

# 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**

Copyright in the whole and every part of this manual belongs to TETCOS LLP and may not be used, sold, transferred, copied or reproduced in whole or in part in any manner or in any media to any person, without the prior written consent of TETCOS LLP. If you use this manual you do so at your own risk and on the understanding that TETCOS LLP shall not be liable for any loss or damage of any kind.

TETCOS LLP may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from TETCOS LLP, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Rev 13.0 (V), March 2021, TETCOS LLP. All rights reserved.

**All trademarks are property of their respective owner.**

### **Contact us at**

TETCOS LLP

# 214, 39<sup>th</sup> A Cross, 7<sup>th</sup> Main, 5th Block Jayanagar,  
Bangalore - 560 041, Karnataka, INDIA.

Phone: +91 80 26630624

E-Mail: [sales@tetcos.com](mailto:sales@tetcos.com)

Visit: [www.tetcos.com](http://www.tetcos.com)

## Table of Contents

<b>1</b>	<b>NetSim – Introduction.....</b>	<b>10</b>
1.1	Introduction to modeling and simulation of networks.....	10
1.2	Versions of NetSim – Academic, Standard & Pro.....	10
1.3	Components (Technology Libraries) in Pro and Standard versions.....	12
<b>2</b>	<b>Installation and License Server Set-up .....</b>	<b>14</b>
2.1	System Requirements .....	14
2.1.1	NetSim Client (installs locally) .....	14
2.1.2	License Server (for running Host-ID/ Dongle locked floating licenses, not applicable for node locked licenses) .....	14
2.2	Installing NetSim.....	14
2.2.1	Silent installation .....	38
2.2.2	Import Compatible Workspaces .....	39
2.3	Setting up License Server .....	41
2.3.1	Installing NetSim RLM Dongle Driver Software (Dongle Based Licenses)	41
2.3.2	Running NetSim License Server .....	43
2.3.3	Running NetSim Software .....	44
<b>3</b>	<b>NetSim GUI.....</b>	<b>45</b>
3.1	Menus in the NetSim Home Screen.....	45
3.1.1	Creating “New” Simulations .....	49
3.1.2	Environment Settings.....	50
3.2	Modeling and Simulating a simple network.....	53
3.2.1	Creating a Network scenario .....	53
3.2.2	Configuring devices and links in the scenario .....	55
3.2.3	Display Settings .....	56
3.2.4	Copy/Paste.....	57
3.2.5	Modeling Application Traffic.....	59
3.2.6	Logging Packet/ Event Trace.....	60
3.2.7	Run Simulation.....	60
3.2.8	ACL Configuration.....	61
3.3	Saving & Opening experiments and Printing results .....	65
3.3.1	Opening Saved Experiments:.....	65
3.3.2	Saving an Experiment.....	66
3.4	NetSim Keyboard Shortcuts .....	66
3.5	NetSim Interactive Simulation .....	67

3.5.1	Simulation specific (Not applicable for file based interactive simulation) ..	70
3.5.2	Ping Command .....	70
3.5.3	Route Commands .....	72
<b>4</b>	<b>Workspaces and Experiments.....</b>	<b>74</b>
4.1	What is an Experiment and what is a workspace in NetSim? .....	74
4.2	How does a user create and save an experiment in workspace? .....	75
4.3	Should each user have a workspace? .....	80
4.4	How does a user export an experiment? .....	80
4.5	How does a user delete an Experiment in a workspace?.....	81
4.6	How does a user create a new workspace? .....	81
4.7	How does a user switch between workspaces? .....	83
4.8	How does a user export a workspace? .....	84
4.9	How does a user import a workspace? .....	88
4.10	How does a user import an experiment? .....	89
4.11	How does a user delete a workspace? .....	90
4.12	How does a user open and modify source codes?.....	91
4.13	Can I use NetSim's default code for my experiments? .....	92
<b>5</b>	<b>Simulating different networks in NetSim .....</b>	<b>93</b>
5.1	Internetworks .....	94
5.1.1	Internetworks Examples .....	94
5.1.2	Internet Documentation.....	94
5.2	Legacy Networks .....	94
5.2.1	Legacy Networks Examples .....	95
5.2.2	Legacy Network Documentation.....	95
5.3	Cellular Networks.....	95
5.3.1	Cellular Networks Examples.....	96
5.3.2	Cellular Networks Documentation .....	96
5.4	Advanced Routing .....	96
5.4.1	Advanced Routing Examples.....	96
5.4.2	Advanced Routing Documentation .....	97
5.5	MANETs.....	97
5.5.1	MANET Examples.....	97
5.5.2	MANET Documentation .....	97
5.6	Wireless Sensor Networks (WSN) .....	98
5.6.1	Wireless Sensor Networks (WSN) Examples .....	98
5.6.2	WSN Library Documentation .....	98
5.7	Internet of Things .....	99

5.7.1	Internet of Things (IOT) Examples .....	99
5.7.2	IOT Library Documentation.....	99
5.8	Software Defined Networks (SDN).....	99
5.8.1	Software Defined Networks (SDN) Examples .....	100
5.8.2	SDN Library Documentation.....	100
5.9	Cognitive Radio.....	100
5.9.1	Cognitive Radio Examples.....	101
5.9.2	Cognitive Radio Library Documentation.....	101
5.10	LTE.....	101
5.10.1	LTE Examples.....	101
5.10.2	LTE Library Documentation .....	102
5.11	5G NR .....	102
5.11.1	5G NR Examples .....	102
5.11.2	5G NR Library Documentation .....	102
5.12	VANETs .....	103
5.12.1	VANET Examples .....	103
5.12.2	VANET Library Documentation .....	103
5.13	Satellite Communication.....	103
5.13.1	Satellite Communication Examples.....	104
5.13.2	Satellite Communication Documentation .....	104
5.14	TDMA Radio Networks .....	104
5.14.1	TDMA Radio Network Examples .....	104
5.14.2	TDMA Radio Network Library Documentation .....	105
<b>6</b>	<b>Applications (Network Traffic Generator) .....</b>	<b>106</b>
6.1	Common properties for all applications .....	107
6.2	Application Types .....	110
6.3	Network Traffic Generation Rate for Different Applications .....	120
6.4	Priority and QoS of Applications.....	122
6.5	Capture real applications and simulate in NetSim .....	122
6.6	Modelling Poisson arrivals in NetSim.....	123
6.7	Application Configuration – Special Conditions .....	124
<b>7</b>	<b>Running Simulation via Command Line Interface .....</b>	<b>125</b>
7.1	Running NetSim via CLI .....	125
7.1.1	Running in CLI Mode when using floating licenses .....	126
7.1.2	Running in CLI Mode when using node-locked or cloud licenses .....	127
7.1.3	Quick edit for copy pastes in CLI mode.....	128
7.2	Understanding the Configuration.netsim file .....	129

7.2.1	How to use Visual Studio to edit the Configuration file? .....	129
7.2.2	Sections of Configuration file .....	130
7.2.3	Sample Configuration file.....	131
7.2.4	Configuration.xsd file .....	131
<b>8</b>	<b>Outputs: Results, Plots and Data Files .....</b>	<b>132</b>
8.1	Result Window and Plots Windows.....	132
8.1.1	Application and Link Throughput Plots.....	132
8.1.2	Buffer Occupancy Plot .....	133
8.1.3	TCP Congestion Window Plot .....	135
8.1.4	Notes on plots .....	136
8.1.5	Link metrics .....	136
8.1.6	Queue Metrics.....	137
8.1.7	Protocol Metrics .....	138
8.1.8	Device Metrics.....	138
8.1.9	Cellular Metrics .....	138
8.1.10	Channel metrics .....	139
8.1.11	Sensor metrics .....	139
8.1.12	Battery Model .....	140
8.1.13	CR metrics .....	141
8.1.14	Application Metrics.....	142
8.1.15	IP Metrics .....	143
8.1.16	Advanced Metrics .....	143
8.1.17	Notes on metrics .....	144
8.1.18	The different results files written at the end of simulation.....	144
8.2	Export to .csv .....	145
8.3	Packet Animation.....	147
8.3.1	Packet animation Table .....	148
8.3.2	Packet animation – Display Settings .....	149
8.3.3	Example on how to use NetSim packet animation feature: .....	149
8.3.4	How to record and save Packet animation as a Video file.....	152
8.4	Packet Trace.....	154
8.4.1	How to set filters to NetSim trace file.....	155
8.4.2	Observing packet flow in the Network through packet trace file .....	156
8.4.3	Analysing Packet Trace using Pivot Tables .....	157
8.4.4	Packet Transmitted / Received Analysis .....	160
8.4.5	Delay analysis .....	163
8.4.6	Throughput analysis.....	168

8.4.7	Plotting with Pivot Charts.....	169
8.4.8	Packet Trace Fields .....	172
8.5	Event Trace (only in Standard/Pro Version) .....	174
8.5.1	NetSim Network Stack and Discrete Event Simulation working .....	174
8.5.2	Event Trace.....	175
8.5.3	Calculation of Delay and Application throughput from event trace.....	177
8.6	Packet Capture & analysis using Wireshark .....	185
8.6.1	Enabling Wireshark Capture in a node for packet capture .....	185
8.6.2	Viewing captured packets.....	186
8.6.3	Filtering captured packets.....	187
8.6.4	Analyzing packets in Wireshark.....	187
8.6.5	Window Scaling.....	188
<b>9</b>	<b>Writing Custom Code in NetSim .....</b>	<b>192</b>
9.1	Writing your own code .....	192
9.1.1	Microsoft Visual Studio 2019 Installation Settings .....	192
9.1.2	Modifying code .....	193
9.1.3	Building DLLs .....	195
9.1.4	Running Simulation.....	196
9.1.5	Source Code Dependencies.....	197
9.1.6	Enabling Additional Security Checks.....	198
9.2	Implementing your code - Examples .....	200
9.2.1	Hello World Program.....	200
9.2.2	Introducing Node Failure in MANET.....	201
9.3	Debugging your code.....	203
9.3.1	Via GUI.....	203
9.3.2	Via CLI.....	211
9.3.3	Co-relating with Event Trace .....	213
9.3.4	Viewing & Accessing variables .....	216
9.3.5	Print to console window in NetSim .....	223
9.4	Creating a new packet and adding a new event in NetSim .....	224
9.5	NetSim API's.....	230
<b>10</b>	<b>Advanced Features .....</b>	<b>233</b>
10.1	Random Number Generator and Seed Values .....	233
10.2	Interfacing MATLAB with NetSim (Std/Pro versions).....	233
10.2.1	Implement Weibull Distribution of MATLAB without using .m file .....	235
10.2.2	Debug and understand communication between NetSim and MATLAB.	244

10.2.3 Implement Weibull Distribution of MATLAB in NetSim using .m file:.....	247
10.2.4 Plot a histogram in MATLAB per a Weibull distribution (using .m file) ....	249
10.3 Interfacing tail with NetSim.....	253
10.4 Adding Custom Performance Metrics .....	257
10.5 Simulation Time and its relation to Real Time (Wall clock) .....	261
10.6 Adding Custom Plots .....	262
10.6.1 Plotting SNR for each UE-gNB pair in 5G NR.....	262
10.7 Environment Variables in NetSim.....	268
10.8 Best practices for running large scale simulations.....	269
10.9 Batch experimentation and automated simulations .....	270
<b>11 NetSim Emulator .....</b>	<b>271</b>
11.1 Introduction .....	271
11.1.1 Simulating and Analyzing Emulation Examples.....	271
<b>12 Troubleshooting in NetSim.....</b>	<b>272</b>
12.1 CLI mode.....	272
12.2 I/O warning displayed in CLI mode .....	272
12.2.1 Connection refused at server<-111> error displayed.....	272
12.2.2 Unable to load license config dll(126) .....	273
12.2.3 “Error in getting License” error in CLI mode .....	273
12.2.4 Unable to load license config dll displayed.....	274
12.3 Configuration.netsim.....	275
12.3.1 Invalid attribute in configuration file attributes .....	275
12.3.2 Error in tags in configuration file attributes .....	275
12.3.3 Error lines in configuration.xsd in the Configuration file.....	276
12.4 Simulation terminates and “NetSim Backend has stopped working” displayed ..	277
12.5 Monitor screen resolution is less than 1024X768 .....	277
12.6 Licensing .....	278
12.6.1 No License for product (-1) error .....	278
12.7 Troubleshooting VANET simulations that interface with SUMO.....	278
12.7.1 Guide for Sumo .....	278
12.7.2 Guide for Python .....	279
12.7.3 VANET Simulation .....	280
12.7.4 Python .....	280
12.7.5 NetSim Core Protocol Library .....	280
<b>13 NetSim Videos .....</b>	<b>281</b>
<b>14 R&amp;D projects in NetSim.....</b>	<b>281</b>

<b>15 NetSim FAQ/Knowledgebase .....</b>	<b>281</b>
--	------------

# 1 NetSim – Introduction

## 1.1 Introduction to modeling and simulation of networks

A network simulator 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.
- **Visualizing the simulation:** Use the packet animator to view the flow of packets.
- **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 comes 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 education institutions while NetSim Pro version addresses the needs of defense and industry. The standard and pro versions are available as components in NetSim v13.0 from which users can choose and assemble. A comparison of the features in the three versions are tabulated below

**Table 1-1.**

Features	Academic	Standard	Pro
<b>Technology Coverage</b>			
Internetworks	Y	Y	Y
Legacy & Cellular Networks	Y	Y	Y
Mobile Adhoc networks	Y	Y	Y
Software Defined Networks	Y	Y	Y
Wireless Sensor Networks	Y	Y	Y
Internet of Things	Y	Y	Y
Cognitive Radio Networks	Y	Y	Y
LTE Networks	Y	Y	Y
5G NR	N	Y	Y
VANET	N	Y	Y
Satellite Communication Networks	N	Y	Y
<b>Performance Reporting</b>			
Performance metrics available for Network and Sub-networks	Y	Y	Y
<b>Packet Animator</b>			
Used to animate the packet flow in network	Y	Y	Y
<b>Packet Trace</b>			
Available in tab ordered .txt format for easy post processing	Y	Y	Y
<b>Event Trace</b>			
Available in tab ordered .txt format for easy post processing	N	Y	Y
<b>Protocol Library Source Codes with Documentation</b>			
Protocol C source codes and appropriate header files with extensive documentation	N	Y	Y
<b>External Interfacing</b>			
Interfacing with SUMO	N		
MATLAB	N	Y	Y
Wireshark	Y		
<b>Integrated debugging</b>			
Users can write their own code, link their code to NetSim and debug using Visual Studio	N	Y	Y
<b>Plots</b>			
Allows users to plot the value of a parameter over simulation time	Y	Y	Y
<b>Simulation Scale</b>	100 Nodes	500 Nodes	~ 10,0000 Nodes
<b>Custom Coding and Modeling Support</b>	N	N	Y
<b>Emulator (Add on)</b>			
Connect to real hardware running live application	N	Y	Y
<b>TDMA Radio Networks (Add On)</b>			
TDMA and DTDMA	N	N	Y
<b>Target Users and Segment</b>	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
<b>Component 1</b> (Base. Required for all components)	<b>Internetworks</b> 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, Friis 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 <b>Common Modules</b> Traffic Generator: Voice, Video, FTP, Database, HTTP, Email, P2P, Custom, CBR. Virtual Network Stack, Simulation Kernel, Command Line Interface Command Line Interpreter Metrics Engine with packet and event trace Plot Generator Packet Animator, Packet Encryption External Interfaces: MATLAB, Wireshark	IEEE 802.3 IEEE 802.11 a/b/g/n/ac/p/e RFCs 2453, 2328, 826, 793, 2001 and 768
<b>Component 2</b>	<b>Legacy &amp; Cellular Networks</b> Aloha – (Pure & Slotted) GSM CDMA	3GPP, ETSI, IMT-MC, IS-95 A/B, IxRTT, 1x-EV-Do, 3xRTT
<b>Component 3</b>	<b>Advanced Routing</b> Multicast Routing - IGMP, PIM, Access Control Lists, Detailed Layer 3 switch mode, Virtual LAN (VLAN), Public IP, Network Address Translation (NAT)	IETF RFC's 1771 & 3121
<b>Component 4</b>	<b>Mobile Adhoc Networks</b> MANET - DSR, AODV, OLSR, ZRP	IETF RFC 4728, 3561, 3626
<b>Component 5</b>	<b>Software Defined Network (SDN)</b>	Based on Open Flow v1.3
<b>Component 6</b> (Component 4 required)	<b>Internet of things (IOT)</b> with RPL protocol Wireless Sensor Networks (WSN)	IEEE 802.15.4 MAC, MANET in L3 RFC 6550
<b>Component 7</b>	<b>Cognitive Radio Networks</b> WRAN	IEEE 802.22

<b>Component 8</b>	<b>Long-Term Evolution Networks:</b> LTE	3GPP
<b>Component 9</b> (Component 4 required)	<b>VANETs:</b> IEEE 1609 WAVE, Basic Safety Message (BSM) protocol per J2735 DSRC, Interface with SUMO for road traffic simulation	IEEE 1609
<b>Component 10</b> (Components 3 and 8 required)	<b>5G NR :</b> 3GPP 38 Series. Full Stack covering SDAP, PDCP, RLC – UM, TM, MAC, PHY – FR1 and FR2, mmWave propagation.	3GPP 38.xxx
<b>Component 11</b> (Component 3 required)	<b>Satellite Communication Networks:</b> 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
<b>TDMA Radio Networks Add on</b> (Pro version only)	<b>TDMA Radio Networks</b> TDMA Link 16, Dynamic TDMA, Frequencies – HF, VHF, UHF Bands, Frequency Hopping	----
<b>Network Emulator Add On</b>	<b>Network Emulator</b> 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: i3 equivalent or above, RAM: 4 GB (Min). 8GB Recommended.
- Monitor resolution: Min - 1024\*768, Max - 1920\*1080. Optional Scale and layout setting: 100%
- Operating system: 64 bit. Win 8 or Win 10, 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 / v13.0: Microsoft Visual Studio 2019 (or higher)

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

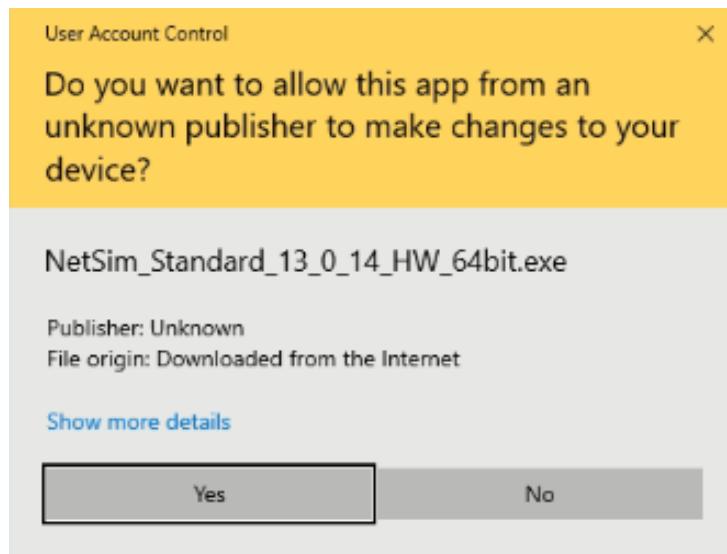
#### 2.1.2 License Server (for running Host-ID/ Dongle locked floating licenses, not applicable for node locked licenses)

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 is 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 server through the network.

### 2.2 Installing NetSim

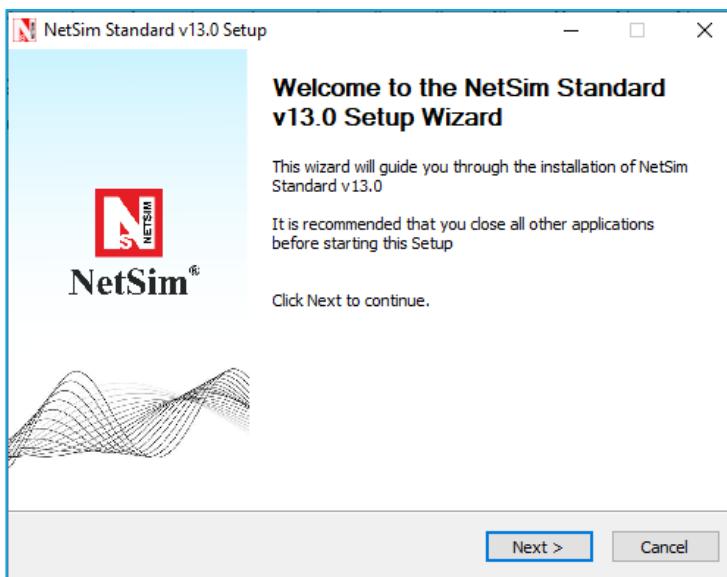
Install the **32-bit or 64-bit build** of NetSim depending on your PC platform. Based on the NetSim version under installation, the version type being displayed in the following window will change.

For example, you will see **NetSim\_Standard\_13\_0\_14\_HW\_64bit.exe** for a Standard version install. Double click on the setup file. Click on **Yes** button to install the software.



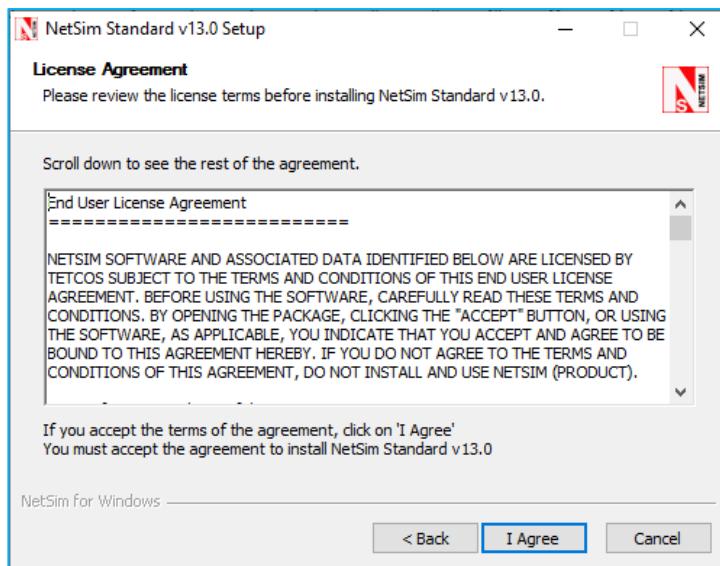
**Figure 2-1:** User Account Control message window appears and select Yes button.

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



**Figure 2-2:** Select Next button 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** button else quit the setup by clicking **Cancel** button.



**Figure 2-3:** Select I Agree button

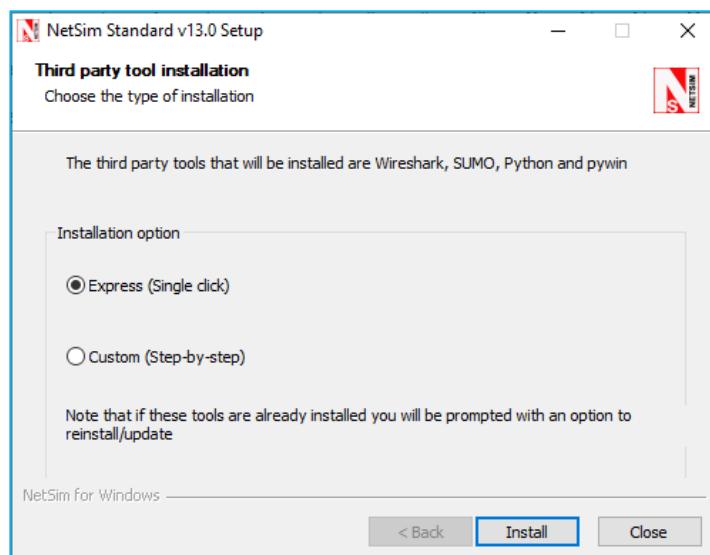
If you agree with the license agreement, you will be prompted to select either one of the installation options, Express (Single-click installation) or Custom (Step-by-Step installation).

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 with NetSim.

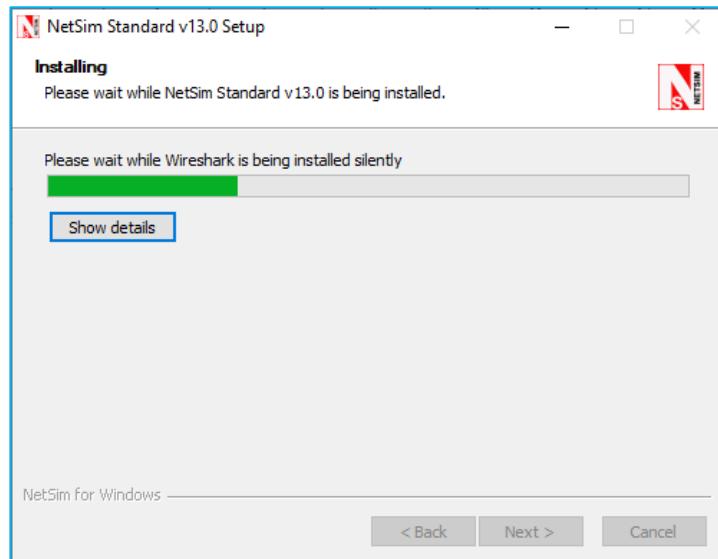
Both the installation methods are explained below:

Choose the **Express (Single click)** option and click on the **Install** button.

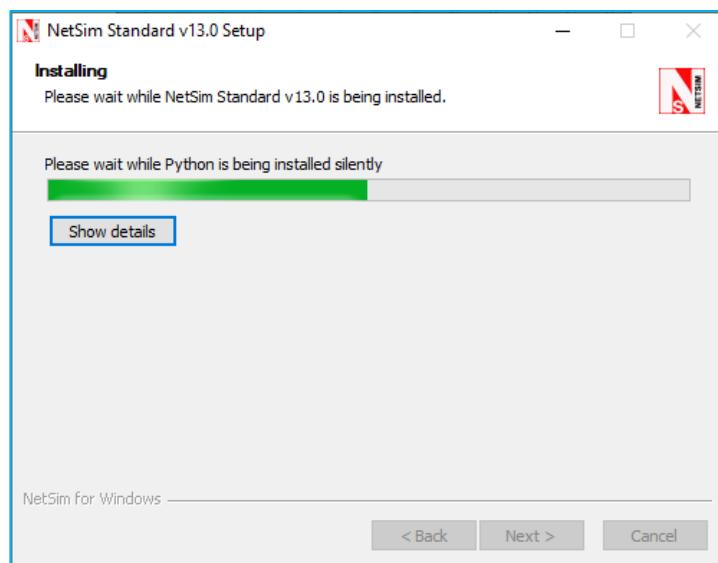


**Figure 2-4:** Select Express (Single click) radio button and Click on install

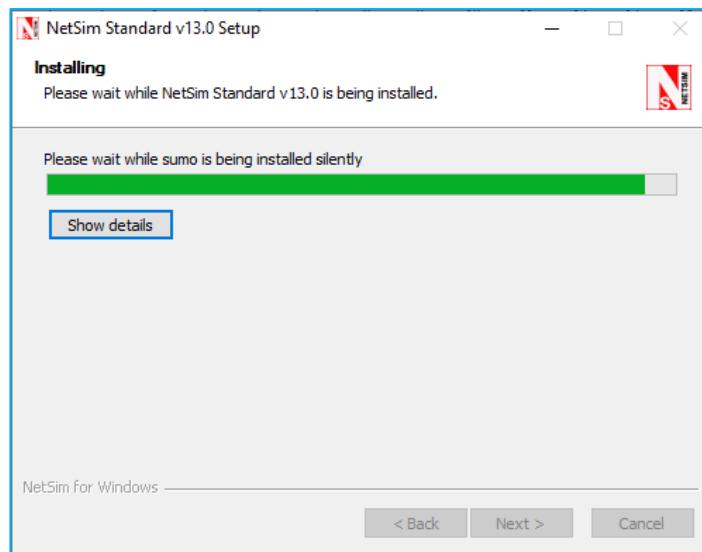
NetSim installation starts and users can see that the third-party tools are getting installed one by one silently.



**Figure 2-5:** Wireshark is being installed Silently.



**Figure 2-6:** Python is being installed Silently.



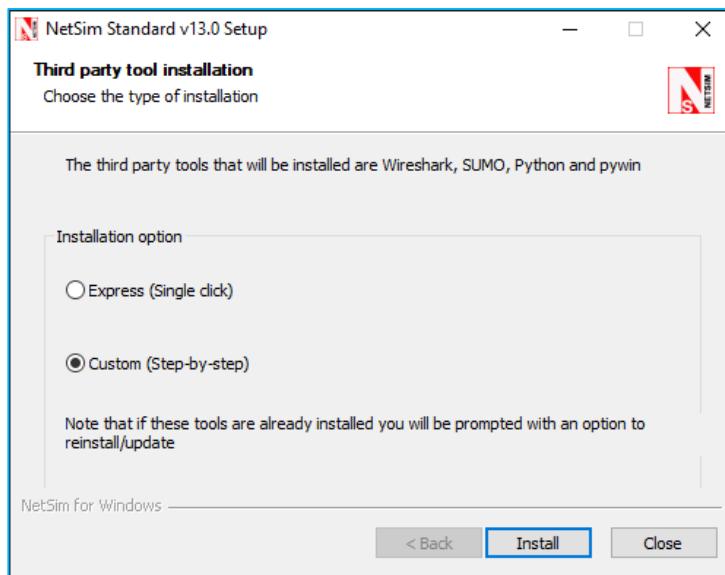
**Figure 2-7:** Sumo is being installed Silently.

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



**Figure 2-8:** Select Finish button to complete the installation process of NetSim.

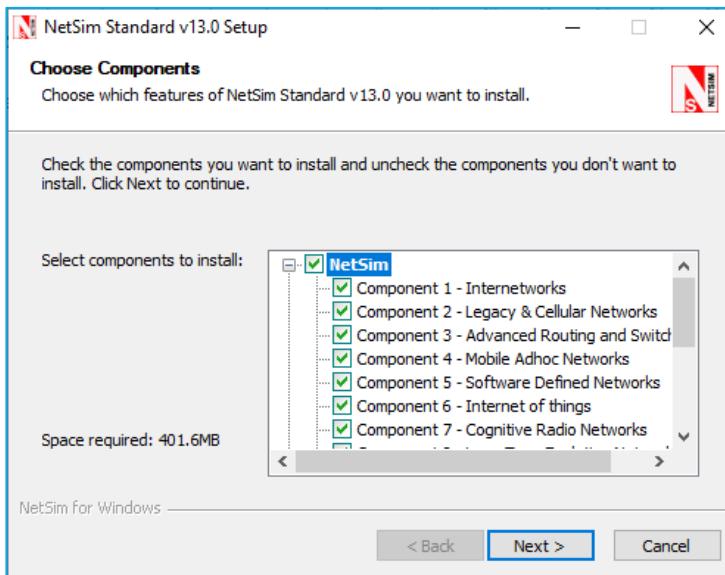
Otherwise, Choose the **Custom (Step-by-step)** option and click on the **Install** button.



**Figure 2-9:** Select Custom (Step-by-Step) radio button and Click on install.

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.

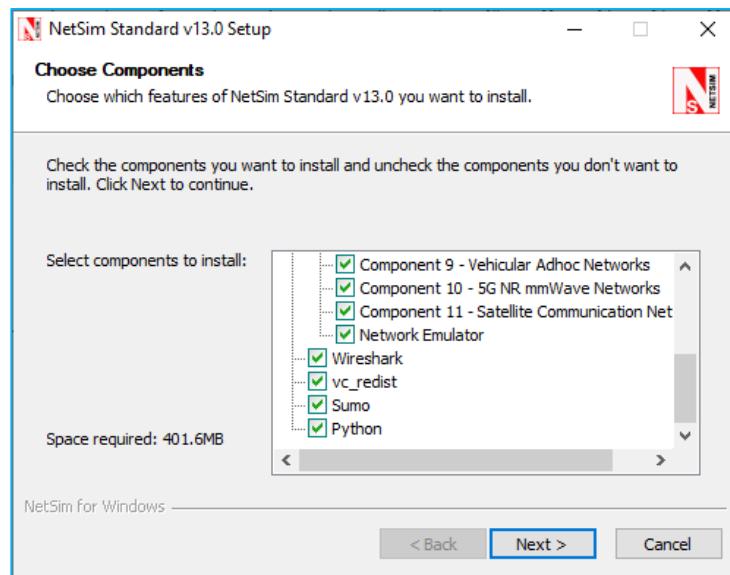
**Note:** In Standard and Pro Versions of NetSim, the Choose Components screen will display only those components for which the licenses are obtained by the user. Also, Network Emulator and Real Time Protocol are available as Add-On along with NetSim.



**Figure 2-10:** 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:

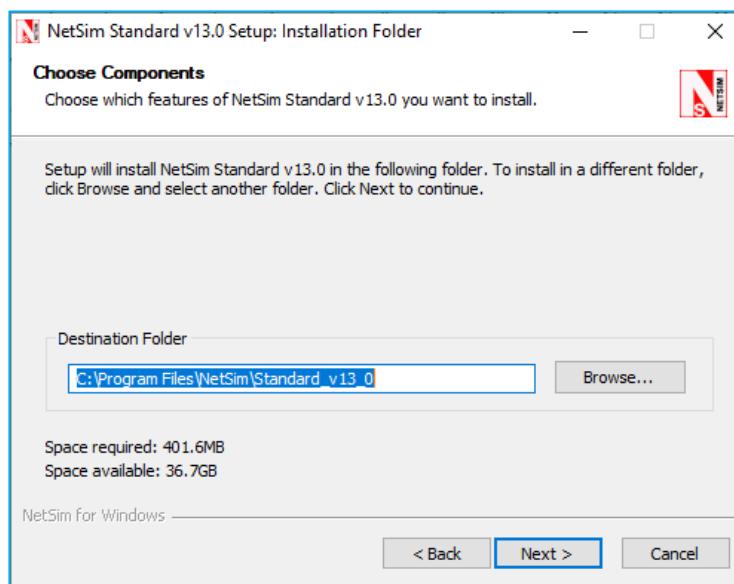
Click on the **Next** button.



**Figure 2-11:** list of third-party tools

**Note:** Sumo and Python comes only as a part of Standard and Pro Version Install.

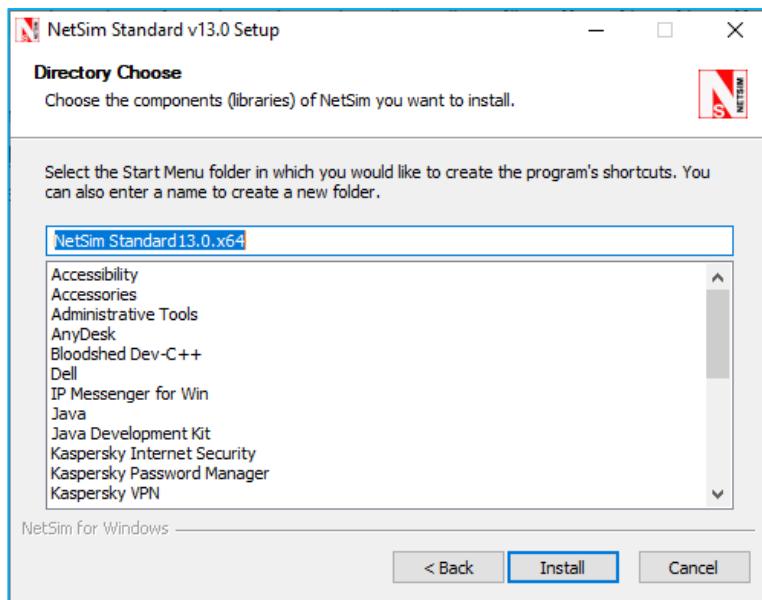
In the next screen, you will be requested to enter the installation path. Select the path in which the software needs to be installed and click on **Next** button.



**Figure 2-12:** NetSim installation directory path

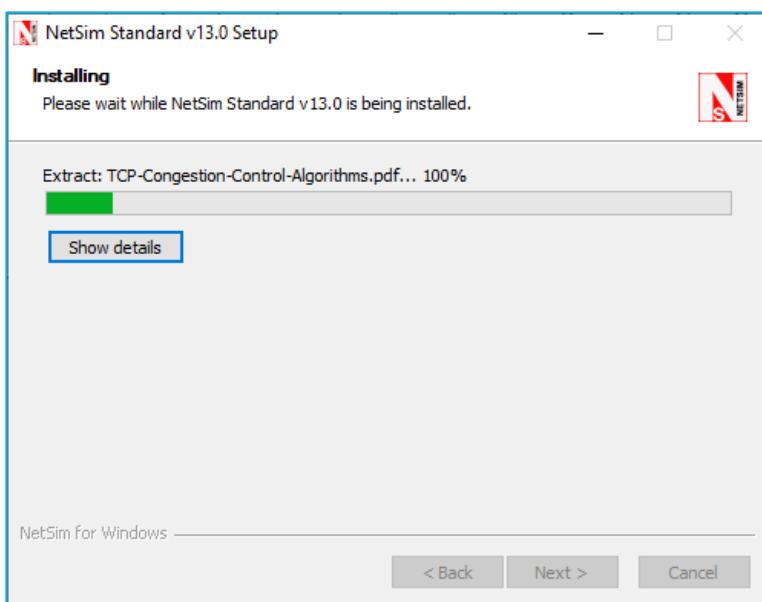
**Note:** In the case you are installing 32bit NetSim in a 64-bit machine, ensure that the path is <OS installed drive> C:/Program Files (x86)/NetSim/Standard\_v13\_0.

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** button to start the installation.



**Figure 2-13:** Start Menu folder name

The installation process begins.



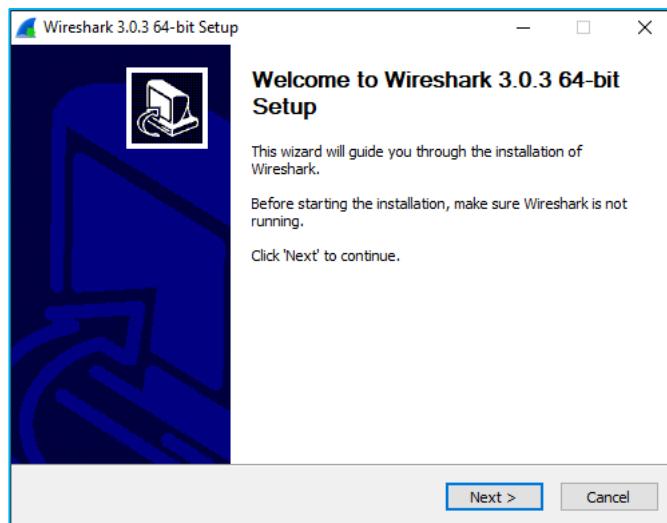
**Figure 2-14:** NetSim Standard v13.0 being installed.

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

For NetSim Academic Version, Adobe Flash Player, WinPcap, Wireshark will be installed.

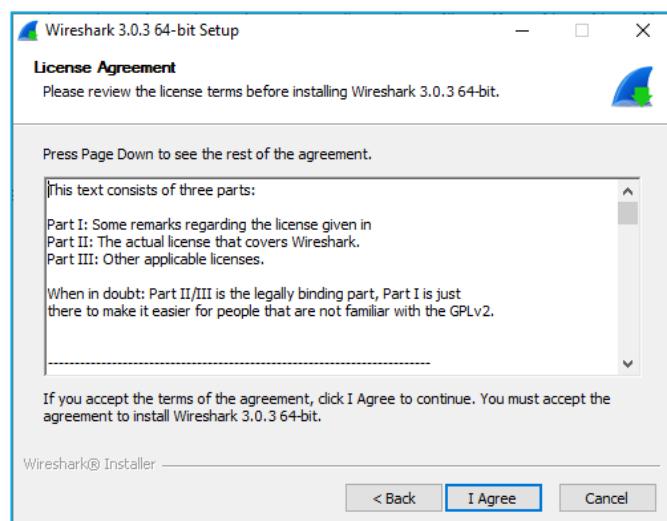
For NetSim Standard and Pro Versions, along with WinPcap and Wireshark installation, Python installation will start automatically. (*If not deselected during 3rd party software selection*)

Click on **Next** button to start Wireshark installation.



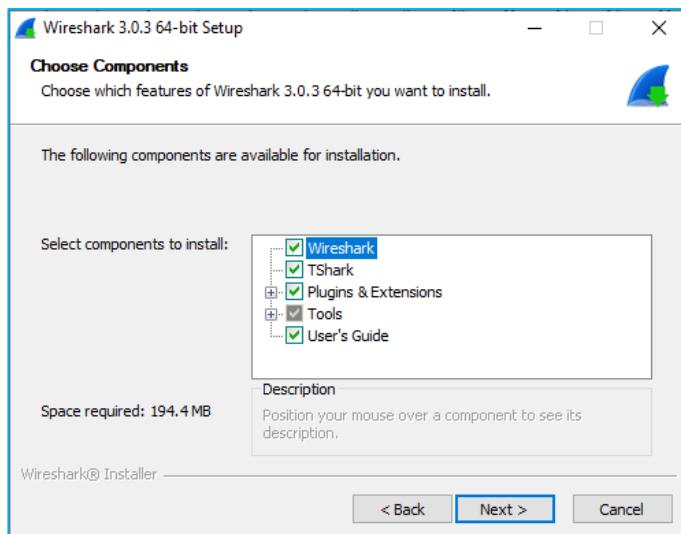
**Figure 2-15:** Select Next button to start Wireshark installation

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



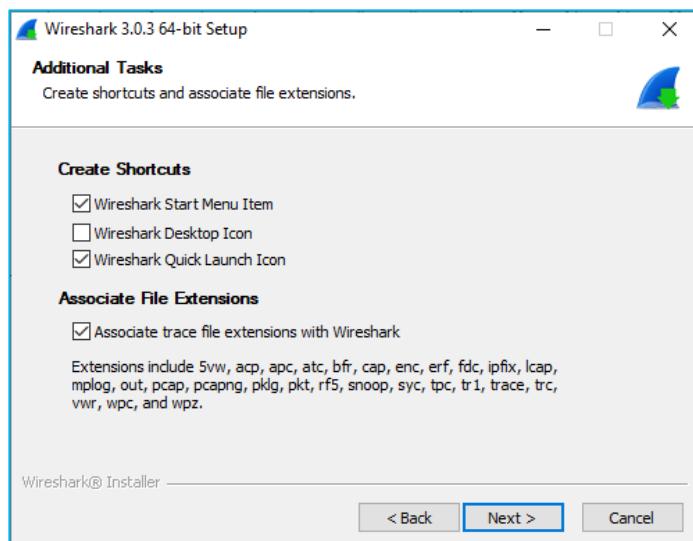
**Figure 2-16:** Wireshark License Agreement window

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



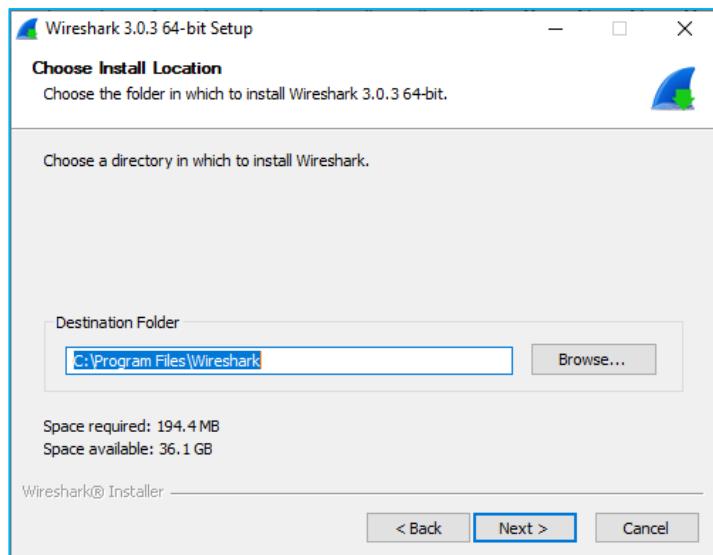
**Figure 2-17:** Choose Wireshark features

Click on **Next** button.



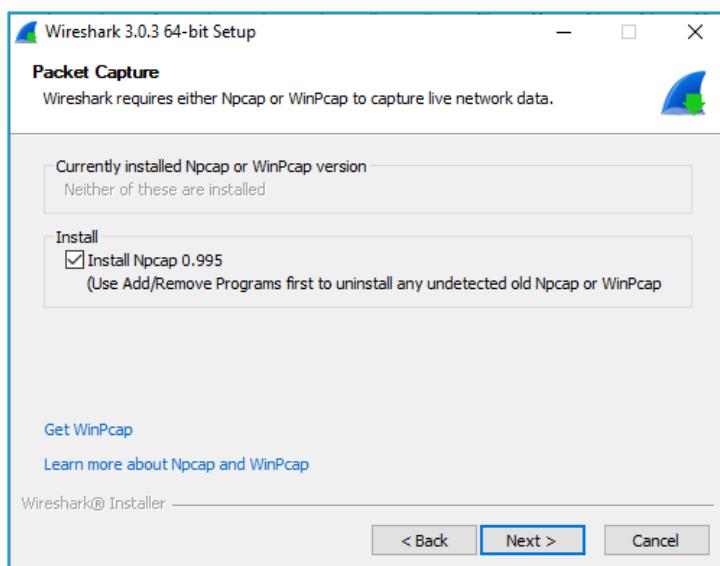
**Figure 2-18:** Select Next button

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



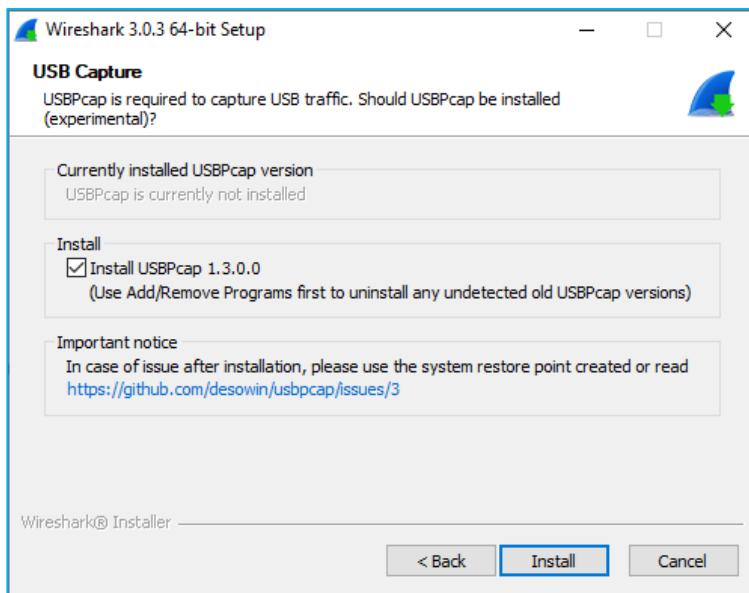
**Figure 2-19:** Wireshark installation directory path

Select **Install Npcap 0.995** and click on **Next** button.



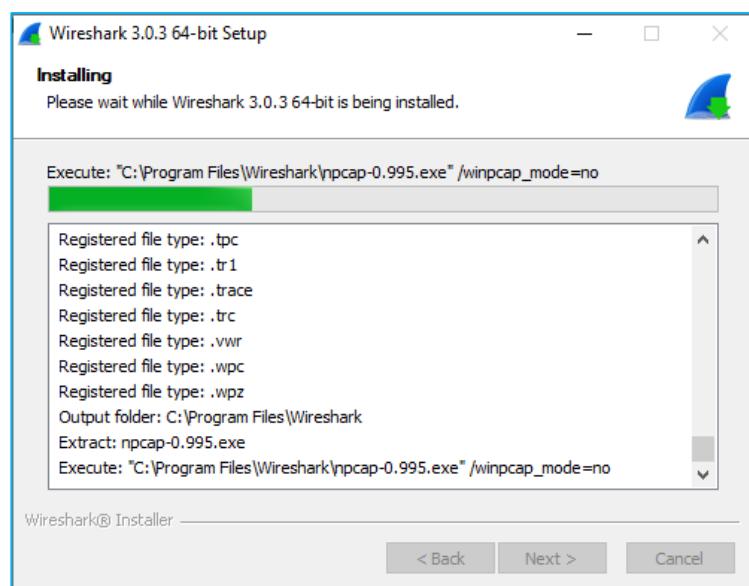
**Figure 2-20:** Select Install Npcap 0.995 in Wireshark window

Select **Install USBPcap 1.3.0.0** and click on **Install** button.



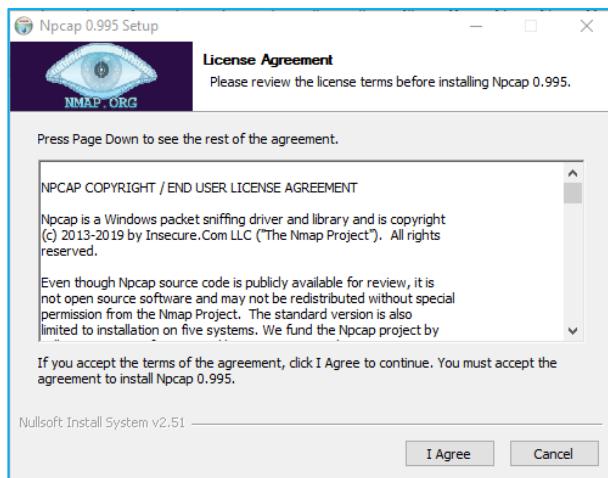
**Figure 2-21:** Select Install USBPcap 1.3.0.0 in Wireshark window

The installation process begins.



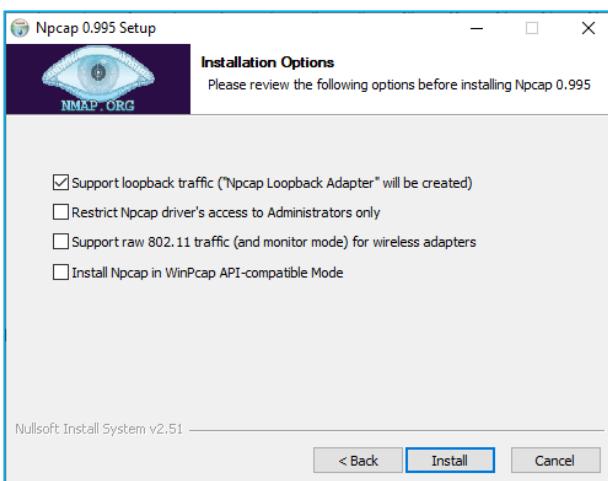
**Figure 2-22:** Wireshark installation process begins

Npcap License Agreement window appears. Click on **I Agree** button.



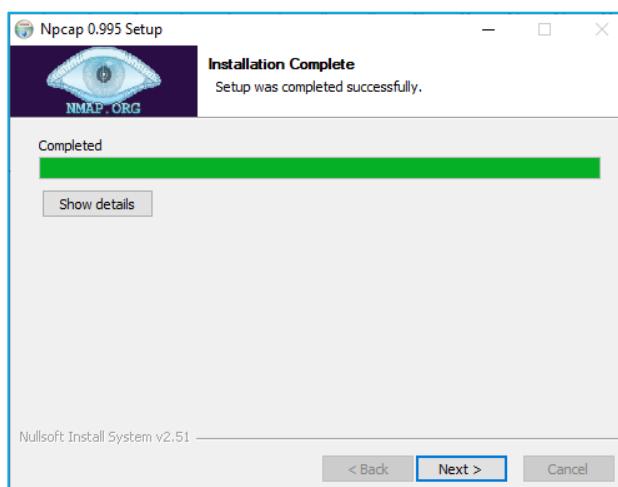
**Figure 2-23:** Npcap License Agreement window

Installation Options window appears. Click on **Install** button.



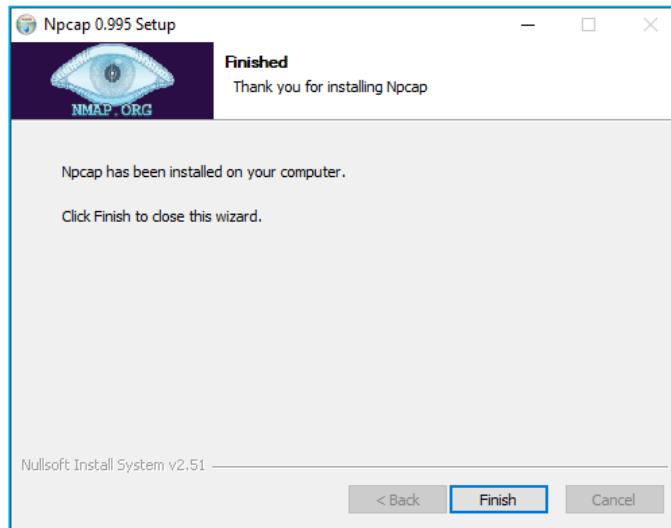
**Figure 2-24:** Select Install options

Once the installation is completed successfully, click on **Next** button.



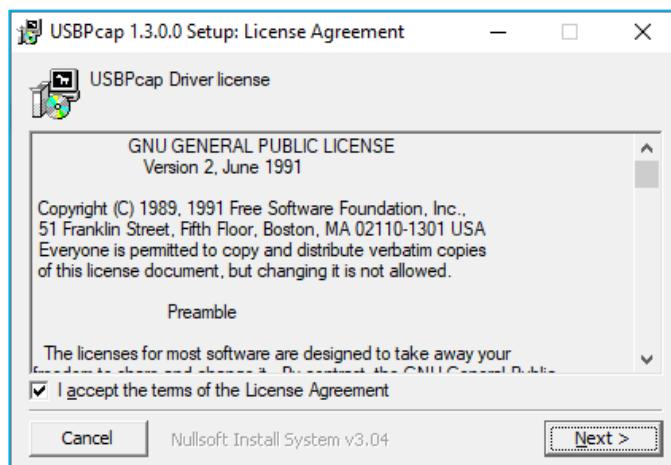
**Figure 2-25:** Select Next button

You will get the Npcap Setup Finished window. Click on **Finish** button.



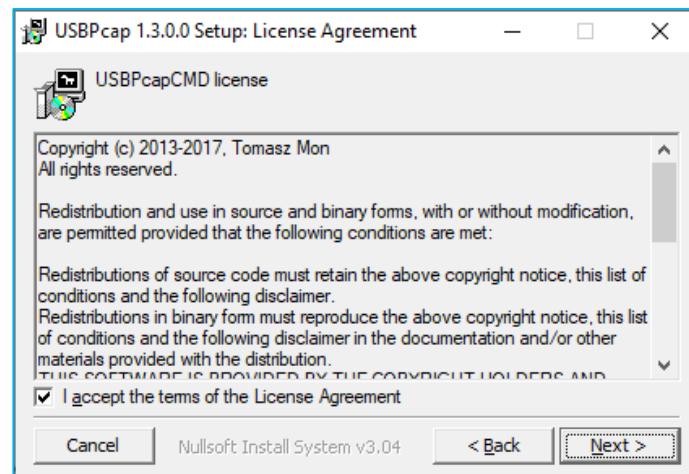
**Figure 2-26:** Select Finish button to complete Npcap

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



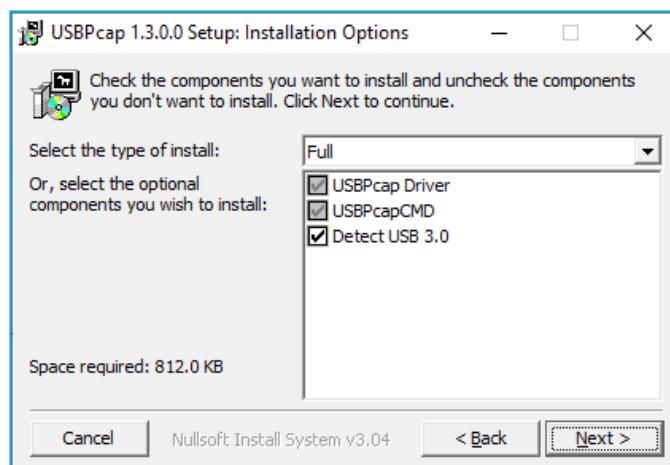
**Figure 2-27:** USBPcap Driver License Agreement window

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



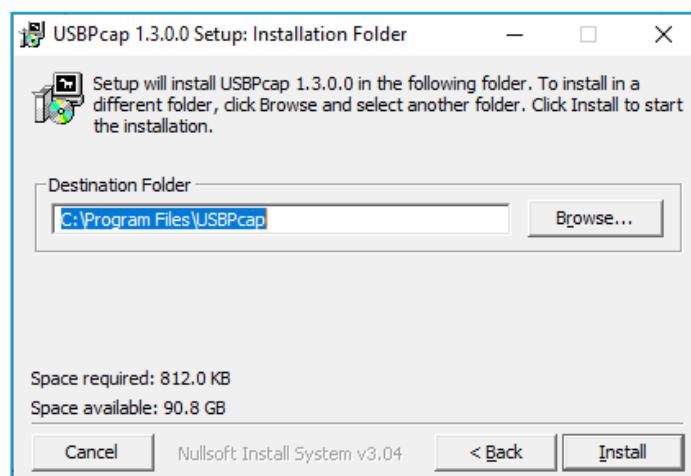
**Figure 2-28:** USBPcap CMD License Agreement window

USBPcap Setup Installation window appears. Click on **Next** button.



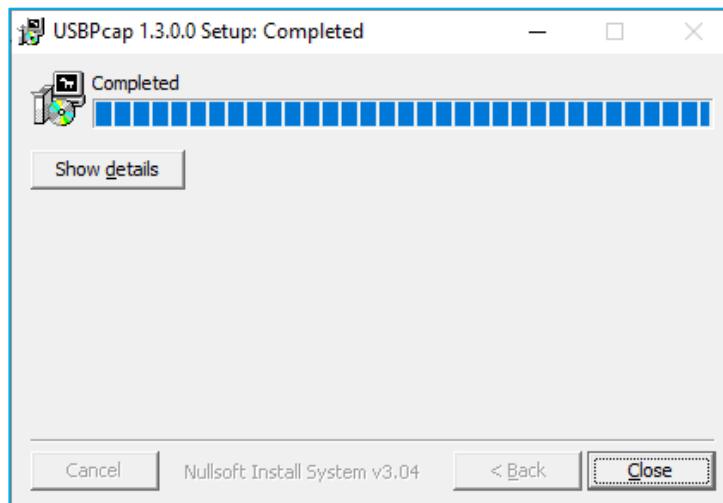
**Figure 2-29:** USBPcap Setup Installation window

Select the path in which USBPcap needs to be installed and click on **Install** button.



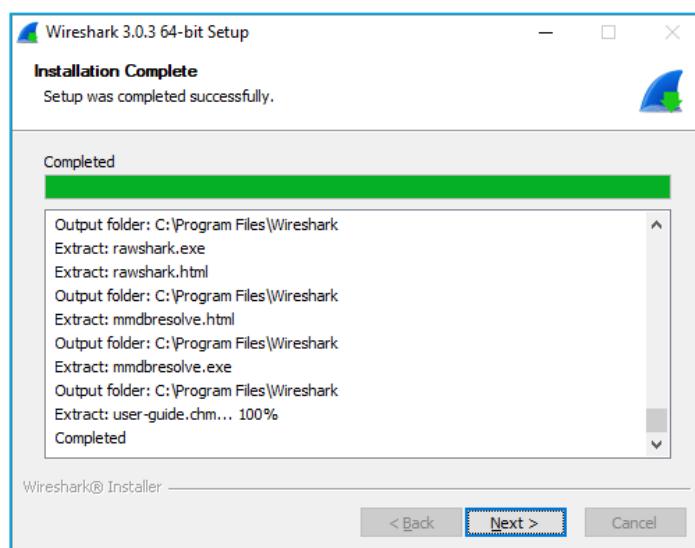
**Figure 2-30:** USBPcap installation path

Once the installation is completed successfully, click on **Close** button.



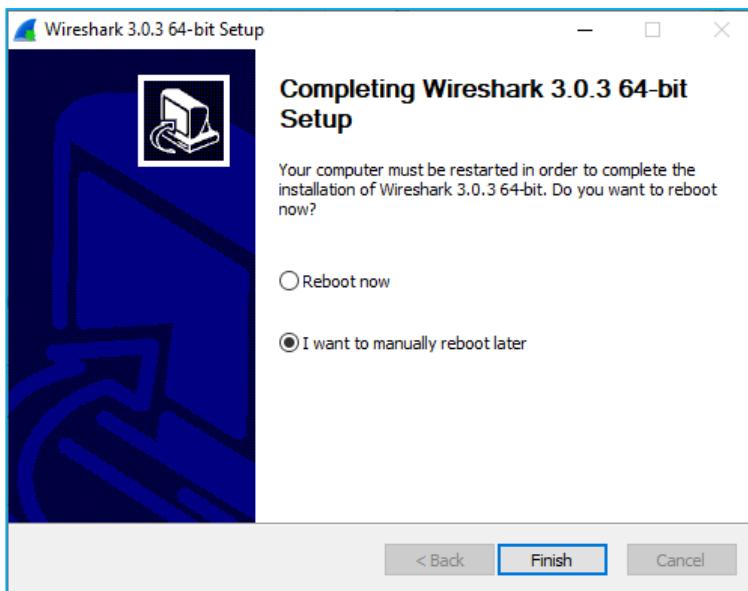
**Figure 2-31:** Installation is completed

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

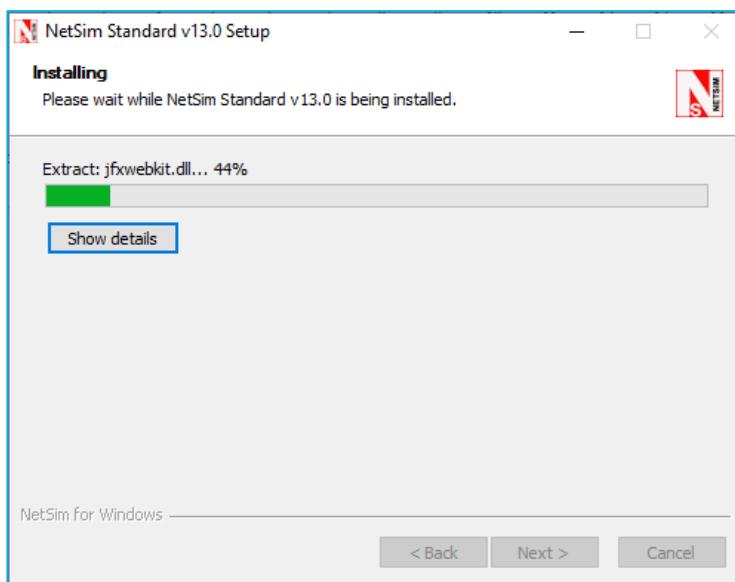


**Figure 2-32:** Installation Complete dialog box and select next button

You will get the Wireshark Completing Setup window. Select the option **I want to manually reboot later** and Click on **Finish** button.



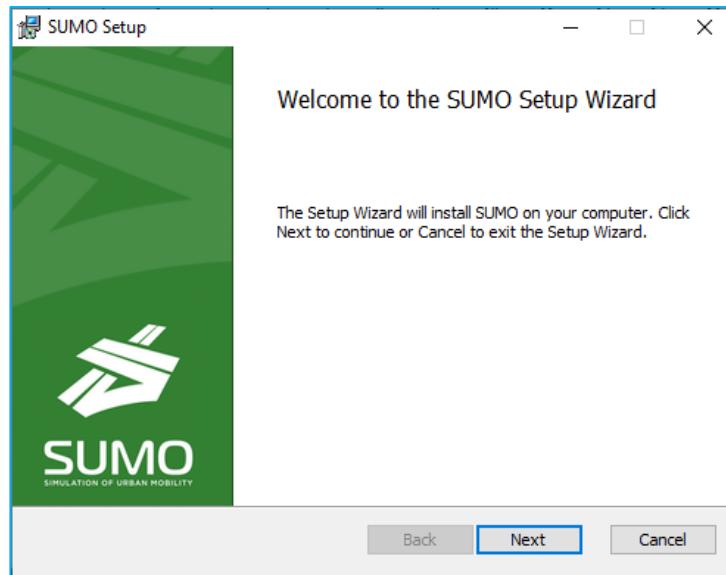
**Figure 2-33:** Select the option I want to manually reboot later and Click on Finish button  
NetSim Standard Setup Installation proceeds further.



**Figure 2-34:** Setup Installation proceeds

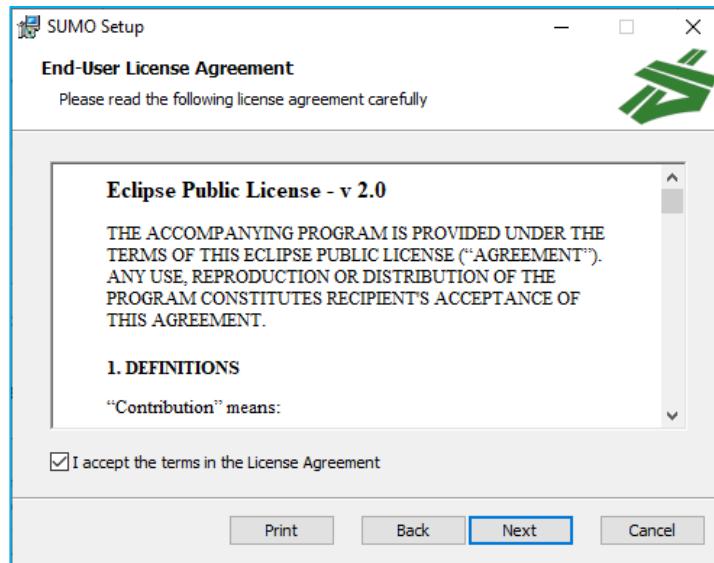
**Note:** During the installation of NetSim Academic version, the supporting software installed is WinPcap.

SUMO Set Up installation wizard appears. Click on **Next** button to continue with Sumo installation.



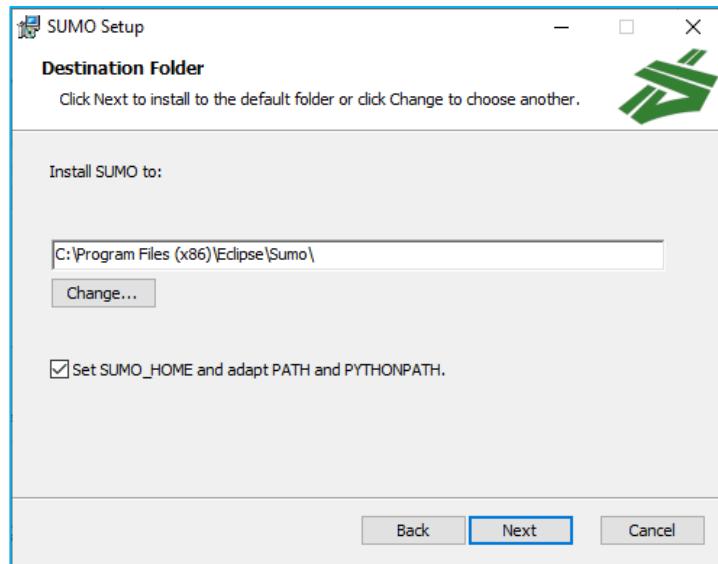
**Figure 2-35:** SUMO Set Up installation wizard

SUMO License Agreement window appears. Click on I accept the terms in the License Agreement checkbox and click on **Next** button.



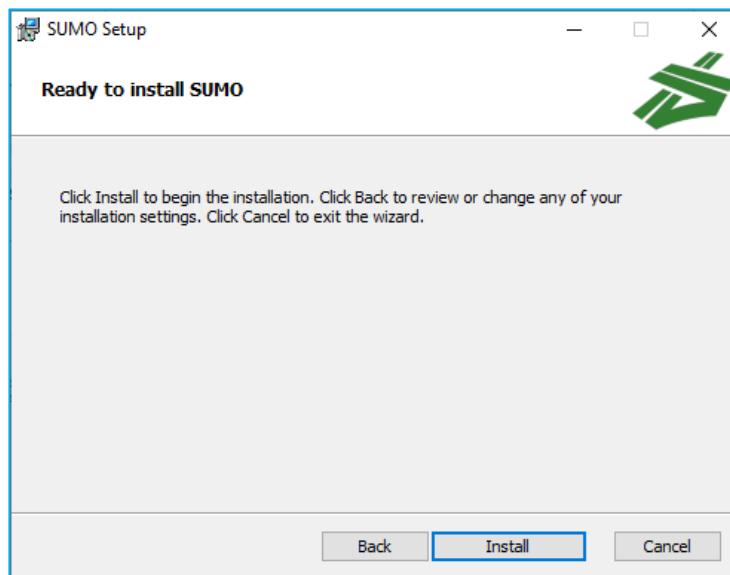
**Figure 2-36:** SUMO License Agreement window

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



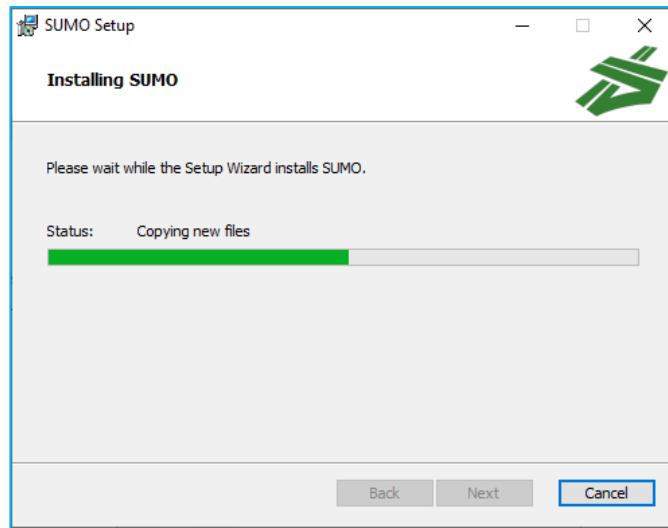
**Figure 2-37:** SUMO installation directory path

SUMO Install Set Up window appears. Click on **Install** button.



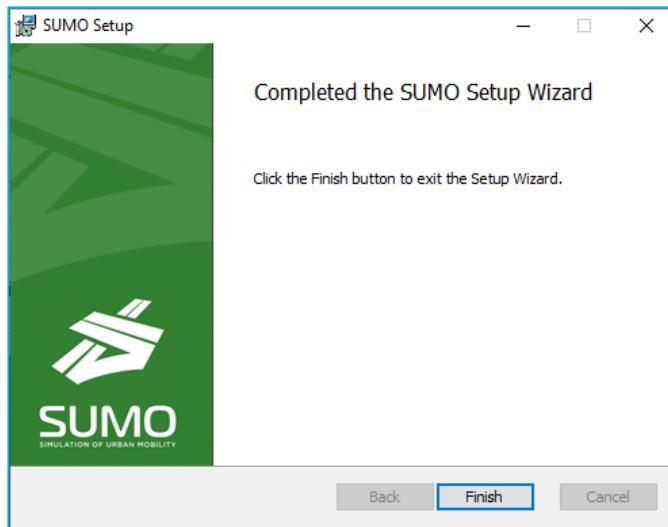
**Figure 2-38:** SUMO Install Set Up window

SUMO Installation begins.



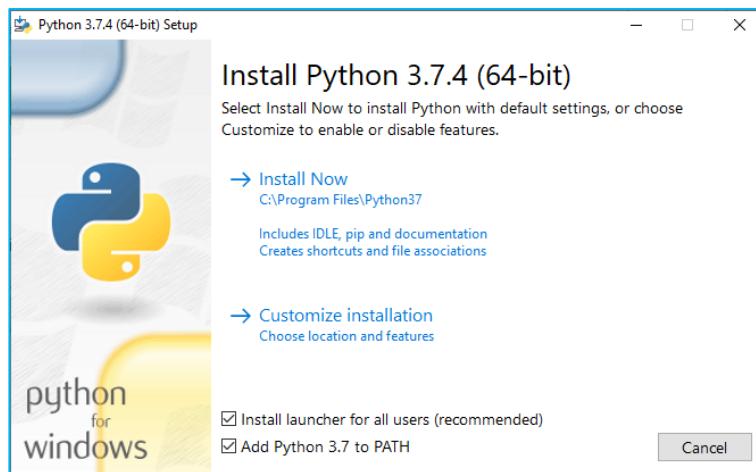
**Figure 2-39:** SUMO Installation begins

Once the installation is completed, click on the **Finish** button.



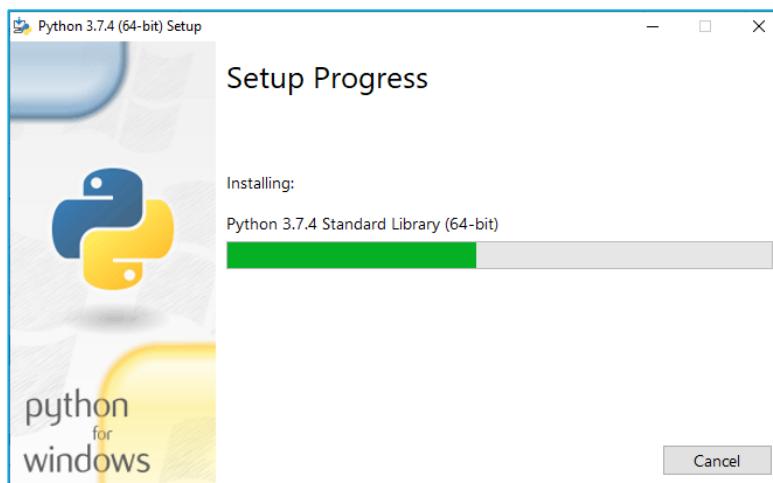
**Figure 2-40:** Select Finish button

And then, Install Python 3.7.4 (64-bit) window appears. To start Python software installation, click on **Install Now** option.



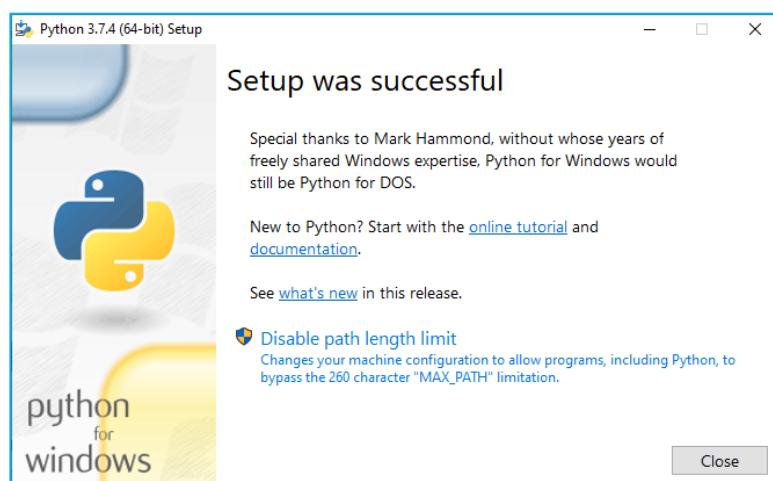
**Figure 2-41:** Install Python 3.7.4 (64-bit) window

Python installation begins.



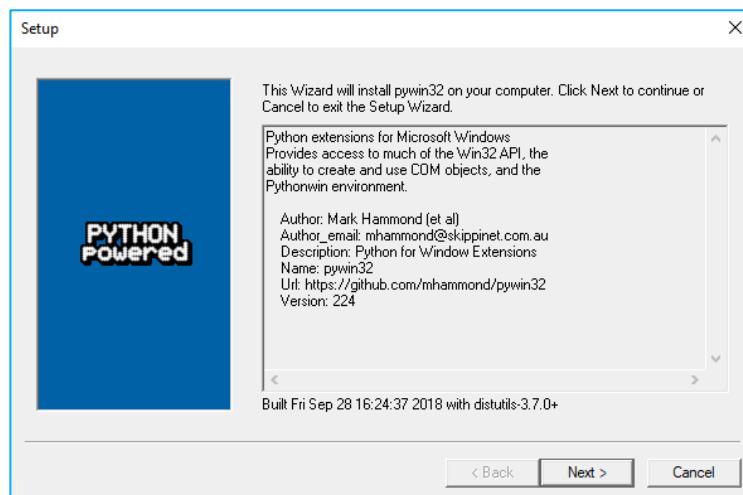
**Figure 2-42:** Python installation

A Setup was successful message will be displayed. Click on **Close** button.



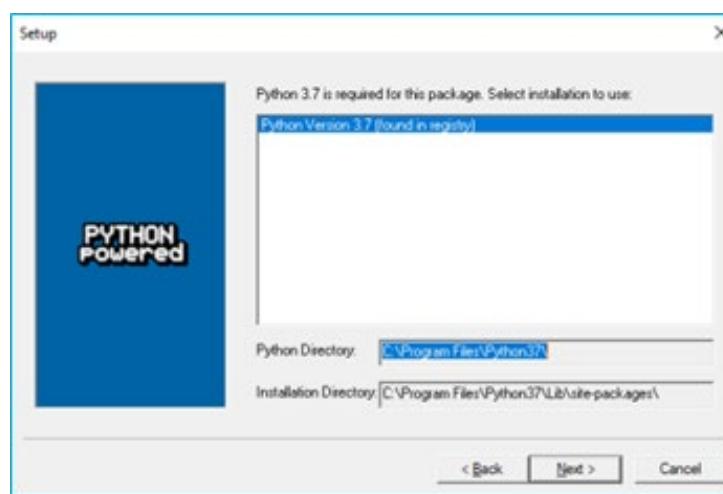
**Figure 2-43:** Select Close Install Python 3.7.4 (64-bit)

This will take you to the **pywin 32-224** installation wizard. To install Pywin32, Click on **Next** button.



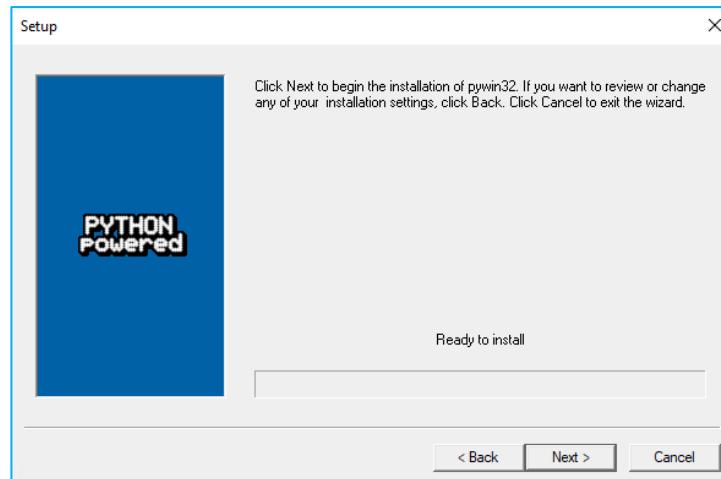
**Figure 2-44:** pywin 32-224 installation wizard window

Select the Python directory and Click on **Next** button.



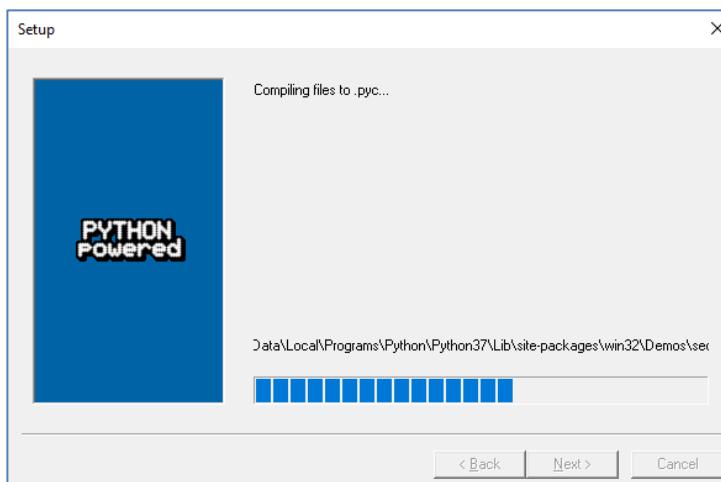
**Figure 2-45:** Python directory path

The Setup is ready to install. Click on **Next** button.



