NetSim User Manual

Contents

1 NetSim – Introduction 12

1.1 Introduction to modeling and simulation of networks 12

1.2 Versions of NetSim – Academic, Standard & Pro 13

1.3 Components in Pro and Standard versions 15

2 Getting Started in NetSim 17

2.1 Installing NetSim in Client 17

2.2 Setting up License Server 24

2.2.1 Installing NetSim RLM Dongle 24

2.2.2 Running RLM Dongle (Server) 26

2.2.3 Running NetSim Software 27

2.3 Menus in NetSim 28

2.3.1 Simulation Menu 29

2.3.2 Programming Menu 30

2.3.3 Basics 31

2.3.4 Help 31

2.4 Modeling and Simulation of a simple network 32

2.4.1 Creating a Network scenario 32

2.4.2 Configuring devices and links in the scenario 33

2.4.3 Modeling Traffic 34

2.4.4 Logging Packet/ Event Trace 35

2.4.5 Simulation 35

2.5 Network – Opening, Saving scenarios and Printing results 36

2.5.1 Opening Saved Experiments 36

2.5.2 Saving an Experiment 36

3 Simulating different networks in NetSim 38

3.1 Internetworks 38

3.1.1 New Experiment 38

3.1.2 Create Scenario 38

3.1.3 Set Node, Link and Application Properties 39

3.1.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 40

3.1.5 Run Simulation 40
3.1.6 Example Configuration files in NetSim 40
3.1.7 TCP – Example Simulations 44
3.1.8 IP Addressing in NetSim 47
3.1.9 Features in WLAN 802.11n/ac 47
3.2 Legacy Networks 53
3.2.1 New Experiment 53
3.2.2 Create Scenario 53
3.2.3 Set Node, Link and Application Properties 53
3.2.4 Modifying/Viewing/Accepting Properties 54
3.2.5 Enable Packet Trace (Optional) 54
3.2.6 Run Simulation 54
3.3 Advanced wireless networks – MANET & Wi-Max 55
3.3.1 New Experiment 55
3.3.2 Create Scenario 55
3.3.3 Set Node, Link and Application Properties 55
3.3.4 Modifying/Viewing/Accepting Properties 56
3.3.5 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 57
3.3.6 Example Configuration files in NetSim 57
3.3.7 Run Simulation 59
3.3.8 Link Layer Acknowledgements and Network Layer Acknowledgements in DSR 60
3.4 BGP 62
3.4.1 New Experiment 62
3.4.2 Create Scenario 62
3.4.3 Set Node, Link and Application Properties 62
3.4.4 Modifying/Viewing/Accepting Properties 63
3.4.5 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 63
3.4.6 Run Simulation 64
3.4.7 Example Configuration files in NetSim 64
3.5 Cellular Networks – GSM/CDMA 67
3.5.1 New Experiment 67
3.5.2 Create Scenario 67
3.5.3 Set Node, Link and Application Properties 67
3.5.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 68

3.5.5 Run Simulation 68
3.5.6 Example Configuration files in NetSim 69
3.6 Wireless Sensor Network 73
3.6.1 Devices in NetSim: 73
3.6.2 Designing WSN Networks 73
3.6.3 Procedure 74
3.6.4 Model features 75
3.6.5 Set Link / Agent Properties 79
3.6.6 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 80
3.6.7 Run Simulation 81
3.6.8 Performance Metrics in WSN 81
3.6.9 The IEEE 802.15.4 Protocol implementation in NetSim 84
3.6.10 CSMA/CA Implementation in NetSim 85
3.6.11 Example Configuration files in NetSim 87
3.7 Internet of Things 89
3.7.1 New Experiment 89
3.7.2 Introduction 89
3.7.3 Set Node, Link and Application Properties 90
3.7.4 Enable Packet Trace, Event Trace & Dynamic Metrics(Optional) 91
3.7.5 Run Simulation 91
3.7.6 RPL protocol in IOT 92
3.7.7 Example Configuration files in NetSim 98
3.8 Zigbee 100
3.8.1 New Experiment 100
3.8.2 Create Scenario 100
3.8.3 Modifying/Viewing/Accepting Properties 100
3.8.4 Set Node, Link and Application Properties 100
3.8.5 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 101
3.8.6 Run Simulation 101
3.9 Cognitive Radio 102
3.9.1 New Experiment 102
3.9.2 Create Scenario 102
3.9.3 Set Node, Link and Application Properties 102
3.9.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 103

3.9.5 Run Simulation 103
3.9.6 Example Configuration files in NetSim 104
3.10 LTE/LTE-A 107
3.10.1 New Experiment 107
3.10.2 Create Scenario 107
3.10.3 Set Node, Link and Application Properties 107
3.10.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 108
3.10.5 Run Simulation 108
3.10.6 Example Configuration files in NetSim 109
3.10.7 Physical speed of the LTE Air Interface 111
3.11 VANETs 114
3.11.1 Introduction 114
3.11.2 IEEE802.11p DSRC/WAVE Protocol Stack: 114
3.11.3 Implementation of 802.11p protocol in NetSim 115
3.11.4 Introduction to NetSim – SUMO interfacing for VANET simulation 116
3.11.5 Run Simulation 119
3.11.6 Example Configuration files in NetSim 121
3.12 Military Radio – TDMA link 16 124
3.12.1 New Experiment 124
3.12.2 Create Scenario 124
3.12.3 Set Node Properties 124
3.12.4 Set Environment Properties 125
3.12.5 Modifying/Viewing/Accepting Properties 125
3.12.6 Set Application Properties 126
3.12.7 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 126
3.12.8 Run Simulation 127
3.13 Military Radio – DTDMA 128
3.13.1 New Experiment 128
3.13.2 Create Scenario 128
3.13.3 Set Node Properties 128
3.13.4 Set Environment Properties 129
3.13.5 Modifying/Viewing/Accepting Properties 130
3.13.6 Set Application Properties 130
3.13.7 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional) 131

3.13.8 Run Simulation 131
3.13.1 Example Configuration files in NetSim 131
3.13.2 DTDMA Packet size 136
3.13.3 Node Join / Leave 138
3.14 Propagation models in NetSim 139
3.14.1 Propagation Loss 139
3.14.2 Path Loss 139
3.14.3 Shadowing 141
3.14.4 Fading 141
3.14.5 SINR Calculation 141
3.14.6 Bit Error Rate (BER) Calculation 142

4 Traffic generator in NetSim (Application Models) 143
4.1 Common properties for all the traffic types 144
4.2 CBR 144
4.3 Custom 145
4.4 Voice 145
4.5 Video 146
4.6 FTP 149
4.7 Database 150
4.8 Peer to Peer 150
4.9 HTTP 151
4.10 Email 151
4.11 Sensor App 152
4.12 Erlang Call 153
4.13 BSM 154
4.14 Emulator 154
4.15 Priority and QoS of Applications 155
4.16 Modelling Poisson arrivals in NetSim 156

5 Running simulation via CLI 158
5.1 Running NetSim via CLI 158
5.1.1 Running CLI via Quick edit mode 162
5.2 Understanding Configuration.xml file 163
5.2.1 How to use Visual Studio to edit the Configuration file? 164

5.2.2 Sections of Configuration file 165
5.2.3 Sample Configuration file 166
5.2.4 Configuration.xsd file 166

6 Analysis 167
6.1 Performance Metrics 167
6.2 Packet Animation 172
6.2.1 Example on how to use NetSim packet animation feature: 174
6.3 Analytics Menu (Multiple Experiments) 177
6.4 Packet Trace 179
6.4.1 Analysing Packet trace using Pivot tables 179
6.4.2 Calculation of Delay, Jitter and Application through put using pivot tables in NetSim event trace 197
6.5 Event Trace (only in Standard/Pro Version) 208
6.5.1 NetSim Network Stack 208
6.6 Trace Data Analysis (only in Standard/Pro Version) 211
6.7 Packet Capture & analysis using Wireshark (www.wireshark.com) (only in Standard/Pro Version) 214
6.7.1 Enabling Wireshark in the network scenario 214
6.7.2 Viewing captured packets 214
6.7.3 Filtering captured packets 215
6.7.4 Analyzing packets in Wireshark 216

7 Custom code in NetSim 220
7.1 Writing your own code 220
7.1.1 Modifying code 220
7.1.2 Building Dlls 222
7.1.3 Building 64 Bit DLL in NetSim 224
7.1.4 Linking Dlls 227
7.1.5 Running Simulation 228
7.2 Implementing your code - Examples 229
7.2.1 Hello World Program 229
7.2.2 Introducing Node Failure in MANET 230
7.2.3 Transferring file from source to destination in WSN 232
7.3 Debugging your code 237

7.3.1 Via GUI 237
7.3.2 How to debug and visualize animation simultaneously 239
7.3.3 Via CLI and co-relating with event trace 242
7.3.4 Viewing & Accessing variables 247
7.3.5 Print to console window in NetSim 255
7.4 NetSim API's 256

8 Advanced Features 258
8.1 Random number Generator and Seed Values 258
8.2 Static Routing 258
8.3 Mobility Models in NetSim 262
8.3.1 Random Walk mobility model 262
8.3.2 Random Waypoint Mobility Model 262
8.3.3 Group mobility 262
8.3.4 File Based Mobility 262
8.4 Interfacing MATLAB with NetSim 265
8.4.1 Implement Nakagami Distribution of MATLAB in NetSim without using .m file 265
8.4.2 Debug and understand communication between NetSim and MATLAB 275
8.4.3 Implement Nakagami Distribution of MATLAB in NetSim using .m file: 279
8.4.4 Plot a histogram in MATLAB using the values generated by Nakagami distribution for NetSim (using .m file) 281
8.5 Adding Custom Performance Metrics 285

9 NetSim Emulator 288
9.1 Introduction 288
9.1.1 Emulation: How Simulation interacts with the real world 288
9.2 Emulation Set-up: 289
9.2.1 Setting up the NetSim Server: 289
9.2.2 Setting up the Client systems (Real Source and Destination system) 290
9.2.3 Setting up the Linux Client Systems (Real Source and Destination system) for Emulation 292
9.2.4 Setting multiple Virtual Machines (VM) to act as Nodes for Emulation 297
9.2.5 Multicasting in NetSim Emulator using JPERF 300
9.3 Emulation examples in NetSim 304
9.3.1 Example Application 1 – PING (One way Communication) 304

9.3.2 Example Application 1 – PING (Two way Communication) 306
9.3.3 Example Application 2 – Video (One way Communication) 307
9.3.4 Example Application 3 – File Transfer using FileZilla (One way Communication) 313
9.3.5 Example Application 4 –Skype (Two way Communication) 316
9.3.6 Example Application 5 – JPerf Network performance measurement graphical tool (One way Communication) 318
9.3.7 Steps to follow for providing pcap file as input to NetSim Emulator 321
9.4 Working of an Emulation Application in NetSim: 325
9.4.1 When I run ping in emulation why is it showing round trip delay but only one direction delay? 327

10 Programming Exercises 331
10.1 Architecture 333
10.2 Creating .exe file for Programming Exercise 334
10.2.1 Using Visual Studio 334
10.2.2 Using GCC 339
10.2.3 Using Dev C++ 341
10.3 Steps to perform Programming Exercise in NetSim 344
10.4 How to De-bug your code linked to NetSim's Programming Exercise 345
10.5 Programming Exercises 348
10.5.1 Address Resolution Protocol 348
10.5.2 Assignment of Sites to Concentrator 351
10.5.3 Cryptography - Substitution - Encryption 354
10.5.4 Cryptography - Substitution - Decryption 356
10.5.5 Cryptography - Transposition - Encryption 359
10.5.6 Cryptography - Transposition – Decryption 360
10.5.7 Cryptography - XOR - Encryption 363
10.5.8 Cryptography - XOR - Decryption 364
10.5.9 Cryptography - Data Encryption Standard (DES) - Encryption 367
10.5.10 Cryptography - Data Encryption Standard (DES) - Decryption 370
10.5.11 Rivest-Shamir - Adleman Algorithm (RSA) 374
10.5.12 Cryptography - Wired Equivalent Privacy (WEP) – Encryption 376
10.5.13 Cryptography - Wired Equivalent Privacy (WEP) - Decryption 378
10.5.14 Distance Vector Routing 381
10.5.15 Distance Host Configuration Protocol 384

10.5.16 Error Correcting Code - Hamming Code 387
10.5.17 Error Detection Code - Cyclic Redundancy Check (CRC) - 12 390
10.5.18 Error Detection Code - Cyclic Redundancy Check (CRC) – 16 393
10.5.19 Error Detection Code - Cyclic Redundancy Check (CRC) - 32 396
10.5.20 Error Detection Code - Cyclic Redundancy Check (CRC) – CCITT 399
10.5.21 Error Detection Code - Longitudinal Redundancy Check (LRC) 402
10.5.22 Framing Sequence – Bit Stuffing 404
10.5.23 Framing Sequence – Character Stuffing 406
10.5.24 Virtual Scheduling Algorithm 409
10.5.25 Address Mask 413
10.5.26 Binary Conversion 415
10.5.27 Classless InterDomain Routing 418
10.5.28 Network Address 422
10.5.29 Special Addresses 424
10.5.30 Subnetting 426
10.5.31 EUI-64 Interface Identifier 429
10.5.32 IPV6 Host Addresses 432
10.5.33 IPV6 Subnetting 435
10.5.34 Leaky Bucket Algorithm 438
10.5.35 Multi Level Multi Access 441
10.5.36 Code Division Multiple Access 443
10.5.37 Time Division Multiple Access 448
10.5.38 Orthogonal Frequency Division Multiple Access 452
10.5.39 PC to PC Communication - Socket Programming TCP 457
10.5.40 PC to PC Communication - Socket Programming UDP 462
10.5.41 PC to PC Communication – Chat Application TCP 468
10.5.42 PC to PC Communication – Chat Application UDP 473
10.5.43 Scheduling - First In First Out (FIFO) 479
10.5.44 Scheduling - Max - Min Fair (MMF) 482
10.5.45 Shortest Path - Floyd's 485
10.5.46 Shortest Path - Link State 488
10.5.47 Sliding Window Protocol - Go Back N 491
10.5.48 Sliding Window Protocol - Selective Repeat 496
10.5.49 Sorting Technique - Bubble Sort 501

10.5.50 Sorting Technique - Insertion Sort 504
10.5.51 Sorting Technique - Quick Sort 507
10.5.52 Sorting Technique - Selection Sort 510
10.5.53 Spanning Tree – Borovska 514
10.5.54 Spanning Tree – Kruskal 517
10.5.55 Spanning Tree – Prims 520
10.5.56 Transmission Flow Control - Go Back N 523
10.5.57 Transmission Flow Control - Selective Repeat 527
10.5.58 Transmission Flow Control - Stop and Wait 531
10.6 Programming exercise - How to practice without NetSim 535

11 Troubleshooting in NetSim 539
11.1 CLI mode 539
11.1.1 I/O warning displayed in CLI mode: 539
11.1.2 Connection refused at server<-111> error displayed: 540
11.1.3 Unable to load license config dll(126) problem: 540
11.1.4 “Error in getting License” error in CLI mode: 541
11.1.5 Unable to load license config dll displayed: 542
11.2 Configuration.xml 543
11.2.1 Blue zigzag lines in configuration file attributes: 543
11.2.2 Red zigzag lines in configuration file attributes: 543
11.2.3 Zigzag lines appearing at configuration.xsd in the Configuration file: 544
11.2.4 Simulation terminates and “NetSim Backend has stopped working” displayed: 545
11.2.5 Monitor screen resolution is less than 1024X768: 545
11.3 Licensing 545
11.3.1 No License for product (-1) error 545
11.4 Troubleshooting for VANET Simulation 547
11.4.1 Guide for Sumo 547
11.4.2 Guide for Python 547
11.4.3 VANET Simulation 548
11.4.4 Python 548
11.4.5 Backend 549

12 Known Issues in NetSim v10 550
12.1 User modified parameters in device properties will not reflect in newly dropped devices 550

12.2 Device properties does not revert to default values 550
12.3 Dynamic Metrics limit is 500 graphs 550
12.4 Running Application between unconnected nodes 550
12.5 RIP Hop count 551
12.6 Default gateway can't be empty 551
12.7 Packet size limit in TDMA (Military Radio) 551
12.8 Packet animation simulation time bar is not functioning properly in WSN, IOT 551
12.9 Broadcast application with RTS threshold value less than 1480 551
12.10 No Radio Range in map view 551
12.11 Scenario crashes if link name field is left empty 552
12.12 Incumbent placement at the end of the grid 552
12.13 Sensors at grid edge with high velocity for agents leads to crash 552
12.14 IP address issue with connection removal between switch and wired node 552
12.15 Network layer ACK type will need to be reset in DTDMA 552
12.16 Wireless link is wrongly established in MANET 552
12.17 In web print we won't get axis value in the plot and in the network metrics table some table headings may be not printed 553
12.18 VANET_EX1-sumo config file, Throwing error in run simulation console during simulation 553
12.19 In Map View print option is not printing the scenario 553

13 What's new in NetSim v10? 554

14 NetSim Videos 555

14.1 R&D projects in NetSim 555

1 NetSim – Introduction

1.1 Introduction to modeling and simulation of networks

A network simulator enables users to virtually create a network along with its components such as devices, links, and applications etc. To study the behavior and performance of the Network.

