NetSim User Manual

Contents

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 academics, industry and defense 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 educations institutions while NetSim Pro version addresses the needs of defense and industry. The standard and pro versions are available as components in NetSim v11 from which users can choose and assemble. The academic version is available as a single product and includes all the technologies shown below. A comparison of the features in the three versions are tabulated below:

Features	Academic	Standard	Pro
Technology Coverage			
Internetworks	Y	Y	Y
Legacy & CellularNetworks	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/LTE-A Networks	Y	Y	Y
VANET	N	Y	Y
Performance Reporting Performance metrics available for Network and Sub-network	Y	Y	Y
Packet Animator Used to animate the packet flow in network	Y	Y	Y
Packet Trace Available in tab ordered .txt format for easy post processing	Y	Y	Y
Event Trace Available in tab ordered .txt format for easy post processing	N	Y	Y
Protocol Library Source Codes with Documentation Protocol C source codes and appropriate header files with extensive documentation	N	Y	Y
External Interfacing Interfacing with SUMO	N	Y	Y
MATLAB	N		
Wireshark	Y		
Integrated debugging Users can write their own code, link their code to NetSim and debug using Visual Studio	N	Y	Y
Plots Allows users to plot the value of a parameter over simulation time	Y	Y	Y
Simulation Scale	100 Nodes	500 Nodes	~ 10,000 Nodes
Custom Coding and Modeling Support	N	N	Y
Emulator (Add on) Connect to real hardware running live application	N	Y	Y
Military Radio (Add On) TDMA and DTDMA	N	N	Y
Target Users and Segment	Educational (Lab Experimentation)	Educational (Research)	Commercial (Industrial and Defense)

1.3 Components (Technology Libraries) in Pro and Std versions

In NetSim v11, users can choose and assemble components (Technology libraries) for Pro and Standard version. The components are as follows:

Component No	Networks / Protocols	International Standards
Component 1 (Base. Required for all components)	**Internetworks** Ethernet - Fast & Gigabit, ARP, Routing - RIP, OSPF, WLAN - 802.11 a / b / g /p / n / ac & e, Propagation models - HATA Urban / Suburban, COST 231 HATA urban / Suburban, Indoor Home / Office / Factory, 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 **Common Modules** Traffic Generator: Voice, Video, FTP, Database, HTTP, Email, P2P, Custom, 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
Component 2	**Legacy & Cellular Networks** Aloha – (Pure & Slotted) GSM CDMA	3GPP, ETSI, IMT-MC, IS-95 A/B, IxRTT, 1x-EV-Do, 3xRTT
Component 3	**Advanced Routing** 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
Component 4	**Mobile Adhoc Networks** MANET - DSR, AODV, OLSR, ZRP Multiple MANETs	IETF RFC 4728, 3561, 3626 IEEE 802.16d
Component 5	**Software Defined Network (SDN)**	Open Flow v1.3
Component 6 (Component 4 required)	**Internet of things (IOT)** with RPL protocol Wireless Sensor Networks (WSN)	IEEE 802.15.4 MAC, MANET in L3 RFC 6550
Component 7	**Cognitive Radio Networks** WRAN	IEEE 802.22
Component 8	**Long-Term Evolution Networks:** LTE, LTE - Advanced, LTE Device to Device (LTE D2D), LTE Femto Cell, LTE VANET	3GPP
Component 9 (Component 4 required)	**VANETs:** IEEE 1609 WAVE, Basic Safety Message (BSM) protocol per J2735 DSRC, Interface with SUMO for road traffic simulation	IEEE 1609

Military Radio Add on (Pro version only)	**Military Radio (Only in PRO Version)** TDMA Link 16, Dynamic TDMA, Frequencies – HF, VHF, UHF Bands, Frequency Hopping	----
Network Emulator Add On	**Network Emulator** Connect real hardware running live applications to NetSim Simulator	----

2 Getting Started in NetSim

2.1 System Requirements

2.1.1 NetSim Client (installs locally):

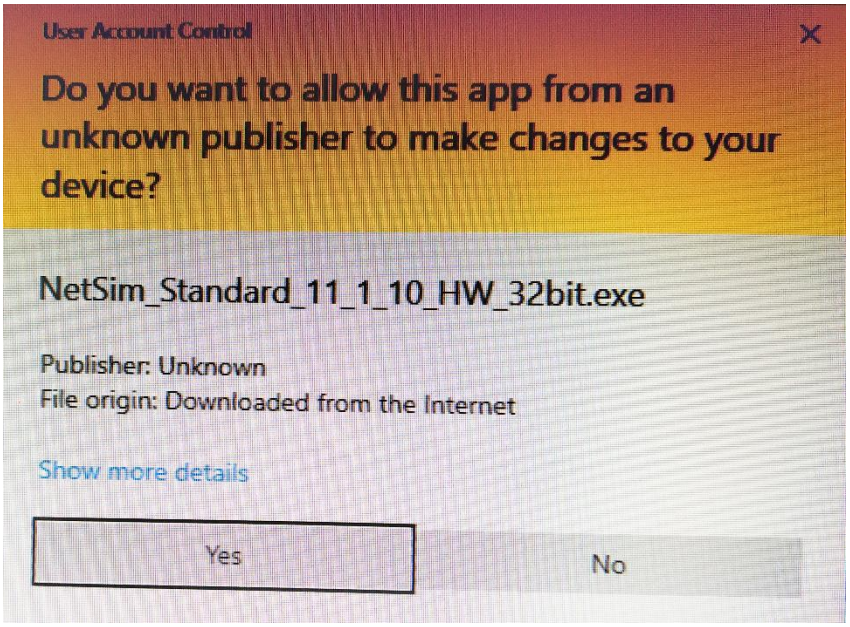
- Hardware: i3 equivalent or above, 4 GB RAM (minimum)
- Monitor resolution: Min - 1024*768, Max - 1920*1028. Optional Scale and layout setting: 100%
- Operating system: Win 7 (SP1 or higher), Win 8 or Win 10
- Software: MS Office, Adobe Reader
- Development Tools: Visual Studio
    - NetSimv8 / v8.1 / v8.3 / v9 / v9.1: Microsoft Visual Studio 2010 (or higher), Community edition (or higher for writing and debugging custom code
    - NetSim v10/ v11: Microsoft Visual Studio 2015 (or higher), Community edition (or higher) is required for writing and debugging custom code

2.1.2 License Server (for running HostID/ Dongle locked floating licenses, not applicable for node locked licenses):

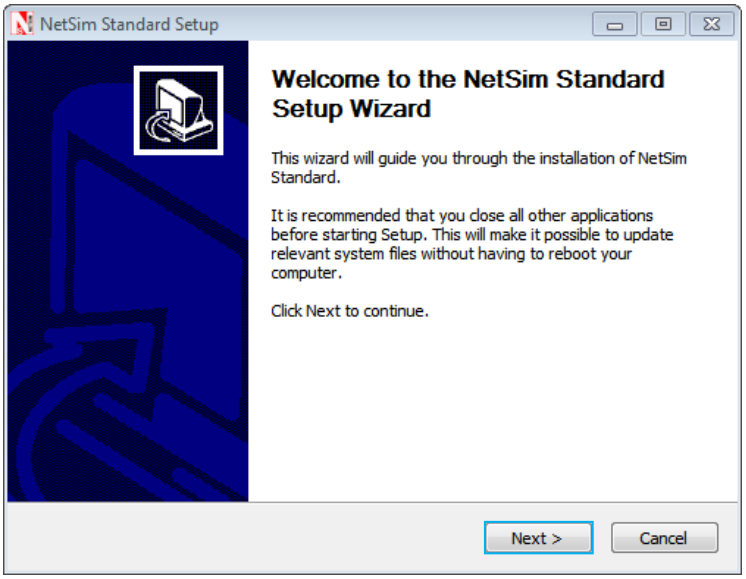
Any one system will have to be made as the license server, and it is to this PC that the license is locked, either via its MAC ID or via a dongle. The dongle is a USB device which controls the licensing. The system (hardware / OS) requirements is same as that applicable for NetSim clients. USB Port is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

2.2 Installing NetSim

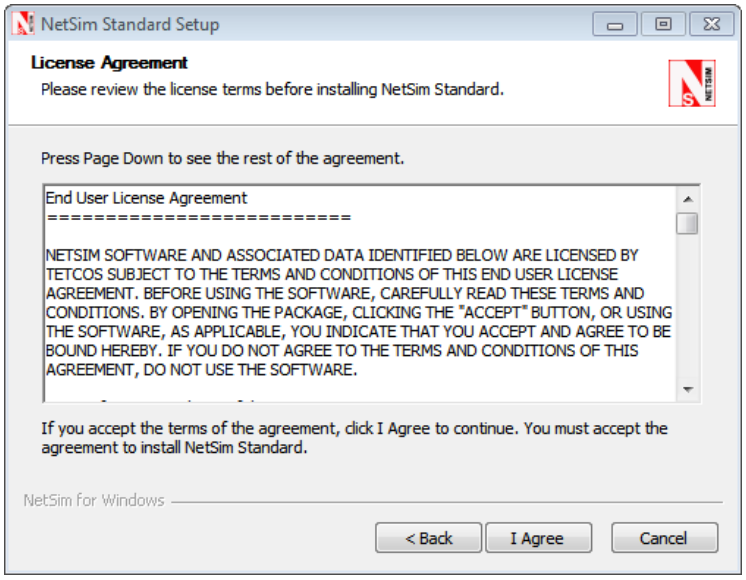
Install the 32-bit or 64-bit build of NetSim depending on your PC platform.



Based on the NetSim version under installation the version type being displayed in the following windows will change. For example, you will see NetSim Standard for a standard version install. Click on **Yes** button to install the software.



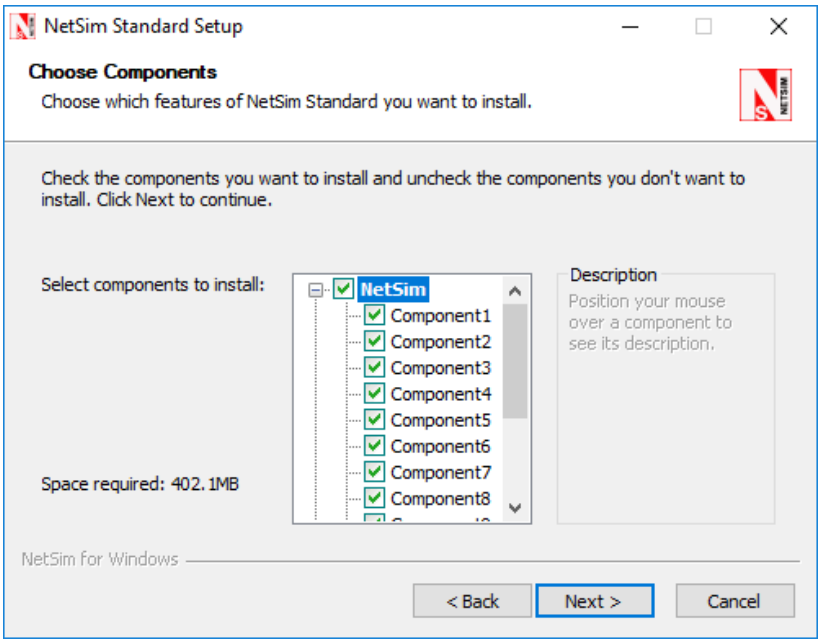
Setup prepares the installation wizard and software installation begins with a **Welcome Screen**.



Click on the **Next** button. In the next screen, License agreement will be displayed.

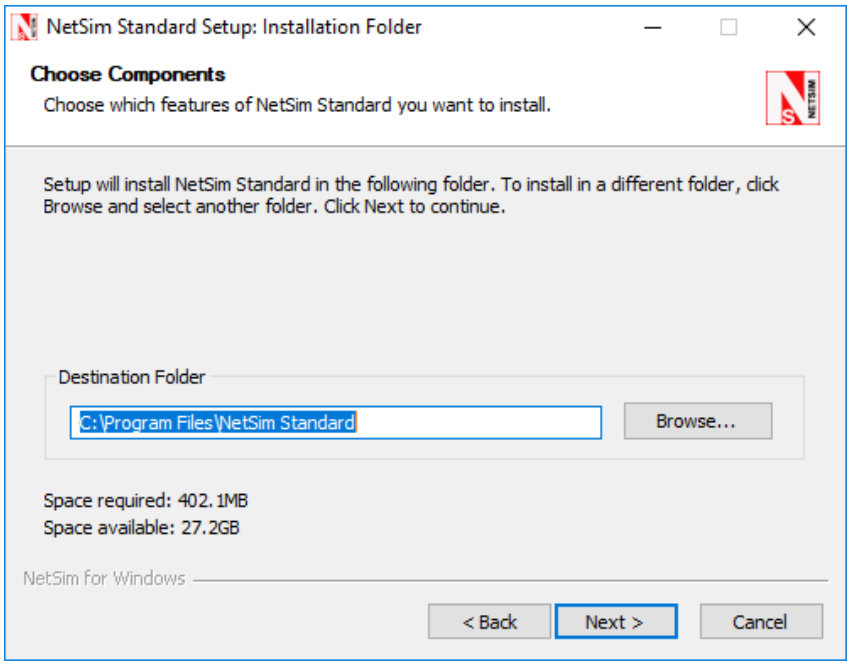
Read the agreement carefully, scroll down to read the complete license agreement. Click "**I Agree**" button else quit the setup by clicking **Cancel** button.

If you agree with the license agreement, you 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 version. Other versions of NetSim are available as a single package.

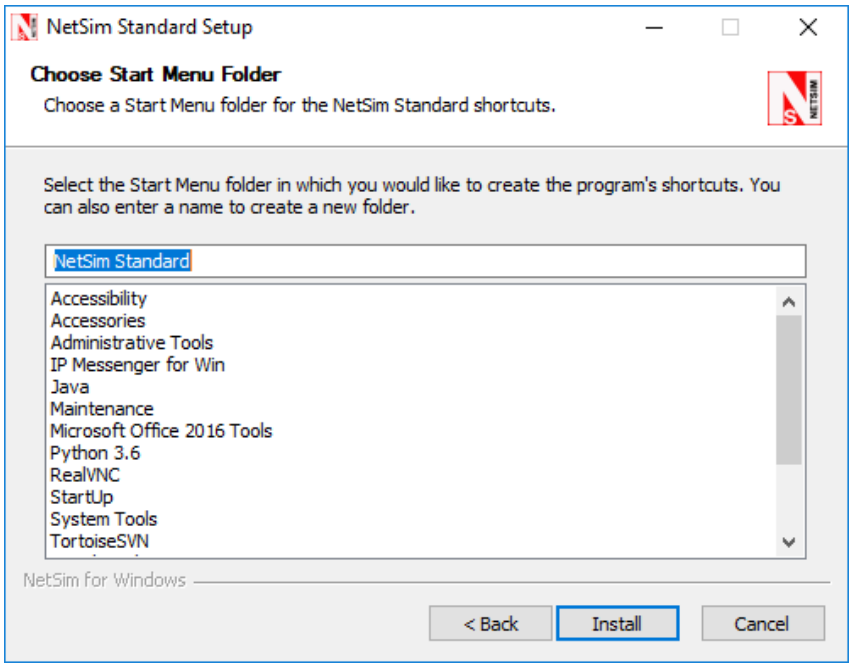


Click on the **Next** button.

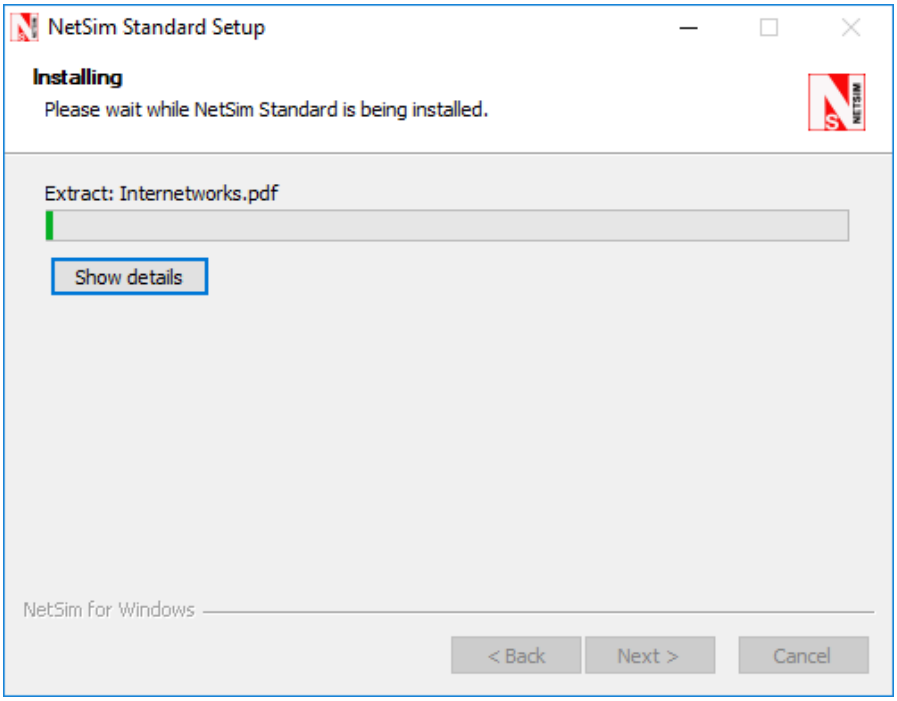
Note: Select all the supporting applications for complete installation of the software. In the next screen, you will be requested to enter the installation path. Select the path in which the software needs to be installed and click on **Next** button.



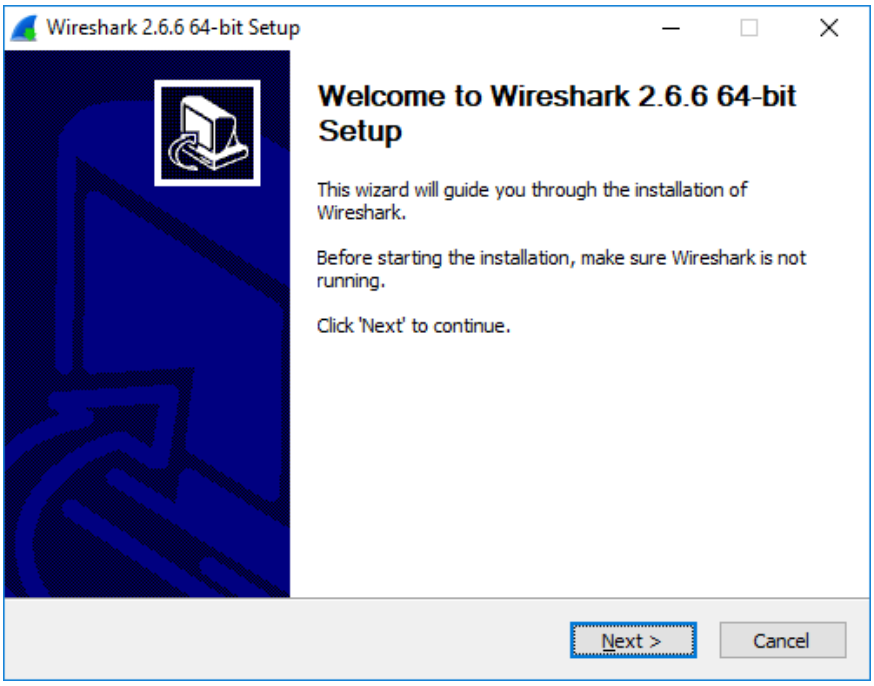
On the next screen, you will be requested to enter the Start menu folder name.



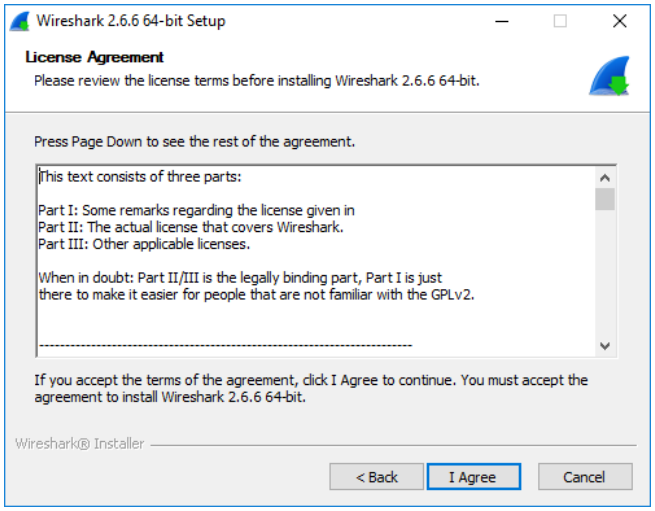
Click on the **Install** button to start the installation. The installation process begins.

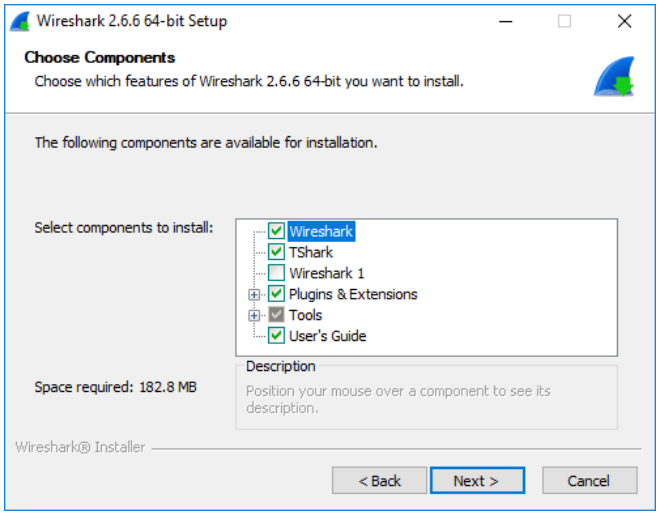


After the installation of required NetSim files, the installation of third party tools begin.

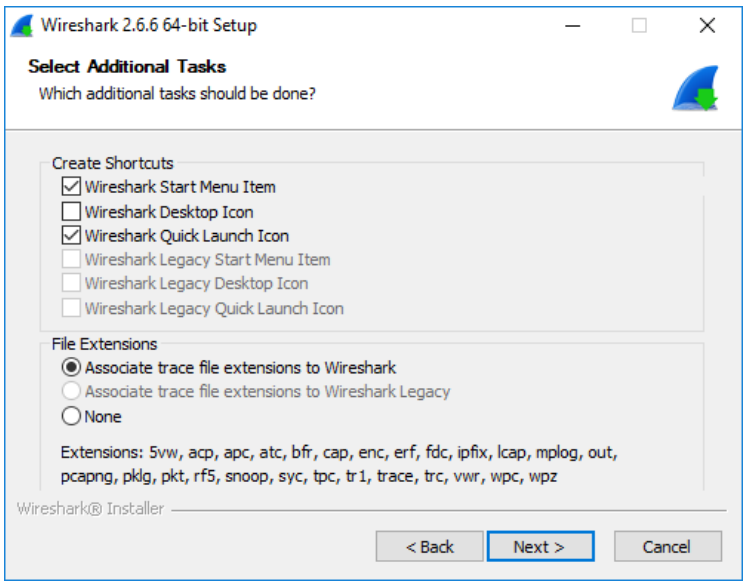


For NetSim Standard Version and Pro Version, Wireshark installation will start by default (if not deselected during 3rd party software selection). Click on Next to start Wireshark installation. In the following window, click on I Agree.

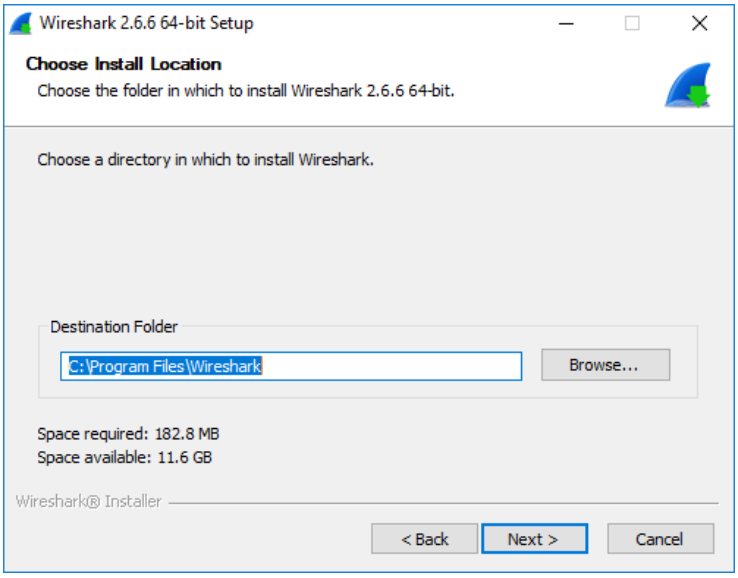




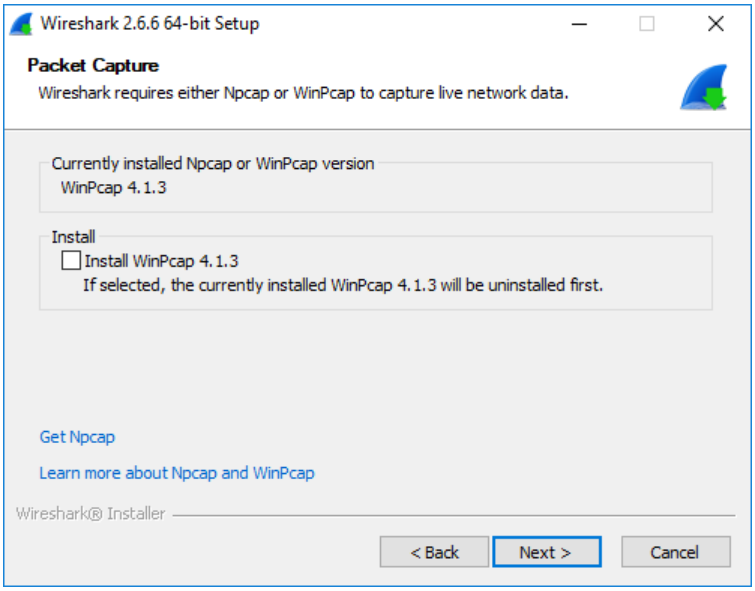
Select all the components and click on **Next.**



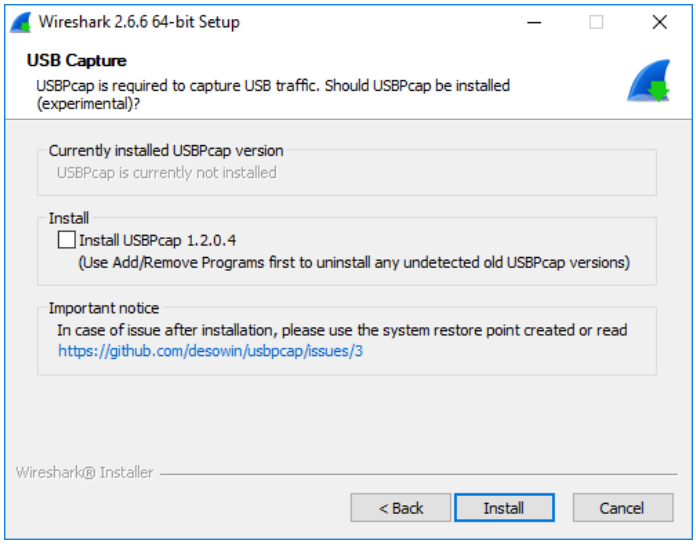
Click on Next to go to Install Location window as shown below.



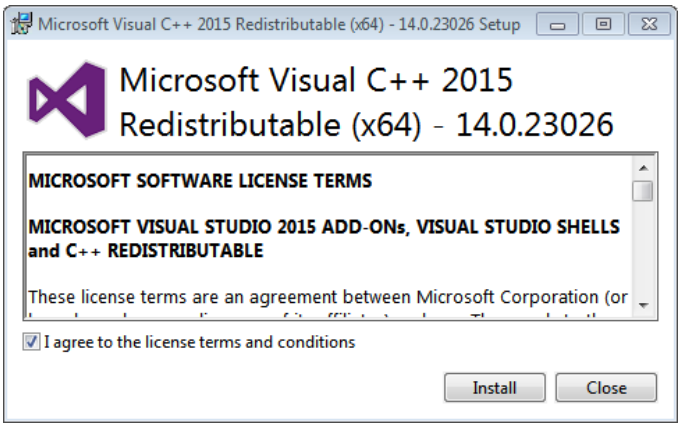
Specify the destination and click on **Next.**



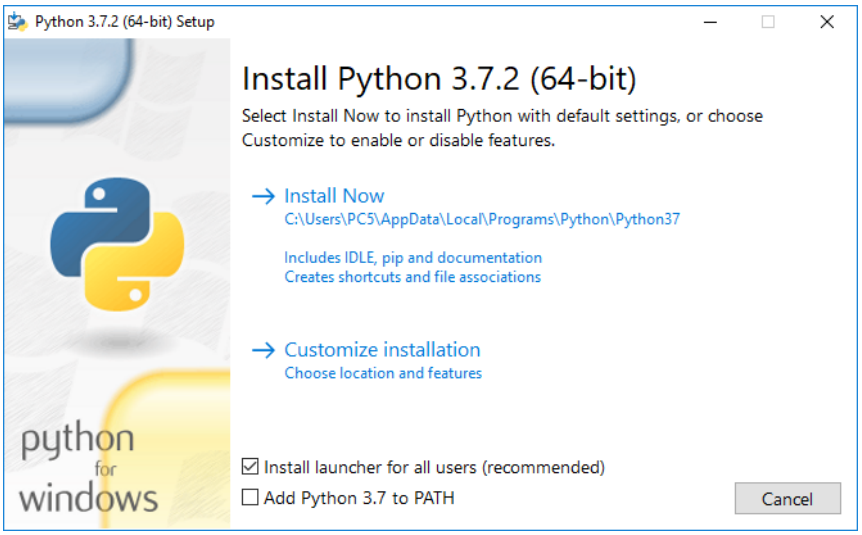
Select **Install WinPcap** and click on **Next.**



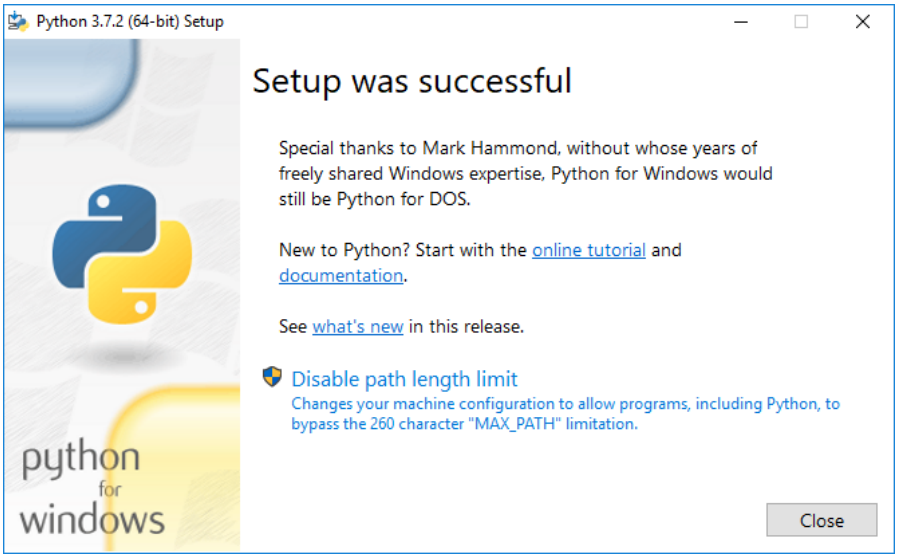
Select **Install USBPcap** and click on **Install.** After the installation of the software, you will be requested to click **Next** and then **Finish** to complete the installation process.



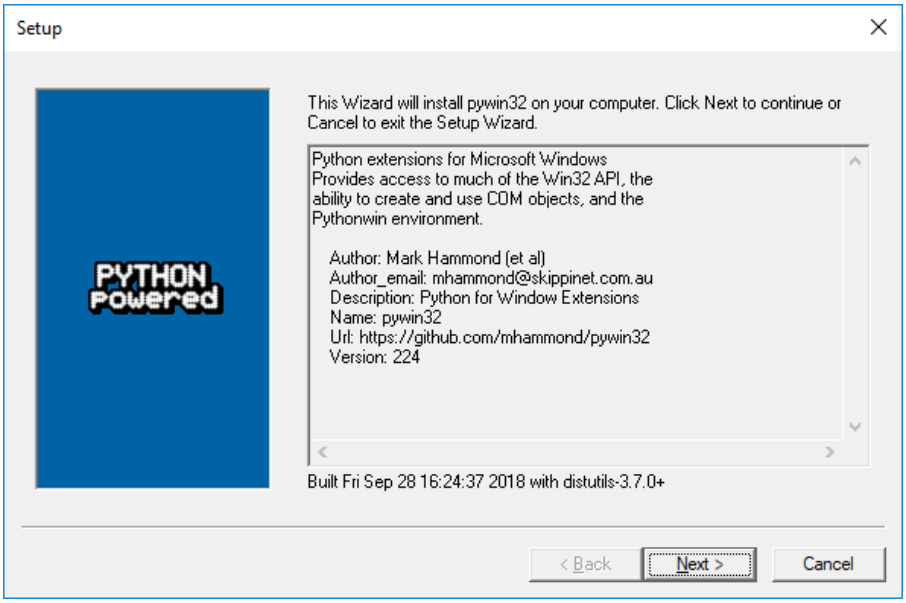
Select I agree and click on Install to continue installation of Microsoft Visual C++ 2015. Once installation is completed, a message box will appear and then click on close.



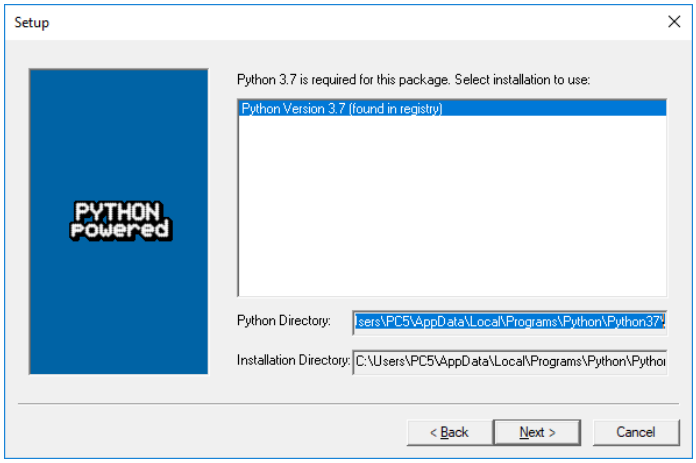
To start Python software installation, select the checkbox of "Add python 3.7 to PATH", then select Install Now and the Installation begins.



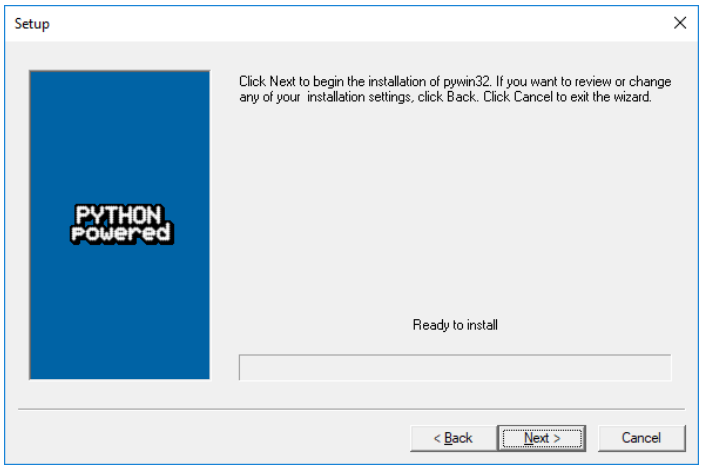
Click on **Close** once setup was successful**.**



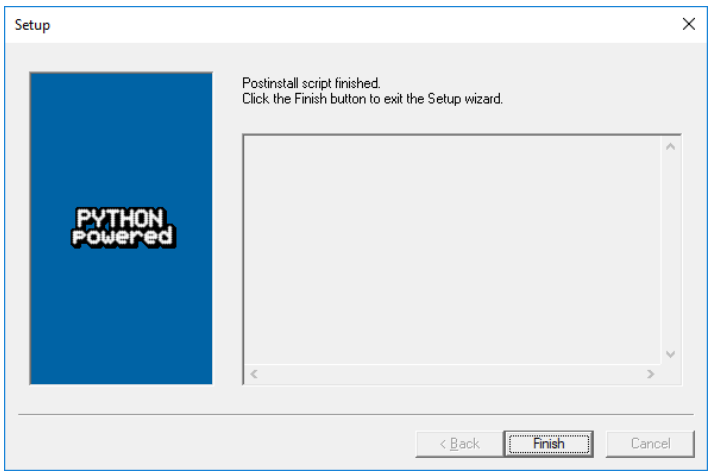
To install Pywin32, Click on **Next**



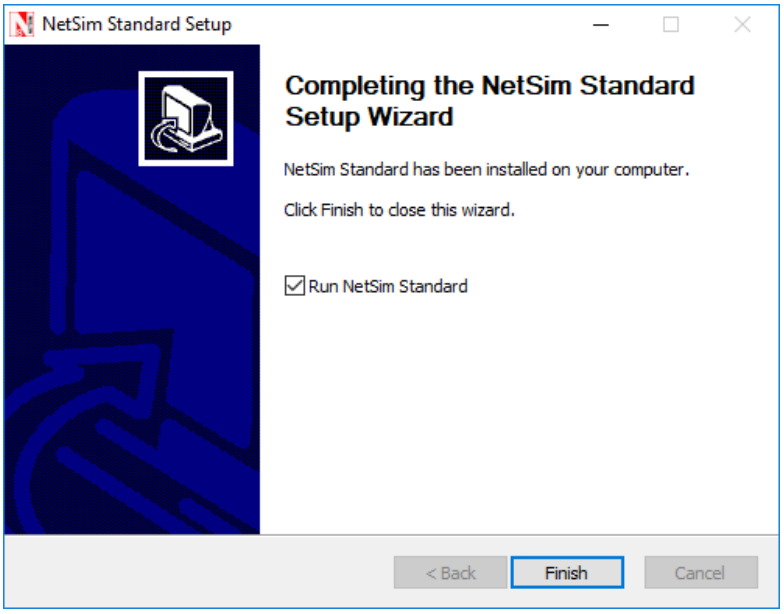
Select the Python directory and Click on **Next**.



Click on **Next** and the installation Pywin32 begins.



Click on **Finish**. This completes the Installation of python software



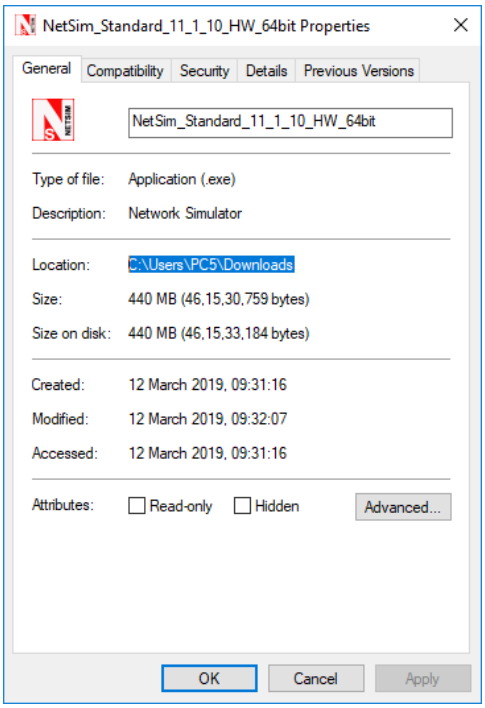
Select "**Run NetSim**" and then click on the **Finish** button.This completes the installation of NetSim Software.

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

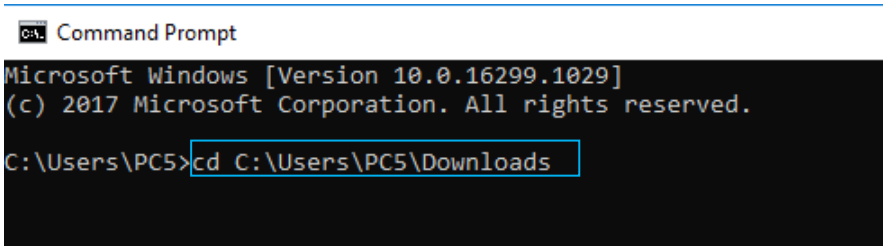
2.2.1 Silent installation

Steps for silent installation in NetSim are as follows:

1 For example, let us take the NetSim Standard 64 bit setup. Right click on NetSim Standard 64bit setup-> go to properties and copy the location



2 Open command prompt and paste the copied location as shown below:->*cd <setup location>*



Then, press enter.

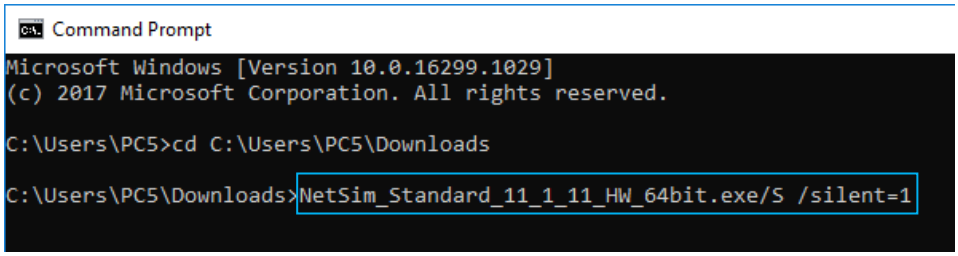
3 Run Execute Command with the following parameters,

NetSim_Standard_11_1_11_HW_64bit.exe/S /silent=1

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

I. silent=1: will install NetSim and third party tools silently

II. /S : will Install NetSim itself Silently



Then press enter.

4 Users can click on “Yes” as shown below to begin installation of NetSim Standard.

NOTE: Complete installation of NetSim may take upto 2 to 3 minutes.

2.3 Setting up License Server

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

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

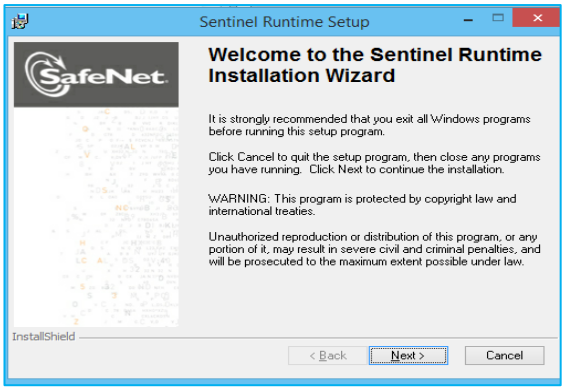
1 Insert the CD-ROM disc in the CD drive.

2 Double click on My Computer and access the CD Drive

3 Double click on **Driver_Software** folder.

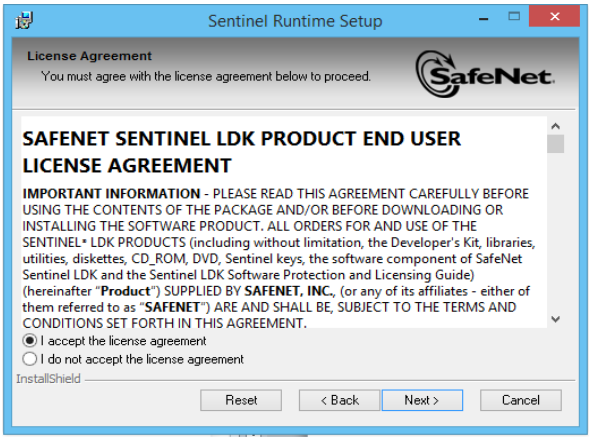
4 Double click on **HASPUserSetup.exe**

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

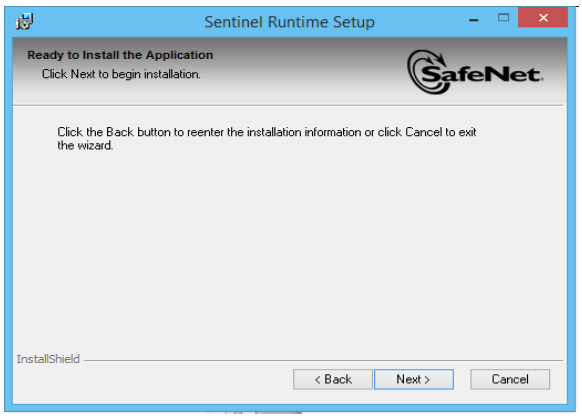


Setup prepares the installation wizard and the software installation begins with a **Welcome Screen**.Click on the **Next** button

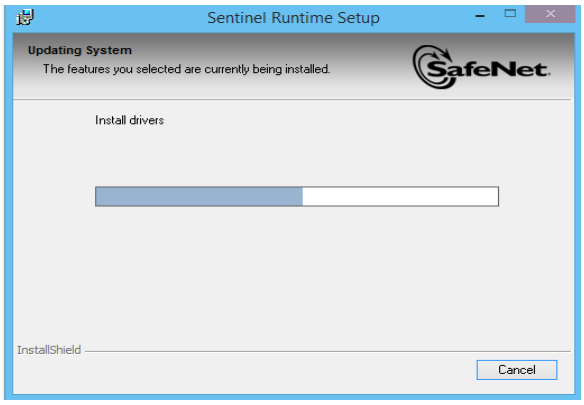
Note: Any other program running during the installation of the Dongle will affect the proper installation of the software.



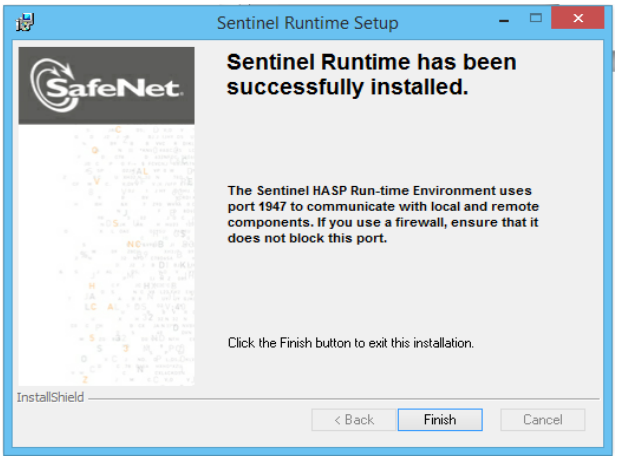
In the next screen, the License agreement is displayed.Read the license agreement carefully, scroll down to read the complete license agreement. If the requirement of the license agreement is accepted select the "**I accept**" button else quit the setup by clicking **Cancel** button.



Click on the **Next button**



The installation process begins.



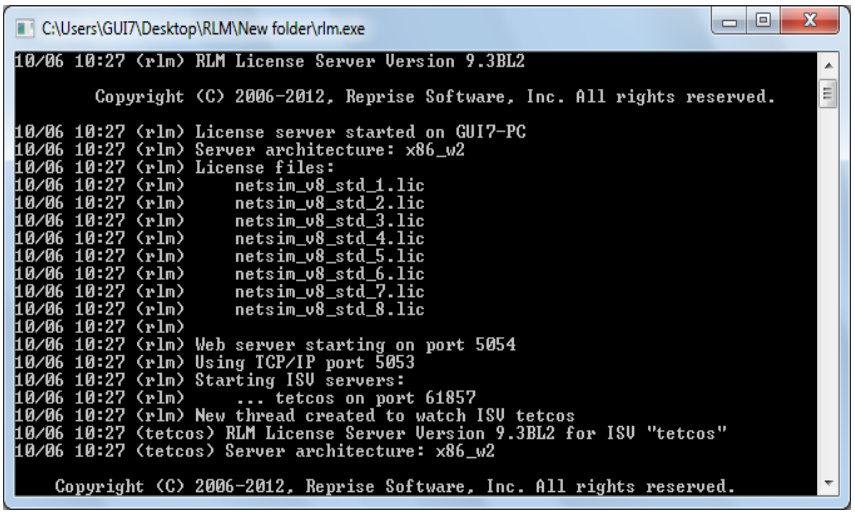
After the installation of the software, you will be requested to click Finish button to complete the installation process. Now the RLM driver software is installed successfully.

If the driver has been successfully installed then upon connecting the Dongle in the USB port red light would glow (Refer picture below). If the driver is not correctly installed this light will not glow when the dongle is connected to the USB port.

2.3.2 Running NetSim License Server

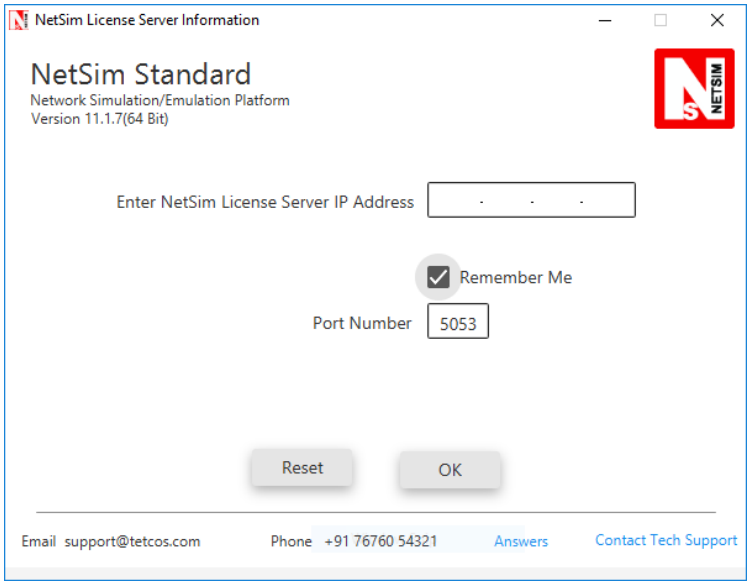
- Copy the **NetSim License Server** folder and paste it on **Desktop**. Check that it has the license file. If not copy the paste the license file into the **License server folder**
- Double click on **NetSim License Server** folder from Desktop.
- Double click on **rlm.exe**
- For hardware dongle-based 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. When you run rlm.exe, the screen will appear as shown below.



2.3.3 Running NetSim Software

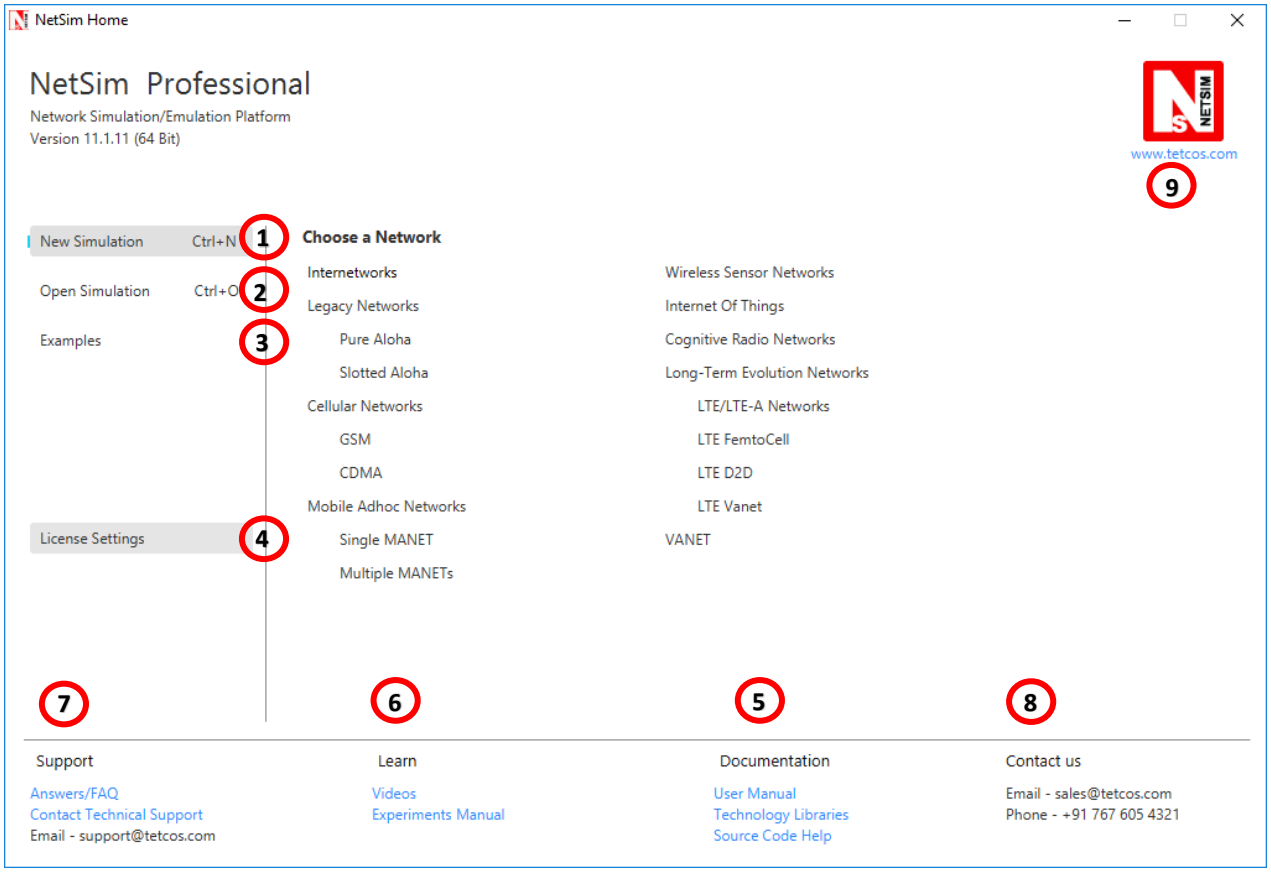
After running rlm.exe, click the NetSim icon in the Desktop.The screen given below will be obtained. Enter the Server IP address where the rlm.exe is running and click OK.



2.3.4 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.



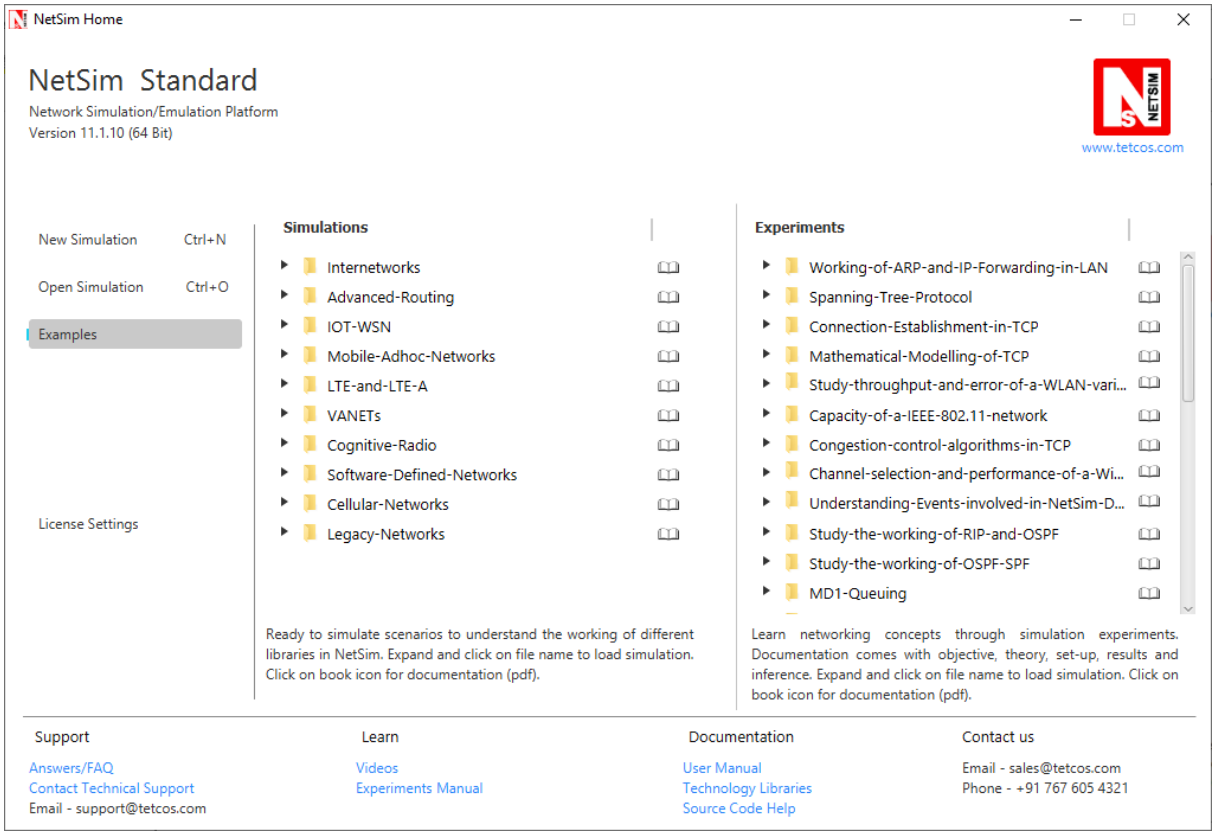
You see the following items on the NetSim Home screen:

1. **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 (Single MANET and Multiple MANETs), Cellular Networks, Wireless Sensor Networks, Internet of Things, Cognitive Radio Networks, LTE/LTE-A Networks (LTE/LTE-A, LTE femtocell, LTE D2D, LTE Vanet) and VANETs.

2. **Open Simulation:** 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.

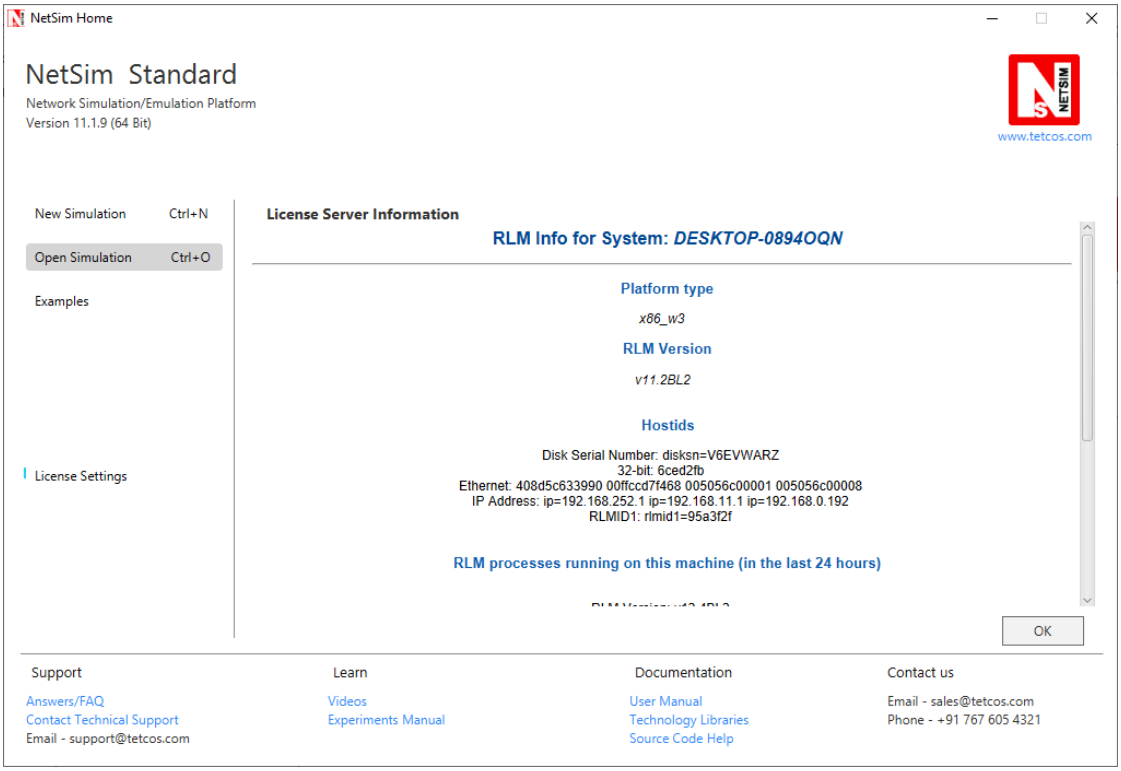


Similarly on the other side, users can find experiments section which has various experiments covering all the technologies in NetSim. Users can choose their experiment by either a double click on it or by a click on the pointer arrow which will take you the samples. Click on the sample to open the particular experiment in NetSim. All the settings to carry out a particular 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 particular 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 that is hosted locally to check the licenses you need to run NetSim.

   See the following image for an example of what the NetSim Home screen displays, if you click the License Server Information menu item.



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.

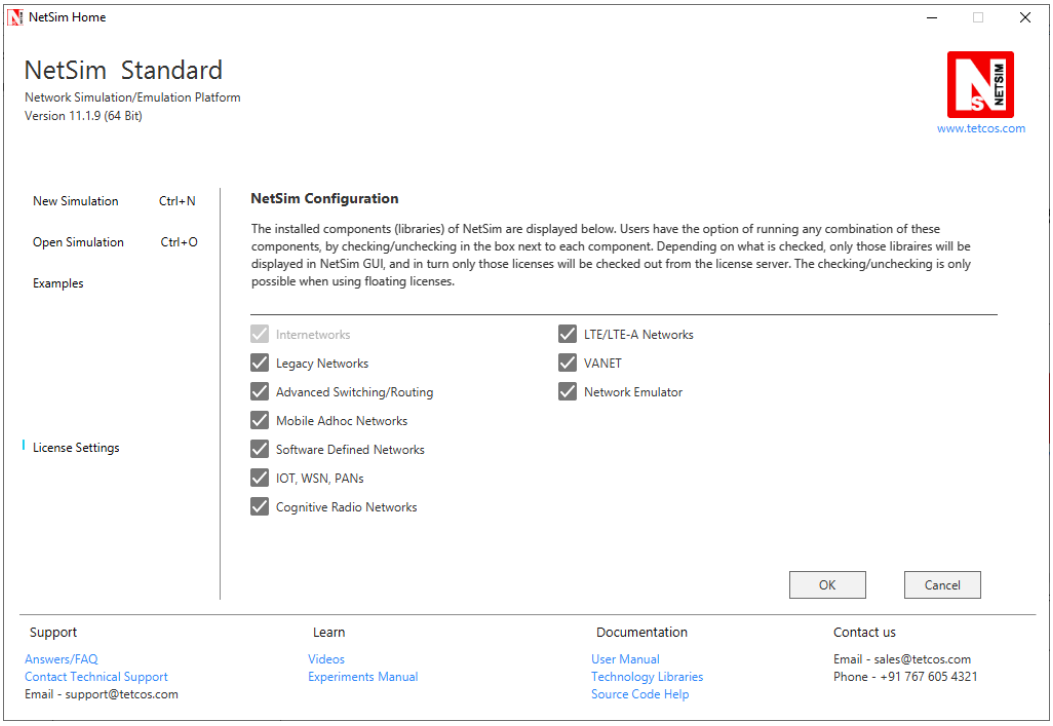
End User License Agreement: Use this menu to view the end user license agreement. You will see the following details on NetSim Home screen, if you click the End User License Agreement menu item: Grant of License and Use of the Services, License Restrictions, License Duration, Upgrade and Support Service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



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 ***NetSim 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 TETCOS videos on the TETCOS channel on

YouTube. 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***, 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.

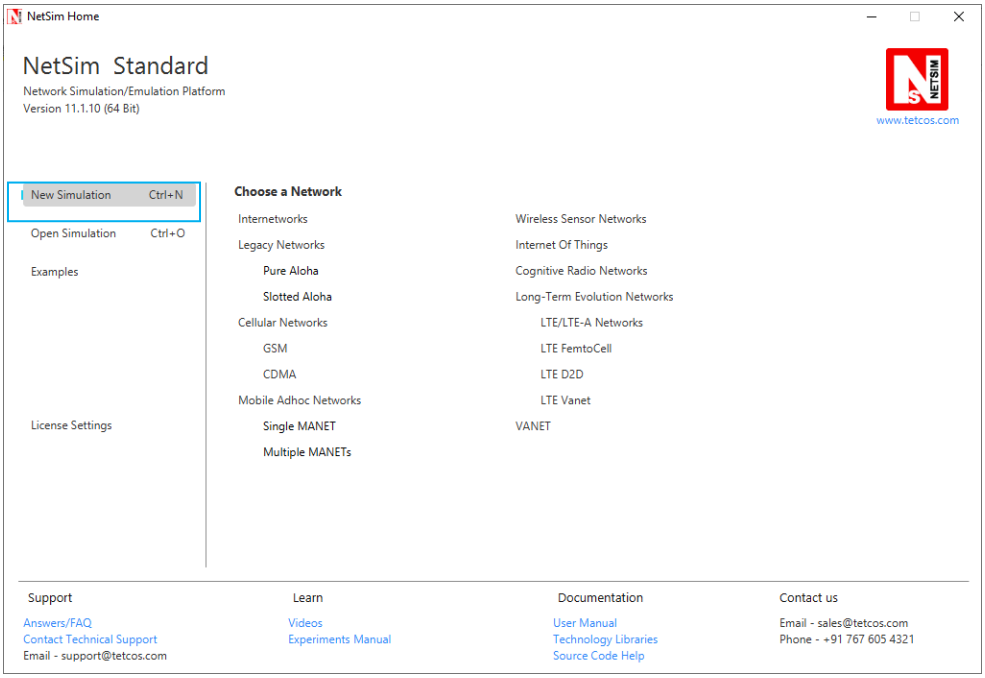
8. **Contact Us:** Use this section to contact us and know more information about our product. You can write to us via ***Email*** to ***sales@tetcos.com*** or contact us via ***Phone*** to our official number +91 76760 54321.

9. **Website:** Use this link ***www.tetcos.com*** to visit our website which consists of vast information which will assist you during all walks of NetSim.

2.3.5 Creating "New" Simulations

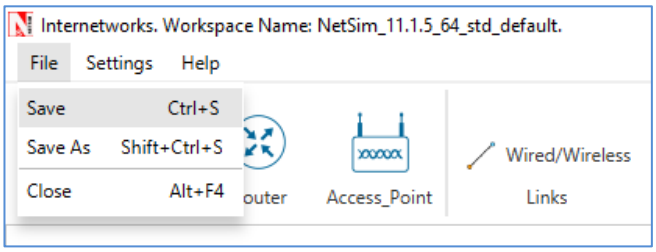
The Simulation window loads up once user selects the desired network technology from the New Menu.

To create a Network scenario



Click on New Simulation and select the desired kind of network to simulate

Save



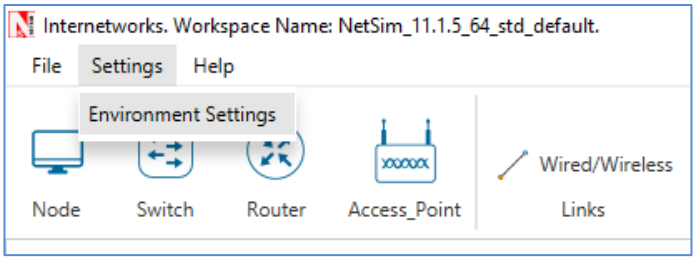
To save experiment, select File → Save, then specify the Experiment Name, Description (Optional) and click Save. The short cut for the same is Ctrl + S.

Save as

To save a previously saved experiment in a different location without overwriting the existing copy, Save As option can be used. The short cut for the same is Shift + Ctrl + S

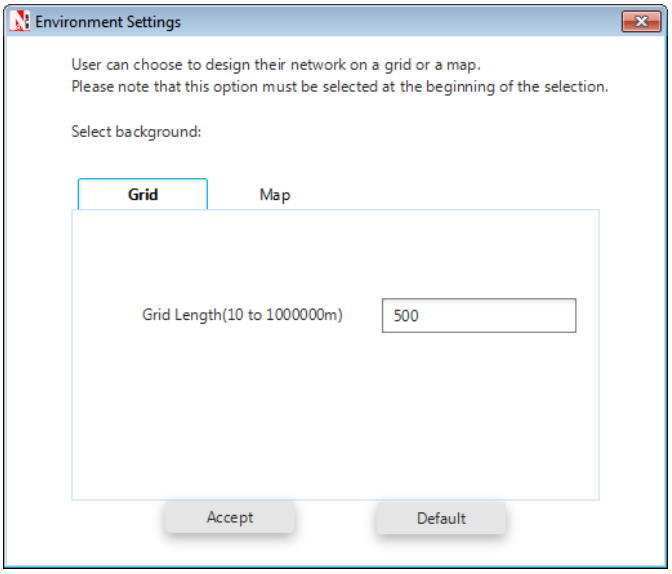
Settings

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



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 whereas for Map view users can configure the latitude and longitude respectively.