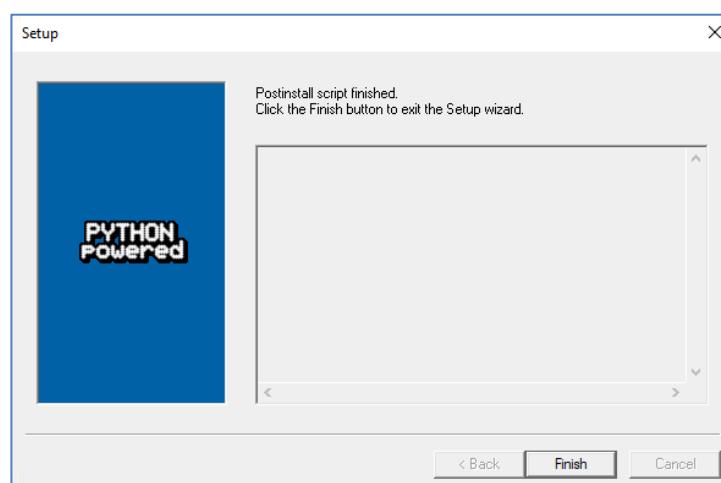
**Figure 2-46:** Select Next button

Pywin32 installation begins.



**Figure 2-47:** Pywin32 installation begins

Once Pywin installation is complete, click on **Finish** button.



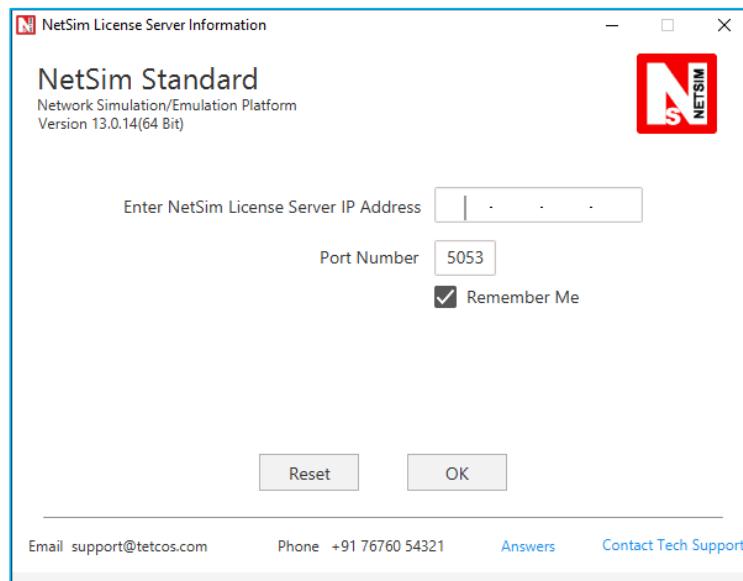
**Figure 2-48:** Pywin installation to Finish button

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



**Figure 2-49:** 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.



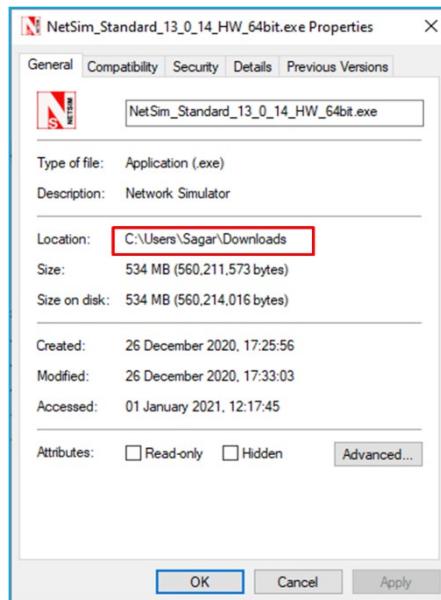
**Figure 2-50:** Enter NetSim License Server IP Address

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. (To set up NetSim License Server - Refer to Section 2.3.1 – Installing NetSim RLM Dongle Driver Software (Dongle Based Licenses).) Click on OK button. Once this is done, NetSim Home screen will appear.

## 2.2.1 Silent installation

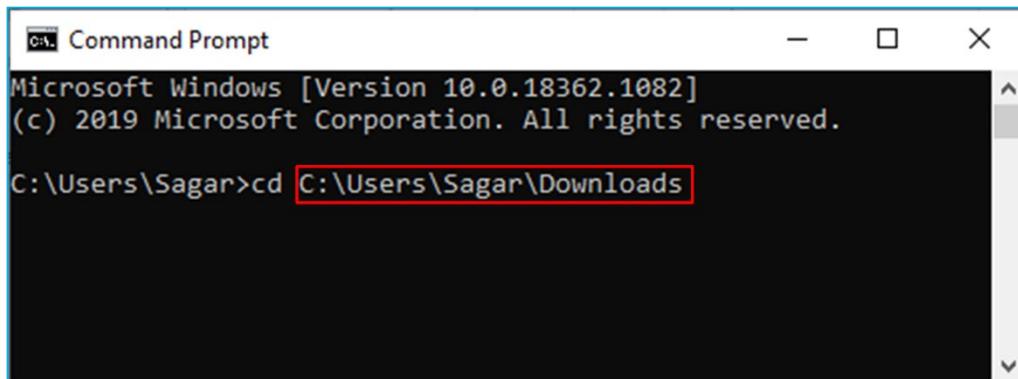
Steps for silent installation in NetSim are as follows.

- For example, let us take the **NetSim\_Standard\_13\_0\_14\_HW\_64bit.exe** setup. Right click on NetSim Standard 64-bit setup → Go to properties and copy the **Location** as shown below.



**Figure 2-51:** NetSim Standard 64-bit setup location

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



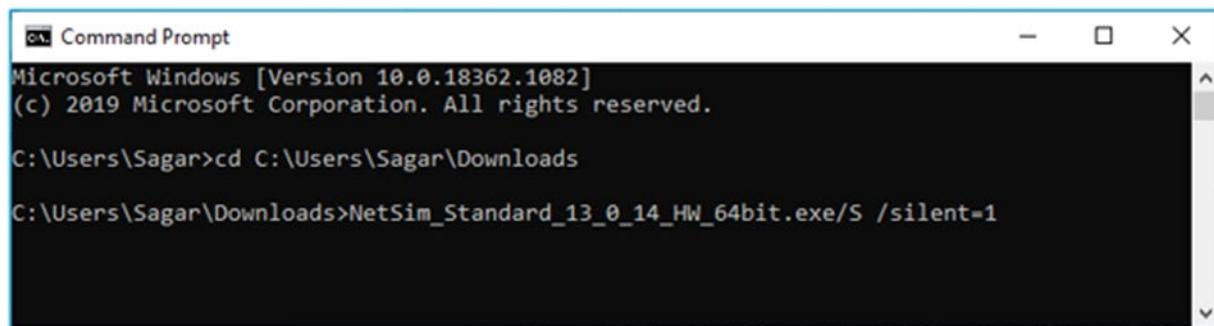
**Figure 2-52:** Enter setup location in command prompt

- Run/Execute Command with the following parameters:

**NetSim\_Standard\_13\_0\_14\_HW\_64bit.exe/S /silent=1**

**><setup location/S<space>/silent=1**

- silent=1:** It will install NetSim and third-party tools silently.
- /S:** It will Install NetSim itself silently.



```
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

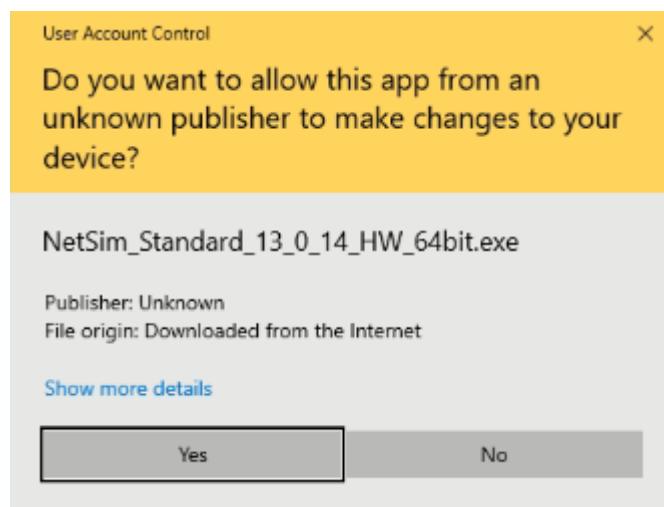
C:\Users\Sagar>cd C:\Users\Sagar\Downloads

C:\Users\Sagar\Downloads>NetSim_Standard_13_0_14_HW_64bit.exe/S /silent=1
```

**Figure 2-53:** Silent installation command in command prompt

4. Press the **Enter** key. The following User Account Control message window appears.

Click on **Yes** button to begin silent installation of NetSim.

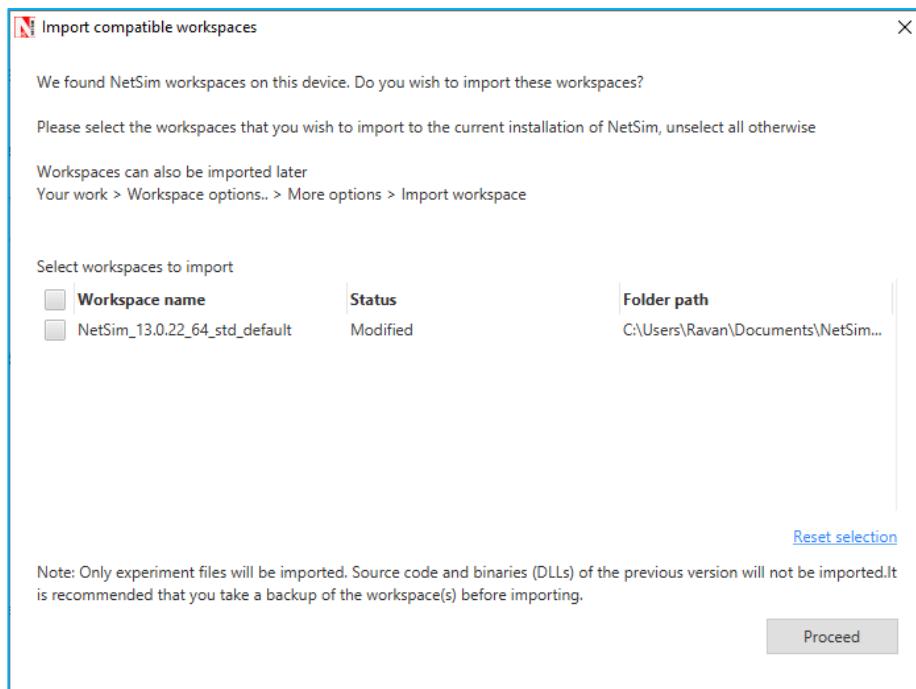


**Figure 2-54:** User Account Control message window appears and select Yes

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

## 2.2.2 Import Compatible Workspaces

After successful installation, NetSim automatically detects if any workspaces (of prior versions/installations) are present. If yes, NetSim opens a window named, “Import Compatible workspaces”, via which users can import these workspaces into new version of NetSim.

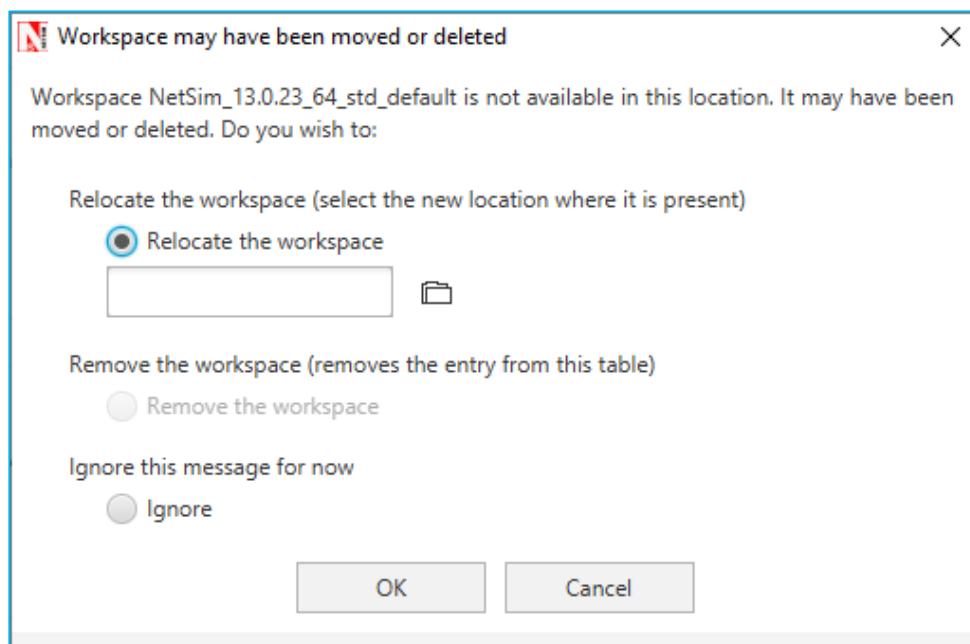


**Figure 2-55:** Import Compatible Workspaces window after installation.

Check/Uncheck suitably and click on Proceed to import compatible workspaces into newer version of NetSim.

A default workspace will be created and set as current workspace if none of the compatible workspace are selected and clicked on Proceed.

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



**Figure 2-56:** Relocate the workspace/Remove the workspace/Ignore

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 button.

## 2.3 Setting up License Server

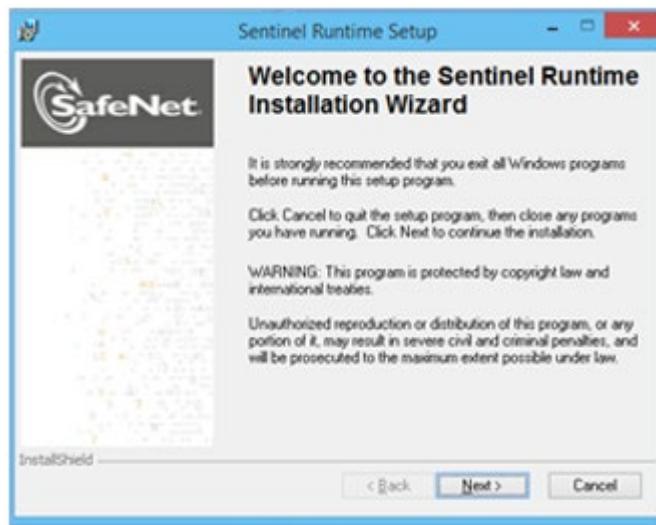
### 2.3.1 Installing NetSim RLM Dongle Driver Software (Dongle Based Licenses)

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.

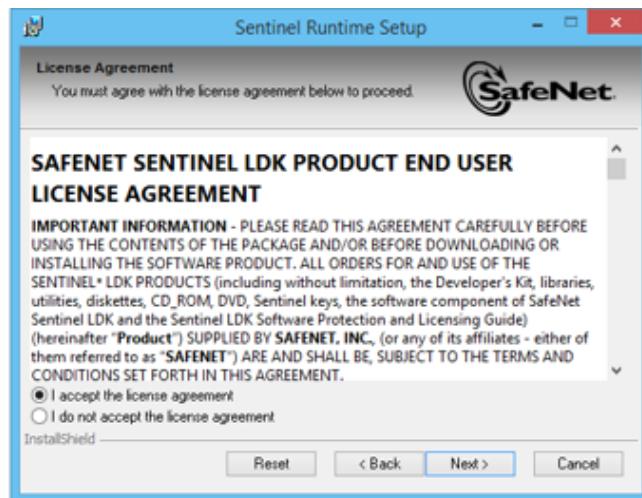
Setup prepares the installation wizard and the driver software installation begins with a Welcome Screen. Click on **Next** button.



**Figure 2-57:** Sentinel Runtime Setup window and select Next button

**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 button else quit the setup by clicking Cancel button.

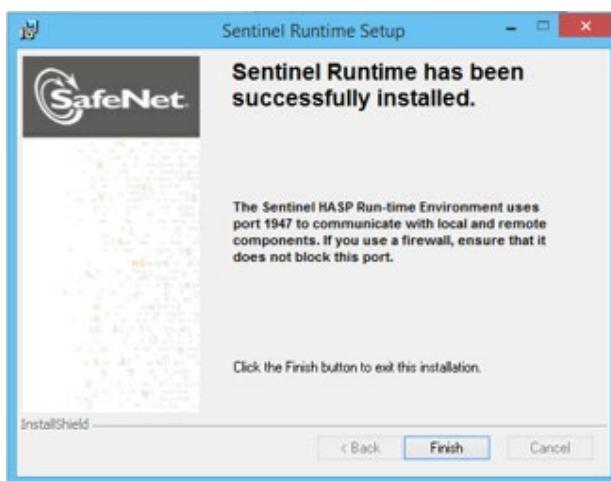


**Figure 2-58:** Sentinel Runtime Setup License Agreement window appears and select Next button  
The installation process begins.



**Figure 2-59:** Installation process begins

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



**Figure 2-60:** Sentinel Runtime is installed successfully and select Finish button

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 2-61**). If the driver is not properly installed, this light will not glow when the dongle is connected to the USB Port.



**Figure 2-61:** Connecting the Dongle in the USB Port

### 2.3.2 Running NetSim License Server

- Copy the **NetSim License Server** folder and paste it on **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.

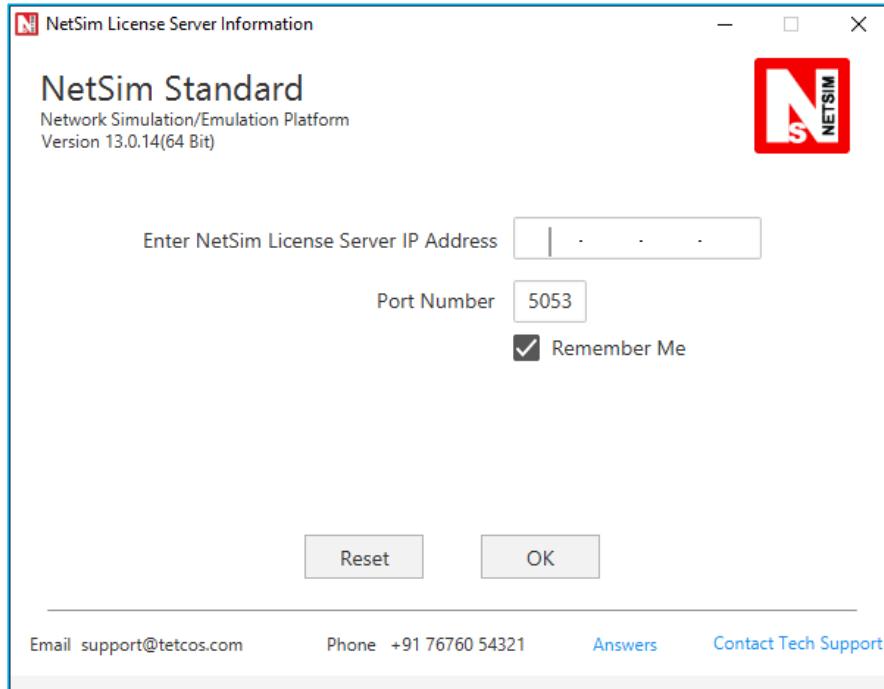
While running **rlm.exe**, the screen will appear as shown below **Figure 2-62**.

```
C:\Users\Tetcos\Desktop\NetSim_License_Server\rlm.exe
02/18 12:18 <rlm> RLM License Server Version 11.2BL2
Copyright (C) 2006-2014, Reprise Software, Inc. All rights reserved.
02/18 12:18 <\lm> License server started on Tetcos-PC
02/18 12:18 <\lm> Server architecture: x86_w3
02/18 12:18 <\lm> License files:
02/18 12:18 <\lm> netsim_std_1.lic
02/18 12:18 <\lm> netsim_std_2.lic
02/18 12:18 <\lm> netsim_std_3.lic
02/18 12:18 <\lm> netsim_std_4.lic
02/18 12:18 <\lm> netsim_std_5.lic
02/18 12:18 <\lm> netsim_std_6.lic
02/18 12:18 <\lm> netsim_std_7.lic
02/18 12:18 <\lm> netsim_std_8.lic
02/18 12:18 <\lm>
02/18 12:18 <\lm> Web server starting on port 5054
02/18 12:18 <\lm> Using TCP/IP port 5053
02/18 12:18 <\lm> ... adding UDP/IP port 5053
02/18 12:18 <\lm> Starting ISU server tetcos on port 49623
02/18 12:18 <\lm> New thread created to watch ISU tetcos
02/18 12:18 <tetcos> RLM License Server Version 11.2BL2 for ISU "tetcos"
02/18 12:18 <tetcos> Server architecture: x86_w3
Copyright (C) 2006-2014, Reprise Software, Inc. All rights reserved.
```

**Figure 2-62:** 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 2-63:** Enter NetSim License Server IP Address

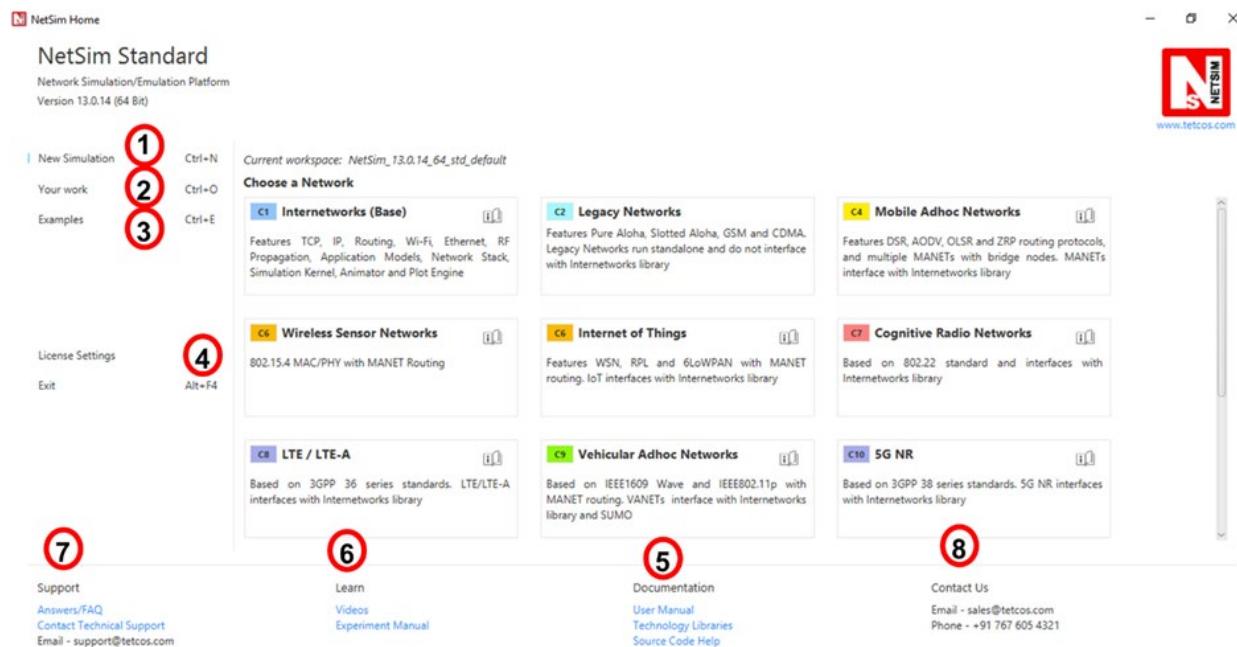
# 3 NetSim GUI

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, Animation Window and Plots Window.

## 3.1 Menus in the NetSim Home Screen

You see the NetSim Home Screen when you run NetSim software for the first time, after checking out a license from the NetSim License Server.

See the following image for an example of the NetSim Home screen as shown below **Figure 3-1.**



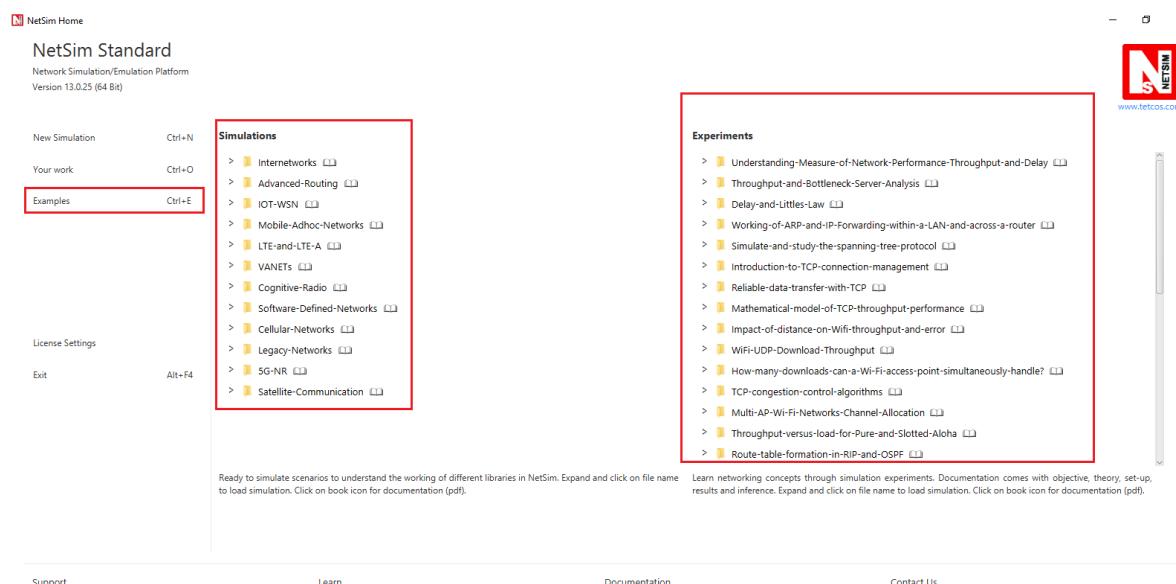
**Figure 3-1:** NetSim Home screen

You see the following items on the NetSim Home screen:

- New Simulation:** Use this menu to simulate different types of networks in NetSim. You can simulate the following the types of networks: Internetworks, Legacy Networks, Mobile Adhoc networks, Cellular Networks, Wireless Sensor Networks, Internet of Things, Cognitive Radio Networks, LTE/LTE-A Networks, 5G NR (newly added component in v12), VANETs, and Satellite Communication. Only the networks (components) for which licenses are available will be shown. The networks (components) shown at the bottom with grey background cannot be directly clicked

and entered. These features can be accessed through other components given the dependencies.

2. **Your work:** Use this menu to load saved configuration files from the current workspace. You can view, modify or re-run existing simulations. Along with this, users can also export the saved files from the current workspace to their preferred location on their PC's.
3. **Examples:** Use this menu to perform simulations of different kinds categorized technology-wise. Users can choose any network which they want to work and further go down by using a double click on it or by a click on the arrow pointer which will take you to the next level. By a click on any simulation file will open a pre-existing simulation file which users can run and analyze the results. Users can click on the book icon present in the right-hand side of each network which opens the corresponding pdf files. This helps the users with all information about the current simulation as well as the entire network technology.

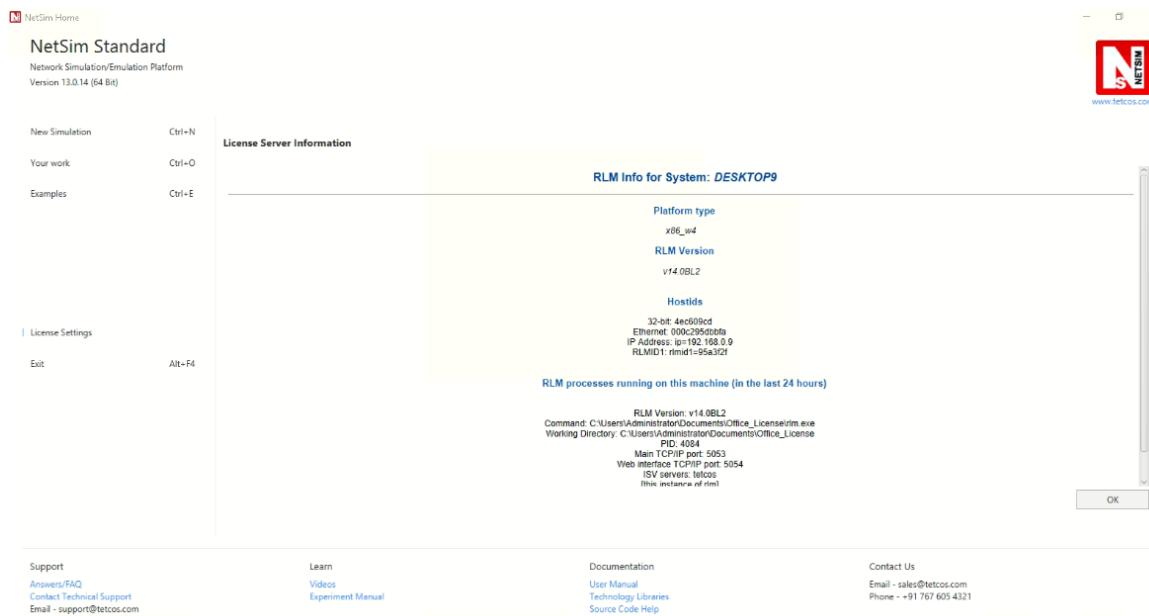


**Figure 3-2: Featured Examples and Experiments list window**

Similarly, on the other side, users can find experiments section which has various experiments covering all the technologies in NetSim. Users can choose their experiment by either a double click on it or by a click on the pointer arrow which will take you the samples. Click on the sample to open the particular experiment in NetSim. All the settings to carry out the experiment are already done. Users can click on the book icon present in the right-hand side of each experiment. This will open the corresponding pdf file for the experiment which consists of detailed description of that experiment.

**4. License Settings:** Use this menu to perform the following. Click on License Settings provides users with three sub-menus related to License information.

**License Server Information:** Use this menu to view details about the NetSim License Server from where the client is checking out licenses.



**Figure 3-3: NetSim License Server Information window**

You will see the following details on the NetSim Home screen, if you click the License Server Info menu item: the type of platform on which NetSim is running, the version of RLM, the Dongle RLM ID, the IP address of the NetSim License Server, and the path to the license files in the server hosting NetSim License Server.

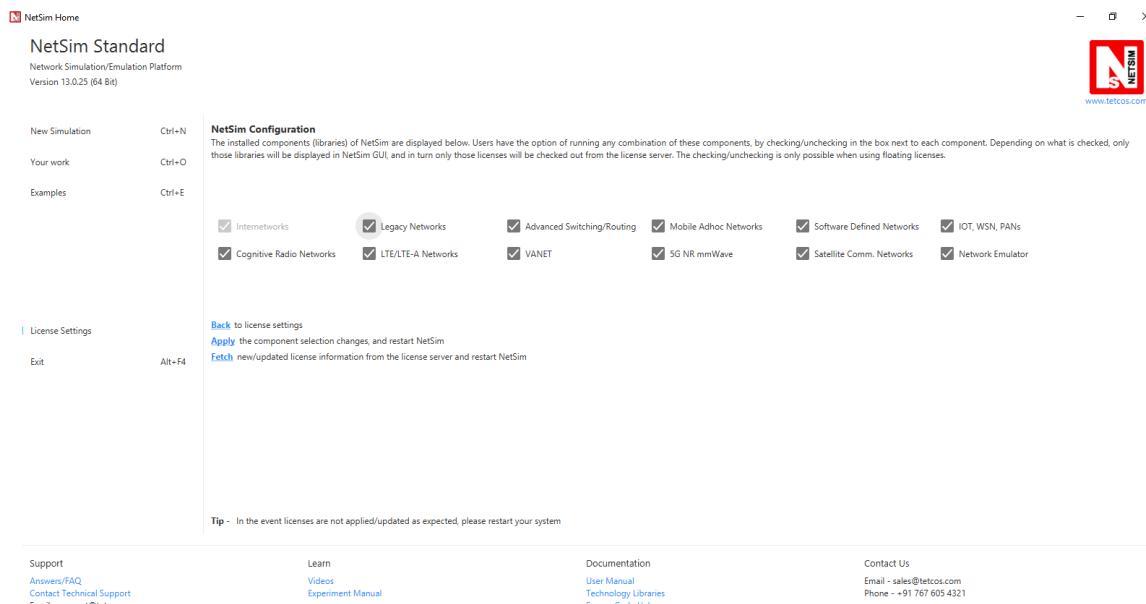
**End User License Agreement:** Use this menu to view the end user license agreement. You will see the following details on NetSim Home screen, if you click the End User License Agreement menu item: Grant of License and Use of the Services, License Restrictions, License Duration, Upgrade and Support Service etc.

**Configure Installed Components/Libraries:** Use this menu to allow NetSim users to simulate only specific types of networks (by the licenses and libraries associated with the types of networks). You control access to types of networks by selecting libraries for specific types of networks that NetSim License Server checks out when NetSim users start NetSim.

NetSim Home screen displays libraries for components for which you have purchased licenses.

**Note:** You can select or clear libraries and control access to NetSim users, only if you are using floating licenses.

See the following image for an example of what the NetSim Home screen displays, if you click the Configure Installed Components/Libraries menu item.



**Figure 3-4:** The Installed Component (Libraries) of NetSim

Use the License Settings menu as follows:

- Select the checkboxes for the component libraries (types of networks) that NetSim users must be allowed to simulate.
- Clear the checkboxes for the component libraries (types of networks) that NetSim users must not be allowed to simulate.

The Internetwork component is greyed out. You cannot clear the Internetworks component because Internetworks is a base component that is required for all the other components to work.

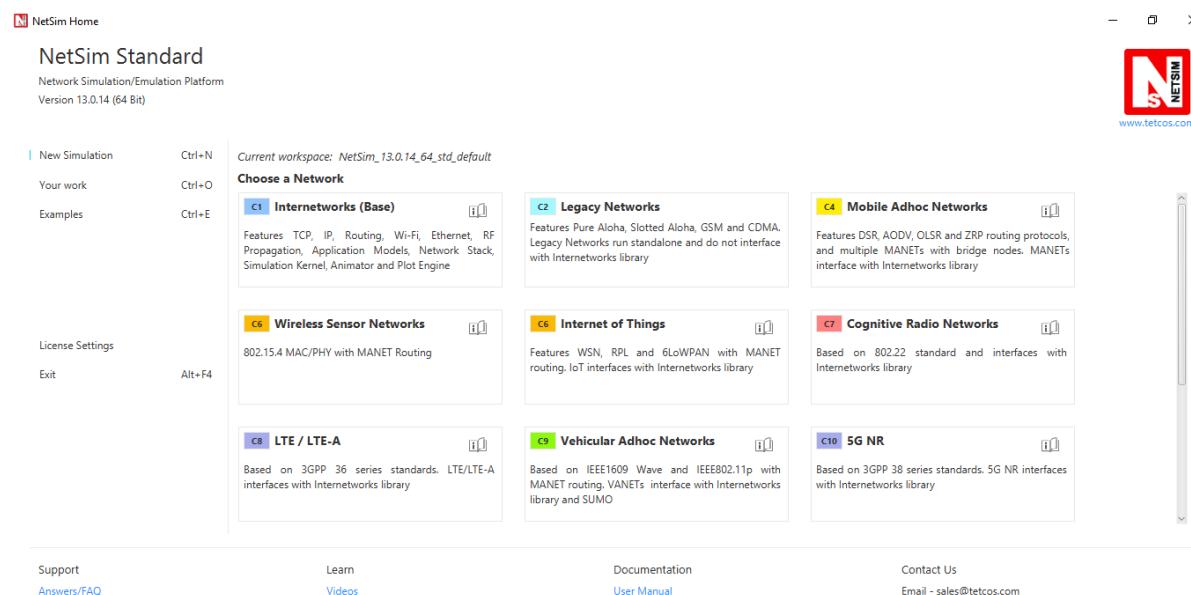
5. **Documentation:** Use this section to open the following NetSim help documents: These include the **User Manual** which consists of complete description about all the features in NetSim and how it can be used by the end users, the **Technology Libraries** which provides users with an access to a detailed description of various Network Technologies present in NetSim through individual pdf files, and **Source code help** which comes along with Standard and Pro Versions of NetSim, allows users to gain a better understanding of the underlying code structure for in-depth analysis.
6. **Learn:** Use this section to learn more about the software which includes the following: **Videos** section can be used to view videos related to NetSim in **TETCOS** YouTube channel. This channel helps users by providing frequent updates on what's new in NetSim, topics related to various network technologies covering different versions of

NetSim, and monthly webinars. The ***Experiments Manual*** section grants you access to a well-designed experiments manual covering various networking concepts which helps users to easily understand different networks and also gain ideas to carry out their own experimentations in NetSim.

7. **Support:** Use this section to reach TETCOS helpdesk. **Contact Technical Support** link can be used to raise a **trouble ticket**, you can also write to us via **Email** to [support@tetcos.com](mailto:support@tetcos.com), and **Answers/FAQ** link grants you access to our **Knowledge Base** which contains answers to all your questions most of the time. Users can utilize the wealth of information present in it, which are further classified into the following: FAQs, Technologies/Protocols, Modelling/UI/Results, and Writing your own code in NetSim.
8. **Contact Us:** Use this section to contact us and know more information about our product. You can write to us via **Email** to [sales@tetcos.com](mailto:sales@tetcos.com) or contact us via **Phone** to our official number **+91 76760 54321**.
9. **Website:** Use this link [www.tetcos.com](http://www.tetcos.com) to visit our website which consists of vast information that will assist you through all walks of NetSim.

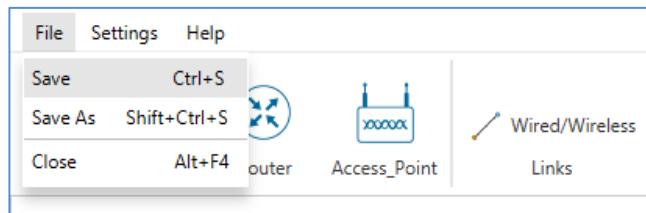
### 3.1.1 Creating “New” Simulations

The Simulation window loads up once user selects the desired network technology from the New Menu. Click on New Simulation and select the desired kind of network to simulate.



**Figure 3-5: NetSim Home Screen**

## Save



**Figure 3-6:** To save experiment, Select File >Save. Save As etc

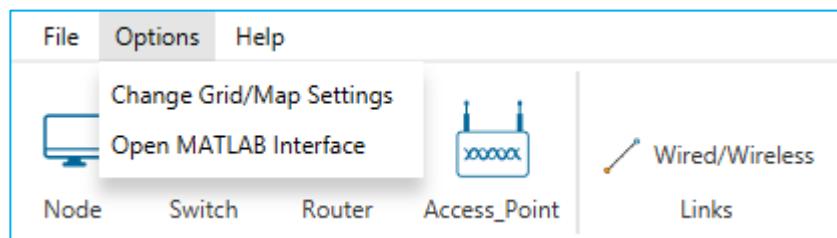
To save experiment, select File → Save, then specify the Experiment Name, Description (Optional) and click Save. The short cut for the same is Ctrl + S.

## Save as

To save an already existing/saved experiment by a different name after performing required modifications/changes to it (without overwriting the existing copy), Save As option can be used. Select File → Save As, then specify the Experiment Name, Description (Optional) and click Save. The short cut for the same is Shift + Ctrl + S and F12.

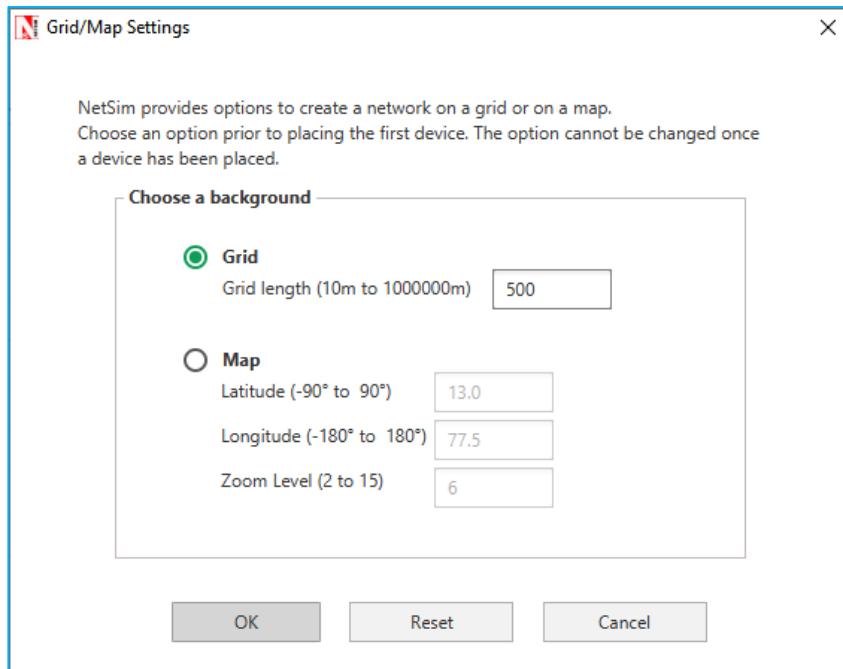
### 3.1.2 Environment Settings

The settings menu provides user's access to the simulation environment settings.



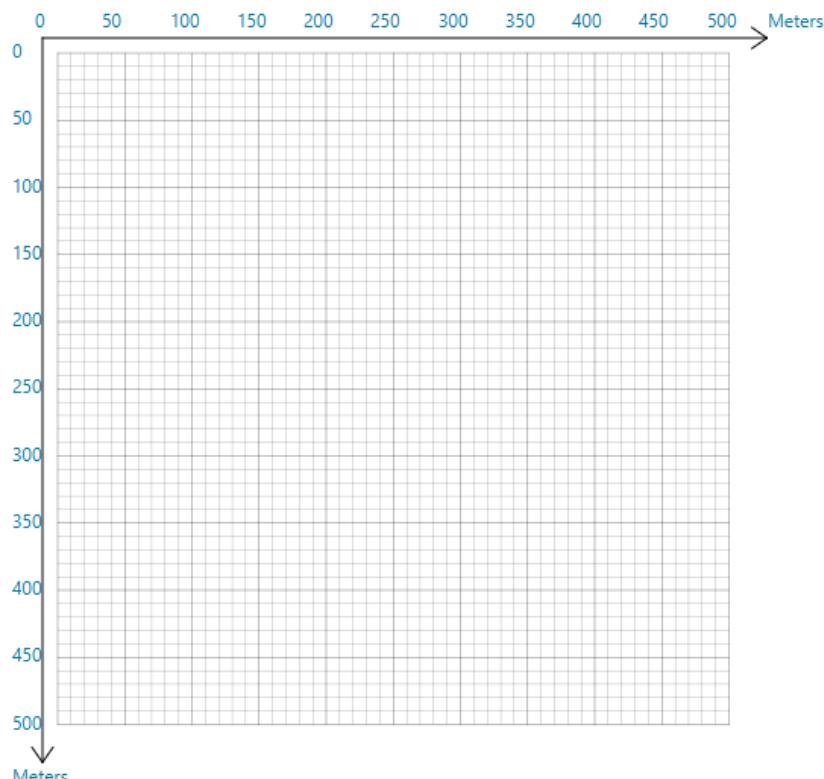
**Figure 3-7:** Environment settings

The Environment Settings window is used to switch between Grid View and Map View backgrounds in supported network technologies. For Grid view, users can configure the Grid environment length in meters as shown **Figure 3-8**.

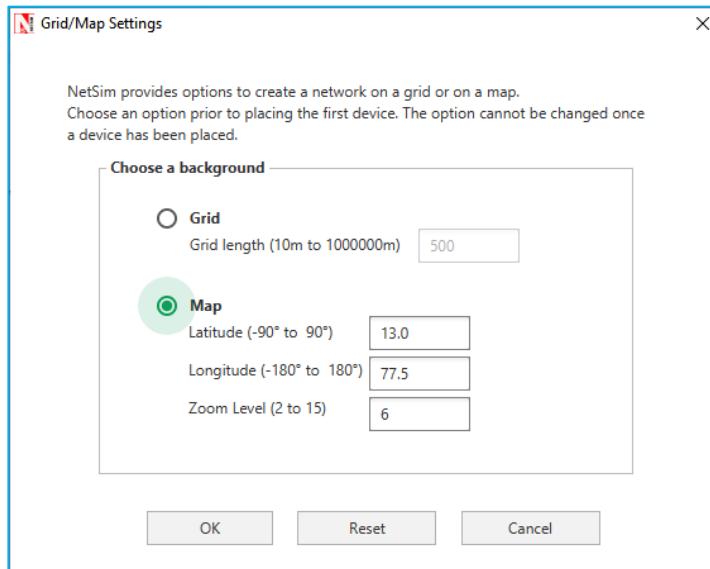
**Figure 3-8:** Grid View Setting

### 3.1.2.1 The Gird

The Grid coordinate system has its origin at the upper left of the drawing area, and positive Y is down while positive X is to the right.

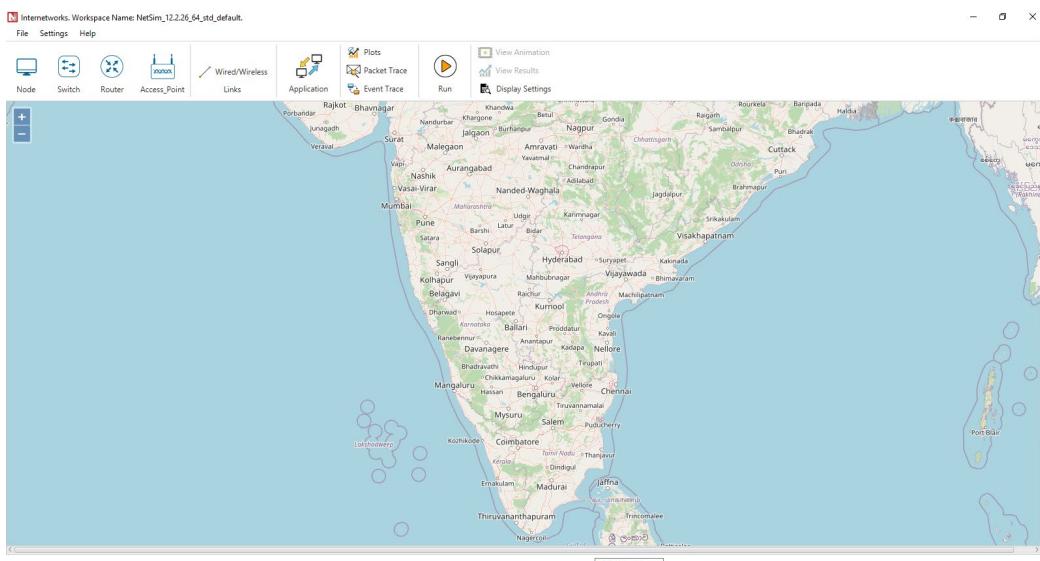
**Figure 3-9:** NetSim Design window

For Map view users can configure the latitude and longitude respectively.



**Figure 3-10: Map View Setting**

Users can zoom in and out of the map to add devices in specific geographical locations.



**Figure 3-11: Map Design window**

## Learn

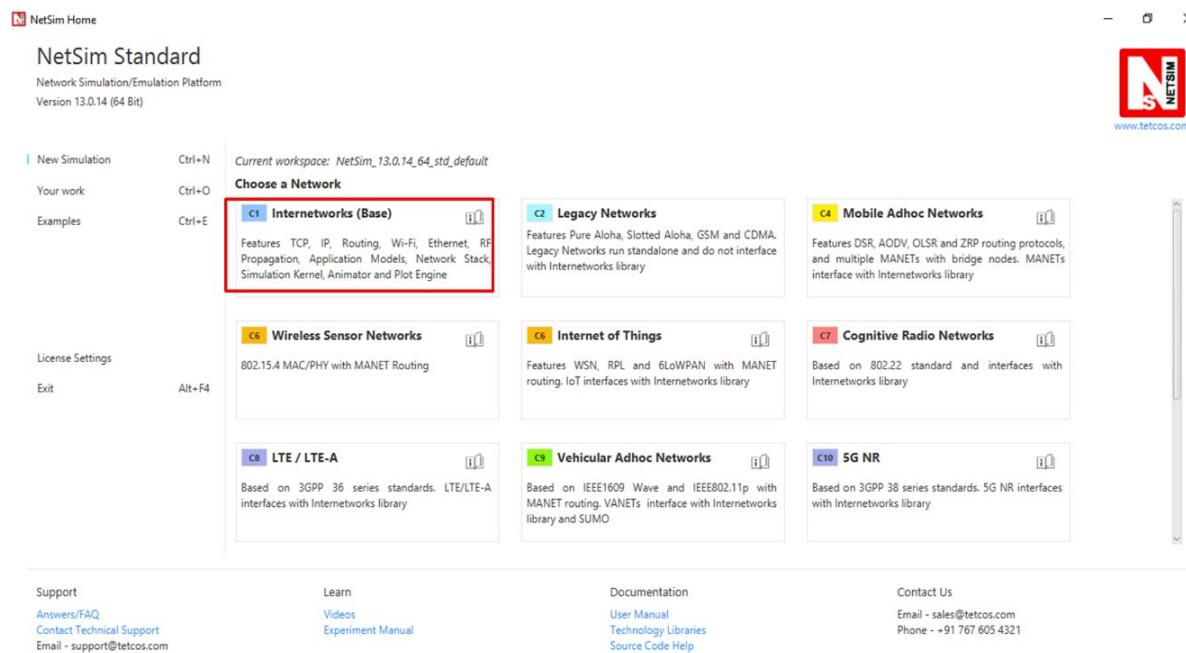
This menu contains link to NetSim Videos on TETCOS YouTube Channel and NetSim Experiments manual.

## Documentation

This menu contains link to NetSim User Manual, Technology Libraries and NetSim Source Code Help.

## 3.2 Modeling and Simulating a simple network

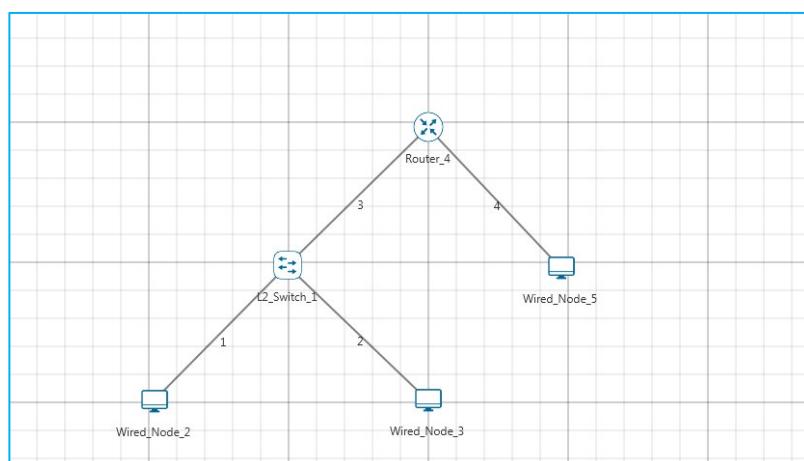
This section will demonstrate how to create a basic network scenario and analyze the results. Let us consider Internetworks. To create a new scenario, Go to **New Simulation → Internetworks** as shown below **Figure 3-12**.



**Figure 3-12:** NetSim Home Screen

### 3.2.1 Creating a Network scenario

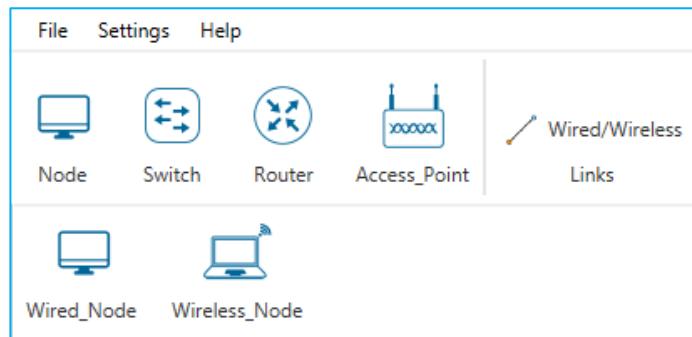
In this example, a network with two subnets is designed. Let us say the subnet 1 consists of two wired nodes connected via a Switch and the other subnet consists of one wired node. Both the subnets are connected using a Router. Traffic in the Network flows from a wired node in subnet 1 to the wired node in subnet 2.



**Figure 3-13:** Network Topology in this experiment

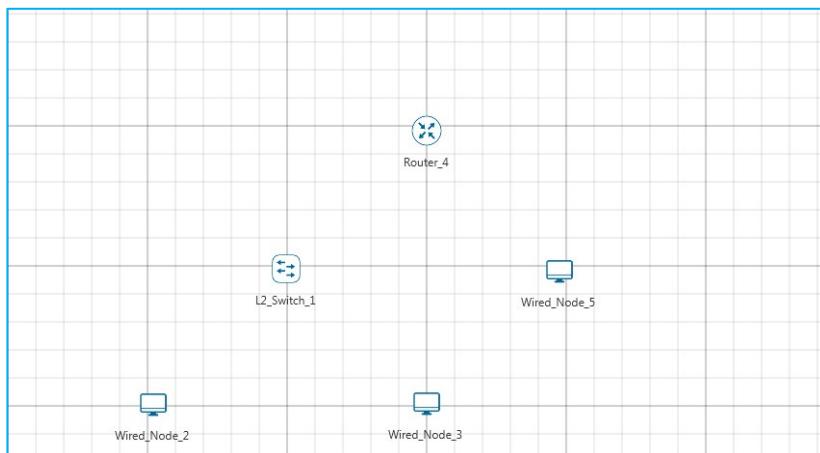
Perform the following steps to create this network design.

**Step 1: Drop the devices.** Click on Node icon and select → Wired Node.



**Figure 3-14:** Internetworks Device Palette in GUI

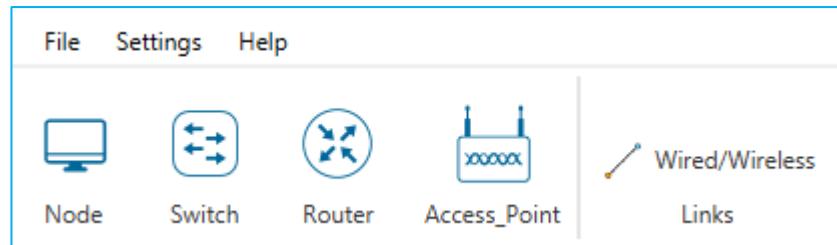
Click on the environment (the grid) where you want the Wired Node to be placed. In this way, place two more wired nodes. Similarly, to place a Switch and a Router, click on the respective device and click on the environment at the desired location.



**Figure 3-15:** Dropped Devices on GUI

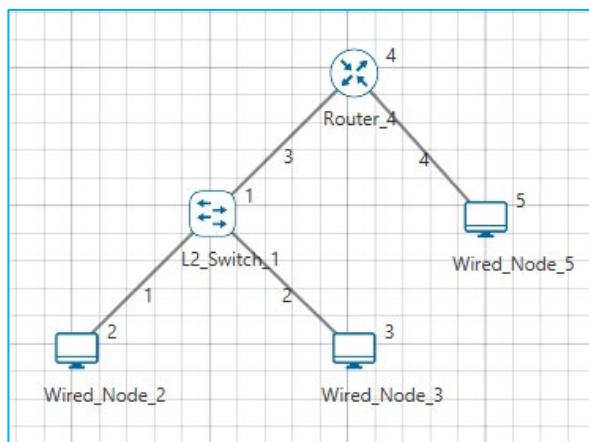
Note that NetSim takes the (x, y) position of any device on the grid is the position of top left corner of the icon and not the center of the icon.

**Step 2: Connecting devices:** Select the link and then left click on one device, free the mouse button, then click on the second device and free the mouse button. The wired links may disappear if you right click anywhere in the environment. Clicking and dragging without freeing the mouse pointer would displace the device in the environment.



**Figure 3-16:** To Connect devices select wired/wireless links

For example, select link and the click on Switch followed by router to connect them. In this manner, continue to link all devices.

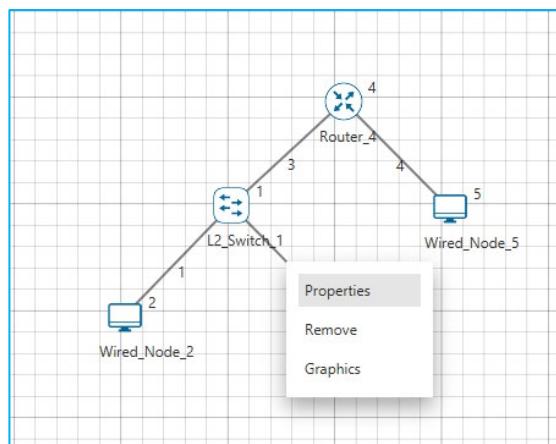


**Figure 3-17:** Network Topology

### 3.2.2 Configuring devices and links in the scenario

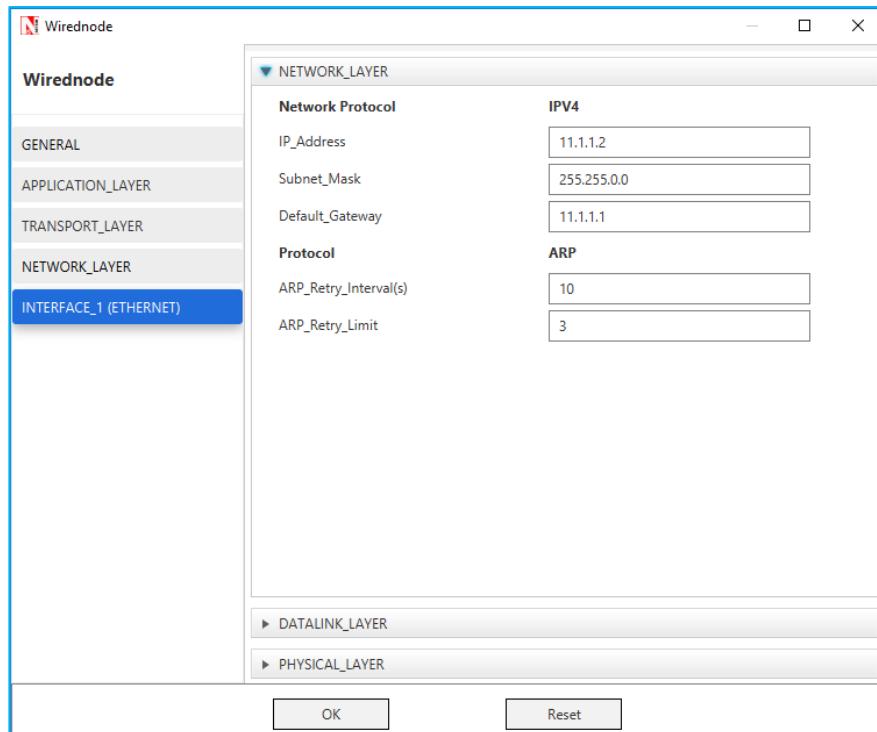
**Step 1:** To configure any device, right click on the device and select properties as shown

**Figure 3-18.**



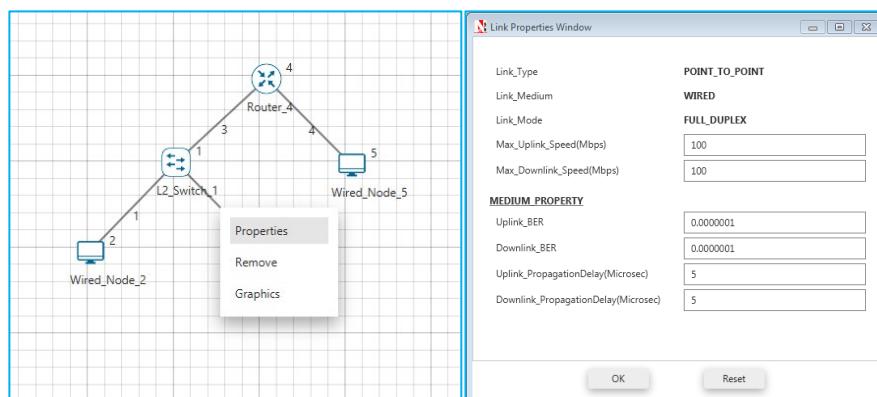
**Figure 3-18:** Right click on the device and select properties

The default properties of any device can be modified per requirement. Then click on OK.



**Figure 3-19:** Network layer Properties window for wired node

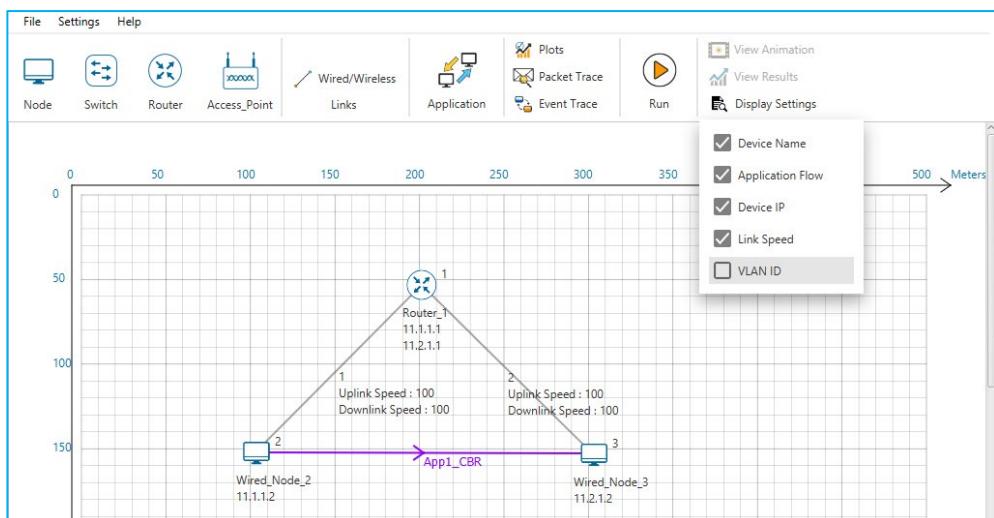
Step 2: To configure the links, right click on any Link and select Properties as shown **Figure 3-20.**



**Figure 3-20:** Wired Link properties window for links

### 3.2.3 Display Settings

In NetSim, users can Turn-On or Turn-Off display information such as IP Address of the devices, link speed etc. For doing this click on Display settings as shown below **Figure 3-21.**



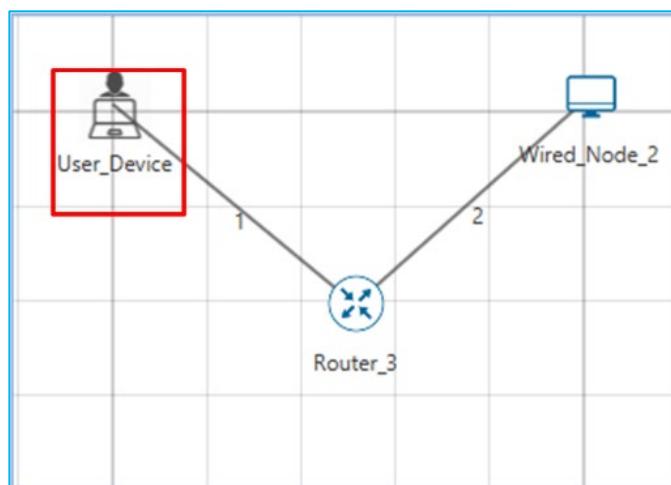
**Figure 3-21:** Turn-On or Turn-Off display information such as IP Address of the devices, link speed etc

In NetSim the device ID serves as a “device identifier” while the IP Address is an “Interface identifier”

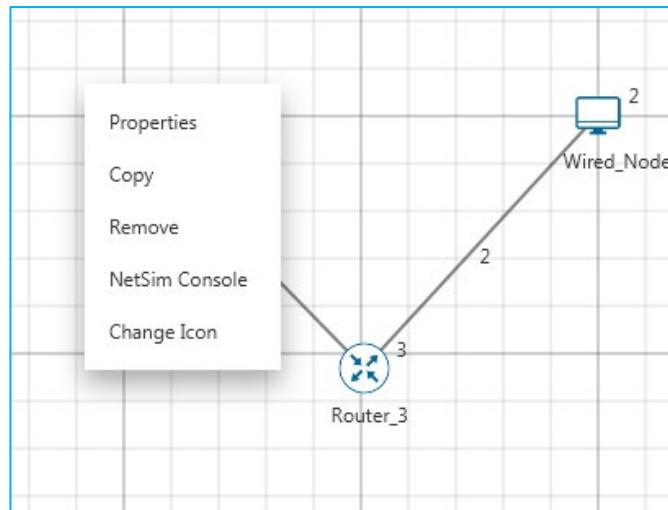
### 3.2.4 Copy/Paste

In NetSim simple copy paste can be used. Using this feature users can copy all the properties of a device and create a new device with similar properties.

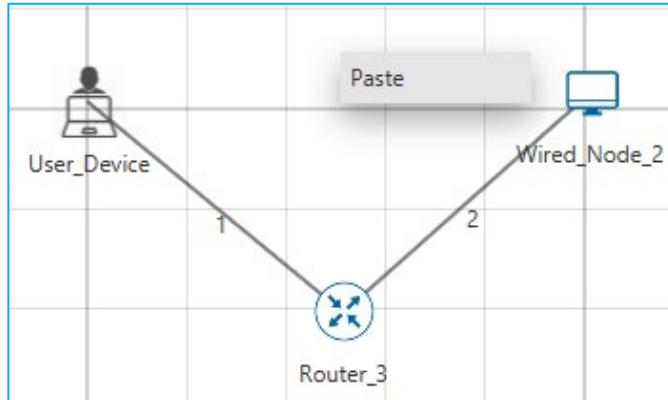
Right click on the device, click on copy and then right click and click paste. The sequence is shown below **Figure 3-22/Figure 3-23/Figure 3-24/Figure 3-25/Figure 3-26**.



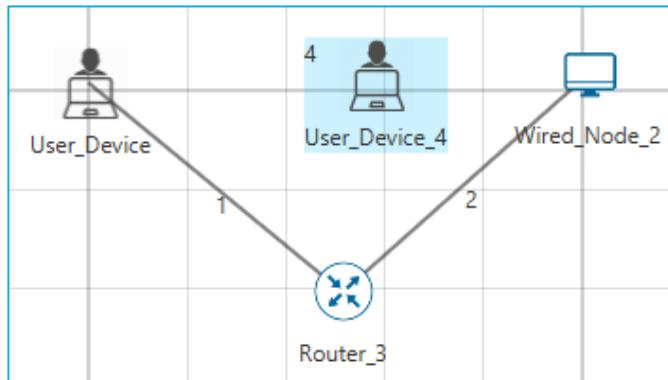
**Figure 3-22:** Devices present on GUI



**Figure 3-23:** Right click on the user device and Select copy

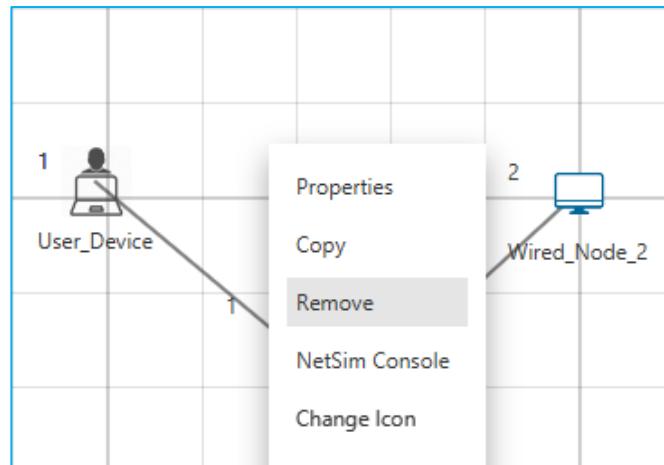


**Figure 3-24:** Right click on the GUI and Select Paste



**Figure 3-25:** Devices Pasted in GUI

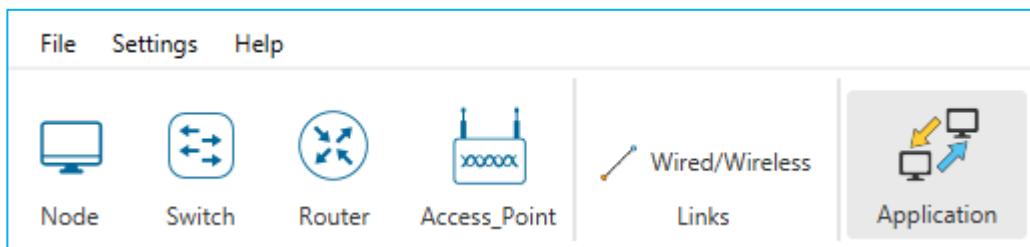
Remove in the device options, is used to delete the device from the grid environment. Given below is an example of removing the device User\_Device\_4 which was previously pasted.



**Figure 3-26:** Right click on User\_Device\_4 and Select Remove

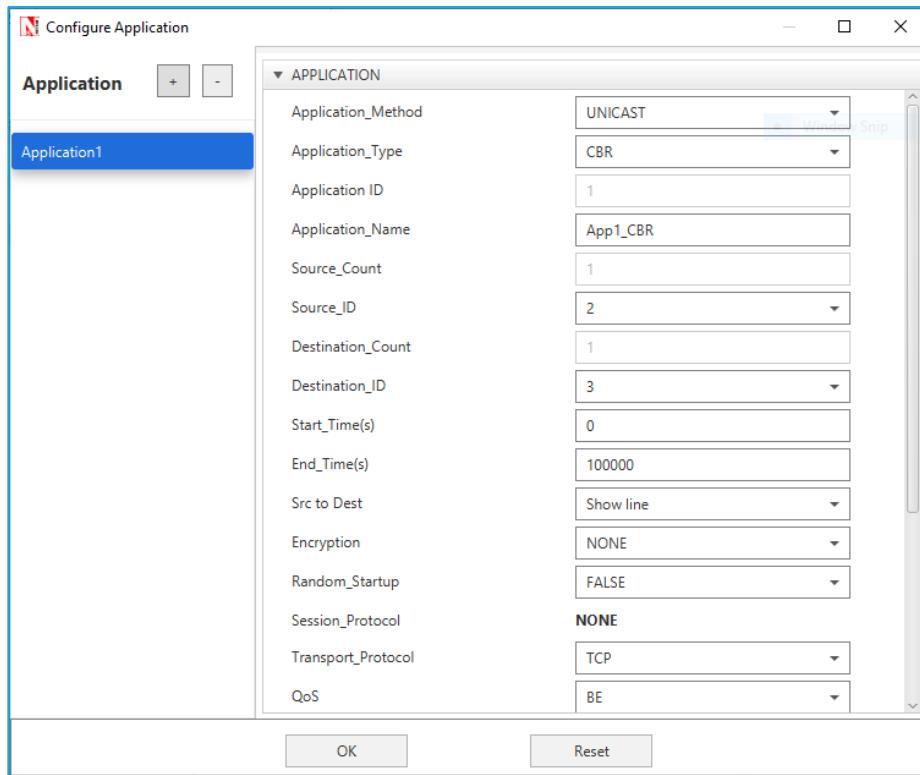
### 3.2.5 Modeling Application Traffic

After the network is configured, user needs to model traffic from Wired Node 2 to Wired Node 3. This is done using the application icon. Click on the Application icon present on the ribbon as shown below **Figure 3-27**.



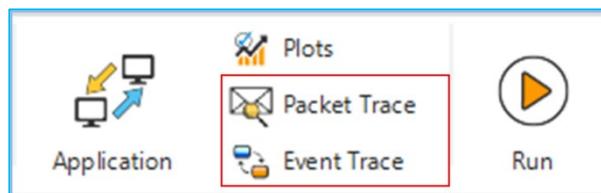
**Figure 3-27:** Select Application icon present on ribbon

In screen shot shown below the Application type is set to CBR, Source\_ID is 2 and Destination\_ID is 3. Click on OK.

**Figure 3-28:** Application Configuration window

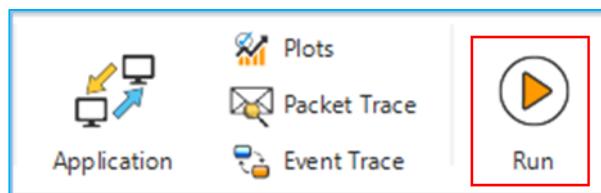
### 3.2.6 Logging Packet/ Event Trace

Packet and Event Trace files are useful for detailed simulation analysis. By default, these are not enabled since it slows down the simulation. To enable logging of Packet Trace / Event Trace click on the icon in the tool bar as shown below. Set the file name and select the required attributes to be logged. For more information, please refer **sections 8.4 and 8.5** respectively.

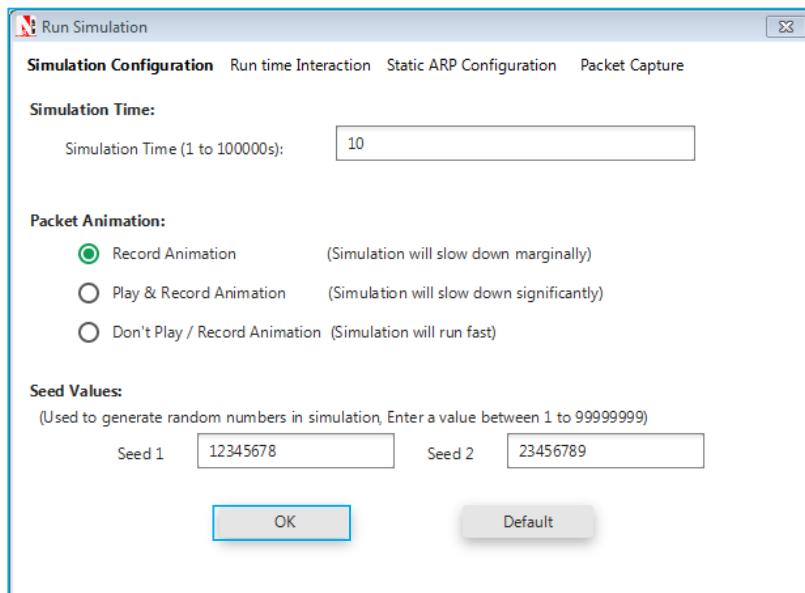
**Figure 3-29:** Packet Trace and Event Trace options present on ribbon

### 3.2.7 Run Simulation

For simulating the network scenario created, click on Run Simulation present in the Ribbon.

**Figure 3-30:** Run Simulation icon in the Ribbon

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



**Figure 3-31:** Run Simulation window

### 3.2.8 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 permit or deny statements that apply to addresses or upper-layer protocols. These lists tell the router what types of packets to: permit or deny. When using an access-list to filter traffic, a permit statement is used to “allow” traffic, while a deny statement is used to “block” traffic.

#### 3.2.8.1 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)

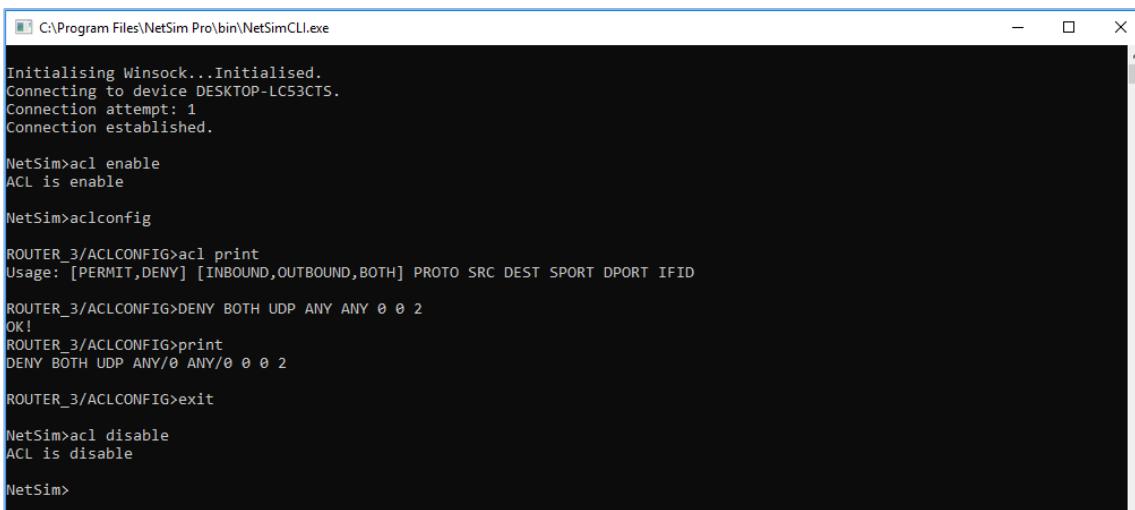
To view ACL usage syntax use: **acl print**

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

### 3.2.8.2 Step to Configure ACL

- To create a new rule in the ACL use command as shown below to block UDP packet in Interface\_2 and Interface\_3 of the Router\_3
- Disable TCP in all nodes (Wired Node and Router)
- 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 500 sec 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 2
OK!
ROUTER_3/ACLCONFIG>DENY BOTH UDP ANY ANY 0 0 3
OK!
ROUTER_3/ACLCONFIG>print
DENY BOTH UDP ANY/0 ANY/0 0 0 2
ROUTER_3/ACLCONFIG>exit
NetSim>acl disable
ACL is disable
NetSim>
```



```
C:\Program Files\NetSim Pro\bin\NetSimCLI.exe
Initialising Winsock...Initialised.
Connecting to device DESKTOP-LC53CTS.
Connection attempt: 1
Connection established.