Some examples of 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 key features essential to any network simulation are -

- **Building the model** – Create a network scenario with devices, links, applications etc
- **Running the simulation** - Run the discrete event simulation (DES) and log different performance metrics
- **Visualizing the simulation**- Use a 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, sub network, link, queue, application etc.
- **Developing your own protocol / algorithm** - Extend existing algorithms by modifying the simulators 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 project work and research while Pro version addresses the needs of defense and industry. The standard and pro versions are available as components in NetSim v10 from which users can choose and assemble. The academic version is available as a single product and includes all the technologies shown below. The main differences between the various versions are tabulated below:

Features	Academic	Standard	Pro
Technology Coverage			
Internetworks	✔	✔	✔
Legacy Networks	✔	✔	✔
BGP	✔	✔	✔
Advanced Wireless Networks	✔	✔	✔
Cellular Networks	✔	✔	✔
Wireless Sensor Networks	✔	✔	✔
Internet of Things	✔	✔	✔
Zigbee	✔	✔	✔
Cognitive Radio Networks	✔	✔	✔
LTE/LTE-A Networks	✔	✔	✔
VANET	✘	✔	✔
Military Radio: TDMA-Link16,HF,VHF,UHF Bands	✘	✘	✔
Basics Understand networking concepts using more than 400 animations	✔	✘	✘
Performance Reporting Performance metrics available for Network and Sub-network	✔	✔	✔
Packet Animator Used to animate the packet flow in network	✔	✔	✔
Packet Trace Available in tab ordered .txt format for easy post processing	✔	✔	✔
Event Trace Available in tab ordered .txt format for easy post processing	✘	✔	✔

Protocol Library Source Codes with Documentation Protocol C source codes and appropriate header files with extensive documentation	✗	✔	✔
External Interfacing Interfacing with SUMO, MATLAB and Wireshark	✗	✔	✔
Integrated debugging Users can write their own code, link their code to NetSim and debug using Visual Studio	✗	✔	✔
Dynamic Metrics Allows users to plot the value of a parameter over simulation time	✗	✔	✔
Simulation Scale	50 Nodes 32 bit build	500 Nodes 32 bit build	1,00,000 Nodes 32bit/64bit build
Custom Coding and Modeling Support	✗	✔	✔
Emulator (Add on) Connect to real hardware running live application	✗	✔	✔
Target Users and Segment	Educational (Lab Experimentation)	Educational (Research)	Commercial (Industrial and Defense)

1.3 Components in Pro and Standard versions

In NetSim v10, users can choose and assemble components for Pro and Standard version. The components are as follows:

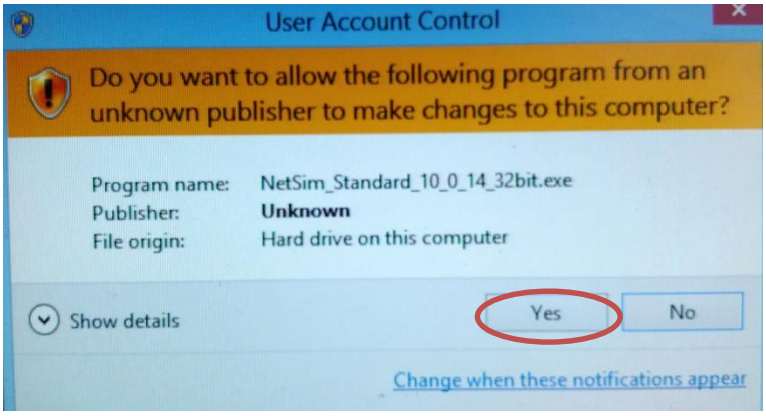
Component No	Networks / Protocols	International Standards
Component 1 (Base. Required for all components)	**Internetworks** Ethernet - Fast & Gigabit, ARP, Routing - RIP, OSPF, WLAN - 802.11 a / b / g /p / n / ac & e, Propagation models - HATA Urban / Suburban, COST 231 HATA urban / Suburban, Indoor 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, Metrics Engine with packet and event trace Plot Generator Packet Animator, Packet Encryption External Interfaces: MATLAB Wireshark	IEEE 802.3 RFC 826 802.11 a/b/g/n/ac/p/e RFC 2453,2328 RFC's 793, 2001 and 768
Component 2	**Legacy Networks** Aloha – (Pure & Slotted) CSMA/CD Token Ring Token Bus	IEEE 802.3 IEEE 802.4 IEEE 802.5 ATM Forum ITU Forum IETF RFC 3031
Component 3	**BGP Networks** Border Gateway Protocol (BGP)	IETF RFC's 1771 & 3121

Component 4	**Advanced Wireless Networks** MANET - DSR, AODV, OLSR, ZRP, and Wi-Max	IETF RFC 4728, 3561, 3626 IEEE 802.16d
Component 5	**Cellular Networks** GSM CDMA	3GPP, ETSI, IMT-MC, IS-95 A/B, IxRTT, 1x-EV-Do, 3xRTT
Component 6 **(Component 4 required)**	**Internet of things (IOT)** with RPL protocol **Wireless Sensor Networks (WSN)** **Personal Area Networks:** ZigBee	IEEE 802.15.4 MAC , MANET in L3
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	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	----
Network Emulator Add On	**Network Emulator – Connect real hardware running live applications to NetSim Simulator**	----

2 Getting Started in NetSim

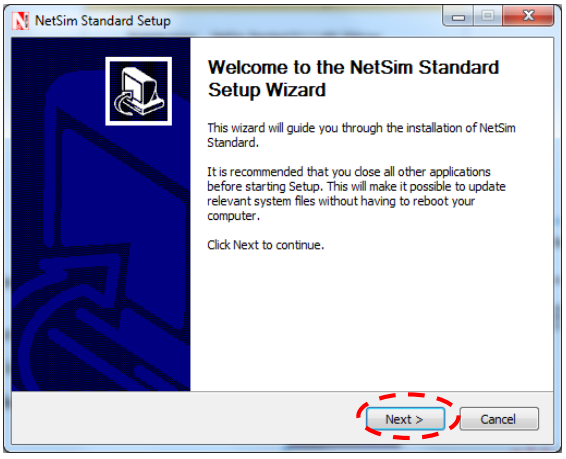
2.1 Installing NetSim in Client

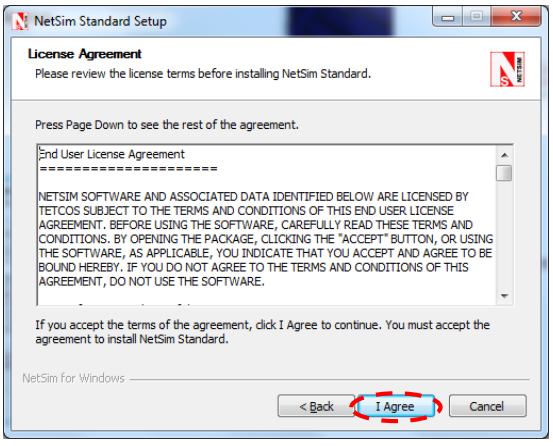
Note: Based on the NetSim version under installation the version type being displayed in the following windows will change. For example you will see NetSim Standard for a standard version install –



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

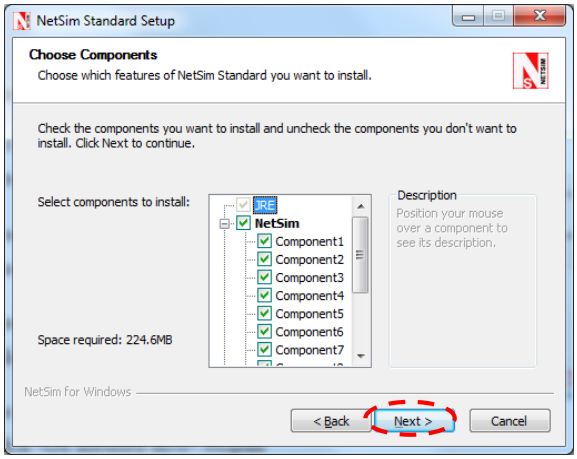
In the next screen, License agreement will be displayed.

Read the agreement carefully, scroll down to read the complete license agreement. If the requirement of the license agreement is accepted 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.

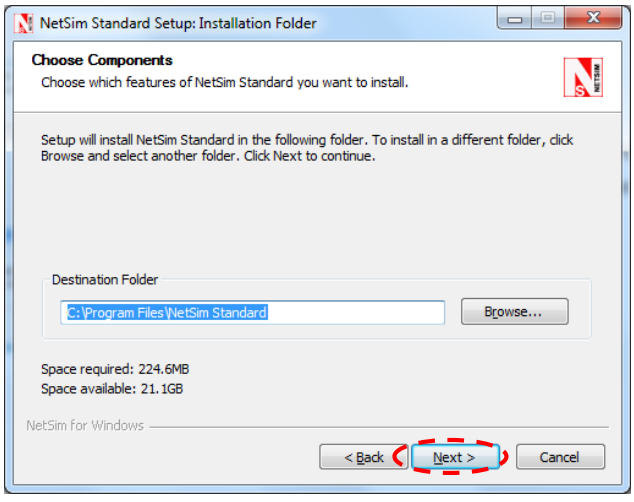
Note: Select all the supporting applications for complete installation of the software.



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.

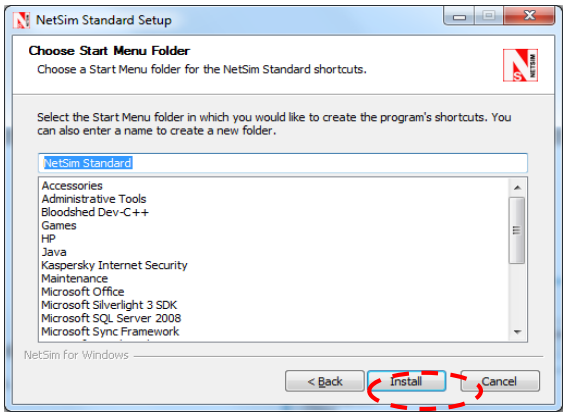
Note: In the case of 64 bit machine, ensure that the path is <OS installed drive>:/Program Files(x86)/NetSim Standard

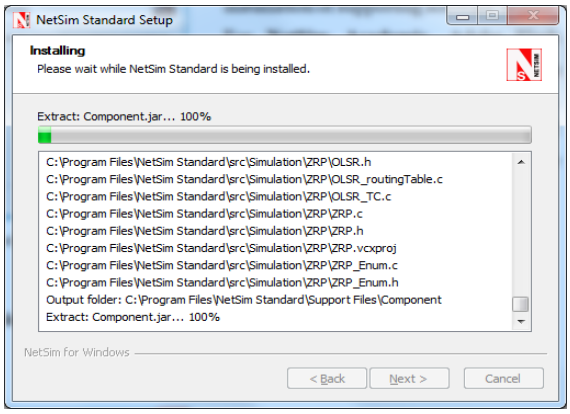


In the next screen, you will be requested to enter the Start menu folder name.

Click on the **Install** button to start the installation.

The installation process begins.



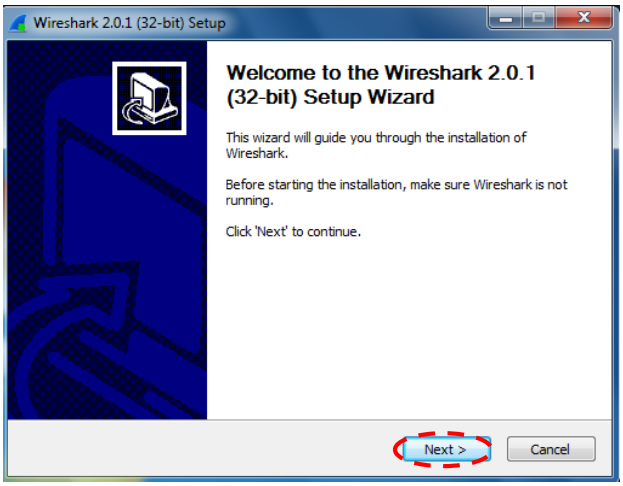


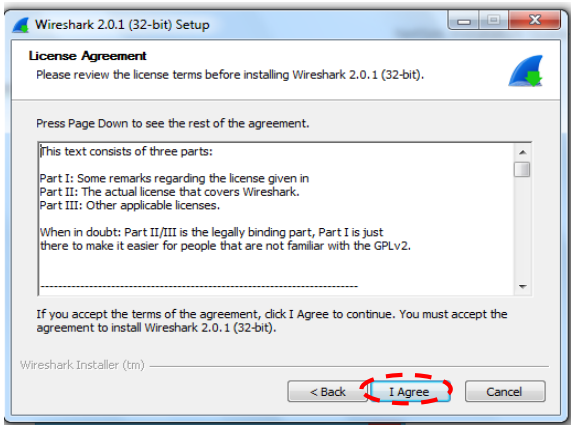
After the installation of required files, the installation of supporting software begins. For **NetSim Academic**, Adobe Flash Player will be installed. For **NetSim Standard Version** and **Pro Version**, gnuplot installation will start by default (if not deselected during 3rd party software selection)

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.

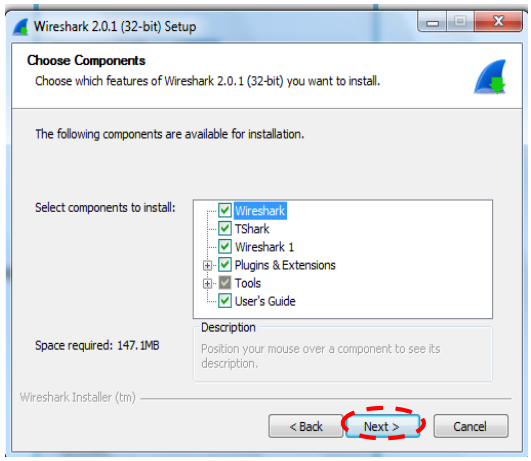
In the following window, click on **I Agree.**

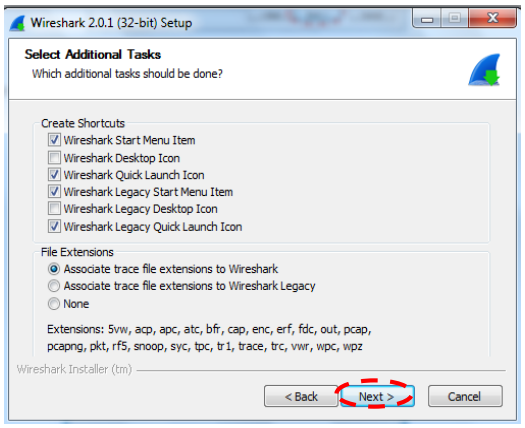




Click on **I Agree.**

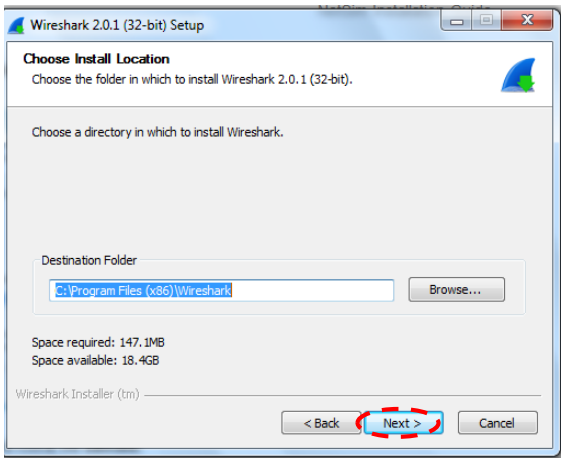
Select all the components and click on **Next.**

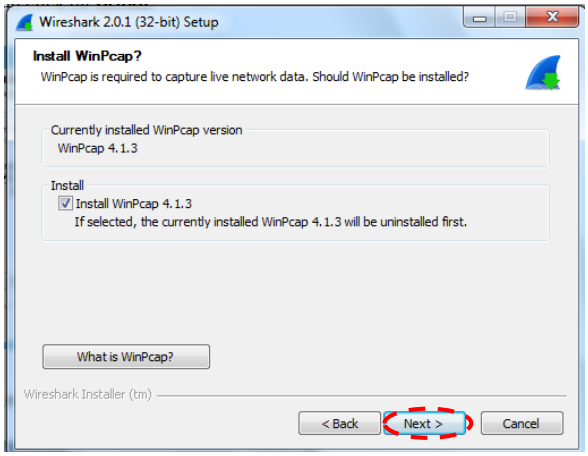




Click on **Next** to go to **Install Location** window as shown below.

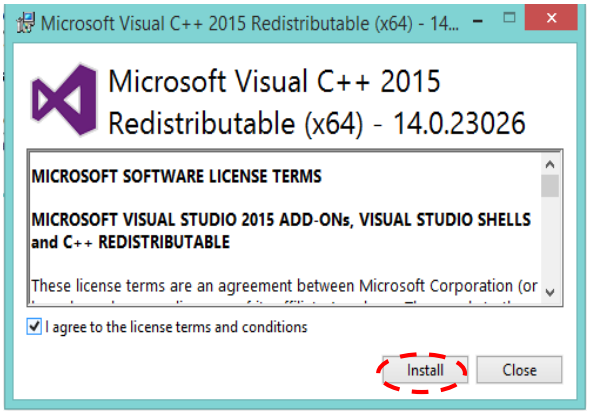
Specify the destination and select **Next.**





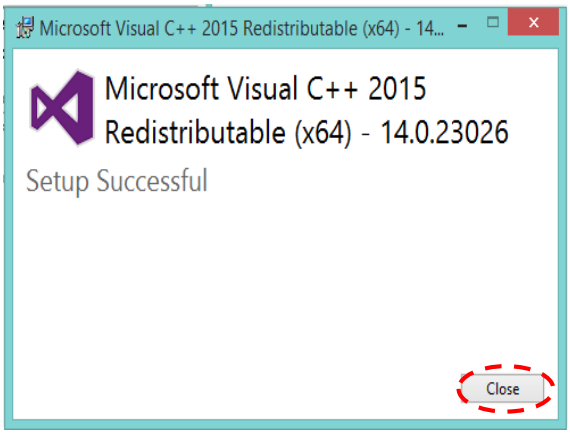
Select **Install WinPcap** and click on **Install.**

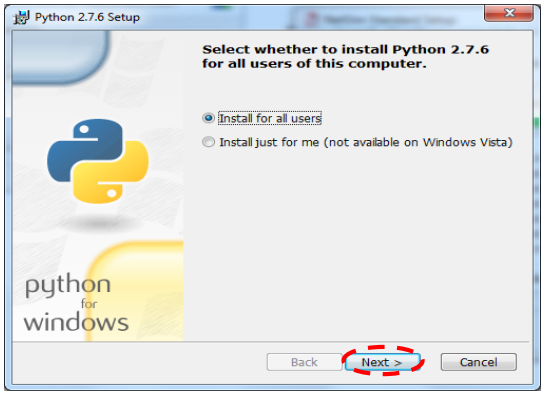
After the installation of the software, you will be requested to click Finish to complete the installation process.



Select I agree and
click on Install to continue installation of Microsoft Visual C++ 2015.

Once installation is completed, a message box will appear as shown in image,
click on close.



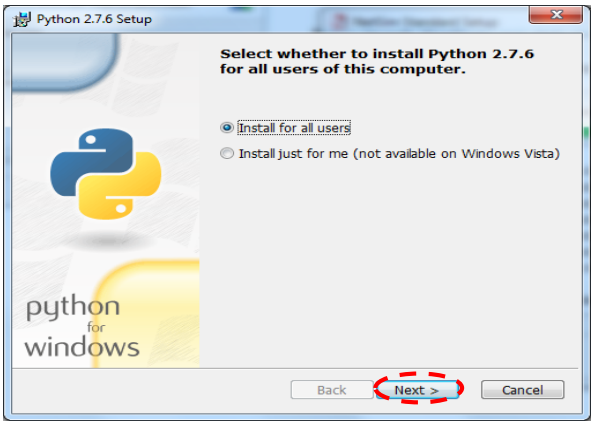


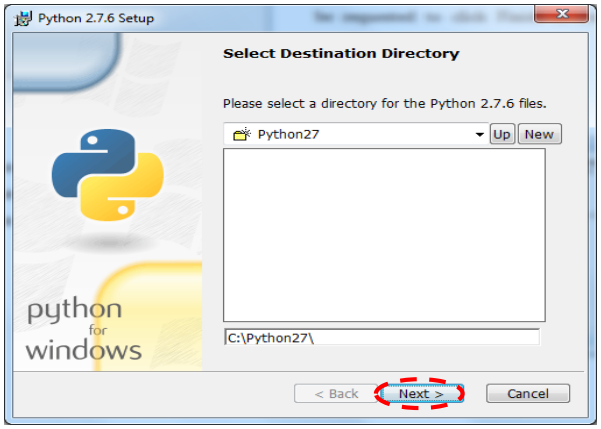
To start Python software installation,
Select whether to install for all users or Particular user alone.
Click on **Next**.

To start Python software installation,

Select whether to install for all users or particular user alone.

Click on **Next**.