Learn

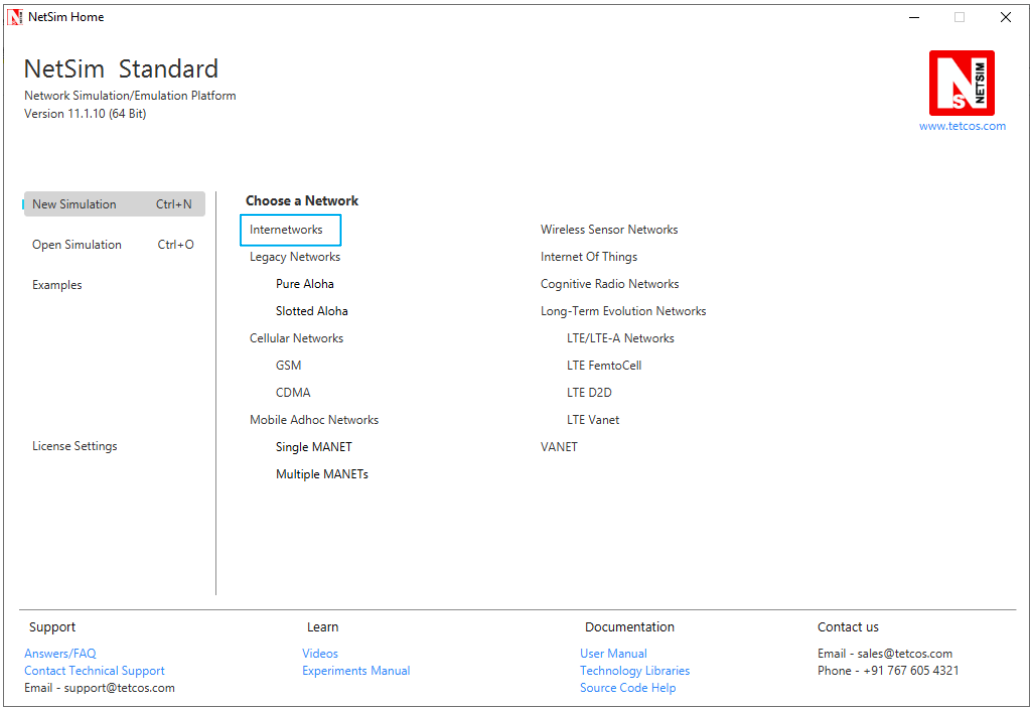
This menu contains link to NetSim Youtube Videos and NetSim Experiment manual.

Documentation

This menu contains link to NetSim user Manual, Technology Libraries and NetSim Source code Help.

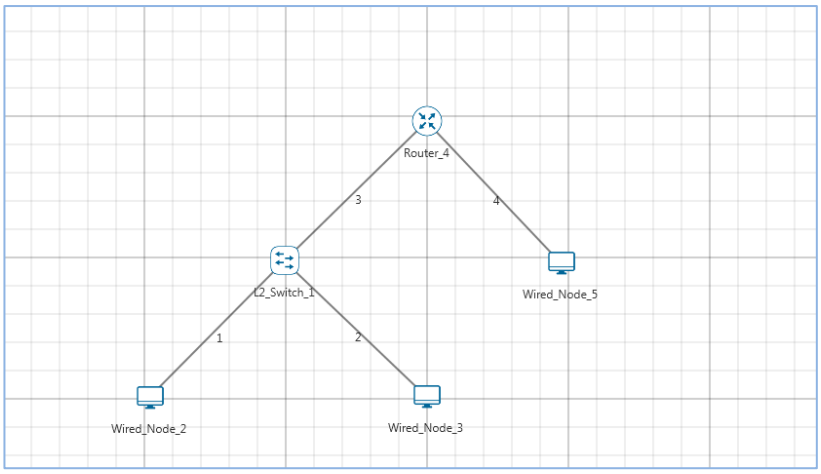
2.4 Modeling and Simulation of a simple network

This section will demonstrate how to create a basic network scenario and analyze in NetSim. Let us consider Internetworks. To create a new scenario, go to New Simulation→ Internetworks



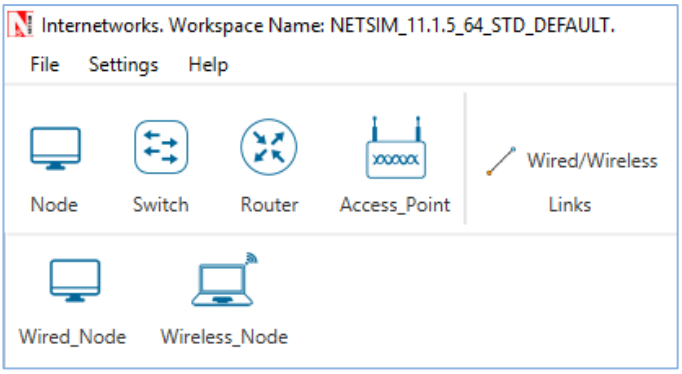
2.4.1 Creating a Network scenario

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.

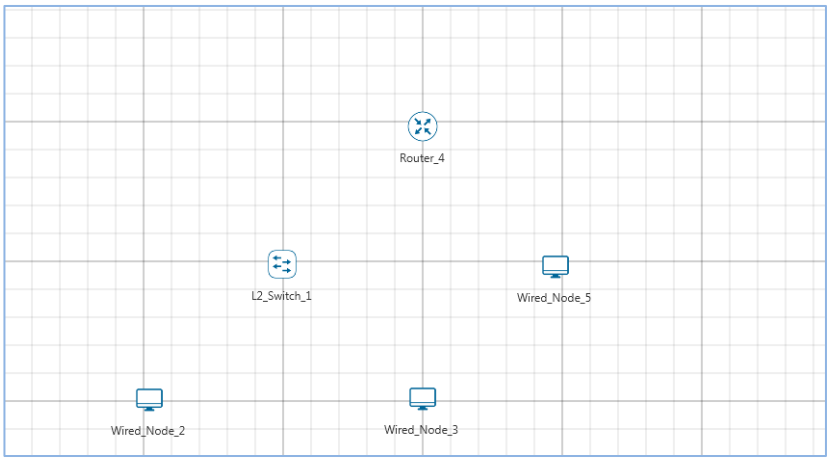


Perform the following steps to create this network design.

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

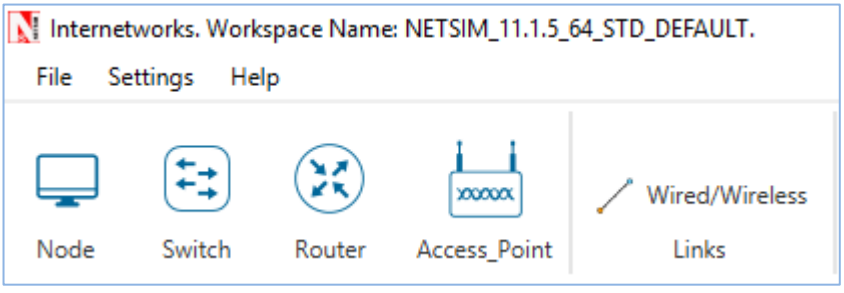


Click on the environment (the grid) where you want the Wired Node to be placed. In this way, place two more wired nodes. Similarly, to place a Switch and a Router, click on the respective device and click on the environment at the desired location.

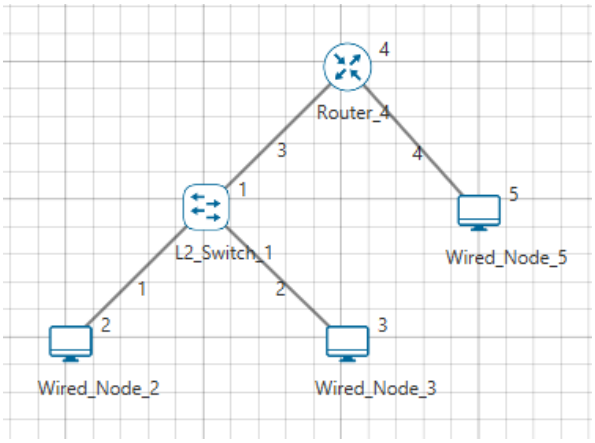


Note: The (x, y) position of any device on the grid is given by top left corner of the icon and not the center of the icon.

Step 2: Connecting devices on the environment. : With respect to establishing connection between devices, and in order to connect two devices, you will have to 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.

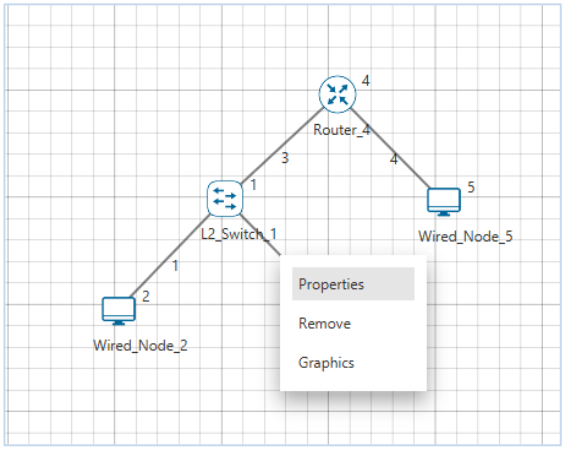


For example, select link and the click on Switch followed by router to connect them. In this manner, continue to link all devices.

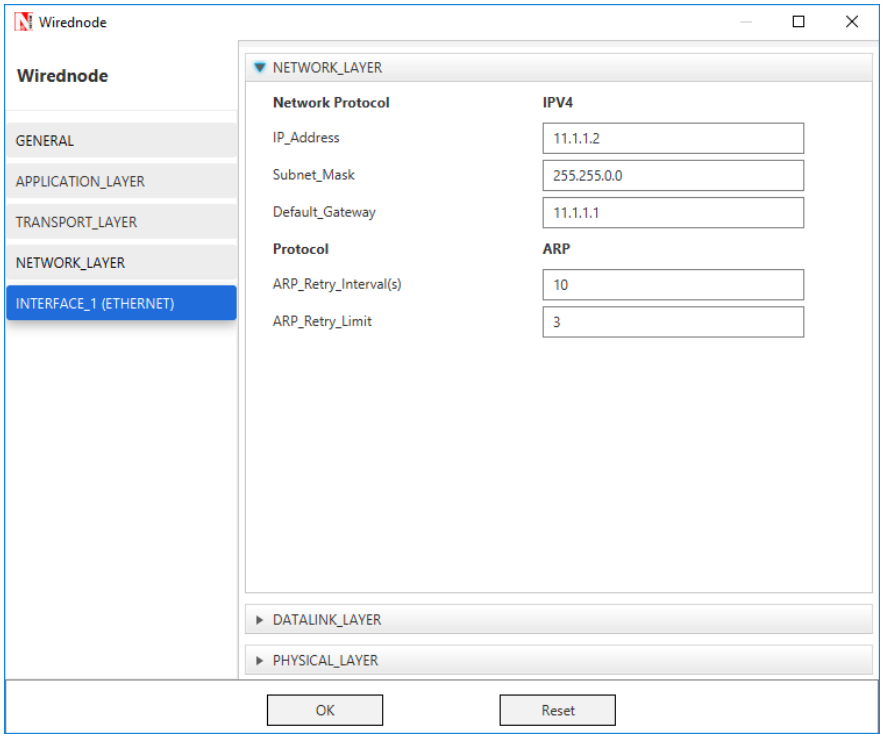


2.4.2 Configuring devices and links in the scenario

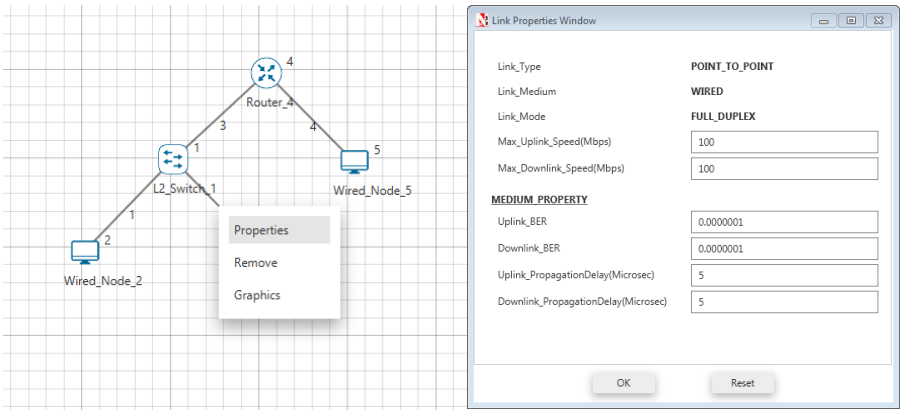
Step 1: To configure any device, right click on the device and select properties



User can set values according to their requirement. Modify the properties of any device and click on Ok. In this example default values are accepted.



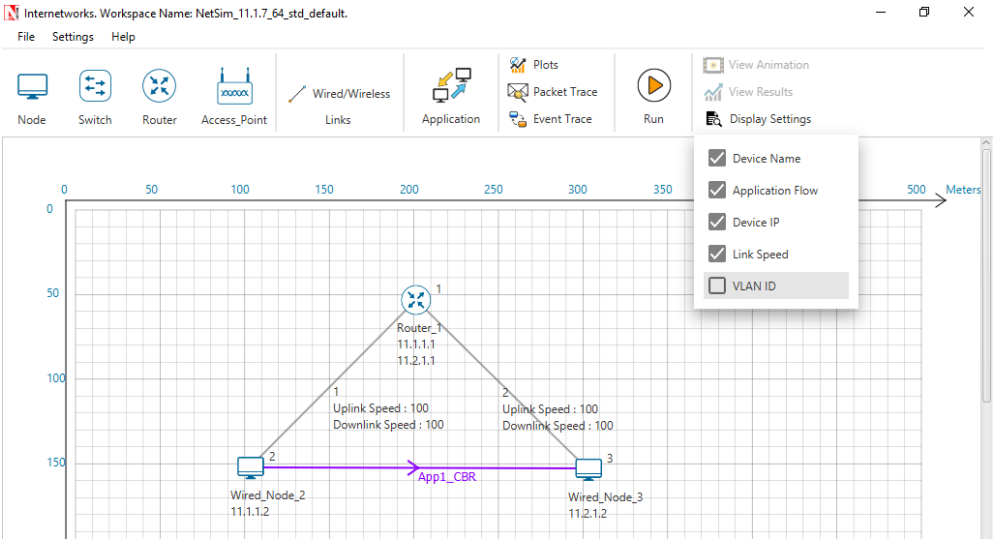
Step 2: To configure the links, right click on any Link and select Properties.



In above scenario, default values are accepted.

2.4.3 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

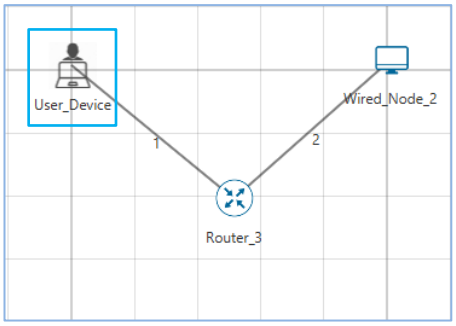


2.4.4 Copy/Paste

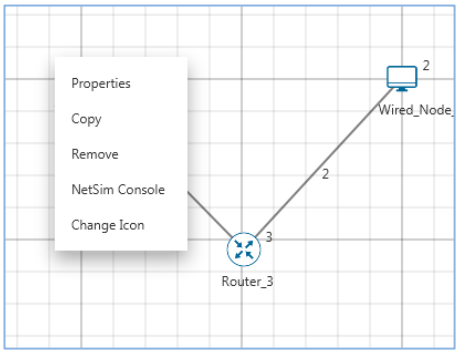
In NetSim simple copy paste can be used. Using this feature users can copy all the properties of a device and create a new device with similar properties.

Working of Copy/ Paste with example

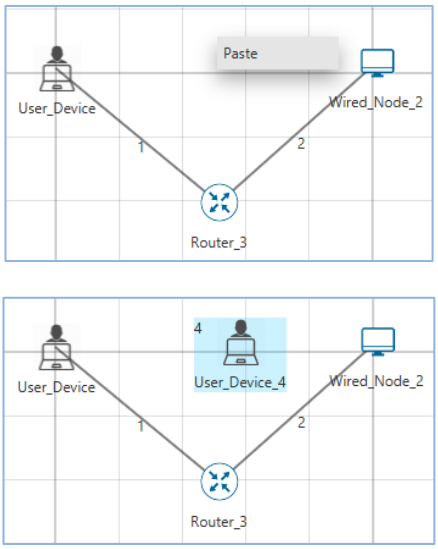
Right click on the device, click on copy on the device that users need to use it to drop similar device



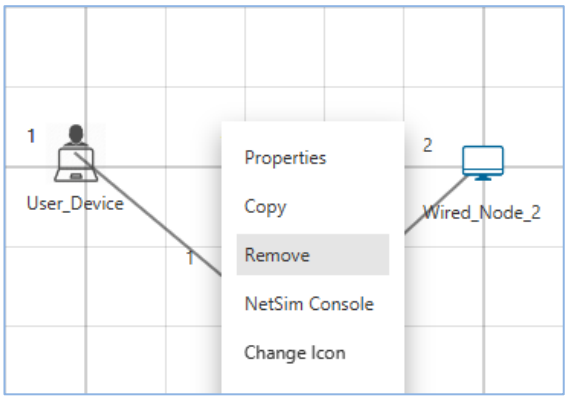
In this example we have considered a Wired Node for which the Device Name and Device Icon is modified. Right click on the device and select Copy.



Right click on the grid environment on the appropriate position that you need to drop the device click on Paste. A new device with similar properties will be added to the grid at the specific location as shown below:

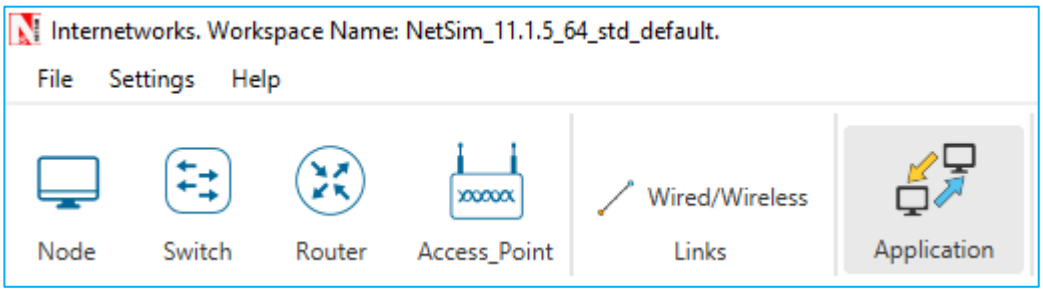


Remove in the device options, is used to delete the device from the grid environment. Given below is an example of removing the device User_Device_4 which was previously pasted.

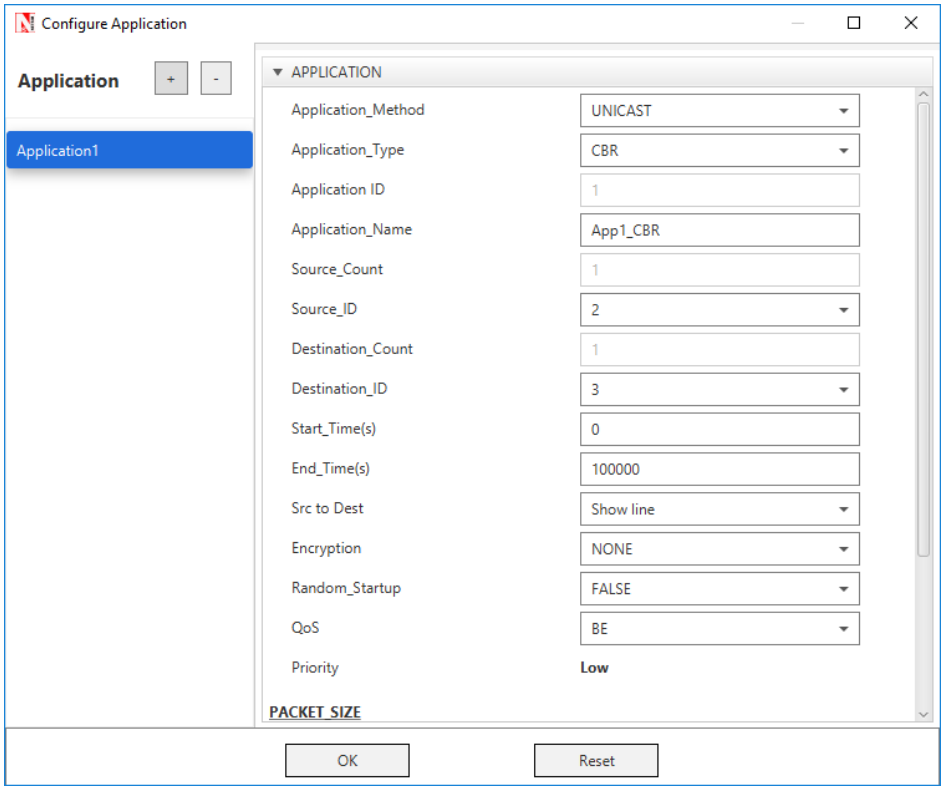


2.4.5 Modeling Traffic

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.



In screen shot shown below the Application type is set to CBR, source_ID is 2 and Destination_ID is 3. Click on **OK.**



Application Configuration:

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

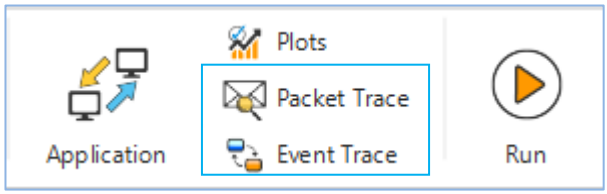
   - If TCP is enabled TCP may stop after the re-try limit is reached (since the SYN packets would not reach the destination)

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

4. Jumbo Frames are not supported in NetSim Ethernet Protocol

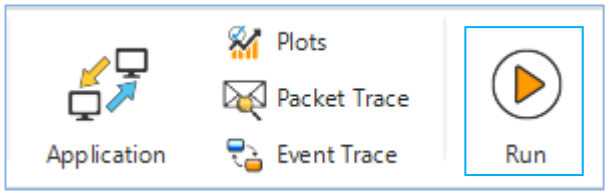
2.4.6 Logging Packet/ Event Trace

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 7.5 and 7.6 respectively.

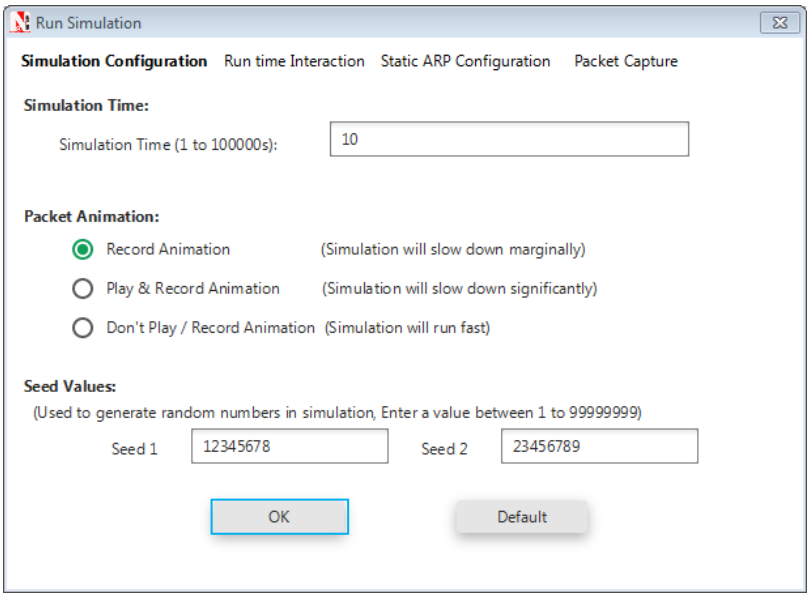


2.4.7 Run Simulation

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



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



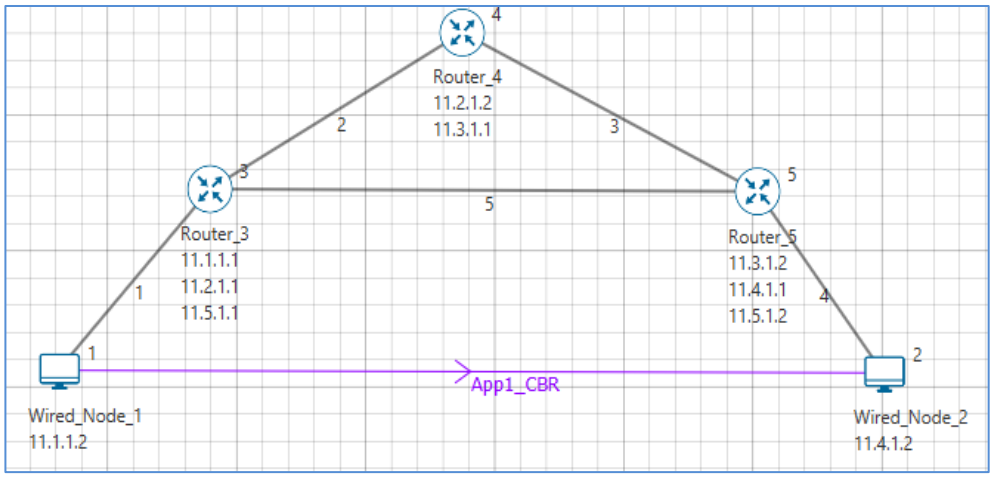
2.4.8 NetSim Interactive Simulation

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.

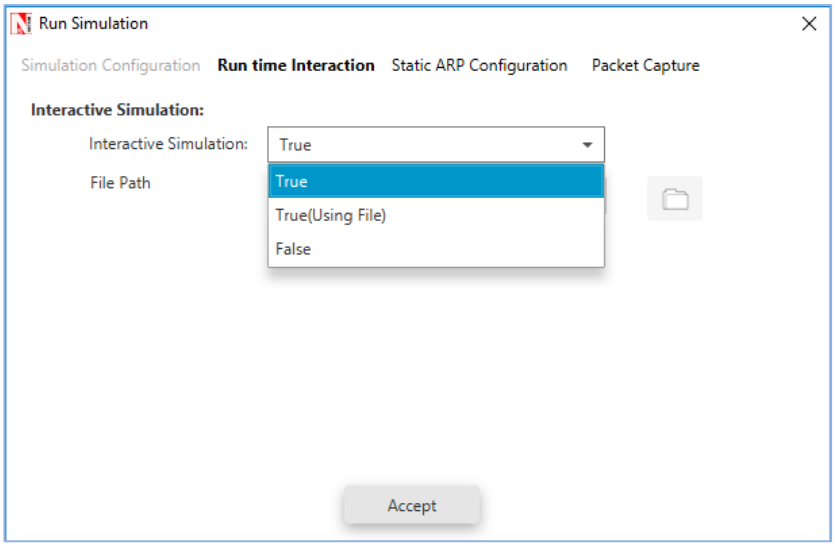
Working: This section will demonstrate how to perform Interactive simulation for a simple network scenario.

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 or otherwise Open the scenario for Interactive Simulation which is available in "<NetSim Install Dir>\Docs\ Sample_Configuration\Internetworks\Interactive Simulation"

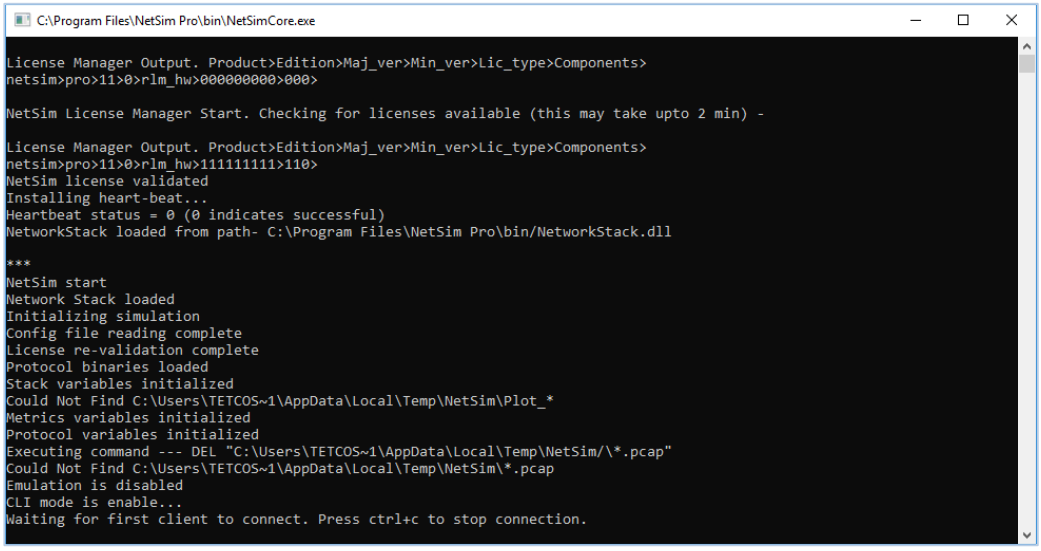


- Click on Application icon present in the top ribbon and set the Application type as CBR. The Source_Id is 1 and Destination_Id is 2.
- Set Start Time as 30 Sec
- Enable Plots and Packet trace options
- Click on run simulation option and In the Run time Interaction tab set Interactive Simulation as True and click on Accept

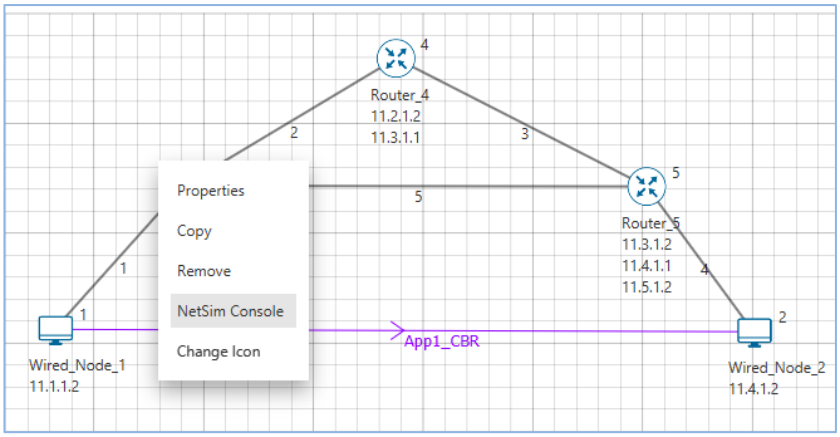


- Click on run simulation and set Simulation Time as 500 sec. (It is recommended to specify a longer simulation time to ensure that there is sufficient time for the user to execute the various commands and see the effect of that before Simulation ends) and click OK

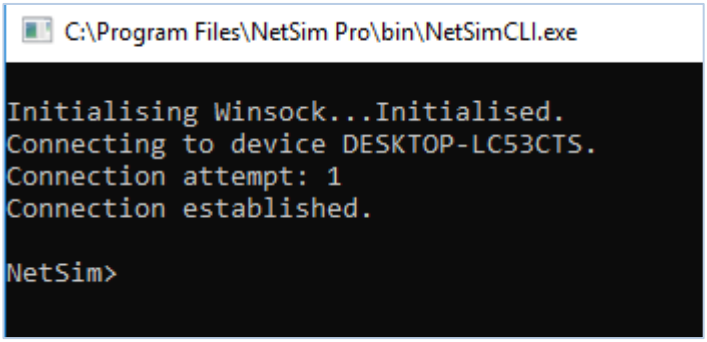
- Simulation (NetSimCore.exe) will start running and will display a message “waiting for first client to connect” as shown below:



- After Simulation window opens, goto Network scenario and right click on Router_3 or any other node and 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:



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

Note: Commands are not a case sensitive

1. Simulation specific (Not applicable for file based interactive simulation)

1. Pause
2. PauseAt
3. Continue
4. Stop
5. Exit
6. 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

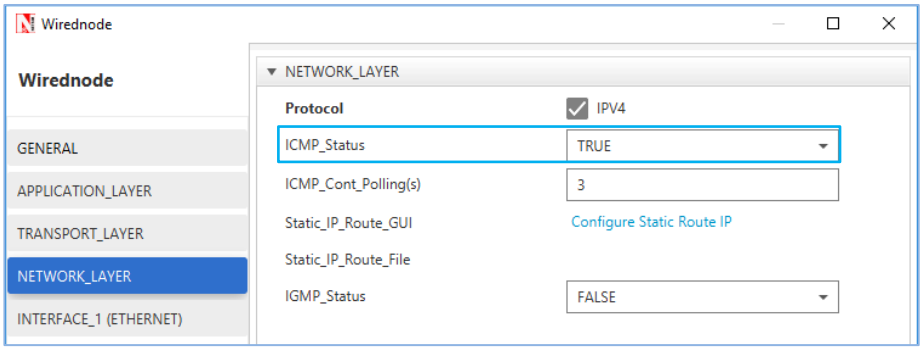
Stop: To stop the currently running simulation (NetSimCore.exe)

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

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

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)

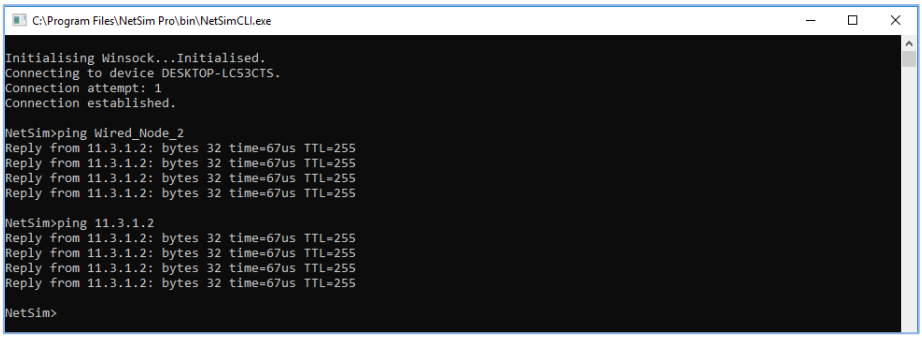


- Right click on Wired_Node_1 and go to properties. Under General properties enable Wireshark Capture option as "Online"

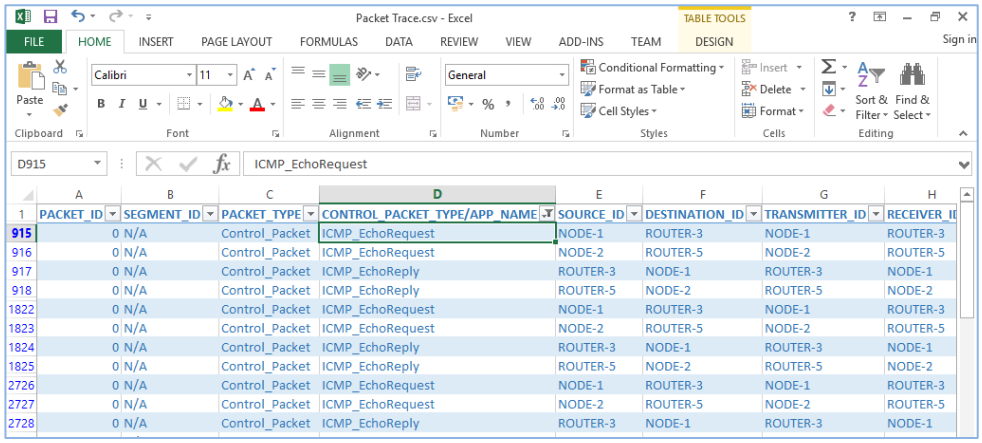
Ping <IP address> e.g. ping 11.4.1.2

Ping <NodeName> e.g. ping Wired_Node_2

Ping Command Results:

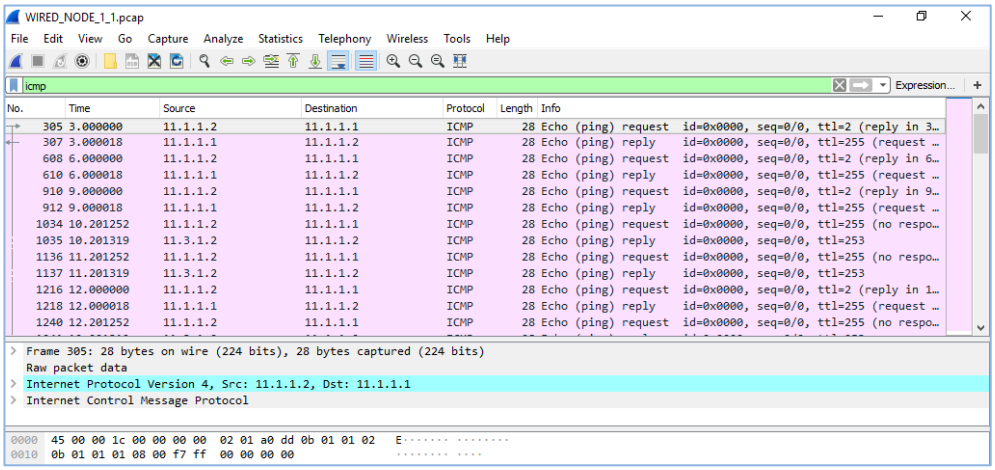


- After simulation open packet trace and filter ICMP_EchoRequest and ICMP_EchoReply from CONTROL_PACKET_TYPE/APP_NAME column



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

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

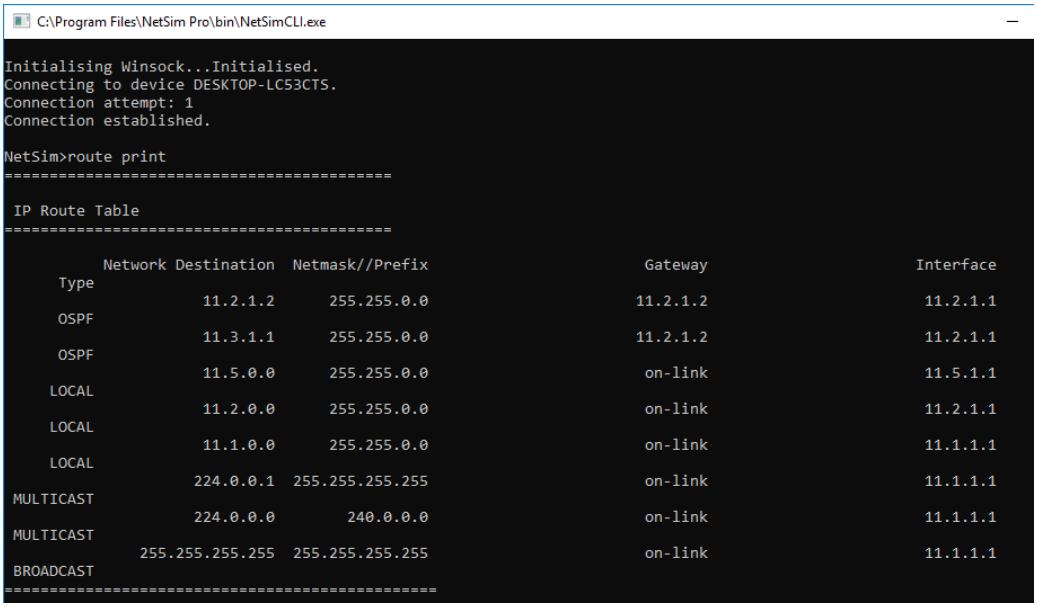


3. Route Commands

1. route print
2. route delete
3. route add

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

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

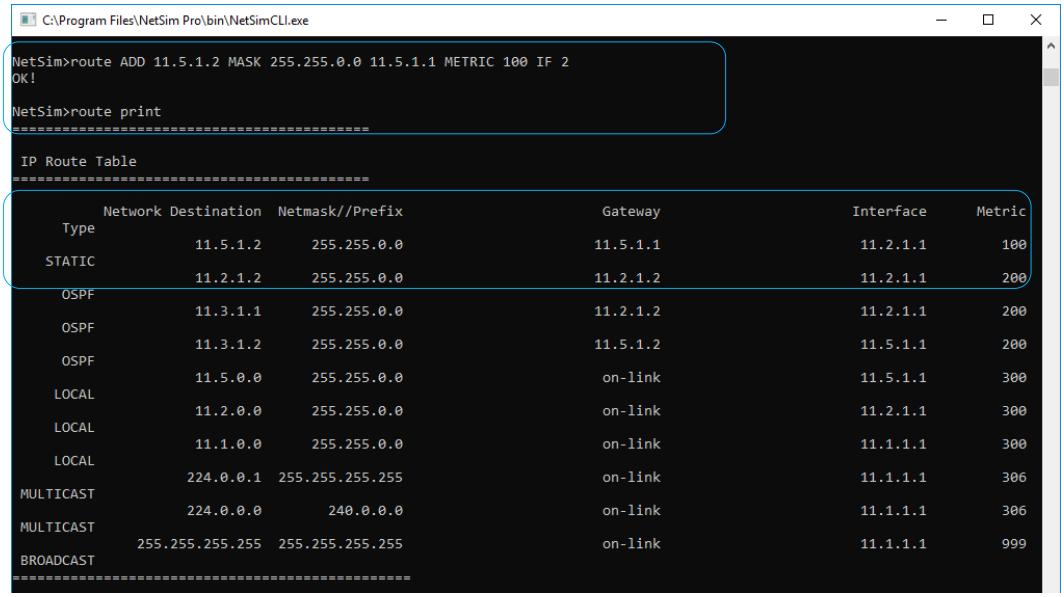
- To check whether route has been deleted or not check again using **route print** command
- To add a static route to the table, you will type a command using the following syntax

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

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

route ADD 11.5.1.2 MASK 255.255.0.0 11.5.1.1 METRIC 100 IF 2

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



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

3. 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.

Commands to configure ACL:

- 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

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***

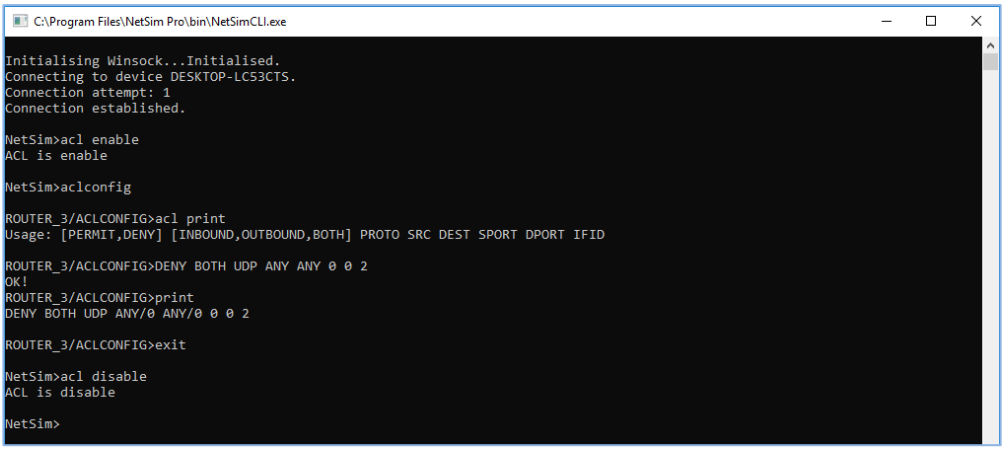
DENY BOTH UDP ANY/0 ANY/0 0 0 2

*ROUTER_3/ACLCONFIG>**exit***

*NetSim>**acl disable***

ACL is disable

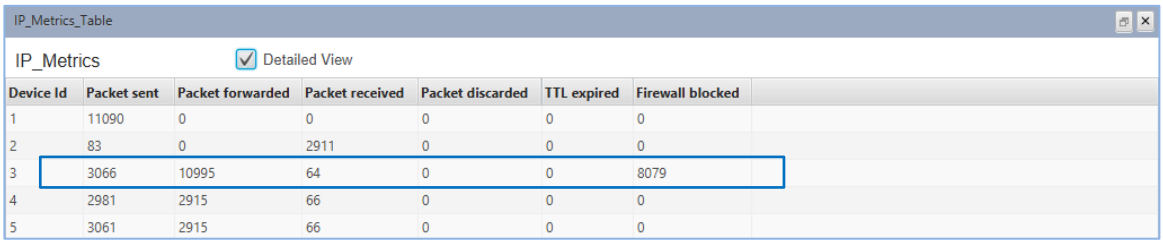
NetSim>



ACL Results:

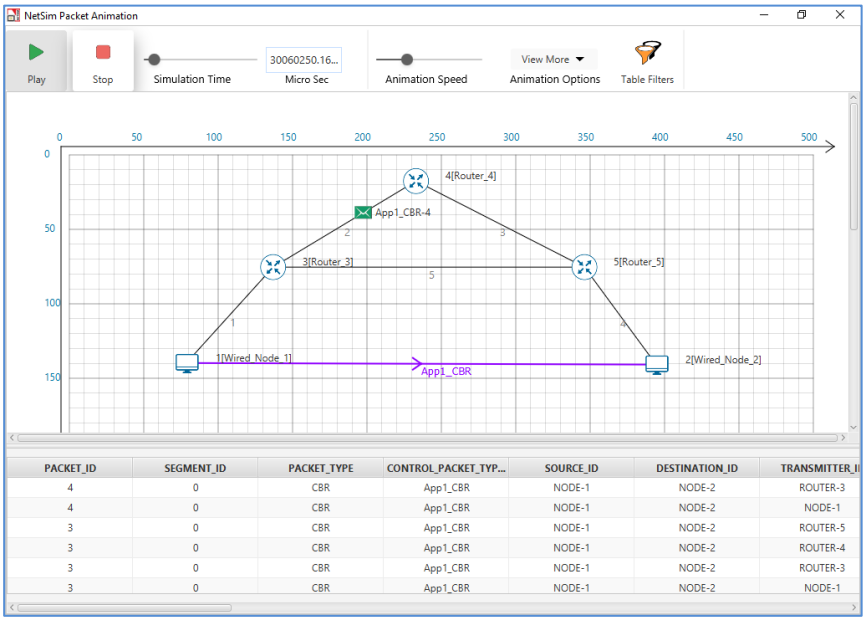
The impact of ACL rule applied over the simulation traffic can be observed in the IP_Metrics_Table in the simulation results window, In Router_3 number of packets blocked by firewall has been shown below

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



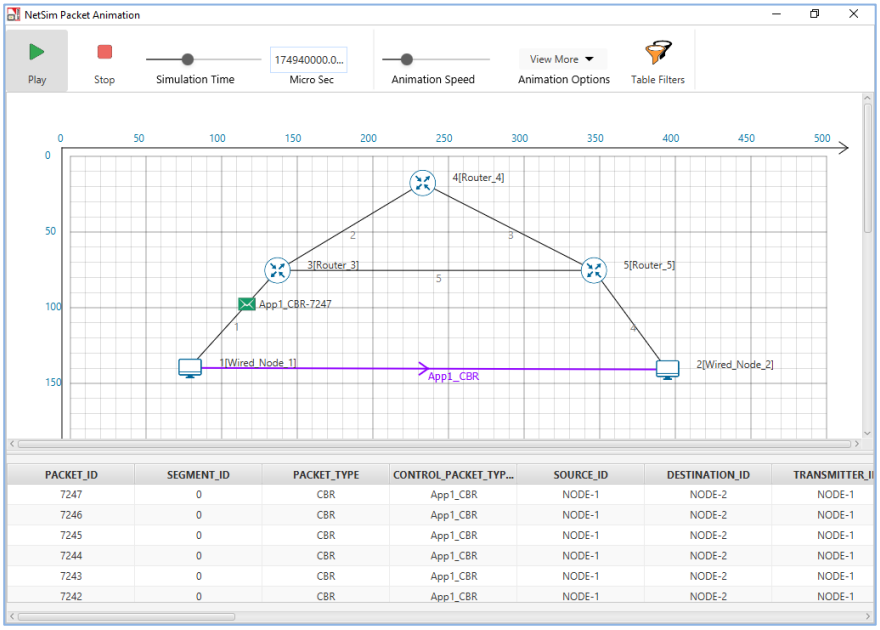
Device Id	Packet sent	Packet forwarded	Packet received	Packet discarded	TTL expired	Firewall blocked
1	11090	0	0	0	0	0
2	83	0	2911	0	0	0
3	3066	10995	64	0	0	8079
4	2981	2915	66	0	0	0
5	3061	2915	66	0	0	0

- 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



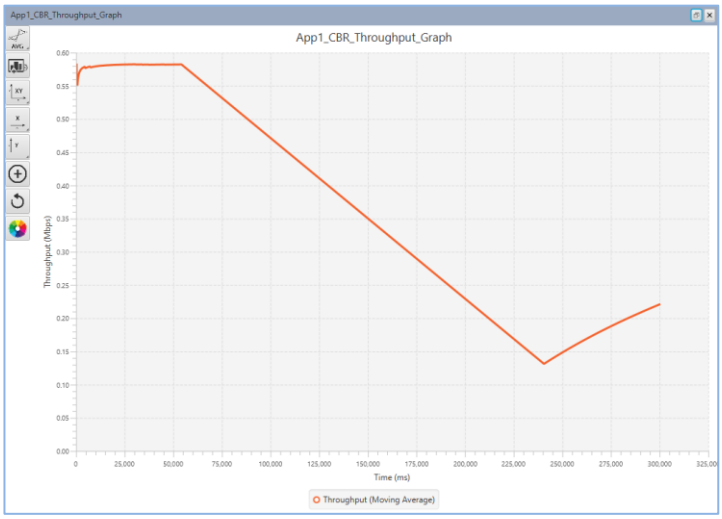
PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYP...	SOURCE_ID	DESTINATION_ID	TRANSMITTER_I
4	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3
4	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1
3	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-5
3	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-4
3	0	CBR	App1_CBR	NODE-1	NODE-2	ROUTER-3
3	0	CBR	App1_CBR	NODE-1	NODE-2	NODE-1

- After applying ACL rule Packet flows up to Router_3 only



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 througput graph) in throughput after setting ACL and disabling ACL respectively.

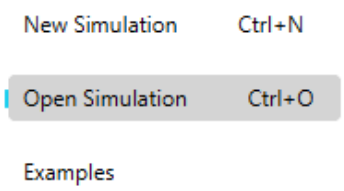
Example: ACL rule applied at around 50sec user can see the drop in throughput in the graph, since router blocks UDP packets in the plot. Once ACL has been disabled at around 240sec router permits packets and hence throughput can be observed in the plot shown below



2.5 Saving & Opening experiments and Printing results

2.5.1 Opening Saved Experiments: Open Network – All Networks

Click on **Open Simulation** (Ctrl+O).

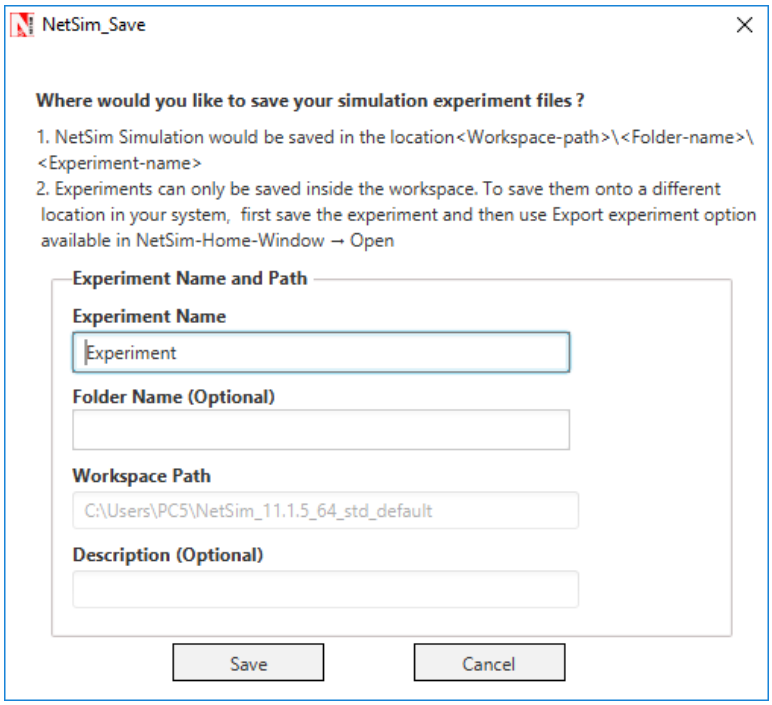


Open saved experiment file you want to open.

2.5.2 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.



Upon saving a number of files would get saved inside the folder, including

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

2.6 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:

Keys	Function
Home Screen	
Ctrl + N	Open a New Network
Ctrl + O	Open a Saved Network
Design Window (Any Network)	
Ctrl + S	Save the Experiment
Shift + Ctrl + S	Save As
Alt+ F4	Close Window
F1	User Manual Help
Ctrl + '+/-'	Zoom In/Zoom Out
Delete	Delete the devices in the Environment with links
Simulation Console	
Ctrl + C	Terminates Simulation in Mid way. Results will be calculated till the time at which the simulation is terminated
Packet Animation Window	
Space bar	To Play/Pause animation

3 Workspaces and Experiments

3.1 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.

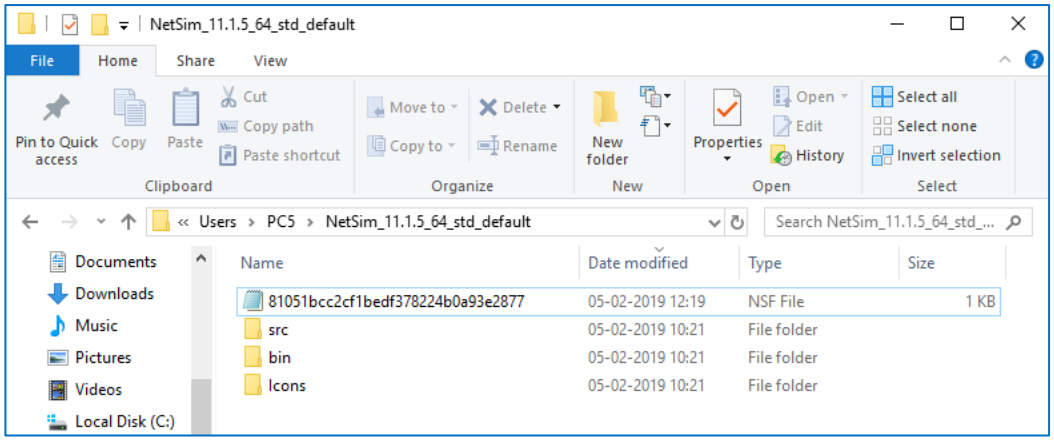
New workspaces need to be created by users when:

1. 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.
2. A user chooses to save and organize a large number of experiments. These can be saved under different workspaces provided by that user.
3. 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)

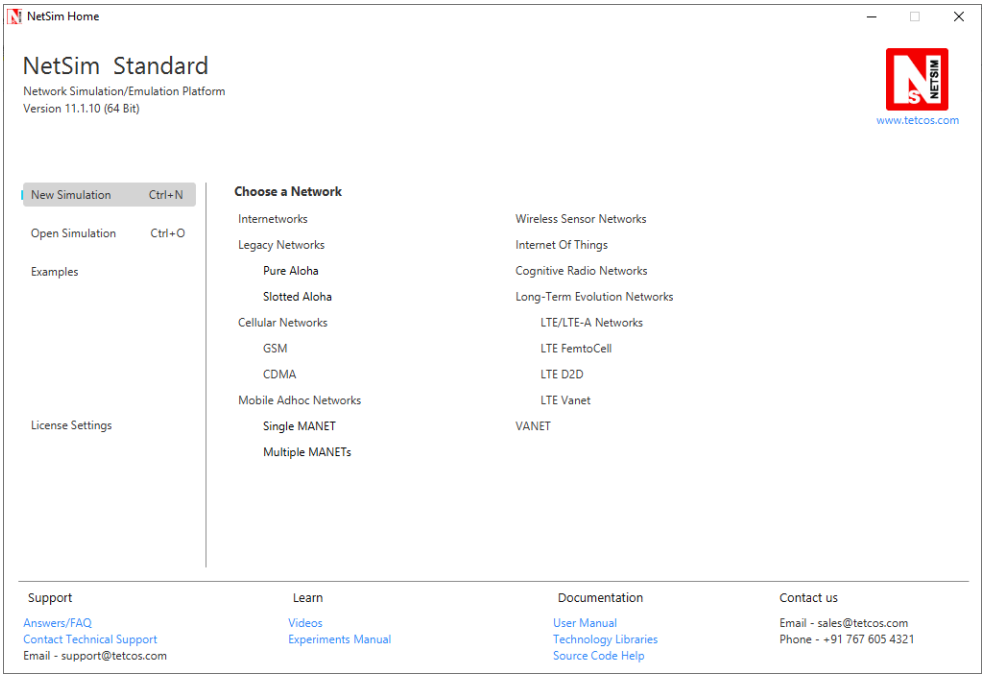
A default workspace is created under current user directory (Ex: C:\Users\PC5) path if when NetSim is run for the first time after installation. This default workspace contains the folders

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

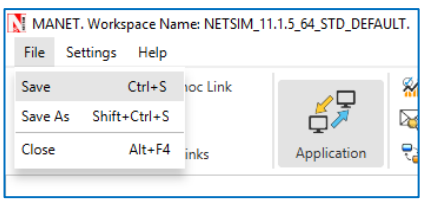


3.2 How does a user create and save an experiment in workspace?

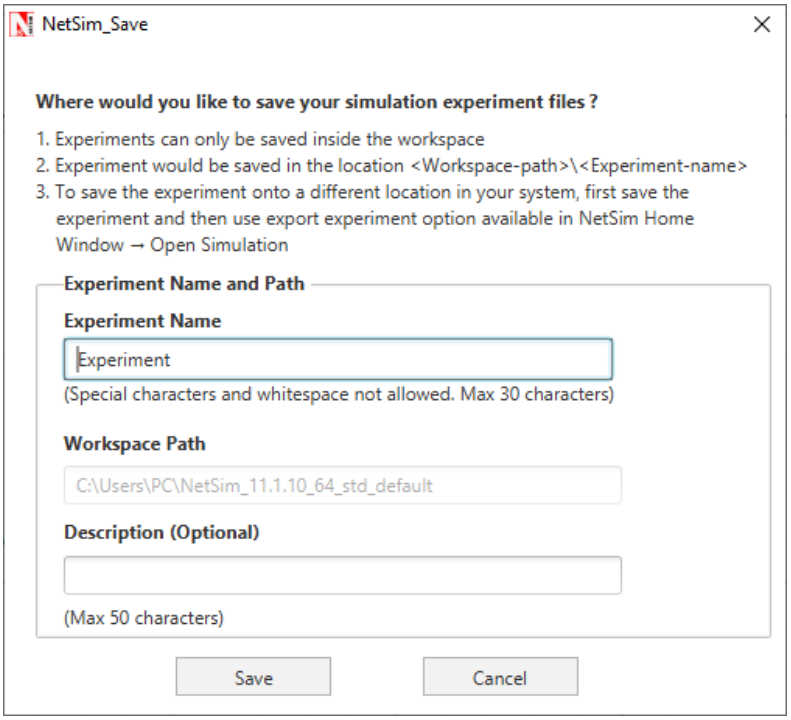
To create an experiment, select New Simulation-> <Any Network> in the NetSim Home Screen.



Create a network and save the experiment by clicking on File->Save button on the top left.

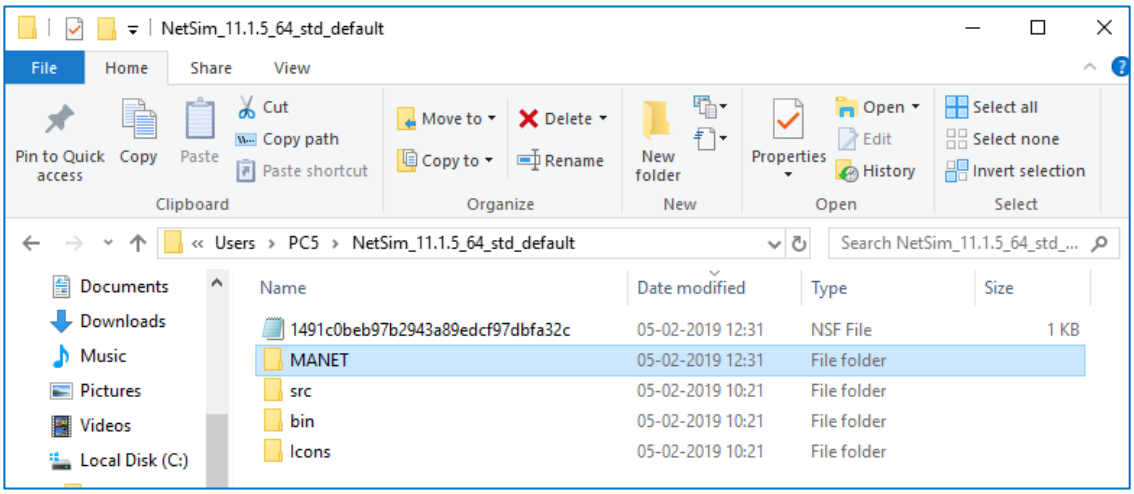


A save popup window appears which contains Experiment Name, Folder Name, Workspace path and Description.

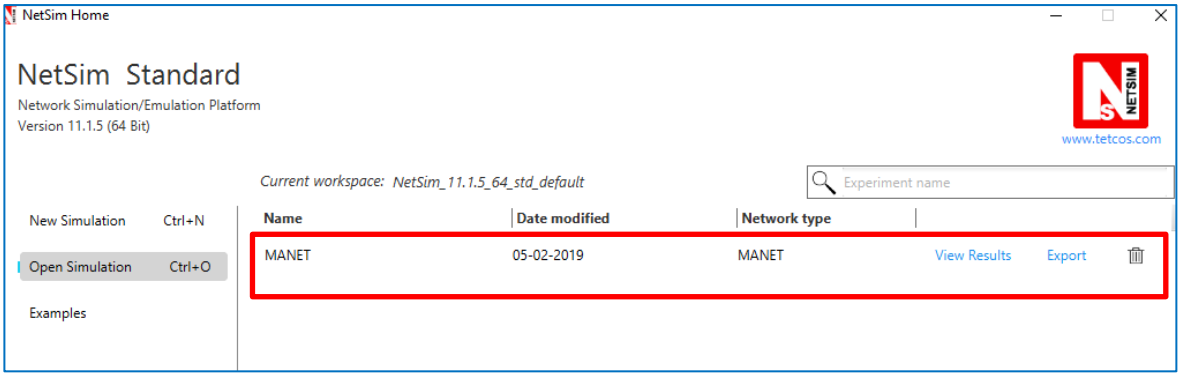


Specify the Experiment Name and Description (Optional) and then click on Save. The workspace path is non-editable. Hence all the experiments will be saved in the default workspace path. After specifying the Experiment Name click on Save.

In our example we saved with the name MANET and this experiment and it can be found in the default workspace path as shown below:



Users can also see the saved experiments in Open Simulation menu shown below:



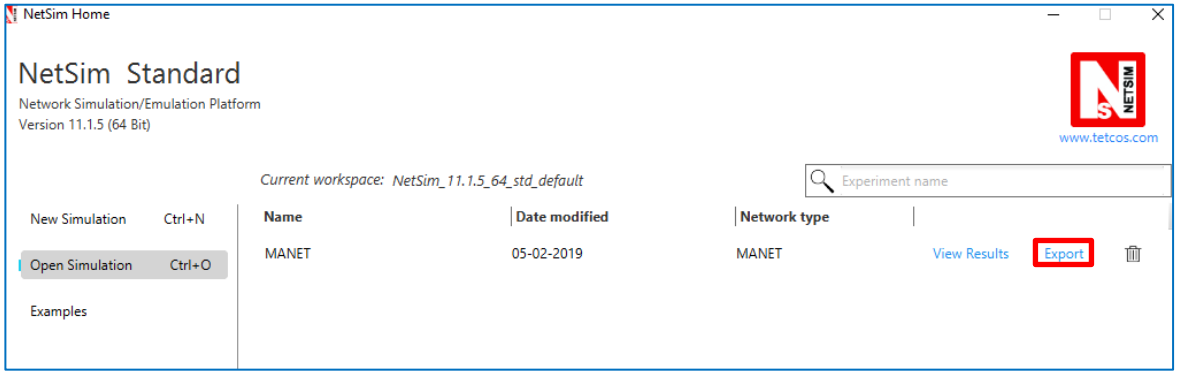
Note: “Save As” option is also available to save the current experiment with a different name.

3.3 Should each user have a workspace?

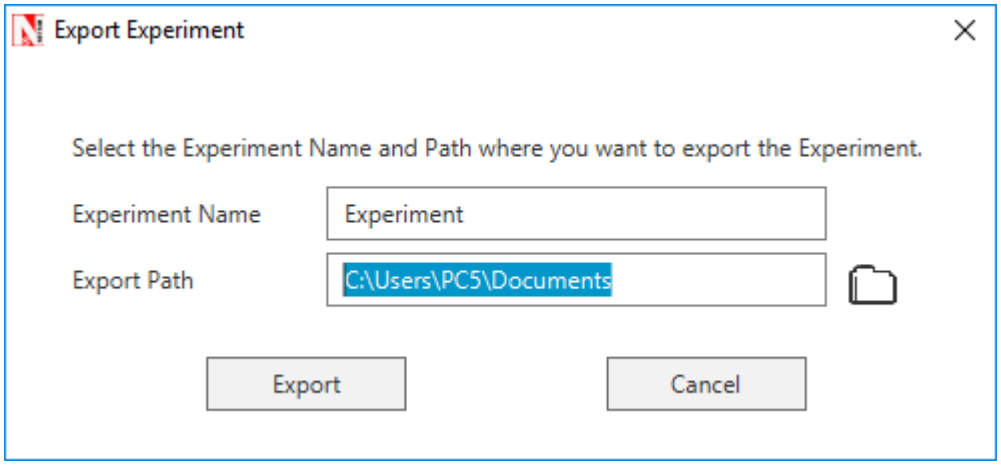
There is no strict association between users and workspaces. A single user can have multiple workspaces (and in turn experiments in each workspace), or multiple users can operate in one workspace

3.4 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 Open Simulation menu in the NetSim Home Screen



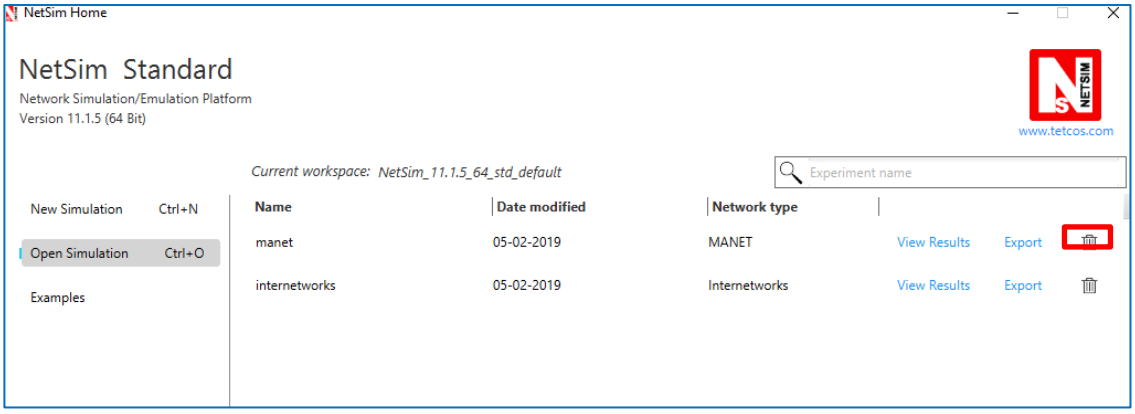
If you click on Export, an Export Experiment pop-up window appears where you can input the experiment name and the Export path.



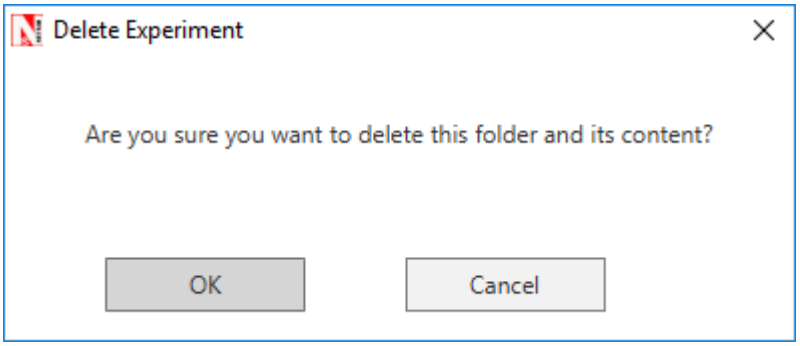
The exported experiments would be saved with *.netsim_exp extension.