NetSim>acl enable
ACL is enable

NetSim>aclconfig

ROUTER_3/ACLCONFIG>acl print
Usage: [PERMIT,DENY] [INBOUND,OUTBOUND,BOTH] PROTO SRC DEST SPORT DPORT IFID

ROUTER_3/ACLCONFIG>DENY BOTH UDP ANY ANY 0 0 2
OK!
ROUTER_3/ACLCONFIG>print
DENY BOTH UDP ANY/0 ANY/0 0 0 2

ROUTER_3/ACLCONFIG>exit

NetSim>acl disable
ACL is disable

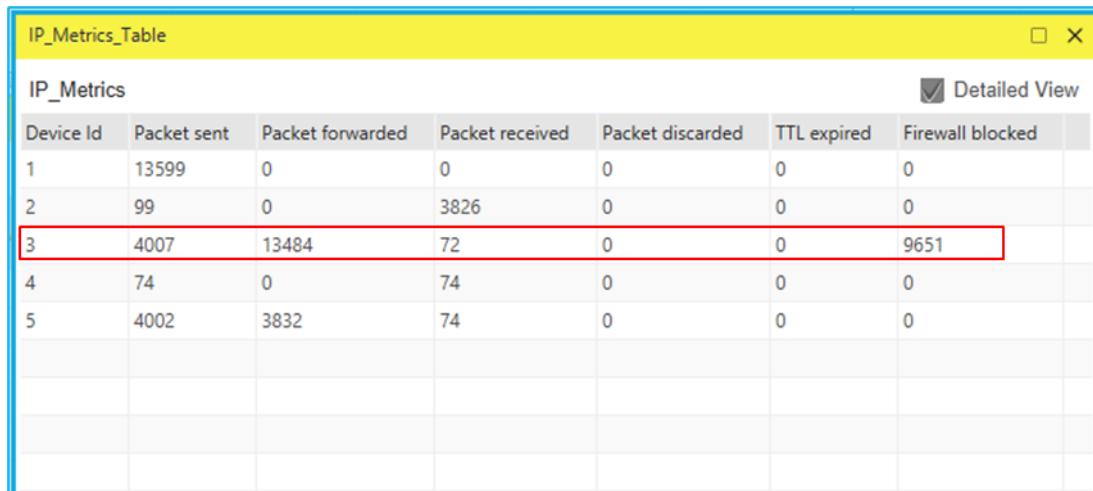
NetSim>
```

**Figure 3-32: ACL Configuration command**

### 3.2.8.3 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

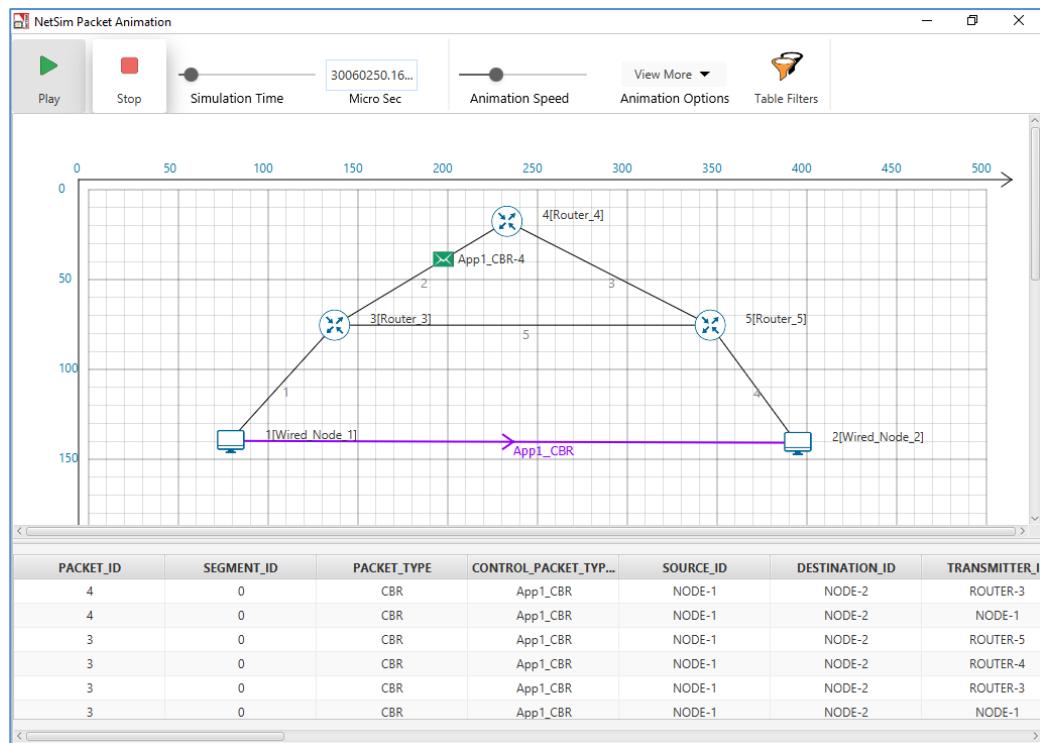
**Note: Results will vary based on time of ACL command are executed**



IP_Metrics_Table							<input checked="" type="checkbox"/> Detailed View
Device Id	Packet sent	Packet forwarded	Packet received	Packet discarded	TTL expired	Firewall blocked	
1	13599	0	0	0	0	0	
2	99	0	3826	0	0	0	
3	4007	13484	72	0	0	9651	
4	74	0	74	0	0	0	
5	4002	3832	74	0	0	0	

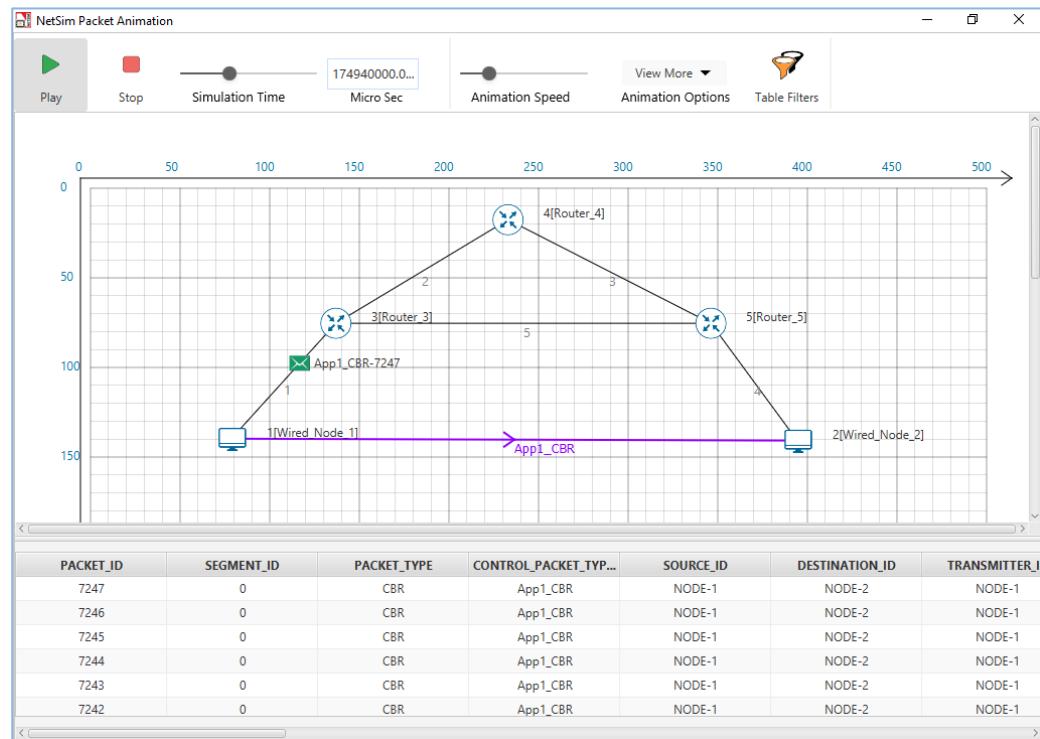
**Figure 3-33: IP Metrics Table in result window**

- Check Packet animation window whether packets has been blocked in Router\_3 or not after entering ACL command to deny UDP traffic.
- Before applying ACL rule there is packet flow from Wired\_Node\_1 to Wired\_Node\_2.



**Figure 3-34:** In Animation Window before applying ACL rules see the packet flow

- After applying ACL rule Packet flows up to Router\_3 only

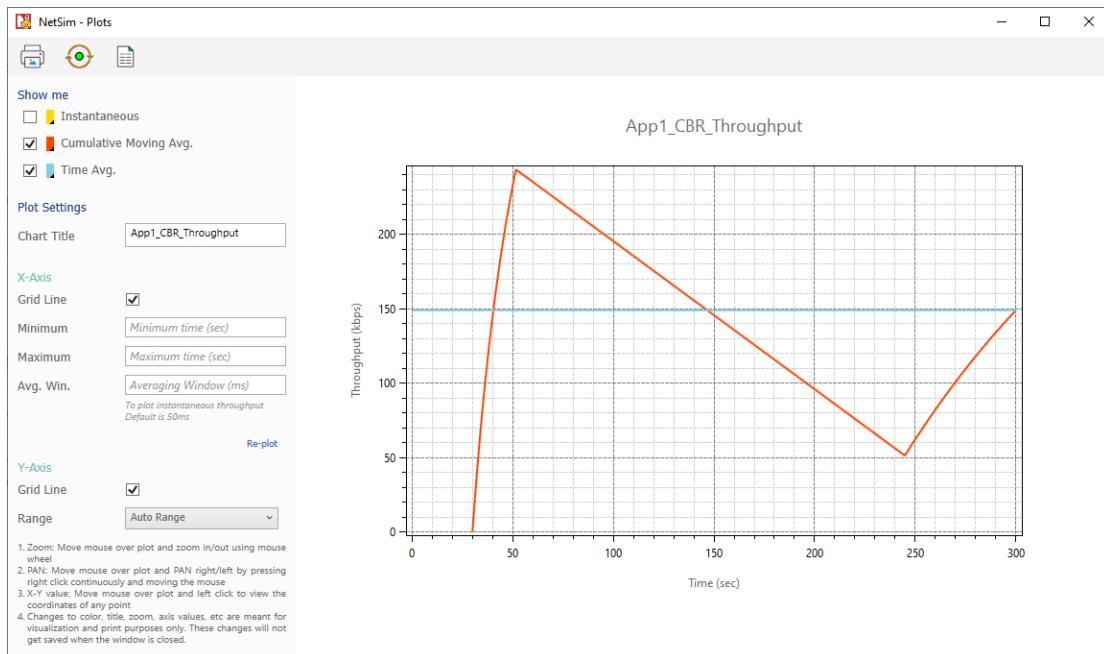


**Figure 3-35:** In Animation Window after applying ACL rules see the packet flow

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 50sec 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 240sec router permits packets and hence throughput can be observed in the plot shown below **Figure 3-36**.

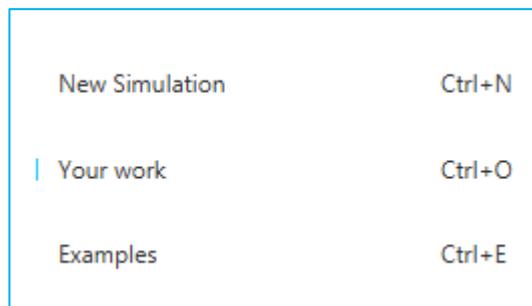


**Figure 3-36:** CBR Application throughput plot

### 3.3 Saving & Opening experiments and Printing results

#### 3.3.1 Opening Saved Experiments:

Click on **Your work** (Ctrl+O).



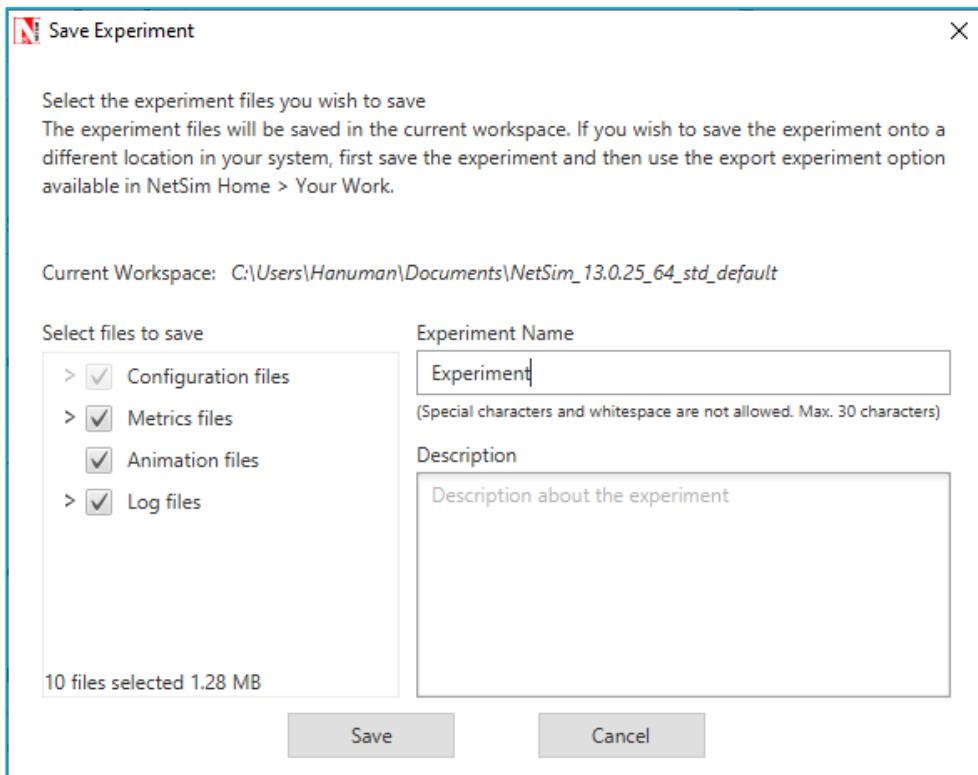
**Figure 3-37:** Select Your work

Click on the saved experiment file you wish to open.

### 3.3.2 Saving an Experiment

**During Simulation:** Users can save by using the short cut CTRL + S.

**After Simulation: From Network Window:** Click on File > Save button on the top left. Next, specify the Experiment Name, Description (Optional) and click on Save.



**Figure 3-38:** Save popup window

Upon saving several files would get saved inside the folder, including:

- Configuration file (xml) & metrics file (xml)
- Trace Files (csv), if enabled, and
- Plot data (txt)

## 3.4 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	Function
<b>Home Screen</b>	
Ctrl + N	Open a New Network
Ctrl + O	Open a Saved Network
Ctrl + E	Open a Examples
<b>Design Window (Any Network)</b>	

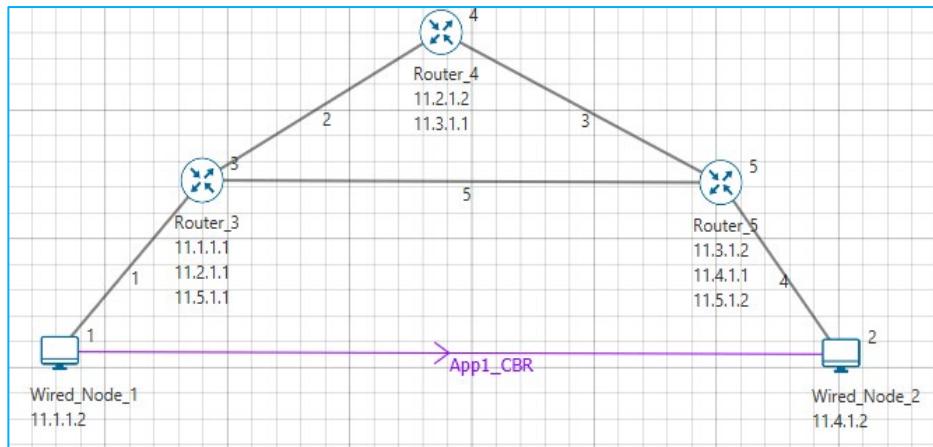
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + R	Open Run Simulation Window
Ctrl + S	Save the Experiment
Shift + Ctrl + S	Save As (To Save under different name)
Ctrl + P	Open Image/Screenshot of the network scenario that is designed in the GUI
Alt+ F4	Close Window
F1	User Manual Help
Ctrl + '+/-'	Zoom In/Zoom Out
Mouse Click (Left)	Select a device
Ctrl + A	Select All devices in the design environment
Ctrl + Mouse Click (Left) and Drag	Select devices within a selected area
Delete	Deletes the selected devices in the Environment along with any links it may have
<b>Simulation Console</b>	
Ctrl + C	Terminates Simulation in Mid way. Results will be calculated till the time at which the simulation is terminated
<b>Packet Animation Window</b>	
Space bar	To Play/Pause animation

**Table 3-1:** NetSim keyboard shortcuts

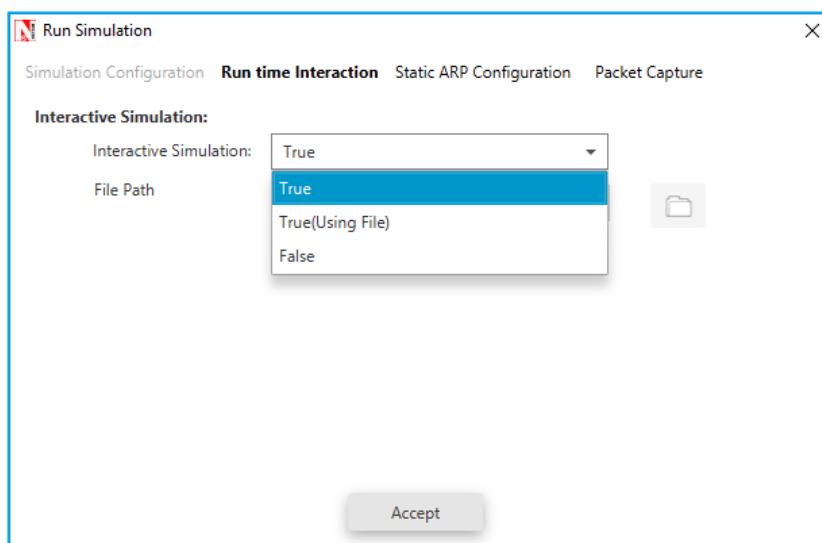
### 3.5 NetSim Interactive Simulation

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 → Internetworks. Click & drop Wired Nodes and Router onto the Simulation Environment and link them as shown below **Figure 3-39** or otherwise Open the scenario for Interactive Simulation which is available in “<NetSim Install Dir>\Docs\ Sample\_Configuration\Internetworks\Interactive Simulation”.

**Figure 3-39:** Network Topology

- Click on Application icon present in the top ribbon and set the Application type as CBR. The Source\_Id is 1 and Destination\_Id is 2.
- Set Start Time as 30 Sec
- Enable Plots and Packet trace options
- Click on run simulation option and In the Run time Interaction tab set Interactive Simulation as True and click on Accept as shown below **Figure 3-40**.

**Figure 3-40:** Run time Interaction tab set Interactive Simulation as True

- 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 OK
- Simulation (NetSimCore.exe) will start running and will display a message "waiting for first client to connect" as shown below **Figure 3-41**.

```

C:\Program Files\NetSim Pro\bin\NetSimCore.exe

License Manager Output. Product>Edition>Maj_ver>Min_ver>Lic_type>Components>
netsim>pro>11>0>rlm_hw>00000000>000>

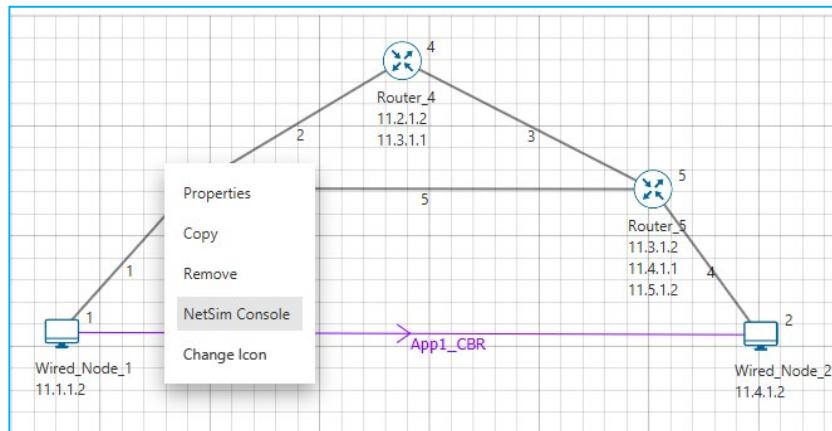
NetSim License Manager Start. Checking for licenses available (this may take upto 2 min) -
License Manager Output. Product>Edition>Maj_ver>Min_ver>Lic_type>Components>
netsim>pro>11>0>rlm_hw>11111111>110>
NetSim license validated
Installing heart-beat...
Heartbeat status = 0 (0 indicates successful)
NetworkStack loaded from path- C:\Program Files\NetSim Pro\bin\NetworkStack.dll

*** 
NetSim start
Network Stack loaded
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\Plot_*
Metrics variables initialized
Protocol variables initialized
Executing command --- DEL "C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\/*.pcap"
Could Not Find C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\*.pcap
Emulation is disabled
CLI mode is enable...
Waiting for first client to connect. Press ctrl+c to stop connection.

```

**Figure 3-41:** Waiting for first client to connect

- After Simulation window opens, goto Network scenario and right click on Router\_3 or any other node and select NetSim Console option as shown **Figure 3-42**.

**Figure 3-42:** Select NetSim Console option

- Now Client (NetSimCLI.exe) will start running and it will try to establish connection with NetSimCore.exe. After connection is established the window will look similar like this shown below **Figure 3-43**.

```

C:\Program Files\NetSim Pro\bin\NetSimCLI.exe

Initialising Winsock...Initialised.
Connecting to device DESKTOP-LC53CTS.
Connection attempt: 1
Connection established.

NetSim>

```

**Figure 3-43:** Connection is established

- After this the command line interface can be used to execute the supported commands

**Note: Commands are not a case sensitive**

### 3.5.1 Simulation specific (Not applicable for file based interactive simulation)

- Pause
- PauseAt
- 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

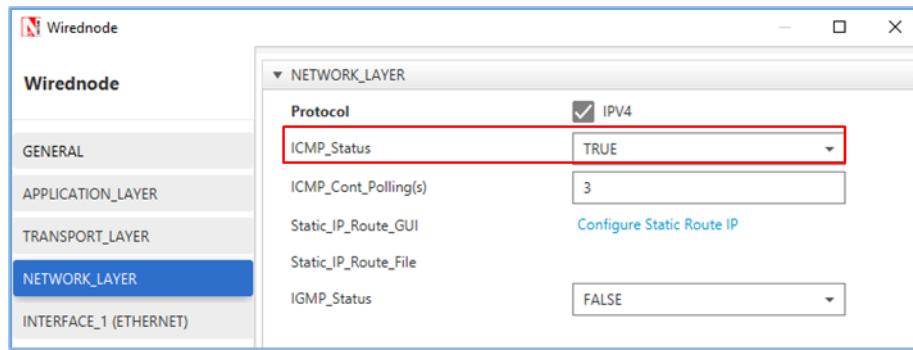
**Stop:** To stop the currently running simulation (NetSimCore.exe)

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

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

### 3.5.2 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 3-44:** Set ICMP\_Status to True in Network layer window.

- Right click on Wired\_Node\_1 and go to properties. Under General properties enable Wireshark Capture option as “Online”

**Ping <IP address> e.g. ping 11.4.1.2**

**Ping <NodeName> e.g. ping Wired\_Node\_2**

### 3.5.2.1 Ping Command Results

```
C:\Program Files\NetSim Pro\bin\NetSimCLI.exe
Initialising Winsock...Initialised.
Connecting to device DESKTOP-LCS3CTS.
Connection attempt: 1
Connection established.

NetSim>ping Wired_Node_2
Reply from 11.3.1.2: bytes 32 time=67us TTL=255

NetSim>ping 11.3.1.2
Reply from 11.3.1.2: bytes 32 time=67us TTL=255

NetSim>
```

**Figure 3-45:** 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
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 3-46:** ICMP Control Packets in Packet Trace

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

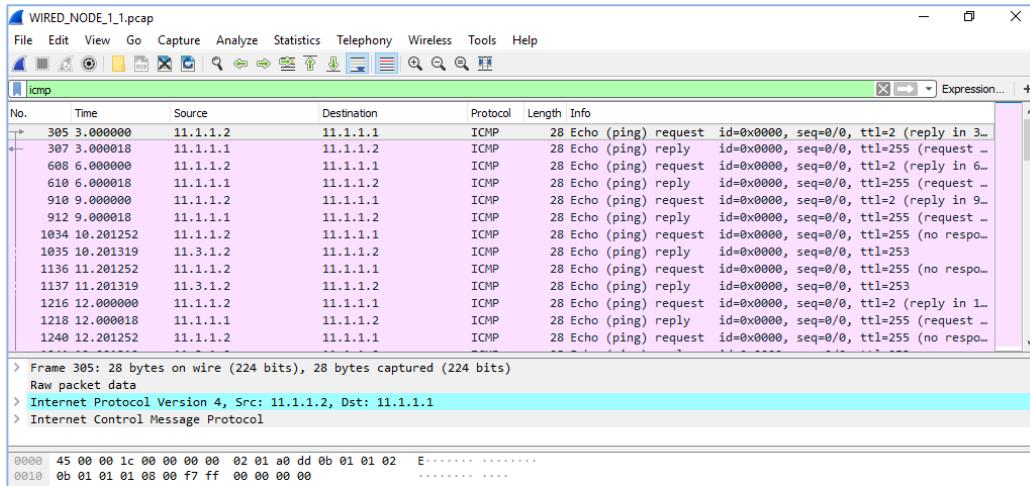


Figure 3-47: ICMP Control Packets in Wireshark

### 3.5.3 Route Commands

- route print
- route delete
- route add

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

**route print**

```
C:\Program Files\NetSim Pro\bin\NetSimCLI.exe
Initialising Winsock...Initialised.
Connecting to device DESKTOP-LCS3CTS.
Connection attempt: 1
Connection established.

NetSim>route print
=====
IP Route Table
=====

      Network Destination  Netmask//Prefix      Gateway      Interface
      Type          11.2.1.2  255.255.0.0        11.2.1.2    11.2.1.1
      OSPF          11.3.1.1  255.255.0.0        11.2.1.2    11.2.1.1
      OSPF          11.5.0.0  255.255.0.0      on-link     11.5.1.1
      LOCAL         11.2.0.0  255.255.0.0      on-link     11.2.1.1
      LOCAL         11.1.0.0  255.255.0.0      on-link     11.1.1.1
      LOCAL         224.0.0.1 255.255.255.255  on-link     11.1.1.1
      MULTICAST     224.0.0.0 240.0.0.0        on-link     11.1.1.1
      MULTICAST     255.255.255.255 255.255.255.255  on-link     11.1.1.1
      BROADCAST    255.255.255.255 255.255.255.255  on-link     11.1.1.1
=====
```

Figure 3-48: 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 11.5.0.0, all we'd have to do is type this command.

**route delete 11.5.1.2**

- To check whether route has been deleted or not check again using **route print** command.
- To add a static route to the table, you 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 11.5.1.1

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

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

The screenshot shows a terminal window titled 'NetSim>' with the following commands and output:

```
C:\Program Files\NetSim Pro\bin\NetSimCLI.exe
NetSim>route ADD 11.5.1.2 MASK 255.255.0.0 11.5.1.1 METRIC 100 IF 2
OK!
NetSim>route print
```

IP Route Table

Type	Network	Destination	Netmask//Prefix	Gateway	Interface	Metric
STATIC		11.5.1.2	255.255.0.0	11.5.1.1	11.2.1.1	100
		11.2.1.2	255.255.0.0	11.2.1.2	11.2.1.1	200
OSPF		11.3.1.1	255.255.0.0	11.2.1.2	11.2.1.1	200
OSPF		11.3.1.2	255.255.0.0	11.5.1.2	11.5.1.1	200
OSPF		11.5.0.0	255.255.0.0	on-link	11.5.1.1	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						

**Figure 3-49:** 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.**

# 4 Workspaces and Experiments

## 4.1 What is an Experiment and what is a workspace in NetSim?

When you design & simulate a network in NetSim it is saved as an experiment. This experiment is saved within a Workspace. In a logical sense, a workspace contains all the source code files, executable files, icons, data files etc.

A workspace can contain one or more experiments.

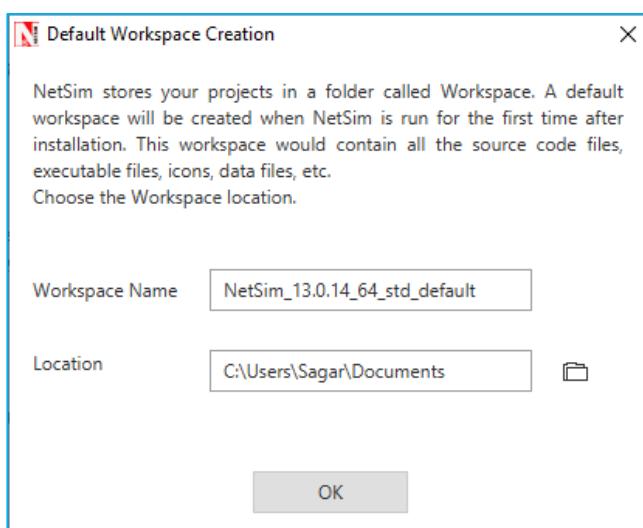
In general, users need not change the workspace and can use the default workspace.

New workspaces need to be created by users when:

- The user wants to modify the underlying source code of NetSim as is typically in research applications. The modified code will be saved in that workspace.
- A user chooses to save and organize a large number of experiments. These can be saved under different workspaces provided by that user.
- The same PC/NetSim build is shared between multiple users.

The default workspace of NetSim will have the Master Source code and the Master Binaries (Compiled files)

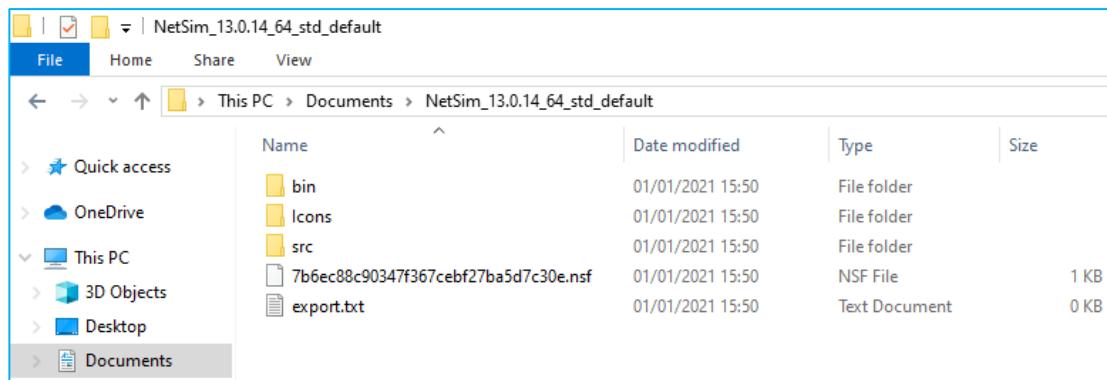
A default workspace is created in a user selected directory when NetSim is run for the first time after installation. Choose the path where user wants the default workspace to be created and click on OK.



**Figure 4-1:** Default workspace created in Documents folder

This default workspace contains the folders.

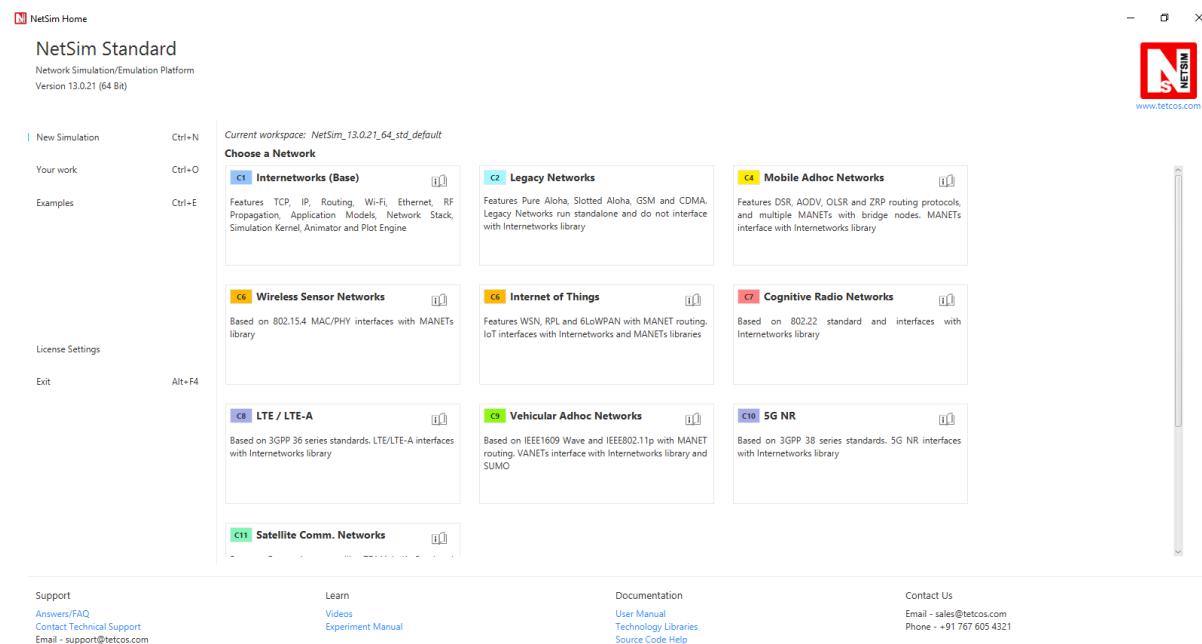
1. \src - contains protocol source codes.
2. \bin\bin\_x86 - contains NetSim binaries for 32-bit and \bin\bin\_x64 contains NetSim binaries for 64-bit.
3. \Icons - contains icons of the devices, links etc.



**Figure 4-2:** Default workspace contains different folders

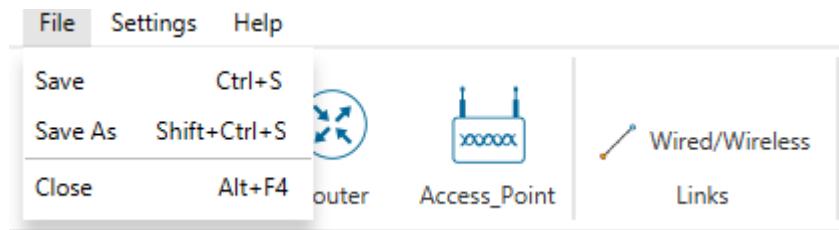
## 4.2 How does a user create and save an experiment in workspace?

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



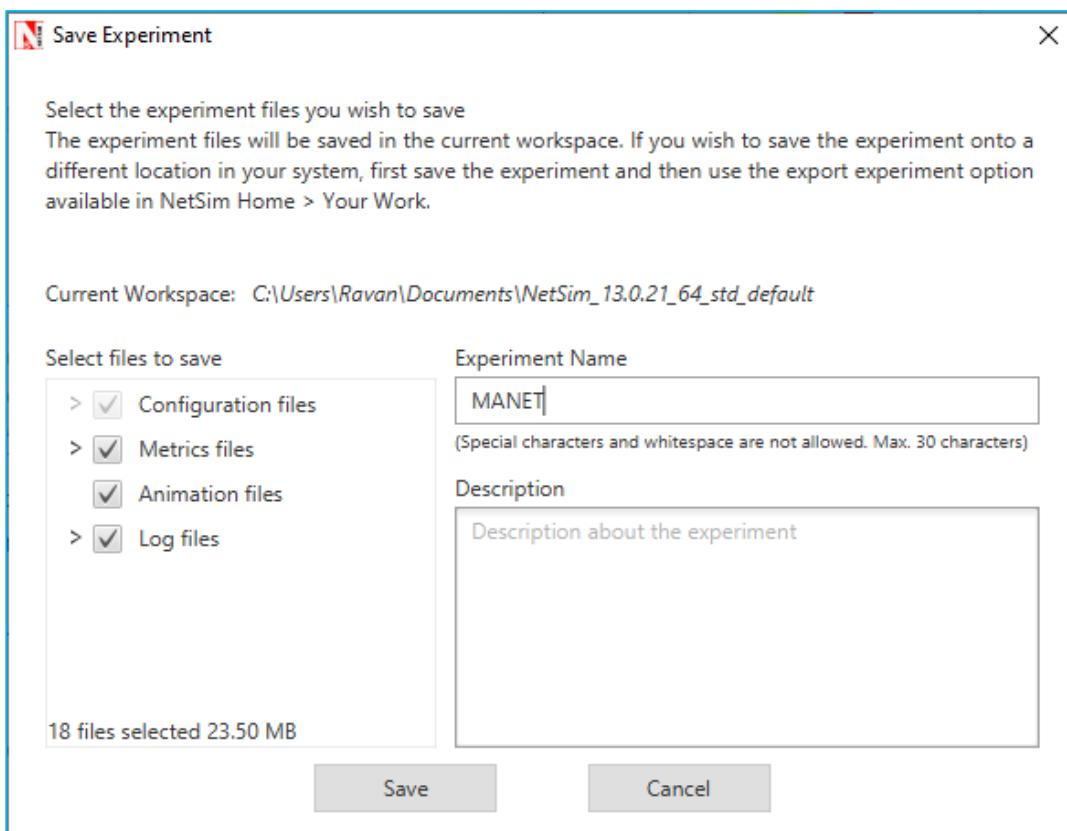
**Figure 4-3:** NetSim Home Screen

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



**Figure 4-4:** Save/Save As the experiment by clicking on File

A save pop-up window appears which contains Experiment Name, Description for the experiment, Workspace Path.



**Figure 4-5:** Save popup window

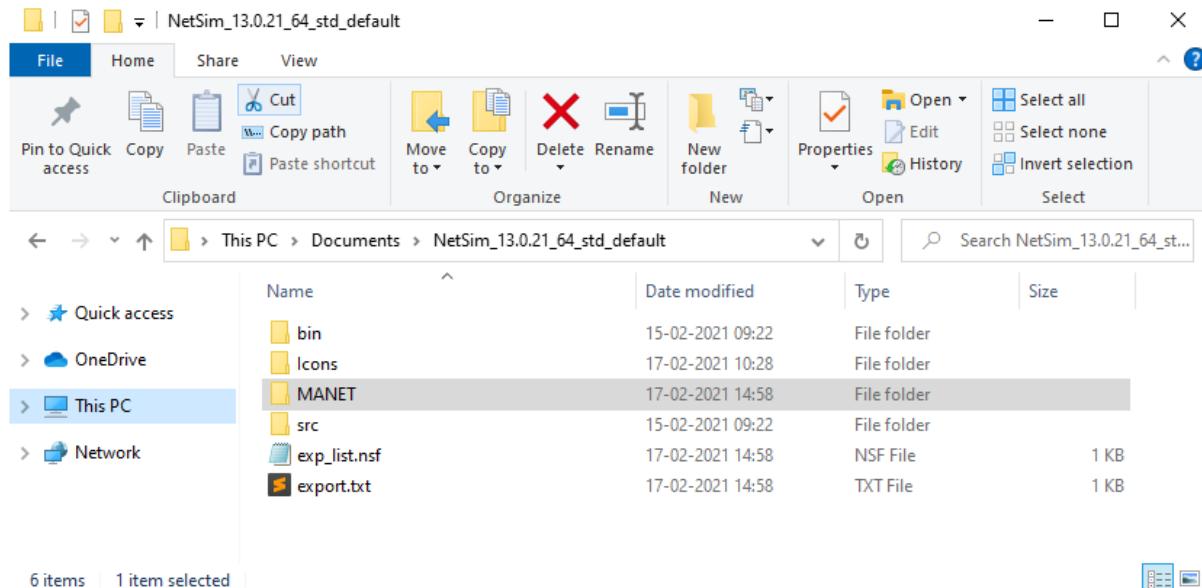
Specify the Experiment Name and Description about the experiment (optional) and click on Save. The workspace path is non-editable. Hence all the experiments will be saved in the current workspace path.

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.xml, Animation files, Event Trace file, Packet Trace file and Plot data (if enabled).

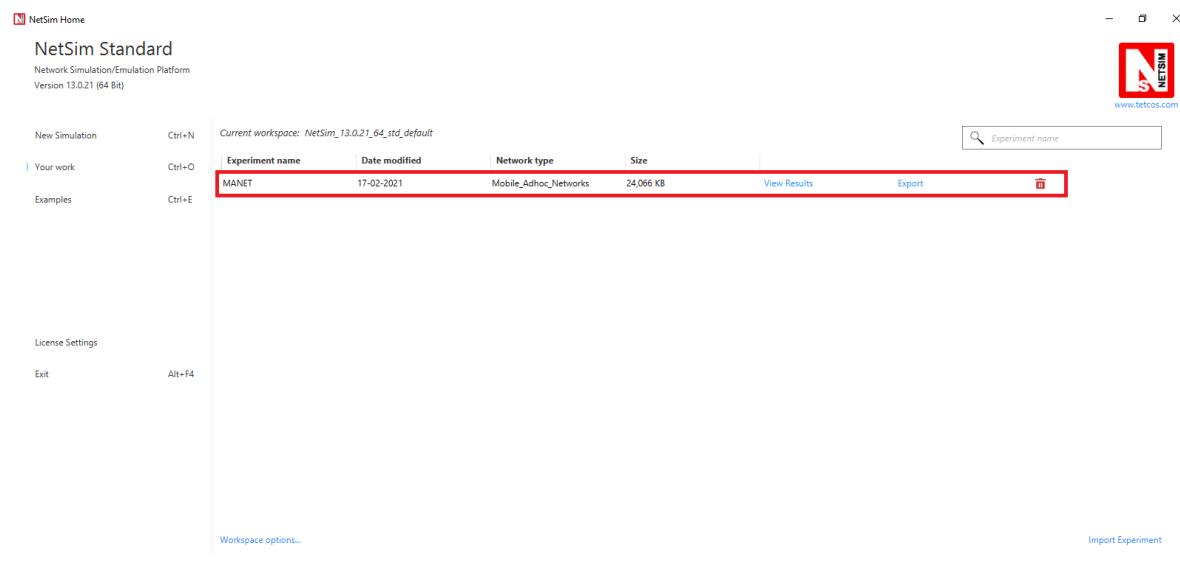
- Optional: Protocol logs (if written) or Custom Log files (if codes have been modified for logging)

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



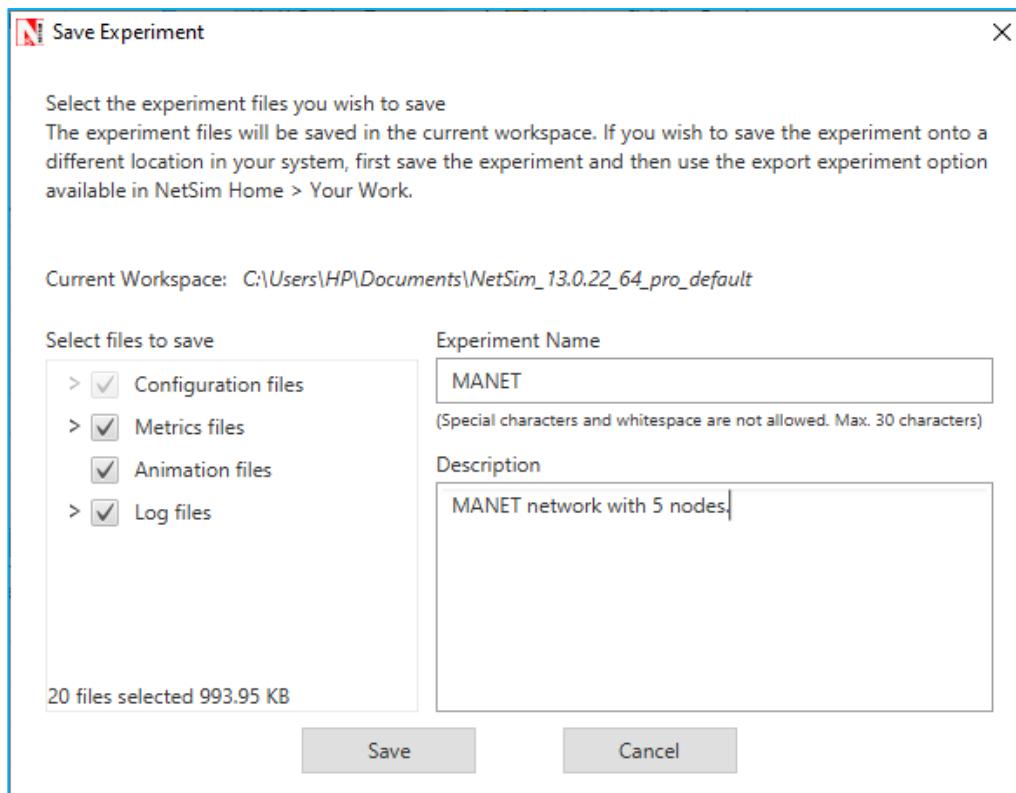
**Figure 4-6:** Manet Example Saved in Workspace

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



**Figure 4-7:** Saved experiments in Your work menu

If Description was provided while saving the experiment.



**Figure 4-8:** Save popup window along with Experiment Name and Description

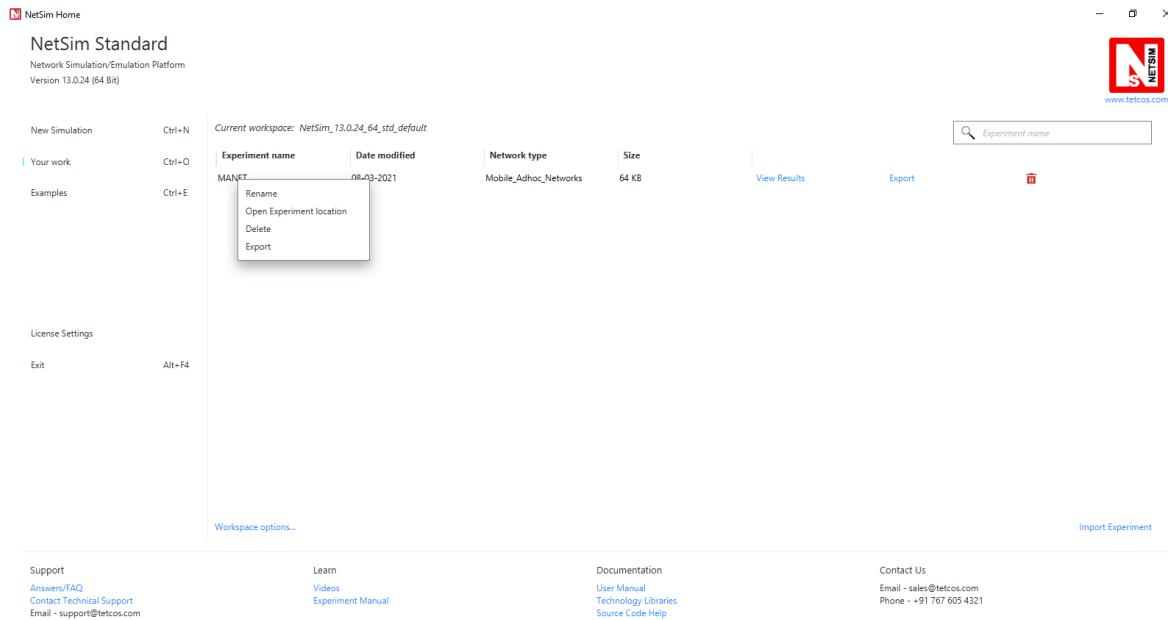
It will be displayed when mouse pointer is placed over the experiment name in the list shown as part of the **Your Work** menu.

Experiment name	Date modified	Network type	Size			
MANET	07-03-2021	Mobile_Adhoc_Networks	994 KB	<a href="#">View Results</a>	<a href="#">Export</a>	

**Figure 4-9:** Saved Experiment Present under Your Work menu

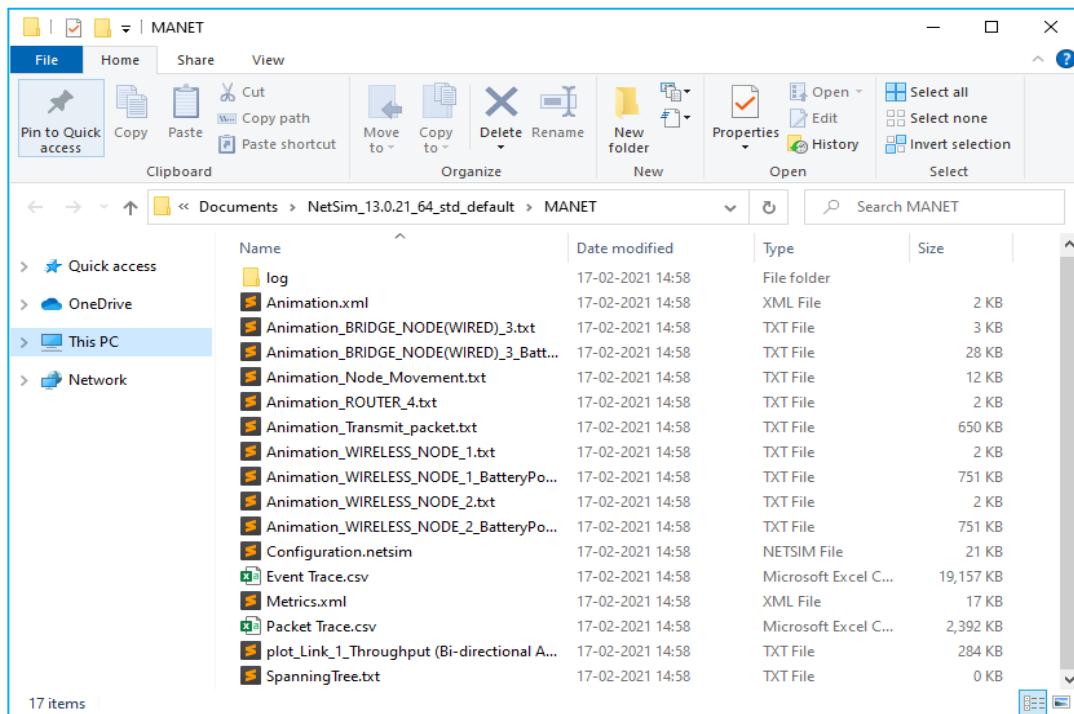
The “Save As” option is also available to save the current experiment with a different name.

Users can **rename** the experiment or **Open the folder** where the experiment is saved or **Delete** the experiment or **Export** the experiment by right-clicking on the experiment in the **Your Work** window as shown below **Figure 4-10**.



**Figure 4-10:** User options to “rename the experiment” or to “open folder” by right clicking on the experiment

In this example, we have saved all the files related to the experiment. Hence, all the files will be stored in the experiment folder **Figure 4-11**.



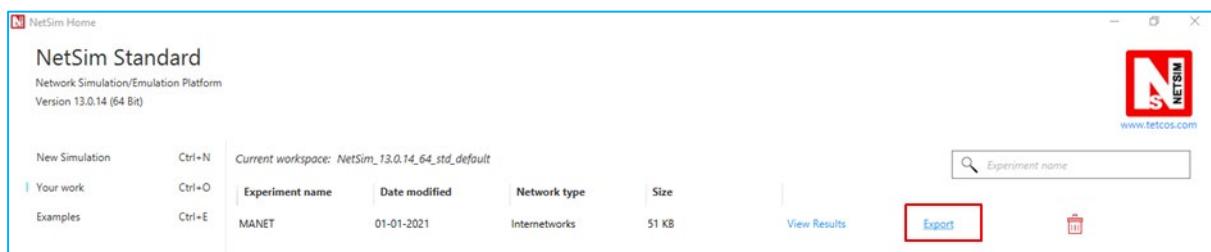
**Figure 4-11:** Simulation output files in experiment folder

## 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.

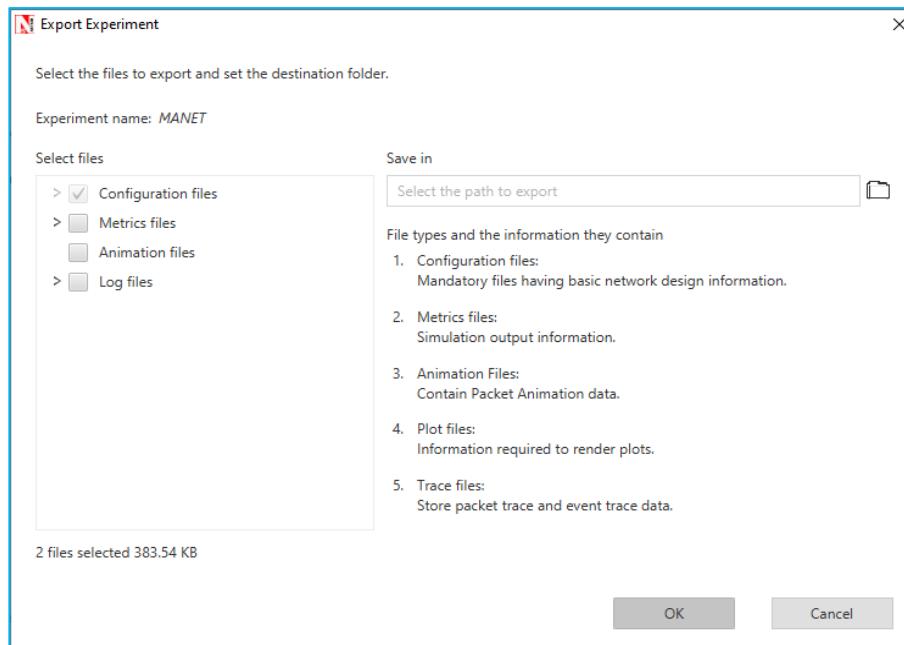
## 4.4 How does a user export an experiment?

To save experiments in a different location, first save the experiment in the current workspace and use export option present under **Your work** in the NetSim Home Screen as shown **Figure 4-12**.



**Figure 4-12:** Export option present in Your work in NetSim Home Screen

If you click on Export, an Export Experiment pop-up window appears where Users can select the files which are to be exported as part of the experiment. The Configuration file is mandatory and other files are optional. The destination path can be selected to export the experiment.

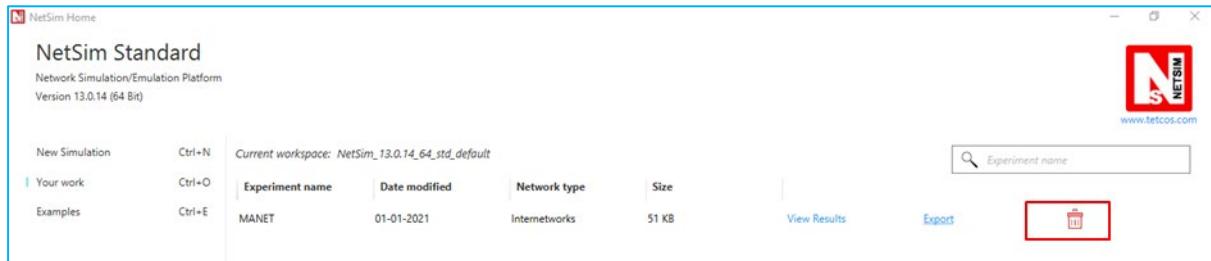


**Figure 4-13:** Export Experiment pop-up window

The exported experiments would be saved with \*.netsim\_exp extension.

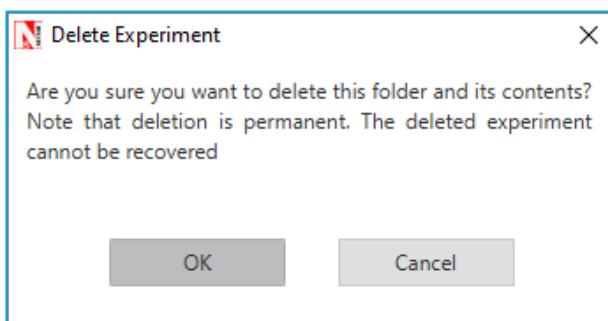
## 4.5 How does a user delete an Experiment in a workspace?

Users can delete experiments by clicking on the delete icon as shown below **Figure 4-13.**



**Figure 4-14:** Delete an Experiment in a workspace

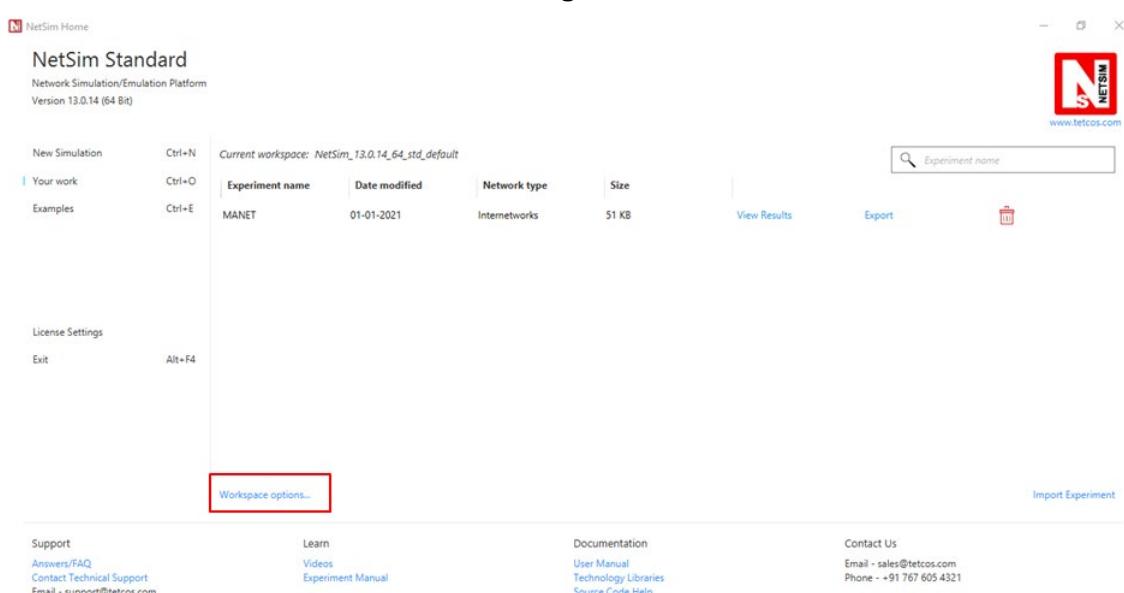
It displays a popup window as shown below and click on OK.



**Figure 4-15:** Delete Experiment window popup

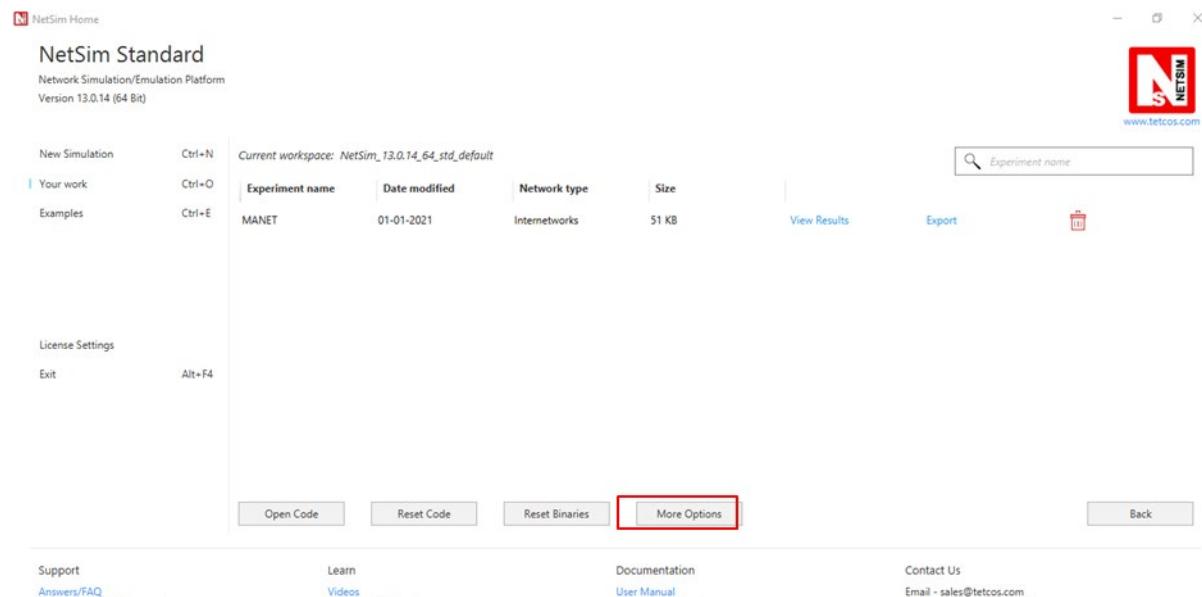
## 4.6 How does a user create a new workspace?

To create a new workspace, click on Wokspace options present in Your work Menu shown below **Figure 4-16.**



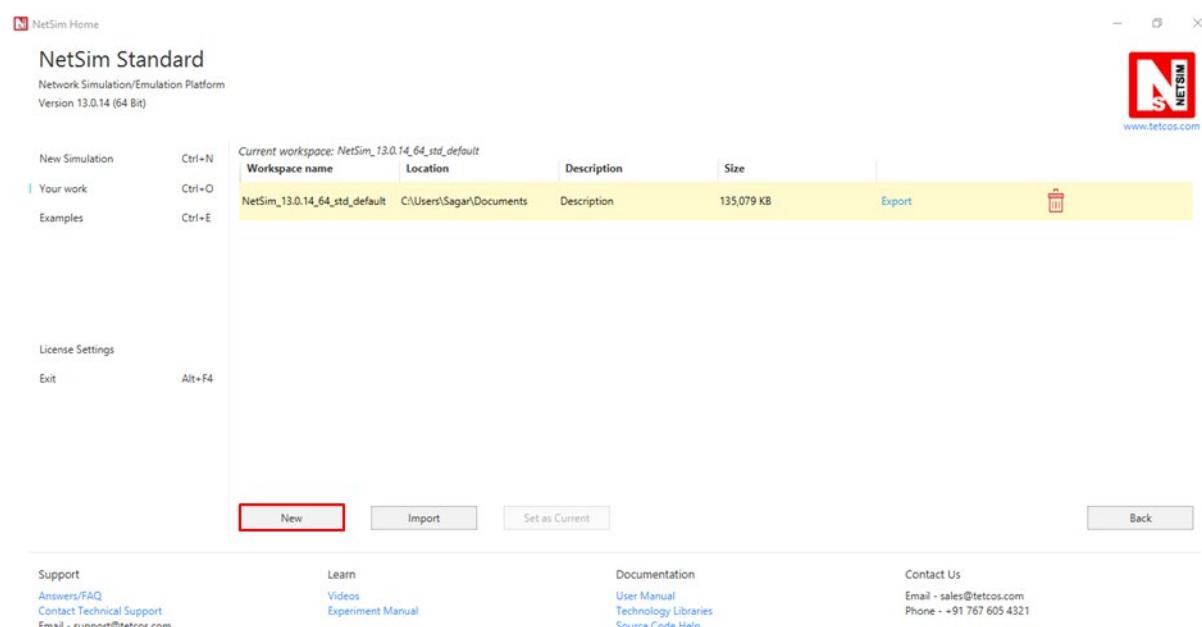
**Figure 4-16:** Select Workspace options present in Your work Menu window

Then select More Options



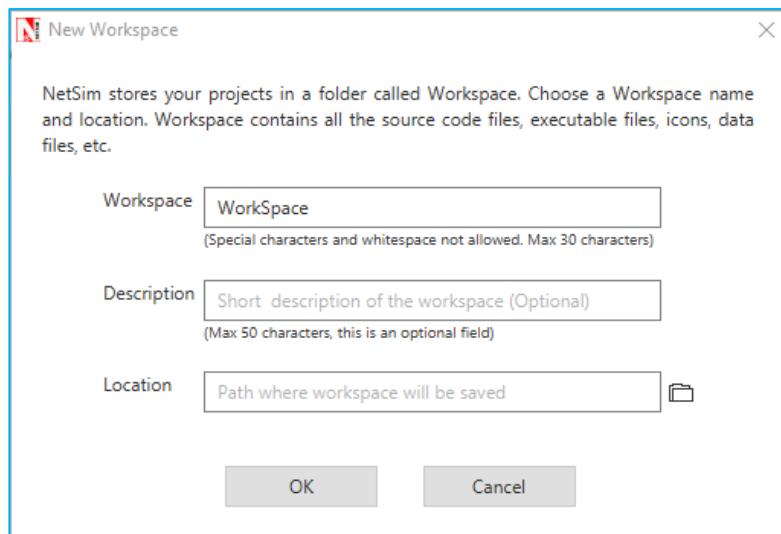
**Figure 4-17: Select More Options**

And select New



**Figure 4-18: Select New option**

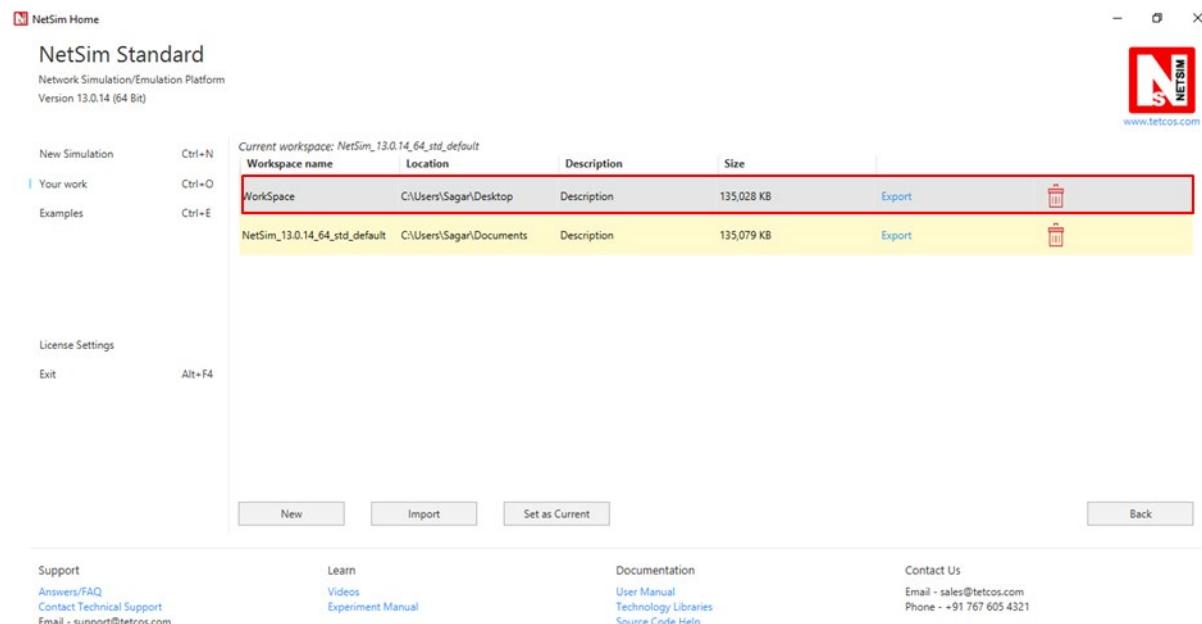
A New Workspace pop-up window appears where you can input the Workspace Name, Description and Workspace Path (where you want to create new workspace).



**Figure 4-19:** New Workspace pop-up window

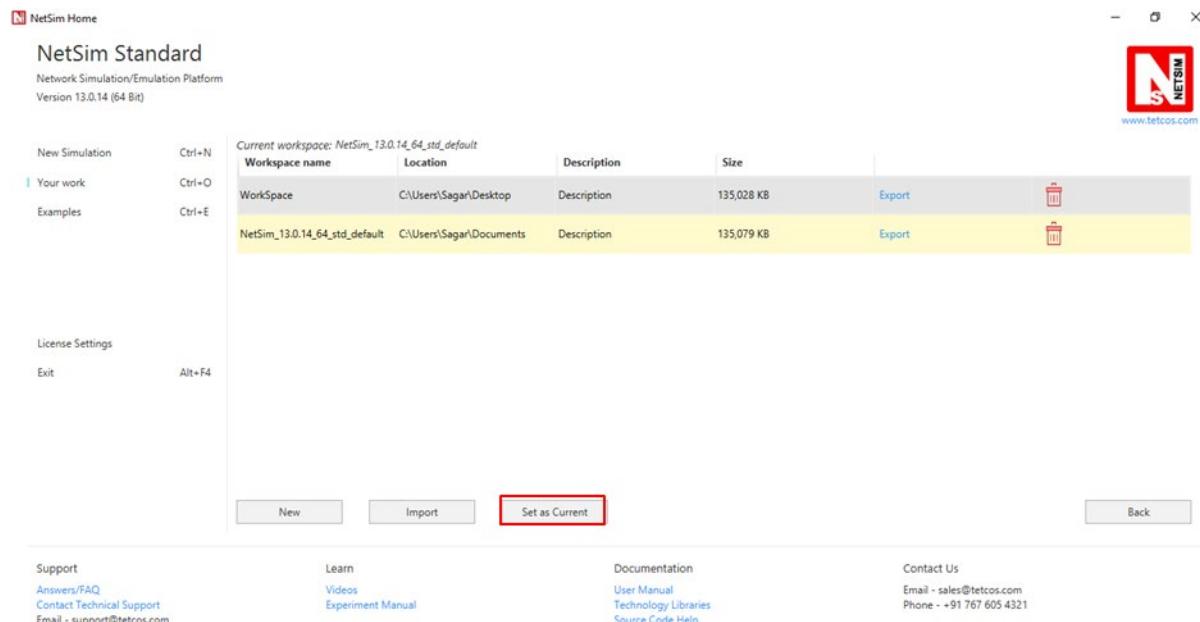
## 4.7 How does a user switch between workspaces?

Users can also switch from one workspace to another workspace. For doing this, Select Your work->Workspace Options->More Options and click on the workspace you want to switch to as shown below **Figure 4-20**.



**Figure 4-20:** Switch between workspaces

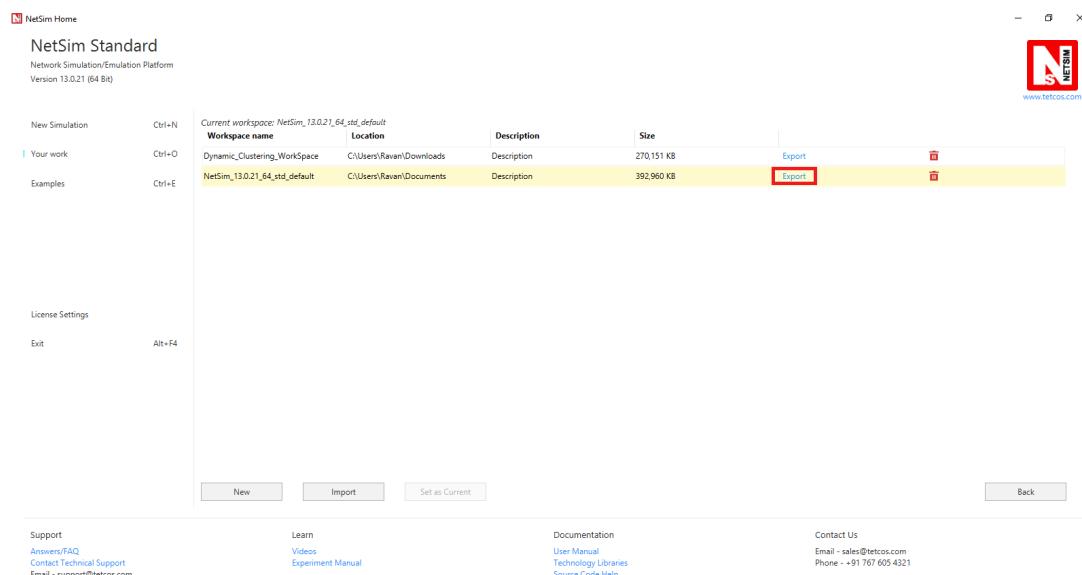
And then select “Set as Current” as shown below **Figure 4-21**.



**Figure 4-21:** Select “Set as Current” workspace

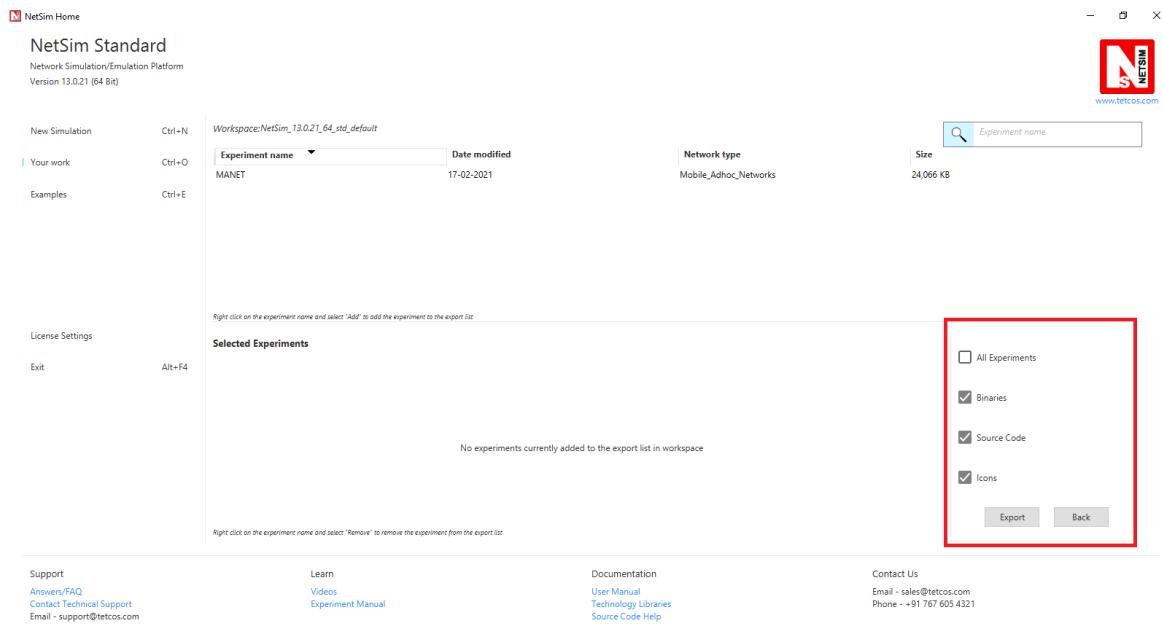
## 4.8 How does a user export a workspace?

Users can export workspaces by selecting Your Work >> Workspace Options >> More Options and then selecting Export option as shown below **Figure 4-22**.



**Figure 4-22:** Select Export option in Your work window

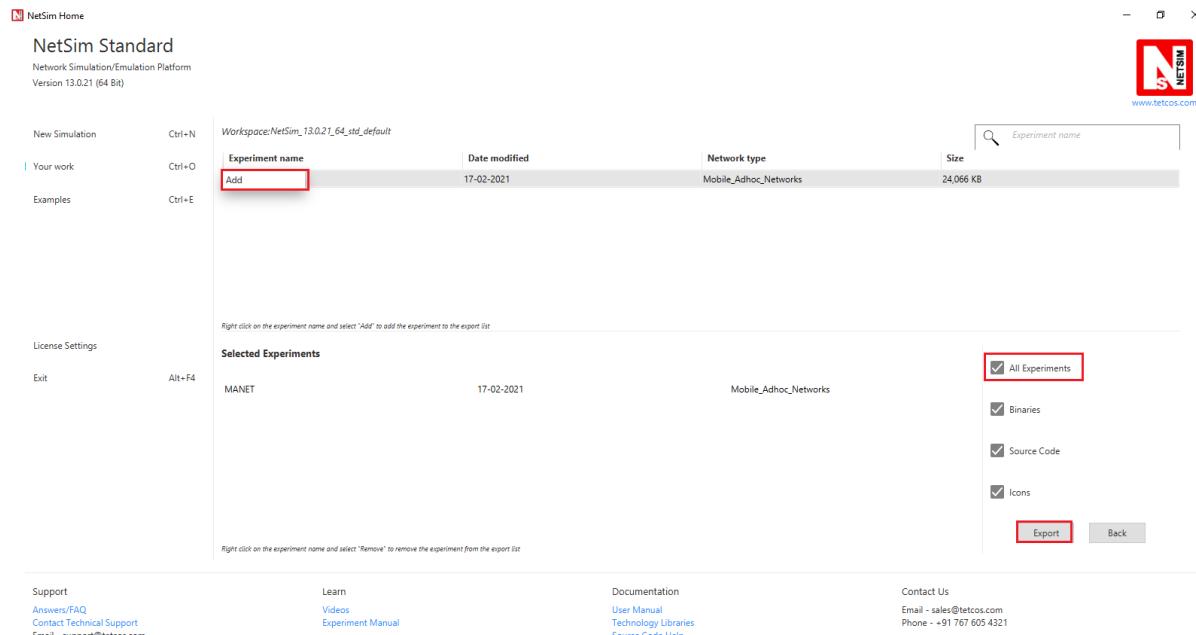
This will open export workspace window with the following options.



**Figure 4-23:** Added by default Binaries, Source Code Icons to Selected Experiments list

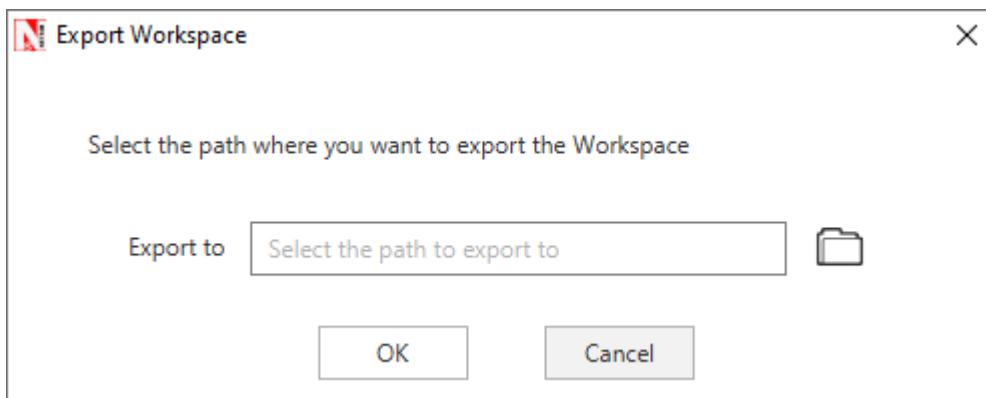
Binaries, Source Code (available only for Standard and Pro) and Icons are added by default as shown above. Users can add all experiments present in the current workspace by clicking on All Experiments check box or if the user want to add particular experiment, user can right click on the desired experiment's name and click on Add option.

After selecting the experiments, user can export the workspace by Clicking on Export button as shown below **Figure 4-24**.



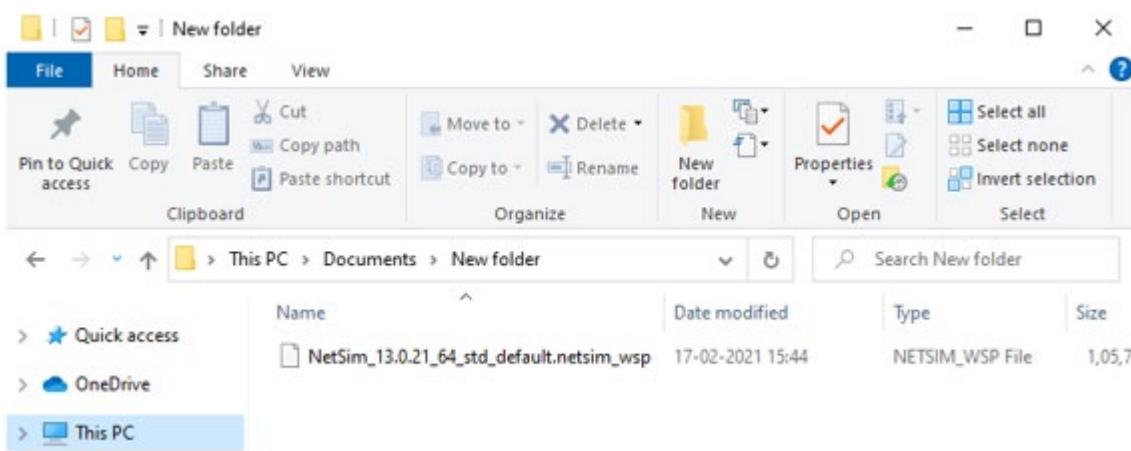
**Figure 4-24:** Select All Experiments check box and then click on Export

This will open a window as shown below where users can enter the path to which the workspace is to be exported and then click on ok as shown in below **Figure 4-25**.



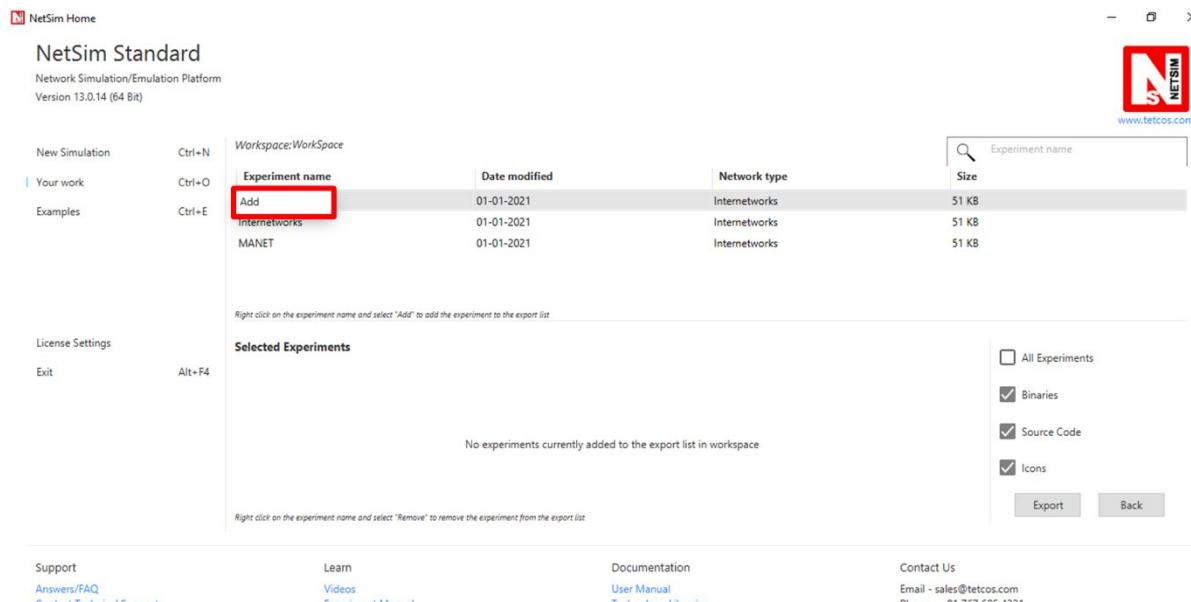
**Figure 4-25:** Select the path by clicking on browse option in Export Workspace window

The workspace will be exported to the path selected with an extension <workspace name>.netsim\_wsp as shown below.



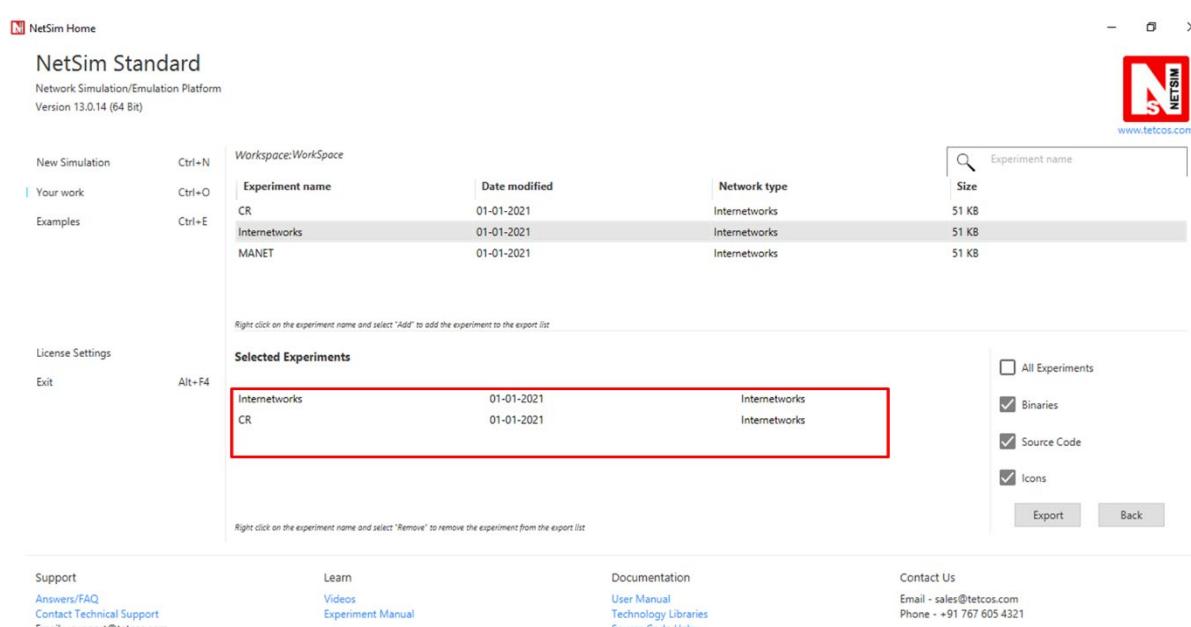
**Figure 4-26:** Workspace exported to Location.

Users can also have options to export individual experiments by right clicking on the experiment and select Add to add the experiment to the export list as shown below **Figure 4-27**.



**Figure 4-27:** Export individual experiments by right clicking on the experiment and select Add to add the experiment to the export list

The added experiments will be available in the export list shown below **Figure 4-28**.



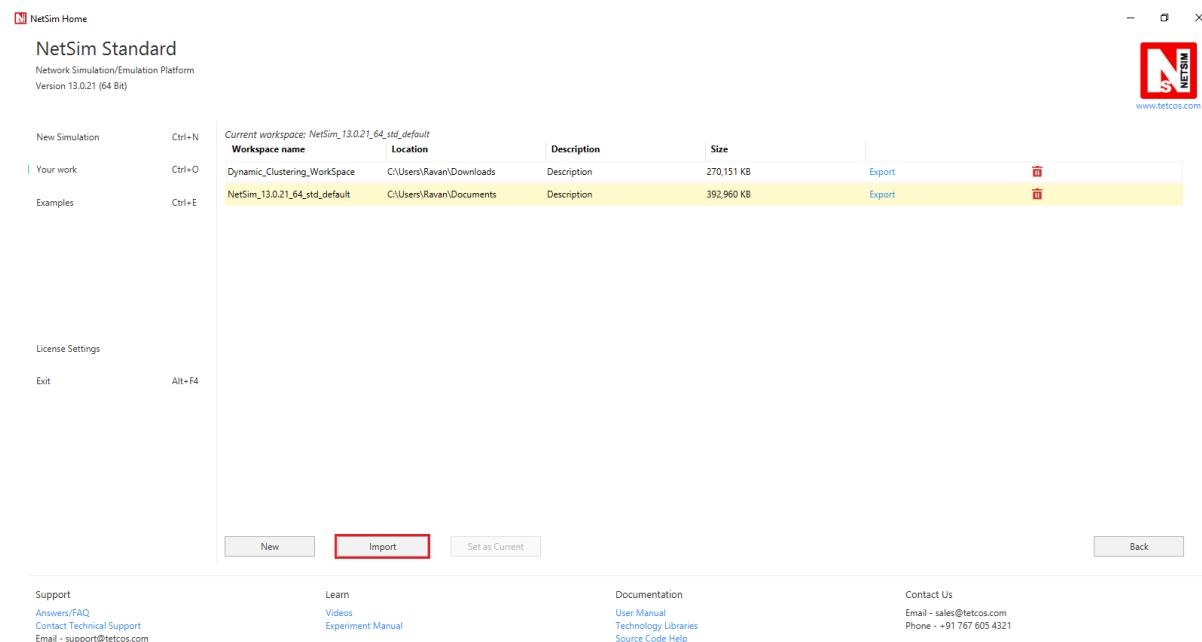
**Figure 4-28:** Added Experiments to the export list

After adding the experiments click on Export and follow the same procedure as explained above

**Note:** Users can import only the exported workspaces.

## 4.9 How does a user import a workspace?

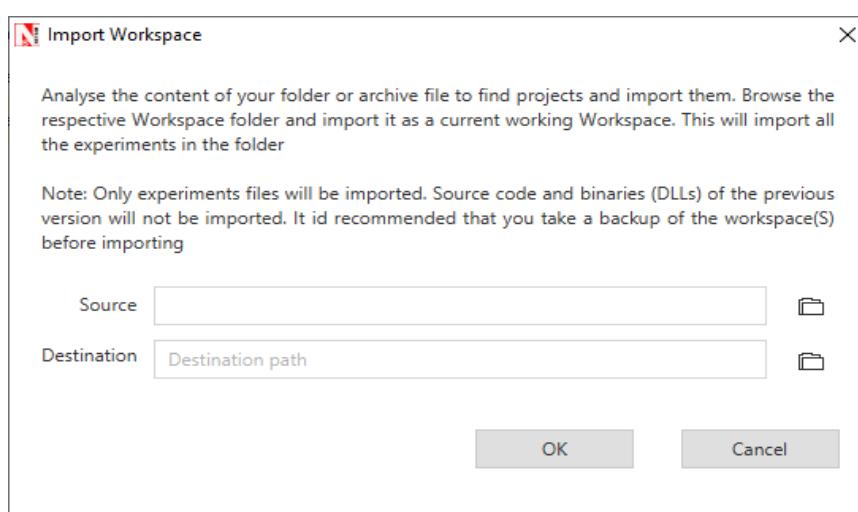
Users can import workspaces by selecting Your work >> Workspace Options >> More Options and then selecting Import option as shown below **Figure 4-29**.



**Figure 4-29:** Select Import option in Your work window

This will display a window where users need to give the source file (exported workspace file) and the Destination, the path where the workspace is to be imported to and then click on ok.

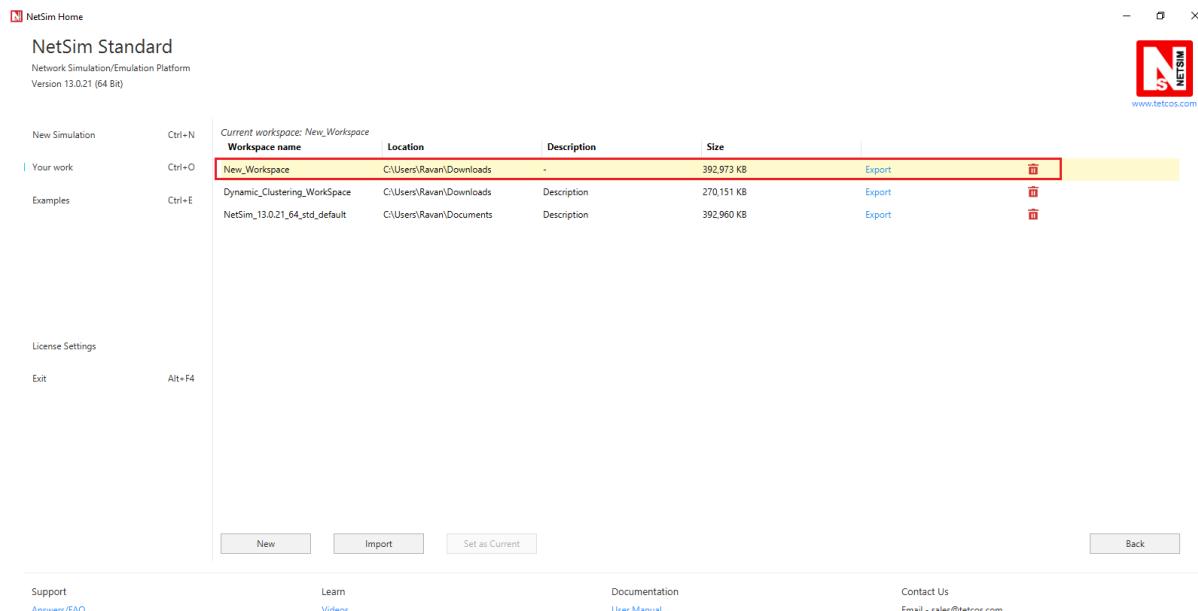
Note: Only exported workspaces with “.netsim\_wsp” extension can be imported.



**Figure 4-30:** Select the path by clicking on browse option for Source and Destination

The imported workspace will be set as the current workspace. The imported workspace can be seen by clicking on Your Work >> Workspace option >> More Options as shown below

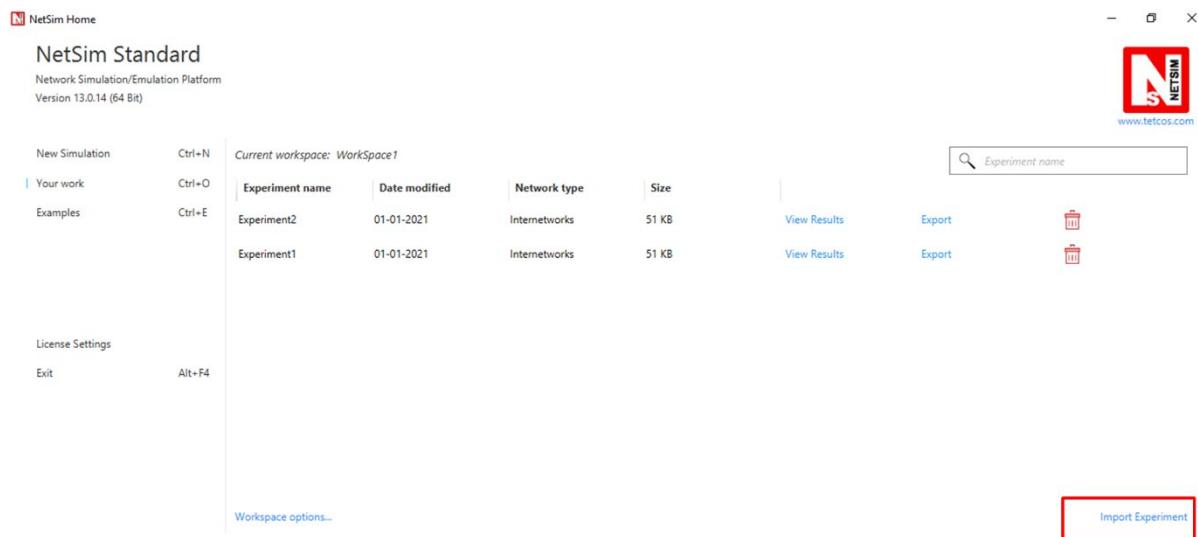
**Figure 4-31.**

**Figure 4-31:** Imported workspace set as the current workspace by default

(Note: Users can import only version 12.2 workspaces into NetSim v13.0)

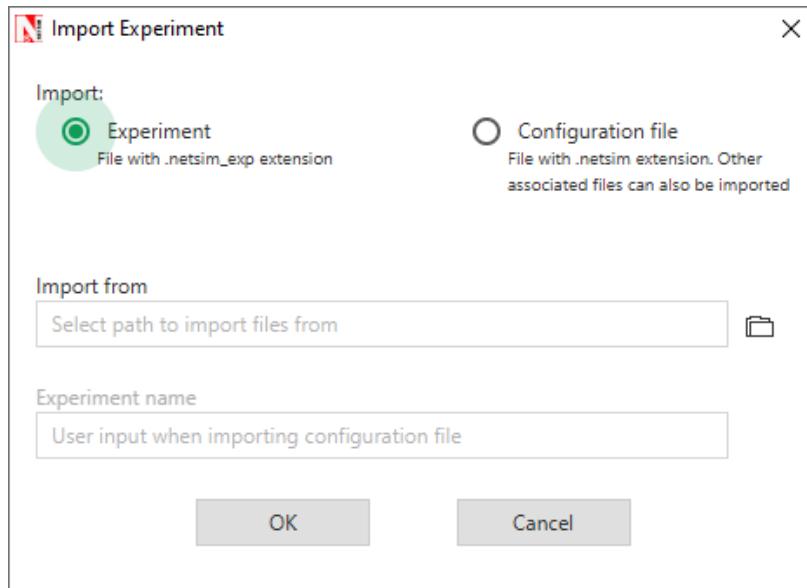
## 4.10 How does a user import an experiment?

Users can also have option to import experiments. For this click on Your work and then select Import Experiment as shown below **Figure 4-32**.

**Figure 4-32:** Import experiments option in Your work window

The following window is displayed where users need to give the path from which the experiment has to be imported and then click on OK.

User can import the configuration file also through giving the path from which the configuration file has to be imported, then describe the experiment name and click on OK.



**Figure 4-33:** Two options for Import Experiments

The imported experiment will be available in the current workspace. To see the imported experiment, click on Your work as shown below **Figure 4-34**.

The screenshot shows the NetSim Standard interface. On the left, there's a menu bar with "NetSim Home", "File" (containing "New Simulation", "Ctrl+N; Your work", "Ctrl+O; Examples", "Ctrl+E", "License Settings", and "Exit"), "Edit" (containing "Alt+F4"), and "Help". The main area is titled "NetSim Standard" and "Network Simulation/Emulation Platform Version 13.0.14 (64 Bit)". It features a logo for "NETSIM" and the website "www.tetcos.com". Below this is a table titled "Current workspace: WorkSpace1" showing three rows of workspace information. The first row, "WorkSpace1", is highlighted with a red border. The second row is "WorkSpace" and the third is "NetSim\_13.0.14\_64\_std\_default". At the bottom are buttons for "New", "Import", "Set as Current", and "Back". The footer contains links for "Support", "Learn", "Documentation", and "Contact Us".

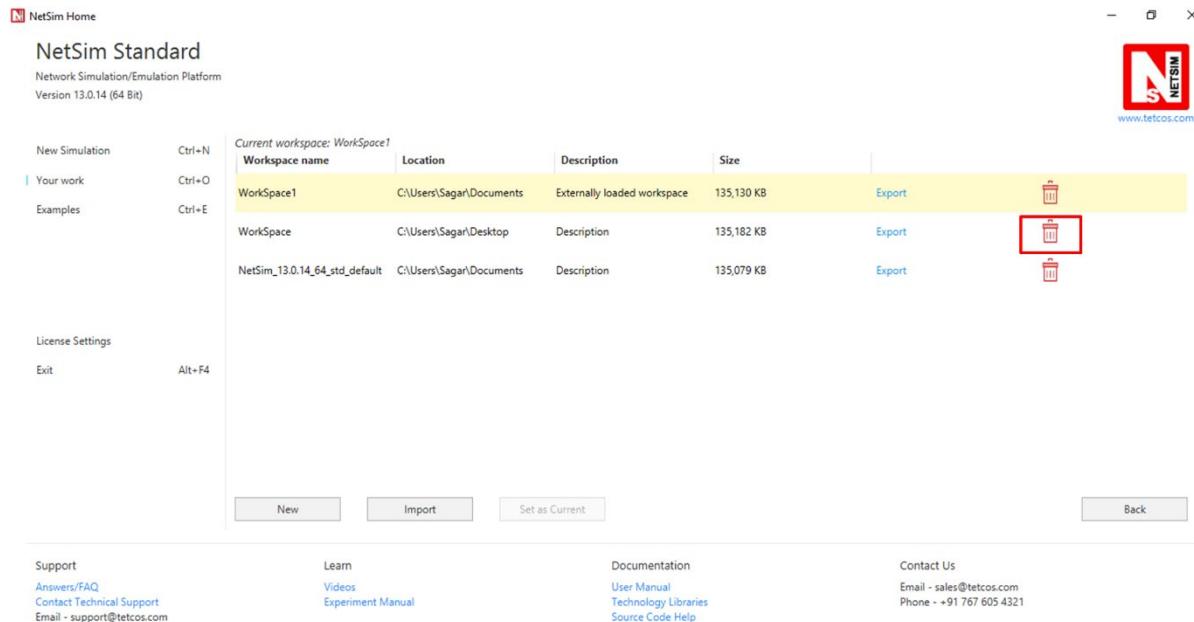
Current workspace: WorkSpace1					
Workspace name	Location	Description	Size	Export	Delete
WorkSpace1	C:\Users\Sagar\Documents	Externally loaded workspace	135,028 KB	Export	
WorkSpace	C:\Users\Sagar\Desktop	Description	135,182 KB	Export	
NetSim_13.0.14_64_std_default	C:\Users\Sagar\Documents	Description	135,079 KB	Export	

**Figure 4-34:** Imported experiment available in the current workspace

**Note:** Users can import only the exported experiments

## 4.11 How does a user delete a workspace?

Users can also delete a workspace by clicking on the delete icon shown below **Figure 4-35**

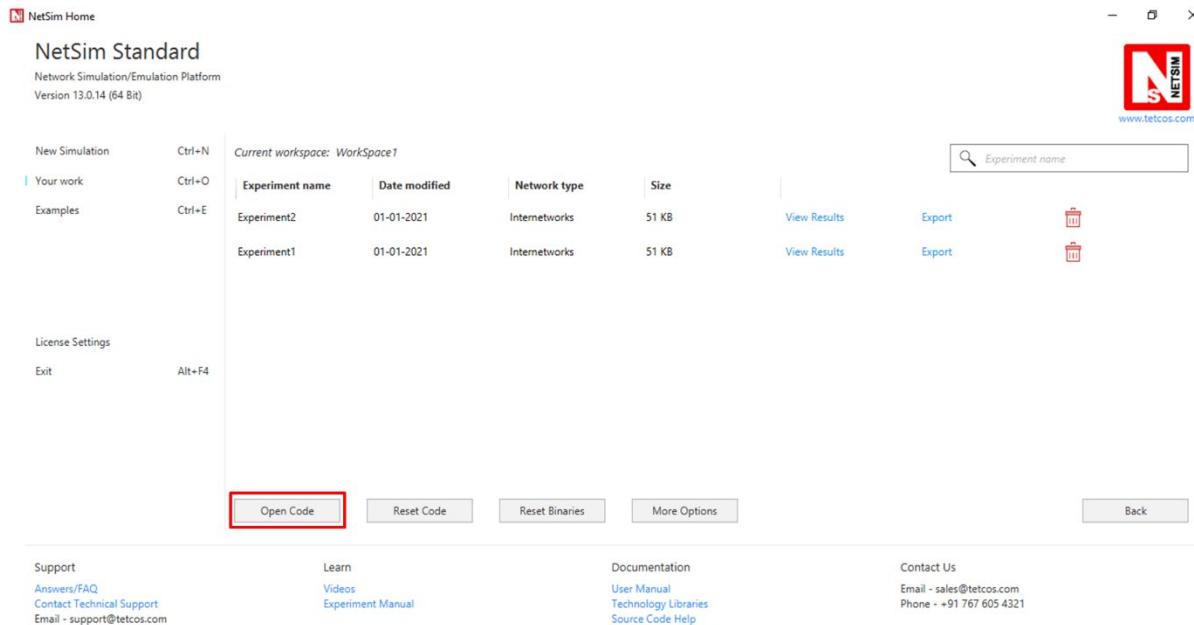


**Figure 4-35:** Delete a workspace by clicking on the delete icon in Your work window

Deleting current workspace is not allowed. Deleting a workspace will delete all saved experiments and code modifications done in that workspace.

## 4.12 How does a user open and modify source codes?

User can also modify the source codes within a workspace. For doing this, select open Your work ->Workspace Options->Open Code as shown below **Figure 4-36**.



**Figure 4-36:** Open code option is available in Your work window

This opens the source codes in MS Visual Studio.

Users can then modify the protocol codes and build the solution. Then users can create a network in NetSim or open the saved experiment which involves the protocol that has been modified and click on run simulation. This simulation will run per the modified code.

The changes in the source codes applies to the current workspace only.

## 4.13 Can I use NetSim's default code for my experiments?

Yes, each workspace will have a Reset option which would set.

1. Binaries (compiled files) to default
2. Code (source C codes) to default

## 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
<b>Internet</b>	All versions
<b>Legacy Network</b>	All versions
<b>Cellular Network</b>	All versions
<b>MANET</b>	All versions
<b>Wireless Sensor Network</b>	All versions
<b>Software Defined Network</b>	All versions
<b>Internet of Things</b>	All versions
<b>Cognitive Radio</b>	All versions
<b>LTE</b>	All versions
<b>5G NR</b>	Available only with NetSim Standard and NetSim Pro versions
<b>VANET</b>	Available only with NetSim Standard and NetSim Pro versions
<b>Satellite Communication</b>	Available only with NetSim Standard and NetSim Pro versions
<b>Network Emulator</b>	Available only with NetSim Standard and NetSim Pro versions
<b>TDMA Radio Networks</b>	Available only with NetSim Pro version
<b>Real Time Protocol</b>	Available only with NetSim Pro version

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

**Note:** *Network Emulator, TDMA Radio Networks, and Real Time Protocol are available only as separate Add-On components along with 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.

Internetworks library in NetSim covers Ethernet, Address Resolution Protocol (ARP), Wireless LAN – 802.11 a / b / g / n / ac 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 **Internet** example you wish to simulate. NetSim UI loads the example.

### 5.1.2 Internet 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
- 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 **Examples**.  
The Example 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 Legacy Networks in examples. The help documentation explains the following:

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

## 5.4 Advanced Routing

NetSim supports the following advanced routing protocols.

- Multicast Routing –
  - Internet Group Management Protocol (IGMP)
  - Protocol Independent Multicast (PIM)
- 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
- 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 > Internet-of-Things** 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
- Reference Documents
- Latest FAQ available online

## 5.7 Internet of Things

Internet of things (IoT) is a network of objects 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 6LowPAN Gateway. WSN for IoT uses the following protocols: AODV 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 > Wireless-Sensor-Networks** 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
- 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
- Model Features
- Featured Examples
- Reference Documents
- 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
- Reference Documents
- Latest FAQ available online

## 5.10 LTE

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 15 / 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, Animation, Traces etc.

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

### 5.11.1 5G NR 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 **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
- Specifications scheduled for next release.

- 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 (24 – 60 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 TDMA Radio Networks

NetSim TDMA 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.*

### 5.14.1 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.

### 5.14.2 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
- Reference Documents
- 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.

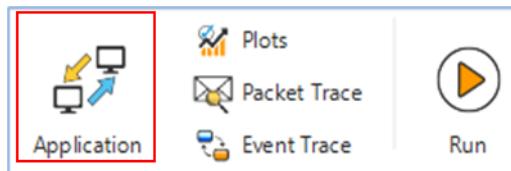
Applications are generally parameterized in terms of packet size, inter-packet arrival time, priority, transport protocol running below etc. Therefore, each application has its own distinctive traffic pattern and creates its own unique load on the network.

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.

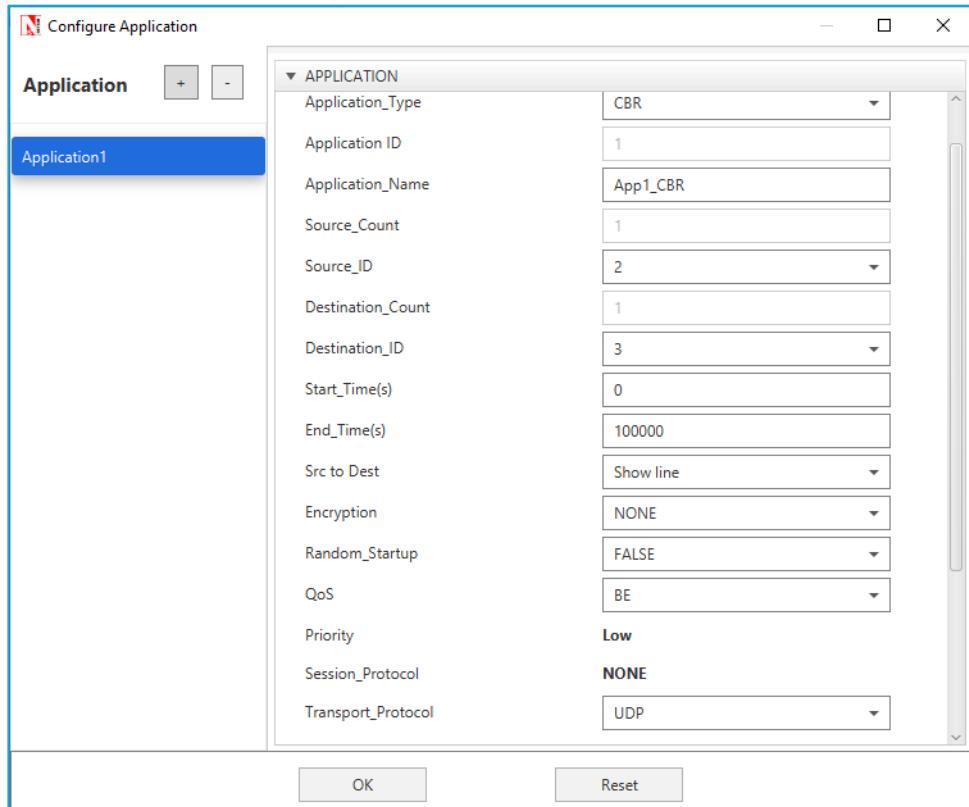
NetSim allows users to model and simulate different types of applications.

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

To set up the application click on the application icon from the tool bar as shown below **Figure 6-1.**



**Figure 6-1:** Application icon from the tool bar



**Figure 6-2:** Application Configuration Window

This properties window allows you to model the application traffic. There may be more than one application you may require for your simulation study. You can add (or) delete one or more applications by clicking on the “+” or “-” symbols present on top left-hand side next to the Application.

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, COAP, 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.

**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'.

For **Multicast** Applications, users can enter the number of multicast destinations in the Destination Count field and specify the Device ID's of the destination devices separated by comma (",") in the Destination ID field. E.g., 6, 7, 8

**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 1<sup>st</sup> second and stops at the 10<sup>th</sup> 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 8.7 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 true, application will start at a random time between 0 and inter-arrival time. Having a random start-up time provides more realism to the model since all applications need not necessarily start at time = 0 in the real world.

**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
<b>UGS - Unsolicited Grant Service</b>	The UGS scheduling service type is designed to support real-time data streams consisting of fixed-size data packets issued at periodic intervals	High
<b>rtPS - Real-time Polling Service</b>	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, for example, for MPEG (Moving Pictures Experts Group) video transmission	Medium
<b>ertPS - Extended real-time Polling Service</b>	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
<b>nrtPS - Non-real-time Polling Service</b>	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
<b>BE - Best Effort</b>	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.

**Session Protocol:** Session Protocol is applicable only for applications that support RTP (Real-time Transport Protocol)

**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

Brief explanation of application types as shown below **Table 6-2.**

Application Type	Properties	Units	Description
<b>CBR – Constant bit Rate</b>	<b>Packet size</b> (Constant distribution) – It is the size of the packet	bytes	<p>Packets of constant size are generated at constant inter arrival times.</p> <p>The generation times would be as follows:</p> <ul style="list-style-type: none"> <li>Packet 1: Application start time</li> <li>Packet 2: Packet 1 + Interarrival Time</li> <li>Packet 3: Packet 2 + Interarrival Time</li> <li>....</li> <li>Packet (n+1): Packet n + Interarrival Time</li> </ul> <p>Ends at Application end time</p>
	<b>Inter Arrival Time</b> (Constant distribution) – It is the gap between two successive packets		
<b>Custom</b>	<b>Packet size</b> (Constant, Exponential distribution) – It is the size of the packet	bytes	<p>It is user defined application where the packet size and inter-arrival time can be set per user requirements.</p>
	<b>Inter Arrival Time</b> (Constant, Exponential distribution) – It is the time	μs	

	gap between two successive packets		
<b>Peer to Peer</b>	<b>File size distribution</b> (Constant, Exponential distribution)	-	Peer-to-peer network does not have the notion of clients or servers but only equal peer nodes that simultaneously functioning as both "clients" and "servers" to the other nodes on the network. <b>Ex</b> – Torrent, LimeWire etc.
	<b>Value</b> – Size of the file	bytes	
	<b>Piece size</b> - Each file is divided into equal sized pieces. This property represents the size of each piece	bytes	
<b>Email</b>	<b>Email send/receive</b> – Represents the rate at which emails are sent/receive	-	Allows users to send/receive email application <b>Ex</b> – Outlook, Apple mail, Gmail etc.
	<b>Duration</b> (Constant, Exponential distribution) - Time between two successive emails	Seconds	
	<b>Email size</b> (Constant, Exponential distribution) – Size of an email	Bytes	
<b>HTTP – Hyper Text Transfer Protocol</b>	<b>Inter Arrival Time</b> (Constant, Exponential distribution) – It is the time gap between two successive HTTP requests	seconds	HTTP is a protocol that utilizes TCP to transfer its information between computers (usually Web servers and clients). TCP should mandatorily be set as the transport layer protocol.
	<b>Page size</b> (Constant, Exponential distribution) – It is the size of each page	bytes	

	<b>Page count</b> – Represents the number of pages	-	
<b>COAP – Constrained Application Protocol</b>	<b>Inter Arrival Time</b> (Constant, Exponential distribution) – It is the time b/w two successive COAP requests	seconds	
	<b>Page size</b> (Constant, Exponential distribution) – It is the size of each page	bytes	
	<b>Response time</b> – It is the time taken by a device to generate response	ms	
	<b>Multicast response</b> – Represents the server responds to multicast response or not	-	
	<b>NSTART</b> – Limit the number of simultaneous outstanding interactions that a client maintains to a given server	-	It is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks and designed for M2M applications
	<b>DEFAULT_LEISURE</b> – This setting is only relevant in multicast scenarios, outside the scope of the EST-coaps draft	-	
	<b>PROBING_RATE</b> : A parameter which specifies the rate of re-sending Non-confirmable messages.	-	

	<b>Ack required</b> – It represents whether the ack for the request/response to be sent or not	-	
<b>FTP – File Transfer Protocol</b>	<b>File size</b> (Constant, Exponential distribution) – It is the size of the file	bytes	<p>It is a standard network protocol used for the transfer of files between a client and server</p> <p><b>Note:</b> Devices must have TCP enabled as the transport layer protocol.</p> <p><b>Ex</b> – FileZilla</p>
	<b>File Inter Arrival Time</b> <b>size</b> (Constant, Exponential distribution) – It is the gap between two files	seconds	<p>The generation times would be as follows:</p> <p>File 1: Application start time</p> <p>File 2: File 1 + Interarrival Time</p> <p>File 3: File 2 + Interarrival Time</p> <p>....</p> <p>File (n+1): File n + Interarrival Time</p> <p>Ends at Application end time</p> <p>The files are in-turn fragmented into packets during the simulation.</p> <p>Users can generate one file by setting</p>

			<i>Application End Time</i> $< (Application Start Time + File Interarrival Time)$
<b>Database</b>	<b>Transaction size</b> (Constant, Exponential distribution) - It represents the size of each transaction	bytes	A database application is a computer program whose primary purpose is entering and retrieving information from a computerized database
	<b>Transaction Inter Arrival Time</b> (Constant, Exponential distribution) – It is the time gap between two successful transactions	μs	<b>Ex</b> – MS Excel, MySQL etc.
<b>Voice</b>	<b>Packet size</b> (Constant, Exponential) – It is the size of the packet	bytes	It allows users to configure voice application between client and server
	<b>Packet Inter Arrival Time</b> (Constant, Exponential distribution) - It is the gap between two successful packets	μs	<b>Note</b> – Distribution is constant only for all codec types except custom  <b>Ex</b> – Skype
	<b>Service type</b> – CBR, VBR	-	
	<b>Suppression models available for VBR</b> – Deterministic, Markov chain	-	
	<b>Success ratio</b> - Sets the ratio of the packets that are not silenced during VBR calls	%	

<b>Video</b>	<b>Model Type</b> - Continuous Normal VB, Continuous State Autoregressive Markov, Quantized State Continuous Time Markov, Simple IPB Composite Model	-	It allows users to configure video application between client and server  <b>Ex – Skype</b>
<b>Erlang Call</b>	<b>Packet size</b> (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
	<b>Packet Inter Arrival Time</b> (Constant, Exponential distribution) - It is the gap between two successful packets	μs	<b>Note</b> – Distribution is constant only for all codec types except custom
	<b>Call duration</b> (Constant, Exponential distribution) – It is the duration of each call	seconds	
	<b>Call Inter Arrival Time</b> (Constant, Exponential distribution) - It is the gap between two successful calls	seconds	
	<b>Service type</b> – VBR, CBR	-	
	<b>Suppression model available for VBR</b> – Deterministic, Markov chain	-	
	<b>Success ratio</b> - Sets the ratio of the packets that	%	

	are not silenced during VBR calls		
<b>Sensor App</b>	<b>Packet size</b> (Constant distribution) – It is the size of the packet	bytes	Used to create application between two sensors <b>Ex</b> – Smart home, Smart water etc.
	<b>Packet Inter Arrival Time</b> (Constant distribution) - It is the gap between two successful packets	μs	
<b>BSM – Basic safety message</b>	<b>Packet size</b> (Constant, Exponential 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
	<b>Packet Inter Arrival Time</b> (Constant, Exponential distribution) - It is the gap between two successful packets	μs	Messages (BSMs). The BSM is a 20-byte packet that is generally broadcast from every vehicle at a nominal rate of 10 Hz. <b>Note</b> - Available only with VANET component <b>Ex</b> – Traffic management
<b>Emulation</b>	<b>Source Real IP</b> - Specifies the real IP Address of source device in Emulation	-	NetSim Emulation application enables users to connect NetSim simulator to real devices and interact with live applications.
	<b>Source Port</b> - Specifies the Port no used for transmission by Application running in source device		<b>Note</b> - Will be present only when Emulator Add-on is licensed
	<b>Destination Real IP</b> - Specifies the real IP		

	Address of destination device in Emulation		
	<p><b>Destination Port -</b></p> <p>Specifies the Port no used for reception by Application running in destination device</p>		

**Table 6-2:** Brief explanation of application types**Voice, Erlang call**

For Voice and Erlang call applications, Codec option is available as follows.

**Codec**

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.

Five different standards of voice codec's available in NetSim are G.711, G.729, G.723, GSM-FR, GSM-EFR which can be selected depending on the variations required. 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. There are two types of standard compression algorithms are used.

- $\mu$ -law algorithm.
- A-law algorithm.

**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) speech codec was developed in early 1990s and was adopted by the 3GPP for mobile telephony. 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. The codec provides voice activity detection (VAD) and comfort noise generation (CNG) algorithms and an inherent packet loss concealment (PLC) algorithm for handling frame erasures. The codec was primarily developed for mobile telephony over GSM networks.

**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).

**CUSTOM:** It is similar to the CUSTOM application type explained in the table above.

## Video Models

### Model Type

- **Continuous Normal VBR** – This model is the simplest of all models. It uses Normal Distribution for the generation of bits per pixel. In this model, consecutive packet sizes are independent of each other.
  - **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 100.
  - **Pixels per frame** -Number of pixels in each frame. This is in the range of 10000 – 100000.
  - **Bits per pixel ( $\mu$ )** – Mean value of the normal distribution used to generate the value of bits per pixel.
  - **Bits per pixel ( $\Sigma$ )** – Standard Deviation of the normal distribution used to generate the value of bits per pixel.
    - The generation rate for video application can be calculated by using the formula Generation Rate (bits per second) = fps \* ppf \* bpp where, fps = frames per second, ppf = pixel per frame, bpp ( $\mu$ ) = bits per pixel (mean)
    - Users can set the above-mentioned parameters in the Application Properties.
- **Continuous State Autoregressive Markov** –This model incorporates the autocorrelation between the frames. Also, current packet size depends on the previous packet size via the first order autoregressive Markov process.
  - **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 50.

- **Pixels per frame** - Number of pixels in each frame. This is in the range of 10000 – 100000.
- **Constants A, B**– First order autoregressive Markov process  $\lambda(n)$  can be generated by the recursive relation  $\lambda(n) = a\lambda(n-1)+bw(n)$ .
- **Eta**– The steady-state average  $E(\lambda)$ and discreet auto covariance  $C(n)$  are given by  $E(\lambda) = (b / (1-a)) \eta$   $C(n)=(b^2/(1-a^2))\eta$  where  $\eta$  is the Gaussian parameter.
- **Quantized State Continuous Time Markov** –In this model the bit rate is quantized into finite discrete levels. This model takes uniform quantization step as A bits/pixel. There are M + 1 possible levels (0, A... MA). Transitions between levels are assumed to occur with exponential rates that may depend on the current level. This model is approximating the bit rate by a continuous time process  $\lambda(t)$  with discrete jumps at random Poisson time.
  - **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 100.
  - **Pixels per frame** - Number of pixels in each frame. This is in the range of 10000 – 100000.
  - **No of Multiplexed Sources**– This model considers the aggregate instantaneous input rate  $\lambda_N(t)$  instead of the single source bit rate  $\lambda(t)$ . The total rate is the sum of N independent random processes each with mean  $E(\lambda)$  and variance  $C(0)$  at steady state. Therefore, the steady state- mean of  $\lambda_N(t)$  will be  $E(\lambda_N) = N \times E(\lambda)$  bits/pixel.
  - **Quantization Level**– This model takes uniform quantization step as A bits/pixel. There are M + 1 possible levels (0, A, MA). Transitions between levels are assumed to occur with exponential rates that may depend on the current level.
- **Simple IPB Composite Model**–In this model, the frames are organized as IBBPBPBPBBIBBPBB... i.e., 12 frames in a Group of Pictures (GOP). Generate  $X_0$  from a Gaussian distribution  $N(0, y 0)$ . Set initial value  $N_0= 0$ ,  $D_0 = 1$ .

For  $k = 1, 2, \dots, N-1$ , calculate  $\Phi_{kj}$ ,  $j = 1, 2, \dots, k$  iteratively using the following formulae

$$N_k = r(k) - \sum_{j=1}^{k-1} \Phi_{k-1,j} r(k-j)$$

$$D_k = D_{k-1} - (N_{2k-1}/D_{k-1})$$

$$\Phi_{kk} = N_k / D_k$$

$$\Phi_{kj} = \Phi_{k-1,j} - \Phi_{kk}\Phi_{k-1,k-j} \quad j=1, \dots, k-1$$

$$m_k = \sum_{j=1}^k \Phi_{kj} X_{k-j}$$

$$y_k = (1 - \Phi_{2kk}) y_{k-1}$$

Finally, each  $X_k$  is chosen from  $N(m_k, \sigma^2)$ . Thus, we get a process  $X$  with ACF approximating to  $r(k)$ .

The auto correlation function can be calculated in a recursive way as follows:

$$r(0) = 1, r(k+1) = ((k+d)/(k+1))r(k)$$

Where  $d = H - 0.5$ .

$H$  is called the Hurst parameter  $k-\beta$  is used as the ACF of a self-similar process. We get the value of  $H$  parameter for a self-similar process using the relationship,

$$\beta = 2 - 2H$$

Distribution of these data is Gaussian. For data to be Beta distributed, the following mapping is being used.

$$Y_k = F^{-1}\beta(FN(X_k)), k > 0$$

$X_k$ : Self-similar Gaussian process,

$FN$ : The cumulative probability of normal distribution,

$F^{-1}\beta$ : The inverse cumulative probability functions of the Beta model.

- **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 50.
- **Gamma I, Gamma B, Gamma P, Eta I, Eta B, Eta P, Beta I, Beta P, Beta B** – Refer i-button help of Simple IPB Composite Model.

## 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

$$\text{Generation Rate (Mbps)} = (\text{Packet size (bytes)} * 8) / \text{Inter arrival time (\mu s)}$$

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

$$\text{Generation rate (Mbps)} = 1460 * 8 / 20000 = 0.584 \text{ Mbps}$$

### Video

The traffic generation rate for Video applications are based on the CONTINUOUS\_NORMAL\_VBR model. This CONTINUOUS\_NORMAL\_VBR model is the

simplest of all video models in NetSim. It uses 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:

$$\text{Generation Rate (bits per second)} = \text{fps} * \text{ppf} * \text{bpp}$$

where, fps = frames per second

ppf = pixel per frame

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

$$\begin{aligned}\text{Generation rate (bps)} &= \text{fps} * \text{ppf} * \text{bpp} \\ &= 20 * 10000 * 0.52 = 104000 \text{ bits per second} = 0.1040 \text{ Mbps}\end{aligned}$$

## Voice

$$\text{Generation Rate (Mbps)} = (\text{Packet size (bytes)} * 8) / \text{Inter arrival time (\mu s)}$$

**Note** – Distribution is constant for all codec types except custom.

## Email

$$\text{Generation Rate (Mbps)} = (\text{Email size (bytes)} * 8) / \text{Duration (s)}$$

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

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

## HTTP

$$\text{Generation Rate (Mbps)} = (\text{Page size (bytes)} * 8 * \text{Page count}) / \text{Inter arrival time (s)}$$

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

$$\text{Generation rate (Mbps)} = 20000 * 8 * 2 / 3000000 = 0.106 \text{ Mbps}$$

## FTP

$$\text{Generation Rate (Mbps)} = (\text{File size (bytes)} * 8) / \text{Inter arrival time (s)}$$

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

$$\text{Generation rate (Mbps)} = 100000 * 8 / 5 = 0.16 \text{ Mbps}$$

## Database

$$\text{Generation Rate (Mbps)} = (\text{Packet size (bytes)} * 8) / \text{Inter arrival time (\mu s)}$$

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

$$\text{Generation rate (Mbps)} = 10000 * 8 / 1000000 = 0.08 \text{ Mbps}$$

## BSM

$$\text{Generation Rate (Mbps)} = (\text{Packet size (bytes)} * 8) / \text{Inter arrival time (\mu s)}$$

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

$$\text{Generation rate (Mbps)} = 20 * 8 / 1000000 = 0.00016 \text{ Mbps}$$

## 6.4 Priority and QoS of Applications

The various application traffic generated in NetSim have the following priority and QoS values as shown below **Table 6-3**.

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

**Table 6-3:** 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.

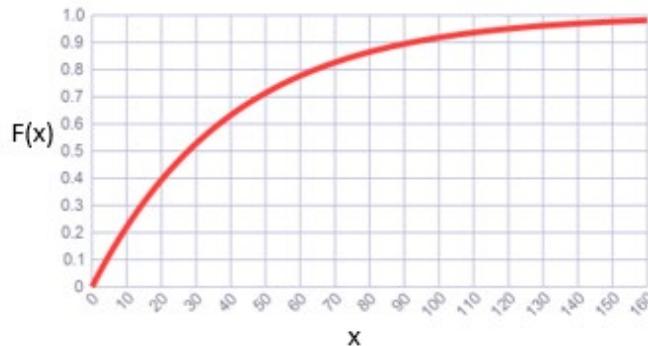
## 6.6 Modelling Poisson arrivals in NetSim

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  $60 / 1200 = 0.05$  seconds. So, let's define a variable  $\lambda = 1 / 0.05 = 20$  and call it the *rate parameter*. The rate parameter  $\lambda$  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  $F(0.05) \approx 0.63$
- The probability of generating a packet within 1 second is  $F(1) \approx 0.9999999998$

In particular, note that after 0.05 seconds – the prescribed average time between packets – the probability is  $F(0.05) \approx 0.63$ .

### 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 that 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\_13.0.14\_64\_std\_default\bin\bin\_x64”** for 64-bit and  
**“C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86”** for 32-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 path where the input and output files of an application is written. This is similar to the temp path of windows OS. For NetSim, the IO path can be got by **Start → Run → %temp%\NetSim**. Once you reach this folder, the user can notice that the path would be something like

**“C:\Users\PC\AppData\Local\Temp\NetSim\std13.0.14\_x64”**

The IO path is the path where the **Configuration.netsim** (*NetSim Configuration file*) of the scenario, that will be simulated, should be present.

App path and IO path can also be same, i.e., Configuration.netsim can be placed inside the app path (*if the app path has the 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\NetSim_13.0.14_64_std_default\bin\bin_x64"
```