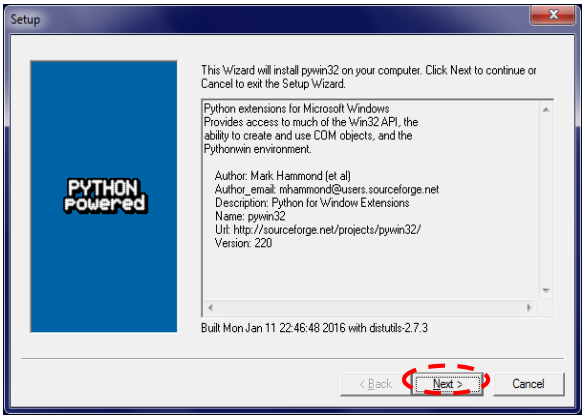
Click on **Next.**



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

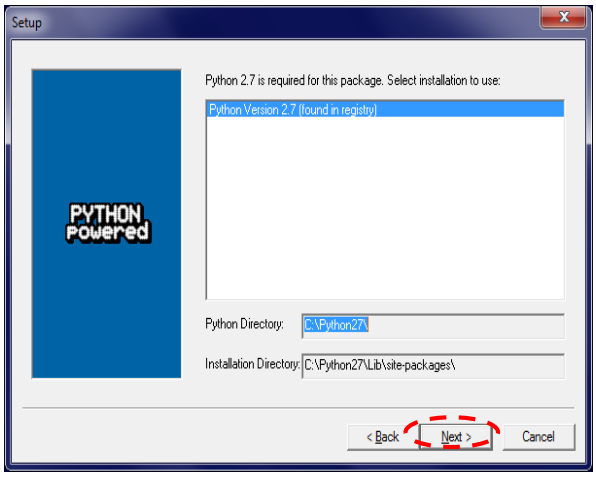
Click on **Finish.**

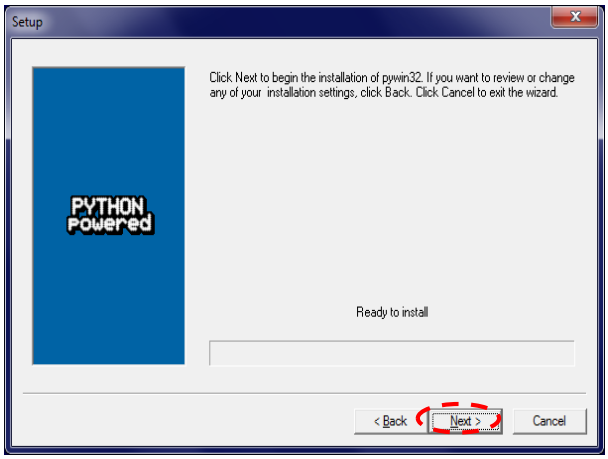




To install Pywin32, Click on **Next**

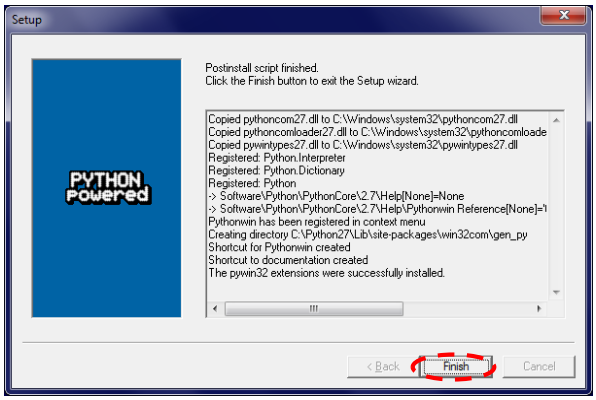
Select the Python directory and Click on Next.





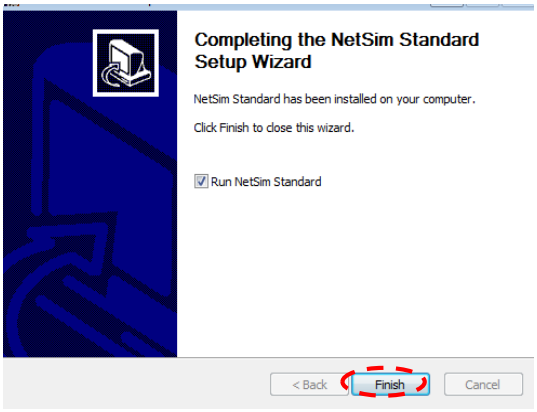
Click on **Next** and the installation Pywin32 begins.

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 are Adobe Flash player and WinPcap.

2.2 Setting up License Server

2.2.1 Installing NetSim RLM Dongle

This section guides you to install the **RLM Dongle** software from the CD-ROM.

- Insert the CD-ROM disc in the CD drive.
- Double click on My Computer and access the CD Drive
- Double click on **Driver_Software** folder.
- Double click on **HASPUserSetup.exe**

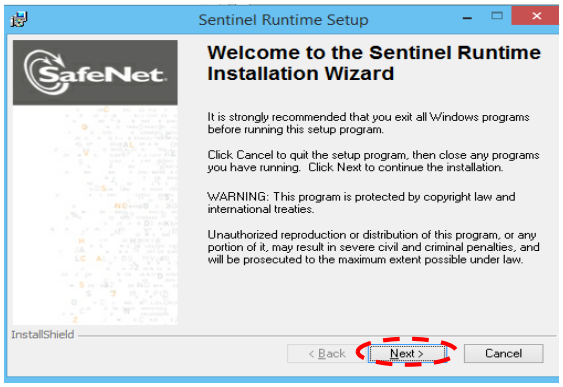
Each prompt displayed during the process tells you what it is about to do and prompts to either **continue** or **Exit**.

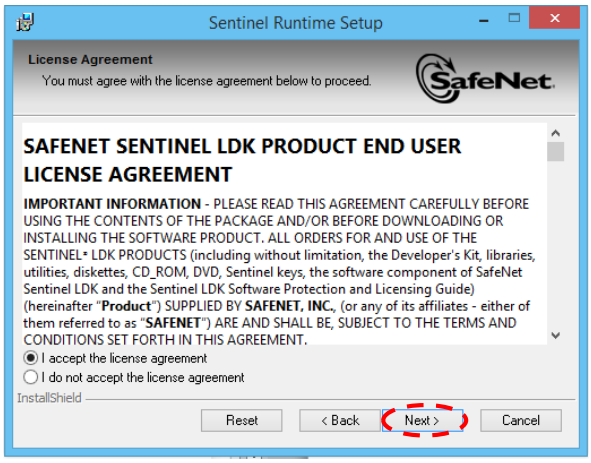
Setup prepares the installation wizard and the software installation begins with a **Welcome Screen**.

Click on the **Next** button

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

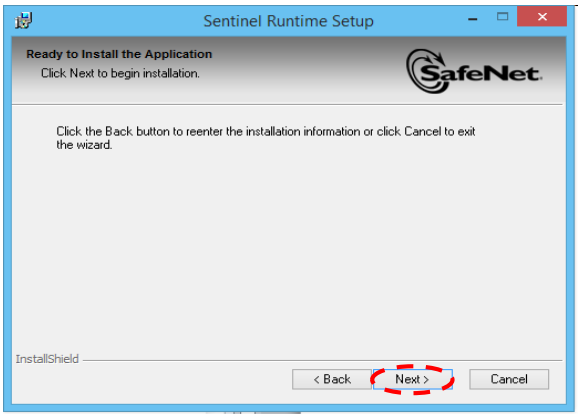
In the next screen, the License agreement is displayed.

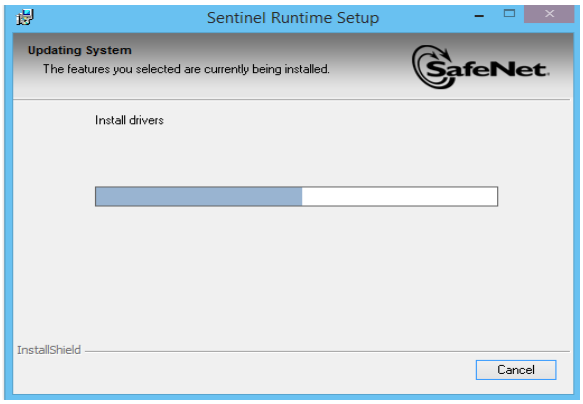




Read the license agreement carefully, scroll down to read the complete license agreement. If the requirement of the license agreement is accepted select the "**I accept**" button else quit the setup by clicking **Cancel** button.

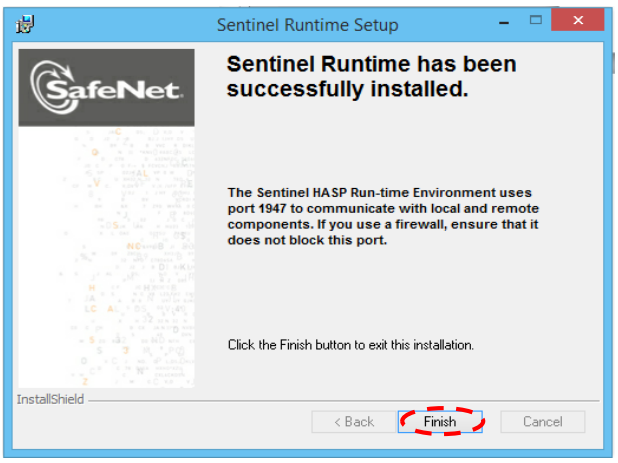
Click on the **Next** button





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.



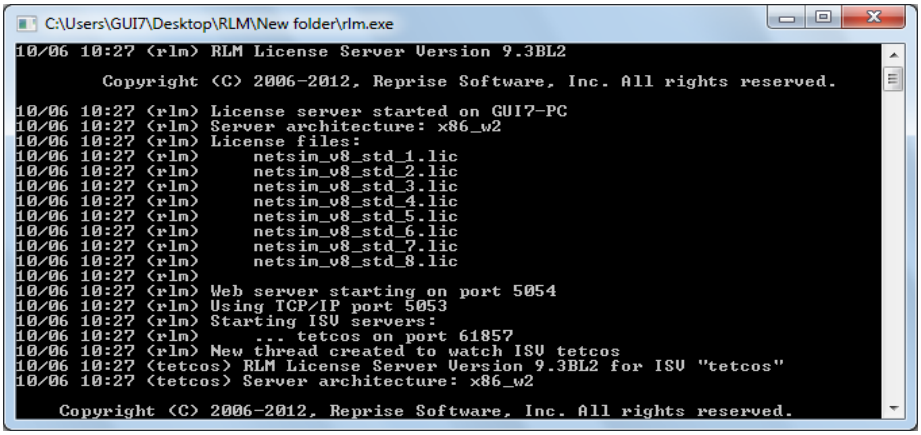
2.2.2 Running RLM Dongle (Server)

After the Driver Software installation, connect the **RLM dongle** to the **system USB port**.

- Double click on My Computer and access the CD Drive
- Copy the **NetSim License Server** folder and paste it on **Desktop**.
- Double click on **NetSim License Server** folder from Desktop.
- Double click on **rlm.exe**

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.

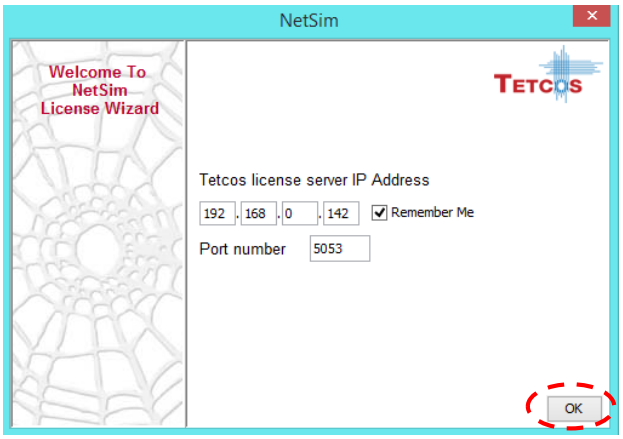
When you run rlm.exe, the screen will appear as shown below.



2.2.3 Running NetSim Software

After running **rlm.exe**, click the NetSim icon in the Desktop.

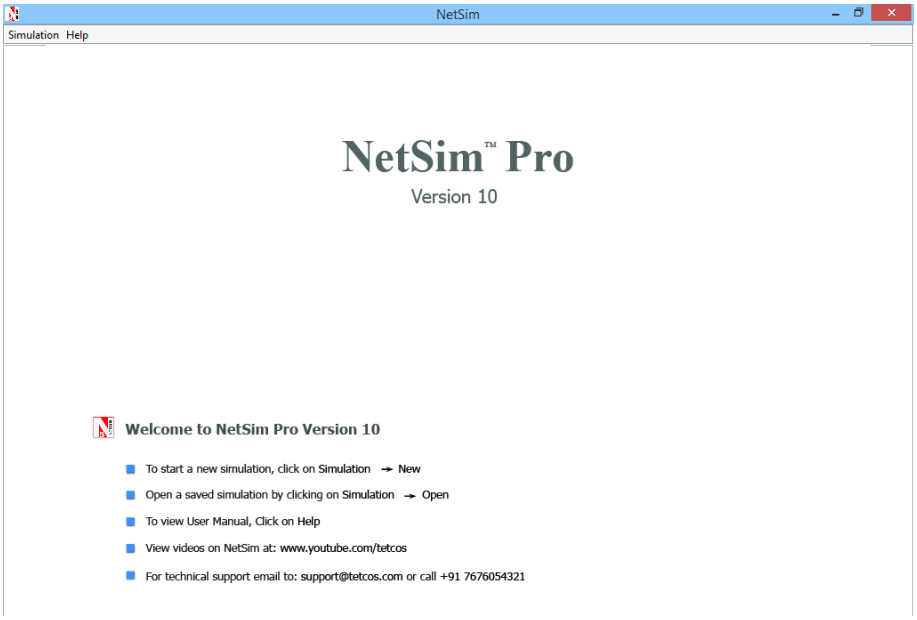
The screen given below will be obtained.



Enter the **Server IP address** where the **rlm.exe** is running, then click ok button.



You have now reached to the main menu of NetSim,

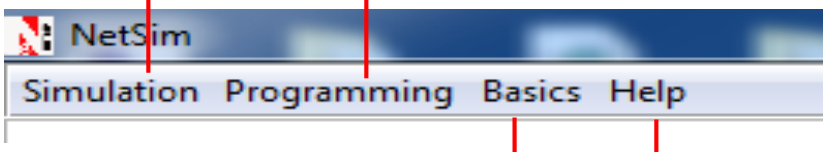


2.3 Menus in NetSim

In Academic/Standard Version

Opens the Simulation menu consisting of New, Open, Save, Save As and Analytics. User can simulate Internetworks, BGP Networks, Advanced Wireless Networks (MANET and Wimax), Cellular, Wireless Sensor Networks, Internet of Things, Zigbee Networks, Cognitive Radio Networks, LTE/LTE-A Networks (LTE/LTE-A, LTE femtocell, LTE D2D) and VANETs.

Opens the Programming menu where different network programming lab exercises are available.

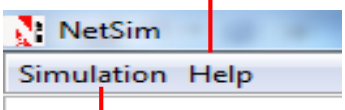


Consists of Animated explanations of networking principles, protocol working and packet formats.
NOTE: Present in Academic version only

Displays all the Help related to NetSim. Help covers Simulation Experiments also.

In Pro Version

Displays all the Help related to NetSim. Help covers Simulation Experiments also.



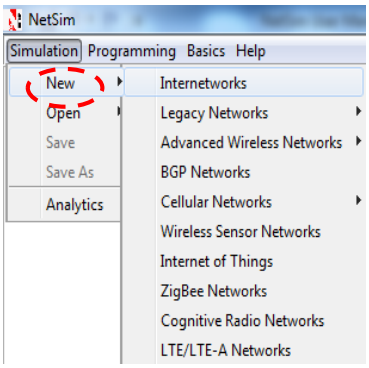
Opens the Simulation menu consisting of New, Open, Save, Save As and Analytics. User can simulate Internetworks, BGP Networks, Advanced Wireless Networks (MANET and Wimax), Cellular, Wireless Sensor Networks, Internet of Things, ZigBee Networks, Cognitive Radio Networks, LTE/LTE-A Networks (LTE/LTE-A ,LTE femtocell, LTE D2D) and VANETs.

2.3.1 Simulation Menu

The **Simulation** menu contains options such as **New**, **Open**, **Save, Save As** and **Analytics**.

2.3.1.1 *New*

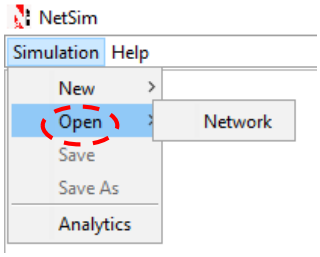
Click on **New** and select the desired protocol from **Internetworks, Legacy Networks, BGP, Advanced Wireless Networks - (MANET and Wi-max), Cellular, Personal Area Networks (Wireless Sensor Network & Internet of Things, ZigBee Networks, Cognitive Radio, LTE/LTE-A Networks (LTE/LTE-A ,LTE femtocell, LTE D2D) and VANETs** to create a new network scenario.



2.3.1.2 *Open*

To open an already created network scenario, go to **Open → Network**.

Refer Section **2.3** for detailed understanding.

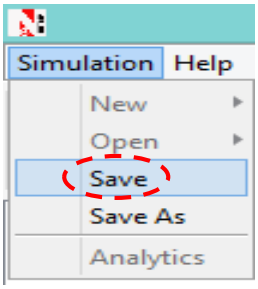


2.3.1.3 *Save*

To save any Network scenario, go to **Save**.

A. To Save any specific experiment, select **Save**, then specify the Experiment Name, Path and click **Ok**.

Refer Section **2.3** for detailed understanding.



2.3.1.4 *Save As*

To save any Network scenario, go to **Save As**.

B. To Save any specific experiment with different name and different path, select **Save As**, then specify the Experiment Name, Path and click **Ok**.

Refer Section **2.3** for detailed understanding.

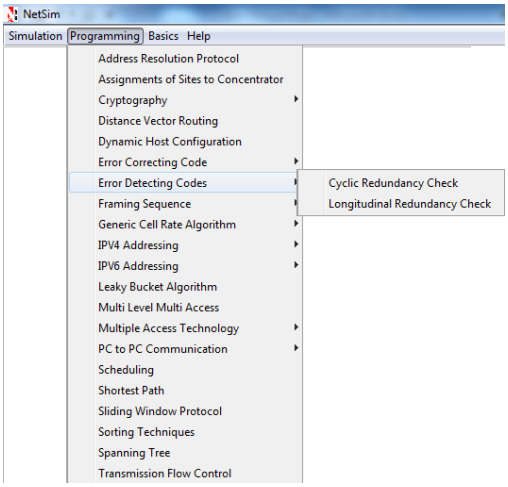
2.3.1.5 *Analytics*

To compare the metrics of various network scenarios of same network, go to **Analytics**. Refer Section **6.4** for detailed understanding.

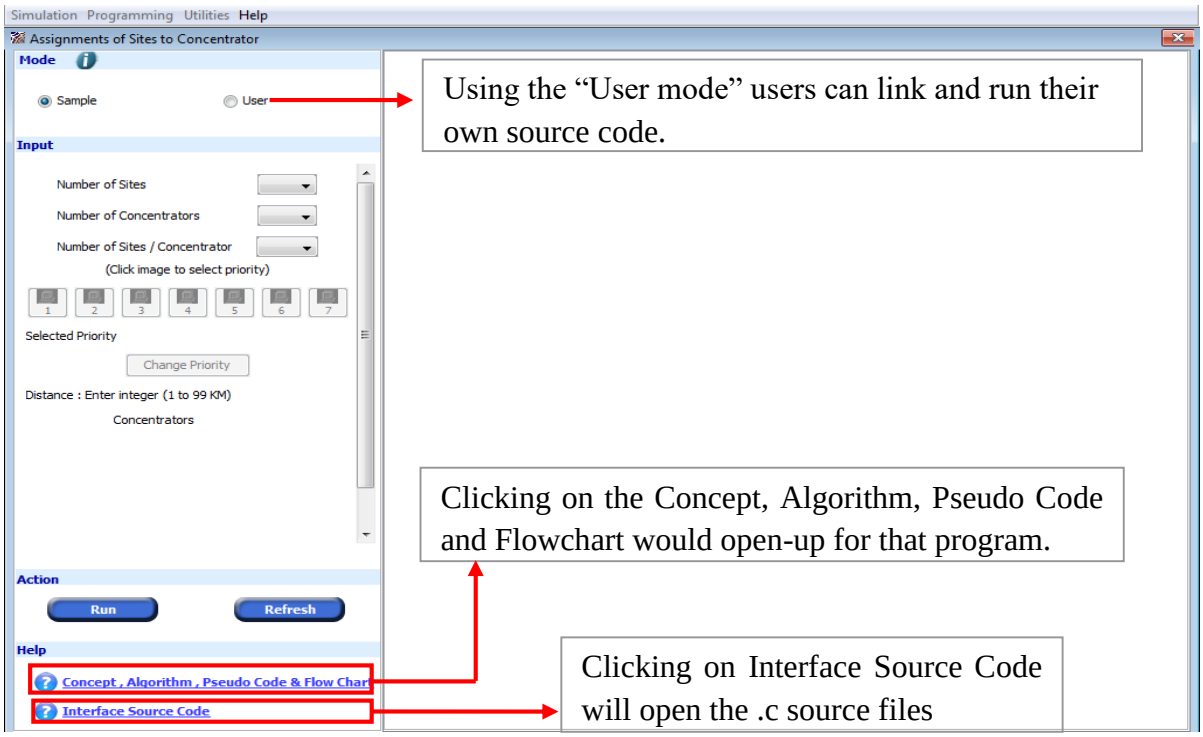
2.3.2 Programming Menu

The **Programming** menu contains network programming exercises. Click on this menu and select the desired programming exercise.