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

Users can delete experiments by clicking on the delete icon as shown below

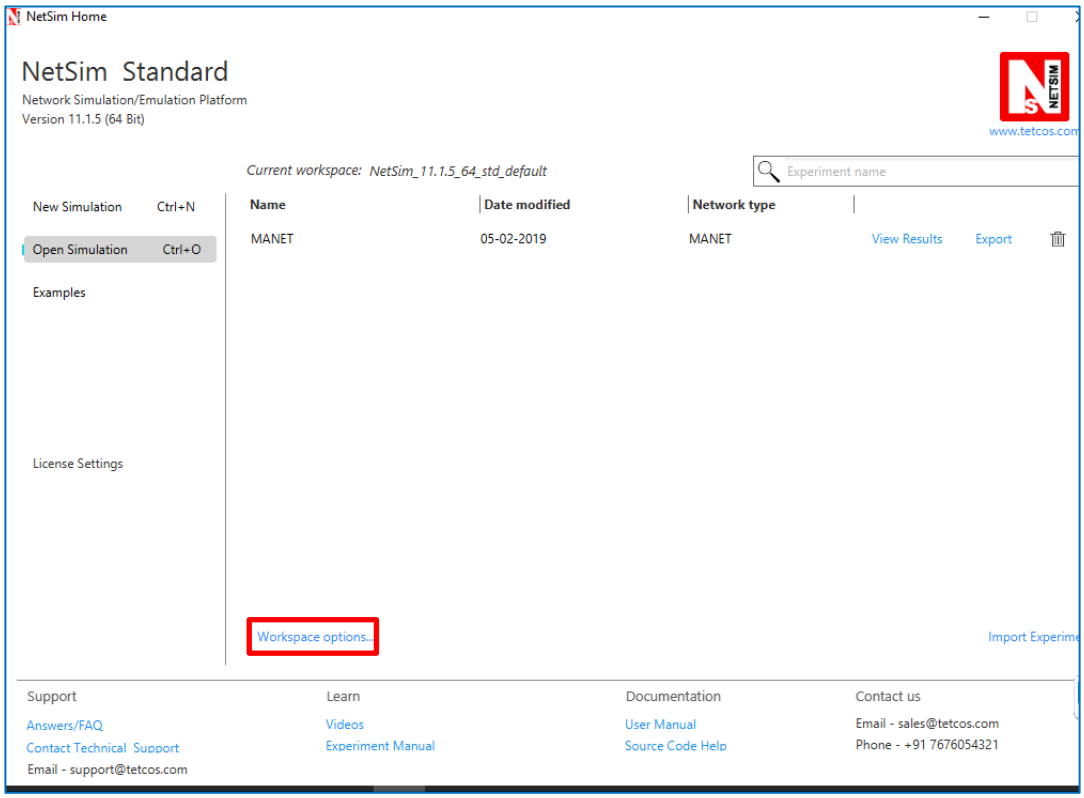


It displays a popup window as shown below and click on OK.

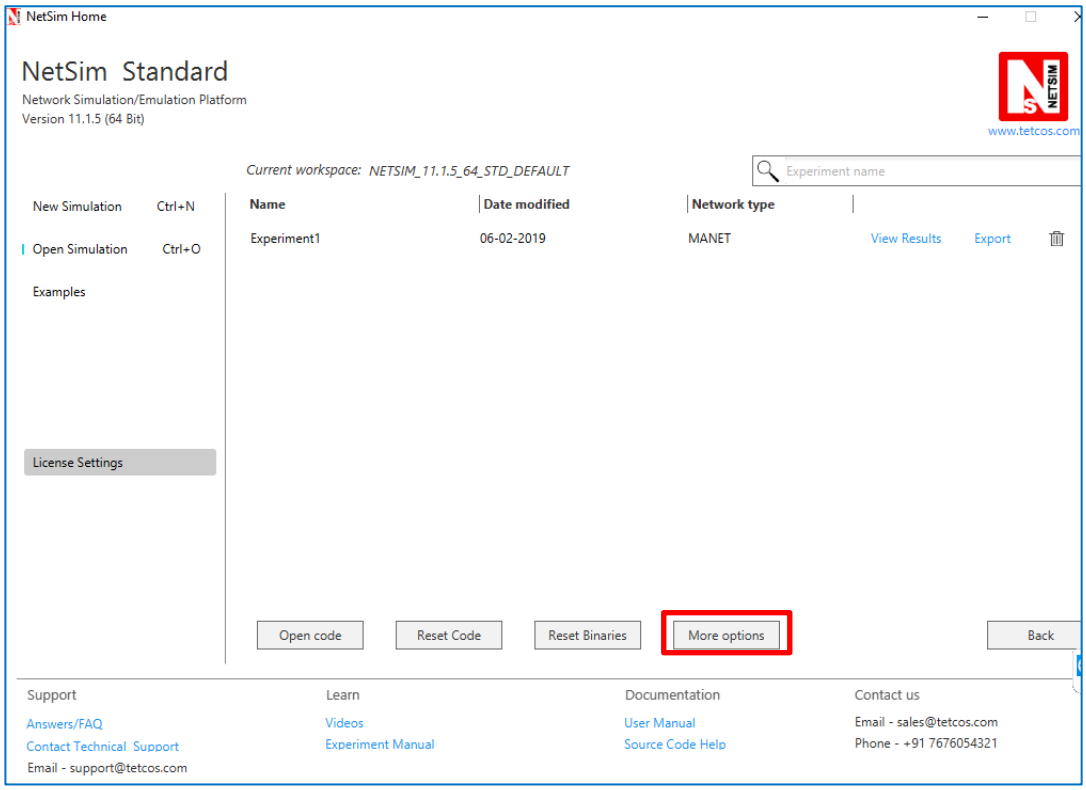


3.6 How does a user create a new workspace?

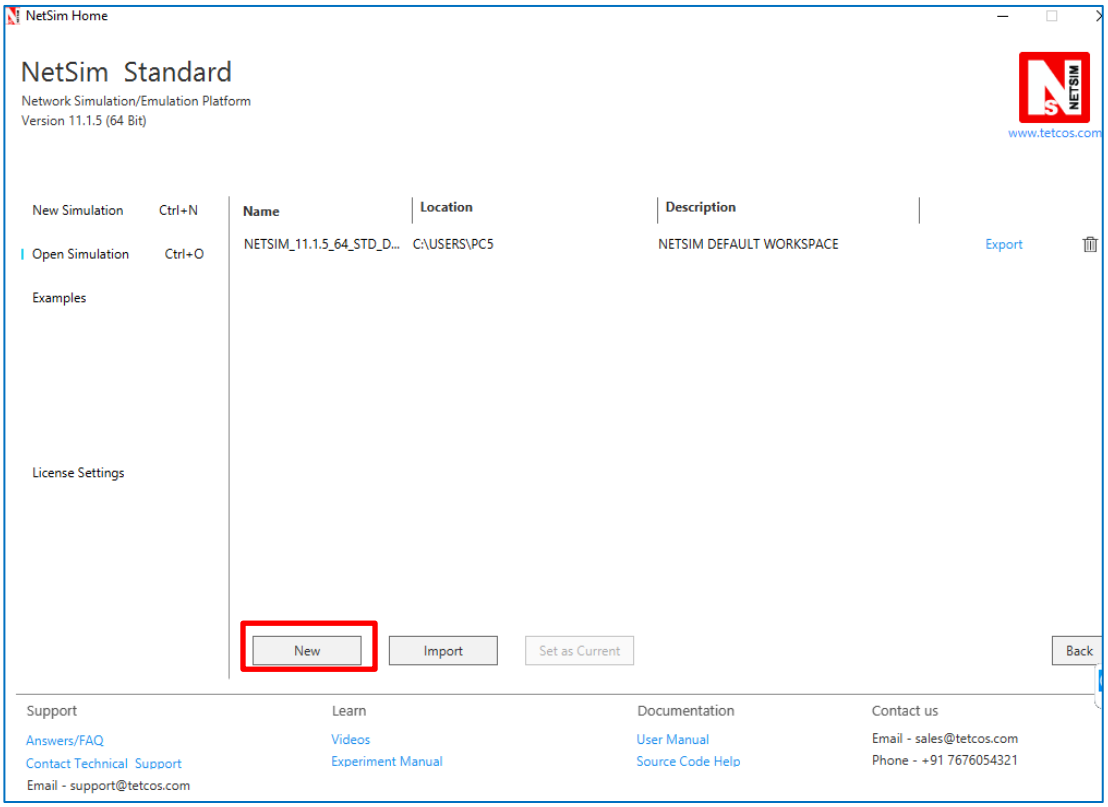
To create a new workspace, click on Wokspace options present in Open Simulation Menu shown below:



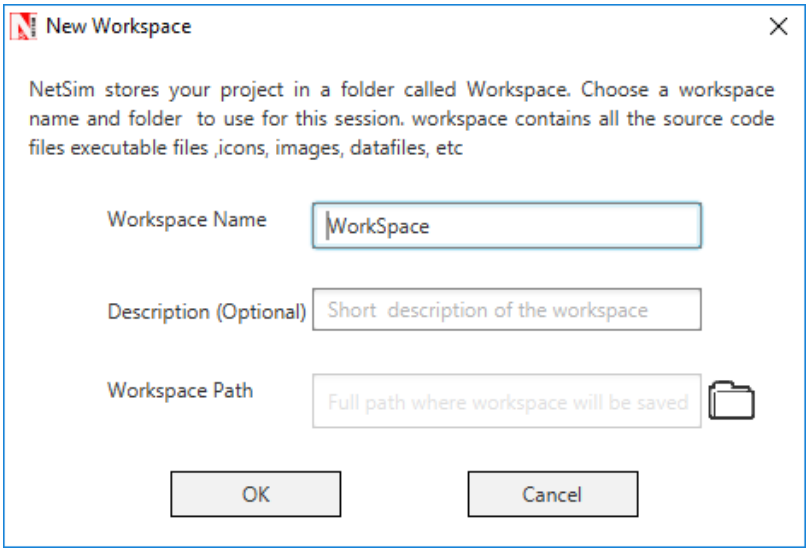
Then select More Options



And select New

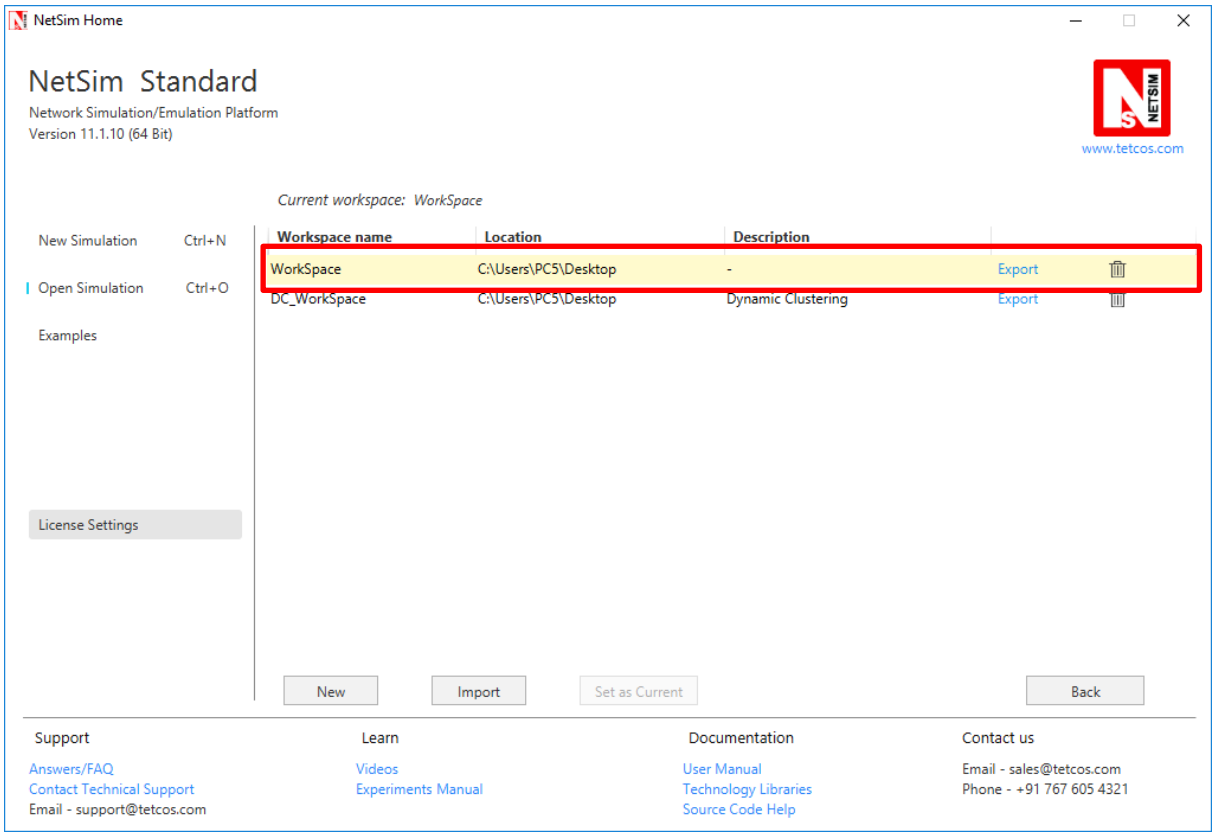


A New Workspace pop-up window appears where you can input the Workspace Name, Description and Workspace Path (where you want to create new workspace).

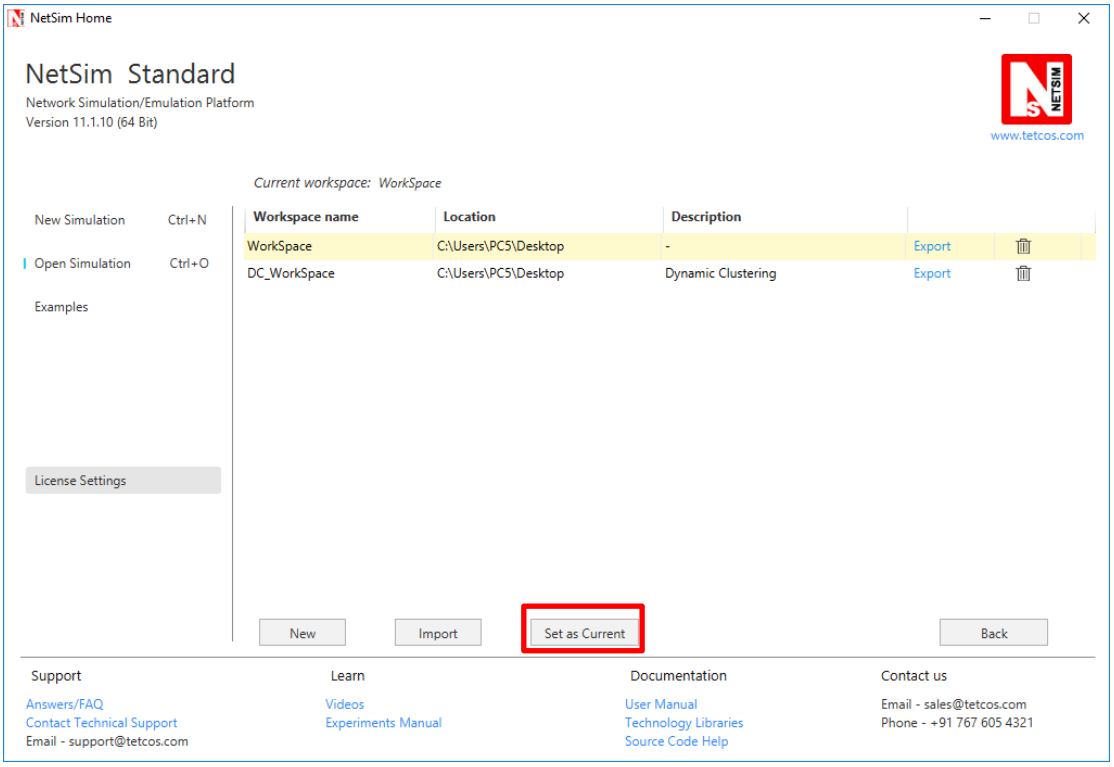


3.7 How does a user switch between workspaces?

Users can also switch from one workspace to another workspace. For doing this, Select Open Simulation->Workspace Options->More Options and click on the workspace you want to switch to.

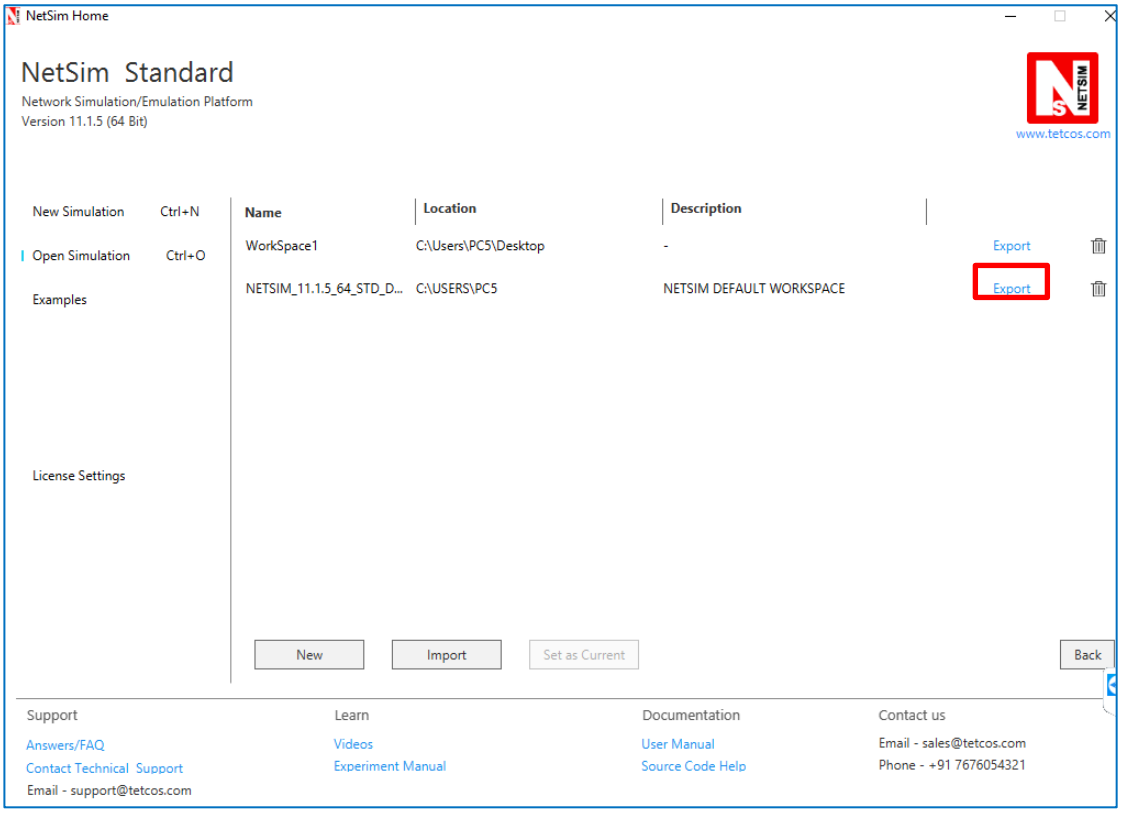


And then select “Set as Current” as shown below:

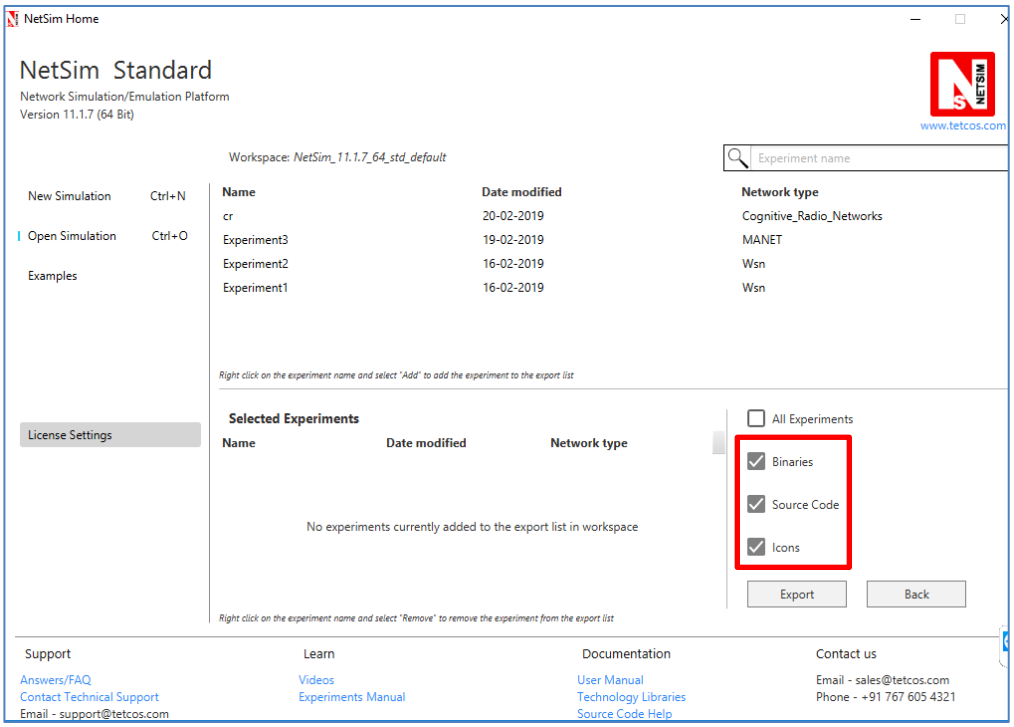


3.8 How does a user export a workspace?

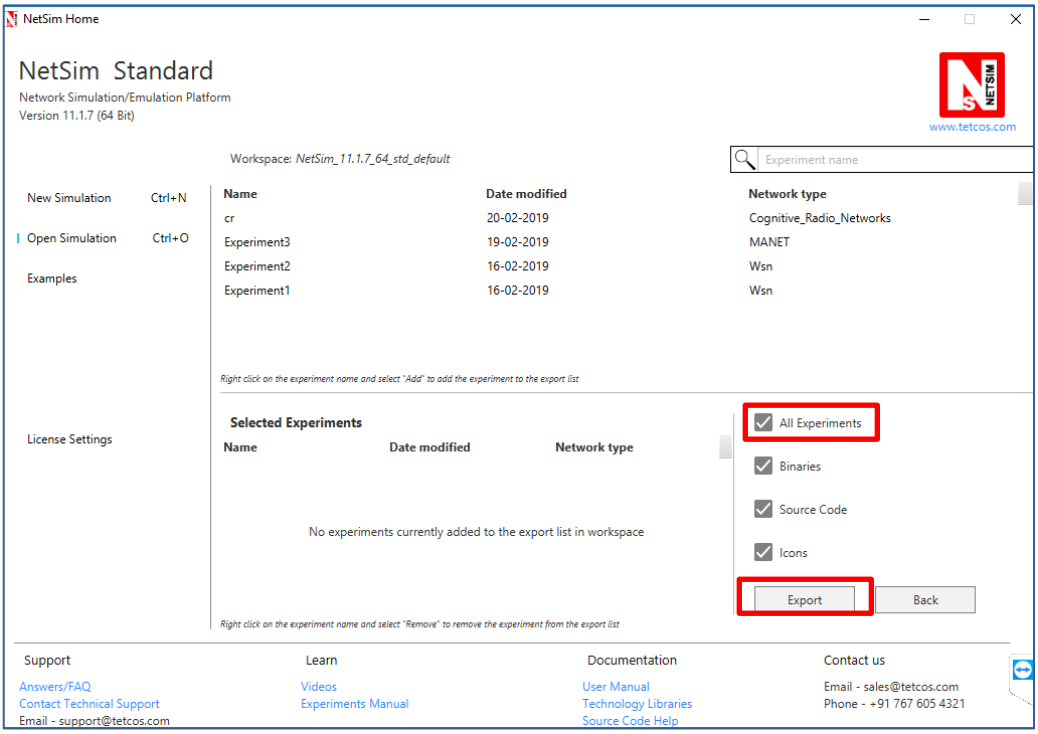
Users can export workspaces by selecting Open Simulation->Workspace Options->More Options. Then select Export as shown below:



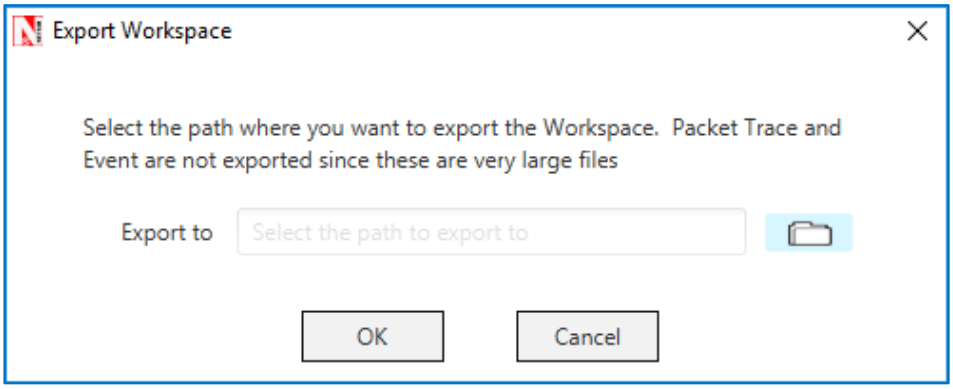
The following window is displayed



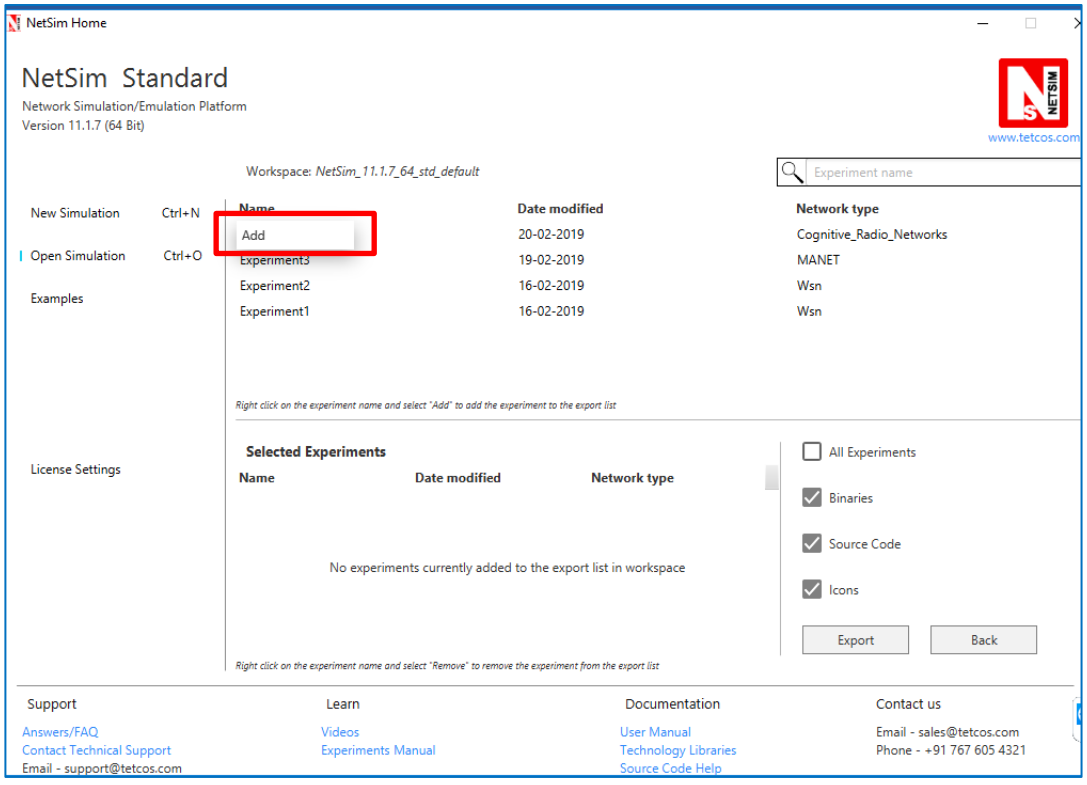
Binaries, Source Code (available only for Standard and Pro) and Icons are added by default as shown above. Users can add all experiments present in the current workspace by clicking on the All Experiments check box and then click on Export as shown below.



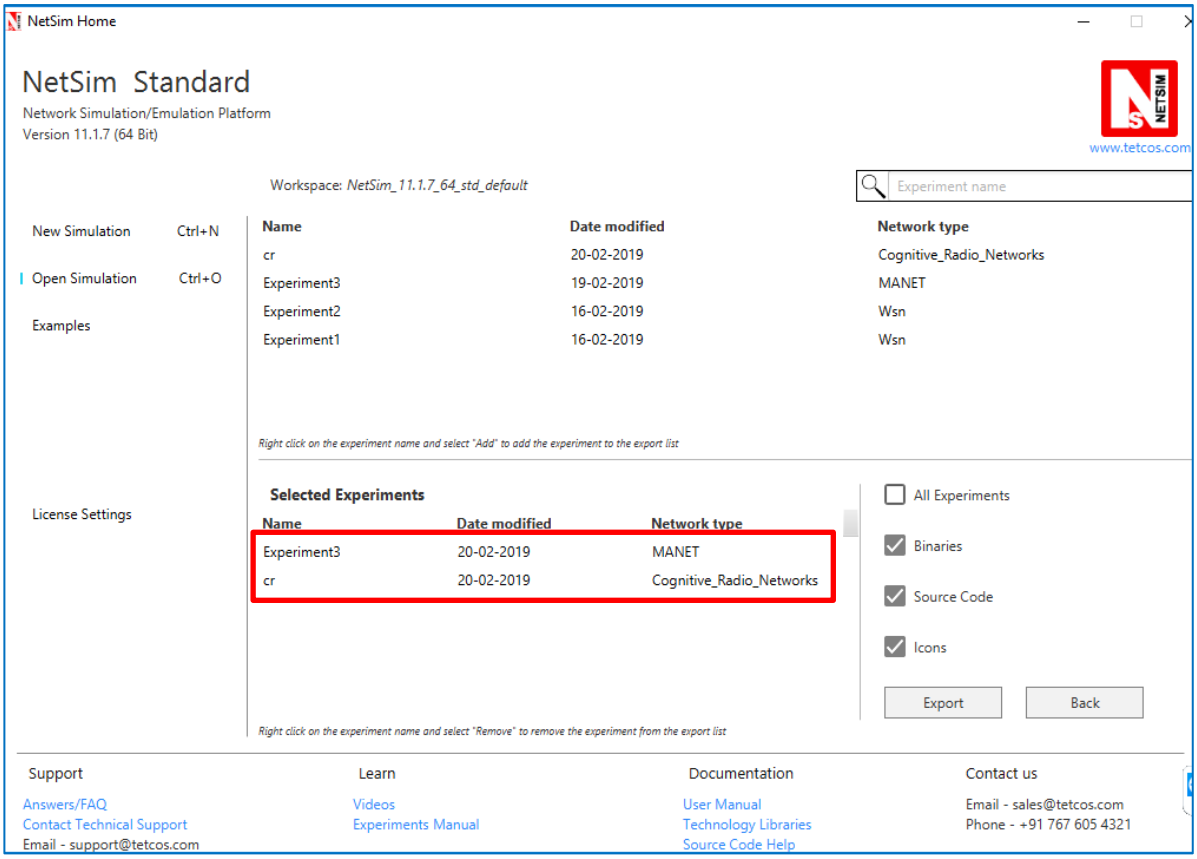
It displays a window shown below where users need to give the path to which the workspace has to be exported and then click on OK.



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



The added experiments will be available in the export list shown below

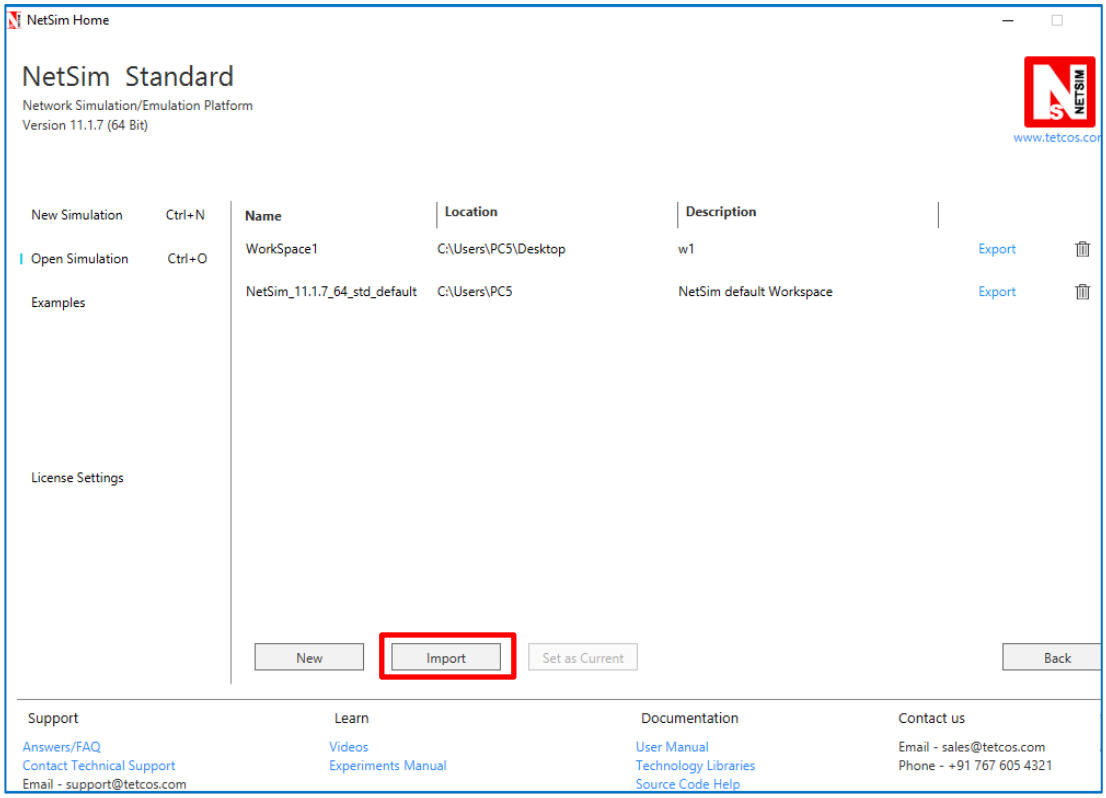


After adding the experiments click on Export and follow the same procedure as explained above

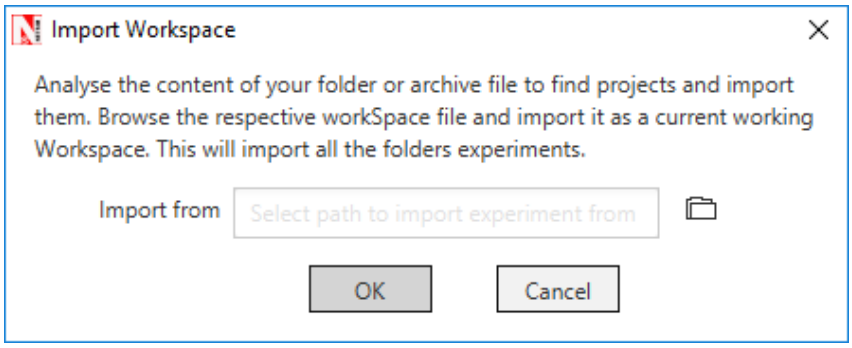
Note: Users can import only the exported workspaces.

3.9 How does a user import a workspace?

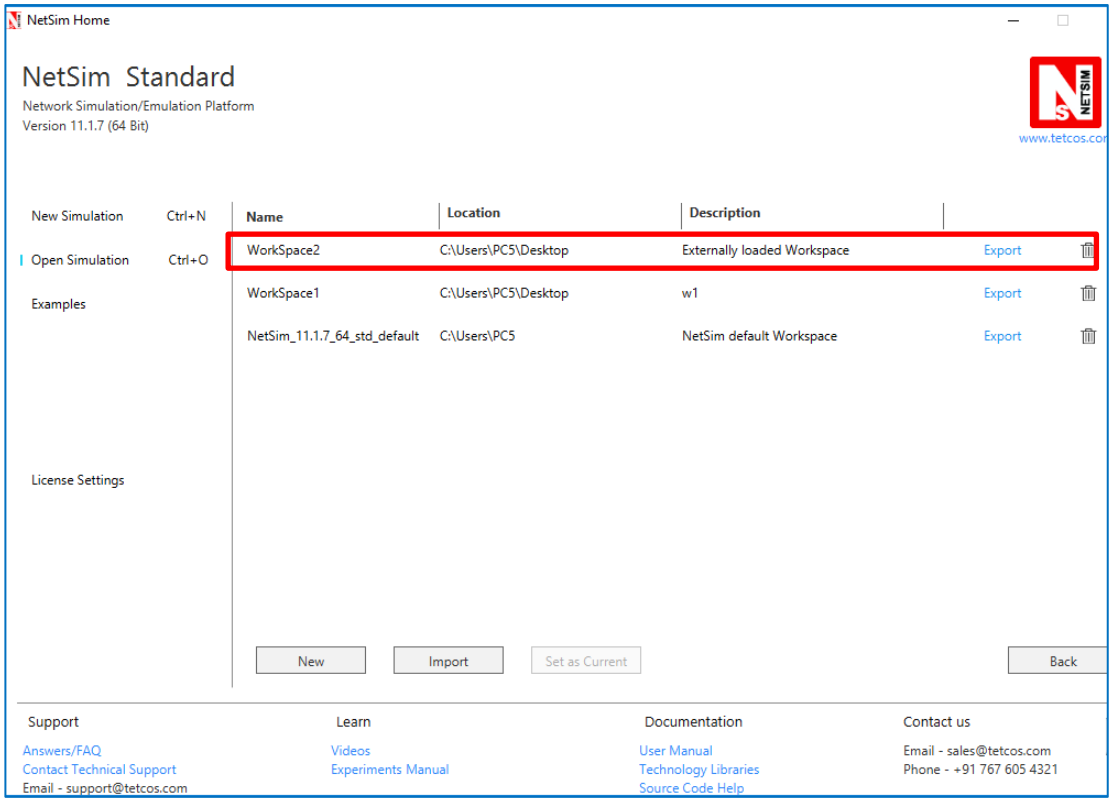
Users can import workspaces by selecting Open Simulation->Workspace Options->More Options. Then select Import as shown below:



It displays a window where users need to give the path of the workspace folder and click on OK as shown below

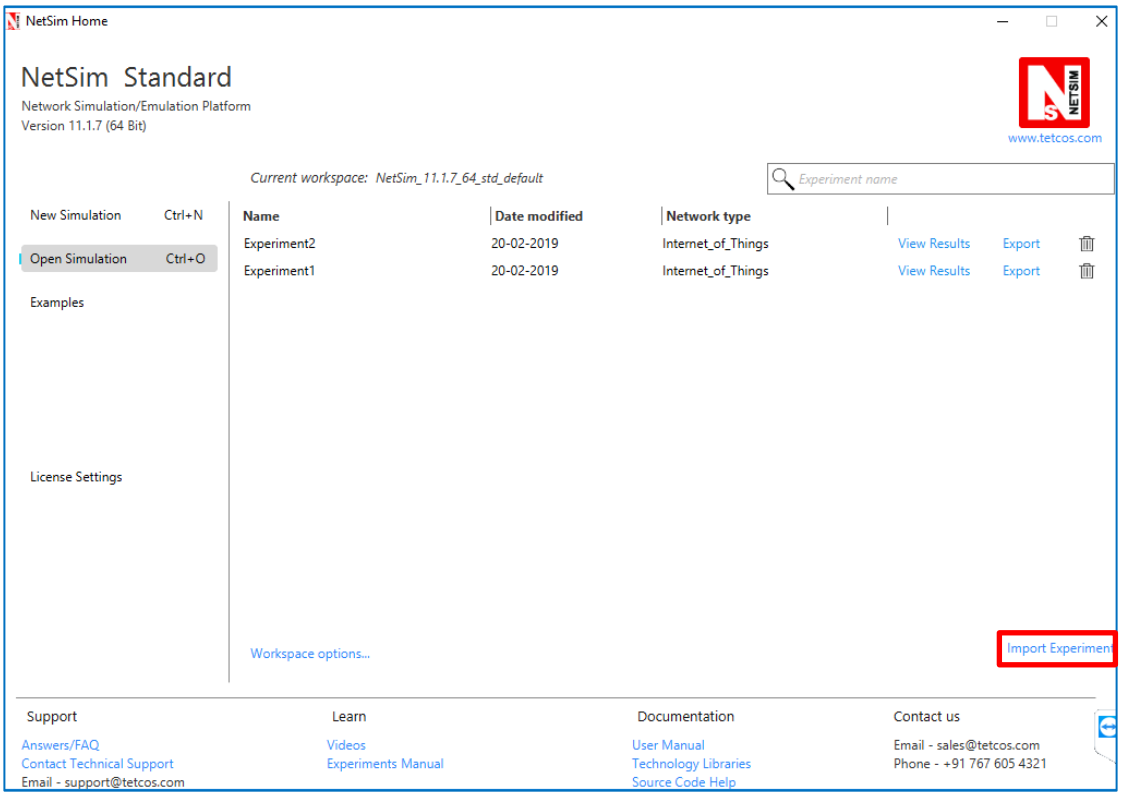


The Imported workspace will be set as the current workspace. To see the imported workspace, click on Open Simulation->Workspace Options->More Options as shown below

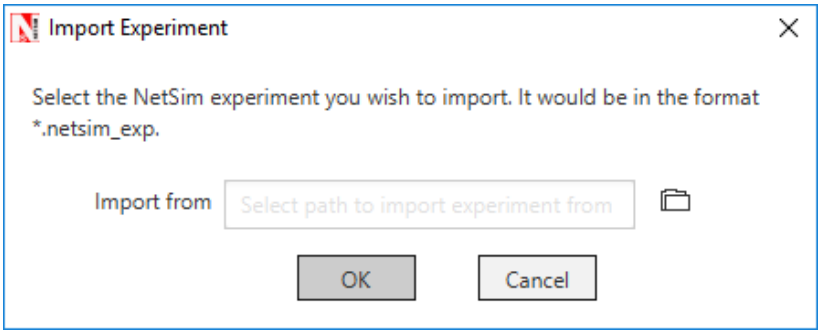


3.10 How does a user import an experiment?

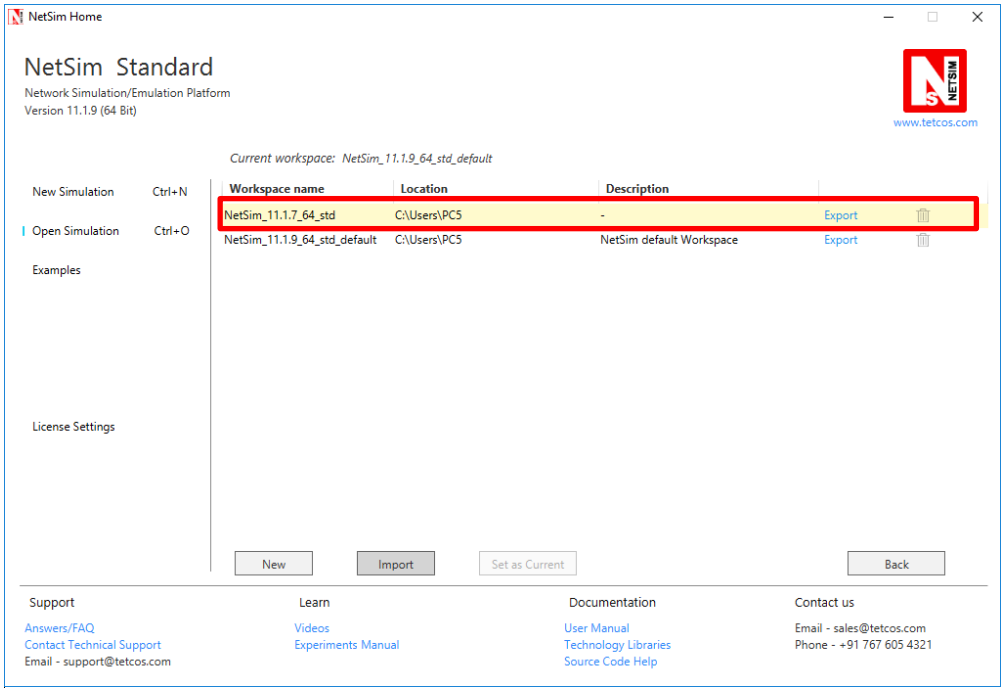
Users can also have option to import experiments. For this click on Open Simulation and then select Import Experiment as shown below:



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



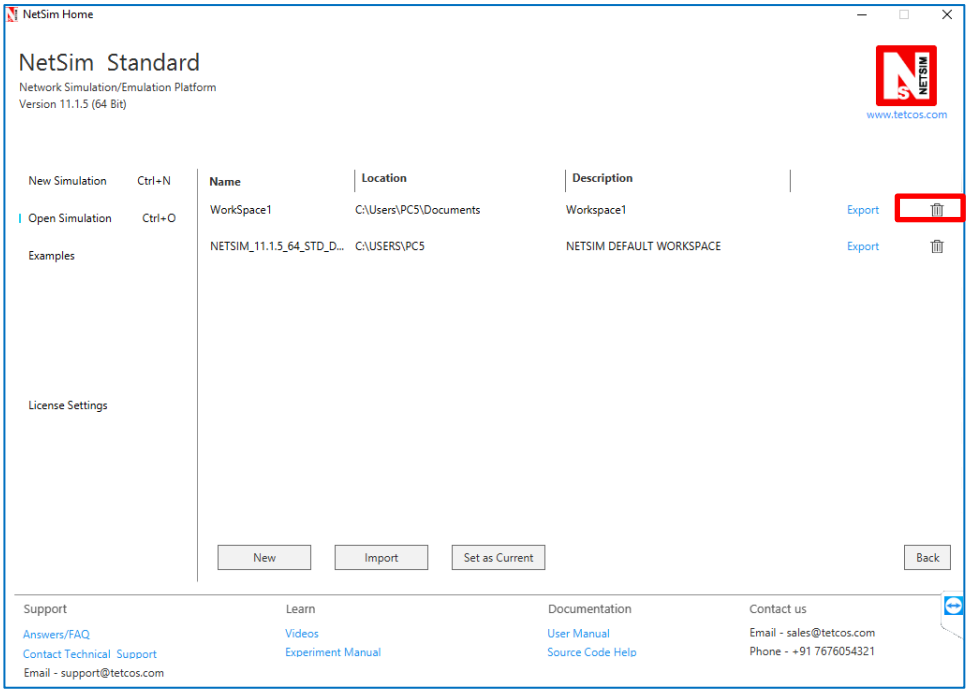
The imported experiment will be available in the current workspace. To see the imported experiment click on Open Simulation as shown below:



Note: Users can import only the exported experiments

3.11 How does a user delete a workspace?

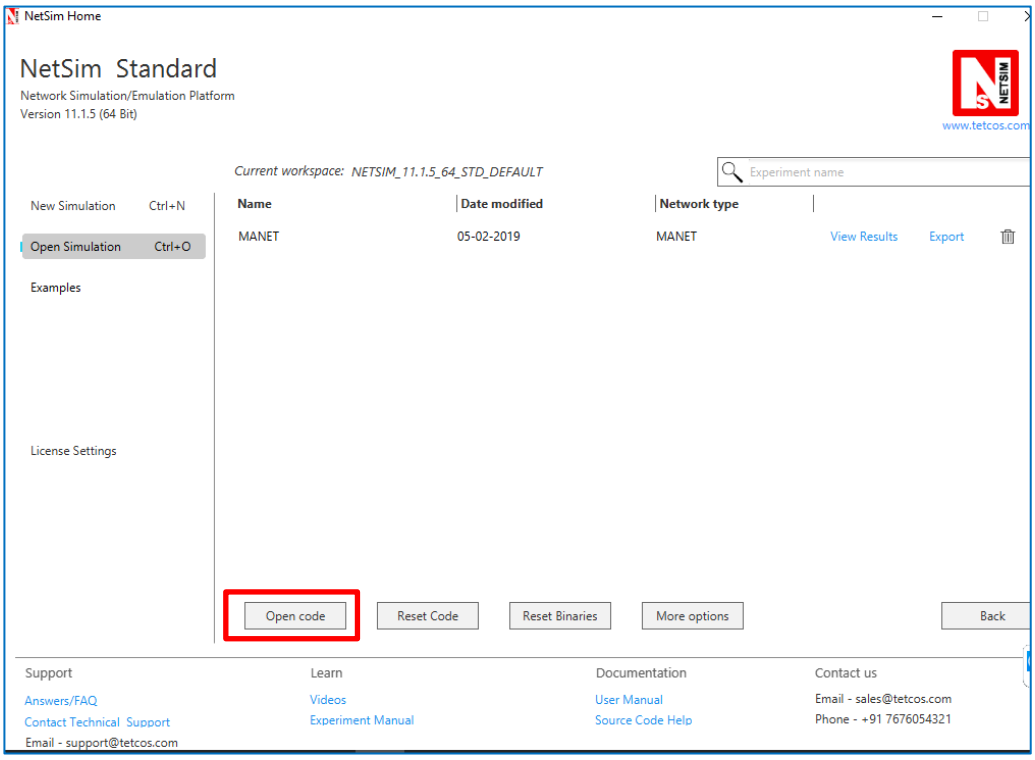
Users can also delete a workspace by clicking on the delete icon shown below:



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

3.12 How does a user open and modify source codes?

User can also modify the source codes within a workspace. For doing this, select open Simulation->Workspace Options->Open Code as shown below.



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.

Note: The changes in the source codes applies to the current workspace only

3.13 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

4 Simulating different networks in NetSim

When you install NetSim, you get access to inbuilt examples to help you understand how the different types of networks work. For information on how to use the NetSim UI to simulate the Examples and create your own simulations, see NetSim User Interface.

The following table lists what versions of NetSim you must use to simulate the networks.

Type of Network	NetSim Versions
Internetwork	All versions
Legacy Network	All versions
Cellular Network	All versions
MANET	All versions
Wireless Sensor Network	All versions
Software Defined Network	All versions
Internet of Things	All versions
Cognitive Radio	All versions
LTE	All versions
VANET	Available only with NetSim Standard and NetSim Pro versions
Military Radio	Available only with NetSim Pro version

4.1 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, 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

4.1.1 Internetworks Examples

To simulate the Examples for different types of Internetworks

1. Go to the NetSim UI and click **Examples**.
   The Example Simulation pane appears at the right.
2. Click the **Internetwork** example you wish to simulate. NetSim UI loads the example.

4.1.2 Internetwork 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 Internetworks in examples. The help documentation explains the following:

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

4.2 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 librarary 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

4.2.1 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.

4.2.2 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:

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

4.3 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 equipments 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.

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

4.3.2 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:

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

4.4 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 – 802.11, TDMA/DTDMA (available in NetSim pro only)
- Single MANET and multiple MANETs using bridge nodes

To simulate MANET, click on New Simulation and then under Mobile Adhoc networks click on either Single MANETs or Multiple MANETs

4.4.1 MANET Examples

To simulate single MANET or multiple MANETs Examples:

1. Go to the NetSim UI and click **Examples**.
   The Example Simulation pane appears at the right.
2. Click the example you want to simulate. NetSim UI loads the example.

4.4.2 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:

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

4.5 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 protcols.

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

4.5.1 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 example you want to simulate. NetSim UI loads the example.

4.5.2 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:

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

4.6 Internet of Things

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

In NetSim, IOT is modeled as a WSN that connects to the internet via a 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

4.6.1 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 example you want to simulate. NetSim UI loads the example.

4.6.2 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:

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

4.7 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 Internworks, MANET, IOT, WSN, Cognitive Radio, LTE or VANETs

4.7.1 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 example you want to simulate. NetSim UI loads the example.

4.7.2 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 Degined Network examples. The help documentation explains the following:

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

4.8 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

4.8.1 Cognitive Radio Examples

To simulate Cognitive Radio Examples:

3. Go to the NetSim UI and click **Examples**.
   The Example Simulation pane appears at the right.
4. Click the example you want to simulate. NetSim UI loads the example.

4.8.2 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:

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

4.9 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 along with LTE FemtoCell, LTE D2D and LTE VANETs (which requires VANET library)

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 a suitable option under Long Term Evolution Networks.

4.9.1 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 example you want to simulate. NetSim UI loads the example.

4.9.2 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:

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

4.10 VANETs

Vehicular Ad-Hoc Network (VANET) is a sub form 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

4.10.1 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 example you want to simulate. NetSim UI loads the example.

4.10.2 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:

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

4.11 Military Radio (TDMA/DTDMA)

Military communications or military signals involve all aspects of communications to convey information in armed forces. Military Communications include text, audio, facsimile, tactical ground-based communications, terrestrial microwave, tropospheric scatter, naval, satellite communications systems and equipment, surveillance and signal analysis, encryption and security and direction-finding and jamming.

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

To simulate Military Radios click on New Simulation, then click on MANET (Either Single or Multiple) and select TDMA/DTDMA in MAC/PHY layer of the devices.

4.11.1 Military Radio Examples

To simulate Military Radio Examples:

3. Go to the NetSim UI and click **Examples**.
   The Example Simulation pane appears at the right.
4. Click the TDMA/DTDMA example you want to simulate. NetSim UI loads the example.

4.11.2 Military Radio (TDMA/DTDMA) 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/DTDMA examples. The help documentation explains the following:

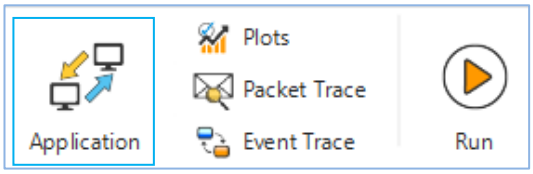
- Simulation GUI
- Model Features
- Featured Examples
- Reference Documents
- Latest FAQ available online

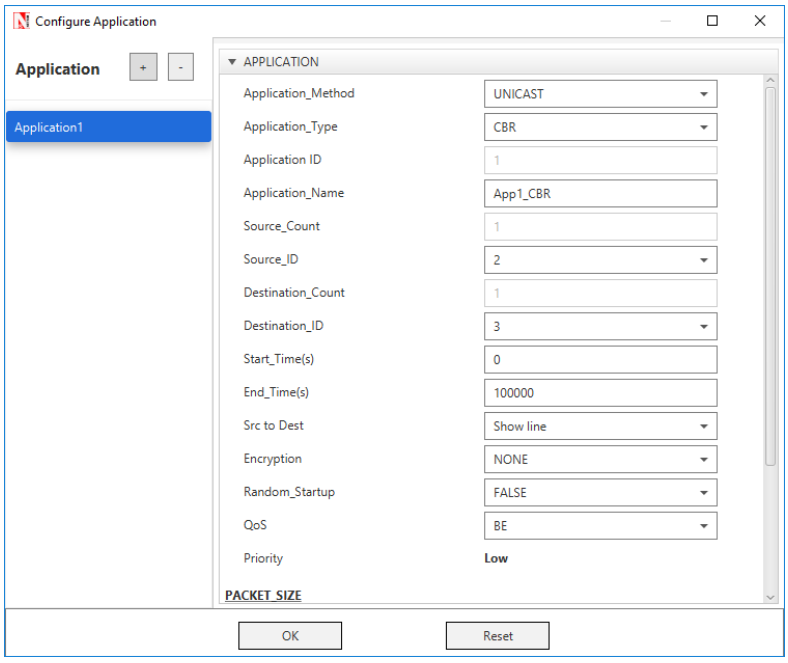
5 Traffic generator (Application models)

NetSim allows users to model and simulate the applications:

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

To set up the application click on the application icon from the tool bar as shown below.





This properties window allows you to model the application traffic. You can add (or) delete one or more applications.

5.1 Common properties for all the traffic types

Application Method: It specifies the type of Application method either 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, Emulation, BSM and Emulation.

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

Application Name: It specifies the name of the application

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

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

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

Destination ID: This property represents the unique identification numbers of the destination. And to model, **Broadcast** applications users can select '0' as the Destination ID.

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.

Note: Suppose Start time is 1 and end time is 10 then application starts generating traffic at 1st second and ends at 10th second.

Encryption: Encrypts the application

Random Startup: If random start up is set true, application will start at +1s. 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.

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

Priority: In a complex network, packets may reach a router from many directions. Priority scheduling algorithm will allow the router to fix priority level for different sources from different directions. Higher priority packets are processed first and sent out.

5.2 Application Types

Brief explanation of application types

Application Type	Properties	Units	Description
CBR – Constant bit Rate	**Packet size** (Constant distribution) – It is the size of the packet	bytes	Packets of constant size are generated at constant inter arrival times.
	Inter Arrival Time (Constant distribution) – It is the gap between two successive packets	µs	
Custom	**Packet size** (Constant, Exponential distribution) – It is the size of the packet	bytes	It is User defined application that can be configured based on user requirements

	Inter Arrival Time (Constant, Exponential distribution) – It is the time gap between two successive packets	µs	
Peer to Peer	**File size distribution** (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. **Ex** – Torrent, Limewire etc.
	Value – Size of the file	bytes	
	Piece size - Each file is divided into equal sized pieces. This property represents the size of each piece	bytes	
Email	**Email send/receive** – Represents the rate at which emails are sent/receive	-	Allows users to send/receive email application **Ex** – Outlook, Apple mail, Gmail etc.
	Duration (Constant, Exponential distribution) - Time between two successive emails	Seconds	
	Email size(Constant, Exponential distribution) – Size of an email	Bytes	
HTTP – Hyper Text Transfer Protocol	**Inter Arrival Time** (Constant, Exponential distribution) – It is the time gap between two successive HTTP requests	seconds	HTTP is a protocol that utilizes TCP to transfer its information between computers (usually Web servers and clients). Hence in NetSim, it is imperative that TCP is enabled in the Source Node. **Ex** - In the URL `http://www.nowhere123.com/docs/index.html`, the communication protocol is HTTP; the hostname is `www.nowhere123.com`. The port number was not specified in the URL, and takes on the default number, which is TCP port 80 for HTTP. The path and file name for the resource to be located is "`/docs/index.html`".
	Page size (Constant, Exponential distribution) – It is the size of each page	bytes	
	Page count – Represents the number of pages	-	
COAP – Constrained Application Protocol	**Inter Arrival Time** (Constant, Exponential distribution) – It is the time b//w two successive COAP requests	seconds	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
	Page size (Constant, Exponential distribution) – It is the size of each page	bytes	

	Response time – It is the time taken by a device to generate response	ms	
	Multicast response – Represents the server responds to multicast response or not	-	
	NSTART – Limit the number of simultaneous outstanding interactions that a client maintains to a given server	-	
	DEFAULT_LEISURE – This setting is only relevant in multicast scenarios, outside the scope of the EST-coaps draft	-	
	PROBING_RATE: A parameter which specifies the rate of re-sending Non-confirmable messages.	-	
	Ack required – It represents whether the ack for the request/response to be sent or not	-	
FTP – File Transfer Protocol	**File size** (Constant, Exponential distribution) – It is the size of the file	bytes	It is a standard network protocol used for the transfer of files between a client and server **Note**: Devices must have TCP enabled in Transport layer for implementing FTP application successfully. **Ex** – Filezilla
	File Inter Arrival Time – It is the gap between two successful files	seconds	
Database	**Transaction size** (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 **Ex** – MS Excel, MySQL etc.
	Transaction Inter Arrival Time (Constant, Exponential distribution) – It is the time gap between two successful transactions	µs	
Voice	**Packet size** (Constant, Exponential) – It is the size of the packet	bytes	It allows users to configure voice application between client and server **Note** – Distribution is constant only for all codec types except custom
	Packet Inter Arrival Time (Constant, Exponential distribution) - It is the gap	µs	

	between two successful packets		**Ex** – Skype, Team Viewer, Google Voice etc.
	Service type – CBR, VBR	-	
	Suppression models available for VBR – Deterministic, Markov chain	-	
	Success ratio - Sets the ratio of the packets that are not silenced during VBR calls	%	
Video	**Model Type** - 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 **Ex** – Skype
Erlang Call	**Packet size** (Constant, Exponential distribution) – It is the size of the packet	bytes	The erlang is a unit of traffic density in a telecommunications system. One erlang is the equivalent of one call **Note** – Distribution is constant only for all codec types except custom
	Packet Inter Arrival Time (Constant, Exponential distribution) - It is the gap between two successful packets	µs	
	Call duration (Constant, Exponential distribution) – It is the duration of each call	seconds	
	Call Inter Arrival Time (Constant, Exponential distribution) - It is the gap between two successful calls	seconds	
	Service type – VBR, CBR	-	
	Suppression model available for VBR – Deterministic, Markov chain	-	
	Success ratio - Sets the ratio of the packets that are not silenced during VBR calls	%	
Sensor App	**Packet size** (Constant distribution) – It is the size of the packet	bytes	Used to create application between two sensors **Ex** – Smart home, Smart water etc.
	Packet Inter Arrival Time (Constant distribution) - It is the gap between two successful packets	µs	

BSM – Basic safety message	**Packet size** (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 Messages (BSMs). The BSM is a 20-byte packet that is generally broadcast from every vehicle at a nominal rate of 10 Hz. **Note** - Available only with VANET component **Ex** – Traffic management
	Packet Inter Arrival Time (Constant, Exponential distribution) - It is the gap between two successful packets	µs	
Emulation	**Source Real IP** - Specifies the real IP Address of source device in Emulation	-	NetSim Emulation application enables users to connect NetSim simulator to real hardware and interact with live applications. **Note** 1. Will be present only when Emulator Add-on is installed. 2. If user wants to modify application.dll and run emulation then application.dll must be built in release mode only. If built in debug mode then emulation won't work.
	Source Port - Specifies the Port no used for transmission by Application running in source device		
	Destination Real IP - Specifies the real IP Address of destination device in Emulation		
	Destination Port - Specifies the Port no used for reception by Application running in destination device		

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.

- μ-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 CUSTOM application type as explained 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 – 50.

- **Pixels per frame** -Number of pixels in each frame. This is in the range of 10000 – 100000.
- **Bits per pixel (μ)** – Mean value of the normal distribution used to generate the value of bits per pixel.
- **Bits per pixel (Σ)** – 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 (μ) = 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 λ(n) can be generated by the recursive relation λ(n) = aλ(n-1)+bw(n).
    - **Eta**– The steady-state average E(λ)and discreet auto covariance C(n) are given by E(λ) = (b / (1-a)) η C(n)=(b2/(1-a2))an where η 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 λ(t) with discrete jumps at random Poisson time.
    - **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.
    - **No of Multiplexed Sources**– This model considers the aggregate instantaneous input rate λN (t) instead of the single source bit rate λ (t). The

total rate is the sum of N independent random processes each with mean E (λ) and variance C (0) at steady state. Therefore, the steady state- mean of λN (t) will be E (λ N) =N x E (λ) 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 IBBPBBPBBPBBIBBPBB… i.e., 12 frames in a Group of Pictures (GOP). Generate X0 from a Gaussian distribution N(0, y 0).Set initial value N0= 0, D0 = 1.

For k = 1, 2,…, N-1, calculate Φkj , j = 1, 2,…,k iteratively using the following formulae

$$Nk = r(k) – j=1\Sigma k-1\Phi k-1,j\ r(k-j)$$

$$Dk = Dk-1 –(N2k-1/Dk-1)$$

$$\Phi kk = Nk / Dk$$

$$\Phi kj = \Phi k-1,\ j-\Phi kk\Phi k-1,k-j\ j=1,…..,k-1$$

$$mk = j = 1\Sigma k\Phi kjXk-j$$

$$y\ k = (1- \Phi 2kk)\ yk-1$$

Finally, each Xk is chosen from N (mk, y k). 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-β 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,

$$B = 2 – 2H$$

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

Yk = F-1β (FN(Xk)), k>0

Xk: Self-similar Gaussian process,

FN: The cumulative probability of normal distribution,

F-1β: 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.

5.3 Generation Rate for Different Applications

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

CBR and Custom application

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

Example: Packet size = 1460Bytes and Inter arrival time = 20000μs.

Generation rate (Mbps) = 1460*8/20000 = 0.584Mbps

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:

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

where, fps = frames per second

ppf = pixel per frame

bpp (μ) = bits per pixel (mean)

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

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

Generation rate (bps) = fps*ppf*bpp

= 20*10000*0.52 = 104000 bits per second = 0.1040 Mbps

Voice

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

Note – Distribution is constant for all codec types except custom

Email

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

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

Generation rate (Mbps) = 20000*8/1 = 0.16Mbps

HTTP

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

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

Generation rate (Mbps) = 20000*8*2/3 = 0.106Mbps

FTP

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

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

Generation rate (Mbps) = 100000*8/5 = 0.16Mbps

Database

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

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

Generation rate (Mbps) = 10000*8/1000000 = 0.08Mbps

BSM

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

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

Generation rate (Mbps) = 20*8/1000000 = 0.00016Mbps

5.4 Priority and QoS of Applications

The various application traffic generated in NetSim have the following priority and QoS values:

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

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

5.5 Modelling Poisson arrivals in NetSim

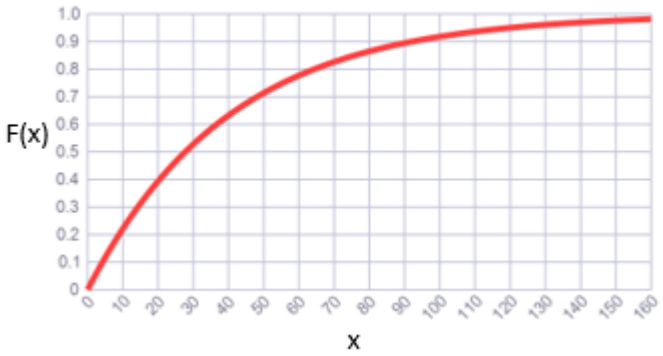
What's a Poisson process, and how is it useful?

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 are randomly generated throughput the hour throughput the 60 seconds, 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 λ = 1/ 0.05 = 20 and call it the *rate parameter*. The rate parameter λ is a measure of frequency: the average rate of events (packets) per unit of time (in this case, seconds).

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

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



Basically, the more time passes, the more likely it is that, a packet is generated. The word "exponential", in this context, actually 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 $F(1) \approx 0.999999998$

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 pretty easy to write this analytically, where U is the random value between 0 and 1:

$$\text{Next Time when a packet is generated} = -\ln(1 - \text{RandNo}) / \lambda$$

This is exactly the code used in NetSim, and this is available in the source C file in ../NetSim_Standard/Simulation/Application/Distribution.c. In the case exponential distribution, you would see

```
case Distribution_Exponential: /*Exponential Distribution Function*/
```

```
fFirstArg = args[0];

nRandOut = fnRandomNo(10000000, &fRand, uSeed, uSeed1);

fRandomNumber = (double) (fRand);

fFirstArg = 1 / fFirstArg;

*fDistOut = (double) -(1 / fFirstArg)* (double) logl(1 - fRandomNumber);
```

The simple way of selecting this via the UI is to selecting exponential distribution for inter-arrival time when modelling application properties.

6 Running Simulation via CLI

6.1 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\NetSim_11.1.7_64_std_default\bin\bin_x64" for 64-bit and "C:\Users\PC\NetSim_11.1.7_32_std_default\bin\bin_x86" for 32-bit. For more information on NetSim workspace refer Section 3 "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\Ram\AppData\Local\Temp\NetSim"

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 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 installed 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\NetSim_11.1.7_64_std_default\bin\bin_x64"

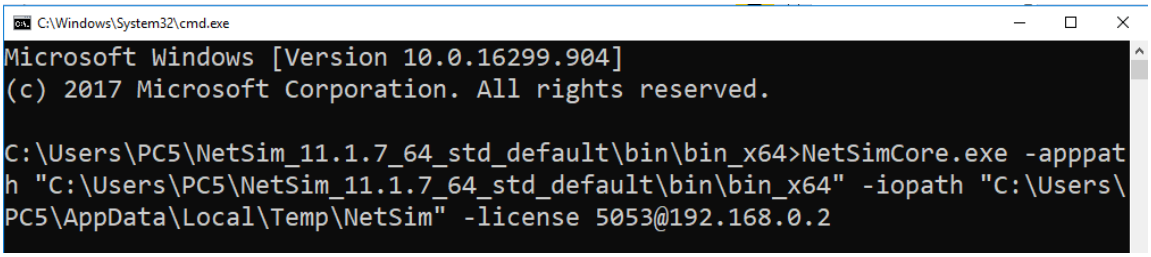
For floating/roaming licenses, type the following in the command prompt.The type of license can be seen by clicking on NetSim Help → About NetSim

>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 and Configuration.xsd (Configuration.xsd is available in the bin folder of NetSim's current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_32_std_default\bin\bin_x86> for 32-bit folder). Refer section 6.2.4 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. 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 192.168.0.2



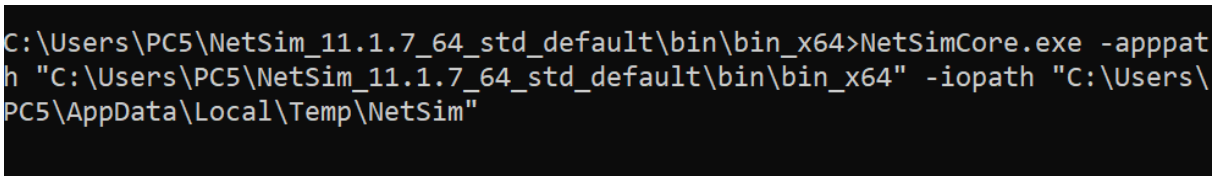
For node-locked licenses, type the following in the command prompt

>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>

Where,

- <app path> contains all files of NetSim including NetSimCore.exe
- <iopath> contains Configuration.netsim and Configuration.xsd

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



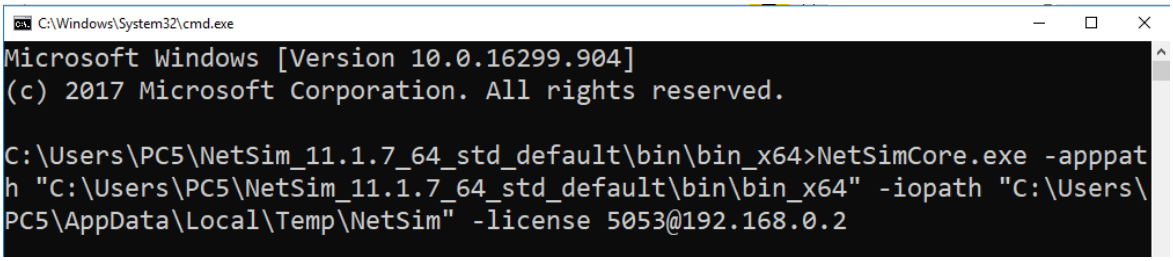
Simulation will be completed successfully and the text files that are requested by the end user in Configuration.netsim will be written in the <iopath>.