#### 7.1.1 Running in CLI Mode when using floating licenses

For floating licenses, type the following in the command prompt. The type of license can be seen by clicking on **Help → About NetSim** in GUI of NetSim.

```
>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\_13.0.14\_64\_std\_default\bin\bin\_x64>** for 64-bit and **<C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x86>** for 32-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 (dongle) 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.

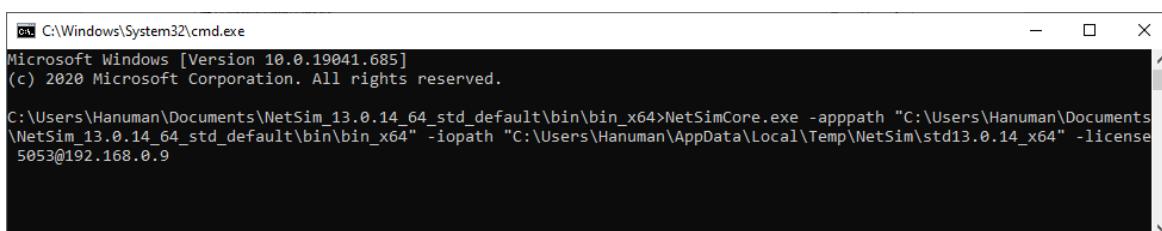


Figure 7-1: Running NetSim through CLI mode for floating license

### 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

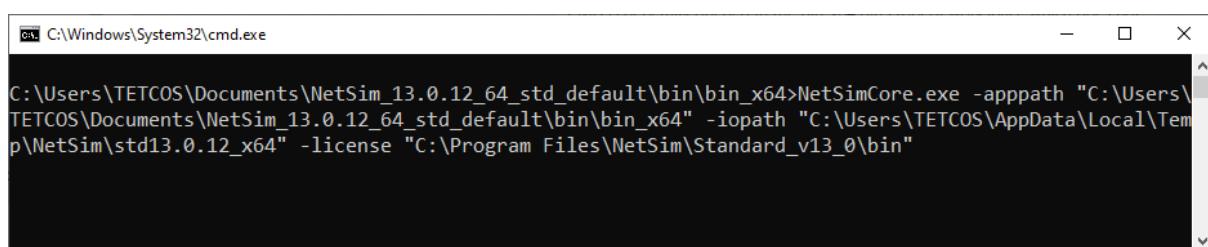
```
>NetSimCore.exe<space>apppath<space><apppath><space>iopath<space><io
path><space>-license<space><license file path>
```

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\Standard\_v13\_0\bin

The following screenshot is the example of running NetSim through CLI for the node locked or cloud license.



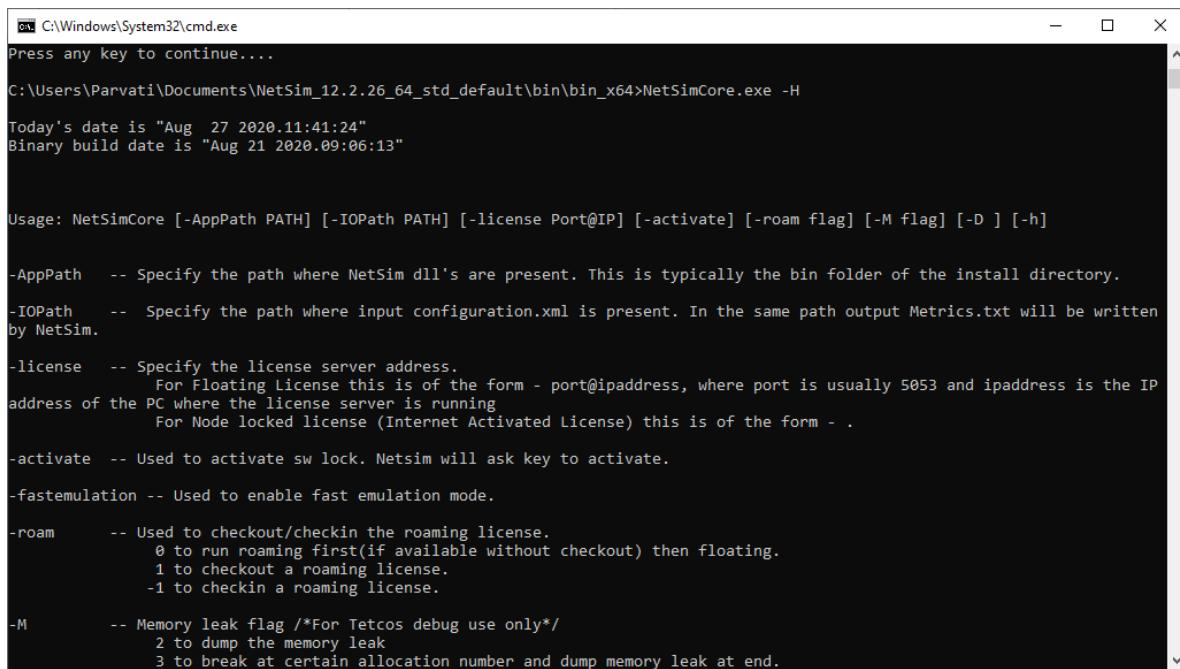
**Figure 7-2:** Running NetSim through CLI mode for Cloud and Node lock licenses

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**



```
C:\Windows\System32\cmd.exe
Press any key to continue.....
C:\Users\Parvati\Documents\NetSim_12.2.26_64_std_default\bin\bin_x64>NetSimCore.exe -H
Today's date is "Aug 27 2020.11:41:24"
Binary build date is "Aug 21 2020.09:06:13"

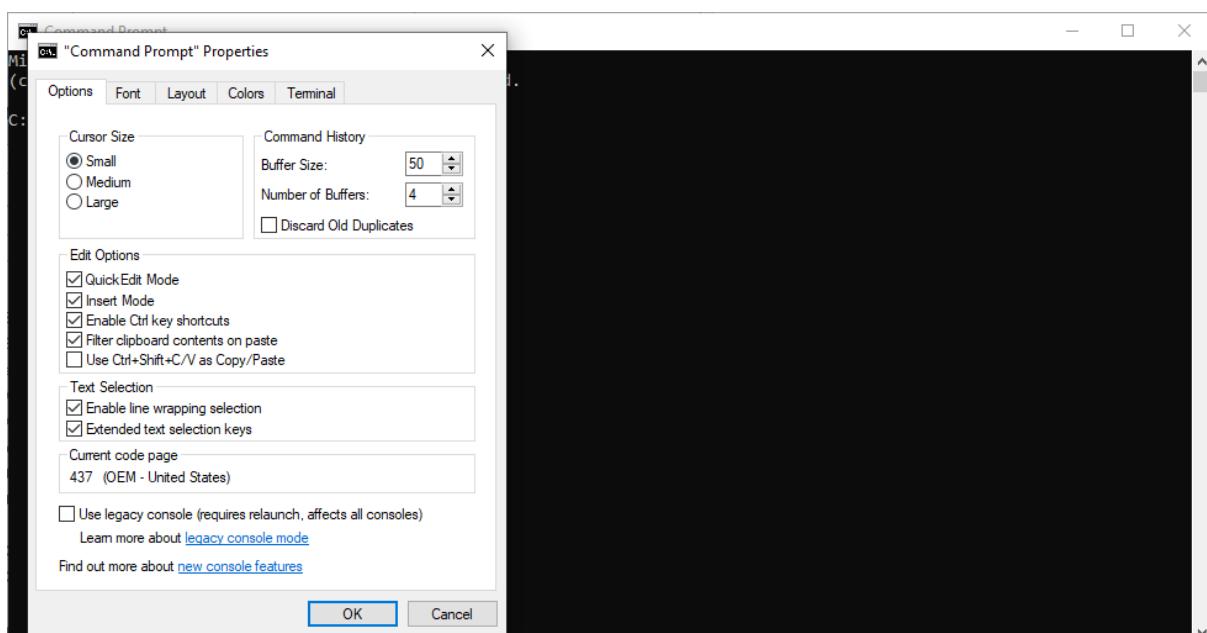
Usage: NetSimCore [-AppPath PATH] [-IOPath PATH] [-license Port@IP] [-activate] [-roam flag] [-M flag] [-D ] [-h]

-AppPath -- Specify the path where NetSim dll's are present. This is typically the bin folder of the install directory.
-IOPath -- Specify the path where input configuration.xml is present. In the same path output Metrics.txt will be written by NetSim.
-license -- Specify the license server address.
          For Floating License this is of the form - port@ipaddress, where port is usually 5053 and ipaddress is the IP address of the PC where the license server is running
          For Node locked license (Internet Activated License) this is of the form - .
-activate -- Used to activate sw lock. Netsim will ask key to activate.
-fastemulation -- Used to enable fast emulation mode.
-roam -- Used to checkout/checkin the roaming license.
          0 to run roaming first(if available without checkout) then floating.
          1 to checkout a roaming license.
          -1 to checkin a roaming license.
-M -- Memory leak flag /*For Tetcos debug use only*/
      2 to dump the memory leak
      3 to break at certain allocation number and dump memory leak at end.
```

**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 writes 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.

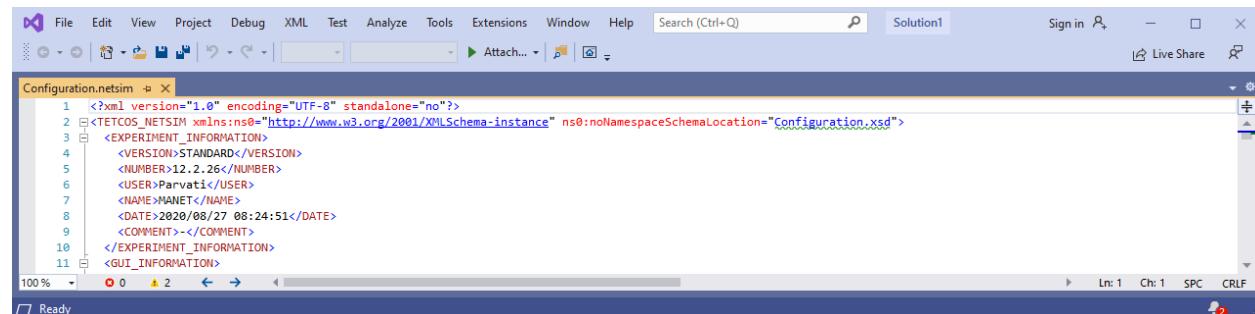
In order to run NetSim through command line (CLI), the user must create the Configuration.netsim file furnishing all the details about the devices, links and the environment of the desired scenario.

### 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.**



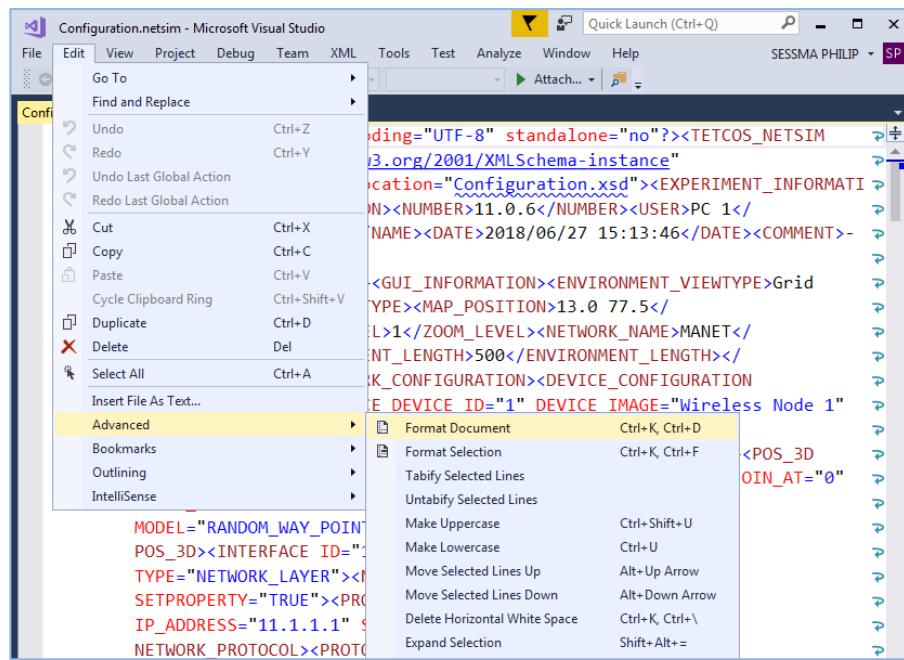
```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TETCOS_NETSIM xmlns:s0="http://www.w3.org/2001/XMLSchema-instance" ns0:noNamespaceSchemaLocation="Configuration.xsd">
  <EXPERIMENT_INFORMATION>
    <VERSION>STANDARD</VERSION>
    <NUMBER>12.2.26</NUMBER>
    <USER>Parvati</USER>
    <NAME>MANET</NAME>
    <DATE>2020/08/27 08:24:51</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 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\_13.0.14\_64\_std\_default \bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default \bin\bin\_x86> for 32-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.

# 8 Outputs: Results, Plots and Data Files

## 8.1 Result Window and Plots Windows

The results of a simulation run are presented in a unified dashboard for convenient analysis. The graphical displays are application throughputs, link throughputs, buffer occupancy and TCP congestion windows. The tabular presentation includes end-to-end delays, jitter, errors, packets generated / received / collided, route tables, TCP Acknowledgments, retransmissions etc.

Results are organized per interface, per device, per application and per link. In addition, summary metrics are aggregated and presented system-wide (network-level). Information in the trace files contain individual packet flow and individual event execution. Protocol log files records a myriad of information pertaining to protocol operation necessary for in-depth analysis and debugging.

The results can be exported as a .csv file and opened in a spread sheet software like Excel. Results can also be exported in .html format and opened in a browser.

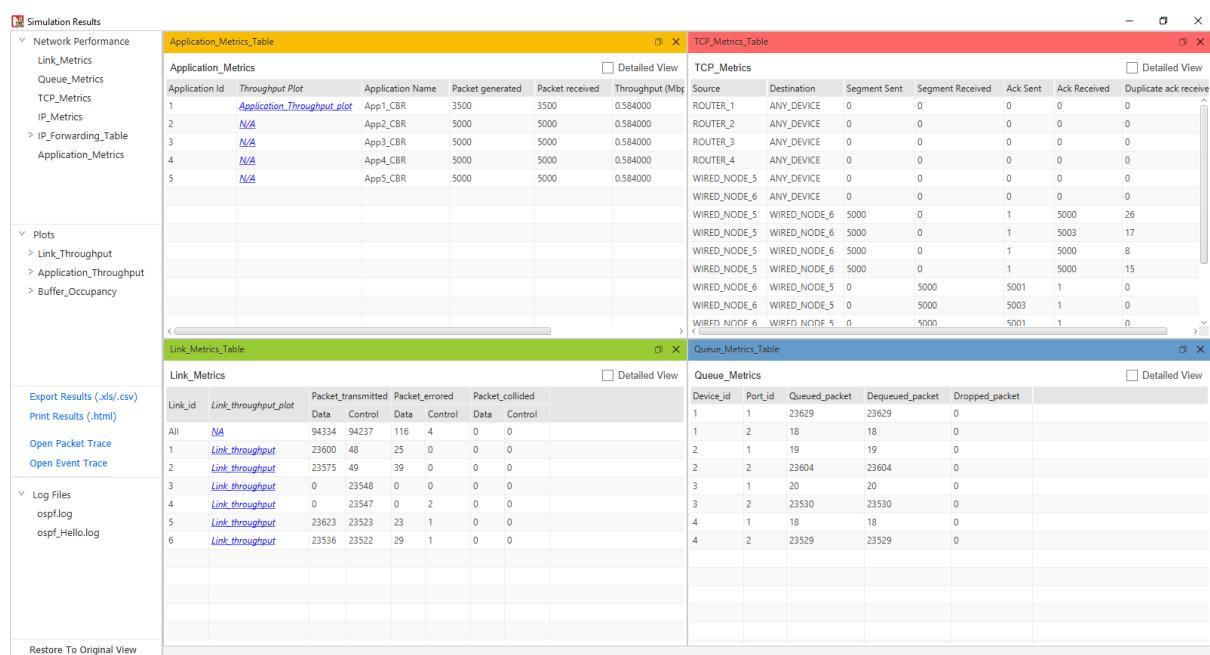
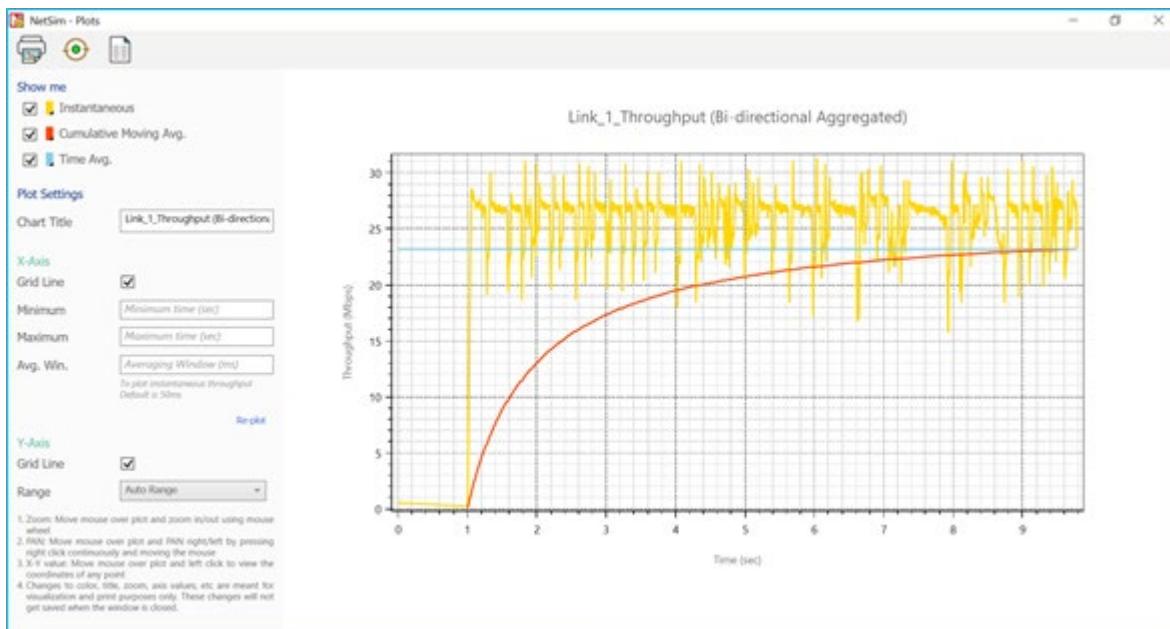


Figure 8-1: Result Window

### 8.1.1 Application and Link Throughput Plots

If plots are enabled, NetSim plots Instantaneous (50 ms averaging window), Cumulative moving average and Time Average for link and application throughputs.



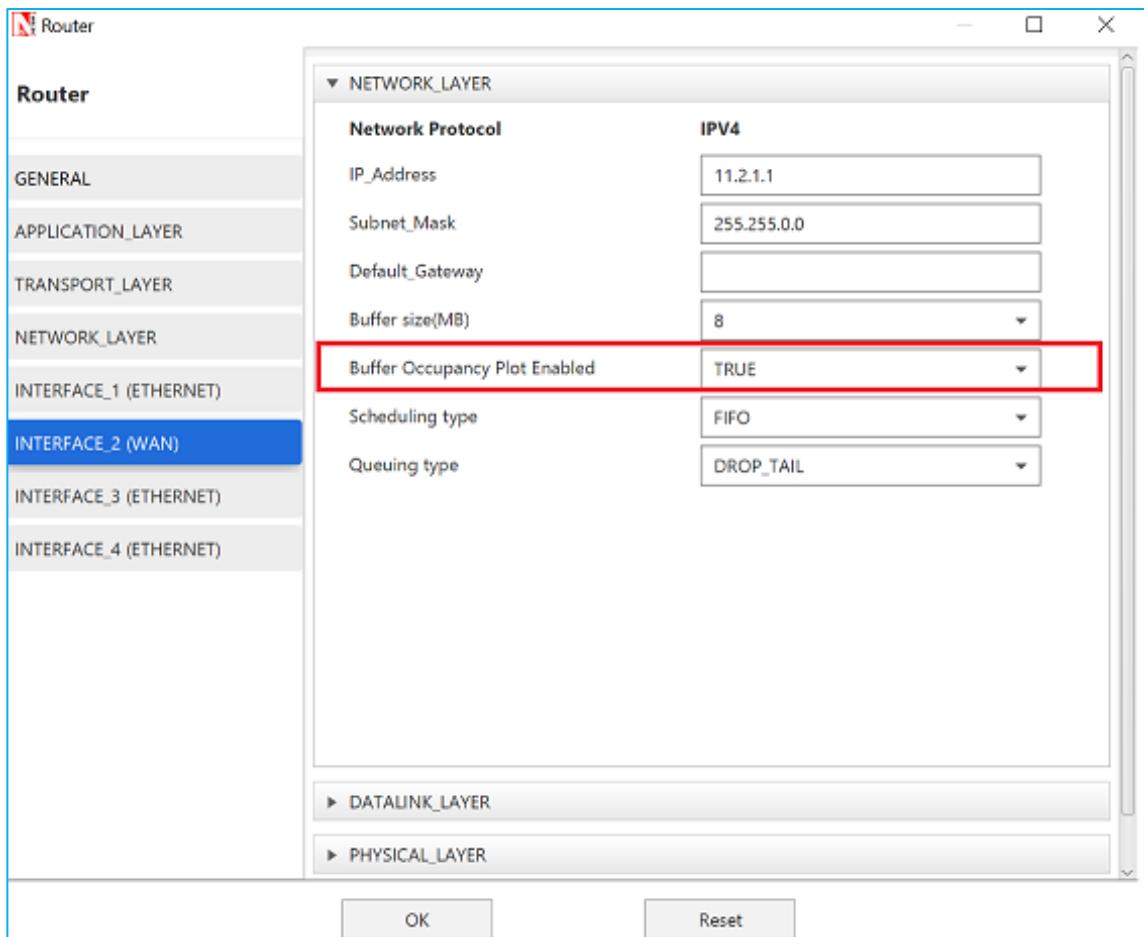
**Figure 8-2:** Link Throughput plot

Guidance for Zooming, Panning, and obtaining XY co-ordinate value are providing on the bottom left of the window.

The ‘re-plot’ option can be used to change the X-value Min and Max, and to change the averaging window for plotting the instantaneous throughput.

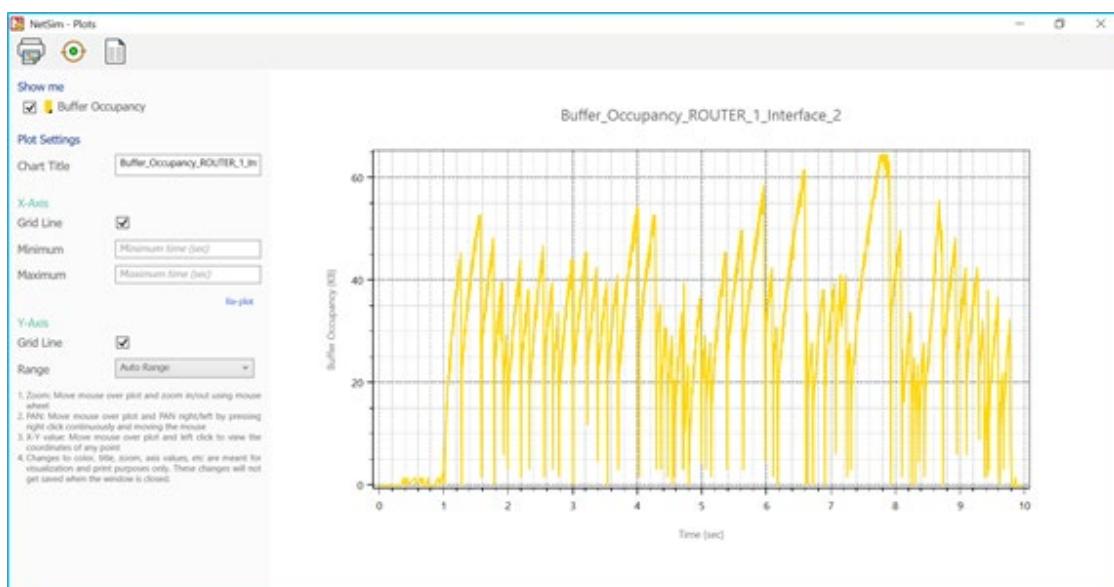
### 8.1.2 Buffer Occupancy Plot

The buffer occupancy over time can be plotted by setting *Buffer Occupancy Plot Enabled* to True. This parameter is available wherever there are buffers in NetSim such as in Router – WAN Port – Network Layer as shown **Figure 8-3**.



**Figure 8-3:** Buffer Occupancy Plot Enabled to True in WAN Port – Network Layer

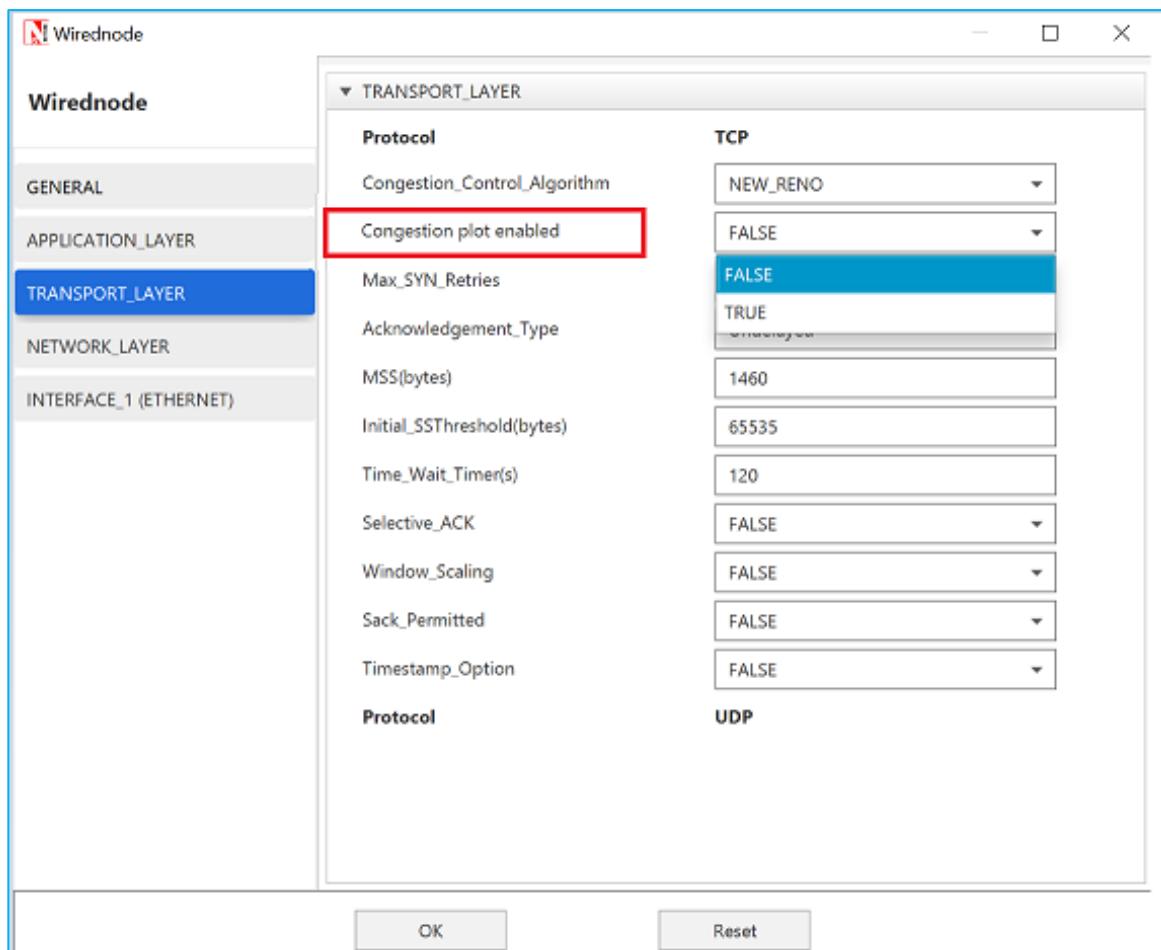
Upon simulation the buffer occupancy plot can be opened from the Results Window and would look like what is shown below **Figure 8-4**.



**Figure 8-4:** Buffer occupancy plot

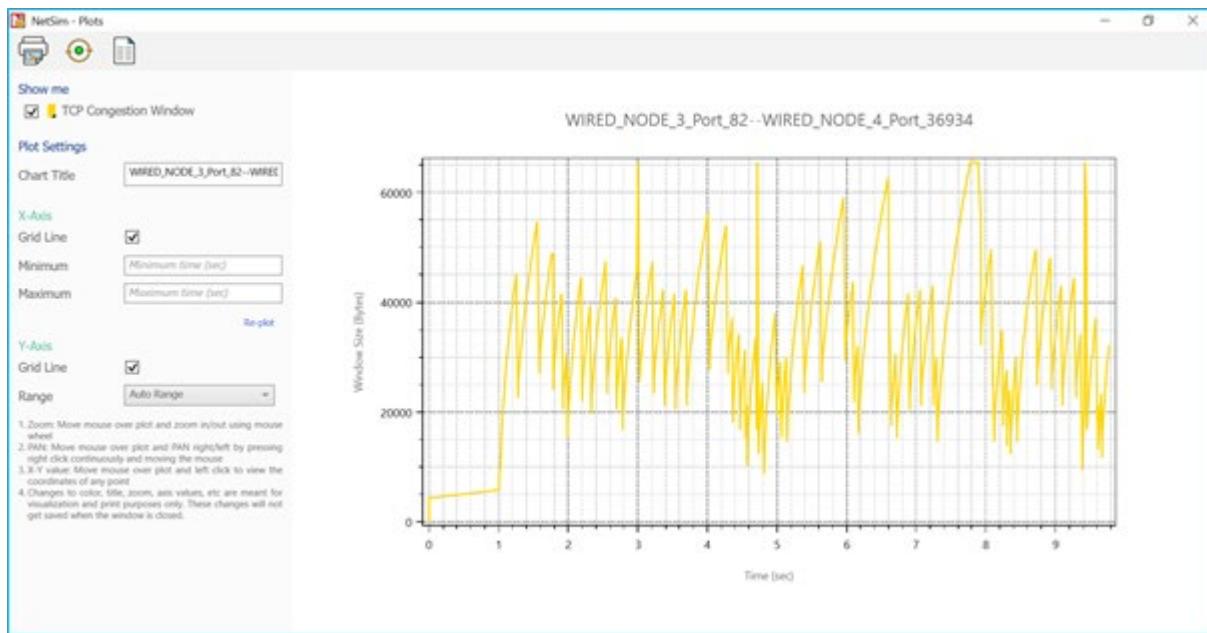
### 8.1.3 TCP Congestion Window Plot

The TCP Congestion window over time can be plotted by setting *Congestion Window Plot Enabled* to True. This parameter is available wherever in the end nodes in NetSim where TCP has been enabled, for example Wired Node – Transport Layer.



**Figure 8-5:** Set TCP Congestion Plot Enabled to True in Transport Layer

Upon simulation the TCP congestion window plot can be opened from the Results Window and would look like what is shown below **Figure 8-6**.



**Figure 8-6:** TCP Congestion window plot

The downsampling algorithm in NetSim plot engine leads to some approximations while plotting, especially in TCP. To see a precise TCP congestion plot window please use Wireshark.

#### 8.1.4 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 these n points.
  - To get a more precise plot users can select the min and max values in the time series and replot.
2. The link throughput is calculated as the sum of throughputs in both directions for a full duplex link.
3. Application throughput is plotted till the last packet reaches or end of simulation time, whichever is earlier.
4. Cumulative Moving Average: This is the average of the metric up until the current time and is defined as.

$$\overline{\theta}(t) = \frac{1}{t} \int_0^t r(u) du$$

#### 8.1.5 Link metrics

Here users can view the values of the metrics obtained based on the overall network and also displays the values of the metrics pertaining to each link.

- **Link\_Id** - It is the unique Id for the link.
- **Link\_throughput\_graph** – Plots throughput vs. Simulation time

#### **Formula:**

$$\text{Link Throughput (in Mbps)} = \frac{\text{Total bytes transmitted over the link} * 8}{\text{Simulation Time (Micro sec)}}$$

Total Bytes transmitted is counted for successful data packets and control packets.

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. Error and collision packets are not included in this calculation and only successful packets are counted for calculation of this metric.

- **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.
- **Packets\_errorred** - 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.
- **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.

#### **8.1.6 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.

### 8.1.7 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.

### 8.1.8 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.

### 8.1.9 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.

#### 8.1.10 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).

#### 8.1.11 Sensor 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 Guarantee time slots. If GTS is allocated, nodes can transmit without contention.

### 8.1.12 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.
- **Transmission Energy** - It is the energy consumed by the respective sensor for transmitting data.

$$\text{TransmissionEnergy (mJ)} = \text{TransmitCurrent (mA)} \times \text{Voltage (V)} \times \text{AmountOfTimeRadioIsInTransmitState(s)}.$$

- **Receiving Energy** - It is the energy consumed by the respective sensor while receiving data.

$$\text{ReceivingEnergy (mJ)} = \text{ReceiveCurrent (mA)} \times \text{Voltage (V)} \times \text{AmountOfTimeRadioIsInReceiveState(s)}.$$

- **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.

$$\text{IdleEnergy (mJ)} = \text{IdleCurrent (mA)} \times \text{Voltage (V)} \times \text{AmountOfTimeRadioIsInIdleState(s)}.$$

- **Sleep Energy** - This is the energy consumed when the respective sensor is in an inactive mode.

$$\text{SleepEnergy (mJ)} = \text{SleepCurrent (mA)} \times \text{Voltage (V)} \times \text{AmountOfTimeRadioIsInSleepState(s)}.$$

## 8.1.13 CR metrics

Displayed if 802.22 cognitive radio is running in the network.

