to load the packet Trace file in MS-Excel. Formats the spread sheet as a table for convenient analysis.



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)	NW_LAYER_ARRIVA
0	N/A	Control_Packet	TCP_SYN	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	20000	
0	N/A	Control_Packet	TCP_SYN	NODE-1	NODE-2	ROUTER-3	NODE-2	N/A	20000	
0	N/A	Control_Packet	TCP_SYNACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	20021.2	
0	N/A	Control_Packet	TCP_SYNACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A	20021.2	
0	N/A	Control_Packet	TCP_ACK	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	20042.4	
0	N/A	Control_Packet	TCP_ACK	NODE-1	NODE-2	ROUTER-3	NODE-2	N/A	20042.4	
1	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	20000	0	
1	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	20000	0	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	20302.8	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A	20302.8	
2	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	40000	0	
2	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	40000	0	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	40254.16	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A	40254.16	
3	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	60000	0	
3	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	60000	0	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	60254.16	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A	60254.16	
4	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	80000	0	
4	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	80000	0	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	80254.16	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A	80254.16	
5	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	100000	0	
5	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	100000	0	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	100254.16	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A	100254.16	
6	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	120000	0	
6	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	120000	0	
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A	120254.16	

Figure 8-32: Sheet 1 is the packet trace

Sheet 2 of the packet trace file has a pivot table – Pivot Table (TX-RX) automatically populated to analyze the packets that were transmitted and received in the network that was simulated. Further users can modify the table by adding or deleting the column headers.



Count			DESTINATION_ID		
SOURCE_ID	CONTROL_PACKET_TYPE/APP_NAME	PACKET_STATUS	NODE-1	NODE-2	Grand Total
NODE-1	App1_CBR	Errored		2	2
		Successful		763	763
	App1_CBR Total			765	765
	TCP_ACK	Successful		2	2
	TCP_ACK Total			2	2
	TCP_SYN	Successful		2	2
	TCP_SYN Total			2	2
NODE-1 Total				769	769
NODE-2	TCP_ACK	Errored	1		1
		Successful	759		759
	TCP_ACK Total		760		760
	TCP_SYNACK	Successful	2		2
	TCP_SYNACK Total		2		2
NODE-2 Total			762		762
Grand Total			762	769	1531

Figure 8-33: Sheet 2 of the packet trace file has a pivot table

Sheet 3 of the packet trace has a blank pivot table – **Pivot Table (Custom)** which can be used to create additional pivot tables from scratch.

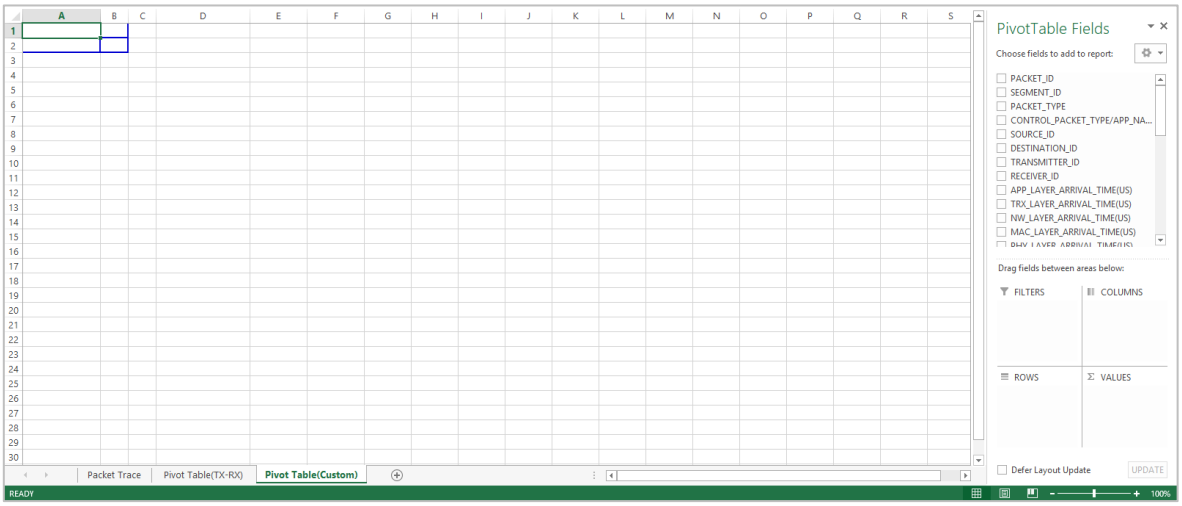


Figure 8-34: Sheet 3 of the packet trace file has a blank pivot table

Steps to analyse the packet trace using pivot tables

Step 1: Click on Packet Trace in the result dashboard, you can find 3 sheets will be created i.e. Packet Trace, Pivot Table (TX-RX), Pivot Table (Custom)

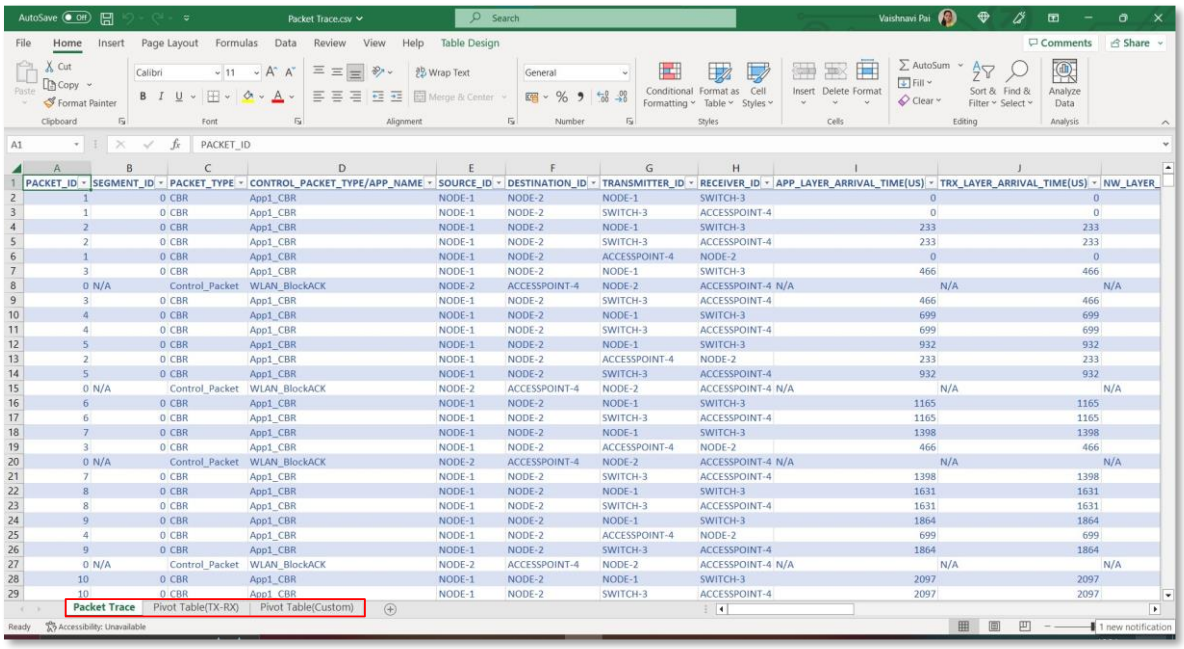


Figure 8-35: Packet Trace, Pivot Table (TX-RX), Pivot Table (Custom) in packet trace

Step 2: Click on Pivot Table (Custom) to create your own pivot table.

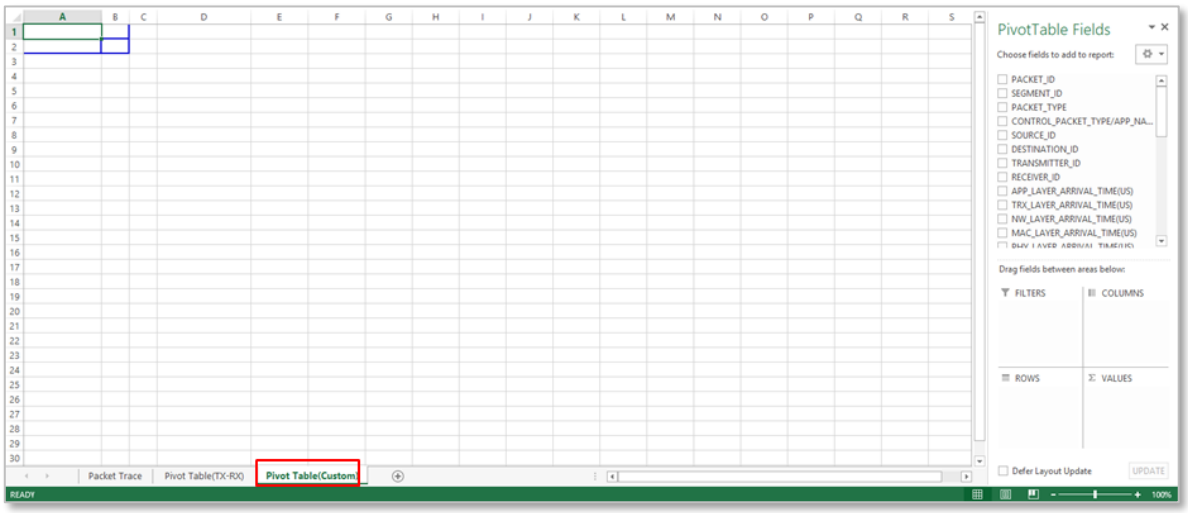


Figure 8-36: Select Blank Pivot Table (Custom) to create your own pivot table

Once you open the sheet PivotTable (Custom), you will need to decide which **fields** to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

8.4.5 Packet Transmitted / Received Analysis

- If you want to analyze packets sent from all sources to all destinations, then check SOURCE_ID, DESTINATION_ID and CONTROL_PACKET_TYPE/APP_NAME as shown below Figure 8-37.

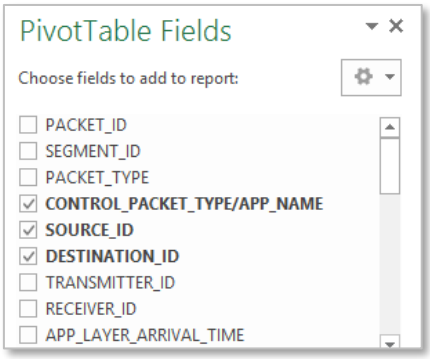


Figure 8-37: Select the check box of SOURCE_ID, DESTINATION_ID and CONTROL_PACKET_TYPE/APP_NAME in PivotTable Fields

- The selected fields will be added to one of the four areas below the Field List. Click SOURCE_ID, hold it and drag to the ROW field. Similarly, DESTINATION_ID to COLUMNS and CONTROL_PACKET_TYPE/APP_NAME to VALUES.

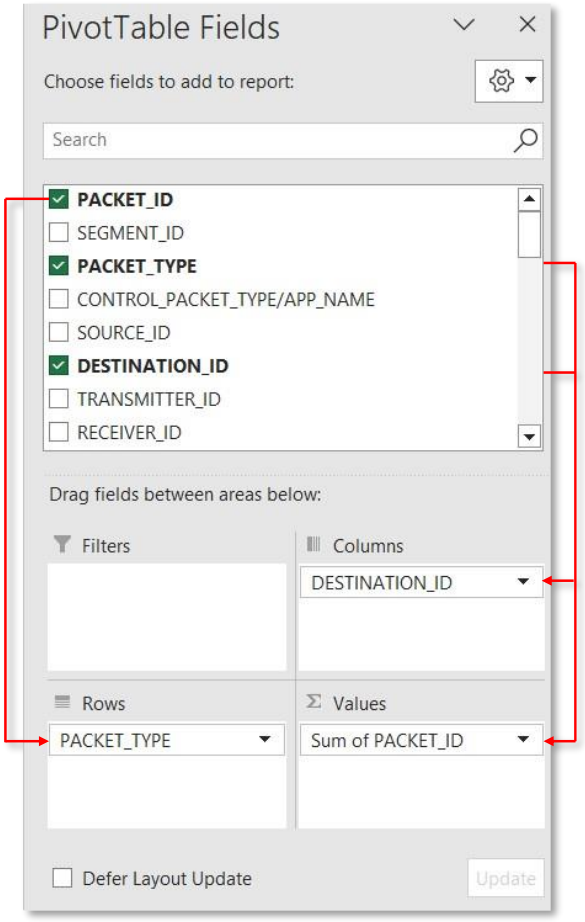


Figure 8-38: Selected fields add to one of the four areas below the Field List

- The PivotTable will calculate and summarize the selected fields. In this example, the PivotTable shows the packets sent from all sources to all destinations.

Count of CONTROL_PACKET_TYPE/APP_NAME	DESTINATION_ID		
SOURCE_ID	NODE-1	NODE-2	Grand Total
NODE-1		356	356
NODE-2	349		349
Grand Total	349	356	705

Figure 8-39: PivotTable Created with selected fields

- The above example shows all the packets which including data packets and control packets.
- If you wish to know how many Data and how many were control packets then, check the PACKET_TYPE and drag it to the ROWS field as shown below Figure 8-40.

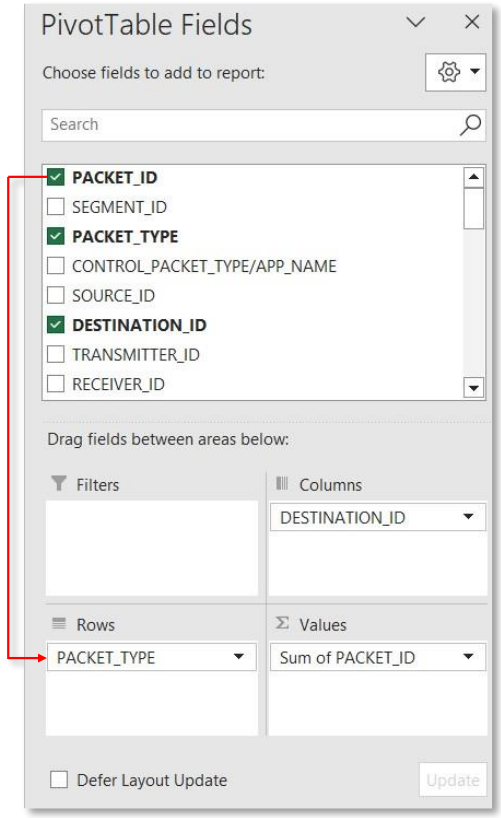


Figure 8-40: Select the PACKET_TYPE Check Box and drag it to the ROWS fields

- This will look like

Count of CONTROL_PACKET_TYPE/APP_NAME		DESTINATION_ID		
SOURCE_ID	PACKET_TYPE	NODE-1	NODE-2	Grand Total
NODE-1	CBR		352	352
	Control_Packet		4	4
NODE-1 Total			356	356
NODE-2	Control_Packet	349		349
NODE-2 Total		349		349
Grand Total		349	356	705

Figure 8-41: PivotTable Created with Packet Type

- Further, if you wish to know how many packets got errored and how many were successful, check the PACKET_STATUS field and drag it to the ROWS field.

Count of CONTROL_PACKET_TYPE/APP_NAME			DESTINATION_ID		
SOURCE_ID	PACKET_TYPE	PACKET_STATUS	NODE-1	NODE-2	Grand Total
NODE-1	CBR	Errored		2	2
		Successful		350	350
	CBR Total			352	352
	Control_Packe	Successful		4	4
	Control_Packet Total			4	4
NODE-1 Total				356	356
NODE-2	Control_Packe	Errored	1		1
		Successful	348		348
	Control_Packet Total		349		349
NODE-2 Total			349		349
Grand Total			349	356	705

Figure 8-42: PivotTable Created with Packet Status

8.4.6 Delay analysis

We explain this using a packet trace generated per the following network scenario.

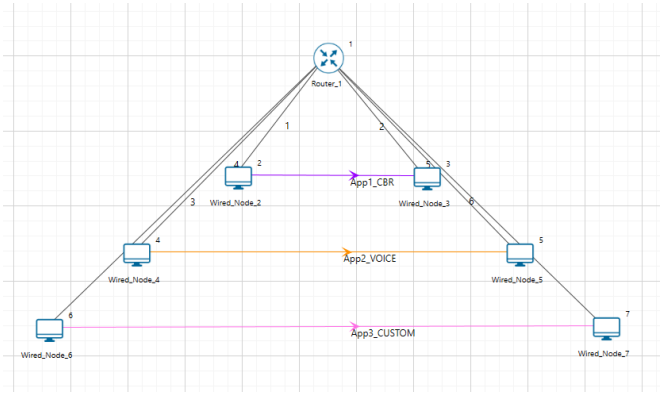


Figure 8-43: Network Topology with different application

Consider network scenario with 1 router and 6 wired nodes. Create 3 applications as per the following Table 8-2.

Application Type	Source Id	Destination Id	Transport Protocol	Packet Size (Bytes)	Inter arrival time (µs)
CBR	2	3	TCP	1460	20000
VOICE	4	5	UDP	1500	20000
CUSTOM	6	7	TCP	1200	20000

Table 8-2: Application Properties

Note: Users need to select Codec as CUSTOM for voice application as shown in the below screenshot Figure 8-44.

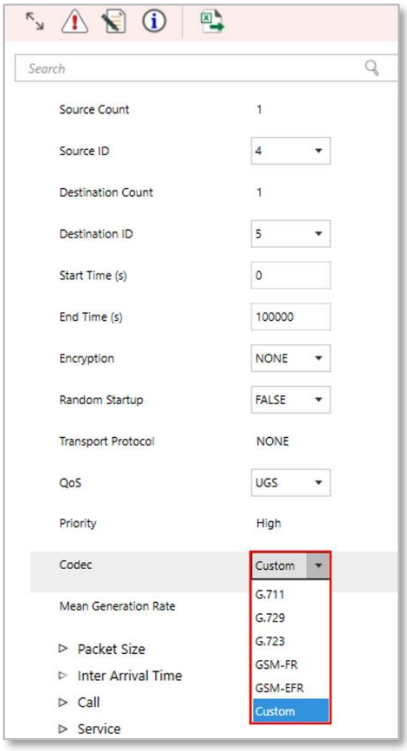
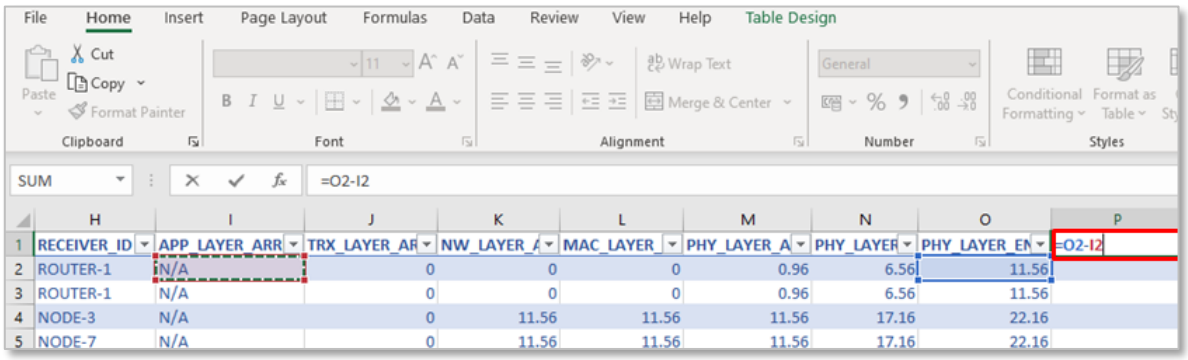


Figure 8-44: Application properties Window

Enable Packet Trace and simulate the scenario for 10 seconds. Open packet trace and perform the following steps:

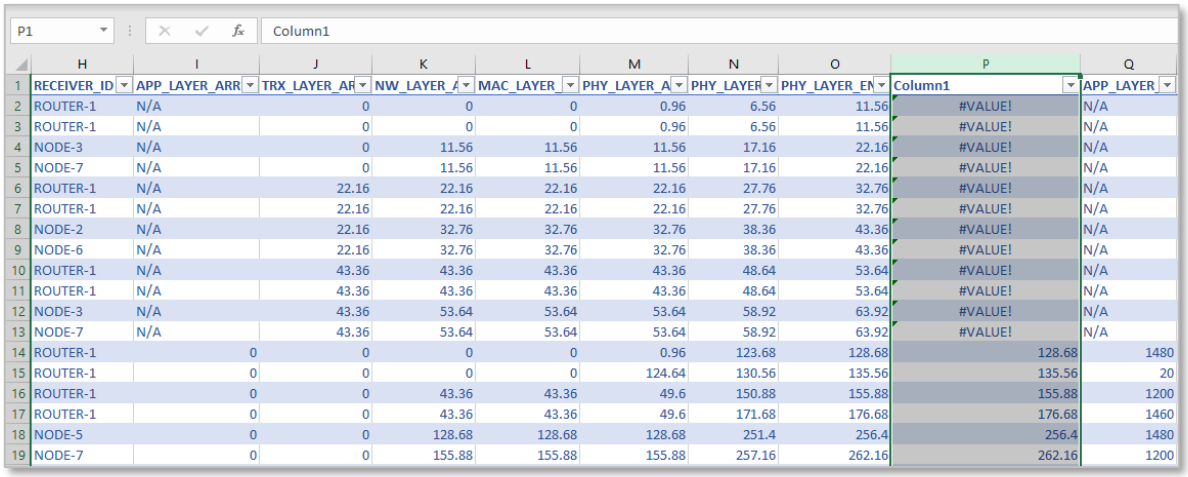
- Insert a column after PHY_LAYER_END_TIME, then select the whole column and calculate delay for each and every packet by using the formula.
  =PHY_LAYER_END_TIME – APPLICATION_LAYER_ARRIVAL_TIME



	H	I	J	K	L	M	N	O	P
1	RECEIVER_ID	APP_LAYER_ARR	TRX_LAYER_AR	NW_LAYER_A	MAC_LAYER	PHY_LAYER_A	PHY_LAYER	PHY_LAYER_EN	=O2-I2
2	ROUTER-1	N/A	0	0	0	0.96	6.56	11.56	
3	ROUTER-1	N/A	0	0	0	0.96	6.56	11.56	
4	NODE-3	N/A	0	11.56	11.56	11.56	17.16	22.16	
5	NODE-7	N/A	0	11.56	11.56	11.56	17.16	22.16	

Figure 8-45: Calculate delay in packet trace using PHY_LAYER_END_TIME – APPLICATION_LAYER_ARRIVAL_TIME then Press CTRL + ENTER

- Then Press CTRL + ENTER. This will calculate delay for the whole column shown below.



	H	I	J	K	L	M	N	O	P	Q
1	RECEIVER_ID	APP_LAYER_ARR	TRX_LAYER_AR	NW_LAYER_A	MAC_LAYER	PHY_LAYER_A	PHY_LAYER	PHY_LAYER_EN	Column1	APP_LAYER
2	ROUTER-1	N/A	0	0	0	0.96	6.56	11.56	#VALUE!	N/A
3	ROUTER-1	N/A	0	0	0	0.96	6.56	11.56	#VALUE!	N/A
4	NODE-3	N/A	0	11.56	11.56	11.56	17.16	22.16	#VALUE!	N/A
5	NODE-7	N/A	0	11.56	11.56	11.56	17.16	22.16	#VALUE!	N/A
6	ROUTER-1	N/A	22.16	22.16	22.16	22.16	27.76	32.76	#VALUE!	N/A
7	ROUTER-1	N/A	22.16	22.16	22.16	22.16	27.76	32.76	#VALUE!	N/A
8	NODE-2	N/A	22.16	32.76	32.76	32.76	38.36	43.36	#VALUE!	N/A
9	NODE-6	N/A	22.16	32.76	32.76	32.76	38.36	43.36	#VALUE!	N/A
10	ROUTER-1	N/A	43.36	43.36	43.36	43.36	48.64	53.64	#VALUE!	N/A
11	ROUTER-1	N/A	43.36	43.36	43.36	43.36	48.64	53.64	#VALUE!	N/A
12	NODE-3	N/A	43.36	53.64	53.64	53.64	58.92	63.92	#VALUE!	N/A
13	NODE-7	N/A	43.36	53.64	53.64	53.64	58.92	63.92	#VALUE!	N/A
14	ROUTER-1	0	0	0	0	0.96	123.68	128.68	128.68	1480
15	ROUTER-1	0	0	0	0	124.64	130.56	135.56	135.56	20
16	ROUTER-1	0	0	43.36	43.36	49.6	150.88	155.88	155.88	1200
17	ROUTER-1	0	0	43.36	43.36	49.6	171.68	176.68	176.68	1460
18	NODE-5	0	0	128.68	128.68	128.68	251.4	256.4	256.4	1480
19	NODE-7	0	0	155.88	155.88	155.88	257.16	262.16	262.16	1200

Figure 8-46: Calculate delay for the whole column

- Name the column as DELAY.
- Go to Insert->PivotTable and click on OK to create a blank Pivot Table with the newly added column listed under the PivotTable Fields.
- Drag and drop DESTINATION_ID, RECEIVER_ID, PACKET_STATUS and CONTROL_PACKET_TYPE/APP_NAME to FILTERS field shown below Figure 8-47.

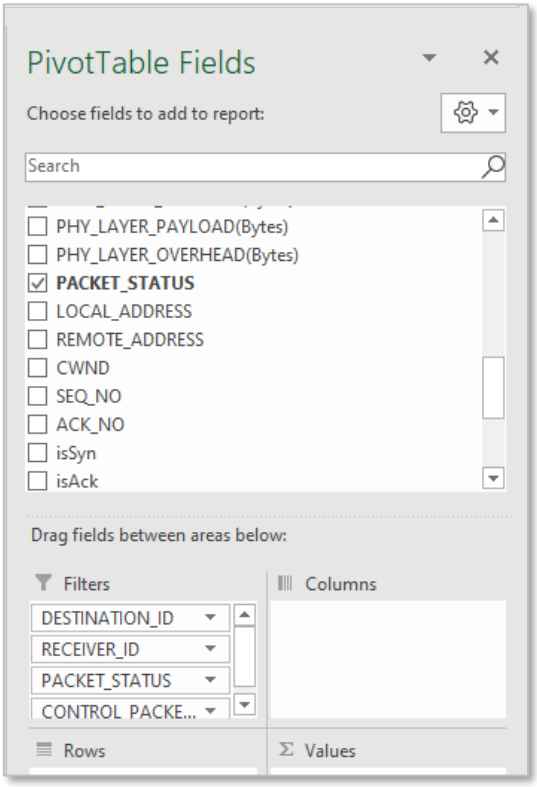


Figure 8-47: Added Selected fields to Filter

- Filter RECEIVER_ID to Node-3 by clicking on the drop down and select OK.

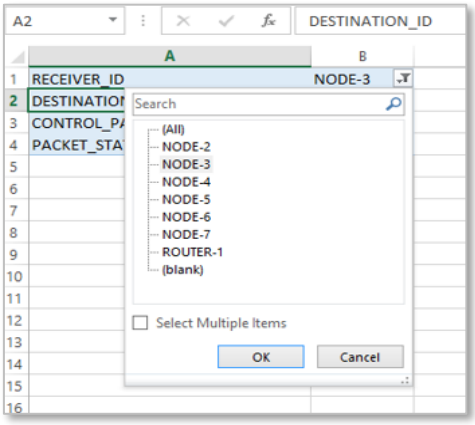


Figure 8-48: Filter RECEIVER_ID to Node-3 by clicking on the drop down

- Similarly filter CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR, DESTINATION_ID to NODE-3 and PACKET_STATUS to Successful

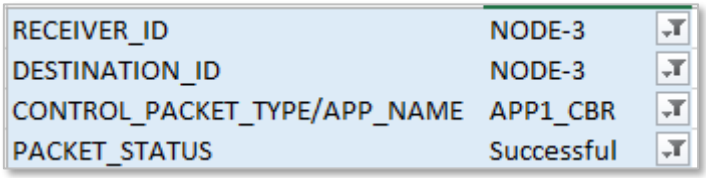


Figure 8-49: Similarly filter other fields as per screenshot

- Drag and drop PACKET_ID to ROWS and the Delay value that we calculated earlier to VALUES area.

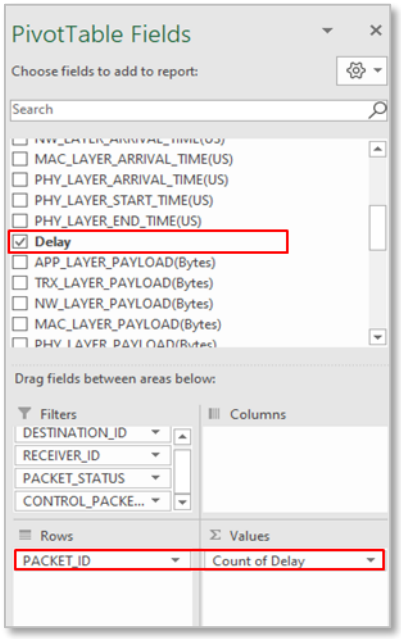


Figure 8-50: Drag and drop DELAY value that we have calculated earlier to ROWS and VALUES field

- Click on Count of DELAY drop down and select Value Field settings, then Select SUM and click on OK.

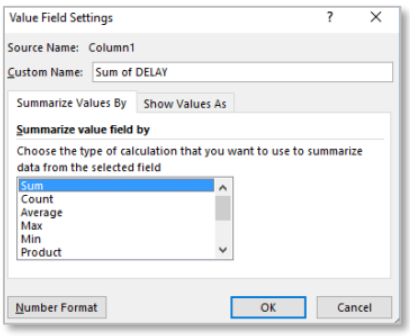


Figure 8-51: Select Count of DELAY drop down and select Value Field settings as SUM

- Again, Drag and drop DELAY to VALUES field.

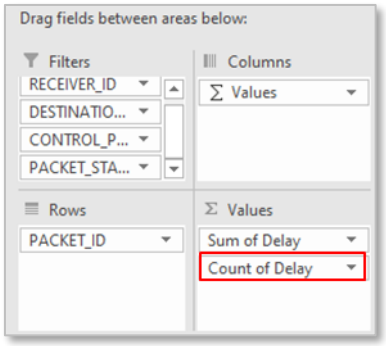


Figure 8-52: Drag and drop DELAY to VALUES field

- Select one cell and calculate the Application Delay, which is the average delay faced by a packet by using the formula.

$$Application\ \ DELAY\ = \frac{Sum\ of\ DELAY\ of\ Successful\ Packets}{Number\ of\ Packets}$$

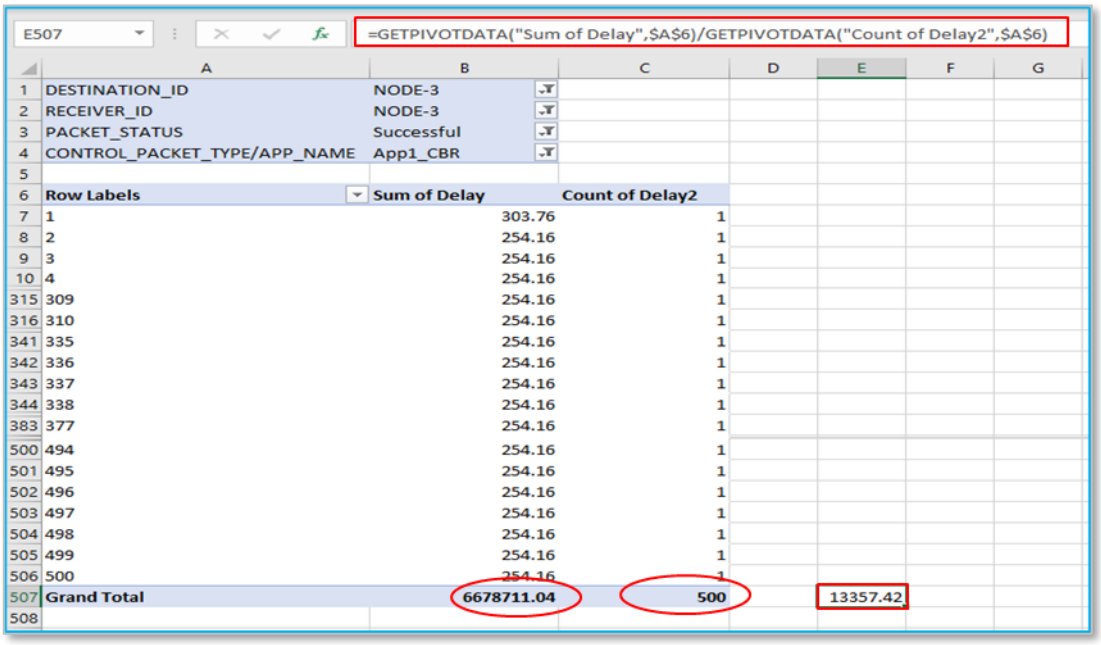
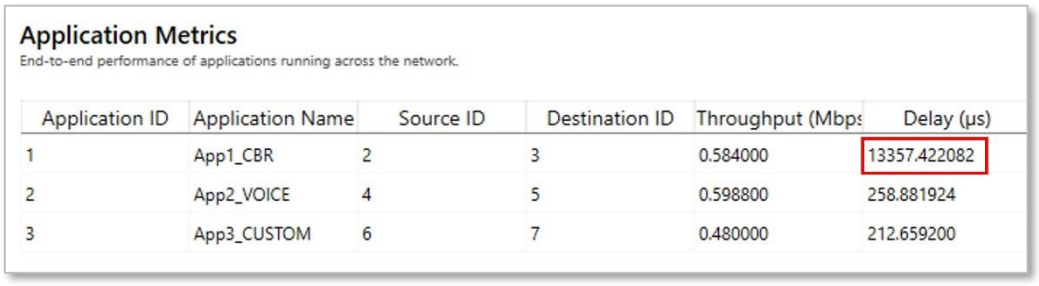


Figure 8-53: Calculated the Application Delay in Pivot table

- Compare the obtained value with the DELAY in Application Metrics



Application Metrics

End-to-end performance of applications running across the network.

Application ID	Application Name	Source ID	Destination ID	Throughput (Mbps	Delay (µs)
1	App1_CBR	2	3	0.584000	13357.422082
2	App2_VOICE	4	5	0.598800	258.881924
3	App3_CUSTOM	6	7	0.480000	212.659200

Figure 8-54: Compare the obtained DELAY with Application Metrics DELAY

- To calculate DELAY for VOICE application, filter DESTINATION_ID to Node-5, RECEIVER_ID to Node-5, CONTROL_PACKET_TYPE/APP_NAME to APP2_VOICE and PACKET_STATUS to Successful
- Similarly calculate and compare DELAY for other applications by following the above procedure.

8.4.7 Throughput analysis

To explain how users can perform Throughput Analysis, we have used same network design example as was used for Delay analysis above.

After loading the packet trace switch to sheet Pivot Table (Custom), drag and drop SOURCE_ID, RECEIVER_ID, CONTROL_PACKET_TYPE / APP_NAME and PACKET_STATUS to FILTERS field.

- Similarly drag and drop APP_LAYER_PAYLOAD to ROWS field and VALUES field.
- Filter SOURCE_ID to NODE-2, CONTROL_PACKET_TYPE APP_NAME to APP1_CBR, PACKET_STATUS to Successful and RECEIVER_ID to NODE-3
- Click on Count of APP_LAYER_PAYLOAD drop down and select Value Field settings, then Select Sum and click on OK.
- The pivot table would look like.

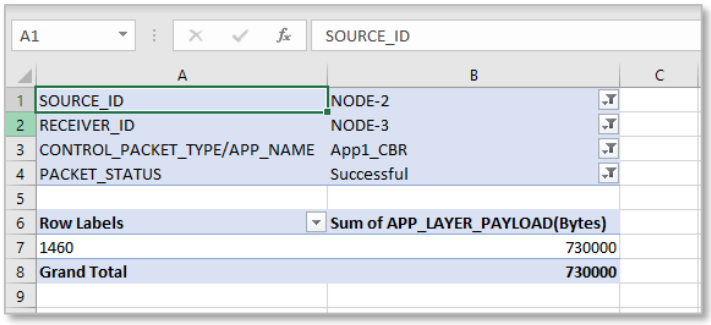


Figure 8-55: Pivot Table

- Select 1 cell and calculate the throughput by using the formula.

$$Application\ throughput\ (Mbps) = \frac{Sum\ of\ \ Successfully\ Received\ App\ Layer\ Payload\ (Bytes) * 8}{Simulation\ Time\ (\mu s\ )}$$

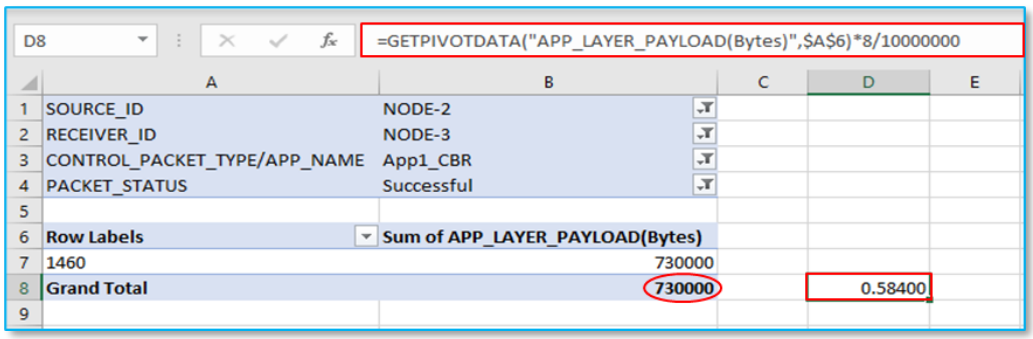
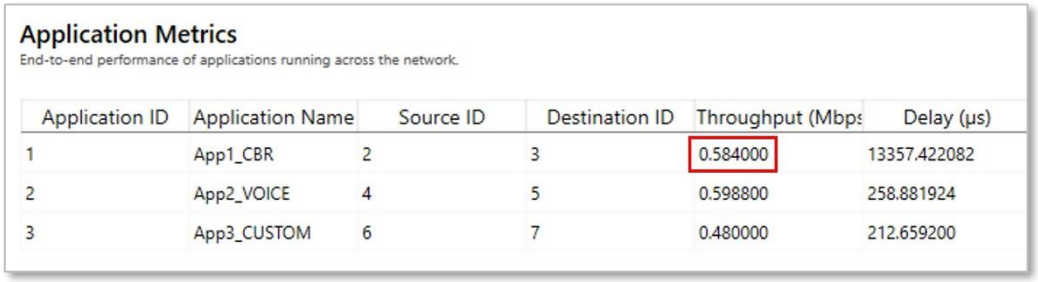


Figure 8-56: Calculate the throughput by using the formula in Pivot Table

EmptyCell=GETPIVOTDATA("APP_LAYER_PAYLOAD(Bytes)",A6,"APP_LAYER_PAYLOAD(Bytes)",1460)*8/10000000

- Now compare the throughput calculated using pivot table with the Application Metrics throughput.



Application Metrics

End-to-end performance of applications running across the network.

Application ID	Application Name	Source ID	Destination ID	Throughput (Mbps	Delay (µs)
1	App1_CBR	2	3	0.584000	13357.422082
2	App2_VOICE	4	5	0.598800	258.881924
3	App3_CUSTOM	6	7	0.480000	212.659200

Figure 8-57: Compared the calculated throughput using pivot table with the Application Metrics throughput

- To calculate THROUGHPUT for VOICE application, filter SOURCE_ID to Node-4, RECEIVER_ID to Node-5, CONTROL_PACKET_TYPE/APP_NAME to APP2_VOICE and PACKET_STATUS to Successful
- Similarly calculate and compare THROUGHPUT for other applications by following the above procedure.

8.4.8 Plotting with Pivot Charts

In a pivot table, you can create a new field that performs a calculation on the sum of other pivot fields.

- Open Packet Trace, switch to sheet Pivot Table (Custom)
- Drag and drop SOURCE_ID, RECEIVER_ID and PACKET_STATUS to FILTERS field, then CONTROL_PACKET_TYPE/APP_NAME, APP_LAYER_PAYLOAD to ROWS field Figure 8-58.

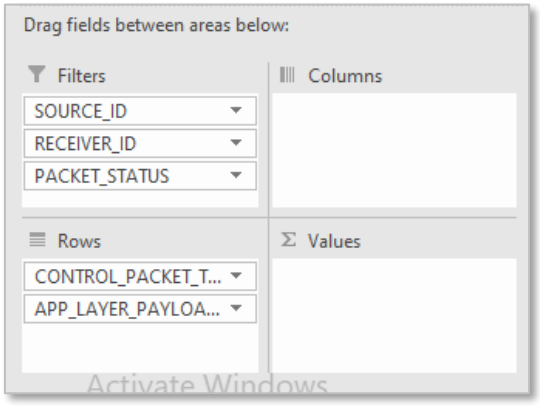


Figure 8-58: Drag and drop Sleeted Fields to one of the four areas below the Field List

- Filter SOURCE_ID to Node 2, Node 4 and Node 6, then RECEIVER_ID to Node 3, Node 5 and Node 7 and PACKET_STATUS to successful

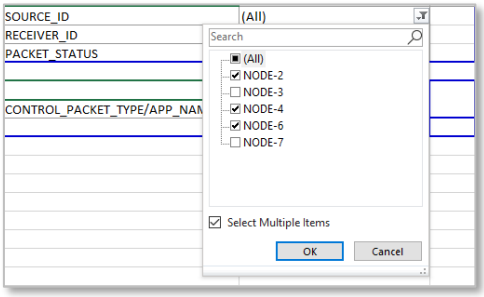


Figure 8-59: Filter Source ID and Destination ID

- Filter CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR, APP2_VOICE and APP3_CUSTOM
- Select a cell in the pivot table, and on the Excel Ribbon, under the PivotTable Tools tab, click the Options tab (PivotTable Analyze tab in Excel 2013).
- In the Calculations group, click Fields, Items, & Sets, and then click Calculated Field.

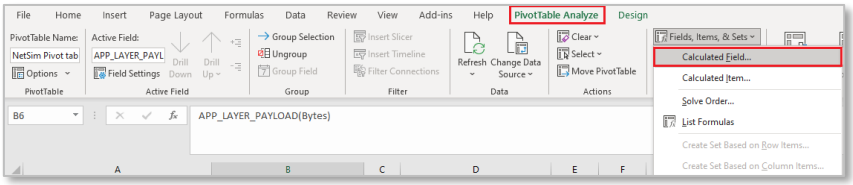


Figure 8-60: In Calculations group Select Calculated Field

- Type a name for the calculated field, Application Throughput.
- Then click on ADD to save the calculated field.

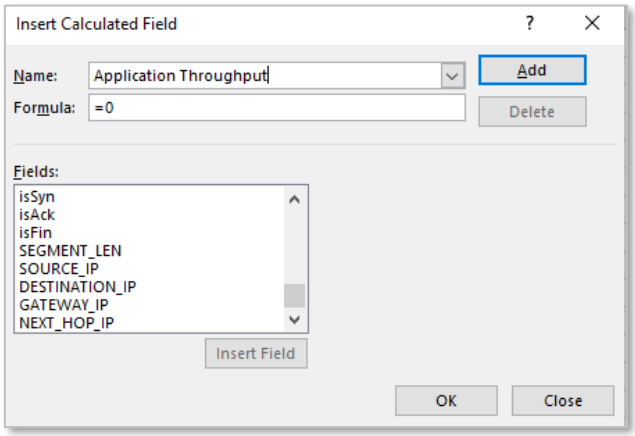


Figure 8-61: Insert calculated field name and select Add

- Click on Formula text box and then select APP_LAYER_PAYLOAD in the Fields list and click on Insert Field.
- Calculate the throughput by using the following formula shown below and click on OK.

$$Application\ throughput\ (Mbps) = \frac{Sum\ of\ \ Successfully\ Received\ App\ Layer\ Payload\ (Bytes) * 8}{Simulation\ Time\ (\mu s\ )}$$

Formula='APP_LAYER_PAYLOAD(Bytes)'*8/10000000

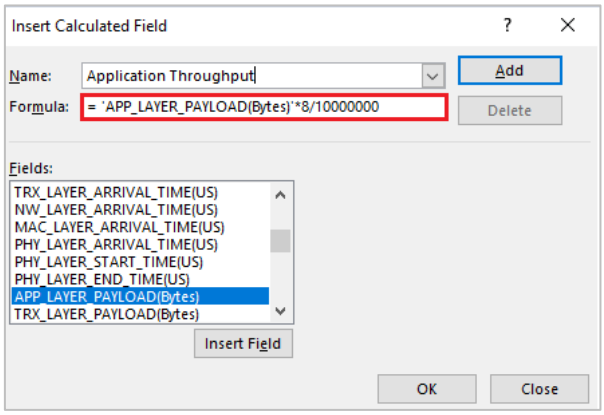


Figure 8-62: Calculate the throughput by using the following formula

- Then Drag and drop the newly added Application throughput to values field

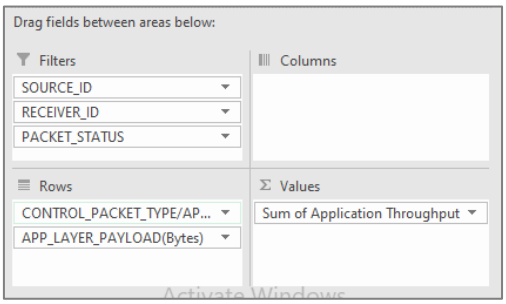


Figure 8-63: Add Application throughput to values field

- Select a cell in the pivot table, and on the Excel Ribbon, under the PivotTable Tools tab, click the Options tab (PivotTable Analyze tab in Excel 2013).
- In the Tools group, click Pivot chart and select OK.

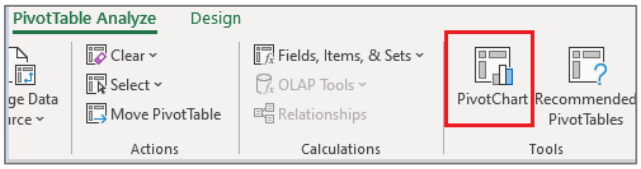


Figure 8-64: In the Tools group Select Pivot chart

- This will display a pivot chart shown below.

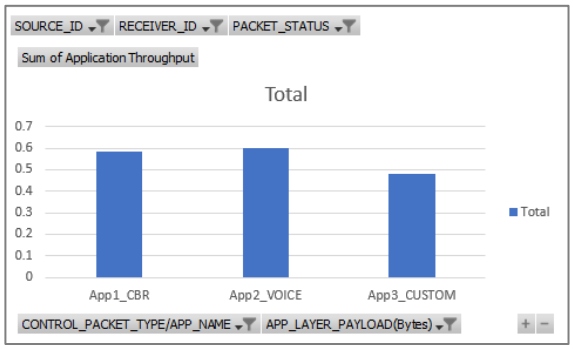


Figure 8-65: Pivot Chart

Note: The procedure may vary with different versions of excel, the given procedure is according to the Excel 2017.