Note: 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 as given below.

```
Executing command --- DEL "C:\Users\Santhosh\AppData\Local\Temp\NetSim\*.pcap"
Could Not Find C:\Users\Santhosh\AppData\Local\Temp\NetSim\*.pcap
Emulation is disabled
Applications created


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

100 % is completed... Simulation Time = 10000.000 ms  Event Id=43935
Total time taken (wall clock) = 906 ms
Total events processed = 43935
Average events per sec (wall clock) = 48493.38
Simulation complete
***


Waiting for Graph to complete...

Graph plotting completed...
Network metrics calculations complete
Protocol specific metrics calculations complete
Protocol binaries freed
Stack memory freed

***

NetSim end

If you are running via CLI go to IO path to view NetSim metrics.
If you are running via UI, you can view NetSim performance metrics in the UI
Press any key to continue....
```

>cd <app path>

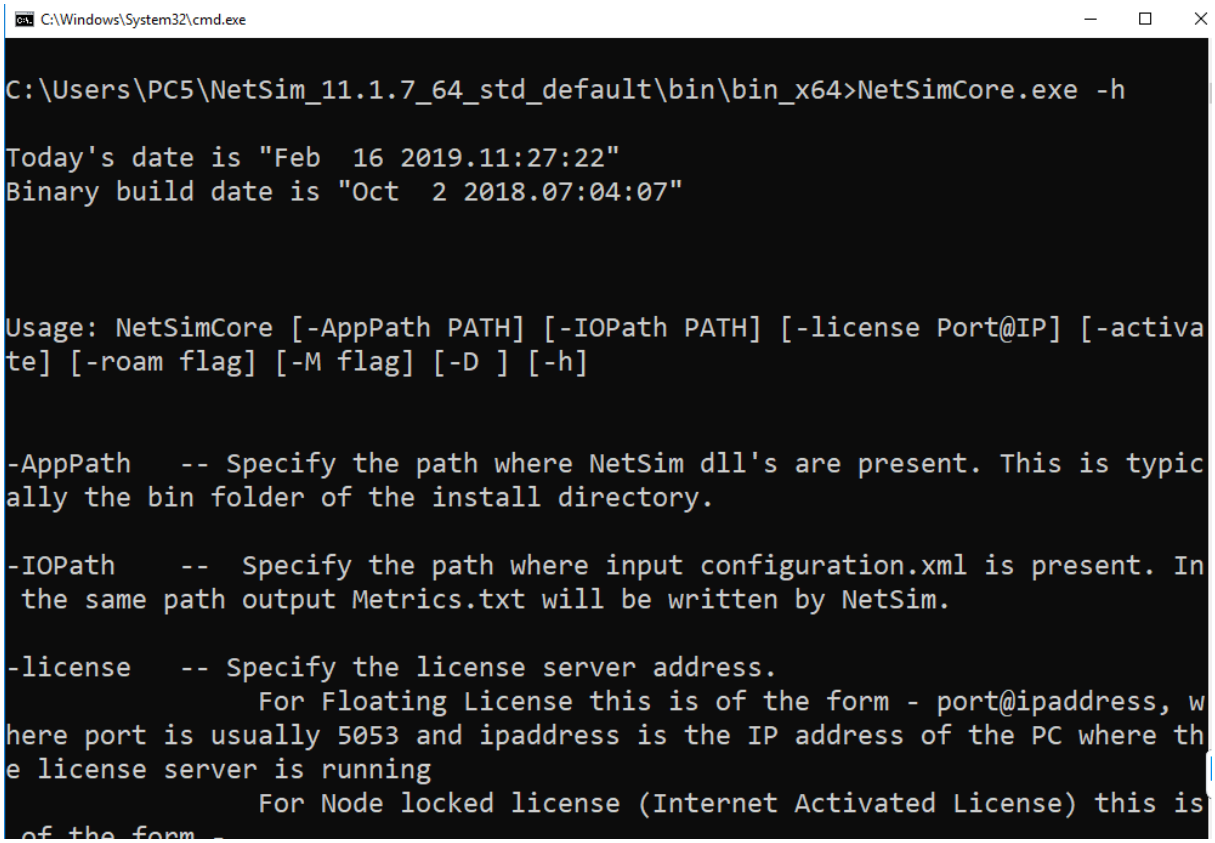
>NetSimCore.exe -apppath <"app path"> -iopath <io path>-license 5053@<License Server IP Address>



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



Running CLI via Quick edit 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 select ok.

6.2 Understanding the Configuration.netsim file

Let's see under the hood to know how NetSim is working

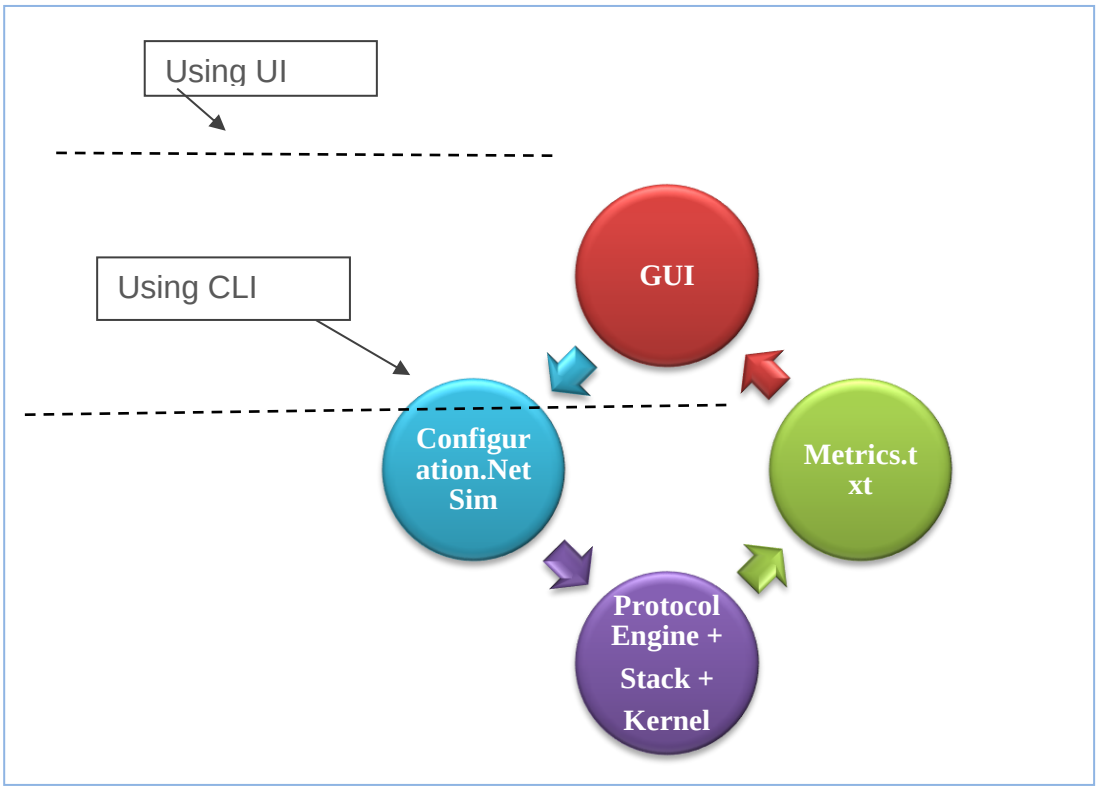


Fig.1. NetSim Architectural Overview

To model a scenario in order to generate metrics in NetSim, GUI will write all the details about the devices used in the scenario and its properties, the links used and their properties, the properties of the environment being used, etc. in ***Configuration.netsim*** just when the user performs the simulation.

The back-end engine that contains dlls and NetSimCore.exe will read this Configuration.netsim, execute the simulation and write output metrics files (in .txt format) to the IO path. Then, the GUI will display the metrics based on the text files written by the backend.

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

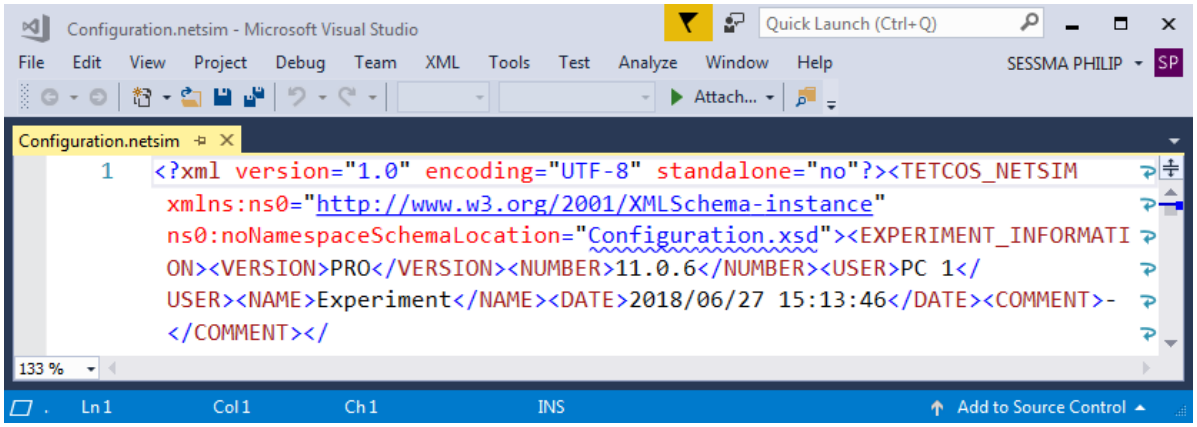
6.2.1 How to use Visual Studio to edit the Configuration file?

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

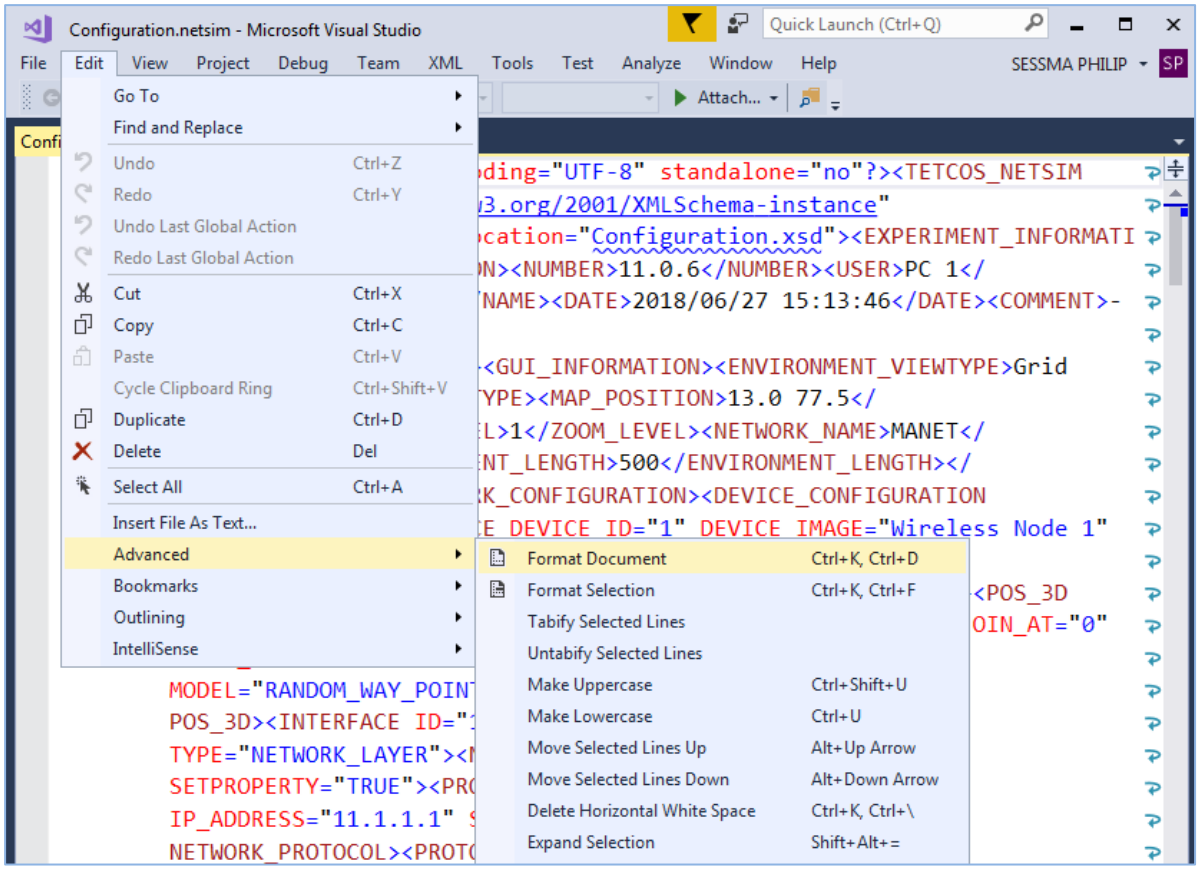
presented with a list of attributes that the element supports. This is known as “IntelliSense”. Using this feature, you can select the options that are required to create the desired scenario.

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.



To reformat click on edit→Advanced→Format Document



6.2.2 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 the 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.

6.2.3 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 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.

6.2.4 Configuration.xsd file

Configuration.xsd is a XML schema Definition file which is present in the bin folder of NetSim’s current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_32_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.

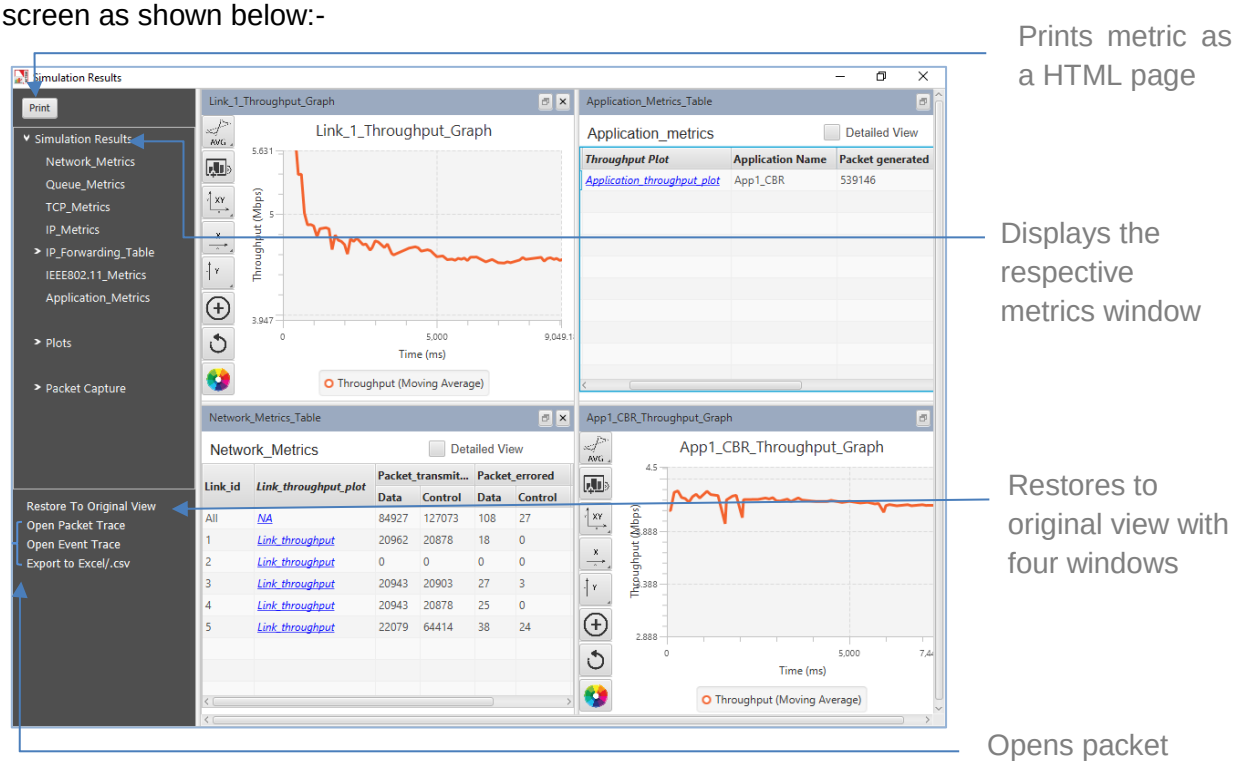
It is not mandatory to place the configuration.xsd file along with the Configuration.netsim file in the iopath. But if it is done, then it will be easier to check & validate changes that are done to the Configuration.netsim file.

7 Results and Analysis

7.1 Result Window

NetSim provides performance metrics at various abstraction levels such as Network Metrics, Queue metrics, TCP Metrics, Application Metrics, etc., at the end of simulation. With the help of metrics, users can analyze the behavior of the modeled network and can compare the impact of different algorithms on end-to-end behavior.

After simulation of a scenario is performed, NetSim Performance Metrics are shown on the screen as shown below:-



The Performance metrics is divided into sections

Network 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

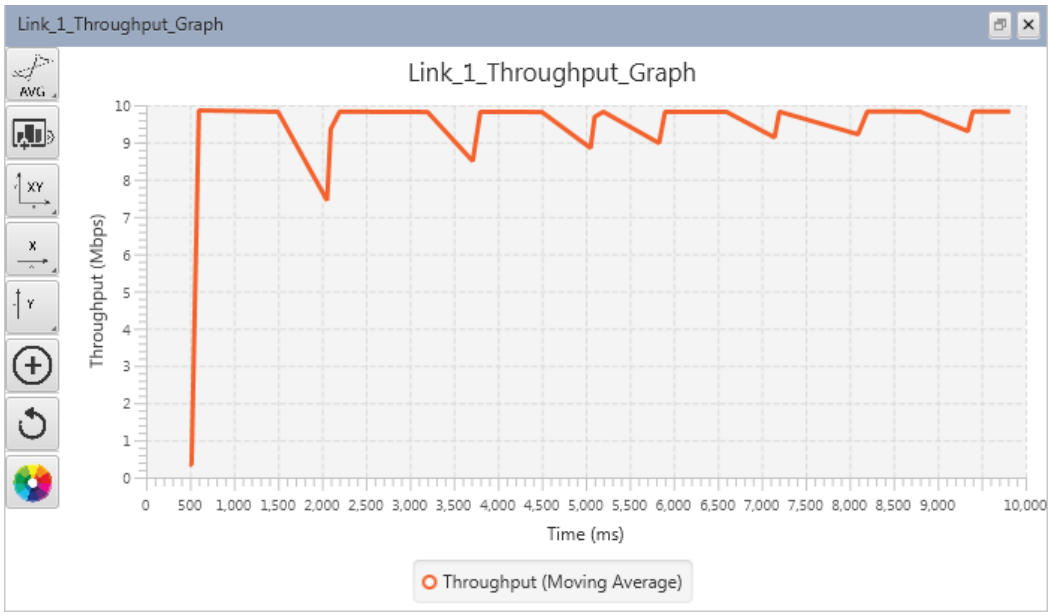
- **Link_ Id**-It is the unique Id for the link.
- **Link_ throughput_ graph –** Plots throughput vs. Simulation time

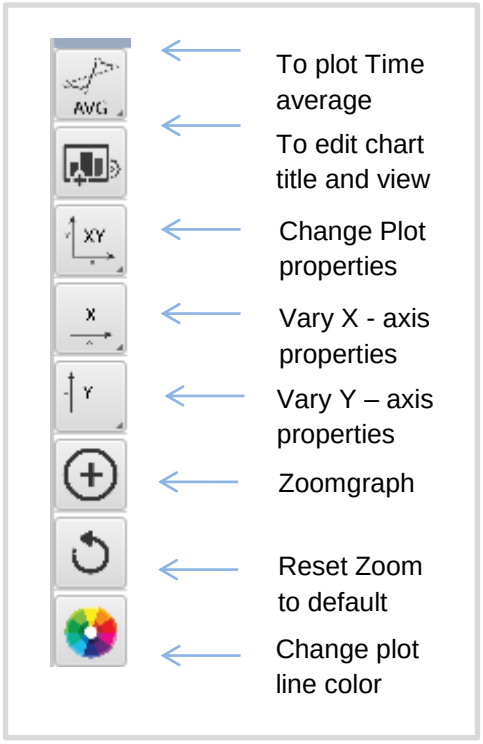
Calculation:

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

Total Bytes transmitted is counted for both 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.





Moving Average: This is the average of the metric up until the current time and is defined as

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

Time Average: This is the average of the metric up to end of simulation current time

- **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_ errored** - Total number of packets error in the link inclusive of data and control packets.
- **Packets_ collided** - Total number of packets collided in the link including data and control packets.
- **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.

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.

Protocol metrics: Displays the protocol based metrics which are implemented in Network scenario. Metrics will vary depending upon the type of network simulated.

TCP Metrics: Displayed if TCP is enabled in any of the devices

- **Source** - It displays the name with ID of the source device which generates TCP packets
- **Destination** - It displays the name with ID of the destination device which receives TCP packets
- **Local Address** - It displays the local IP address with port number of the device present in source column
- **Remote Address** It represents the remote IP address with port number for the source and destination

- **Syn Sent** - It is the number of syn packets sent by the source
- **Syn Ack sent** - It is the number of syn ack packets sent by the destination
- **Segment sent** - It is the number of segments sent by a source
- **Segment received** - It is the number of segments received by a destination
- **Segment retransmitted** - It is the number of segments retransmitted by the source
- **Ack sent** - It is the number of acknowledgements sent by a source to destination in response to TCP syn ack and the number of acks sent by destination to source in response to the successful reception of data packet
- **Ack received** - It is the number of acknowledgements received by source in response to data packets and the number of acks received by destination in response to syn ack packet
- **Duplicate segment received** - It is the number of duplicate segments received by destination
- **Out of order segments received** - It is the number of out of ordered packets received by destination
- **Duplicate ack received** - It is the number of duplicate acknowledgements received by source
- **Times RTO expired** - It is the number of times RTO timer expired at source

UDP Metrics: Displayed if UDP is enabled in any of the devices

- **Device Id** - It is the Id of a device in which UDP is enabled
- **Local Address** - It represents the IP address with port number of the local device (either source or destination)
- **Foreign Address** - It represents the IP address with port number of the remote device (either source or destination)
- **Datagram sent** - It is the number of datagrams sent by source
- **Datagram received** - It is the number of datagrams received by destination

IP Metrics:

- **Device_Id** - It displays the Id's of the layer 3 devices
- **Packet sent** - It is the number of packets sent by a source, intermediate devices (Router or L3 switch)
- **Packet forwarded** - It is the number of packets forwarded by intermediate devices (Router or L3 switch)
- **Packet received** - It is the number of data packets received by destination, routing packets (OSPF, RIP etc.) received by Routers

- **TTL expired** - Time-to-live (TTL) is a value in an Internet Protocol (IP) packet that tells a network router whether or not the packet has been in the network too long and should be discarded
- **Firewall blocked** - It is the number of packets blocked by firewall at routers

IEEE802.11_Metrics: Displayed if WLAN is running in the network

- **Device_Id** - It represents the Id's of the wireless devices which supports 802.11 (WLAN)
- **Interface_Id** - It represents the interface Id's of the wireless nodes
- **Frame Sent** - It is the Number of frames sent by Access Point
- **Frame Received** - It is the number of frames received by a wireless node
- **RTS Sent** - It is the number of Request to send (RTS) packets sent by a Wireless Node. RTS/CTS frames are sent prior to transmission when the packet size exceeds RTS threshold. The access point receives the RTS and responds with a CTS frame. The station must receive a CTS frame before sending the data frame. The CTS also contains a time value that alerts other stations to hold off from accessing the medium while the station initiating the RTS transmits its data.
- **RTS Received** - It is the number of RTS packets received by an Access Points
- **CTS Sent** - It is the number of Clear to send (CTS) packets sent by an Access Points
- **CTS Received** - It is the number of CTS packets received by Wireless Nodes
- **Successful Backoff** - It is the number of successful backoffs running at a wireless node. In the IEEE 802.11 Wireless Local Area Networks (WLANs), network nodes experiencing collisions on the shared channel need to backoff for a random period of time, which is uniformly selected from the Contention Window (CW). Backoff is a timer which is decreased as long as the medium is sensed to be idle for a DIFS, and frozen when a transmission is detected on the medium, and resumed when the channel is detected as idle again for a DIFS interval
- **Failed Backoff** - It is the number of failed backoffs at wireless node

LTE metrics: Displayed if LTE is running in the network

- **Device Id** - It is the unique Id of a device
- **Packet transmitted** - It is the number of packets transmitted by eNB and UE which is having application
- **Bytes transmitted** - It is the number of bytes transmitted by eNB and UE
- **Packet received** - It is the number of packets received by a eNB and UE
- **Bytes received** - It is the number of bytes received by eNB and UE

- **Handover count** - It represents the number of handovers occurred in a UE. Handover is a procedure for changing the serving cell of a UE

AODV/DSR Metrics: Displayed if AODV is running in the network

- **Device Id** - It is the Id of the wireless node
- **RREQ sent** - It is the number of Route Request packets sent by wireless node during Route Discovery process
- **RREQ forwarded** - It is the number of Route Request packets forwarded by wireless node during Route Discovery process
- **RREP sent** - It is the number of Route Reply packets sent by wireless node when route is found during Route Discovery process
- **RREP forwarded** - It is the number of Route Reply packets forwarded by a wireless node when route is found during Route Discovery process
- **RERR sent** - It is the total number of Route Error packets sent by a wireless node during Route Maintenance
- **RERR forwarded** - It is the total number of Route Error packets forwarded by a wireless node during Route Maintenance
- **Packet originated** - It is the total number of packets originated in a source node
- **Packet transmitted** - It is the total number of packets transmitted by a source node and intermediate device (LoWPAN Gateway or sink node)
- **Packet dropped** - It is the total number of packets dropped by a wireless node

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

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

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)

Sensor metrics: Displayced 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 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 has 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 enabed, 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 competes 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

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
- **Receiving Energy** - It is the energy consumed by the respective sensor while receiving data
- **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
- **Sleep Energy** - This is the energy consumed when the respective sensor is in an inactive mode

CR metrics: Displayed if 802.22 cognitive radio is running in the network

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

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

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

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.

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.

- **Packets 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.
- **Packets 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.

Calculation:

$$Application\ Throughput(in\ Mbps) = \frac{Total\ payload\ delivered\ to\ destination\ (bytes)\ *\ 8}{Simulation\ Time\ (Micro\ sec)}$$

- **Delay** - It is the average amount of time taken calculated for all the packets to reach the destination from the source.

Note about metrics: NetSim uses the Douglas–Peucker algorithm for plotting the throughput (link/application) over time. This is an iterative end-point fit algorithm, which takes a curve composed of line segments and finds a similar curve with fewer points.

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.

Note about packet transmission: The Network Stack forms the core of NetSim's architecture. The Stack consists of five IN and OUT events: PHYSICAL_IN, MAC_IN, NETWORK_IN, TRANSPORT_IN, APPLICATION_IN and APPLICATION_OUT, TRANSPORT_OUT, NETWORK_OUT, MAC_OUT, PHYSICAL_OUT. All the packets when transferred between devices go through the above events in order. IN events occur when the packet is entering a device and all the OUT events occur when packet leaves a device.

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

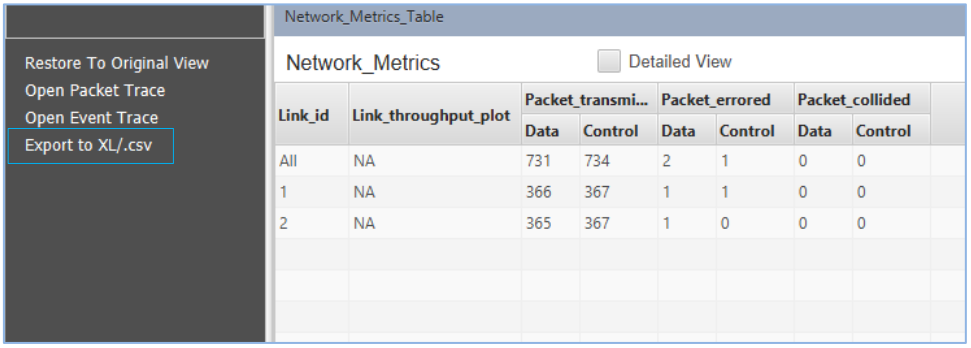
S. No	File	Contents
1	Metrics.xml	Contains the metrics of the network that is simulated recently.
2	Node.pcap	Contains the information of captured packets that is recently simulated.
3	LicenseErrorLog.txt	Contains the status of the communication between the NetSim dongle and the client.
4	ConfigLog.txt	This file will be written while reading the Configuration file. Provides errors if there are errors in the configuration file.
5	LogFile.txt	Contains the logs as the control flows across various layers in the Network Stack
6	PacketTrace.csv	Contains the detailed packet information. This file will be written only when Packet Trace is enabled.
7	EventTrace.csv	Contains the information about each event. This file will be written only when Event Trace is enabled.
8	Animation.txt	Contains the information about the flow of the packet.
9	Static ARP.txt	Contains the information about the dropped devices like Ip address and mac address.

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.

7.2 Export to .csv

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



XL/CSV fle

Network_Metrics

Link_id	Link_throughput_plot	Packet_transmitted Data	Packet_transmitted Control	Packet_errored Data	Packet_errored Control	Packet_collided Data	Packet_collided Control	Bytes_transmitted(bytes)	Payload_transmitted(bytes)	Overhead_transmitted(bytes)
All	NA	473	476	2	1	0	0	753230	687660	65570
1	NA	237	238	1	1	0	0	377378	344560	32818
2	NA	236	238	1	0	0	0	375852	343100	32752

Queue_Metrics

Device_id	Port_id	Queued_packet	Dequeued_packet	Dropped_packet
3	1	236	236	0
3	2	238	238	0

TCP_Metrics

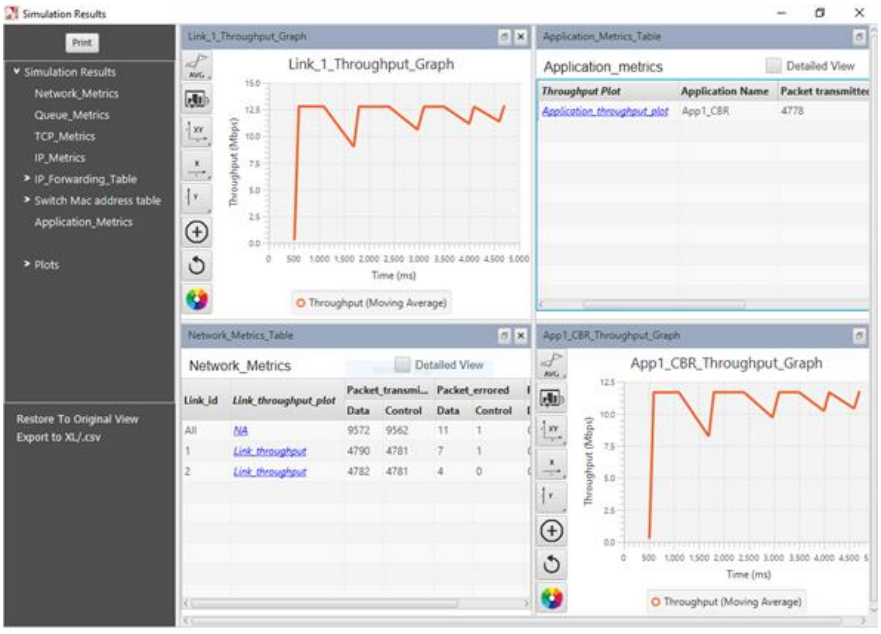
Source	Destination	Local Address	Remote Address	Syn Sent	Syn-Ack Sent	Segment Sent	Segment Received	Segment Retransmitted	Ack Sent	Ack Received	Duplicate segment
WIRED_NODE_1	ANY_DEVICE	11.1.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	
WIRED_NODE_2	ANY_DEVICE	11.2.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	
ROUTER_3	ANY_DEVICE	11.1.1.1:0	0.0.0.0:0	0	0	0	0	0	0	0	
WIRED_NODE_1	WIRED_NODE_2	11.1.1.2:82	11.2.1.2:36934	1	0	234	0	3	1	234	
WIRED_NODE_2	WIRED_NODE_1	11.2.1.2:36934	11.1.1.2:82	0	1	0	234	0	235	1	

IP_Metrics

Device Id	Packet sent	Packet forwarded	Packet received	Packet discarded	TTL expired	Firewall blocked
1	239	0	235	0	0	0

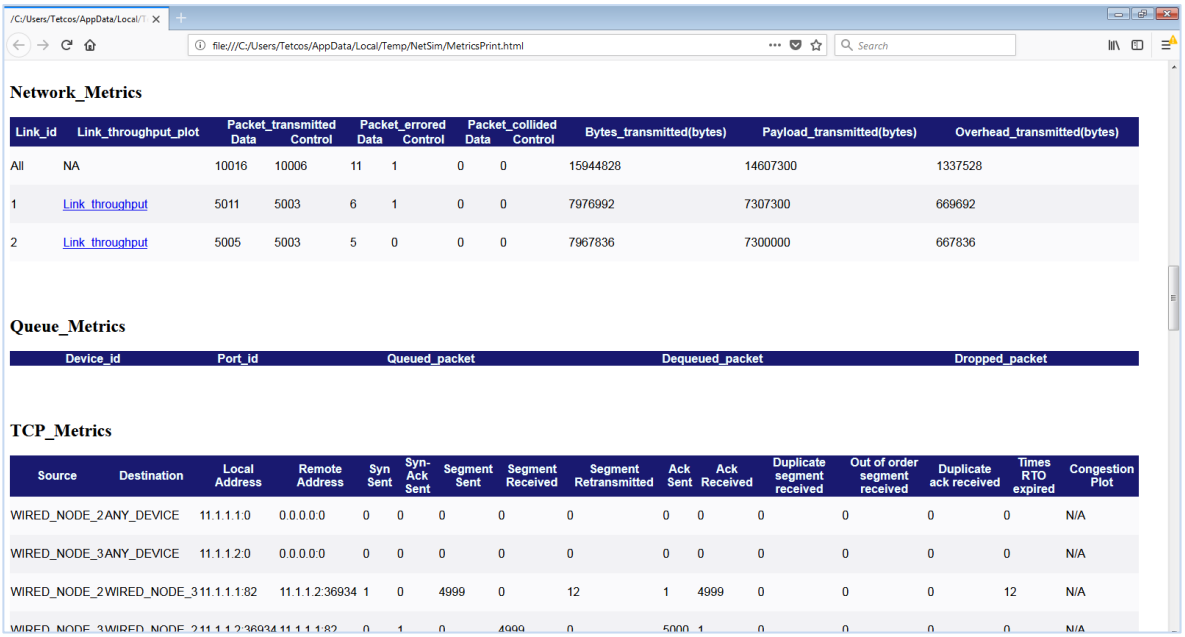
7.3 Print

A web formatted (html file) report can be generated for simulations performed in NetSim, using the Print button present in the 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)



Network_Metrics

Link_id	Link_throughput_plot	Packet_transmitted Data	Packet_transmitted Control	Packet_errored Data	Packet_errored Control	Packet_collided Data	Packet_collided Control	Bytes_transmitted(bytes)	Payload_transmitted(bytes)	Overhead_transmitted(bytes)
All	NA	10016	10006	11	1	0	0	15944828	14607300	1337528
1	Link_throughput	5011	5003	6	1	0	0	7976992	7307300	669692
2	Link_throughput	5005	5003	5	0	0	0	7967836	7300000	667836

Queue_Metrics

Device_id	Port_id	Queued_packet	Dequeued_packet	Dropped_packet

TCP_Metrics

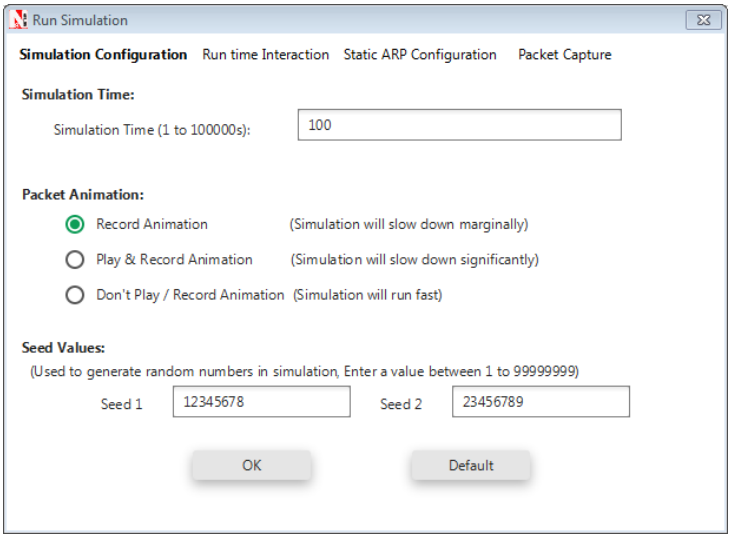
Source	Destination	Local Address	Remote Address	Syn Sent	Syn-Ack Sent	Segment Sent	Segment Received	Segment Retransmitted	Ack Sent	Ack Received	Duplicate segment received	Out of order segment received	Duplicate ack received	Times RTO expired	Congestion Plot
WIRED_NODE_2	ANY_DEVICE	11.1.1.1:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	0	N/A
WIRED_NODE_3	ANY_DEVICE	11.1.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0	0	0	N/A
WIRED_NODE_2	WIRED_NODE_3	11.1.1.1:82	11.1.1.2:36934	1	0	4999	0	12	1	4999	0	0	0	12	N/A
WIRED_NODE_3	WIRED_NODE_2	11.1.1.2:36934	11.1.1.1:82	0	1	0	4999	0	5000	1	0	0	0	0	N/A

- 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 as per your need.

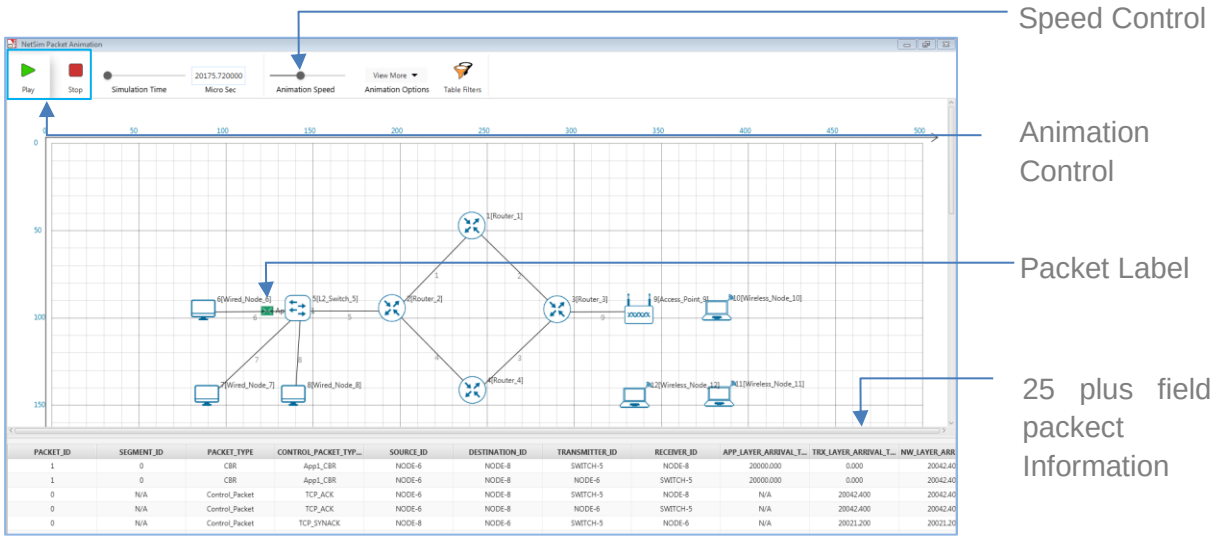
7.4 Packet Animation

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.



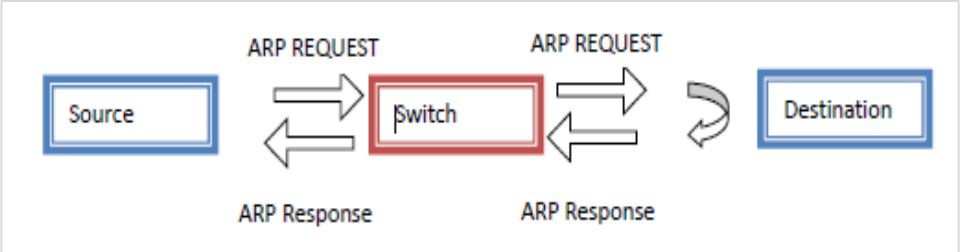
The packet animation would then be recorded and the user can view the animation from the NetSim Packect Animation window as shown below:



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. Packet Animation table is also provided for users to see the flow of packets along with packet animation. 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. The View More Animation options can be used to view plots, IP address of devices, Battery Level, Route tables etc alongside animation.

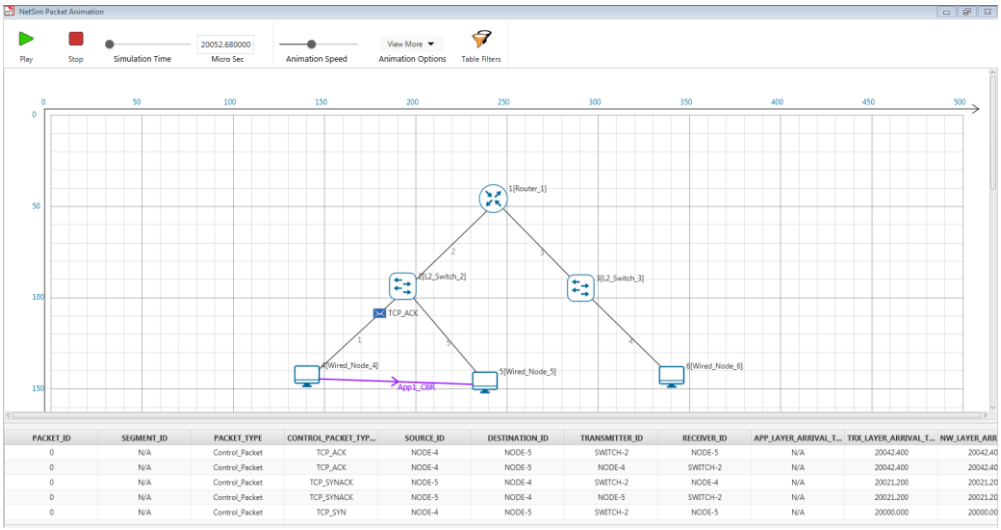
7.4.1 Example on how to use NetSim packet animation feature:

Case 1: ARP PROTOCOL- WORKING



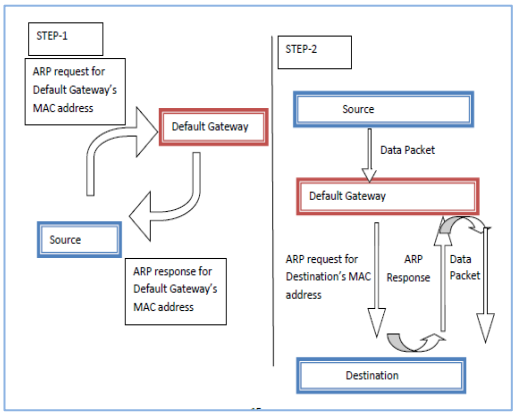
- Create a scenario with 3 wired nodes, 2 switches and 1 router and connect it based on the following scenario.
- Disable TCP in all the wired nodes.
- Click on application and set Source_Id and Destination_Id as 1 and 2 respectively.
- Set Simulation time = 100s. After clicking on Run Simulation, edit **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:



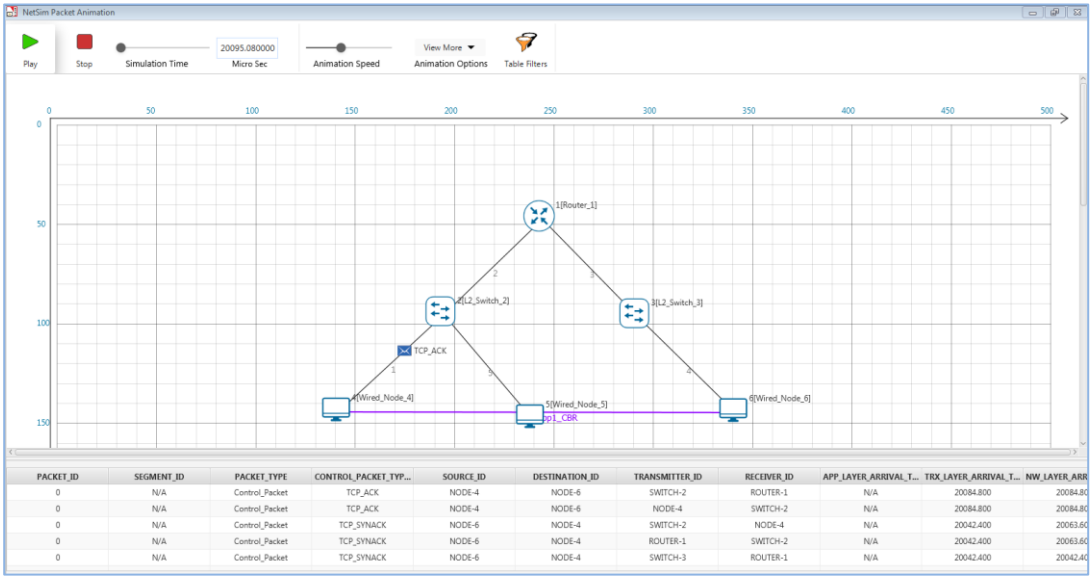
- NODE-1 sends ARP_Request which is then broadcasted by SWITCH-4.
- During the process the devices that receive the ARP_Request packet (Switch, Router, and Node-2) will update their ARP table or the switch table.
- NODE -2 sends the ARP_Reply to NODE-1 via SWITCH-4.
- Now NODE-1 updates its ARP table 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



- 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:



- 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.

7.5 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.

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).

7.5.1 How to set filters to NetSim trace file

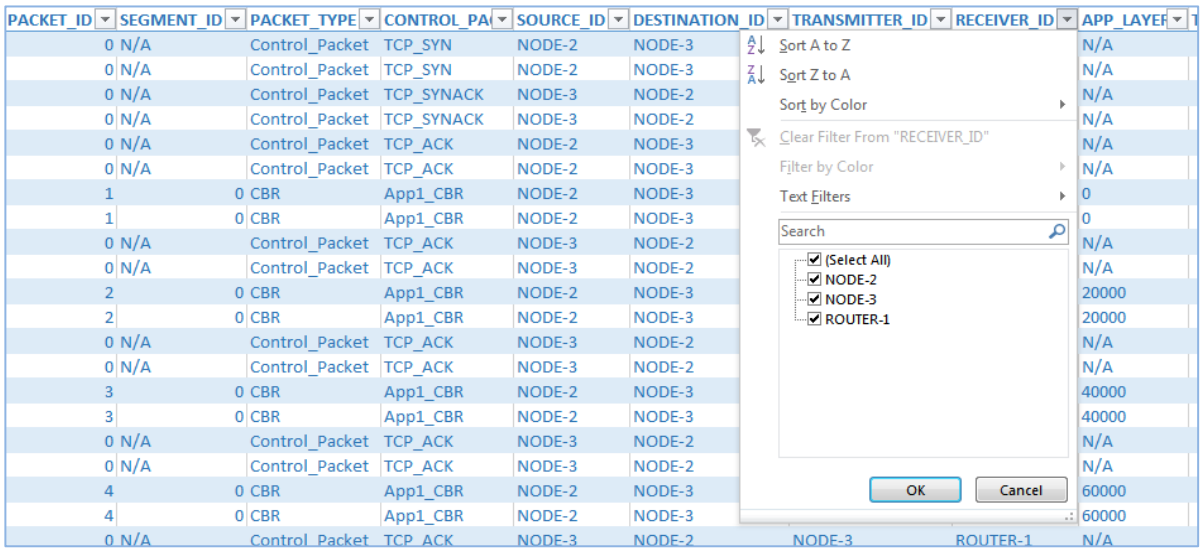
Step 1: Open the trace file. (In this example packet trace is opened)

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

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.

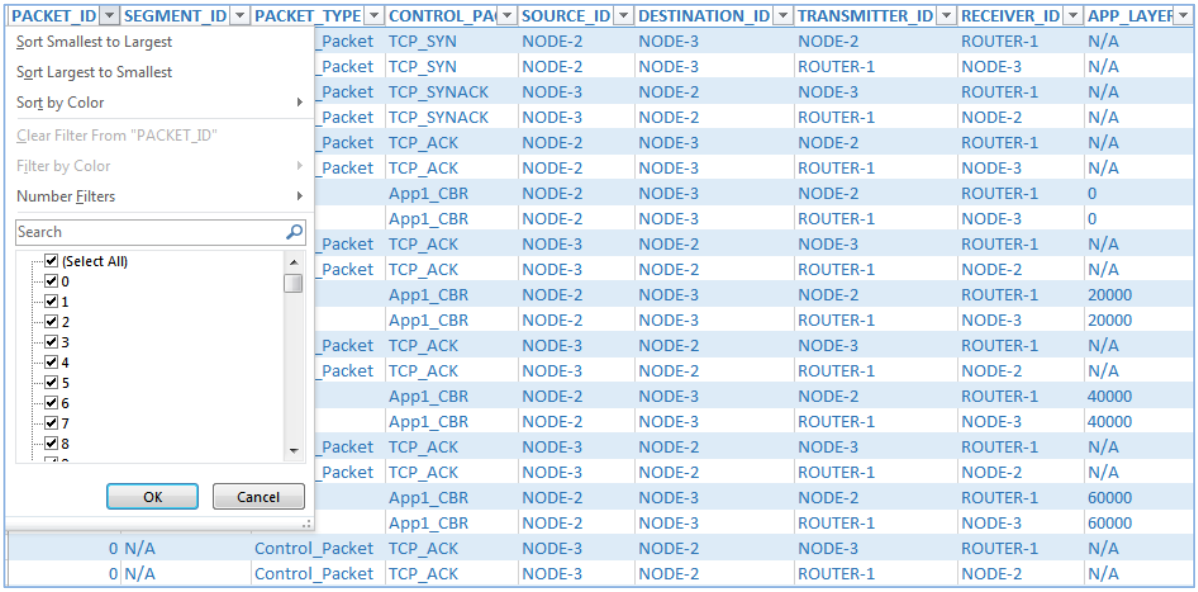


PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PA	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER
0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
1	0	CBR	App1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1	0
2	0	CBR	App1_CBR	NODE-2	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

Typically, filters can be set to observe “Errored/Collided/Successful “packets, packets of destination and packets of source.

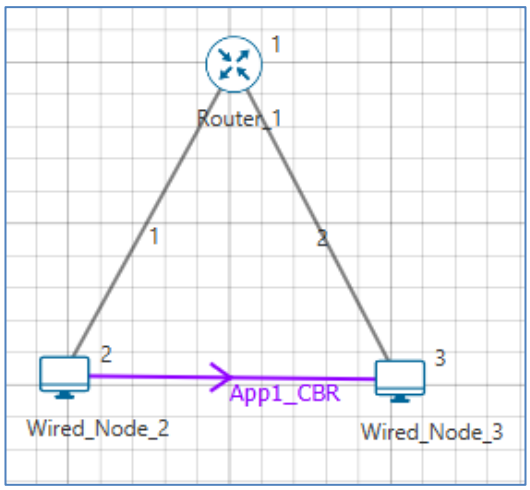
7.5.2 Observing packet flow in the Network through packet trace file

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_LAYEF
		_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
		_Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3	N/A
		_Packet	TCP_SYNACK	NODE-3	NODE-2	NODE-3	ROUTER-1	N/A
		_Packet	TCP_SYNACK	NODE-3	NODE-2	ROUTER-1	NODE-2	N/A
		_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	N/A
		_Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3	N/A
			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-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	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

Scenario is as shown below and 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.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PA	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYEF
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

Note: In the trace file device IDs are shown not device names. Wired Node 1's ID is 2 so it is Shown as NODE-2, Wired Node 2's ID is 3 so it is shown as NODE -3, Router-1' ID is 1 so it is shown as ROUTER-1. Device IDs are shown on the top of the device icon in the above scenario.

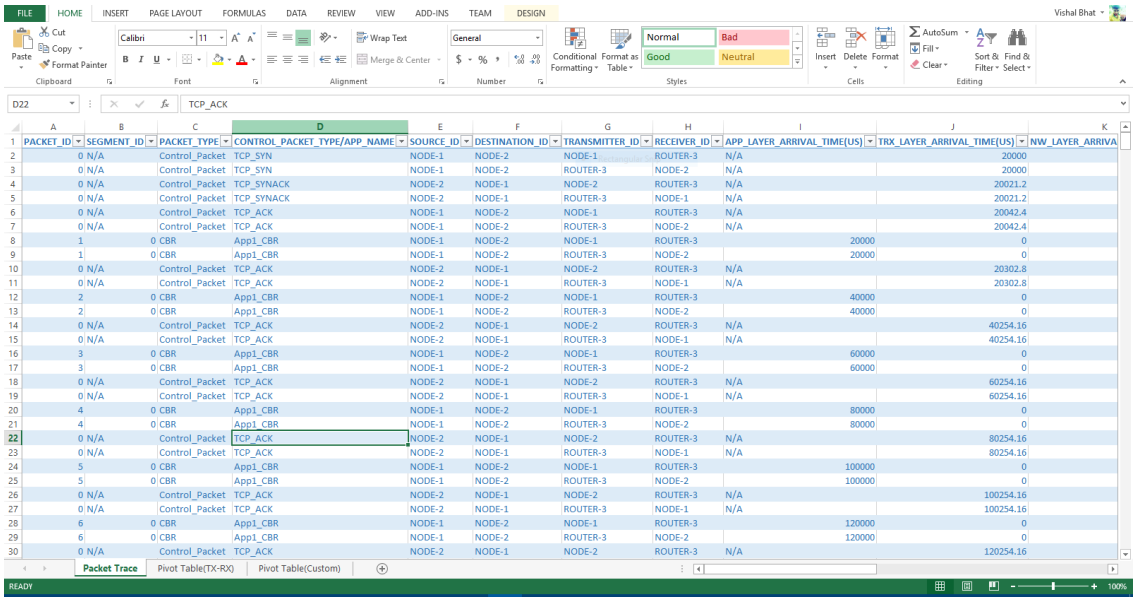
In a scenario source and destinations are fixed but transmitter and receiver are changed. For example, in the above scenario NODE-2 is the source and NODE-3 is the destination, but when NODE- 2 sending the packet to the ROUTER-1 then NODE-2 is the transmitter and ROUTER-1 is the receiver. When ROUTER-1 sending the packet to the NODE-3, ROUTER-1 is the transmitter and NODE-3 is the receiver.

7.5.3 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

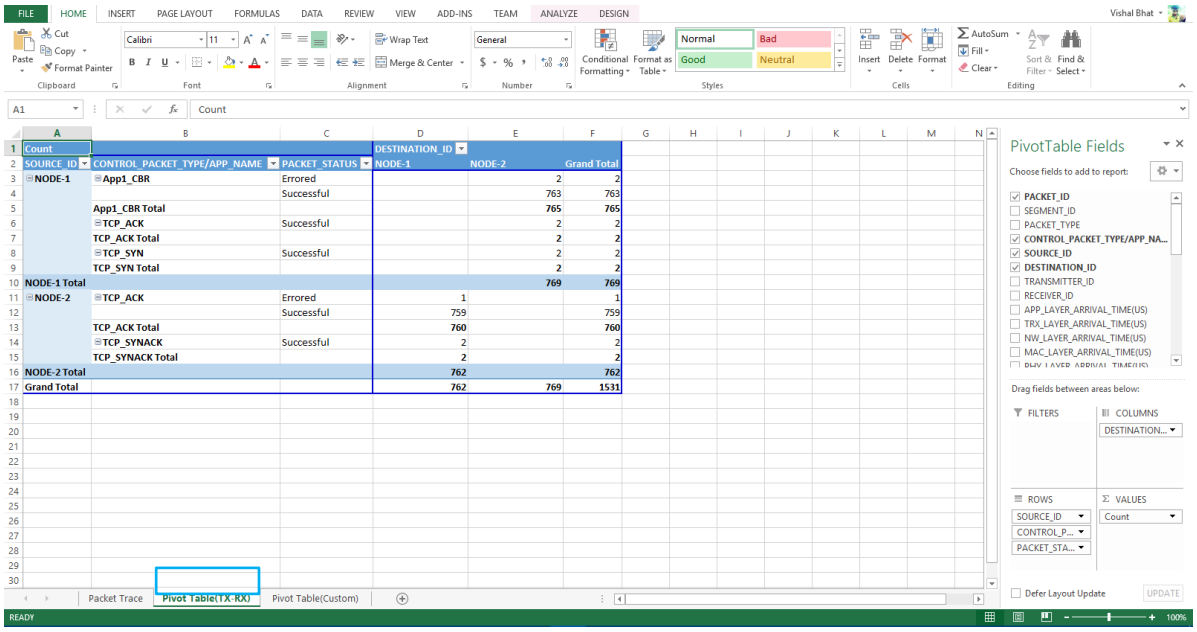
PivotTables are a great way to summarize, analyse, explore, and present your data, and you can create them with just a few clicks. PivotTables are highly flexible and can be quickly adjusted depending on how you need to display your results. You can also create Pivot Charts based on PivotTables that 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.



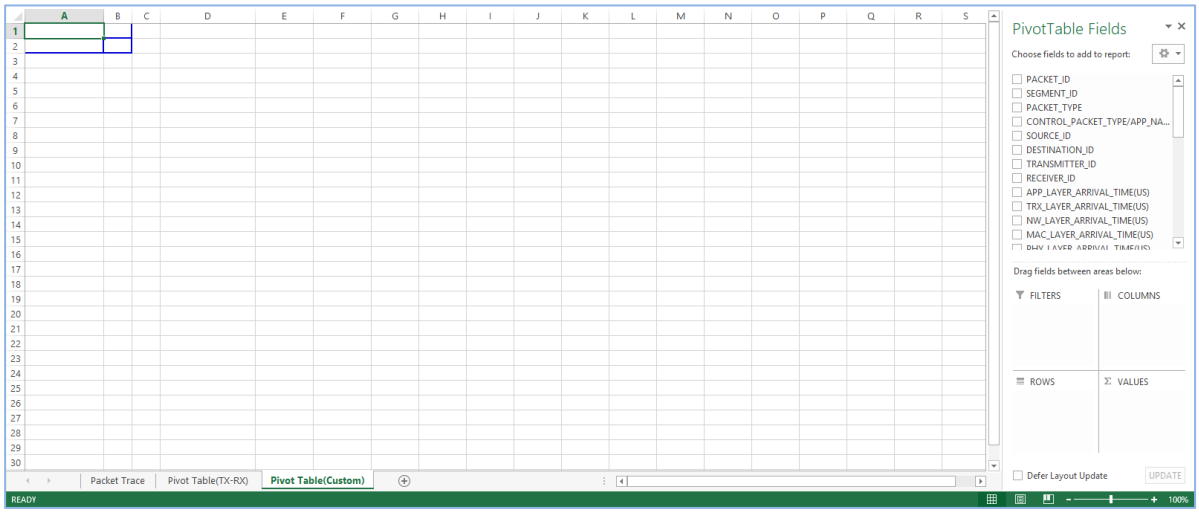
PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME(US)	TRX_LAYER_ARRIVAL_TIME(US)	NW_LAYER_ARRIVA
2	0 N/A	Control_Packet	TCP_SYN	NODE-1	NODE-2	NODE-1	ROUTER-3	N/A	20000	
3	0 N/A	Control_Packet	TCP_SYN	NODE-1	NODE-2	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	

Sheet 2 of the packet trace file has a pivot table – Pivot Table (TX-RX) automatically populated to analyze the packets that were transmitted and received in the network that was simulated. Further users can modify the table by adding or deleting the column headers.



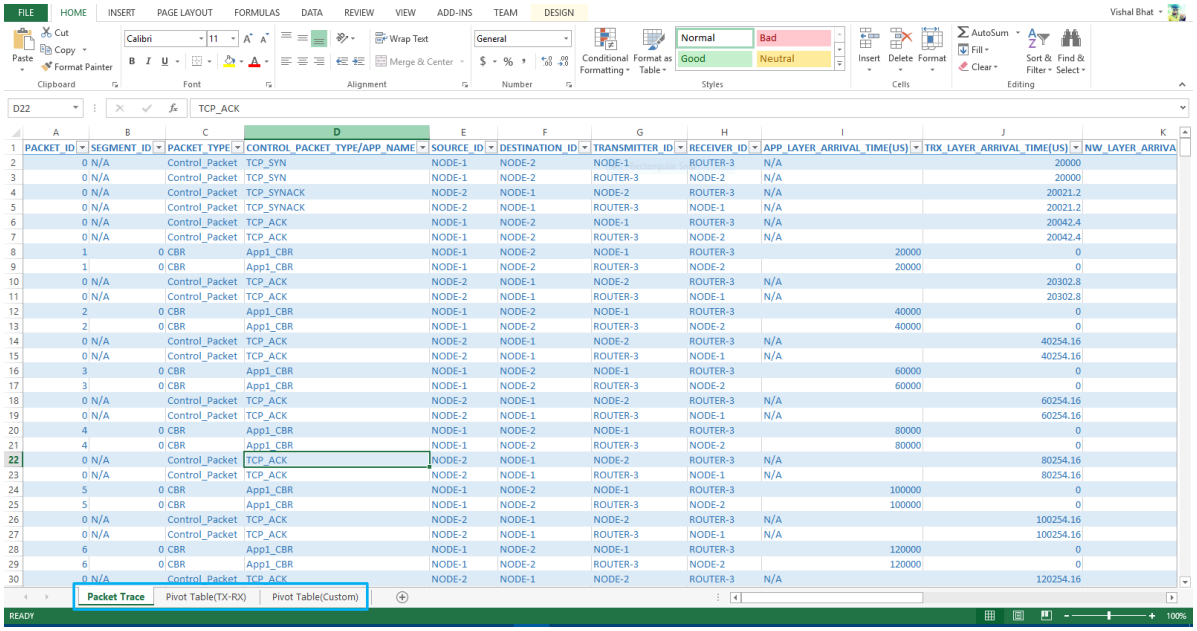
Count			DESTINATION_ID		
SOURCE_ID	CONTROL_PACKET_TYPE/APP_NAME	PACKET_STATUS	NODE-1	NODE-2	Grand Total
NODE-1	App1_CBR	Errored		2	2
		Successful		763	763
	App1_CBR Total			765	765
	TCP_ACK	Successful		2	2
	TCP_ACK Total			2	2
	TCP_SYN	Successful		2	2
	TCP_SYN Total			2	2
NODE-1 Total				769	769
NODE-2	TCP_ACK	Errored	1		1
		Successful	759		759
	TCP_ACK Total		760		760
	TCP_SYNACK	Successful	2		2
	TCP_SYNACK Total		2		2
NODE-2 Total			762		762
Grand Total			762	769	1531

Sheet 3 of the packet trace has a black pivot table – **Pivot Table (Custom)** which can be used to create additional pivot tables from scratch.

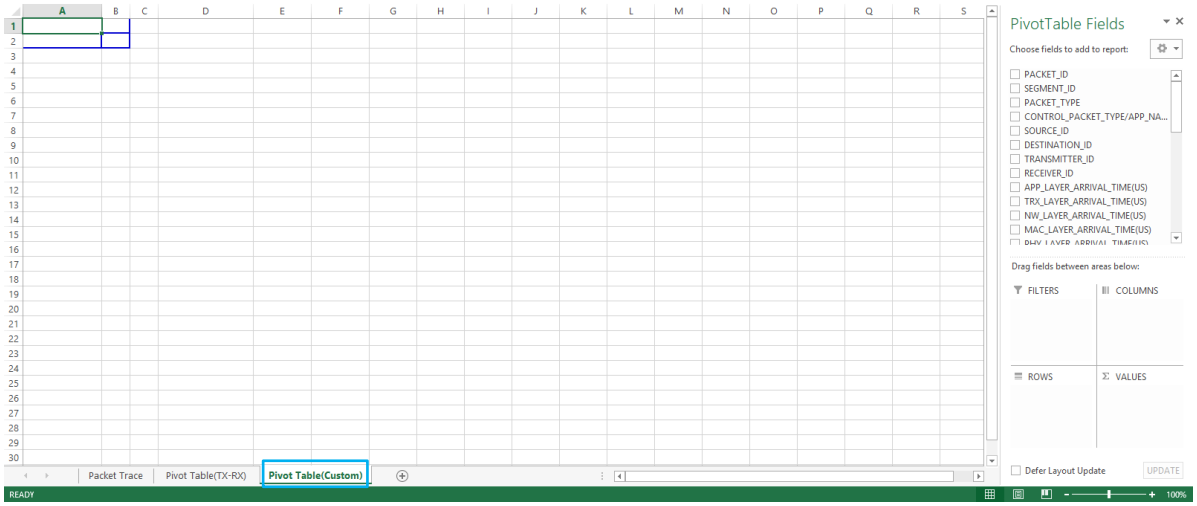


Steps to analyse the packet trace using pivot tables

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)



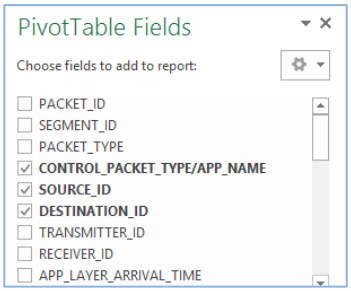
Step 2: Click on Pivot Table (Custom) to create your own pivot table.



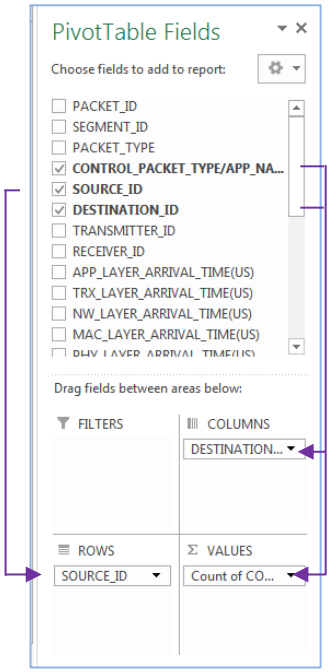
Once you open the sheet PivotTable (Custom), you'll need to decide which **fields** to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

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.



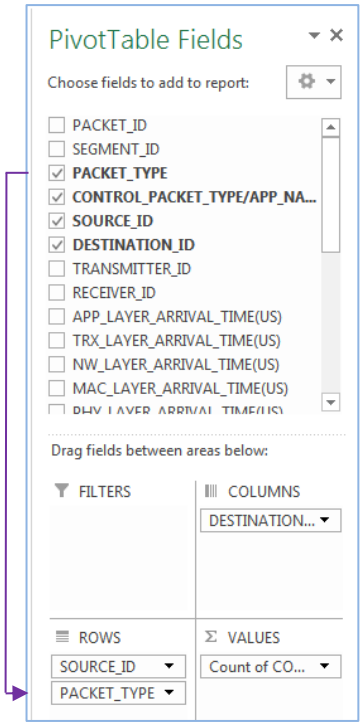
- The selected fields will be added to one of the four areas below the Field List. Click SOURCE_ID, hold it and drag to the ROW field. Similarly, DESTINATION_ID to COLUMNS and CONTROL_PACKET_TYPE/APP_NAME VALUES



- 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

- 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.