Note: This menu is available only in Academic and Standard Version.



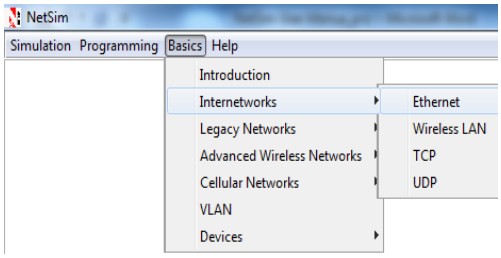
Upon selection, the following screen will appear. Please refer Section 9 (NetSim Academic/Standard User Manual) for detailed help.



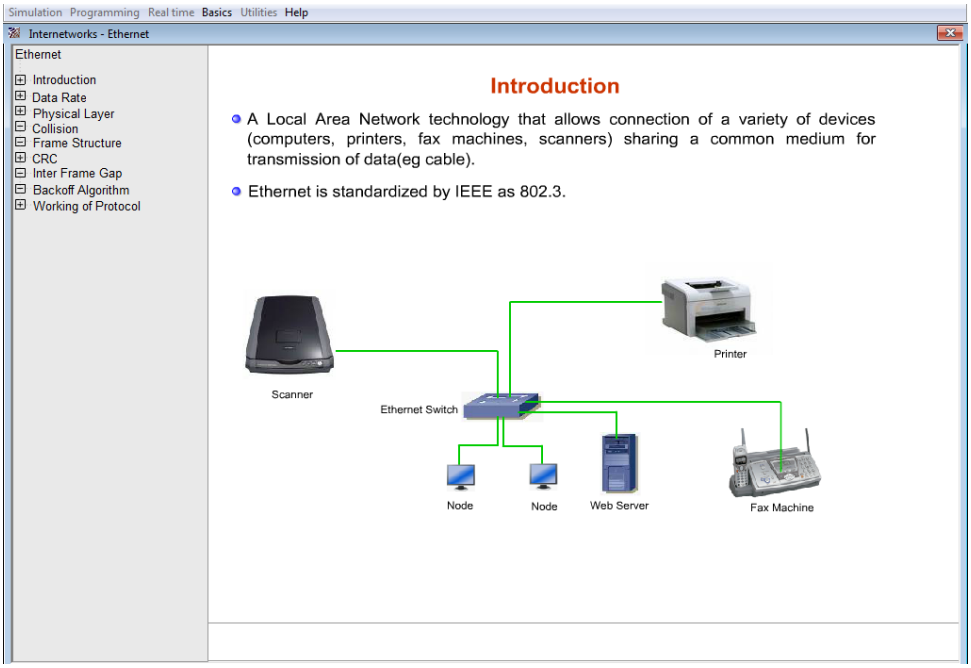
2.3.3 Basics

In this menu, various network concepts are explained using animations.

Note: This menu is available only in Academic Version.



For example: On selecting "**Ethernet**" under "**Internetworks**", following screen will be displayed.



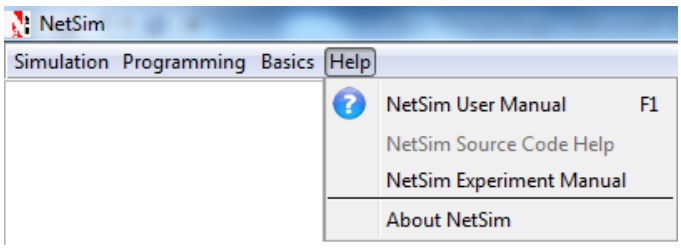
2.3.4 Help

This menu contains all the **Help** related to **NetSim**.

NetSim User Manual - It contains the documentation about the working of NetSim.

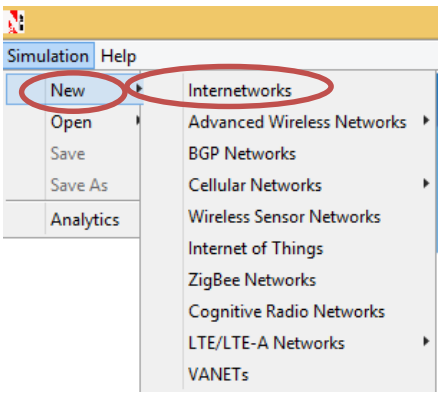
NetSim Source Code Help – It contains the documentation about network protocol source code *(Available with NetSim Standard as well as Pro Version)*

NetSim Experiment Manual – It contains selected experiments which can be performed in NetSim.



2.4 Modeling and Simulation of a simple network

This section will demonstrate how to create a basic network scenario and analyze in NetSim.



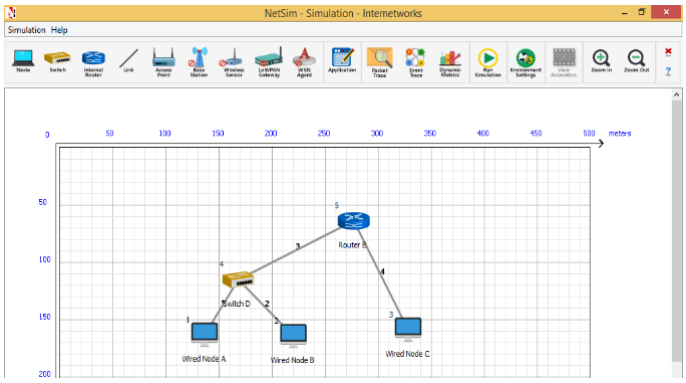
Let us consider Internetworks. To create a new scenario, go to Simulation → New → Internetworks

2.4.1 Creating a Network scenario

Consider we want to design a network with two subnets in Internetworks.

Let us say the subnet 1 consists of two wired nodes connected via a Switch and the other subnet 2 is a wired node. Both the subnets are connected using a Router. Traffic in Network is generated from a wired node in subnet 1 to the wired node in subnet 2.

Perform the following steps to create the network scenario which looks like this:



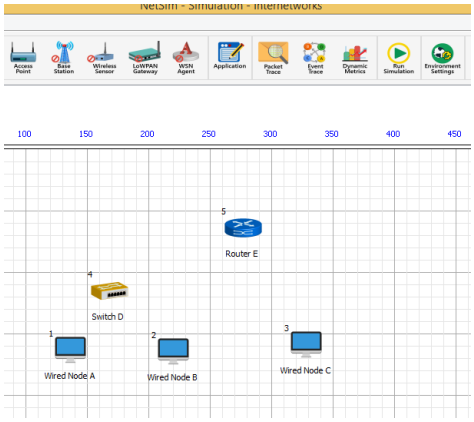


Step 1: Drop the devices

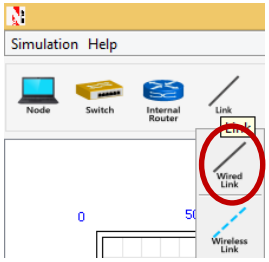
Click on Node icon and select → Wired Node

Click on the environment (the grid in the center) 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.

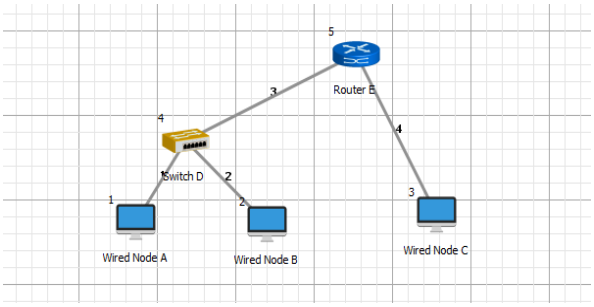


Step 2: Connecting devices on the environment



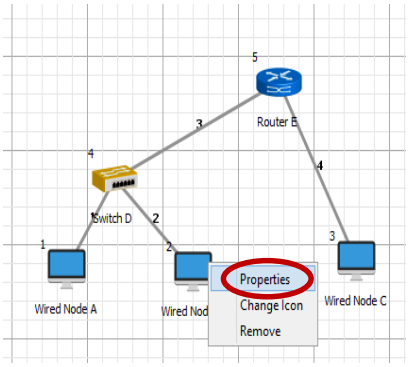
In order to connect devices present in the environment, click on Link and select Wired Link.

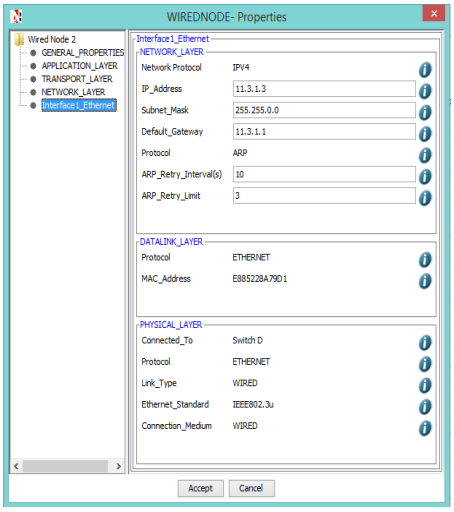
Click and select the devices successively where link is required. For example, select wired link and select Switch and Router successively to connect them. In this manner, continue to link all devices.



2.4.2 Configuring devices and links in the scenario

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

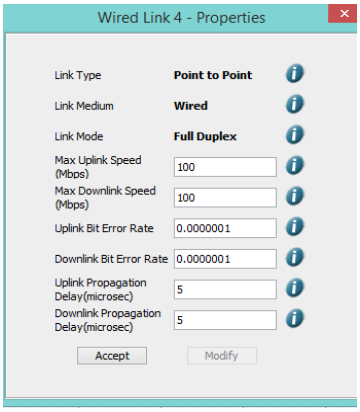




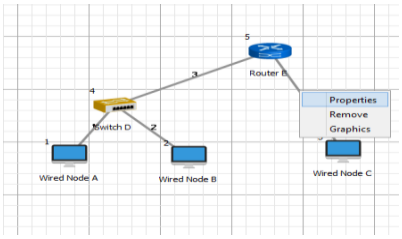
User can set values according to requirement. Modify the properties of any device and click on Accept.

In above scenario, default values already present in the properties are accepted.

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



User can set values according to requirement.

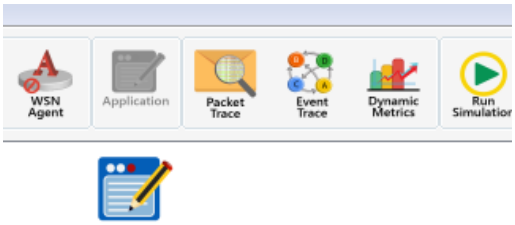


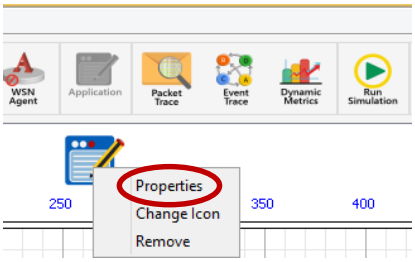
In above scenario, default values already present in the properties are accepted.

2.4.3 Modeling Traffic

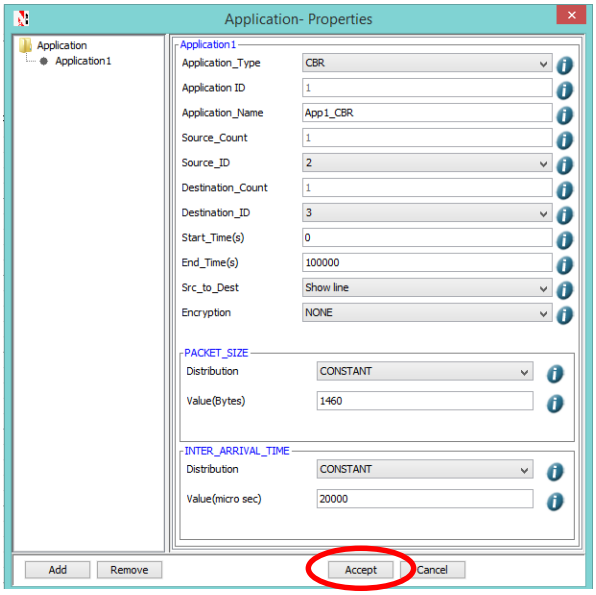
After the network is configured, user needs to model traffic from Wired Node B to Wired Node C.

Select the Application Button and click on the gap between the Grid Environment and the ribbon. Now right click on **Application** and select **Properties.**





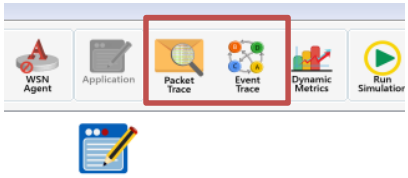
In above scenario, default values already present in the properties are accepted. The Source_ID is 2 and Destination_ID is 5. Click on Accept.



2.4.4 Logging Packet/ Event Trace

If the user wants to analyze the simulation result using Packet and Event Trace, then they need to be enabled before simulation.

Click Packet Trace / Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer section 6.5 and 6.6 respectively.

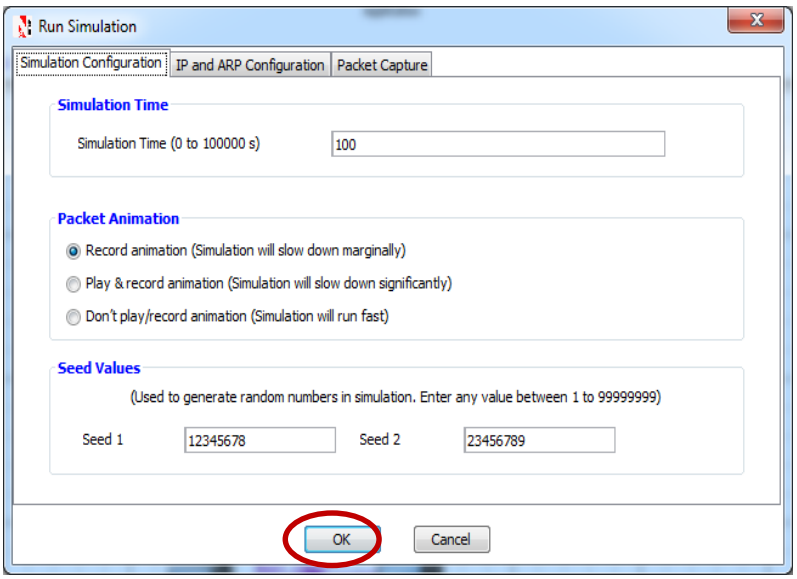


2.4.5 Simulation

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



Set the Simulation Time to 10 seconds. Select OK.

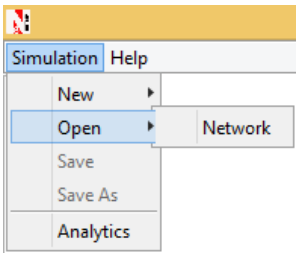


2.5 Network – Opening, Saving scenarios and Printing results

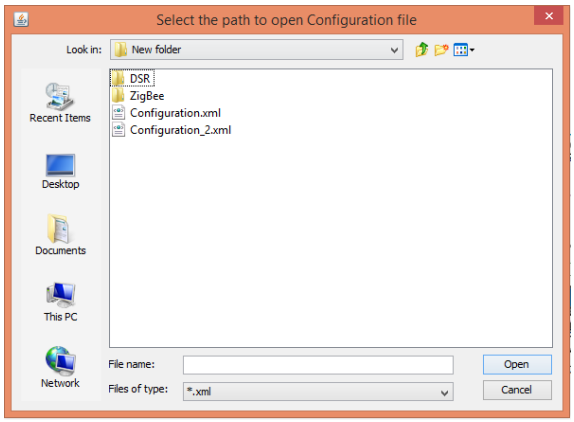
2.5.1 Opening Saved Experiments

2.5.1.1 Open Network – All Networks

Go to **Simulation → Open → Network** menu to open saved experiments. The following steps need to be followed:



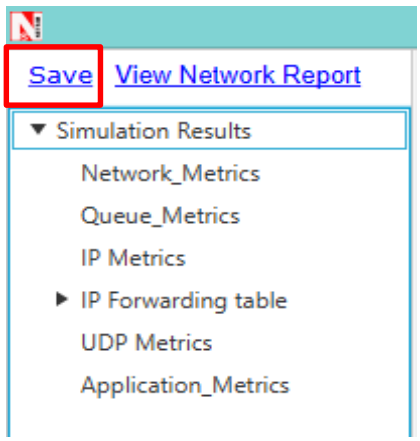
Open saved experiment folder and select the configuration file you want to open.



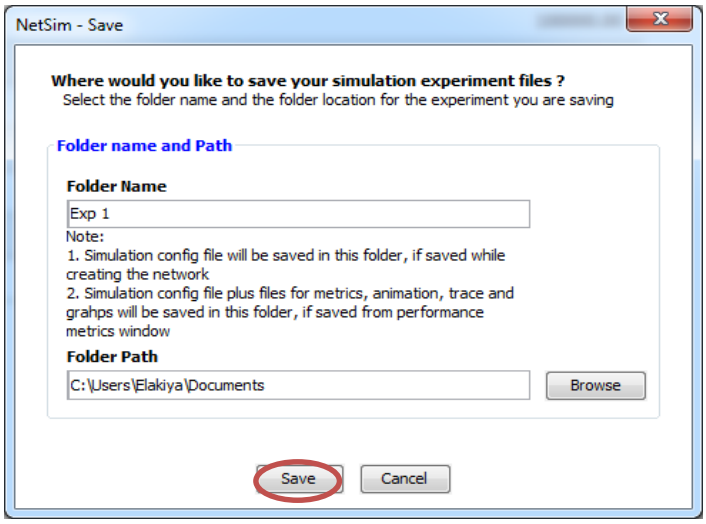
2.5.2 Saving an Experiment

2.5.2.1 For all Networks

Step 1: After simulation of the network, on the top left corner of Performance metrics screen, click on the "Save Network and Metric as" button



Step 2: Specify the **Experiment Name** and **Save Path** and click on **Save.**


NetSim - Save
Where would you like to save your simulation experiment files ?
Select the folder name and the folder location for the experiment you are saving
Folder name and Path
Folder Name
Exp 1
Note:
1. Simulation config file will be saved in this folder, if saved while creating the network
2. Simulation config file plus files for metrics, animation, trace and grahps will be saved in this folder, if saved from performance metrics window
Folder Path
C:\Users\Elakiya\Documents
Browse
Save
Cancel

3 Simulating different networks in NetSim

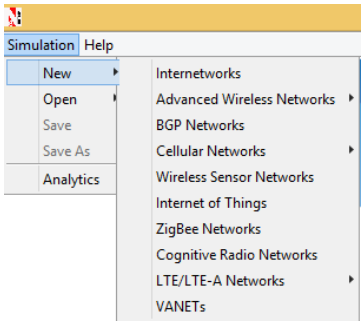
3.1 Internetworks

Internetwork simulation is available in NetSim with LAN-WAN-LAN modeling capability. Internetwork runs Ethernet, Wireless LAN, IP Routing and TCP / UDP and allows users to log packet and event (in NetSim Standard/Pro version only) traces.