8.4.9 Packet Trace Fields

GENERAL FIELDS	DESCRIPTION
PACKET_ID	Specifies the ID of the Data Packets. For control packets this value is set to 0 For every application packet IDs are assigned in serial order. The Packet ID is not a unique number. It is the tuple {Application ID, Packet_ID} that is unique.
SEGMENT_ID	Specifies the ID of the segment of the Data Packet. Segmentation is done in transport layer. If the packet size (generated in the APP layer) is greater than the maximum segment size in TRANSPORT layer, packet will get segmented. For control packets it is N/A
PACKET_TYPE	Specifies the type of application that generates the packet. It can be Control Packet, Custom, CBR, Peer_to_peer, E-Mail, DataBase, FTP, Video, Voice, HTTP.
CONTROL_PACKET_TYPE	Specifies the type of Control Packet transmitted. Following are the Protocol specific control packets WLAN: WLAN_ACK, WLAN_BlockACK OSPF: OSPF_HELLO, OSPF_D-D, OSPF_LSR, OSPF_LSU, OSPF_LSA RIP: RIP_Message GSM: GSM_Channel_Request, GSM_Channel_Granted, GSM_Call_Request, GSM_Channel_Request_For_Incoming, GSM_Call_Accepted CDMA: CDMA_Channel_Request, CDMA_Channel_Granted, CDMA_Call_Request, CDMA_Channel_Request_For_Incoming, CDMA_Call_Accepted DSR, AODV, ZRP, OLSR: RREQ, RREP, NDP_HELLO_MESSAGE, OLSR_TC_MESSAGE Zigbee: Zigbee_BEACON_FRAME, Zigbee_ACK Cognitive Radio: SCH, FCH, DS-MAP, US-MAP, UCD, DCD, BW_REQUEST, UCS_NOTIFICATION LTE: LTE_Measurement_Report, LTE_RRC_CONNECTION_SETUP, LTE_RLC_SDU, LTE_RRC_CONNECTION_REQUEST, LTE_RRC_CONNECTION_SETUP_COMPLETE, LTE page, LTE Ack etc.
SOURCE_ID	Specifies the <Device-type>-<ID> of the source set in the application. Note that if the device name is changed the new name will not reflect in the trace.
DESTINATION_ID	Specifies the <Device-type>-<ID> of the destination set in the application. Note that if the device name is changed the new name will not reflect in the trace. If the application is a broadcast application the destination field will show 0
TRANSMITTER_ID	Specifies the <Device-type>-<ID> of the current node which is transmitting the packet. Note that if the device name is changed the new name will not reflect in the trace. The difference between a Source node and a Transmitter, is that when the Source remains constant across the entire packet transmission whereas the transmitter ID changes with each hop of the packet.
RECEIVER_ID	Specifies the <Device-type>-<ID> of the current node which is receiving the packet. Note that if the device name is changed the new name will not reflect in the trace. The difference between a Destination node and a Receiver, is that when the Destination remains constant across the entire packet transmission whereas the receiver ID changes with each hop of the packet.
APP_LAYER_ARRIVAL_TIME (μs)	Specifies the time at which packet is at the Application_Layer of Source_ID (or Transmitter_ID). This is usually the time at which the packet is generated at Source_ID

TRX_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches the Transport_layer from the application layer. This will usually be the same as Application_layer_Arrival_Time unless there are TCP re-transmissions
NW_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches the Network_Layer of Transmitter_ID if this is a Router (or) Time at which packet reaches the Network_layer of previous Router / Source_ID (immediate previous Layer 3 or higher device) if current device is Switch / Access Point.
MAC_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches MAC_Layer of Transmitter_ID
PHY_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches PHY_layer of Transmitter_ID
PHY_LAYER_START_TIME (µs)	Specifies the time at which packet starts betting transmitted in the link between Transmitter_ID and Receiver_ID
PHY_LAYER_END_TIME (µs)	Specifies the time at which packet reaches Phy_Layer of Receiver_ID
APP_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Application Layer
TRX_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Transport Layer
NW_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Network Layer
MAC_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Data Link Layer
PHY_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Physical Layer
PHY_LAYER_OVERHEAD (Bytes)	Specifies the size of the overhead in Physical layer
PACKET_STATUS	Specifies whether the Packet is Successful, Collided or Errored
LOCAL_ADDRESS	Specifies the Port Number at Source Node. Port Numbers are chosen randomly by NetSim.
REMOTE_ADDRESS	Specifies the Port Number at Destination Node. Port Numbers are chosen randomly by NetSim.
CWND (bytes)	Specifies the current size of the TCP congestion window
SEQ_NO	If TCP is enabled, it specifies the TCP Sequence number of the packet
ACK_NO	If TCP is enabled, it specifies the TCP Acknowledgement number of the packet
isSyn	If TCP is enabled, it specifies whether the packet is TCP_SYN or not
isAck	If TCP is enabled, it specifies whether the packet is TCP_ACK/TCP_SYN_ACK or not
isFin	If TCP is enabled, it specifies whether the packet is TCP_FIN or not
SEGMENT_LENGTH	Specifies the segment length of the packet
SOURCE_IP	Specifies the IP address of the source
DESTINATION_IP	Specifies the IP address of the destination
GATEWAY_IP	Specifies the IP address of the device which is transmitting a packet
NEXT_HOP_IP	Specifies the IP address of the next hop

Table 8-3: Packet Trace Fields and Description

Note:

- Each line in the packet trace represents one hop of one packet.
- The packet trace is logged in ascending order of time as measured in Phy_Layer_End_Time.

8.5 Event Trace (only in Standard/Pro Version)

8.5.1 NetSim Network Stack and Discrete Event Simulation working

NetSim's Network Stack forms the core of NetSim and its architectural aspects are diagrammatically explained below. It exactly mirrors the TCP/IP stack and has the following five layers.

- Application Layer – CBR, Voice, Video, HTTP, etc.
- Transport Layer – TCP, UDP
- Network Layer – IP, OSPF, AODV, OLSR etc.
- MAC Layer – 802.11, 802.15.4, LTE etc.
- Physical Layer – Wired (P2P, P2MP, MP2MP), Wireless (RF Propagation)

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. In addition to these events there can be ***TIMER*** events associated with each protocol.

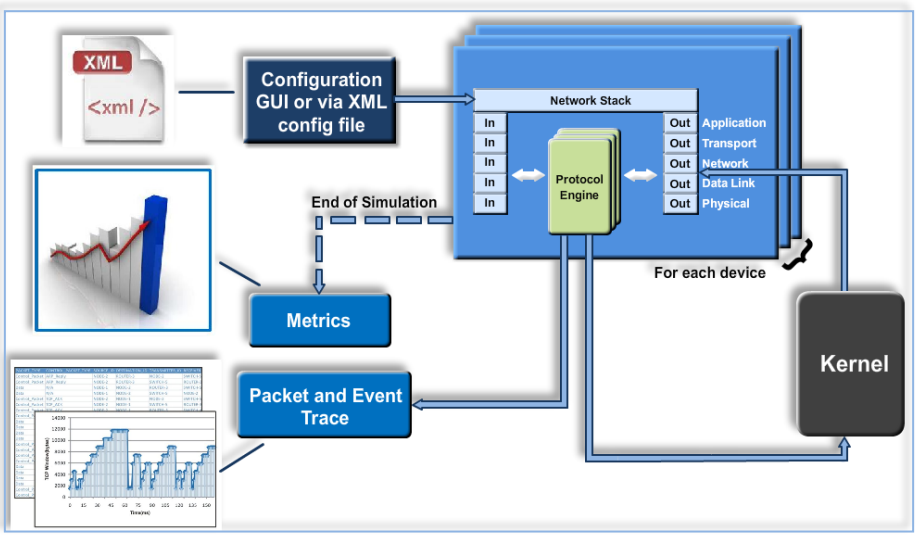


Figure 8-66: Flow of one packet from a Wired node to a Wireless node

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 and TIMER events if applicable.

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.

8.5.2 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. Apart from a host of information, the event trace has two special information fields for diagnostics.

- A log of the file name and line number from where the event was generated (Please refer **"Writing Custom Code in NetSim → Debugging your code → Via CLI"**) and
- Previous event which triggered the current event.

Note: Turning on Event Trace will slow down the simulation significantly

NetSim provides users with the option of turning on "Event Traces".

How to enable Event Trace via GUI?

If NetSim runs via GUI, event trace can be enabled by clicking on Configure Reports tab and unable Event trace.

How to enable Event Trace via CLI?

If NetSim runs via CLI, then the event trace can be turned on by enabling the event trace in the STATISTICS_COLLECTION tag of the configuration file. Following is a screenshot of a Configuration.netsim file with Event Trace disabled:

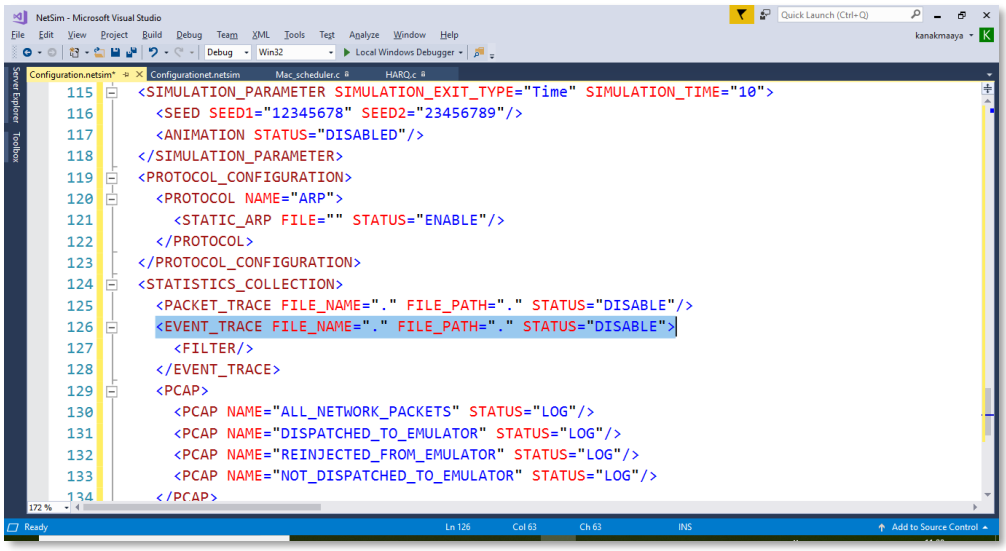


Figure 8-67: Open Configuration.netsim in Visual Studio and Event Trace disabled

You can see that the STATUS is set to DISABLE, file name and file path are not set. To enable Event trace these parameters can be modified by editing the Configuration file. Open Configuration.netsim file and provide the file name, path and set status as Enable. Following is a screenshot of a Configuration.netsim file with Event Trace enabled:

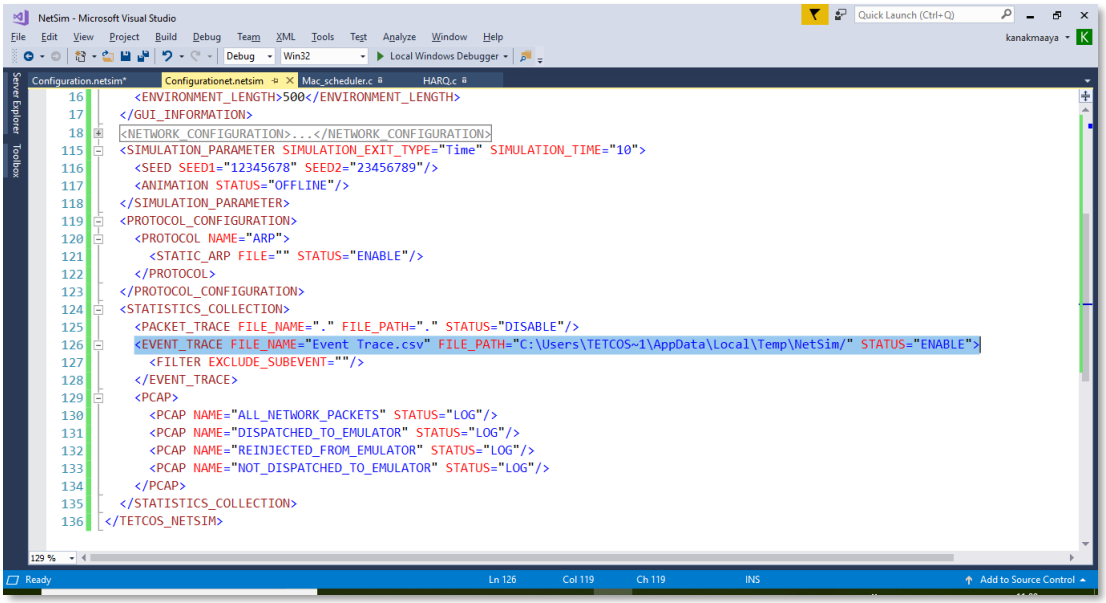


Figure 8-68: Event Trace enabled in Configuration.netsim file

Event Trace Metrics:

Event_Id	Specifies the ID of the Event
Event_Type	Specifies the type of event being performed, for e.g. - APPLICATION_IN, APPLICATION_OUT, MAC_OUT, MAC_IN, PHYSICAL_OUT, PHYSICAL_IN, etc.
Event_Time	Specifies the time (in microseconds) at which the event is being executed
Device_Type	Specifies the type of device in which the current event is being executed

Device_Id	Specifies the ID of device in which the current event is being executed
Interface_Id	Specifies the Interface_Id of device in which the present event is being executed.
Application_Id	Specifies the ID of the Application on which the specific event is executed
Packet_Id	Specifies the ID of the packet on which the current event is being executed
Segment_Id	Specifies the ID of the segment of packet on which the current event is being executed
Protocol_Name	Specifies the Protocol which is presently executed
Subevent_Type	Specifies the protocol sub event which is being executed. If the sub event value is 0, it indicates interlayer communication (Ex: MAC_OUT called by NETWORK_OUT) or a TIMER_EVENT which has no sub event.
Packet_Size	Specifies the size of packet during the current event
Prev_Event_Id	Specifies the ID of the event which generated the current event.

Table 8-4: Event Trace fields and Descriptions

8.5.3 Calculation of Delay and Application throughput from event trace

1. Consider the scenario as explained in the section **8.4.5** Delay analysis.

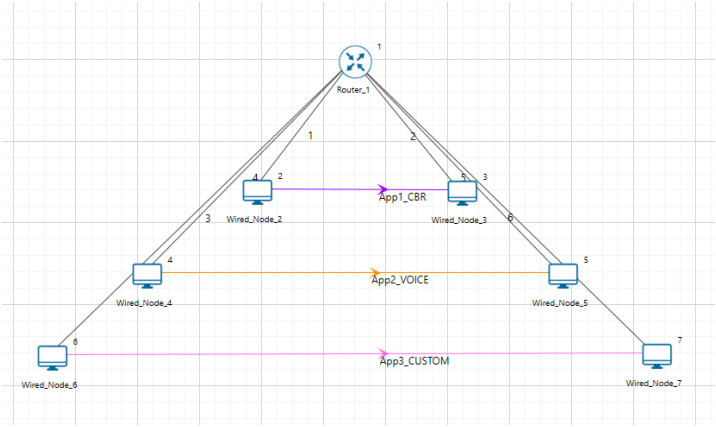


Figure 8-69: Network Scenario to calculate delay and throughput

2. Enable Event trace and simulate the scenario for 10 seconds,
3. Open event trace from the simulation results windows as shown in the below Figure 8-70.

Note: Event tracing is available only in NetSim standard and pro versions.

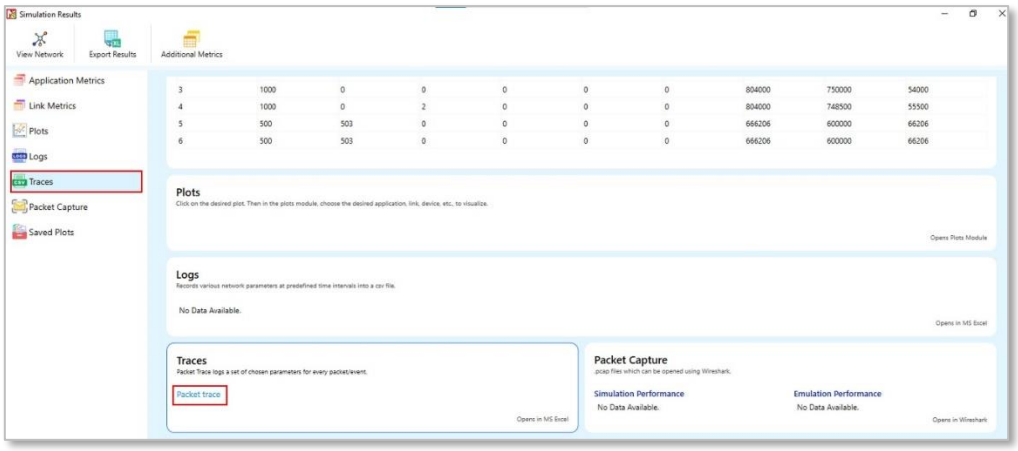
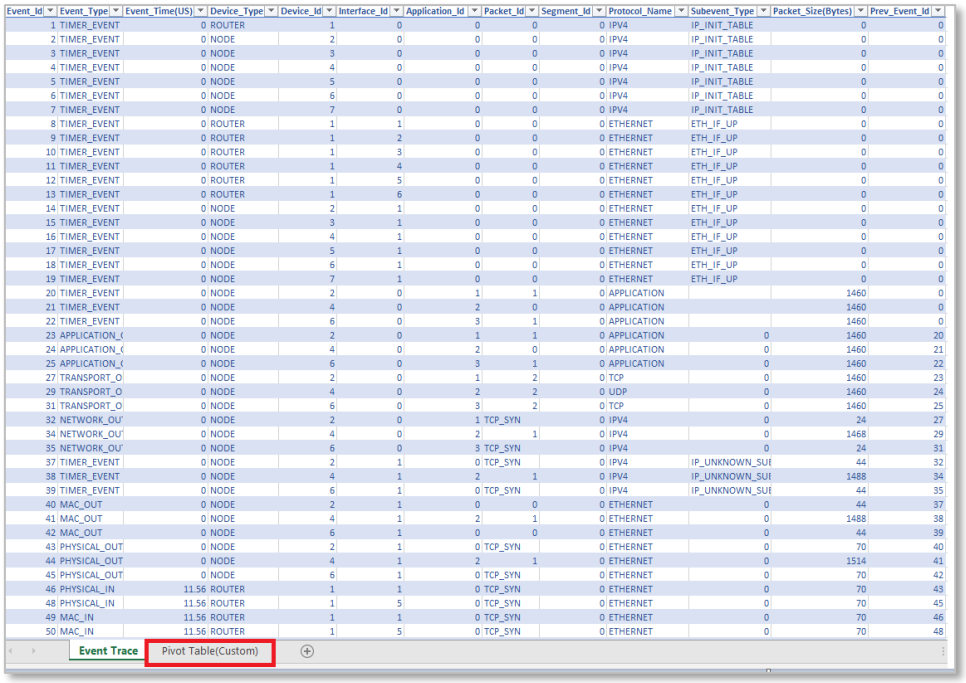


Figure 8-70: Select Event Trace option in results window.

4. Click on **Pivot Table (Custom)** in excel sheet as shown below.



Event_Id	Event_Type	Event_Time(US)	Device_Type	Device_Id	Interface_Id	Application_Id	Packet_Id	Segment_Id	Protocol_Name	Subevent_Type	Packet_Size(Bytes)	Prev_Event_Id
1	TIMER_EVENT	0	ROUTER	1	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
2	TIMER_EVENT	0	NODE	2	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
3	TIMER_EVENT	0	NODE	3	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
4	TIMER_EVENT	0	NODE	4	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
5	TIMER_EVENT	0	NODE	5	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
6	TIMER_EVENT	0	NODE	6	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
7	TIMER_EVENT	0	NODE	7	0	0	0	0	IPV4	IP_INIT_TABLE	0	0
8	TIMER_EVENT	0	ROUTER	1	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
9	TIMER_EVENT	0	ROUTER	1	2	0	0	0	ETHERNET	ETH_IF_UP	0	0
10	TIMER_EVENT	0	ROUTER	1	3	0	0	0	ETHERNET	ETH_IF_UP	0	0
11	TIMER_EVENT	0	ROUTER	1	4	0	0	0	ETHERNET	ETH_IF_UP	0	0
12	TIMER_EVENT	0	ROUTER	1	5	0	0	0	ETHERNET	ETH_IF_UP	0	0
13	TIMER_EVENT	0	ROUTER	1	6	0	0	0	ETHERNET	ETH_IF_UP	0	0
14	TIMER_EVENT	0	NODE	2	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
15	TIMER_EVENT	0	NODE	3	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
16	TIMER_EVENT	0	NODE	4	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
17	TIMER_EVENT	0	NODE	5	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
18	TIMER_EVENT	0	NODE	6	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
19	TIMER_EVENT	0	NODE	7	1	0	0	0	ETHERNET	ETH_IF_UP	0	0
20	TIMER_EVENT	0	NODE	2	0	1	1	0	APPLICATION		1460	0
21	TIMER_EVENT	0	NODE	4	0	2	0	0	APPLICATION		1460	0
22	TIMER_EVENT	0	NODE	6	0	3	1	0	APPLICATION		1460	0
23	APPLICATION_(	0	NODE	2	0	1	1	0	APPLICATION	0	1460	20
24	APPLICATION_(	0	NODE	4	0	2	0	0	APPLICATION	0	1460	21
25	APPLICATION_(	0	NODE	6	0	3	1	0	APPLICATION	0	1460	22
27	TRANSPORT_O	0	NODE	2	0	1	2	0	TCP	0	1460	23
29	TRANSPORT_O	0	NODE	4	0	2	2	0	UDP	0	1460	24
31	TRANSPORT_O	0	NODE	6	0	3	2	0	TCP	0	1460	25
32	NETWORK_OU	0	NODE	2	0	1	TCP_SYN	0	IPV4	0	24	27
34	NETWORK_OU	0	NODE	4	0	2	1	0	IPV4	0	1468	29
35	NETWORK_OU	0	NODE	6	0	3	TCP_SYN	0	IPV4	0	24	31
37	TIMER_EVENT	0	NODE	2	1	0	TCP_SYN	0	IPV4	IP_UNKNOWN_SUI	44	32
38	TIMER_EVENT	0	NODE	4	1	2	1	0	IPV4	IP_UNKNOWN_SUI	1488	34
39	TIMER_EVENT	0	NODE	6	1	0	TCP_SYN	0	IPV4	IP_UNKNOWN_SUI	44	35
40	MAC_OUT	0	NODE	2	1	0	0	0	ETHERNET	0	44	37
41	MAC_OUT	0	NODE	4	1	2	1	0	ETHERNET	0	1488	38
42	MAC_OUT	0	NODE	6	1	0	0	0	ETHERNET	0	44	39
43	PHYSICAL_OUT	0	NODE	2	1	0	TCP_SYN	0	ETHERNET	0	70	40
44	PHYSICAL_OUT	0	NODE	4	1	2	1	0	ETHERNET	0	1514	41
45	PHYSICAL_OUT	0	NODE	6	1	0	TCP_SYN	0	ETHERNET	0	70	42
46	PHYSICAL_IN	11.56	ROUTER	1	1	0	TCP_SYN	0	ETHERNET	0	70	43
48	PHYSICAL_IN	11.56	ROUTER	1	5	0	TCP_SYN	0	ETHERNET	0	70	45
49	MAC_IN	11.56	ROUTER	1	1	0	TCP_SYN	0	ETHERNET	0	70	46
50	MAC_IN	11.56	ROUTER	1	5	0	TCP_SYN	0	ETHERNET	0	70	48

Event Trace | Pivot Table(Custom)

Figure 8-71: Select Pivot Table (Custom) in excel sheet

5. A blank **PivotTable** and **Field List** will appear on a new worksheet.

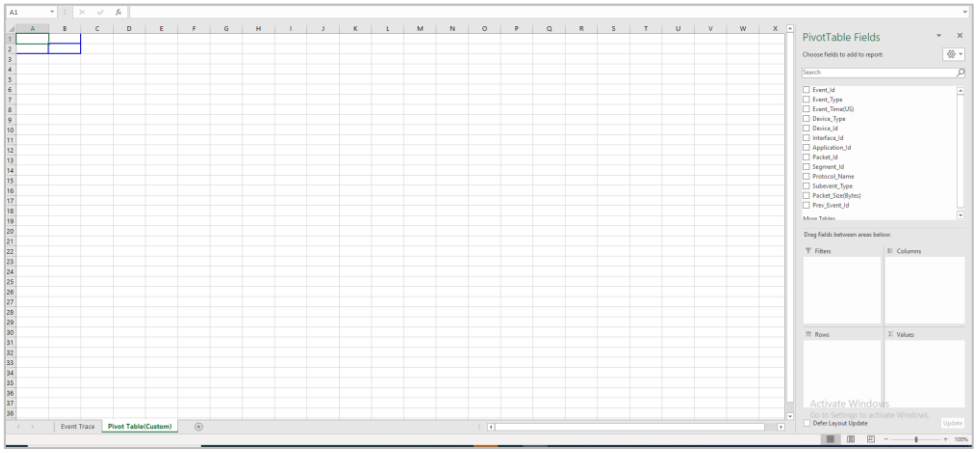


Figure 8-72: A blank PivotTable

6. Once PivotTable worksheet open, you will need to decide which fields to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

8.5.3.1 Application Delay Analysis:

1. Drag and drop the Event_Type, Protocol_Name Fields into FILTERS, Packet_Id into ROWS and Device_Id into COLUMNS.
2. Drag and Drop Event_Time Field into VALUES twice, then both will show Sum of Event_Time. Recheck that you have dropped the Event_Time field twice.

3. Click on the second Event_Time field in the VALUES and select the Value Field Settings.

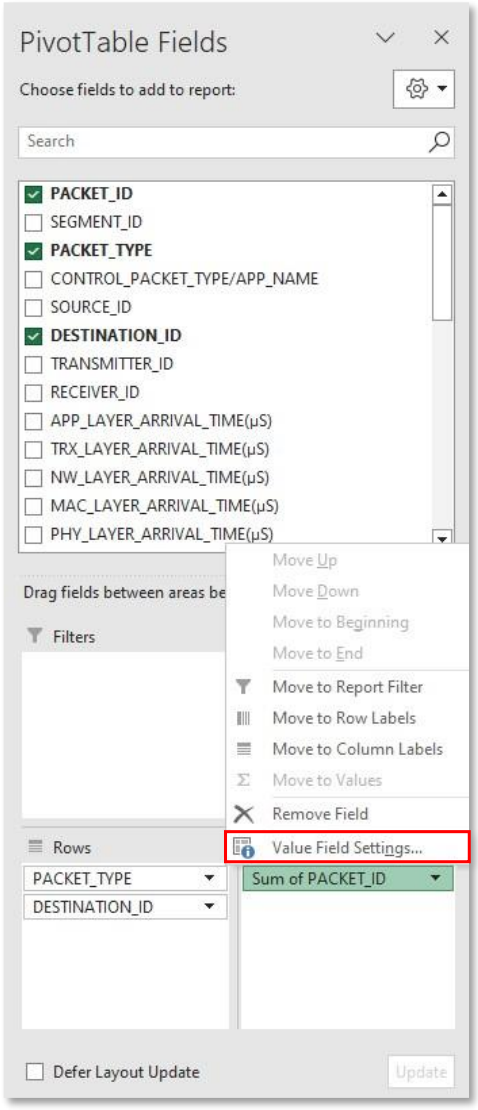


Figure 8-73: Select Second Event_Time field in the VALUES and select the Value Field Settings

4. A window named **Value Field Settings** opens then select **Count** option and click **OK** .

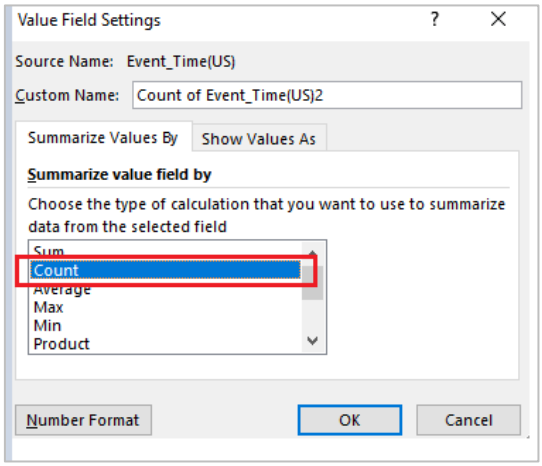


Figure 8-74: Select Summarize value field by Count

5. Then finally the **Pivot Table Fields** will be as shown below Figure 8-75.

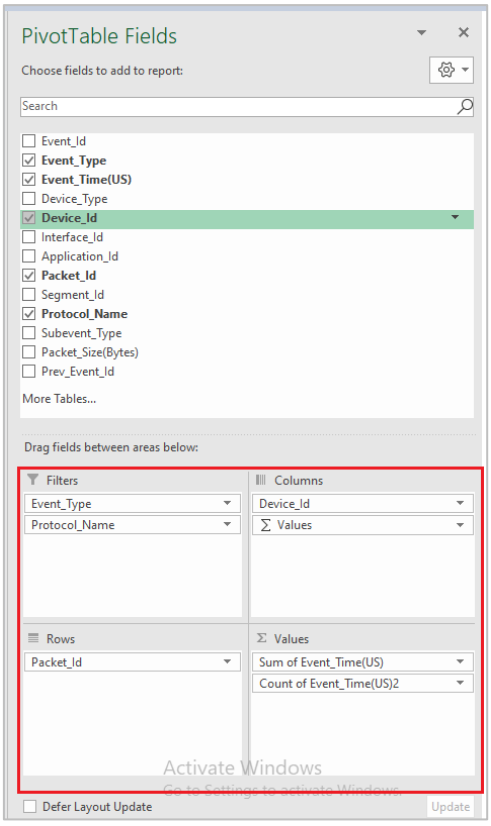


Figure 8-75: Selected Fields to one of the four areas Field list

6. In the Event_Type select APPLICATION_IN and APPLICATION_OUT, **Protocol_Name** select **APPLICATION** and in **Column Labels** select the **Source_Id** and **Destination_Id**. In our example source node ID is 2 and destination node ID is 3.

Note: After selecting the dropdown, to check and uncheck the check box proceed by selecting the selecting the multiple items check box

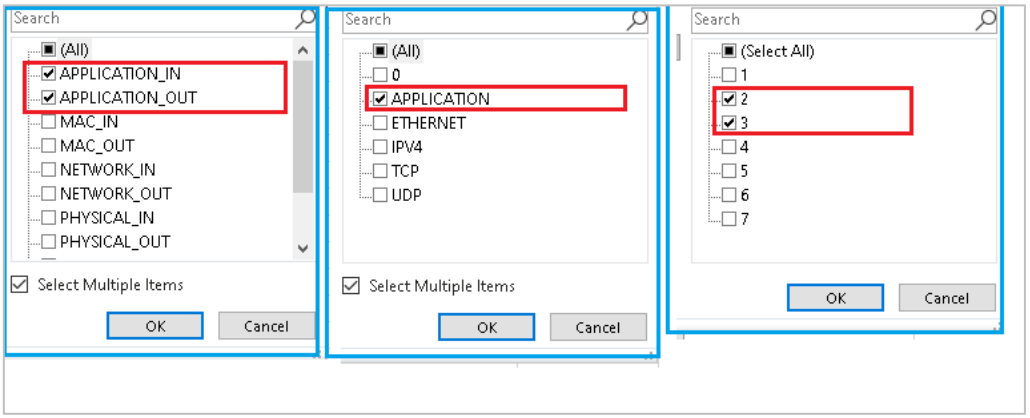


Figure 8-76: Select the Event type, Protocol Name, Source and Destination ID etc

7. The Pivot Table created will be as shown below.

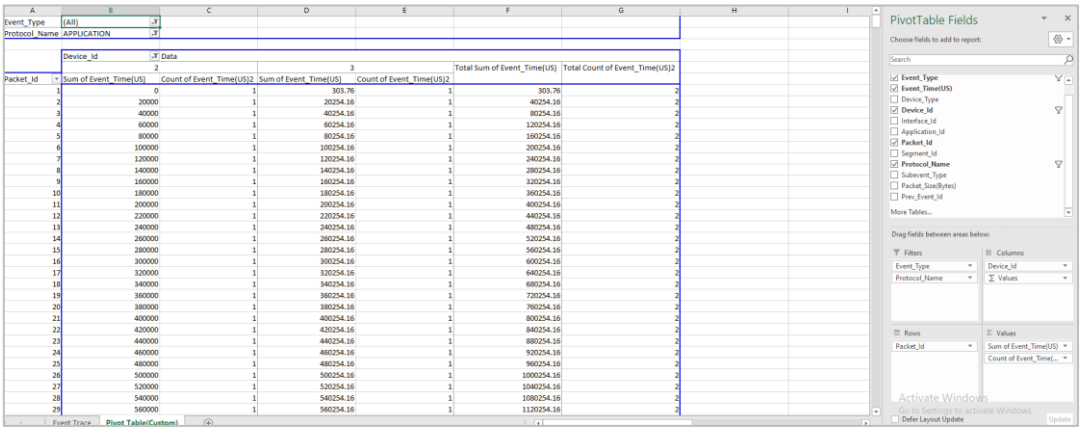


Figure 8-77: Created Pivot Table

8. Select the entire empty column H then and enter the formula **=IF(AND(LEN(A1), INT(A1)=A1),F1-G1*B1)** in function and press **CTRL+ENTER**

F column is Total Sum of Event_Time, G Column is Total Count of Event_Time, B Column is Sum of Event_time(µs) of the Source.

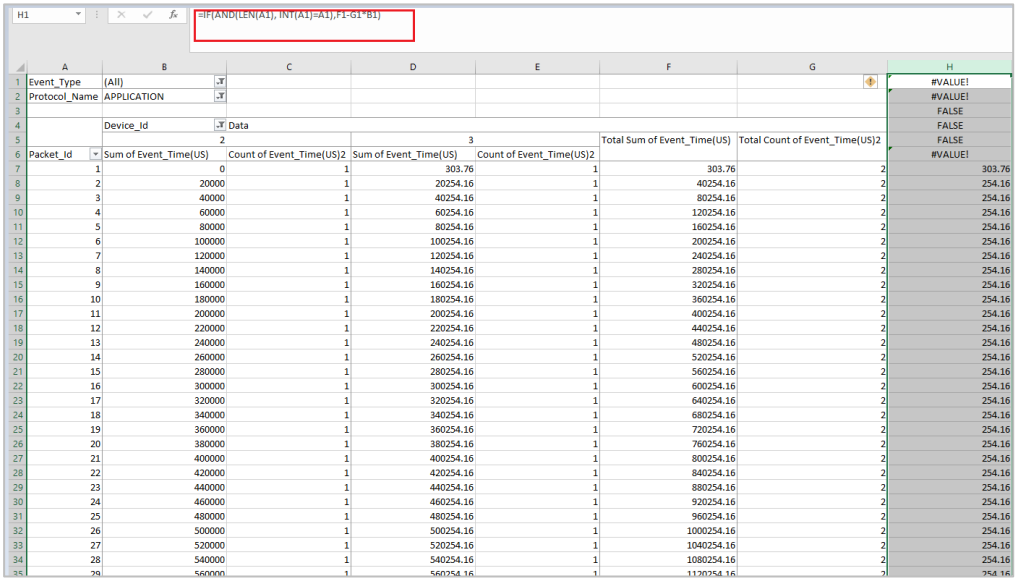


Figure 8-78: Select Entire empty column H then and enter the formula =IF(AND(LEN(A1), INT(A1) = A1), F1-G1*B1) in function and press CTRL+ENTER

$$App\ Delay = \frac{Sum\ of\ the\ Delays\ of\ the\ sucessfully\ received\ application\ data\ packets\ by\ the\ destination}{Total\ number\ of\ sucessful\ appplication\ data\ packets\ received\ by\ the\ destination}$$

Note: If the packet size is > 1500 then fragmentation occurs, and the packet is received as multiple segments. In NetSim the destination counts each segment as a different packet.

Then in an empty cell enter

=SUMIF(H:H,">0")/GETPIVOTDATA("Count of Event_Time(US)2",A4,"Device_Id",3) where

GETPIVOTDATA ("Count of Event_Time(US)2",A4,"Device_Id",3) gives the total number of packets received by the destination (in this case 3). This will give the exact Application Delay.

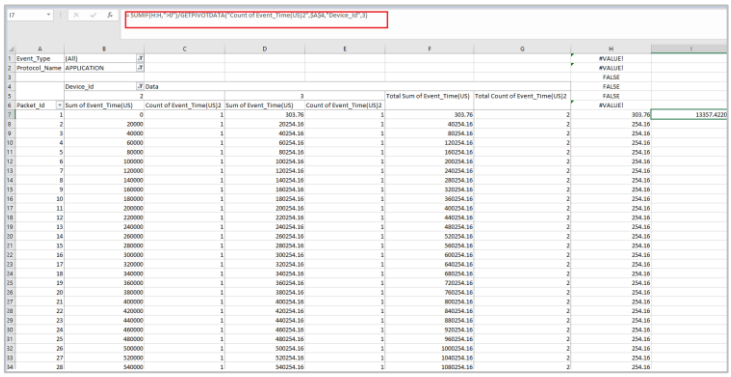


Figure 8-79: Calculated Application Delay using Formula in Pivot table

Compare with the Delay in Application_Metrics_Tables and it would exactly match. There might be slight difference in the decimals due to Excel's round offs.

Application Metrics

End-to-end performance of applications running across the network.

Application ID	Application Name	Source ID	Destination ID	Throughput (Mbps	Delay (µs)
1	App1_CBR	2	3	0.584000	13357.422082
2	App2_VOICE	4	5	0.598800	258.881924
3	App3_CUSTOM	6	7	0.480000	212.659200

Figure 8-80: Compare the Application_Metrics_Tables Delay and Pivot table Delay

8.5.3.2 Application Throughput Analysis

1. For Application Throughput drag and drop **Event_type**, **Protocol_Name** Fields in **FILTERS**, **Device_Id** in **ROWS**, **Packet_Size(Bytes)** into **VALUES.** Change the **Value Field Settings** of **Packets_Size(Bytes)** to **SUM** as mentioned in Delay Analysis.

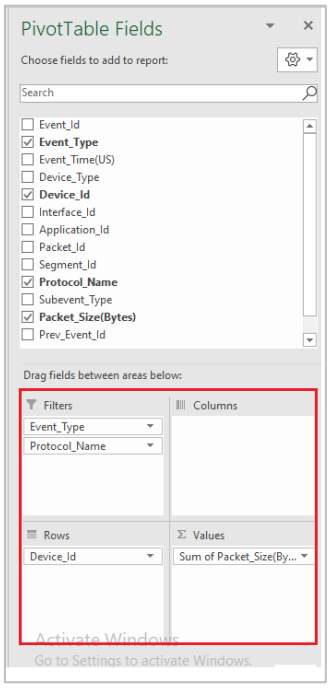


Figure 8-81: Selected Fields to one of the four areas Field list

Then Select the Event_Type as APPLICATION_IN, Protocol_Name as APPLICATION and Device_Id as the Destination (in this case destination id will be 3 since we are calculating for APP1 CBR)

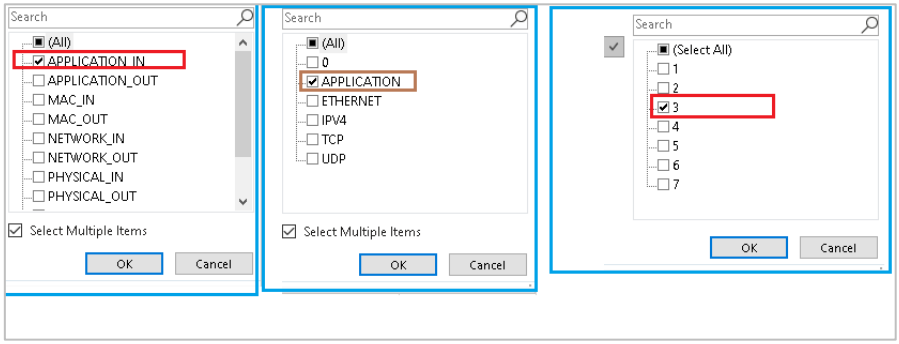


Figure 8-82: Select the Event type, Protocol Name, Source and Destination ID etc

$$\text{App Throughput} = \frac{Total Payload application succesfully received by the destination}{Simulation time}$$

Then in an empty cell type =GETPIVOTDATA ("Packet_Size(Bytes)",A4)*8/10000000

This gives the Application Throughput in Mbps (Multiplied by 8 to convert Bytes to bits, and divided by 100000 to convert into Mega)

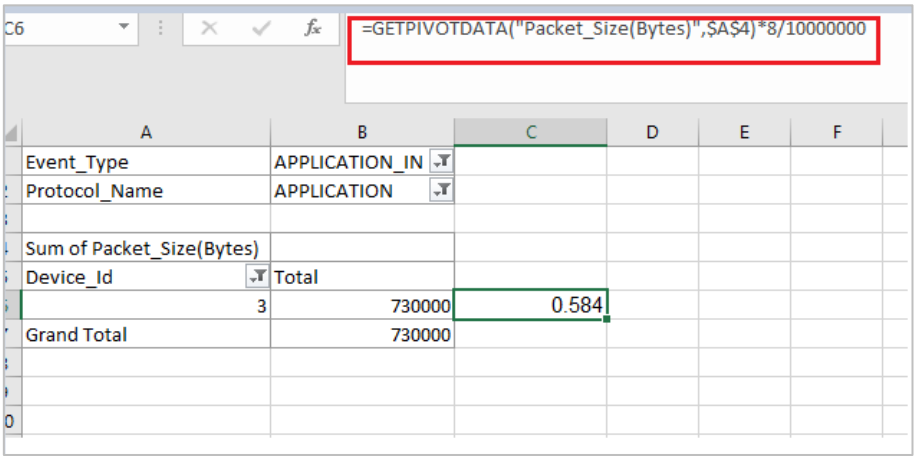
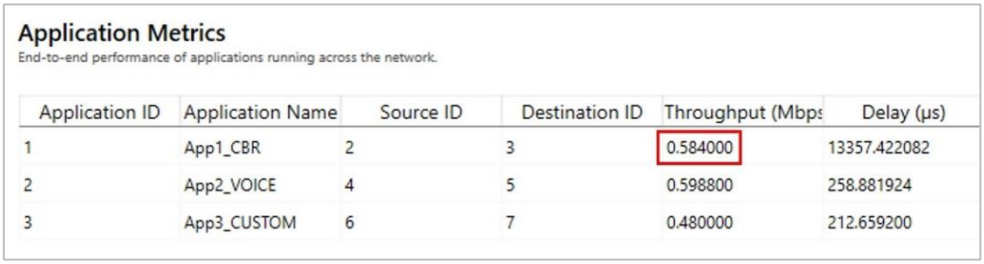


Figure 8-83: Calculate Application Throughput using formula

Compare with the Application throughput in the **Application_Metrics_Table**



Application Metrics

End-to-end performance of applications running across the network.

Application ID	Application Name	Source ID	Destination ID	Throughput (Mbps	Delay (µs)
1	App1_CBR	2	3	0.584000	13357.422082
2	App2_VOICE	4	5	0.598800	258.881924
3	App3_CUSTOM	6	7	0.480000	212.659200

Figure 8-84: Compare the Application_Metrics_Tables throughput and Pivot table throughput

8.6 Mobility Viewer

NetSim Mobility Viewer enables the user to observe the path taken by the nodes during the simulation. The Mobility Viewer is available for mobile ad-hoc networks, sensor networks, 4G, 5G networks, Cellular, VANETs, and underwater acoustic networks.

The mobility path for the file-based mobility model can be observed prior to the simulation, whereas the path for other models like random walk, random waypoint, group mobility, and pedestrian mobility can only be observed post-simulation.

To enable mobility viewer in the above networks, create a scenario with desired mobility model and click on configure tab and plots. Then in the right panel click on mobility log and click on run simulation.

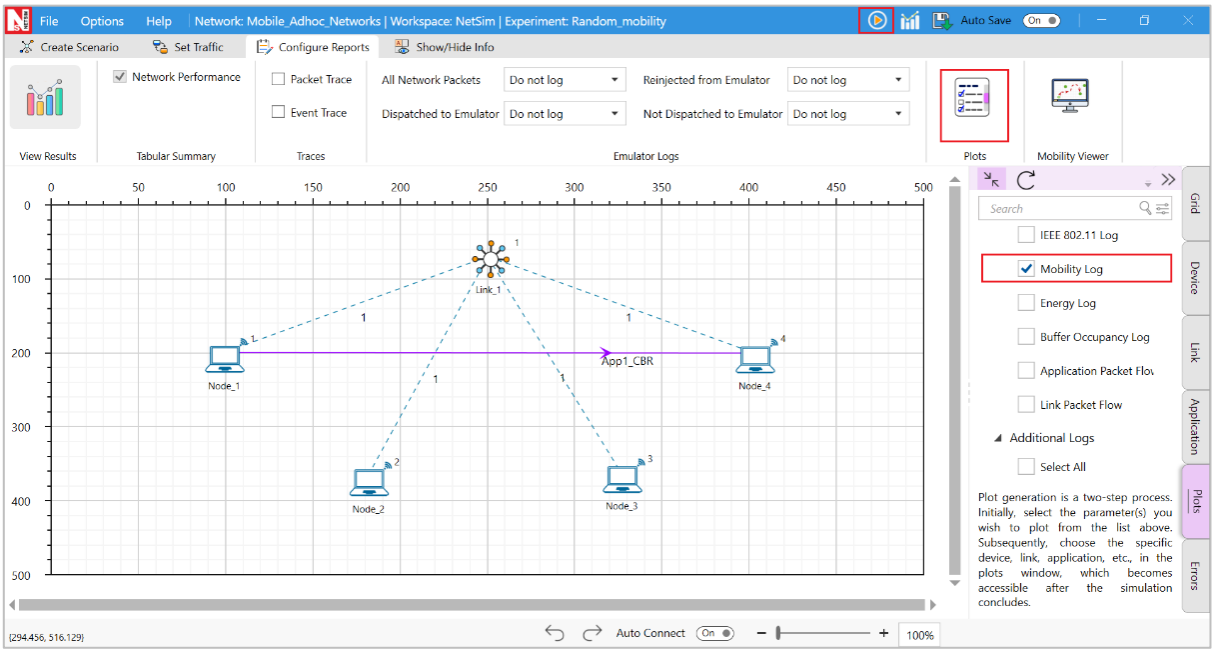


Figure 8-85: Enabling Mobility log prior to simulation

After simulation, in the design window click on the Mobility Viewer and set the properties and click on View Mobility Path.

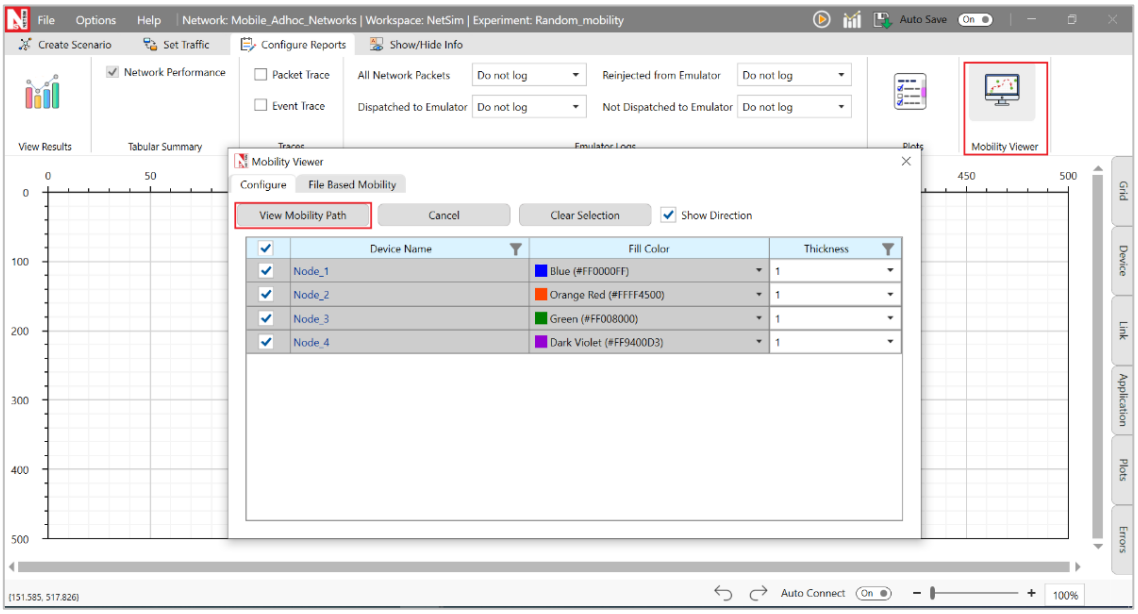


Figure 8-86: Mobility viewer property settings

In the image below, we have considered the Random Waypoint mobility model.