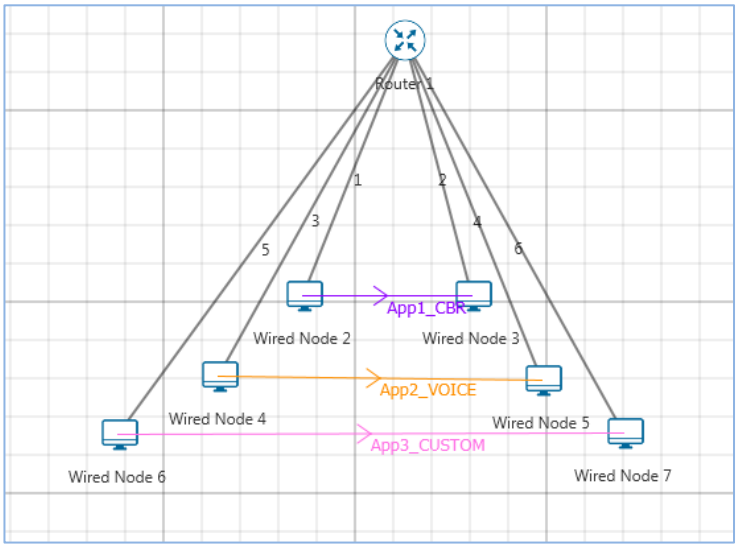
- This will look like

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

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

Count of CONTROL_PACKET_TYPE/APP_NAME			DESTINATION_ID		
SOURCE_ID	PACKET_TYPE	PACKET_STATUS	NODE-1	NODE-2	Grand Total
NODE-1	CBR	Errored		2	2
		Successful		350	350
	CBR Total			352	352
	Control_Packe	Successful		4	4
	Control_Packet Total			4	4
NODE-1 Total				356	356
NODE-2	Control_Packe	Errored	1		1
		Successful	348		348
	Control_Packet Total		349		349
NODE-2 Total			349		349
Grand Total			349	356	705

Delay analysis: To explain how users can do the delay analysis, we have chosen the packet trace generated per the following network scenario

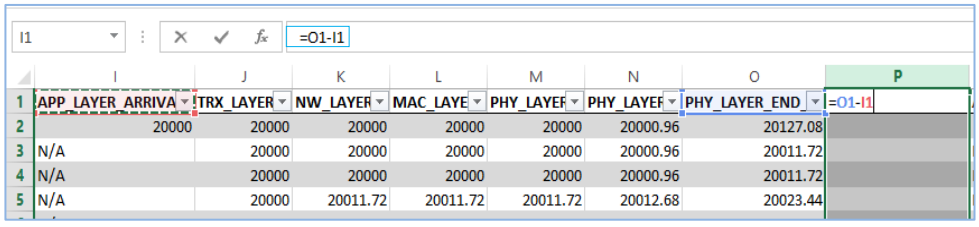


Create a network scenario with 1 router and 6 wired nodes. Create 3 applications as per the following

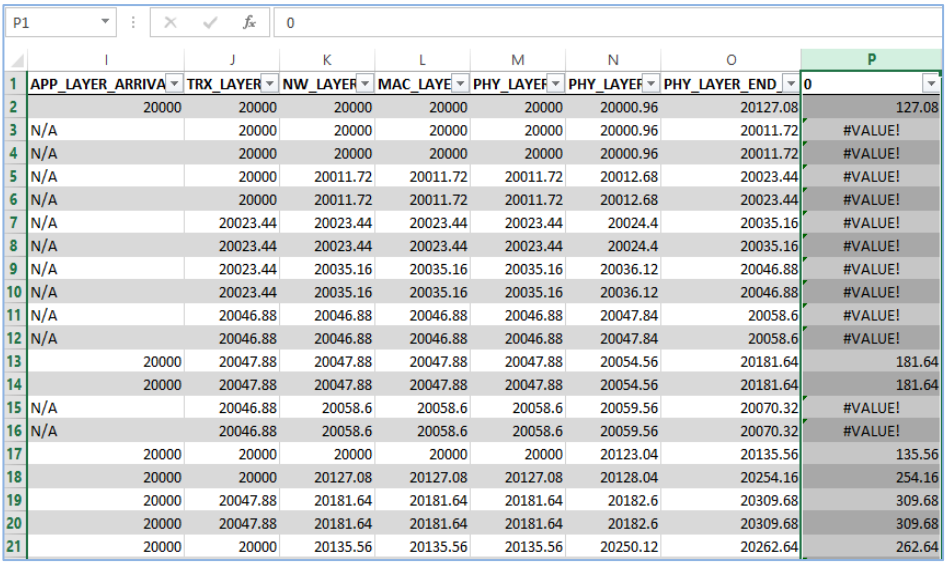
Application Type	Source Id	Destination Id	Packet Size (Bytes)	Inter arrival time (µs)
CBR	2	3	1460	20000
VOICE	4	5	1500	20000
CUSTOM	6	7	1200	20000

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

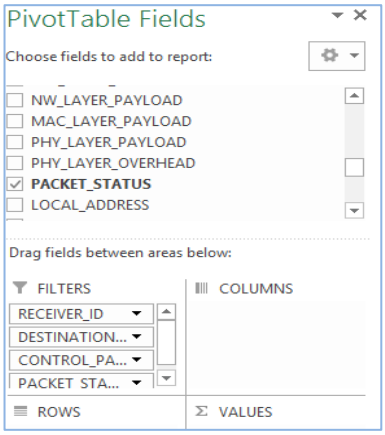


- Then Press CTRL + ENTER. This will calculate delay for the whole column shown below

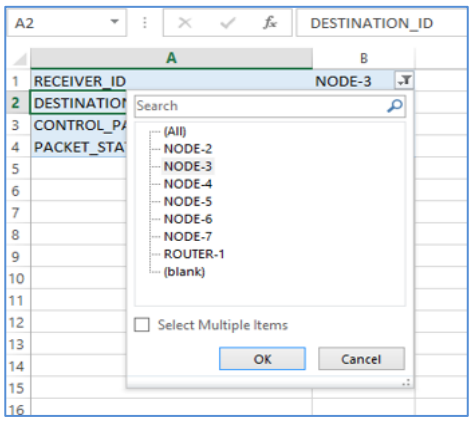


	I	J	K	L	M	N	O	P
1	APP_LAYER_ARRIVA	TRX_LAYER	NW_LAYER	MAC_LAYE	PHY_LAYER	PHY_LAYER	PHY_LAYER_END_	0
2	20000	20000	20000	20000	20000	20000.96	20127.08	127.08
3	N/A	20000	20000	20000	20000	20000.96	20011.72	#VALUE!
4	N/A	20000	20000	20000	20000	20000.96	20011.72	#VALUE!
5	N/A	20000	20011.72	20011.72	20011.72	20012.68	20023.44	#VALUE!
6	N/A	20000	20011.72	20011.72	20011.72	20012.68	20023.44	#VALUE!
7	N/A	20023.44	20023.44	20023.44	20023.44	20024.4	20035.16	#VALUE!
8	N/A	20023.44	20023.44	20023.44	20023.44	20024.4	20035.16	#VALUE!
9	N/A	20023.44	20035.16	20035.16	20035.16	20036.12	20046.88	#VALUE!
10	N/A	20023.44	20035.16	20035.16	20035.16	20036.12	20046.88	#VALUE!
11	N/A	20046.88	20046.88	20046.88	20046.88	20047.84	20058.6	#VALUE!
12	N/A	20046.88	20046.88	20046.88	20046.88	20047.84	20058.6	#VALUE!
13	20000	20047.88	20047.88	20047.88	20047.88	20054.56	20181.64	181.64
14	20000	20047.88	20047.88	20047.88	20047.88	20054.56	20181.64	181.64
15	N/A	20046.88	20058.6	20058.6	20058.6	20059.56	20070.32	#VALUE!
16	N/A	20046.88	20058.6	20058.6	20058.6	20059.56	20070.32	#VALUE!
17	20000	20000	20000	20000	20000	20123.04	20135.56	135.56
18	20000	20000	20127.08	20127.08	20127.08	20128.04	20254.16	254.16
19	20000	20047.88	20181.64	20181.64	20181.64	20182.6	20309.68	309.68
20	20000	20047.88	20181.64	20181.64	20181.64	20182.6	20309.68	309.68
21	20000	20000	20135.56	20135.56	20135.56	20250.12	20262.64	262.64

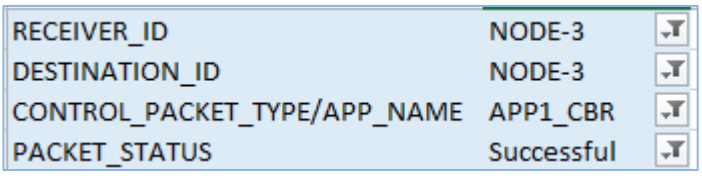
- Name the column as DELAY
- Go to Insert->PivotTable and click on OK to create a blank Pivot Table with the newly added column listed under the PivotTable Fields.
- Drag and drop DESTINATION_ID, RECEIVER_ID, PACKET_STATUS and CONTROL_PACKET_TYPE/APP_NAME to FILTERS field shown below



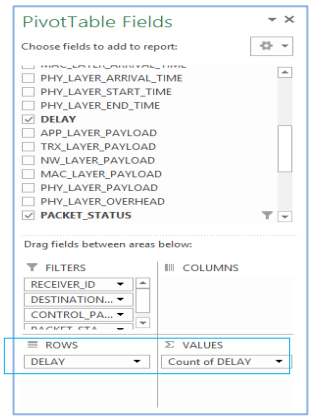
- Filter RECEIVER_ID to Node-3 by clicking on the drop down and select OK



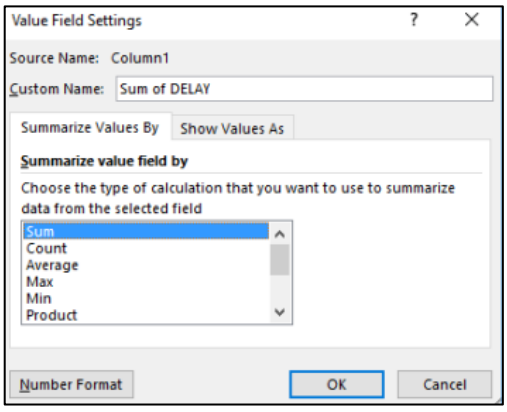
- Similarly filter CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR, DESTINATION_ID to NODE-3 and PACKET_STATUS to Successful



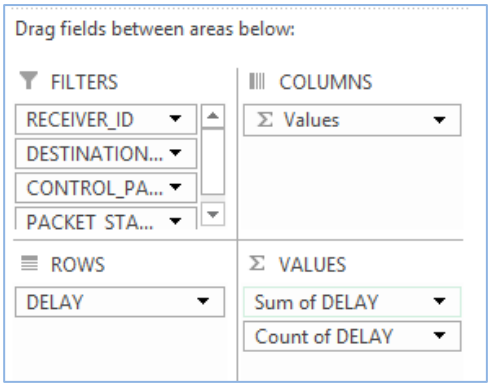
- Drag and drop DELAY value that we have calculated earlier to ROWS and VALUES field



- Click on Count of DELAY drop down and select Value Field settings, then Select SUM and click on OK.

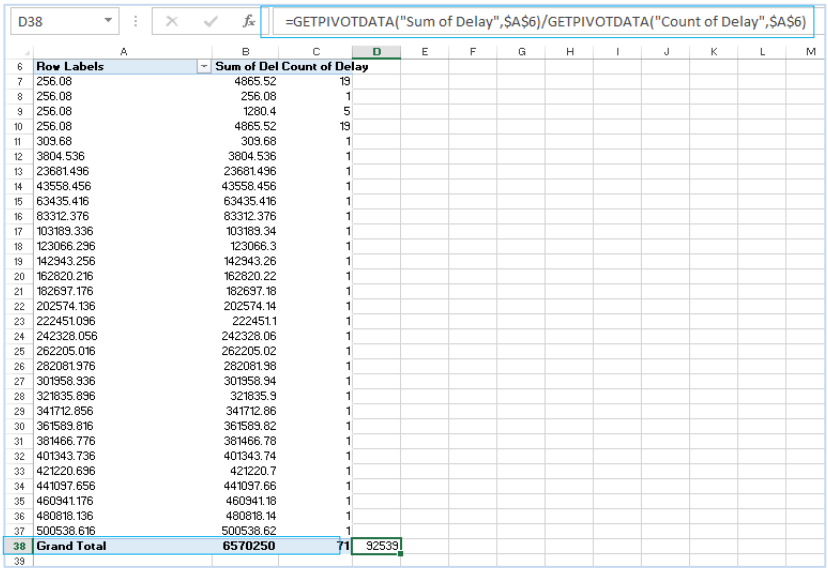


- Again Drag and drop DELAY to VALUES field.



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

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



Row Labels	Sum of Del	Count of Delay	
256.08	4865.52	19	
256.08	256.08	1	
256.08	1280.4	5	
256.08	4865.52	19	
309.68	309.68	1	
3804.536	3804.536	1	
23681.496	23681.496	1	
43558.456	43558.456	1	
63435.416	63435.416	1	
83312.376	83312.376	1	
103189.336	103189.34	1	
123066.296	123066.3	1	
142943.256	142943.26	1	
162820.216	162820.22	1	
182697.176	182697.18	1	
202574.136	202574.14	1	
222451.096	222451.1	1	
242328.056	242328.06	1	
262205.016	262205.02	1	
282081.976	282081.98	1	
301958.936	301958.94	1	
321835.896	321835.9	1	
341712.856	341712.86	1	
361589.816	361589.82	1	
381466.776	381466.78	1	
401343.736	401343.74	1	
421220.696	421220.7	1	
441097.656	441097.66	1	
460941.176	460941.18	1	
480818.136	480818.14	1	
500538.616	500538.62	1	
Grand Total	**6570250**	71	92539

- Compare the obtained value with the DELAY in Application Metrics

Application_Metrics_Table

Application_metrics Detailed View

Application Id	Application Name	Throughput (Mbps)	Delay(microsec)
1	APP1_CBR	0.082928	92538.737092
2	APP2_VOICE	0.597600	258.440482
3	APP3_CUSTOM	0.048960	215.530980

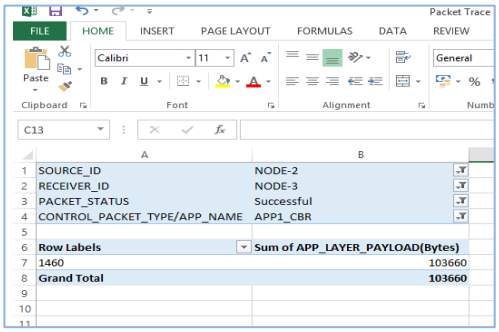
- To calculate DELAY for VOICE application, filter DESTINATION_ID to Node-5, RECEIVER_ID to Node-5, CONTROL_PACKET_TYPE/APP_NAME to APP2_VOICE and PACKET_STATUS to Successful

- Similarly calculate and compareDELAY for other applications by following the above procedure

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.

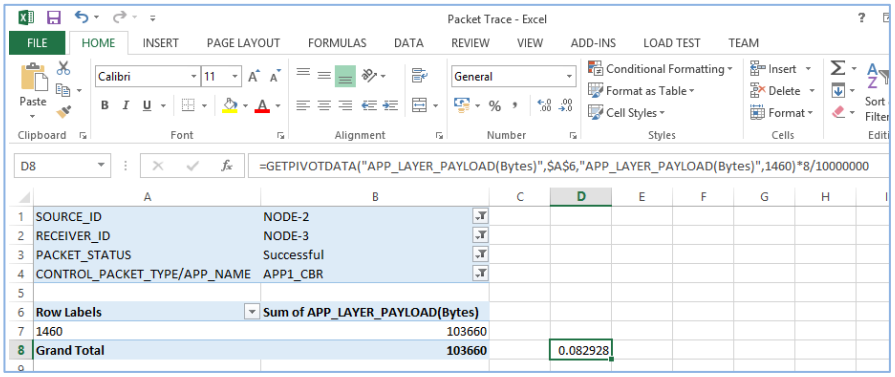
- Similarly drag and drop APP_LAYER_PAYLOAD to ROWS field and VALUES field
- Filter SOURCE_ID to NODE-2, CONTROL_PACKET_TYPE APP_NAME to APP1_CBR,PACKET_STATUS to Successful and RECEIVER_ID to NODE-3
- Click on Count of APP_LAYER_PAYLOAD drop down and select Value Field settings, then Select Sum and click on OK.
- The pivot table would look like



	A	B
1	SOURCE_ID	NODE-2
2	RECEIVER_ID	NODE-3
3	PACKET_STATUS	Successful
4	CONTROL_PACKET_TYPE/APP_NAME	APP1_CBR
5		
6	**Row Labels**	**Sum of APP_LAYER_PAYLOAD(Bytes)**
7	1460	103660
8	**Grand Total**	**103660**

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

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



	A	B	C	D	E	F	G	H
1	SOURCE_ID	NODE-2						
2	RECEIVER_ID	NODE-3						
3	PACKET_STATUS	Successful						
4	CONTROL_PACKET_TYPE/APP_NAME	APP1_CBR						
5								
6	**Row Labels**	**Sum of APP_LAYER_PAYLOAD(Bytes)**						
7	1460	103660						
8	**Grand Total**	**103660**		0.082928				

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_Metrics_Table

Application_metrics Detailed View

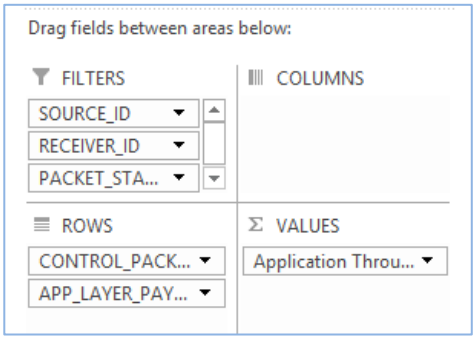
Application Id	Application Name	Throughput (Mbps)	Delay(microsec)
1	APP1_CBR	0.082928	92538.737092
2	APP2_VOICE	0.597600	258.440482
3	APP3_CUSTOM	0.048960	215.530980

- 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 compareTHROUGHPUT for other applications by following the above procedure.

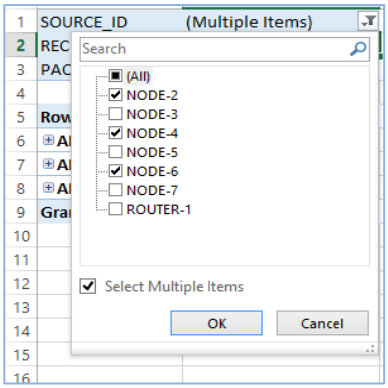
Plotting with Pivot Charts

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

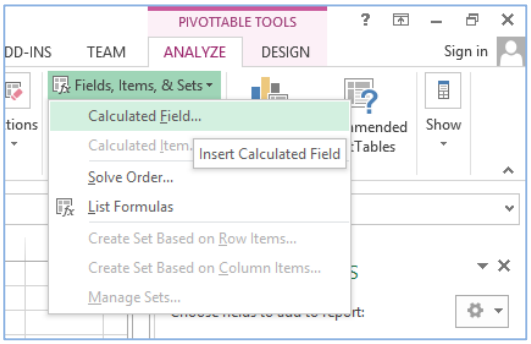
- Open Packet Trace, switch to sheet Pivot Table (Custom)
- Drag and drop SOURCE_ID, RECEIVER_ID and PACKET_STATUS to FILTERS field, then CONTROL_PACKET_TYPE/APP_NAME, APP_LAYER_PAYLOAD to ROWS field and APP_LAYER_PAYLOAD to VALUES field as shown below



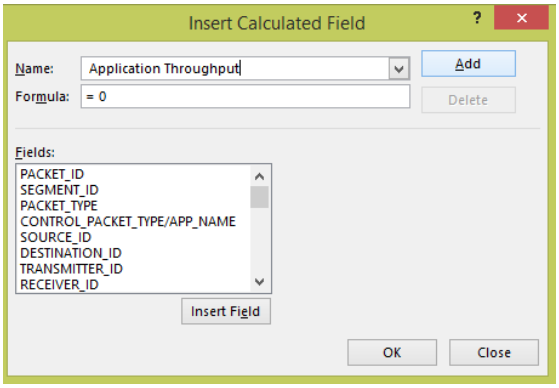
- Filter SOURCE_ID to Node 2, Node 4 and Node 6, then RECEIVER_ID to Node 3, Node 5 and Node 7 and PACKET_STATUS to successful



- 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.

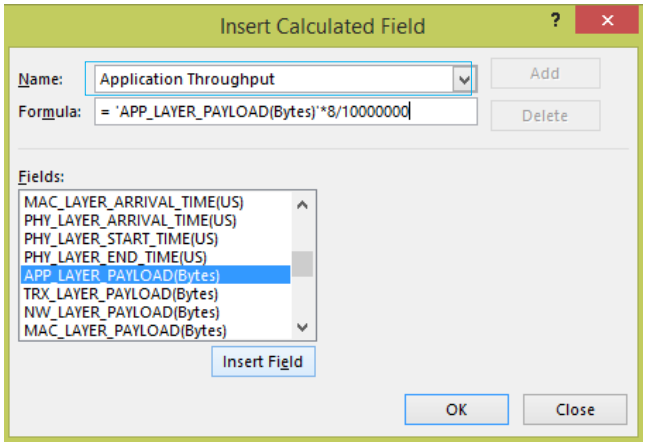


- Type a name for the calculated field, Application Throughput
- Then click on ADD to save the calculated field.



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

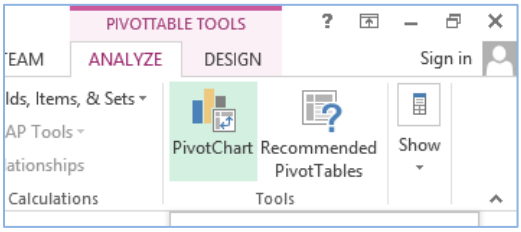
$$Application\ throughput\ (Mbps) = \frac{Sum\ of\ \ Successfully\ Received\ App\ Layer\ Payload\ (Bytes) * 8}{Simulation\ Time\ (\mu s\ )}$$



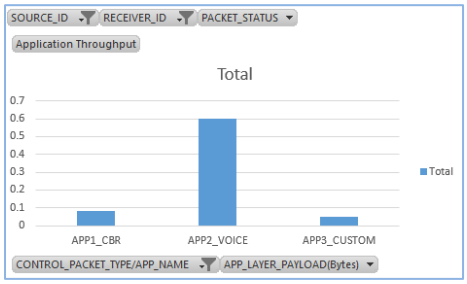
- This would look like

SCREEN SHOT FOR COMPARISON

- 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.



- This will display a pivot chart shown below



Packet Trace Fields:

GENERAL FIELDS	DESCRIPTION
PACKET_ID	Specifies the ID of the Data Packets. For control packets this value is set to 0
SEGMENT_ID	Specifies the ID of the segment of the Data Packet. Segmentation is done in transport layer. If the packet size (generated in the APP layer)

	is greater than the maximum segment size in TRANSPORT layer, packet will get segmented. For control packets it is N/A
PACKET_TYPE	Specifies the type of application that generates the packet. It can be Control Packet, Custom, CBR, Peer_to_peer, E-Mail, DataBase, FTP, Video, Voice, HTTP.
CONTROL_PACKET_TYPE	Specifies the type of Control Packet transmitted. Following are the Protocol specific control packets WLAN: WLAN_ACK, WLAN_BlockACK OSPF: OSPF_HELLO, OSPF_D-D, OSPF_LSR, OSPF_LSU, OSPF_LSA RIP: RIP_Message GSM: GSM_Channel_Request, GSM_Channel_Granted, GSM_Call_Request, GSM_Channel_Request_For_Incoming, GSM_Call_Accepted CDMA: CDMA_Channel_Request, CDMA_Channel_Granted, CDMA_Call_Request, CDMA_Channel_Request_For_Incoming, CDMA_Call_Accepted DSR, AODV, ZRP, OLSR: RREQ, RREP, NDP_HELLO_MESSAGE, OLSR_TC_MESSAGE Zigbee: Zigbee_BEACON_FRAME, Zigbee_ACK Cognitive Radio: SCH, FCH, DS-MAP, US-MAP, UCD, DCD, BW_REQUEST, UCS_NOTIFICATION LTE:LTE_Measurement_Report, LTE_RRC_CONNECTION_SETUP,LTE_RLC_SDU, LTE_RRC_CONNECTION_REQUEST, LTE_RRC_CONNECTION_SETUP_COMPLETE, LTE page, LTE Ack etc.
SOURCE_ID	Specifies the Node ID of the source set in the application
DESTINATION_ID	Specifies the Node ID of the destination set in the application. If the application is a broadcast application the destination field will show 0
TRANSMITTER_ID	Specifies the current node which is transmitting the packet. The difference between a Source node and a Transmitter, is that when the Source remains constant across the entire packet transmission whereas the transmitter ID changes with each hop of the packet.
RECEIVER_ID	Specifies the current node which is receiving the packet. The difference between a Destination node and a Receiver, is that when the Destination remains constant across the entire packet transmission whereas the receiver ID changes with each hop of the packet.
APP_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet is at the Application_Layer of Source_ID (or Transmitter_ID). This is usually the time at which the packet is generated at Source_ID
TRX_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches the Transport_layer from the application layer. This will usually be the same as Application_layer_Arrival_Time unless there are TCP re-transmissions
NW_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches the Network_Layer of Transmitter_ID if this is a Router (or) Time at which packet reaches the Network_layer of previous Router / Source_ID (immediate previous Layer 3 or higher device) if current device is Switch / Access Point.
MAC_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches MAC_Layer of Transmitter_ID
PHY_LAYER_ARRIVAL_TIME (µs)	Specifies the time at which packet reaches PHY_layer of Transmitter_ID
PHY_LAYER_START_TIME (µs)	Specifies the time at which packet starts betting transmitted in the link between Transmitter_ID and Receiver_ID

PHY_LAYER_END_TIME (µs)	Specifies the time at which packet reaches Phy_Layer of Receiver_ID
APP_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Application Layer
TRX_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Transport Layer
NW_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Network Layer
MAC_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Data Link Layer
PHY_LAYER_PAYLOAD (Bytes)	Specifies the size of the Payload at Physical Layer
PHY_LAYER_OVERHEAD (Bytes)	Specifies the size of the overhead in Physical layer
PACKET_STATUS	Specifies whether the Packet is Successful, Collided or Errored
LOCAL_ADDRESS	Specifies the Port Number at Source Node. Port Numbers are chosen randomly by NetSim.
FOREIGN_ADDRESS	Specifies the Port Number at Destination Node. Port Numbers are chosen randomly by NetSim.
CWND (bytes)	Specifies the current size of the TCP congestion window
SEQ_NO	If TCP is enabled, it specifies the TCP Sequence number of the packet
ACK_NO	If TCP is enabled, it specifies the TCP Acknowledgement number of the packet
RTT (seconds)	Specifies the Round Trip Time for the packet
RTO (seconds)	Specifies the Retransmission Timeouts
CONNECTION_STATE	Specifies the state of TCP connection
isSyn	If TCP is enabled, it specifies whether the packet is TCP_SYN or not
isAck	If TCP is enabled, it specifies whether the packet is TCP_ACK/TCP_SYN_ACK or not
isFin	If TCP is enabled, it specifies whether the packet is TCP_FIN or not
SEGMENT_LENGTH	Specifies the segment length of the packet
SOURCE_IP	Specifies the IP address of the source
DESTINATION_IP	Specifies the IP address of the destination
GATEWAY_IP	Specifies the IP address of the device which is transmitting a packet
NEXT_HOP_IP	Specifies the IP address of the next hop

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.

7.6 Event Trace (only in Standard/Pro Version)

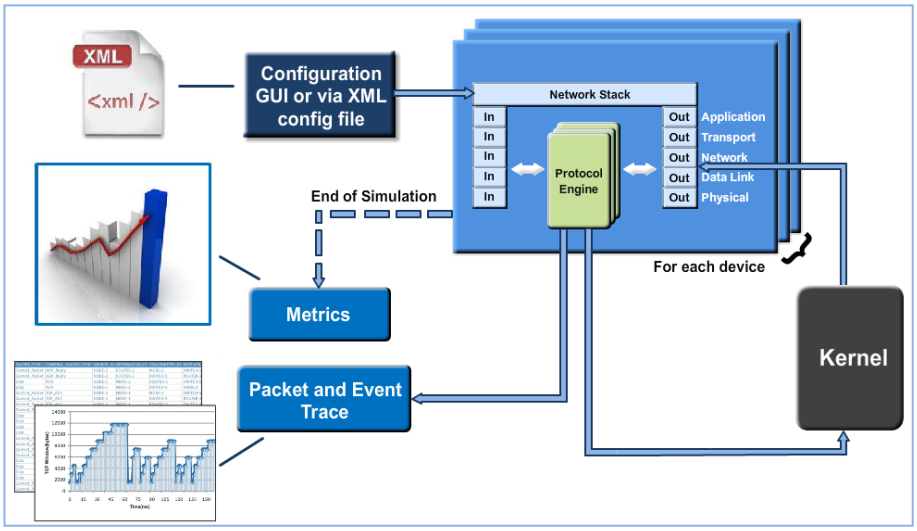
7.6.1 NetSim Network Stack and Discrete Event Simulation working

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

1. Application Layer – CBR, Voice, Video, HTTP, COAP etc
2. Transport Layer – TCP, UDP
3. Network Layer – IP, OSPF, AODV, OLSR etc
4. MAC Layer – 802.11, 802.15.4, LTE etc
5. 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.



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.

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

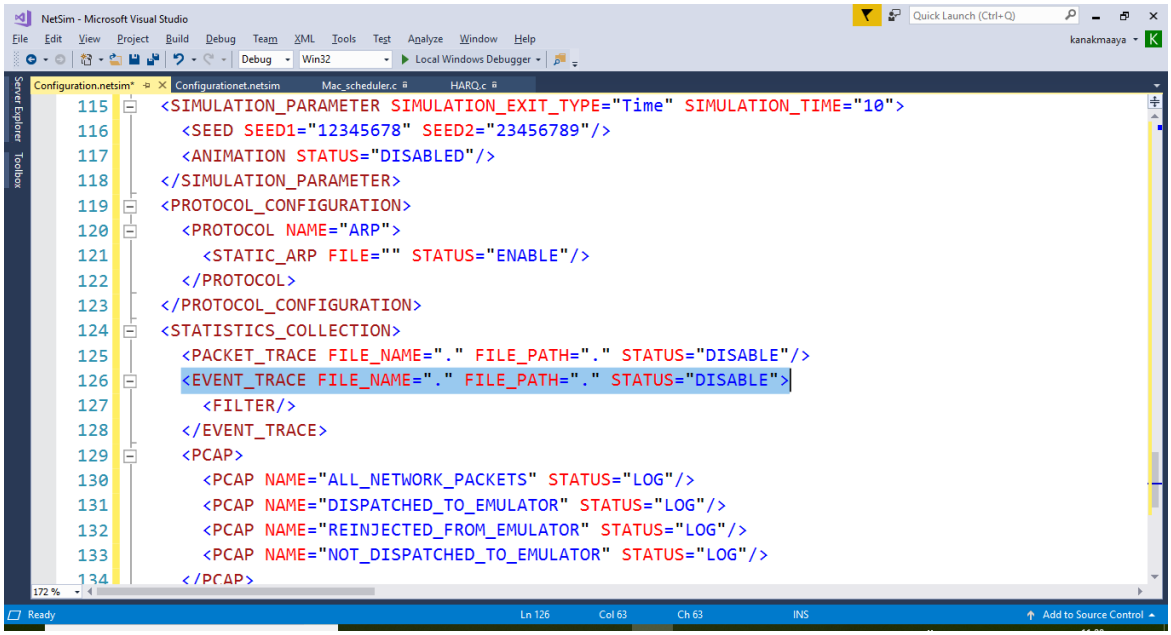
NetSim provides users with the option of turning on "Event Traces".

How to enable Event Trace via GUI?

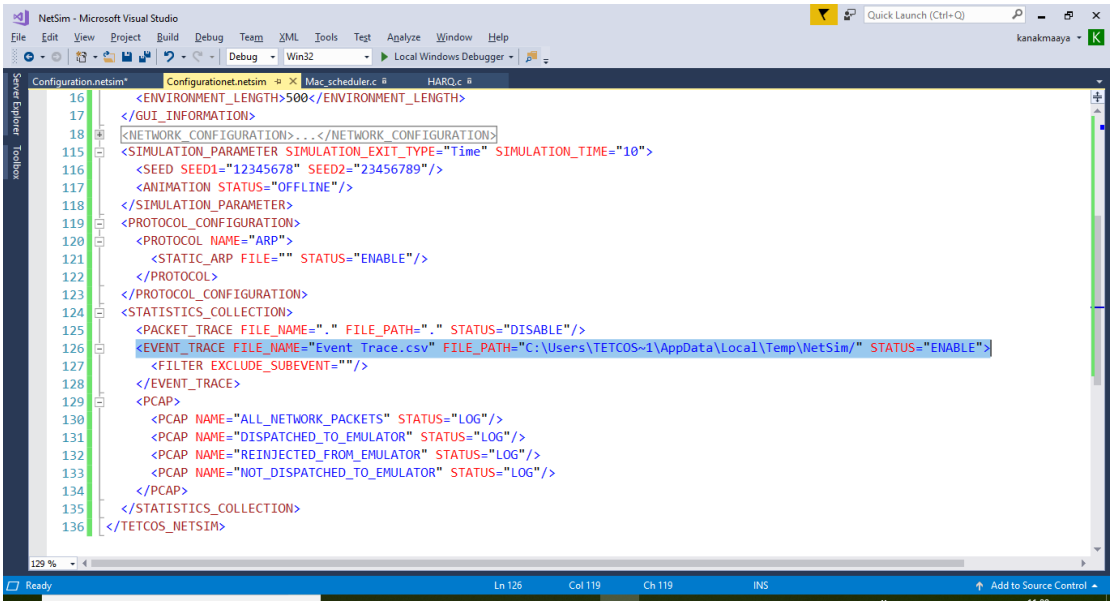
If NetSim runs via GUI, event trace can be turned on by clicking the Event Trace icon in the tool bar and selecting the required fields in the event trace.

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:



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:



Event Trace Metrics:

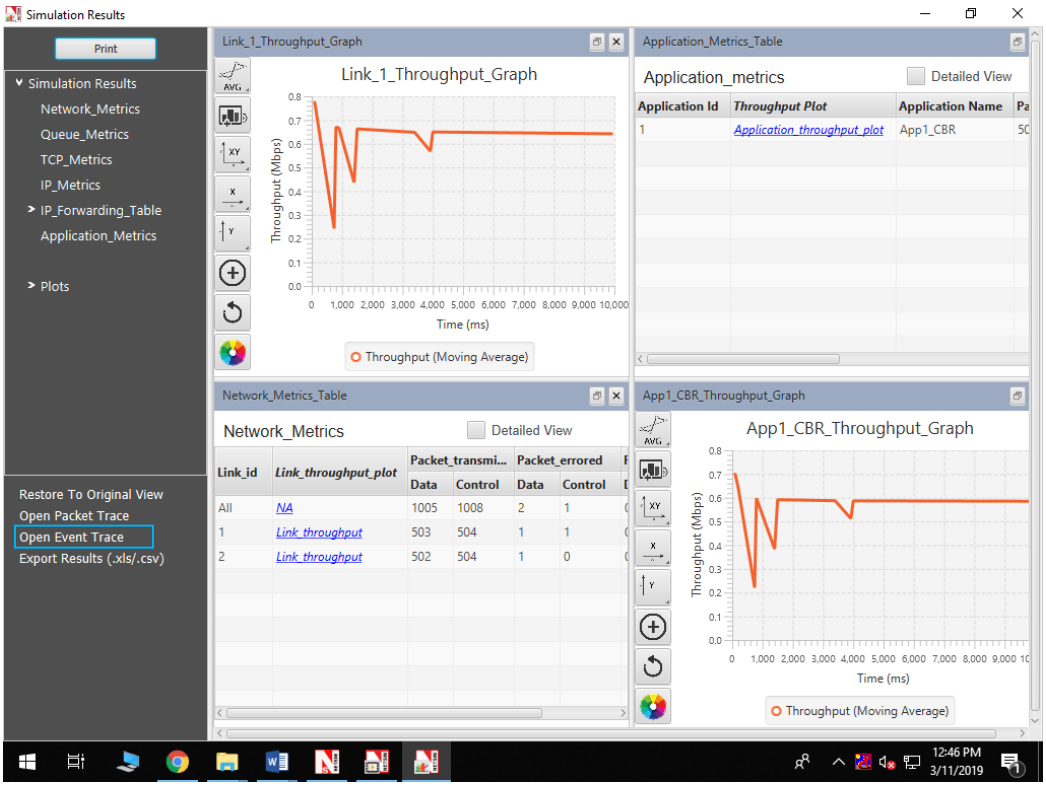
Event_Id	Specifies the ID of the Event
Event_Type	Specifies the type of event being performed, for eg - APPLICATION_IN, APPLICATION_OUT, MAC_OUT, MAC_IN, PHYSICAL_OUT, PHYSICAL_IN,etc
Event_Time	Specifies the time(in microseconds) at which the event is being executed

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.

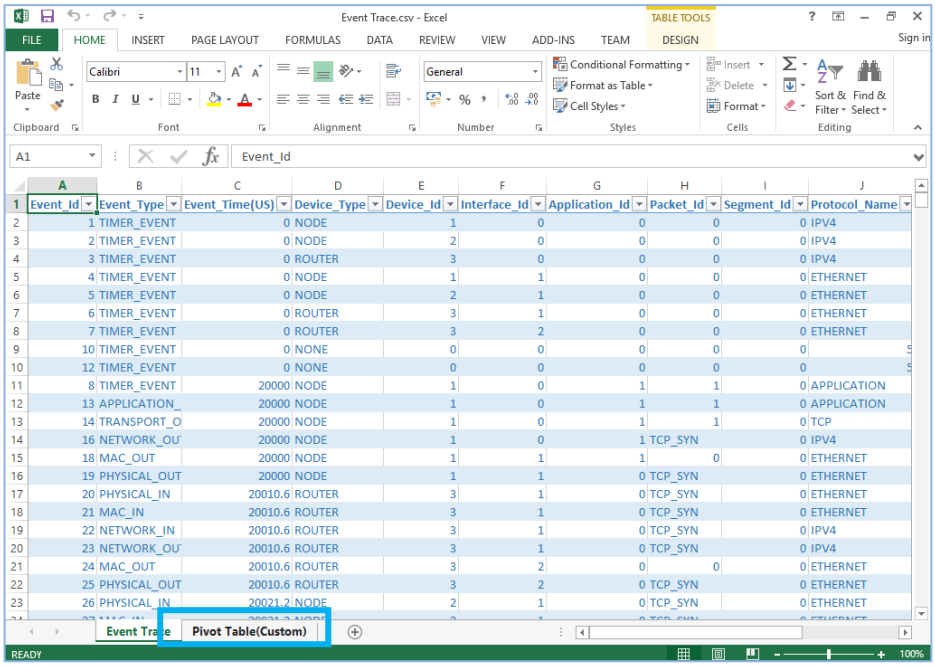
7.6.2 Calculation of Delay, Jitter and Application throughput from event trace

1. Enable Event trace and run simulation
2. Click Event Trace option (Open Event Trace) in the Simulation results window.

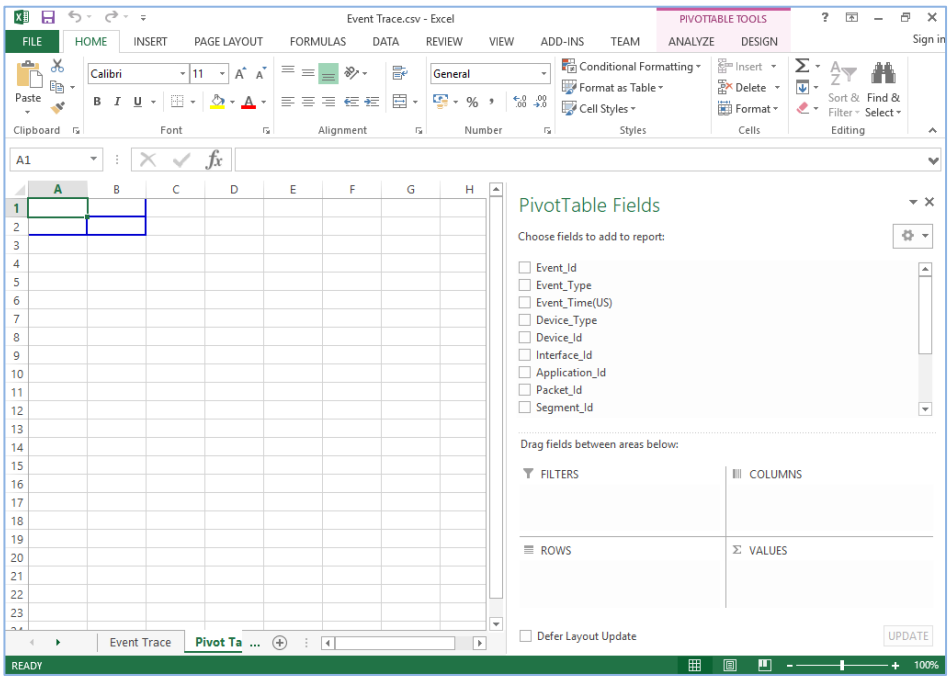
Note: Event tracing is available only in NetSim standard and pro versions.



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



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

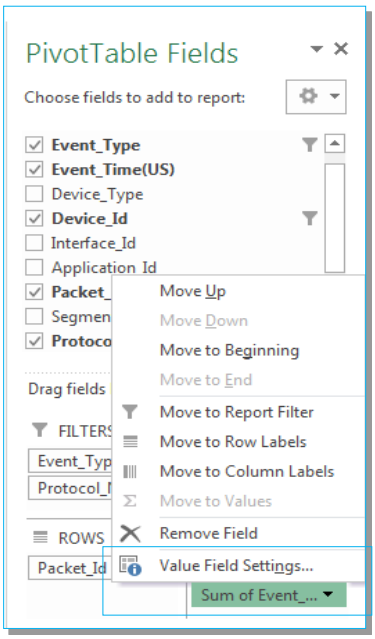


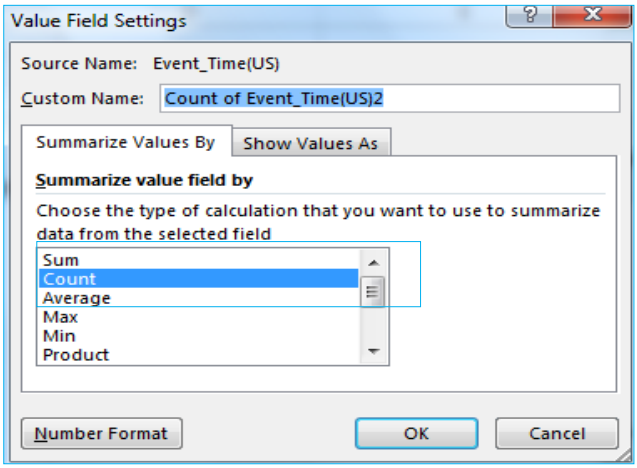
5. Once PivotTable worksheet open, you'll need to decide which **fields** to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

7.6.2.1 Application Delay Analysis:

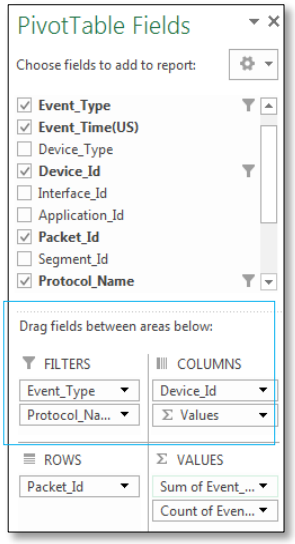
1. Drag and drop the Event_Type, Protocol_Name Fields into FILTERS, Packet_Id into ROWS and Device_Id into COLUMNS.

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.

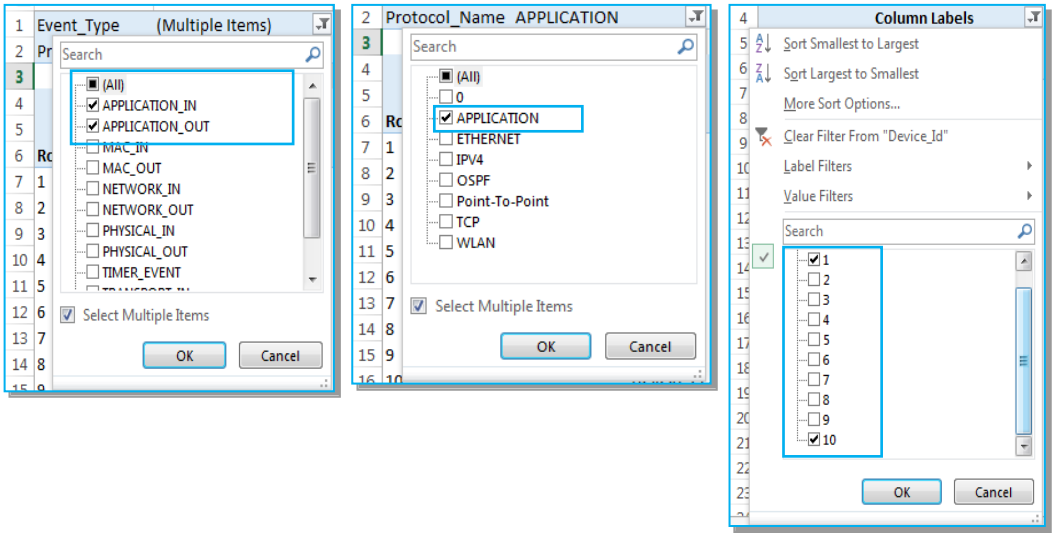




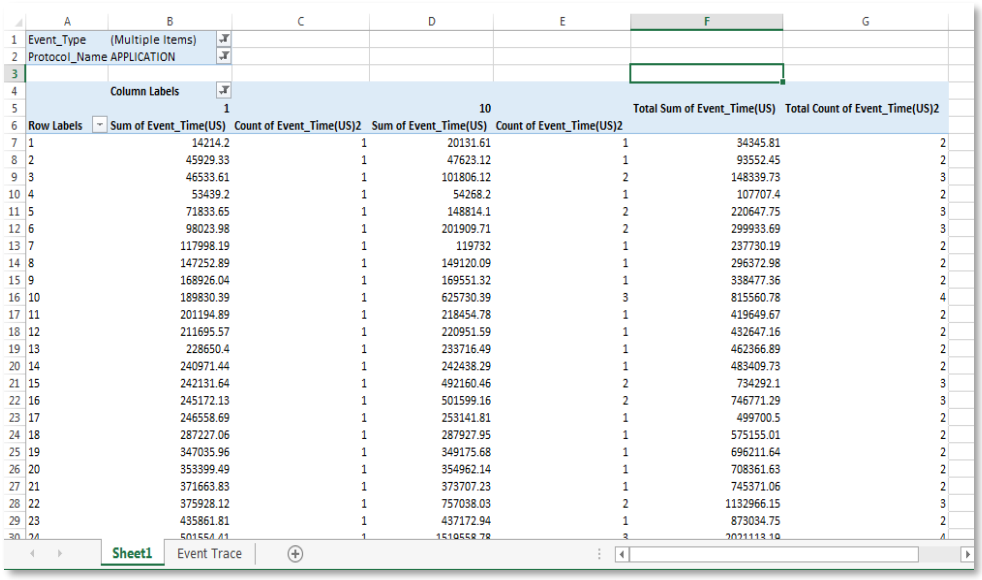
4. A window named **Value Field Settings** opens then select **Count** option and click **OK** button
5. Then finally the **Pivot Table Fields** will be as shown below.



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



And the Pivot Table created will be as shown (1 in the table is Source_Id and 10 is the Destination_Id)

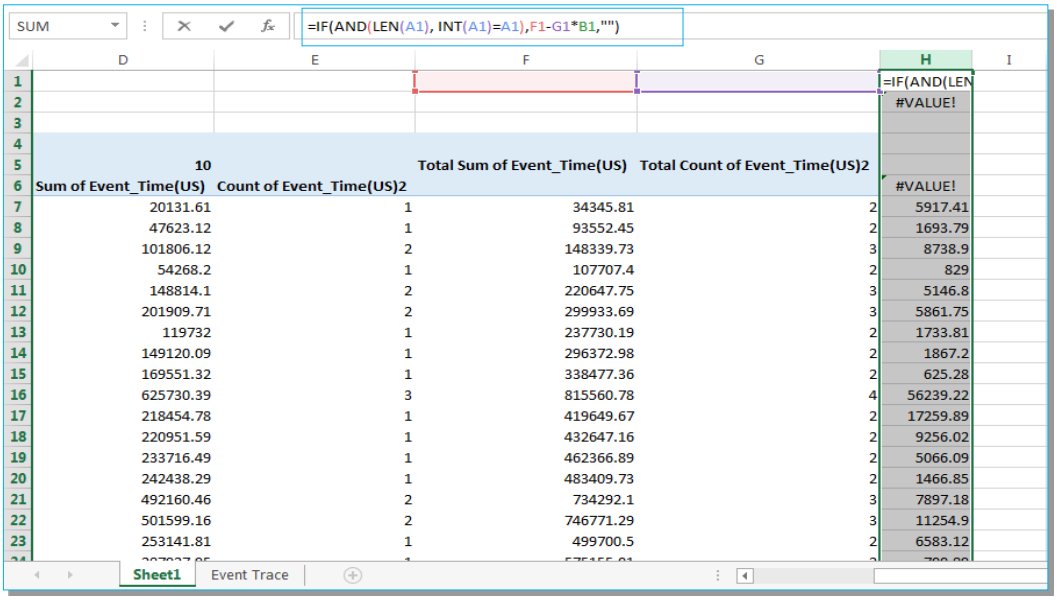


	A	B	C	D	E	F	G
1	Event_Type	(Multiple Items)					
2	Protocol_Name	APPLICATION					
3							
4		Column Labels					
5		1		10		Total Sum of Event_Time(US)	Total Count of Event_Time(US)2
6	Row Labels	Sum of Event_Time(US)	Count of Event_Time(US)2	Sum of Event_Time(US)	Count of Event_Time(US)2		
7	1	14214.2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

Sheet1 Event Trace

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 =IF(AND(LEN(A1), INT(A1)=A1),F1-G1*B1,"")

	D	E	F	G	H	I
1					=IF(AND(LEN	
2					#VALUE!	
3						
4						
5	10		Total Sum of Event_Time(US)	Total Count of Event_Time(US)2		
6	Sum of Event_Time(US)	Count of Event_Time(US)2			#VALUE!	
7	20131.61	1	34345.81	2	5917.41	
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	

Sheet1 Event Trace

App Delay = $\frac{Sum\ of\ the\ Delays\ of\ the\ sucessfully\ received\ application\ data\ packets\ by\ the\ destination}{Total\ number\ of\ sucessful\ appplication\ data\ packets\ received\ by\ the\ destination}$

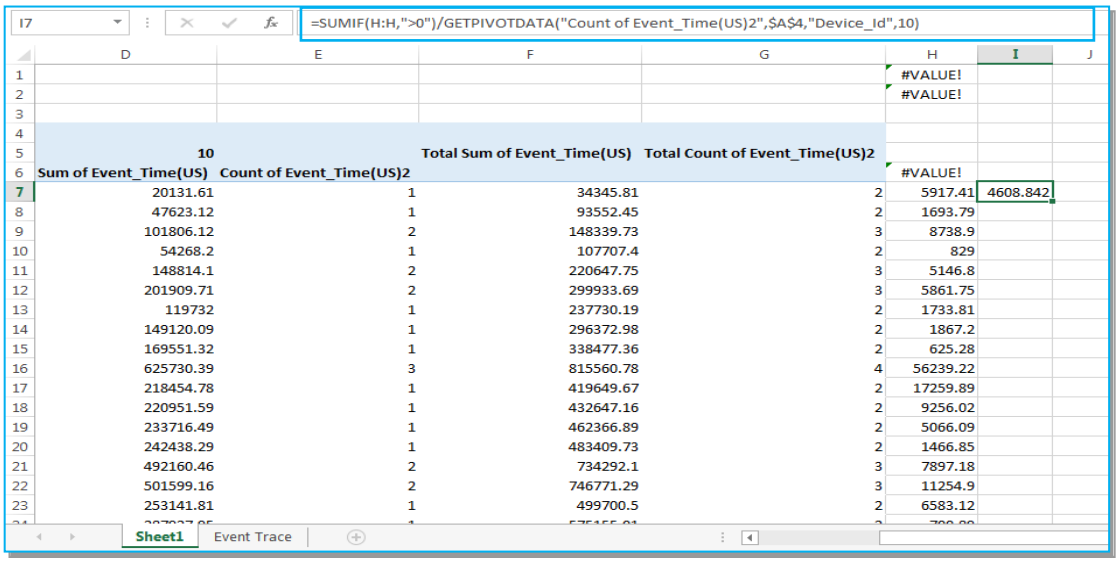
Note: If the packet size is > 1500 then fragmentation occurs and the packet is received as multiple segments. In NetSim the destination counts each segment as different packet.

Then in an empty cell enter

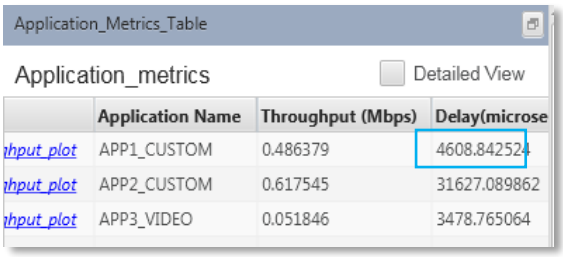
=SUMIF(H:H,">0")/GETPIVOTDATA("Count of Event_Time(US)2",$A$4,"Device_Id",10)

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



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.

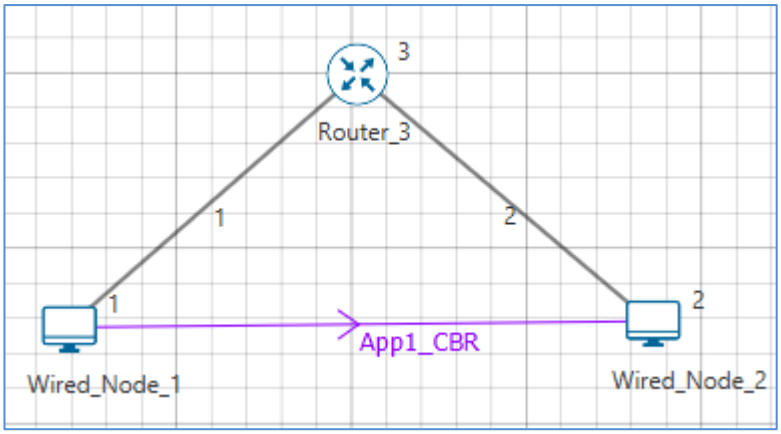


7.6.2.2 Application Jitter Analysis

In NetSim 'jitter' is defined as the variance of delay. Variance is statistically defined as the square of deviation from the mean.

Note: This calculation is only valid only when there is with no packet segmentation. Packets will be segmented in the Transport Layer if they exceed the Maximum Segment Size (MSS), which is 1460 Bytes by default.

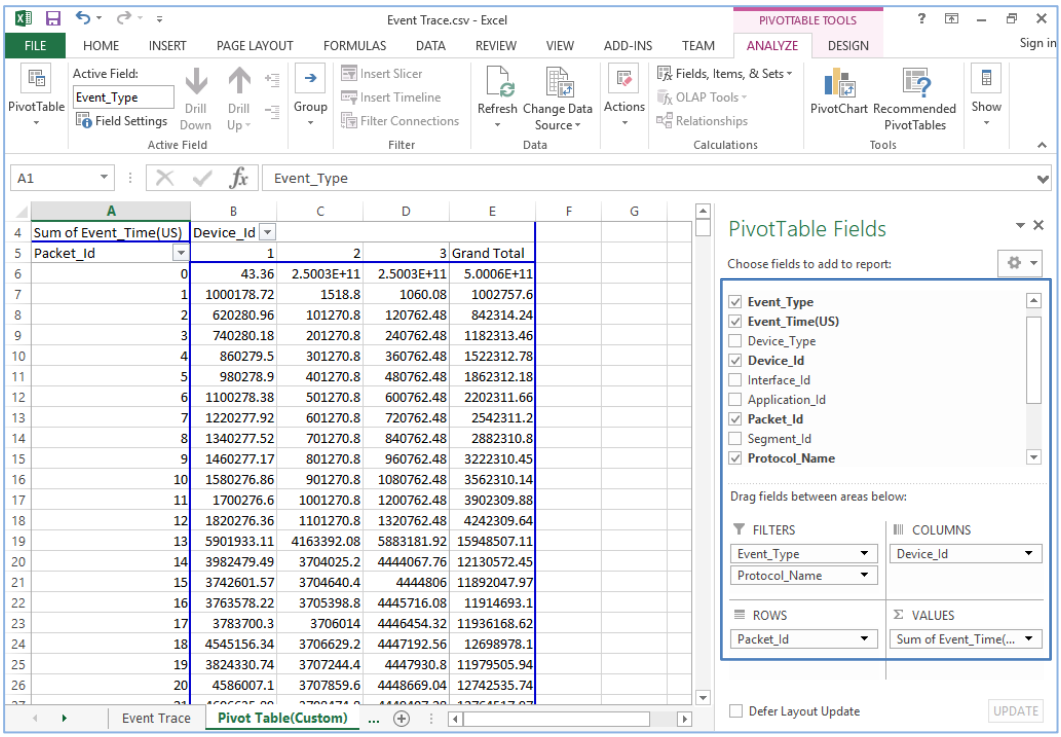
Sample Scenario:



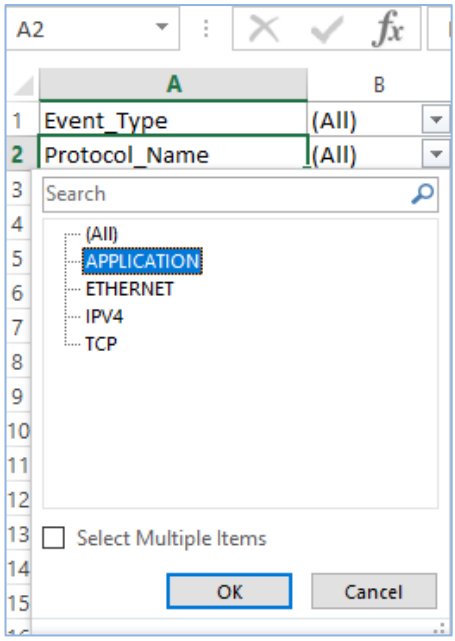
1. Open Event trace, Goto Pivot table(Custom) Sheet
2. For Jitter Calculation **Pivot table Fields**

   FILTERS to have of **Event_Type**, **Protocol_Name**, COLUMNS to have **Device_Id**, ROWS to have **Packet_Id** and VALUES to have **Event_Time (US)** (Note that **Event_Time** is added only once and not twice as was done for delay calculation)

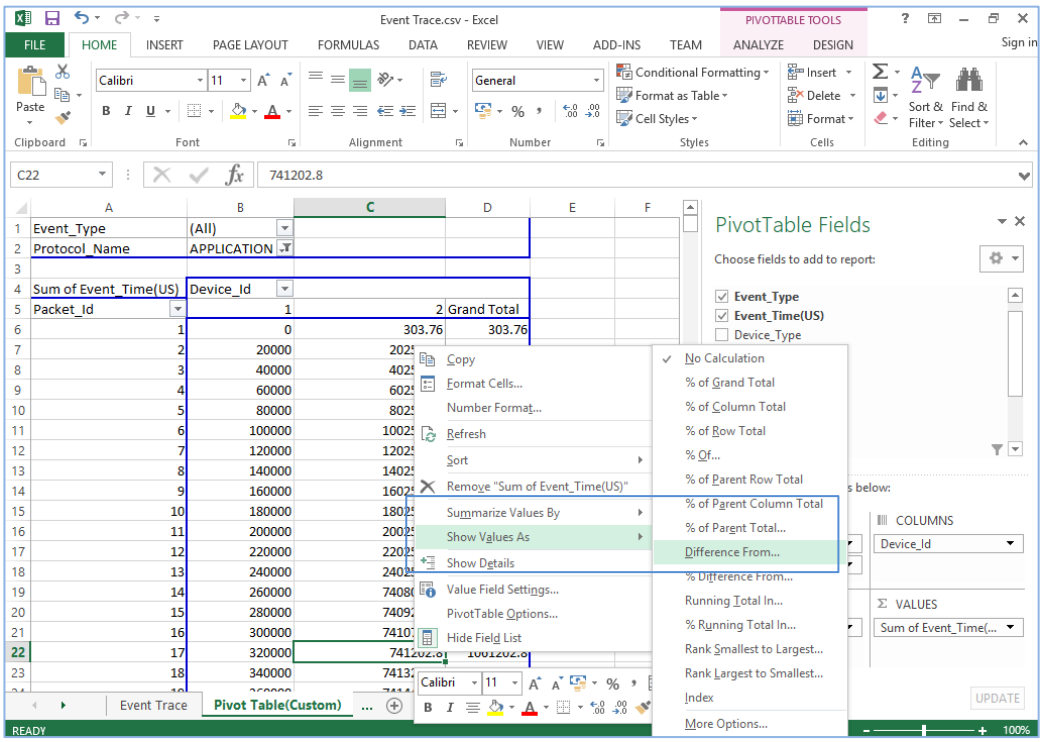
   Pivot table Fields should be as shown below



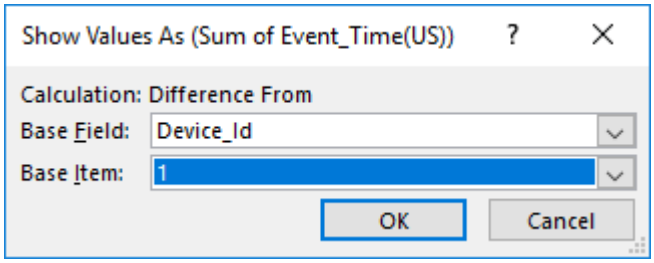
3. Then select **Protocol_Name**(2nd Row) in the top of the sheet and select **APPLICATION** only and then click Ok



4. Then select any cell of **Destination_Node_Id (2 in this case)** column and right click and select **Show Values As** option, it dropdown a list in which you select **Difference From** option



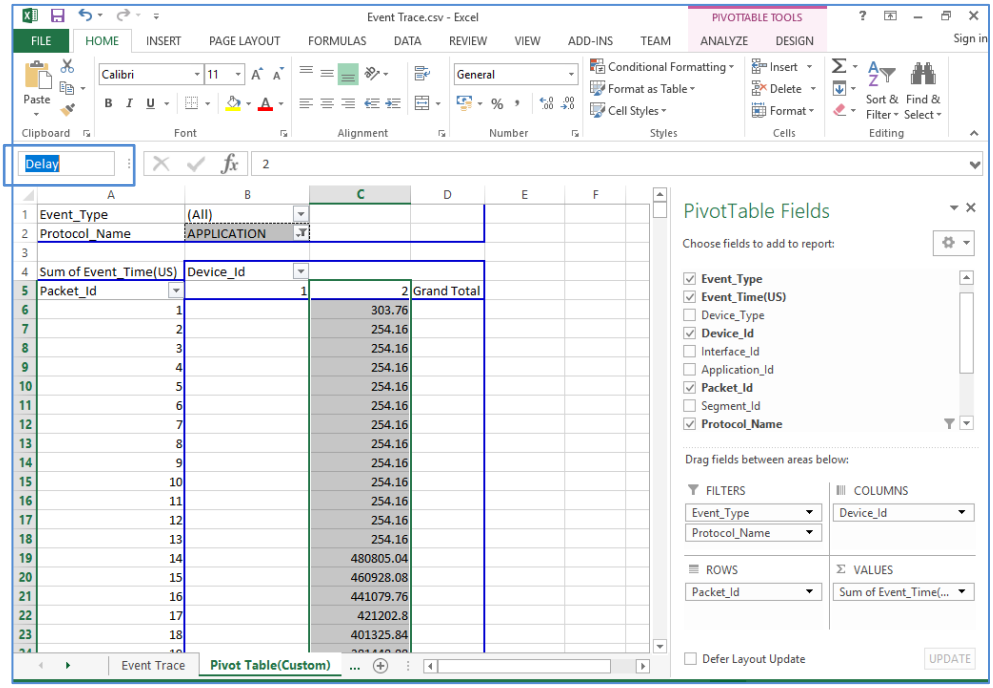
5. Then it opens a window named **Show Values As** in that select **Base Field** as **Device_Id** Base Item as the **Source_Node_Id (1 in this case)** and click **OK** button.



6. Then the Pivot Table will look like this and the **Destination_Node_Id (2 in this Case)** column shows the delay of each packet

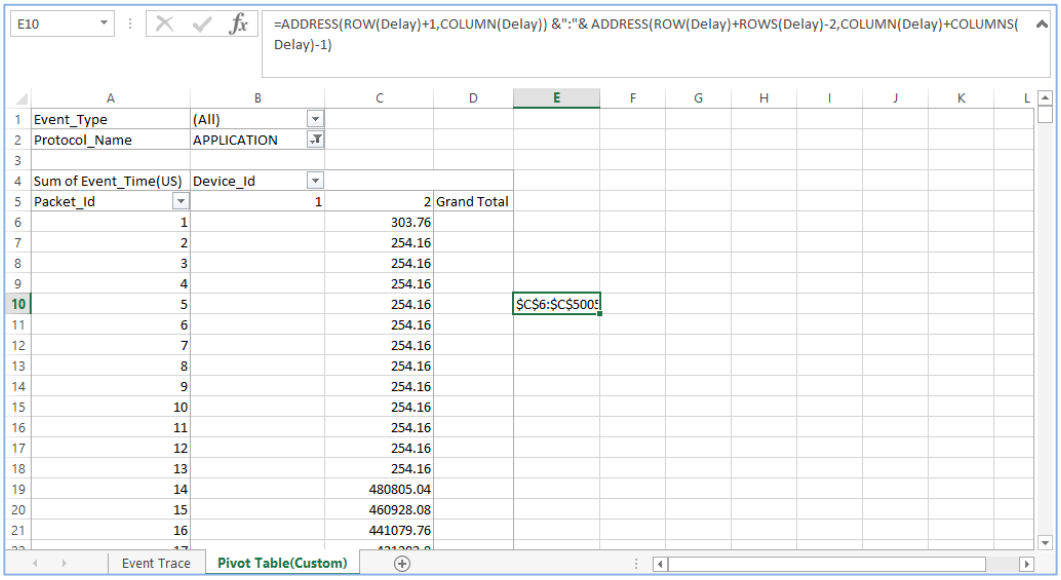
Event_Type	(All)		
Protocol_Name	APPLICATION		
Sum of Event_Time(US)	Device_Id		
Packet_Id	1	2	Grand Total
1		303.76	
2		254.16	
3		254.16	
4		254.16	
5		254.16	
6		254.16	
7		254.16	
8		254.16	
9		254.16	
10		254.16	
11		254.16	
12		254.16	
13		254.16	
14		480805.04	
15		460928.08	
16		441079.76	
17		421202.8	

7. Place the mouse cursor on the top of **Destination_Node_Id(2 in this Case)** then left click it will automatically selects the rows of the Destination_Node_Id, then select the **Name Box** and name the row as **Delay** as shown below



8. Then in an empty cell type the following formula and press enter

=ADDRESS(ROW(Delay)+1,COLUMN(Delay)) &":"& ADDRESS(ROW(Delay)+ROWS(Delay)-2,COLUMN(Delay)+COLUMNS(Delay)-1)

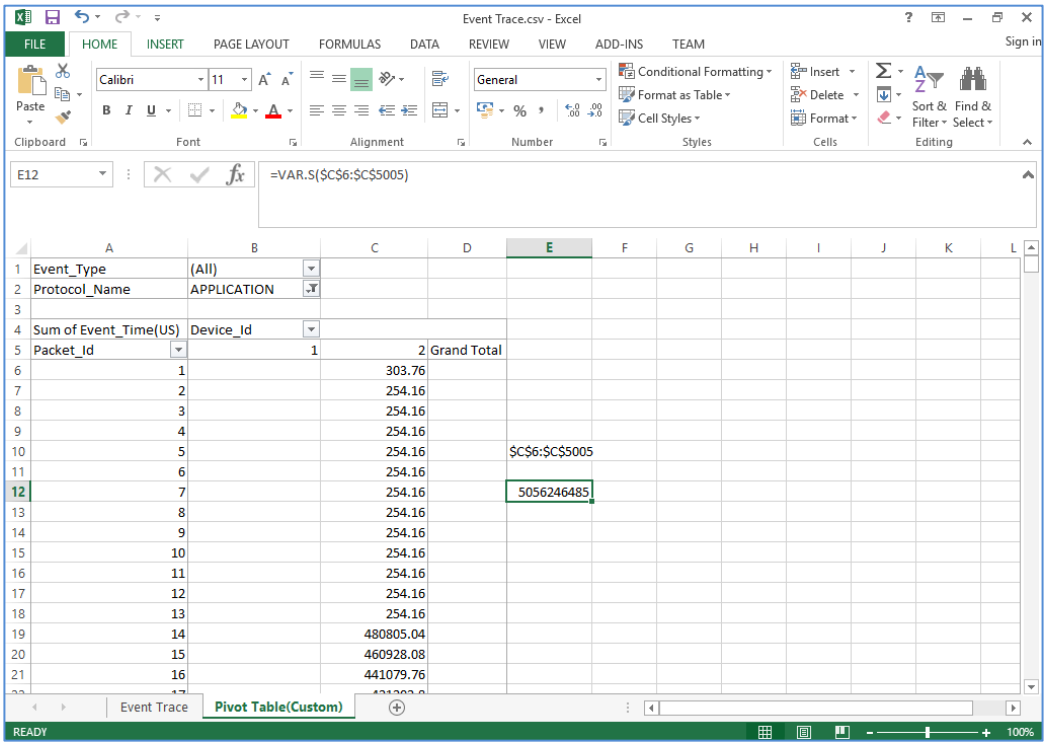


This will give you the **Row Address ($C$6:$C$5005)**

9. Then in an empty cell type the following formula and press enter

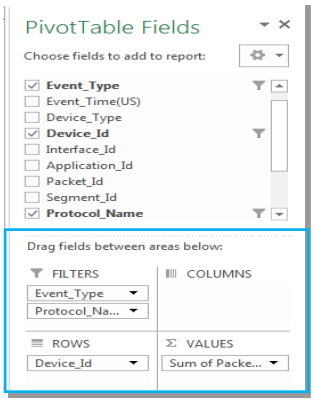
=VAR.S (Row Address), where the Row Address is one that you got in the previous step.