### 8.1.13.1 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.

### 8.1.13.2 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.

#### 8.1.13.3 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.

#### 8.1.13.4 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.

#### 8.1.14 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

$$\text{Application Throughput}(Mbps) = \frac{\text{Total payload delivered to destination (bytes)} * 8}{\text{Time last received packet at App layer}(\mu s) - \text{App Start Time}(\mu s)}$$

If Simulation Time < Application End Time, then

$$\text{Application Throughput(in Mbps)} = \frac{\text{Total payload delivered to destination (bytes)} * 8}{\text{Simulation Time}(\mu s) - \text{App Start Time}(\mu s)}$$

- **Jitter**

$$\text{Jitter}(\mu s) = \frac{\text{TotalPacket Jitter of all successfull packets}}{\text{Total Number of successfully recieved packets} - 1}$$

$$\text{Packet Jitter } (\mu s) = |\text{EndtoEnd Delay of Current packet} - \text{EndtoEnd Delay of Previous Packet}|$$

- **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 later. It would APP\_IN time at destination – APP\_OUT time at source.

### 8.1.15 IP Metrics

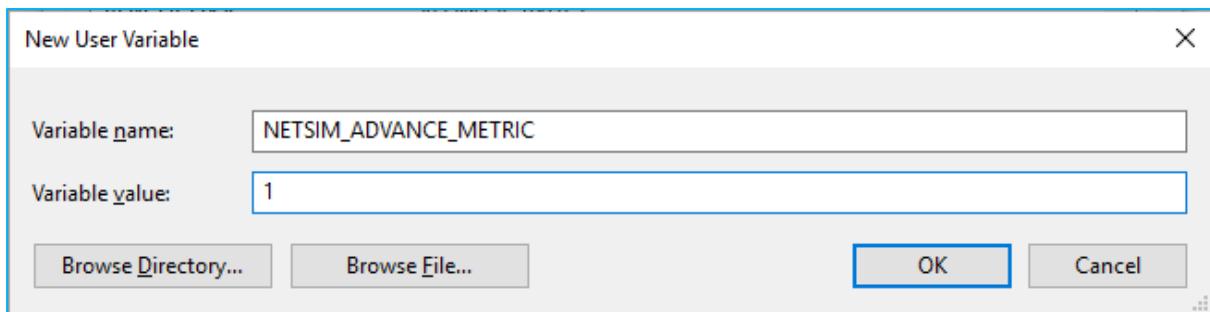
IP layer metrics calculated for the overall network and displayed for each device.

- **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.

### 8.1.16 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 8-7:** Environment Variables window

The Application metrics table in the results dashboard will display an additional column –

Duplicate packet received as shown below **Figure 8-8**

Application_Metrics_Table								
Application_Metrics								
Application Id	Throughput Plot	Application Name	Source Id	Destination Id	Packet generated	Packet received	Duplicate packet received	Payload generated (bytes)
1	<a href="#">Application_Throughput_plot</a>	App1_SENSOR_APP	1	3	100	240	162	5000

**Figure 8-8:** Application metrics table in results window

**Note:** Note that keeping track of duplicate packets will slow down the simulation

### 8.1.17 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 / Multicast 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/Multicast application metrics is architected in NetSim.

### 8.1.18 The different results files written at the end of simulation

The following table lists the various files that will be written in the NetSim install directory/ IO path on completion of simulation.

S. No	File	Contents
1	Metrics.xml	Contains the metrics of the network that is simulated recently.
2	Node.pcap	Contains the information of captured packets that is recently simulated.
3	LicenseErrorLog.txt	Contains the status of the communication between the NetSim dongle and the client

<b>4</b>	ConfigLog.txt	This file will be written while reading the Configuration file. Provides errors if there are errors in the configuration file.
<b>5</b>	LogFile.txt	Contains the logs as the control flows across various layers in the Network Stack
<b>6</b>	PacketTrace.csv	Contains the detailed packet information. This file will be written only when Packet Trace is enabled.
<b>7</b>	EventTrace.csv	Contains the information about each event. This file will be written only when Event Trace is enabled.
<b>8</b>	Animation.txt	Contains the information about the flow of the packet.
<b>9</b>	Static ARP.txt	Contains the information about the dropped devices like Ip address and mac address.

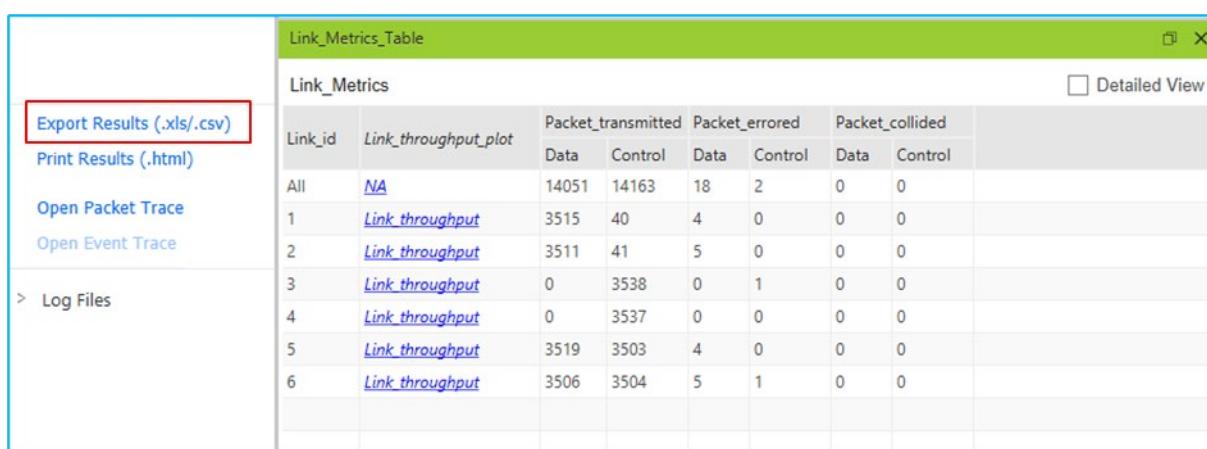
**Table 8-1:** Different results files written at the end of simulation in I/O Path

If NetSim runs via the UI, then the metrics will be displayed automatically at the end of simulation with illustrative tables.

If NetSim runs via CLI, then the metrics will be written into Metrics.txt and MetricsGraph.txt.

## 8.2 Export to .csv

In NetSim Result Dashboard, users can use the option **Export Results (.xls/.csv)** to export all the metrics file to XL/CSV file for the further computation or analysis using it.



Link_Metrics_Table						
Link_Metrics						
Link_id	Link_throughput_plot	Packet_transmitted		Packet_errored		Packet_collided
		Data	Control	Data	Control	
All	<a href="#">NA</a>	14051	14163	18	2	0 0
1	<a href="#">Link_throughput</a>	3515	40	4	0	0 0
2	<a href="#">Link_throughput</a>	3511	41	5	0	0 0
3	<a href="#">Link_throughput</a>	0	3538	0	1	0 0
4	<a href="#">Link_throughput</a>	0	3537	0	0	0 0
5	<a href="#">Link_throughput</a>	3519	3503	4	0	0 0
6	<a href="#">Link_throughput</a>	3506	3504	5	1	0 0

**Figure 8-9:** 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)	
All	NA	300	543	0	0	0	486948	438000	48948	
1	Link_throughput	100	170	0	0	0	163900	146000	17900	
2	Link_throughput	100	203	0	0	0	159148	146000	13148	
3	Link_throughput	100	170	0	0	0	163900	146000	17900	

Queue_Metrics				
Device_id	Port_id	Queued_r	Dequeued_r	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 Addi	Remote A	Syn Sent	Syn-Ack S	Segment t	Segment f	Segment l	Ack Sent	Ack Recei	Duplicate	Out of ord	Duplicate Times	RTC Congestion Plot
WIRED_NODE_1	ANY_DEVICE	11.1.1.20	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	N/A
WIRED_NODE_2	ANY_DEVICE	11.3.1.20	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	N/A
ROUTER_3	ANY_DEVICE	11.1.1.10	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	N/A
ROUTER_4	ANY_DEVICE	11.2.1.20	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	N/A
WIRED_NODE_1	WIRED_NODE_2	11.1.1.28	11.3.1.2:3(	1	0	10	1	0	2	11	0	1	1	N/A
WIRED_NODE_2	WIRED_NODE_1	11.3.1.23	11.1.1.2:8	0	1	0	11	0	11	1	0	0	0	N/A

**Figure 8-10:** Option Export Results (.xls/.csv) to export all the metrics

A web formatted (html file) report can be generated for simulations performed in NetSim, using the Print button present in the results window as shown below **Figure 8-11**.

Link_Metrics						
Link_id	Link_throughput_plot	Packet_transmitted	Packet_errorred	Packet_collided	Data	Control
All	NA	14051	14163	18	2	0
1	Link_throughput	3515	40	4	0	0
2	Link_throughput	3511	41	5	0	0
3	Link_throughput	0	3538	0	1	0
4	Link_throughput	0	3537	0	0	0
5	Link_throughput	3519	3503	4	0	0
6	Link_throughput	3506	3504	5	1	0

**Figure 8-11:** Print Results(.html) in Results window

The report that is generated contains:

- A screenshot of the network scenario created in NetSim GUI.
- All the metrics tables that were part of the Simulation Results Window
- Dynamic Metrics Plots (if Dynamic Metrics is enabled prior to Simulation).

Queue_Metrics				
Device_id	Port_id	Queued_packet	Dequeued_packet	Dropped_packet
3	2	38	38	0
3	3	3870	3870	0
4	1	37	37	0
4	2	37	37	0
5	1	36	36	0
5	3	35	35	0

TCP_Metrics														
Source	Destination	Local Address	Remote Address	Syn Sent	Syn-Ack Sent	Segment Sent	Segment Received	Segment Retransmitted	Ack Sent	Ack Received	Duplicate segment received	Out of order segment received	Duplicate ack received	Times RTO expire
WIRED_NODE_1	ANY_DEVICE	11.1.2.0	0.0.0.0	0	0	0	0	0	0	0	0	0	0	0
WIRED_NODE_2	ANY_DEVICE	11.4.1.2.0	0.0.0.0	0	0	0	0	0	0	0	0	0	0	0
ROUTER_3	ANY_DEVICE	11.1.1.0	0.0.0.0	0	0	0	0	0	0	0	0	0	0	0
ROUTER_4	ANY_DEVICE	11.2.1.2.0	0.0.0.0	0	0	0	0	0	0	0	0	0	0	0
ROUTER_5	ANY_DEVICE	11.3.1.2.0	0.0.0.0	0	0	0	0	0	0	0	0	0	0	0

IP_Metrics						
Device Id	Packet sent	Packet forwarded	Packet received	Packet discarded	TTL expired	Firewall blocked
1	13599	0	0	0	0	0
2	99	0	3826	0	0	0
3	4007	13484	72	0	0	9851
4	74	0	74	0	0	0
5	4002	3832	74	0	0	0

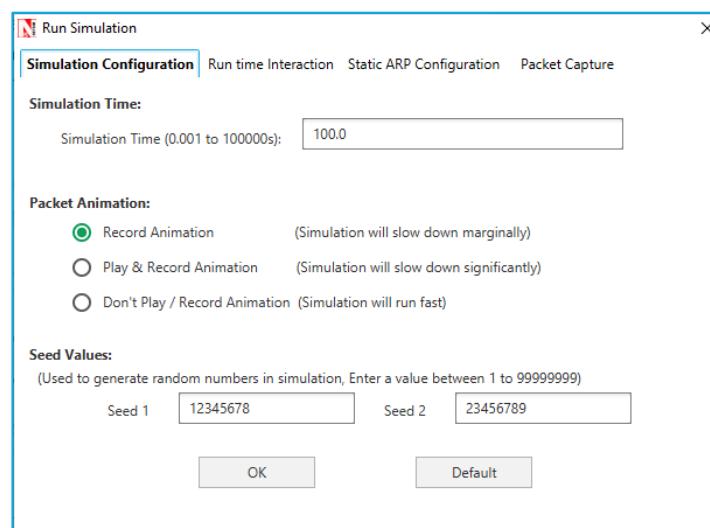
**Figure 8-12:** html report generated in PDF format

- The report that is generated makes it convenient for documentation, reference, study and further analysis.
- This html report can be printed as PDF or printed out by selecting printer options.

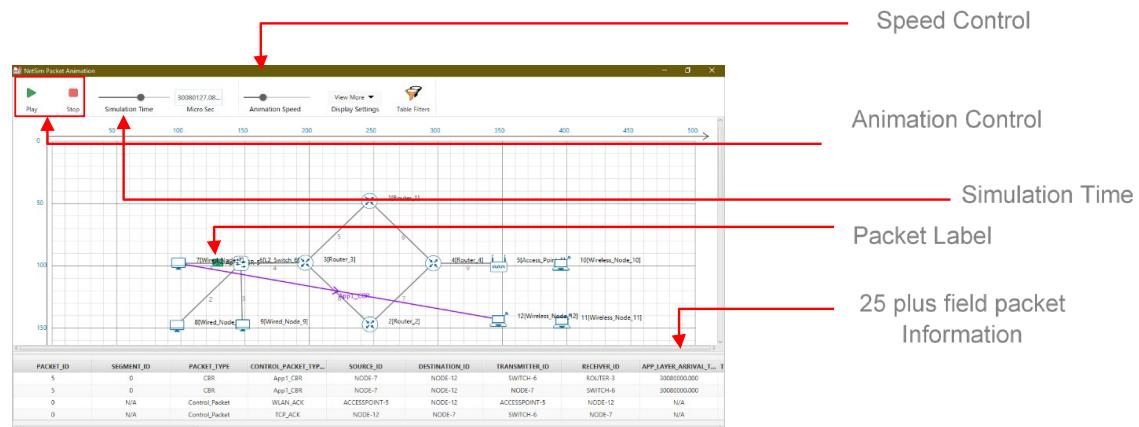
## 8.3 Packet Animation

NetSim provides the feature to play and record animations to the user. Packet animation enables users to watch traffic flow through the network for in-depth visualization and analysis. Users have the following options before running simulation:

- Record the animation.
- Don't play/ record animation and
- Play and record animation while running simulation.

**Figure 8-13:** 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 8-14**.

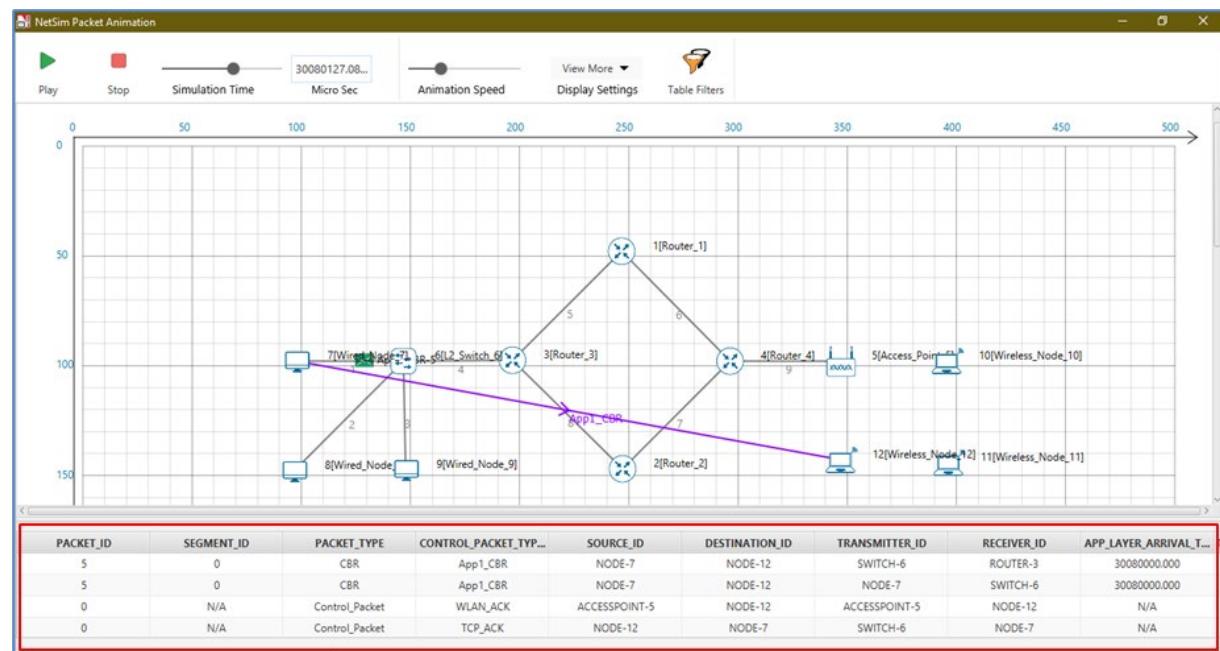


**Figure 8-14:** Packet Animation window

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.

### 8.3.1 Packet animation Table

Packet Animation table is also provided for users to see the flow of packets along with packet animation.



**Figure 8-15:** Packet Animation table in animation window

The “**Table Filters**” option available in the Packet Animator Window allows users to filter the parameters that will be displayed in the Packet Trace Window displayed alongside animation.

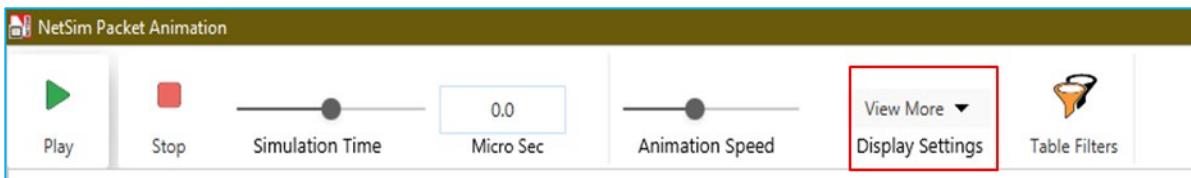


**Figure 8-16:** Table Filters option available in the Packet Animator Window

**Note:** Packet Animation table would be displayed only if Packet Trace is enabled in the network before running the simulation.

### 8.3.2 Packet animation – Display Settings

NetSim Packet Animation can be customized using the View More drop-down list provided with the display settings as shown below **Figure 8-17**.



**Figure 8-17:** Display Settings in Packet animation window

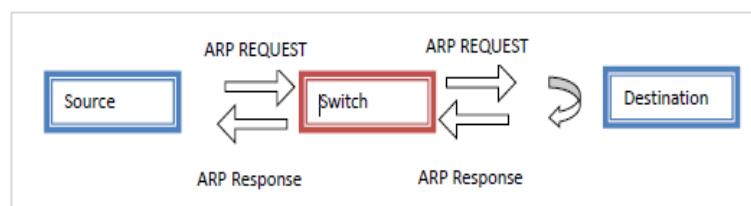
The View More Animation options can be used to view (enable/disable)

- Device Name
- IP address of devices
- VLAN ID
- Application Flow
- Node Movement
- Packet Flow
- Battery Level
- Route tables etc alongside animation

**Note:** The options displayed under **View more** drop down are dependent on the network that is simulated and features that are enabled.

### 8.3.3 Example on how to use NetSim packet animation feature:

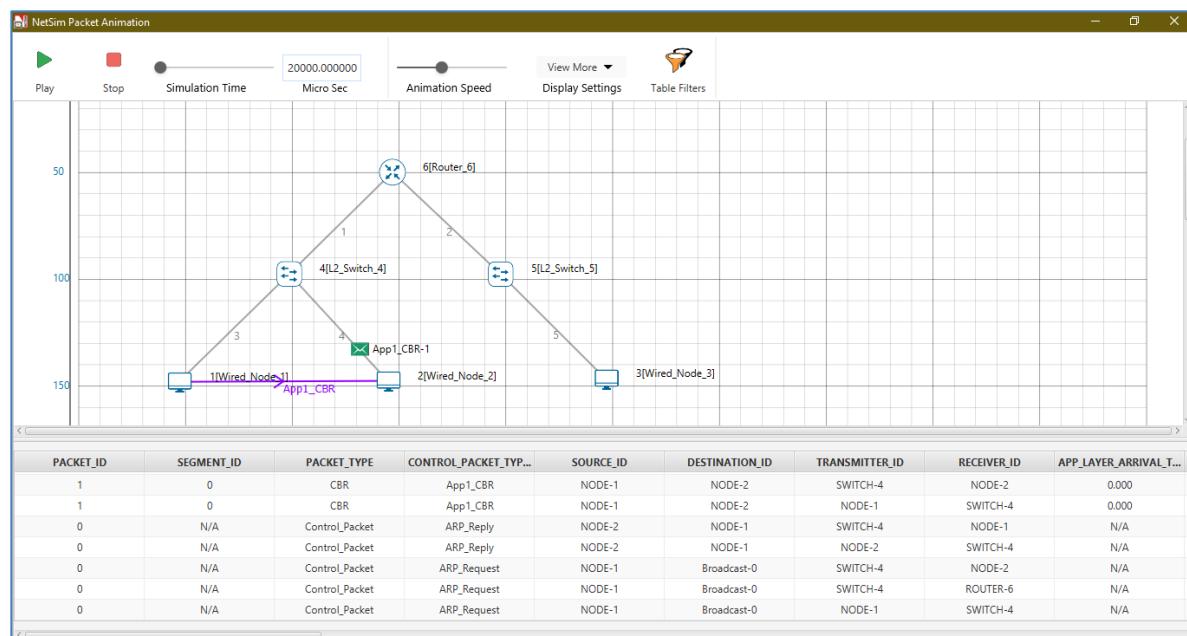
#### Case 1: ARP PROTOCOL - WORKING



**Figure 8-18:** Intra LAN IP Forwarding

- Create a scenario with 3 wired nodes, 2 switches and 1 router and connect it based on the following scenario.
- Disable TCP in all the wired nodes.
- Click on application and set Source\_Id and Destination\_Id as 1 and 2 respectively.
- Set Simulation time = 100s. After clicking on Run Simulation, edit **Static ARP Configuration** tab by setting Static ARP as Disable. Click on OK button to simulate.

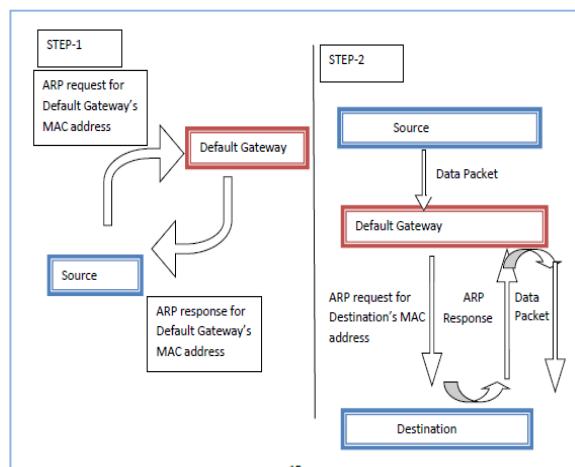
Now click on packet animation and analyse the following:



**Figure 8-19:** Packet animation window

- NODE-1 sends ARP\_Request which is then broadcasted by SWITCH-4.
- During the process the devices that receive the ARP\_Request packet (Switch, Router, and Node-2) will update their ARP table or the switch table.
- NODE -2 sends the ARP\_Reply to NODE-1 via SWITCH-4.
- Now NODE-1 updates its ARP table with the MAC address of NODE-2 on receiving the ARP\_Reply.
- After this step, NODE-1 starts sending data packets to NODE-2 since the source now has both IP and MAC addresses of destination.

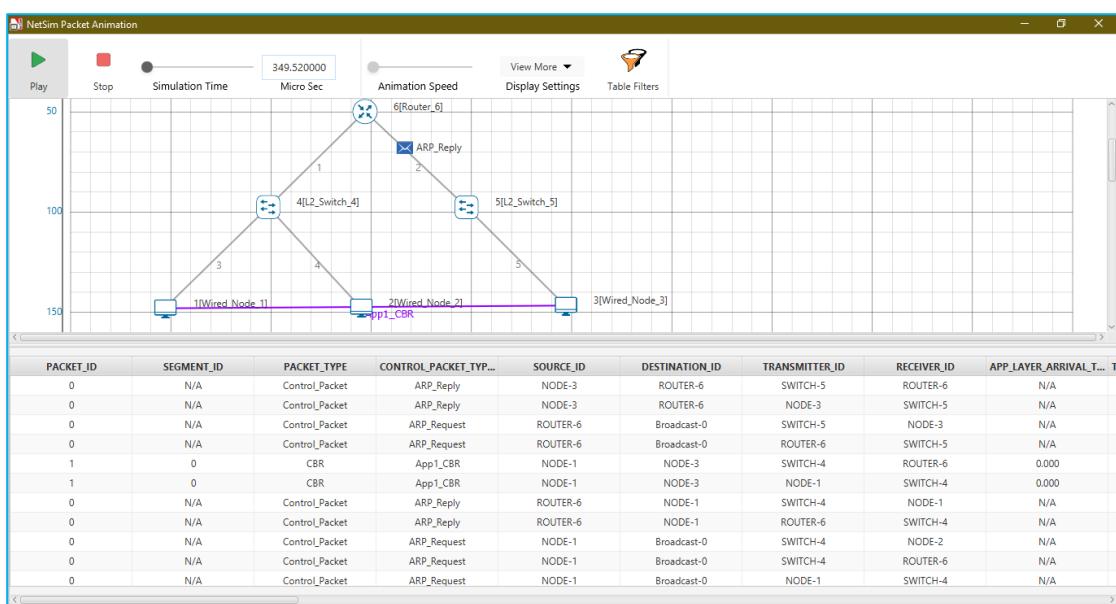
## Case 2: Across-Router-IP-forwarding



**Figure 8-20: Across Router IP Forwarding**

- Follow all the steps till Step 2 and perform the following sample.
- To run the simulation, click on the Application icon and set the Source\_Id and Destination\_Id as 1 and 3 respectively.
- Click on Run Simulation and set Simulation time as 100 sec.
- Then go to **Static ARP Configuration** tab and set Static ARP as Disable. Click on OK button to simulate.

Click on packet animation to analyse the following:



**Figure 8-21: Packet animation window**

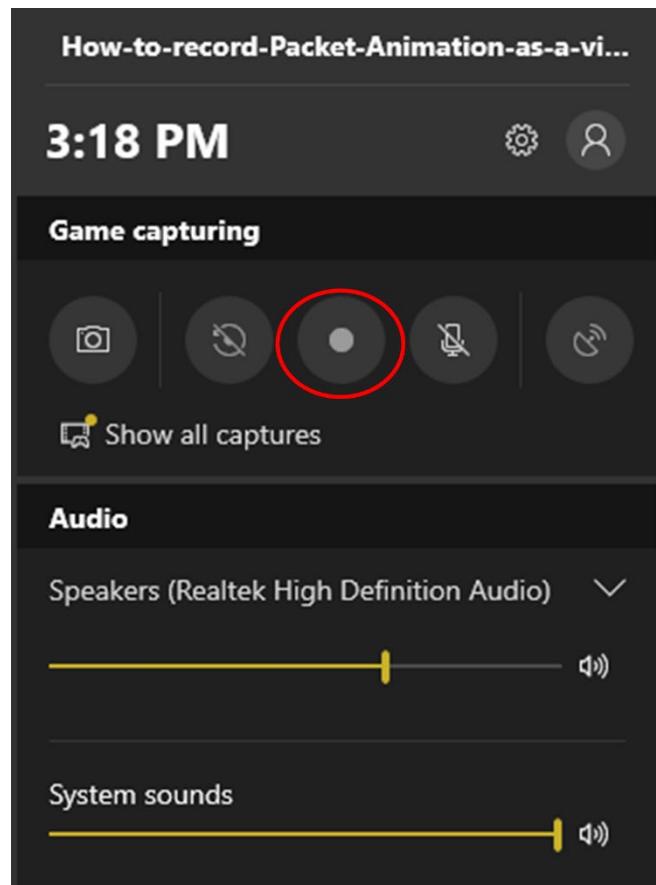
- NODE-1 transmits ARP\_Request which is further broadcasted by SWITCH-4. ROUTER-6 sends ARP\_Reply to NODE-1 which goes through SWITCH-4. Then NODE-1 starts to send data to NODE-3.

- If the router has the address of NODE-3 in its routing table, ARP protocol ends here and data transfer starts that is PACKET\_ID 1 is being sent from NODE-1 to NODE-3.
- In other case, Router sends ARP\_Request to appropriate subnet and after getting the MAC ADDRESS of the NODE-3, it forwards the packet which it has received from NODE-1.
- When a node has to send data to a node with known IP address but unknown MAC address, it sends an ARP request. If destination is in same subnet as the source (found through subnet mask) then it sends the ARP (broadcast ARP message) request, otherwise it forwards it to the default gateway.
- Former case happens in case of intra-LAN communication. The destination node sends an ARP response which is then forwarded by the switch to the initial node. Then data transmission starts.
- In latter case, a totally different approach is followed. Source sends the ARP request to the default gateway and gets back the MAC address of default gateway. (If it knows which router to send then it sends ARP request to the corresponding router and not to Default gateway).
- When source sends data to default gateway (a router in this case), the router broadcasts ARP request for the destined IP address in the appropriate subnet. On getting the ARP response from destination, router then sends the data packet to destination node.

#### 8.3.4 How to record and save Packet animation as a Video file

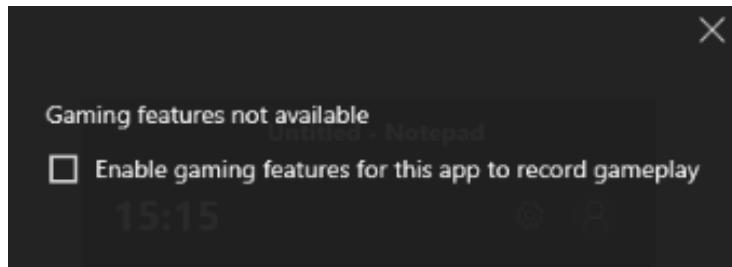
**Note:** The following procedure applies to Windows 10 Operating system only. Users with other versions of Windows can use third-party video capture tools (Link to a list of common tools) to save NetSim packet animation as a video.

To quickly capture NetSim packet animation, launch the packet animation window. Before playing the animation, press **Windows key + G** on the keyboard to open Game bar. (or Select windows settings and then select Gaming option for Game bar related settings). Now start recording by pressing record option as shown below. (Shortcut to start recording **Windows key + Alt + R**)



**Figure 8-22:** In packet animation window press **Windows key + G** on the keyboard and Select Start recording

Then select the checkbox “Enable gaming features for this app to record gameplay” option.

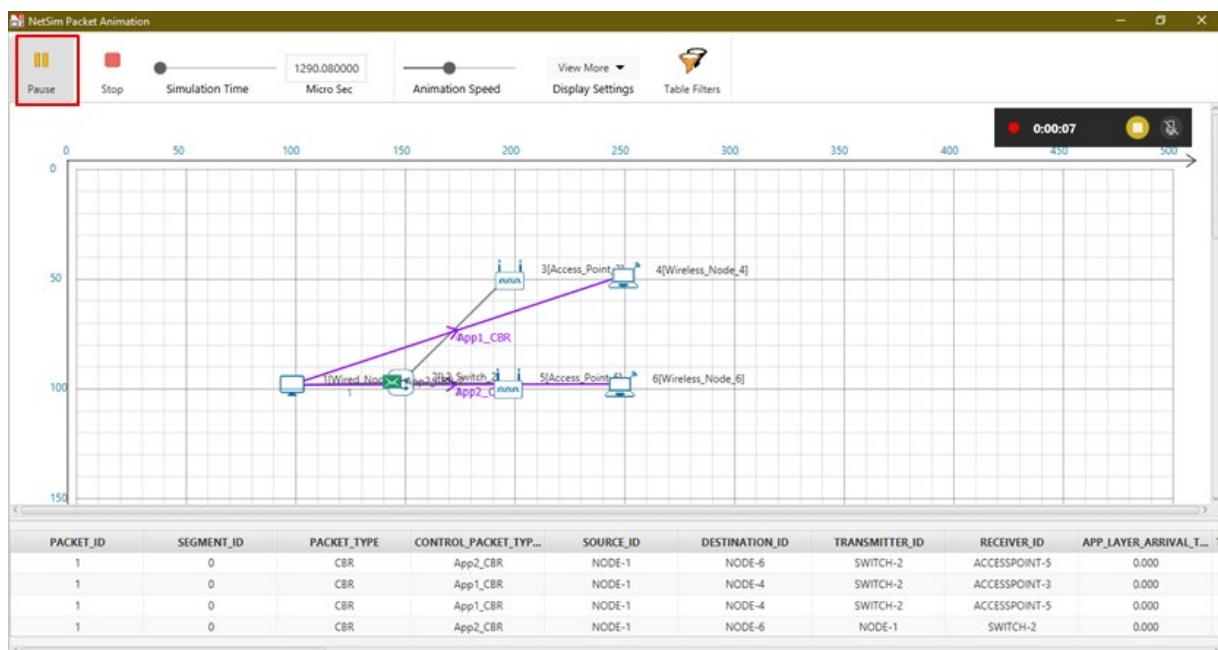


**Figure 8-23:** Select the checkbox “Enable gaming features for this app to record gameplay” option  
Once you select the checkbox, recording window will open as shown below.



**Figure 8-24:** Recording window

Now start playing the animation in NetSim using play button in packet animation window.



**Figure 8-25:** Start playing the animation in animation window

Once the animation has been recorded stop recording (Shortcut to stop recording **Windows logo key + Alt + R**). Recorded clips will be saved in windows default videos folder (E.g.: C:\Users\PC\Videos\Captures).

## 8.4 Packet Trace

NetSim allows users to generate trace files which provide detailed packet information useful for performance validation, statistical analysis and custom code de-bugging. 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:** By default, packet tracing option is turned off. Turning on Packet Trace will slow down the simulation significantly. After simulation, users would get the “open packet trace” link in the metrics window (will also get **Packet\_Trace.csv** file in the saved folder).

#### 8.4.1 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_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
1	0 CBR	App1_CBR		NODE-2	NODE-3	ROUTER-1	NODE-3	0
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A

**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 8-28**.

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	↑ Sort A to Z	N/A	
0 N/A		Control_Packet	TCP_SYN	NODE-2	NODE-3	↓ Sort Z to A	N/A	
0 N/A		Control_Packet	TCP_SYNACK	NODE-3	NODE-2	Sort by Color	N/A	
0 N/A		Control_Packet	TCP_SYNACK	NODE-3	NODE-2	Clear Filter From "RECEIVER_ID"	N/A	
0 N/A		Control_Packet	TCP_ACK	NODE-2	NODE-3	Filter by Color	N/A	
0 N/A		Control_Packet	TCP_ACK	NODE-2	NODE-3	Text Filters	0	
1	0 CBR	App1_CBR		NODE-2	NODE-3	Search 	(Select All)	
1	0 CBR	App1_CBR		NODE-2	NODE-3		<input checked="" type="checkbox"/> NODE-2	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		<input checked="" type="checkbox"/> NODE-3	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		<input checked="" type="checkbox"/> ROUTER-1	
2	0 CBR	App1_CBR		NODE-2	NODE-3		OK	
2	0 CBR	App1_CBR		NODE-2	NODE-3		Cancel	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		...	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		...	
3	0 CBR	App1_CBR		NODE-2	NODE-3		...	
3	0 CBR	App1_CBR		NODE-2	NODE-3		...	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		...	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		...	
4	0 CBR	App1_CBR		NODE-2	NODE-3		...	
4	0 CBR	App1_CBR		NODE-2	NODE-3		...	
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2		...	

**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.

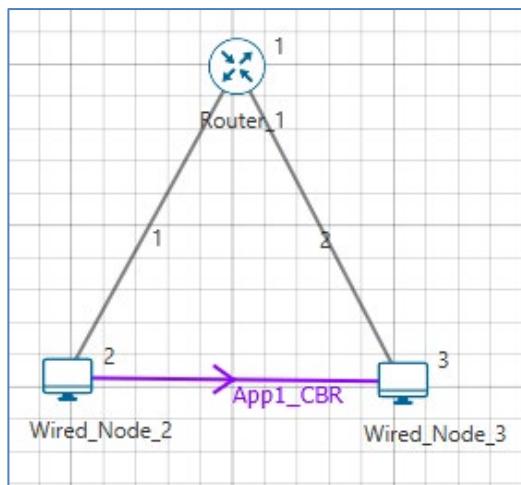
#### 8.4.2 Observing packet flow in the Network through packet trace file

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.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PA	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER
Sort Smallest to Largest		Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
Sort Largest to Smallest		Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3	N/A
Sort by Color		Packet	TCP_SYNACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
Clear Filter From "PACKET_ID"		Packet	TCP_SYNACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A
Filter by Color		Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
Number Filters		Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3	N/A
Search		App1_CBR		NODE-2	NODE-3	NODE-2	ROUTER-1	0
		App1_CBR		NODE-2	NODE-3	ROUTER-1	NODE-3	0
		Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
		Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A
		App1_CBR		NODE-2	NODE-3	NODE-2	ROUTER-1	20000
		App1_CBR		NODE-2	NODE-3	ROUTER-1	NODE-3	20000
		Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
		Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A
		App1_CBR		NODE-2	NODE-3	NODE-2	ROUTER-1	40000
		App1_CBR		NODE-2	NODE-3	ROUTER-1	NODE-3	40000
		Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
		Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A
		App1_CBR		NODE-2	NODE-3	NODE-2	ROUTER-1	60000
		App1_CBR		NODE-2	NODE-3	ROUTER-1	NODE-3	60000
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
0 N/A		Control_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A

**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's 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.

#### 8.4.3 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, analyse, 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.

	A	B	C	D	E	F	G	H	I	J	K
	PACKET_ID	SEGMENT_ID	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_ARRIVAL_TIME(US)	
2	0 N/A		Control_Packet TCP_SYN	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A		20000	
3	0 N/A		Control_Packet TCP_SYN	NODE-1	NODE-2	ROUTER-3	NODE-2	N/A		20000	
4	0 N/A		Control_Packet TCP_SYNACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A		20021.2	
5	0 N/A		Control_Packet TCP_SYNACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A		20021.2	
6	0 N/A		Control_Packet TCP_ACK	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A		20042.4	
7	0 N/A		Control_Packet TCP_ACK	NODE-1	NODE-2	ROUTER-3	NODE-2	N/A		20042.4	
8	1	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	20000		0	
9	1	0 CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	20000		0	
10	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A		20302.8	
11	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A		20302.8	
12	2	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	40000		0	
13	2	0 CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	40000		0	
14	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A		40254.16	
15	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A		40254.16	
16	3	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	60000		0	
17	3	0 CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	60000		0	
18	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A		60254.16	
19	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A		60254.16	
20	4	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	80000		0	
21	4	0 CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	80000		0	
22	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A		80254.16	
23	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A		80254.16	
24	5	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	100000		0	
25	5	0 CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	100000		0	
26	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A		100254.16	
27	0 N/A		Control_Packet TCP_ACK	NODE-2	NODE-1	ROUTER-3	NODE-1	N/A		100254.16	
28	6	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	120000		0	
29	6	0 CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3	NODE-2	120000		0	
30	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.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	Count			DESTINATION_ID										
2	SOURCE_ID	CONTROL_PACKET_TYPE/APP_NAME		DESTINATION_ID										
3	NODE-1	App1_CBR		Errored		2	2							
4				Successful		763	763							
5		App1_CBR Total				765	765							
6		TCP_ACK		Successful		2	2							
7		TCP_ACK Total				2	2							
8		TCP_SYN		Successful		2	2							
9		TCP_SYN Total				2	2							
10	NODE-1 Total					769	769							
11	NODE-2	TCP_ACK		Errored		1	1							
12				Successful		759	759							
13		TCP_ACK Total				760	760							
14		TCP_SYNACK		Successful		2	2							
15		TCP_SYNACK Total				2	2							
16	NODE-2 Total					762	762							
17	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

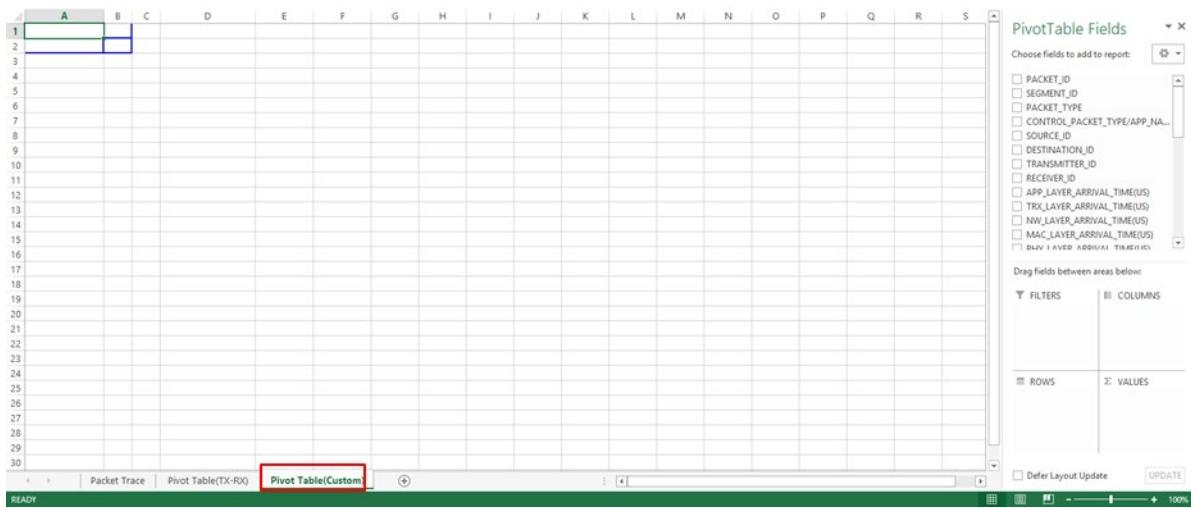
### Steps to analyse the packet trace using pivot tables

**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)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	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_ARRIVAL_TIME(US)								
2	0 N/A	Control_Packet	TCP_SYN	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A				20000							
3	0 N/A	Control_Packet	TCP_SYNACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				20021.2							
4	0 N/A	Control_Packet	TCP_SYNACK	NODE-2	NODE-1	NODE-1	ROUTER-3	N/A				20021.2							
5	0 N/A	Control_Packet	TCP_ACK	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A				20042.4							
6	0 N/A	Control_Packet	TCP_ACK	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A				20042.4							
7	0 N/A	Control_Packet	TCP_ACK	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A				20042.4							
8	1	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	20000			0							
9	1	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A	20000			0							
10	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				20302.8							
11	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-1	ROUTER-3	N/A				20302.8							
12	2	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	40000			0							
13	2	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A	40000			0							
14	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				40254.16							
15	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-1	ROUTER-3	N/A				40254.16							
16	3	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	60000			0							
17	3	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A	60000			0							
18	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				60254.16							
19	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-1	ROUTER-3	N/A				60254.16							
20	4	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	80000			0							
21	4	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A	80000			0							
22	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				80254.16							
23	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-1	ROUTER-3	N/A				80254.16							
24	5	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	100000			0							
25	5	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A	100000			0							
26	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				100254.16							
27	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-1	ROUTER-3	N/A				100254.16							
28	6	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	120000			0							
29	6	0 CBR	App1_CBR	NODE-1	NODE-2	NODE-2	ROUTER-3	N/A	120000			0							
30	0 N/A	Control_Packet	TCP_ACK	NODE-2	NODE-1	NODE-2	ROUTER-3	N/A				120254.16							

**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.

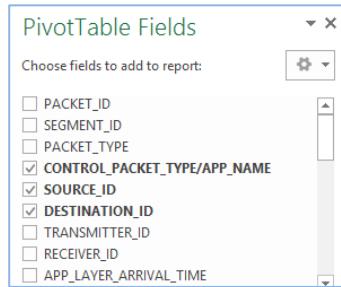


**Figure 8-36:** Select Blank Pivot Table (Custom) to create your own pivot table

Once you open the sheet PivotTable (Custom), you'll 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.

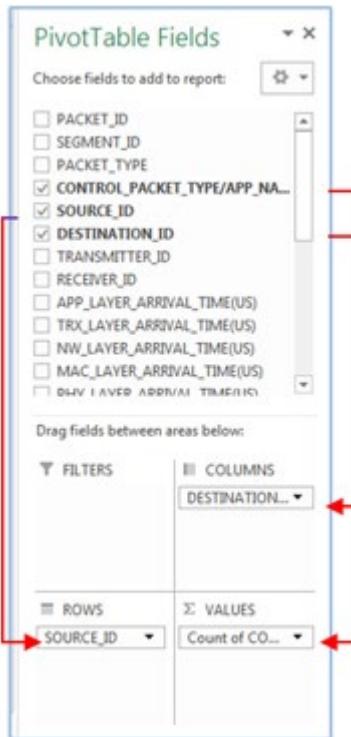
#### 8.4.4 Packet Transmitted / Received Analysis

- If you want to analyse 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.



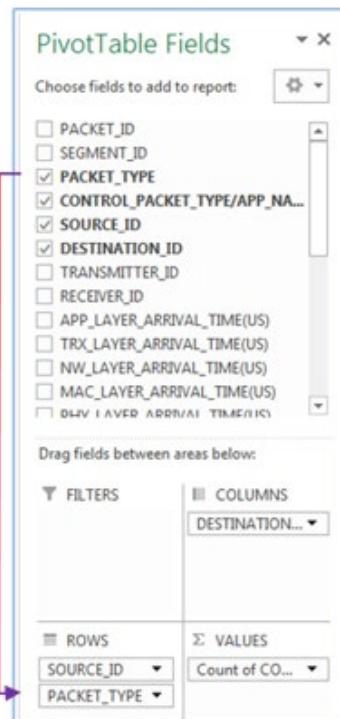
**Figure 8-38:** Selected fields add to one of the four areas below the Field List

- 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

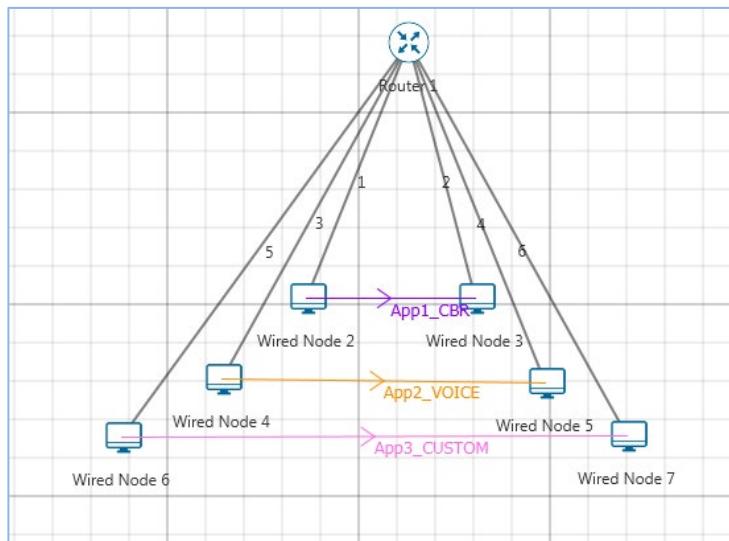
- Further, if you wish to know how many packets got errored and how many were successful, check the PACKET\_STATUS field and drag it to the ROWS field.

Count of CONTROL_PACKET_TYPE/APP_NAME		DESTINATION_ID		
SOURCE_ID	PACKET_TYPE	PACKET_STATUS	NODE-1	NODE-2
NODE-1	CBR	Errored		2
		Successful		350
	CBR Total		352	352
	Control_Packet	Successful		4
		Control_Packet Total	4	4
NODE-1 Total			356	356
NODE-2	Control_Packet	Errored	1	1
		Successful	348	348
	Control_Packet Total		349	349
NODE-2 Total			349	349
Grand Total			349	356

**Figure 8-42:** PivotTable Created with Packet Status

#### 8.4.5 Delay analysis

We explain this using a packet trace generated per the following network scenario.



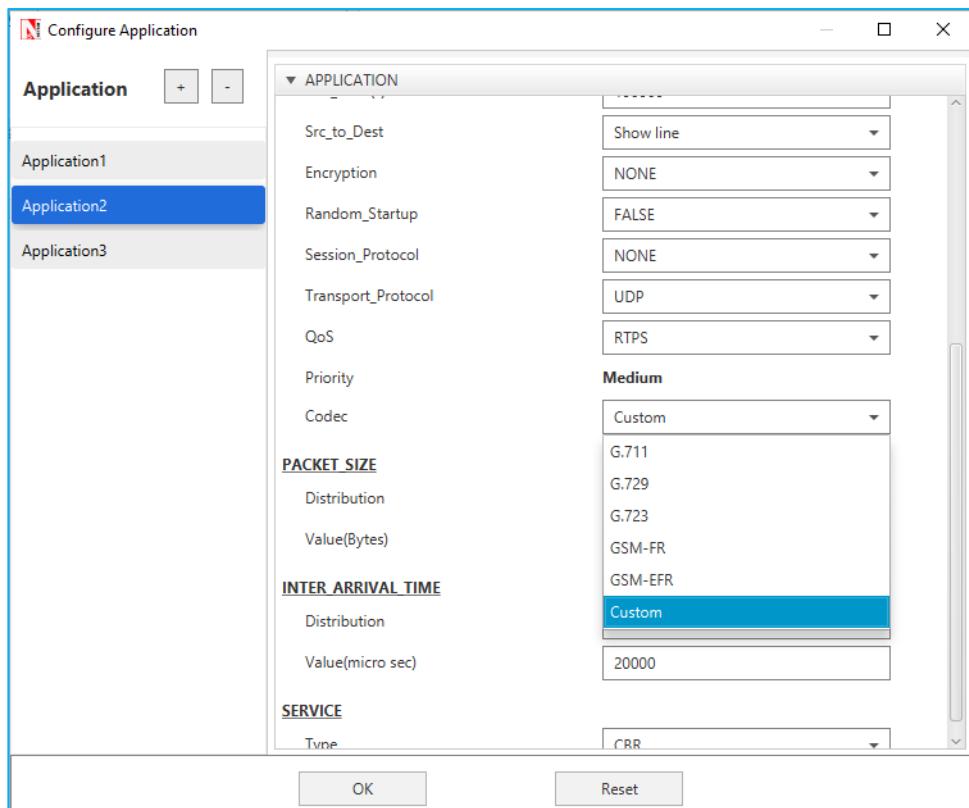
**Figure 8-43:** Network Topology with different application

Create a 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

The screenshot shows a Microsoft Excel spreadsheet with the following data:

	H	I	J	K	L	M	N	O	P
1	RECEIVER_ID	APP_LAYER_ARR	TRX_LAYER_AR	NW_LAYER	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

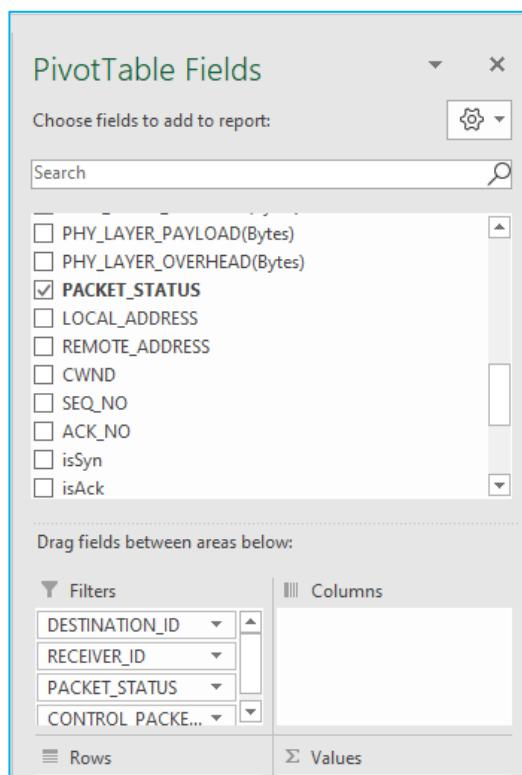
**Figure 8-45:** Calculate delay in packet trace using PHY\_LAYER\_END\_TIME – APPLICATION\_LAYER\_ARRIVAL\_TIME then Press CTRL + ENTER

- Then Press CTRL + ENTER. This will calculate delay for the whole column shown below.

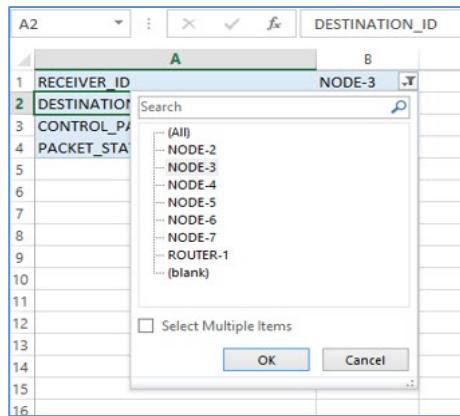
P1	H	I	J	K	L	M	N	O	P	Q
	RECEIVER_ID	APP_LAYER_ARR	TRX_LAYER_AR	NW_LAYER_F	MAC_LAYER	PHY_LAYER_A	PHY_LAYER	PHY_LAYER_EN	Column1	APP_LAYER
1	ROUTER-1	N/A		0	0	0	0.96	6.56	11.56	#VALUE!
2	ROUTER-1	N/A		0	0	0	0.96	6.56	11.56	#VALUE!
3	ROUTER-1	N/A		0	11.56	11.56	11.56	17.16	22.16	#VALUE!
4	NODE-3	N/A		0	11.56	11.56	11.56	17.16	22.16	#VALUE!
5	NODE-7	N/A		0	11.56	11.56	11.56	17.16	22.16	#VALUE!
6	ROUTER-1	N/A		22.16	22.16	22.16	22.16	27.76	32.76	#VALUE!
7	ROUTER-1	N/A		22.16	22.16	22.16	22.16	27.76	32.76	#VALUE!
8	NODE-2	N/A		22.16	32.76	32.76	32.76	38.36	43.36	#VALUE!
9	NODE-6	N/A		22.16	32.76	32.76	32.76	38.36	43.36	#VALUE!
10	ROUTER-1	N/A		43.36	43.36	43.36	43.36	48.64	53.64	#VALUE!
11	ROUTER-1	N/A		43.36	43.36	43.36	43.36	48.64	53.64	#VALUE!
12	NODE-3	N/A		43.36	53.64	53.64	53.64	58.92	63.92	#VALUE!
13	NODE-7	N/A		43.36	53.64	53.64	53.64	58.92	63.92	#VALUE!
14	ROUTER-1	0	0	0	0	0.96	123.68	128.68	128.68	1480
15	ROUTER-1	0	0	0	0	0	124.64	130.56	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

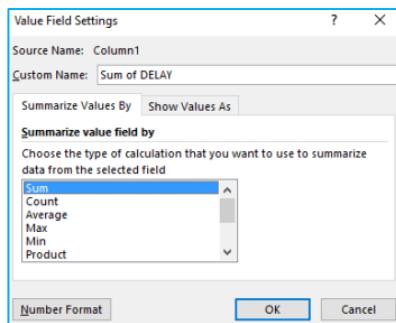
RECEIVER_ID	NODE-3
DESTINATION_ID	NODE-3
CONTROL_PACKET_TYPE/APP_NAME	APP1_CBR
PACKET_STATUS	Successful

**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

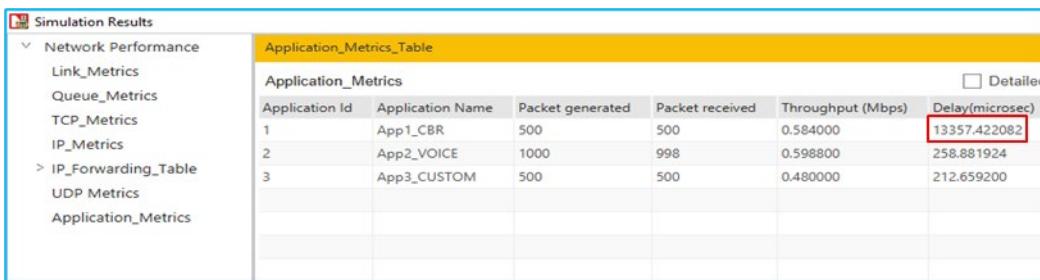
- Select one cell and calculate the Application Delay, which is the average delay faced by a packet by using the formula

$$\text{Application Delay} = \frac{\text{Sum of DELAY of Successful Packets}}{\text{Number of Packets}}$$

	A	B	C	D	E	F	G
1	DESTINATION_ID	NODE-3					
2	RECEIVER_ID	NODE-3					
3	PACKET_STATUS	Successful					
4	CONTROL_PACKET_TYPE/APP_NAME	App1_CBR					
5							
6	Row Labels	Sum of Delay	Count of Delay2				
7	1	303.76	1				
8	2	254.16	1				
9	3	254.16	1				
10	4	254.16	1				
315	309	254.16	1				
316	310	254.16	1				
341	335	254.16	1				
342	336	254.16	1				
343	337	254.16	1				
344	338	254.16	1				
383	377	254.16	1				
500	494	254.16	1				
501	495	254.16	1				
502	496	254.16	1				
503	497	254.16	1				
504	498	254.16	1				
505	499	254.16	1				
506	500	254.16	1				
507	Grand Total	6678711.04	500				
508							

**Figure 8-53:** Calculated the Application Delay in Pivot table

- Compare the obtained value with the DELAY in Application Metrics



Application_Metrics_Table					
Application Id	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)
1	App1_CBR	500	500	0.584000	13357.422082
2	App2_VOICE	1000	998	0.598800	258.881924
3	App3_CUSTOM	500	500	0.480000	212.659200

**Figure 8-54:** Compare the obtained DELAY with Application Metrics DELAY

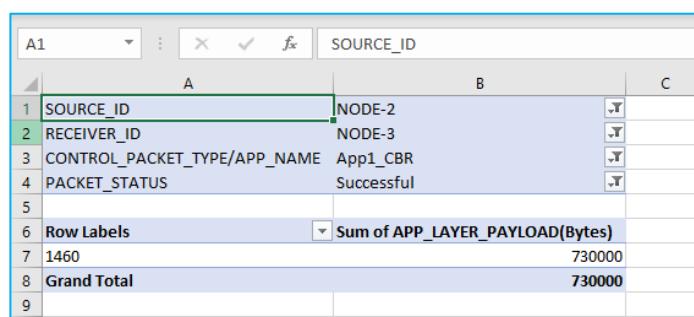
- To calculate DELAY for VOICE application, filter DESTINATION\_ID to Node-5, RECEIVER\_ID to Node-5, CONTROL\_PACKET\_TYPE/APP\_NAME to APP2\_VOICE and PACKET\_STATUS to Successful
- Similarly calculate and compare DELAY for other applications by following the above procedure.

#### 8.4.6 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.



A1	B	C
SOURCE_ID	NODE-2	
RECEIVER_ID	NODE-3	
CONTROL_PACKET_TYPE/APP_NAME	App1_CBR	
PACKET_STATUS	Successful	
Row Labels	Sum of APP_LAYER_PAYLOAD(Bytes)	
1460	730000	
Grand Total	730000	

**Figure 8-55:** Pivot Table

- Select 1 cell and calculate the throughput by using the formula.

$$\text{Application throughput (Mbps)} = \frac{\text{Sum of Successfully Received App Layer Payload (Bytes)} * 8}{\text{Simulation Time (\mu s)}}$$

	A	B	C	D	E
1	SOURCE_ID	NODE-2			
2	RECEIVER_ID	NODE-3			
3	CONTROL_PACKET_TYPE/APP_NAME	App1_CBR			
4	PACKET_STATUS	Successful			
5					
6	Row Labels	Sum of APP_LAYER_PAYLOAD(Bytes)			
7	1460		730000		
8	Grand Total		730000	0.58400	
9					

**Figure 8-56:** Calculate the throughput by using the formula in Pivot Table

EmptyCell=GETPIVOTDATA("APP\_LAYER\_PAYLOAD(Bytes)",\$A\$6,"APP\_LAYER\_PAYLOAD(Bytes)",1460)\*8/10000000

- Now compare the throughput calculated using pivot table with the Application Metrics throughput.

Application Id	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)
1	App1_CBR	500	500	0.584000	13357.422082
2	App2_VOICE	1000	998	0.598800	258.881924
3	App3_CUSTOM	500	500	0.480000	212.659200

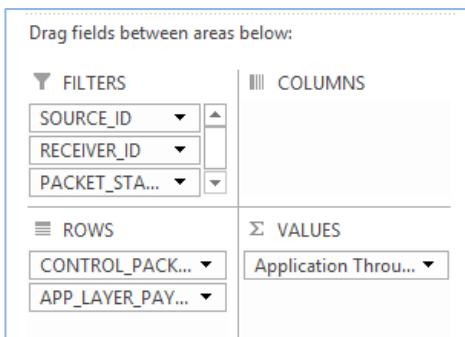
**Figure 8-57:** Compared the calculated throughput using pivot table with the Application Metrics throughput

- 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.

#### 8.4.7 Plotting with Pivot Charts

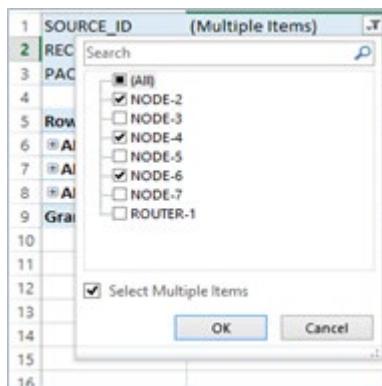
In a pivot table, you can create a new field that performs a calculation on the sum of other pivot fields.

- 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 and APP\_LAYER\_PAYLOAD to VALUES field as shown below **Figure 8-58**.



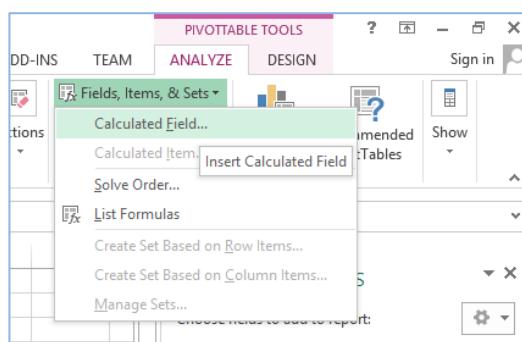
**Figure 8-58:** Drag and drop Selected Fields to one of the four areas  
below the Field List

- 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



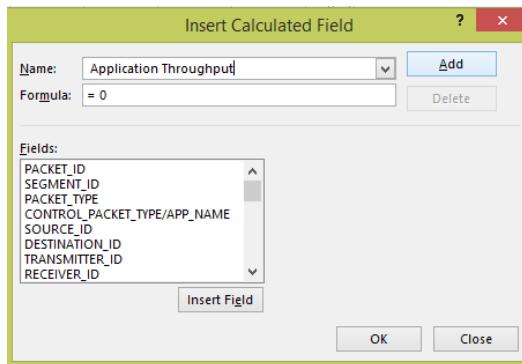
**Figure 8-59:** Filter Source ID and Destination ID

- 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 (Analyse 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

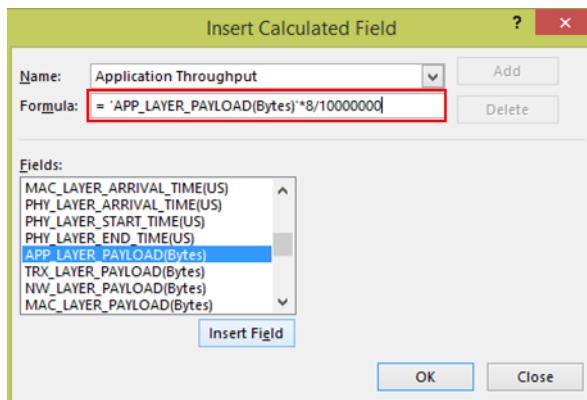
- Type a name for the calculated field, Application Throughput.
- Then click on ADD to save the calculated field.



**Figure 8-61:** Insert calculated field name and select Add

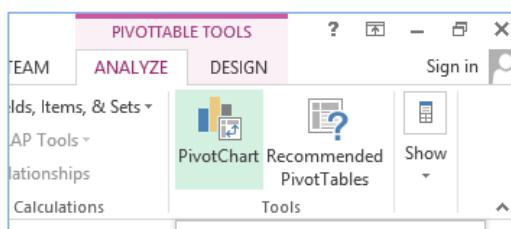
- Click on Formula text box and then select APP\_LAYER\_PAYLOAD in the Fields list and click on Insert Field.
- Calculate the throughput by using the following formula shown below and click on OK.

$$\text{Application throughput (Mbps)} = \frac{\text{Sum of Successfully Received App Layer Payload (Bytes) * 8}}{\text{Simulation Time (\mu s)}}$$



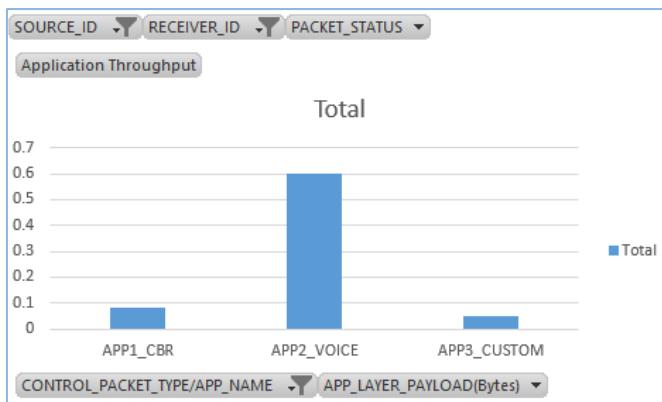
**Figure 8-62:** Calculate the throughput by using the following formula

- Select a cell in the pivot table, and on the Excel Ribbon, under the PivotTable Tools tab, click the Options tab (Analyze tab in Excel 2013).
- In the Tools group, click Pivot chart and select OK.



**Figure 8-63:** In the Tools group Select Pivot chart

- This will display a pivot chart shown below.

**Figure 8-64:** Pivot Chart

**(Note:** The procedure may vary with different versions of excel, the given procedure is according to the Excel 2013.)

#### 8.4.8 Packet Trace Fields

GENERAL FIELDS	DESCRIPTION
<b>PACKET_ID</b>	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.
<b>SEGMENT_ID</b>	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
<b>PACKET_TYPE</b>	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.
<b>CONTROL_PACKET_TYPE</b>	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.
<b>SOURCE_ID</b>	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.

<b>DESTINATION_ID</b>	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
<b>TRANSMITTER_ID</b>	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.
<b>RECEIVER_ID</b>	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.
<b>APP_LAYER_ARRIVAL_TIME (μs)</b>	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
<b>TRX_LAYER_ARRIVAL_TIME (μs)</b>	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
<b>NW_LAYER_ARRIVAL_TIME (μs)</b>	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.
<b>MAC_LAYER_ARRIVAL_TIME (μs)</b>	Specifies the time at which packet reaches MAC_Layer of Transmitter_ID
<b>PHY_LAYER_ARRIVAL_TIME (μs)</b>	Specifies the time at which packet reaches PHY_layer of Transmitter_ID
<b>PHY_LAYER_START_TIME (μs)</b>	Specifies the time at which packet starts betting transmitted in the link between Transmitter_ID and Receiver_ID
<b>PHY_LAYER_END_TIME (μs)</b>	Specifies the time at which packet reaches Phy_Layer of Receiver_ID
<b>APP_LAYER_PAYLOAD (Bytes)</b>	Specifies the size of the Payload at Application Layer
<b>TRX_LAYER_PAYLOAD (Bytes)</b>	Specifies the size of the Payload at Transport Layer
<b>NW_LAYER_PAYLOAD (Bytes)</b>	Specifies the size of the Payload at Network Layer
<b>MAC_LAYER_PAYLOAD (Bytes)</b>	Specifies the size of the Payload at Data Link Layer
<b>PHY_LAYER_PAYLOAD (Bytes)</b>	Specifies the size of the Payload at Physical Layer
<b>PHY_LAYER_OVERHEAD (Bytes)</b>	Specifies the size of the overhead in Physical layer
<b>PACKET_STATUS</b>	Specifies whether the Packet is Successful, Collided or Errored
<b>LOCAL_ADDRESS</b>	Specifies the Port Number at Source Node. Port Numbers are chosen randomly by NetSim.
<b>FOREIGN_ADDRESS</b>	Specifies the Port Number at Destination Node. Port Numbers are chosen randomly by NetSim.
<b>CWND (bytes)</b>	Specifies the current size of the TCP congestion window
<b>SEQ_NO</b>	If TCP is enabled, it specifies the TCP Sequence number of the packet
<b>ACK_NO</b>	If TCP is enabled, it specifies the TCP Acknowledgement number of the packet

<b>RTT (seconds)</b>	Specifies the Round-Trip Time for the packet
<b>RTO (seconds)</b>	Specifies the Retransmission Timeouts
<b>CONNECTION_STATE</b>	Specifies the state of TCP connection
<b>isSyn</b>	If TCP is enabled, it specifies whether the packet is TCP_SYN or not
<b>isAck</b>	If TCP is enabled, it specifies whether the packet is TCP_ACK/TCP_SYN ACK or not
<b>isFin</b>	If TCP is enabled, it specifies whether the packet is TCP_FIN or not
<b>SEGMENT_LENGTH</b>	Specifies the segment length of the packet
<b>SOURCE_IP</b>	Specifies the IP address of the source
<b>DESTINATION_IP</b>	Specifies the IP address of the destination
<b>GATEWAY_IP</b>	Specifies the IP address of the device which is transmitting a packet
<b>NEXT_HOP_IP</b>	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)

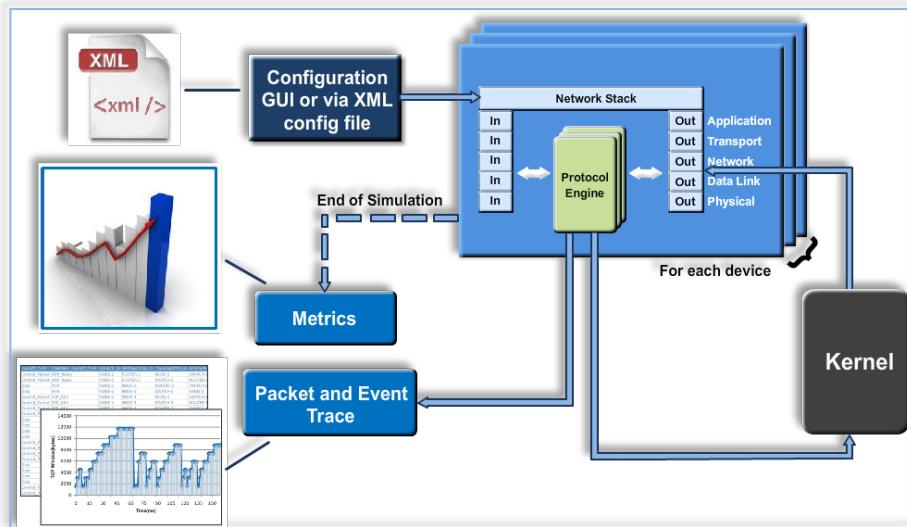
### 8.5.1 NetSim Network Stack and Discrete Event Simulation working

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, COAP 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 8-65:** 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

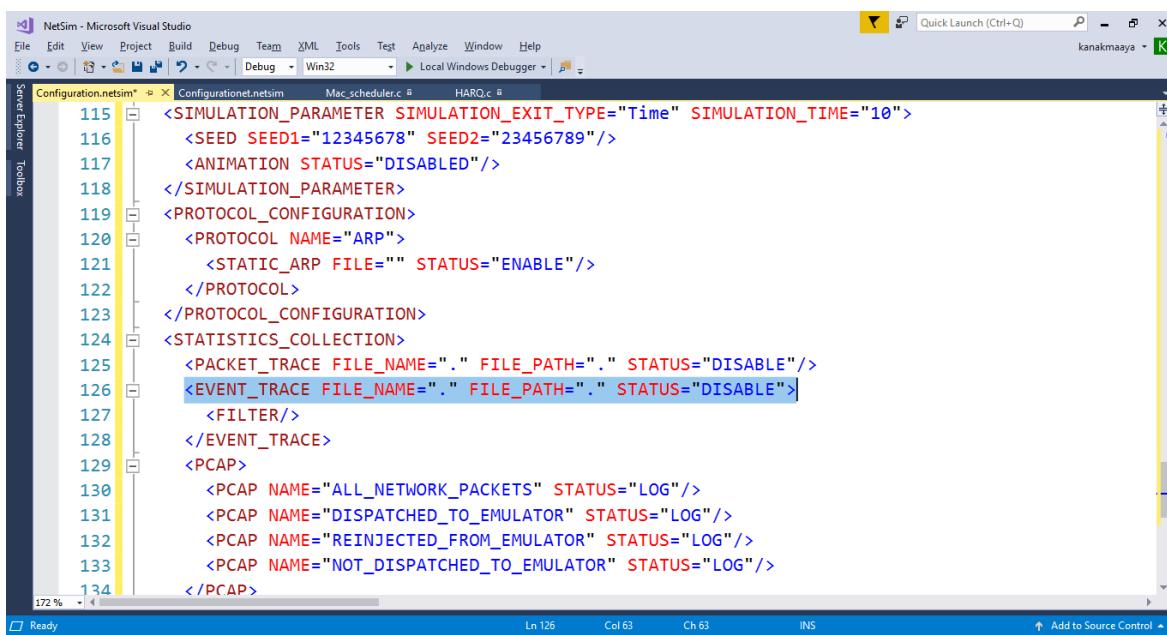
NetSim provides users with the option of turning on "Event Traces".

### How to enable Event Trace via GUI?

If NetSim runs via GUI, event trace can be turned on by clicking the Event Trace icon in the tool bar and selecting the required fields in the 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:



```

<SIMULATION_PARAMETER SIMULATION_EXIT_TYPE="Time" SIMULATION_TIME="10">
    <SEED SEED1="12345678" SEED2="23456789"/>
    <ANIMATION STATUS="DISABLED"/>
</SIMULATION_PARAMETER>
<PROTOCOL_CONFIGURATION>
    <PROTOCOL NAME="ARP">
        <STATIC_ARP FILE="" STATUS="ENABLE"/>
    </PROTOCOL>
</PROTOCOL_CONFIGURATION>
<STATISTICS_COLLECTION>
    <PACKET_TRACE FILE_NAME="." FILE_PATH="." STATUS="DISABLE"/>
    <EVENT_TRACE FILE_NAME="." FILE_PATH="." STATUS="DISABLE">
        <FILTER/>
    </EVENT_TRACE>
    <PCAP>
        <PCAP NAME="ALL_NETWORK_PACKETS" STATUS="LOG"/>
        <PCAP NAME="DISPATCHED_TO_EMULATOR" STATUS="LOG"/>
        <PCAP NAME="REINJECTED_FROM_EMULATOR" STATUS="LOG"/>
        <PCAP NAME="NOT_DISPATCHED_TO_EMULATOR" STATUS="LOG"/>
    </PCAP>
</STATISTICS_COLLECTION>

```

**Figure 8-66:** 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:

```

<ENVIRONMENT_LENGTH>500</ENVIRONMENT_LENGTH>
</GUI_INFORMATION>
<NETWORK_CONFIGURATION>...</NETWORK_CONFIGURATION>
<SIMULATION_PARAMETER SIMULATION_EXIT_TYPE="Time" SIMULATION_TIME="10">
<SEED SEED1="12345678" SEED2="23456789"/>
<ANIMATION_STATUS="OFFLINE"/>
</SIMULATION_PARAMETER>
<PROTOCOL_CONFIGURATION>
<PROTOCOL NAME="ARP">
<STATIC_ARP FILE="" STATUS="ENABLE"/>
</PROTOCOL>
</PROTOCOL_CONFIGURATION>
<STATISTICS_COLLECTION>
<PACKET_TRACE FILE_NAME="" FILE_PATH="" STATUS="DISABLE"/>
<EVENT_TRACE FILE_NAME="Event_Trace.csv" FILE_PATH="C:\Users\TETCOS~1\AppData\Local\Temp\NetSim/" STATUS="ENABLE">
<FILTER EXCLUDE_SUBEVENT=""/>
</EVENT_TRACE>
<PCAP>
<PCAP NAME="ALL_NETWORK_PACKETS" STATUS="LOG"/>
<PCAP NAME="DISPATCHED_TO_EMULATOR" STATUS="LOG"/>
<PCAP NAME="REINJECTED_FROM_EMULATOR" STATUS="LOG"/>
<PCAP NAME="NOT_DISPATCHED_TO_EMULATOR" STATUS="LOG"/>
</PCAP>
</STATISTICS_COLLECTION>
</TETCOS_NETSIM>

```

Figure 8-67: 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.