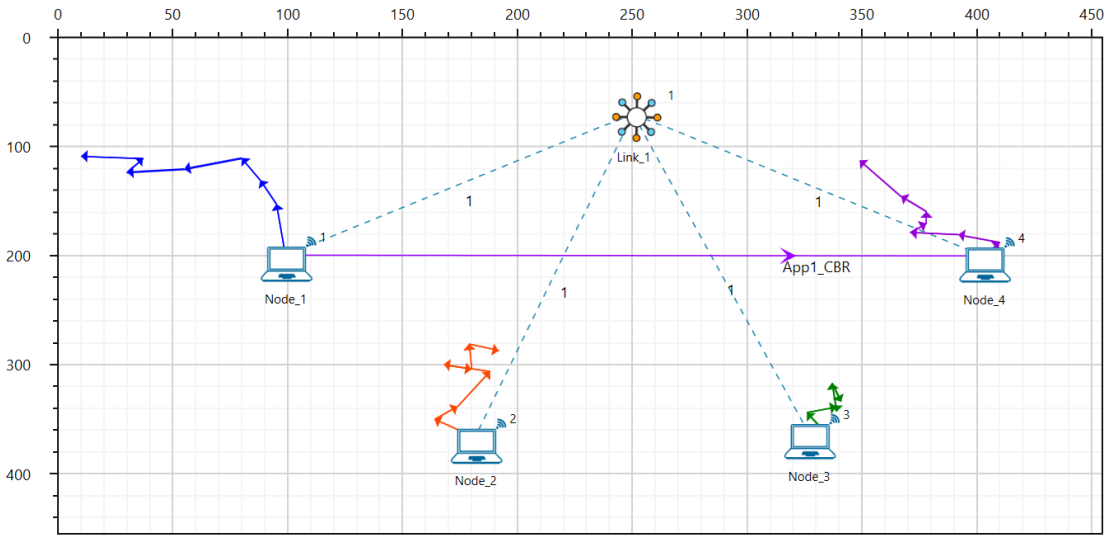


Figure 8-87: Mobility path for random way point model includes a pause time of 10 sec.

Similarly screenshots of the paths for other mobility models using the mobility viewer are shown below.

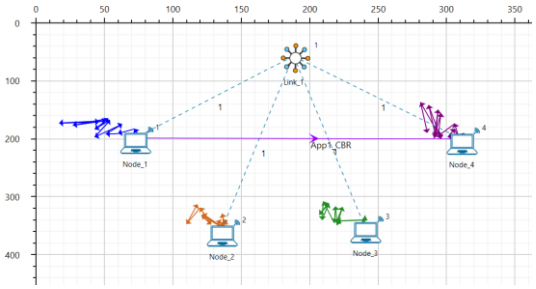


Figure 8-88:Mobility path for Random walk model

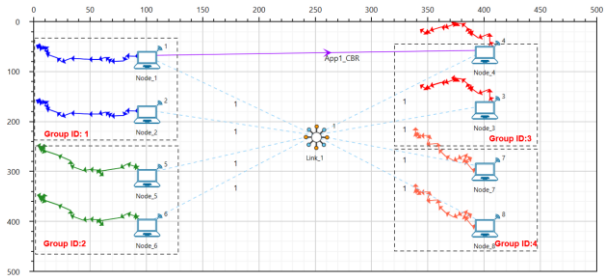


Figure 8-89: Mobility path for group mobility involving 8 nodes defined into 4 groups

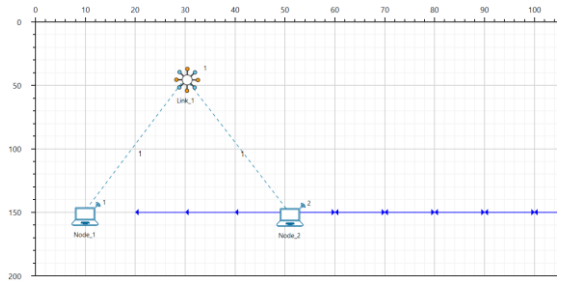


Figure 8-90: Mobility path for file-based mode where node 2 moves away and returns to the transmitter

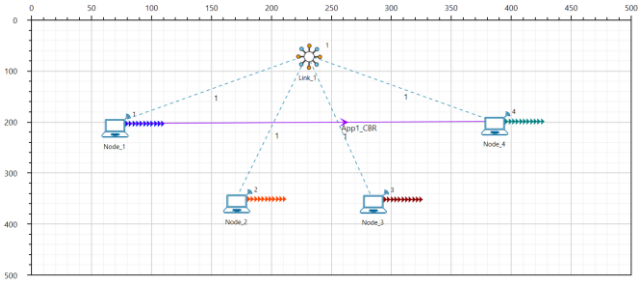


Figure 8-91: Mobility path for pedestrian model with a stop probability of 0

8.7 Packet Capture & analysis using Wireshark.

8.7.1 Enabling Wireshark Capture in a node for packet capture

NetSim provides functionality to capture packets in the virtual nodes. The pcap file written by NetSim contains fields of packet layer 3 and above. This pcap file can be opened using the popular software, Wireshark (formerly Ethereal).

To enable packet capture in Wireshark, Right Click on the device where wireshark should be run. In the properties, go to General Properties and set the Wireshark Capture parameter as Online.

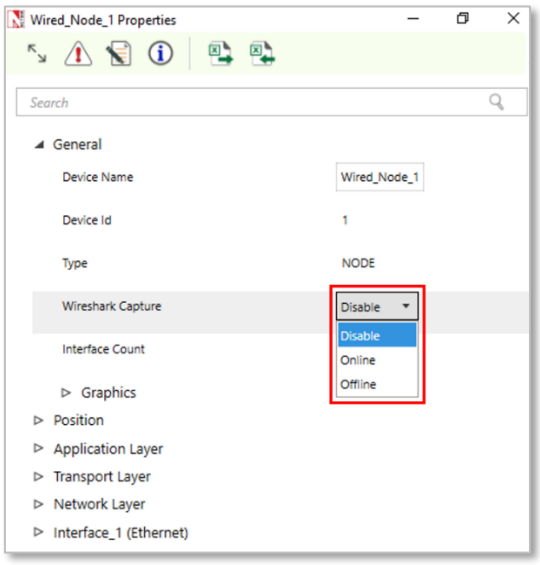


Figure 8-92: Enable Wireshark in General Properties for wired node

Wireshark Capture Options	
Online	Online option will initiate a live interactive packet capture, displaying packets while running simulation
Offline	Offline option will initiate silent packet capture and generate a pcap file which can be opened using Wireshark post-simulation
Disable	Packets are not captured by Wireshark during simulation.

Table 8-5: Wireshark Capture Options and Description

8.7.2 Viewing captured packets

If enabled, Wireshark Capture automatically starts during simulation and displays all the captured packets. To view the details of the packet displayed, click-on the packet as shown below Figure 8-93.

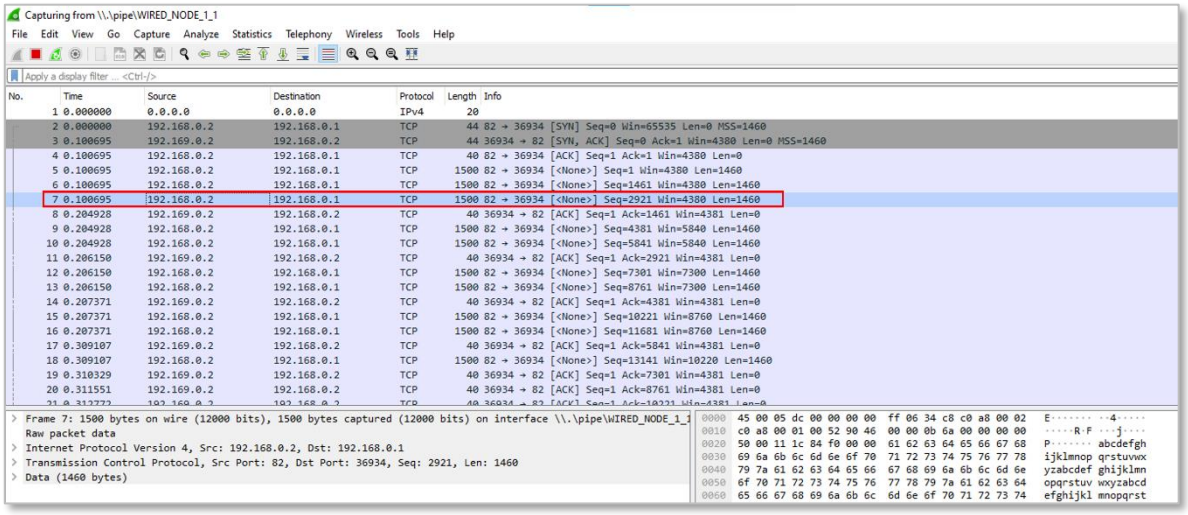


Figure 8-93: Packets Captured in Wireshark

The detail of the contents of the selected packet can be seen in the below panes as shown below Figure 8-94.

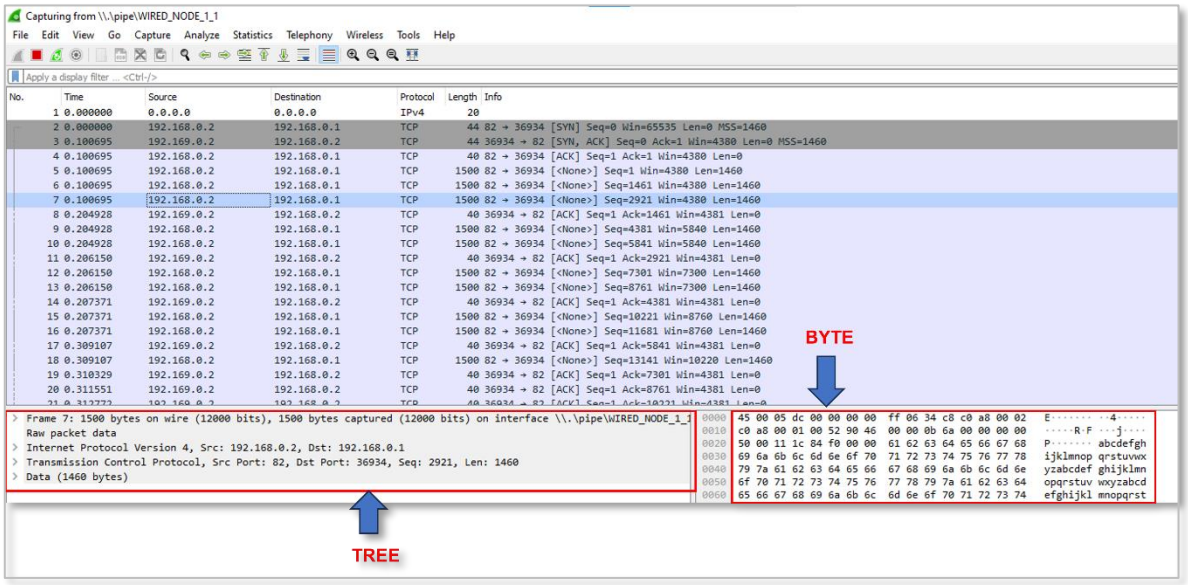


Figure 8-94: Packet Information panes

In the above figure, the details of the packet are displayed in both tree form and bytes form. In the tree form, user can expand the data by clicking on the part of the tree and view detailed information about each protocol in each packet.

8.7.3 Filtering captured packets

Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. Packets can be filtered by protocol, presence of a field, values of field's etc. To select packets based on protocol, type the protocol in which you are interested in the Filter: field of the Wireshark window and presenter to initiate the filter. In the figure below Figure 8-95, tcp protocol is filtered.

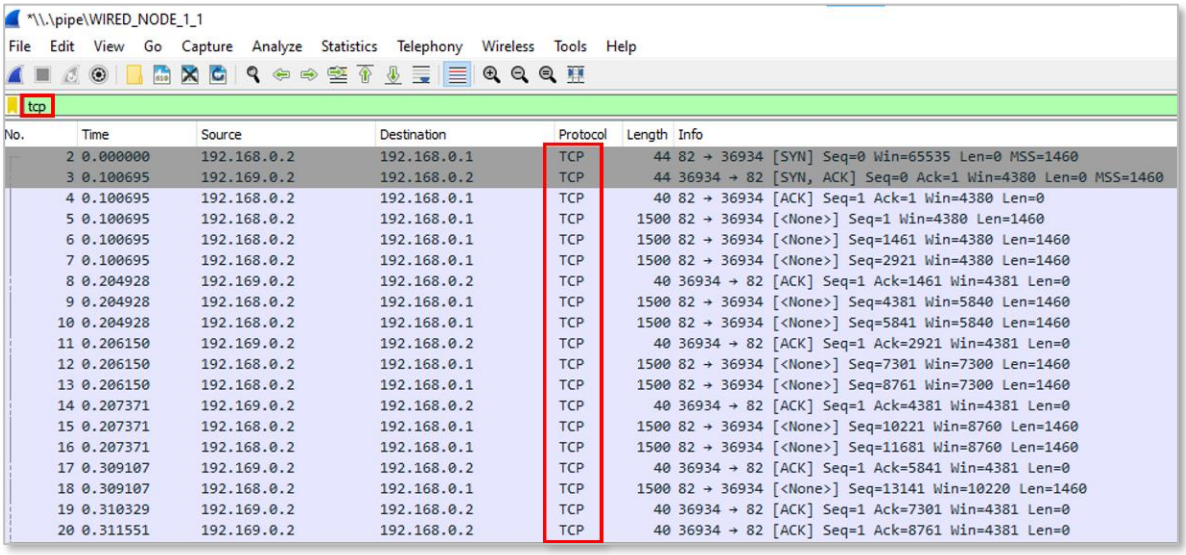


Figure 8-95: TCP Protocol is filtered in Wireshark

You can also build display filters that compare values using a number of different comparison operators like ==, != , >, <, <=, etc. Following is an example displaying filtered packets whose SYN Flag and ACK Flag are set to 1 in a TCP Stream.

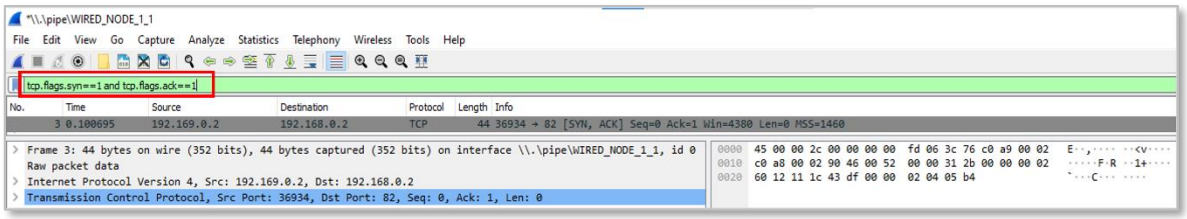


Figure 8-96: Filtered SYN Flag and ACK Flag are set to 1 in a TCP Stream

8.7.4 Analyzing packets in Wireshark

8.7.4.1 Analyzing Conversation using graphs

A network conversation is the traffic between two specific end points. For example, an IP conversation is all the traffic between two IP addresses. In Wireshark, Go to Statistics Menu→ Conversations as shown below Figure 8-97.

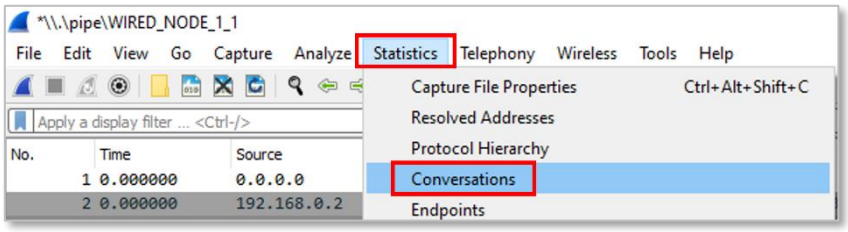


Figure 8-97: In Wireshark, Go to Statistics Menu >Conversations

Different types of protocols will be available. User can select the specific conversation by going to the desired protocol. For example, in the following diagram, we have selected TCP.

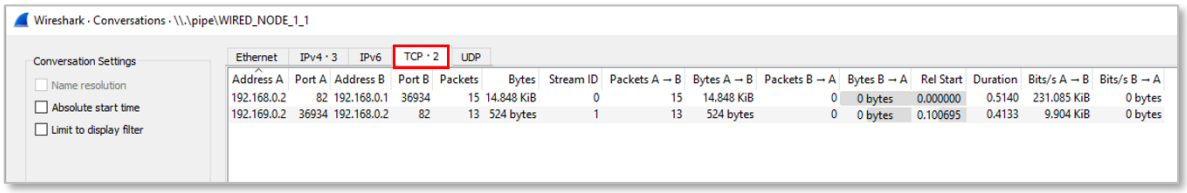


Figure 8-98: TCP Wireshark Conversion for Wired Nodes

User can also analyze each of the conversation and can create graphs by selecting them and clicking on **"Graph".**

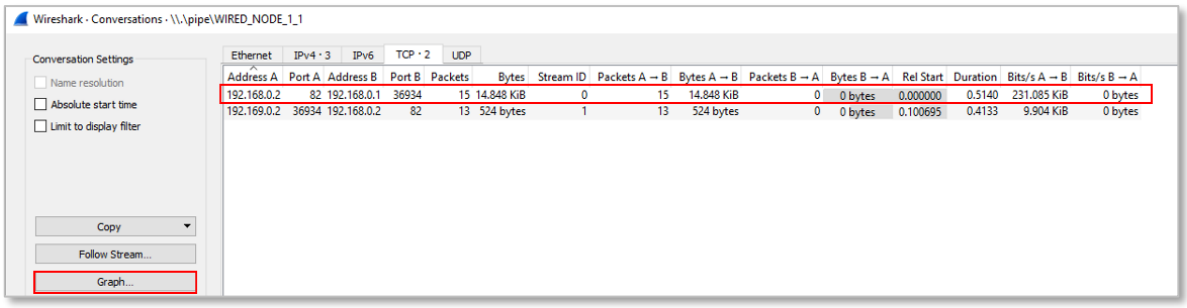


Figure 8-99: Select Graph in Wireshark Conversion

Different types of graphs are possible for Round Trip time, Throughput, Time/Sequence (Stevens), Time/Sequence (tcptrace) and Window Scaling

8.7.5 Window Scaling

Click on data packet i.e. <None>.

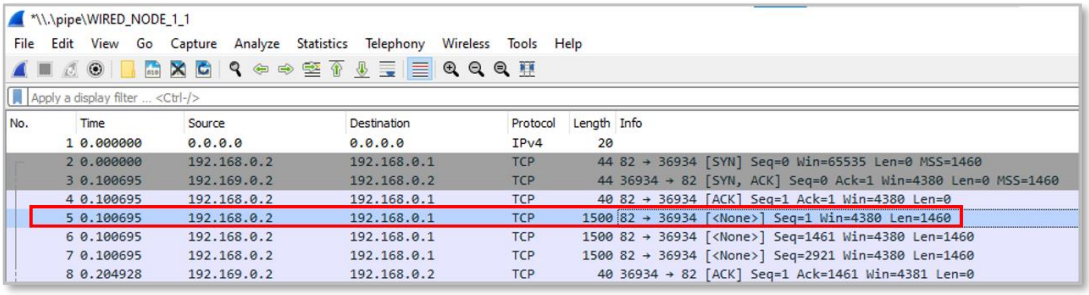


Figure 8-100: Select one data Packet <None>in Wireshark

Choose statistics→TCP Stream Graph→Window Scaling.

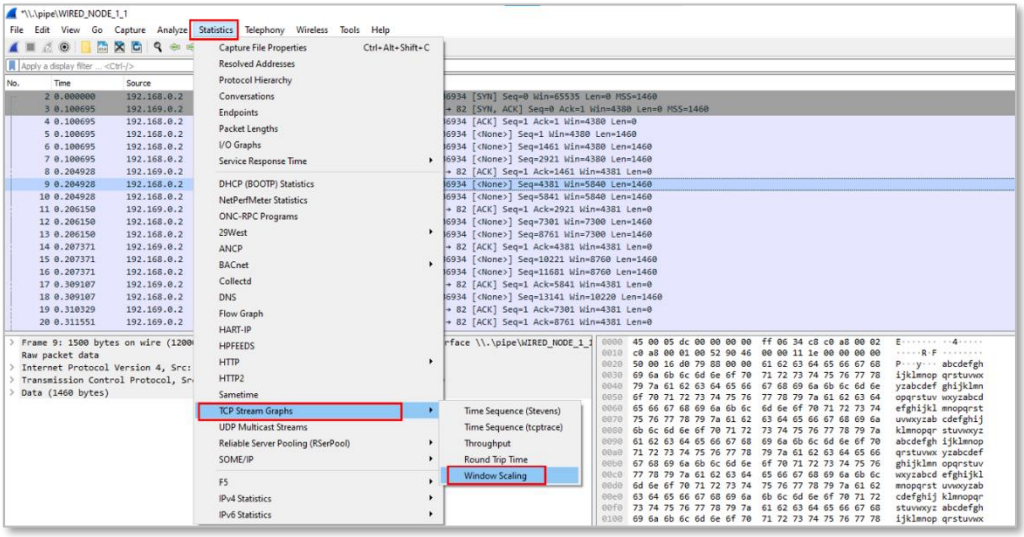


Figure 8-101: Statistics>TCP Stream Graph>Window Scaling

Click on Switch Direction in the window scaling graph window.

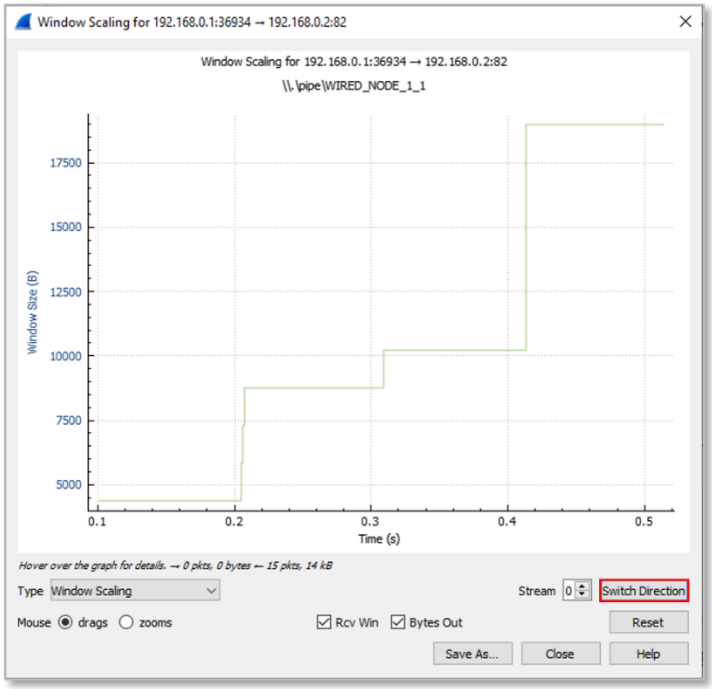


Figure 8-102: TCP Window Scaling Plot

8.7.5.1 Comparing the packet lengths

To analyze the packet sizes of all packets transmitted, go to ***Statistics Menu→Packet lengths***. Users can also set filter to analyze a collection of specific packets only. For example, tcp filter is set to obtain the packet length below Figure 8-103.

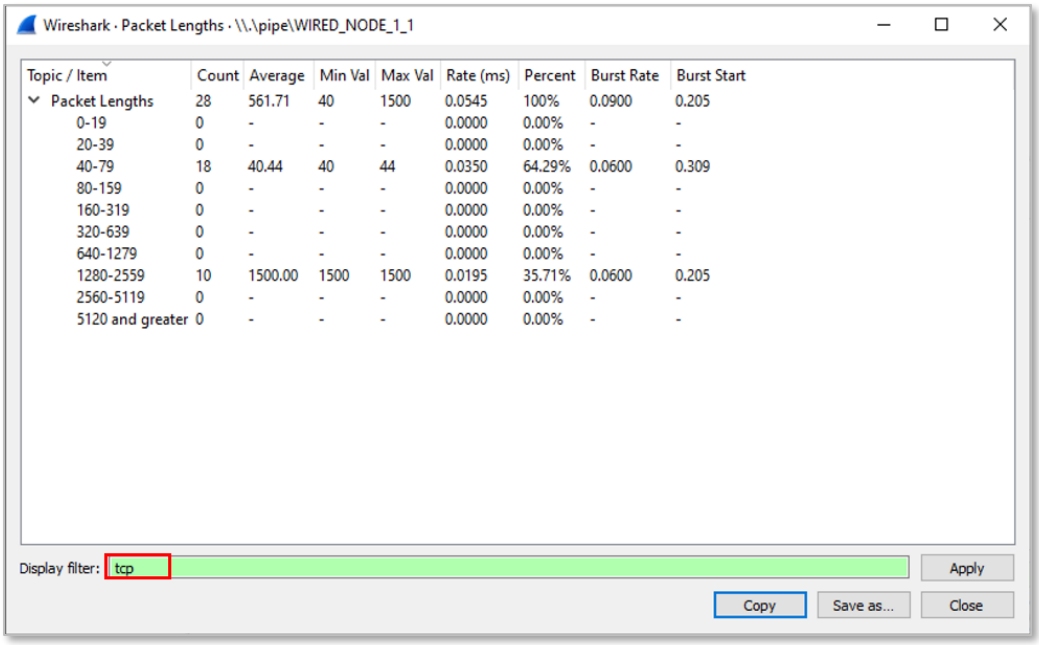


Figure 8-103: Comparing the packet lengths in Wireshark.

8.7.5.2 Creating IO graphs

To get the graph, go to ***Statistics Menu → IO Graph***.

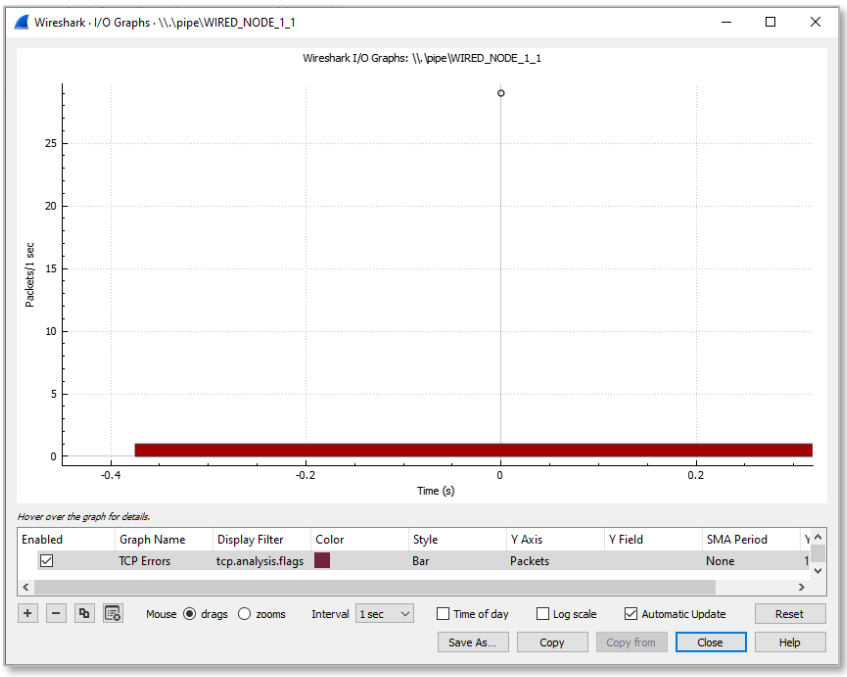


Figure 8-104: Statistics Menu > IO Graph in Wireshark

8.7.5.3 Creating Flow graphs

The flow graph feature provides a quick and easy to use way of checking connections between a client and a server. It can show where there might be issues with a TCP connection, such as timeouts, re-transmitted frames, or dropped connections. To access flow graph, go to ***Statistics Menu → Flow Graph*** and select the flow type. By default, you can see the flow graph of all the packets. To get the TCP flow, select TCP flow in "Flow Type" dropdown box and you will obtain the flow as shown Figure 8-105.

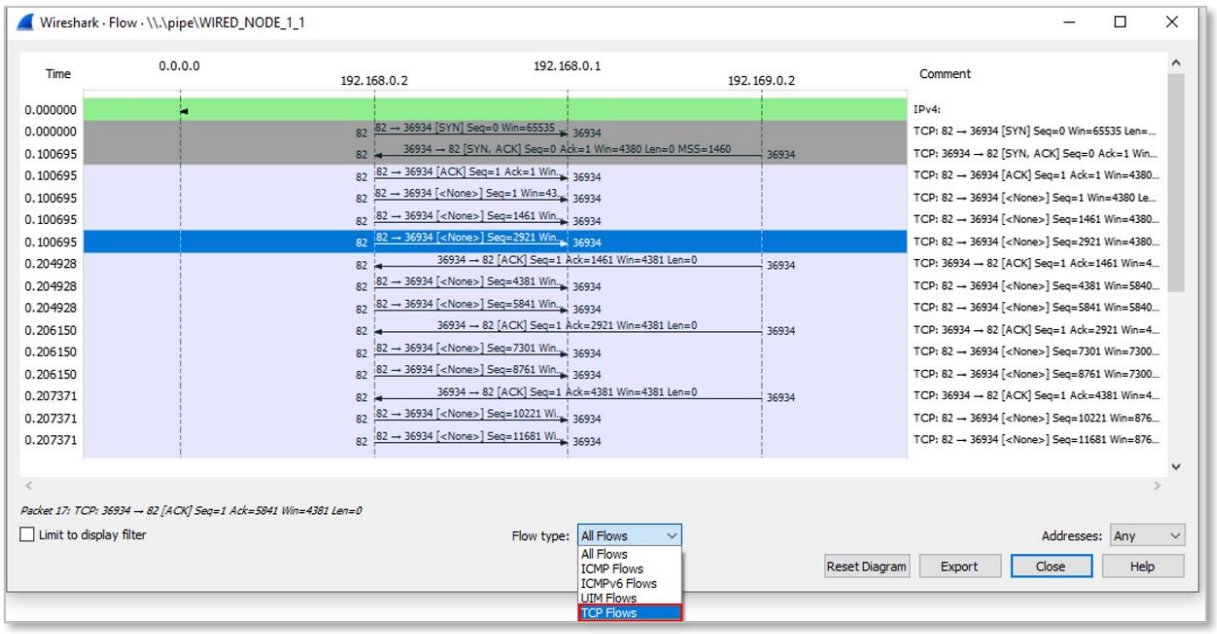


Figure 8-105: Statistics Menu > Flow Graph in Wireshark

8.7.6 Protocols supported in Wireshark – NetSim interfacing

When interacting with NetSim, Wireshark by default logs the following protocols.

Routing protocols
RIP (Routing Information Protocol)
OSPF (Open Shortest Path First)
WSMP (Wireless Mesh Standard Protocol)
Ad hoc routing protocols
RPL (Routing Protocol for Low-Power and Lossy Networks)
DSR (Dynamic Source Routing)
AODV (Ad hoc On-Demand Distance Vector)
OLSR (Optimized Link State Routing)
Internet Control Message Protocol (ICMP)
ICMP Echo Request/Reply
ICMP Destination Unreachable
ICMP Time Exceeded
ICMP Parameter Problem
Transport protocols
TCP (Transmission Control Protocol)

UDP (User Datagram Protocol)
Data link layer
802.11 (Wi-Fi)

Table 8-6: Protocols logged by Wireshark when interfacing with NetSim

Note: Protocols such as LTE_NR, 802.22, Zigbee, OpenFlow are not logged currently in Wireshark.

9 Writing Custom Code in NetSim

9.1 Writing your own code

NetSim allows the user to write the custom code for all the protocols by creating a DLL (Dynamic Link Library) for their custom code.

There are various important steps in this process, and each of these steps has various options as explained in the subsequent pages.

9.1.1 Microsoft Visual Studio 2022 Installation Settings

NetSim requires only a few components of Visual Studio Community 2022 edition to be installed. Upon starting the installer:

1. Under the **Workloads** tab users can select **Desktop Development with C++** as shown below Figure 9-1.

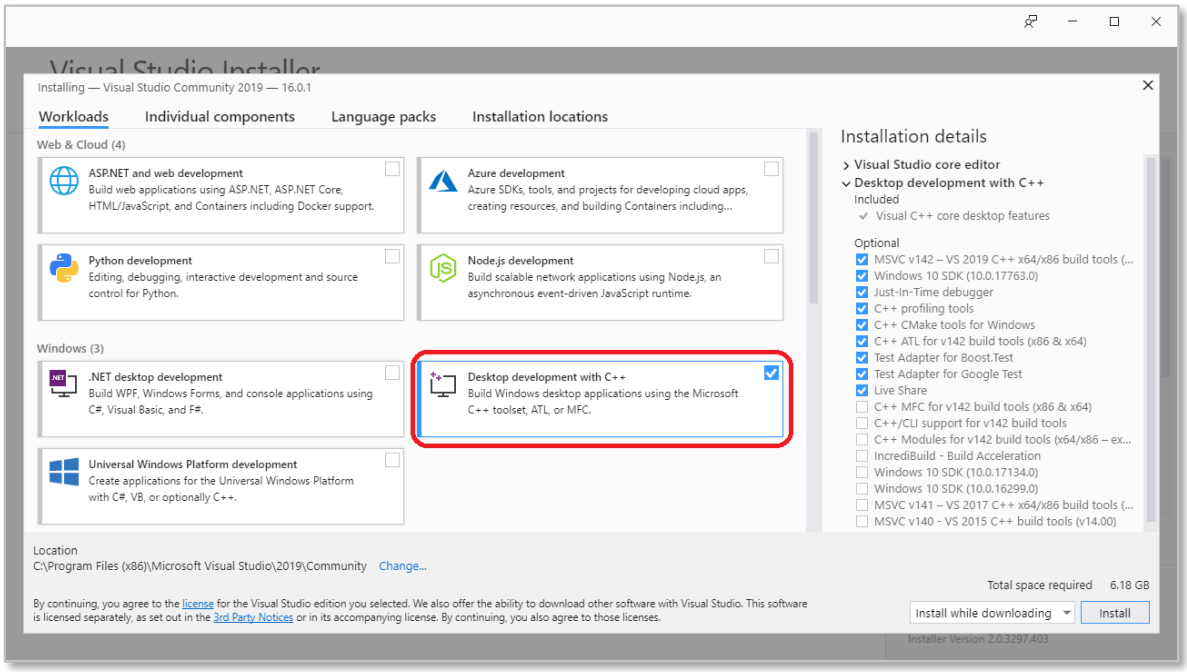


Figure 9-1: In Workloads tab select Desktop Development with C++

2. Under the Individual components tab select MSVC v140-VS 2015 C++ toolset for desktop (x86, x64).

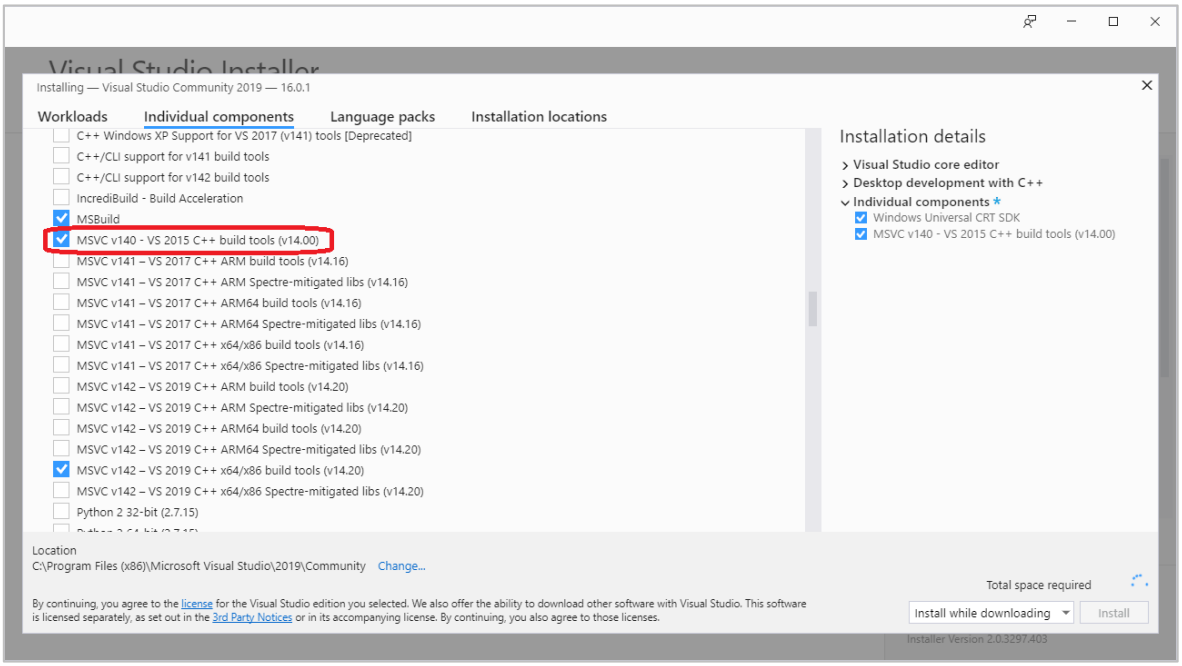


Figure 9-2: In Individual components tab select MSVC v140-VS 2015 C++ toolset for desktop (x86, x64).

9.1.2 Modifying code

DLL is the shared library concept, implemented by Microsoft. All DLL files have a .dll file extension. DLLs provide a mechanism for sharing code and data to upgrade functionality without requiring applications to be re-linked or re-compiled. NetSim requires Visual Studio Compiler for building DLL's.

Refer **section 4.13** "How does a user open and modify source codes" to open NetSim Source Codes

1. After this you may modify the source codes of any project. You can also add new files to the project if required. As an example, let us make a simple source code modification to TCP. Inside Solution Explorer pane in Visual Studio, double click on TCP project. Then open TCP.c file by double clicking on it. Using the drop down list of functions that are part of the current file, choose **fn_Netsim_TCP_Init()**.

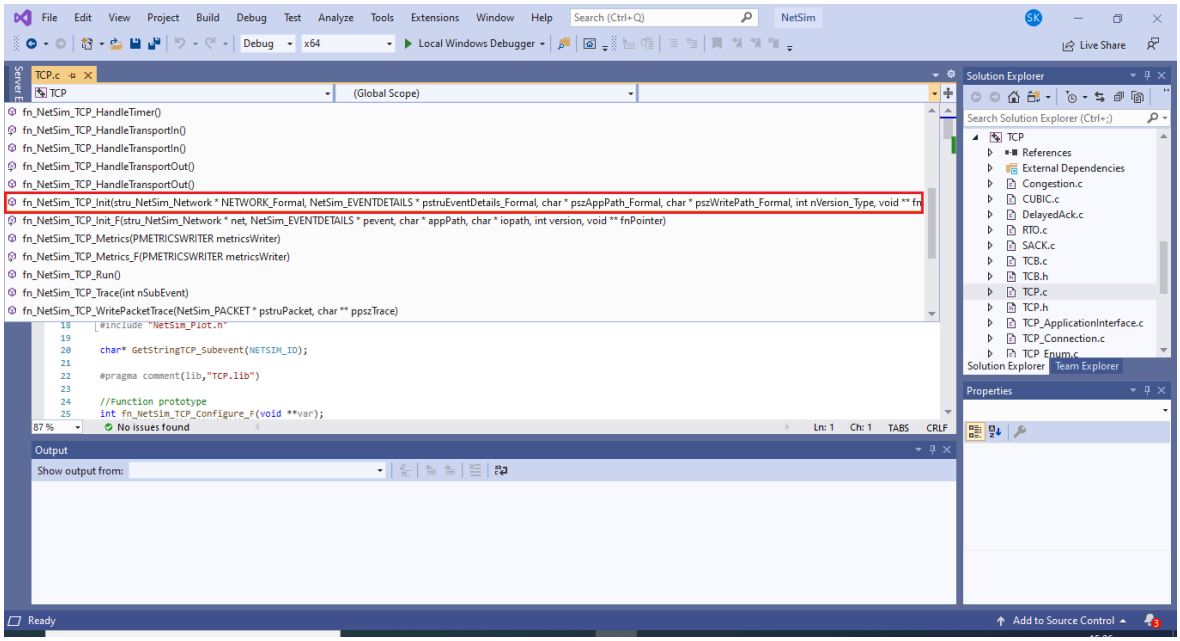


Figure 9-3: Select fn_Netsim_TCP_Init() in TCP.C in Visual Studio

2. Add the line **fprintf(stderr, "\nSource is Modified\n");** statement inside the **fn_Netsim_TCP_Init()** function as shown below to print “Source is modified”. **_getch();** is added in the next line for the simulation to wait until it gets a user input.

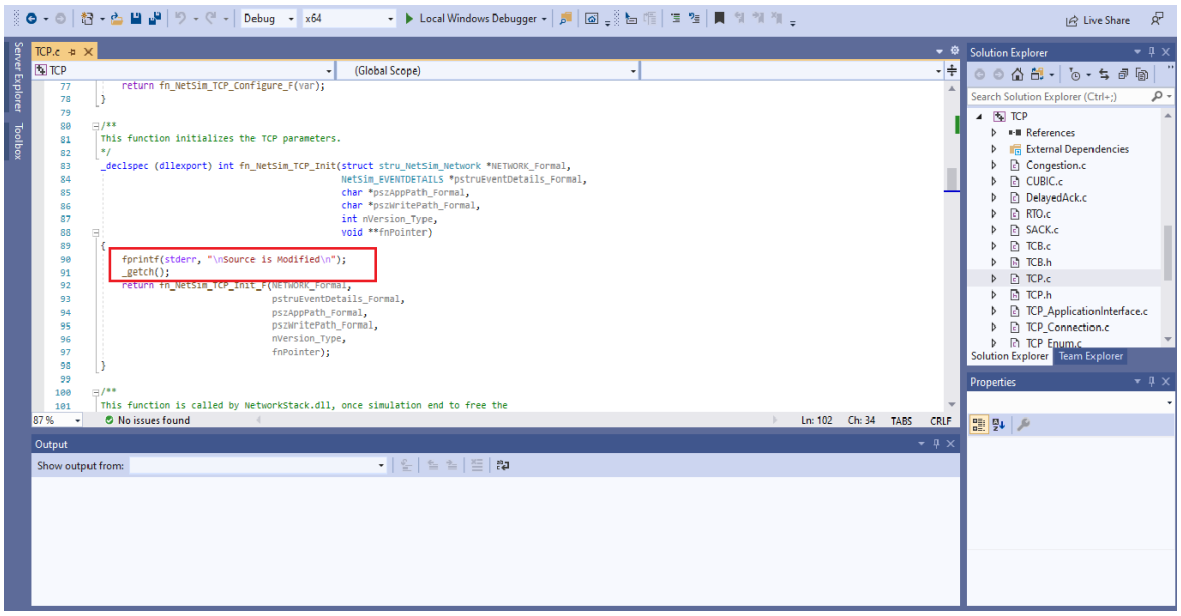


Figure 9-4: Source Code modified in fn_Netsim_TCP_Init() function in TCP project

9.1.3 Building DLLs

1. Identify the build of NetSim that is installed in your system from NetSim Home Screen as shown below.

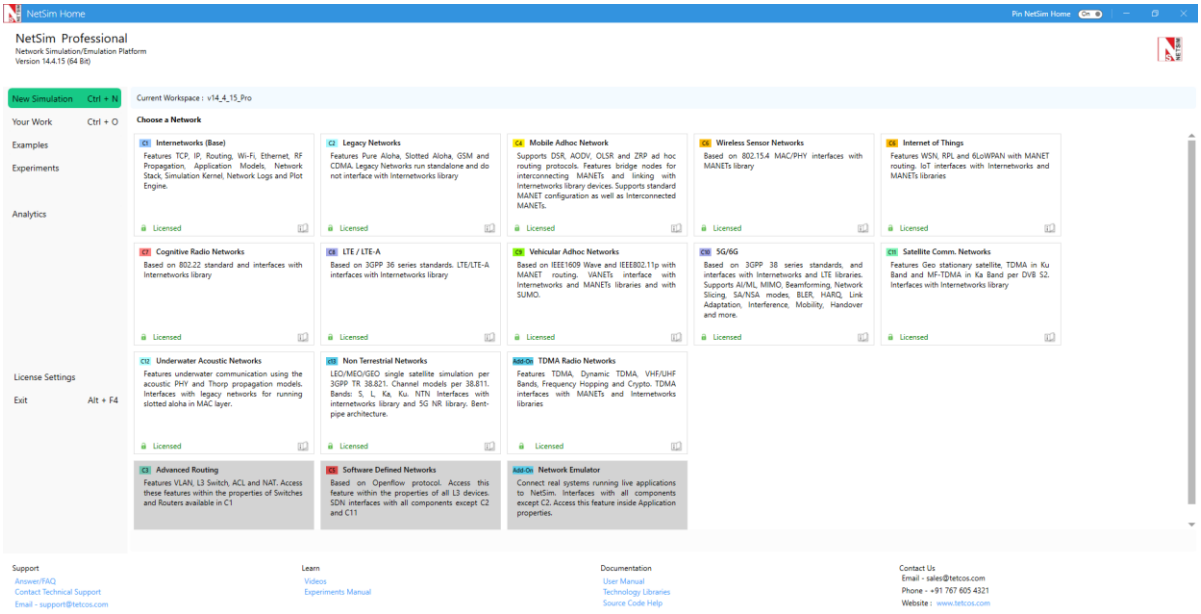


Figure 9-5: In NetSim Home Screen Identify the build of NetSim

2. By default, x64 is chosen for 64-bit version of NetSim. These changes will be automatically applied to all projects that are displayed in the Solution Figure 9-6.

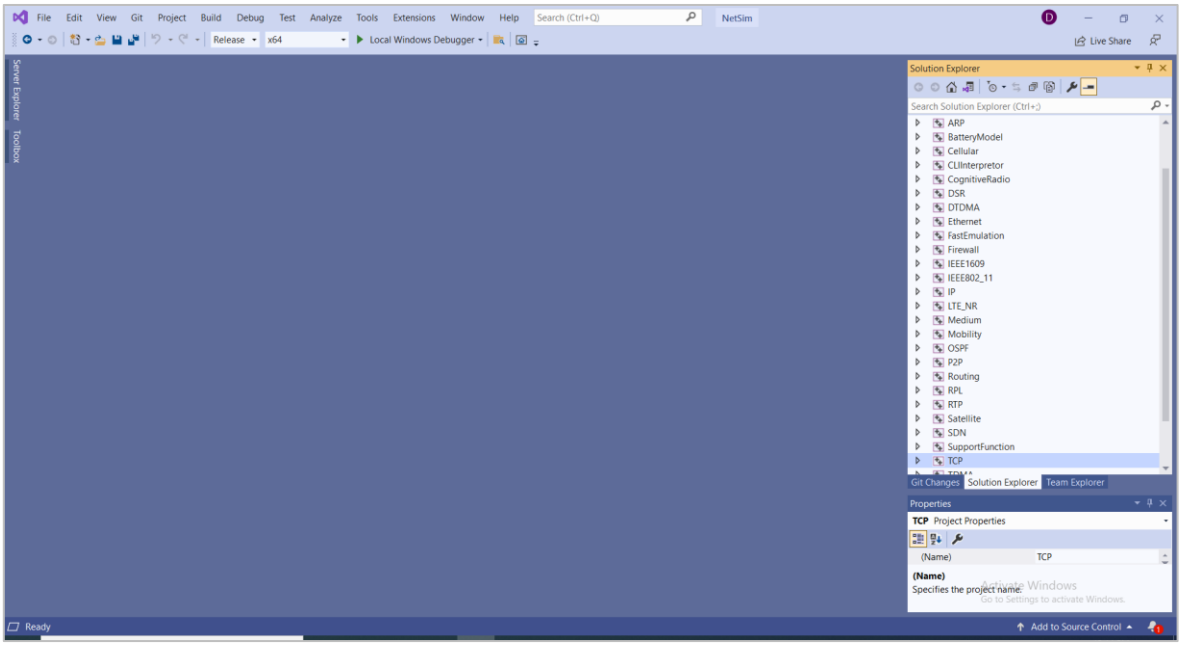


Figure 9-6: Based on the build of NetSim select x64 in Visual studio

3. Now rebuild the network by right clicking on the project header and selecting Rebuild creates a DLL file in the bin folder of NetSim's current workspace path.