For example **=VAR.S ($C$6:$C$5005) .**This will give you the Application Jitter.

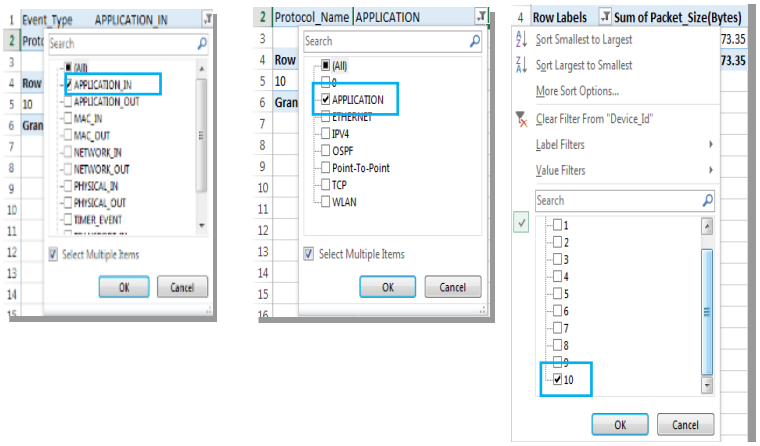


7.6.2.3 Application Throughput Analysis:

1. For Application Throughput drag and drop **Event_type**, **Protocol_Name** Fields in **FILTERS**, **Device_Id** in **ROWS**, **Packet_Size(Bytes)** into **VALUES.** Change the **Value Field Settings** of **Packets_Size(Bytes)** to **SUM** as mentioned in Delay Analysis.



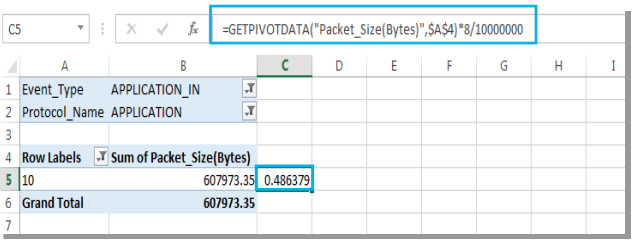
Then Select the Event_Type as APPLICATION_IN, Protocol_Name as APPLICATION and Device_Id as the Destination (in this case 10).



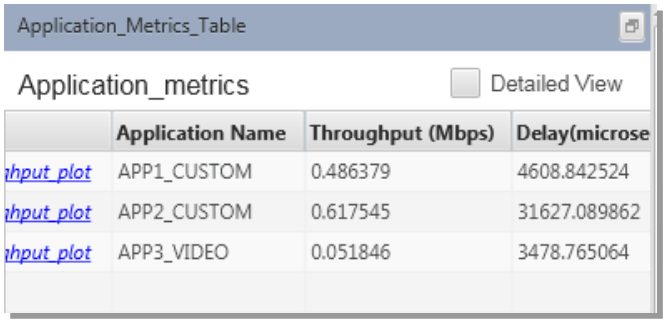
2. App Throughput = $\frac{Total Payload application succesfully received by the destination}{Simulation time}$

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)



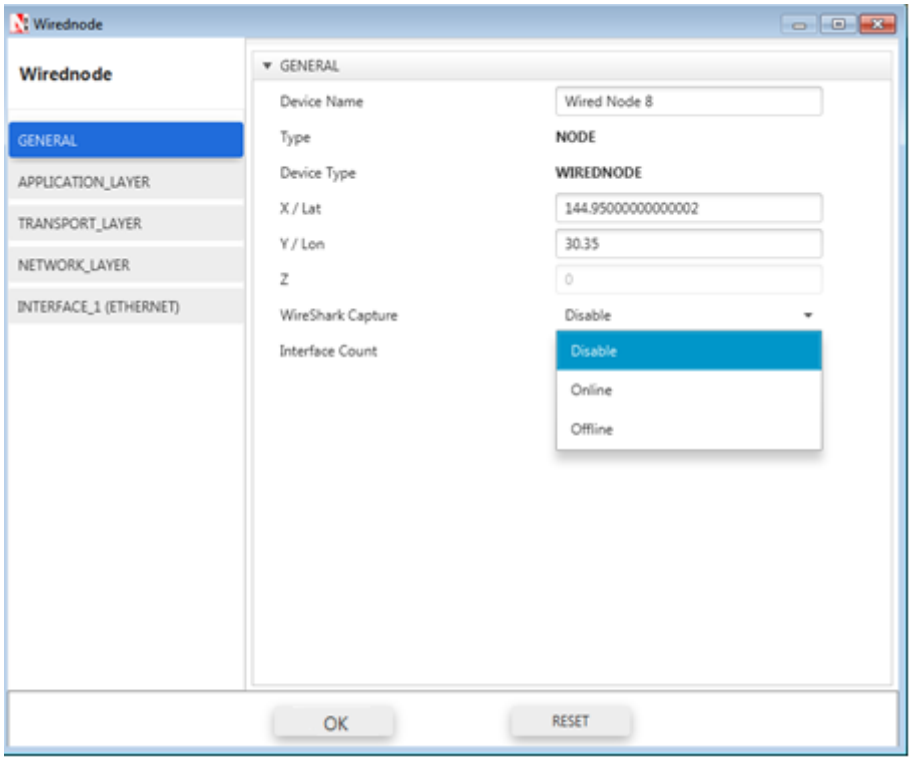
Compare with the Application throughput in the **Application_Metrics_Table**



7.7 Packet Capture & analysis using Wireshark

7.7.1 Enabling Wireshark Capture in a node for packet capture

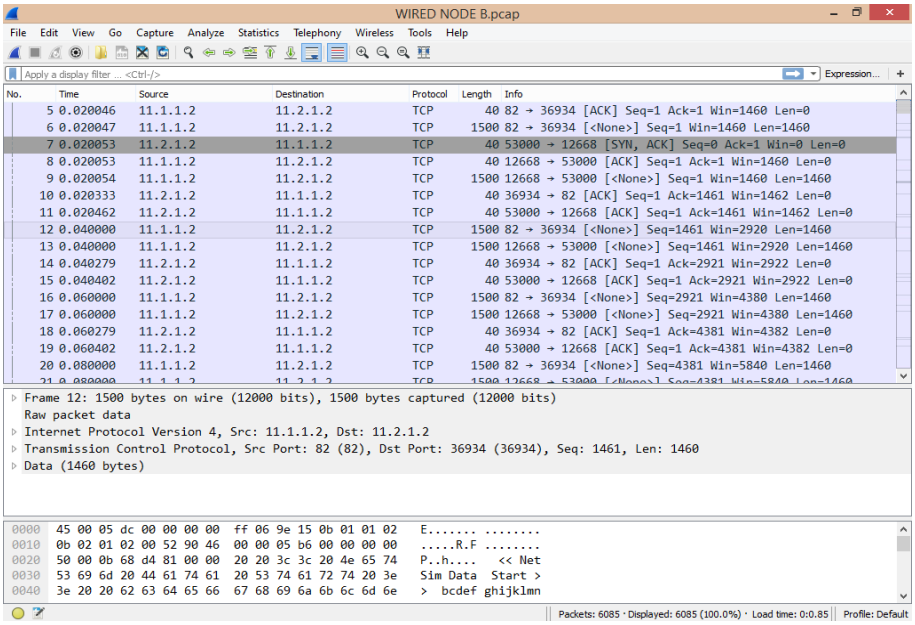
In NetSim, to enable packet capture in Wireshark, Right Click on the device where Wireshark should capture packets. In the properties, go to Global_Properties and set the Wireshark Capture parameter as Online



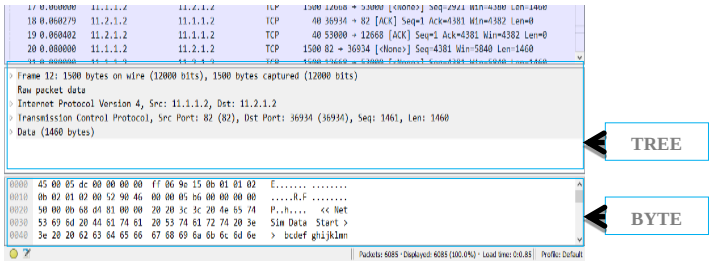
Wireshark Capture Option	
Online	Online will initiate a live interactive packet caputure, displaying packets while running simulation
Offline	Offline option will initiate silent packet caputure and generate a **pcap** file which can be opened using Wireshark post-simulation
Disable	Packets are not captured during simulation.

7.7.2 Viewing captured packets

If enabled, Wireshark Capture automatically starts during simulation and displays all the captured packets. To view the details of the packet displayed, click-on the packet as shown below:



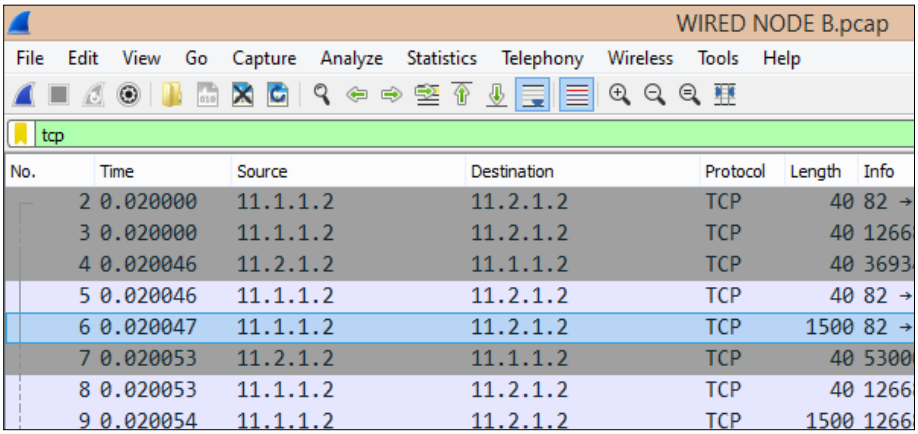
The detail of the contents of the selected packet can be seen in the below panes.



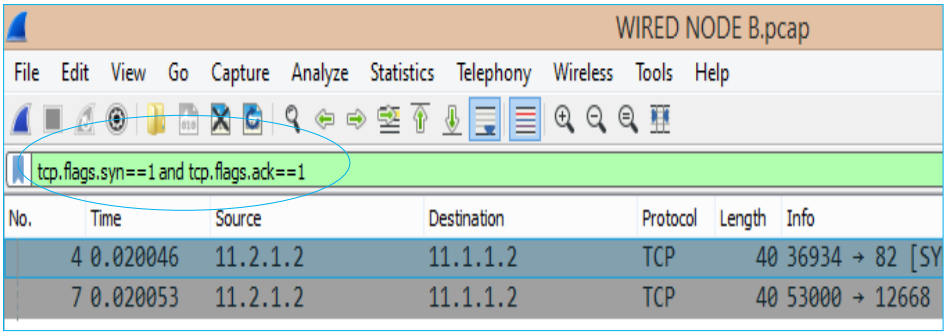
In the above figure, the details of the packet are displayed in both tree form and bytes form. In the tree form, user can expand the data by clicking on the part of the tree and view detailed information about each protocol in each packet.

7.7.3 Filtering captured packets

Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. Packets can be filtered by protocol, presence of a field, values of field's etc.To select packets based on protocol, type the protocol in which you are interested in the Filter: field of the Wireshark window and presenter to initiate the filter. In the figure below, tcp protocol is filtered.



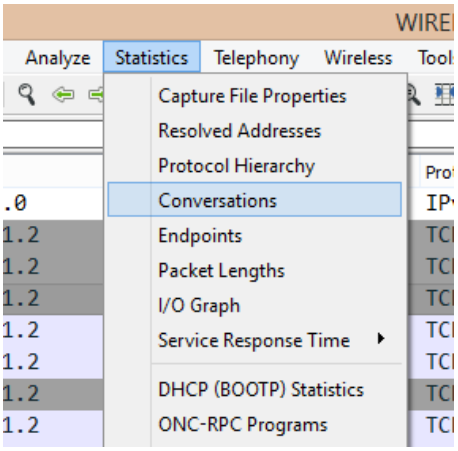
Youcan also build display filters that compare values using a number of different comparison operators like ==, != , >, <, <=, etc.Following is an example displaying filtered packets whose SYN Flag and ACK Flag are set to 1 in a TCP Stream.



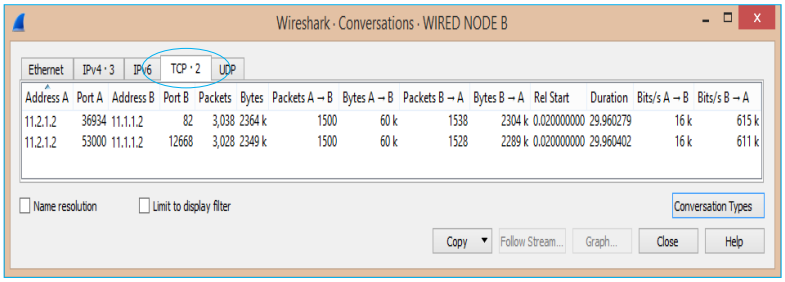
7.7.4 Analyzing packets in Wireshark

Analyzing Conversation using graphs

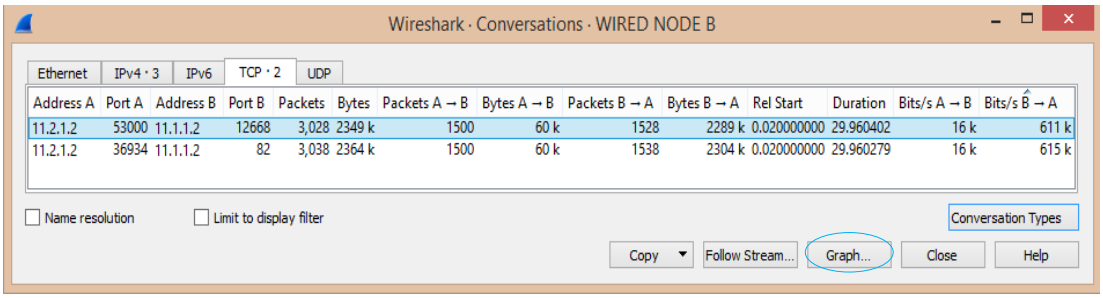
A network conversation is the traffic between two specific end points.For example, an IP conversation is all the traffic between two IP addresses.In Wireshark, Go to Statistics Menu→ Conversations



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.



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

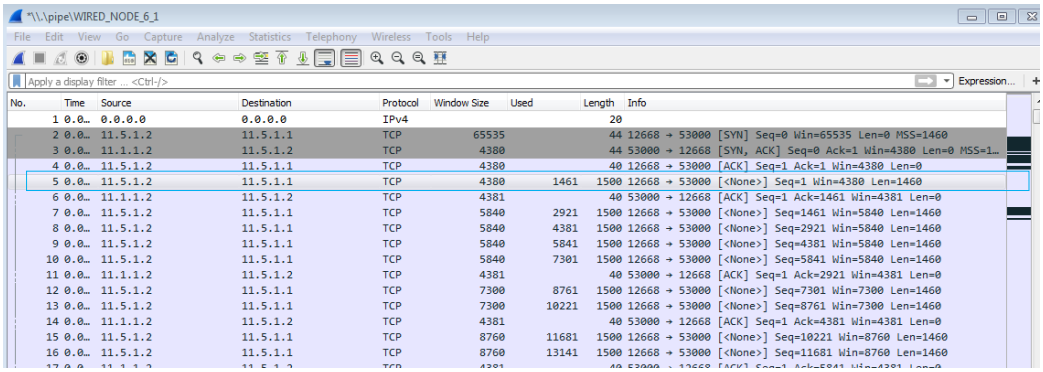


The different types of graphs possible are

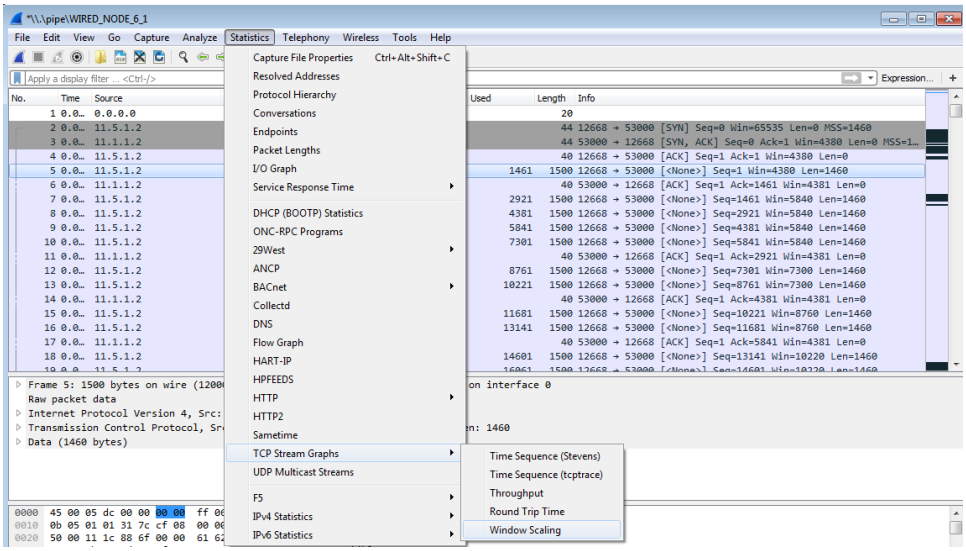
- Round Trip time
- Throughput
- Time/Sequence (Stevens)
- Time/Sequence (tcptrace)
- Window Scaling

7.7.5 Window Scaling

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



Choose statistics→TCP Stream Graph→Window Scaling



Click on Switch Direction in the window scaling graph window

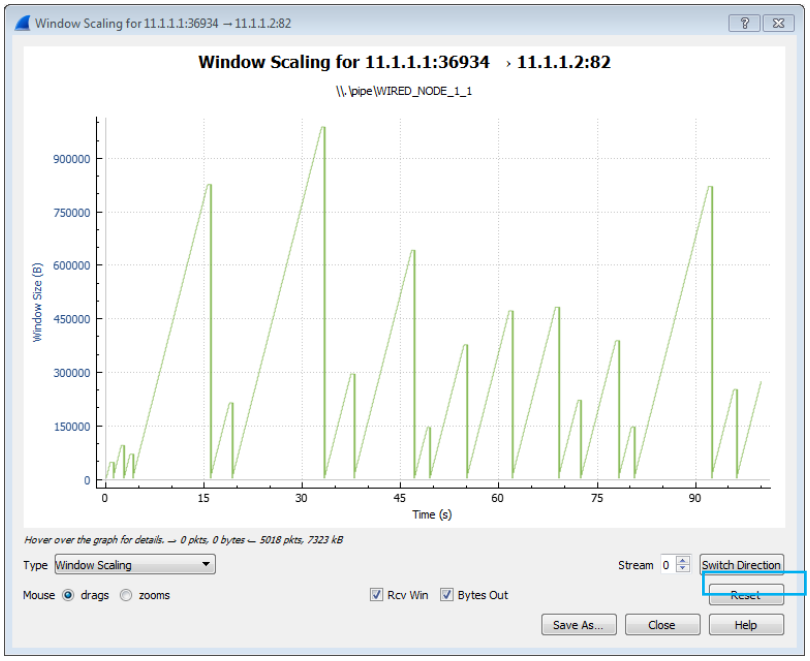
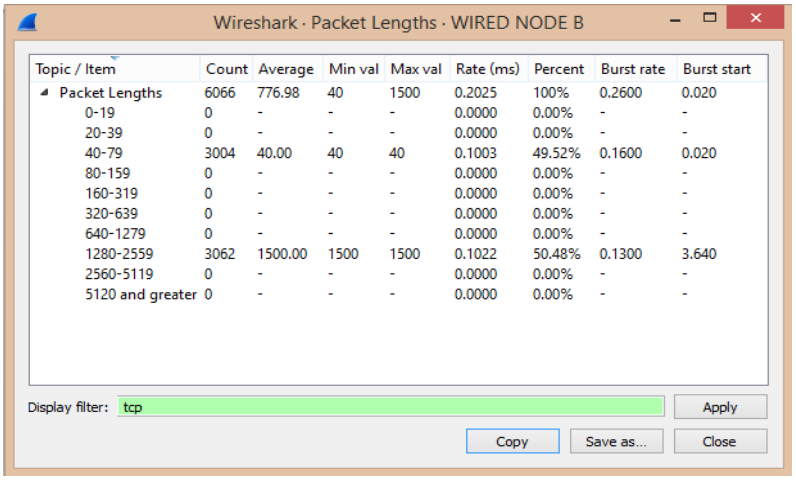


Fig: TCP congestion window Plot

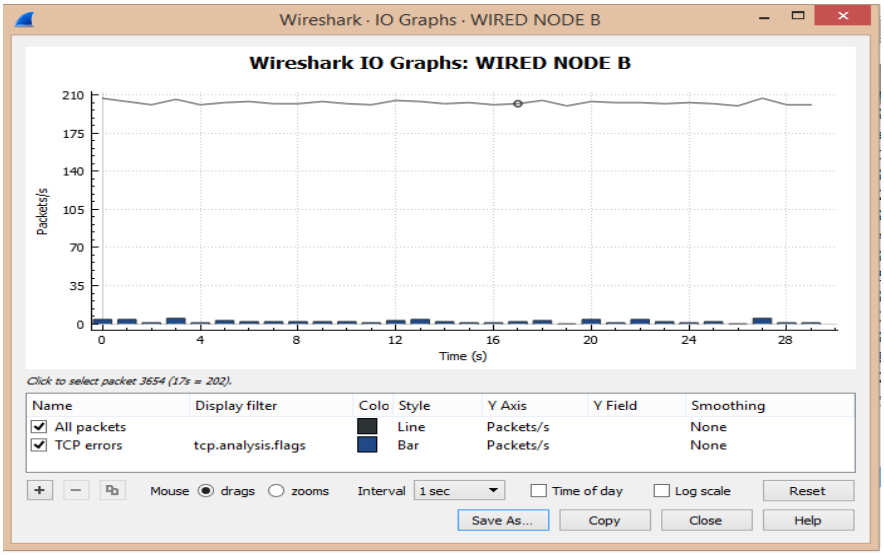
Comparing the packet lengths

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:



Creating IO graphs

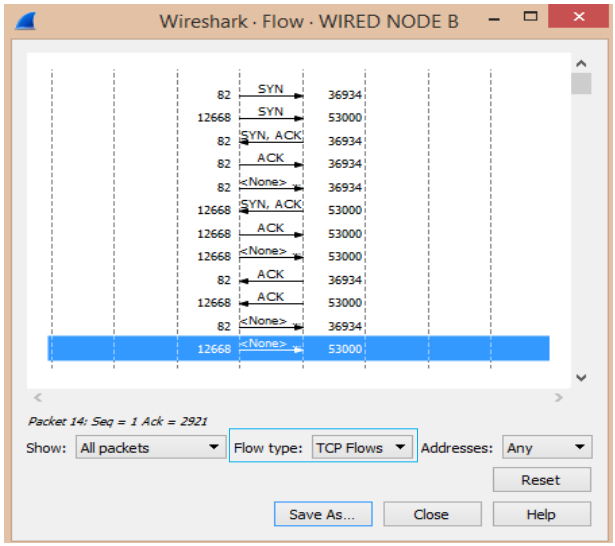
To get the graph, go to ***Statistics Menu → IO Graph***.



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:



8 Writing Custom Code in NetSim

8.1 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:

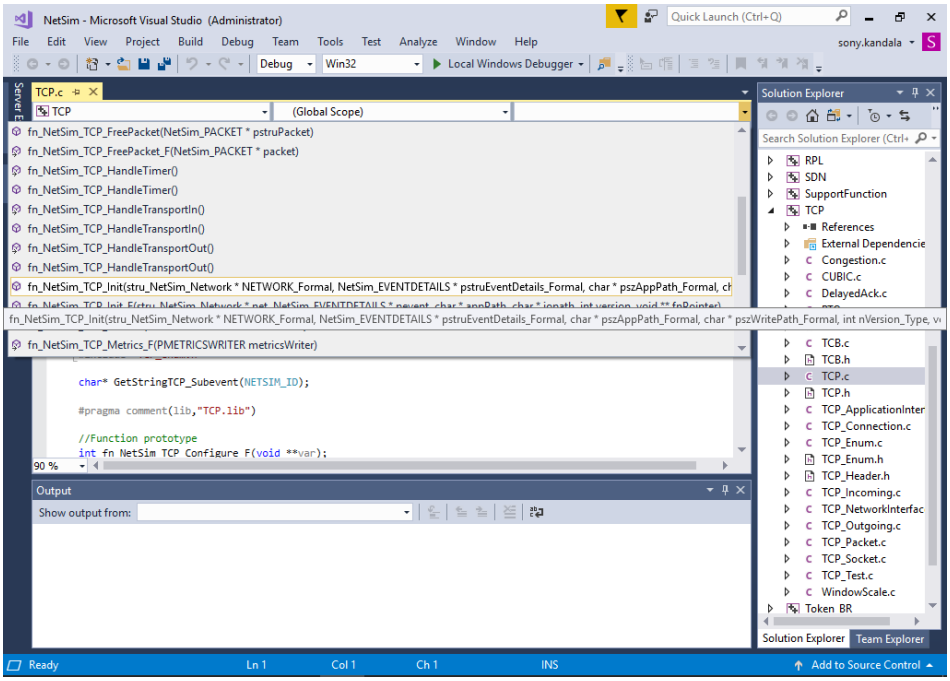
8.1.1 Modifying code

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.

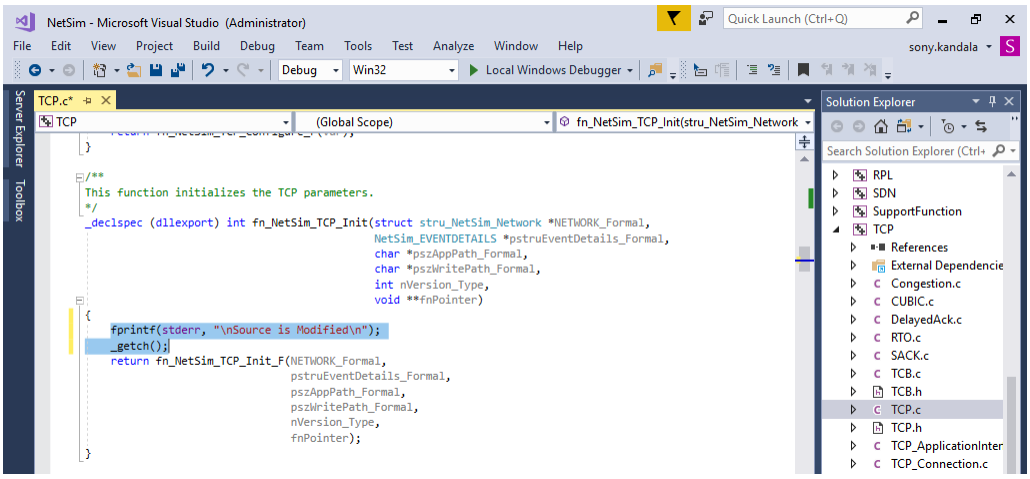
Note: Make sure that Visual Studio 2015 or above is installed in your system.

Refer section 3.12 section "How does a user open and modify source codes" to open NetSim Source Codes

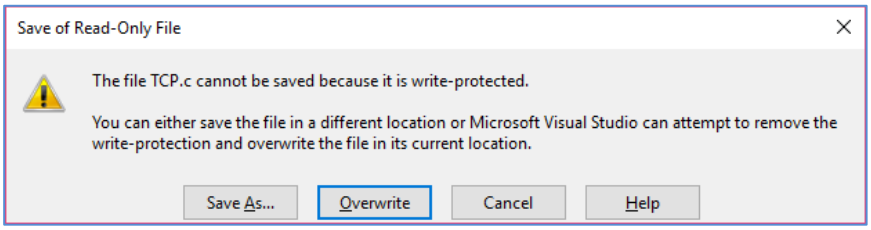
1. After this you may modify the source codes of any project. You can also add new files to the project if required. As an example let us make a simple source code modification to TCP. Inside Solution Explorer pane in Visual Studio, double click on TCP project. Then open TCP.c file by double clicking on it. Using the drop down list of functions that are part of the current file, choose **fn_Netsim_TCP_Init()**.



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”.

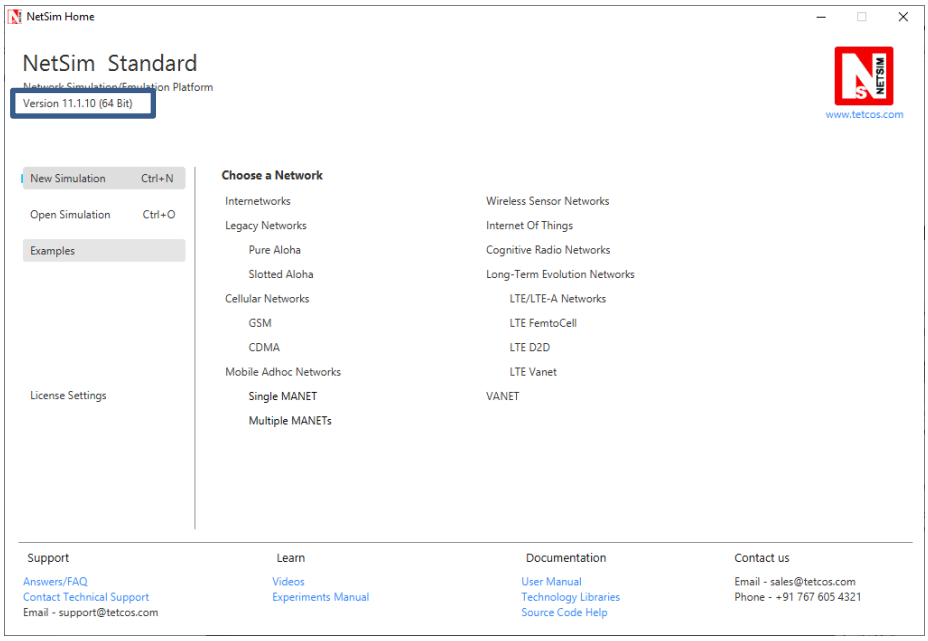


3. Once this is done click to save the changes and overwrite the file (in case of write protection).

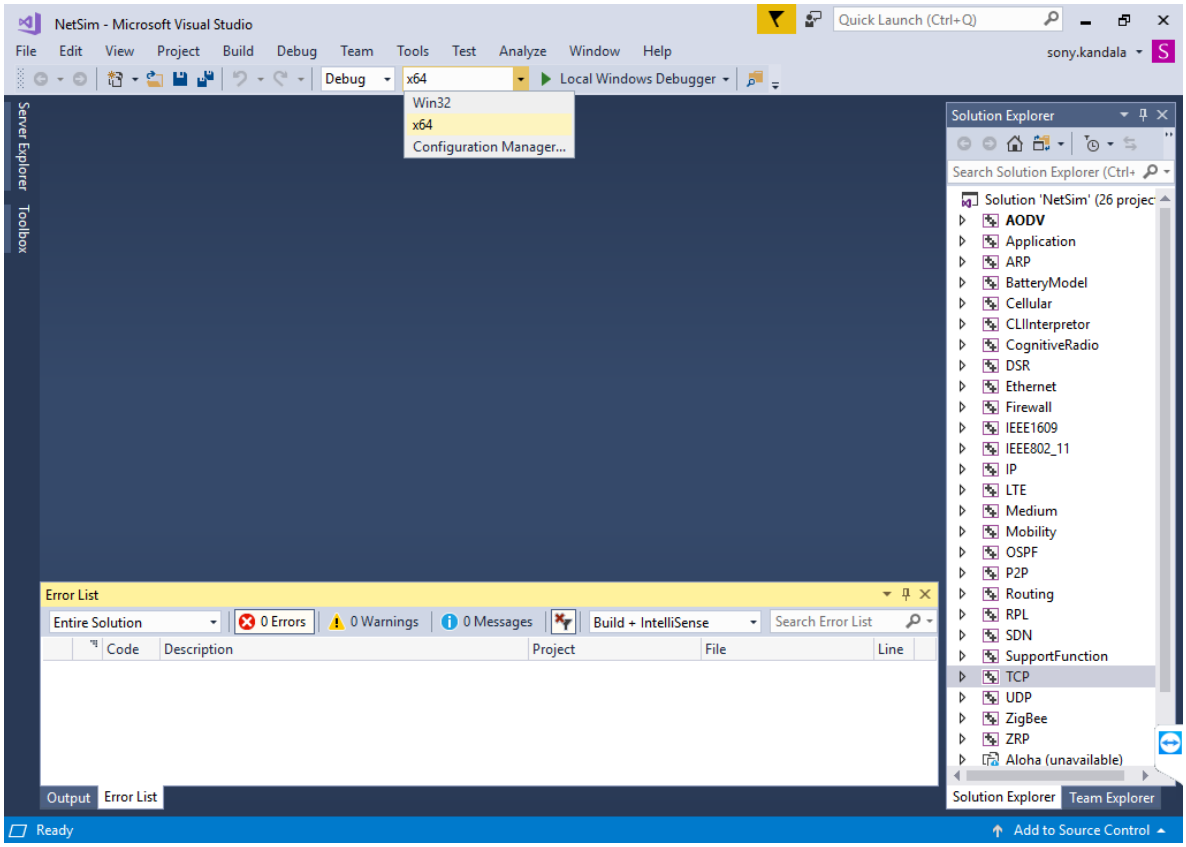


8.1.2 Building Dlls

1 Identify the build of NetSim that is installed in your system from NetSim Home Screen as shown below:

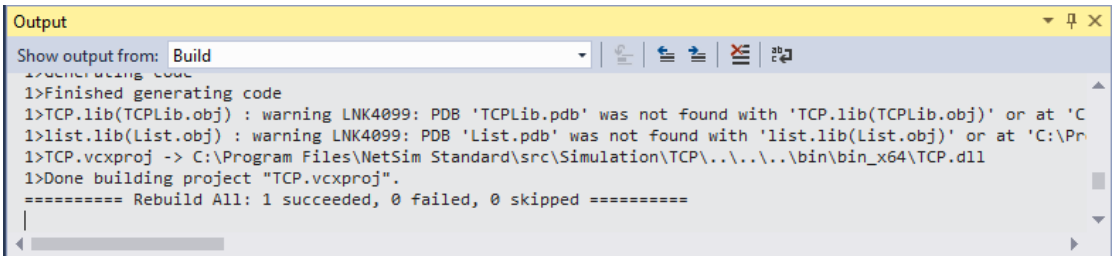


2 Based on the build of NetSim installed modify the solution platform in Visual studio from the drop down as shown below:



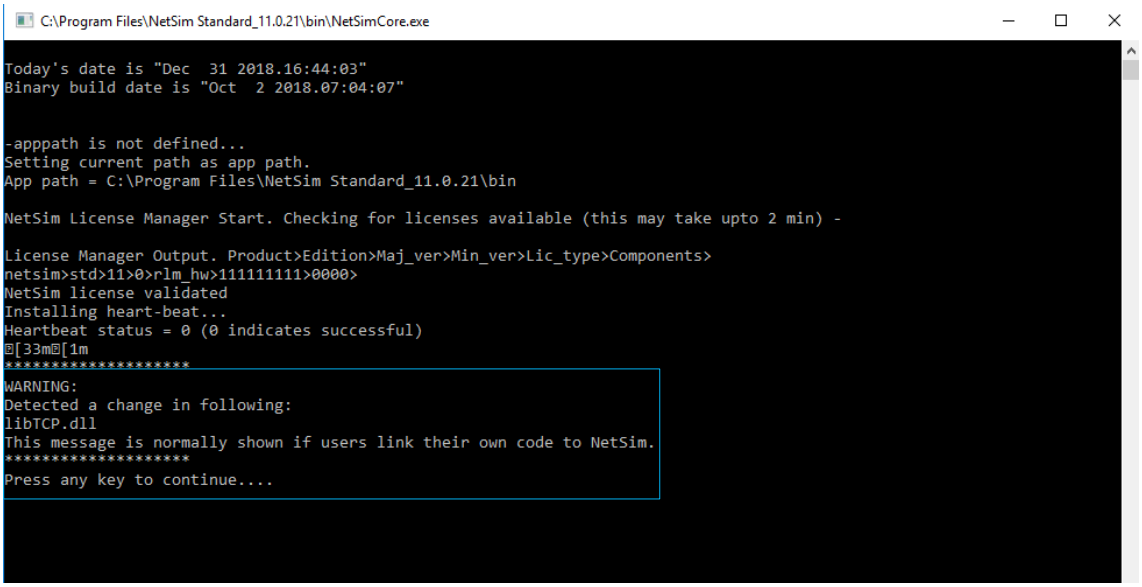
3 Choose x64 for 64 bit version of NetSim and Win32 for 32 bit version of NetSim. After changing the solution platform, changes will be automatically applied to all projects that are displayed in the Solution Explorer.

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\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_32_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:



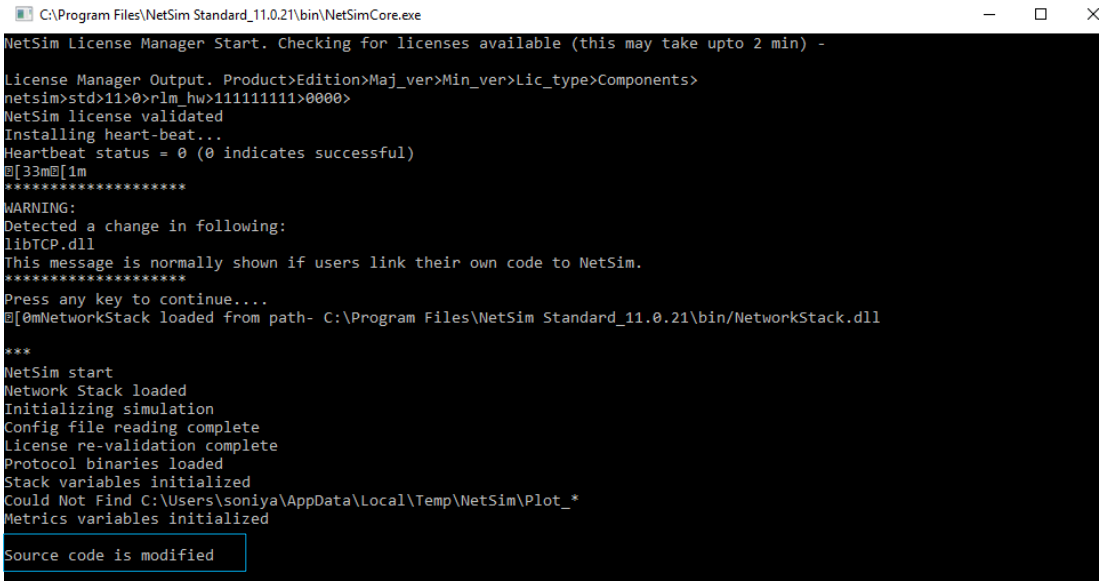
8.1.3 Running Simulation

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:



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:



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_x32 for 32-bit and bin\bin_x64 for 64-bit).

8.1.4 Source Code Dependencies

The following are the list of projects that are part of NetSim source codes present in <NetSim_Install_Directory>/src/Simulation directory and their dependencies:

PROJECT	DEPENDENCY
Application	IP
Cellular	Application
CLIInterpertor	Firewall, IP
Cognitive Radio	Application
Ethernet	Firewall
IEEE802_11	Battery Model
OSPF	IP
Routing	IP
RPL	IP
ZigBee	Battery Model
ZRP	IP
Aloha	-
AODV	-

ARP`	-
Battery Model	-
CSMACD	-
DSR	-
Firewall	-
IEEE1609	-
IP	-
LTE	-
Mobility	-
P2P	-
SDN	-
Support Function	-
TCP	-
Token_BR	-
UDP	-
DTDMA	-
TDMA	-

For Eg: To perform modfications to Application Project, IP folder will also be required in addition to lib folder, Include folder and NetSim.sln file.

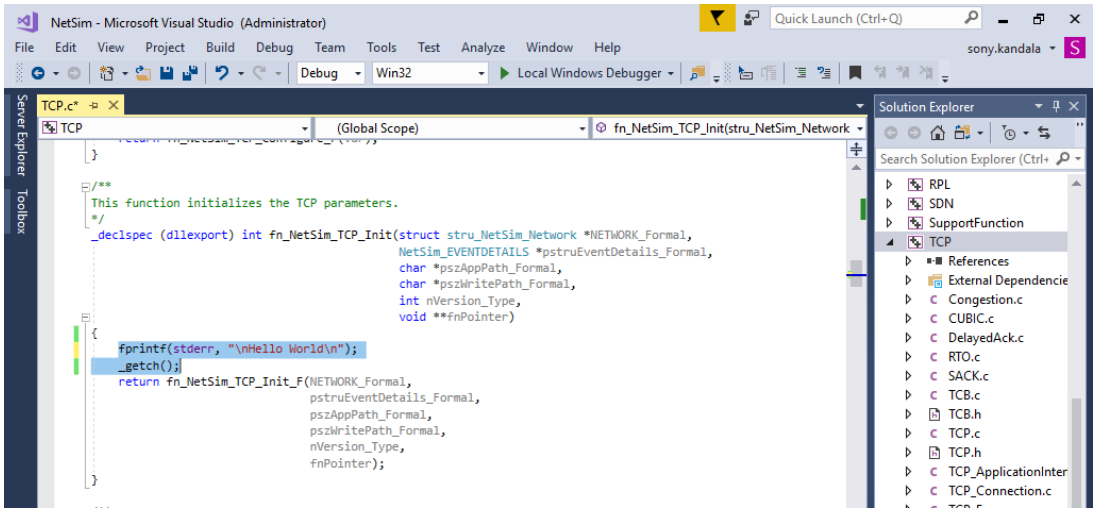
8.2 Implementing your code - Examples

8.2.1 Hello World Program

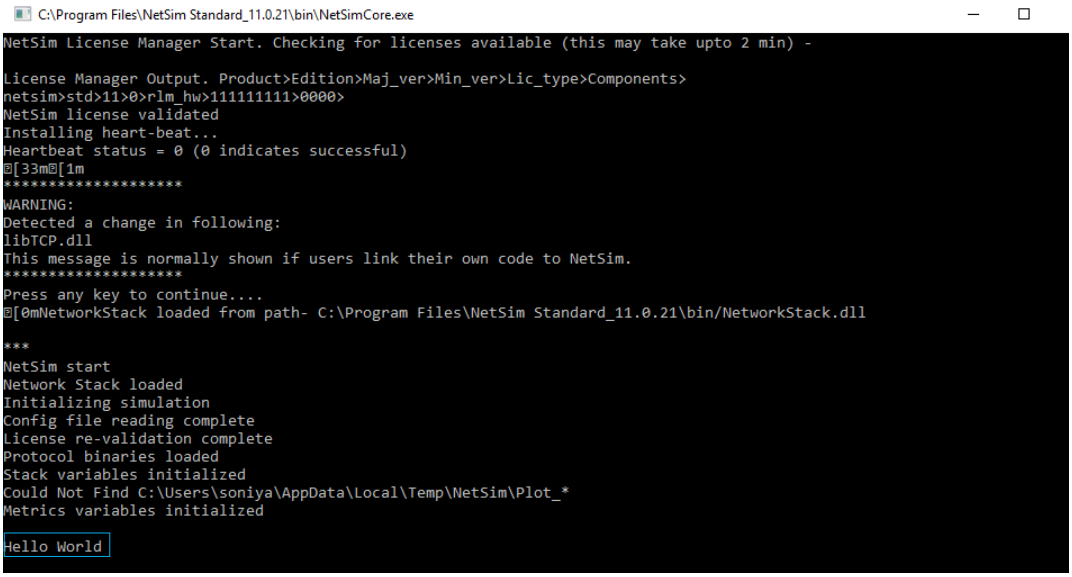
Objective: Print Hello World from TCP protocol.

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();



Build DLL as explained in Section 8.1 and run the simulation, you can see the following output on the console.



Press enter then simulation will continue.

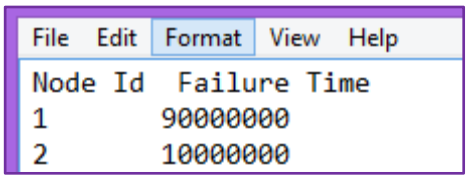
8.2.2 Introducing Node Failure in MANET

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\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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 from10th sec onwards and fail Node Id 1 from 90th sec onwards, the NodeFailure.txt file will be as follows:



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 8.1.

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.

8.3 Debugging your code

This section is helpful to debug the code which user has written. To write your own code please refer Section 8.1.

8.3.1 Via GUI

Debugging your code via GUI there are two methods available

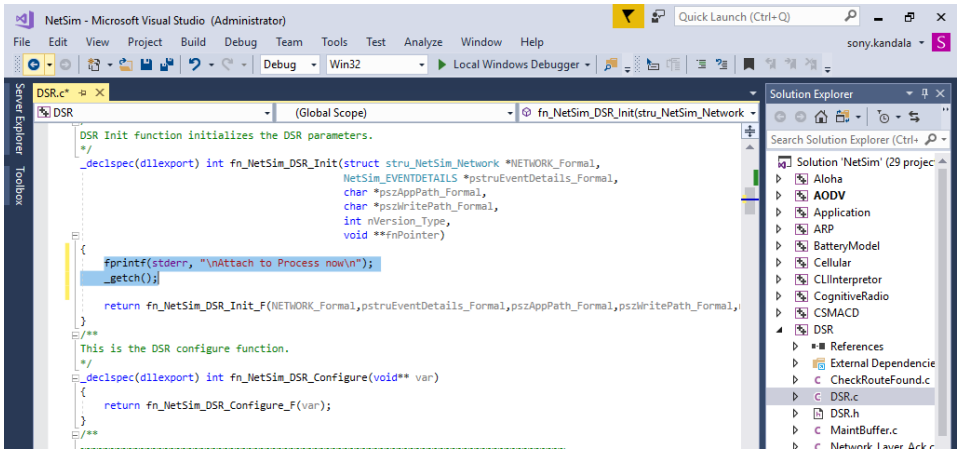
- **Using _getch()**
- **Using Environment Variables (NETSIM_BREAK)**

8.3.1.1 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

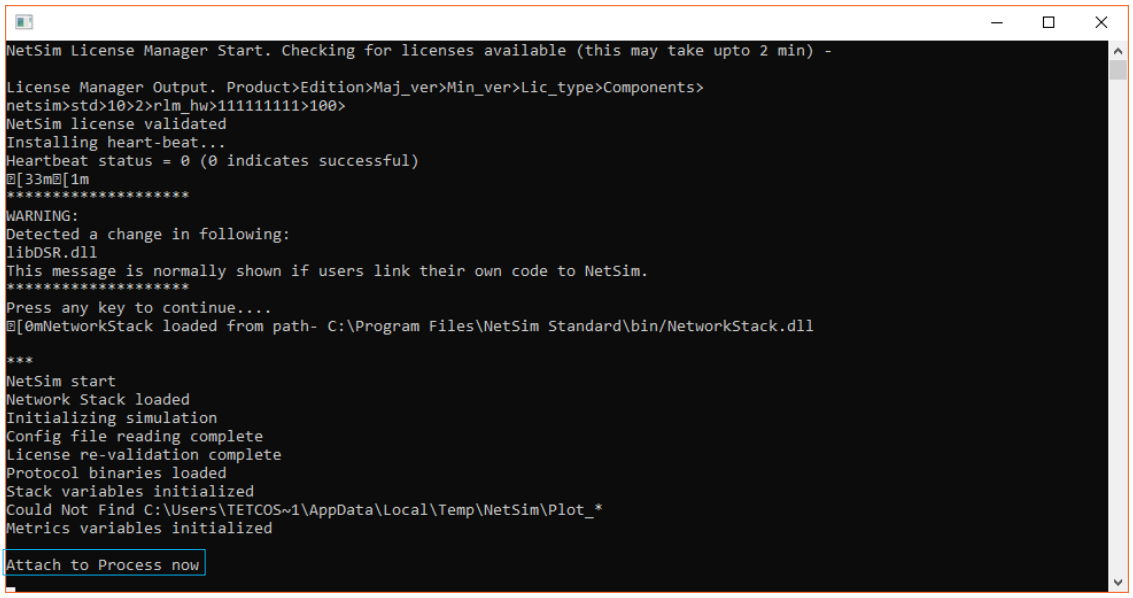
fprintf(stderr,"\nAttach to Process now\n");

_getch();



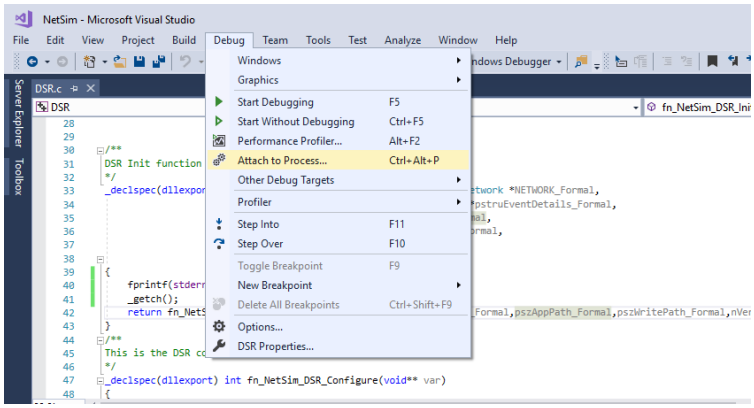
Step 2: Build the DSR protocol as explained in Section 8.1. Do not close Visual Studio.

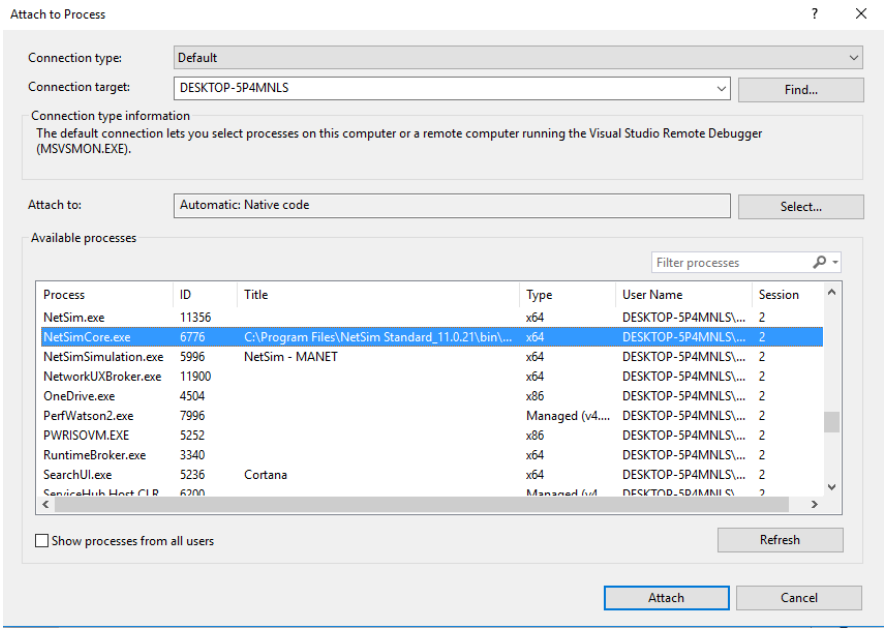
Step 3: In NetSim, create a network scenario where the protocol is being used and start the simulation. In the console window user would get a warning message shown in the below screenshot and the simulation will pause for user input (because of _getch() added in the init function)



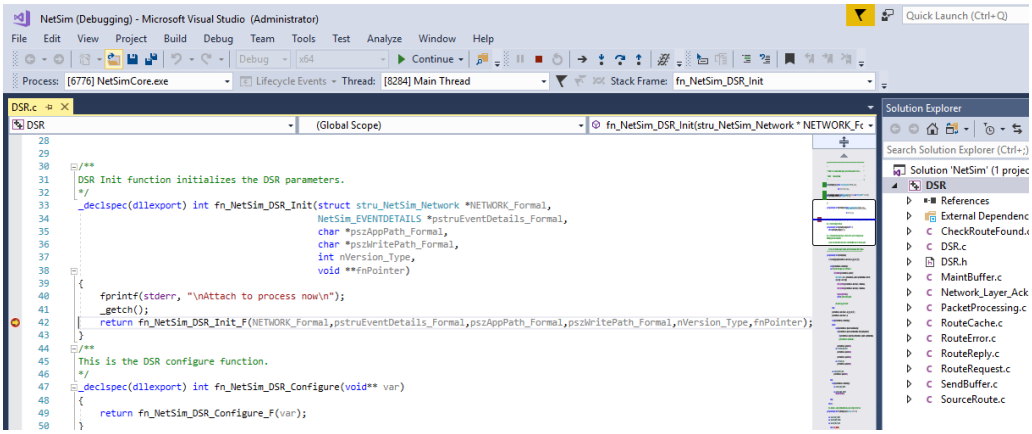
Step 4: In Visual Studio, put break point inside the source code where you want to debug.

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





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. 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 and continue execution, press Shift+F5 (key). This then gives the control back to NetSim, for normal execution to continue.

8.3.1.2 Using Environment Variable

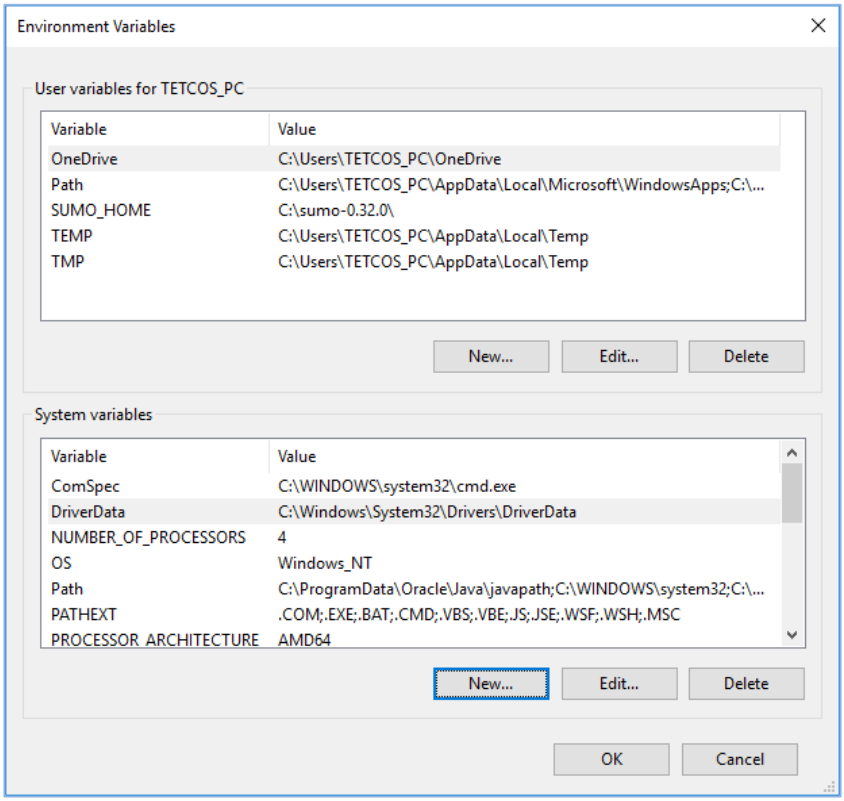
This section is helpful to Debug Using Environment Variable (NETSIM_BREAK). To set Environment variable follow the steps as shown

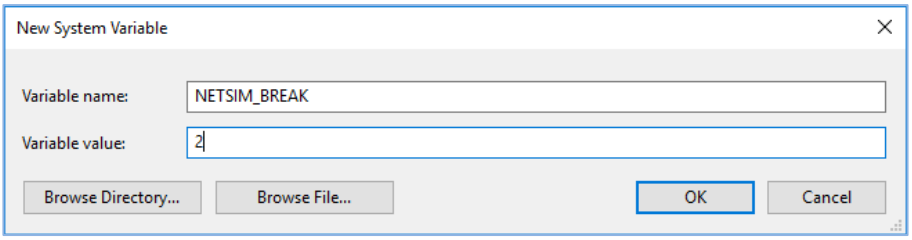
Note: Setting NETSIM_BREAK Environment Variable will cause the simulation to slow down and it is recommended to remove this Environment Variables after debugging the simulation

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

Step 2: Go to Advanced System setting → Advanced Tab → Environment Variables option

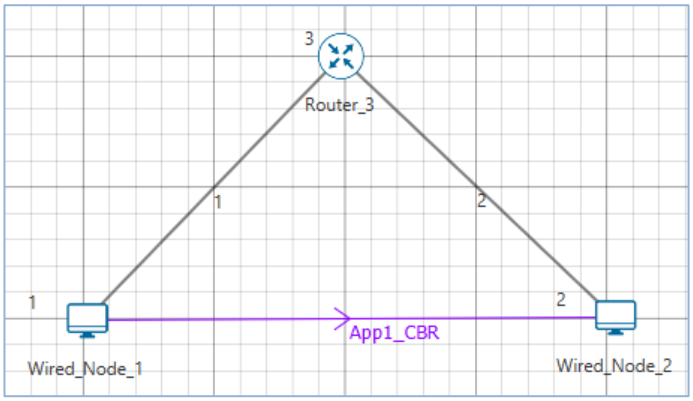
Step 3: Click New in System variables. Type “NETSIM_BREAK” as Variable name and any positive integer as variable value (e.g. 2). Click OK. The value of the variable is the event ID at which you want NetSim Simulation to break. In this example we have set the value to 2, which means that the simulation will break at the previous event.





Step 4: Open NetSim and then open the source codes. Please refer section 3.12 “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)



Step 6: In this example we are placing a break point in TCP source code and thus TCP should be enabled in Transport Layer of the devices.

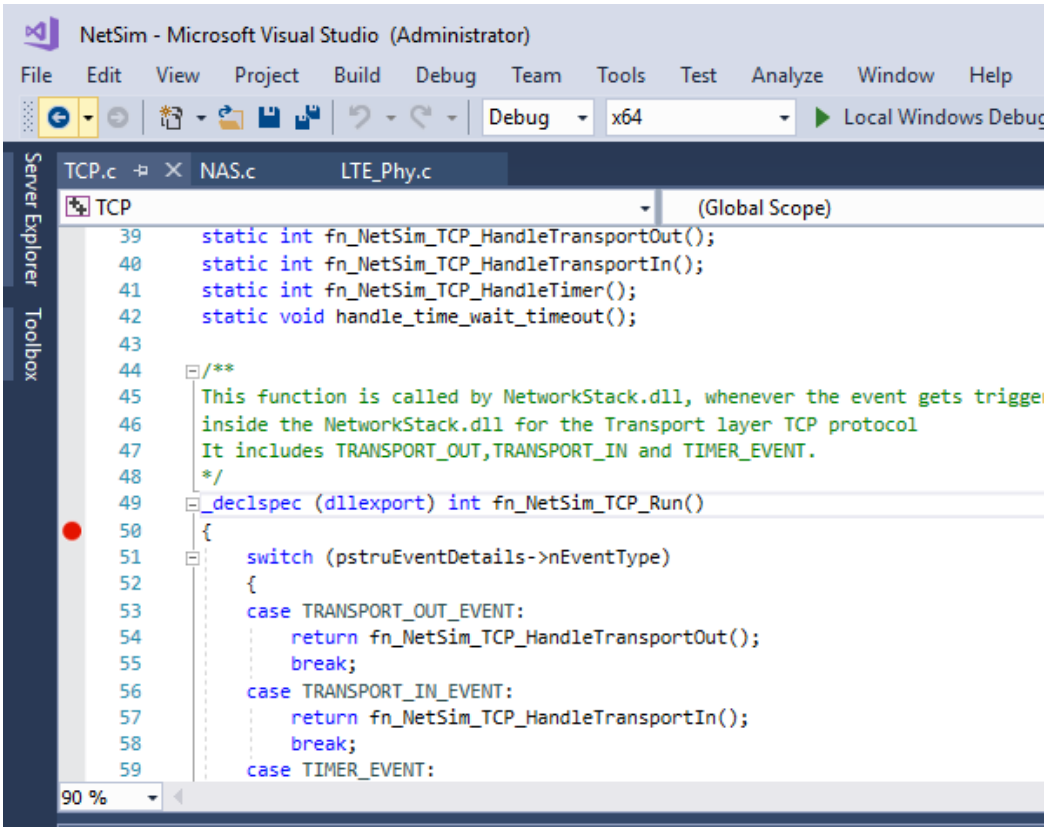
Step 7: Enable Event trace option and run the simulation

Simulation will break at event ID 1 as we have set the environment variable to 2 as shown below:

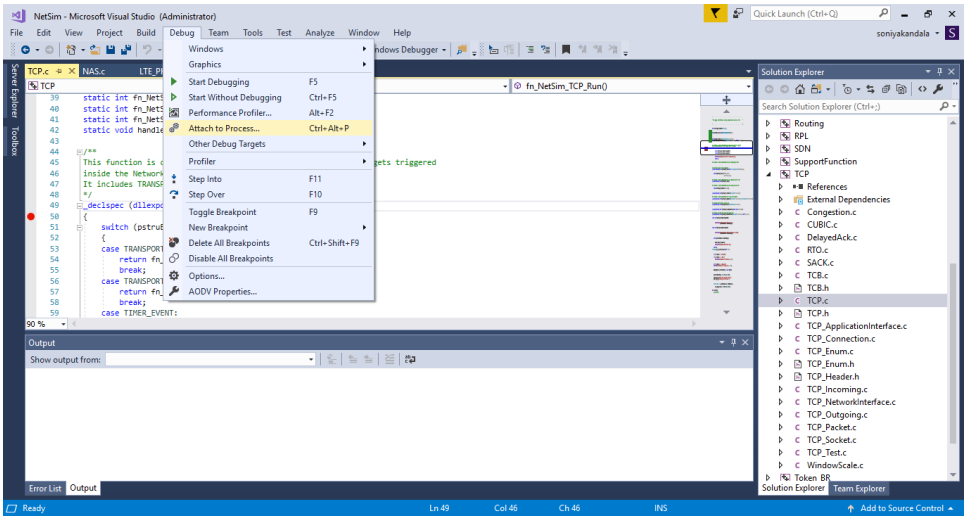
Here NetSim breakpoint has been triggered.

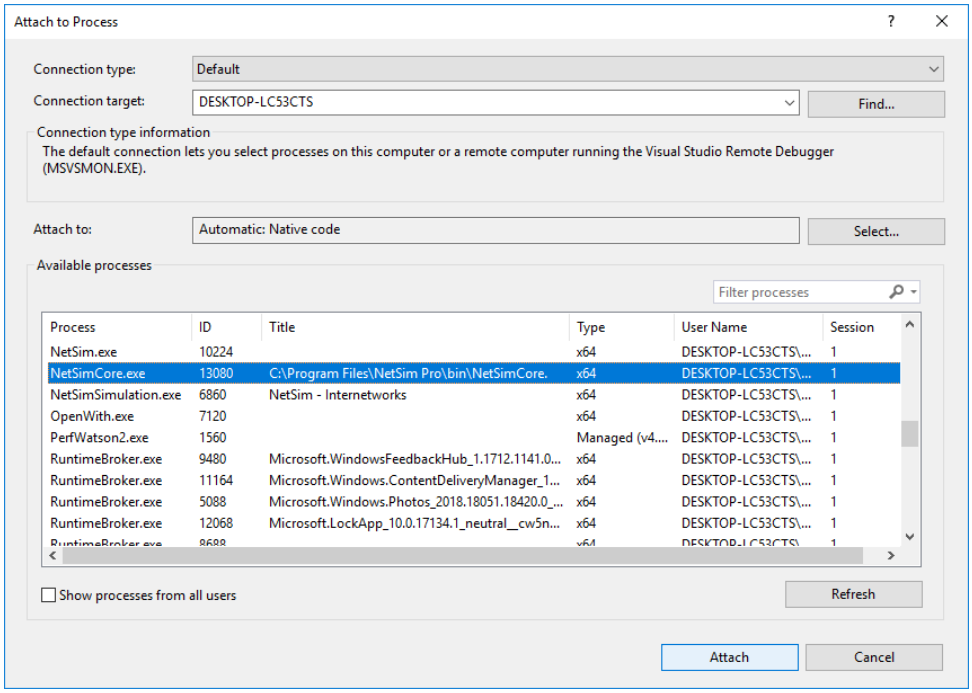
Step 10: Inside Solution Explorer pane in Visual Studio, double click on TCP project. Then open TCP.c file by double clicking on it. Using the drop down list of functions that are part of the current file, choose **fn_NetSim_TCP_Run()**

Step 11: 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



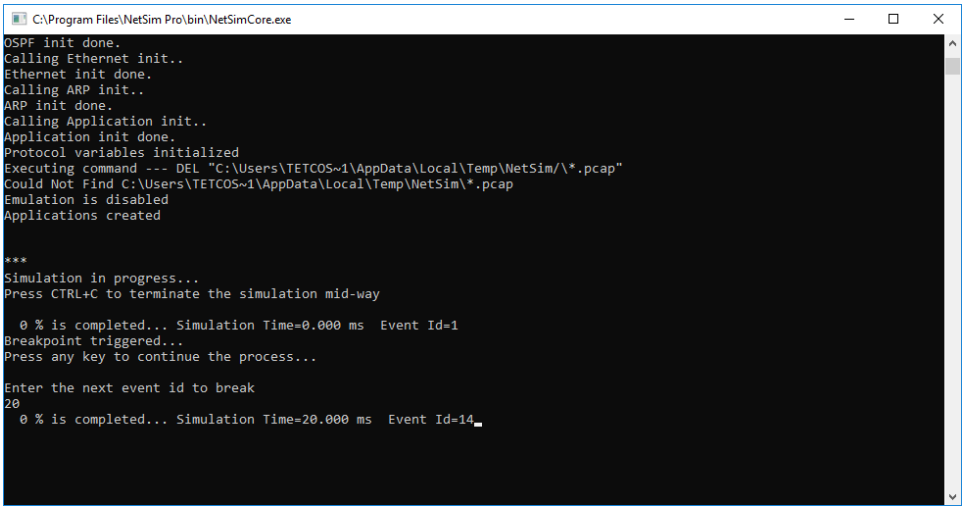
Step 12: Go to “Debug→Attach to Process” in Visual studio as shown and select NetSimCore.exe from the list of processes displayed.



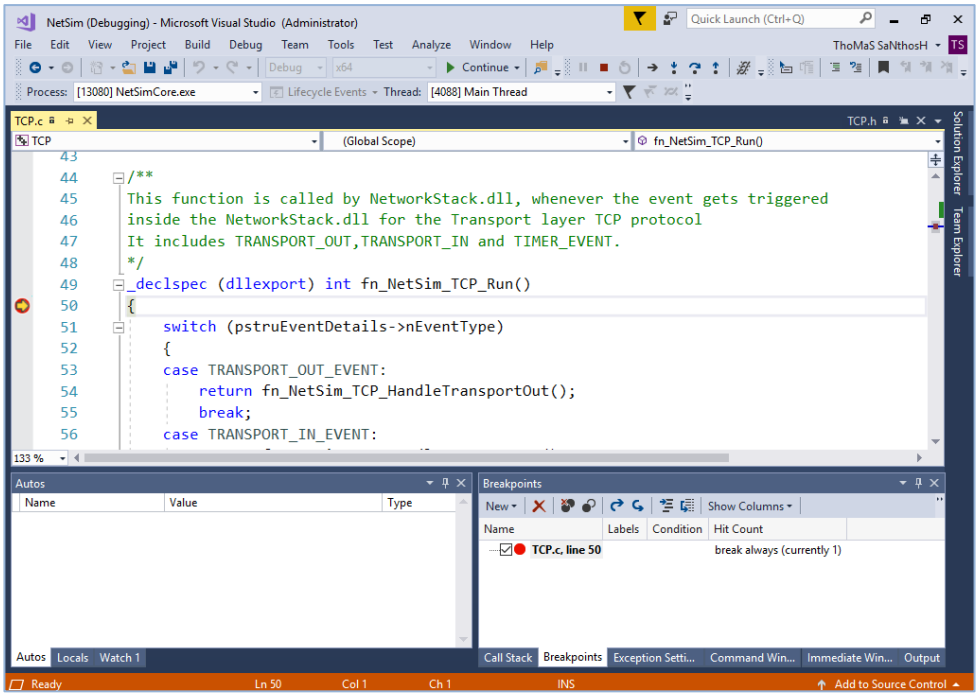


Click on Attach. Press any key in the command window to continue the process.