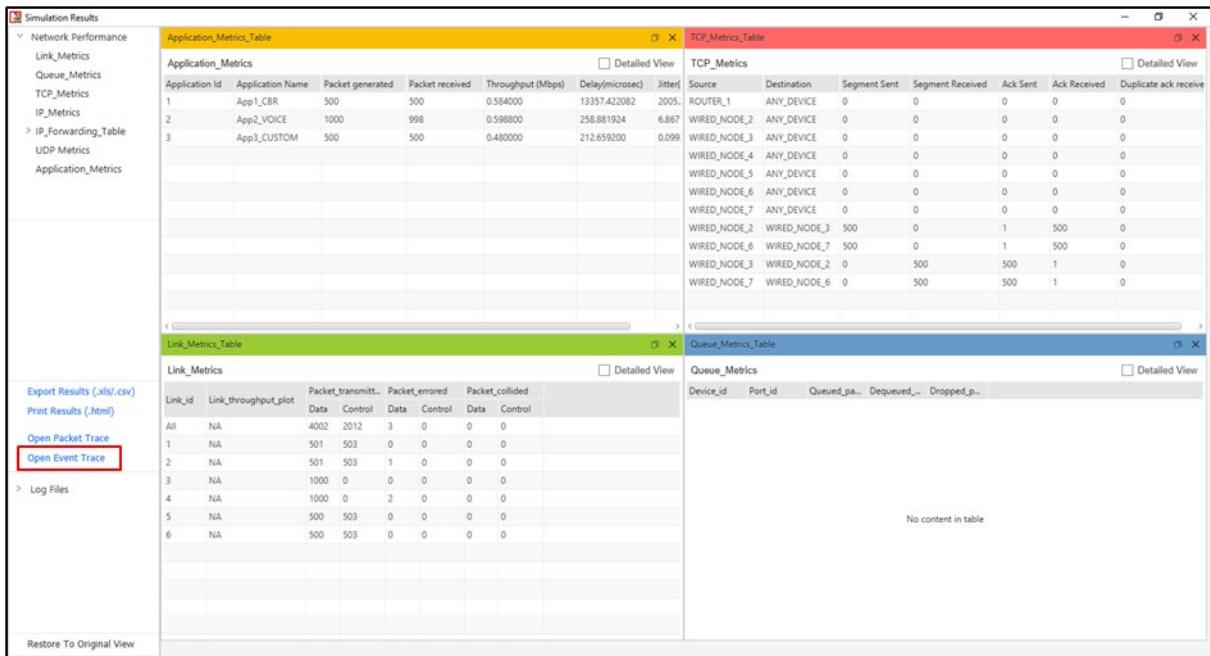
Table 8-4: Event Trace fields and Descriptions

### 8.5.3 Calculation of Delay and Application throughput from event trace

1. Enable Event trace and run simulation.

2. Click Event Trace option (Open Event Trace) in the Simulation results window as shown in below **Figure 8-68**.

**Note: Event tracing is available only in NetSim standard and pro versions.**



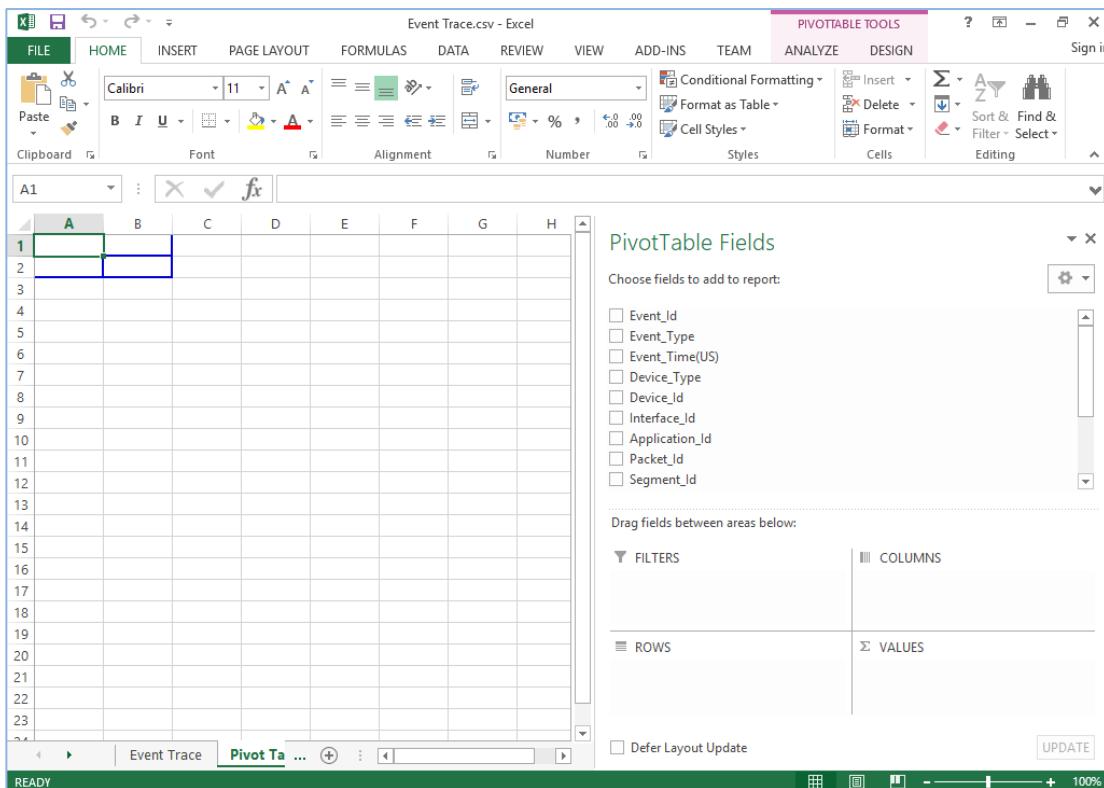
**Figure 8-68:** Select Event Trace option in results window

3. Click on **Pivot Table (Custom)** in excel sheet as shown below **Figure 8-69**.

A	B	C	D	E	F	G	H	I	J	
1	Event_Id	Event_Type	Event_Time(US)	Device_Type	Device_Id	Interface_Id	Application_Id	Packet_Id	Segment_Id	Protocol_Name
2	1	TIMER_EVENT	0	NODE	1	0	0	0	0	0 IPV4
3	2	TIMER_EVENT	0	NODE	2	0	0	0	0	0 IPV4
4	3	TIMER_EVENT	0	ROUTER	3	0	0	0	0	0 IPV4
5	4	TIMER_EVENT	0	NODE	1	1	0	0	0	0 ETHERNET
6	5	TIMER_EVENT	0	NODE	2	1	0	0	0	0 ETHERNET
7	6	TIMER_EVENT	0	ROUTER	3	1	0	0	0	0 ETHERNET
8	7	TIMER_EVENT	0	ROUTER	3	2	0	0	0	0 ETHERNET
9	10	TIMER_EVENT	0	NONE	0	0	0	0	0	0
10	12	TIMER_EVENT	0	NONE	0	0	0	0	0	0
11	8	TIMER_EVENT	20000	NODE	1	0	1	1	0	0 APPLICATION
12	13	APPLICATION_	20000	NODE	1	0	1	1	0	0 APPLICATION
13	14	TRANSPORT_O	20000	NODE	1	0	1	1	0	0 TCP
14	16	NETWORK_OU	20000	NODE	1	0	1	TCP_SYN	0	0 IPV4
15	18	MAC_OUT	20000	NODE	1	1	1	0	0	0 ETHERNET
16	19	PHYSICAL_OUT	20000	NODE	1	1	0	TCP_SYN	0	0 ETHERNET
17	20	PHYSICAL_IN	20010.6	ROUTER	3	1	0	TCP_SYN	0	0 ETHERNET
18	21	MAC_IN	20010.6	ROUTER	3	1	0	TCP_SYN	0	0 ETHERNET
19	22	NETWORK_IN	20010.6	ROUTER	3	1	0	TCP_SYN	0	0 IPV4
20	23	NETWORK_OU	20010.6	ROUTER	3	1	0	TCP_SYN	0	0 IPV4
21	24	MAC_OUT	20010.6	ROUTER	3	2	0	0	0	0 ETHERNET
22	25	PHYSICAL_OUT	20010.6	ROUTER	3	2	0	TCP_SYN	0	0 ETHERNET
23	26	PHYSICAL_IN	20021.2	NODE	2	1	0	TCP_SYN	0	0 ETHERNET
24	27	MAC_IN	20021.2	NODE	2	1	0	TCP_SYN	0	0 ETHERNET

**Figure 8-69:** Select Pivot Table (Custom) in excel sheet

4. A blank **PivotTable** and **Field List** will appear on a new worksheet.

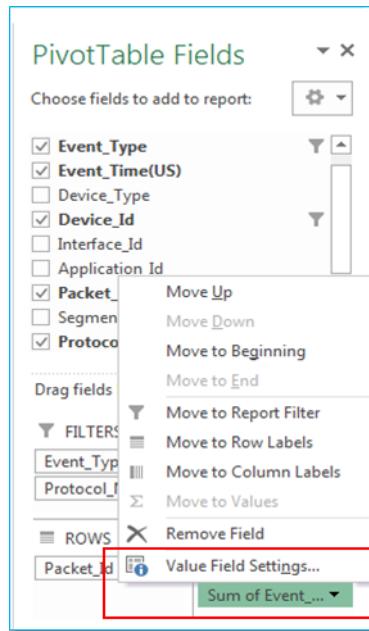


**Figure 8-70:** A blank PivotTable

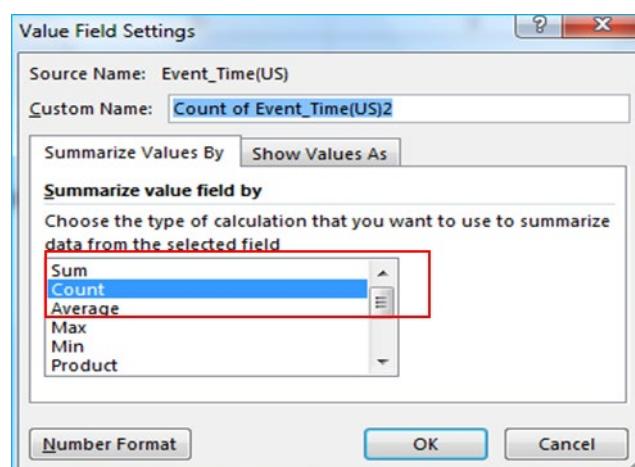
5. Once PivotTable worksheet open, you'll need to decide which **fields** to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

#### 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.

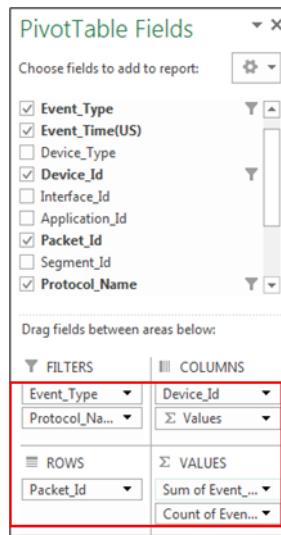


**Figure 8-71:** Select Second Event\_Time field in the VALUES and select the Value Field Settings



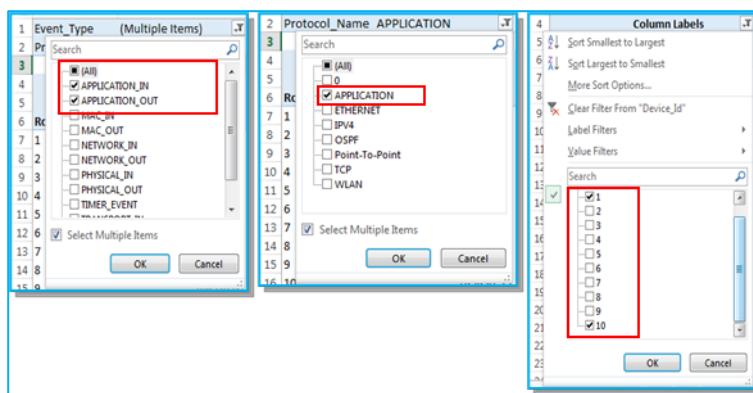
**Figure 8-72:** Select Summarize value field by Count

4. A window named **Value Field Settings** opens then select **Count** option and click **OK** button.
5. Then finally the **Pivot Table Fields** will be as shown below **Figure 8-73**.



**Figure 8-73:** 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 1 and destination node ID is 10



**Figure 8-74:** Select the Event type, Protocol Name, Source and Destination ID etc

And the Pivot Table created will be as shown (1 in the table is Source\_Id and 10 is the Destination\_Id)

	A	B	C	D	E	F	G
1	Event_Type (Multiple Items)						
2	Protocol_Name APPLICATION						
3	Column Labels						
4		1	10	Total Sum of Event_Time(US)	Total Count of Event_Time(US)		
5	Row Labels	Sum of Event_Time(US)	Count of Event_Time(US)	Sum of Event_Time(US)	Count of Event_Time(US)		
6	7	14214.2	1	20131.61	1	34345.81	2
7	8	45929.33	1	47623.12	1	93552.45	2
8	9	46533.61	1	101806.12	2	148339.73	3
9	10	53439.2	1	54268.2	1	107707.4	2
10	11	71833.65	1	148814.1	2	220647.75	3
11	12	98023.98	1	201909.71	2	299933.69	3
12	13	117998.19	1	119732	1	237730.19	2
13	14	147252.89	1	149120.09	1	296372.98	2
14	15	168926.04	1	169551.32	1	338477.36	2
15	16	189830.39	1	625730.39	3	815560.78	4
16	17	201194.89	1	218454.78	1	419649.67	2
17	18	211695.57	1	220951.59	1	432647.16	2
18	19	228650.4	1	233716.49	1	462366.89	2
19	20	240971.44	1	242438.29	1	483409.73	2
20	21	242131.64	1	492160.46	2	734292.1	3
21	22	245172.13	1	501599.16	2	746771.29	3
22	23	246558.69	1	253141.81	1	499700.5	2
23	24	287227.06	1	287927.95	1	575155.01	2
24	25	347035.96	1	349175.68	1	696211.64	2
25	26	353399.49	1	354962.14	1	708361.63	2
26	27	371663.83	1	373707.23	1	745371.06	2
27	28	375928.12	1	757038.03	2	1132966.15	3
28	29	435861.81	1	437172.94	1	873034.75	2
29	30	501551.41	1	1510551.78	2	2021113.10	4

**Figure 8-75:** Created Pivot Table

7. 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.

SUM	D	E	F	G	H	I
					=IF(AND(LEN(A1), INT(A1)=A1),F1-G1*B1,"")	
1					#VALUE!	
2						
3						
4						
5						
6	10			Total Sum of Event_Time(US)	Total Count of Event_Time(US)	
7	Sum of Event_Time(US)	Count of Event_Time(US)				
8	20131.61	1		34345.81	2	5917.41
9	47623.12	1		93552.45	2	1693.79
10	101806.12	2		148339.73	3	8738.9
11	54268.2	1		107707.4	2	829
12	148814.1	2		220647.75	3	5146.8
13	201909.71	2		299933.69	3	5861.75
14	119732	1		237730.19	2	1733.81
15	149120.09	1		296372.98	2	1867.2
16	169551.32	1		338477.36	2	625.28
17	625730.39	3		815560.78	4	56239.22
18	218454.78	1		419649.67	2	17259.89
19	220951.59	1		432647.16	2	9256.02
20	233716.49	1		462366.89	2	5066.09
21	242438.29	1		483409.73	2	1466.85
22	492160.46	2		734292.1	3	7897.18
23	501599.16	2		746771.29	3	11254.9
	253141.81	1		499700.5	2	6583.12
						700.00

**Figure 8-76:** 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**

$$\text{App Delay} = \frac{\text{Sum of the Delays of the successfully received application data packets by the destination}}{\text{Total number of successful application data packets received by the destination}}$$

**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 different packet.

Then in an empty cell enter

=SUMIF(H:H,>0)/GETPIVOTDATA("Count of Event\_Time(US)2",\$A\$4,"Device\_Id",10)

where

**GETPIVOTDATA ("Count of Event\_Time(US)2",\$A\$4,"Device\_Id",10)** gives the total number of packets received by the destination (in this case 10). This will give the exact Application Delay.

	D	E	F	G	H	I	J
1							
2							
3							
4							
5	10		Total Sum of Event_Time(US)	Total Count of Event_Time(US)			
6	Sum of Event_Time(US)	Count of Event_Time(US)2				#VALUE!	
7	20131.61	1	34345.81		2	5917.41	4608.842
8	47623.12	1	93552.45		2	1693.79	
9	101806.12	2	148339.73		3	8738.9	
10	54268.2	1	107707.4		2	829	
11	148814.1	2	220647.75		3	5146.8	
12	201909.71	2	299933.69		3	5861.75	
13	119732	1	237730.19		2	1733.81	
14	149120.09	1	296372.98		2	1867.2	
15	169551.32	1	338477.36		2	625.28	
16	625730.39	3	815560.78		4	56239.22	
17	218454.78	1	419649.67		2	17259.89	
18	220951.59	1	432647.16		2	9256.02	
19	233716.49	1	462366.89		2	5066.09	
20	242438.29	1	483409.73		2	1466.85	
21	492160.46	2	734292.1		3	7897.18	
22	501599.16	2	746771.29		3	11254.9	
23	253141.81	1	499700.5		2	6583.12	
							700.00

**Figure 8-77:** Calculated Application Delay using Formula in Pivot table

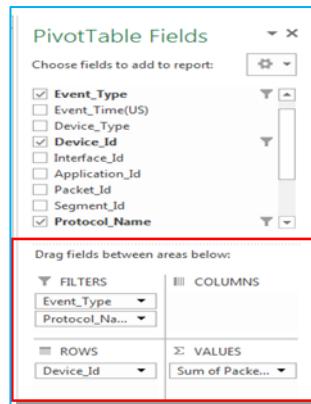
Compare with the Delay in Application\_Metrics\_Tables and it would exactly match. There might be slight difference in the decimals due to Excel's round offs.

Application_Metrics			<input type="checkbox"/> Detailed View
Received	Throughput (Mbps)	Delay(microsec)	
	0.486379	4608.2524	
	0.617545	31627.089862	
	0.051846	3478.765064	

**Figure 8-78:** Compare the Application\_Metrics\_Tables Delay and Pivot table Delay

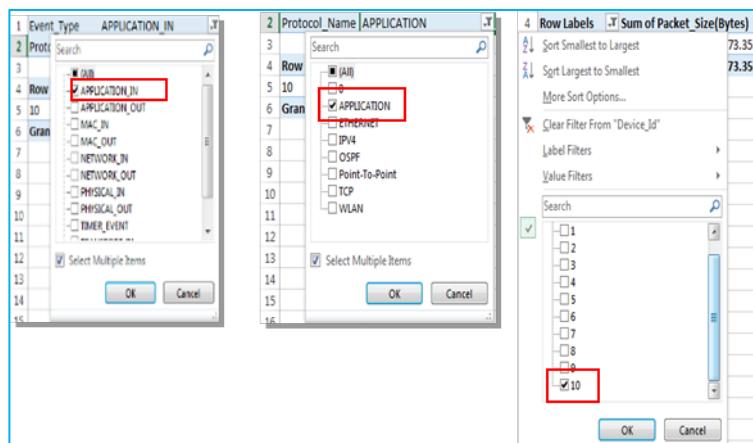
### 8.5.3.2 Application Throughput Analysis

- 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 8-79:** 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 10).

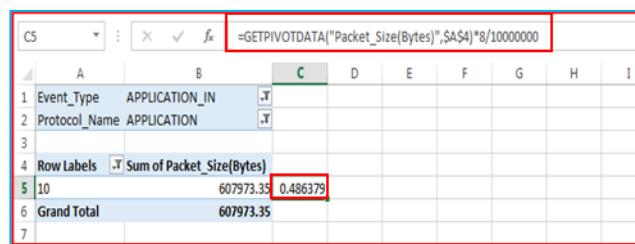


**Figure 8-80:** Select the Event type, Protocol Name, Source and Destination ID etc

$$\text{2. App Throughput} = \frac{\text{Total Payload applications successfully received by the destination}}{\text{Simulation time}}$$

Then in an empty cell type =GETPIVOTDATA ("Packet\_Size(Bytes)",\$A\$4)\*8/10000000

This give the Application Throughput in Mbps (Multiplied by 8 to convert Bytes to bits, and divided by 100000 to convert into Mega)



**Figure 8-81:** Calculate Application Throughput using formula

Compare with the Application throughput in the **Application\_Metrics\_Table**

Application_Metrics_Table		
Application_Metrics		<input type="checkbox"/> Detailed View
Received	Throughput (Mbps)	Delay(microsec)
	0.486379	4608.2524
	0.617545	31627.089862
	0.051846	3478.765064

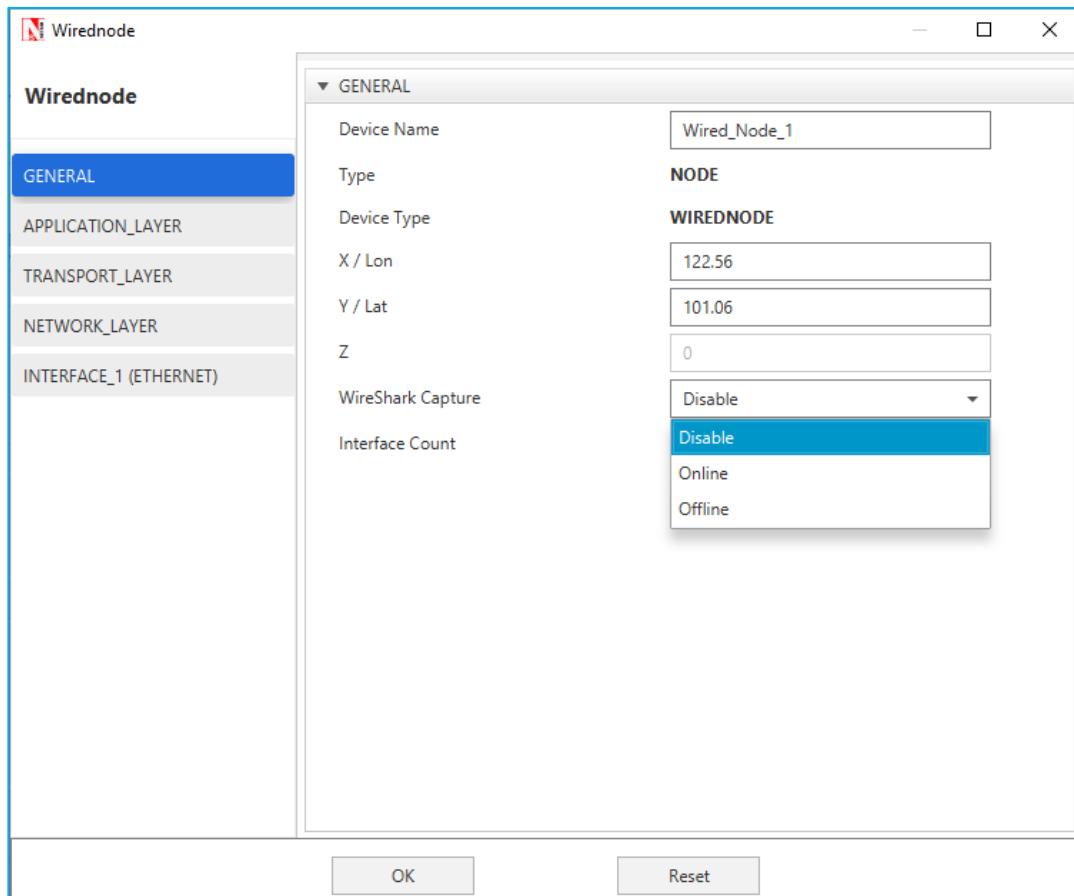
**Figure 8-82:** Compare the Application\_Metrics\_Tables throughput and Pivot table throughput

## 8.6 Packet Capture & analysis using Wireshark

### 8.6.1 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 8-83:** Enable Wireshark in General Properties for wired node

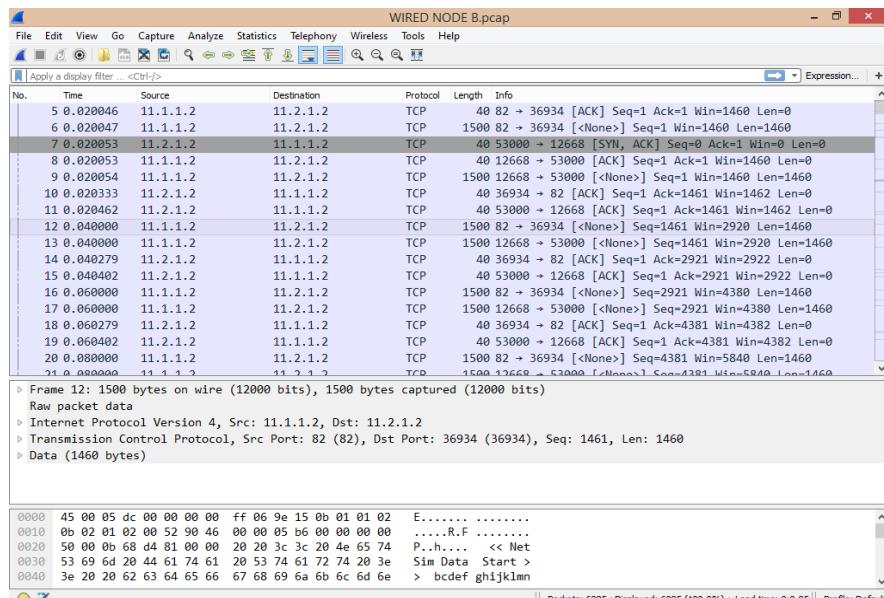
### Wireshark Capture Options

<b>Online</b>	Online option will initiate a live interactive packet capture, displaying packets while running simulation
<b>Offline</b>	Offline option will initiate silent packet capture and generate a <b>pcap</b> file which can be opened using Wireshark post-simulation
<b>Disable</b>	Packets are not captured by Wireshark during simulation.

**Table 8-5:** Wireshark Capture Options and Description

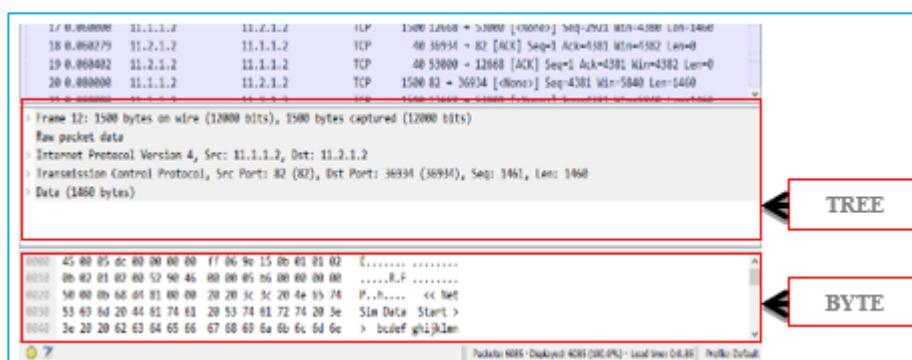
#### 8.6.2 Viewing captured packets

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 8-84**.



**Figure 8-84:** Packets Captured in Wireshark

The detail of the contents of the selected packet can be seen in the below panes as shown below **Figure 8-85**.



**Figure 8-85:** Packet Information in below 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.

### 8.6.3 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 8-86**, tcp protocol is filtered.

No.	Time	Source	Destination	Protocol	Length	Info
2	0.020000	11.1.1.2	11.2.1.2	TCP	40	82 →
3	0.020000	11.1.1.2	11.2.1.2	TCP	40	1266
4	0.020046	11.2.1.2	11.1.1.2	TCP	40	3693
5	0.020046	11.1.1.2	11.2.1.2	TCP	40	82 →
6	0.020047	11.1.1.2	11.2.1.2	TCP	1500	82 →
7	0.020053	11.2.1.2	11.1.1.2	TCP	40	5300
8	0.020053	11.1.1.2	11.2.1.2	TCP	40	1266
9	0.020054	11.1.1.2	11.2.1.2	TCP	1500	1266

**Figure 8-86:** 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.

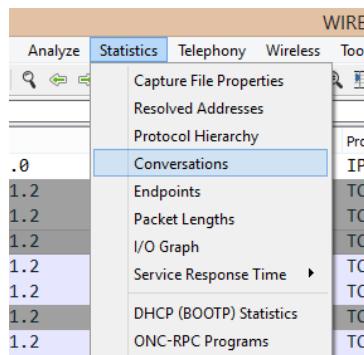
No.	Time	Source	Destination	Protocol	Length	Info
4	0.020046	11.2.1.2	11.1.1.2	TCP	40	36934 → 82 [SYN]
7	0.020053	11.2.1.2	11.1.1.2	TCP	40	53000 → 12668

**Figure 8-87:** Filtered SYN Flag and ACK Flag are set to 1 in a TCP Stream

### 8.6.4 Analyzing packets in Wireshark

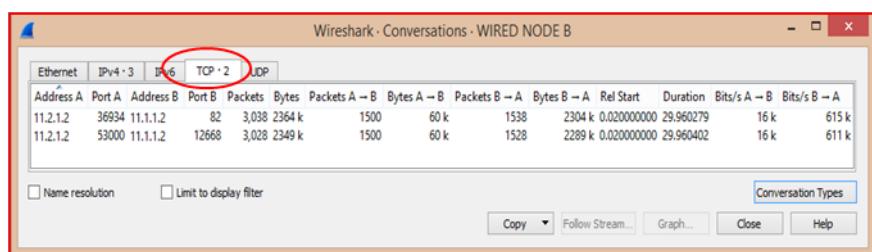
#### 8.6.4.1 Analyzing Conversation using graphs

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 8-88**.



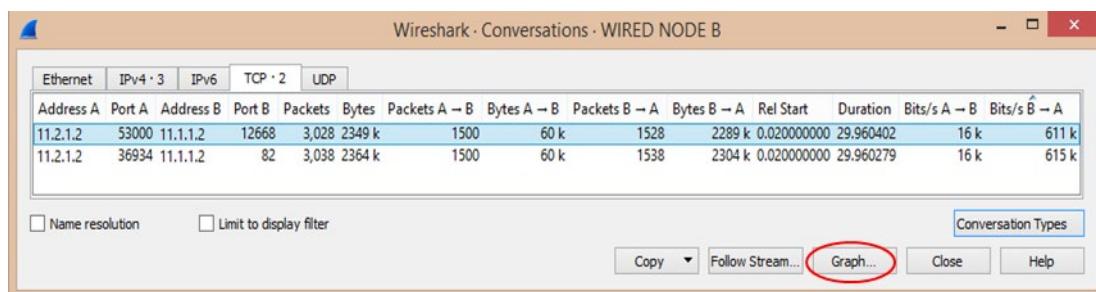
**Figure 8-88:** In Wireshark, Go to Statistics Menu > Conversations

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 8-89:** TCP Wireshark Conversion for Wired Nodes

User can also analyze each of the conversations and can create graphs by selecting them and clicking on “Graph”.



**Figure 8-90:** 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

### 8.6.5 Window Scaling

Click on data packet i.e. <None>.

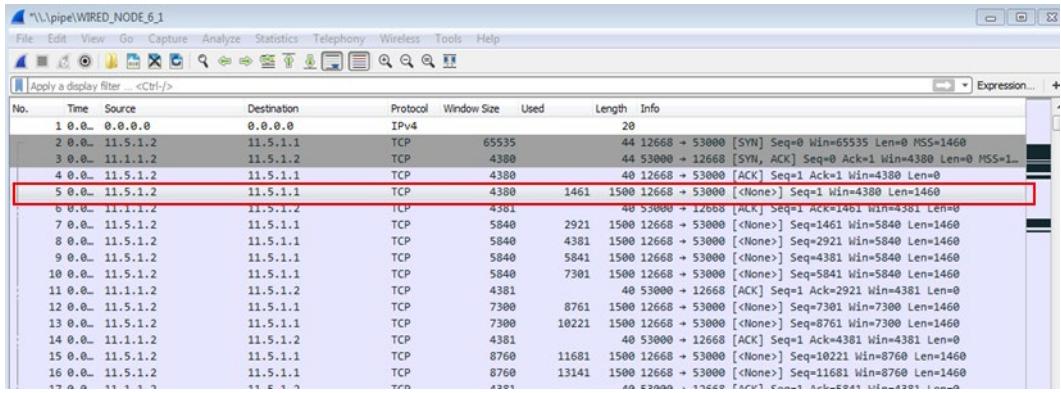


Figure 8-91: Select one data Packet &lt;None&gt;in Wireshark

Choose statistics→TCP Stream Graph→Window Scaling.

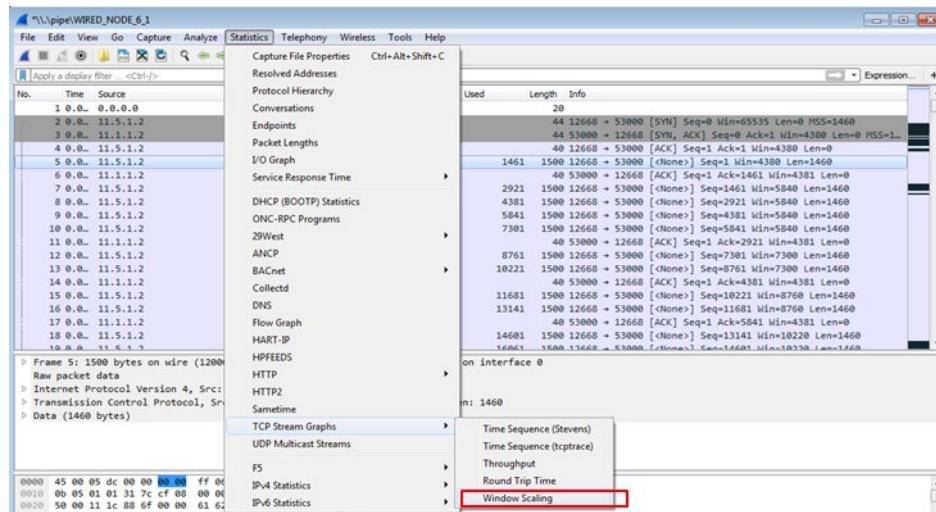


Figure 8-92: Statistics&gt;TCP Stream Graph&gt;Window Scaling

Click on Switch Direction in the window scaling graph window.

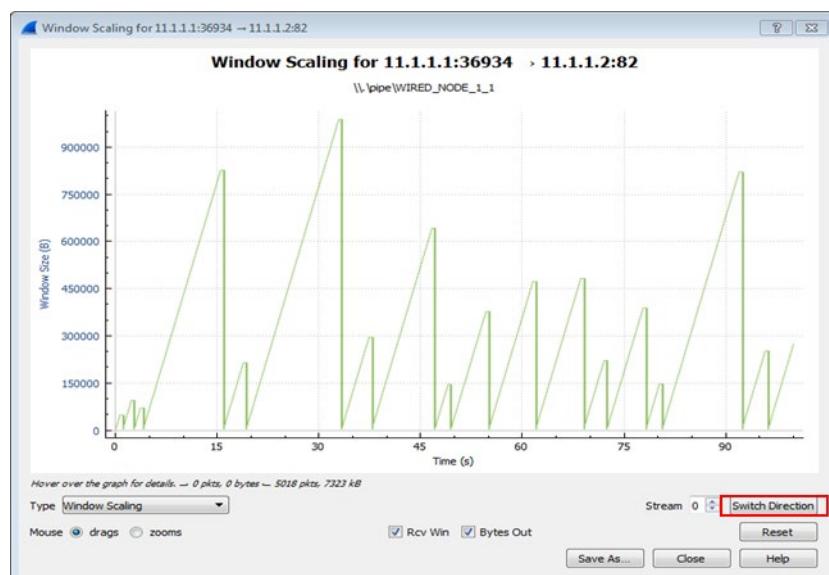
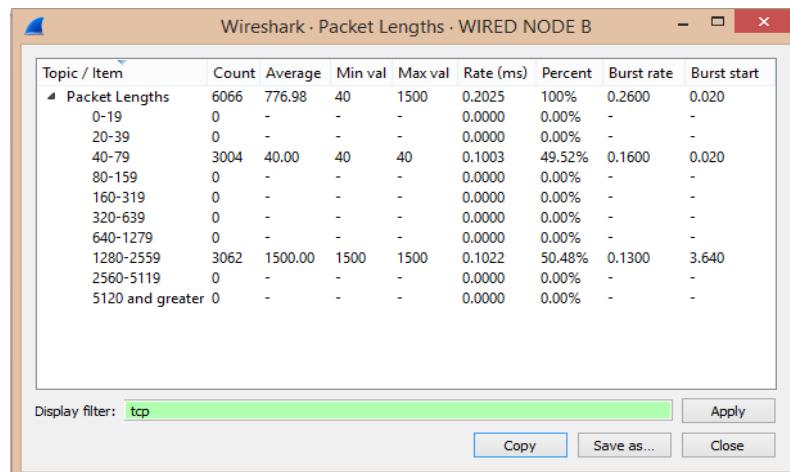


Fig 8-93: TCP congestion window Plot

### 8.6.5.1 Comparing the packet lengths

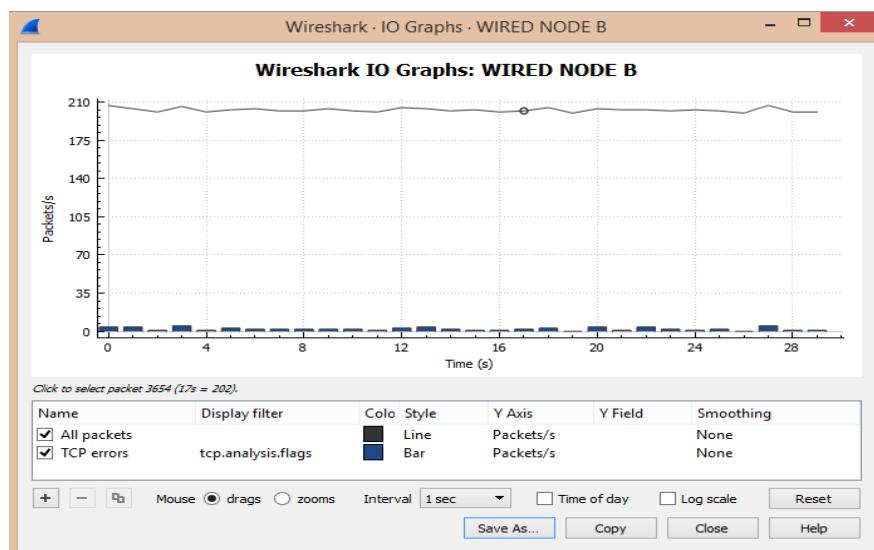
To analyze the packet sizes of all packets transmitted, go to **Statistics Menu → Packet lengths**. Users can also set filter to analyze a collection of specific packets only. For example, **tcp** filter is set to obtain the packet length below **Figure 8-94**.



**Figure 8-94:** Comparing the packet lengths in Wireshark

### 8.6.5.2 Creating IO graphs

To get the graph, go to **Statistics Menu → IO Graph**.

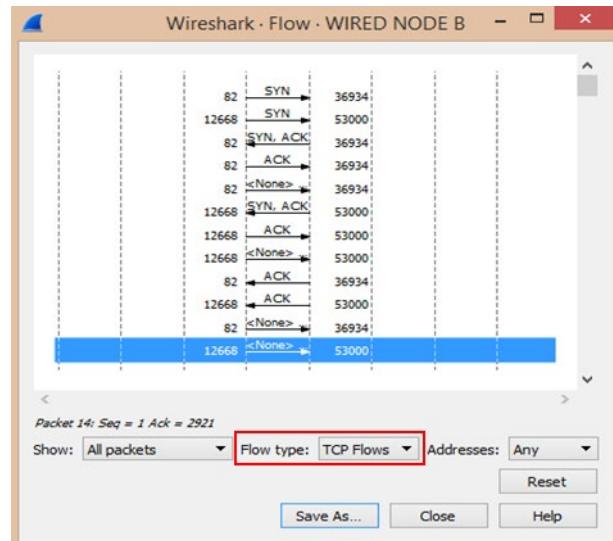


**Figure 8-95:** Statistics Menu > IO Graph in Wireshark

### 8.6.5.3 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 8-96**.



**Figure 8-96:** Statistics Menu > Flow Graph in Wireshark

# 9 Writing Custom Code in NetSim

## 9.1 Writing your own code

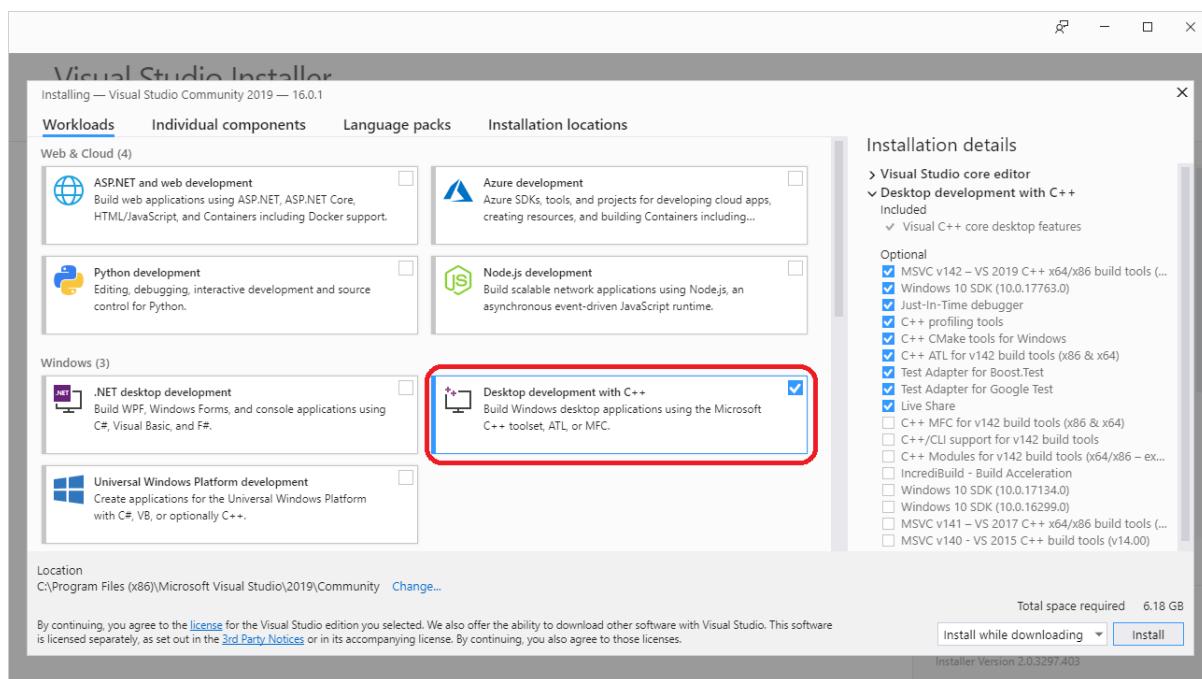
NetSim allows the user to write the custom code for all the protocols by creating a DLL (Dynamic Link Library) for their custom code.

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 2019 Installation Settings

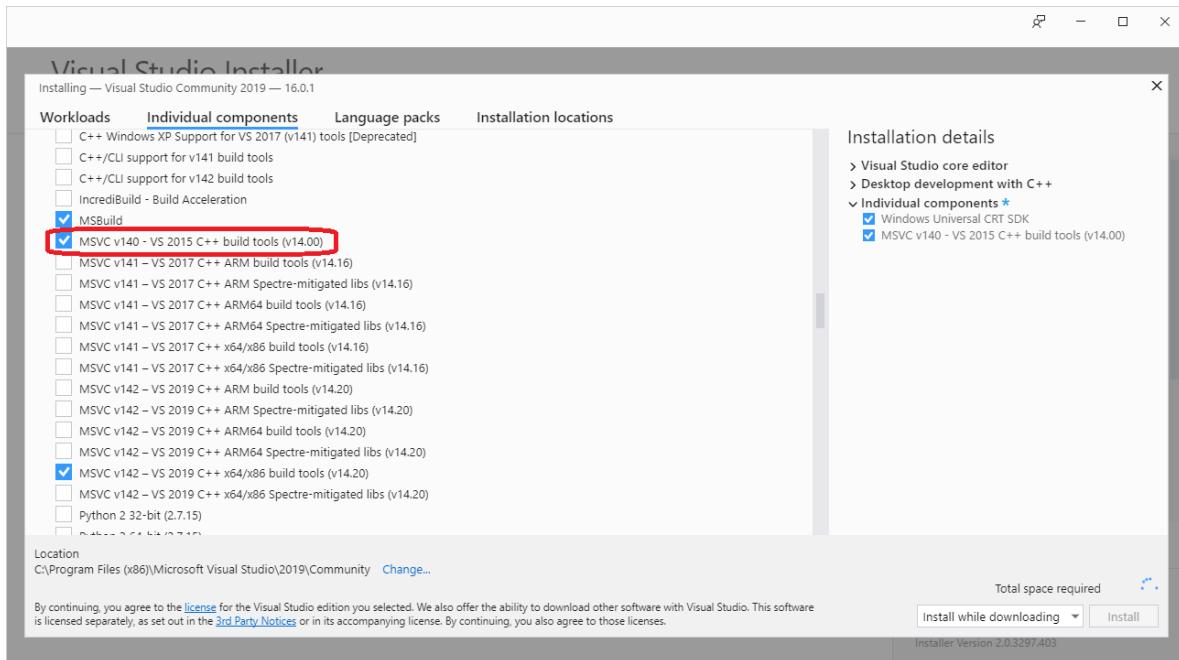
NetSim requires only a few components of Visual Studio Community 2019 edition to be installed. Upon starting the installer:

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 **VC++2015.3 V140 toolset for desktop (x86,x64)**.



**Figure 9-2:** In Individual components tab select VC++2015.3 V140 toolset for desktop (x86,x64).

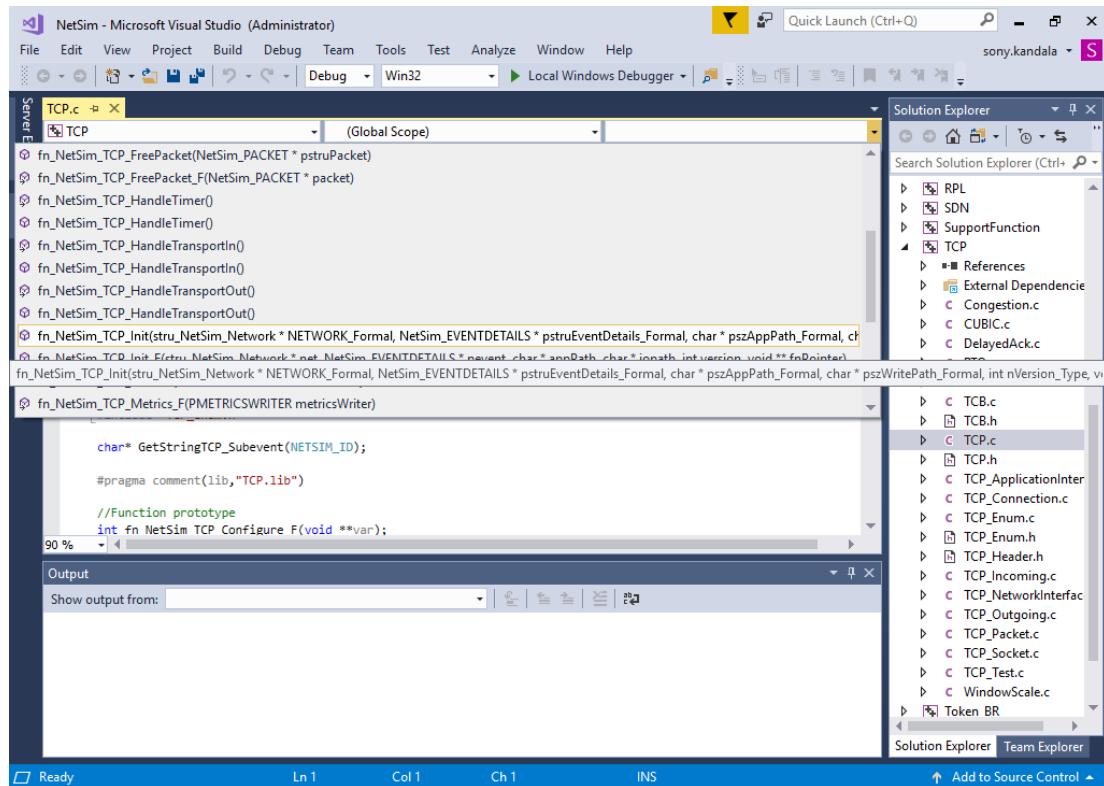
### 9.1.2 Modifying code

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. It is not possible to directly execute a DLL, since it requires an EXE for the operating system to load it through an entry point. NetSim requires Visual Studio Compiler for building DLL's.

**Note:** Make sure that Visual Studio 2015 or above is installed in your system.

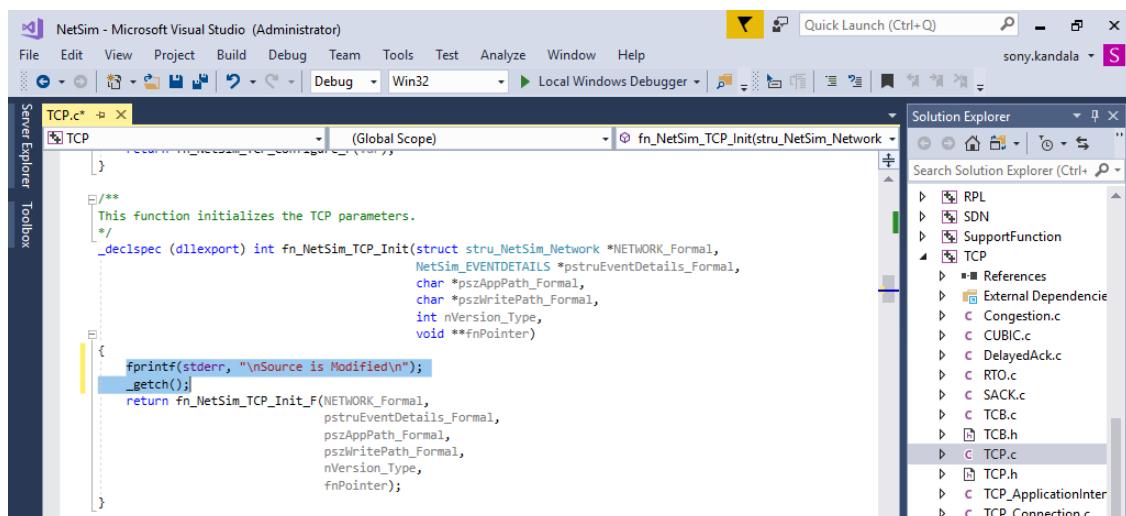
Refer **section 4.12** section “How does a user open and modify source codes” to open NetSim Source Codes

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()**.



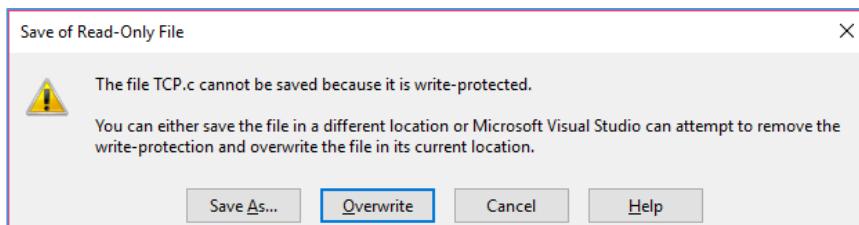
**Figure 9-3:** Select fn\_Netsim\_TCP\_Init() in TCP.C in Visual Studio

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

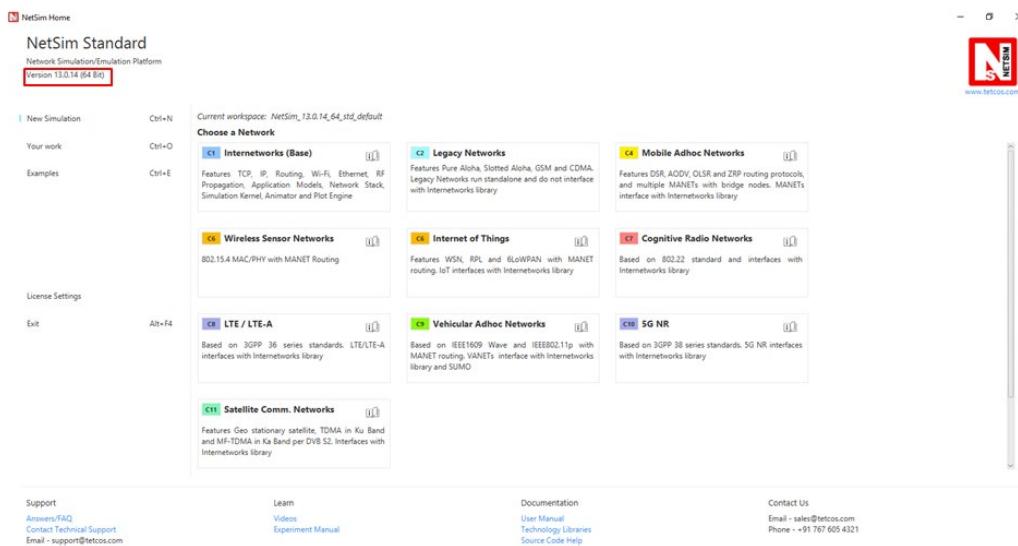
3. Once this is done click to save the changes and overwrite the file (in case of write protection).



**Figure 9-5:** Select Overwrite in save of Read Only File

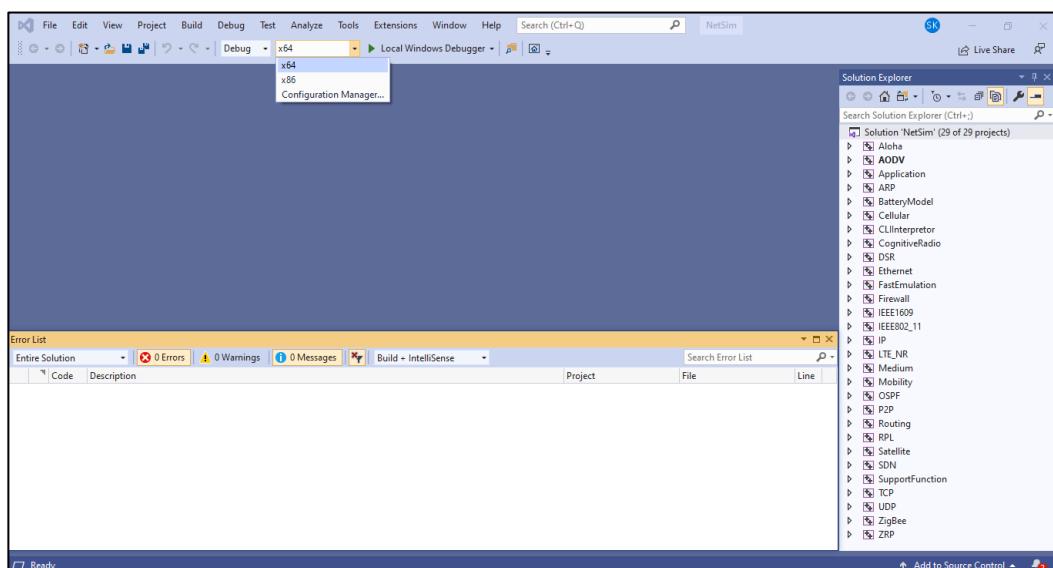
### 9.1.3 Building DLLs

1. Identify the build of NetSim that is installed in your system from NetSim Home Screen as shown below **Figure 9-6**.



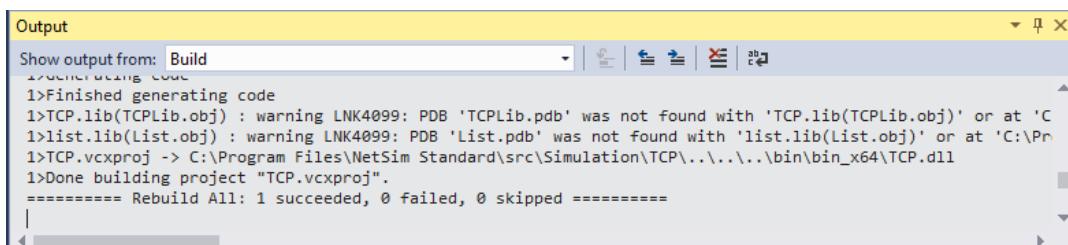
**Figure 9-6:** In NetSim Home Screen Identify the build of NetSim

2. Based on the build of NetSim installed modify the solution platform in Visual studio from the drop down as shown below **Figure 9-7**.



**Figure 9-7:** Based on the build of NetSim select Win32/x64 in Visual studio

3. Choose x64 for 64 bit version of NetSim and Win32 for 32 bit version of NetSim. After changing the solution platform, changes will be automatically applied to all projects that are displayed in the Solution Explorer.
4. 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\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x86> for 32-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 9-8**.



The screenshot shows the 'Output' window from Visual Studio. The status bar at the bottom says 'Rebuild All: 1 succeeded, 0 failed, 0 skipped'. The main area displays the build log:

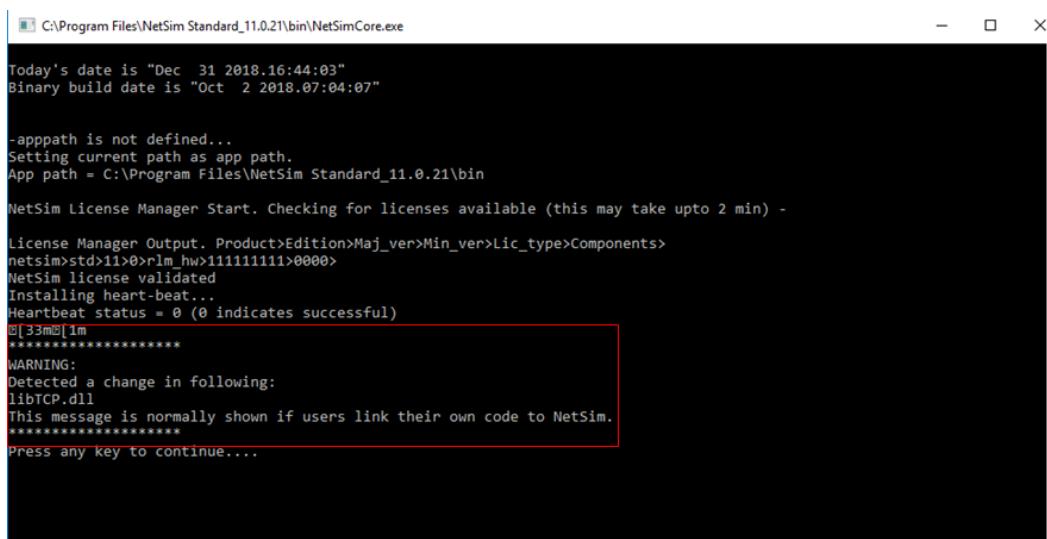
```

Output
Show output from: Build
1>Building code
1>Finished generating code
1>TCP.lib(TCPLib.obj) : warning LNK4099: PDB 'TCPLib.pdb' was not found with 'TCP.lib(TCPLib.obj)' or at 'C:\Program Files\NetSim Standard\src\Simulation\TCP\..\..\bin\bin_x64\TCP.dll'
1>list.lib(List.obj) : warning LNK4099: PDB 'List.pdb' was not found with 'list.lib(List.obj)' or at 'C:\Program Files\NetSim Standard\src\Simulation\list\..\..\bin\bin_x64\list.dll'
1>TCP.vcxproj -> C:\Program Files\NetSim Standard\src\Simulation\TCP\..\..\bin\bin_x64\TCP.dll
1>Done building project "TCP.vcxproj".
===== Rebuild All: 1 succeeded, 0 failed, 0 skipped =====
|
```

**Figure 9-8:** Build is successful a message similar to the following will be displayed in the output window

#### 9.1.4 Running Simulation

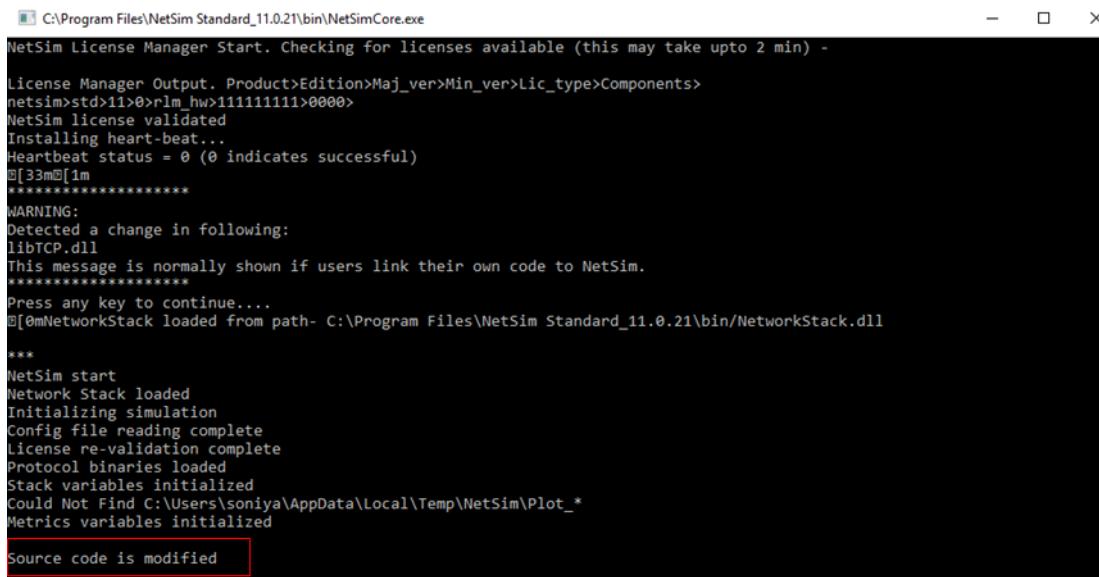
1. After rebuilding the code, user can run the simulation via GUI (Please refer **section 3**). In this case, user can create a scenario in any network which involves TCP protocol. Running the simulation with the custom DLL will initially display a warning message as shown below **Figure 9-9**.



**Figure 9-9:** 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\_x86 for 32-bit and 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 9-10**.



```
C:\Program Files\NetSim Standard_11.0.21\bin\NetSimCore.exe
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>std>i1>0>rlm_hw>1111111111>0000>
NetSim license validated
Installing heart-beat...
Heartbeat status = 0 (0 indicates successful)
@[33m@1m
*****
WARNING:
Detected a change in following:
libTCP.dll
This message is normally shown if users link their own code to NetSim.
*****
Press any key to continue....
@[0mNetworkStack loaded from path- C:\Program Files\NetSim Standard_11.0.21\bin\NetworkStack.dll
***
NetSim start
Network Stack loaded
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\soniya\AppData\Local\Temp\NetSim\Plot_*
Metrics variables initialized
Source code is modified
```

**Figure 9-10:** 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\_x86 for 32-bit and 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
<b>Application</b>	IP
<b>Cellular</b>	Application
<b>CLlInterpertor</b>	Firewall, IP
<b>Cognitive Radio</b>	Application
<b>Ethernet</b>	Firewall
<b>IEEE802_11</b>	Battery Model
<b>OSPF</b>	IP
<b>Routing</b>	IP
<b>RPL</b>	IP
<b>ZigBee</b>	Battery Model
<b>ZRP</b>	IP
<b>Aloha</b>	-
<b>AODV</b>	-

<b>ARP`</b>	-
<b>Battery Model</b>	-
<b>CSMACD</b>	-
<b>DSR</b>	-
<b>Firewall</b>	-
<b>IEEE1609</b>	-
<b>IP</b>	-
<b>LTE NR</b>	
<b>Mobility</b>	-
<b>P2P</b>	-
<b>SDN</b>	-
<b>Support Function</b>	-
<b>TCP</b>	-
<b>Token_BR</b>	-
<b>UDP</b>	-
<b>UWAN</b>	
<b>TDMA</b>	-
<b>TDMA</b>	-
<b>Satellite Comm. Networks</b>	-

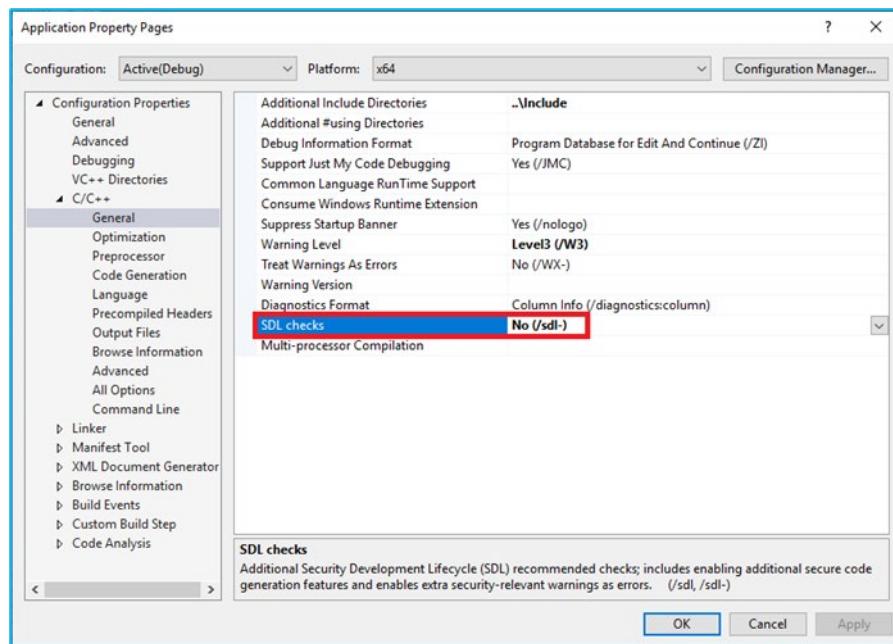
**Table 9-1:** Source Code Dependencies

For E.g.: 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 9-11:** Enabling Additional Security Checks application properties window

If SDL checks is enabled (/sdl), following warnings will be treated as errors:

Warning enabled by	Equivalent command-line switch	Description
<b>C4146</b>	/we4146	A unary minus operator was applied to an unsigned type, resulting in an unsigned result.
<b>C4308</b>	/we4308	A negative integral constant converted to unsigned type, resulting in a possibly meaningless result.
<b>C4532</b>	/we4532	Use of continue, break or goto keywords in a __finally/finally block has undefined behavior during abnormal termination.
<b>C4533</b>	/we4533	Code initializing a variable will not be executed.
<b>C4700</b>	/we4700	Use of an uninitialized local variable.
<b>C4703</b>	/we4703	Use of a potentially uninitialized local pointer variable.
<b>C4789</b>	/we4789	Buffer overrun when specific C run-time (CRT) functions are used.
<b>C4995</b>	/we4995	Use of a function marked with pragma deprecated.
<b>C4996</b>	/we4996	Use of a function marked as deprecated.

**Table 9-2:** 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

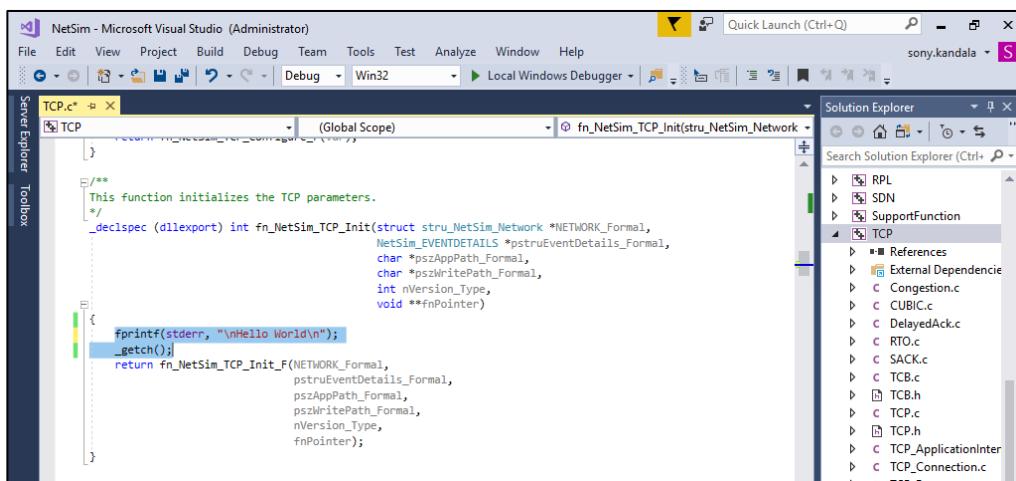
### 9.2.1 Hello World Program

**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, "\nHello World\n");
```

```
_getch();
```



**Figure 9-12:** Hello World Printf Statement added in TCP Project

Build DLL as explained in **Section 9.1.3** and run the simulation, you can see the following output on the console.

```
C:\Program Files\NetSim Standard_11.0.21\bin\NetSimCore.exe
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>std>1>0>lm_hw>111111111>00000>
NetSim license validated
Installing heart-beat...
Heartbeat status = 0 (0 indicates successful)
@[33mb[1m
*****
WARNING:
Detected a change in following:
libTCP.dll
This message is normally shown if users link their own code to NetSim.
*****
Press any key to continue...
@[0mNetworkStack loaded from path- C:\Program Files\NetSim Standard_11.0.21\bin\NetworkStack.dll

***
NetSim start
Network Stack loaded
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\soniya\AppData\Local\Temp\NetSim\Plot_*
Metrics variables initialized
Hello World
```

**Figure 9-13:** 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\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-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 from 10<sup>th</sup> sec onwards and fail Node Id 1 from 90<sup>th</sup> sec onwards, the NodeFailure.txt file will be as follows **Figure 9-14**.

File	Edit	Format	View	Help
Node Id	Failure Time			
1	90000000			
2	10000000			

**Figure 9-14:** 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 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 Animation

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.

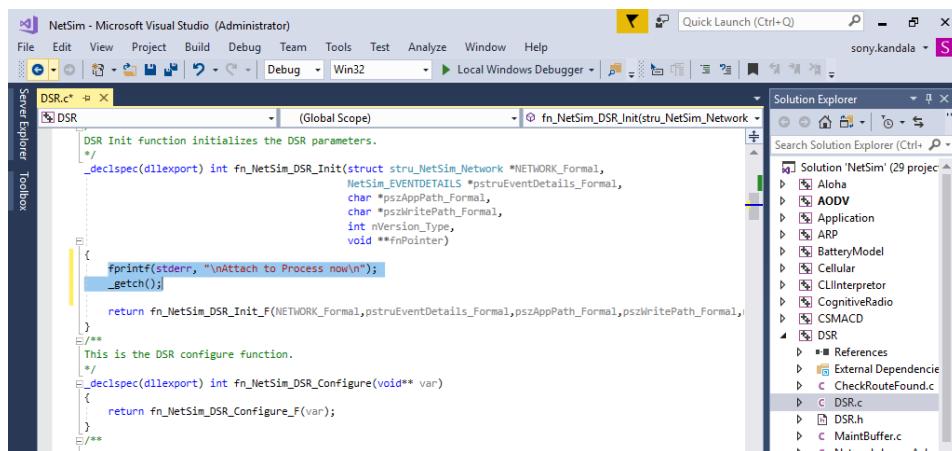
- **Using \_getch()**
- **Using Environment Variables (NETSIM\_BREAK)**

#### 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 DSR protocol and add the following lines of code in the init function as shown in the below screenshot **Figure 9-15**.

```
fprintf(stderr, "\nAttach to Process now\n");
```

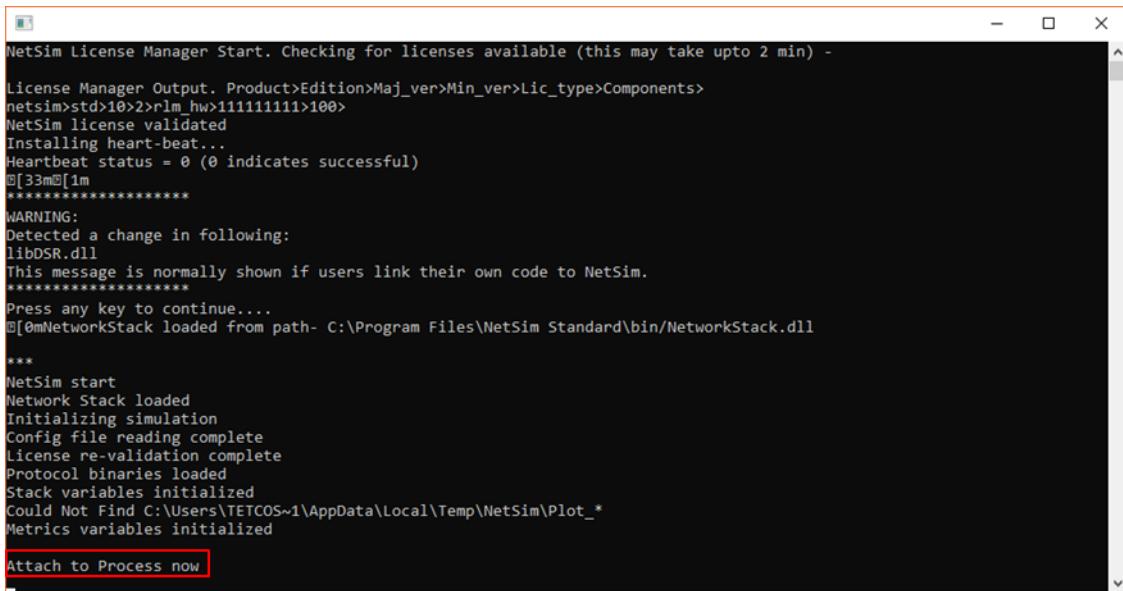
```
_getch();
```



**Figure 9-15:** Added Two line of Code in DSR protocol

**Step 2:** Build the DSR 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 9-16** and the simulation will pause for user input (because of \_getch() added in the init function)



```

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>std>10>2>rlm_hw>11111111>100>
NetSim license validated
Installing heart-beat...
Heartbeat status = 0 (0 indicates successful)
@[33m@[1m
*****
WARNING:
Detected a change in following:
libDSR.dll
This message is normally shown if users link their own code to NetSim.
*****
Press any key to continue....
@[0mNetworkStack loaded from path- C:\Program Files\NetSim Standard\bin\NetworkStack.dll

*** 
NetSim start
Network Stack loaded
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\Plot_*
Metrics variables initialized

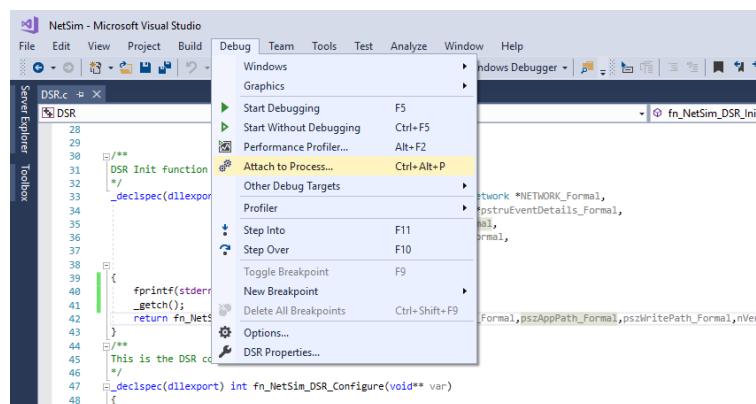
Attach to Process now

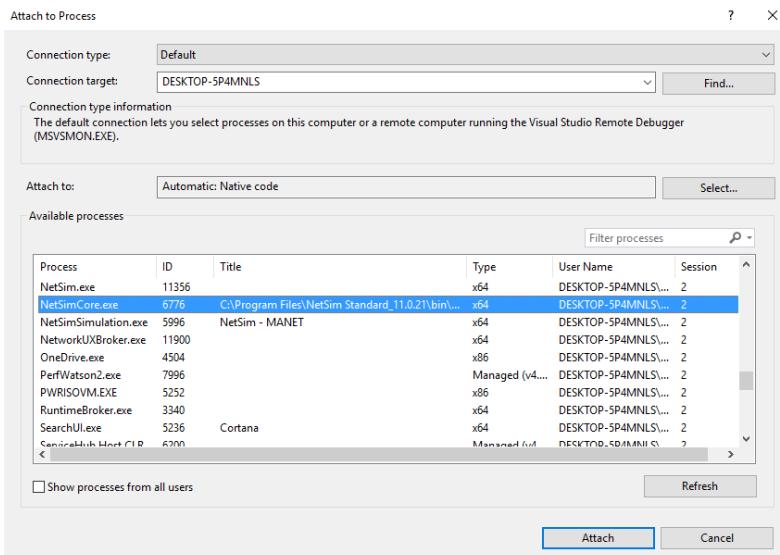
```

**Figure 9-16:** Attach to Process Statement written to console

**Step 4:** In Visual Studio, put break point inside the source code where you want to debug.

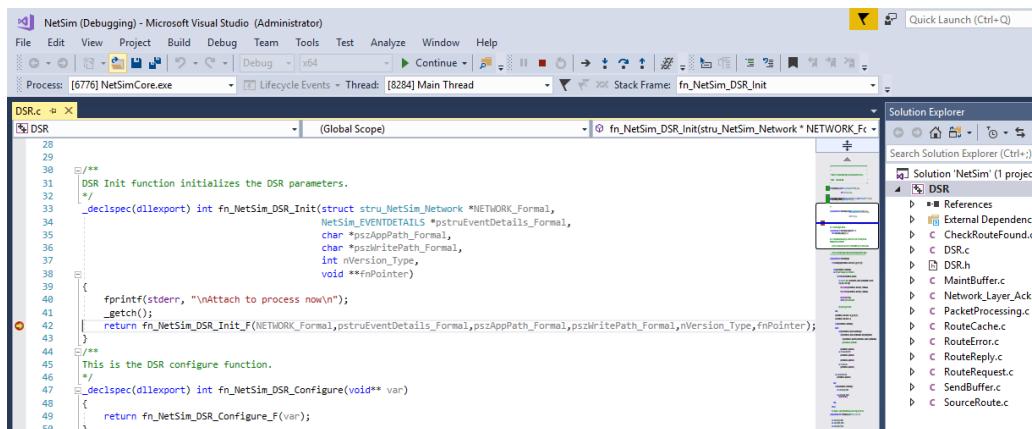
**Step 5:** Go to “Debug→Attach to Process” in Visual studio as shown and attach to NetSimCore.exe.

**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

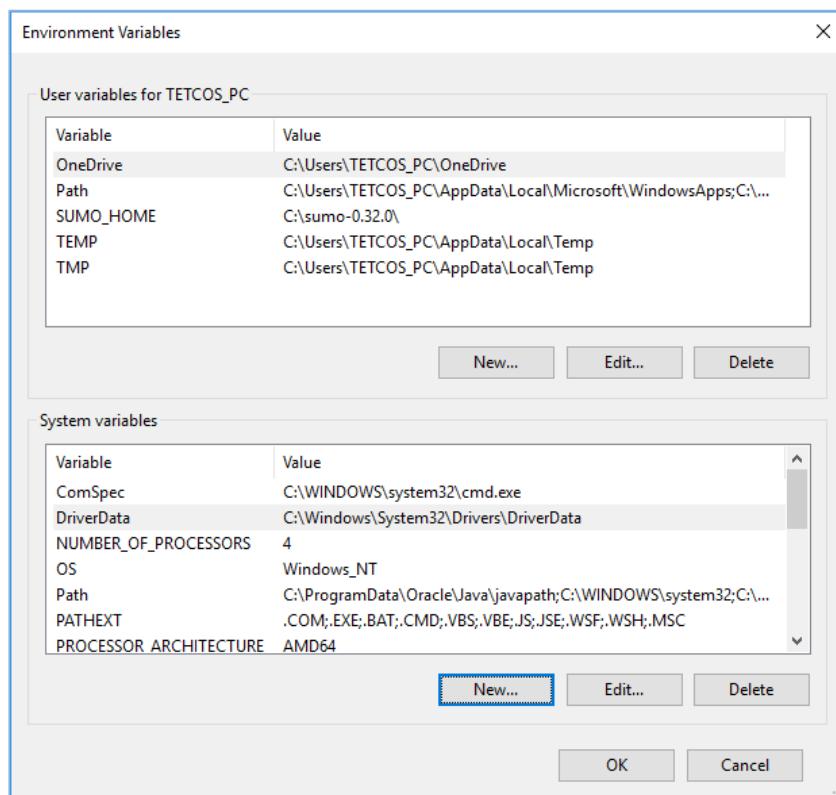
This section is helpful to Debug Using Environment Variable (NETSIM\_BREAK). To set Environment variable follow the steps as shown.

**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

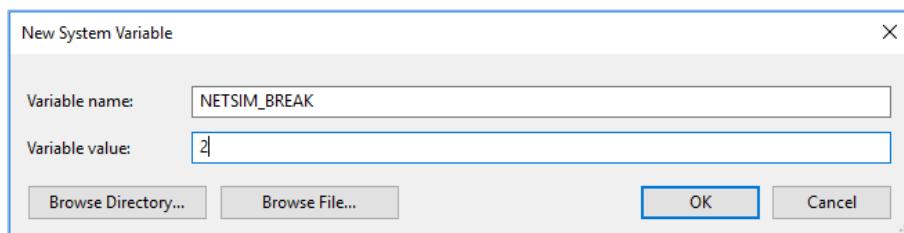
**Step 1:** Right click on My Computer\ This PC and select Properties.

**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.



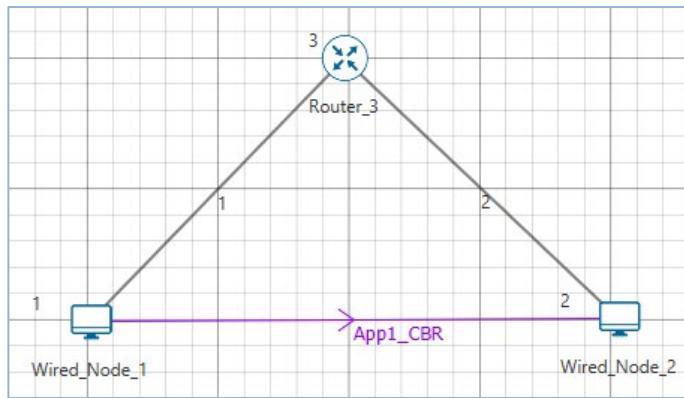
**Figure 9-20:** Environment Variable Window



**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 4.12** “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**.