< C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64> for 64-bit which contains your modifications. If build is successful a message similar to the following will be displayed in the output window as shown below.

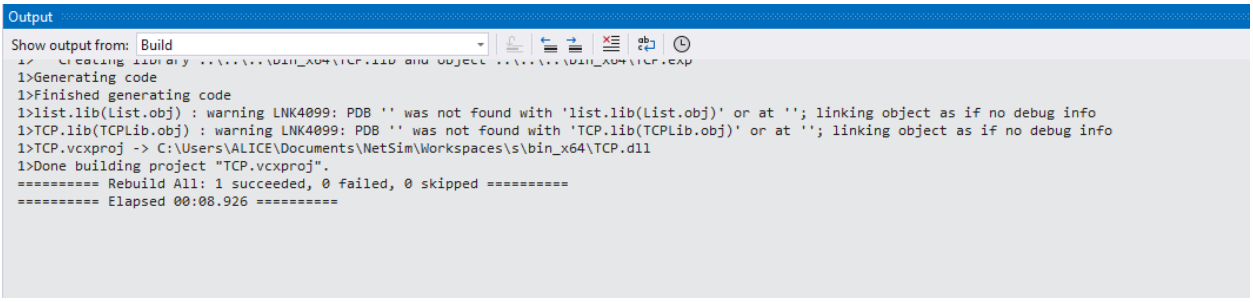


Figure 9-7: Build is successful a message like the following will be displayed in the output window

9.1.3.1 Error:

The solution file settings in Visual Studio (for all protocol source codes) is of latest version, i.e., Visual Studio 2022. Build will be successful when users run this version of Visual Studio. Users running older version of Visual studio will face errors related to Platform Set during the compilation of code. The Platform Toolset properties for different versions of VS are given below.

Version	Platform Set
Visual Studio 2015	v140
Visual Studio 2017	v141
Visual Studio 2019	v142
Visual Studio 2022	v143

Table 9-1: Platform Toolset Properties of different visual studio versions.

If a user has an older version of VS installed in their system they can migrate to the latest version of VS (and uninstall the older version of VS), or perform the following changes

- Go to Project 'Properties' > Under 'General' > Select Platform toolset as it is mentioned in the table above. (i.e.) for VS2022 - select v143, then save (Click OK), then rebuild the code.

9.1.4 Running Simulation

1. After rebuilding the code, user can run the simulation via GUI (Please refer **section 3**). In this case, user can create a scenario in any network which involves TCP protocol. Running the simulation with the custom DLL will initially display a warning message as shown below.

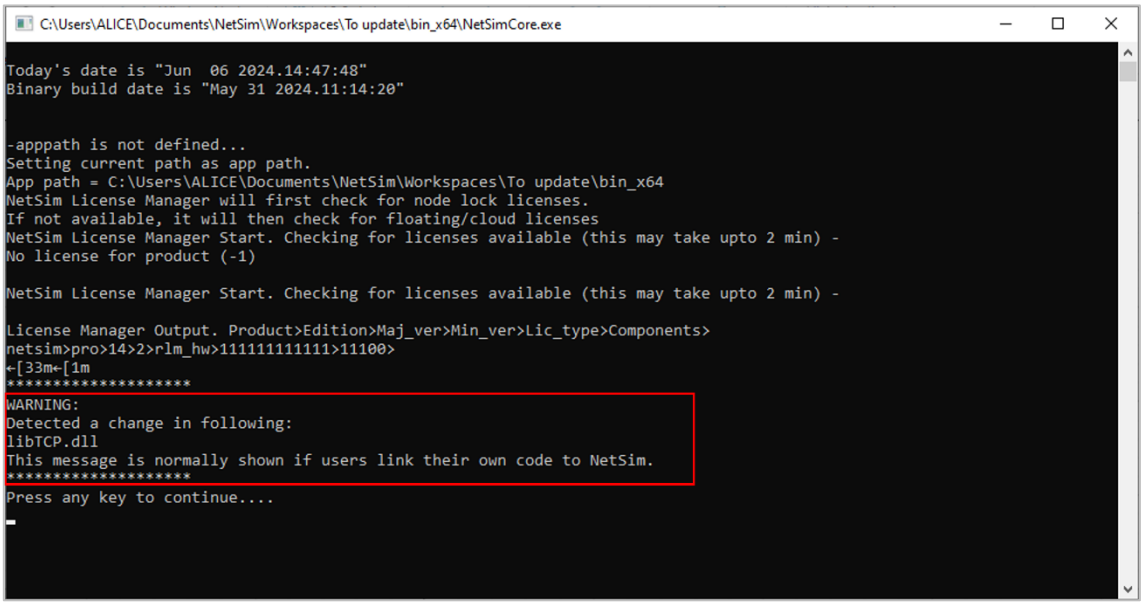


Figure 9-8: Modified Project display a DLL warning message to NetSim Console

2. The warning message lists the DLL files which have been modified in the bin folder (bin\bin_x64 for 64-bit) of NetSim's current workspace path. After pressing any key, the statement "Source is modified" will be printed to console as shown below.

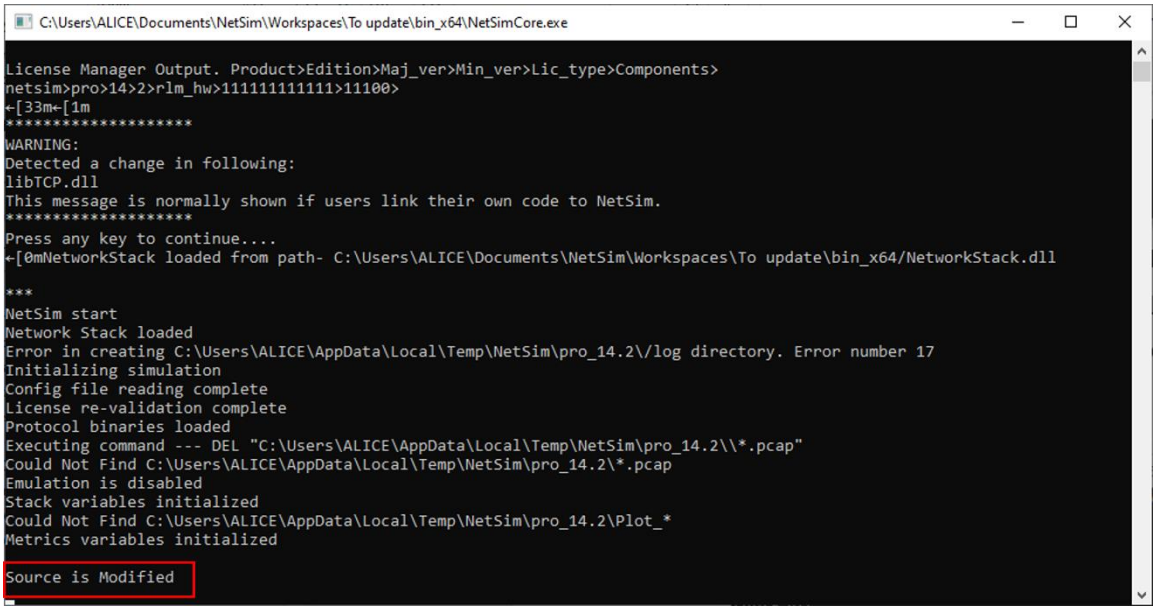


Figure 9-9: Printf Statement written to console

3. Press any key to proceed with the simulation.
4. The warning message will not be displayed if no DLL's are modified in the bin folder of current workspace path (bin\bin_x64 for 64-bit).

9.1.5 Source Code Dependencies

The following are the list of projects that are part of NetSim source codes present in **<NetSim Install Directory>/src/Simulation** directory and their dependencies:

PROJECT	DEPENDENCY
Application	IP
Cellular	Application
CLIInterpertor	Firewall, IP
Cognitive Radio	Application
Ethernet	Firewall
IEEE802_11	Battery Model
OSPF	IP
Routing	IP
RPL	IP
ZigBee	Battery Model
ZRP	IP
Aloha	-
AODV	-
ARP	-
Battery Model	-
Medium	-
DSR	-
Firewall	-
IEEE1609	-
IP	-
LTE NR	
Mobility	-
P2P	-
SDN	-
Support Function	-
TCP	-
FastEmulation	-
UDP	-
UWAN	Battery Model
DTDMA	-
TDMA	-
Satellite Comm. Networks	-

Table 9-2: Source Code Dependencies

For example, to perform modifications to Application Project, IP folder will also be required in addition to lib folder, Include folder and NetSim.sln file.

9.1.6 Enabling Additional Security Checks

SDL – Security Development Lifecycle checks adds recommended Security Development Lifecycle. These checks include extra security-relevant warnings as errors, and additional secure code-generation features.

/sdl enables a superset of the baseline security checks provided by /GS and overrides /GS-. By default, **/sdl** is off. **/sdl-** disables the additional security checks.

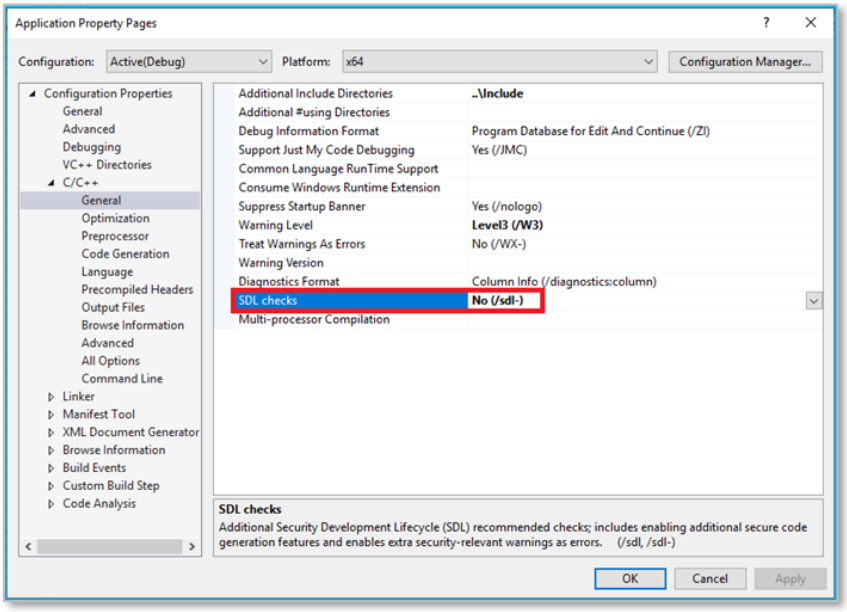


Figure 9-10: Enabling Additional Security Checks application properties window

If SDL checks is enabled (/sdl), following warnings will be treated as errors:

Warning enabled by /sdl	Equivalent command-line switch	Description
C4146	/we4146	A unary minus operator was applied to an unsigned type, resulting in an unsigned result.
C4308	/we4308	A negative integral constant converted to unsigned type, resulting in a possibly meaningless result.
C4532	/we4532	Use of continue, break or go to keywords in a __finally/finally block has undefined behavior during abnormal termination.
C4533	/we4533	Code initializing a variable will not be executed.
C4700	/we4700	Use of an uninitialized local variable.
C4703	/we4703	Use of a potentially uninitialized local pointer variable.
C4789	/we4789	Buffer overrun when specific C run-time (CRT) functions are used.
C4995	/we4995	Use of a function marked with pragma deprecated.
C4996	/we4996	Use of a function marked as deprecated.

Table 9-3: SDL Warnings Messages

Reference: https://docs.microsoft.com/en-us/cpp/build/reference/sdl-enable-additional-security-checks?view=vs-2019

9.2 Implementing your code - Examples

9.2.1 Hello World Program

Objective: Print Hello World from TCP protocol.

Implementation: Add fprintf (stderr, “<MESSAGE>”) statement inside the source code of TCP as shown below to print “Hello World” when custom built dll is executing.

```
fprintf(stderr, "\n Hello World\n");
_getch();
```

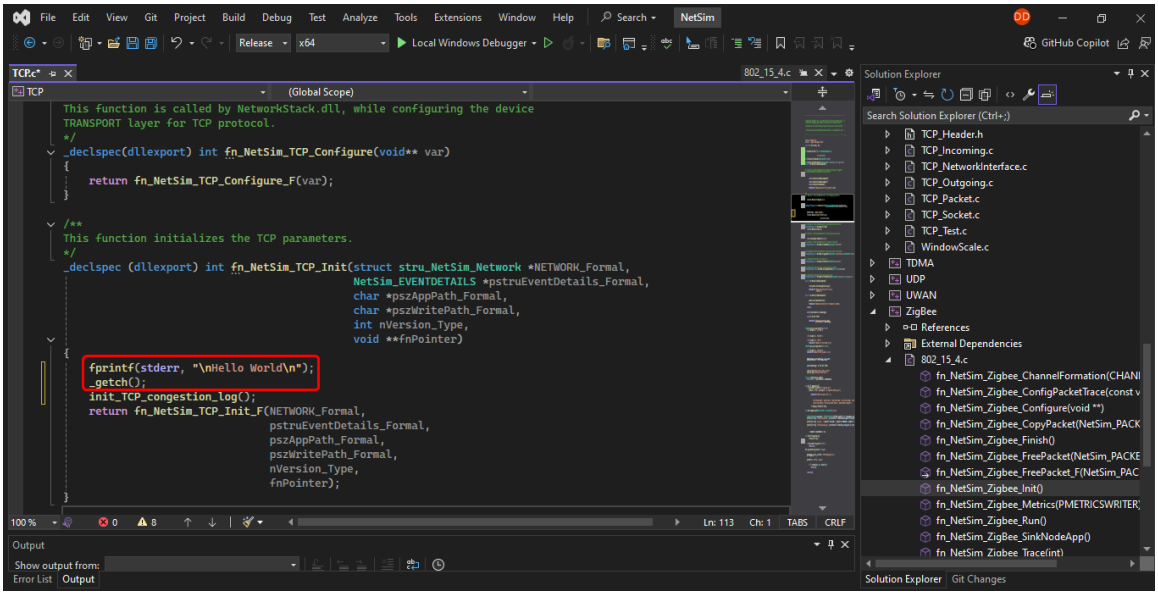


Figure 9-11: Hello World Printf Statement added in TCP Project

Build DLL as explained in 9.1.3 and run the simulation, you can see the following output on the console.

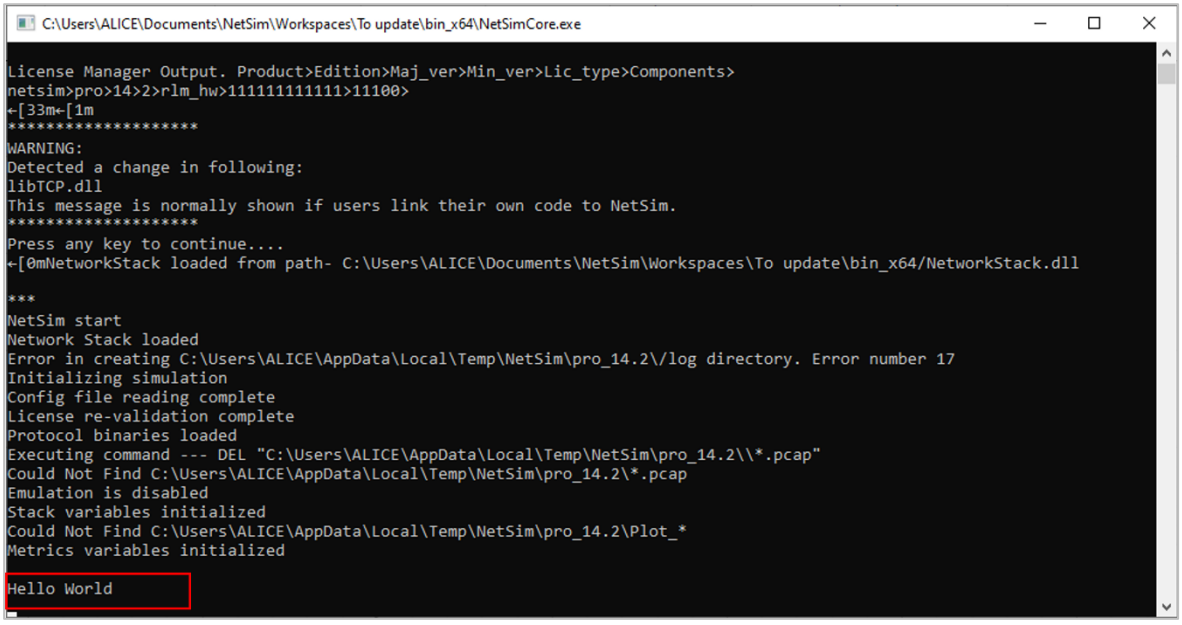


Figure 9-12: Hello World Statement written to console

Press enter then simulation will continue.

9.2.2 Introducing Node Failure in MANET

Objective: Node failure using MANET-DSR using Device Id.

Implementation: Identify the Device ID of the particular node to be failed.

Step 1: Create a file with the name NodeFailure.txt inside the bin folder of NetSim's current workspace path

<C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64> for 64-bit.

The file will contain two columns: one being the Node ID of the device to be failed and other being the failure time (in microseconds).

For example, to fail Node Id 2 from10th sec onwards and fail Node Id 1 from 90th sec onwards, the NodeFailure.txt file will be as follows Figure 9-14.

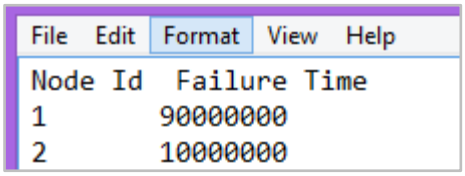


Figure 9-13: NodeFailure.txt file

Step 2: Go to DSR.c in DSR protocol.

Step 3: The function fn_NetSim_DSR_Init() will execute before the protocol execution starts. So, in this function, we will read the NodeFailure.txt and save information regarding which nodes will fail at which time. Add the following code inside the specified function.

```
int i;
FILE* fp1;
char* pszFilepath;
char pszConfigInput[1000];
pszFilepath = fnpAllocateMemory(36, sizeof(char) * 50);
strcpy(pszFilepath, pszAppPath);
strcat(pszFilepath, "/NodeFailure.txt");
fp1 = fopen(pszFilepath, "r");
i = 0;
if (fp1)
{
	while (fgets(pszConfigInput, 500, fp1) != NULL)
	{
		sscanf(pszConfigInput, "%d   %d", &NodeArray[i], &TimeArray[i]);
		i += 1;
	}
	fclose(fp1);
}
```

Step 4: The fn_NetSim_DSR_Run() is the main function to handle all the protocol functionalities. So, add the following code to the function at the start.

```
int i, nFlag = 1;
```

```
if (nFlag)
{
        for (i = 0; i < 100; i++)
                if ((pstruEventDetails->nDeviceId == NodeArray[i]) &&
                        (pstruEventDetails->dEventTime >= TimeArray[i]))
                {
                        pstruEventDetails->nInterfaceId = 0;
                        pstruEventDetails->pPacket = NULL;
                        return 0;
                }
}
```

Step 5: Add the following code inside DSR.h header file.

```
//Node failure model
int NodeArray[200];
int TimeArray[200];
```

Step 6: Build DLL as explained in **Section 9.1.3**.

Step 7: Create a scenario in MANET with DSR Protocol. where data packets should be travelling from source to destination through the mentioned node in NodeFailure.txt file. For that user can increase the pathloss exponent value and the distance among the nodes. User can utilize Packet trace to check the node failure (i.e. no packets are forwarded by failed nodes) after the mentioned time.

9.3 Debugging your code

This section is helpful to debug the code which user has written. To write your own code please refer **Section 9.1.2**.

9.3.1 Via GUI

Debugging your code via GUI there are two methods available.

- **Using _getch()**
- **Using Environment Variables (NETSIM_BREAK)**

9.3.1.1 Using _getch()

Step 1: Perform the required modification of the protocol source code and add _getch() (used to hold the program execution until the user enters a character) statement inside init function of the

modified protocol. For example, take TCP protocol and add the following lines of code in the init function as shown in the below screenshot Figure 9-14.

fprintf(stderr, "\nAttach to Process now\n");
_getch();

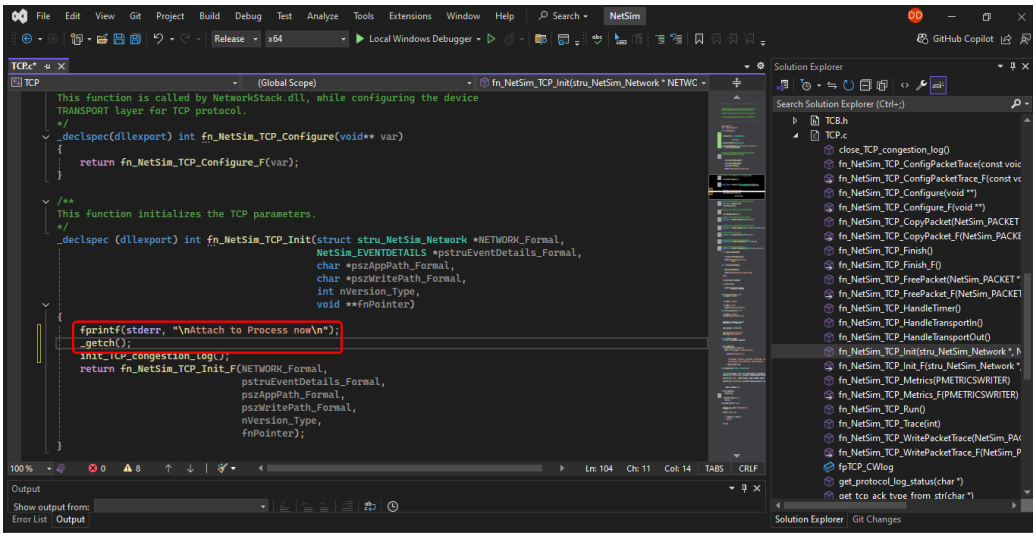


Figure 9-14: Added Two line of Code in TCP protocol

Step 2: Build the TCP protocol as explained in section **9.1.3**. Do not close Visual Studio.

Step 3: In NetSim, create a network scenario where the protocol is being used and start the simulation. In the console window user would get a warning message shown in the below screenshot Figure 9-15 and the simulation will pause for user input (because of _getch() added in the init function)

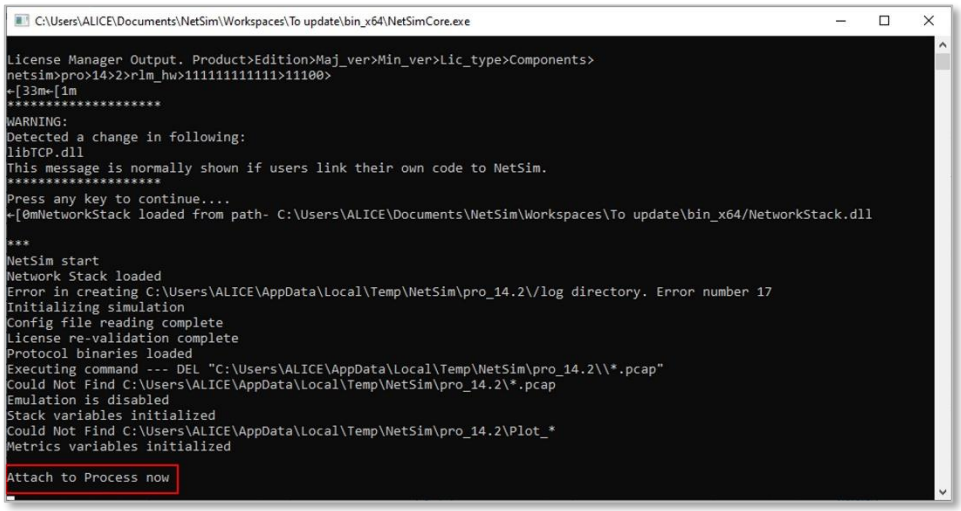


Figure 9-15: Attach to Process Statement printed to console, and waiting for user to attach NetSim process to Visual Studio and an user input since _getch() function is called.

Step 4: In Visual Studio, put break point inside the source code where you want to debug.

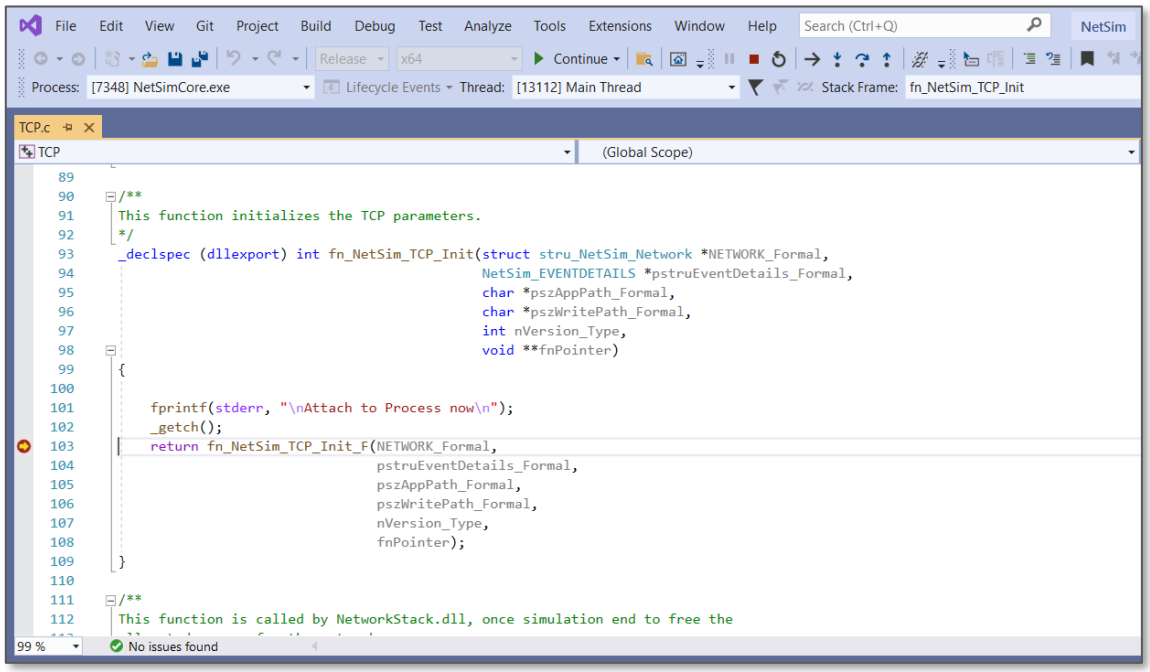


Figure 9-16: Adding breakpoint in Visual Studio

Step 5: Go to "Debug→Attach to Process" in Visual studio as shown and attach to NetSimCore.exe.

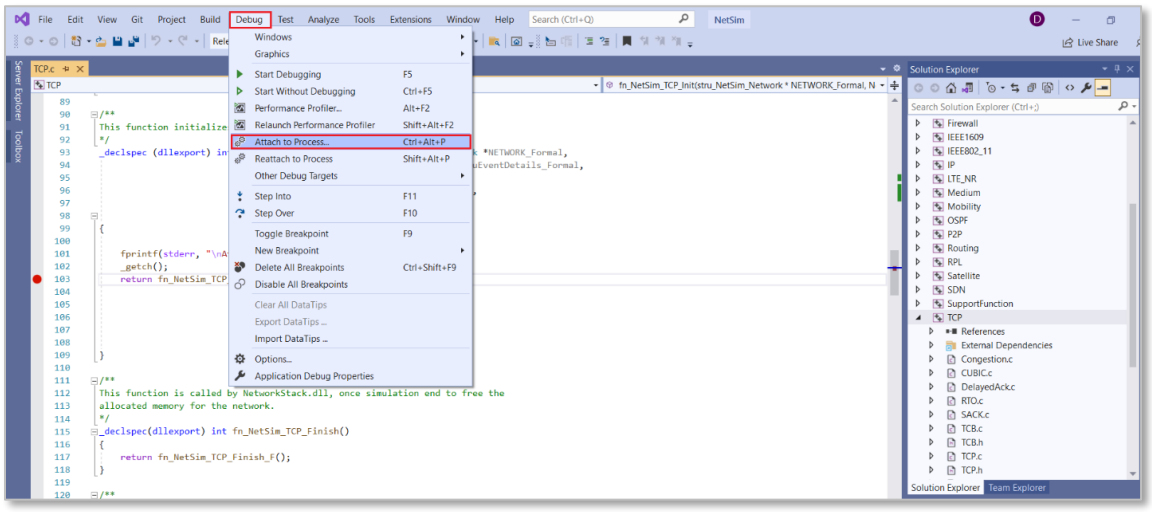


Figure 9-17: Debug > Attach to Process in Visual studio

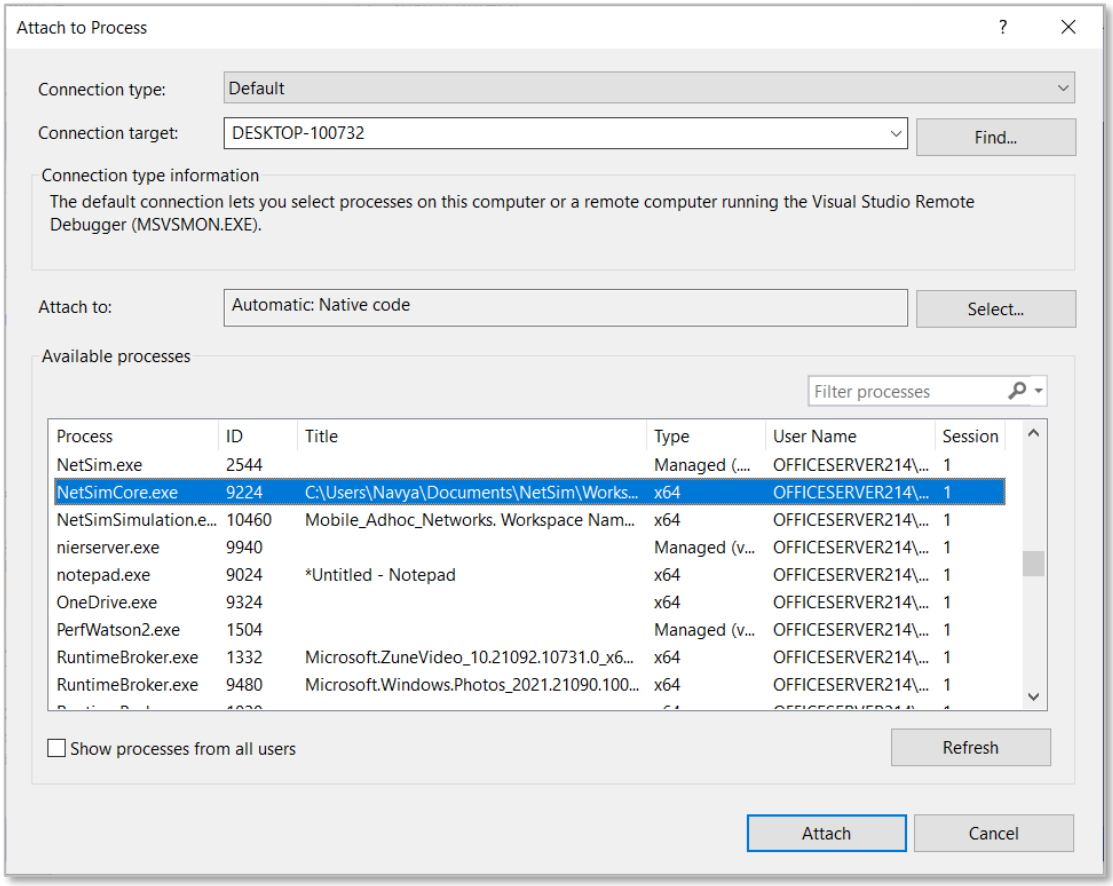


Figure 9-18: Select NetSimCore.exe in Attach to Process Window

Click on Attach. Press enter in the command window. Then control goes to the project and stops at the break point in the source code as shown below Figure 9-19. All debugging options like step over (F10), step into (F11), step out (Shift + F11), continue (F5) are available.

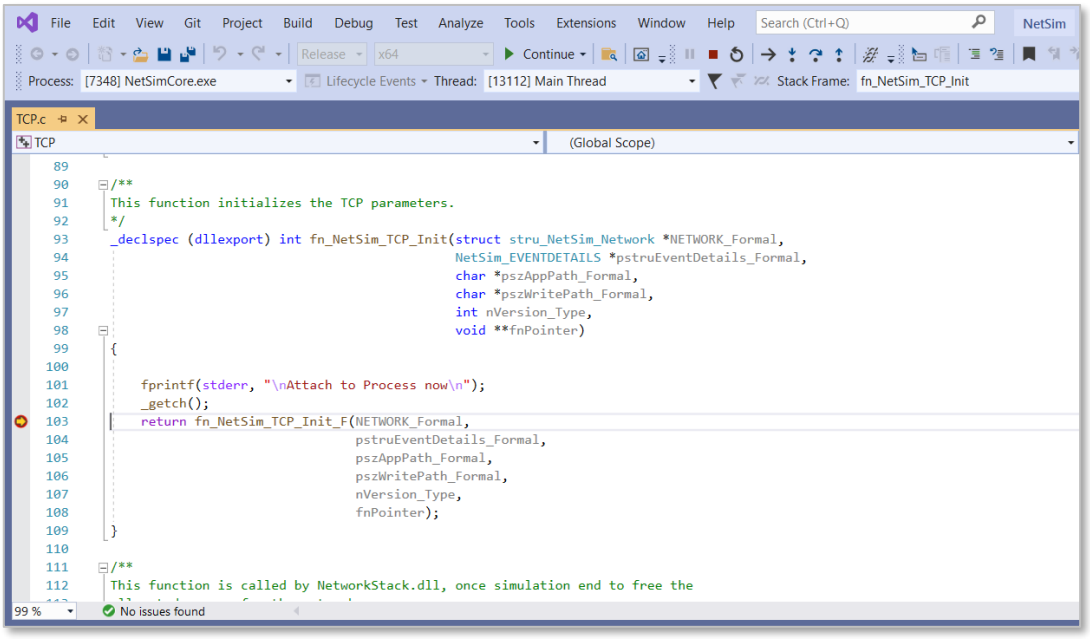


Figure 9-19: Control goes to the project and stops at the break point in the source code

After execution of the function, the control goes back to NetSim and then comes back to the custom code the next time the function is called in the simulation.

To stop debugging and continue execution, press Shift+F5 (key). This then gives the control back to NetSim, for normal execution to continue.

9.3.1.2 Using Environment Variable

This section is helpful to Debug Using Environment Variable (NETSIM_BREAK). To set Environment variable follow the steps as shown.

Note: Setting NETSIM_BREAK Environment Variable will cause the simulation to slow down and it is recommended to remove this Environment Variables after debugging the simulation

Step 1: Right click on My Computer\ This PC and select Properties.

Step 2: Go to Advanced System setting → Advanced Tab → Environment Variables option

Step 3: Click New in System variables. Type “NETSIM_BREAK” as Variable name and any positive integer as variable value (e.g., 2). Click OK. The value of the variable is the event ID at which you want NetSim Simulation to break. In this example we have set the value to 2, which means that the simulation will break at the previous event.

Note: After adding NETSIM_BREAK in environment variable, users should restart the computer

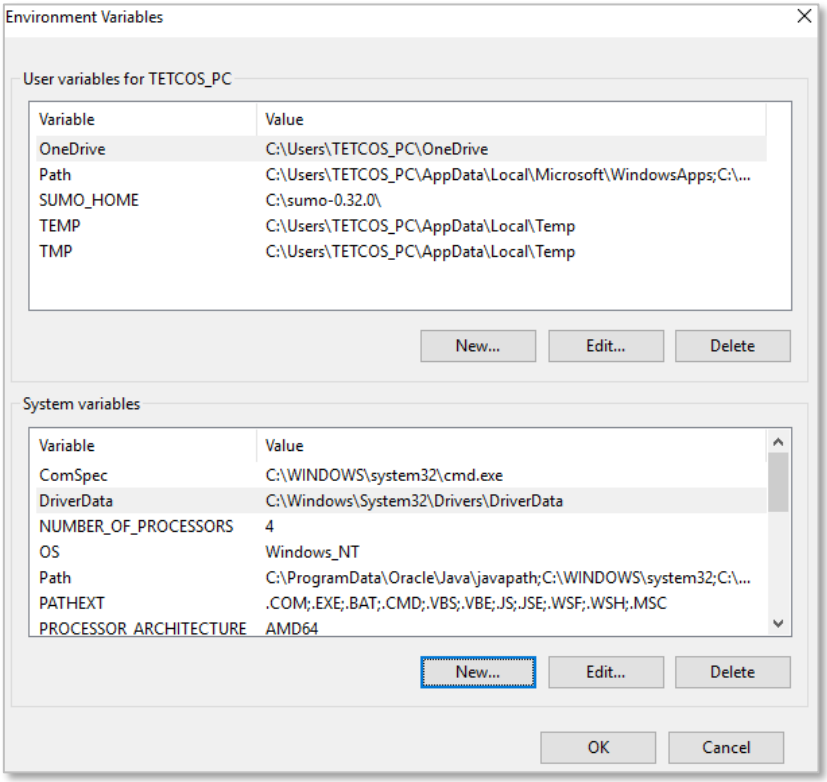


Figure 9-20: Environment Variable Window

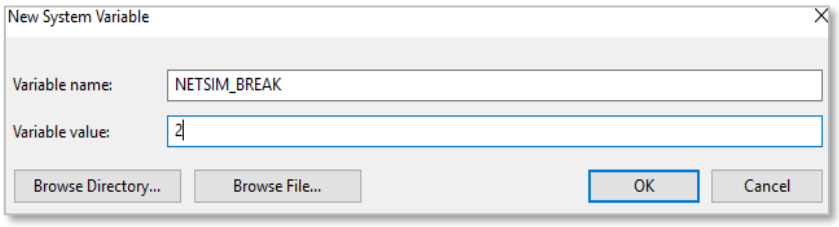


Figure 9-21: Add Variable name and Variable Value in New in System Variable

Step 4: Open NetSim and then open the source codes. Please refer **Section 4.13** "How does a user open and modify source codes" for more information.

Step 5: Create a network scenario in NetSim (Internetworks or any other networks)

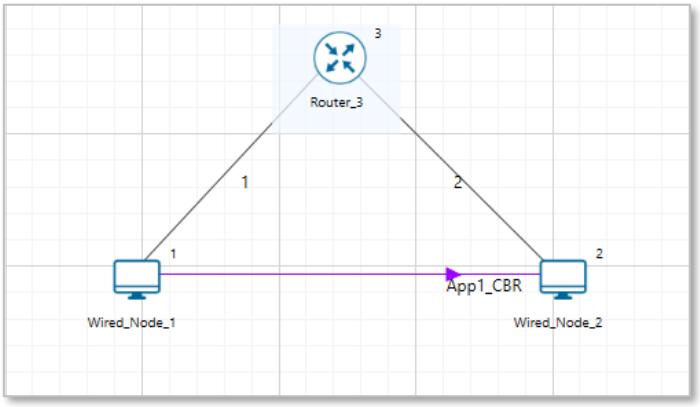


Figure 9-22: Network Topology

Step 6: In this example we are placing a break point in TCP source code and thus TCP should be select in Application properties window.

Step 7: Enable Event trace option and run the simulation.

Simulation will break at event ID 1 as we have set the environment variable to 2 as shown below.

Figure 9-23: Simulation will break at event ID 1

Here NetSim breakpoint has been triggered.

Step 8: Inside Solution Explorer pane in Visual Studio, double click on TCP project. Then open TCP.c file by double clicking on it. Using the drop down list of functions that are part of the current file, choose **fn_NetSim_TCP_Run().**

Step 9: In Visual Studio, Set the breakpoint in the code by clicking on the grey area on the left of the line or by right clicking on the line and selecting Breakpoint->Insert Breakpoint.

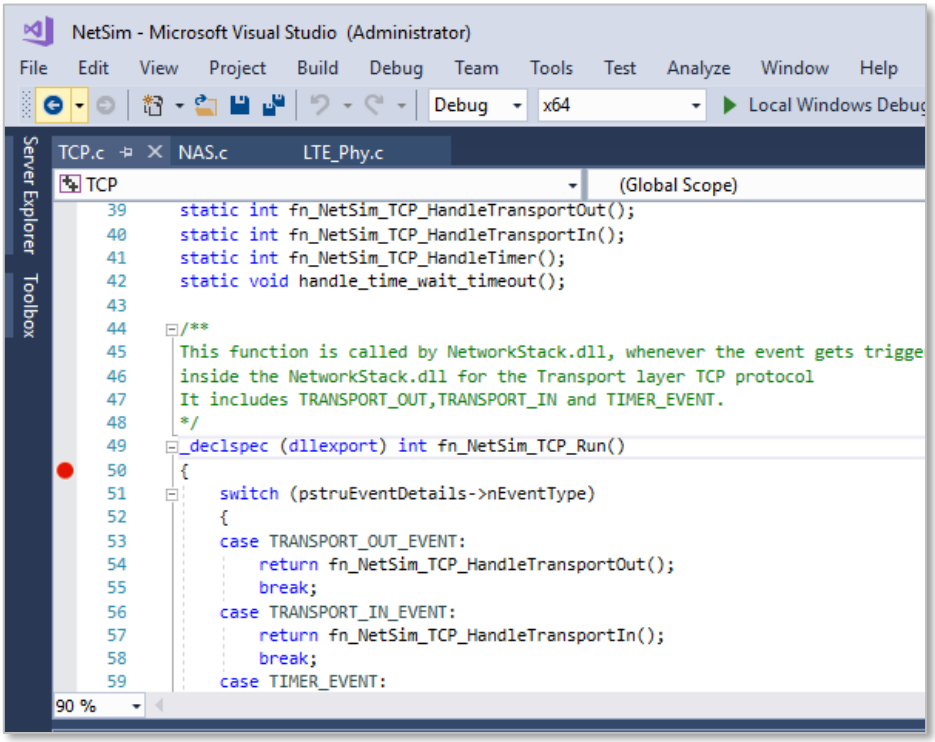


Figure 9-24: Added Break point in line number 50

Step 10: Go to “Debug→Attach to Process” in Visual studio as shown Figure 9-25 and select NetSimCore.exe from the list of processes displayed.

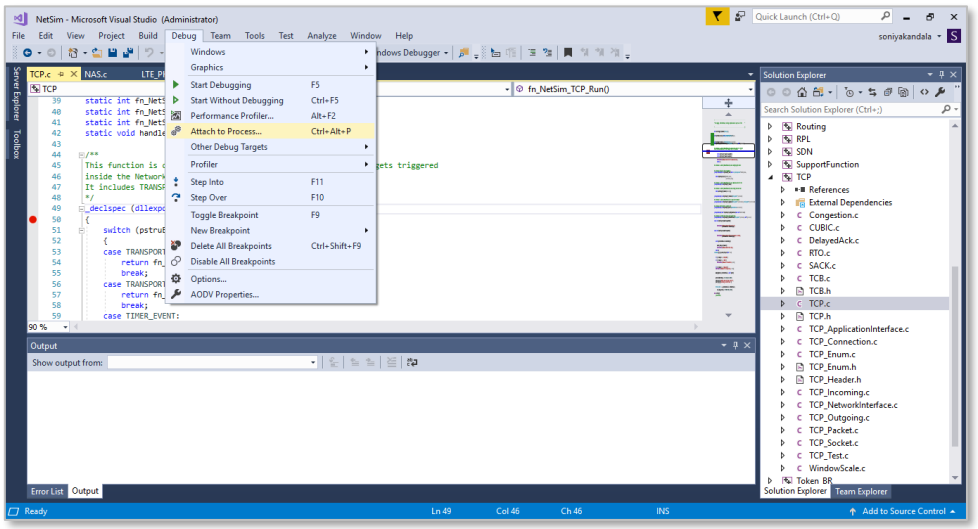


Figure 9-25: Debug > Attach to Process in Visual studio

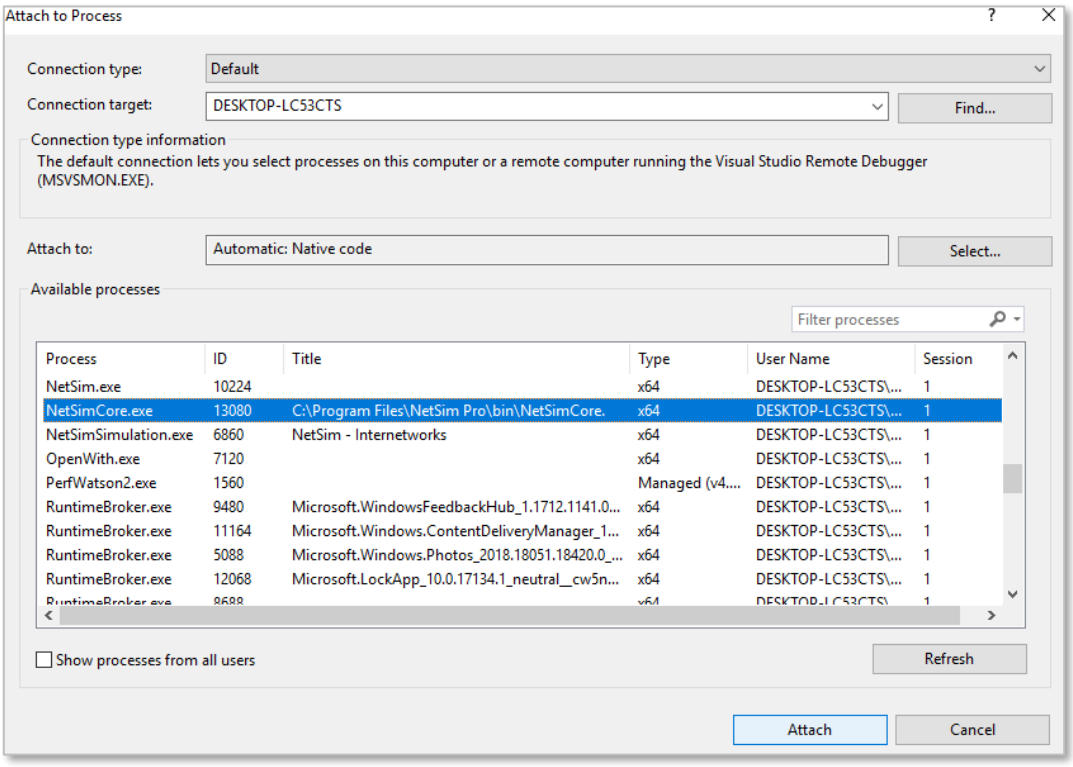


Figure 9-26: Select NetSimCore.exe in Attach to Process Window

Click on Attach. Press any key in the command window to continue the process.