Step 13: Now we need to enter next event ID to break

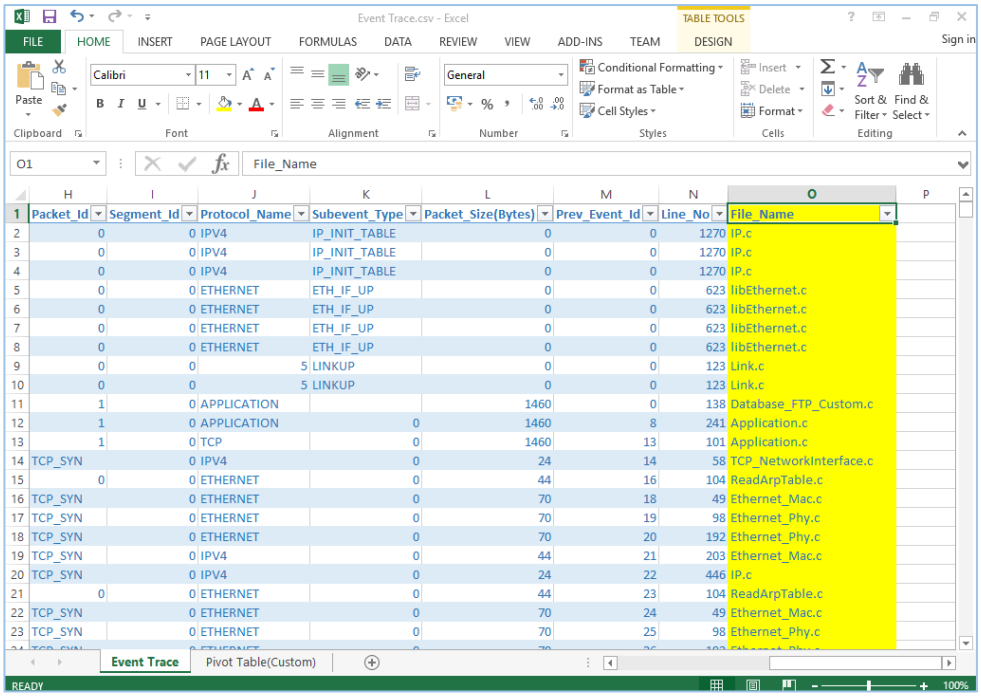


Then control goes to the project and stops at the break point in the source code (NetSim will break where ever user has set the breakpoint) as shown below. All debugging options like **step over (F10), step into (F11), step out (Shift + F11), continue (F5)** are available.



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 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
0	0	IPV4	IP_INIT_TABLE	0	0	1270	IP.c
0	0	IPV4	IP_INIT_TABLE	0	0	1270	IP.c
0	0	IPV4	IP_INIT_TABLE	0	0	1270	IP.c
0	0	ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
0	0	ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
0	0	ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
0	0	ETHERNET	ETH_IF_UP	0	0	623	libEthernet.c
0	0	5	LINKUP	0	0	123	Link.c
0	0	5	LINKUP	0	0	123	Link.c
1	0	APPLICATION		1460	0	138	Database_FTP_Custom.c
1	0	APPLICATION	0	1460	8	241	Application.c
1	0	TCP	0	1460	13	101	Application.c
TCP_SYN	0	IPV4	0	24	14	58	TCP_NetworkInterface.c
0	0	ETHERNET	0	44	16	104	ReadArpTable.c
TCP_SYN	0	ETHERNET	0	70	18	49	Ethernet_Mac.c
TCP_SYN	0	ETHERNET	0	70	19	98	Ethernet_Phy.c
TCP_SYN	0	ETHERNET	0	70	20	192	Ethernet_Phy.c
TCP_SYN	0	IPV4	0	44	21	203	Ethernet_Mac.c
TCP_SYN	0	IPV4	0	24	22	446	IP.c
0	0	ETHERNET	0	44	23	104	ReadArpTable.c
TCP_SYN	0	ETHERNET	0	70	24	49	Ethernet_Mac.c
TCP_SYN	0	ETHERNET	0	70	25	98	Ethernet_Phy.c

8.3.2 Via CLI

Modify the DSR protocol and build the code. Create a scenario on MANET then follow the below steps.

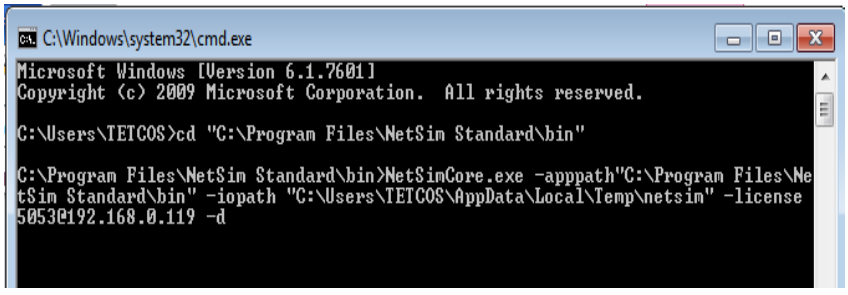
Step 1: Open the Command prompt.Press “windows+R” and type “cmd”.

Step 2: To run the NetSim via CLI copy the path where “NetSimCore.exe” is present.

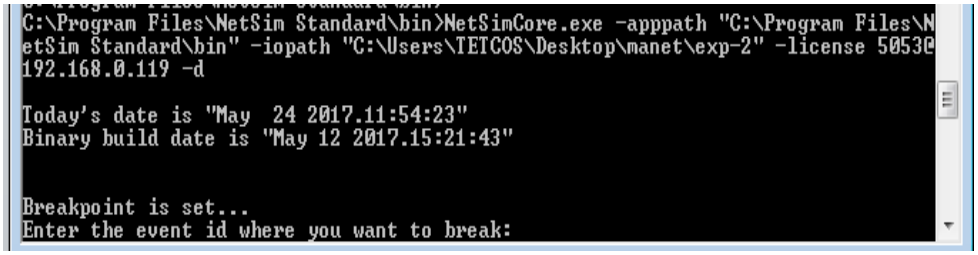
>cd <apppath>

>NetSimCore.exe<space>-apppath<space><apppath><space>-iopath<space><iopath><space>-license<space>5053@<ServerIP Address><space> -d

Step 3: Type the following command.

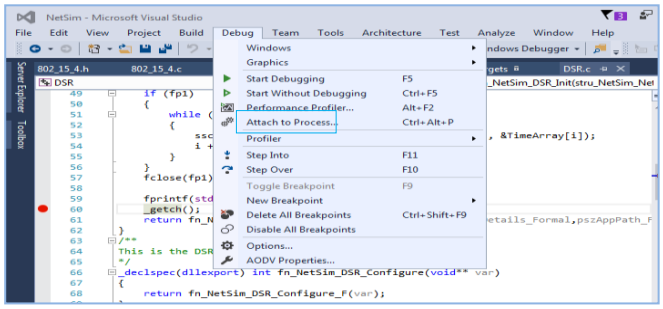


Press enter, now you can see the following screen.

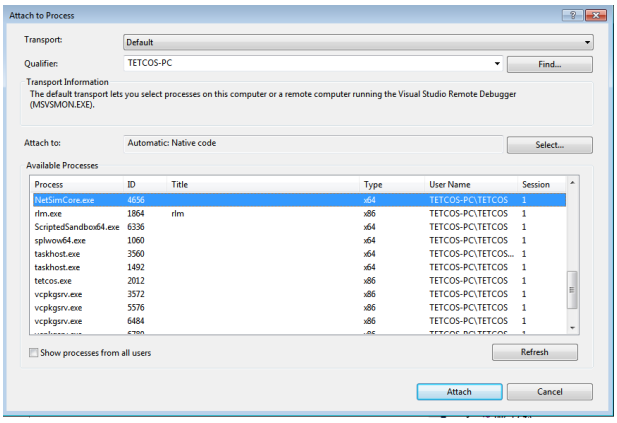


Step 4: Open the Project in Visual Studio and put break point inside the source code.

Step 5: Go to "Debug→Attach to Process"



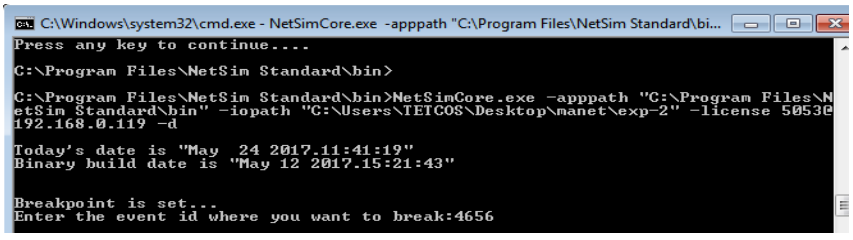
Attach to NetSimCore.exe.



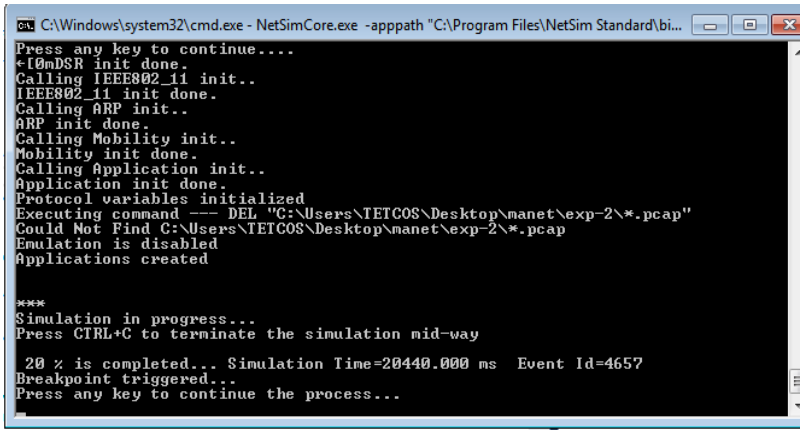
Click on Attach.

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

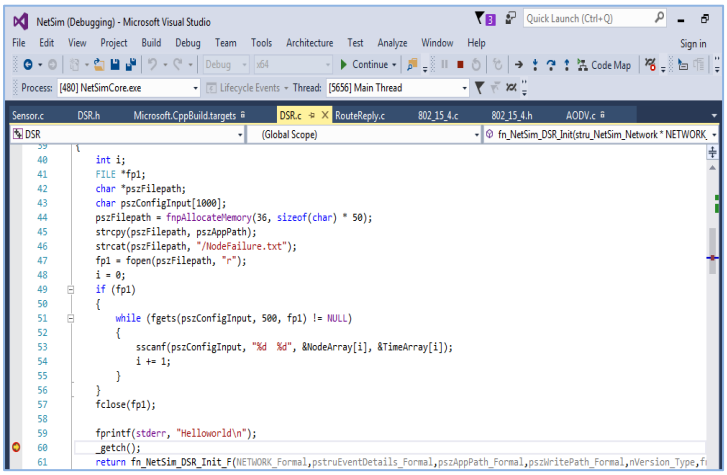
Note: If you don't want to stop at any event you can specify 0 as event id.



Execution will stop 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.



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.

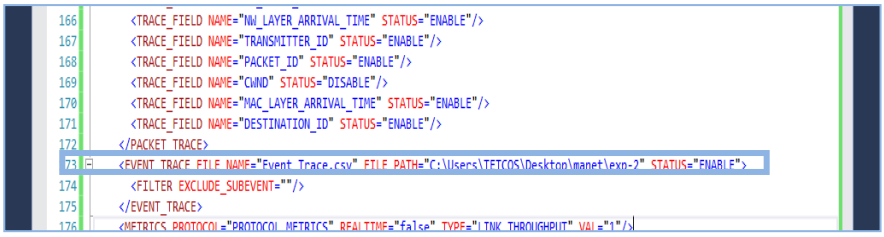
To stop debugging press Shift+F5. This thengives the control back to NetSim, for normal execution to continue.

8.3.3 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.

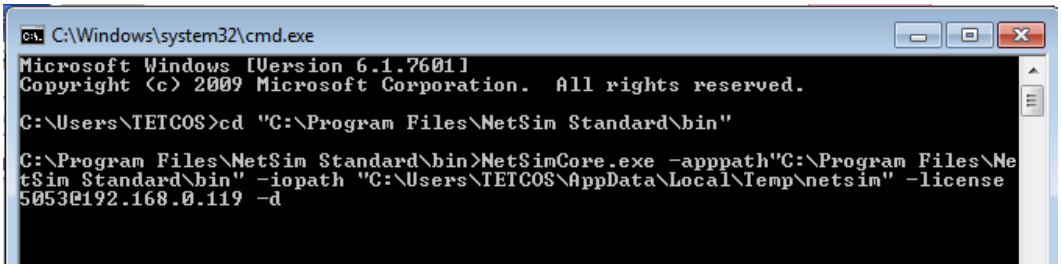
Procedure 1

Step 1: Open configuration.netsim file and provide the file name, path and set status as Enable.

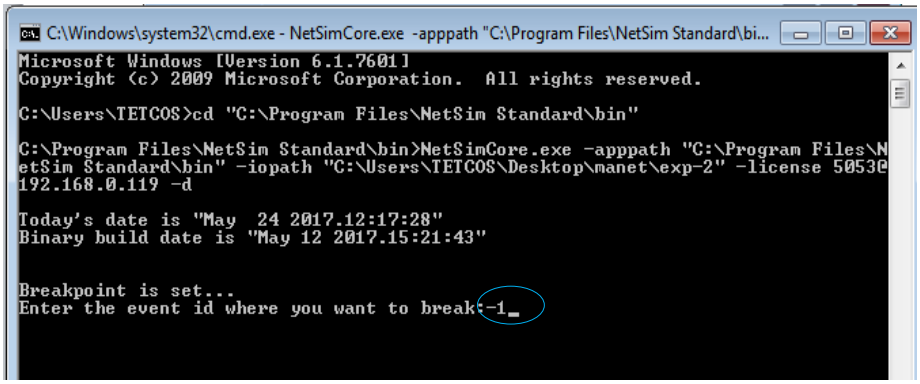


Step 2: Run the NetSim via CLI in debug mode (Refer NetSim Help in chapter 6→Running Simulation via CLI) with –d as the fourth parameters

Press enter



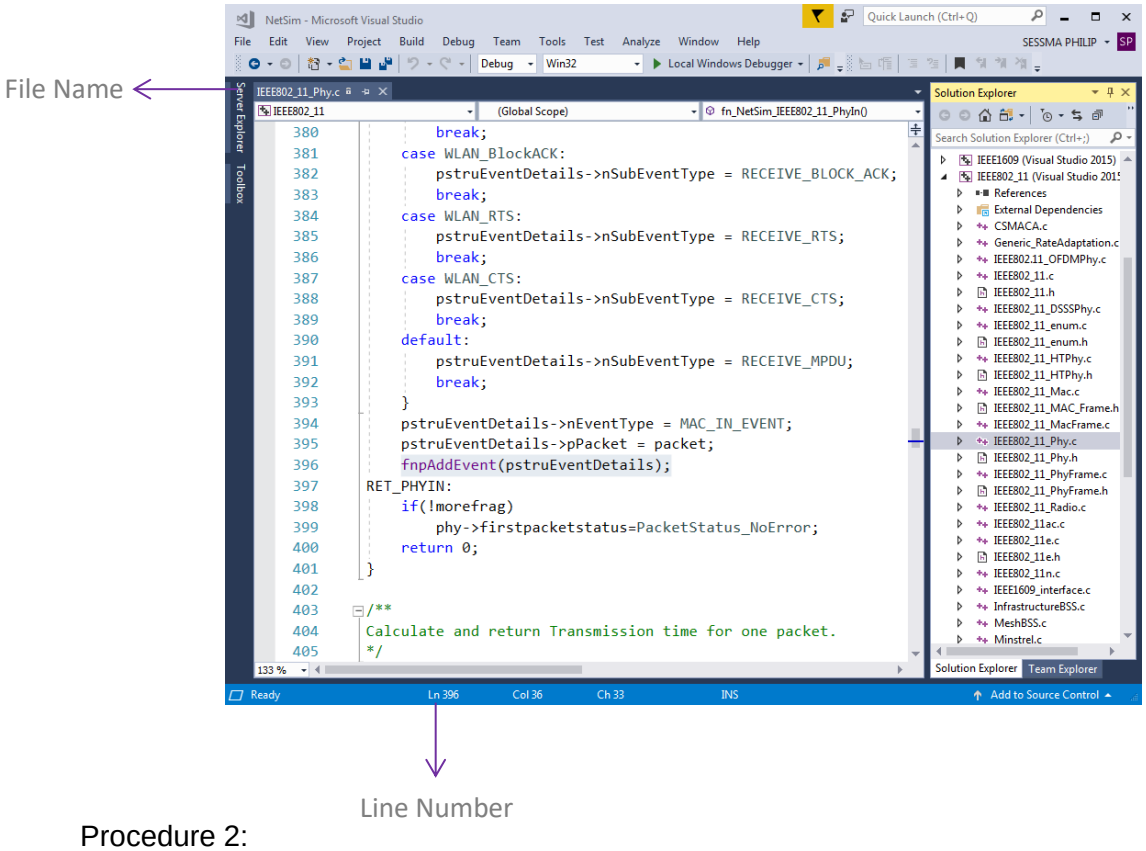
Step 3: Enter -1 as the event ID



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

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.

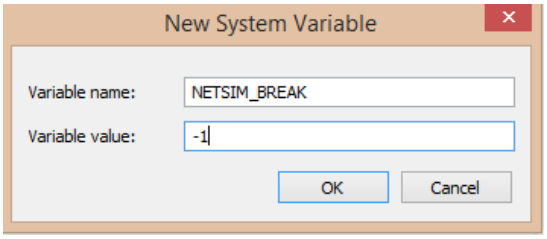


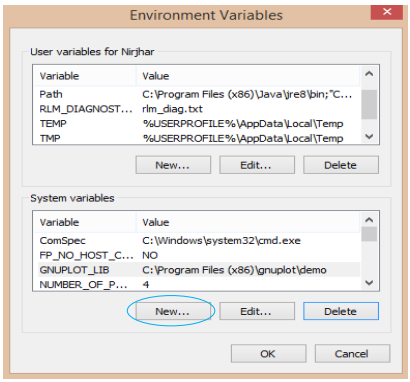
Procedure 2:

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

Step 2: Go to Advanced System setting → Advanced Tab → Environment Variables

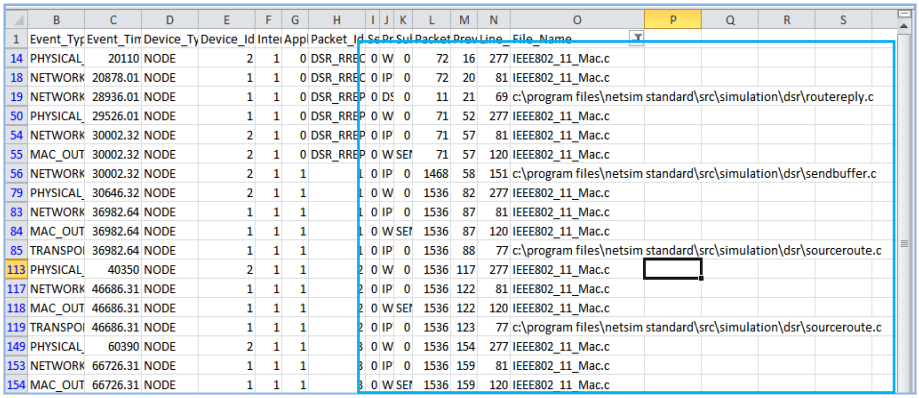
Step 3: Click New. Type “NETSIM_BREAK” as Variable name and any negative integer as Variable value. Click OK.





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.

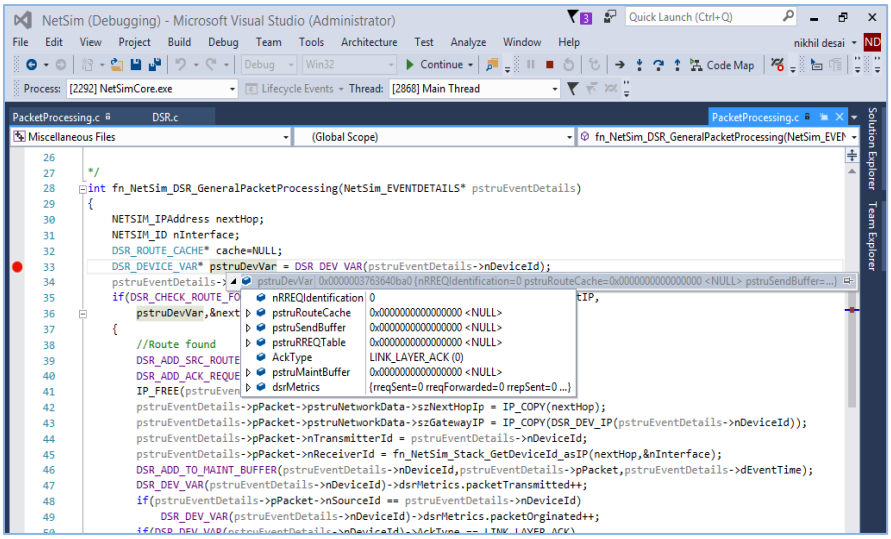


	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Event_Typ	Event_Tim	Device_Ty	Device_Id	Inter	Appl	Packet_Id	Se	Pr	Sul	Packet	Prev	Line_	File_Name				
14	PHYSICAL	20110	NODE	2	1	0	DSR_RREC	0	W		0	72	16	277	IEEE802_11_Mac.c			
18	NETWORK	20878.01	NODE	1	1	0	DSR_RREC	0	IP		0	72	20	81	IEEE802_11_Mac.c			
19	NETWORK	28936.01	NODE	1	1	0	DSR_RREP	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_RREP	0	W		0	71	52	277	IEEE802_11_Mac.c			
54	NETWORK	30002.32	NODE	2	1	0	DSR_RREP	0	IP		0	71	57	81	IEEE802_11_Mac.c			
55	MAC_OUT	30002.32	NODE	2	1	0	DSR_RREP	0	W	SEI		71	57	120	IEEE802_11_Mac.c			
56	NETWORK	30002.32	NODE	2	1	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		1	0	W	0	1536	82	277	IEEE802_11_Mac.c			
83	NETWORK	36982.64	NODE	1	1	1		1	0	IP	0	1536	87	81	IEEE802_11_Mac.c			
84	MAC_OUT	36982.64	NODE	1	1	1		1	0	W	SEI	1536	87	120	IEEE802_11_Mac.c			
85	TRANSPOI	36982.64	NODE	1	1	1		1	0	IP	0	1536	88	77	c:\program files\netsim standard\src\simulation\dsr\sourceroute.c			
113	PHYSICAL	40350	NODE	2	1	1		2	0	W	0	1536	117	277	IEEE802_11_Mac.c			
117	NETWORK	46686.31	NODE	1	1	1		2	0	IP	0	1536	122	81	IEEE802_11_Mac.c			
118	MAC_OUT	46686.31	NODE	1	1	1		2	0	W	SEI	1536	122	120	IEEE802_11_Mac.c			
119	TRANSPOI	46686.31	NODE	1	1	1		2	0	IP	0	1536	123	77	c:\program files\netsim standard\src\simulation\dsr\sourceroute.c			
149	PHYSICAL	60390	NODE	2	1	1		3	0	W	0	1536	154	277	IEEE802_11_Mac.c			
153	NETWORK	66726.31	NODE	1	1	1		3	0	IP	0	1536	159	81	IEEE802_11_Mac.c			
154	MAC_OUT	66726.31	NODE	1	1	1		3	0	W	SEI	1536	159	120	IEEE802_11_Mac.c			

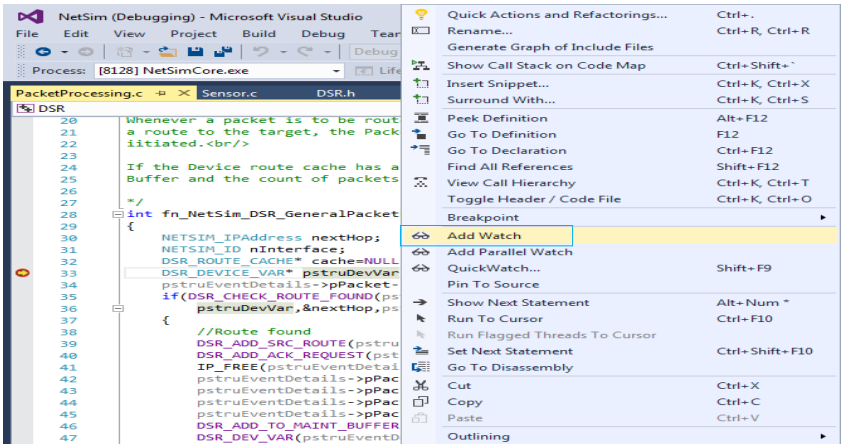
8.3.4 Viewing & Accessing variables

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.



Adding the variable to watch –

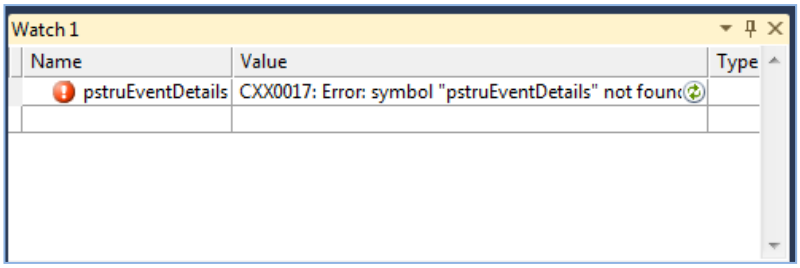


Watch the change in the variable as the code progress by right clicking on the variable & clicking on "add watch" tab. This is useful if to continuously monitor the change in the variable as the code progresses.

Viewing external variables

During the process of debug users would come across variables that are defined outside the source file being built as a .dll. Such variables cannot be viewed directly when added in the watch tab, as this would throw the error

CX0017: Error:symbol “Variable_Name”not found.

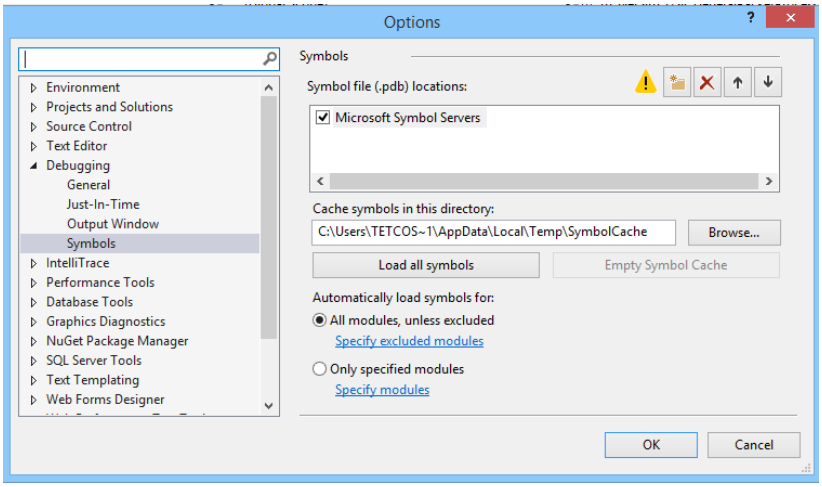


In the call stack window one can find the file in which that variable is situated. Right click on the dll file name in the call stack window, in this case NetworkStack.dll. Then in the pull down menu which appears, select "load symbols from" and give the path of the pdb(program database) file.

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.

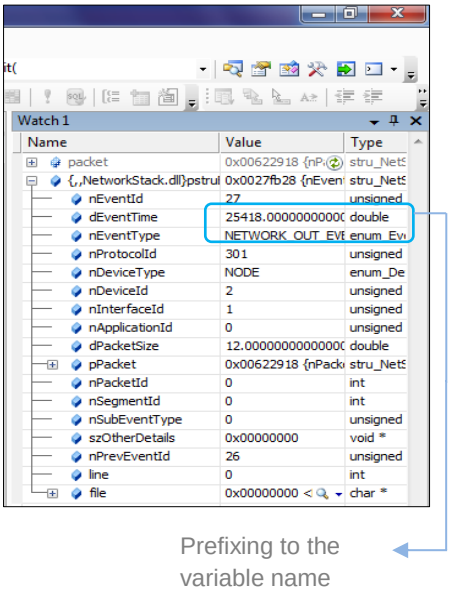
When a user debugs a project in the Visual Studio IDE, the debugger knows exactly where to find the .pdb and source files for the code. If the user wants to debug code outside their project source code, such as the Windows or third-party code the project calls, the user has to specify the location of 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 for the latest copy of these debug files. Go to Tools > options>debugging>load all symbols.



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).



Accessing External Variables:

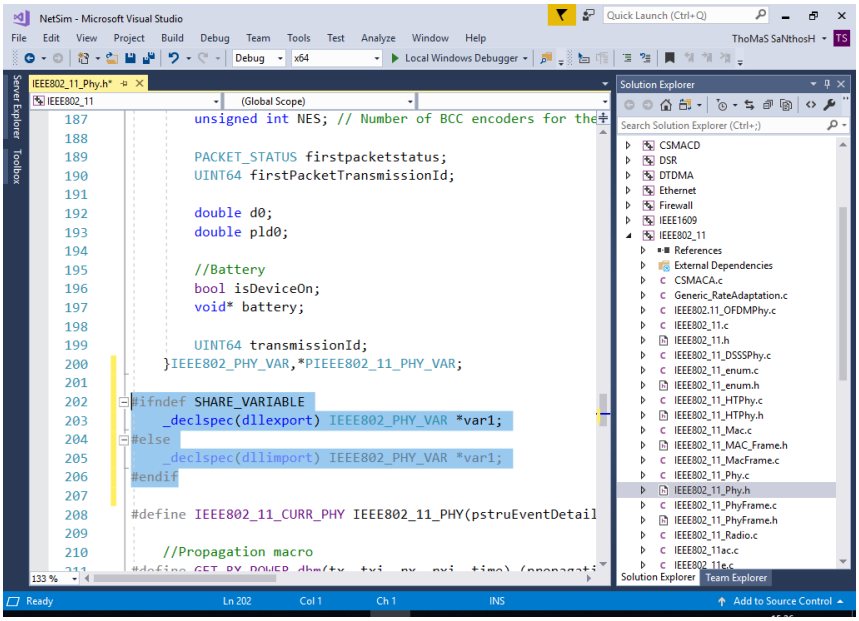
Each protocol in NetSim has a separate Dll file which contains the variables and functions which can be shared. In case of cross layer protocol implementations variables of one protocol may have to be accessed from another Dll.

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
```

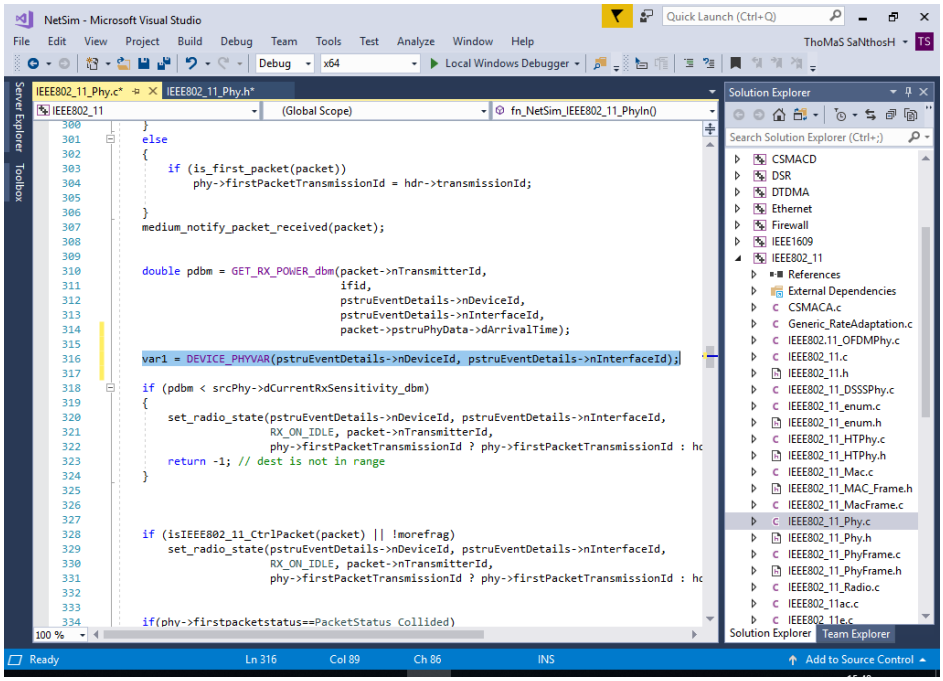
In the example, the code line must be written in IEEE802_11_Phy.h file present inside IEEE802_11 folder.



In the main function where a user wishes to find the dReceivedPower_mw, the variable must be assigned the respective value. In the above case, the following line of code must be written inside fn_NetSim_IEEE802_11_PhyIn() function in IEEE802_11_Phy.c file present inside IEEE802_11 folder.

var1 = DEVICE_PHYVAR(pstruEventDetails->nDeviceId,pstruEventDetails->nInterfaceId);

Note that the parameters given in the macro or any function which assigns a value to the variable must be defined beforehand in the code. Here nDeviceId and nInterfaceId are defined beforehand.



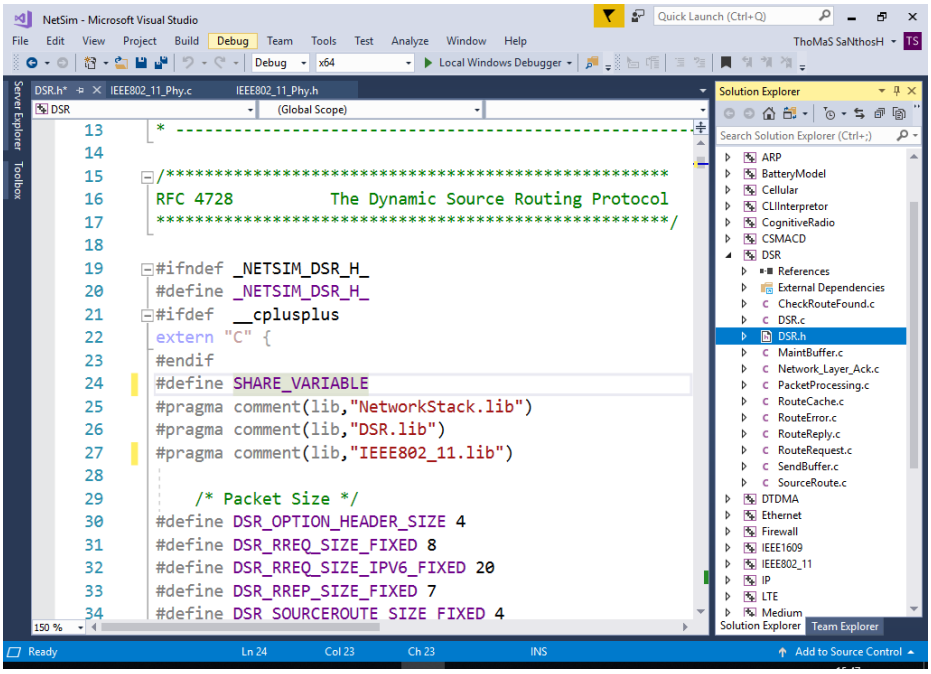
The IEEE802_11 project must be built and the resulting **libIEEE802.11.dll** file which gets created in the bin folder of NetSim's current workspace <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x86> for 32-bit and <C:\Users\PC\NetSim_11.1.7_32_std_default\bin\bin_x64> for 64-bit NetSim.

The Object file **IEEE802_11.lib** which also got created with all file must be copied and pasted in the lib folder located in the current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\src\Simulation\lib_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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

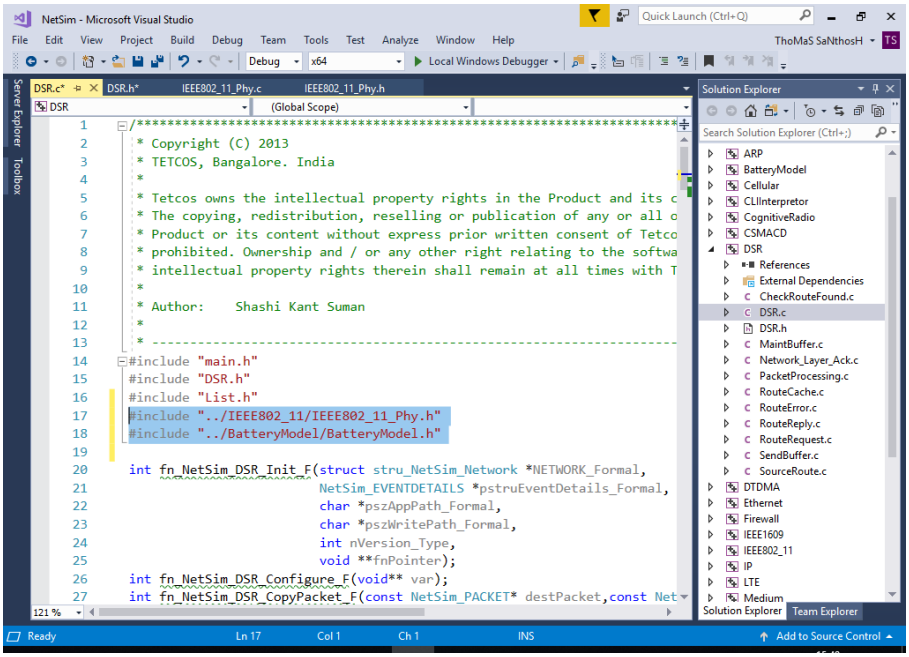
#define SHARE_VARIABLE

#pragma comment(lib,"IEEE802_11.lib")



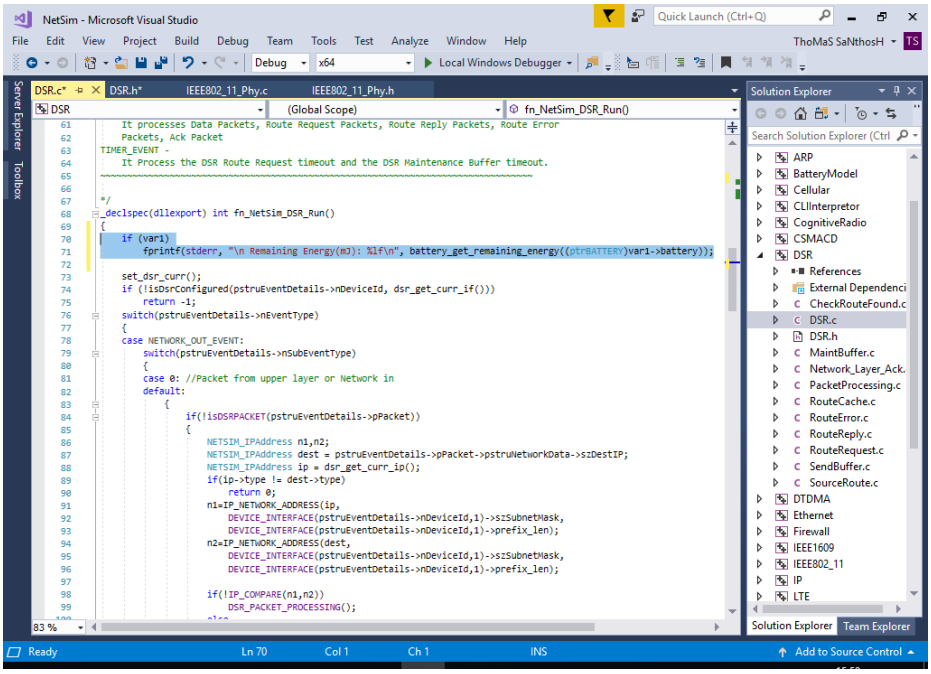
Add the following lines of code to the DSR.c file as shown below:

#include "../IEEE802_11/IEEE802_11_Phy.h"
#include "../BatteryModel/BatteryModel.h"

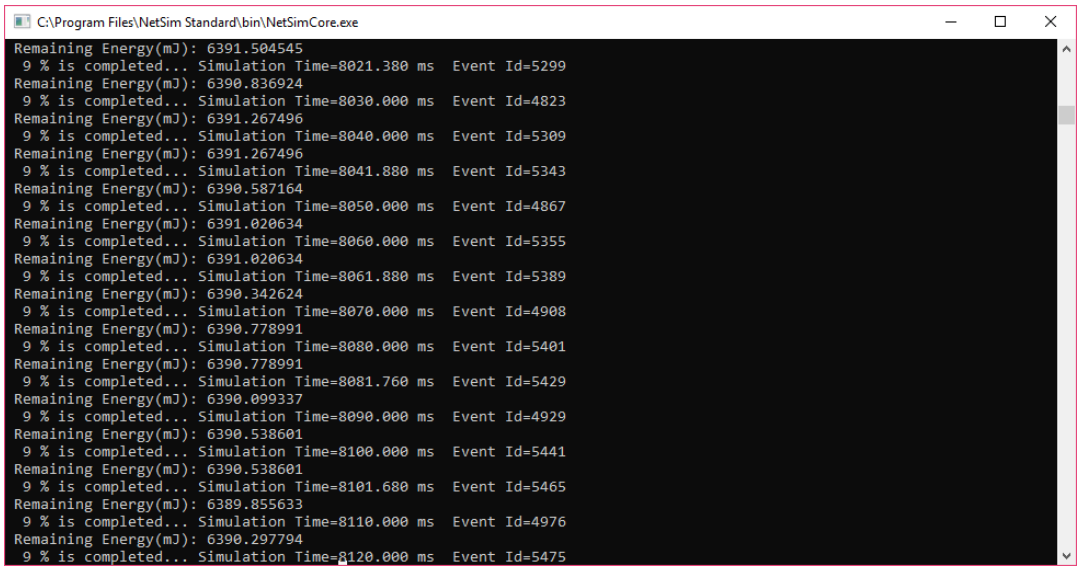


In the fn_NetSim_DSR_Run() function add the following lines of code to print the value of dReceivedPower_mw variable from DSR project.

```
if (var1)
                fprintf(stderr, "\n Remaining Energy(mJ): %lf\n"
                ,battery_get_remaining_energy((ptrBATTERY)var1->battery));
```



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\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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:



8.3.5 Print to console window in NetSim

Users can try printing the Device ID, Application ID, Duplicate Ack Count etc.

To print to console: Print node positions in MANET

Open Mobility Project, and in Mobility.c file go to fn_NetSim_Mobility_Run() function. Inside the default case add following codes

fprintf(stderr,"\n The position of %s at time %.2lfms is X=%.2lf and Y = %.2lf \n",DEVICE_NAME(pstruEventDetails->nDeviceId),

pstruEventDetails->dEventTime,

DEVICE_POSITION(pstruEventDetails->nDeviceId)->X,

DEVICE_POSITION(pstruEventDetails->nDeviceId)->Y);

_getch();

```
419                 pstruEventDetails->dEventTime-=pstruMobilityVar->dCalculationInterval;
420             }
421
422             //call all the callback function
423             for(nLoop=0;nLoop<nCallBackCount;nLoop++)
424             {
425                 fnMobilityCallBack[nLoop](pstruEventDetails->nDeviceId);
426             }
427         }
428         fprintf(stderr, "\n The position of %s at time %.2lfms is X=%.2lf and Y = %.2lf \n", DEVICE_NAME(pst
429             pstruEventDetails->dEventTime,
430             DEVICE_POSITION(pstruEventDetails->nDeviceId)->X,
431             DEVICE_POSITION(pstruEventDetails->nDeviceId)->Y);
432         _getch();
433
434         break;
435     }
436     return 1;
437 };
```

Building Mobility project creates libMobility.dll inside the binary folder of NetSim's current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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.

8.4 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:

```
/** 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,
};
```

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:

```
enum enum_IEEE_802_15_4_ControlPacket_Type
{
    BEACON_FRAME = MAC_PROTOCOL_IEEE802_15_4*100+1,
    SLEEP_FRAME,
    ACK_FRAME,
    MY_PACKET,
};
```

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.

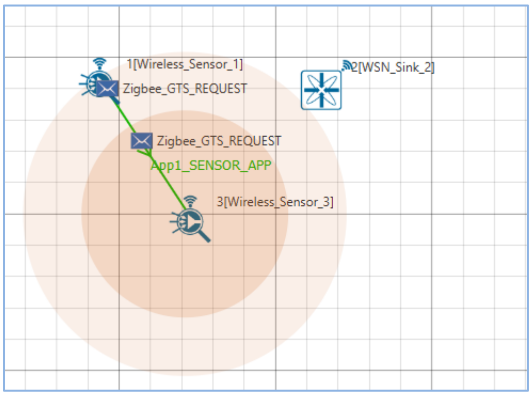
9. fn_NetSim_Zigbee_Trace() is an API to print the trace details to the event trace. So inside 802_15_4.c file add the following lines of code in red color inside fn_NetSim_Zigbee_Trace(int nSubEvent) as shown below:-

```
_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\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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

    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.



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.

PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3
0	N/A	Control_Packet	MY_PACKET	SINKNODE-3	Broadcast-0	SINKNODE-3

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			Protocol_Name	Subevent_Type
4	TIMER_EVENT	0	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
9	TIMER_EVENT	1000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
4280	TIMER_EVENT	2000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
8519	TIMER_EVENT	3000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
12820	TIMER_EVENT	4000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
17025	TIMER_EVENT	5000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
21223	TIMER_EVENT	6000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
25452	TIMER_EVENT	7000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
29735	TIMER_EVENT	8000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT
33993	TIMER_EVENT	9000000	SINKNODE	3	1	0	0	0	IEEE802.15.4	MY_EVENT

8.5 NetSim API's

NetSim provides a wide variety of APIs for protocol developers. These are available in

1 **packet.h** – Packet related APIs

- Create a new packet
    - fn_NetSim_Packet_CreatePacket_dbg(int nLayer,int line,const char* file);
- Copy a packet into a new packet
    - fn_NetSim_Packet_CopyPacket_dbg(const NetSim_PACKET* pstruPacket,int line,const char* file);
- Create error in packet
    - fn_NetSim_Packet_DecideError(double dBER, long double dPacketSize);
- Free a packet
    - fn_NetSim_Packet_FreePacket_dbg(NetSim_PACKET** pstruPacket,int line,char* file);

2 **stack.h** – Network / device / link and event related APIs

- Calculate distance between nodes.
    - fn_NetSim_Utilities_CalculateDistance(NetSim_COORDINATES* coordinate1,NetSim_COORDINATES* coordinates2);
- Stores the event details. Only one time memory is allocated. Most used variable
    - struct stru_NetSim_EventDetails* pstruEventDetails;
- Retrieve values from xml file
    - GetXmlVal(void* var,char* name,void* xmlNode,int flag, XMLDATATYPE type);

3 **list.h** -- Optimized list operation calls since NetSim uses lists extensively

- Add elments in list
    - list_add(void** list,void* mem,size_t offset,int (*check)(void* current,void* mem));

- Sorting the list
    - list_sort(void** list,size_t offset,int (*check)(void* current, void* mem));

4 **IP_Addressing.h** – For setting & getting IP address per the appropriate format

- Set Ip address of any node
    - NETSIM_IPAddress
- Checking ip address is broadcast or multicast
    - isBroadcastIP(NETSIM_IPAddress ip);
    - isMulticastIP(NETSIM_IPAddress ip);

For detailed help please refer the appropriateheader (.h) files inside:/NetSim_Standard/src/simulation/include or read through the doxygen source code documentation available inside 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)(**const**NetSim_PACKET* pstruDestPacket,**const** NetSim_PACKET* pstruSrcPacket):
- This is for copying protocol specific parameters / data into the packed
- **int** (*fn_NetSim_protocol_FreePacket)(**const** NetSim_PACKET* pstruPacket): The this to free the protocol specific parameters / data in the packet
- (*fn_NetSim_protocol_Metrics)(**const** FILE* fpMetrics): This is to write the metrics file upon completion of the simulation
- **int** (*fn_NetSim_protocol_Finish)(): To release all memory after completion
- char* (*fn_NetSim_protocol_ConfigPacketTrace)(**constvoid*** xmlNetSimNode); To configure the packet packet trace in terms of the parameters to be logged

  char* (*fn_NetSim_protocol_WritePacketTrace)(**const** NetSim_PACKET*); To configure the event trace in terms of the parameters to be logged.

9 Advanced Features

9.1 Random number Generator and Seed Values

All network simulations involve an element of randomness. Some examples are -

It is possible to configure the traffic sources in the simulation to generate traffic in a perfectly regular pattern. However, this is typically not the case in the real world. For example, Node back-off's after collisions are random to resolve contention issues. The exact bit which is errored, based on Bit error probability of a wireless channel, is decided randomly

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 PC or the 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. When simulations are run for a network configuration with different seed values, the results will likely be slightly different.

More advanced users get "Confidence" by analyzing a set of results with different seed values for the same network scenario.

9.2 Interfacing MATLAB with NetSim (Std/Pro versions)

9.2.1 Implement Rician Distribution of MATLAB in NetSim without using .m file

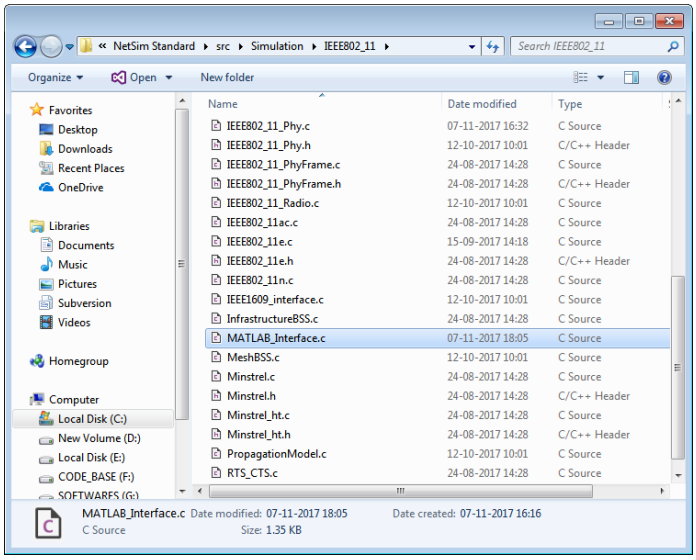
In this example we will replace the default Rayleigh Fading (part of the path loss calculation) used in NetSim, with a Fading Power calculated using the Rician Distribution from MATLAB

Procedure:

i. Create a MATLAB_Interface.c file inside the IEEE802_11 folder which can be found in the current workspace location of the NetSim that you are running and it would be something like “C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64” for 64-bit and “C:\Users\PC\NetSim_11.1.7_32_std_default\bin\bin_x86” for 32-bit. For more information on NetSim workspace refer Section 3 “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[100];
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 rician_noncentrality = 1, rician_scale = 2;

    engPutVariable(ep, "h", h);
    //use ProbDistUnivParam() function for matlab 2016
    sprintf(buf, "h=ProbDistUnivParam('rician',[%d %d])", rician_noncentrality,
    rician_scale);
    //use makedist() function for matlab 2017
    //sprintf(buf, "h=makedist('rician',%d %d)", rician_noncentrality, rician_scale);

    status = engEvalString(ep, buf);