3.1.1 New Experiment

In the **Simulation** menu,

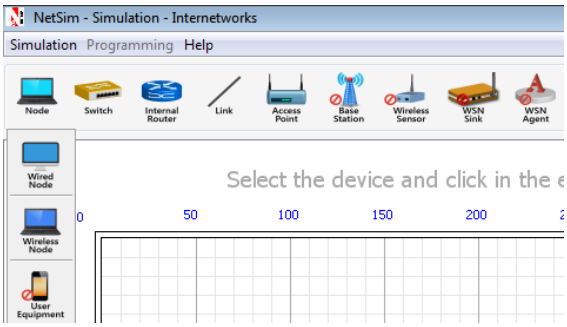
Select → **New** → **Internetworks**



3.1.2 Create Scenario

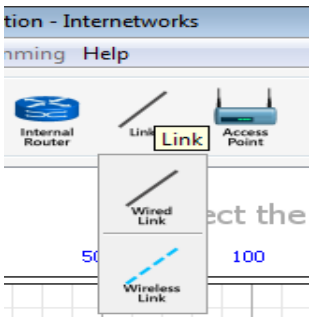
Internetworks come with the palette of various devices like Switch, Router, Wired Node, Wireless Node, AP, etc.

Select the desired devices application icon in the toolbar and click and drop on the environment.



To remove devices or application, right click on the particular icon and then click Remove.

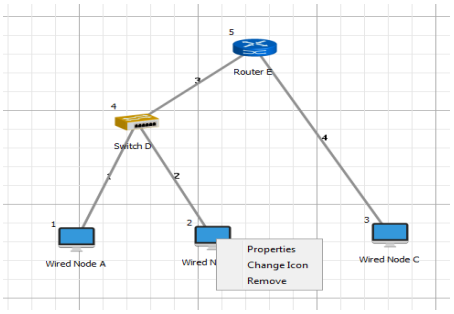
Select the appropriate link in the toolbar and connect the devices by clicking on the device 1 and device 2.



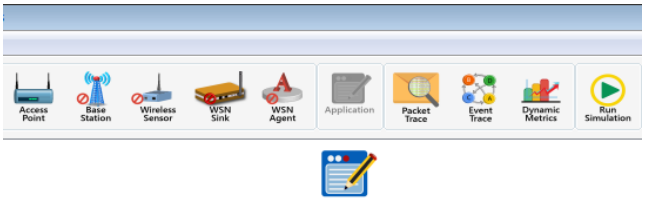
3.1.3 Set Node, Link and Application Properties

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements. Routing Protocol in Application Layer of router and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global except for IEEE802.11e i.e. changing properties in one node will automatically reflect in the others in that network..

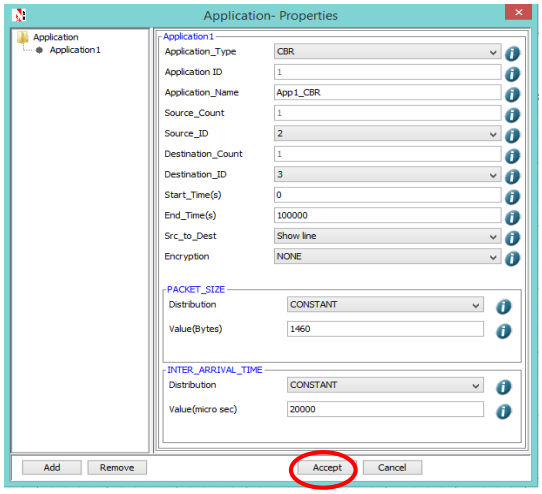
  Note: Mobility of Wireless nodes is not available in infrastructure mode and is only available in Adhoc mode. Hence mobility for wireless nodes can only be set when running MANET simulation.



- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.

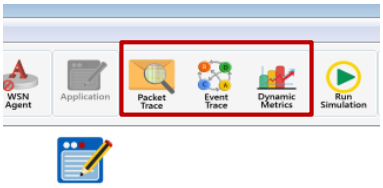


- Set the values according to requirement and click Accept.



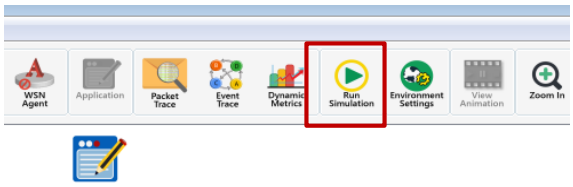
3.1.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.1.5 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

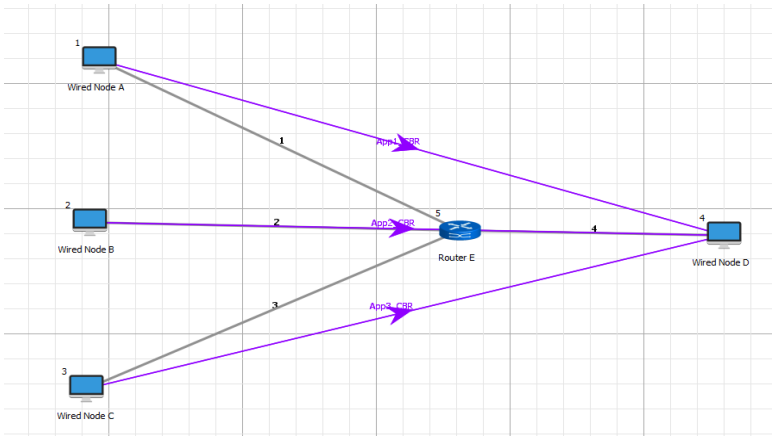
3.1.6 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

Internetworks – Example Simulations:

Example 1: Generation rate and Service Rate

Open the scenario for service rate which is available in "<NetSim Install Dir>\Docs\ Sample_Configuration\Internetworks\Generation_rate_service_rate"



Settings done in example config file:

1. TCP (Transport Layer) -> Disable in all wired nodes
2. Generation rate = 10Mbps for each application.

Formula to calculate Generation Rate:-

*Generation Rate = (Packet size (bytes) * 8) / Inter arrival time (µs)*

3. The traffic generation rate can be modified by changing application properties. Note that the generation rate should be less than or equal to service rate for steady-state simulation, where the service rate is defined as the data rate supported by the Bottle-neck link. In this case there is no bottle neck link since all link support upto 100 Mbps.
4. Simulate and note the throughput.
5. Click on edit network and change the link speed from Router E to Wired Node D from the default 100 Mbps to 25 Mbps. In this case, the link between Router and Wired Node D becomes the Bottle-neck link since the link rate is less than the generation rate of 30 Mbps (10 * 3).

Inference:

Sample 1:

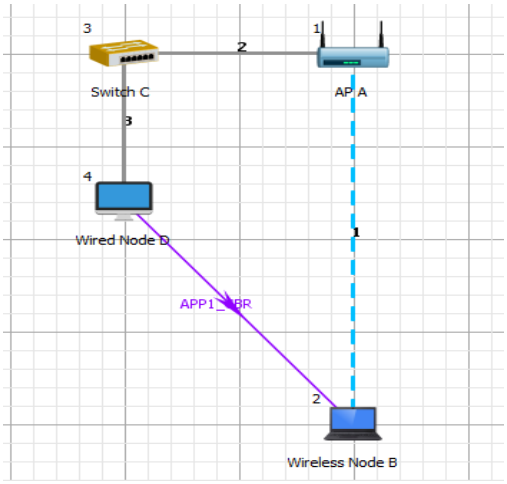
In this scenario router receives packets from three links at the rate of 10mbps each, a total of 30 mbps. And the router-node link supports 100mbps. Hence there is no queuing / drops at the Router. The application throughput would be approximately equal to the generation rate.

Sample 2:

In this case, the bottleneck link supports only 25 mbps. Due to this packets get accumulated in the router's buffer, which overflows after reaching its limit. After this router starts dropping the packets which can be observed in the Queue metrics at the end of the simulation. The application throughput would be approximately equal to the bottle neck link capacity.

Example 2: Packet Aggregation in 802_11n

Open the scenario for packet aggregation which is available in "<NetSim Install Dir>\Docs\ Sample_Configuration\Internetworks\802_11n_packet_aggregation"



Settings done in example config file:

1. Grid length = 50m * 50m.
2. Distance between AP and Node is 20m.
3. Packet Trace -> Enable
4. TCP -> Disable in Wireless and Wired Node.
5. WLAN Standard -> 802.11n.
6. For sample 1, No. of packets aggregated -> 1.
7. Propagation model -> Path Loss only, Path Loss Model -> Log Distance and Path loss Exponent -> 3.
8. CBR application with 100Mbps generation rate.
9. Simulate for 10 sec.
10. Note down the application throughput in the results window after simulation. In further increase the No. of aggregated packets (changing AP -> Interface_Wireless ->

No_of_Packet_to_Aggregate) and note down the application throughputs. It would be as shown in below table:-

Number of Packets Aggregated	Application Throughput
1	23.8 Mbps
5	44.5 Mbps
10	53.9 Mbps

Inference:

- Packet aggregation is responsible for joining multiple MSDUs into a single MPDU that can be delivered to the physical layer as a single unit for transmission. As we increase the number of packet aggregated increase which results in lesser number of ack's so. Hence more data packets are transmitted per unit time leading to higher application throughput.
- For Number of Packets aggregated -> 5, we will get 5 successive packets followed by a WLAN_Block_Ack (which is used to acknowledge that 5 packets are received successfully. Users can observe this in Packet Trace by filtering Tx_ID as Access Point and Rx_ID as Wireless Node B.
- Note that in the early stages of the simulation the AP would transmit whatever the number of frames/packets in its buffer. It will not wait for 5 packets to be aggregated (if number of packets aggregated -> 5). If Access Point buffer has more than 5 packets, it will aggregate 5 packets and then send. After sending 5 Packets it will receive one WLAN_Block_ack.

Example 3: 802.11 MIMO

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \Internetworks\802_11n_MIMO"

Settings done in sample network:

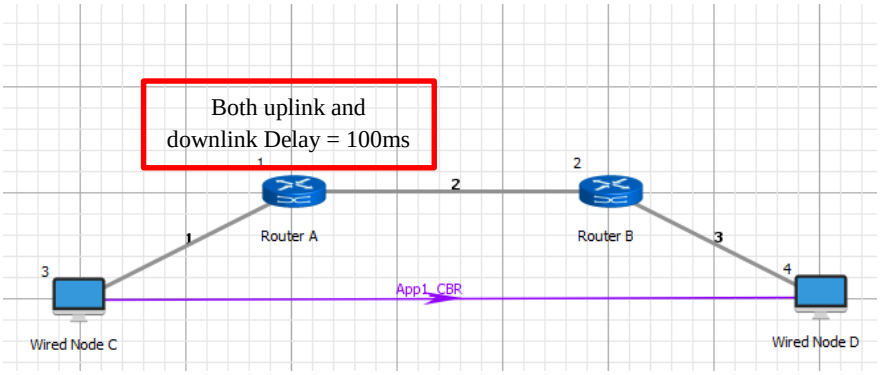
1. Grid length = 50m X 50m.
2. Distance between AP and Node is 20m.
3. Packet Trace -> Enable
4. TCP -> Disable in Wireless and Wired Node.
5. WLAN Standard -> 802.11n.
6. Propagation model -> Path Loss only, Path Loss Model -> Log Distance and Path loss Exponent -> 3.
7. CBR application with 50Mbps generation rate.
8. Simulate for 10 sec.

Number of Tx and Rx Antennae	Throughput
1 x 1	23.8 Mbps
2 x 2	30.85 Mbps
3 x 3	33.13 Mbps
4 x 4	35.66 Mbps

MIMO is a method for multiplying the capacity of a radio link using multiple transmit and receive antennas to exploit multipath propagation. Increasing the Transmitter and Receiver Antenna count results in more Data rate (link capacity) and hence leading to an increased application throughput.

3.1.7 TCP – Example Simulations

Example 1: Window Scaling



Open the scenario which is available in "<NetSim Install Dir>\Docs\ Sample_Configuration \internetworks\window_scaling"

The throughput of a communication is limited by two windows: the congestion window and the receive window. The congestion window tries not to exceed the capacity of the network (congestion control); the receive window tries not to exceed the capacity of the receiver to process data (flow control).

The **TCP window scale option** is an option to increase the receive window size allowed in Transmission Control Protocol above its former maximum value of 65,535 bytes.

TCP window scale option is needed for efficient transfer of data when the bandwidth-delay product is greater than 64K. For instance, if a transmission line of 1.5 Mbit/second was used over a satellite link with a 513 millisecond round trip time (RTT), the bandwidth-delay product is (1,500,000 * 0.513) = 769,500 bits or about 96,187 bytes.

Using a maximum window size of 64 KB only allows the buffer to be filled to (65,535 / 96,187) = 68% of the theoretical maximum speed of 1.5 Mbits/second, or 1.02 Mbit/s.

By using the window scale option, the receive window size may be increased up to a maximum value of 1,073,725,440 bytes. This is done by specifying a one byte shift count in the header options field. The true receive window size is left shifted by the value in shift count. A maximum value of 14 may be used for the shift count value. This would allow a single TCP connection to transfer data over the example satellite link at 1.5 Mbit/second utilizing all of the available bandwidth.

Settings done in example config file:

1. TCP -> Enable
2. Window Scaling -> FALSE
3. Application Generation rate -> 10Mbps (packet Size = 1460 Bytes, Inter arrival time = 1168μs)
4. Bit error rate -> 0 in all wired links
5. Wireshark -> offline (Node3)
6. Link1 & Link3 speed -> 100Mbps, delay ->5μs
7. Link2 speed -> 10Mbps, delay -> 100ms (i.e., both uplink and downlink delay = 100ms -> Round trip time = 200ms)
8. Simulate for 100s and note down the throughput
9. Window Scaling -> TRUE
10. Simulate for 100s and note down the throughput

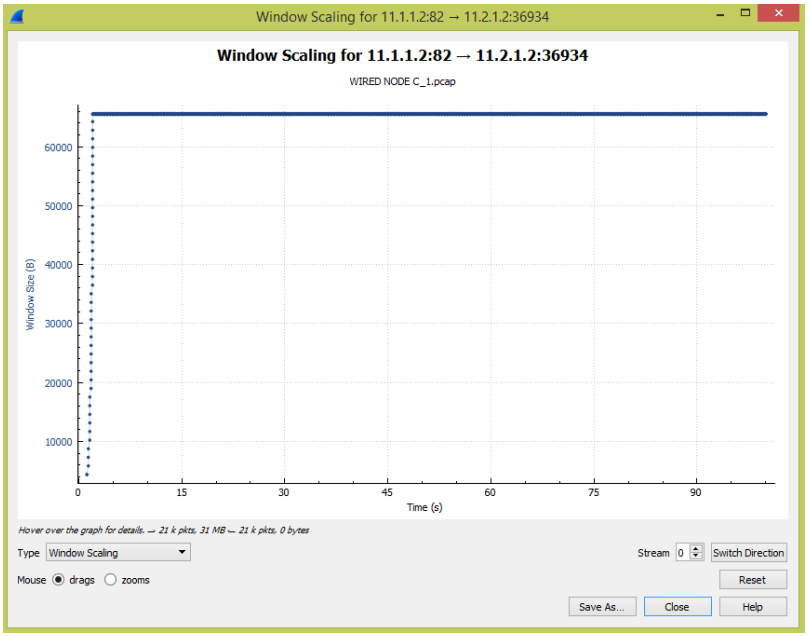
Output:

Window Scaling	Application_Throughput (Mbps)
FALSE	2.50
TRUE	8.60

Throughput calculation: (without Window Scaling)

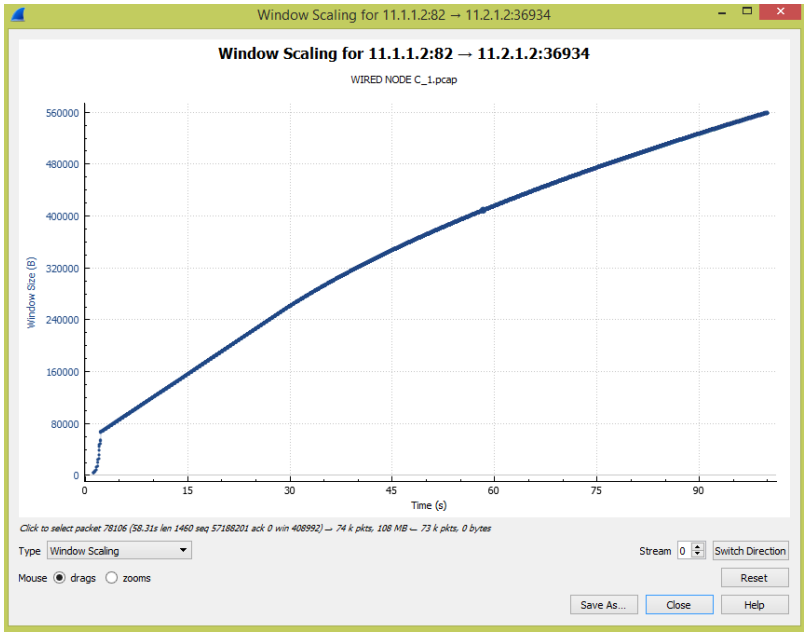
Thoeretical Throughput = Window size / Round trip time = 65535*8/200ms = 2.62Mbps

Inference (Without Window scaling):



In case 1 the Application_Throughput is 2.50 Mbps less than the theoretical throughput since it initially takes some time for the window to reach 65535 B Users can notice the Window size in wireshark. Please refer section 6.7.4 in User Manual for creating wireshark graphs

With Window scaling:



From the above screenshot users can notice that the window size grows upto 560192 Bytes (which is > 65535Bytes) because of Window Scaling. This leads to a higher Application_Throughput compared to the case without window scaling.

3.1.8 IP Addressing in NetSim

When you create a network using the GUI, NetSim will automatically configure the IP address of the devices in the scenario.

Consider the following scenarios:
If you create a network with two wired nodes and a switch, the IP addresses are assigned as 10.0.1.2 and 10.0.1.3 for the two wired nodes. The default subnet mask is assigned to be 255.255.0.0. It can be edited to 255.0.0.0 (Class A) or 255.255.255.0 (Class C) subnet masks. Both the nodes are in the same network (10.0.0.0).

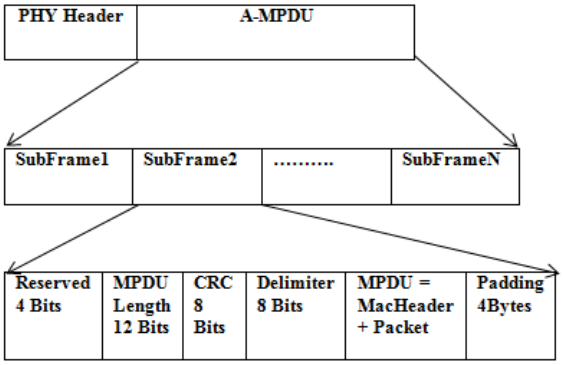
Similarly, if you create a network with a router and two wired nodes, the IP addressed are assigned as 11.1.1.2 and 11.2.1.2 for the two wired nodes. The subnet mask is default as in above case, i.e., 255.255.0.0. The IP address of the router is 11.1.1.1 and 11.2.1.1 respectively for the two interfaces. Both the nodes are in different networks (11.1.0.0 and 11.2.0.0) in this case.
The same logic is extended as the number of devices is increased.