Step 11: Now we need to enter next event ID to break.

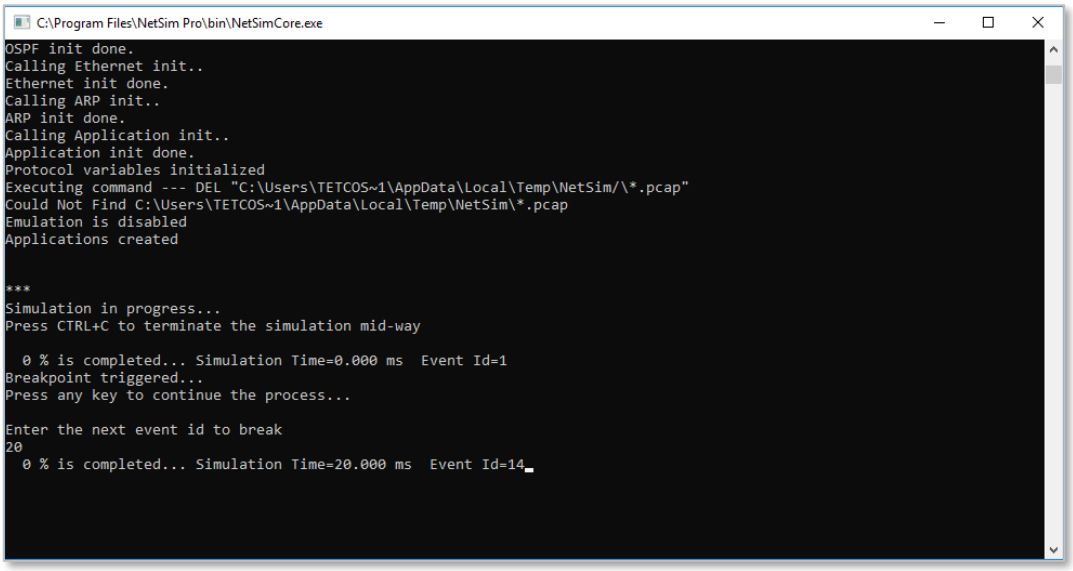


Figure 9-27: Enter next event ID to break

Then control goes to the project and stops at the break point in the source code (NetSim will break wherever user has set the breakpoint) as shown below Figure 9-28. All debugging options like **step over (F10), step into (F11), step out (Shift + F11), continue (F5)** are available.

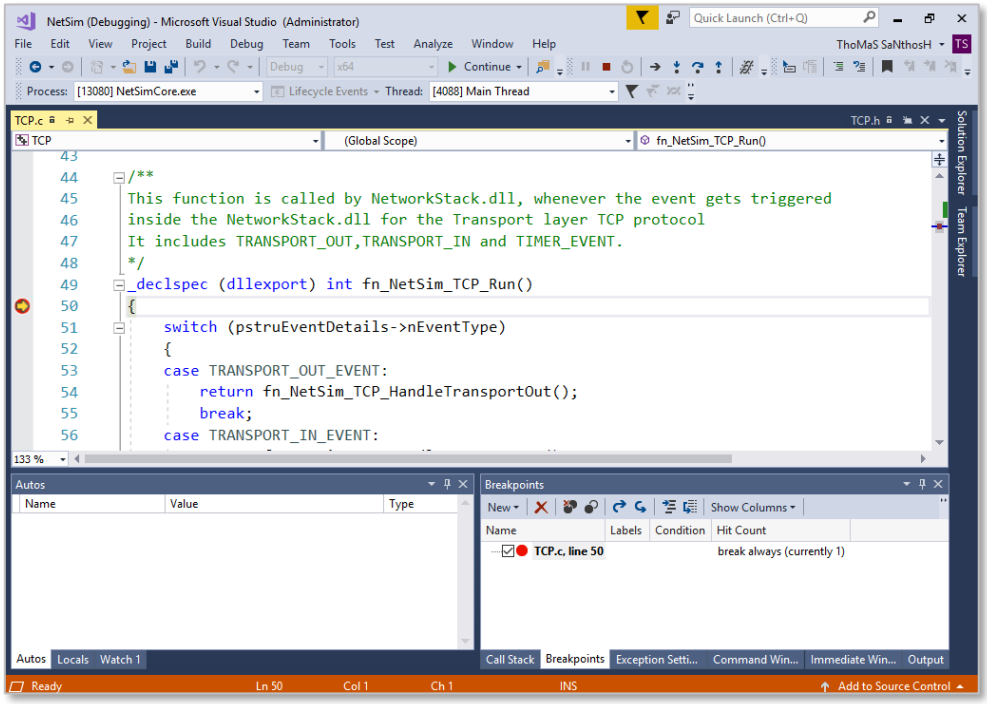


Figure 9-28: Control goes to the project and stops at the break point in the source code.

After execution of the function, the control goes back to NetSim and then comes back to the custom code the next time the function is called in the simulation. To stop debugging and continue execution, press **Shift+F5** (key). This then gives the control back to NetSim, for normal execution to continue.

If NETSIM_BREAK environment variable is set, NetSim **event trace file** additionally logs the file name and line number of the source code where the event was added as shown below:

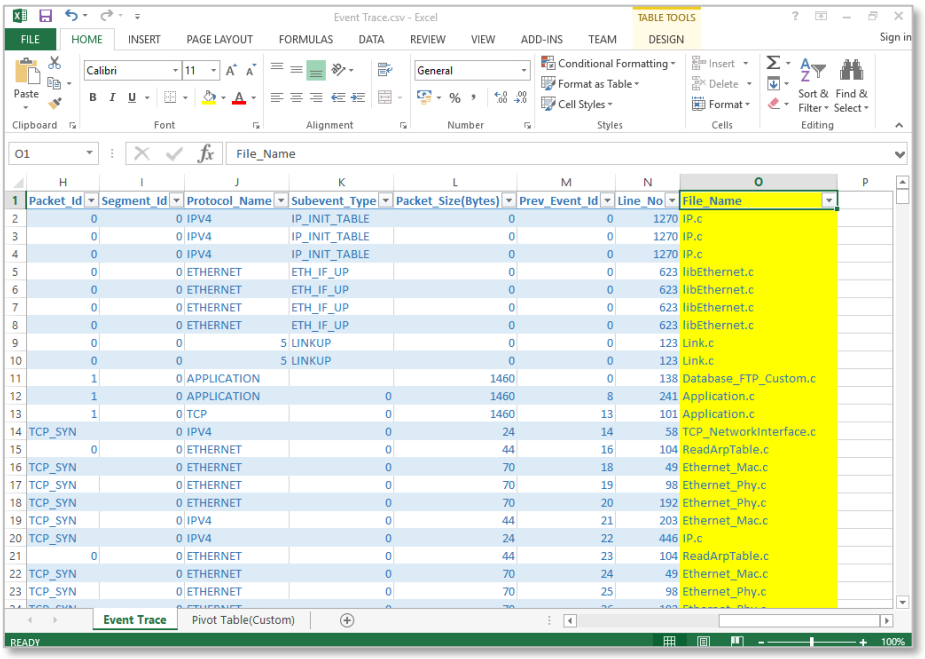


Figure 9-29: NetSim event trace file additionally added two columns the file name and line number of the source code.

9.3.2 Via CLI

Modify the TCP protocol and build the code. Create a scenario in Internetworks then follow the below steps.

Step 1: Open the Command prompt. Press “windows+R” and type “cmd”.

Step 2: To run the NetSim via CLI copy the path where “NetSimCore.exe” is present.

>cd <apppath>

>NetSimCore.exe<space>-apppath<space><apppath><space>-iopath<space><iopath><space>-license<space>5053@<ServerIP Address><space> -d

Step 3: Type the following command.

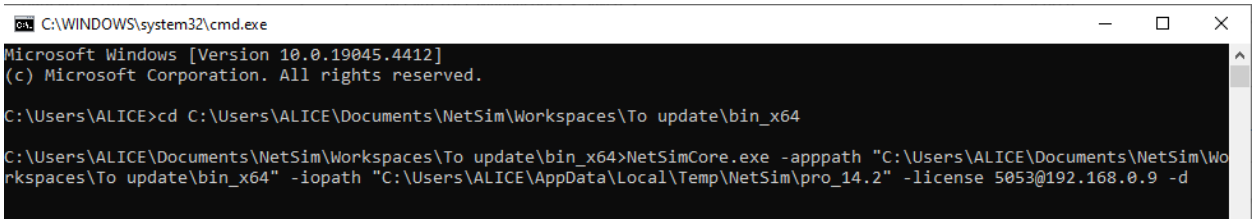


Figure 9-30: Run the NetSim via CLI Mode use the following Command.

Press enter, now you can see the following screen.

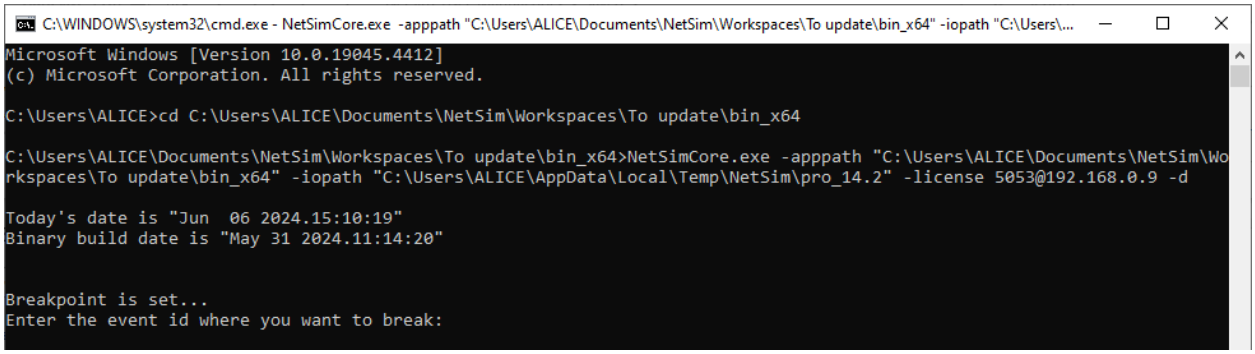


Figure 9-31: Enter the Event ID

Step 4: Open the Project in Visual Studio and put break point inside the source code.

Step 5: Go to “Debug→Attach to Process”.

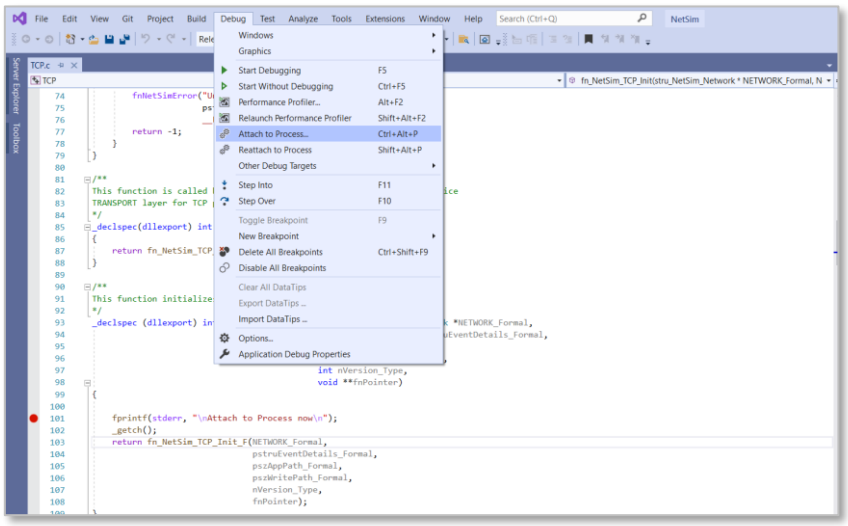


Figure 9-32: Debug > Attach to Process

Attach to NetSimCore.exe.

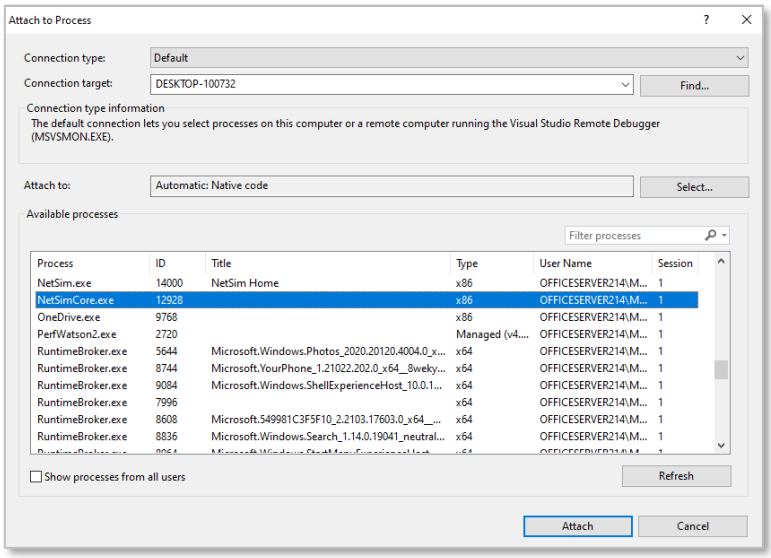


Figure 9-33: Select NetSimCore.exe in Attach to Process Window

Click on Attach.

Step 6: Go to command prompt which is already opened in Step 3. Enter the Event Id.

Note: If you do not want to stop at any event you can specify 0 as event id.

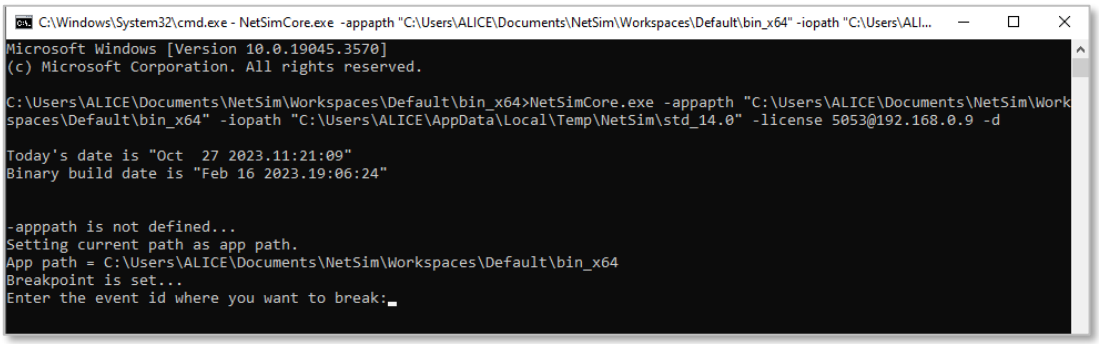


Figure 9-34: Enter the Event Id

Execution will stop at the specified event.

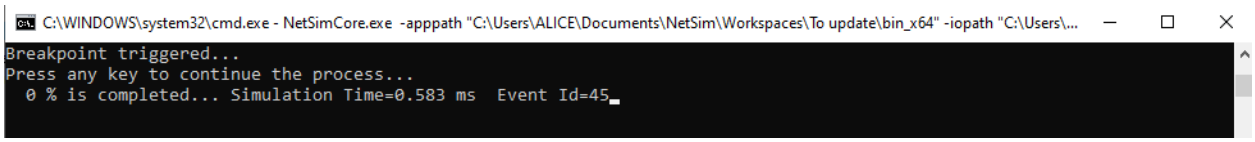


Figure 9-35: Execution stops at the specified event

Press enter then control goes to the project and stops at the break point in the source code as shown below.

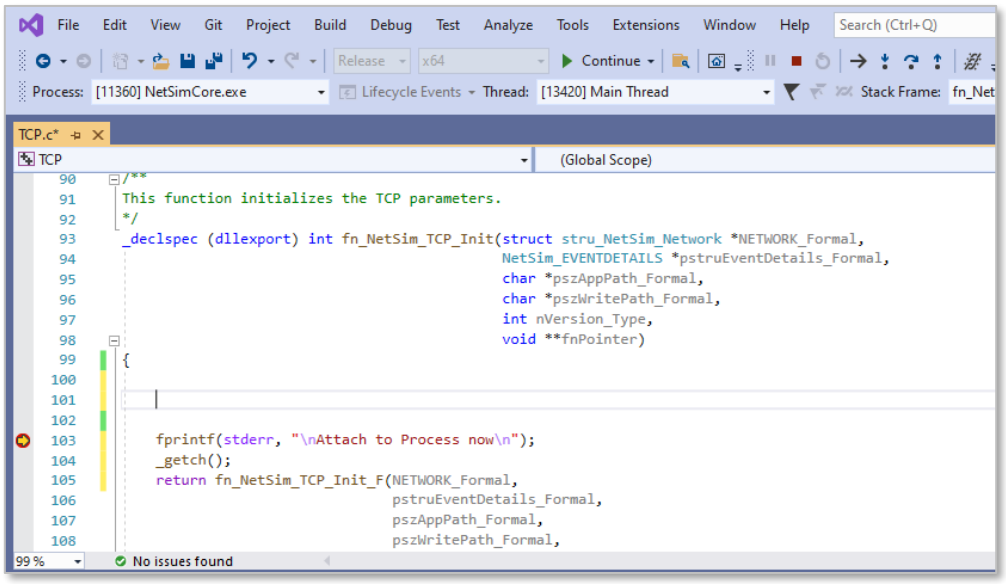


Figure 9-36: Control goes to the project and stops at the break point in the source code

All debugging options like step over (F10), step into (F11), step out (Shift + F11), continue (F5) are available.

After execution of the function, the control goes back to NetSim and then comes back to the custom code the next time the function is called in the simulation.

To stop debugging press Shift+F5. This then gives the control back to NetSim, for normal execution to continue.

9.3.3 Co-relating with Event Trace

To debug your own (custom) code, it is often helpful to know which section of the code (file name & line number) generated the event under study. There are 2 ways to enable this feature.

Procedure 1

Step 1: Open configuration.netsim file and provide the file name, path and set status as Enable.

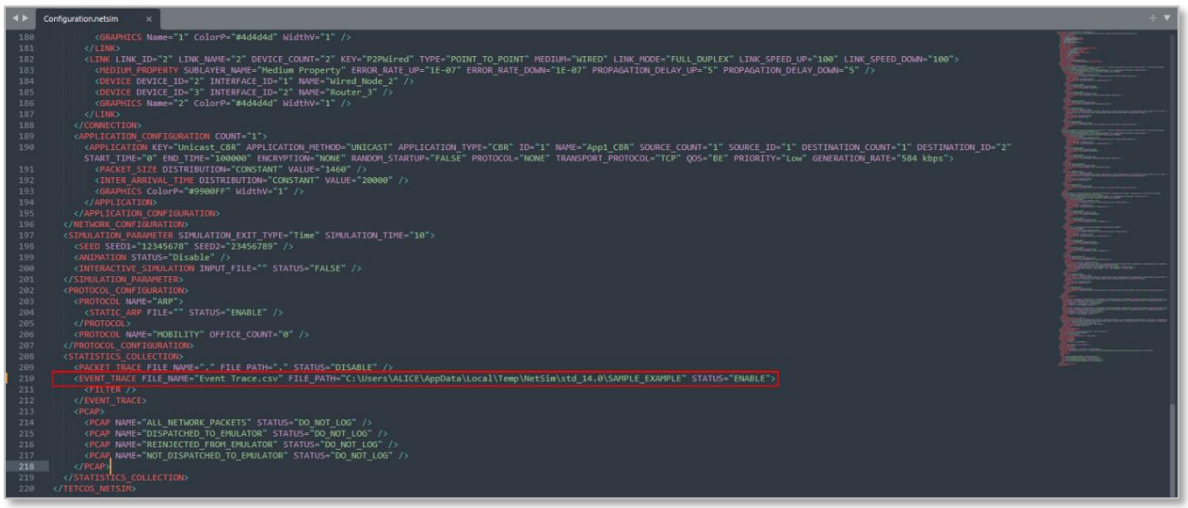


Figure 9-37: Enable Event Trace by editing Configuration.netsim and provide the file name, path and set status as Enable

Step 2: Run the NetSim via CLI in debug mode (Refer **Section 7**→Running Simulation via CLI) with –d as the fourth parameters.

Press enter.

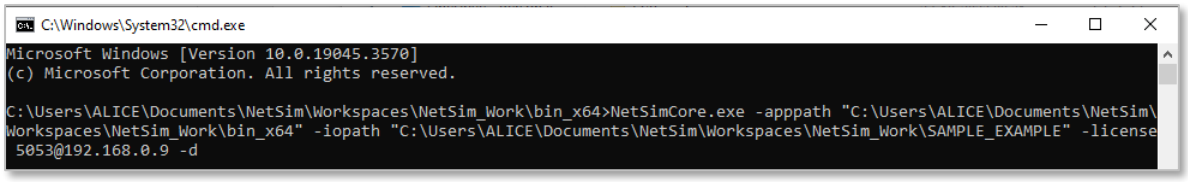


Figure 9-38: Run the NetSim via CLI

Step 3: Enter -1 as the event ID.

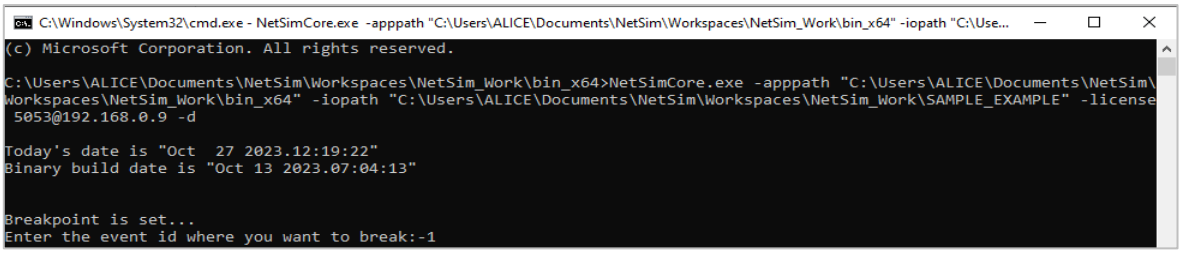


Figure 9-39: Enter -1 as the event ID

Upon running, NetSim will write the file name and line number of the source code that generated each event.

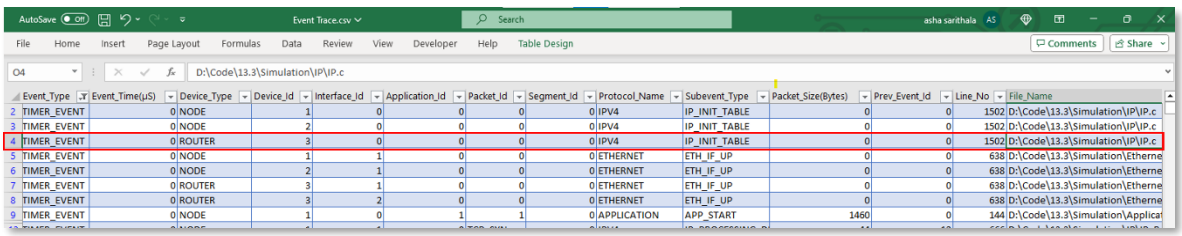


Figure 9-40: NetSim writes the file name and line number of the source code in Event Trace

In the above trace file Event Id 46 is triggered inside the IEEE802_11_Phy.c file which is present in IEEE802_11 project. Since all the lib files are opaque to the end user, you cannot see the source code of the lib file. However, Event Id 56 is triggered at line number 420 of IEEE802_11_Phy.c file and you can find the location of the event by opening the IEEE802_11_Phy.c file as shown below.

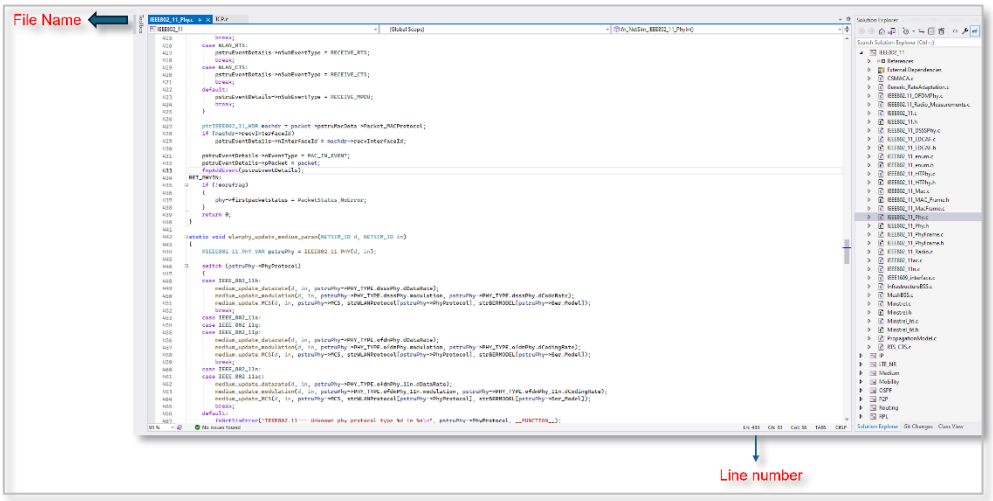


Figure 9-41: IEEE802_11_Phy.c file Code in Visual Studio

Procedure 2:

Step 1: Right click on my computer and select Properties.

Step 2: Go to Advanced System setting → Advanced Tab → Environment Variables.

Step 3: Click New. Type “NETSIM_BREAK” as Variable name and any negative integer as Variable value. Click OK.

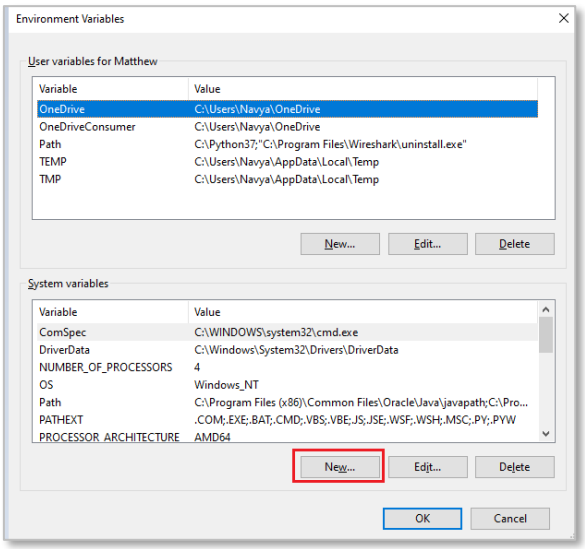


Figure 9-42: Environment Variables Window

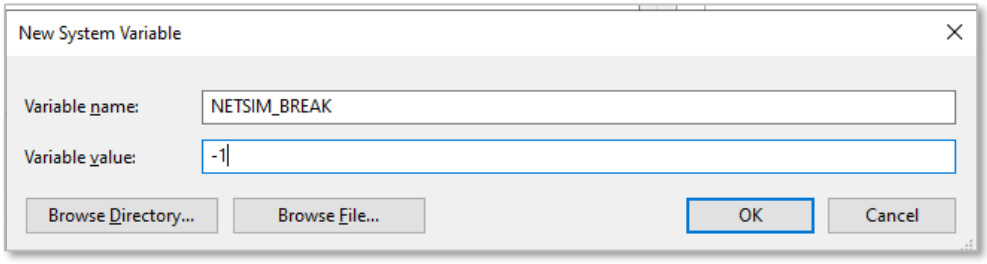


Figure 9-43: Enter Variable name and Value in New System Variable

Step 4: Restart the system.

Step 5: Now perform simulation in NetSim (Enable event trace in GUI). Upon running, NetSim will write the file name and line number of the source code that generated each event.

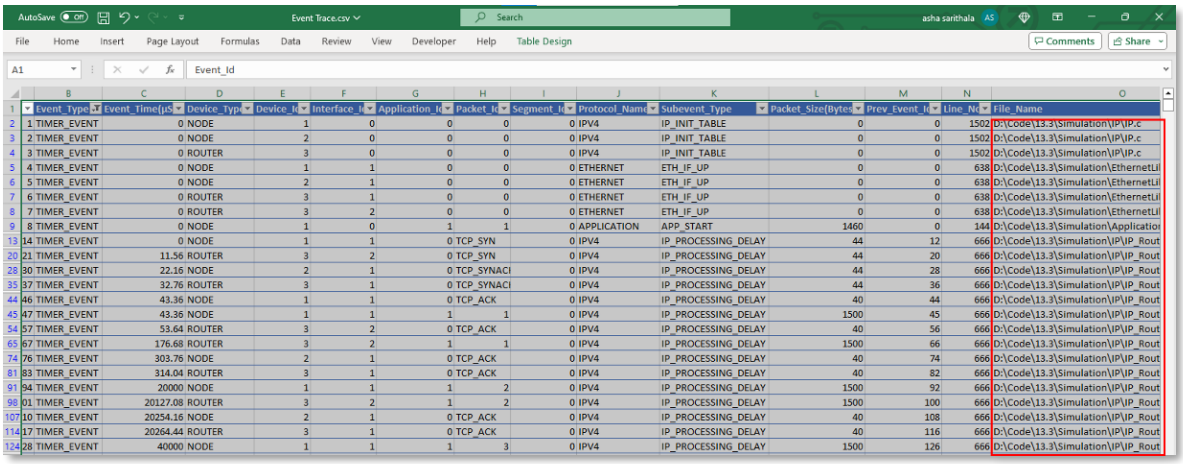


Figure 9-44: NetSim writes the file name and line number of the source code in Event trace

9.3.4 Viewing & Accessing variables

Viewing variables while debugging code

To see the value of a variable, when debugging the mouse over the variable name in the code. A text box with variable contents appears. If the variable is a structure and contains other variables, then click on the plus sign which is there to the left of the text box. Users can pin the variable to watch by clicking on the pin icon to the right of that variable in the text box.

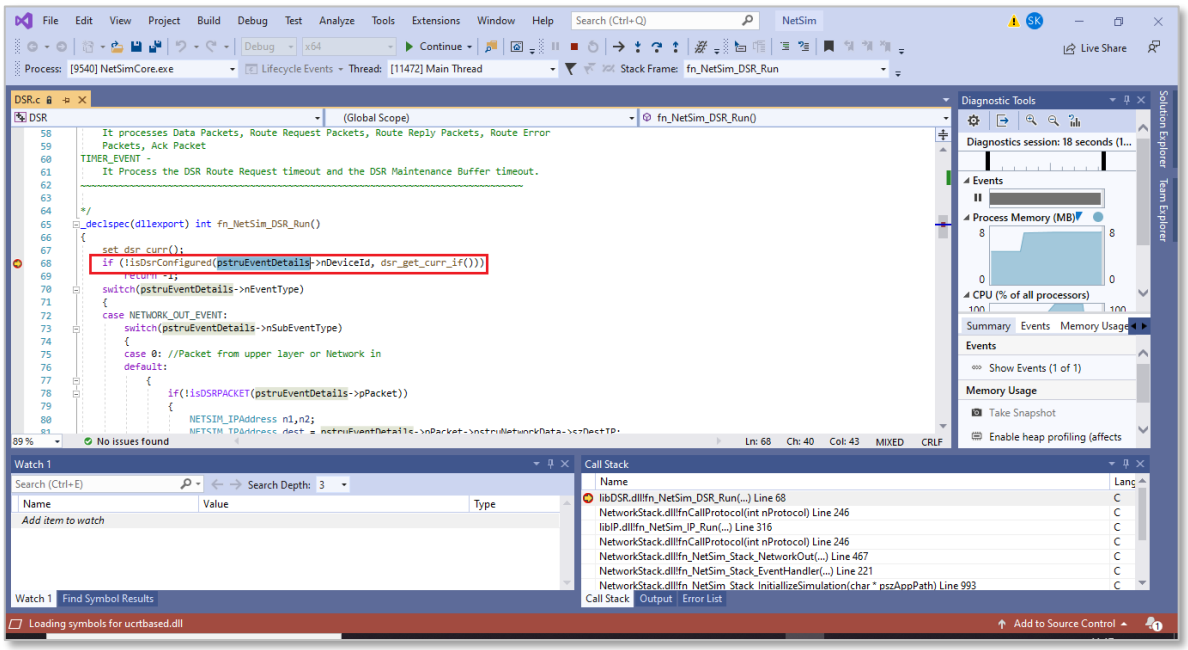


Figure 9-45: Viewing variables while debugging code

Adding the variable to watch

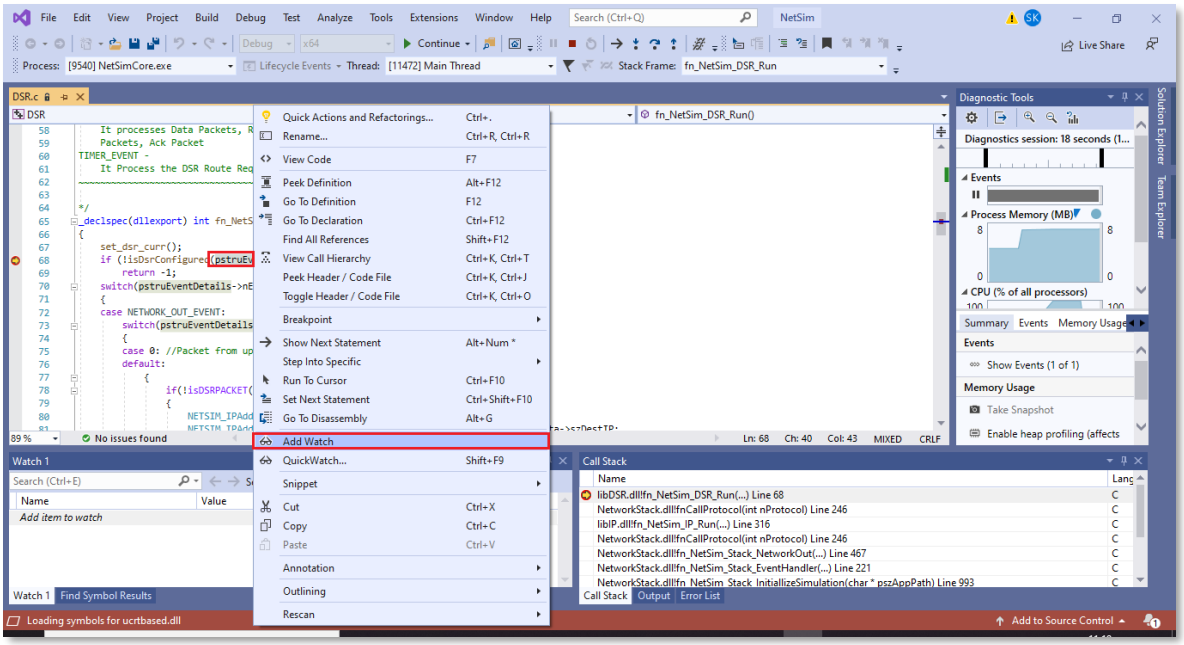


Figure 9-46: Adding the variable to watch

Watch the change in the variable as the code progress by right clicking on the variable & clicking on "**add watch**" tab. This is useful if to continuously monitor the change in the variable as the code progresses.

Viewing external variables

During the process of debug users would come across variables that are defined outside the source file being built as a .dll. Such variables cannot be viewed directly when added in the watch tab, as this would throw the error.

Error: Identifier “pstruEventDetails” is undefined

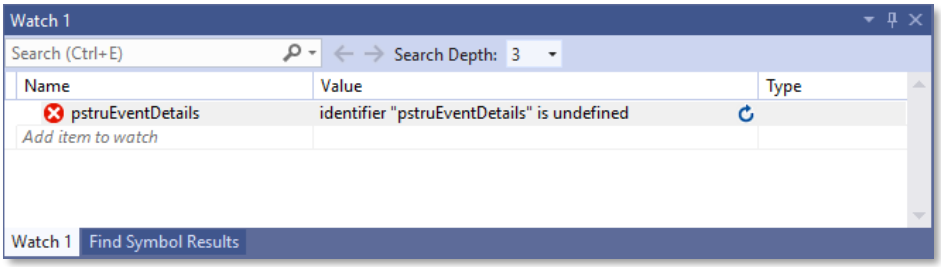


Figure 9-47: Viewing external variables

In the watch window, the variable which the user has to watch should be edited by double clicking on it and prefixing {,,**NetworkStack.dll**} to the variable name and pressing enter. (The name of the respective file in which the variable is defined should be mentioned - in this case **NetworkStack.dll**).

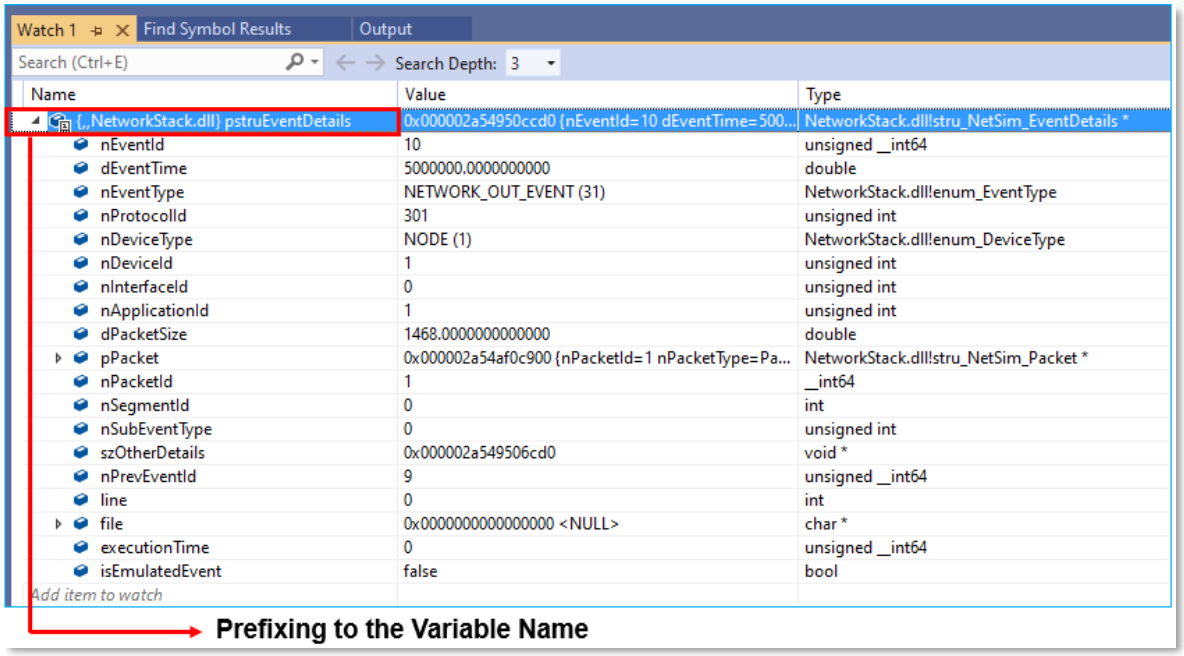


Figure 9-48: Variable In the watch window

Accessing External Variables

Each protocol in NetSim has a separate DLL file which contains the variables and functions which can be shared. In case of cross layer protocol implementations variables of one protocol may have to be accessed from another DLL.

An example is given below showing how Physical layer parameters of devices running IEEE802.11 can be accessed in the Network Layer with DSR protocol configured.

The variable **battery** is defined in a structure **stru_802_11_Phy_Var** which is part of **IEEE802_11_Phy.h** file**.** So the user will have to access a pointer of type **stru_802_11_Phy_Var**. In the header file where the structure definition is given, the following line of code must be written –

```
#ifndef SHARE_VARIABLE
        _declspec(dllexport) IEEE802_PHY_VAR* var1;
#else
        _declspec(dllimport) IEEE802_PHY_VAR* var1;
#endif
```

In the example, the code line must be written in IEEE802_11_Phy.h file present inside IEEE802_11 folder.

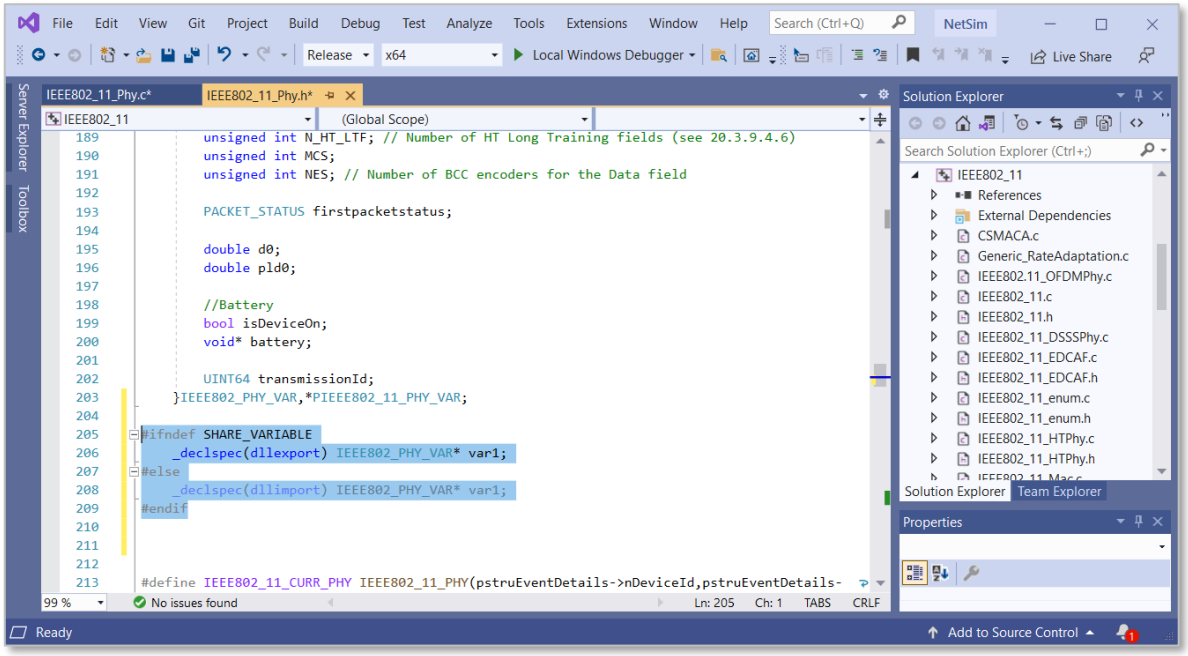


Figure 9-49: Code Modification done in IEEE802_11_Phy.h file present inside IEEE802_11 folder

In the main function where a user wishes to find the dReceivedPower_mw, the variable must be assigned the respective value. In the above case, the following line of code must be written inside fn_NetSim_IEEE802_11_PhyIn() function in IEEE802_11_Phy.c file present inside IEEE802_11 folder.

```
var1 = DEVICE_PHYVAR(pstruEventDetails->nDeviceId, pstruEventDetails->nInterfaceId);
```

Note that the parameters given in the macro or any function which assigns a value to the variable must be defined beforehand in the code. Here nDeviceId and nInterfaceId are defined beforehand.

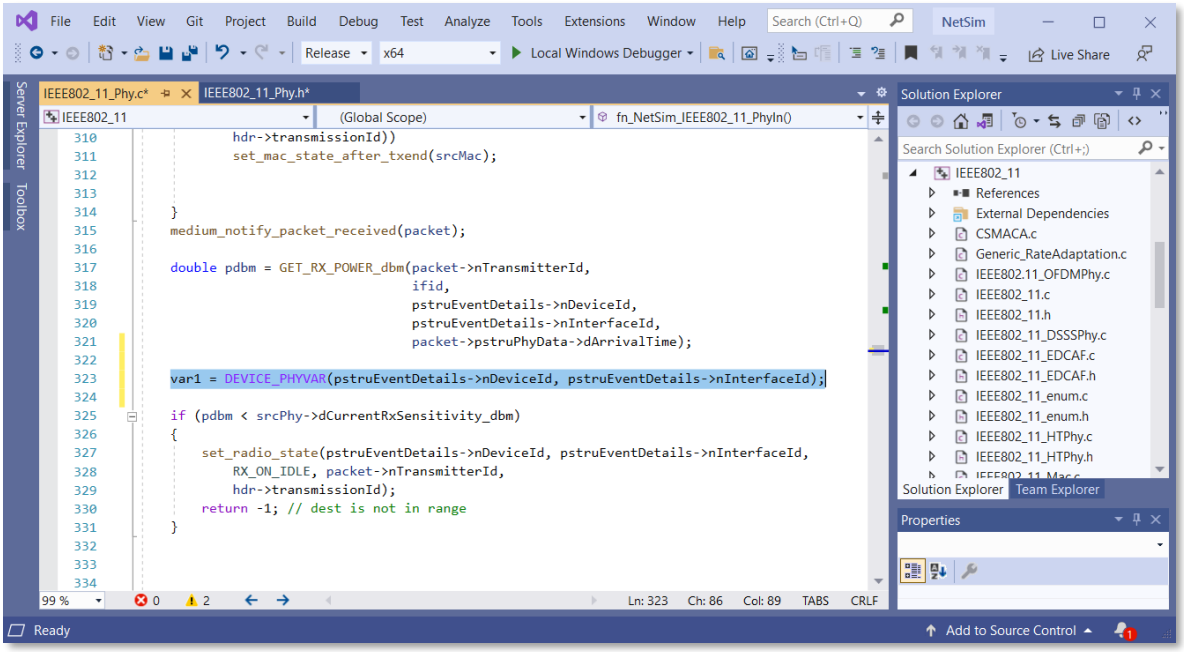


Figure 9-50: Code Modification done in IEEE802_11_Phy.c file present inside IEEE802_11 folder

The IEEE802_11 project must be built and the resulting libIEEE802.11.dll file which gets created in the bin_x64 folder of NetSim's current workspace.

C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64 for 64-bit

The Object file IEEE802_11.lib which is also got created in the bin_x64 folder located in the current workspace path

<C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64 >

Place the created IEEE802_11.lib inside simulation **lib x64** or **lib** folder as shown in below path.