    engPutVariable(ep, "i", i);
    sprintf(buf, "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;
}
```

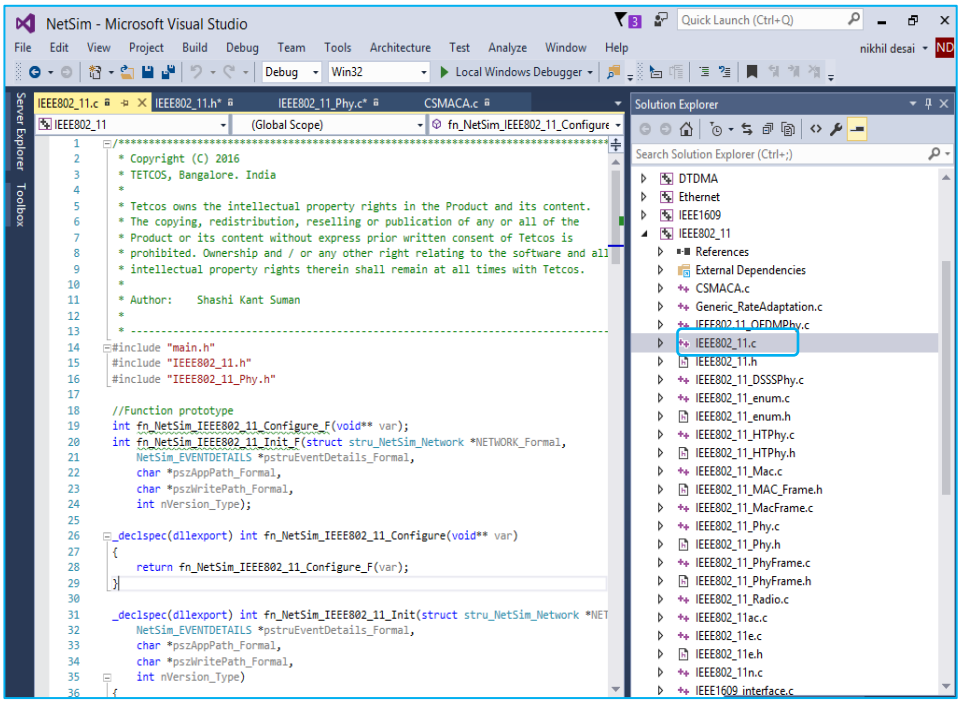


4. Now open the code
5. Right click on “IEEE802_11 Project” present in “Solution Explorer” window and select Add → Existing Item and select the MATLAB_Interface.c file.
6. MATLAB_Interface.c file contains the following functions

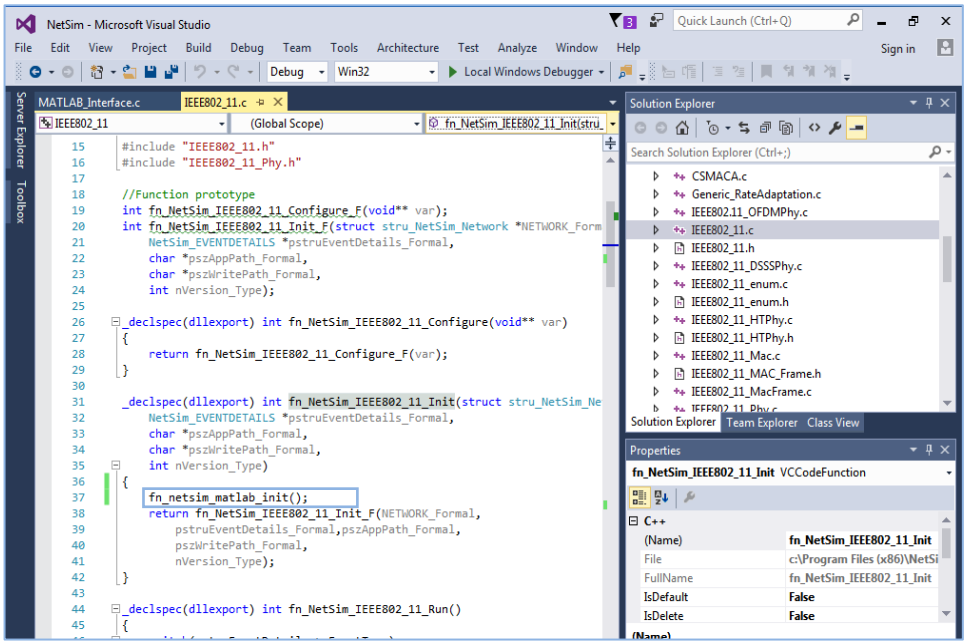
   a) fn_netsim_matlab_init() - Opens the MATLAB Engine

   b) fn_netsim_matlab_run() - Communicates with MATLAB Command Window

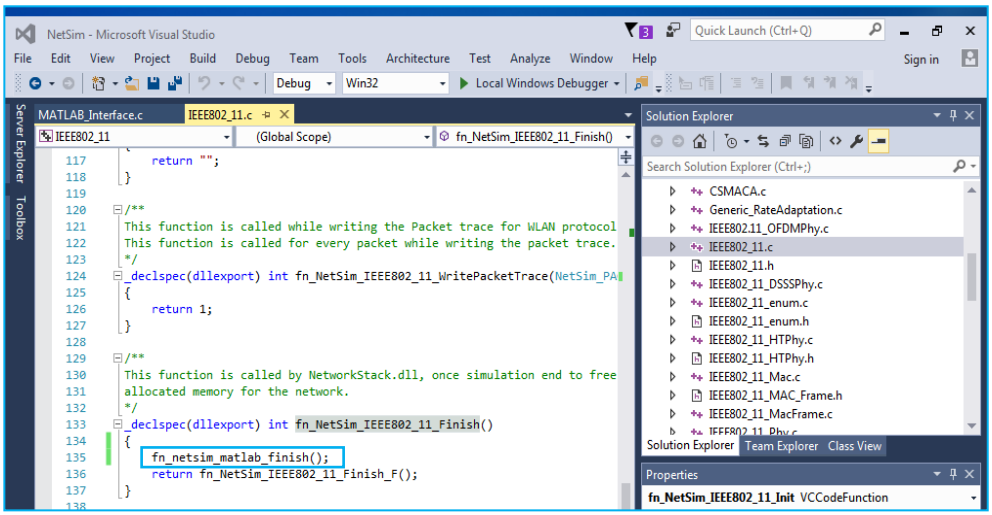
   c) fn_netsim_matlab_finish() - Closes the MATLAB Engine
7. In the Solution Explorer double click on the IEEE802_11.c file.



8. Add a call to fn_netsim_matlab_init(); inside the fn_NetSim_IEEE802_11_Init() function.



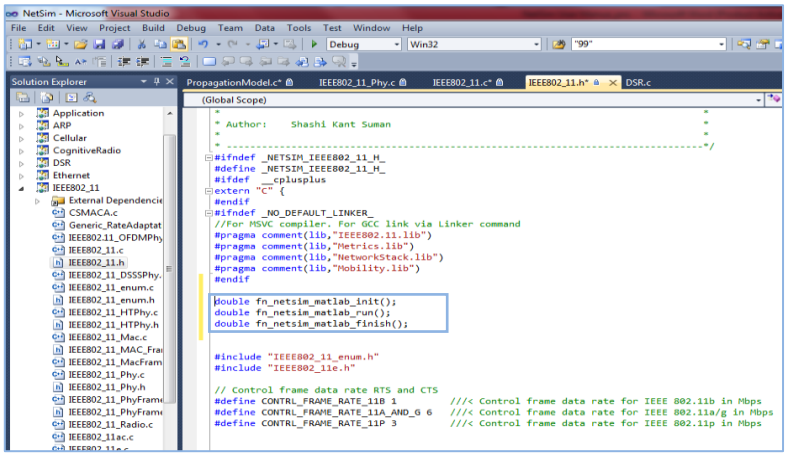
9. Similarly add a call to fn_netsim_matlab_finish(); inside the fn_NetSim_IEEE802_11_Finish() function.



10. 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();



11. In the Solution Explorer double click on the IEE802_11_PHY.c file.
12. Inside fn_Netsim_IEEE802_11_PHYIn() function comment the lines,

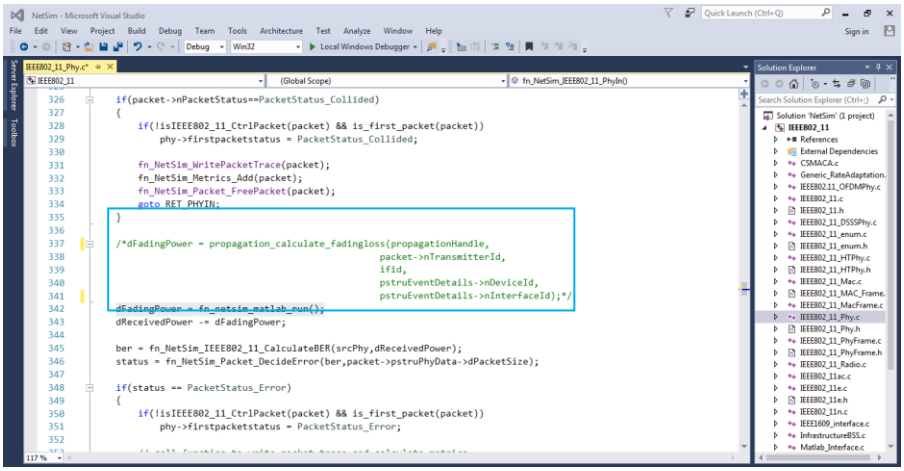
dFadingPower = propagation_calculate_fadingloss(propagationHandle,

packet->nTransmitterId,ifid,pstruEventDetails->nDeviceId,

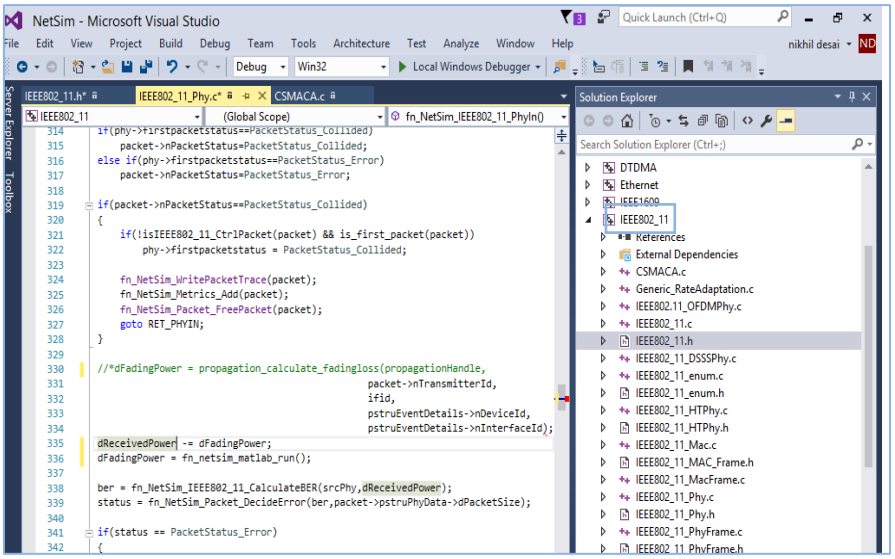
pstruEventDetails->nInterfaceId);

Make a call to the fn_netsim_matlab_run() function by adding the following line,

dFadingPower = fn_netsim_matlab_run();



13. 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:

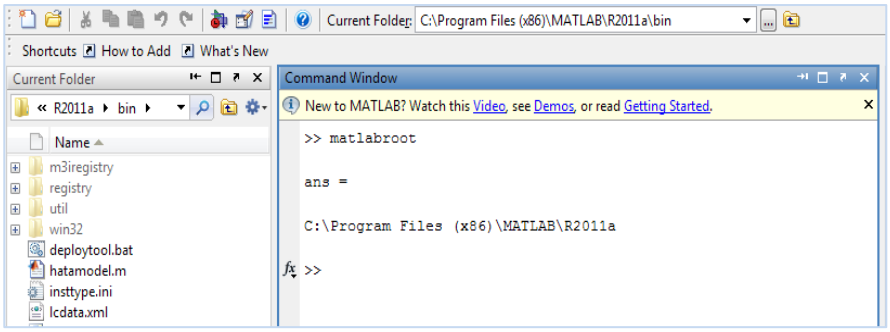


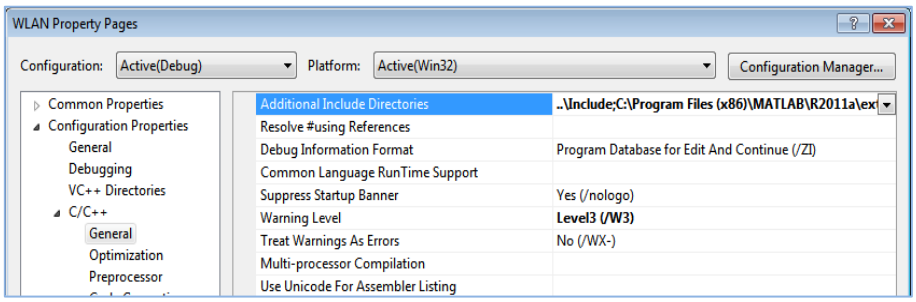
- Under C/C++ → General, add the following directory to the field ADDITIONAL INCLUDE DIRECTORIES:

<Path where MATLAB is installed>\extern\include

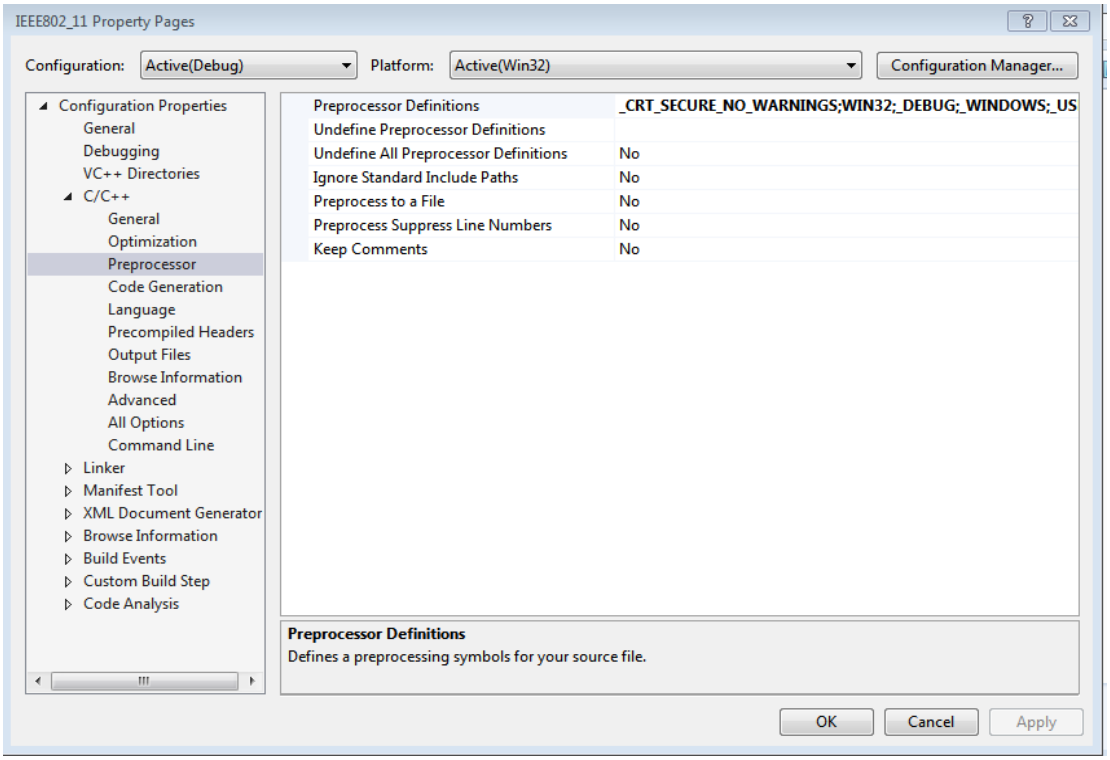
NOTE: To determine path where MATLAB is installed, entering the following command in the MATLAB command prompt:

matlabroot

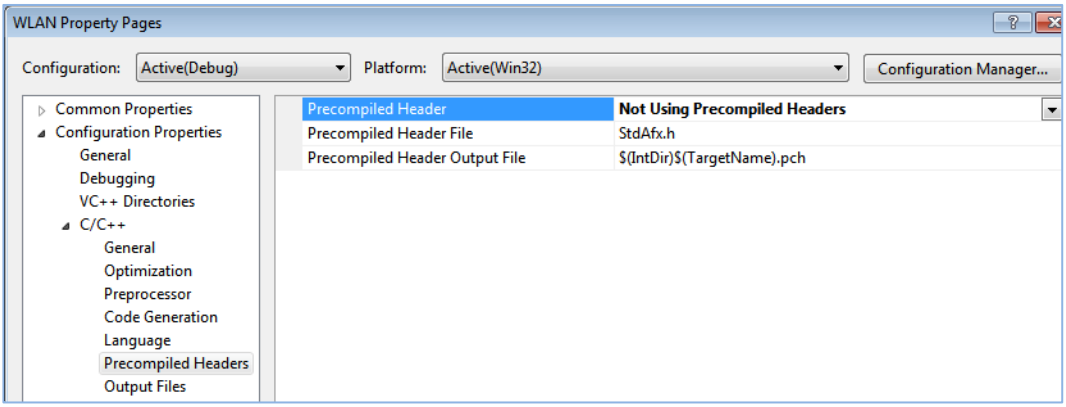




Go to the Preprosessor-->In Preprocessor Defination add _CRT_SECURE_NO_WARNINGS; to the beginning of the existing entries as shown below and click on Apply and then click on OK.

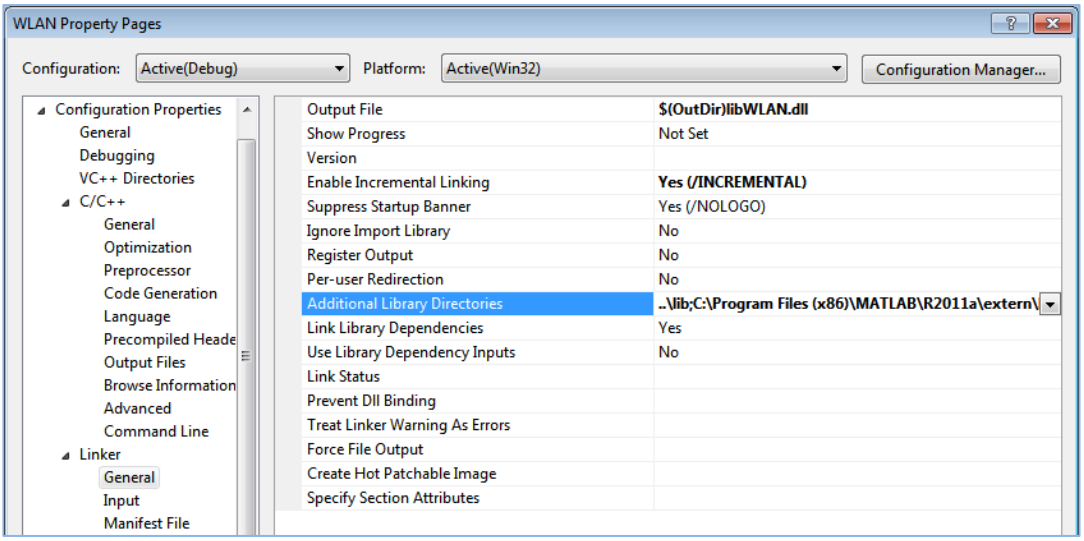


Under C/C++ → Precompiled Headers, set PRECOMPILED HEADERS as "Not Using Precompiled Headers".

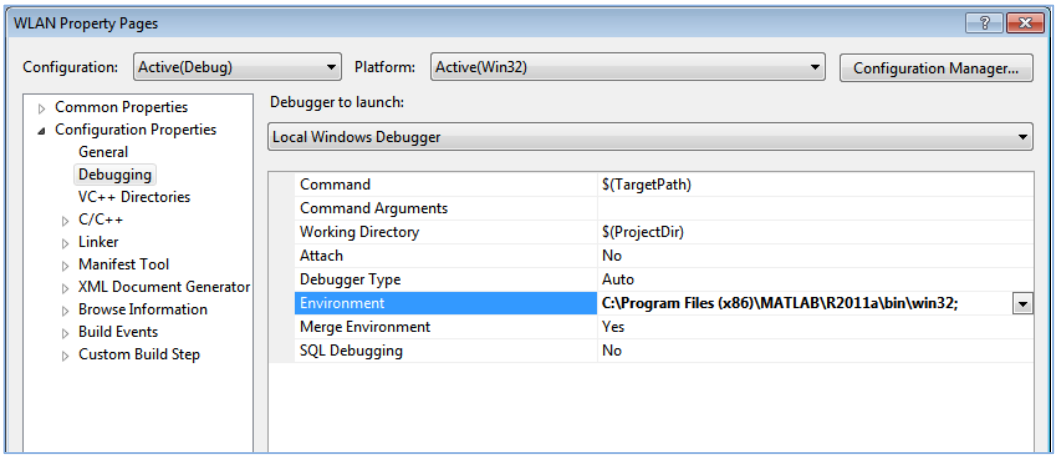


- Under Linker → General, add the directory to the field ADDITIONAL LIBRARY DIRECTORIES:

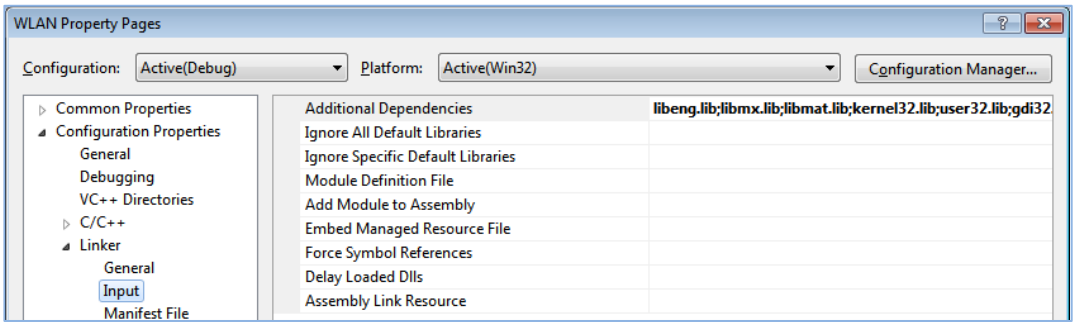
  <Path where MATLAB is installed>\extern\lib\win32\microsoft



- Under Configuration Properties →Debugging, Add the following Target path in the ENVIRONMENT: <Path where MATLAB is installed>\bin\win32



- Under Linker → Input, add the following names to the field marked ADDITIONAL DEPENDENCIES: libeng.lib;libmx.lib;libmat.lib;



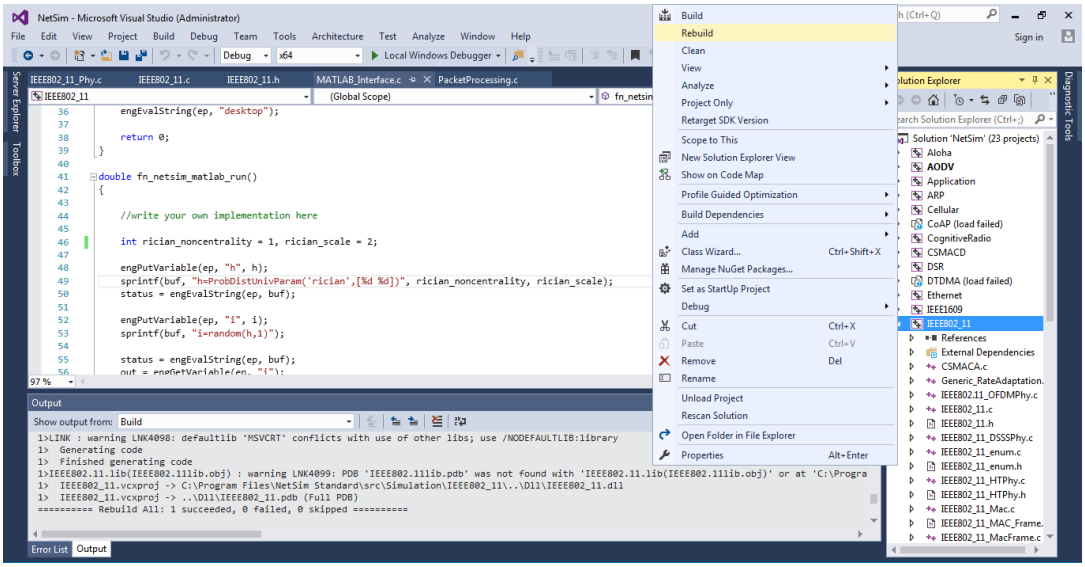
- Make sure that the following directory is in the environment variable PATH:

  <Path where MATLAB is installed>\bin\win32

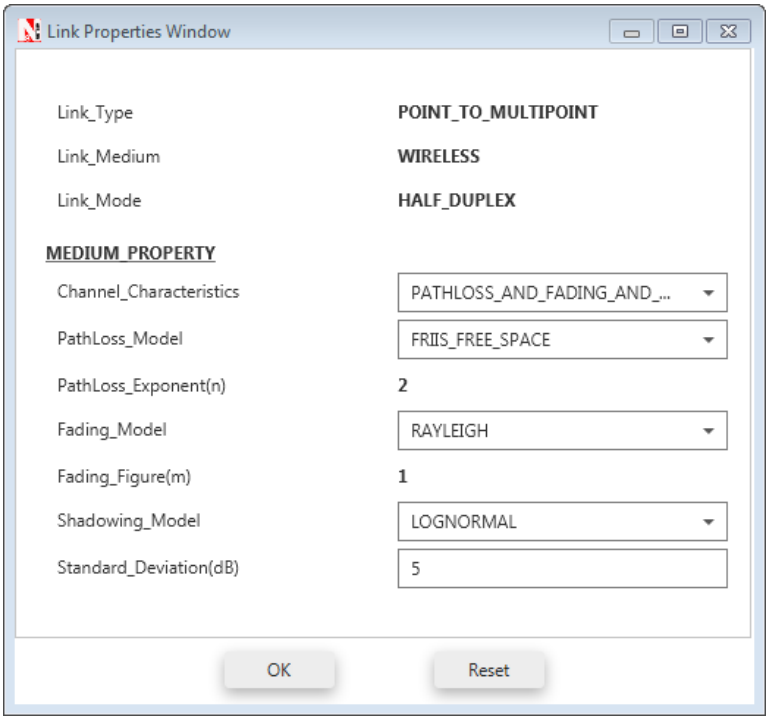
NOTE: To do step 14, check the Windows system path by clicking on Start → Right click on Computer → Properties → Advanced System Settings → Environment variables → System Variables → Open "Path" for editing.

Note: 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 has to change the above mentioned matlab paths to 64-bit matlab paths

- Now Right Click on IEEE802_11 project and select Rebuild



- A new libIEEE802.11.dll gets created in the bin folder of NetSim's current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_32_std_default\bin\bin_x86> for 32-bit. [For more information, follow steps provided in Section8.1 "Writing your own code".
- Run 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.



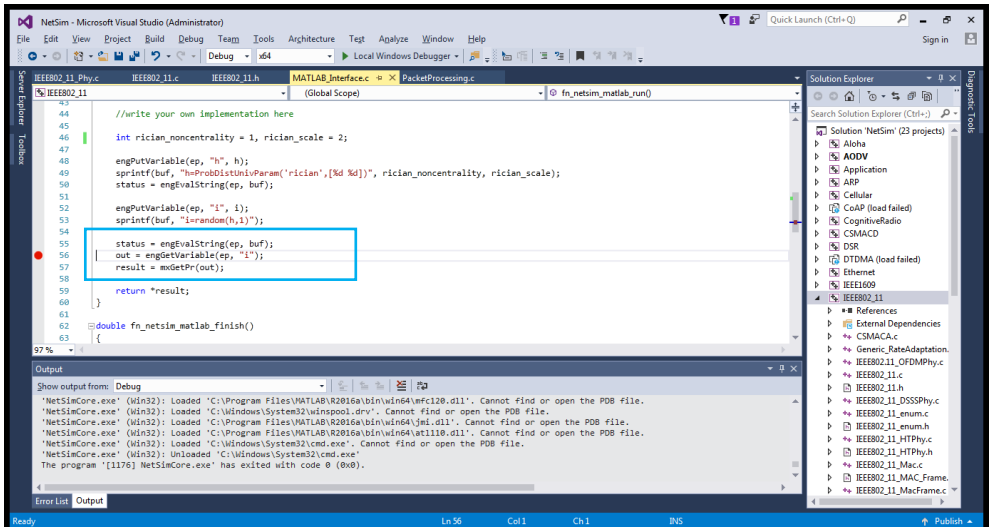
- Perform Simulation. You will find that once the Simulation starts MATLAB command window starts and gets closed once the simulation is over.

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:

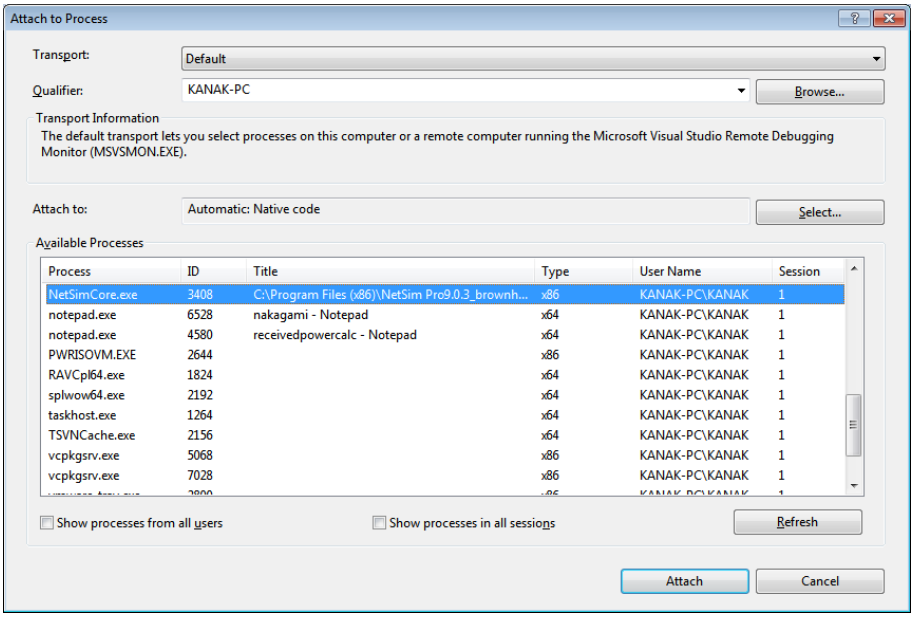
!matlab -regserver

9.2.2 Debug and understand communication between NetSim and MATLAB

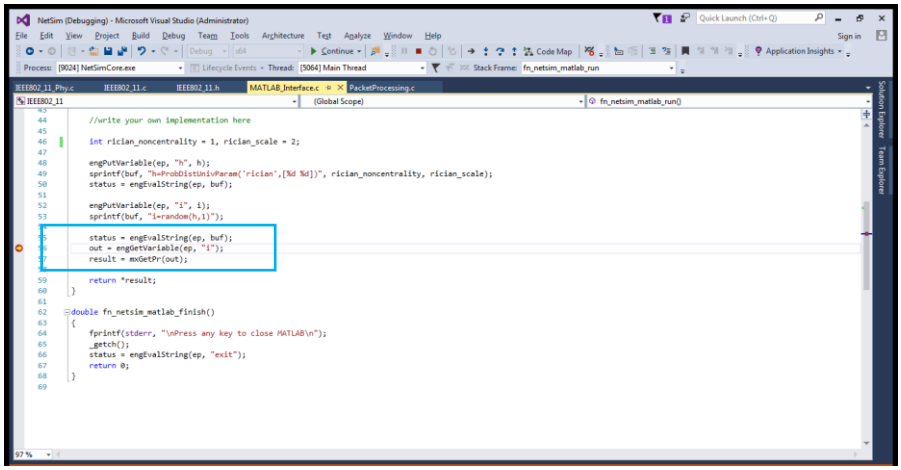
1. In the Solution Explorer double click on MATLAB_Interface.c file and place a breakpoint inside the fn_netsim_matlab_run() function before the return statement.



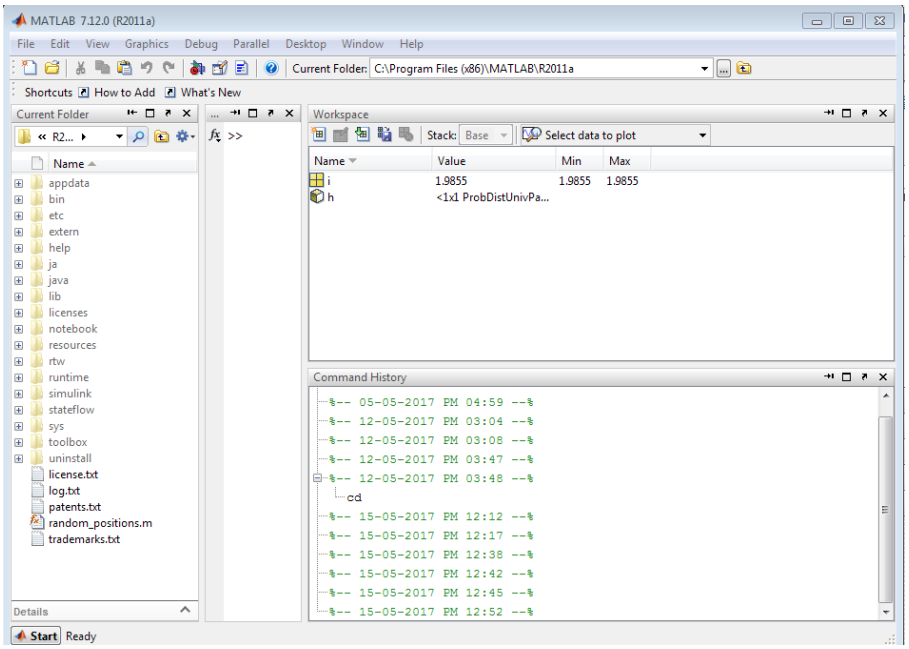
2. Rebuild the code.
3. Now run the NetSim Scenario. The simulation window stops for user interrupt.
4. In Visual studio, go to Debug → Attach to Process.
5. From the list of Processes select NetSimCore.exe and click on Attach.



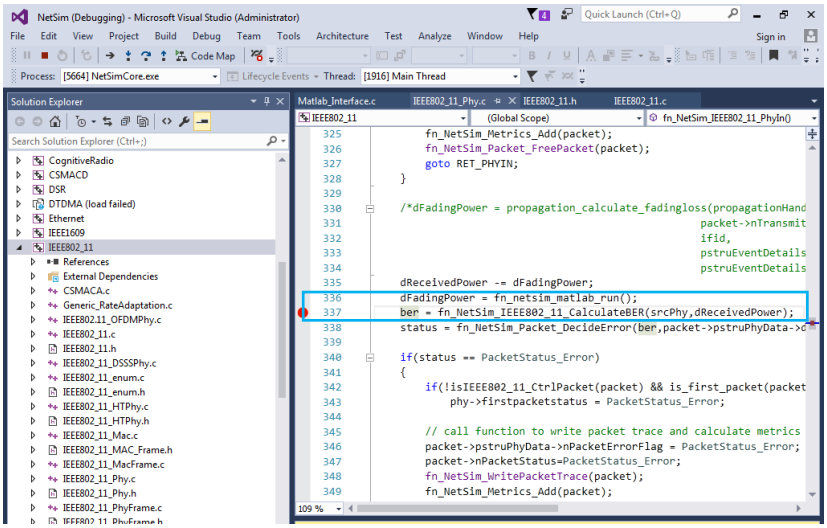
6. Now go to the Simulation window and press Enter.
7. MATLAB Command Window and MATLAB Desktop Window will start and breakpoint in Visual Studio gets triggered.



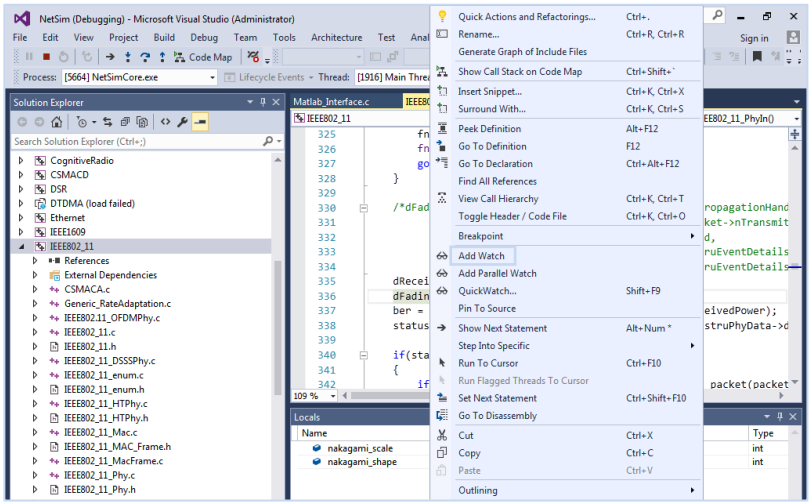
8. Now when debugging (say, by pressing F5 each time) you will find the computation taking place in the MATLAB Workspace.



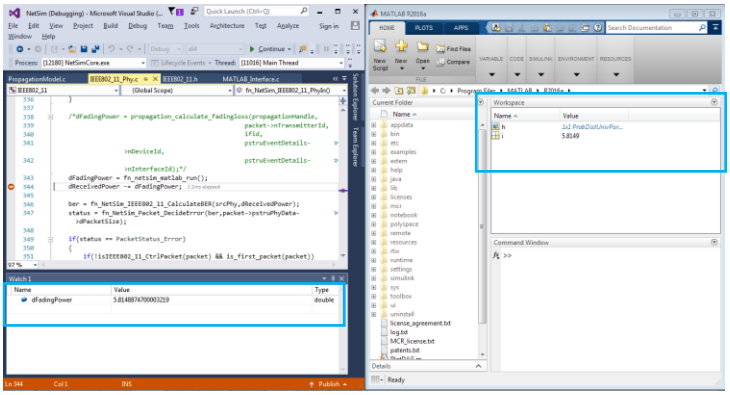
9. This value of i obtained from MATLAB is used to calculate fading power instead of the already available models in NetSim.
10. Now place another breakpoint after the line dFadingPower = fn_netsim_matlab_run()



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.



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.



9.2.3 Implement Rician Distribution of MATLAB in NetSim using .m file:

Procedure:

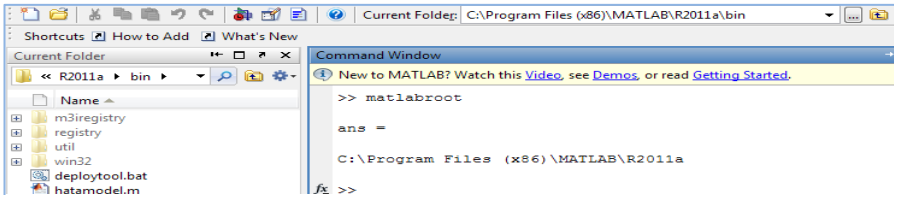
1. Create a file named rician_distribution.m file inside <Path where MATLAB is installed>. The rician_distribution.m file contains the following code:

```
function WLAN= rician_distribution(noncentrality,scale)
%use ProbDistUnivParam() function for matlab 2016
h=ProbDistUnivParam('rician',[noncentrality,scale]);
%use makedist() function for matlab 2017
%h=makedist('rician',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



3. You will have to create a MATLAB_Interface.c file in the IEEE802_11 folder similar to the previous example. The funcitons 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 rician_noncentrality = 1, rician_scale = 2;
  engPutVariable(ep, "h", h);
  sprintf(buf, "k=rician_distribution(%d,%d)", rician_noncentrality, rician_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 Rician Distribution of MATLAB in NetSim without using .m file" above.
5. A call to the rician_distribution () function inside the rician_distribution.m file is made, and rician_noncentrality and rician_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\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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.

9.2.4 Plot a histogram in MATLAB per a Rician 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'rician'
        %use ProbDistUnivParam function for matlab 2016
        h=ProbDistUnivParam('rician',[varargin{1},varargin{2}]);
        %use makedist function for matlab 2017
        %h=makedist('rician',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
```

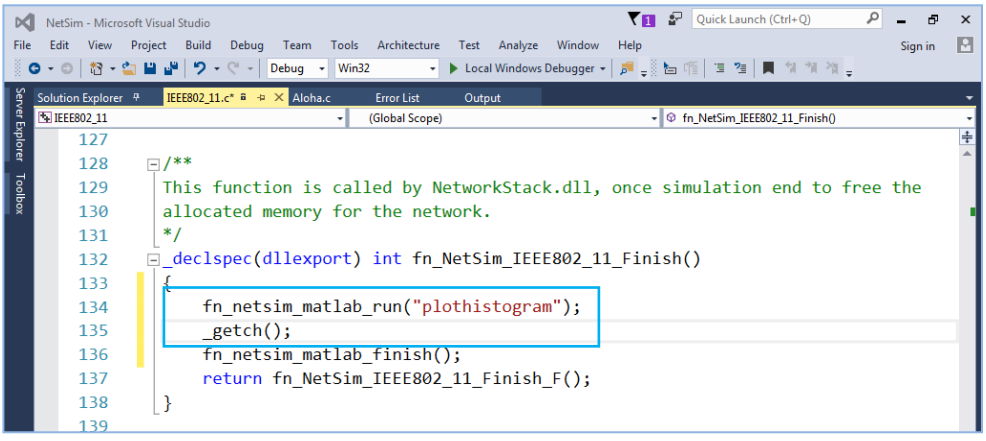
2. Modify the function fn_netsim_matlab_run() that is part of MATLAB_Interfacing.c which was used in the previous example.

```
double fn_netsim_matlab_run(char *arr)
{
                //write your own implementation here

                int rician_noncentrality = 1, rician_scale = 2;
```

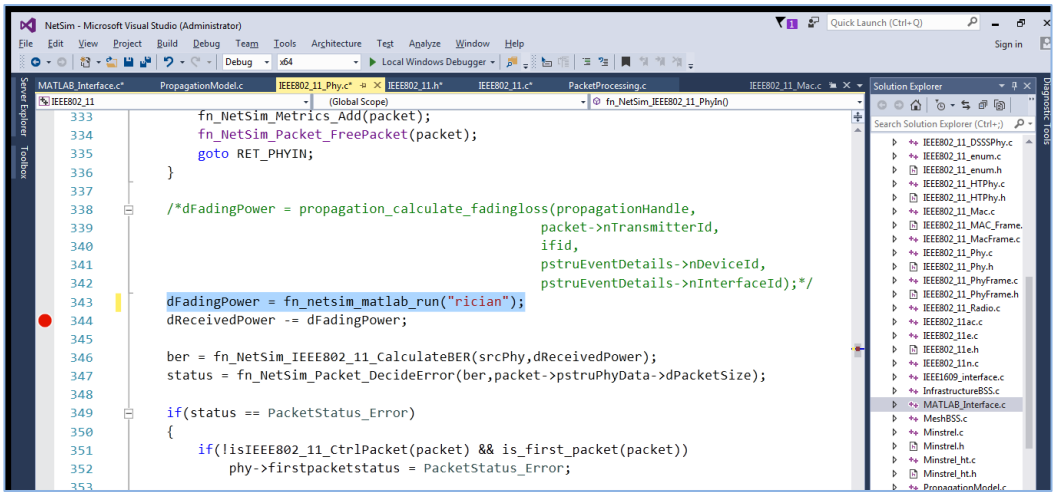
```
		if (strcmp(arr, "rician") == 0)
		{
			engPutVariable(ep, "h", h);
			sprintf(buf, "h=NETSIM_MATLAB('rician',%d,%d)", rician_noncentrality,
rician_scale);
			status = engEvalString(ep, buf);
			out = engGetVariable(ep, "h");
			result = mxGetPr(out);
			return *result;
		}
		else if (strcmp(arr, "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 rician Distribution of MATLAB in NetSim without using .m file” above.

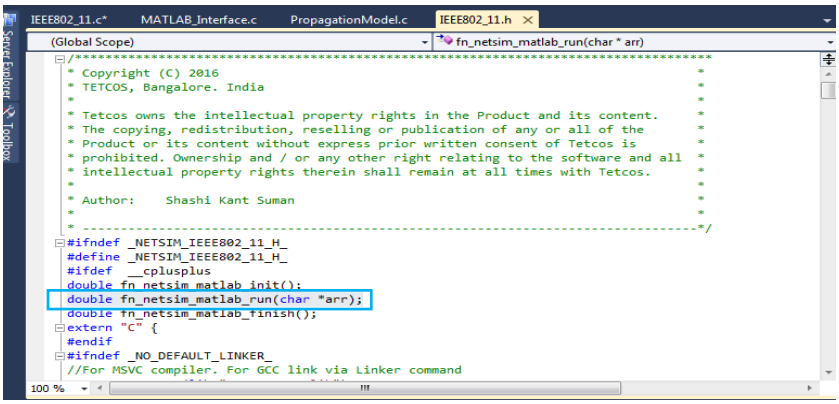
- A call to the NetSim_MATLAB() function inside the NetSim_MATLAB.m file is made, for fading power calculation with parameters distribution(‘rician’), rician_noncentrality and rician_scale parameters are passed from NetSim.
- A call to the NetSim_MATLAB() function inside the NetSim_MATLAB.m file is made, 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.



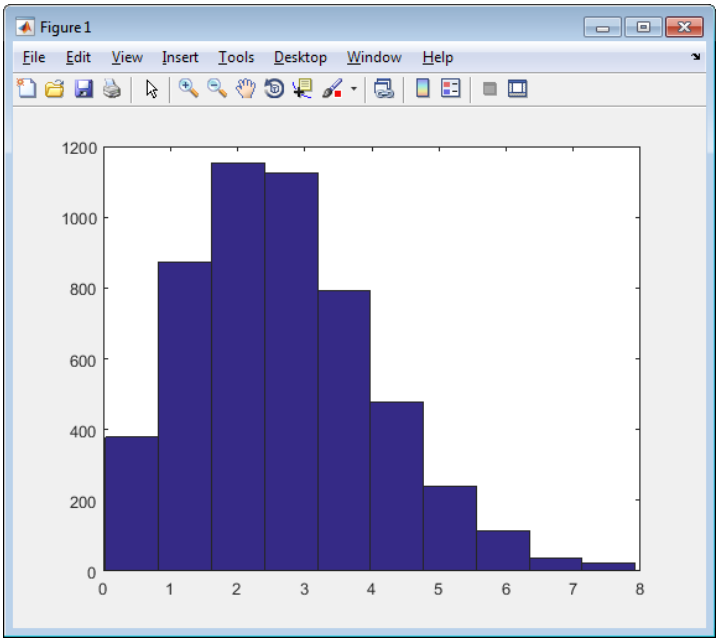
- Similarly in the call made to fn_netsim_matlab_run() function in IEEE802_11_Phy.c file add the parameter “rician” as shown below:-



- Also modify the function definition of fn_netsim_matlab_run() function in IEEE802_11.h file as shown below:



- 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\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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 rician distribution.



- The graph and the MATLAB windows gets 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.

9.3 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 will 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

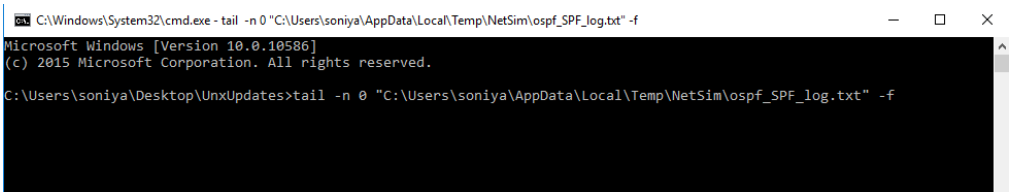
PART 2:

Steps to log NetSim files using tail console

- Open bin folder of NetSim's current workspace path (<C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_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\soniya\AppData\Local\Temp\NetSim\ospf_SPF_log.txt" –f

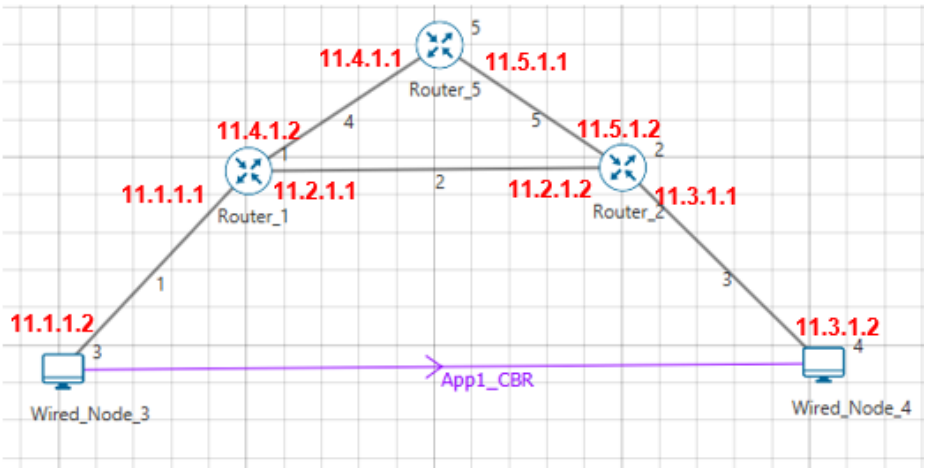
Note: Users need to change the path of the file. In this example we are using ospf_log.txt file



- Open solution file and add the following line in fn_NetSim_OSPF_Init() function in ospf.c file present inside OSPF project

```
OSPF.c
OSPF                    (Global Scope)          fn_NetSim_OSPF_Init(stru_
67
68      _declspec (dllexport) int fn_NetSim_OSPF_Init(struct stru_NetSim_Network *NETWORK_Formal,
69                                                    NetSim_EVENTDETAILS *pstruEventDetails_Formal,
70                                                    char *pszAppPath_Formal,
71                                                    char *pszWritePath_Formal,
72                                                    int nVersion_Type,
73                                                    void **fnPointer)
74      {
75          _getch();
76          register_ospf_log();
77          return fn_NetSim_OSPF_Init_F(NETWORK_Formal,
78                                       pstruEventDetails_Formal,
79                                       pszAppPath_Formal,
80                                       pszWritePath_Formal,
81                                       nVersion_Type,
82                                       fnPointer);
83      }
84
```

- Rebuild the project
- Upon rebuilding, **libOSPF.dll** will get created in the bin folder of NetSim's current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC\NetSim_11.1.7_32_std_default\bin\bin_x86> for 32-bit.
- Create a scenario in NetSim as per the screenshot below and run simulation

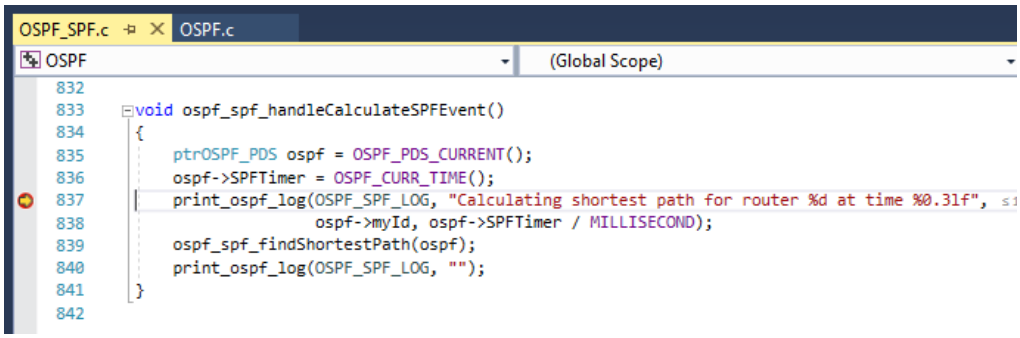


- In the console window user would get a warning message shown in the below screenshot (because of changed DLL) and then the simulation will pause for user input (because of _getch() added in the init function)

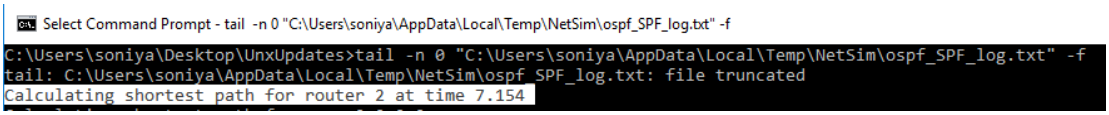
```
C:\Program Files\NetSim Standard_11.0.21\bin\NetSimCore.exe
Today's date is "Dec  05 2018.17:59:05"
Binary build date is "Oct  2 2018.07:04:07"


-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>11>0>rlm_hw>111111111>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....
```

- In Visual Studio, put break point inside all the functions in OSPF_SPF.c file present inside OSPF project

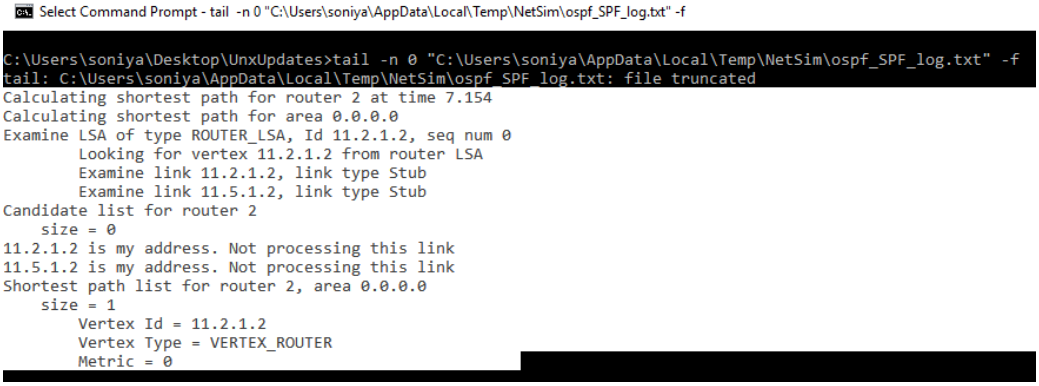
- Go to “Debug->Attach to Process” in Visual studio and attach to NetSimCore.exe.
- Press enter in the command window. Then control goes to the project and stops at the break point in the source code as shown below



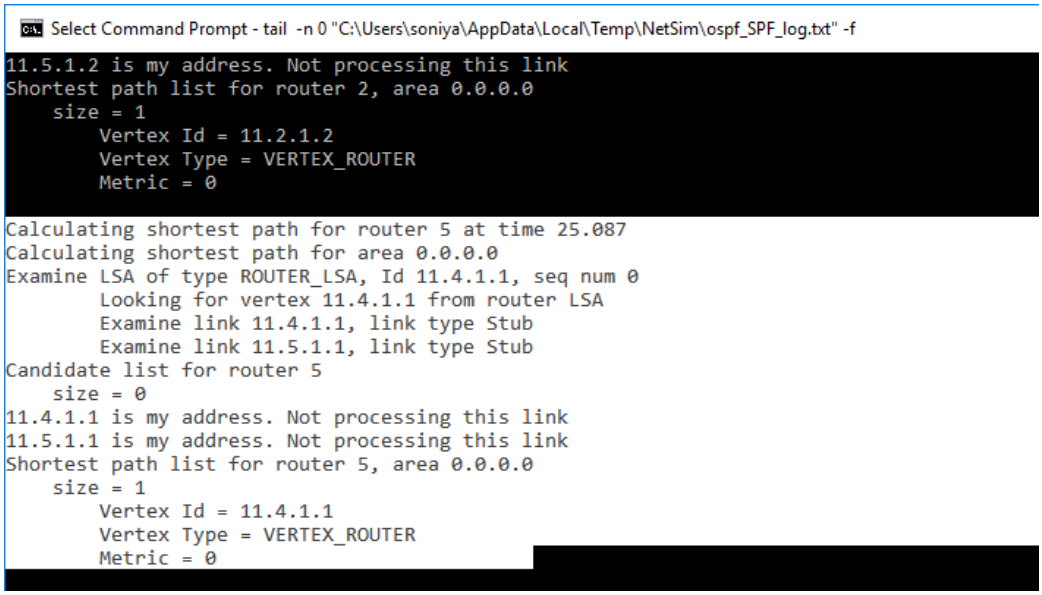
- Press F5 and check the tail console to watch the ospf_SPF log would look like the following screenshot which calculates the shortest path for Router2



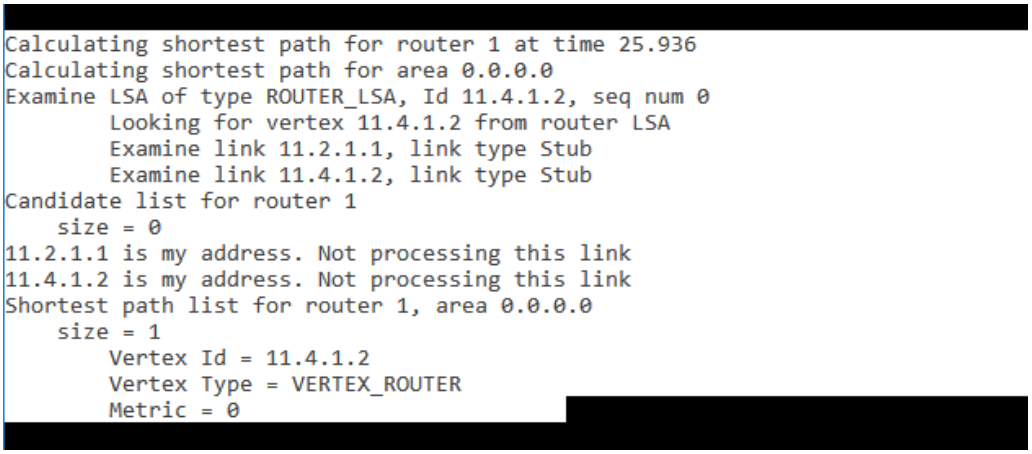
- Keep pressing F5 will add the ospf_SPF log to the tail console. The below screenshot examines the WAN links connected to Router2 i.e. 11.2.1.2 and 11.5.1.2



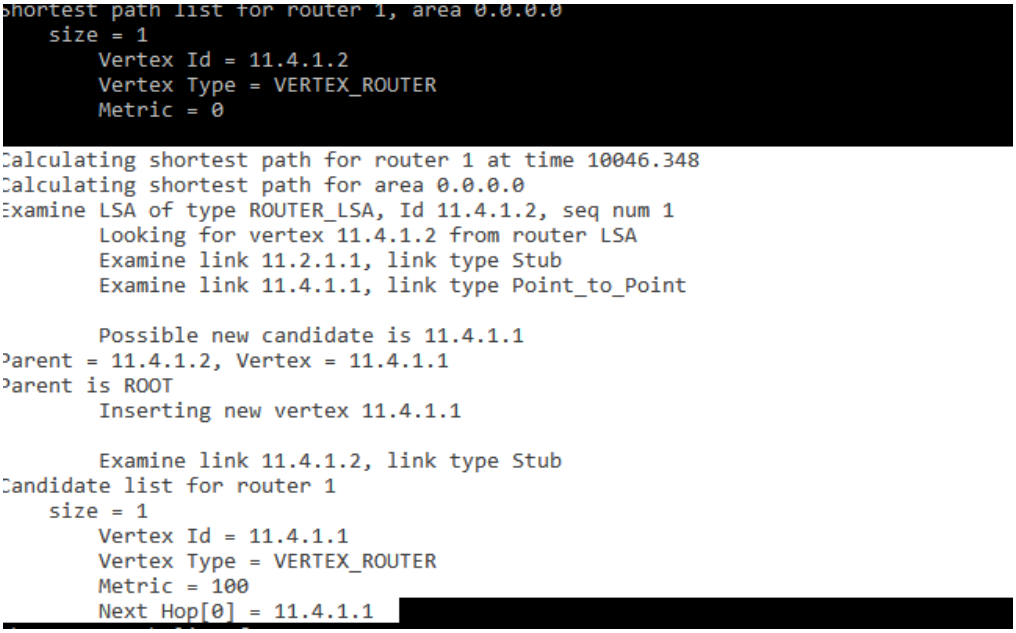
- In the above screenshot, the shortest path for Router2 is 11.2.1.2 with Metrics 0 since it is one of the Router2’s interface
- The below screenshot calculates the shortest path for Router5 and examines the WAN links connected to Router5 i.e. 11.4.1.1 and 11.5.1.1



- In the above screenshot, 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 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



- 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 1st interface) connected to Router1



- Like this, 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 section1

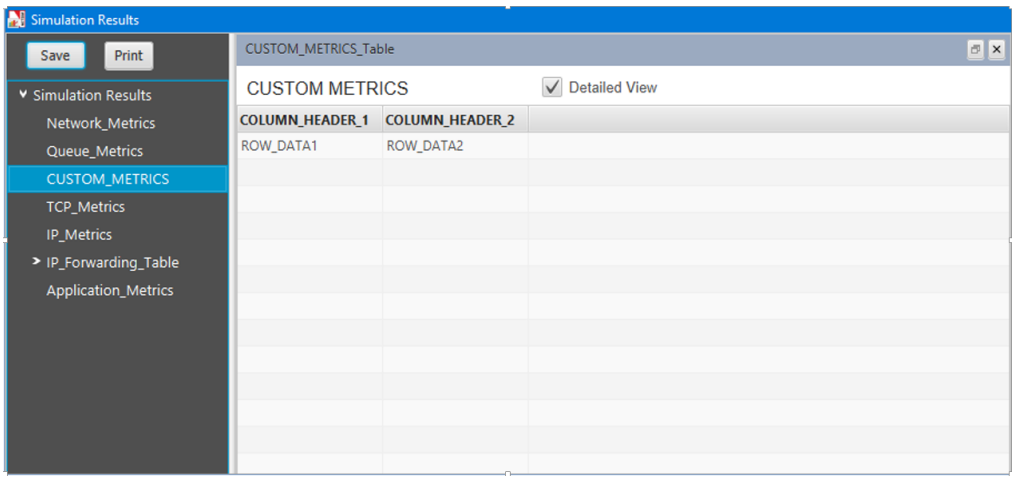
9.4 Adding Custom Performance Metrics

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 has to be done via code 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 Eg: 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);
  }
```



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

```
string NetSim_Node_name[100];
double NetSim_Off_Time[100];
string NetSim_Node_state[100];
```

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;
        int i;
        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 (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)
			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 - 1]);
	}
	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:

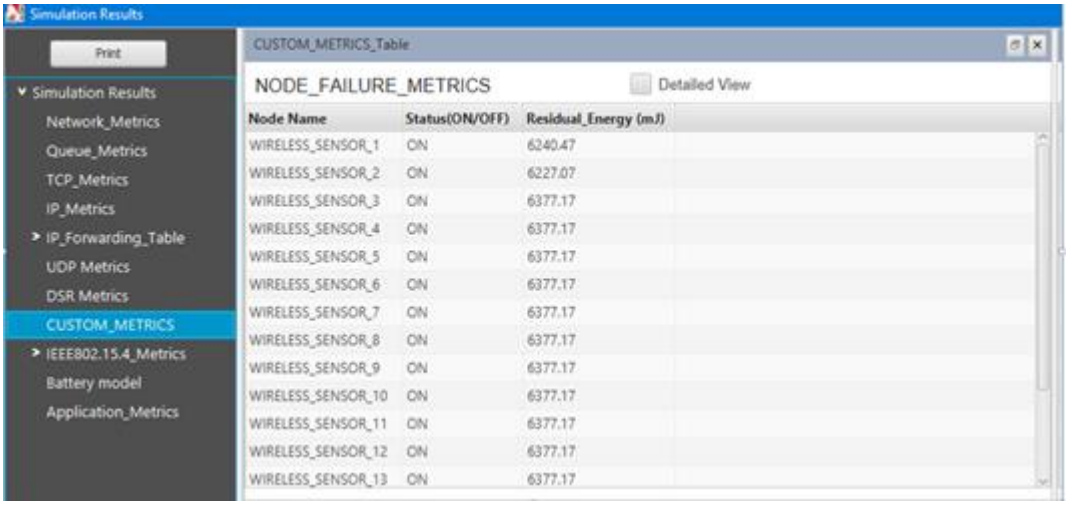
Copy the below code (in red colour) at the end of ChangeRadioState.c file

```
if(isChange)
	{
		phy->nOldState = nOldState;
		phy->nRadioState = nNewState;
		NetSim_Node_state[nDeviceId - 1] = "ON";
		NetSim_Node_name[nDeviceId - 1] =
NETWORK->ppstruDeviceList [nDeviceId - 1] -> szDeviceName;
	}
	else
	{
		phy->nRadioState = RX_OFF;
		WSN_MAC(nDeviceId)->nNodeStatus = OFF;
		NetSim_Off_Time[nDeviceId - 1] = ldEventTime;
```

```
                NetSim_Node_state[nDeviceId - 1] = "OFF";
                NetSim_Node_name[nDeviceId - 1] =
NETWORK->ppstruDeviceList [nDeviceId - 1] -> szDeviceName;
        }

        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:



9.5 Adding Custom Plots

SNR measured by UE can be logged into a text file in a format expected by NetSim metrics window. Upon adding a reference to the SNR plot text files in the Metrics.xml file users can conveniently obtain SNR plots in NetSim Metrics window without having to use other tools like MS Excel, GNU Plot etc. Following is one such example where we log the SNR for each UE in a separate text file and obtain plots in NetSim Metrics window 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 project through the solution explorer and open the **LTE.c** file. In the function **fn_NetSim_LTE_Init()**, modify the function definition as shown below:

```
_declspec(dllexport) int fn_NetSim_LTE_Init()
{
int i = 0, j = 0;
FILE* fp;
char filename[BUFSIZ], nodename[BUFSIZ];
j = fn_NetSim_LTE_Init_F();
```

```
for (i = 0; i < NETWORK->nDeviceCount; i++)
{
if (DEVICE_TYPE(i + 1) == UE &&
DEVICE_MACLAYER(i + 1, 1)->nMacProtocolId == MAC_PROTOCOL_LTE)
{
sprintf(nodename, "LTE_SNR_UE_%d", i + 1);
sprintf(filename, "%s\\%s.txt", pszIOPath, nodename);
fp = fopen(filename, "w+");
if (fp)
{
fprintf(fp, "#Type=Line\n#Heading=SNR Vs Time\n#XLabel=Time(micro\
sec)\n#Num_Y=1\n#YLabel=SNR(dBm)");
fclose(fp);
}

}
}
return j;
return fn_NetSim_LTE_Init_F();
}
```

Step 3:In the function fn_NetSim_LTE_Metrics(), add the lines of code highlighted in red as shown below:

```
_declspec(dllexport) int fn_NetSim_LTE_Metrics(PMETRICSWRITER file)
{
PMETRICSNODE menu = NULL;
PMETRICSNODE rmenu = NULL;
PMETRICSNODE submenu = NULL;
PMETRICSNODE submenu1 = NULL;
char filename[BUFSIZ], nodename[BUFSIZ];
int i = 0;
if (!rmenu)
rmenu = init_metrics_node(MetricsNode_Menu, "Plots", NULL);
if (!menu)
menu = init_metrics_node(MetricsNode_Menu, "LTE_PLOTS", NULL);
add_node_to_menu(rmenu, menu);
for (i = 0; i < NETWORK->nDeviceCount; i++)
{
if (DEVICE_TYPE(i + 1) == UE &&
DEVICE_MACLAYER(i + 1, 1)->nMacProtocolId == MAC_PROTOCOL_LTE)
{
sprintf(nodename, "LTE_SNR_UE_%d", (i + 1));
sprintf(filename, "%s\\%s.txt", pszIOPath, nodename);
submenu1 = init_metrics_node(MetricsNode_Menu, nodename, NULL);
add_node_to_menu(menu, submenu1);
submenu = init_metrics_node(MetricsNode_Plot, nodename, filename);
add_node_to_menu(submenu1, submenu);
}
}
if (menu)
write_metrics_node(file, WriterPosition_Current, NULL, rmenu);
return fn_NetSim_LTE_Metrics_F(file);
}
```

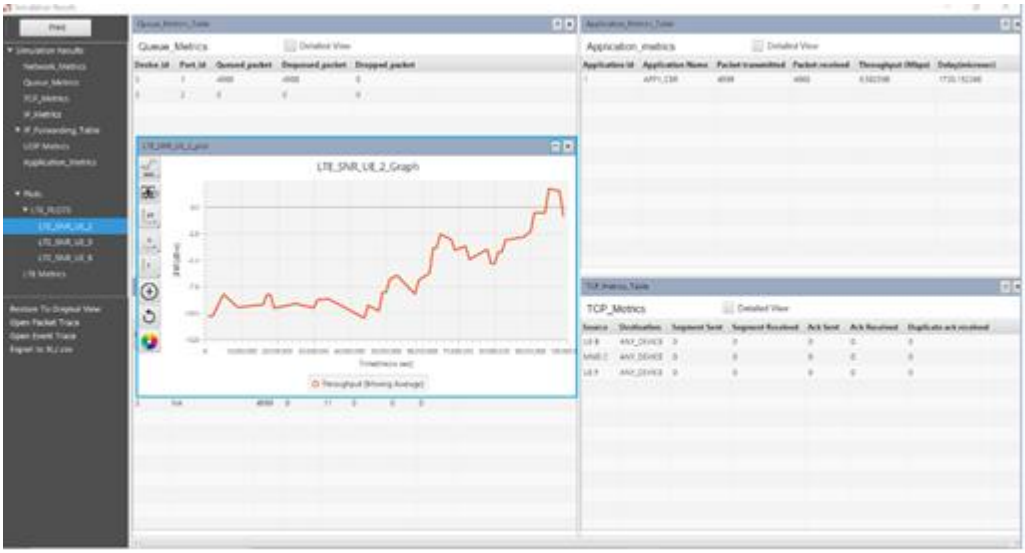
Step 4: Open LTE_Phy.c file and add the lines of code highlighted in red in the function fn_NetSim_LTE_Mobility() as shown below:

```
int fn_NetSim_LTE_Mobility(NETSIM_ID nDeviceId)
{
FILE* fp = NULL;
char filename[BUFSIZ], nodename[BUFSIZ];
double dETime = 0;
unsigned int j;
if(DEVICE_MACLAYER(nDeviceId,1)->nMacProtocolId == MAC_PROTOCOL_LTE)
{
.
.
.
.
else
{
info->DLInfo[j].nCQIIndex--;
info->ULInfo[j].nCQIIndex--;
}
}
}
sprintf(nodename, "LTE_SNR_UE_%d.txt", fn_NetSim_GetDeviceIdByConfigId(\
info->nUEId));
sprintf(filename, "%s\\%s", pszIOPath, nodename);
fp = fopen(filename, "a+");
dETime = pstruEventDetails->dEventTime;
if (fp)
{
fprintf(fp, "\n%lf,%lf", dETime, info->ULInfo[0].dSNR);
fclose(fp);
}
}
else
{
//Do nothing other mac protocol
}
return 1;
}
```

Step 5: Save the changes and right click on LTE module in the solution explorer and select Rebuild.

Step 6: Upon successful build, you will get a new libLTE.dll file in the bin folder of NetSim's current workspace path <C:\Users\PC\NetSim_11.1.7_64_std_default\bin\bin_x64> for 64-bit and <C:\Users\PC5\NetSim_11.1.7_32_std_default\bin\bin_x86> for 32-bit.

Step 7: Now on running any simulation in LTE, you will get individual SNR plots for each UE, in the NetSim Metrics window under **Plots->LTE_PLOTS** as shown below:



9.6 Environment Variables in NetSim

1 NETSIM_PACKET_FILTER = <filter_string>

2 // used by NetSim developers to debug. Emulator code to passes filter string to // windivert. See windivert doc for more information.

3 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

4 NETSIM_EMULATOR = 1 // Set by application dll or user to notify NetSim internal modules to run in emulation mode

5 NETSIM_CUSTOM_EMULATOR = 1 // To notify NetworkStack to not load emulation dll and to only do time sync.

6 NETSIM_SIM_AREA_X = <int> // Area used by Mobility functions for movement of device. Set by config file parser or user.

7 NETSIM_SIM_AREA_Y = <int> // Same as above

8 NETSIM_ERROR_MODE = 1 // if set then windows won't popup gui screen for error reporting on exception.

9 NETSIM_BREAK = <int>

10 // Event id at which simulation will break and wait for user input.

11 // Equivalent to -d command in CLI mode.

12 NETSIM_AUTO = <int> // If set NetSim won't ask for keypress after simulation. //Useful to run batch simulations.

13 NETSIM_IO_PATH = <path>

14 // IO path of NetSim from where it will read Config file and write output file.

15 // Equivalent to -IOPATH command in CLI mode.

```
16 NETSIM_MAP = 1 // Set by Networkstack to inform other modules that simulation is
   running per map view.
17 NETSIM_ADVANCE_METRIC
18 // If set, NetSim provides a set of extra metrics.
19 // In application metrics, you can see duplicate packets received.
20 NETSIM_CONFIG_NAME = <FILE NAME>
21 // Config file name. This file must present in IOPath.
22 // If not set default value is Configuration.netsim
23 NETSIM_NEG_ID = 1 // If set, then control packets will have negative id.
```

10 NetSim Emulator

NOTE: Emulator will be featured in NetSim only if license for Emulator Add-on is available

10.1 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.

10.1.1 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 trasfer using Filezilla, Skype etc

1. Go to the NetSim UI and click Examples.

   The Example Simulation pane appears at the right.
2. Click the type of Emulation example you want to simulate.

   NetSim UI loads the example and displays the stage and the UI options to simulate.
3. To view help documentation for Emulation, click the ‘Book’ link located next the Emulation node.

11 Troubleshooting in NetSim

This section discusses some common issues and solutions:

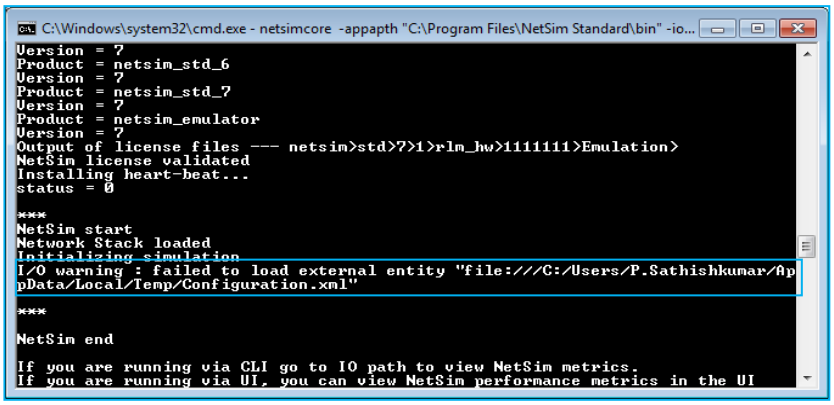
11.1 CLI mode

While running NetSim via CLI for the scenarios described in the Configuration file, you may bump into few problems.

Note: 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.

11.2 I/O warning displayed in CLI mode

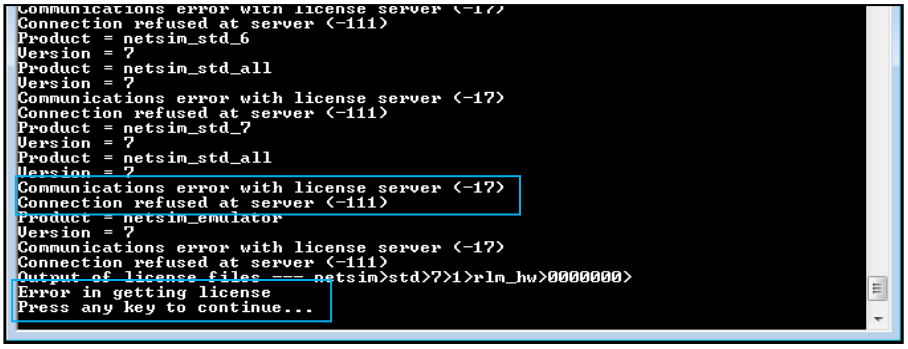
Reason: While typing the CLI command if you enter wrong I/O Path, or if there is no Configuration.netsim file then the following error is thrown



Solution: Please check the I/O path.

11.2.1 Connection refused at server<-111> error displayed:

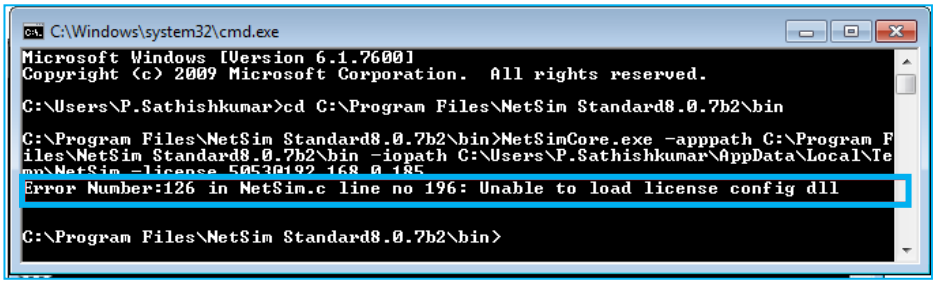
Reason: Unable to communicate with the license server



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 –appppath is typed instead of –apppath then this message will be shown. So, check those details.

11.2.2 Unable to load license config dll(126):

Apppath and I/O path have white spaces

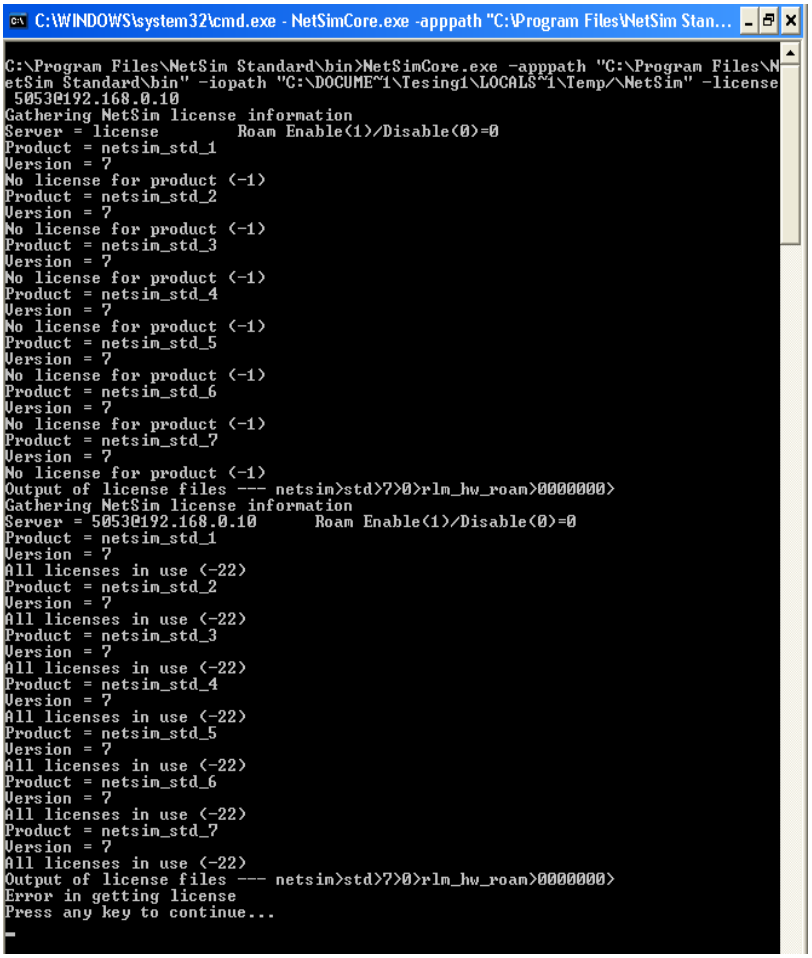


Solution:If the folder name contains white space, then mention the folder path within double quoteswhile specifying the folder name in the command prompt. For example, if app path containswhite space, then the app path must be mentioned within double quotes in the command prompt.

11.2.3 "Error in getting License" error in CLI mode:

Simulation does not commence. "No license for product (-1)" is displayed in the command prompt.

Example:

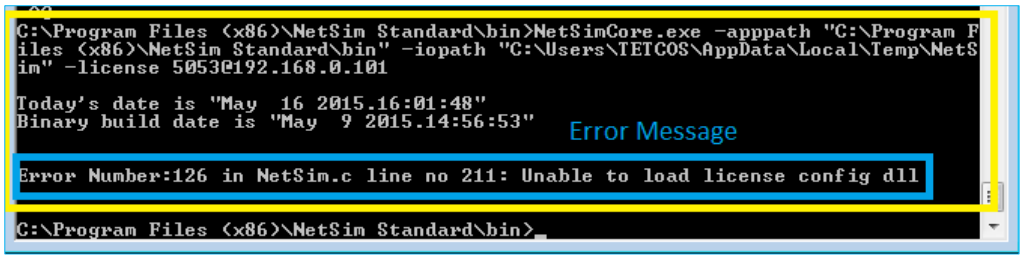


Solution:NetSim is based on the client-server architecture. When NetSim runs in the client machine, it will check for the license in the same machine, first. If license is not available in the same machine, then “No license for product (-1)” will be displayed in the command prompt and the server machine 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.

11.2.4 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.



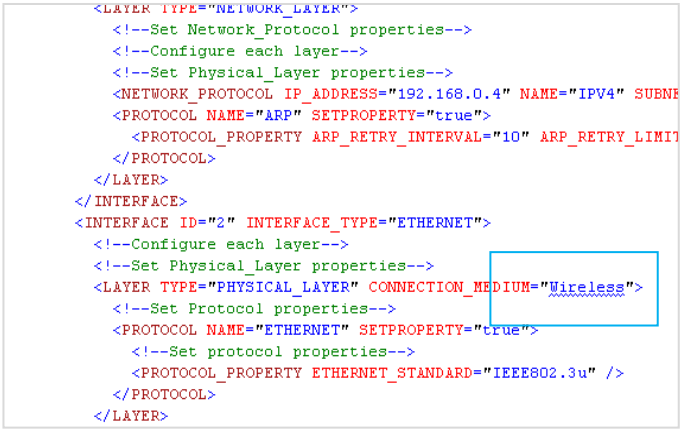
Solution:Type the command manually or copy paste the command/iopath from notepad.

11.3 Configuration.netsim

11.3.1 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.



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.

11.3.2 Error in tags in configuration file attributes:

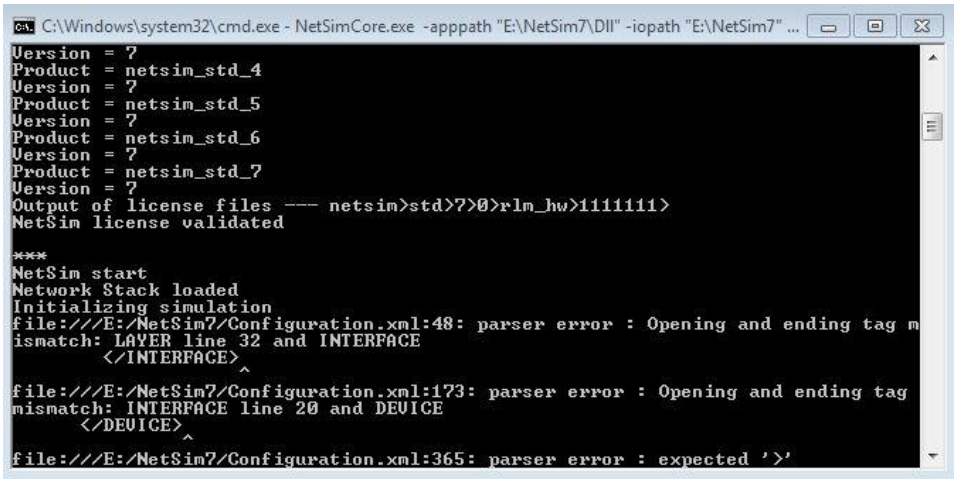
Simulation does not commence and error is displayed at the command prompt. Also, red lines appearing at the tag specifying the Layer in the Configuration file

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="AAAAAAAAAAAA">
      <!--Set Protocol properties-->
      <PROTOCOL_PROPERTY />
    </PROTOCOL>

    <!--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>
```

When NetSim is made to run through CLI, then the following error gets displayed in the command prompt.

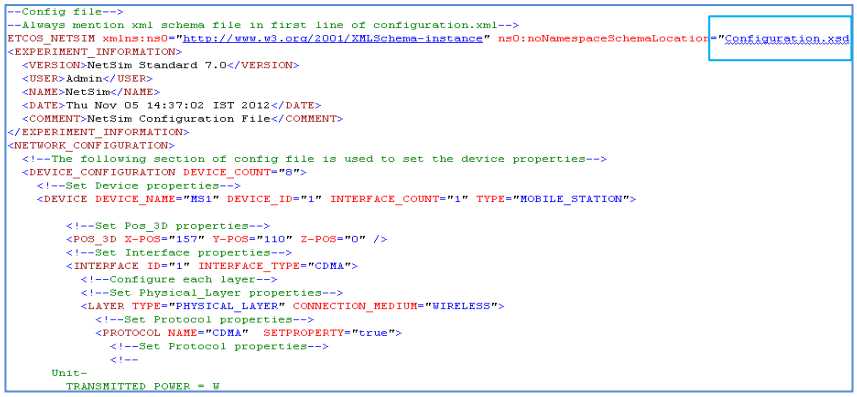


Solution: The bug can be fixed by setting the closing tag correctly in the Configuration file

11.3.3 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.

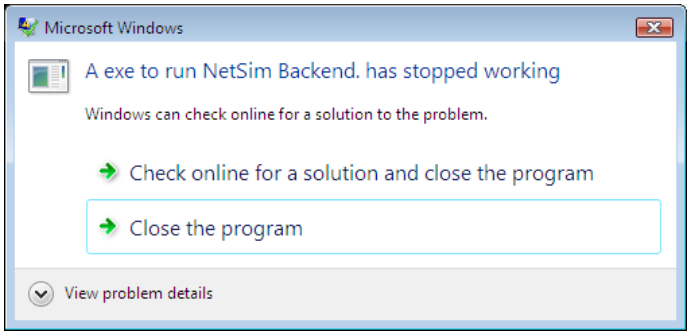


Solution: The bug can be fixed by placing the Configuration file and schema in the same folder.

11.4 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:



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.

11.5 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 **and** 768. For example, 1260 X 720 will also show this error

Solution: Change your monitor resolution to 1024 X 768 or above.

11.6 Licensing

11.6.1 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.

11.7 Troubleshooting VANET simulations that interface with SUMO

11.7.1 Guide for Sumo

- Link for the Sumo Website - http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/for help related to Sumo.
- In case sumo Configuration files do not open, Right click on any Sumo Configuration file, go to properties→open with→sumo.
- While Running NetSim Vanet Simulation – If any message pops up as **“SUMO_HOME” Not found**→ Go to My computer → System Properties → Advanced system settings → Environment Variables. Add an Environment variable as “SUMO_HOME”.
- Sumo Configuration File must contain the paths of the Vehicle routes and Networks file.
- Set the exact End Time for Sumo Simulation in Sumo Configuration File.

11.7.2 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()
```

11.7.3 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.

11.7.4 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 backend.
- “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.

11.7.5 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.

12 Known Issues in NetSim v11.1

List of known issues in v11.1 is available at

https://tetcos.freshdesk.com/solution/articles/14000089935-list-of-known-issues-in-netsim-v11

13 NetSim Videos

In order to have a better understanding of NetSim, users can access YouTube channel of Tetcos at www.youtube.com/tetcos and check out the various videos available

14 R&D projects in NetSim

Example R & D projects in NetSim is available in 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