3.1.9 Features in WLAN 802.11n/ac

3.1.9.1 IEEE 802.11n Standard

Mac aggregation and block acknowledgement are two important enhancements to 802.11n standard. In the aggregation scheme, several MPDU's (MAC Protocol Data Units) are aggregated in to a single A-MPDU (Aggregated MPDU).

The A-MPDU's are created before sending to PHY layer for transmission. The MAC does not wait for MPDU's before aggregation. It aggregates the already present packets in the queue to form an A-MPDU. The maximum size of A-MPDU is 65535 bytes. The maximum size of each MPDU is 4KB. In A-MPDU, each MPDU has a delimiter of 32bits at the beginning and padding at the end. These padding bytes ensure that size of MPDU is a multiple of 4bytes.



In 802.11n, a single block acknowledgement is sent for the entire A-MPDU. The block ack acknowledges each packet that is received. It consists of a bitmap (compressed bitmap) of 64bits or 8 bytes. This bitmap can acknowledge upto 64 packets, 1bit for each packet.

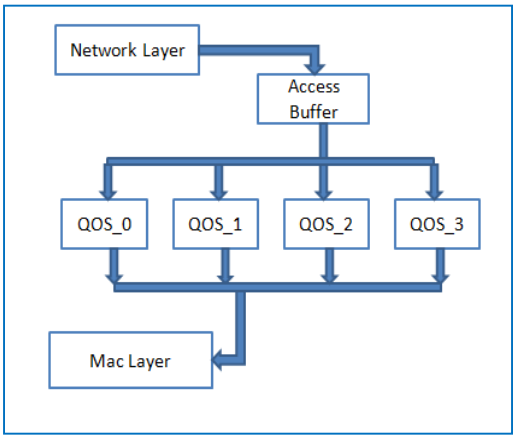
The value of a bitmap field is 1 if respective packet is received without error else it is 0. Only the error packets are resent until a retry limit is reached. The number of packets in an A-MPDU is restricted to 64 since the size of block ack bitmap is 64bits.

Octets:2	2	6	6	2	2	8	4
Frame Control	Duration /ID	RA	TA	BARControl	BA Starting Sequence Control	BitMap	FCS

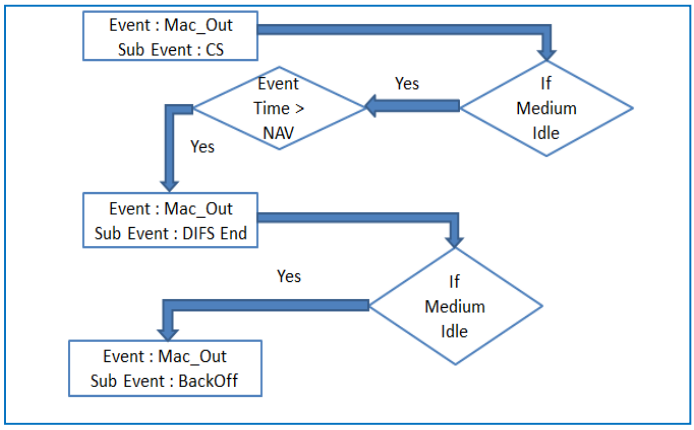
Block Ack Control Packet

3.1.9.2 Details of 802.11 n implementation in NetSim –

- NetSim aggregates packets in terms of numbers and not size.
- A user can vary the number of packets to aggregate by changing the appropriate parameters in the GUI.
- NetSim ignores the padding bytes added to the MPDU since its effect is negligible.
- NetSim aggregates packets to the same receiver id and not to the destination ID.



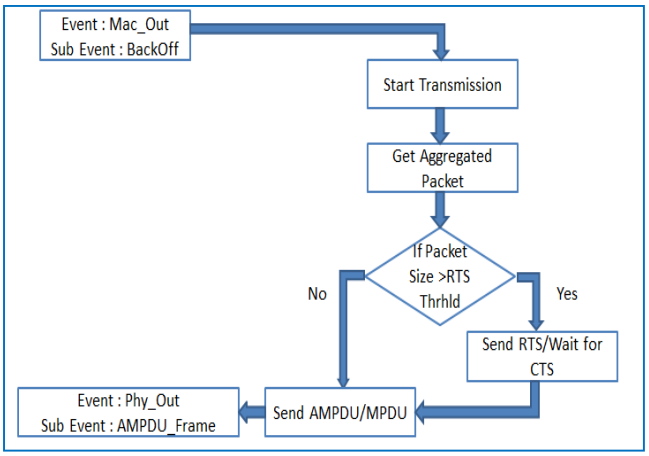
Packets arriving from the NETWORK Layer gets queued up in an access buffer from which they are sorted according to their priority in the respective QOS buffer according to the IEEE 802.11e standard. An event MAC_OUT with SubEvent CS (Carrier Sense – CSMA) is added to check if the medium is free



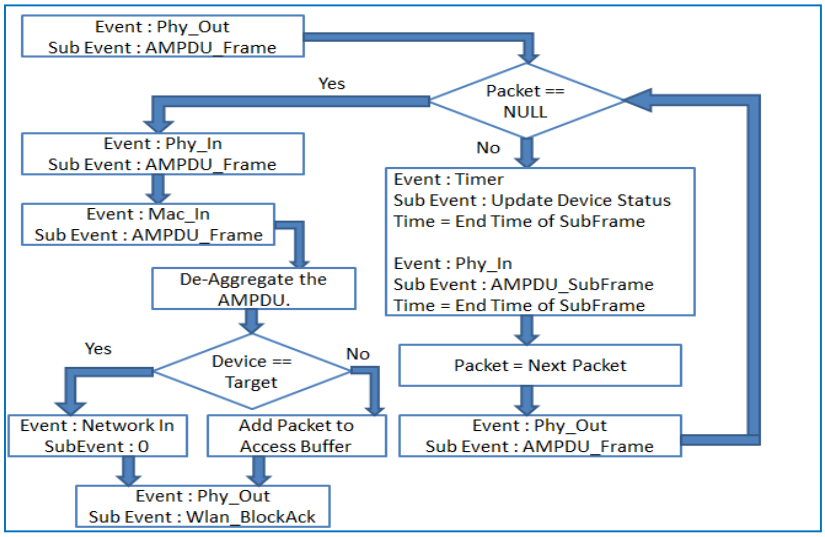
In CS, if the medium is free, then the NAV is checked. This is enabled if RTS/CTS mechanism is enabled which can be done so by adjusting the RTS Threshold. If the Present_Time>NAV, then an Event MAC_OUT with SubEvent DIFS End is added at the time Present_Time + DIFS time.

The medium is checked at the end of DIFS time period and a random time BackOff is calculated based on the Contention Window (CW). An Event MAC_OUT with SubEvent Backoff is added at time Present_Time + BackOff Time.

Once Backoff is successful, NetSim starts the transmission process wherein it gets the aggregated packet from the QOS buffer and stores it in the Retransmit buffer. If the A-MPDU size is > RTS Threshold, then it enables RTS/CTS mechanism which is an optional feature.



NetSim sends the packet by calling the PHY_OUT Event with SubEvent AMPDU_Frame. Note that the implementation of A-MPDU is in the form of a linked list.



Whenever a packet is transmitted, the medium is made busy and a Timer Event with SubEvent Update Device Status is added at the transmission end time to set the medium again as idle.

Events PHY_OUT SubEvent AMPDU_SubFrame, Timer Event SubEvent Update Device Status and Event PHY_IN SubEvent AMPDU_SubFrame are added in succession for each MPDU (Subframe of the aggregated packet). This is done for collision calculations. If two stations start transmission simultaneously, then some of the SubFrames may collide. Only those collided SubFrames will be retransmitted again. The same logic is followed for an Errored packet. However, if the PHY header (the first packet) is errored or collided, the entire A-MPDU is resent.

At the receiver, the device de-aggregates the packet in the MAC Layer and generates a block ACK which is sent to the transmitter. If the receiver is an intermediate node, the de-aggregated packets are added to the access buffer of the receiver in addition to the packets which arrive from Network layer. If the receiver is the destination, then the received packets are sent to the Network layer. At the transmitter side, when the device receives the block acknowledgement, it retransmits only those packets which are errored. The rest of the packets are deleted from the retransmit buffer. This is done till all packets are transmitted successfully or a retransmit limit is reached after which next set of packets are aggregated to be sent.

3.1.9.3 802.11ac MAC and PHY Layer Implementation

Improvements in 802.11ac compared to 802.11n

Feature	802.11n	802.11ac
Spatial Streams	Up to 4 streams	Up to 8 streams
MIMO	Single User MIMO	Multi-User MIMO
Channel Bandwidth	20 and 40 MHz	20, 40, 80 and 160 MHz (optional)
Modulation	BPSK, QPSK, 16QAM and 64QAM	BPSK, QPSK, 16QAM, 64QAM and 256QAM (optional)
Max Aggregated Packet Size	65536 octets	1048576 octets

MAC layer improvements include only the increment of number of aggregated packets from 1 to 64. The MCS index for different modulation and coding rates are as follows:-

MCS Index	Modulation	Code Rate
0	BPSK	1/2
1	QPSK	1/2
2	QPSK	3/4
3	16QAM	1/2
4	16QAM	3/4
5	64QAM	2/3
6	64QAM	3/4
7	64QAM	5/6
8	256QAM	3/4
9	256QAM	5/6

Receiver sensitivity for different modulation schemes in 802.11ac (for a 20MHz Channel bandwidth) are as follows:

MCS Index	Receiver Sensitivity (in dBm)
0	-82
1	-79
2	-77
3	-74
4	-70
5	-66
6	-65
7	-64
8	-59
9	-57

Number of subcarriers for different channel bandwidths

PHY Standard	Subcarriers	Capacity relative to 20MHz in 802.11ac
802.11n/802.11ac 20MHz	Total 56, 52 Usable (4 pilot)	x1.0
802.11n/802.11ac 40MHz	Total 114, 108 Usable (6 pilot)	x2.1
802.11n/802.11ac 80MHz	Total 242, 234 Usable (8 pilot)	x4.5
802.11n/802.11ac 160MHz	Total 484, 468 Usable (16 pilot)	x9.0

Now with the knowledge of MCS index and bandwidth of the channel data rate is set in the following manner

Step1: Get the number subcarriers that are usable for the given bandwidth of the medium.

Step2: Get the Number of Bits per Sub Carrier (NBPSC) from selected MCS

Step3: Number of Coded Bits Per Symbol (NCBPS) = NBPSC*Number of Subcarriers

Step4: Number of Data Bits Per Symbol (NDBPS) = NCBPS*Coding Rate

Step5: Physical level Data Rate = NDBPS/Symbol Time (4micro sec for long GI and 3.6 micro sec for short GI)

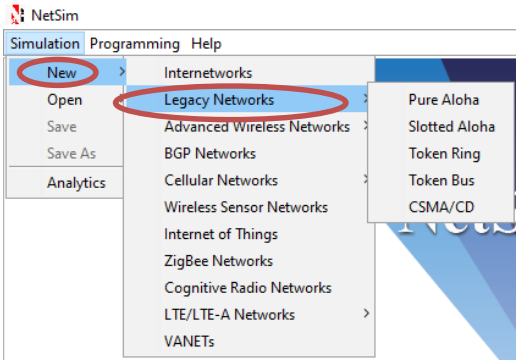
3.2 Legacy Networks

3.2.1 New Experiment

In the **Simulation** menu select → **New** → **Legacy Networks**

For example, to arrive **Pure Aloha**,

In the **Simulation** menu select → **New** → **Legacy Networks** → **Pure Aloha**.



3.2.2 Create Scenario

Adding Node:

- **Click** on the **Node** icon in the tool bar and click and drop inside the grid. (*Note*: This is applicable for Pure Aloha and Slotted Aloha)
- Nodes cannot be connected directly to each other because an intermediate connecting component (such as Hub or Concentrator) is required. (Note: This is applicable for CSMA/CD, Token Bus and Token Ring)

Adding Hub:

- **Click** on the **Hub** icon in the tool bar and click it onto the environment. By default a Hub has 24 ports. (*Note*: This is applicable for CSMA/CD and Token Bus).

Adding Concentrator:

- **Click** on the **Concentrator** icon in the tool bar and click it onto the environment. By default a Concentrator consists of 24 ports. (*Note*: This is applicable for Token Ring).

3.2.3 Set Node, Link and Application Properties

Set Node Properties

Right Click on the appropriate node and select Properties.

Set the Properties for the devices and links

Right click over the devices and then select Properties to set the properties of the links and the devices.

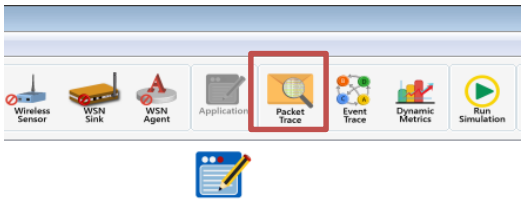
3.2.4 Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

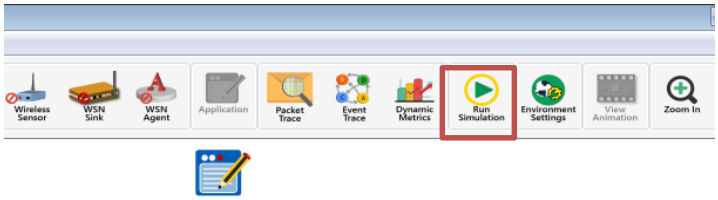
3.2.5 Enable Packet Trace (Optional)

Click Packet Trace icon in the tool bar. To get detailed help, please refer section 6.5respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.2.6 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

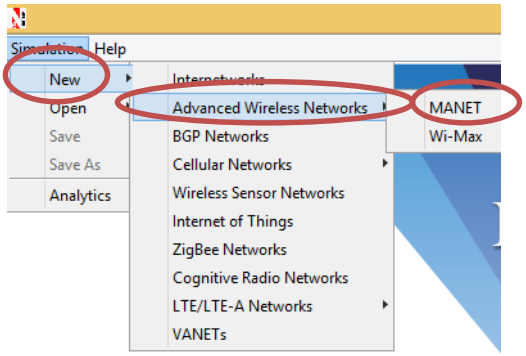
3.3 Advanced wireless networks – MANET & Wi-Max

3.3.1 New Experiment

In the **Simulation** menu select → **New** → **Advanced Wireless Networks**

For example, to arrive **MANET**,

In the **Simulation** menu select → **New** → **Advanced Wireless Networks** → **MANET**



3.3.2 Create Scenario

Adding Wireless Node (*Note*: This is applicable for MANET)



- **Click** on the **Node** icon in the tool bar, select **Wireless Node** and click and drop it inside the **grid**. One must be aware that TCP is disabled by default. (Note: A Node cannot be placed on another Node. A Node cannot float outside of the grid.)

Adding Base Station and Subscriber (*Note*: This is applicable for Wi-MAX)

- **Click** on the **Base Station** icon in the tool bar and **click** it onto the environment.
- **Click** on the **Wi-Max Subscriber** icon after clicking **Node** icon in the tool bar. **Click** and **drop** it onto the environment.

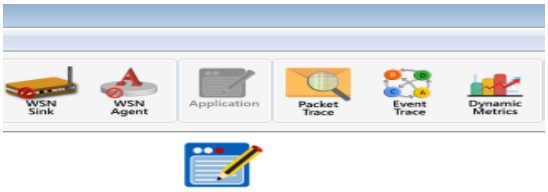
3.3.3 Set Node, Link and Application Properties

For MANET and Wi-Max

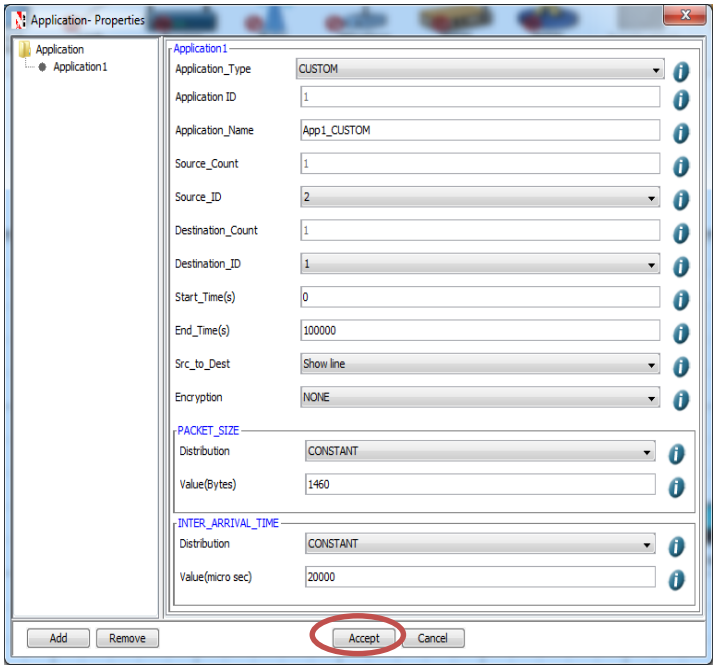
- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements.
    - In case of Wi-Max, Routing Protocol in Application Layer of router and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global i.e. changing properties in one node will automatically reflect in the others in that network.

- In case of MANET, in Wireless Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer, Physical Layer and Power are Global i.e. changing properties in one node will automatically reflect in the others in that network.

- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.



- Set the values according to requirement and click Accept.

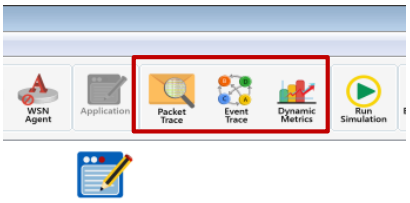


3.3.4 Modifying/Viewing/Accepting Properties

On opening an already configured properties of environment, the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

3.3.5 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



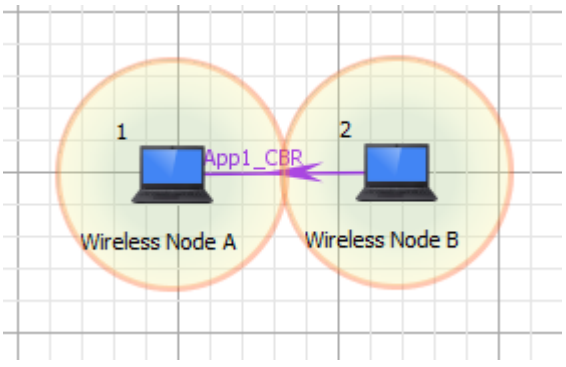
3.3.6 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

MANET – Example Simulations:

Example 1: Node Range

Range of a wireless node depends on Transmitter power, RF propagation losses, Distance, Receiver sensitivity, Frequency etc. Radio range option is available in NetSim. To enable this option click on Environment settings, then click on radio range utility tab, then select ON and then click on OK.



Users can see the variation in range of a node by changing the following parameters.

Transmitter Power:

Increase in transmitter power increases the received power when all other parameters are constant. Increased received power leads to higher SNR and hence higher Phy Data rates and lesser error.

Open the scenario which is available in "<NetSim Install Dir>\Docs\Sample_Configuration \MANET\Transmitter_power"

Settings done in example config file:

1. TCP (Transport Layer) -> Disable in all wireless nodes
2. CBR applications with default generation rates
3. Distance between 2 wireless nodes = 150m
4. Channel Characteristics -> Pathloss only, Pathloss model -> LOG DISTANCE, Pathloss exponent -> 4
5. Transmitter power = 10mW
6. Simulate for 10 seconds and note down the Application_throughput
7. Similarly do the experiment by changing Transmitter power as per the following

Sample	Transmitter Power (mW)	Application_Throughput (Mbps)
1	10	0
2	20	0.2791
3	30	0.5828
4	40	0.5828
5	50	0.5828

RF propagation losses:

Path Loss or attenuation of RF signals occurs naturally with distance. Losses can be increased by increasing the path loss exponent (η). This option is available in channel characteristics. Users can compare the results by changing the path loss exponent (η) value.