<C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace\src\Simulation\

lib_x64>for 64-bit

Now expand the DSR project in solution explorer. For accessing the IEEE802_11 variable, the following lines must be added in DSR.h file.

```
#define SHARE_VARIABLE
#pragma comment(lib,"IEEE802_11.lib")
```

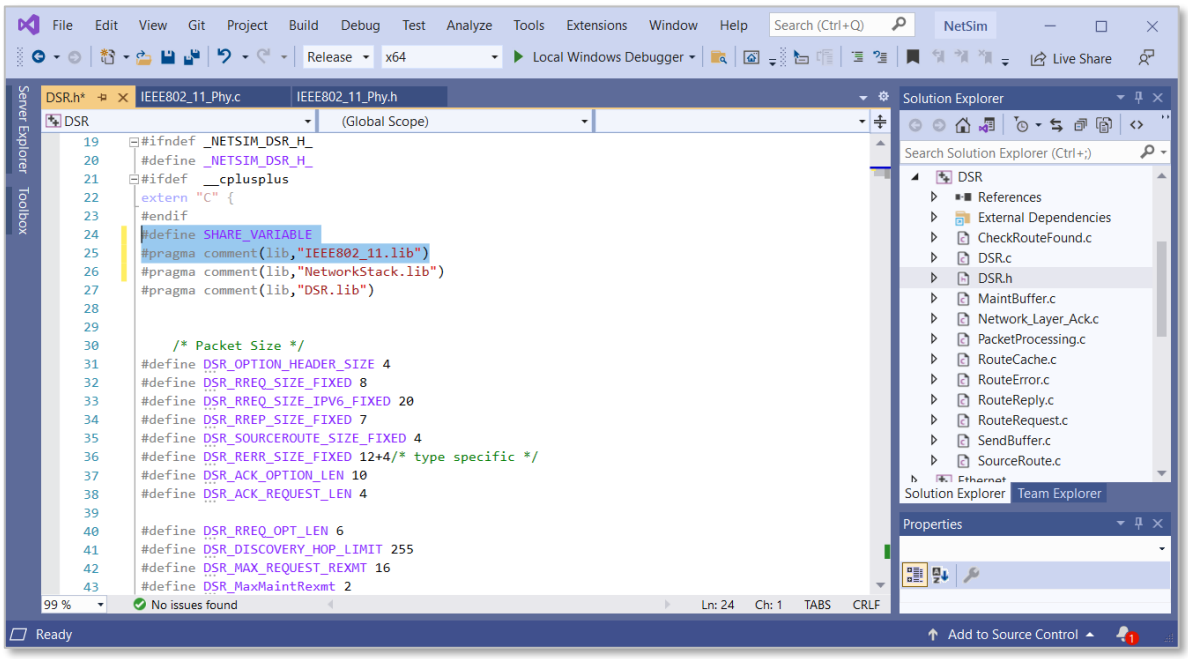


Figure 9-51: Accessing the IEEE802_11 variable, Modification done in DSR.h file

Add the following lines of code to the DSR.c file as shown below.

```
#include "../IEEE802_11/IEEE802_11_Phy.h"
#include "../BatteryModel/BatteryModel.h"
```

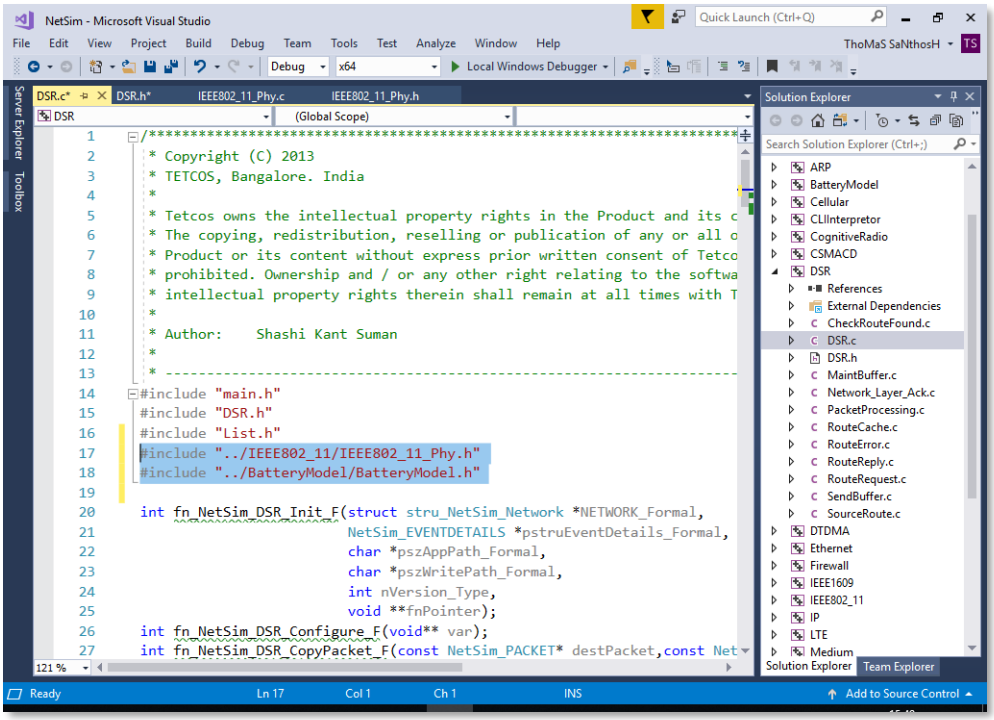


Figure 9-52: Add the following lines of code to the DSR.c file in DSR Project

In the fn_NetSim_DSR_Run() function add the following lines of code to print the value of dReceivedPower_mw variable from DSR project.

```
if (var1)
        fprintf(stderr, "\n Remaining Energy(mJ): %lf\n",
battery_get_remaining_energy((ptrBATTERY)var1->battery));
```

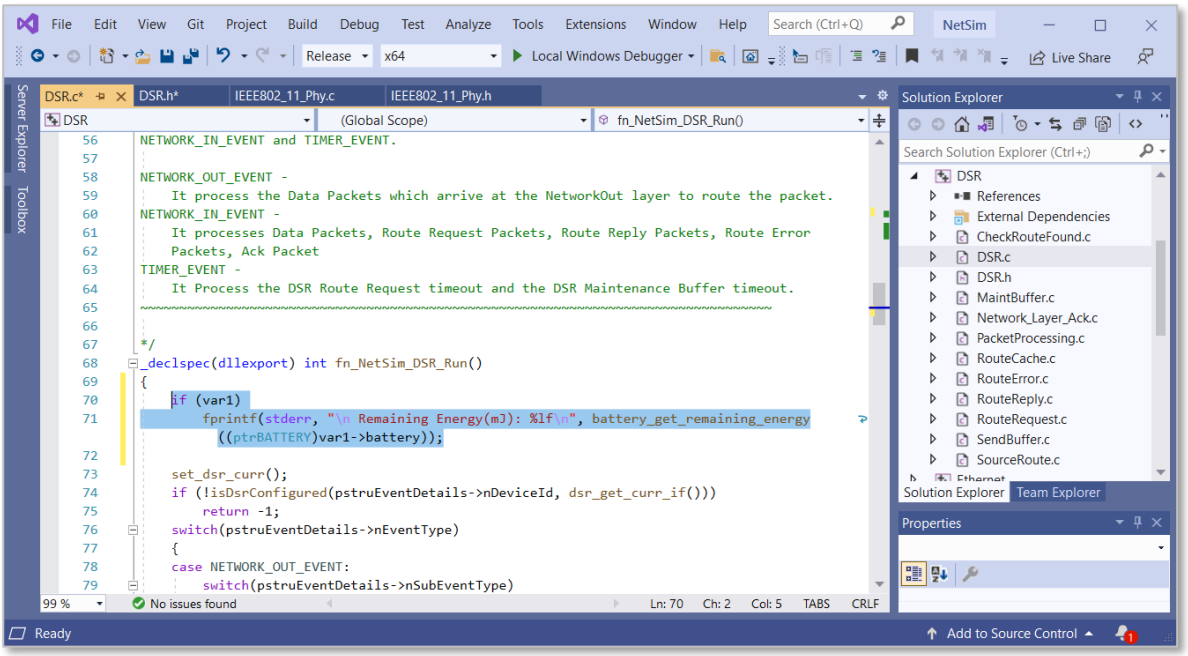


Figure 9-53: Code Related to Remaining Energy for nodes

The DSR project must be built and the resulting libDSR.dll file gets created in the bin_x64 folder of NetSim's current workspace path

< C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64> for 64-bit. When a scenario is run, the remaining energy of the node will be printed to the simulation console as shown below.

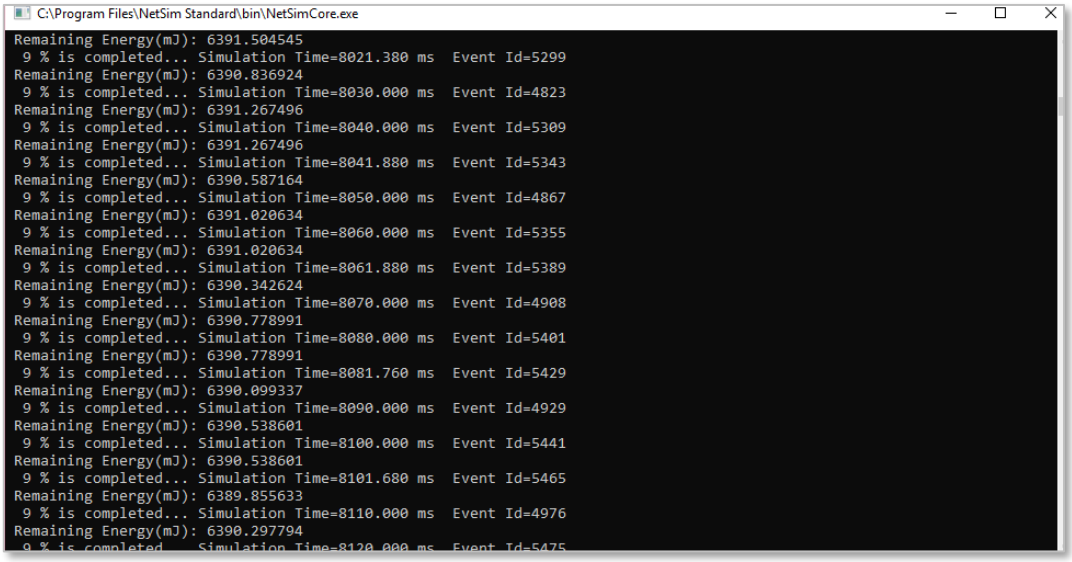


Figure 9-54: Remaining energy of the node printed in NetSim console

9.3.5 Print to console window in NetSim

Users can try printing the Device ID, Application ID, Duplicate Ack Count etc.

To print to console: Print node positions in MANET

Open Mobility Project, and in Mobility.c file go to fn_NetSim_Mobility_Run() function. Inside the default case add following codes

```
fprintf(stderr, "\n The position of %s at time %.2lfms is X=%.2lf and Y = %.2lf \n",
DEVICE_NAME(pstruEventDetails->nDeviceId),
                pstruEventDetails->dEventTime,
                DEVICE_POSITION(pstruEventDetails->nDeviceId)->X,
                DEVICE_POSITION(pstruEventDetails->nDeviceId)->Y);

_getch();
```

```
419                 pstruEventDetails->dEventTime-=pstruMobilityVar->dCalculationInterval;
420             }
421
422             //call all the callback function
423             for(nLoop=0;nLoop<nCallBackCount;nLoop++)
424             {
425                 fnMobilityCallBack[nLoop](pstruEventDetails->nDeviceId);
426             }
427         }
428         fprintf(stderr, "\n The position of %s at time %.2lfms is X=%.2lf and Y = %.2lf \n", DEVICE_NAME(pst
429             pstruEventDetails->dEventTime,
430             DEVICE_POSITION(pstruEventDetails->nDeviceId)->X,
431             DEVICE_POSITION(pstruEventDetails->nDeviceId)->Y);
432         _getch();
433
434         break;
435     }
436     return 1;
437 };
```

Figure 9-55: Code for Print node positions in MANET

Building Mobility project creates libMobility.dll inside the binary folder(bin_x64) of NetSim's current workspace path

<C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64 > for 64-bit.

Create a scenario in MANET and configure the mobility model of the nodes. During simulation users can notice that the positions of the nodes are displayed in the console w.r.t. the simulation time.

9.4 Creating a new packet and adding a new event in NetSim

In this example we show how users can create their own packet & event in 802.15.4 Zigbee. The same methodology can be applied to any network / protocol.

1. Open the Source codes in Visual studio using the NetSim.sln file of your current workspace folder.

2. Go to ZigBee project and Open 802_15_4.h file and add a subevent called “MY_EVENT” inside enum_IEEE802_15_4_Subevent_Type as shown below.

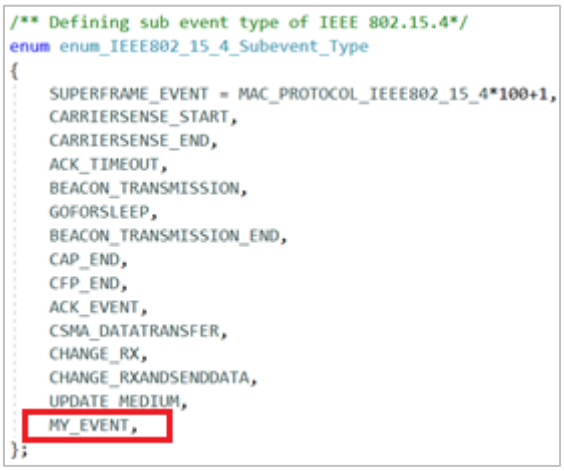


Figure 9-56: Add “MY_EVENT” inside enum_IEEE802_15_4_Subevent_Type

3. To add a new packet in NetSim first user has to initialize their new packet name inside 802_15_4.h file. Let us assume the new packet be “MY_PACKET” and it is a control packet. So, user has to define it inside the following enum as shown below Figure 9-57

```
enum enum_IEEE_802_15_4_ControlPacket_Type
{
    BEACON_FRAME = MAC_PROTOCOL_IEEE802_15_4*100+1,
    SLEEP_FRAME,
    ACK_FRAME,
    MY_PACKET,
};
```

Figure 9-57: Add “MY PACKET” inside enum_IEEE802_15_4_ControlPacket_Type

4. We assume that MY_PACKET has the same fields as a Zigbee Ack and hence we are adding the following ack frame to 802_15_4.h file (Add this code just above the enum enum_IEEE_802_15_4_ControlPacket_Type {} defenition):

```
struct stru_My_Frame
{
        int nBeaconId;
        int nSuperFrameId;
        int nBeaconTime;
        double dPayload;
        double dOverhead;
        double dFrameSize;
};
typedef struct stru_My_Frame MY_FRAME;
enum enum_IEEE_802_15_4_ControlPacket_Type
```

```
        {
```

5. Open 802_15_4.c file, go to the case TIMER_EVENT and add the following code to the subevent type.

```
case SUBEVENT_GETLINKQUALITY:
{
        -------------------------
}
break;
case MY_EVENT:
{
        //my event//
        fprintf(stderr, "My_event");
        pstruEventDetails->dEventTime = pstruEventDetails->dEventTime + 1 * SECOND;
        pstruEventDetails->nDeviceId = nGlobalPANCoordinatorId;
        pstruEventDetails->nInterfaceId = 1;
        pstruEventDetails->nEventType = TIMER_EVENT;
        pstruEventDetails->nSubEventType = MY_EVENT;
        pstruEventDetails->nProtocolId = MAC_PROTOCOL_IEEE802_15_4;
        fnpAddEvent(pstruEventDetails);
        fn_NetSim_WSN_MY_PACKET();
        //my event//
}
break;
```

Here we are adding a new event inside the timer event, and this event will occur every 1 second in the GlobalPANCoordinator. i.e., sink node. In this event, fn_NetSim_WSN_MY_PACKET() is called as explained in step 5.

6. Inside 802_15_4.c file, add the following code at the end of the file for sending ack (broadcast):

```
int fn_NetSim_WSN_MY_PACKET()
{
        double dTime;
        NETSIM_ID nDeviceId = pstruEventDetails->nDeviceId;
        NETSIM_ID nInterfaceId = pstruEventDetails->nInterfaceId;
        IEEE802_15_4_MAC_VAR* pstruMacVar = DEVICE_MACVAR(nDeviceId, nInterfaceId);
        IEEE802_15_4_PHY_VAR* pstruPhyVar = DEVICE_PHYVAR(nDeviceId, nInterfaceId);
        NetSim_PACKET* pstruPacket = pstruEventDetails->pPacket;
```

```
NetSim_PACKET* pstruAckPkt;
MY_FRAME* pstruAck;
dTime = pstruEventDetails->dEventTime;

// Create MY_Frame
pstruAckPkt = fn_NetSim_Packet_CreatePacket(MAC_LAYER);
pstruAckPkt->nPacketType = PacketType_Control;
pstruAckPkt->nPacketPriority = Priority_High;
pstruAckPkt->nControlDataType = MY_PACKET;
pstruAck = fnpAllocateMemory(1, sizeof(MY_FRAME));

// Update packet fields
pstruAckPkt->nSourceId = nDeviceId;
pstruAckPkt->nTransmitterId = nDeviceId;
pstruAckPkt->nReceiverId = 0;
add_dest_to_packet(pstruAckPkt, pstruAckPkt->nReceiverId);
pstruAckPkt->pstruMacData->Packet_MACProtocol = pstruAck;
pstruAckPkt->pstruMacData->dArrivalTime = dTime;
pstruAckPkt->pstruMacData->dStartTime = dTime;
pstruAckPkt->pstruMacData->dEndTime = dTime;
pstruAckPkt->pstruMacData->dPacketSize =
        pstruAckPkt->pstruMacData->dOverhead;
pstruAckPkt->pstruMacData->nMACProtocol = MAC_PROTOCOL_IEEE802_15_4;
pstruAckPkt->nPacketId = 0;
strcpy(pstruAckPkt->szPacketType, "MY_PACKET");
// Add SEND ACK subevent
pstruEventDetails->dEventTime = dTime;
pstruEventDetails->dPacketSize =
        pstruAckPkt->pstruMacData->dPacketSize;
pstruEventDetails->nSubEventType = 0;
pstruEventDetails->nEventType = PHYSICAL_OUT_EVENT;
pstruEventDetails->pPacket = pstruAckPkt;
fnpAddEvent(pstruEventDetails);

//Free the packet
fn_NetSim_Packet_FreePacket(pstruPacket);
pstruPacket = NULL;
```

```
        return 0;
}
```

7. Inside the above function NetSim API fn_NetSim_Packet_CreatePacket(MAC_LAYER); is used. This is the API which creates a new packet in NetSim. Since in this example, new packet is created in MAC layer, it is passed as an argument. Users can give the respective Layer name for creating packets in any other layers. In the above code users can see the following line:

```
strcpy(pstruAckPkt->szPacketType, "MY_PACKET");
```

8. In 802_15_4.c file, go to fn_NetSim_Zigbee_Init() function and add the following code to call the timer_event. i.e. MY_EVENT

```
_declspec(dllexport) int fn_NetSim_Zigbee_Init()
{
        init_linkpacketlog();
        RadioMeasurements_802_15_4_Init();
        //MY_EVENT
        pstruEventDetails->nDeviceId = nGlobalPANCoordinatorId;
        pstruEventDetails->nInterfaceId = 1;
        pstruEventDetails->dEventTime = pstruEventDetails->dEventTime;
        pstruEventDetails->nEventType = TIMER_EVENT;
        pstruEventDetails->nSubEventType = MY_EVENT;
        pstruEventDetails->nProtocolId = MAC_PROTOCOL_IEEE802_15_4;
        fnpAddEvent(pstruEventDetails);
        //MY_EVENT
        return fn_NetSim_Zigbee_Init_F();
}
```

In the above function, subevent type, “MY_EVENT” is called. So this function calls the MY_EVENT, timer event to execute.

9. fn_NetSim_Zigbee_Trace() is an API to print the trace details to the event trace. So inside 802_15_4.c file add the following lines of code inside fn_NetSim_Zigbee_Trace(int nSubEvent) as shown below:-

```
_declspec (dllexport) char* fn_NetSim_Zigbee_Trace(int nSubEvent)
{
        if (nSubEvent == MY_EVENT)
                return "MY_EVENT";
```

```
        return (fn_NetSim_Zigbee_Trace_F(nSubEvent));
}
```

10. Save the code and build Zigbee project, libZigBee.dll will get created in the bin folder of NetSim's current workspace path.

< C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64 > for 64-bit.

11. Create a basic scenario in WSN with 2 sensors and 1 sink node.
12. While creating the application between two sensors, set

    Application Type - Sensor App

    Interarrival Time – 5000

13. Enable packet trace and event trace and Run the simulation for 10 seconds.
14. Open packet trace and users can filter the control packet and see all the packet details of "MY_PACKET" written in the packet trace.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3

Figure 9-58: Filter the Packet Type to control packet and See "MY_PACKET" in Packet Trace

15. To analyze the "MY_EVENT" users can open event trace and filter the subevent type as "MY_EVENT". Here users can analyze that the event occurs for every 1 seconds.

Event_Id	Event_Type	Event_Tir	Device_Type		Ir	A		!	Protocol_Name	Subevent_Type
4	TIMER_EVENT	0	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
9	TIMER_EVENT	1000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
4280	TIMER_EVENT	2000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
8519	TIMER_EVENT	3000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
12820	TIMER_EVENT	4000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
17025	TIMER_EVENT	5000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
21223	TIMER_EVENT	6000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
25452	TIMER_EVENT	7000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
29735	TIMER_EVENT	8000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
33993	TIMER_EVENT	9000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT

Figure 9-59: Filter Subevent type to "MY_EVENT"

9.5 NetSim API's

NetSim provides a wide variety of APIs for protocol developers. These are available in

1. **packet.h** – Packet related APIs
   - Create a new packet.
     - fn_NetSim_Packet_CreatePacket_dbg(int nLayer,int line,const char* file);
   - Copy a packet into a new packet.
     - fn_NetSim_Packet_CopyPacket_dbg(const NetSim_PACKET* pstruPacket,int line,const char* file);
   - Create error in packet.
     - fn_NetSim_Packet_DecideError(double dBER, long double dPacketSize);
   - Free a packet
     - fn_NetSim_Packet_FreePacket_dbg(NetSim_PACKET**pstruPacket,int line,char* file);
2. **stack.h** – Network / device / link and event related APIs
   - Calculate distance between nodes.
     - fn_NetSim_Utilities_CalculateDistance(NetSim_COORDINATES* coordinate1,NetSim_COORDINATES* coordinates2);
   - Stores the event details. Only one-time memory is allocated. Most used variable
     - struct stru_NetSim_EventDetails* pstruEventDetails;
   - Retrieve values from xml file.
     - GetXmlVal(void* var,char* name,void* xmlNode,int flag, XMLDATATYPE type);
3. **list.h** -- Optimized list operation calls since NetSim uses lists extensively.
   - Add elments in list.
     - list_add(void** list,void* mem,size_t offset,int (*check)(void* current,void* mem));
   - Sorting the list
     - list_sort(void** list,size_t offset,int (*check)(void* current, void* mem));
4. **IP_Addressing.h** – For setting & getting IP address per the appropriate format.
   - Set Ip address of any node.

- NETSIM_IPAddress

- Checking ip address is broadcast or multicast.
  - isBroadcastIP(NETSIM_IPAddress ip);
  - isMulticastIP(NETSIM_IPAddress ip);

For detailed help please refer the appropriateheader (.h) files inside:/NetSim_Standard/src/simulation/include or read through the doxygen source code documentation available inside Home Page under Documentation > source code Help

- Include all the header (.h) files from the include folder.
- NetworkStack.lib is a “import library” file and has the definitions for the functions present in the NetworkStack.dll
- When developing new protocols users should create their own protocol.h and declare all the protocol specific variables here. Stack & packet related variables should be used from stack.h and packet.h

NetSim Network Stack calling individual Protocol.

Every protocol should provide the following APIs as hooks to the network stack:

- **int**(*fn_NetSim_protocol_init)(**conststruct** stru_NetSim_Network*,**conststruct** stru_NetSim_EventDetails*,**constchar*,constchar*,int,constvoid****);
- Using this API the stack passes all the relevant pointers to variables, paths etc needed for the protocol. Inside this function a) local variables should be initialized, b) Initial events if any should be written, eg: Hello packet in RIP, STP in Ethernet c) File pointers for reading & writing protocol_specific_IO files.
- **int** (*fn_NetSim_protocol_Configure)(**conststruct** stru_NetSim_Network*,**int** nDeviceId, **int** nINterfaceID, **int** nlayertype, fnpAllocateMemory, fnpFreeMemory, fpConfigLog);
- The stack calls this API when reading the config file. Upon reaching the appropriate protocol definition in the XML file, the stack calls this and passes all these pointers to the protocol.
- **int** (*fn_NetSim_protocol_run)(): This is called by the stack to run the protocol
- **char*** (*fn_NetSim_protocol_trace)(**int**): This called by the stack to write the event trace
- **int**(*fn_NetSim_protocol_CopyPacket)(**const**NetSim_PACKET* pstruDestPacket,**const** NetSim_PACKET* pstruSrcPacket):
- This is for copying protocol specific parameters / data into the packed
- int (*fn_NetSim_protocol_FreePacket)(**const** NetSim_PACKET* pstruPacket): The this to free the protocol specific parameters / data in the packet
- (*fn_NetSim_protocol_Metrics)(**const** FILE* fpMetrics): This is to write the metrics file upon completion of the simulation

- **int** (*fn_NetSim_protocol_Finish)(): To release all memory after completion
- char* (*fn_NetSim_protocol_ConfigPacketTrace)(**constvoid*** xmlNetSimNode); To configure the packet packet trace in terms of the parameters to be logged
- char* (*fn_NetSim_protocol_WritePacketTrace)(**const** NetSim_PACKET*); To configure the event trace in terms of the parameters to be logged.

10 Advanced Features

10.1 Random Number Generator and Seed Values

NetSim includes protocol and traffic models which include stochastic behavior. Typical examples are a) Wi-Fi node's random back-off after collisions, and b) packet error decision by comparing a random number chosen between 0 and 1 against the packet error probability.

NetSim uses an in-built prime modulus combined with a linear congruential Random Number Generator (RNG) to generate the randomness. It is a single stream RNG with a period of $\approx 2 \times 10^{18}$. Two seeds values are used to initialize the RNG. These seeds can be input from NetSim GUI or via the config file. Having the same set of seed values ensures that for a particular network configuration the same output results will be got, irrespective of the system or time at which the simulation is run. This ensures repeatability of experimentation.

Modifying the seed value will lead to the generation of a different stream of random numbers and thereby lead to a different sequence of events in NetSim. Therefore, all simulation results are dependent on the initial seeding of the RNG.

10.2 Confidence in simulation results and error bars

Since NetSim's models include stochastic behavior, results are dependent on the initial seeding of the random number generator. Because a particular random seed selection can potentially result in an anomalous, or non-representative behavior, it is important for each model configuration to be exercised with several random number seeds, to be able to determine standard or typical behavior.

The field of statistics provides methods for calculating confidence in an estimate, based on a trial or series of random trials. To calculate confidence intervals, users can do the following:

1. Run N simulations for the same model configuration, with a different initial seed (for the random number generator) for each run.
2. For any output metric X, calculate the mean (average) $\bar{X}$ of the N samples.

$$\bar{X} = \frac{1}{N}(X_1 + X_2 + \cdots + X_N)$$

3. Calculate the standard deviation σ of the N samples.

$$\sigma = \sqrt{\left(\frac{1}{N-1}\right) \times \Sigma\,(X_i - \bar{X})^2}$$

4. Confidence interval limits can be expressed as

$$\Theta_{\text{lower}} = \bar{X} - z_\alpha \times \left(\frac{\sigma}{\sqrt{(N)}}\right),\ \Theta_{\text{upper}} = \bar{X} + z_\alpha \times \left(\frac{\sigma}{\sqrt{(N)}}\right)$$

This statement can be thought of as assigning a probability to the condition that the true mean μ is within a particular distance of the random sample $\bar{X}$, as shown below

$$Prob\left[\bar{X} - z_\alpha \times \left(\frac{\sigma}{\sqrt{(N)}}\right) < \mu < \bar{X} + z_\alpha \times \left(\frac{\sigma}{\sqrt{(N)}}\right)\right] = \alpha$$

Confidence level (α)	z_α
99%	2.575
98%	2.327
95%	1.960
90%	1.645
80%	1.282

Table 10-1:Value of z_α for different confidence intervals

5. The above expression for confidence assumes the distribution of the output metric is Normal, which is true (from the Central Limit Theorem) if the number of runs $N \geq 30$. If the number of repetitions is less than 30 then it is better to use the t-statistic based confidence-interval (details can be found in standard statistics textbooks).

Error bars are plotted as a vertical bar centered at the mean and ending at the upper and lower confidence limits

10.3 Interfacing MATLAB with NetSim (Std/Pro versions)

NetSim provides run-time interfacing with MATLAB so that users do not have to rewrite code in C for features that are already available in MATLAB and instead simply reuse MATLAB code. Lot of work related to machine learning, artificial intelligence and specialized mathematical algorithms which can be used for networking research, can be carried out using existing MATLAB code.

Figure 10-1: Interfacing MATLAB with NetSim

This interfacing feature can be used to either replace an existing functionality in NetSim or to incorporate additional functionalities supported by MATLAB. Any existing command /function/algorithm in MATLAB or a MATLAB M-script can be used.

In general, the following are done when a user interfaces NetSim to MATLAB:

- Initialize a MATLAB engine process in parallel with NetSim,
- Execute MATLAB workspace commands,
- Pass parameters to MATLAB workspace,

- Read parameters from MATLAB workspace,
- Generate dynamic three-dimensional plots,
- Make calls to functions that are part of MATLAB M-scripts or .m files, and
- Terminate the MATLAB engine process at NetSim simulation end.

Guidelines to interface NetSim with MATLAB

- Analyze what parameters the function or code in MATLAB expects as input.
- Identify the relevant variables in NetSim to be passed as input to MATLAB.
- Make calls from relevant places of NetSim source code to
    - Pass parameters from NetSim to MATLAB.
    - To read computed parameters from MATLAB workspace
- Identify and update the appropriate simulation variables in NetSim.

NetSim offers Socket interfacing to interact with MATLAB during runtime.

The socket interface that offers simplified NetSim API's that can be used for interactions with MATLAB. The steps are lot simpler since no additional settings will be required in the source code project settings.

10.3.1 NetSim-MATLAB Socket Interface

NetSim-MATLAB Socket Interface was introduced in NetSim v14.0. It provides several inbuilt APIs that can be called from the underlying protocol C source codes to interact with MATLAB. Following is some of the APIs with syntax and description:

NetSim API's to interact with MATLAB	Description
netsim_matlab_interface_configure(char* appPath)	Starts a MATLAB engine process and adds the appPath to the top of the search path for current MATLAB session.
netsim_matlab_interface_close();	Sends exit command to MATLAB to terminate the session.
netsim_matlab_send_ascii_command(char* format, ...)	Sends commands, variables and values as string to MATLAB.
netsim_matlab_get_value(char* out, int outLen, char* name, char* type)	Gets the value of a MATLAB variable as a string. Currently supports double data type only.
ptrMXArray netsim_matlab_get_array(char* name)	This function retrieves a MATLAB matrix containing double-precision values specified by the variable name. It returns a pointer to an mxArray structure (ptrMXArray) with the following components: arr: A flat array containing all entries of the MATLAB matrix, irrespective of its dimensions. dimn: An array specifying the dimensions of the MATLAB matrix. shape: An array containing the size of each dimension.

	netsim_matlab_send_array
void netsim_matlab_send_array(double* arr, size_t* shape, size_t dimn, char* name)	This function sends a C array of double-precision values to MATLAB and creates an equivalent variable in MATLAB. It takes the following arguments: arr: The pointer to the C double type array. shape: An array representing the size of each dimension. dimn: The dimension of the array. name: The name of the equivalent MATLAB variable.

Table 10-2: MATLAB Engine API functions

These functions are defined as part of the *NetSim_utility.h* file which is part of the Include directory of NetSim Source Codes. This header can be included in the C files where these APIs are to be called from.

10.3.1.1 Prerequisites for MATLAB Socket Interfacing

- An installed version of MATLAB R2023(b) or lower version in the same system where NetSim is installed.

Note: Execute the below command in the command prompt where MATLAB_Interface.exe is placed followed by IP address of the system where the NetSim is installed.

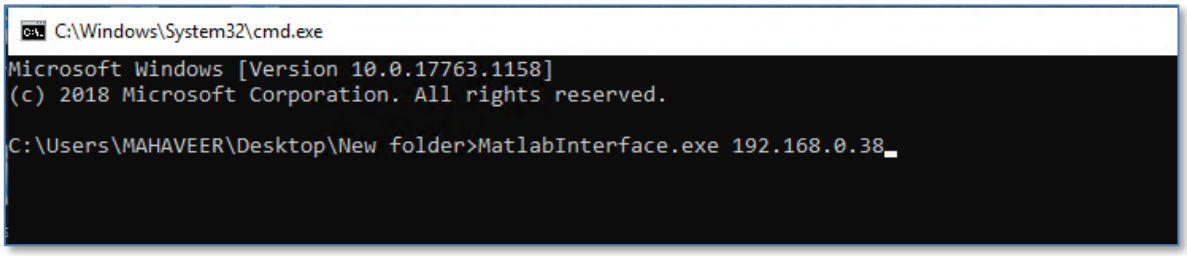


Figure 10-2: Command to start the MATLAB interface

- Registration of MATLAB as a COM server by one of the following methods
    - Start Command Prompt as administrator and execute the following command:

matlab -regserver

Note: If you have multiple versions of MATLAB installed on your computer, the best practice is to run the matlab command from the matlab root folder.

- Enter the following command in the command line of MATLAB version that you want to interface with NetSim:

matlab -regserver

10.3.1.2 Implement Weibull Distribution of MATLAB without using .m file

In this example we will replace the default Rayleigh Fading (part of the path loss calculation) used in NetSim, with a Fading Power calculated using the Weibull Distribution from MATLAB Socket Interfacing.

Procedure

1. Open NetSim Source codes in Visual Studio by going to Your Work -> Open, Reset and Compare option -> Open Code, in the Home Screen of NetSim.
2. In the Solution Explorer under **IEEE802_11** project double click on the **IEEE802_11.c** file.

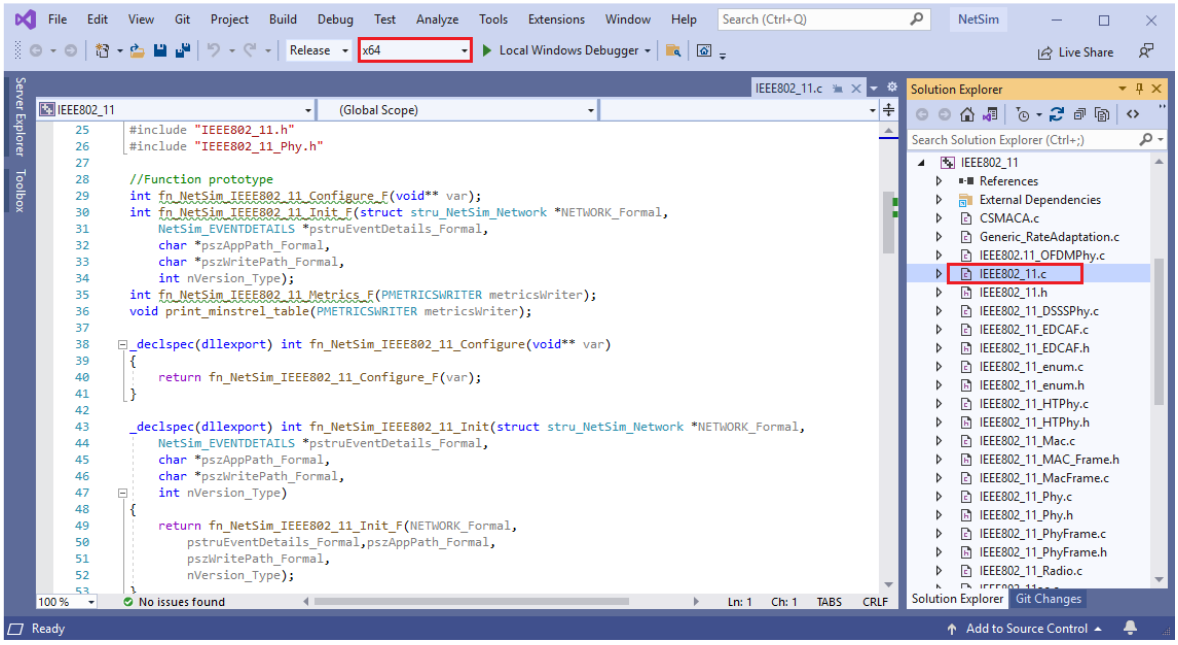


Figure 10-3: Solution Explorer double click on the IEEE802_11.c file

3. The NetSim_utility.h header file is included in IEEE802_11.c

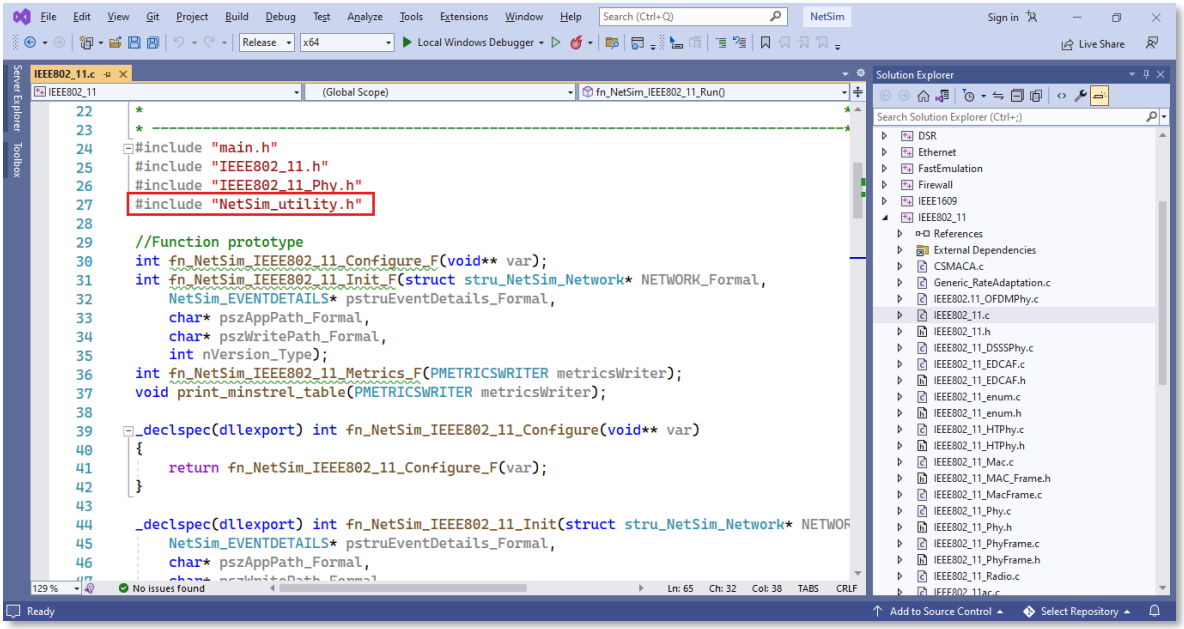


Figure 10-4: NetSim_utility.h file in IEEE802_11.c file

4. Add a call to **netsim_matlab_interface_configure()** API, passing pszAppPath as an argument, inside the fn_NetSim_IEEE802_11_Init() function.

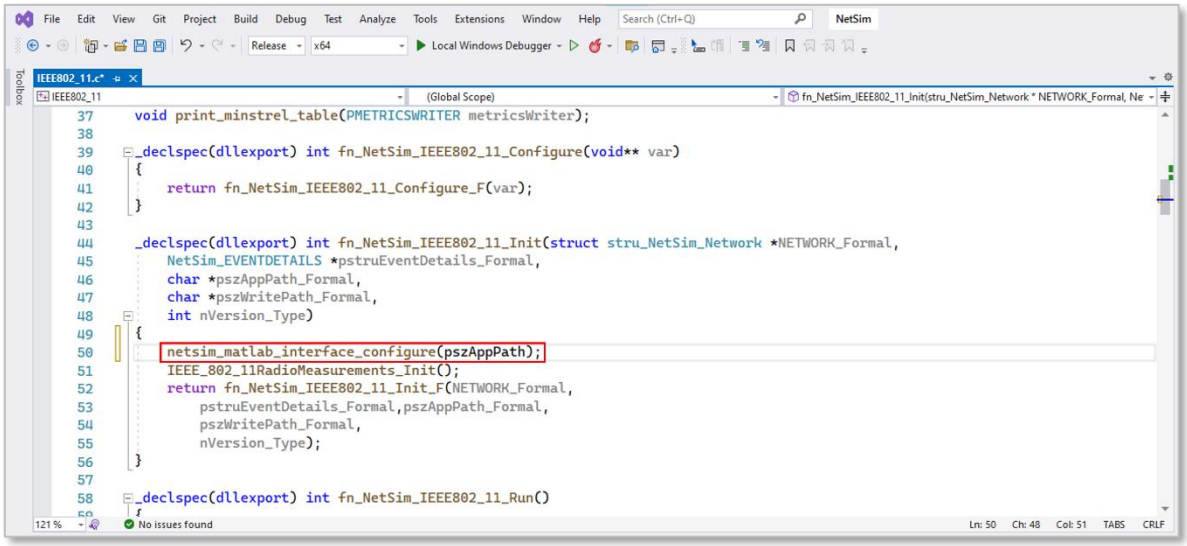


Figure 10-5: Adding a call to netsim_matlab_interface_configure API inside the fn_NetSim_IEEE802_11_Init() function

5. Similarly add a call to netsim_matlab_interface_close(); inside the fn_NetSim_IEEE802_11_**Finish()** function.

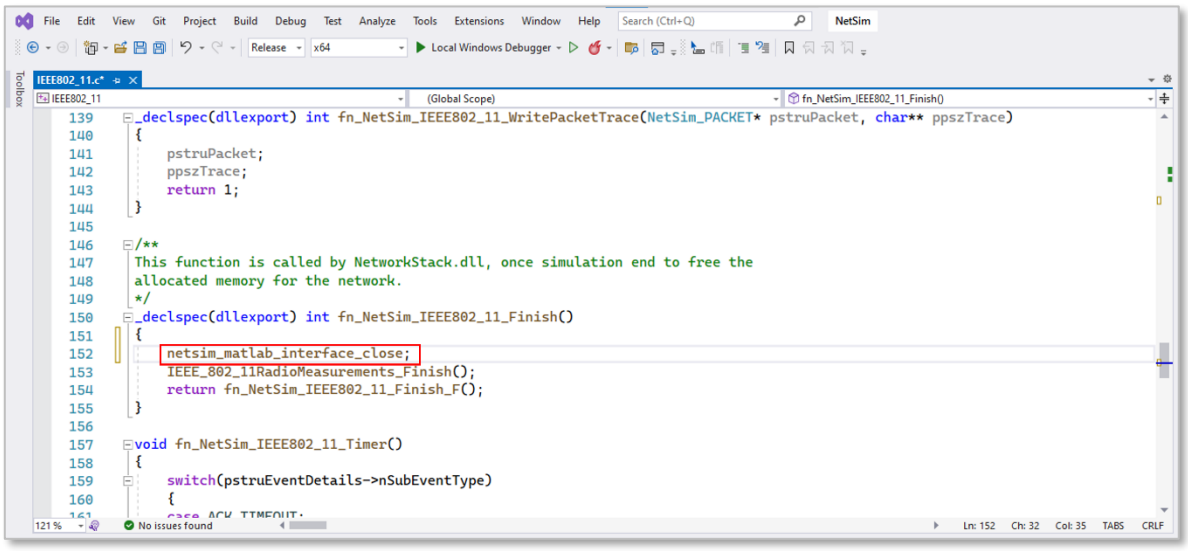


Figure 10-6: Adding a call to netsim_matlab_interface_close() API inside the fn_NetSim_IEEE802_11_Finish() function

6. In the Solution Explorer under expand the Medium project and double click on the Medium.c file.
7. Add a line in the beginning of the file to include the NetSim_utility.h header file.

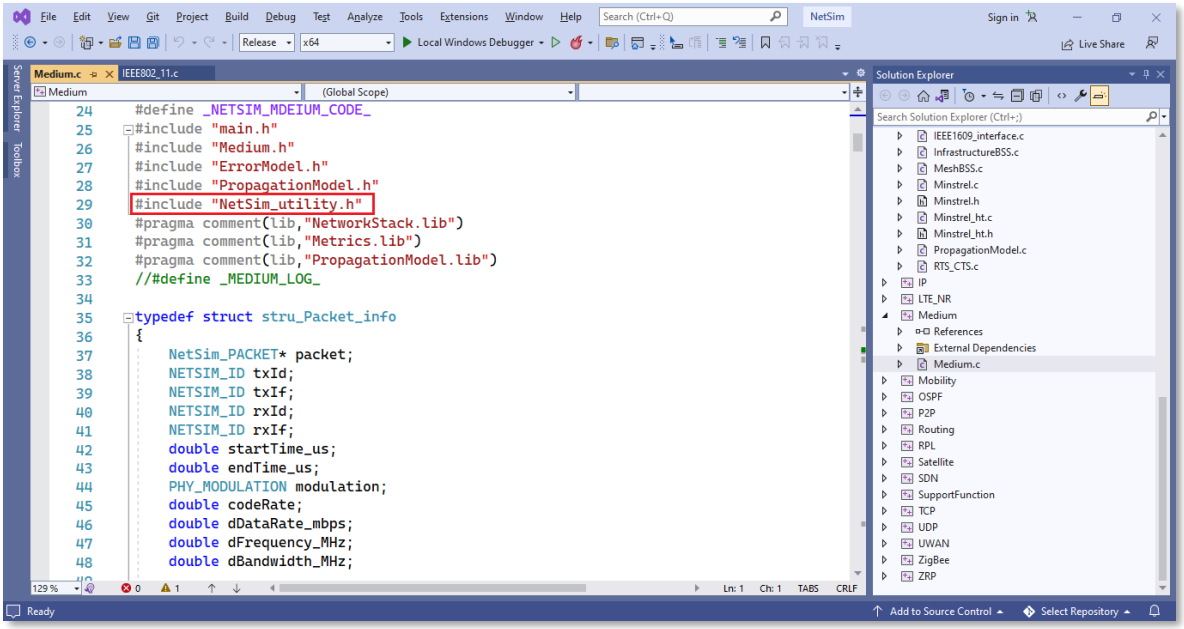


Figure 10-7: Including the NetSim_utility.h file in Medium.c file

8. Define a new function static double matlab_calculate_fadingloss() above the existing static void medium_mark_packet_error(ptrPACKETINFO info) function.

```
static double matlab_calculate_fadingloss()
{
    char buf[BUFSIZ];
```

```
    double result;
    int weibull_noncentrality = 1, weibull_scale = 2;
    //use ProbDistUnivParam() function for matlab 2016 or lower
    //sprintf_s(buf, BUFSIZ, "h=ProbDistUnivParam('weibull',[%d %d])",
    //weibull_noncentrality, weibull_scale);//
    //use makedist() function for matlab 2017 or Higher
    sprintf_s(buf, BUFSIZ, "h=makedist('weibull',%d,%d)", weibull_noncentrality,
            weibull_scale);
    netsim_matlab_send_ascii_command(buf);
    sprintf_s(buf, BUFSIZ, "i=random(h)");//
    netsim_matlab_send_ascii_command(buf);
    netsim_matlab_get_value(buf, BUFSIZ, "i", "double");
    result = atof(buf);
    return result;
}
```

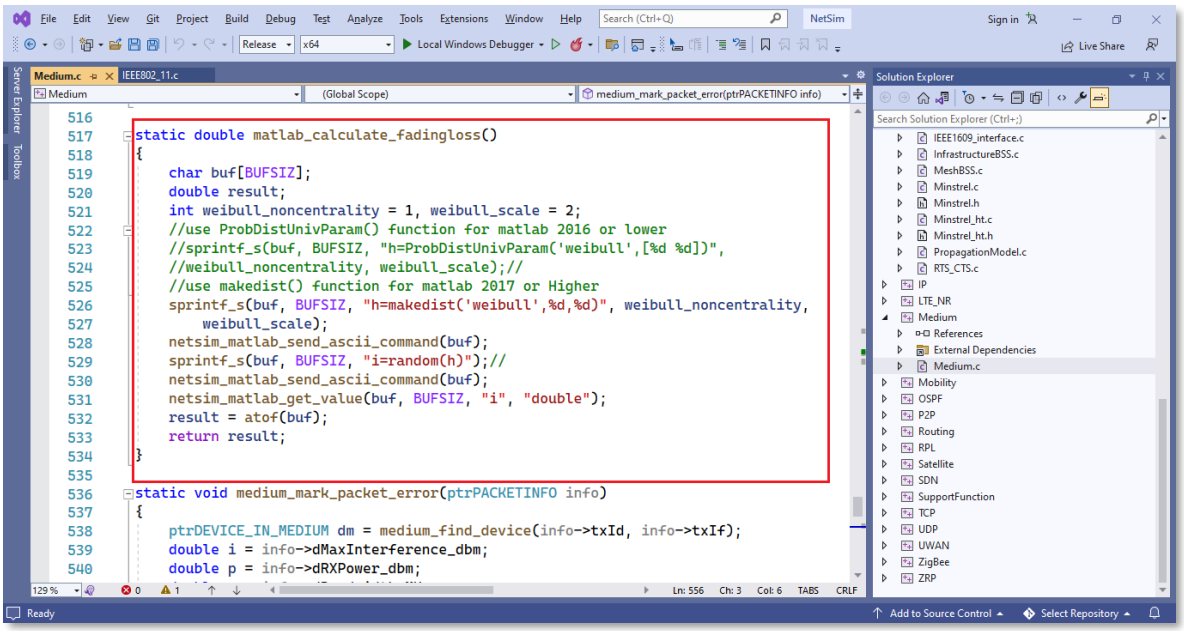


Figure 10-8: Defining a new function matlab_calculate_facingloss() in the Medium.c file

9. Inside **medium_mark_packet_error()** function comment the lines,

```
    double f = _propagation_calculate_fadingloss(dm->propagationinfo_find(
                info->txId, info->txIf,
```

info->rxId, info->rxIf));

10. Make a call to the **fn_netsim_matlab_run()** function by adding the following line,

```
double f = matlab_calculate_fadingloss();
```

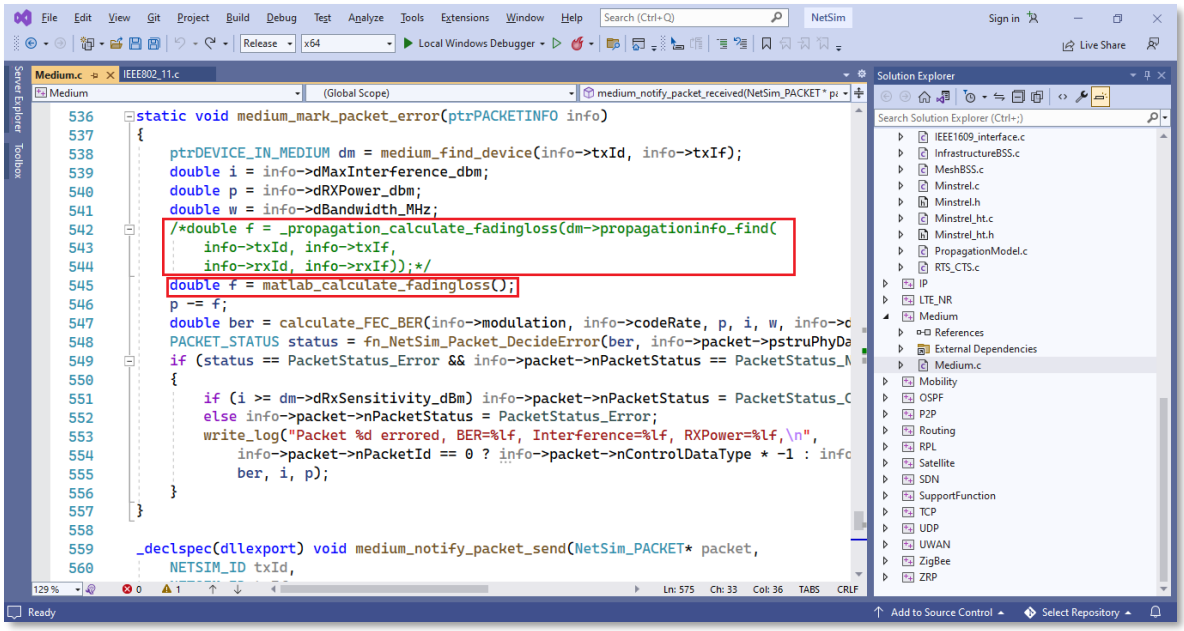


Figure 10-9: Call to the matlab_caculate_fadingloss() function

Note: In MATLAB Socket Interfacing, compilation of MATLAB Engine is not required since the MATLAB APIs are already called in function.