```
Select C:\Program Files\NetSim Pro\bin\NetSimCore.exe
Calling IP init..
IP init done.
Calling UDP init..
UDP init done.
Calling OSPF init..
OSPF init done.
Calling Ethernet init..
Ethernet init done.
Calling ARP init..
ARP init done.
Calling Application init..
Application init done.
Protocol variables initialized
Executing command --- DEL "C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\*.pcap"
Could Not Find C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\*.pcap
Emulation is disabled
Applications created

*** 
Simulation in progress...
Press CTRL+C to terminate the simulation mid-way

 0 % is completed... Simulation Time=0.000 ms  Event Id=1
Breakpoint triggered...
Press any key to continue the process...
-
```

**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.

NetSim - Microsoft Visual Studio (Administrator)

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug x64 Local Windows Debugger

TCP.c NAS.c LTE\_Phy.c

(Global Scope)

```

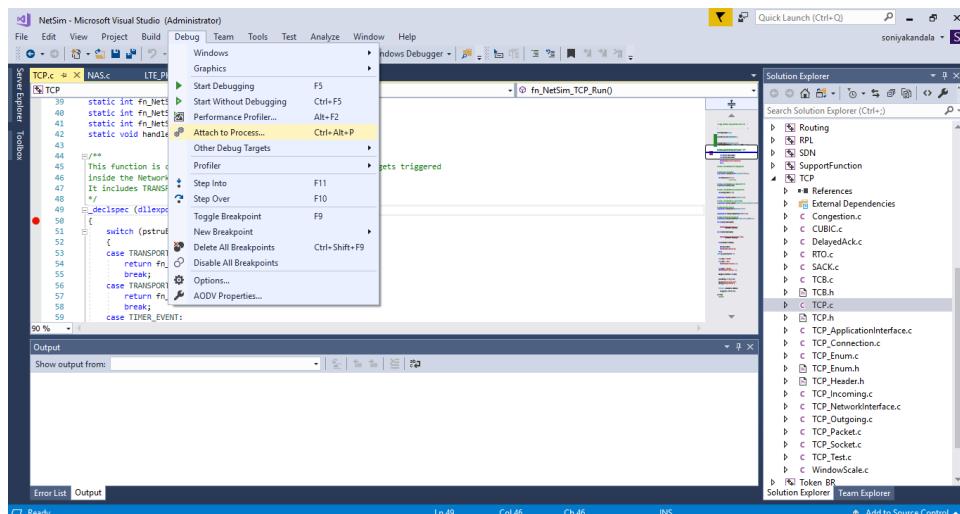
39     static int fn_NetSim_TCP_HandleTransportOut();
40     static int fn_NetSim_TCP_HandleTransportIn();
41     static int fn_NetSim_TCP_HandleTimer();
42     static void handle_time_wait_timeout();

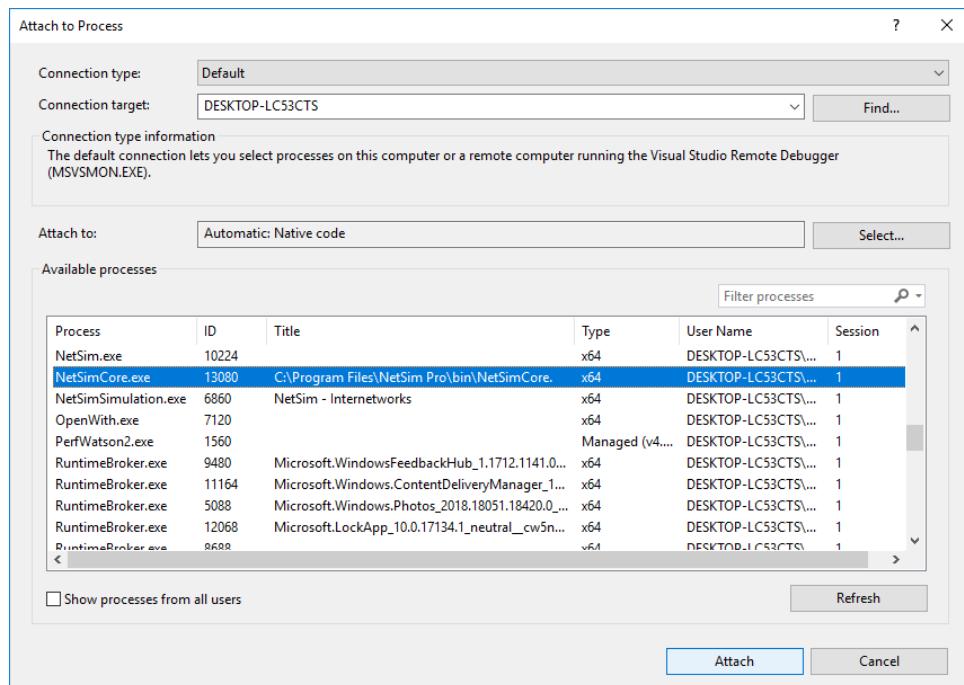
44     /**
45      This function is called by NetworkStack.dll, whenever the event gets triggered
46      inside the NetworkStack.dll for the Transport layer TCP protocol
47      It includes TRANSPORT_OUT,TRANSPORT_IN and TIMER_EVENT.
48     */
49     _declspec (dllexport) int fn_NetSim_TCP_Run()
50     {
51         switch (pstruEventDetails->nEventType)
52         {
53             case TRANSPORT_OUT_EVENT:
54                 return fn_NetSim_TCP_HandleTransportOut();
55                 break;
56             case TRANSPORT_IN_EVENT:
57                 return fn_NetSim_TCP_HandleTransportIn();
58                 break;
59             case TIMER_EVENT:

```

**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.

```
C:\Program Files\NetSim Pro\bin\NetSimCore.exe
OSPF init done.
Calling Ethernet init..
Ethernet init done.
Calling ARP init..
ARP init done.
Calling Application init..
Application init done.
Protocol variables initialized
Executing command --- DEL "C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\*.pcap"
Could Not Find C:\Users\TETCOS~1\AppData\Local\Temp\NetSim\*.pcap
Emulation is disabled
Applications created

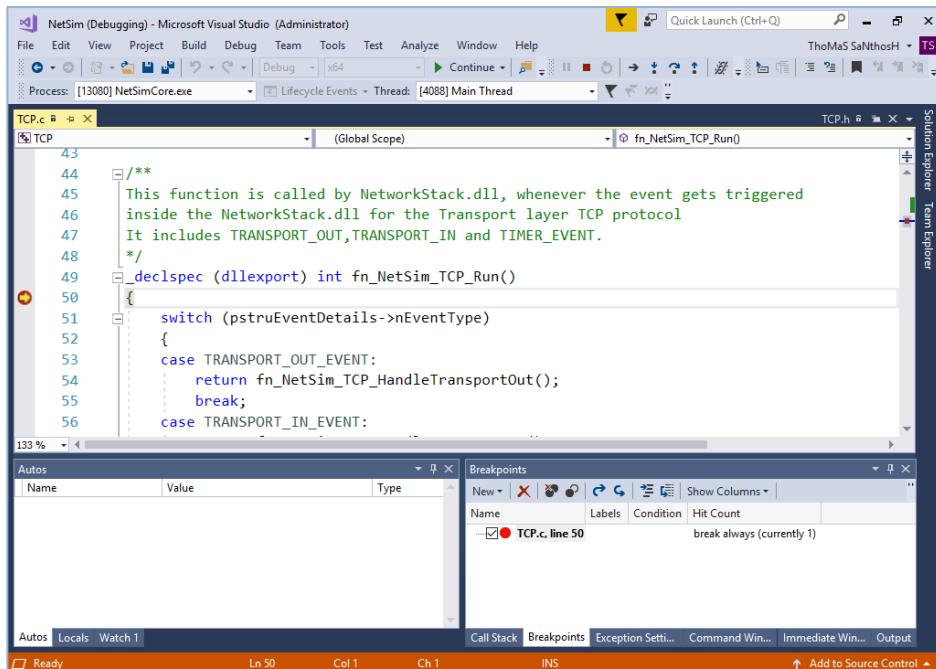
*** 
Simulation in progress...
Press CTRL+C to terminate the simulation mid-way

 0 % is completed... Simulation Time=0.000 ms Event Id=1
Breakpoint triggered...
Press any key to continue the process...

Enter the next event id to break
20
 0 % is completed... Simulation Time=20.000 ms Event Id=14
```

**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:

Packet_Id	Segment_Id	Protocol_Name	Subevent_Type	Packet_Size(Bytes)	Prev_Event_Id	Line_No	File_Name
2	0	0 IPV4	IP_INIT_TABLE	0	0	1270	IP.c
3	0	0 IPV4	IP_INIT_TABLE	0	0	1270	IP.c
4	0	0 IPV4	IP_INIT_TABLE	0	0	1270	IP.c
5	0	0 ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
6	0	0 ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
7	0	0 ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
8	0	0 ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
9	0	0	5 LINKUP	0	0	123	Link.c
10	0	0	5 LINKUP	0	0	123	Link.c
11	1	0 APPLICATION		1460	0	138	Database_FTP_Custom.c
12	1	0 APPLICATION		1460	8	241	Application.c
13	1	0 TCP		0	1460	13	101 Application.c
14	TCP_SYN	0 IPV4		0	24	14	58 TCP_NetworkInterface.c
15	0	0 ETHERNET		0	44	16	104 ReadArpTable.c
16	TCP_SYN	0 ETHERNET		0	70	18	49 Ethernet_Mac.c
17	TCP_SYN	0 ETHERNET		0	70	19	98 Ethernet_Phy.c
18	TCP_SYN	0 ETHERNET		0	70	20	192 Ethernet_Phy.c
19	TCP_SYN	0 IPV4		0	44	21	203 Ethernet_Mac.c
20	TCP_SYN	0 IPV4		0	24	22	446 IP.c
21	0	0 ETHERNET		0	44	23	104 ReadArpTable.c
22	TCP_SYN	0 ETHERNET		0	70	24	49 Ethernet_Mac.c
23	TCP_SYN	0 ETHERNET		0	70	25	98 Ethernet_Phy.c
24	TCP_SYN	0 ETHERNET		0	70	26	104 ReadArpTable.c

**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 DSR protocol and build the code. Create a scenario on MANET 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.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\TETCOS>cd "C:\Program Files\NetSim Standard\bin"

C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath"C:\Program Files\NetSim Standard\bin" -iopath "C:\Users\TETCOS\Desktop\manet\exp-2" -license 5053@192.168.0.119 -d
```

**Figure 9-30:** Run the NetSim via CLI Mode use the following Command.

Press enter, now you can see the following screen.

```
C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "C:\Users\TETCOS\Desktop\manet\exp-2" -license 5053@192.168.0.119 -d

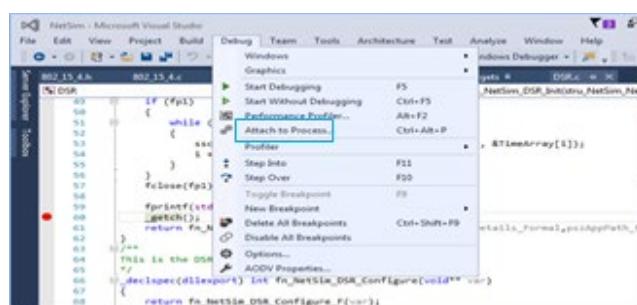
Today's date is "May 24 2017.11:54:23"
Binary build date is "May 12 2017.15:21:43"

Breakpoint is set...
Enter the event id where you want to break:
```

**Figure 9-31:** Enter the Event ID

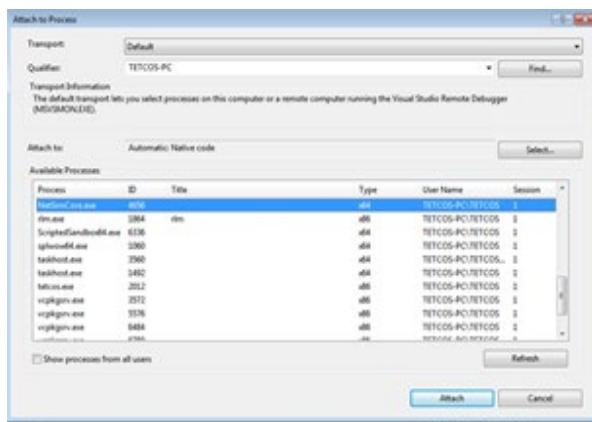
**Step 4:** Open the Project in Visual Studio and put break point inside the source code.

**Step 5:** Go to “Debug→Attach to Process”.



**Figure 9-32:** Debug > Attach to Process

Attach to NetSimCore.exe.

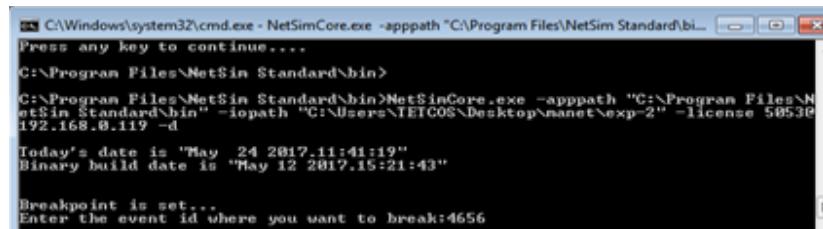


**Figure 9-33:** Select NetSimCore.exe in Attach to Process Window

Click on Attach.

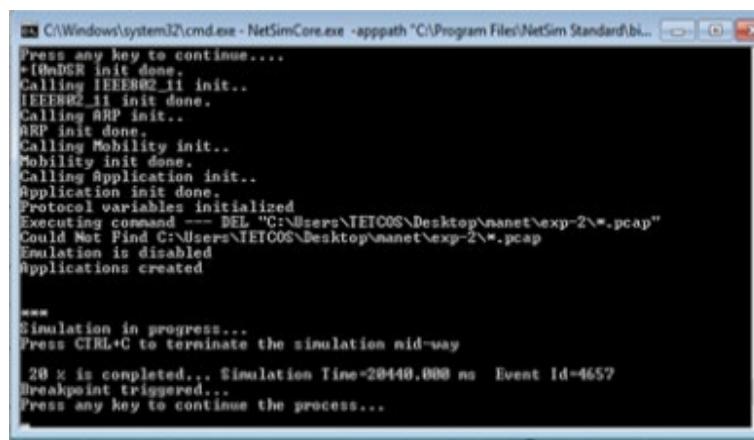
**Step 6:** Go to command prompt which is already opened in Step 3. Enter the Event Id.

**Note:** If you don't want to stop at any event you can specify 0 as event id.



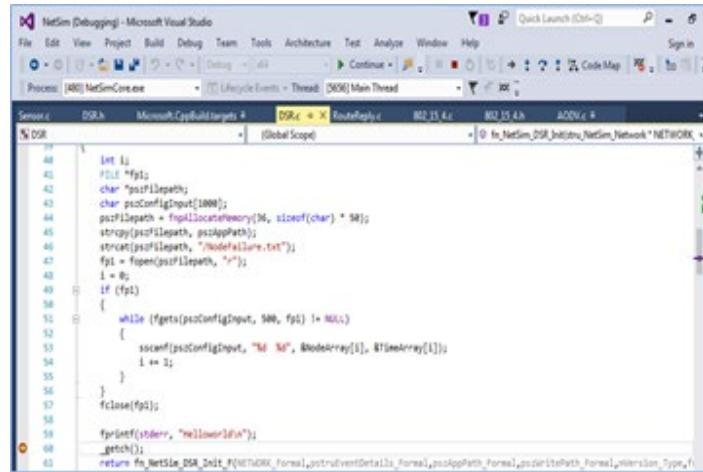
**Figure 9-34:** Enter the Event Id

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**.



**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.

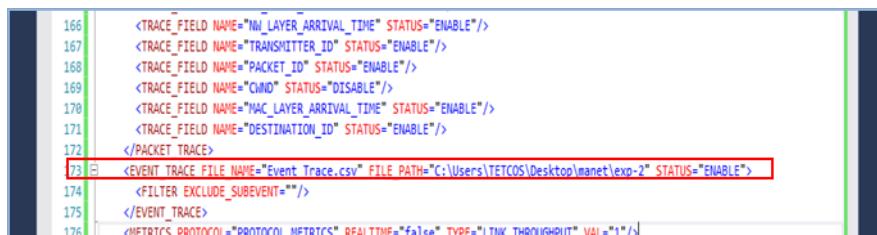
To stop debugging press Shift+F5. This then gives the control back to NetSim, for normal execution to continue.

### 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 NetSim Help in **Section 7→Running Simulation via CLI**) with –d as the fourth parameters.

Press enter.

```
C:\Windows\system32\cmd.exe - NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "C:\Users\TETCOS\Desktop\manet\exp-2" -license 50530 192.168.0.119 -d
```

**Figure 9-38:** Run the NetSim via CLI

**Step 3:** Enter **-1** as the event ID.

```
C:\Windows\system32\cmd.exe - NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "C:\Users\TETCOS\Desktop\manet\exp-2" -license 50530 192.168.0.119 -d

Today's date is "May 24 2017.12:17:28"
Binary build date is "May 12 2017.15:21:43"

Breakpoint is set...
Enter the event id where you want to break:-1
```

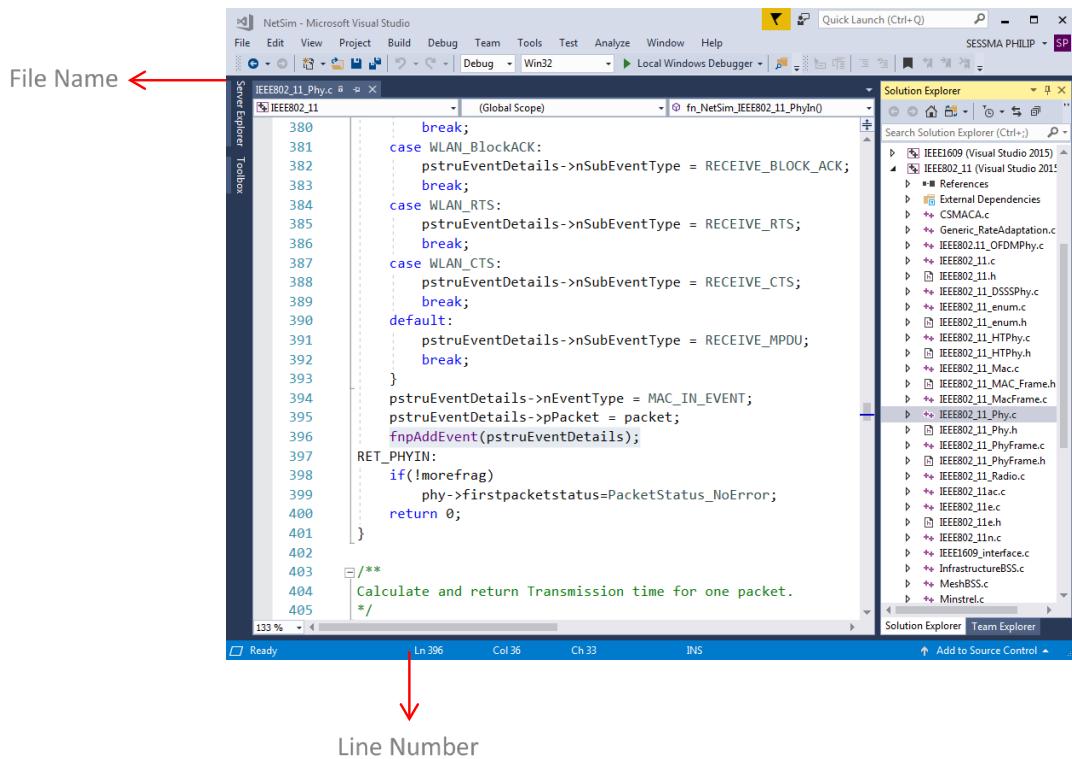
**Figure 9-39:** Enter **-1** as the event ID

Upon running, NetSim will write the file name and line number of the source code that generated each event.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
37	43	MAC_IN	5020998.01	NODE	2	1	0	DSR_RREQ	0	WLAN	RECEIVE_MPDU	72	41	396	IEEE802_11_Phy.c
38	44	NETWORK_IN	5020998.01	NODE	2	1	0	DSR_RREQ	0	IPV4		0	72	43	81 IEEE802_11_Mac.c
39	45	NETWORK_OUT	5029056.01	NODE	2	1	0	DSR_RREP	0	DSR		0	11	44	69 RouteReply.c
40	46	MAC_OUT	5029056.01	NODE	2	1	0	0	0	WLAN		0	31	45	104 ReadArpTable.c
41	47	MAC_OUT	5029056.01	NODE	2	1	0	DSR_RREP	0	WLAN	CS	31	46	101 CSMACA.c	
42	48	MAC_OUT	5029106.01	NODE	2	1	0	DSR_RREP	0	WLAN	DIFS_END	31	47	132 CSMACA.c	
43	49	MAC_OUT	5029126.01	NODE	2	1	0	DSR_RREP	0	WLAN	BACKOFF	31	48	189 CSMACA.c	
44	50	MAC_OUT	5029146.01	NODE	2	1	0	DSR_RREP	0	WLAN	BACKOFF	31	49	241 CSMACA.c	
45	51	MAC_OUT	5029166.01	NODE	2	1	0	DSR_RREP	0	WLAN	BACKOFF	31	50	241 CSMACA.c	
46	52	PHYSICAL_OUT	5029166.01	NODE	2	1	0	DSR_RREP	0	WLAN		0	71	51	290 IEEE802_11_Mac.c
47	55	TIMER_EVENT	5029410.01	NODE	2	1	0	DSR_RREP	0	WLAN	UPDATE_DEVICE	71	52	270 IEEE802_11_Phy.c	
48	53	PHYSICAL_IN	5029410.45	NODE	1	1	0	DSR_RREP	0	WLAN		0	71	52	523 IEEE802_11_Phy.c
49	56	MAC_IN	5029410.45	NODE	1	1	0	DSR_RREP	0	WLAN	RECEIVE_MPDU	71	53	396 IEEE802_11_Phy.c	
50	57	NETWORK_IN	5029410.45	NODE	1	1	0	DSR_RREP	0	IPV4		0	71	56	81 IEEE802_11_Mac.c
51	58	MAC_OUT	5029410.45	NODE	1	1	0	DSR_RREP	0	WLAN	SEND_ACK	71	56	131 IEEE802_11_Mac.c	
52	59	NETWORK_OUT	5029410.45	NODE	1	1	0	TCP_SYN	0	IPV4		0	24	57	151 SendBuffer.c
53	62	MAC_OUT	5029410.45	NODE	1	1	0	0	0	WLAN		0	52	59	104 ReadArpTable.c
54	60	PHYSICAL_OUT	5029420.45	NODE	1	1	0	WLAN_ACK	0	WLAN		0	14	58	298 CSMACA.c
55	64	TIMER_EVENT	5029724.45	NODE	1	1	0	WLAN_ACK	0	WLAN	UPDATE_DEVICE	14	60	270 IEEE802_11_Phy.c	
56	65	MAC_OUT	5029724.45	NODE	1	1	0	TCP_SYN	0	WLAN	CS	52	64	101 CSMACA.c	
57	63	PHYSICAL_IN	5029724.89	NODE	2	1	0	WLAN_ACK	0	WLAN		0	14	60	523 IEEE802_11_Phy.c
58	67	MAC_IN	5029724.89	NODE	2	1	0	WLAN_ACK	0	WLAN	RECEIVE_ACK	14	63	396 IEEE802_11_Phy.c	
59	54	TIMER_EVENT	5029727	NODE	2	1	0	0	0	WLAN	ACK_TIMEOUT	0	52	432 CSMACA.c	
60	66	MAC_OUT	5029774.45	NODE	1	1	0	TCP_SYN	0	WLAN	DIFS_END	52	65	132 CSMACA.c	

**Figure 9-40:** NetSim writes the file name and line number of the source code in Event Trace

**Note:** In the above trace file Event Id 56 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 396 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.



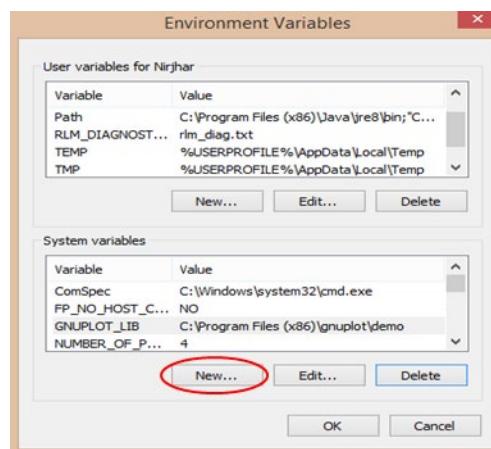
**Figure 9-41:** IEEE802\_11\_Phy.c file Code in Visual Studio

## **Procedure 2:**

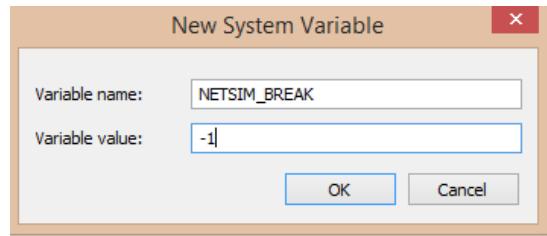
Step 1: Right click on my computer and select Properties.

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.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Event_Ty	Event_Tin	Device_Ty	Device_Id	InterApp_Packet														
14	PHYSICAL	20110	NODE	2	1	0	DSR_RREQ	0	W	0	72	16	277	IEEE802_11_Macc					
18	NETWORK	20878.01	NODE	1	1	0	DSR_RREQ	0	IP	0	72	20	81	IEEE802_11_Macc					
19	NETWORK	28936.01	NODE	1	1	0	DSR_RREQ	0	DS	0	11	21	69	c:\program files\netsim standard\src\simulation\dsr\routereply.c					
50	PHYSICAL	29526.01	NODE	1	1	0	DSR_RREQ	0	W	0	71	52	277	IEEE802_11_Macc					
54	NETWORK	30002.32	NODE	2	1	0	DSR_RREQ	0	IP	0	71	57	81	IEEE802_11_Macc					
55	MAC_OUT	30002.32	NODE	2	1	0	DSR_RREQ	0	W	SET	71	57	120	IEEE802_11_Macc					
56	NETWORK	30002.32	NODE	2	1	1		0	IP	0	1468	58	151	c:\program files\netsim standard\src\simulation\dsr\sendbuffer.c					
79	PHYSICAL	30646.32	NODE	2	1	1		0	W	0	1536	82	277	IEEE802_11_Macc					
83	NETWORK	36982.64	NODE	1	1	1		0	IP	0	1536	87	81	IEEE802_11_Macc					
84	MAC_OUT	36982.64	NODE	1	1	1		0	W	SET	1536	87	87	120 IEEE802_11_Macc					
85	TRANSPOI	36982.64	NODE	1	1	1		0	IP	0	1536	88	77	c:\program files\netsim standard\src\simulation\dsr\sourceroute.c					
111	PHYSICAL	40350	NODE	2	1	1		0	W	0	1536	117	277	IEEE802_11_Macc					
117	NETWORK	46686.31	NODE	1	1	1		0	IP	0	1536	122	81	IEEE802_11_Macc					
118	MAC_OUT	46686.31	NODE	1	1	1		0	W	SET	1536	122	120	IEEE802_11_Macc					
119	TRANSPOI	46686.31	NODE	1	1	1		0	IP	0	1536	123	77	c:\program files\netsim standard\src\simulation\dsr\sourceroute.c					
149	PHYSICAL	60390	NODE	2	1	1		0	W	0	1536	154	277	IEEE802_11_Macc					
153	NETWORK	66726.31	NODE	1	1	1		0	IP	0	1536	159	81	IEEE802_11_Macc					
154	MAC_OUT	66726.31	NODE	1	1	1		0	W	SET	1536	159	120	IEEE802_11_Macc					

**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 hover 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.

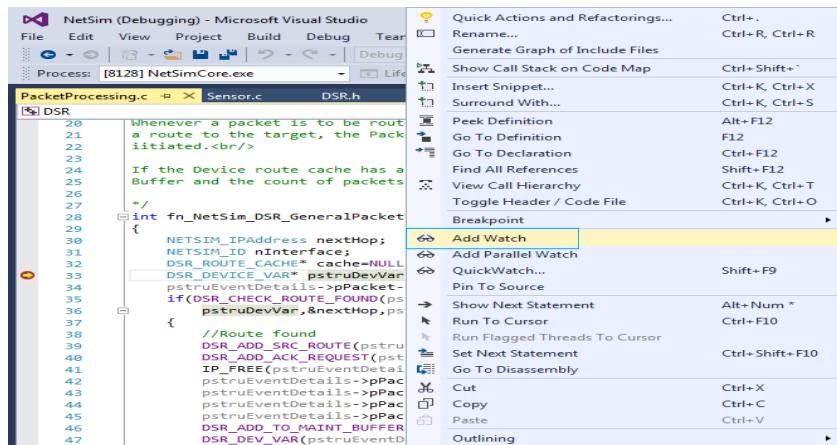
```

26
27
28 int fn_NetSim_DSR_GeneralPacketProcessing(NetSim_EVENTDETAILS* pstruEventDetails)
29 {
30     NETSIM_IPAddress nextHop;
31     NETSIM_ID nInterface;
32     DSR_ROUTE_CACHE* cache=NULL;
33     DSR_DEVICE_VAR* pstruDevVar = DSR_DEV_VAR(pstruEventDetails->nDeviceId);
34     pstruEventDetails->nReqIdentification = 0;
35     if(DSR_CHECK_ROUTE_ID)
36         pstruDevVar->next->pstruRouteCache = 0x0000000000000000<NULL>;
37     {
38         //Route found
39         DSR_ADD_SRC_ROUTE
40         DSR_ADD_ACK_REQUEST
41         IP_FREE(pstruEventDetails->pPacket->pstruNetworkData->nextHopIp = IP_COPY(nextHop));
42         pstruEventDetails->pPacket->pstruNetworkData->zGatewayIp = IP_COPY(DSR_DEV_IP(pstruEventDetails->nDeviceId));
43         pstruEventDetails->pPacket->nTransmitterId = pstruEventDetails->nDeviceId;
44         pstruEventDetails->pPacket->nReceiverId = fn_NetSim_Stack_getDeviceId_asIP(nextHop,&nInterface);
45         DSR_ADD_TO_PASSTHROUGH_BUFFER(pstruEventDetails->nDeviceId,pstruEventDetails->pPacket,pstruEventDetails->dEventTime);
46         DSR_DEV_VAR(pstruEventDetails->nDeviceId)->dsrMetrics.packetTransmitted++;
47         if(pstruEventDetails->pPacket->nDeviceId == pstruEventDetails->nDeviceId)
48             DSR_DEV_VAR(pstruEventDetails->nDeviceId)->dsrMetrics.packetOriginal++;
49     }
50 }

```

**Figure 9-45:** Viewing variables while debugging code

## Adding the variable to watch



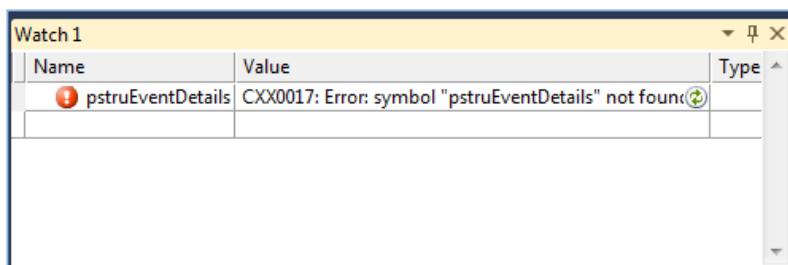
**Figure 9-46:** Adding the variable to watch

Watch the change in the variable as the code progresses 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.

CX0017: Error:symbol "Variable\_Name"not found.



**Figure 9-47:** Viewing external variables

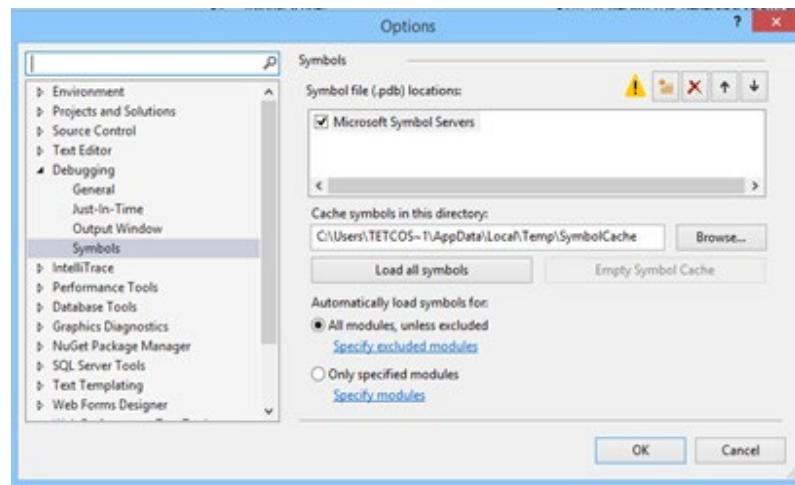
In the call stack window one can find the file in which that variable is situated. Right click on the dll file name in the call stack window, in this case NetworkStack.dll. Then in the pull-down menu which appears, select "load symbols from" and give the path of the pdb (program database) file.

A program database (.pdb) file, also called a symbol file, maps the identifiers that a user creates in source files for classes, methods, and other code to the identifiers that are used in the compiled executables of the project. The .pdb file also maps the statements in the source code to the execution instructions in the executables. The debugger uses this information to

determine: the source file and the line number displayed in the Visual Studio IDE and the location in the executable to stop at when a user sets a breakpoint. A symbol file also contains the original location of the source files, and optionally, the location of a source server where the source files can be retrieved from.

When a user debugs a project in the Visual Studio IDE, the debugger knows exactly where to find the .pdb and source files for the code. If the user wants to debug code outside their project source code, such as the Windows or third-party code the project calls, the user has to specify the location of the .pdb (and optionally, the source files of the external code) and those files need to exactly match the build of the executables.

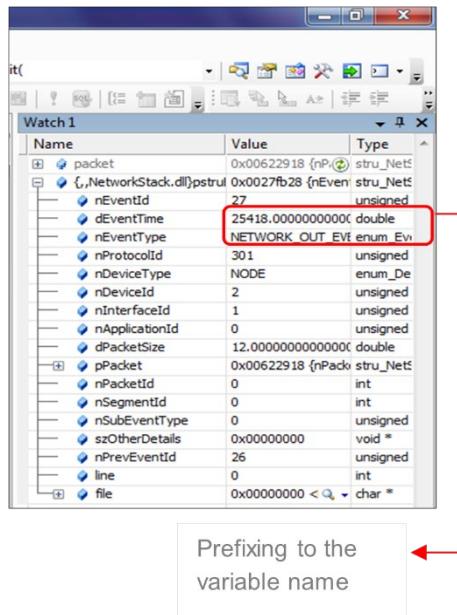
The pdb files are usually available in NetSim's install directory, else write to [support@tetcos.com](mailto:support@tetcos.com) for the latest copy of these debug files. Go to Tools > options>debugging>load all symbols.



**Figure 9-48:** Go to Tools > options > debugging > load all symbols in Visual Studio

**Note:** If the **load symbols** menu option is greyed, then it means symbols are already loaded

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 9-49:** 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.

```

187     unsigned int NES; // Number of BCC encoders for the channel
188
189     PACKET_STATUS firstpacketstatus;
190     UINT64 firstPacketTransmissionId;
191
192     double d0;
193     double pld0;
194
195     //Battery
196     bool isDeviceOn;
197     void* battery;
198
199     UINT64 transmissionId;
200 }IEEE802_PHY_VAR,*PIEEE802_11_PHY_VAR;
201
202 #ifndef SHARE_VARIABLE
203     _declspec(dllexport) IEEE802_PHY_VAR *var1;
204 #else
205     _declspec(dllimport) IEEE802_PHY_VAR *var1;
206 #endif
207
208 #define IEEE802_11_CURR_PHY IEEE802_11_PHY(pstruEventDetail)
209
210 //Propagation macro
211 //Function GET_BY_POWER_Ahms/tu_tui_nv_nvi_timei (parameters)

```

**Figure 9-50:** Code Modification done in IEEE802\_11\_Phys.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\_Phyln() function in IEEE802\_11\_Phyc 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.

```

308     }
309     else
310     {
311         if (is_first_packet(packet))
312             phy->firstPacketTransmissionId = hdr->transmissionId;
313
314         medium_notify_packet_received(packet);
315
316         double pdbm = GET_RX_POWER_dBm(packet->TransmitterId,
317                                         ifid,
318                                         pstruEventDetails->nDeviceId,
319                                         pstruEventDetails->nInterfaceId,
320                                         packet->pstruPhyData->dArrivalTime);
321
322         var1 = DEVICE_PHYVAR(pstruEventDetails->nDeviceId, pstruEventDetails->nInterfaceId);
323
324         if (pdBm < srcPhy->CurrentRxSensitivity_dBm)
325         {
326             set_radio_state(pstruEventDetails->nDeviceId, pstruEventDetails->nInterfaceId,
327                             RX_ON_IDLE, packet->nTransmitterId,
328                             phy->firstPacketTransmissionId ? phy->firstPacketTransmissionId : hc->last_tx);
329             return -1; // dest is not in range
330         }
331
332         if (isIEEE802_11_CtrlPpacket(packet) || !morefrag)
333             set_radio_state(pstruEventDetails->nDeviceId, pstruEventDetails->nInterfaceId,
334                             RX_ON_IDLE, packet->nTransmitterId,
335                             phy->firstPacketTransmissionId ? phy->firstPacketTransmissionId : hc->last_tx);
336
337         if(phy->firstpacketstatus==PacketStatus Collided)
338     }

```

**Figure 9-51:** Code Modification done in IEEE802\_11\_Phyc 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 folder of NetSim's current workspace

<C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and

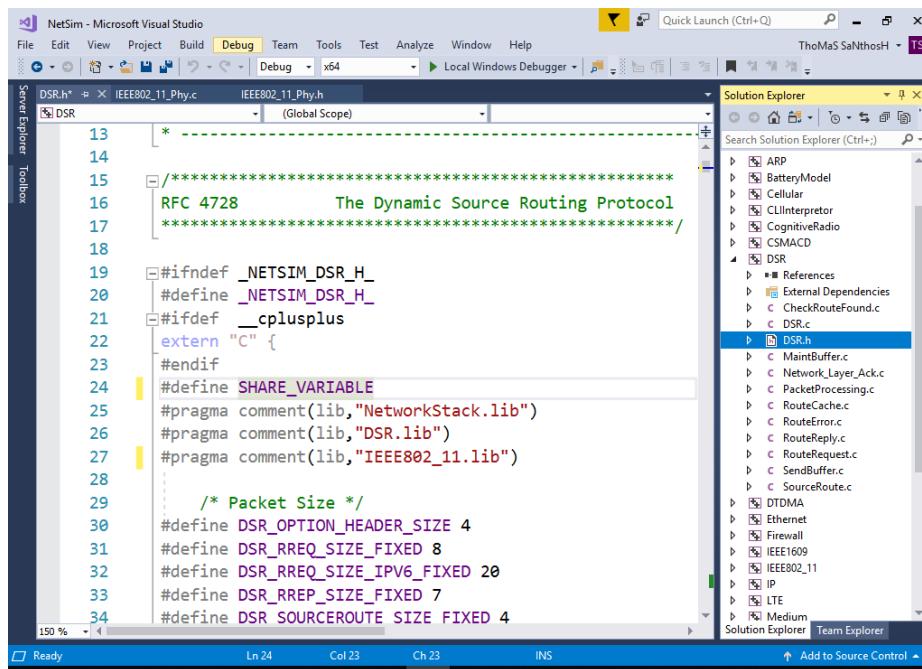
<C:\Users\PC\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit NetSim.

The Object file **IEEE802\_11.lib** which is also got created in the lib folder located in the current workspace path

<C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\src\Simulation\lib\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\src\Simulation\lib> for 32-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

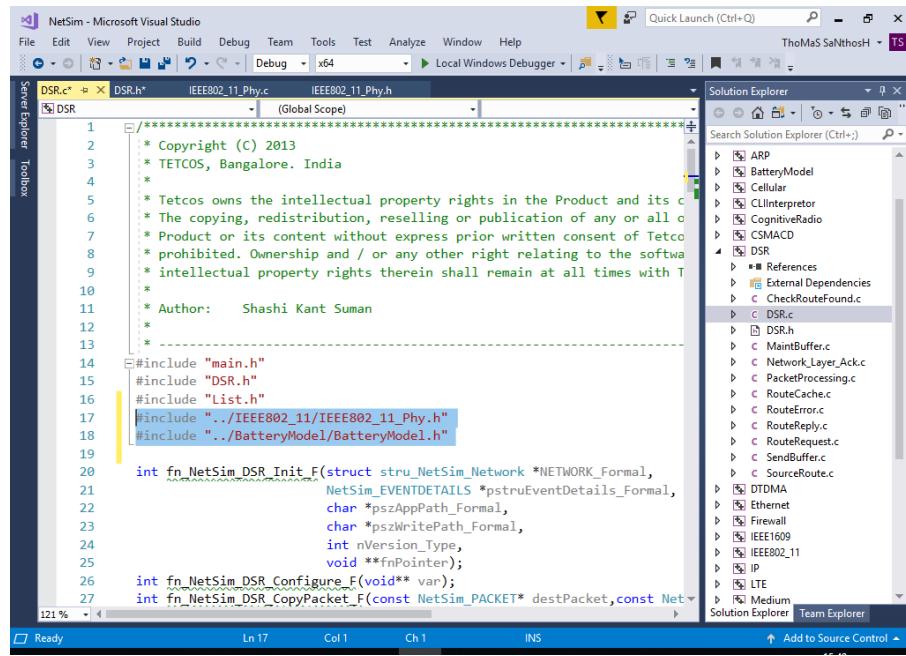
```
#define SHARE_VARIABLE
#pragma comment(lib,"IEEE802_11.lib")
```



**Figure 9-52:** 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 **Figure 9-53.**

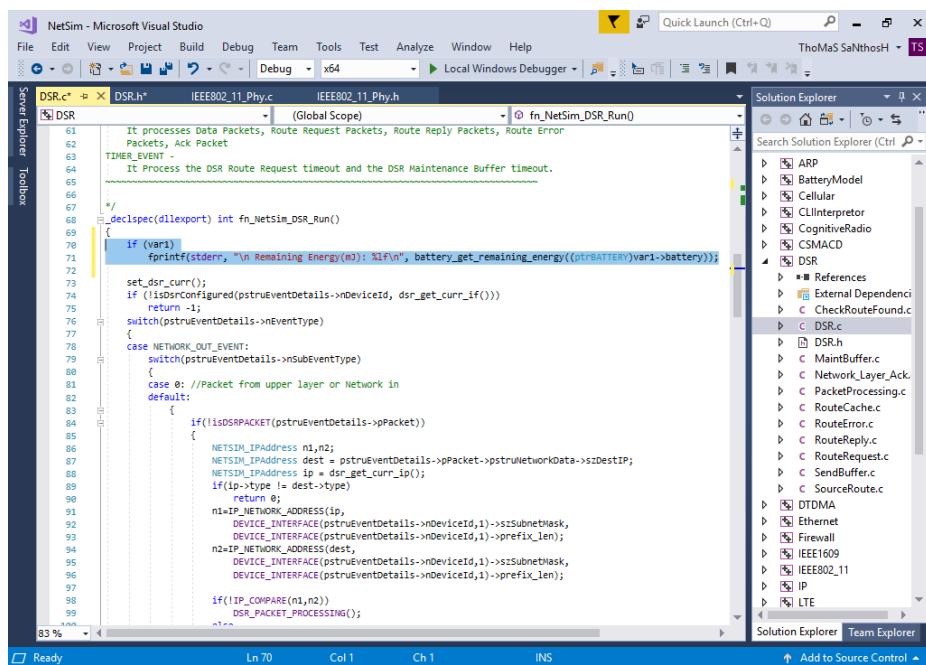
```
#include "../IEEE802_11/IEEE802_11_Phys.h"
#include "../BatteryModel/BatteryModel.h"
```



**Figure 9-53:** 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.

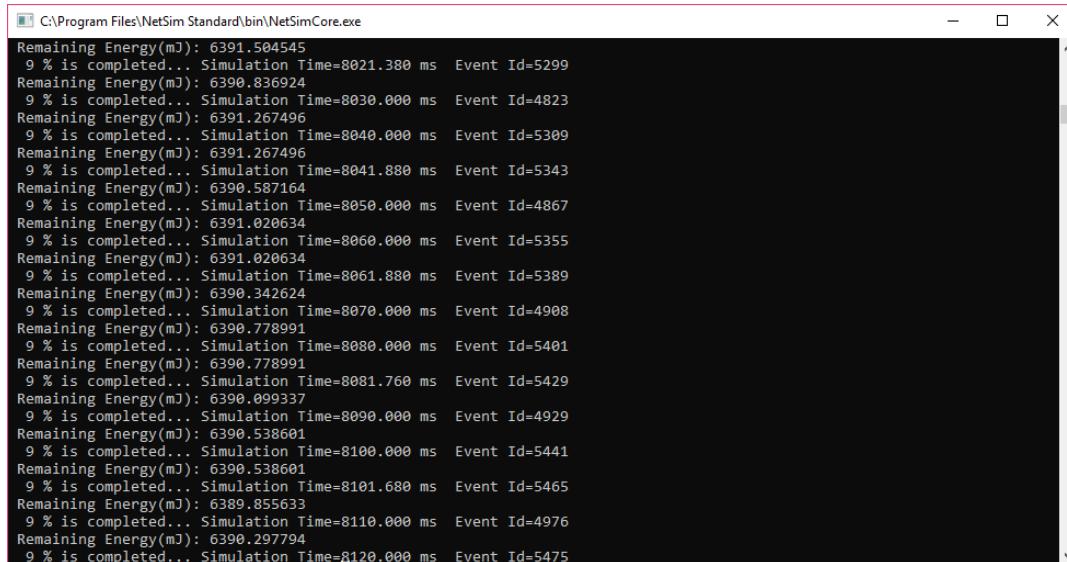
```
if (var1)
    fprintf(stderr, "\n Remaining Energy(mJ): %I\n"
, battery_get_remaining_energy((ptrBATTERY)var1->battery));
```



**Figure 9-54:** Code Related to Remaining Energy for nodes

The DSR project must be built and the resulting libDSR.dll file gets created in the bin folder of NetSim's current workspace path

<C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and  
 <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit. When a scenario is run, the remaining energy of the node will be printed to the simulation console as shown below **Figure 9-55.**



The screenshot shows a Windows command-line interface window titled 'C:\Program Files\NetSim Standard\bin\NetSimCore.exe'. The window contains a list of text entries, each consisting of a timestamp, an event ID, and a message indicating the remaining energy of a node. The messages are repeated every 10ms, showing a gradual decrease from 6391.504545 mJ to 6390.297794 mJ over approximately 10 seconds.

```

Remaining Energy(mJ): 6391.504545
9 % is completed... Simulation Time=8021.380 ms Event Id=5299
Remaining Energy(mJ): 6390.836924
9 % is completed... Simulation Time=8030.000 ms Event Id=4823
Remaining Energy(mJ): 6391.267496
9 % is completed... Simulation Time=8040.000 ms Event Id=5309
Remaining Energy(mJ): 6391.267496
9 % is completed... Simulation Time=8041.880 ms Event Id=5343
Remaining Energy(mJ): 6390.587164
9 % is completed... Simulation Time=8050.000 ms Event Id=4867
Remaining Energy(mJ): 6391.020634
9 % is completed... Simulation Time=8060.000 ms Event Id=5355
Remaining Energy(mJ): 6391.020634
9 % is completed... Simulation Time=8061.880 ms Event Id=5389
Remaining Energy(mJ): 6390.342624
9 % is completed... Simulation Time=8070.000 ms Event Id=4908
Remaining Energy(mJ): 6390.778991
9 % is completed... Simulation Time=8080.000 ms Event Id=5401
Remaining Energy(mJ): 6390.778991
9 % is completed... Simulation Time=8081.760 ms Event Id=5429
Remaining Energy(mJ): 6390.099337
9 % is completed... Simulation Time=8090.000 ms Event Id=4929
Remaining Energy(mJ): 6390.538601
9 % is completed... Simulation Time=8100.000 ms Event Id=5441
Remaining Energy(mJ): 6390.538601
9 % is completed... Simulation Time=8101.680 ms Event Id=5465
Remaining Energy(mJ): 6389.855633
9 % is completed... Simulation Time=8110.000 ms Event Id=4976
Remaining Energy(mJ): 6390.297794
9 % is completed... Simulation Time=8120.000 ms Event Id=5475

```

**Figure 9-55:** 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 %.2fms is X=% .2lf and Y = % .2lf \n", DEVICE_NAME(pstruEventDetails->nDeviceId),
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 9-56:** Code for Print node positions in MANET

Building Mobility project creates libMobility.dll inside the binary folder of NetSim's current workspace path <C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-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.
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 9-57**.

```

/** Defining sub event type of IEEE 802.15.4*/
enum enum_IEEE802_15_4_Subevent_Type
{
    SUPERFRAME_EVENT = MAC_PROTOCOL_IEEE802_15_4*100+1,
    CARRIERSENSE_START,
    CARRIERSENSE_END,
    ACK_TIMEOUT,
    BEACON_TRANSMISSION,
    GOFORSLEEP,
    BEACON_TRANSMISSION_END,
    CAP_END,
    CFP_END,
    ACK_EVENT,
    CSMA_DATATRANSFER,
    CHANGE_RX,
    CHANGE_RXANDSENDDATA,
    UPDATE_MEDIUM,
    MY_EVENT,
};

```

**Figure 9-57:** 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 9-58**.

```
enum enum_IEEE_802_15_4_ControlPacket_Type
{
    BEACON_FRAME = MAC_PROTOCOL_IEEE802_15_4*100+1,
    SLEEP_FRAME,
    ACK_FRAME,
    MY_PACKET,
};
```

**Figure 9-58:** 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");
//to see the packet in animation
// 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");
```

This is used visualize the packet transmission in the packet animation.

8. In 802\_15\_4.c file, goto fn\_NetSim\_Zigbee\_Init() function and add the following code in red color to call the timer\_event. i.e. MY\_EVENT

```
declspec (dllexport) int fn_NetSim_Zigbee_Init(struct stru_NetSim_Network
*NETWORK_Formal,\NetSim_EVENTDETAILS
*pstruEventDetails_Formal,char *pszAppPath_Formal,\char
*pszWritePath_Formal,int nVersion_Type,void **fnPointer)
{
    pstruEventDetails=pstruEventDetails_Formal;
    NETWORK=NETWORK_Formal;
    pszAppPath =pszAppPath_Formal;
    pszIOPath = pszWritePath_Formal;
    //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
    fn_NetSim_Zigbee_Init_F(NETWORK_Formal,pstruEventDetails_Formal,psz
    AppPath_Formal,\pszWritePath_Formal,nVersion_Type,fnPointer);
    return 0;
}
```

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 in red color 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\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit. Create a basic scenario in WSN with 2 sensors and 1 sink node.
  11. While creating the application between two sensors, set

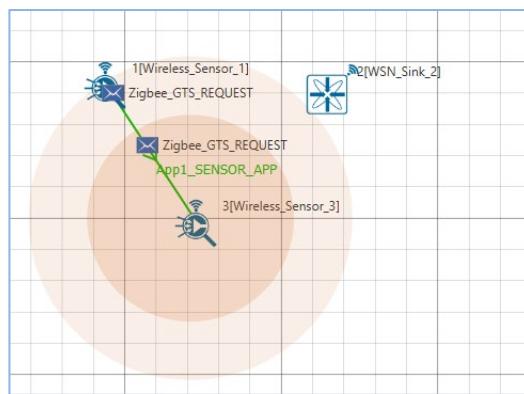
11. While creating the application between two sensors, set

## Application Type - Sensor App

## Interarrival Time - 5000

12. Run simulation for 10 seconds.

13. Play packet animation. Here users can see that Sink node broadcasts "MY\_PACKET" to the sensor.



**Figure 9-59:** In animation window Sink node broadcasts “MY\_PACKET” to the sensor

14. Also, open packet trace and users can filter the control packet and see all the packet details of “MY\_PACKET” written in the packet trace.

**Figure 9-60:** Filter the Packet Type to control packet and See “MY\_PACKET” in Packet Trace

15. To analyse the “MY\_EVENT” users can open event trace and filter the subevent type as “MY\_EVENT”. Here users can analyse that the event occurs for every 1 seconds.

Event_Id	Event_Type	Event_Tir	Device_Type	I	A	S	Protocol_Name	Subevent_Type
4	TIMER_EVENT	0	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
9	TIMER_EVENT	1000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
4280	TIMER_EVENT	2000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
8519	TIMER_EVENT	3000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
12820	TIMER_EVENT	4000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
17025	TIMER_EVENT	5000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
21223	TIMER_EVENT	6000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
25452	TIMER_EVENT	7000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
29735	TIMER_EVENT	8000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT
33993	TIMER_EVENT	9000000	SINKNODE	3	1	0	0 IEEE802.15.4	MY_EVENT

**Figure 9-61:** 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_Utils_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 elements 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\_Address.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 appropriate header (.h) files inside:/NetSim\_Standard/src/simulation/include or read through the doxygen source code documentation available inside NetSim →Help →NetSim 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)(constNetSim\_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 behaviour. 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 linear congruential Random Number Generator (RNG) to generate the randomness. The RNG uses two seeds values to initialize the RNG. 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 set of random numbers and thereby lead to a different sequence of events in NetSim. Therefore, the results are dependent on the initial seeding of the RNG. Because a particular random seed can potentially result in an anomalous, or non-representative behavior, it is important for each network scenario to be simulated with several random number seeds, to ascertain typical performance.

To calculate confidence intervals, users can run multi-seed parametric experiments, where one or more input parameters are varied, and for each input parameter value, multiple random number seeds are used to obtain a set of performance metrics.

## 10.2 Interfacing MATLAB with NetSim (Std/Pro versions)

NetSim provides run-time interfacing with MATLAB so that users don't have to rewrite code in C for features that are already available in MATLAB. They can 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.

#### Pre-requisites for Interfacing NetSim with MATLAB

- MATLAB Interfacing requires an installed version of MATLAB. Engine API functions cannot be run on a machine that only has the MATLAB Runtime.
- Both NetSim and MATLAB should use the same build; either 32-bit or 64-bit.

MATLAB functions can be called from NetSim's underlying protocol C source codes using MATLAB APIs. Following are some of the MATLAB Engine API functions that can be used from NetSim C source codes:

Engine	Type for MATLAB engine
<b>engOpen</b>	Start MATLAB engine session
<b>engOpenSingleUse</b>	Start MATLAB engine session for single, nonshared use
<b>engClose</b>	Quit MATLAB engine session
<b>engEvalString</b>	Evaluate expression in string
<b>engGetVariable</b>	Copy variable from MATLAB engine workspace
<b>engPutVariable</b>	Put variable into MATLAB engine workspace
<b>engGetVisible</b>	Determine visibility of MATLAB engine session
<b>engSetVisible</b>	Show or hide MATLAB engine session
<b>engOutputBuffer</b>	Specify buffer for MATLAB output

**Table 10-1:** MATLAB Engine API functions

#### 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.

### 10.2.1 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.

**Note:** This example uses 32-bit version of NetSim and MATLAB. Settings will slightly vary in case of 64-bit version of the software.

#### Procedure:

1. Create a MATLAB\_Interface.c file inside the IEEE802\_11 folder which can be found in the current workspace location of NetSim that you are running and it would be something like

"C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\src\Simulation\IEEE802\_11" For more information on NetSim workspace refer Section 4 "Workspaces and Experiments". Write the following code inside the MATLAB\_Interface.c file:

```
/*
 *
 * This is a simple program that illustrates how to call the MATLAB
 * Engine functions from NetSim C Code.
 *
 */
#include<windows.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include"engine.h"
#include"mat.h"
#include"mex.h"

char buf[BUFSIZ];
Engine *ep;
```

```

int status;
mxArray *h = NULL, *i = NULL, *j = NULL, *k = NULL;
mxArray *out;
double *result;
double fn_netsim_matlab_init()
{
/*
 * Start the MATLAB engine
 */
fprintf(stderr, "\nPress any key to Initialize MATLAB\n");
_getch();
if(!(ep = engOpen(NULL))) {
    MessageBox((HWND)NULL, (LPCWSTR)"Can't start MATLAB engine",
               (LPCWSTR) "MATLAB_Interface.c", MB_OK);
    exit(-1);
}

engEvalString(ep, "desktop");

return 0;
}

double fn_netsim_matlab_run()
{

//write your own implementation here

int weibull_noncentrality = 1, weibull_scale = 2;

engPutVariable(ep, "h", h);
//use ProbDistUnivParam() function for matlab 2016
sprintf_s(buf, BUFSIZ, "h=ProbDistUnivParam('weibull',[%d %d])",
          weibull_noncentrality, weibull_scale);
//use makedist() function for matlab 2017
//sprintf_s(buf, BUFSIZ, "h=makedist('weibull',%d,%d)", weibull_noncentrality,
//          weibull_scale);

status = engEvalString(ep, buf);

```

```

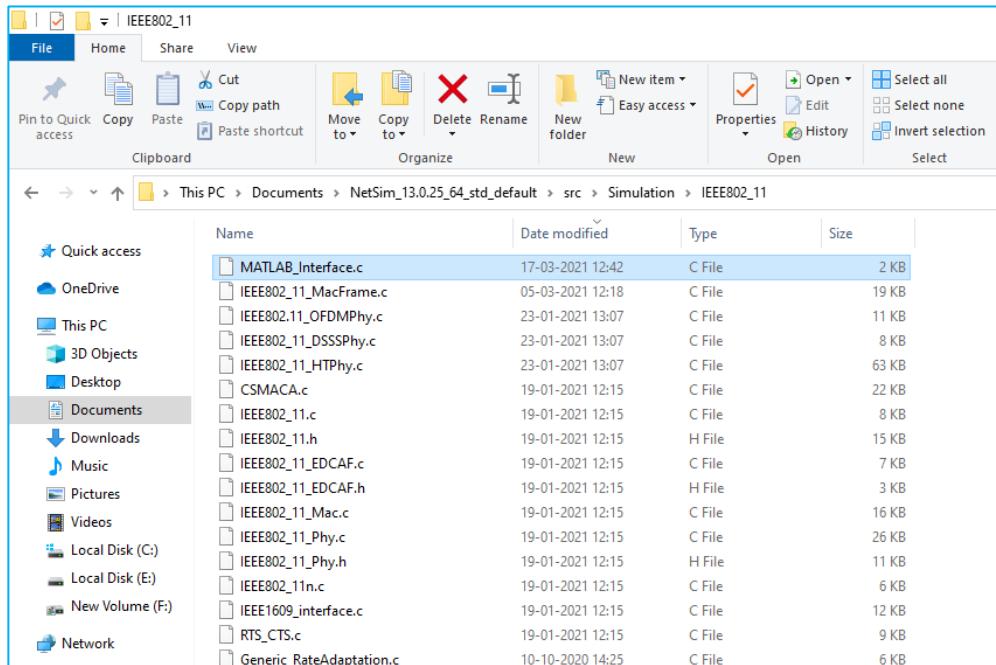
engPutVariable(ep, "i", i);
sprintf_s(buf, BUFSIZ, "i=random(h,1)");

status = engEvalString(ep, buf);
out = engGetVariable(ep, "i");
result = mxGetPr(out);

return *result;
}

double fn_netsim_matlab_finish()
{
    fprintf(stderr, "\nPress any key to close MATLAB\n");
    _getch();
    status = engEvalString(ep, "exit");
    return 0;
}

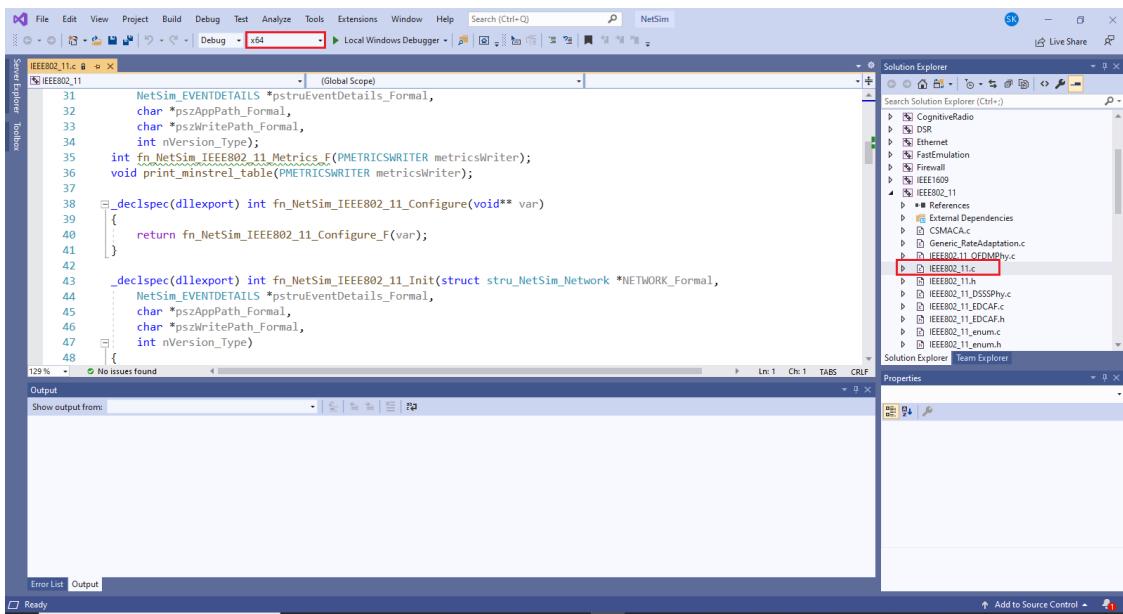
```



**Figure 10-2:** Create a MATLAB\_Interface.c inside the IEEE802\_11 folder

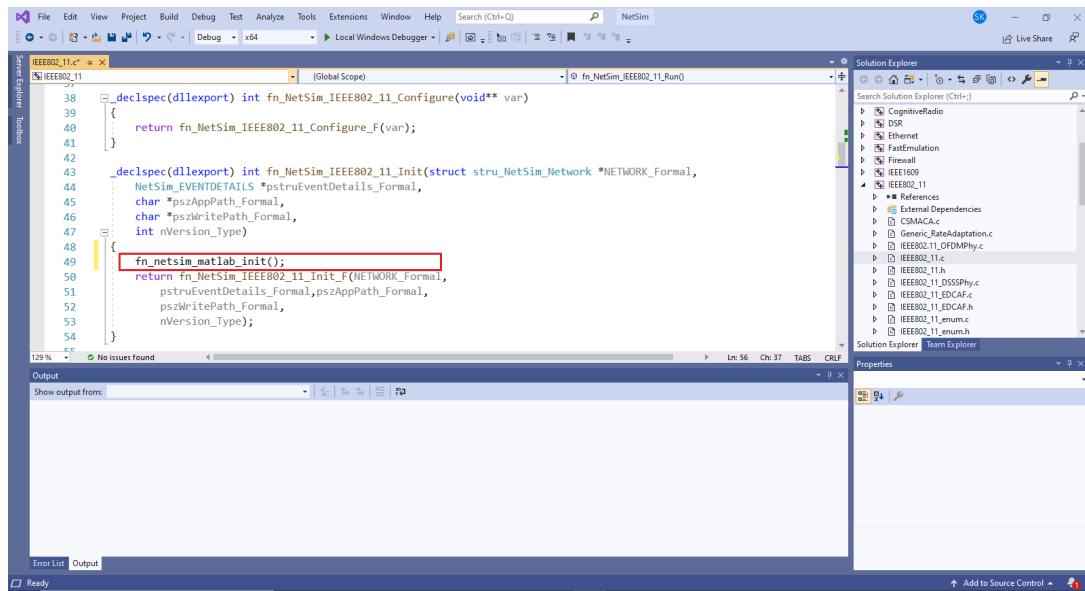
2. Now open the code and Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively.
3. Right click on “IEEE802\_11 Project” present in “Solution Explorer” window and select Add → Existing Item and select the MATLAB\_Interface.c file.

4. MATLAB\_Interface.c file contains the following functions.
  - i. fn\_netsim\_matlab\_init() - Opens the MATLAB Engine
  - ii. fn\_netsim\_matlab\_run() - Communicates with MATLAB Command Window
  - iii. fn\_netsim\_matlab\_finish() - Closes the MATLAB Engine
5. In the Solution Explorer double click on the IEEE802\_11.c file.



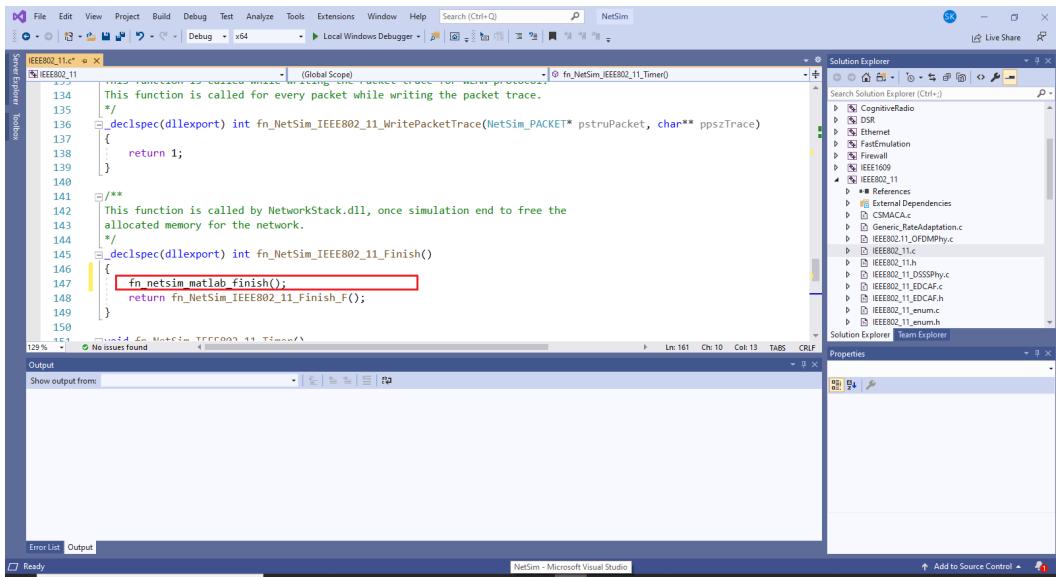
**Figure 10-3:** Solution Explorer double click on the IEEE802\_11.c file

6. Add a call to fn\_netsim\_matlab\_init(); inside the fn\_NetSim\_IEEE802\_11\_Init() function.



**Figure 10-4:** Added a fn\_netsim\_matlab\_init(); inside the fn\_NetSim\_IEEE802\_11\_Init() function

7. Similarly add a call to fn\_netsim\_matlab\_finish(); inside the fn\_NetSim\_IEEE802\_11\_Finish() function.



**Figure 10-5:** Added a `fn_netsim_matlab_finish()`; inside the `fn_NetSim_IEEE802_11_Finish()` function

8. In the Solution Explorer double click on the IEEE802\_11.h file. Add definitions of the following functions

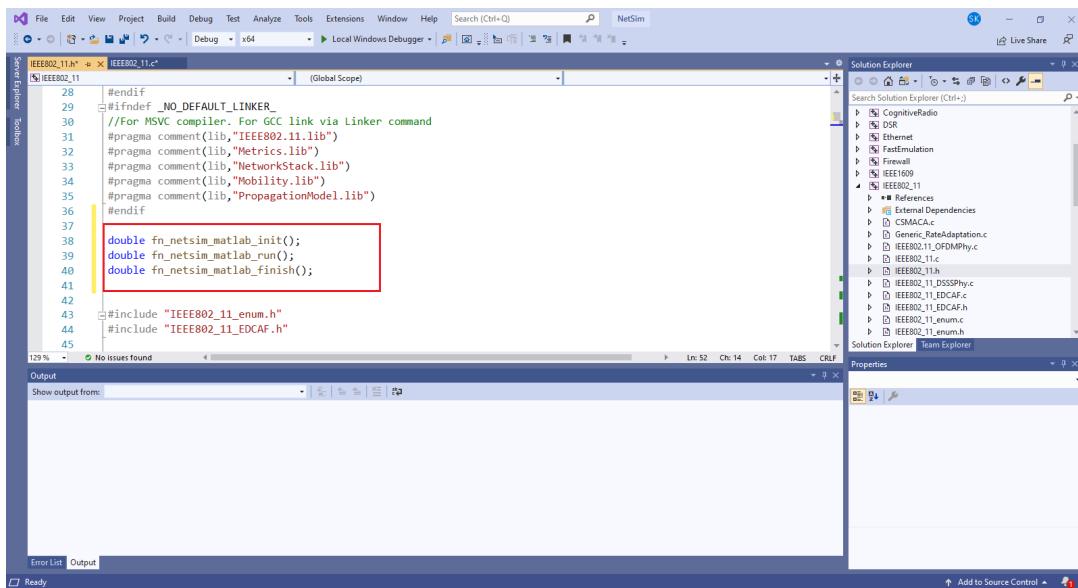
```

double fn_netsim_matlab_init();

double fn_netsim_matlab_run();

double fn_netsim_matlab_finish();

```



**Figure 10-6:** Added Matlab definitions in IEEE802\_11.h file

9. In the Solution Explorer double click on the IEE802\_11\_PHY.c file.
10. Inside `fn_Netsim_IEEE802_11_PHYIn()` function comment the lines,

```

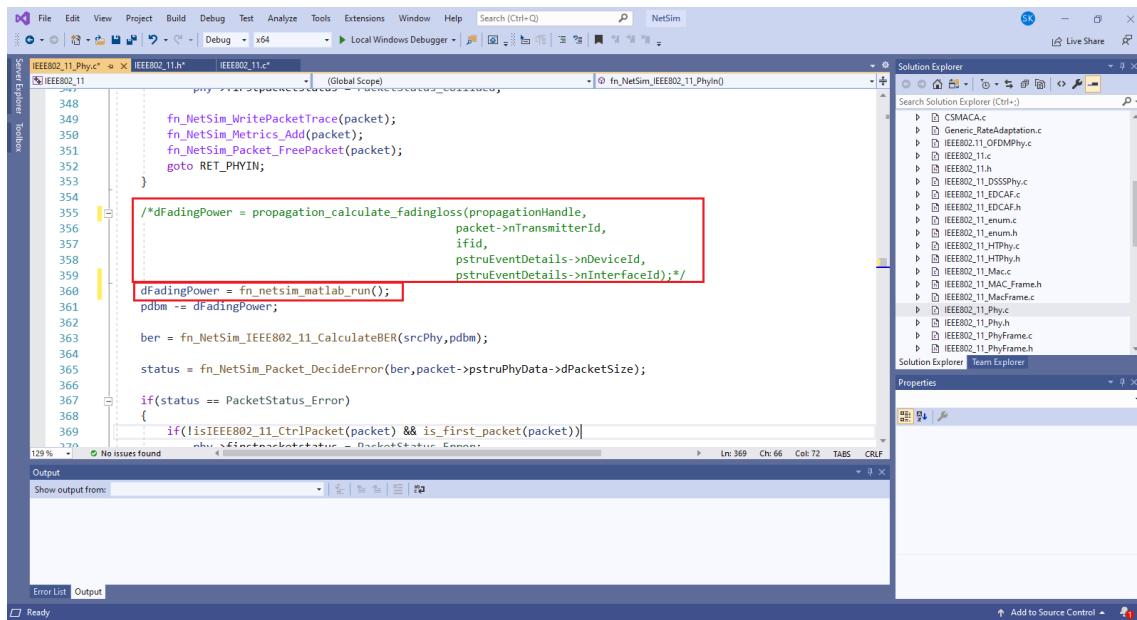
dFadingPower = propagation_calculate_fadingloss(propagationHandle,

```

```
packet->nTransmitterId,ifid,pstruEventDetails->nDeviceId, pstruEventDetails->nInterfaceId);
```

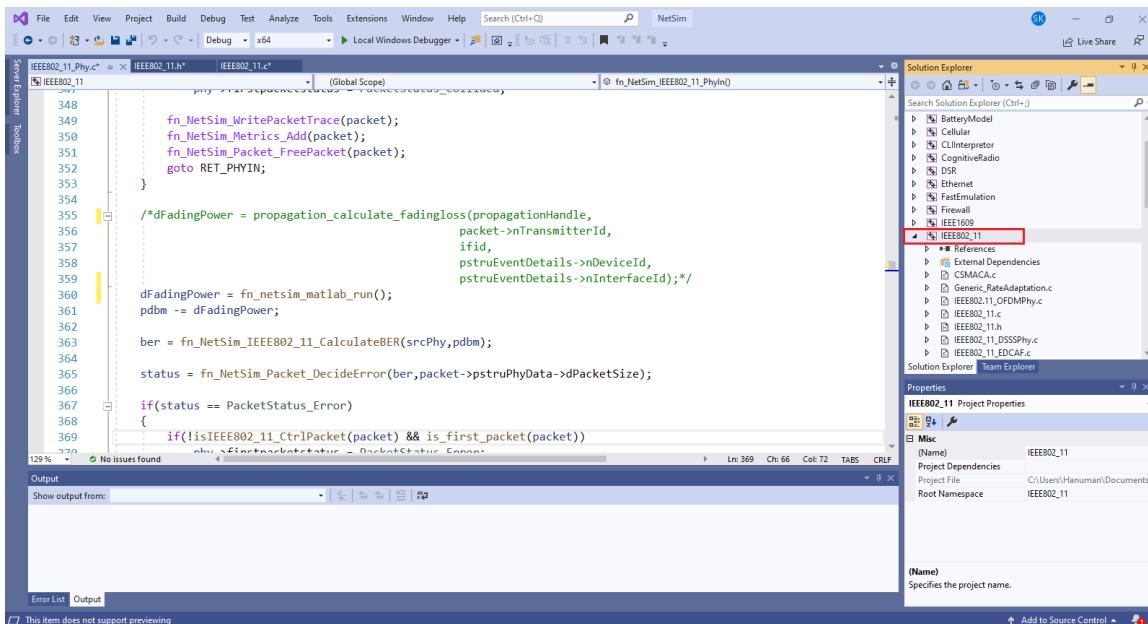
11. Make a call to the fn\_netsim\_matlab\_run() function by adding the following line,