Distance:

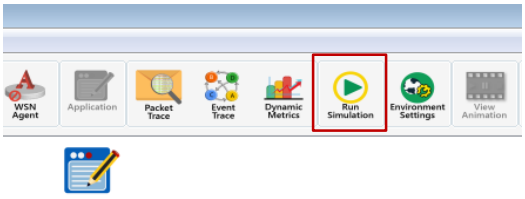
As the distance between two devices increases the received signal power reduces as propagation loss increases with distance. As the received power reduces, the underlying Phy rate the channel can handle also drops.

Receiver sensitivity:

Receiver sensitivity is the lowest power level at which the receiver can detect an RF signal and demodulate data. Improving the sensitivity on the receiver (making it more negative) will allow the radio to detect weaker signals, and can increase the transmission range.

3.3.7 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

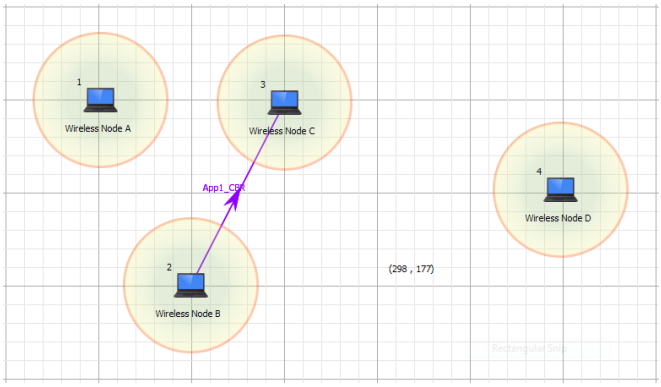
Note on MANET implementation in NetSim:

- If user wants to implement HTTP application among Nodes, TCP must be enabled in Source Node as TCP is set to disable by default.
- OLSR is a proactive link-state routing protocol, which uses hello and topology control (TC) messages to discover and then disseminate link state information throughout the mobile ad hoc network.
- Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths. For topology control (TC) messages to disseminate throughout, it requires 5 or more seconds depending upon the network size. In general, it is (5.5 secs + Tx_Time * network size). Hence an application must start at least 6-7 seconds after the network commences and it can be done by setting the "Starting time" parameter in Application.

What does the radio range of a MANET node represent?

The standard defines the range of a node is the distance at which the PER (packet error rate) of a 1024B packet is 10 %

This means that nodes even beyond the range will be able to communicate, but they may do so with high error.



The factors which affect range are:

a) Transmitter power (More Tx power implies higher range)

b) Path loss (Higher path loss exponent leads to lower range)

c) Receiver sensitivity (Lower Rx sensitivity leads to higher range. Rx sensitivity is in negative dB, hence lower means more negative).

3.3.8 Link Layer Acknowledgements and Network Layer Acknowledgements in DSR

Route Maintenance is the mechanism by which a source node S is able to detect, while using a source route to some destination node D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works.

Using Link-Layer Acknowledgements

If the MAC protocol in use provides feedback as to the successful delivery of a data packet (such as is provided for unicast packets by the link-layer acknowledgement frame defined by IEEE 802.11), then the use of the DSR Acknowledgement Request and Acknowledgement options is not necessary. If such link-layer feedback is available, it SHOULD be used instead of any other acknowledgement mechanism for Route Maintenance, and the node SHOULD NOT use either passive acknowledgements or network-layer acknowledgements for Route Maintenance.

When using link-layer acknowledgements for Route Maintenance, the retransmission timing and the timing at which retransmission attempts are scheduled are generally controlled by the particular link layer implementation in use in the network. For example, in IEEE 802.11, the link-layer acknowledgement is returned after a unicast packet as a part of the basic access method of the IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol; the time at which the acknowledgement is expected to arrive and the time at which the next retransmission attempt (if necessary) will occur are controlled by the MAC protocol implementation.

Using Network-Layer Acknowledgements

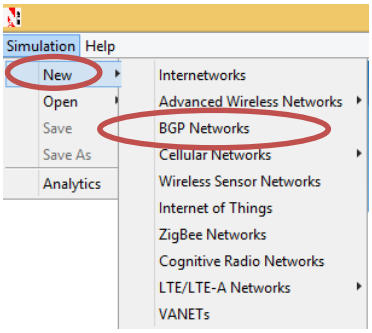
When a node originates or forwards a packet and has no other mechanism of acknowledgement available to determine reachability of the next-hop node in the source route for Route Maintenance, that node SHOULD request a network-layer acknowledgement from that next- hop node. To do so, the node inserts an Acknowledgement Request option in the DSR Options header in the packet. The Identification field in that Acknowledgement Request option MUST be set to a value unique over all packets recently transmitted by this node to the same next-hop node.

When using network-layer acknowledgements for Route Maintenance, a node SHOULD use an adaptive algorithm in determining the retransmission timeout for each transmission attempt of an acknowledgement request. For example, a node SHOULD maintain a separate round-trip time (RTT) estimate for each node to which it has recently attempted to transmit packets, and it SHOULD use this RTT estimate in setting the timeout for each retransmission attempt for Route Maintenance.

3.4 BGP

3.4.1 New Experiment

In the **Simulation** menu select **Simulation → New → BGP Networks**



3.4.2 Create Scenario

Adding Border Router:

- **Click and drop** the **Border Router** icon from the tool bar. (*Note*: Maximum you can have 3 Autonomous systems in a single scenario.)

Adding Internal Router:

- **Click** on the **Internal Router** icon in the tool bar and drop the Internal Router onto the Autonomous systems created. By default a Router has eight ports.

Establishing Connections

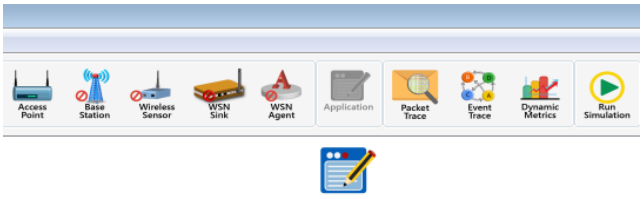
The steps for connecting devices in BGP networks are as follows,

- The connections between two **wired nodes** cannot be made in the network.
- The connection possibilities are
    - **Wired Node** to **Internal Router**
    - **Internal Router** to **Border Router**
    - **Border Router** to **Border Router**

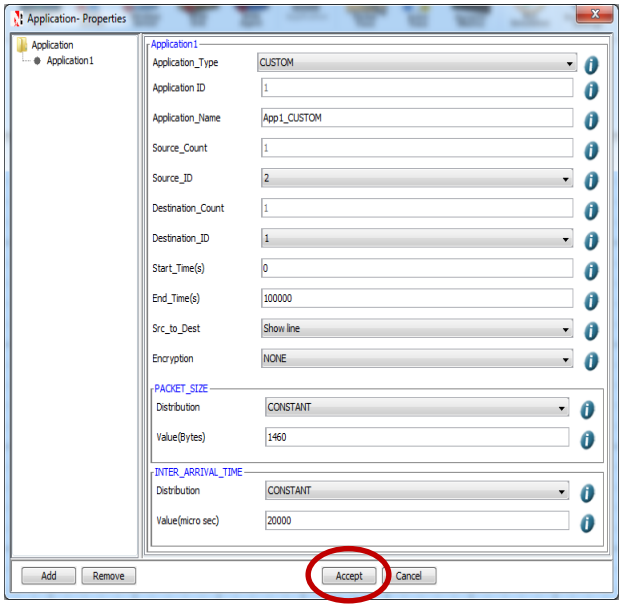
3.4.3 Set Node, Link and Application Properties

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements. Routing Protocol in Application Layer of router and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global i.e. changing properties in one node will automatically reflect in the others in that network.

- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.



- Set the values according to requirement and click Accept.



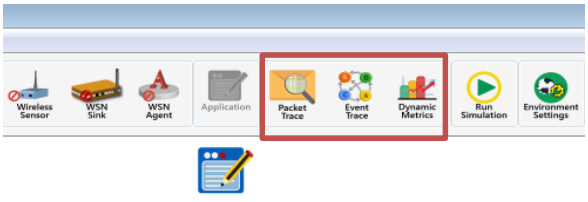
3.4.4 Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

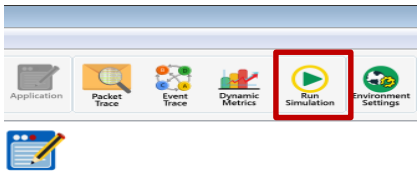
3.4.5 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.4.6 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

3.4.7 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder"<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

BGP– Example Simulations:

BGP is an exterior gateway protocol (EGP), which performs routing between multiple autonomous systems or domains and exchanges routing and reachability information with other BGP systems.

BGP is based on path vector routing. In NetSim, RIP and OSPF are the interior routing protocol for BGP routers that are available.

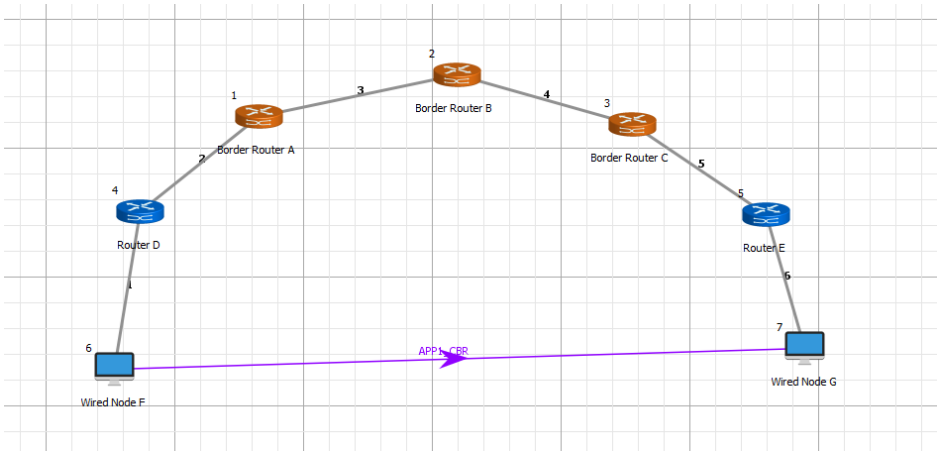
If we change any Interior routing protocol from BGP to OSPF or vice versa it should be change in all the other Routers since it is a global property.

The keep-alive message notifies BGP peers that a device is active. Keep-alives are sent often enough to keep the sessions from expiring.

Example 1: Keep Alive Messages

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \BGP\Keep_Alive_Mesages".

The main mechanism BGP uses to make sure neighbors are still alive is using the hold time and KEEPALIVE messages. The **hold time** specifies how long a router will wait for incoming BGP messages before it assumes the neighbor is dead. When a router has no UPDATE messages to send, it periodically sends KEEPALIVE messages, which, well, keep the BGP session alive.



Settings done in sample network:

1. In the Border Routers, in application layer -> Exterior Routing protocol is by default BGP.
2. Interior routing protocol is by default set as OSPF.
3. Keep alive message timer by default is set as 30 seconds and Hold timer as 90 seconds.
4. Disable TCP in both Wired nodes.
5. Set CBR application and accept default properties.
6. Enable Packet Trace and run simulation for 100 seconds.
7. Open Packet trace and click on the filter option. Filter the control packet type as BGP_KEEPALIVE messages.
8. Total number of keep alive messages sent can be noted down from the packet trace.
9. Save the experiment and run simulation changing the Keep alive timer for 20 and 10 seconds and note down the number of BGP_KEEPALIVE messages sent using the packet trace.

Result:

Keep alive Timer in second	Number of keep alive messages
30	24
20	30
10	60

Inference:

KEEPALIVE messages are sent periodically (every 30 seconds by default) to ensure that the remote peer is still available.

If a router does not receive a KEEPALIVE from a peer for a Hold-time period (by default, **90 seconds**), the router declares that peer dead.

Example 2 –BGP_Open_message:

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \BGP\Keep_Open".

The delay Open timer is optional session attribute is used to delay the sending of an OPEN message on a connection. If the open message is accepted, a Keep alive message confirming the OPEN is send back.

It is nothing but the delay for the open message that will be send by the BGP routers after TCP connection establishment.

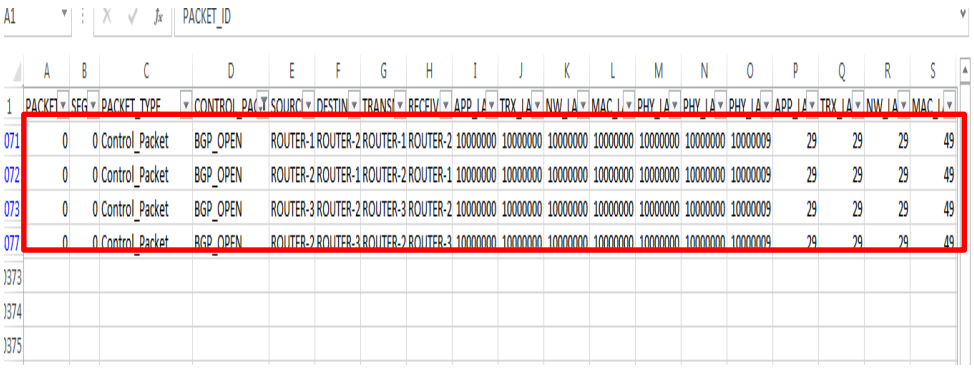
We can analyse the delay by providing the delay value in GUI.

Settings done in sample network:

1. In the Border Routers, in application layer -> Exterior Routing protocol is by default BGP.
2. Interior routing protocol is by default set as OSPF.
3. Set delay open timer to 10 second in all the border routers
4. Disable TCP in both Wired nodes.
5. Set CBR application and accept default properties.
6. Enable Packet Trace and run simulation for 100 seconds.

Result:

Open packet trace and Filter the BGP_OPEN messages.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	PACKET	SEG	PACKET_TYPE	CONTROL_PAC	SOURCI	DESTIN	TRANSI	RECEIV	APP_LA	TRX_LA	NW_LA	MAC_L	PHY_LA	PHY_LA	PHY_LA	APP_LA	TRX_LA	NW_LA	MAC_L
071		0	0 Control_Packet	BGP_OPEN	ROUTER-1	ROUTER-2	ROUTER-1	ROUTER-2	10000000	10000000	10000000	10000000	10000000	10000000	10000009	29	29	29	49
072		0	0 Control_Packet	BGP_OPEN	ROUTER-2	ROUTER-1	ROUTER-2	ROUTER-1	10000000	10000000	10000000	10000000	10000000	10000000	10000009	29	29	29	49
073		0	0 Control_Packet	BGP_OPEN	ROUTER-3	ROUTER-2	ROUTER-3	ROUTER-2	10000000	10000000	10000000	10000000	10000000	10000000	10000009	29	29	29	49
077		0	0 Control_Packet	BGP_OPEN	ROUTER-2	ROUTER-3	ROUTER-2	ROUTER-3	10000000	10000000	10000000	10000000	10000000	10000000	10000009	29	29	29	49
373																			
374																			
375																			

Here users can observe that all the BGP_OPEN messages leaving application layer after 10 second that is the delay provided.

NOTE:

Open delay timer value can't be greater than the simulation time. If you provide this then there won't be any BGP_OPEN messages in the packet trace.

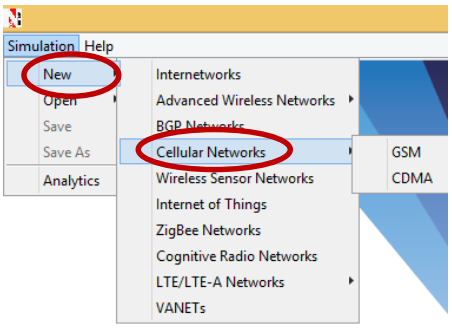
3.5 Cellular Networks – GSM/CDMA

3.5.1 New Experiment

In the **Simulation** menu select → **New** → **Cellular Networks**

For Example, to arrive **CDMA**

In the **Simulation** menu select → **New** → **Cellular Networks** → **CDMA**



3.5.2 Create Scenario

Adding Base Transceiver Station (BTS) - Click on the **BTS** icon in the toolbar and click it onto the **environment**.

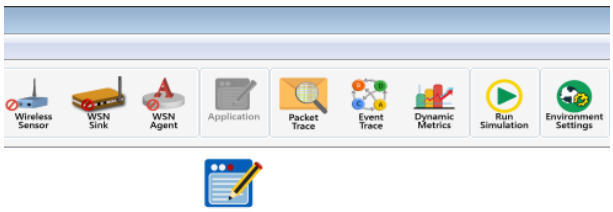
Adding Mobile Switching Centre (MSC) - Click and drop **MSC** in the environment.

Adding Mobile Station (MS) -

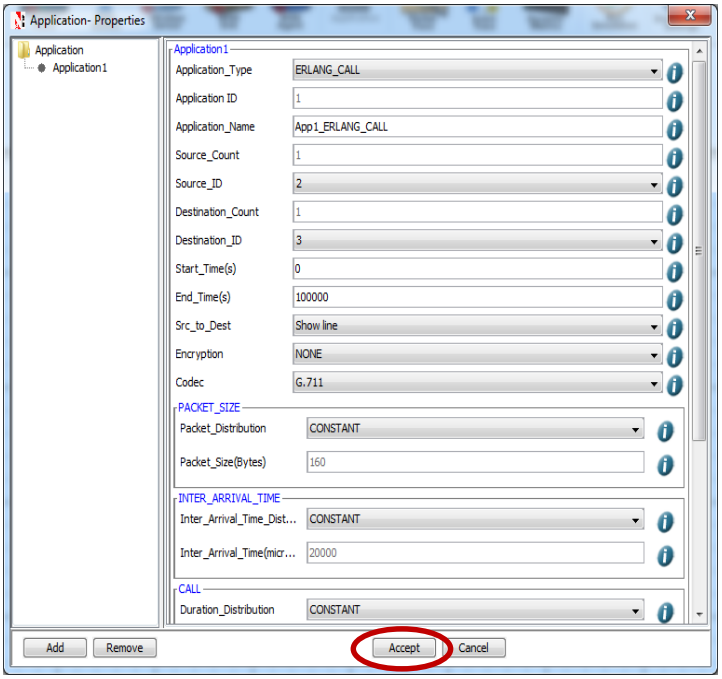
- **Click** on the **Mobile Station** icon in the tool bar, click and drop it on the **Base Station coverage** area.
- **Mobile Station** cannot be placed on another **Mobile Station**. It has to be clicked and placed on the **Base Station** coverage area.

3.5.3 Set Node, Link and Application Properties

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements.

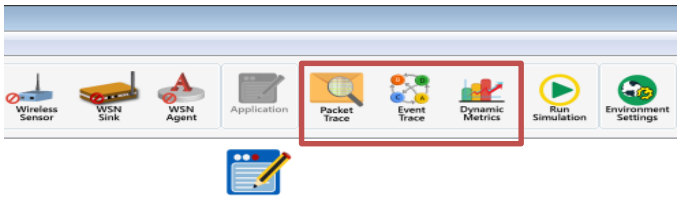


- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.
- Set the values according to requirement and click Accept.