11. Now rebuild the IEEE802_11 and Medium source code projects one after the other, by right clicking on the project and selecting Rebuild option.
12. Upon successful build, NetSim automatically takes care of updating the binaries of IEEE802_11 and Medium source code projects which will be used for further simulations.
13. Run NetSim in Administrative mode. Create a Network scenario involving IEEE802_11 say MANET, right click on the Adhoc link and select properties. Make sure that the Channel Characteristics is set to PathLoss and Fading and Shadowing. Perform the simulation.

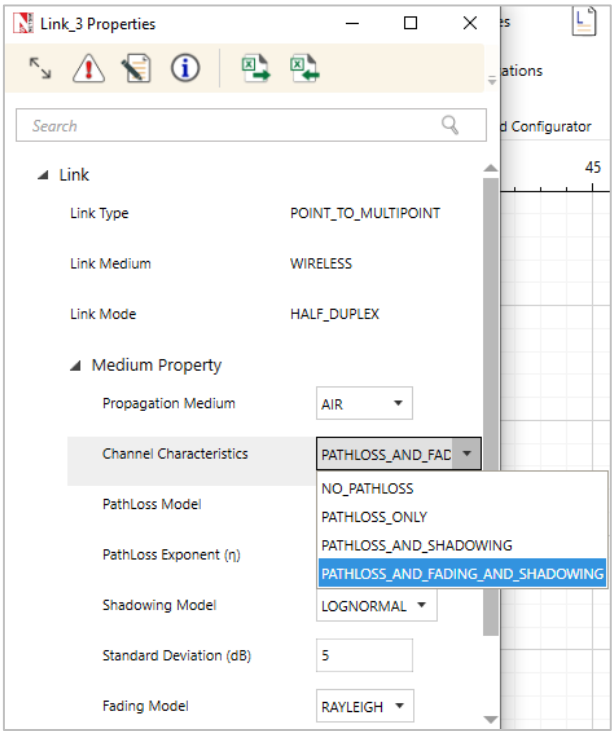


Figure 10-10: Wireless Link Properties Window in MANET

14. If MATLAB is installed in the same system where NetSim is installed. MATLAB Interface process can be initiated directly from the design window of NetSim.

- Go to “Options” in design window and select the Open MATLAB Interface option as shown below:

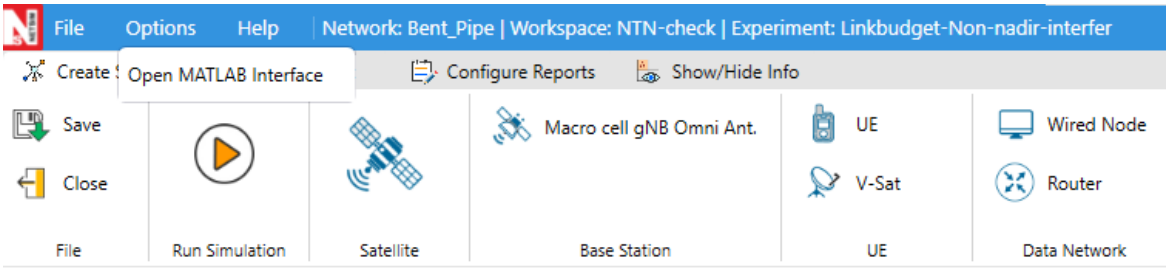


Figure 10-11: Option to open MATLAB Interface

- Click on the OK when the below window is displayed. This opens the MATLAB Session.

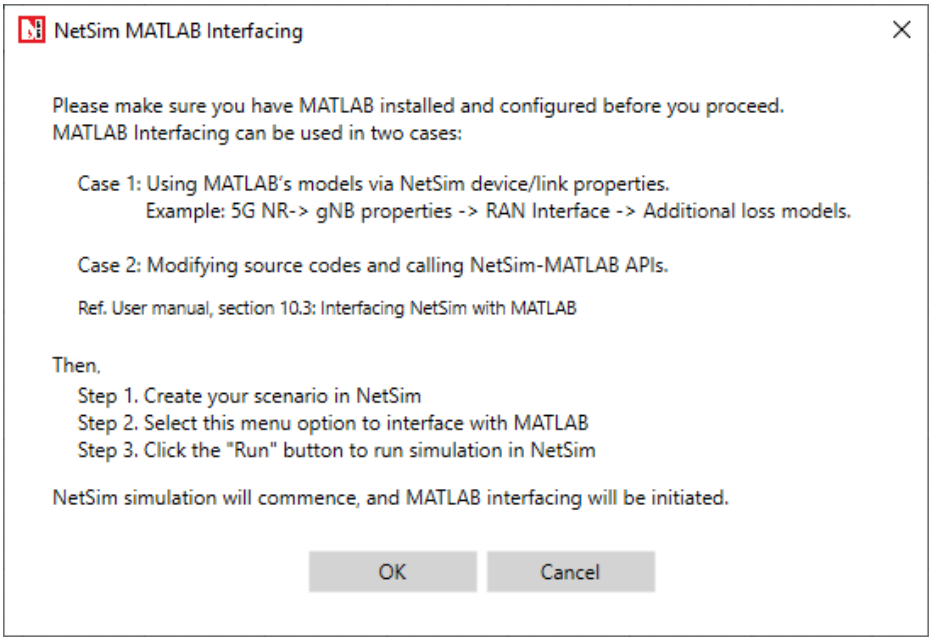


Figure 10-12: NetSim MATLAB Interfacing warning message

15. After clicking on “OK”, MATLAB interface will get initiated. MATLAB Command window will open as shown below

16. Click on run simulation ,after which the simulation NetSim starts to simulate. During the simulation communication between MATLAB and NetSim will be established as shown below

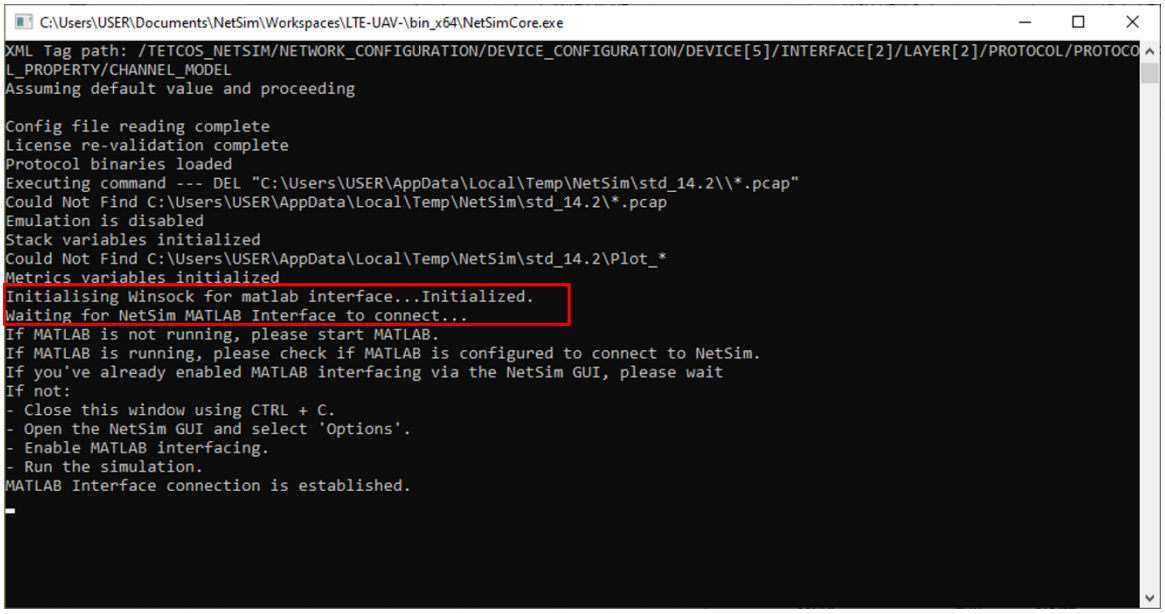


Figure 10-13: NetSim Simulation console for MATLAB Interface process to connect

17. The dFading power value will be displayed in the MATLAB Console window as below. At simulation end the MATLAB Interface process will get terminated.

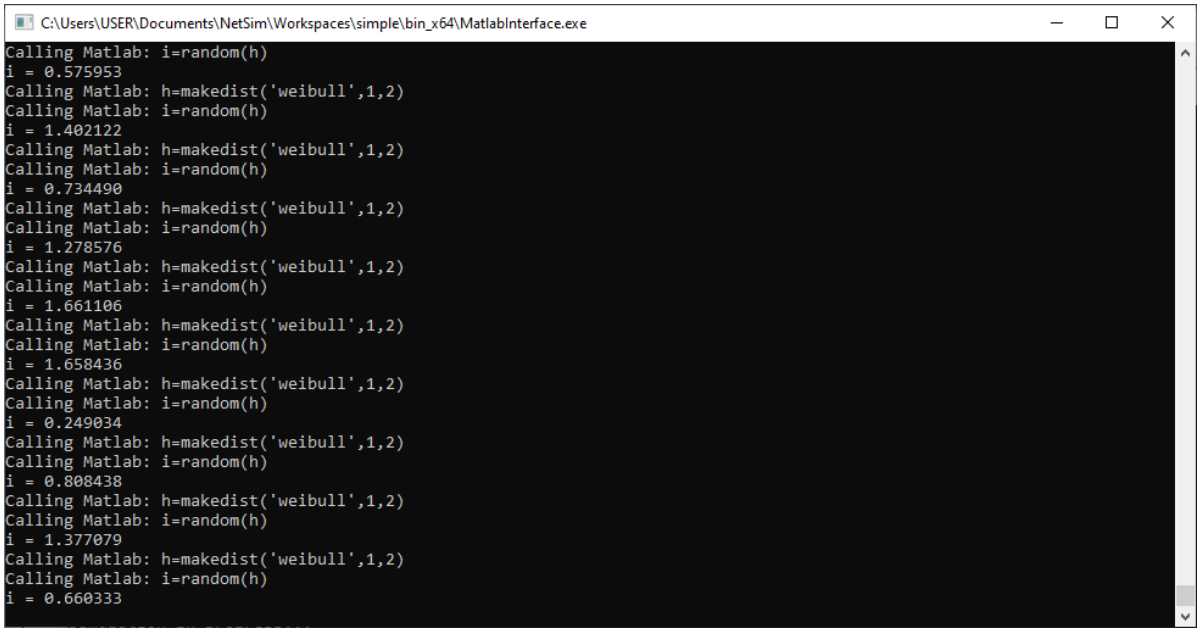


Figure 10-14: Runtime MATLAB interfacing window with I value

10.3.1.3 Debug and understand communication between NetSim and MATLAB

1. In the Solution Explorer under Medium project double click on Medium.c file and place a breakpoint inside the **matlab_calculate_fadingloss()** function before the return statement.

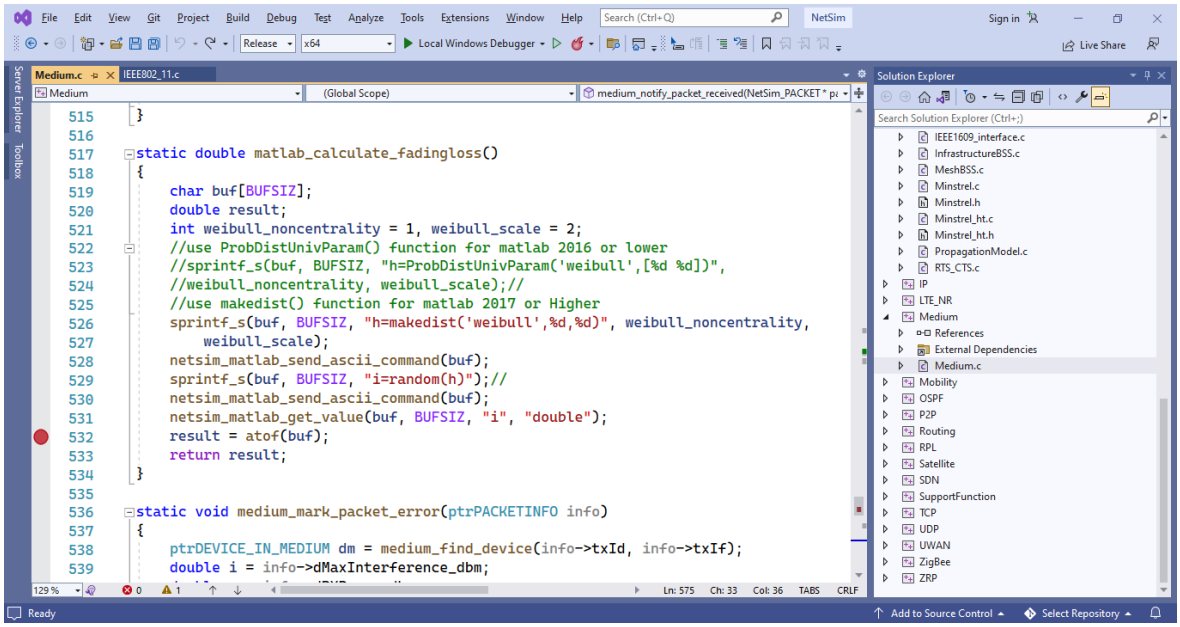


Figure 10-15: Placing a breakpoint inside matlab_calculate_fadingloss() function

2. Now run the NetSim Scenario. The simulation window stops for user interrupt.
3. In Visual studio, go to Debug =>Attach to Process.
4. From the list of Processes select NetSimCore.exe and click on Attach.

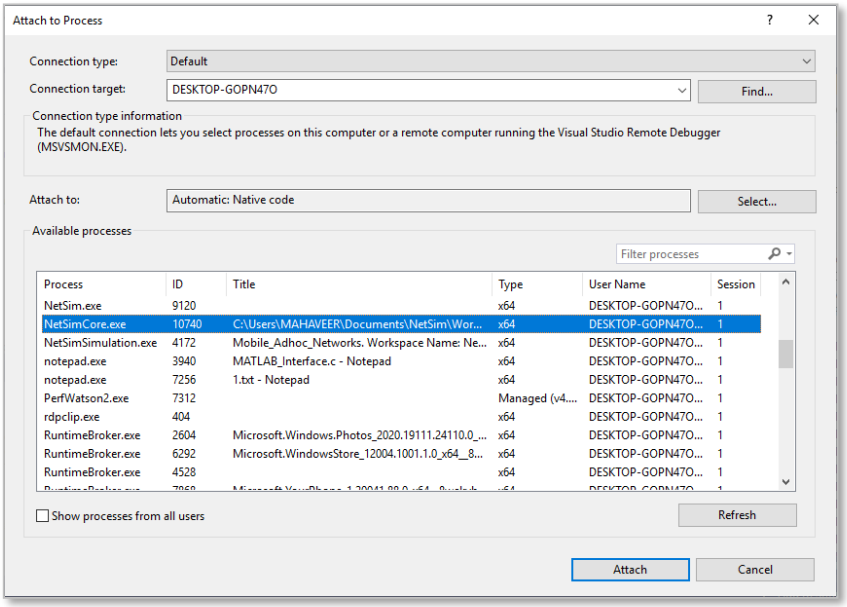


Figure 10-16: Select NetSimCore.exe in Attach to Process Window

5. Now go to the Simulation window and press Enter. Simulation will pause at Waiting for matlab interface to connect, go to Design window/GUI -> Options -> Open MATLAB interface ->click on OK
6. MATLAB Command Window will start and breakpoint in Visual Studio gets triggered.

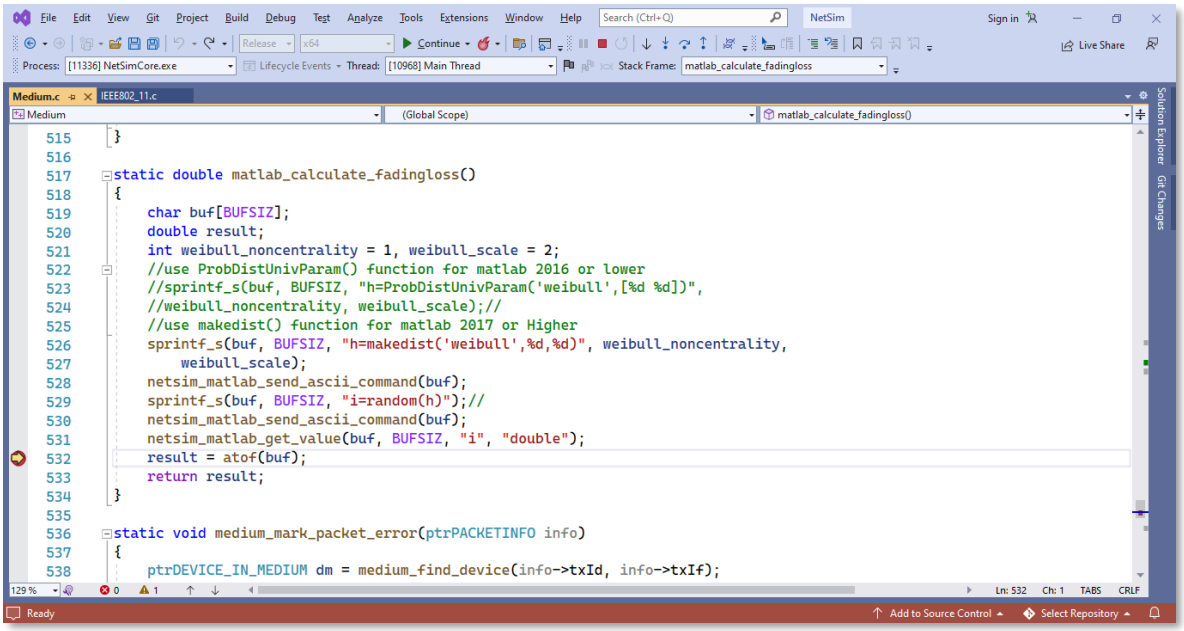


Figure 10-17: Breakpoint getting triggered in Visual Studio.

7. Now place another breakpoint after the line double f = matlab_calculate_fadingloss(); in the function medium_mark_packet_error() function and press continue or F5 key.

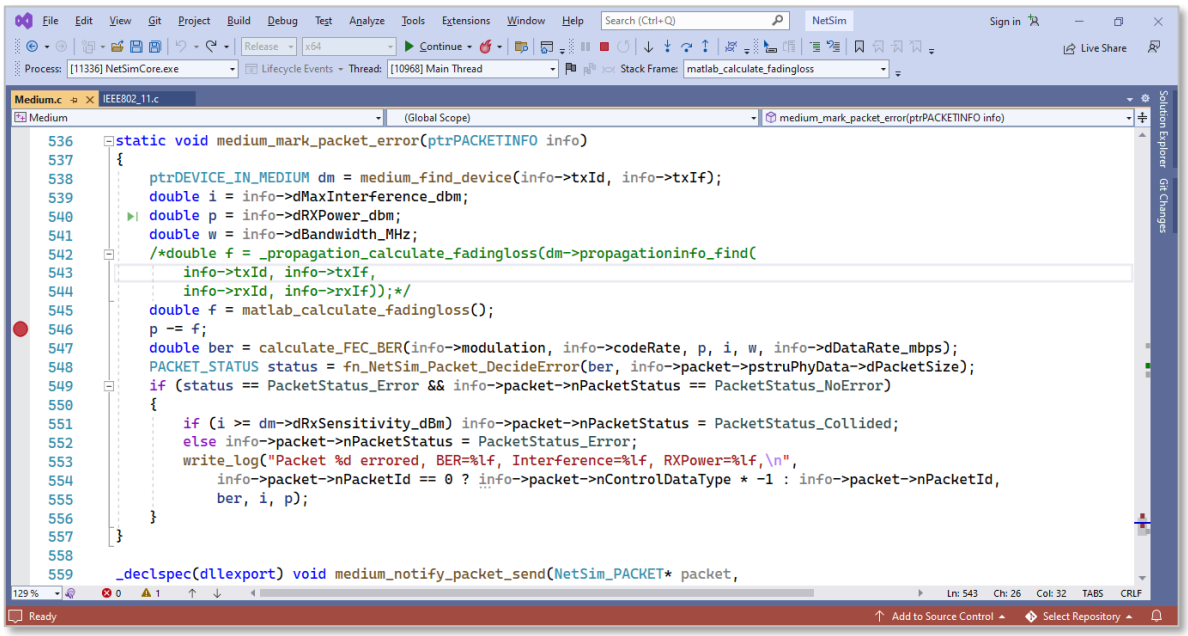


Figure 10-18: Adding Breakpoint under in the medium_mark_packet_error() function

8. Once the second breakpoint gets triggered, add the variable f which stores the fadingloss value, in Medium.c file, to watch. For this, right click on the variable f and select “Add Watch” option. You will find a watch window containing the variable name and its value in the bottom left corner.

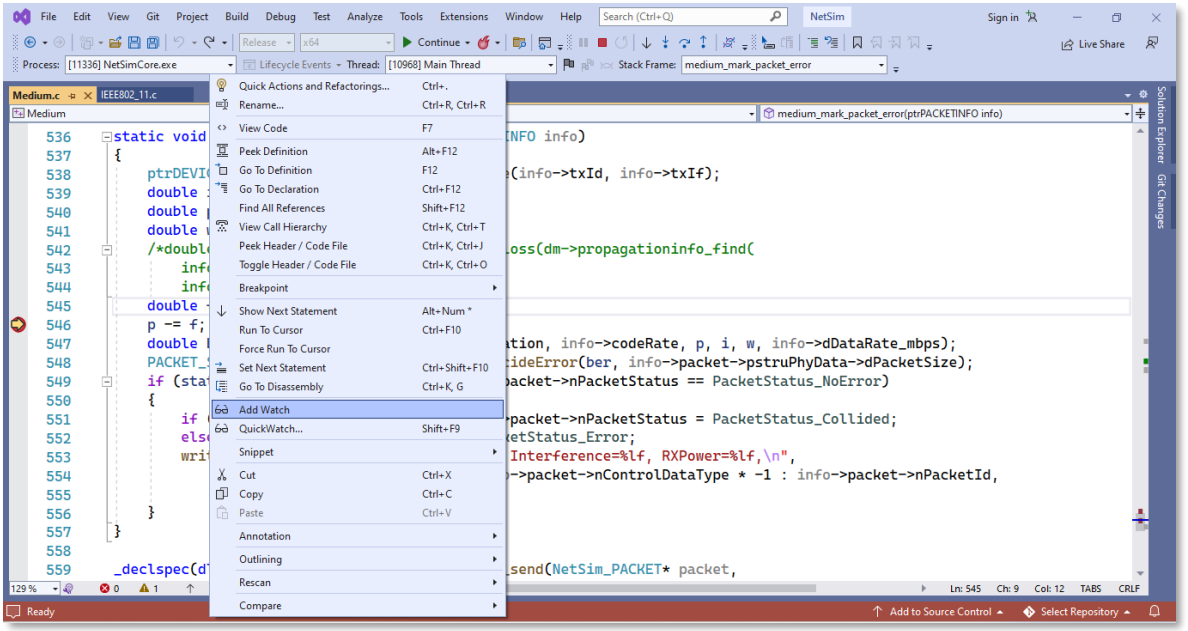


Figure 10-19: Right Clicking on the variable f and adding it to Watch

9. Now when debugging (say by pressing F5 each time) you will find that the watch window displays the value of f whenever the control reaches the recently set breakpoint. You will also find that the value of f shown in the Visual Studio Watch window and the value of i in the MATLAB Console window are similar.

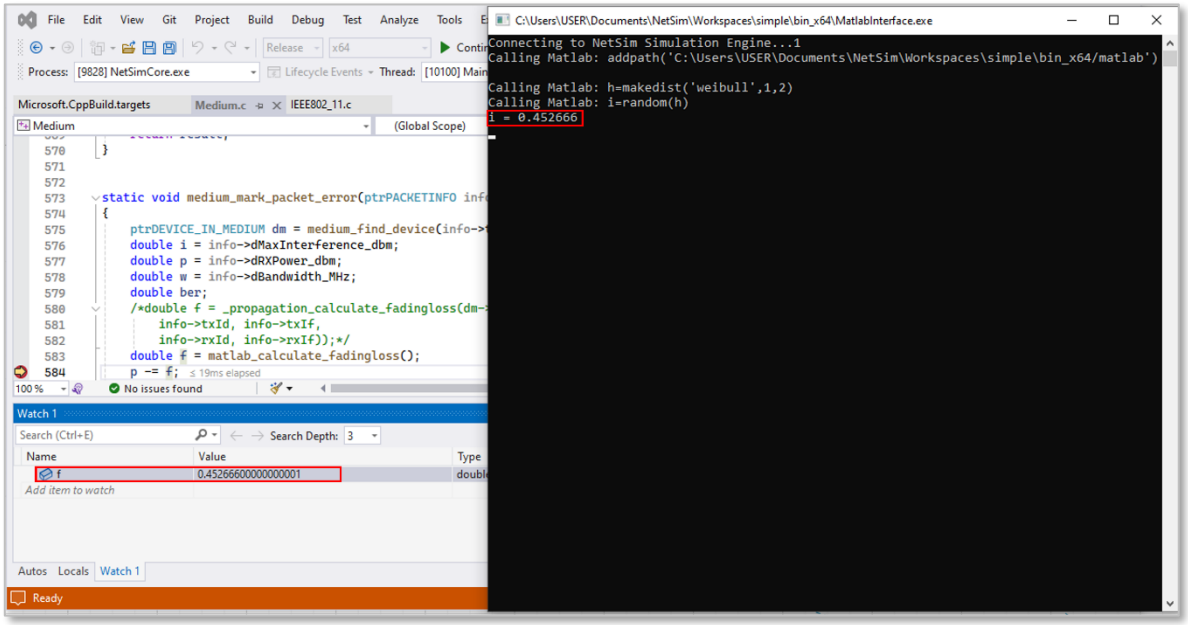


Figure 10-20: The Visual Studio Watch window variable f and the value of i in the MATLABInterface window are similar

10.3.1.4 Implement Weibull Distribution of MATLAB in NetSim using .m file

Procedure:

1. Create a file weibull_distribution.m file with the following code:

```
function out= weibull_distribution(noncentrality,scale)

        %use ProbDistUnivParam() function for matlab 2016

        %h=ProbDistUnivParam('weibull',[noncentrality,scale]);

        %use makedist() function for matlab 2017

        h=makedist('weibull',noncentrality,scale);

        i=random(h,1);

        out=i;

end
```

2. Create matlab folder inside the bin_x64 directory of the current workspace

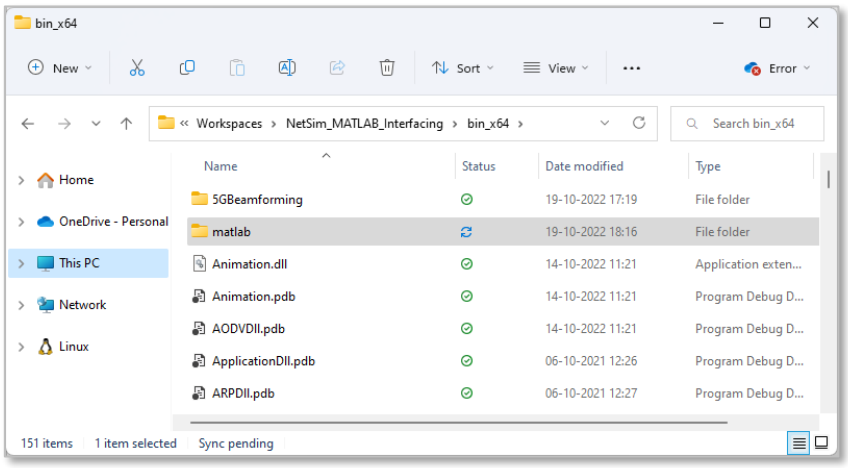


Figure 10-21: Bin folder of x64 of current workspace

3. Place the weibull_distribution.m file inside matlab folder.

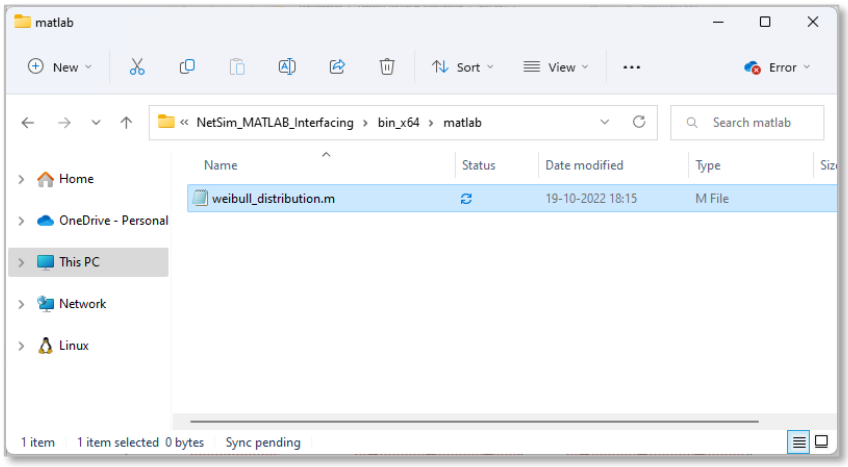


Figure 10-22: Place Weibull_distribution.m file inside matlab folder

4. Modify the function static double matlab_calculate_fadingloss() that we previously created in the Medium.c file to call the newly created Weibull_distribution.m

```
static double matlab_calculate_fadingloss()
{
        char buf[BUFSIZ];
        double result;
        int weibull_noncentrality = 1, weibull_scale = 2;
        //use ProbDistUnivParam() function for matlab 2016 or lower
        //sprintf_s(buf, BUFSIZ, "h=ProbDistUnivParam('weibull',[%d %d])",
        //weibull_noncentrality, weibull_scale);//
        //use makedist() function for matlab 2017 or Higher
        sprintf_s(buf, BUFSIZ, "k=weibull_distribution(%d,%d)", weibull_noncentrality,
                weibull_scale);
```

```
        netsim_matlab_send_ascii_command(buf);
        netsim_matlab_get_value(buf, BUFSIZ, "k", "double");
        netsim_matlab_send_ascii_command(buf);
        result = atof(buf);
        return result;
    }
```

5. Right click on the Medium source code project and select Rebuild.
6. Proceed to run simulation as explained in the previous sections.
7. You will find that once the Simulation is run MATLAB Command Window starts to execute and fadngloss value will be printed based on the call to weibull_distribution.m file.

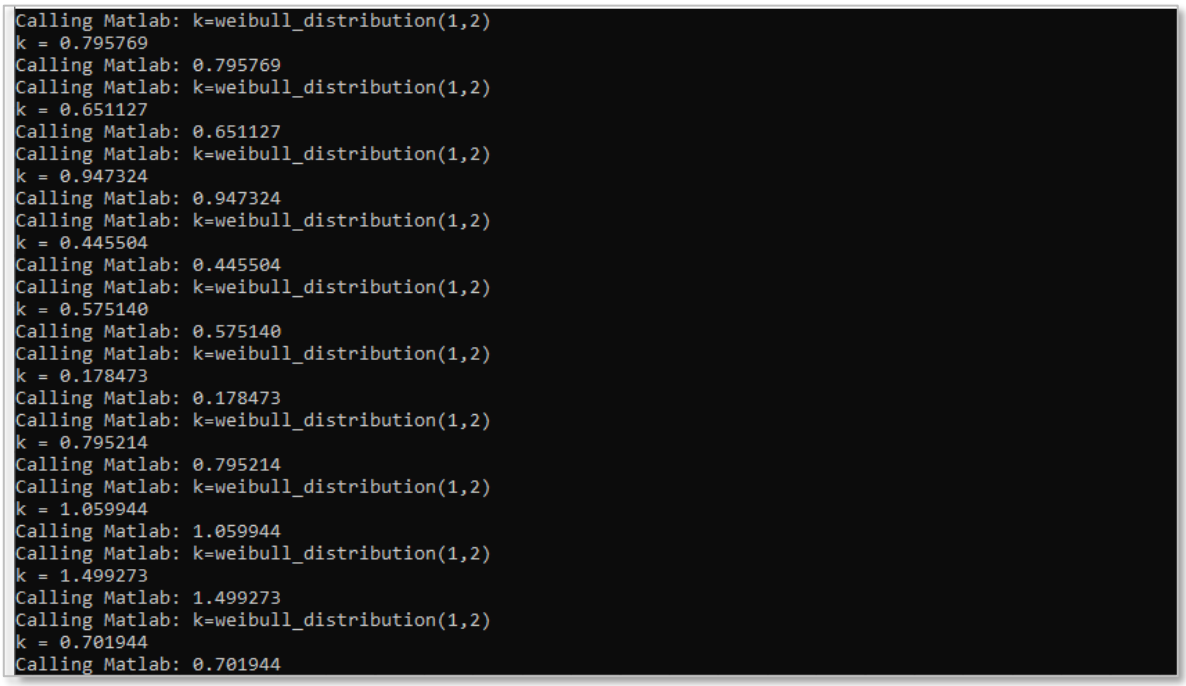


Figure 10-23: Runtime MATLAB interfacing window with k value

10.4 Interfacing Python with NetSim

NetSim provides run-time interfacing with Python so that users do not need to rewrite code in C for functionalities that already exist in Python. Instead, users can reuse existing Python libraries. A large amount of work related to **machine learning, artificial intelligence, data analytics, and mathematical modeling**—which can be valuable for networking research—can be directly implemented using existing Python packages such as **NumPy, SciPy, TensorFlow, scikit-learn**, etc.

Figure 10-24: Interfacing Python with NetSim

This interfacing feature can be used to either **replace existing functionalities** in NetSim or **add new features** that are better supported in Python. Any existing Python script, module, or function can be invoked from within the NetSim simulation environment.

In general, the following are done when a user interfaces NetSim to Python:

- Initialize a Python script or server process that communicates with NetSim.
- Send parameters from NetSim to Python.
- Receive results or computed outputs from Python.
- Handle various data types via structured message formats.
- Make calls to user-defined Python functions or modules.

Guidelines for Interfacing NetSim with Python

- Determine which parameters, variables or results from NetSim need to be passed to Python.
- Run the Python script that listens for NetSim connections.
- Use the provided NetSim socket API functions to package and send data.
- Make calls from relevant places of NetSim source code to
    - Pass parameters from NetSim to Python.
    - Data are serialized into byte streams, transferred via sockets, and deserialized in Python.
- The Python side automatically reconstructs received segments into native types.
- Users can directly operate on them, perform computations, and send results back.
- Identify and update the appropriate simulation variables in NetSim.

NetSim offers Socket interfacing to interact with Python during runtime.

This socket interface exposes simplified NetSim APIs for bidirectional communication with Python, making the integration process easier. Since socket-based communication does not require additional compiler or project settings, users can quickly connect NetSim simulations with external Python scripts for dynamic computation, visualization, or automation.

10.4.1 NetSim-Python Socket Interface

It provides several inbuilt APIs that can be called from the underlying protocol C source codes to interact with Python. Following are some of the APIs with syntax and description:

NetSim API's to interact with Python	Description
void* **Init_netsim_python_Interface()**	Initializes a socket and waits for a connection from python. This must be called once before any data exchange begins.
Message* **Init_Message**()	Allocates and initializes a new message structure used to send data segments from NetSim to Python. Call this before adding variables you want to send to Python.
void **add_variable_to_message**(void* message_v,….,VAR_END)	Adds one or more variables (int, double, string, arrays, etc.) into a message.
SEGMENT* **msg_get_value**(MESSAGE* m); void **msg_reset**(MESSAGE* m)	msg_get_value() helps retrieves the next segment in the message based on internal cursor. msg_reset() will reset the message internal cursor so you can read variables again from the beginning.
int **send_receive_message**(void* handle, void* message_v, void** outMessage)	Sends a message containing simulation data to Python, and receives the reply if Python sends any data.
uint32_t **get_variablecount**(void* message_v)	Returns the total number of variables (segments) currently present in the given message.
void **debug_print_message**(MESSAGE* m, int want_raw)	Prints the contents of a message (both variable names and values) to the console or log file.
void **free_message**(void* message_v)	Frees all dynamically allocated memory associated with a message, including its segments and data buffers. Must be called once a message is no longer needed to prevent memory leaks.
int **init_logging**(const char* path, int to_console)	Initializes the logging system, used to log all internal operations in a file. Can change the settings to log anything on display(terminal).
void **netsim_python_interface_start**(char* filepath)	Starts the Python Client program automatically, without having to start it manually. The path to the file, including the filename has to be passed as an argument.
LOG_INFO() **LOG_WARN()** **LOG_ERR()**	These functions help with logging, which is initiated by init_logging.

Table 10-3: Interface API functions (NetSim Side)

These functions are defined as part of the *NetSim_Python_Interface.h* file which is part of the Include directory of NetSim Source Codes. This header can be included in the C files where these APIs would be used. To know more details about how to use the functions, see the header file (*NetSim_Python_Interface.h*).

Similarly, APIs on the python side are defined. Following is some of the APIs with syntax and description:

Python API's to interact with Netsim	Description
connect(host,port)	Establishes a TCP connection to NetSim. Must be called before sending or receiving any messages.
Message()	Creates a new empty message object. Each message can contain multiple variables of different data types.
add_variable(message, var_type, value, length)	Adds a variable or array (int, double, bool, string, etc.) into a Message. Call this once per value or array you want to send.
get_value() and **reset_cursor**()	get_value() retrieves variables from a received message one segment at a time. reset_cursor() moves the message cursor back to the beginning so values can be read again. These are methods defined under Message class.
send_and_receive(socket, message, expect_reply)	Send a prepared message to NetSim and optionally receivethe reply message.
debug_print_message(message, want_raw, title)	Prints all variables inside a message for debugging. can also display raw bytes if required.
setup_logging(log_path, to_console, to_file)	Enables logging for all Python–NetSim communication. Logs can be written to console or file for debugging and troubleshooting.

Table 10-4: Interface API functions (Python Side)

These functions are defined as part of the *netsim_socket_client.pyd* file that can be found inside NetSim_Python_Interface_Client directory of NetSim Simulation source code folder. This will be required to call above functions. Add your Python code to the directory *NetSim_Python_Interface_Client* and import the functions that you want to call.

To know more details about how to use the functions see the sample python file (*sample.py*) that is available in the *NetSim_Python_Interface_Client* directory.

10.4.1.1 Prerequisites for Python Socket Interfacing

Before using the NetSim–Python interface, ensure that your Python environment is properly configured and compatible with NetSim's socket communication layer.

- Python **3.8 or higher** (recommended: 3.10+). Earlier versions are not supported due to differences in socket and multiprocessing behavior.
- The following standard or third-party modules are required:
  - Socket, Struct, logging, numpy
- Ensure that the chosen socket port (**5555**) is not blocked by a firewall or already in use.

10.5 Interfacing tail with NetSim

What is a tail command?

The **tail** command is a command-line utility for outputting the last part of files given to it via standard input. It writes results to standard output. By default, tail returns the last ten lines of each file that it is given. It may also be used to follow a file in real-time and watch as new lines are written to it.

PART 1:

Tail options

- The following command is used to log the file.
  tail " path_to_file " -f

where -f option is used to watch a file for changes with the tail command pass the -f option. This will show the last ten lines of a file and will update when new lines are added. This is commonly used to watch log files in real-time. As new lines are written to the log the console will update will new lines.

- If users do not want the last ten lines of the file, then use the following command.
  tail -n 0 " path_to_file " –f

where –n option is used to show the last n number of lines.

- If you want to open more than 1 file then use the following command
  tail –n 0 " path_to_file " " path_to_file " –f

PART 2:

Steps to log NetSim files using tail console.

Note: Before Executing below steps, User need to generate/Simulate a scenario of a network with specific network logs enabled to which tail command is being executed.

- Open command window from Install directory path of Netsim (<C:\Program Files\NetSim\Pro_v14_4\bin) which contains tail.exe
- Type the following command to open ospf_hello.log.txt file and press enter.

  tail -n 0 “<NetSim_IOPath>\log\ospf_hello.log” -f

  For example,

  tail -n 0 "C:\Users\ALICE\AppData\Local\Temp\NetSim\pro_14.4\log\log\ospf_hello.log" -f

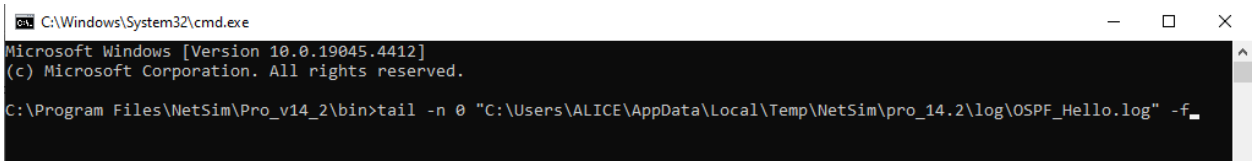


Figure 10-25: Enter the ospf_hello.log.txt file path is Command Prompt

- Open solution file and add the following line in fn_NetSim_OSPF_Init() function in ospf.c file present inside OSPF project

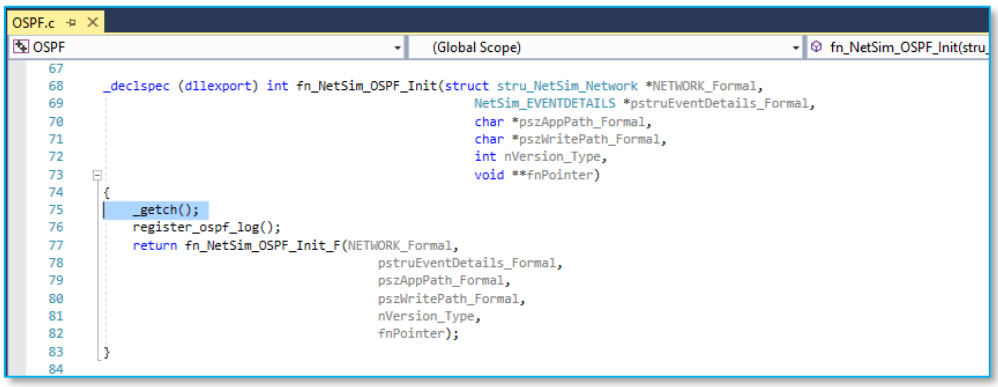


Figure 10-26: Add the following line in fn_NetSim_OSPF_Init() function in ospf.c file present inside OSPF project

- Rebuild the project.
- Upon rebuilding, **libOSPF.dll** will get created in the bin folder of NetSim's current workspace path <C:\Users\PC\Documents\NetSim\Workspaces\<Your default workspace>\bin_x64 > for 64-bit
- Create a scenario in NetSim as per the screenshot below and run simulation.