```
dFadingPower = fn_netsim_matlab_run();
```



**Figure 10-7:** Call to the fn\_netsim\_matlab\_run() function

12. To compile a MATLAB engine application in the Microsoft Visual Studio (2017) environment, Right click on the IEEE802\_11 project and select PROPERTIES in the solution explorer. Once this window has opened, make the following changes:



**Figure 10-8:** Right click on the IEEE802\_11 project and select Properties

13. Under C/C++ → General, add the following directory to the field ADDITIONAL INCLUDE DIRECTORIES:

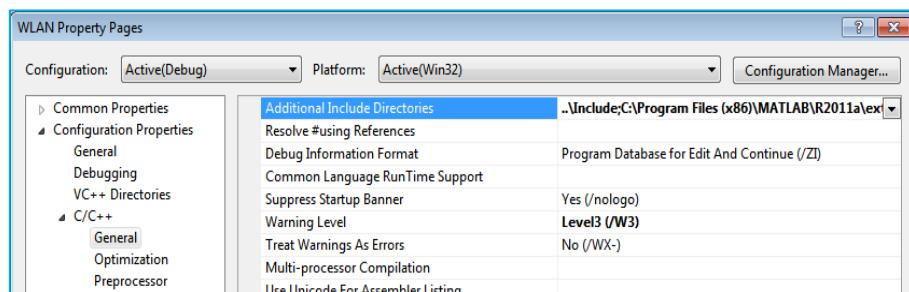
<Path where MATLAB is installed>\extern\include

**NOTE:** To determine path where MATLAB is installed, entering the following command in the MATLAB command prompt:

matlabroot

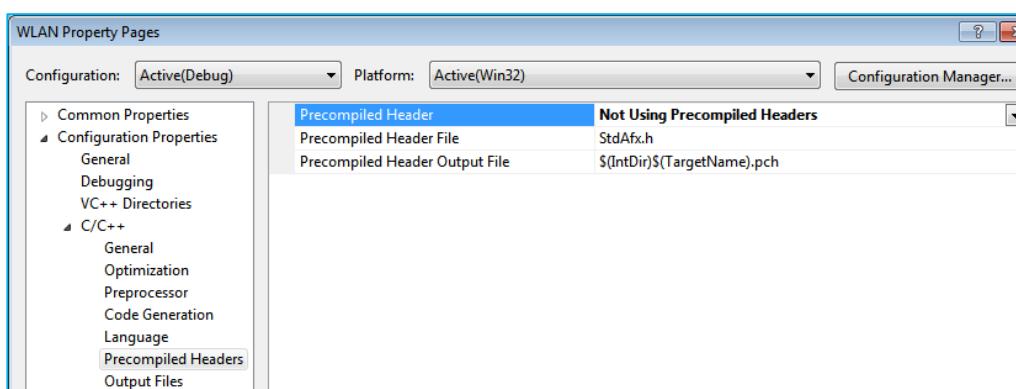


**Figure 10-9: MATLAB Command Prompt**



**Figure 10-10: Determine MATLAB Path in Additional Include Directories**

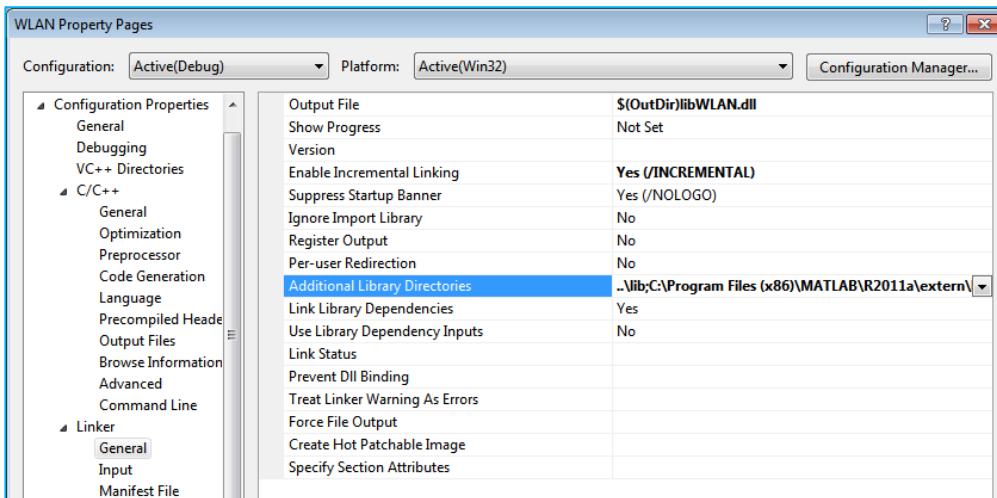
14. Under C/C++ → Precompiled Headers, set PRECOMPILED HEADERS as "Not Using Precompiled Headers".



**Figure 10-11: Select Precompiled Header as "Not Using Precompiled Headers"**

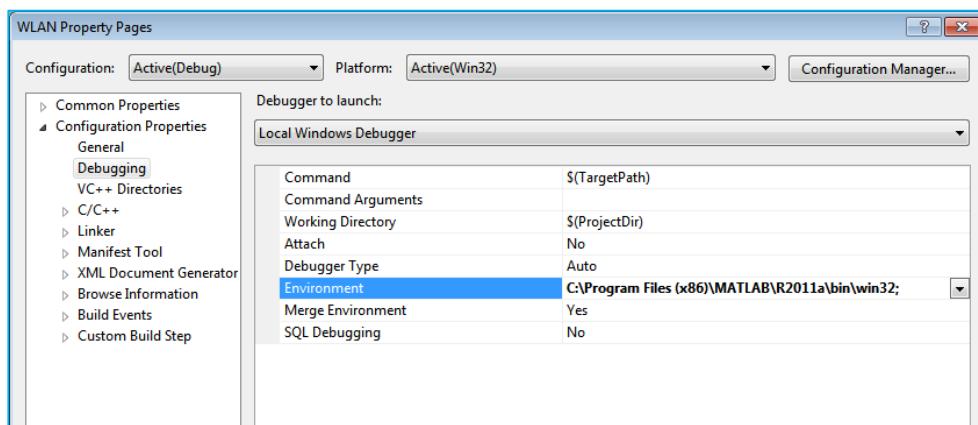
15. Under Linker → General, add the directory to the field ADDITIONAL LIBRARY DIRECTORIES:

<Path where MATLAB is installed>\extern\lib\win32\microsoft



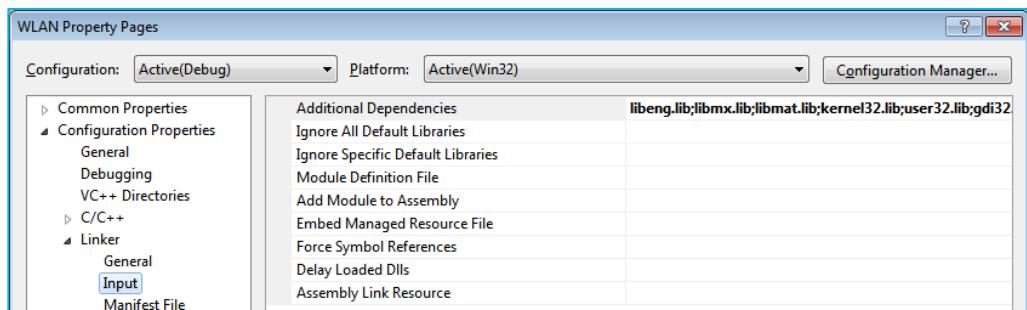
**Figure 10-12:** Set Additional Library Directories to <Path where MATLAB is installed>\extern\lib\win32\microsoft

16. Under Configuration Properties → Debugging, Add the following Target path in the ENVIRONMENT: <Path where MATLAB is installed>\bin\win32



**Figure 10-13:** Set Environment to <Path where MATLAB is installed>\bin\win32

17. Under Linker → Input, add the following names to the field marked ADDITIONAL DEPENDENCIES: libeng.lib;libmx.lib;libmat.lib;



**Figure 10-14:** Set Additional Dependencies as libeng.lib;libmx.lib;libmat.lib;

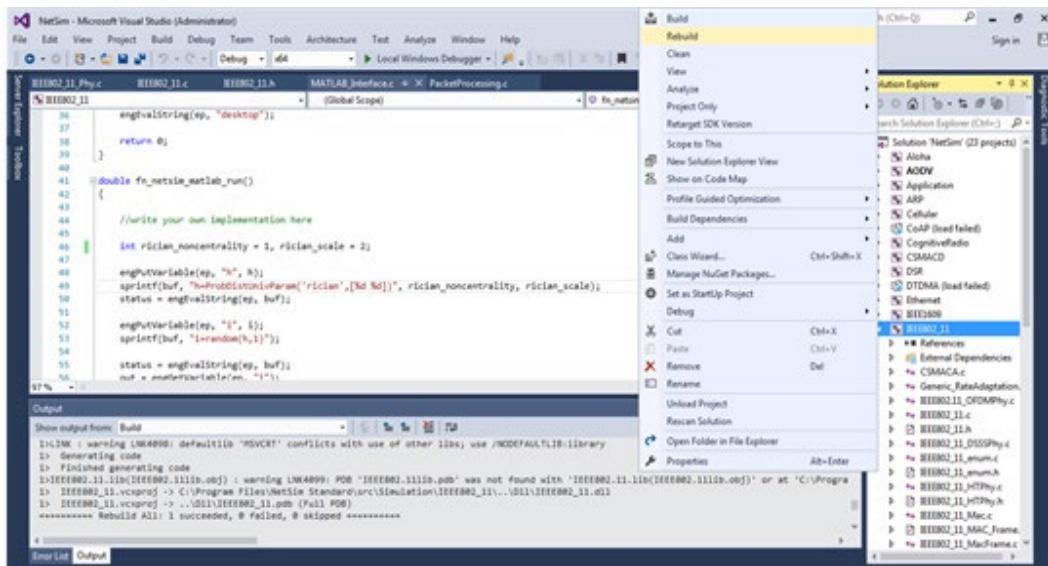
18. Make sure that the following directory is in the environment variable PATH:

<Path where MATLAB is installed>\bin\win32.

#### Notes:

1. To do step 14, check the Windows system path by clicking on Start → Right click on Computer → Properties → Advanced System Settings → Environment variables → System Variables → Open "Path" for editing.
2. If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 32-bit application, the directory in the MATLAB 32-bit installation must be the first one on the PATH). To run 64-bit NetSim, users have to change the above mentioned matlab paths to 64-bit matlab paths.

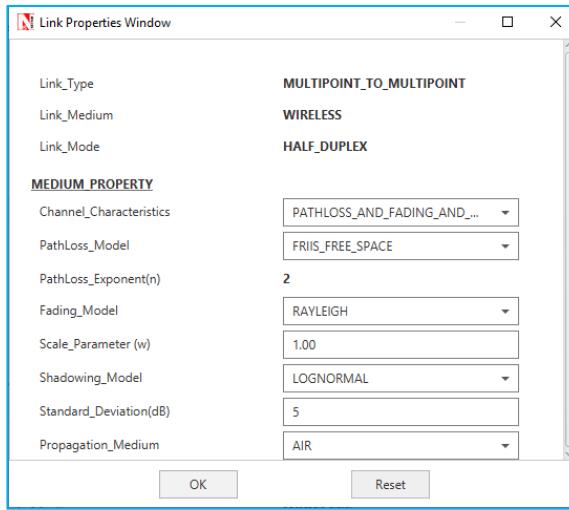
19. Now Right Click on IEEE802\_11 project and select Rebuild.



**Figure 10-15:** Rebuild IEEE802\_11 project

20. A new libIEEE802.11.dll gets created in the bin folder of NetSim's current workspace path <C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit. For more information, follow steps provided in Section 9.1 "Writing your own code".

21. 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.



**Figure 10-16:** Wireless Link Properties Window

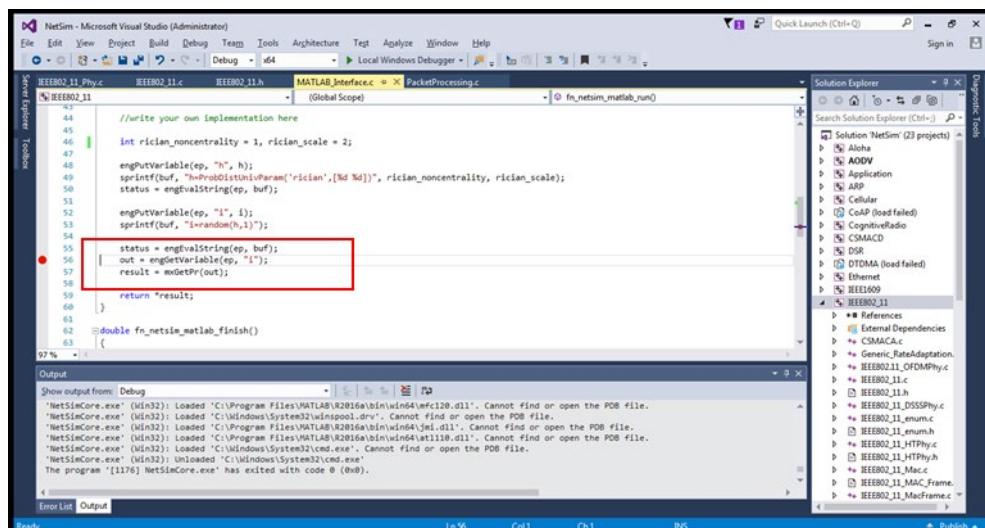
22. Perform Simulation. You will find that once the Simulation starts MATLAB command window starts and gets closed once the simulation is over.

**Note:** On Windows systems, engOpen opens a COM channel to MATLAB. The MATLAB software you registered during installation starts. If you did not register during installation, enter the following command at the MATLAB prompt:

```
!matlab -regserver
```

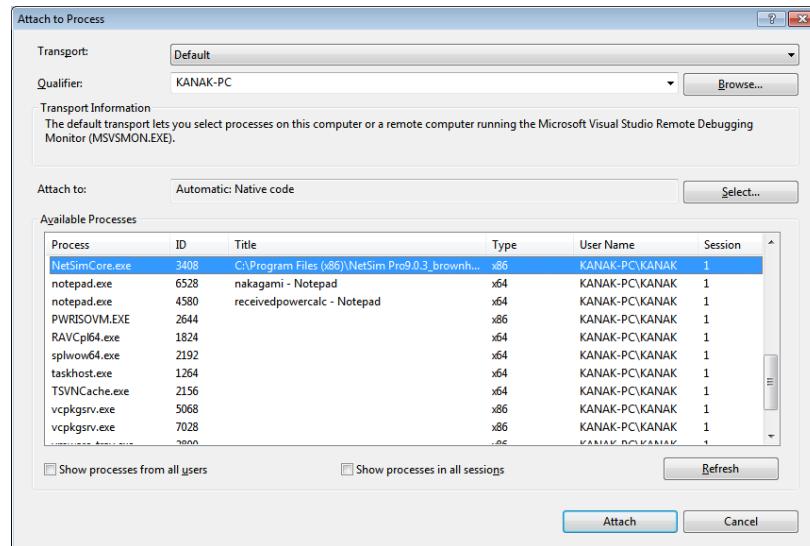
## 10.2.2 Debug and understand communication between NetSim and MATLAB

- In the Solution Explorer double click on MATLAB\_Interface.c file and place a breakpoint inside the fn\_netsim\_matlab\_run() function before the return statement.



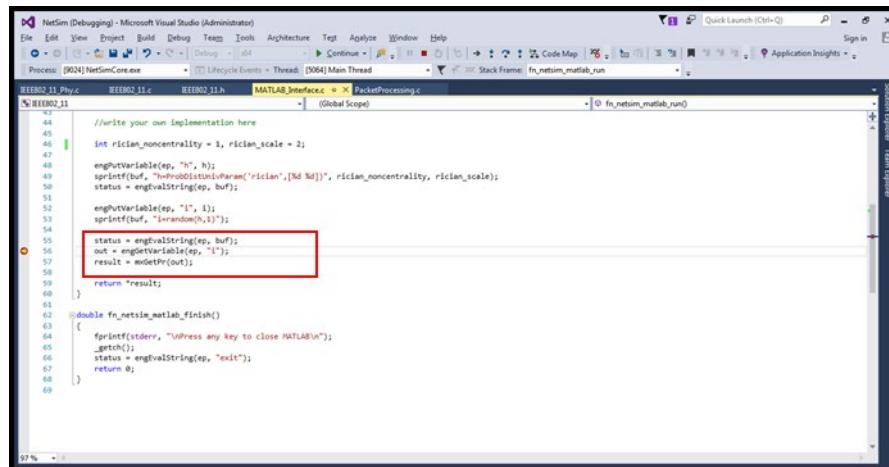
**Figure 10-17:** Place a breakpoint inside the fn\_netsim\_matlab\_run() function

2. Rebuild the code.
3. Now run the NetSim Scenario. The simulation window stops for user interrupt.
4. In Visual studio, go to Debug → Attach to Process.
5. From the list of Processes select NetSimCore.exe and click on Attach.



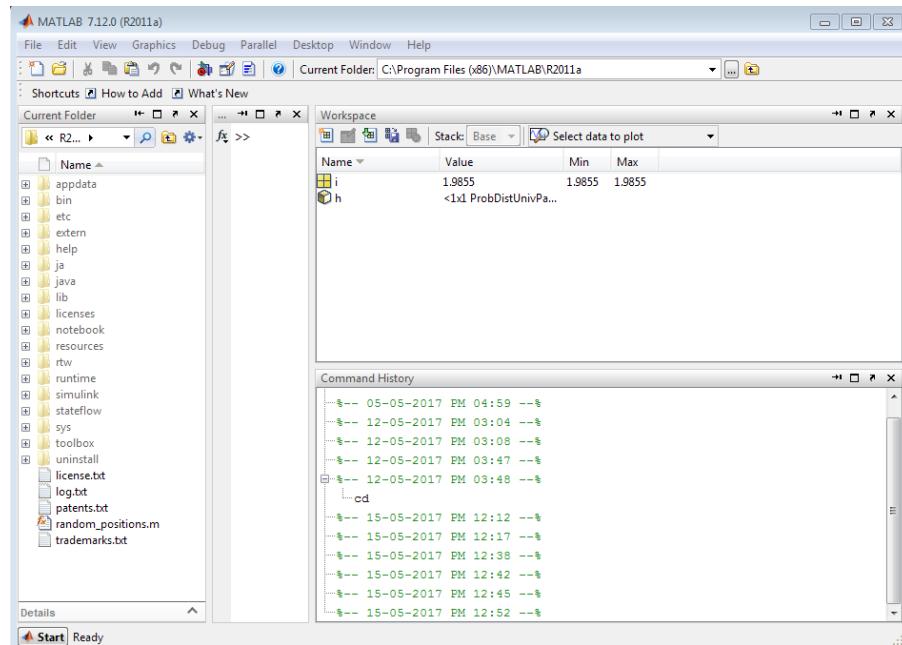
**Figure 10-18:** Select NetSimCore.exe in Attach to Process Window

6. Now go to the Simulation window and press Enter.
7. MATLAB Command Window and MATLAB Desktop Window will start and breakpoint in Visual Studio gets triggered.

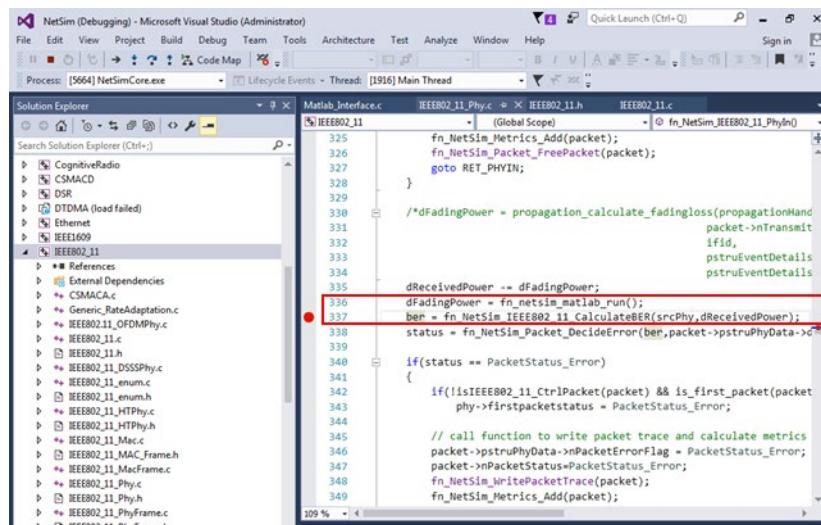


**Figure 10-19:** Once Simulation Start and breakpoint gets triggered in Visual Studio

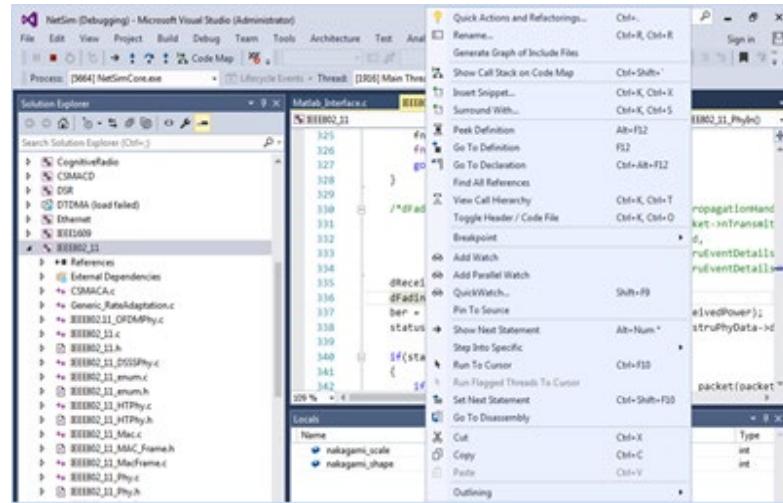
8. Now when debugging (say, by pressing F5 each time) you will find the computation taking place in the MATLAB Workspace.

**Figure 10-20:** MATLAB Workspace

9. This value of i obtained from MATLAB is used to calculate fading power instead of the already available models in NetSim.
10. Now place another breakpoint after the line dFadingPower = fn\_netsim\_matlab\_run()

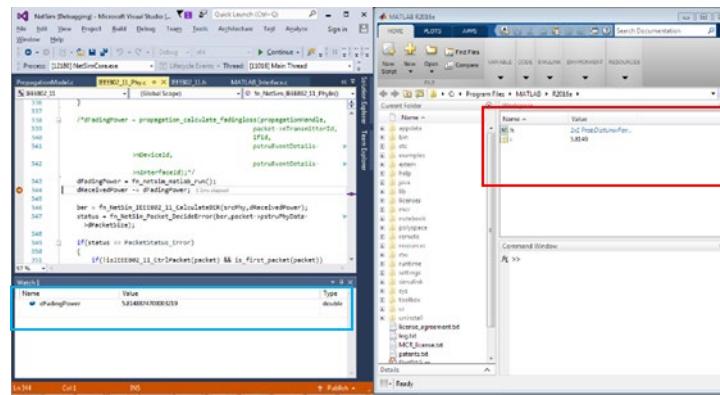
**Figure 10-21:** Added Another breakpoint after the line dFadingPower = fn\_netsim\_matlab\_run()

11. Add the variable dFadingPower in IEEE802\_11.Phy.c file, to watch. For this, right click on the variable dFadingPower and select “Add Watch” option. You will find a watch window containing the variable name and its value in the bottom left corner.



**Figure 10-22:** Right Click on dFadingPower and Select “Add Watch” option

12. Now when debugging (say by pressing F5 each time) you will find that the watch window displays the value of dFadingPower whenever the control reaches the recently set breakpoint. You will also find that the value of dFadingPower in the Visual Studio Watch window and the value of i in the MATLAB workspace window are similar.



**Figure 10-23:** The Visual Studio Watch window and the value of i in the MATLAB workspace window are similar

### 10.2.3 Implement Weibull Distribution of MATLAB in NetSim using .m file:

#### Procedure:

1. Create a file named weibull\_distribution.m file inside <Path where MATLAB is installed>. The weibull\_distribution.m file contains the following code:

```
function WLAN= weibull_distribution(noncentrality,scale)
%use ProbDistUnivParam() function for matlab 2016

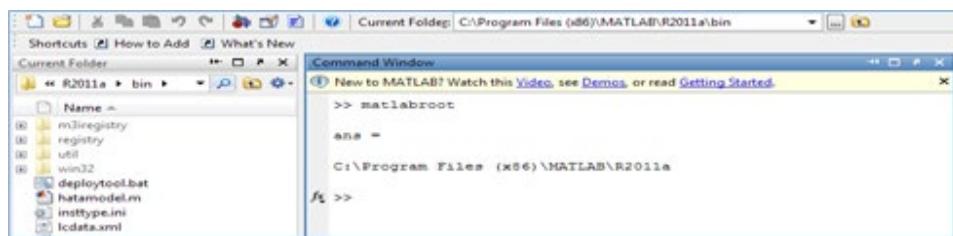
h=ProbDistUnivParam('weibull',[noncentrality,scale]);
%use makedist() function for matlab 2017
```

```
%h=makedist('weibull',noncentrality,scale);
i=random(h,1);
WLAN=i;
```

2. Place this file in the MATLAB's default working directory. This will usually be MATLAB's root directory or the bin folder in MATLAB's installation path.

**NOTE:** To determine path where MATLAB is installed, entering the following command in the MATLAB command prompt:

matlabroot



**Figure 10-24:** MATLAB Command Prompt

3. You will have to create a MATLAB\_Interface.c file in the IEEE802\_11 folder similar to the previous example. The functions fn\_netsim\_matlab\_init() and fn\_netsim\_matlab\_finish() will remain the same. Modify the function fn\_netsim\_matlab\_run() that is part of MATLAB\_Interfacing.c which was used in the previous example as shown below:

```
double fn_netsim_matlab_run()
{
    //write your own implementation here
    int weibull_noncentrality = 1, weibull_scale = 2;
    engPutVariable(ep, "h", h);
    sprintf_s(buf,BUFSIZ,      "k=weibull_distribution(%d,%d)",      weibull_noncentrality,
    weibull_scale);
    status = engEvalString(ep, buf);
    out = engGetVariable(ep, "k");
    result = mxGetPr(out);
    return *result;
}
```

4. Follow steps 2 to 14 as explained in the section “Implement Weibull Distribution of MATLAB in NetSim without using .m file” above.

5. A call to the weibull\_distribution () function inside the weibull\_distribution.m file is made, and weibull\_noncentrality and weibull\_scale parameters are passed from NetSim.
6. Right Click on IEEE802\_11 project and select Rebuild.
7. A new libIEEE802.11.dll will get created in the bin folder of NetSim's current workspace path <C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit. Open NetSim in Administrative mode. Create a Network scenario involving IEEE802\_11 say MANET, right click on the environment and select properties. Make sure that the Channel Characteristics is set to PathLoss and Fading and Shadowing.
8. You will find that once the Simulation is run MATLAB Command Window starts and gets closed once the Simulation is over. You can also debug the code to understand the communication between NetSim and MATLAB as explained in the DEBUGGING section above.

#### 10.2.4 Plot a histogram in MATLAB per a Weibull distribution (using .m file)

**Procedure:**

1. Create a file NETSIM\_MATLAB.m file containing the following code:

```
function WLAN=NETSIM_MATLAB(choice,varargin)
switch(choice)
    case'weibull'
        %use ProbDistUnivParam function for matlab 2016
        h=ProbDistUnivParam('weibull',[varargin{1},varargin{2}]);
        %use makedist function for matlab 2017
        %h=makedist('weibull',varargin{1}, varargin{2});
        i=random(h,1);
        fid = fopen('plotvalues.txt','a+');
        fprintf(fid,'%f',i);
        fprintf(fid,'\r\n');
        fclose('all');
        WLAN=i;
    case'plothistogram'
```

```

fid=fopen('plotvalues.txt');

mx=fscanf(fid,'%f');

hist(mx);

fclose('all');

end

```

2. Modify the function fn\_netsim\_matlab\_run() that is part of MATLAB\_Interfacing.c which was used in the previous example.

```

double fn_netsim_matlab_run(char *arr)
{
    //write your own implementation here

    int weibull_noncentrality = 1, weibull_scale = 2;

    if (strcmp(arr, "weibull") == 0)
    {
        engPutVariable(ep, "h", h);
        sprintf_s(buf,BUFSIZ, "h=NETSIM_MATLAB('weibull',%d,%d)",
        weibull_noncentrality, weibull_scale);
        status = engEvalString(ep, buf);
        out = engGetVariable(ep, "h");
        result = mxGetPr(out);
        return *result;
    }
    else if (strcmp(arr, "plothistogram") == 0)
    {
        status = engEvalString(ep, "NETSIM_MATLAB('plothistogram')");
        return 0;
    }
    else
        return 0;
}

```

Follow steps 2 to 11 as explained in the section on “Implement weibull Distribution of MATLAB in NetSim without using .m file” above.

- A call to the NetSim\_MATLAB() function inside the NetSim\_MATLAB.m file is made, for fading power calculation with parameters distribution('weibull'), weibull\_noncentrality and weibull\_scale parameters are passed from NetSim.
- A call to the NetSim\_MATLAB() function inside the NetSim\_MATLAB.m file is made, for plotting histogram for the values generated by MATLAB.
- Also add the following call to fn\_netsim\_matlab\_run() function along with a \_getch() to plot the histogram before closing the MATLAB Engine.

```

127
128     /**
129      This function is called by NetworkStack.dll, once simulation end to free the
130      allocated memory for the network.
131   */
132   __declspec(dllexport) int fn_NetSim_IEEE802_11_Finish()
133   {
134       fn_netsim_matlab_run("plothistogram");
135       _getch();
136       fn_netsim_matlab_finish();
137       return fn_NetSim_IEEE802_11_Finish_F();
138   }
139

```

**Figure 10-25:** Added fn\_netsim\_matlab\_run() function along with a \_getch() to get histogram plot before closing the MATLAB Engine

- Similarly in the call made to fn\_netsim\_matlab\_run() function in IEEE802\_11\_Phys.c file add the parameter “weibull” as shown below:-

```

333         fn_NetSim_Metrics_Add(packet);
334         fn_NetSim_Packet_FreePacket(packet);
335         goto RET_PHYIN;
336     }
337
338     /*dFadingPower = propagation_calculate_fadingloss(propagationHandle,
339     packet->nTransmitterId,
340     ifid,
341     pstruEventDetails->nDeviceId,
342     pstruEventDetails->nInterfaceId);*/
343     dFadingPower = fn_netsim_matlab_run("rician");
344     dReceivedPower -= dFadingPower;
345
346     ber = fn_NetSim_IEEE802_11_CalculateBER(srcPhy,dReceivedPower);
347     status = fn_NetSim_Packet_DecideError(ber,packet->pstruPhyData->dPacketSize);
348
349     if(status == PacketStatus_Error)
350     {
351         if(!isIEEE802_11_CtrlPacket(packet) && is_first_packet(packet))
352             phy->firstpacketstatus = PacketStatus_Error;
353     }

```

**Figure 10-26:** Added the parameter “weibull” in fn\_netsim\_matlab\_run() function in IEEE802\_11\_Phys.c file

- Also modify the function definition of fn\_netsim\_matlab\_run() function in IEEE802\_11.h file as shown below:

```

IEEE802_11.c* MATLAB_Interface.c PropagationModel.c IEEE802_11.h
(Global Scope) fn_netsim_matlab_run(char *arr)

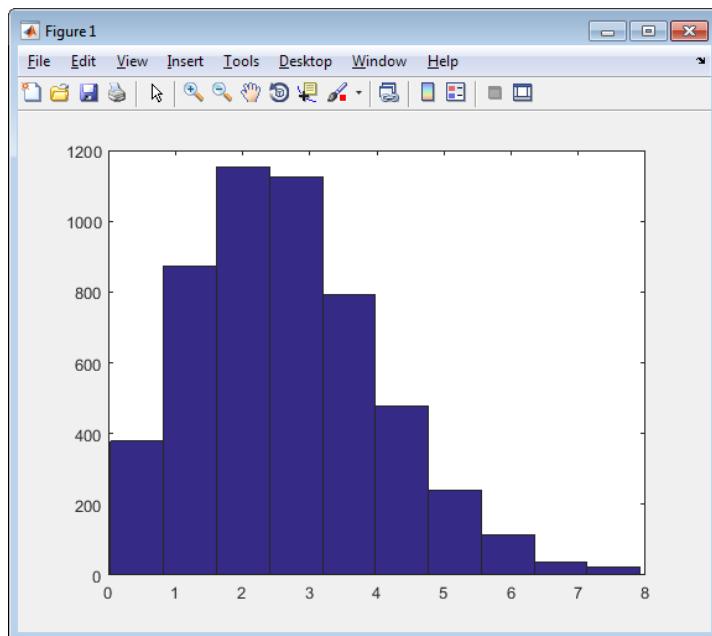
/*
 * Copyright (C) 2016
 * TETCOS, Bangalore, India
 *
 * Tetcos owns the intellectual property rights in the Product and its content.
 * The copying, redistribution, reselling or publication of any or all of the
 * Product or its content without express prior written consent of Tetcos is
 * prohibited. Ownership and / or any other right relating to the software and all
 * intellectual property rights therein shall remain at all times with Tetcos.
 *
 * Author: Shashi Kant Suman
 */

#ifndef _NETSIM_IEEE802_11_H_
#define _NETSIM_IEEE802_11_H_
#ifndef __cplusplus
double fn_netsim_matlab_init();
double fn_netsim_matlab_run(char *arr);
double fn_netsim_matlab_finish();
#endif
extern "C" {
#endif
#ifndef _NO_DEFAULT_LINKER
//For MSVC compiler. For GCC link via Linker command
#endif
100 %

```

**Figure 10-27:** Modify the Function definition of fn\_netsim\_matlab\_run() function in IEEE802\_11.h file

- Right Click on IEEE802\_11 project and select Rebuild will create a new libIEEE802.11. in the bin folder of NetSim's current workspace path <C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit.
- Open NetSim in Administrative mode. Create a Network scenario involving IEEE802\_11 say MANET, right click on the environment and select properties. Make sure that the Channel Characteristics is set to PathLoss and Fading and Shadowing.
- You will find that once the Simulation is run MATLAB Command Window starts and once the Simulation is over a histogram is displayed in MATLAB for the values that were generated using weibull distribution.



**Figure 10-28:** Histogram plot is displayed in MATLAB

- The graph and the MATLAB windows get closed once you press any key.

You can also debug the code to understand the communication between NetSim and MATLAB as explained in the DEBUGGING section above.

## 10.3 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 with new lines.

- If users don't 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.

- Open bin folder of NetSim's current workspace path (<C:\Users\PC\Documents\NetSim\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit) which contains tail.exe
- Open command window and change the directory to bin path of NetSim's current workspace path (bin\bin\_x86 for 32-bit and bin\bin\_x64 for 64-bit)
- Type the following command to open ospf\_log.txt file and press enter.

```
tail -n 0 "C:\Users\PC\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f
```

**Note: Users need to change the path of the file. In this example we are using ospf\_log.txt file**

```
C:\Windows\System32\cmd.exe - tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\soniya\Desktop\UnxUpdates>tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f
```

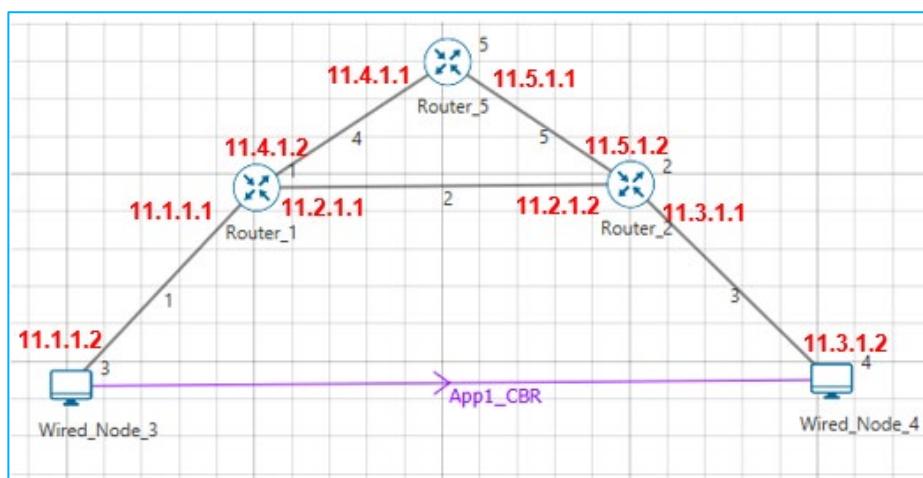
**Figure 10-29:** Enter the ospf\_log.txt file path in Command Prompt

- Open solution file and add the following line in fn\_NetSim OSPF\_Init() function in ospf.c file present inside OSPF project

```
OSPF.c ②
OSPF (Global Scope) fn_NetSim OSPF_Init(struct stru_NetSim_Network *NETWORK_Formal, NetSim_EVENTDETAILS *pstrEventDetails_Formal, char *pszAppPath_Formal, char *pszWritePath_Formal, int nVersion_Type, void **fnPointer)
{
    _getch();
    register_ospf_log();
    return fn_NetSim OSPF_Init_F(NETWORK_Formal, pstrEventDetails_Formal, pszAppPath_Formal, pszWritePath_Formal, nVersion_Type, fnPointer);
}
```

**Figure 10-30:** 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\_13.0.14\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_13.0.14\_32\_std\_default\bin\bin\_x86> for 32-bit.
- Create a scenario in NetSim as per the screenshot below and run simulation.

**Figure 10-31:** Network Topology

- In the console window user would get a warning message shown in the below screenshot **Figure 10-32** (because of changed DLL) and then the simulation will pause for user input (because of `_getch()` added in the init function)

```
C:\Program Files\NetSim Standard_11.0.21\bin\NetSimCore.exe

Today's date is "Dec 05 2018.17:59:05"
Binary build date is "Oct 2 2018.07:04:07"

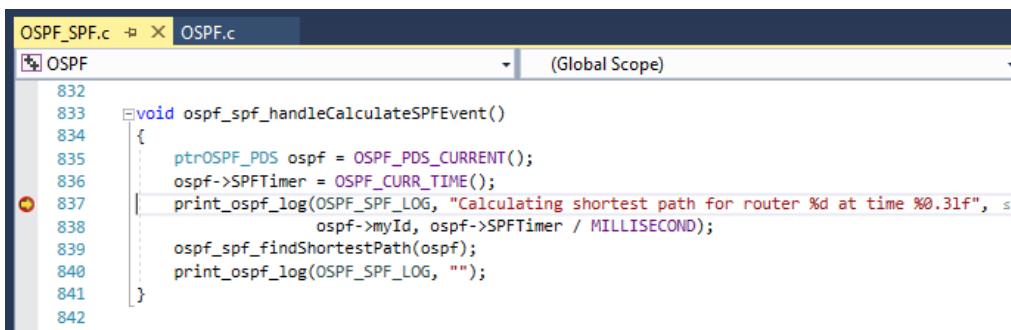
-apppath is not defined...
Setting current path as app path.
App path = C:\Program Files\NetSim Standard_11.0.21\bin

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>std>1>x>rim_hu>1111111111>0000>
NetSim license validated
Installing heart-beat...
Heartbeat status = 0 (0 indicates successful)
[33m[1m
*****
WARNING:
Detected a change in following:
libOSPF.dll
This message is normally shown if users link their own code to NetSim.
*****
Press any key to continue....
```

**Figure 10-32:** Modified Project DLL Warning Message in NetSim Console

- In Visual Studio, put break point inside all the functions in OSPF\_SPF.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 10-33**.



**Figure 10-33:** Control goes to the project and stops at the break point in the source code.

- Press F5 and check the tail console to watch the ospf\_SPF log would look like the following screenshot **Figure 10-34** which calculates the shortest path for Router2.

```
C:\ Select Command Prompt - tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f  
C:\Users\soniya\Desktop\UnxUpdates>tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f  
tail: C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt: file truncated  
Calculating shortest path for router 2 at time 7.154
```

**Figure 10-34:** Calculates the shortest path for Router 2 in Console

- Keep pressing F5 will add the ospf\_SPF log to the tail console. The below screenshot Figure 10-35 examines the WAN links connected to Router2 i.e., 11.2.1.2 and 11.5.1.2

```
OK Select Command Prompt - tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f
C:\Users\soniya\Desktop\UnxUpdates>tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f
tail: C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt: file truncated
Calculating shortest path for router 2 at time 7.154
Calculating shortest path for area 0.0.0.0
Examine LSA of type ROUTER_LSA, Id 11.2.1.2, seq num 0
    Looking for vertex 11.2.1.2 from router LSA
    Examine link 11.2.1.2, link type Stub
    Examine link 11.5.1.2, link type Stub
Candidate list for router 2
    size = 0
11.2.1.2 is my address. Not processing this link
11.5.1.2 is my address. Not processing this link
Shortest path list for router 2, area 0.0.0.0
    size = 1
    Vertex Id = 11.2.1.2
    Vertex Type = VERTEX_ROUTER
    Metric = 0
```

**Figure 10-35:** WAN links connected to Router 2 i.e., 11.2.1.2 and 11.5.1.2

- In the above screenshot, the shortest path for Router2 is 11.2.1.2 with Metrics 0 since it is one of the Router2's interface.
- The below screenshot **Figure 10-36** calculates the shortest path for Router5 and examines the WAN links connected to Router5 i.e., 11.4.1.1 and 11.5.1.1

```
OK Select Command Prompt - tail -n 0 "C:\Users\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" -f
11.5.1.2 is my address. Not processing this link
Shortest path list for router 2, area 0.0.0.0
    size = 1
    Vertex Id = 11.2.1.2
    Vertex Type = VERTEX_ROUTER
    Metric = 0

Calculating shortest path for router 5 at time 25.087
Calculating shortest path for area 0.0.0.0
Examine LSA of type ROUTER_LSA, Id 11.4.1.1, seq num 0
    Looking for vertex 11.4.1.1 from router LSA
    Examine link 11.4.1.1, link type Stub
    Examine link 11.5.1.1, link type Stub
Candidate list for router 5
    size = 0
11.4.1.1 is my address. Not processing this link
11.5.1.1 is my address. Not processing this link
Shortest path list for router 5, area 0.0.0.0
    size = 1
    Vertex Id = 11.4.1.1
    Vertex Type = VERTEX_ROUTER
    Metric = 0
```

**Figure 10-36:** WAN links connected to Router 5 i.e., 11.4.1.1 and 11.5.1.1

- In the above screenshot **Figure 10-36**, the shortest path for Router5 is 11.4.1.1 with Metrics 0 since it is one of the Router5's interface.
- The below screenshot **Figure 10-37** calculates the shortest path for Router1 and examines the WAN links connected to Router1 i.e., 11.2.1.1 and 11.4.1.2

```

Calculating shortest path for router 1 at time 25.936
Calculating shortest path for area 0.0.0.0
Examine LSA of type ROUTER_LSA, Id 11.4.1.2, seq num 0
    Looking for vertex 11.4.1.2 from router LSA
    Examine link 11.2.1.1, link type Stub
    Examine link 11.4.1.2, link type Stub
Candidate list for router 1
    size = 0
11.2.1.1 is my address. Not processing this link
11.4.1.2 is my address. Not processing this link
Shortest path list for router 1, area 0.0.0.0
    size = 1
        Vertex Id = 11.4.1.2
        Vertex Type = VERTEX_ROUTER
        Metric = 0

```

**Figure 10-37:** WAN links connected to Router1 i.e., 11.2.1.1 and 11.4.1.2

- In the above screenshot, the shortest path for Router1 is 11.4.1.2 with Metrics 0 since it is one of the Router5's interface.
- As shown in the below screenshot, the router1 calculates another new entry i.e. 11.4.1.1 with metrics 100 since it is the next hop (Router5's 1<sup>st</sup> interface) connected to Router1

```

shortest path list for router 1, area 0.0.0.0
    size = 1
        Vertex Id = 11.4.1.2
        Vertex Type = VERTEX_ROUTER
        Metric = 0

Calculating shortest path for router 1 at time 10046.348
Calculating shortest path for area 0.0.0.0
Examine LSA of type ROUTER_LSA, Id 11.4.1.2, seq num 1
    Looking for vertex 11.4.1.2 from router LSA
    Examine link 11.2.1.1, link type Stub
    Examine link 11.4.1.1, link type Point_to_Point

        Possible new candidate is 11.4.1.1
Parent = 11.4.1.2, Vertex = 11.4.1.1
Parent is ROOT
    Inserting new vertex 11.4.1.1

        Examine link 11.4.1.2, link type Stub
Candidate list for router 1
    size = 1
        Vertex Id = 11.4.1.1
        Vertex Type = VERTEX_ROUTER
        Metric = 100
        Next Hop[0] = 11.4.1.1

```

**Figure 10-38:** Router1 calculates another new entry

- 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 **Section 1**.

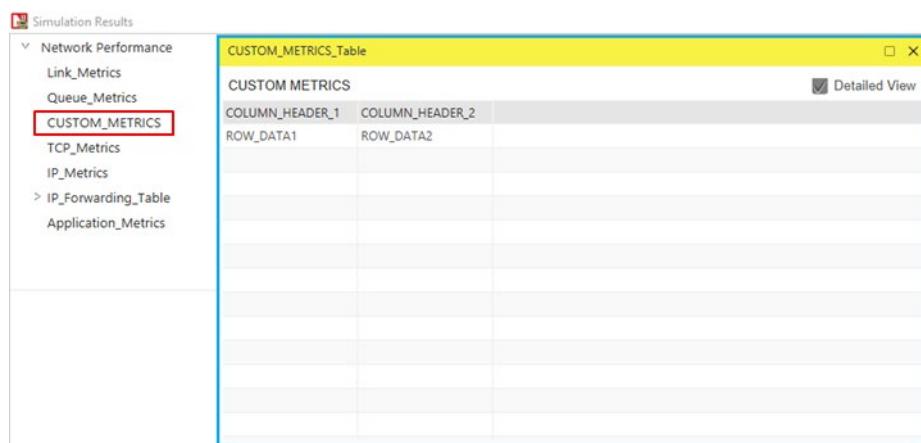
## 10.4 Adding Custom Performance Metrics

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 10-39:** Added Custom metrics table to the Simulation Results 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 8.1 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.

```
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  
        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] = IdEventTime;
}

return isChange;
}

```

**Step 5:**

Build DLL with the modified code and run a Wireless Sensor Network scenario. 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 10-40**.

The screenshot shows the NetSim Simulation Results window. On the left, there is a tree view with nodes like Network Performance, Link\_Metrics, Queue\_Metrics, TCP\_Metrics, IP\_Metrics, IP\_Forwarding\_Table, UDP Metrics, DSR Metrics, CUSTOM\_METRICS (which is selected and highlighted with a red border), and IEEE802.15.4\_Metrics. To the right of this tree view is a table titled "CUSTOM\_METRICS\_Table" with the header "NODE\_FAILURE\_METRICS". The table contains 13 rows, each representing a wireless sensor with its name, status, and residual energy. The data is as follows:

Node Name	Status(ON/OFF)	Residual_Energy (mJ)
WIRELESS_SENSOR_1	ON	5432.34
WIRELESS_SENSOR_2	ON	6479.94
WIRELESS_SENSOR_3	ON	6479.94
WIRELESS_SENSOR_4	ON	6479.94
WIRELESS_SENSOR_5	ON	6479.94
WIRELESS_SENSOR_6	ON	6479.94
WIRELESS_SENSOR_7	ON	6479.94
WIRELESS_SENSOR_8	ON	6479.94
WIRELESS_SENSOR_9	ON	6479.94
WIRELESS_SENSOR_10	ON	6479.94
WIRELESS_SENSOR_11	ON	6479.94
WIRELESS_SENSOR_12	ON	6479.94
WIRELESS_SENSOR_13	ON	6479.94

**Figure 10-40:** Added Custom metrics table to the Simulation Results window

## 10.5 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.

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. Therefore, 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

## 10.6 Adding Custom Plots

NetSim's plot option can be used to obtain Link and application throughput plots, which can be accessed from the results dashboard after the simulation. In addition, TCP Congestion Window plots and Buffer occupancy plots can be obtained by enabling the respective option in the device properties.

Users can also log additional parameters with respect to time and get them plotted in NetSim results dashboard. Following are some of the API's which are part of NetSim\_Plots.h file that can be used for this purpose:

- `fn_NetSim_Install_Metrics_Plot()` - This function creates a plot log file and returns a value of type PNETSIMPLOT which can be stored in a pointer and be used for adding values using `add_plot_data_formatted()`. This function can generally be called at simulation start. This function can be called one time for each plot that is to be generated.

For Eg: If a plot is to be generated for each node. Then this function needs to be called the number of device times.

- `add_plot_data_formatted()` - This function can be used to add values to the plot log created using the call to `fn_NetSim_Install_Metrics_Plot()`. This function needs to be called each time you want to add new values to the plot log file. The call should be made at appropriate section of code where the value being plotted changes with time.

### 10.6.1 Plotting SNR for each UE-gNB pair in 5G NR

SNR measured by UE's from each gNB can be logged and plotted as part of NetSim results window without having to use additional tools. Following is one such example where we log the SNR for each UE-gNB pair and obtain plots at the end of the simulation.

**Step 1:** Open NetSim source code in NetSim current workspace. For more information, please refer section “3.12 How does a user open and modify source codes”.

**Step 2:** Go to LTE\_NR project through the solution explorer and open the **LTE\_NR.c** file. In the function `fn_NetSim_LTE_NR_Init()`, modify code as shown below:

```
_declspec(dllexport) int fn_NetSim_LTE_NR_Init()
{
    //custom plot
    int ret = fn_NetSim_LTE_NR_Init_F();
    for (NETSIM_ID r = 0; r < NETWORK->nDeviceCount; r++)

```

```

{
    for (NETSIM_ID rin = 0; rin < DEVICE(r + 1)->nNumOflInterface; rin++)
    {
        if (!isLTE_NRInterface(r + 1, rin + 1))
            continue;

        ptrLTENR_PROTODATA data = LTENR_PROTODATA_GET(r + 1, rin + 1);
        switch (data->deviceType)
        {
            case LTENR_DEVICETYPE_UE:
            {
                for (NETSIM_ID r1 = 0; r1 < NETWORK->nDeviceCount; r1++)
                {
                    for (NETSIM_ID rin1 = 0; rin1 < DEVICE(r1 + 1)->nNumOflInterface; rin1++)
                    {
                        if (!isLTE_NRInterface(r1 + 1, rin1 + 1))
                            continue;

                        ptrLTENR_PROTODATA data = LTENR_PROTODATA_GET(r1 + 1, rin1 + 1);
                        switch (data->deviceType)
                        {
                            case LTENR_DEVICETYPE_GNB:
                            {
                                char heading[BUFSIZ], plotname[BUFSIZ];
                                sprintf(heading, "UE_%d_GNB_%d_SNR", r + 1, r1 + 1);
                                sprintf(plotname, "plot UE_%d_GNB_%d_SNR", r + 1, r1 + 1);
                                fn_NetSim_Install_Metrics_Plot(Plot_Custom, "LTE_NR SNR Plot", heading,
                                "SNR(dB)", 1, plotname);
                            }
                            break;
                        default:
                            break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}

break;

default:
    break;
}

break;
}

}

return ret;
//custom plot
}

```

**Step 3:** In the file LTENR\_GNBRRCC.c go to the function

fn\_NetSim\_LTENR\_RRC\_GENERATE\_UE\_MEASUREMENT\_REPORT() and add the lines of code highlighted in red as shown below:

```

void fn_NetSIM_LTENR_RRC_GENERATE_UE_MEASUREMENT_REPORT()

{
    NETSIM_ID d = pstruEventDetails->nDeviceId;

    NETSIM_ID in = pstruEventDetails->nInterfaceId;

    ptrLTENR_UERRC ueRRC = LTENR_UERRC_GET(d, in);

    ptrLTENR_GNBRRCC gnbRRC = LTENR_GNBRRCC_GET(ueRRC->SelectedCellID, ueRRC->SelectedCellIF);

    ptrLTENR_RRC_UE_MEASUREMENT_REPORT report = NULL;

    ptrLTENR_RRC_UE_MEASUREMENT_REPORT temp = NULL;

    ptrLTENR_GNBPHY phy = NULL;

```

```
for (NETSIM_ID r = 0; r < NETWORK->nDeviceCount; r++)  
  
{  
  
    for (NETSIM_ID rin = 0; rin < DEVICE(r + 1)->nNumOfInterface; rin++)  
  
    {  
  
        if (!isLTE_NRInterface(r + 1, rin + 1))  
  
            continue;  
  
        ptrLTENR_PROTODATA data = LTENR_PROTODATA_GET(r + 1, rin + 1);  
  
        switch (data->deviceType)  
  
        {  
  
            case LTENR_DEVICETYPE_GNB:  
  
                temp = MEASUREMENT_REPORT_ALLOC();  
  
                temp->ueID = d;  
  
                temp->cellID = r + 1;  
  
                temp->cellIF = rin + 1;  
  
                temp->rs_type = RS_TYPE_SSB;  
  
                temp->reportAmount = ReportAmount_r1;  
  
                temp->reportInteval = gnbRRC->ueMeasReportInterval;  
  
                phy = LTENR_GNBPHY_GET(r + 1, rin + 1);  
  
                temp->sinr = LTENR_PHY_RRC_RSRP_SINR(r + 1, rin + 1, d, in);  
  
                //custom plot  
        }  
    }  
}
```

```

char plotname[BUFSIZ];

sprintf(plotname, "%s\\plot_UE_%d_GNB_%d_SNR.txt",pszIOPath, d, r + 1);

FILE* fp = fopen(plotname, "a+");

if (fp)

{

    fprintf(fp, "%lf,%lf\n",pstruEventDetails->dEventTime,temp->sinr);

    fclose(fp);

}

//custom plot

LIST_ADD_LAST((void**)&report, temp);

break;

default:

break;

}

}

ptrLTENR_RRC_Hdr hdr = calloc(1, sizeof * hdr);

hdr->msg = report;

hdr->msgType = LTENR_MSG_RRC_UE_MEASUREMENT_REPORT;

hdr->SenderID = d;

```

```
hdr->SenderIF = in;
```

```
fn_NetSim_LTENR_RRC_ADD_HDR_INTO_PACKET(pstruEventDetails->pPacket, hdr,
ueMEASID, LTENR_MSG_RRC_UE_MEASUREMENT_REPORT);
```

```
LTENR_CallPDCPOut();
```

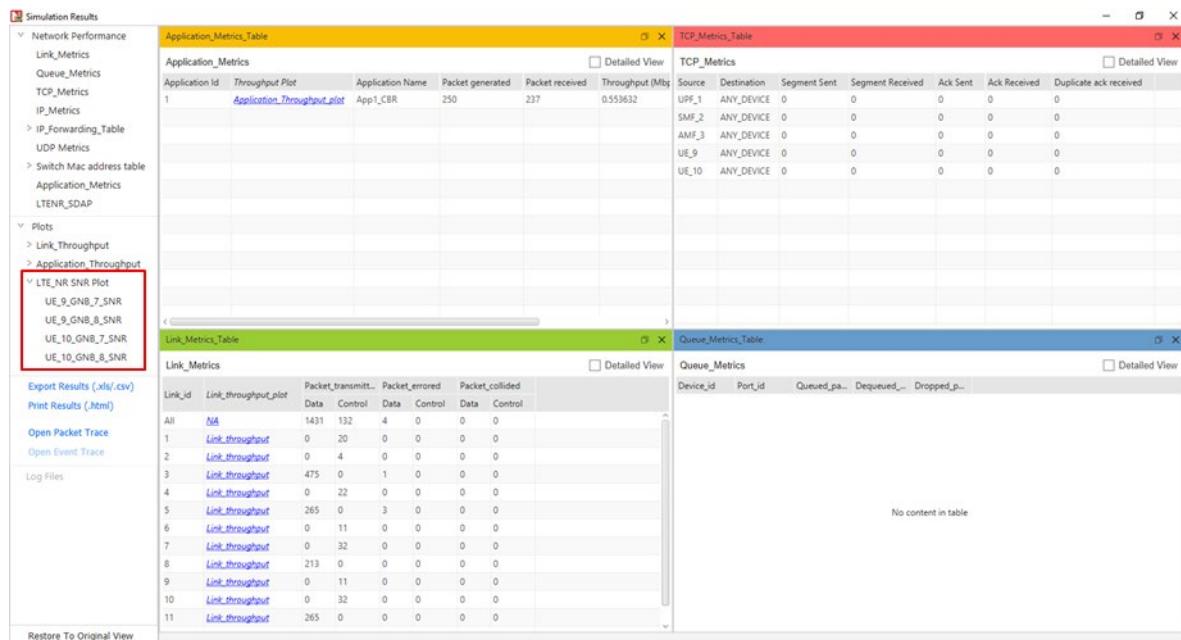
```
}
```

**Step 4:** In **LTENR.c** add the following lines `#include "NetSim_Plot.h"`.

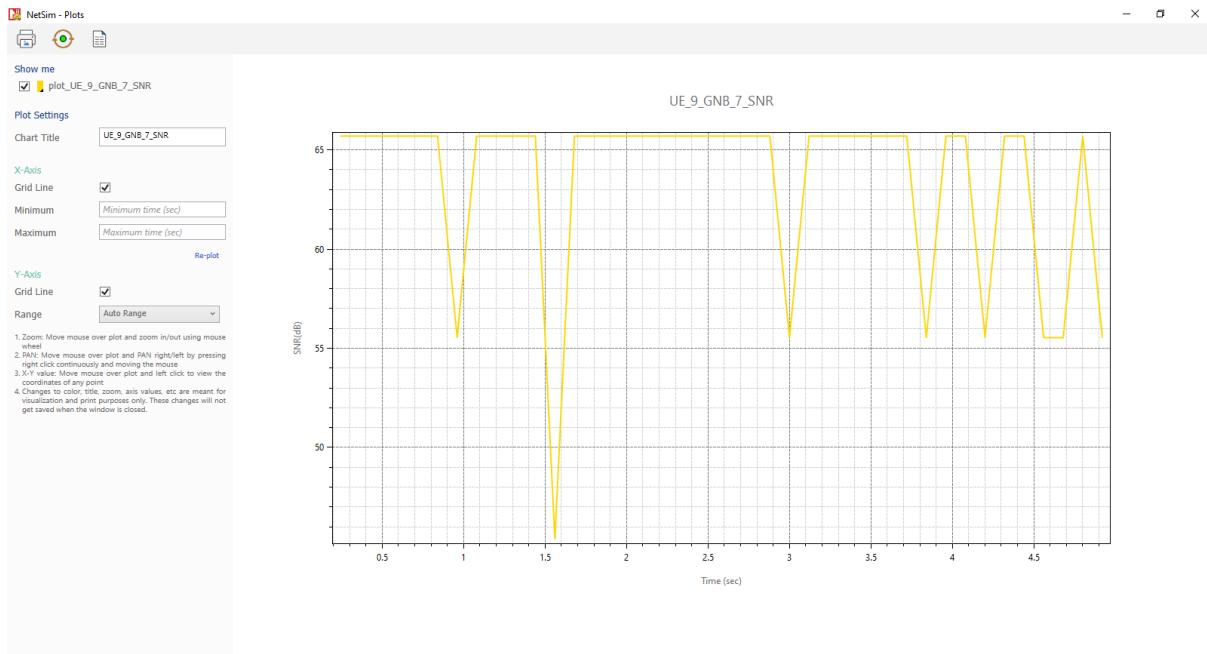
**Step 5:** Save the changes and right-click on the **LTE\_NR** module in the solution explorer and select Rebuild.

**Step 6:** Upon a successful build, NetSim will automatically update the modified binaries in the respective binary folder.

**Step 7:** Now on running any simulation in LTE/5G NR networks, you will get individual SNR plots for each UE-GNB/UE-ENB pair, in the NetSim Metrics window under Plots ->LTE\_NR SNR Plot shown below **Figure 10-41**.



**Figure 10-41:** LTE\_NR SNR Plots in Result Window



**Figure 10-42:** SNR Plot

The above results are based on the Handover in 5G NR Experiment which is part of NetSim v13.0 experiment manual. The plot shows how the SNR drops as UE 3 moves away from GNB 1.

## 10.7 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. Equivalent to -d command in CLI mode.

9. NETSIM\_AUTO = <int> // If set NetSim will not ask for keypress after simulation. //Useful to run batch simulations.
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.  
In application metrics, you can see duplicate packets received.
13. NETSIM\_CONFIG\_NAME = <FILE NAME> // Config file name. This file must present in IOPath. If not set default value is Configuration.netsim
14. NETSIM\_NEG\_ID = 1 // If set, then control packets will have negative id.
15. NETSIM\_PACKET\_DBG = 1 // If set, then Simulation engine will log the packet creation and freeing
16. NETSIM\_MEMCHECK = 1 // If set, then simulation will enable memory check.
17. NETSIM\_MEMCHECK\_1 = x // Lower event id
18. NETSIM\_MEMCHECK\_2 = x // Upper event id

## 10.8 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 - Animation, Plots, Traces and Logs
- External Interfacing – MATLAB, Wireshark, SUMO

NetSim GUI limitation on total number of Nodes is as follows:

- NetSim Academic – 100

- NetSim Standard – 500
- NetSim Professional – No software limit

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 Animation – NetSim writes one file per node and windows limits the number of simultaneously opened files to 512.
- Disable plots, traces and logs to speed up the simulation.
- If plots are enabled NetSim writes one file for each link and for each application. Therefore, it is recommended users only select those links/applications for which they wish to plot output performance. Plots for all other applications/links should be disabled.
- 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.

## 10.9 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:

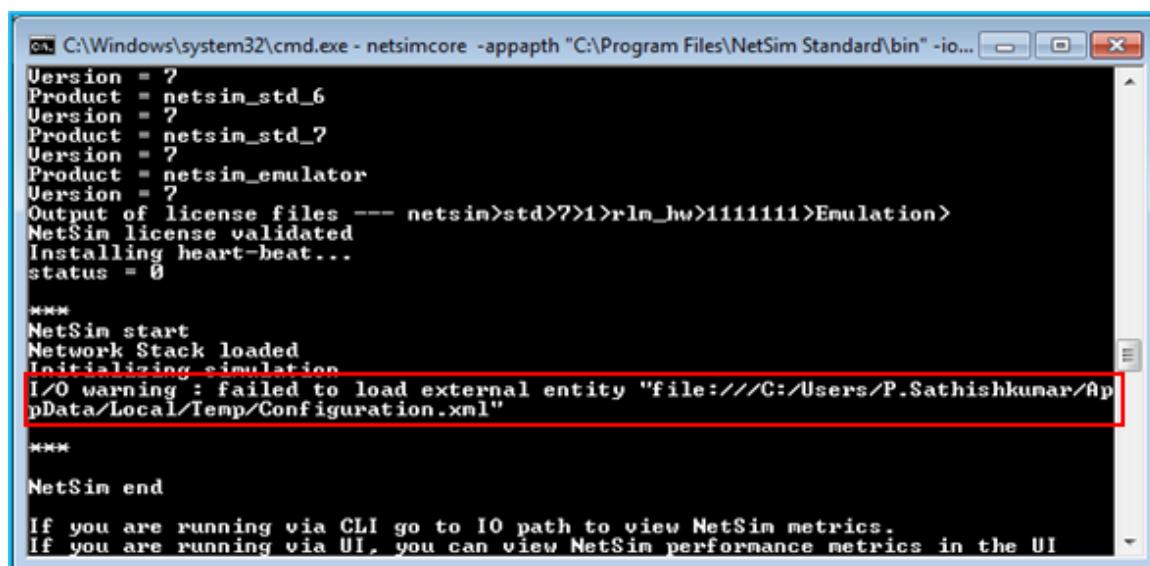
## 12.1 CLI mode

While running NetSim via CLI for the scenarios described in the Configuration file, you may bump into few problems.

Note that while running NetSim via CLI, try to ensure that there are no errors in the Configuration.netsim file. The file, ConfigLog.txt, written to the windows temp path would show errors, if any, found by NetSim's config parser.

## 12.2 I/O warning displayed in CLI mode

**Reason:** While typing the CLI command if you enter wrong I/O Path, or if there is no Configuration.netsim file then the following error is thrown



```
C:\Windows\system32\cmd.exe - netsimcore -appapth "C:\Program Files\NetSim Standard\bin" -io...
Version = ?
Product = netsim_std_6
Version = ?
Product = netsim_std_7
Version = ?
Product = netsim_emulator
Version = ?
Output of license files --- netsim>std>?>1>rlm_jw>111111>Emulation>
NetSim license validated
Installing heart-beat...
status = 0

***  

NetSim start  

Network Stack loaded  

Initializing simulation  

I/O warning : failed to load external entity "file:///C:/Users/P.Sathishkumar/AppData/Local/Temp/Configuration.xml"  

***  

NetSim end

If you are running via CLI go to IO path to view NetSim metrics.  

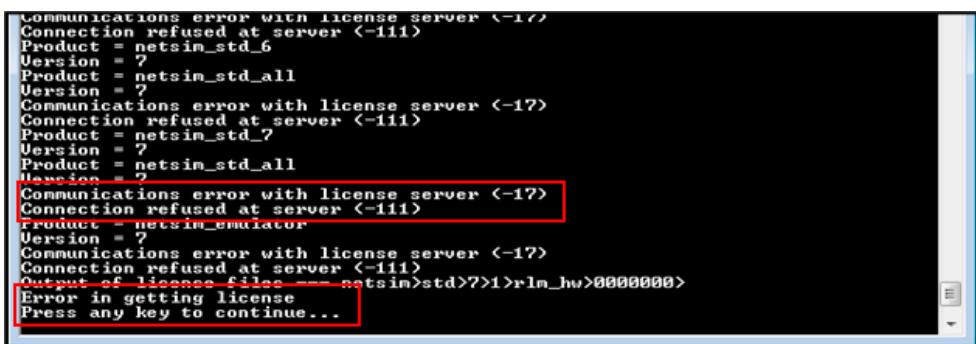
If you are running via UI, you can view NetSim performance metrics in the UI
```

Figure 12-1: I/O warning displayed in CLI mode

**Solution:** Please check the I/O path.

### 12.2.1 Connection refused at server<-111> error displayed

**Reason:** Unable to communicate with the license server



```

communications error with license server <-1>
Connection refused at server <-111>
Product = netsim_std_6
Version = ?
Product = netsim_std_all
Version = ?
Communications error with license server <-17>
Connection refused at server <-111>
Product = netsim_std_7
Version = ?
Product = netsim_std_all
Version = ?
Communications error with license server <-17>
Connection refused at server <-111>
Product = netsim_emulator
Version = ?
Communications error with license server <-17>
Connection refused at server <-111>
Output of license file --- netsim>std>?>1>rlm_hw>0000000>
Error in getting license
Press any key to continue...

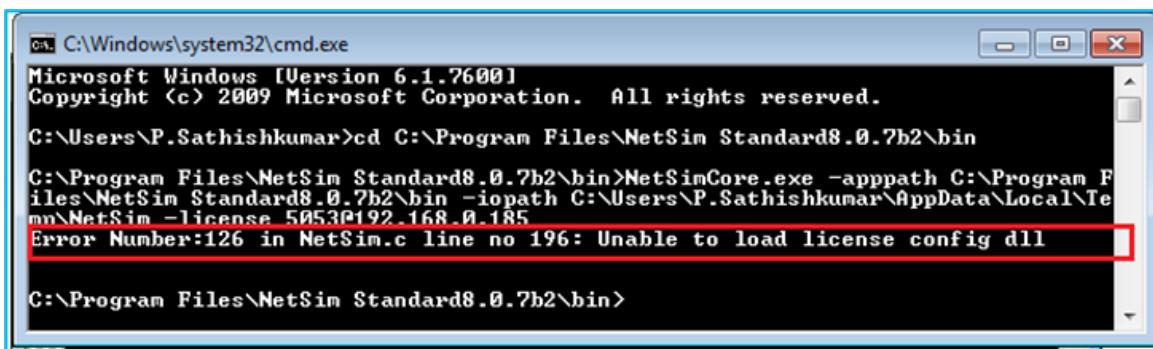
```

**Figure 12-2:** Connection refused at server<-111> error displayed

**Solution:** In this example, license server IP address is 192.168.0.185 but it is given as 192.168.0.180. Here server IP address is wrong. Same error message is shown for wrong port number, wrong tag name like—apppath,-iopath,-license. For example, if –apppath is typed instead of –apppath then this message will be shown. So, check those details.

### 12.2.2 Unable to load license config dll(126)

**Reason:** Apppath and I/O path have white spaces



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\P.Sathishkumar>cd C:\Program Files\NetSim Standard8.0.7b2\bin
C:\Program Files\NetSim Standard8.0.7b2\bin>NetSimCore.exe -apppath C:\Program F
iles\NetSim Standard8.0.7b2\bin -iopath C:\Users\P.Sathishkumar\AppData\Local\Te
mn\NetSim -license 50530192.168.0.185
Error Number:126 in NetSim.c line no 196: Unable to load license config dll
C:\Program Files\NetSim Standard8.0.7b2\bin>

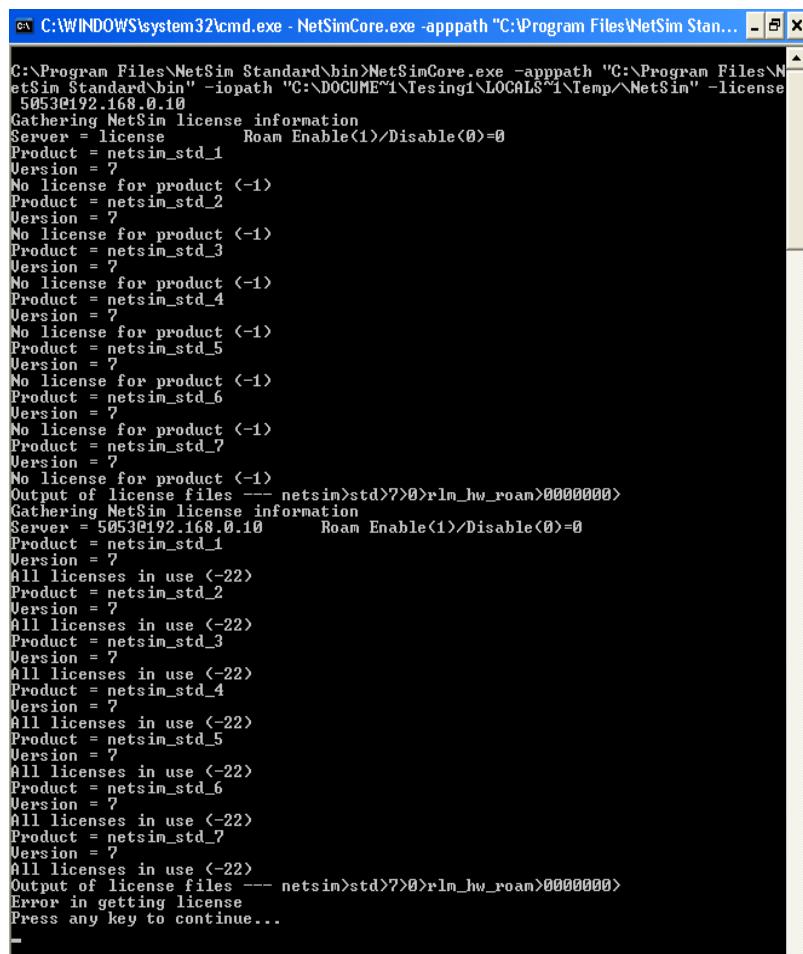
```

**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.

### 12.2.3 “Error in getting License” error in CLI mode

Simulation does not commence. “No license for product (-1)” is displayed in the command prompt.

**Example:**


```
C:\> C:\Windows\system32\cmd.exe - NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "C:\DOCUMENTS\1\Testing1\LOCALS\1\Temp\NetSim" -license 5053@192.168.0.10
Gathering NetSim license information
Server = license Roam Enable(1)/Disable(0)=0
Product = netsim_std_1
Version = ?
No license for product <-1>
Product = netsim_std_2
Version = ?
No license for product <-1>
Product = netsim_std_3
Version = ?
No license for product <-1>
Product = netsim_std_4
Version = ?
No license for product <-1>
Product = netsim_std_5
Version = ?
No license for product <-1>
Product = netsim_std_6
Version = ?
No license for product <-1>
Product = netsim_std_7
Version = ?
No license for product <-1>
Output of license files --- netsim>std>?>0>rlm_hw_roam>0000000>
Gathering NetSim license information
Server = 5053@192.168.0.10 Roam Enable(1)/Disable(0)=0
Product = netsim_std_1
Version = ?
All licenses in use <-22>
Product = netsim_std_2
Version = ?
All licenses in use <-22>
Product = netsim_std_3
Version = ?
All licenses in use <-22>
Product = netsim_std_4
Version = ?
All licenses in use <-22>
Product = netsim_std_5
Version = ?
All licenses in use <-22>
Product = netsim_std_6
Version = ?
All licenses in use <-22>
Product = netsim_std_7
Version = ?
All licenses in use <-22>
Output of license files --- netsim>std>?>0>rlm_hw_roam>0000000>
Error in getting license
Press any key to continue...
-
```

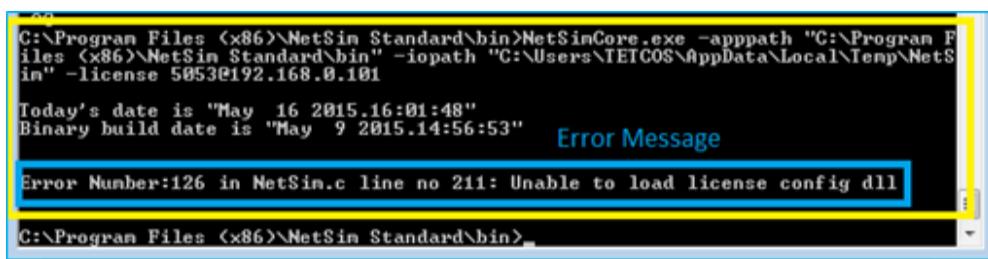
**Figure 12-4:** “Error in getting License” error in CLI mode

**Solution:** NetSim is based on the client-server architecture. When NetSim runs in the client machine, it will check for the license in the same machine, first. If license is not available in the same machine, then “No license for product (-1)” will be displayed in the command prompt and the server machine will be checked for the availability of license. If no license is available in the server machine also, then again “No license for product (-1)” will be displayed in the command prompt.

If “No license for product(-1)” is displayed in the command prompt two times, then check in the NetSim license server to know about the availability of license and adjust the number of current users of NetSim, in order to get the license.

#### 12.2.4 Unable to load license config dll displayed

**Reason:** If the command/iopath provided by the user is first written in MS Word and then copy pasted to Command prompt, some special characters (not visible in command prompt) gets inserted and on execution, license config dll is not found.



**Figure 12-5:** Unable to load license config dll

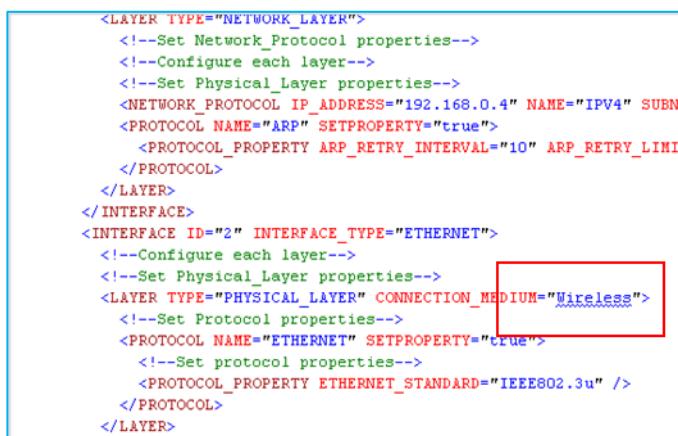
**Solution:** Type the command manually or copy paste the command/iopath from notepad.

## 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 12-6**.



**Figure 12-6:** 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.

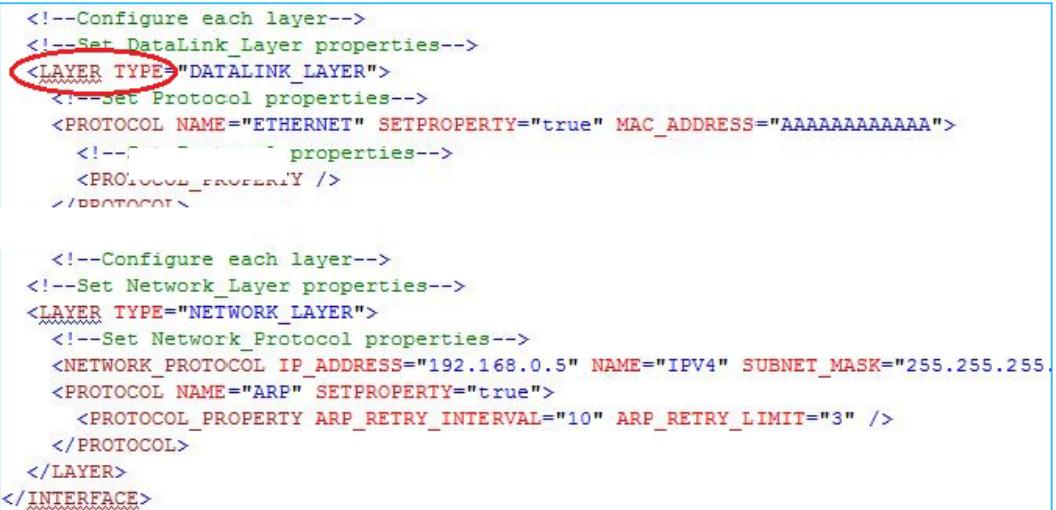
**Note:** If the schema file and the configuration file are not present in the same folder, the zigzag lines won't appear. So, place the Configuration file and Schema File in the same location or change the path of schema file in the configuration file.

### 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.



```

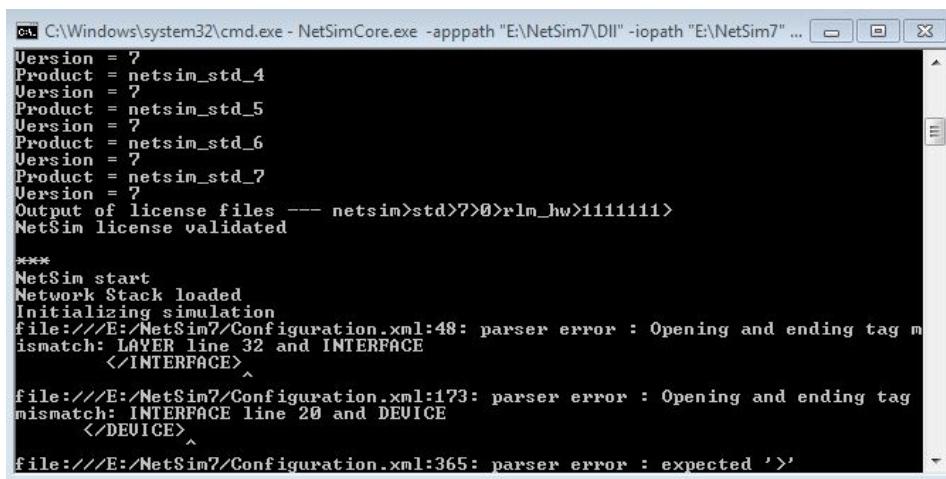
<!--Configure each layer-->
<!--Set DataLink_Layer properties-->
<LAYER TYPE="DATALINK_LAYER">
    <!--Set Protocol properties-->
    <PROTOCOL NAME="ETHERNET" SETPROPERTY="true" MAC_ADDRESS="AAAAAAAAAAAAA">
        <!--Set Protocol properties-->
        <PROTOCOL_PROPERTY />
    </PROTOCOL>
</LAYER>

<!--Configure each layer-->
<!--Set Network_Layer properties-->
<LAYER TYPE="NETWORK_LAYER">
    <!--Set Network_Protocol properties-->
    <NETWORK_PROTOCOL IP_ADDRESS="192.168.0.5" NAME="IPV4" SUBNET_MASK="255.255.255.>
        <PROTOCOL NAME="ARP" SETPROPERTY="true">
            <PROTOCOL_PROPERTY ARP_RETRY_INTERVAL="10" ARP_RETRY_LIMIT="3" />
        </PROTOCOL>
    </LAYER>
</INTERFACE>

```

**Figure 12-7:** 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 - NetSimCore.exe -apppath "E:\NetSim7\Dll" -iopath "E:\NetSim7" ...
Version = 7
Product = netsim_std_4
Version = 7
Product = netsim_std_5
Version = 7
Product = netsim_std_6
Version = 7
Product = netsim_std_7
Version = 7
Output of license files --- netsim>std>?>0>rlm_hw>1111111>
NetSim license validated

*** 
NetSim start
Network Stack loaded
Initializing simulation
file:///E:/NetSim7/Configuration.xml:48: parser error : Opening and ending tag mismatch: LAYER line 32 and INTERFACE
</INTERFACE>
file:///E:/NetSim7/Configuration.xml:173: parser error : Opening and ending tag mismatch: INTERFACE line 20 and DEVICE
</DEVICE>
file:///E:/NetSim7/Configuration.xml:365: parser error : expected '>'
```

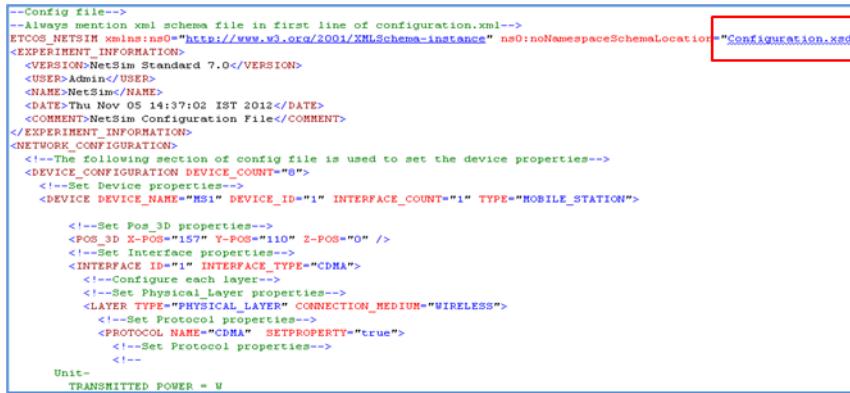
**Figure 12-8:** NetSim Run through CLI and following error displayed in the command prompt

**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.



```
--Config file-->
--Always mention xml schema file in first line of configuration.xml-->
<?xml version="1.0" encoding="UTF-8"?> <!--ns0: noNamespaceSchemaLocation="Configuration.xsd"-->
<EXPERIMENT_INFORMATION>
<VERSION>NetSim Standard 7.0</VERSION>
<USER>Admin</USER>
<NAME>NetSim</NAME>
<DATE>Thu Nov 05 14:37:02 IST 2012</DATE>
<COMMENT>NetSim Configuration File</COMMENT>
</EXPERIMENT_INFORMATION>
<NETWORK_CONFIGURATION>
<!--The following section of config file is used to set the device properties-->
<DEVICE_CONFIGURATION DEVICE_COUNT="8">
<!--Set Device properties-->
<DEVICE DEVICE_NAME="MS1" DEVICE_ID="1" INTERFACE_COUNT="1" TYPE="MOBILE_STATION">
<!--Set Pos_3D properties-->
<POS_3D X-POS="157" Y-POS="110" Z-POS="0" />
<!--Set Interface properties-->
<INTERFACE ID="1" INTERFACE_TYPE="CDMA">
<!--Configure each layer-->
<!--Set Physical_Layer properties-->
<LAYER TYPE="PHYSICAL_LAYER" CONNECTION_MEDIUM="WIRELESS">
<!--Set Protocol properties-->
<PROTOCOL NAME="CDMA" SETPROPERTY="true">
<!--Set Protocol properties-->
<!--
Unit-
TRANSMITTED POWER = u

```

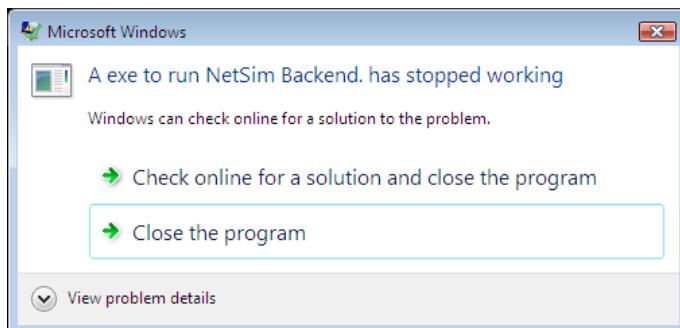
**Figure 12-9:** Error lines in configuration.xsd in the Configuration file

**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, “An exe to run NetSim backend has stopped working” is thrown.

### Example:



**Figure 12-10:** Simulation terminates and “NetSim Backend has stopped working” displayed  
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 Monitor screen resolution is less than 1024X768

While starting NetSim, error shows the monitor screen resolution is less than 1024 X 768.

**Reason:** This error will come if monitor resolution is less than 1024 X 768. For example, 1260 X 720 will also show this error.

**Solution:** Change your monitor resolution to 1024 X 768 or above.

## 12.6 Licensing

### 12.6.1 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.

## 12.7 Troubleshooting VANET simulations that interface with SUMO

### 12.7.1 Guide for Sumo

- Link for the Sumo Website - [http://www.dlr.de/ts/en/desktopdefault.aspx/tabcid-9883/16931\\_read-41000](http://www.dlr.de/ts/en/desktopdefault.aspx/tabcid-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.

## 12.7.2 Guide for Python

- Any Python 2.7 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()

```

### 12.7.3 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.

### 12.7.4 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.

### 12.7.5 NetSim Core Protocol Library

- After establishing the connection, NetSim VANET Library checks for GUI flag, and sends ‘1’ if animation status is online.
- 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.

## 13 NetSim Videos

In order to have a better understanding of NetSim, users can access YouTube channel of Tetcos at [www.youtube.com/tetcos](http://www.youtube.com/tetcos) and check out the various videos available.

## 14 R&D projects in NetSim

Example R & D projects in NetSim is available in [www.tetcos.com/file-exchange](http://www.tetcos.com/file-exchange).

## 15 NetSim FAQ/Knowledgebase

NetSim knowledgebase with hundreds on FAQs on how NetSim works is available at <https://tetcos.freshdesk.com/support/home>

List of known issues in v13.0 is available at

<https://support.tetcos.com/support/solutions/articles/14000101817-list-of>