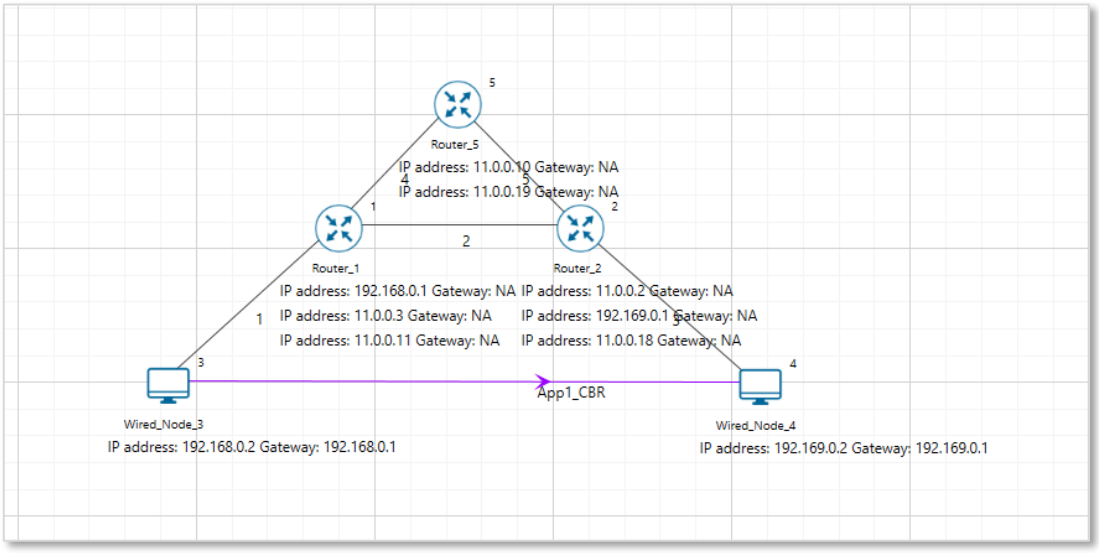


Figure 10-27: Network Topology

- In the console window user would get a warning message shown in the below screenshot Figure 10-28 (because of changed DLL) and then the simulation will pause for user input (because of _getch() added in the init function)

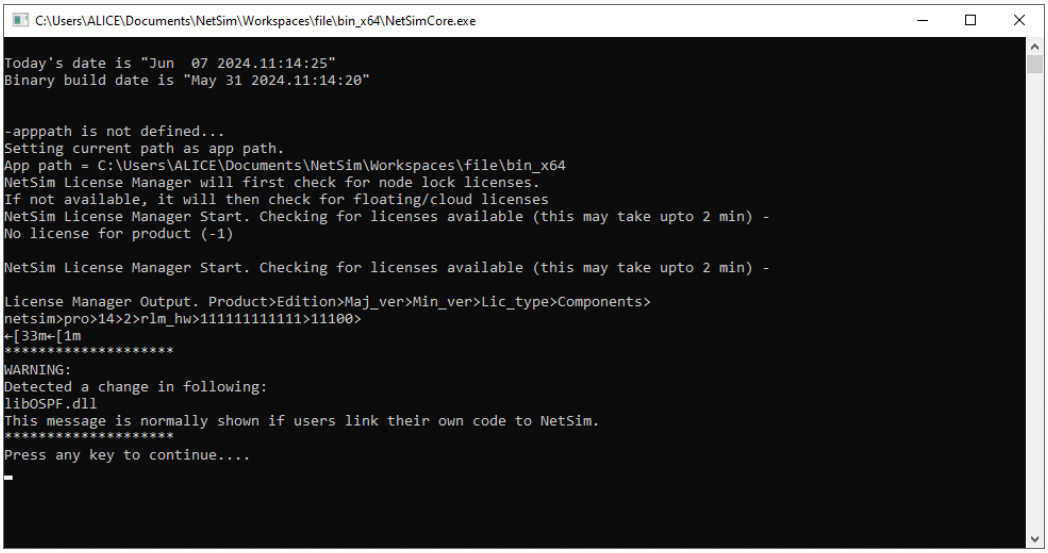


Figure 10-28: Modified Project DLL Warning Message in NetSim Console

- In Visual Studio, put break point inside all the functions in OSPF_Hello.c file present inside OSPF project.
- Go to “Debug->Attach to Process” in Visual studio and attach to NetSimCore.exe.
- Press enter in the command window. Then control goes to the project and stops at the break point in the source code as shown below Figure 10-29.

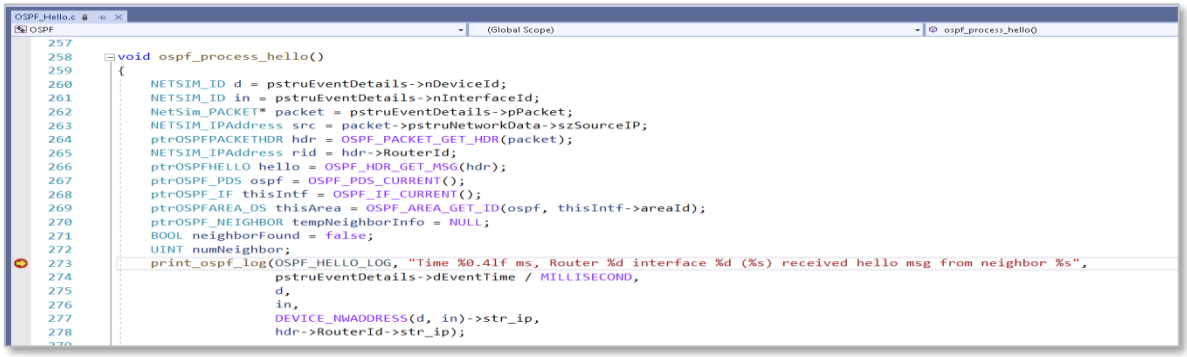


Figure 10-29: Control goes to the project and stops at the break point in the source code.

- Once after pressing enter in command window, check the tail console to watch the ospf_hello.log would look like the following screenshot Figure 10-30.
- Below Screenshot shows that scheduling time of hello interval for each Routers connected in Network.

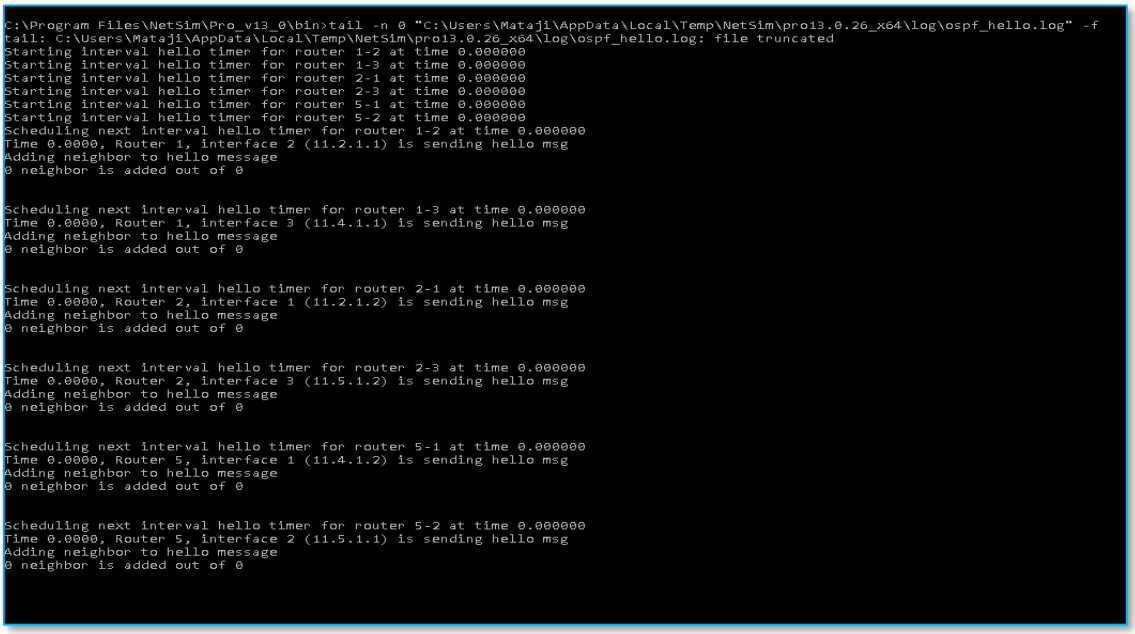


Figure 10-30: Scheduling hello interval for Routers.

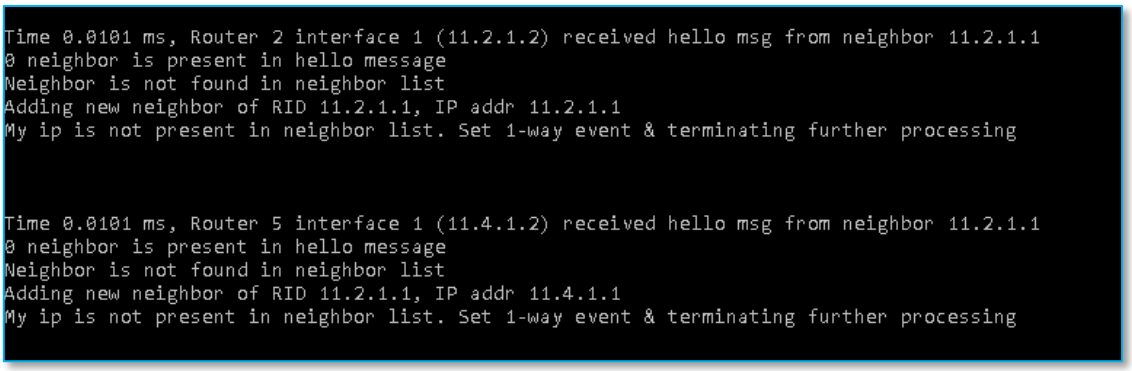


Figure 10-31: Adding new neighbor and terminating process if it is 1-way event

- Above Screenshot shows of adding neighbor to hello message, at Time 0.0101 ms, Router 5 interface 1 (11.4.1.2) received hello msg from neighbor 11.2.1.1.

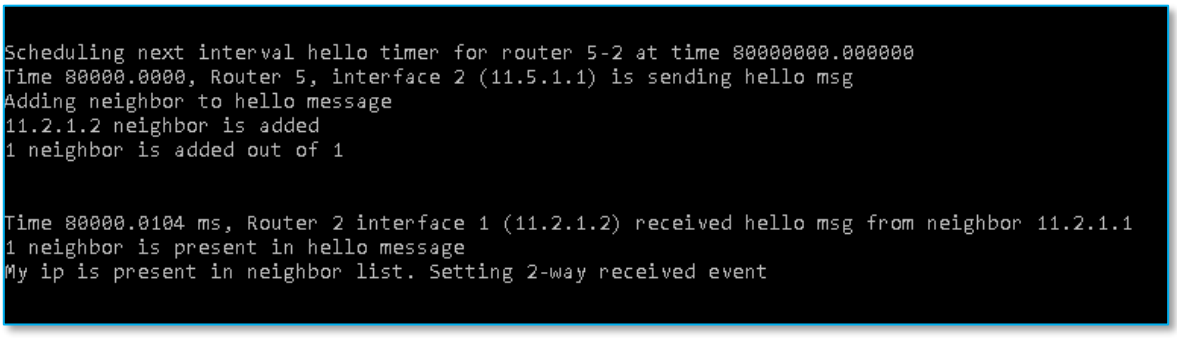


Figure 10-32: Adding Neighbor and Performing 2-way Event if neighbor is present in Router's list.

Above Screenshot indicates the Router 2's interface 1 (11.2.1.2) has received hello message from neighbor 11.2.1.1, if neighbor is present in Router's list 2-way event will be performed.

- Similarly, users can debug the code and observe how the OSPF tables get filled.

- Users can also open multiple files by using the command given in Part 1.

10.6 Adding Custom Performance Metrics

NetSim allows users to add additional metrics tables to the Simulation Results window in addition to the default set of tables that are available at the end of any simulation. This can be achieved by editing the source codes of the respective protocol.

General format to add custom metrics in Result window:

Every protocol has a main C file which contains a Metrics () function. For E.g., TCP project will have a TCP.c file, UDP will have an UDP.c file etc. In the following example we have added a new table as part of TCP protocol. TCP.c file contains fn_NetSim_TCP_Metrics() function where code related to custom metrics is added as shown below:

```
_declspec(dllexport) int fn_NetSim_TCP_Metrics(PMETRICSWRITER metricsWriter)
{
//CUSTOM METRICS
  //Set table name
  PMETRICSNODE table = init_metrics_node(MetricsNode_Table, "CUSTOM METRICS",
  NULL);
  //set table headers
  add_table_heading(table, "COLUMN_HEADER_1", true, 0);
  add_table_heading(table, "COLUMN_HEADER_2", false, 0);
  //Add table data
  add_table_row_formatted(false, table, "%s,%s,", "ROW_DATA1","ROW_DATA2");
  PMETRICSNODE menu = init_metrics_node(MetricsNode_Menu,"CUSTOM_METRICS",
  NULL);
  //Add table to menu
  add_node_to_menu(menu, table);
  //Write to Metrics file
  write_metrics_node(metricsWriter, WriterPosition_Current, NULL, menu);
  delete_metrics_node(menu);
  //CUSTOM METRICS
  return fn_NetSim_TCP_Metrics_F(metricsWriter);
  }
```

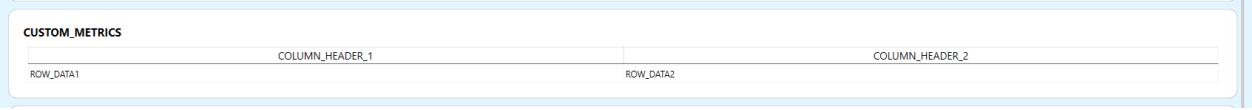


Figure 10-33: Added Custom metrics table to the Additional metrics window.

For illustration, an example regarding Wireless Sensor Network is provided. In this example, parameters such as Sensor Node Name, Residual Energy, State (On/Off) and turn–off time are tracked and added to a new table in the Simulation Results window.

Refer Section 9.1 on writing your own code, for more information.

After loading the source codes in Visual Studio, perform the following modifications:

Step 1: Copy the provided code at the top in 802_15_4.h file present in Zigbee project.

```
double NetSim_Off_Time[100]; //Supports upto Device ID 100. Array size can be increased for higher number of Devices/Device ID's
```

Step 2:

Add the header file in 802_15_4.c file.

```
#include "../BatteryModel/BatteryModel.h"
```

Step 3:

Copy the below code (in red colour) in 802_15_4.c file (inside fn_NetSim_Zigbee_Metrics() function)

```
/** This function write the metrics in metrics.txt */
_declspec(dllexport) int fn_NetSim_Zigbee_Metrics(PMETRICSWRITERmetricsWriter)
 {
//CUSTOM METRICS
    ptrIEEE802_15_4_PHY_VAR phy;
    ptrBATTERY battery;
    char radiostate[BUFSIZ];
    NETSIM_ID nDeviceCount = NETWORK->nDeviceCount;
    //Set table name
                PMETRICSNODE table = init_metrics_node(MetricsNode_Table, "NODE_FAILURE_METRICS", NULL);
    //set table headers
    add_table_heading(table, "Node Name", true, 0);
    add_table_heading(table, "Status(ON/OFF)", true, 0);
    add_table_heading(table, "Residual_Energy (mJ)", true, 0);
```

```
add_table_heading(table, "Time - Turned OFF (microseconds)", false, 0);
for (int i = 1; i <= nDeviceCount; i++)
{
    sprintf(radiostate, "ON");
    phy = WSN_PHY(i);
    if (strcmp(DEVICE(i)->type, "SENSOR"))
        continue;
    if (WSN_MAC(i)->nNodeStatus == 5 || phy->nRadioState==RX_OFF)
        sprintf(radiostate, "OFF");
    //Add table data
   if(phy->battery)
     add_table_row_formatted(false, table, "%s,%s,%.2lf,%.2lf,", DEVICE_NAME(i), radiostate,
battery_get_remaining_energy((ptrBATTERY)phy->battery), NetSim_Off_Time[i]);
}
  PMETRICSNODE menu = init_metrics_node(MetricsNode_Menu, "CUSTOM_METRICS",
NULL);
add_node_to_menu(menu, table);
write_metrics_node(metricsWriter, WriterPosition_Current, NULL, menu);
delete_metrics_node(menu);
//CUSTOM METRICS
return fn_NetSim_Zigbee_Metrics_F(metricsWriter);
}
```

Step 4:

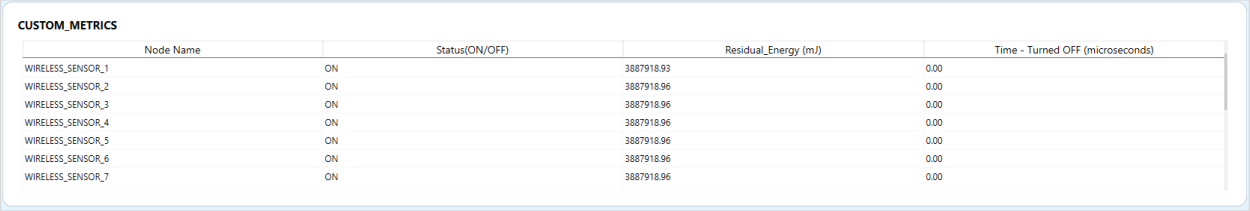
Copy the below code (in red colour) at the end of ChangeRadioState.c file.

```
if(isChange)
        {
                phy->nOldState = nOldState;
                phy->nRadioState = nNewState;
        }
```

```
else
{
        phy->nRadioState = RX_OFF;
        WSN_MAC(nDeviceId)->nNodeStatus = OFF;
        NetSim_Off_Time[nDeviceId] = ldEventTime;
}
return isChange;
}
```

Step 5:

Build DLL with the modified code and run a Wireless Sensor Network scenario by considering power source as battery model. After Simulation, user will notice a new Performance metrics named “Custom Metrics” is added. The newly added NODE_FAILURE_METRICS table is shown below Figure 10-34.



CUSTOM_METRICS

Node Name	Status(ON/OFF)	Residual_Energy (mJ)	Time - Turned OFF (microseconds)
WIRELESS_SENSOR_1	ON	3887918.93	0.00
WIRELESS_SENSOR_2	ON	3887918.96	0.00
WIRELESS_SENSOR_3	ON	3887918.96	0.00
WIRELESS_SENSOR_4	ON	3887918.96	0.00
WIRELESS_SENSOR_5	ON	3887918.96	0.00
WIRELESS_SENSOR_6	ON	3887918.96	0.00
WIRELESS_SENSOR_7	ON	3887918.96	0.00

Figure 10-34: Added Custom metrics table to the Additional metrics window.

10.7 Simulation Time and its relation to Real Time (Wall clock)

The notion of time in a simulation is not directly related to the actual time that it takes to run a simulation (as measured by a wall-clock or the computer's own clock), but is a variable maintained by the simulation program. NetSim uses a virtual clock which ticks virtual time. Virtual time starts from zero progresses as a positive real number.

Time is as a global parameter. All components of the network share the same time throughout the simulation independently of where they are physically located or how they are logically connected to the network. There are not differences among the local time of the communication network components.

This virtual time is referred to as simulation time to clearly distinguish it from real (wall-clock) time. NetSim is a discrete event simulator (DES), and in any DES, the progression of the model over simulation time is decomposed into individual events where change can take place. The flow of

time is only between events and is not continuous. Therefore, simulation time is not allowed to progress during an event, but only between events. In fact, the simulation time is always equal to the time at which the current event occurs. Thus, simulation time can be viewed as a variable that "jumps" to track the time specified for each new event.

The answer to the question "Will NetSim run for 10 seconds if Simulation time is set to 10 sec?" is, the simulation may take more than 10 seconds (Wall clock) if the network scenario is very large and heavy traffic load. It may take a much shorter time (wall clock) for small networks with low traffic loads.

Note that when running in "Emulation mode" simulation time and wall clock will be exactly synchronized since it involves the transfer of real packets across the virtual network in NetSim.

In NetSim, the current simulation time can be got using -pstruEventDetails->dEventTime

10.8 Environment Variables in NetSim

1. NETSIM_PACKET_FILTER = <filter_string> //used by NetSim developers to debug. Emulator code to passes filter string to windivert. See windivert doc for more information.
2. NETSIM_EMULATOR_LOG = <log_file_path> // Used by Real time sync function to log get event and add event. Used by NetSim developers to debug.
3. NETSIM_EMULATOR = 1 // Set by application DLL or user to notify NetSim internal modules to run in emulation mode
4. NETSIM_CUSTOM_EMULATOR = 1 // To notify NetworkStack to not load emulation DLL and to only do time sync.
5. NETSIM_SIM_AREA_X = <int> // Area used by Mobility functions for movement of device. Set by config file parser or user.
6. NETSIM_SIM_AREA_Y = <int> // Same as above
7. NETSIM_ERROR_MODE = 1 // if set then windows won't popup gui screen for error reporting on exception.
8. NETSIM_BREAK = <int> // Event id at which simulation will break and wait for user input.
9. Equivalent to -d command in CLI mode.
10. NETSIM_IO_PATH = <path> // IO path of NetSim from where it will read Config file and write output file. Equivalent to -IOPATH command in CLI mode.
11. NETSIM_MAP = 1 // Set by Networkstack to inform other modules that simulation is running per map view.
12. NETSIM_ADVANCE_METRIC // If set, NetSim provides a set of extra metrics.
13. In application metrics, you can see duplicate packets received.
14. NETSIM_CONFIG_NAME = <FILE NAME> // Config file name. This file must present in IOPath. If not set default value is Configuration.netsim

15. NETSIM_NEG_ID = 1 // If set, then control packets will have negative id.
16. NETSIM_PACKET_DBG = 1 // If set, then Simulation engine will log the packet creation and freeing
17. NETSIM_MEMCHECK = 1 // If set, then simulation will enable memory check.
18. NETSIM_MEMCHECK_1 = x // Lower event id
19. NETSIM_MEMCHECK_2 = x // Upper event id

10.9 Best practices for running large scale simulations

As we scale simulations, the number of events processed and the memory consumed increase. Simulation scale can be defined in terms of:

1. Number of Nodes
   - End nodes
   - Intermediate devices
2. Total traffic in the network
   - Number of traffic sources
   - Average generation rate per source
3. Simulation time

The simulators performance is additionally affected by:

- Protocols running
- Network Parameters such as Topology, Mobility, Wireless Channel etc.
- Enabling/Disabling - Plots, Traces and Logs
- External Interfacing – MATLAB, Wireshark, SUMO

NetSim GUI limitation on total number of Nodes is as follows:

- NetSim Academic:
  - Intermediate devices: 20
  - End nodes: 100
- NetSim Standard:
  - Intermediate devices: 100
  - End nodes: 500
- NetSim Professional :
  - Intermediate devices: 500
  - End nodes: 2500

Recommended best practices for running large scale simulations are:

- Run 64-bit Build of NetSim
- Use the latest Windows 10 Build.
- Use a system running a high-end processor with minimum 32 GB RAM
- Disable plots, traces and logs to speed up the simulation.
- NetSim writes one packet trace and one event trace file per simulation. If users wish to open this file in MS-Excel, please note Excel's limit of 1 Million rows.
- Packet trace and Event trace can be disabled to speed up the simulation.
- Running simulations via CLI mode will save memory.

10.10 Batch experimentation and automated simulations

NetSim Batch Automation allows users to execute a series of simulations without manual intervention. Consider a requirement, where a user wishes to create and simulate hundreds of network scenarios and store and analyse the performance metrics of each simulation run. It is impossible to do this manually using the GUI. This requirement can be met using NetSim Batch Automation script which runs NetSim via CLI.

A related requirement of advanced simulation users is a multi-parameter sweep. When you sweep one or more parameters, you change their values between simulation runs, and compare and analyze the performance metrics from each run.

The documentation and codes for batch experimentation script is available TETCOS - https://www.tetcos.com/file-exchange.html

11 NetSim Emulator

Note: Emulator will be featured in NetSim only if license for Emulator Add-on is available

11.1 Introduction

A network simulator mimics the behavior of networks but cannot connect to real networks. NetSim Emulator enables users to connect NetSim simulator to real hardware and interact with live applications.

11.1.1 Simulating and Analyzing Emulation Examples

To simulate the different types of Emulations Examples such as PING (both one-way and two-way communications), Video (one-way communication), File transfer using FileZilla, Skype etc.

1. Refer to the Emulation Technology Library document, which explains the following:
   i. Introduction to Emulation.
   ii. How to set up and configure Emulation Server in NetSim
   iii. NetSim Emulation Features with added examples
   iv. Latest FAQs
2. To access the Emulation Technology Library document,
   i. You can access from the Technology Libraries link present under Documentation in NetSim Homescreen
   ii. From the Help Menu inside the design window, choose Technology Libraries Manuals → Emulation.

12 Troubleshooting in NetSim

This section discusses some common issues and solutions:

12.1 CLI mode

While running NetSim via CLI for the scenarios described in the Configuration file, you may bump into a few problems. While running NetSim via CLI, try to ensure that there are no errors in the Configuration.netsim file. The ConfigLog.txt file written to the windows temp path would show errors, if any, found by NetSim's config parser.

12.2 Warnings when running CLI mode

12.2.1 I/O warning

Reason: While typing the CLI command if you enter the wrong I/O Path, or if there is no Configuration.netsim file then the following error is thrown

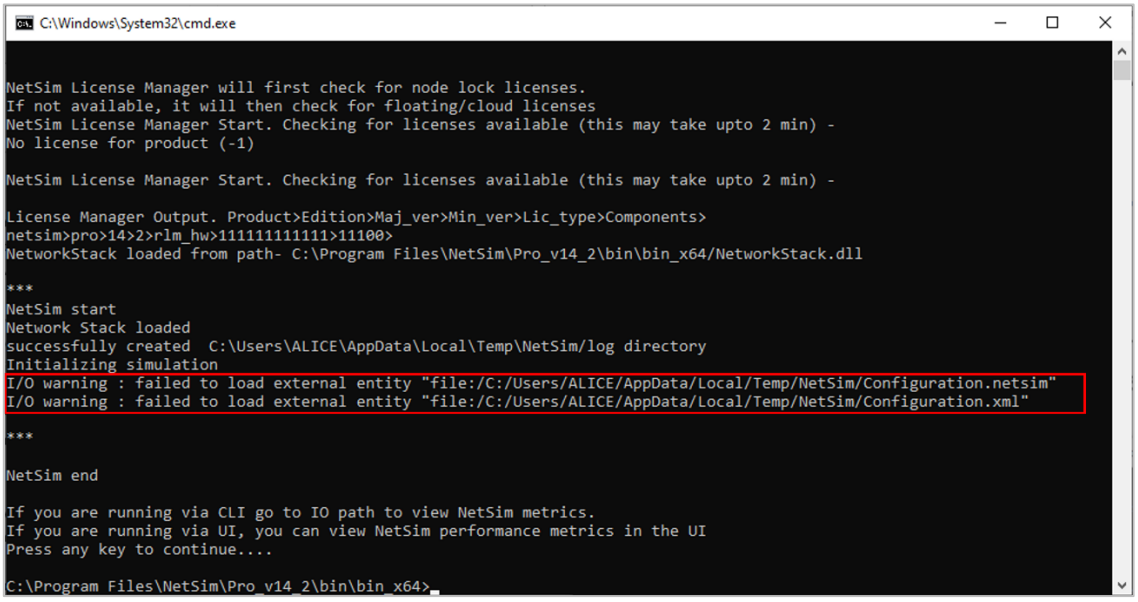


Figure 12-1: I/O warning displayed in CLI mode

Solution*:* Check and correct the I/O path.

12.2.2 Error in getting License displayed

Reason: Unable to communicate with the license server

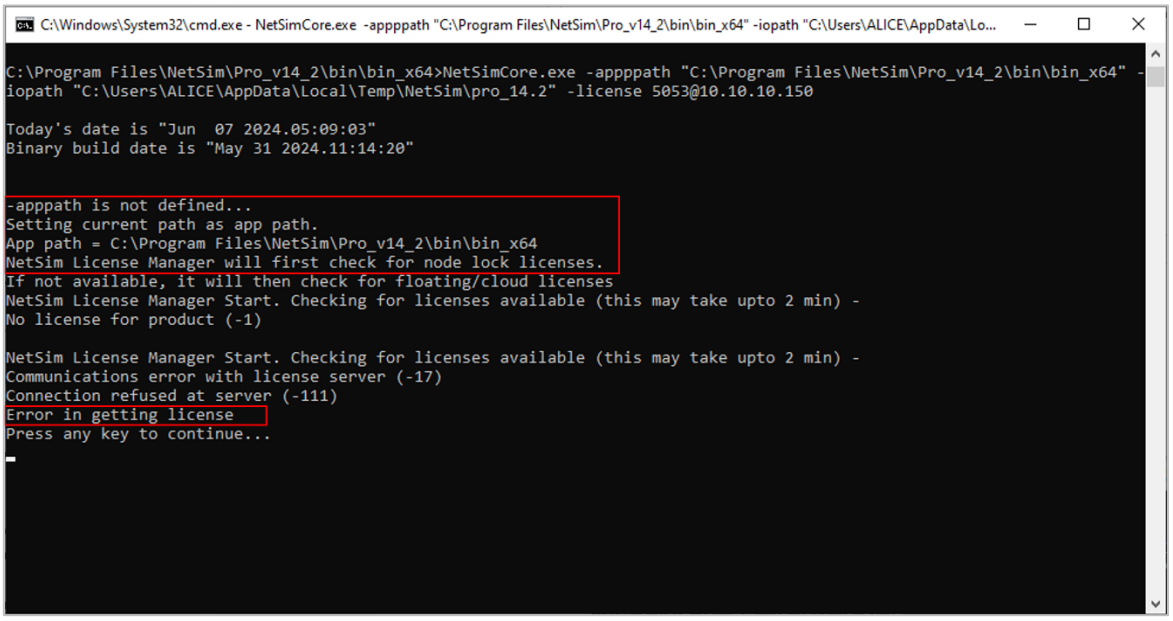


Figure 12-2: Error in getting license

Solution: In this example, license server IP address is 10.10.10.122 but it is entered as 10.10.10.150. This is a case where the server IP address is wrong. The same error message is shown for wrong port number, wrong tag name like–apppath, -iopath, -license …, etc. Therefore, this message is also shown if –appppath is typed instead of –apppath. Users are advised to check the command line arguments carefully.

12.2.3 Unable to load license config DLL(126)

Reason: Apppath and I/O path have white spaces

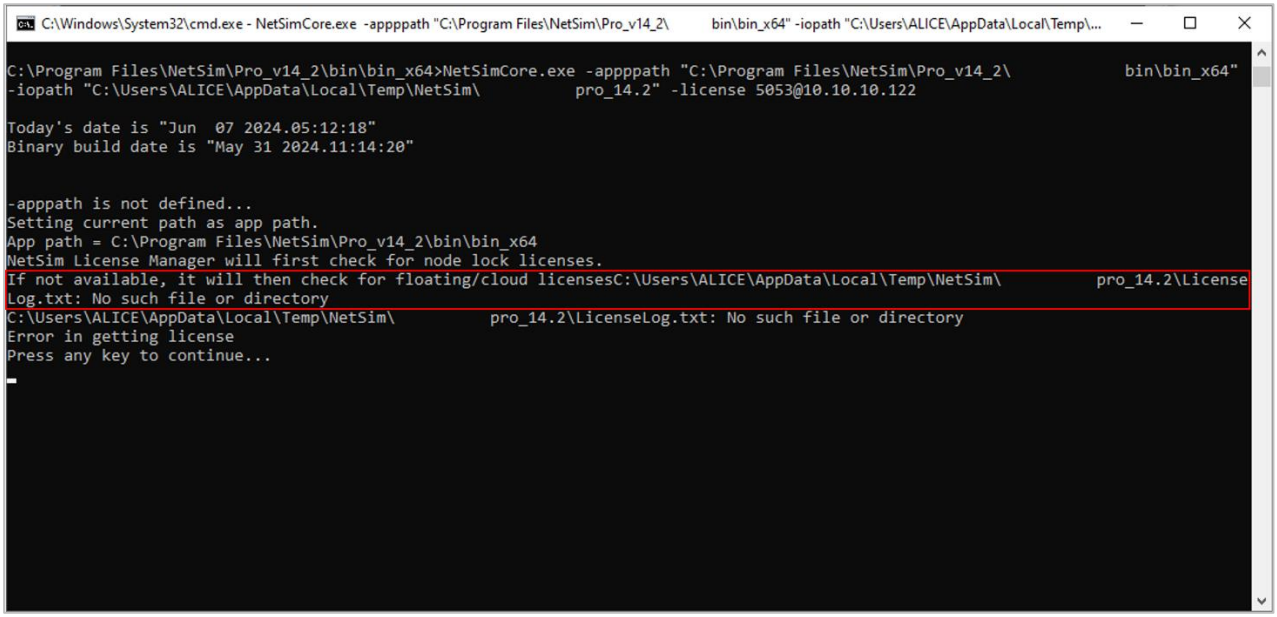


Figure 12-3: Unable to load license config DLL

***Solution**:* If the folder name contains white space, then mention the folder path within double quotes while specifying the folder name in the command prompt. For example, if app path contains white space, then the app path must be mentioned within double quotes in the command prompt.

12.2.4 "License is NULL" error in CLI mode

Simulation does not commence. "License is NULL" is displayed in the command prompt.

Example:

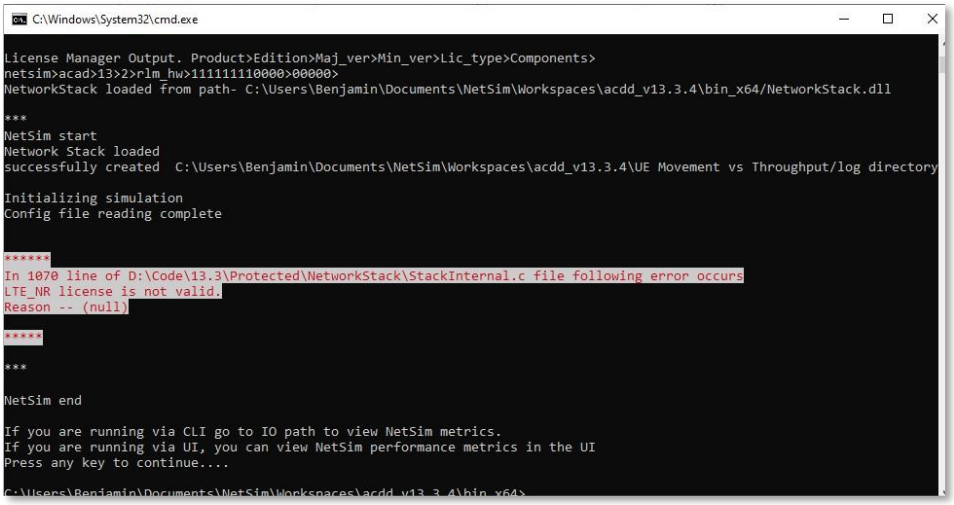


Figure 12-4: "License is NULL" error in CLI mode

Solution: This indicates unavailability of licenses for the network (component) being simulated.

12.3 Configuration.netsim

12.3.1 Invalid attribute in configuration file attributes

Specific attributes in the Configuration file are highlighted with zigzag lines.

***Reason**:* If invalid input is given in the Configuration file, then the corresponding attribute is highlighted as blue lines as shown in the figure given below Figure 12-5.

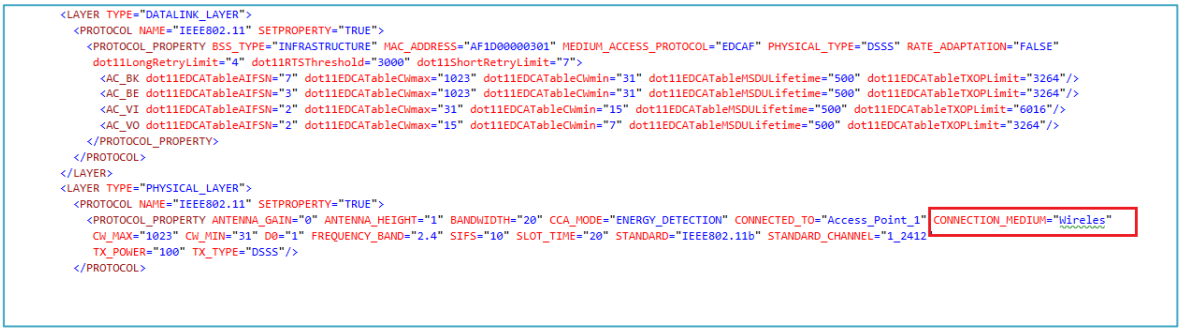


Figure 12-5: Invalid attribute in configuration file

Solution: To resolve this issue mouse over the corresponding attribute, in order to get the tool tip that furnishes the details about the valid input for that attribute.

Note: If the schema file and the configuration file are not present in the same folder, the zigzag lines won't appear. So, place the Configuration file and Schema File in the same location or change the path of schema file in the configuration file.

12.3.2 Error in tags in configuration file attributes

Simulation does not commence, and error is displayed at the command prompt. Also, red lines appearing at the tag specifying the Layer in the Configuration file

Reason*:* This issue arises mainly when the closing tag is not specified correctly for a particular layer in the Configuration file.

Example: If the closing tag is not specified for the Data link Layer, then the zigzag lines appear at the starting tags of Data link Layer and the Network Layer.

```
    </PROTOCOL>
  </LAYER>
  <LAYER TYPE="DATALINK_LAYER">
    <PROTOCOL NAME="IEEE802.11" SETPROPERTY="TRUE">
      <PROTOCOL_PROPERTY BSS_TYPE="INFRASTRUCTURE" MAC_ADDRESS="AF1D00000301" MEDIUM_ACCESS_PROTOCOL="EDCAF" PHYSICAL_TYPE="DSSS" RATE_ADAPTATION="FALSE"
       dot11LongRetryLimit="4" dot11RTSThreshold="3000" dot11ShortRetryLimit="7">
        <AC_BK dot11EDCATableAIFSN="7" dot11EDCATableCWmax="1023" dot11EDCATableCWmin="31" dot11EDCATableMSDULifetime="500" dot11EDCATableTXOPLimit="3264"/>
        <AC_BE dot11EDCATableAIFSN="3" dot11EDCATableCWmax="1023" dot11EDCATableCWmin="31" dot11EDCATableMSDULifetime="500" dot11EDCATableTXOPLimit="3264"/>
        <AC_VI dot11EDCATableAIFSN="2" dot11EDCATableCWmax="31" dot11EDCATableCWmin="15" dot11EDCATableMSDULifetime="500" dot11EDCATableTXOPLimit="6016"/>
        <AC_VO dot11EDCATableAIFSN="2" dot11EDCATableCWmax="15" dot11EDCATableCWmin="7" dot11EDCATableMSDULifetime="500" dot11EDCATableTXOPLimit="3264"/>
      </PROTOCOL_PROPERTY>
    </PROTOCOL>

  <LAYER TYPE="PHYSICAL_LAYER">
    <PROTOCOL NAME="IEEE802.11" SETPROPERTY="TRUE">
      <PROTOCOL_PROPERTY ANTENNA_GAIN="0" ANTENNA_HEIGHT="1" BANDWIDTH="20" CCA_MODE="ENERGY_DETECTION" CONNECTED_TO="Access_Point_1" CONNECTION_MEDIUM="WIRELESS"
       CW_MAX="1023" CW_MIN="31" D0="1" FREQUENCY_BAND="2.4" SIFS="10" SLOT_TIME="20" STANDARD="IEEE802.11b" STANDARD_CHANNEL="1_2412"
       TX_POWER="100" TX_TYPE="DSSS"/>
    </PROTOCOL>
  </LAYER>
</INTERFACE>
<LAYER TYPE="APPLICATION_LAYER"/>
<LAYER TYPE="TRANSPORT_LAYER">
```

Figure 12-6: Error in tags in configuration file

When NetSim is made to run through CLI, then the following error gets displayed in the command prompt.

```
C:\Windows\System32\cmd.exe
NetSim License Manager Start. Checking for licenses available (this may take upto 2 min) -
No license for product (-1)

NetSim License Manager Start. Checking for licenses available (this may take upto 2 min) -

License Manager Output. Product>Edition>Maj_ver>Min_ver>Lic_type>Components>
netsim>pro>14>2>rlm_hw>111111111111>11100>
NetworkStack loaded from path- C:\Program Files\NetSim\Pro_v14_2\bin\bin_x64/NetworkStack.dll

***
NetSim start
Network Stack loaded
Error in creating C:\Users\ALICE\AppData\Local\Temp\NetSim\pro_14.2/log directory. Error number 17
Initializing simulation
file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.netsim:99: parser error : Opening and ending tag mismatch: LAYER line 87 and I
NTERFACE
        </INTERFACE>
                    ^
file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.netsim:121: parser error : Opening and ending tag mismatch: INTERFACE line 78
and DEVICE
      </DEVICE>
                ^
file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.netsim:842: parser error : expected '>'
    </DEVICE_CONFIGURATION>
                ^
file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.netsim:873: parser error : Opening and ending tag mismatch: DEVICE_CONFIGURATI
ON line 25 and NETWORK_CONFIGURATION
  </NETWORK_CONFIGURATION>
                          ^
file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.netsim:897: parser error : Opening and ending tag mismatch: NETWORK_CONFIGURAT
ION line 24 and TETCOS_NETSIM
file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.netsim:897: parser error : Premature end of data in tag TETCOS_NETSIM line 2
I/O warning : failed to load external entity "file:/C:/Users/ALICE/AppData/Local/Temp/NetSim/pro_14.2/Configuration.xml"

***

NetSim end

If you are running via CLI go to IO path to view NetSim metrics.
If you are running via UI, you can view NetSim performance metrics in the UI
Press any key to continue....
```

Figure 12-7: NetSim Run through CLI and following error displayed in the command prompt

Solution: The bug can be fixed by setting the closing tag correctly in the Configuration file

12.3.3 Error lines in configuration.xsd in the Configuration file

Blue lines appear at configuration.xsd in the Configuration file.

Reason: This issue arises when the schema and the configuration file are not in the same folder.

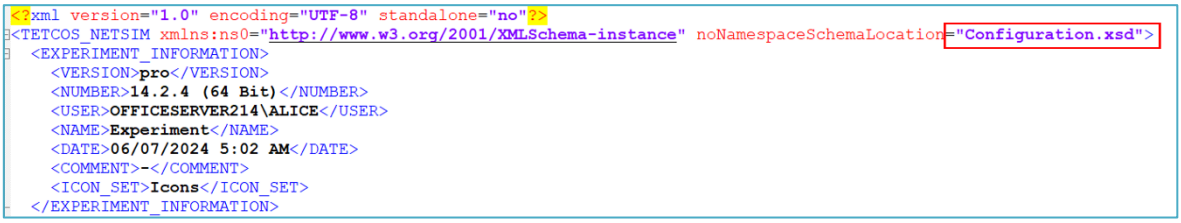


Figure 12-8: Error lines in configuration.xsd in the Configuration file

Solution: The bug can be fixed by placing the Configuration file and schema in the same folder.

12.4 Simulation terminates and “NetSim Backend has stopped working” displayed

Simulation terminates and exhibits unpredictable behavior. An error message stating, “NetSim Crashed or Terminated” is thrown.

Example:

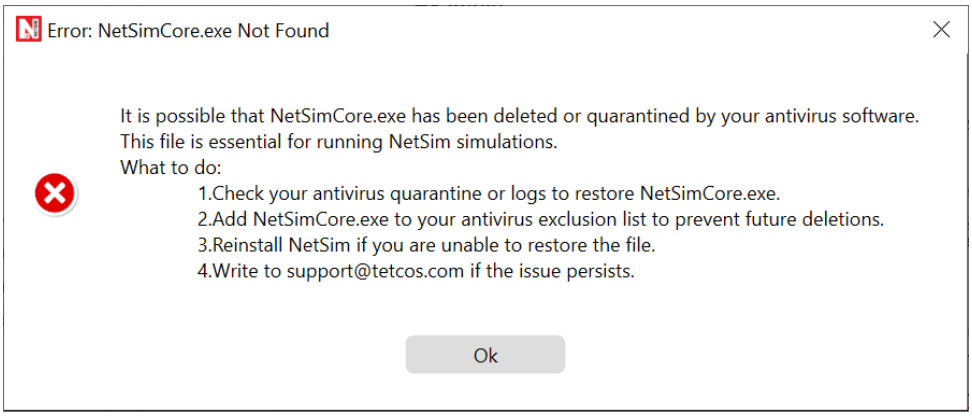


Figure 12-9: NetSim Crashed or Terminated window.

This problem arises if there is any flaw in the Configuration.netsim or in the DLL.

Solution: Check whether the desired scenario has been configured properly in the Configuration.netsim.

12.5 Licensing

12.5.1 No License for product (-1) error

NetSim dongle is running in the server system. When running the NetSim in the Client system showing “No License for product (-1)” error.

Possible Reasons:

1. Firewall in the client system is blocking the Network traffic.
2. No network connection between Client and Server.
3. License Server is not running in the Server system.

Solution:

1. The installed firewall may block traffic at 5053 port used for licensing. So, either the user can stop the firewall, or may configure it to allow port 5053.
2. Contact the Network-in-charge and check if the Server system can be pinged from client.
3. Check whether License Server is running in the Server system or not.

12.6 Troubleshooting VANET simulations that interface with SUMO

12.6.1 Guide for Sumo

- Link for the Sumo Website –

  http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/ for help related to Sumo.

- In case sumo Configuration files do not open, Right click on any Sumo Configuration file, go to properties→open with→sumo.
- While Running NetSim Vanet Simulation – If any message pops up as **“SUMO_HOME” Not found**→ Go to My computer → System Properties → Advanced system settings → Environment Variables. Add an Environment variable as “SUMO_HOME”.
- Sumo Configuration File must contain the paths of the Vehicle routes and Networks file.
- Set the exact End Time for Sumo Simulation in Sumo Configuration File.

12.6.2 Guide for Python

- Any Python 3.11.1 version Installer would work fine for running simulations.
- If you have installed python by an external Installer, make sure the Python Path is set. It would be set automatically by python installer that comes with NetSim.

- In case **“Pywin 32**” is not getting installed, or during simulation, error occurs as “**win32 modules not found**” try the code below (Run it as a python Code).

```
import sys
from _winreg import *
# tweak as necessary
version = sys.version[:3]
installpath = sys.prefix
regpath = "SOFTWARE\\Python\\Pythoncore\\%s\\" % (version)
installkey = "InstallPath"
pythonkey = "PythonPath"
pythonpath = "%s;%s\\Lib\\;%s\\DLLs\\" % (
    installpath, installpath, installpath
)
def RegisterPy():
    try:
        reg = OpenKey(HKEY_CURRENT_USER, regpath)
    except EnvironmentError as e:
        try:
            reg = CreateKey(HKEY_CURRENT_USER, regpath)
            SetValue(reg, installkey, REG_SZ, installpath)
            SetValue(reg, pythonkey, REG_SZ, pythonpath)
            CloseKey(reg)
        except:
            print "*** Unable to register!"
            return
        print "--- Python", version, "is now registered!"
        return
    if (QueryValue(reg, installkey) == installpath and
        QueryValue(reg, pythonkey) == pythonpath):
        CloseKey(reg)
        print "=== Python", version, "is already registered!"
        return
    CloseKey(reg)
    print "*** Unable to register!"
    print "*** You probably have another Python installation!"
if __name__ == "__main__":
    RegisterPy()
```

12.6.3 VANET Simulation

i. Changing Vehicle (Node) Names, Moving or deleting vehicles etc. are disabled in Vanets Simulation.
ii. On running simulation, Backend calls Python file.
iii. NetSim protocol engine waits for the Pipes connection to be established.

12.6.4 Python

- SUMO_HOME Environment variable is checked. If Environment variable is not present, Error is displayed as “key interrupt error” in SUMO_HOME.
- Python File waits for Pipes connection. (“waiting for pipes to connect”).
- It reads initial data as GUI enable/disable from simulation engine.
- “Checking sumo” is printed. If the environment variable SUMO_HOME points to wrong directory, error is displayed.
- Sumo Simulation is started where Sumo Binary is checked (To check Sumo.exe or Sumo GUI are working in the system or not). Then a TCP connection is made
- A while loop runs – It follows the following procedure.

  i. Send Garbage value to Backend to clear pipe buffers (pipes).
  ii. Read Vehicle name from NetSim (pipes).
  iii. Compare with each vehicle present in Sumo. If vehicle is present –Then write confirmation (pipes) and read its position from NetSim (2pipes for X and Y coordinates). Also, sumo is stepped forward for every first vehicle in the list of current vehicles in sumo.

- If vehicle not present, fail (‘f’) is sent.
- Pipes and TCP closed.

13 NetSim Videos

In order to have a better understanding of NetSim, users can access YouTube channel of Tetcos at www.youtube.com/tetcos and check out the various videos available.

13.1.1 NetSim Core Protocol Library

- As simulation proceeds, NetSim VANET library sends vehicle name to python, and receives XY positions, which are passed from python.
- Positions are updated and simulation proceeds.

14 R&D projects in NetSim

Example R & D projects in NetSim is available in http://www.tetcos.com/file-exchange.html

15 NetSim FAQ/Knowledgebase

NetSim knowledgebase with hundreds on FAQs on how NetSim works is available at https://tetcos.freshdesk.com/support/home