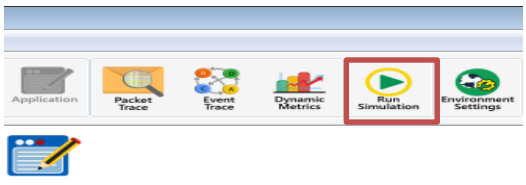
3.5.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.5.5 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

3.5.6 Example Configuration files in NetSim

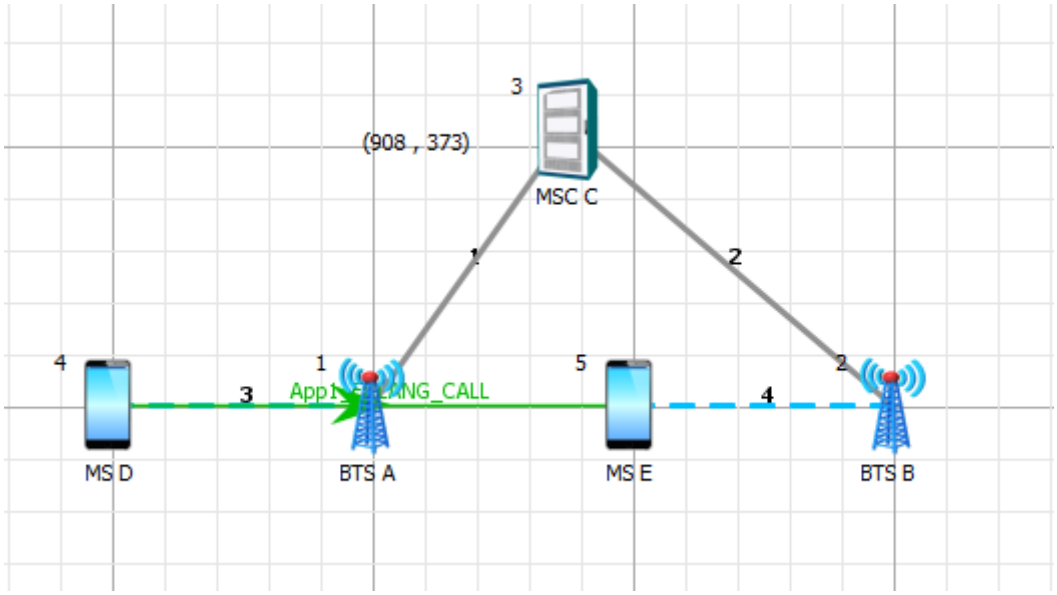
Sample configuration files for all networks are available inside the folder"<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

Cellular – Example Simulations:

Example 1: GSM Handover

Handover refer to the process of transferring an ongoing call or data session from one channel connected to the core network to another channel.

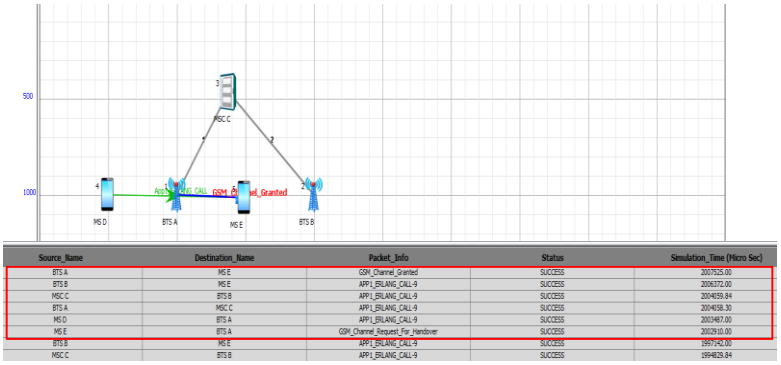
Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \Cellular\GSM\GSM_handover".



Settings done in sample network:

1. Network grid settings is set as 5000 * 5000 by default.
2. Place BTSA be at (1000, 1000) and the BTSB be at (2000, 1000).
3. Place MS D at 500, 1000 and set mobility to zero.
4. Place MS E at 1500, 1000and set mobility to 100 m/s.
5. Generate traffic from MS D to MS E and set the following properties.
    i. Call Inter arrival time = 1s
    ii. Call duration = 1000 s

6. Run simulation for say 250 seconds.
7. Check packet animation you are likely to see the mobile undergoing hand over between the two BS's.



Source_Name	Destination_Name	Packet_Info	Status	Simulation_Time (Micro Sec)
BTS A	MS E	GSM_Channel_Granted	SUCCESS	2007525.00
BTS B	MS E	APP1_ERLANG_CALL-9	SUCCESS	2006372.00
MSC C	BTS B	APP1_ERLANG_CALL-9	SUCCESS	2004559.84
BTS A	MSC C	APP1_ERLANG_CALL-9	SUCCESS	2004038.30
MS D	BTS A	APP1_ERLANG_CALL-9	SUCCESS	2003487.00
MS E	BTS A	GSM_Channel_Request_For_Handover	SUCCESS	2002910.00
BTS B	MS E	APP1_ERLANG_CALL-9	SUCCESS	1997142.00
MSC C	BTS B	APP1_ERLANG_CALL-9	SUCCESS	1994829.84

8. As in the simulation results window, under the cellular metrics users can find information about Handover Requests that were send by each Mobile Station (MS).
9. Open the packet trace and users see that hand over request packets will be shown at this time.

Note: In GSM hand over is based on distance (hard hand over), while in LTE it is based on SNR (soft hand over)

Example 2: CDMA – call blocking probability

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \Cellular\CDMA\call_blocking_probability".

Settings done in sample network:

1. A sample network is created with 60 MS, 1 BTS and 1MSC.
2. All the MS are placed in the range of BTS B (default 1 Km).
3. BTS B is connected via Wired Link to MSC A.
4. In BTS properties, under interface1_CDMA, Voice activity factor is set as 0.9 by default
5. 30 applications are created as follows:

   Application 1: Source id -3 and Destination id -4,

   Application 1: Source id - 5 and Destination id - 6,

   Application 1: Source id - 7 and Destination id - 8,

   Application 1: Source id - 9 and Destination id - 10,

   Likewise totally add 30applications up to Source id - 61 to Destination id– 62 and accept default properties.

(We have considered MSC to be device ID 1, and BTS as device ID 2)

6. Run simulation for 500 seconds.
7. Save the experiment and change the voice activity factor to 0.8, 0.7, 0.6…..0.1 and run the simulations.

Result:

In the metrics window, Cellular Metrics →Channel metrics, the channel count is mentioned. Users can see that when the system Voice activity factor decreases from 1.0 to 0.1, the number of channels increases from 43 to 427.

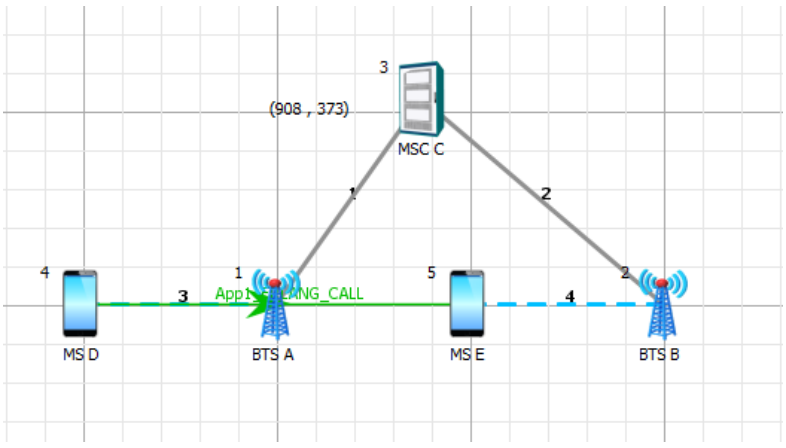
In MS metrics, the call generated and call blocked is shown for each MS. Add all the calls generated to obtain Total call generated, and add calls blocked for all MS Ids to obtain Total call blocked.

Calculate **call blocking probability** as ratio of Total call blocked to Total call generated.

When voice activity factor is decreased the number of channels available increases. Thus the system has more number of channels to handle the same number of calls (Note - Number of MS is constant and their properties are same across all experiments. So, they generate approximately same number of calls throughout). As the number of channels increases the call blocking probability decreases.

Example 3: CDMA Handoff

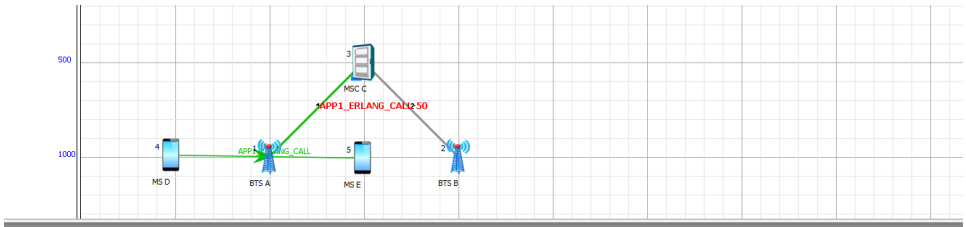
Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \Cellular\CDMA\CDMA_handover".



Settings done in sample network:

1. Network grid settings is set as 5000 * 5000 by default.

2. Place BTSA be at (1000, 1000) and the BTSB be at (2000, 1000).
3. Place MS D at 500, 1000 and set mobility to zero.
4. Place MS E at 1500, 1000 and set mobility to 100 m/s.
5. Generate traffic from MS D to MS E and set the following properties.
    i. Call Inter arrival time = 1s
    ii. Call duration = 1000 s
6. Run simulation for 250 seconds.
7. Check packet animation, you are likely to see the mobile undergoing hand over between the two BS's.



Source_Name	Destination_Name	Packet_Info	Status	Simulation_Time (Micro Sec)
BTS A	MS E	CDMA_Channel_Granted	SUCCESS	2002910.00
MS D	BTS A	APP1_ERLANG_CALL-50	SUCCESS	2000000.00
MS E	BTS A	CDMA_Channel_Request_For_Handover	SUCCESS	2000000.00
MSC C	BTS B	APP1_ERLANG_CALL-49	SUCCESS	1984753.54
BTS A	MSC C	APP1_ERLANG_CALL-49	SUCCESS	1984740.74
MS D	BTS A	APP1_ERLANG_CALL-49	SUCCESS	1980000.00
BTS B	MS E	APP1_ERLANG_CALL-48	SUCCESS	1965990.00
MSC C	BTS B	APP1_ERLANG_CALL-48	SUCCESS	1964753.54
BTS A	MSC C	APP1_ERLANG_CALL-48	SUCCESS	1964740.74
MS D	BTS A	APP1_ERLANG_CALL-48	SUCCESS	1960000.00
BTS B	MS E	APP1_ERLANG_CALL-47	SUCCESS	1947530.00
MSC C	BTS B	APP1_ERLANG_CALL-47	SUCCESS	1944753.54
BTS A	MSC C	APP1_ERLANG_CALL-47	SUCCESS	1944740.74
MS D	BTS A	APP1_ERLANG_CALL-47	SUCCESS	1940000.00
BTS B	MS E	APP1_ERLANG_CALL-46	SUCCESS	1929070.00
MSC C	BTS B	APP1_ERLANG_CALL-46	SUCCESS	1924753.54
BTS A	MSC C	APP1_ERLANG_CALL-46	SUCCESS	1924740.74
MS D	BTS A	APP1_ERLANG_CALL-46	SUCCESS	1920000.00

8. As in the simulation results window, under the cellular metrics users can find information about Handover Requests that were send by each Mobile Station (MS).
9. Also using packet trace, users can analyze the detailed information about the handover request packets

3.6 Wireless Sensor Network

NetSim Wireless Sensor Network library is based on IEEE802.15.4 standard also known as Zigbee. This protocol works on MAC and Physical Layer, while MANET protocol runs in the network layer, and UDP in the transport layer.

IEEE 802.15.4 uses either Beacon Enabled or Disabled Mode for packet transmission. In Beacon Enabled Mode nodes use slotted CSMA/CA algorithm for transmitting packets else they use Unslotted CSMA/CA

In NetSim WSN simulations, sensors sense "agents" and forward the sensed data to the "Sink" node.

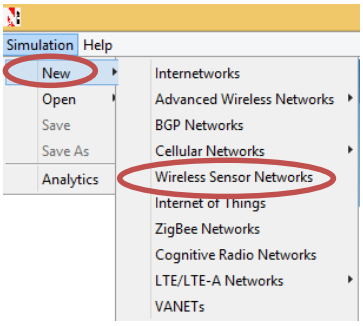
3.6.1 Devices in NetSim:

The Devices that are involved in WSN are:

1. **Sink node:** Sink node is the principal controller in WSN. It is to this sink node that all sensors send their data. In NetSim, users can drop only one sink node in a WSN.
2. **Sensor:** Sensors sense agents and then pass this sensed data through the network to Sink node. Therefore the sensors act as both sensors and routers. Sensors in NetSim are abstract in terms of what they sense, and NetSim focuses on the network communication aspects after sensing is performed. In Netsim Standard version, tje maximum number of sensors that can be placed in environment is 500. Sensor range is 0-100 meters and sensing interval is 0-5000 milliseconds.
3. **Agent:** Agent is an abstraction for physical phenomenon like temperature, pressure, humidity, sound, etc. The "sensors" sense the agent within its sensing range and upon sensing application (or information) packets are generated automatically by NetSim. In Netsim standard version, a maximum of 5 agents are allowed.

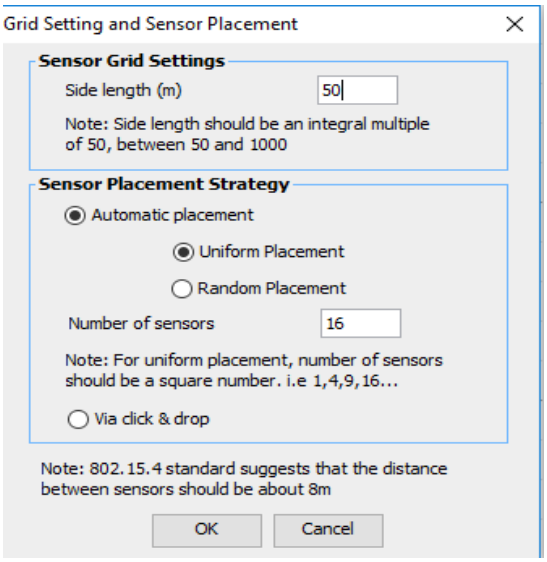
3.6.2 Designing WSN Networks

In the Simulation menu, select Simulation → New → Wireless Sensor Networks.



3.6.3 Procedure

In Grid Setting and Sensor Placement window, Side length of grid Environment can be changed which should be in a multiples of 50 m.



For automatic sensor placement, users can either chose from uniform placement or random placement options.

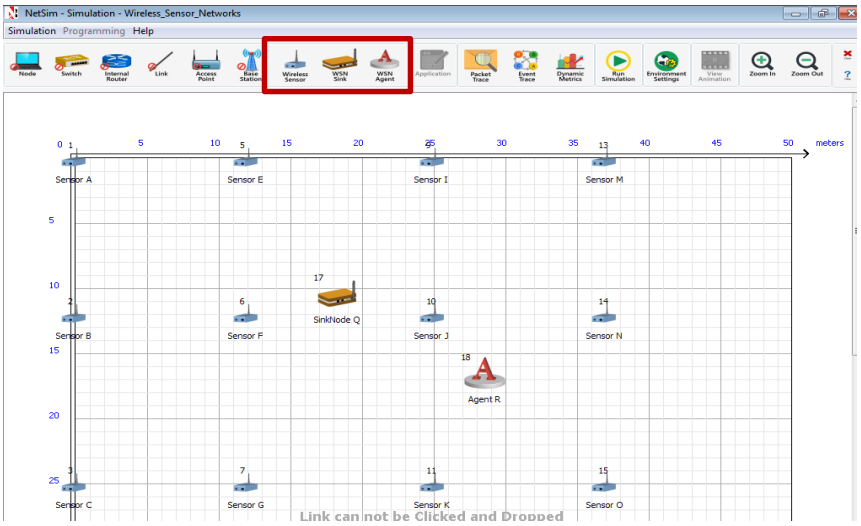
In random placement we can drop up to 500 sensors

In uniform placement number of sensors should be a square number since they are placed equally along the area of a square.

Users can click and drop sensors in the grid environment by chosing, “via click & drop” option.

Based on the user’s sensor placement strategy (in this example, uniform placement is chosen), sensors are placed in the grid environment as shown in below figure. Sensors are uniformly placed according to given inputs in the grid environment.

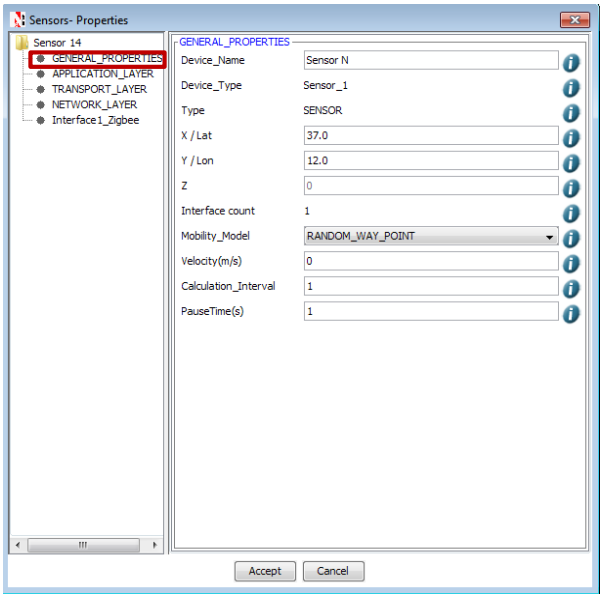
- Click and drop 1 sink node to the grid environment in our desired location.
- Click and drop 1 agent in desired location. (Users can drop upto a maximum of 5 agents).



3.6.4 Model features

GENERAL PROPERTIES –

Right click on any sensor and click on properties.The general properties of the sensor is shown:-



- **Device name** is the name of sensor which is editable and will reflect in the GUI before and after simulation.

- **X /Lat and Y/Lon** are the coordinate of particular sensor

- **Z coordinate** by default will be zero (this is reserved for future use)

Interface count is 1 since sensors share the wireless Multipoint-to-Multipoint medium.

- **Mobility Models in NetSim**

Mobility models represent the movement of the sensor and how their location and velocity. Typical mobility models provided in NetSim are as follows:

- **Random Walk mobility model**

  It is a simple mobility model based on random directions and speeds.

- **Random Waypoint Mobility Model**

  It includes pause time between changes in direction and/or speed.

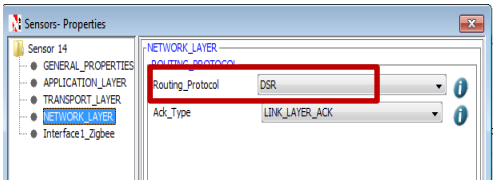
- **Group mobility**

  It is a model which describes the behaviour of sensors as they move together. i.e. the sensors having common group id will move together.

TRANSPORT LAYER – In transport layer, by default UDP is enabled in Transport layer. To run with TCP, users have to enable TCP protocol.

NETWORK LAYER- NetSim supports the following MANET routing protocols:

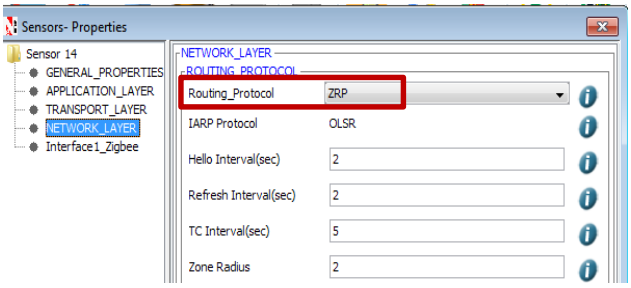
1. **DSR (Dynamic source routing)**
   In Sensor nework, by default Link Layer Ack is enabled. If Network Layer ack is enabled users will seet



   DSR_ACK in addition to Zigbee_ACK in MAC layer.

2. **AODV (Ad-hoc on demand distance vector routing)**

3. **ZRP (Zone routing protocol):** For interior routing mechanism Netsim uses OLSR protocol.



Hello interval describes the interval in which it will discover its neighbour routes.

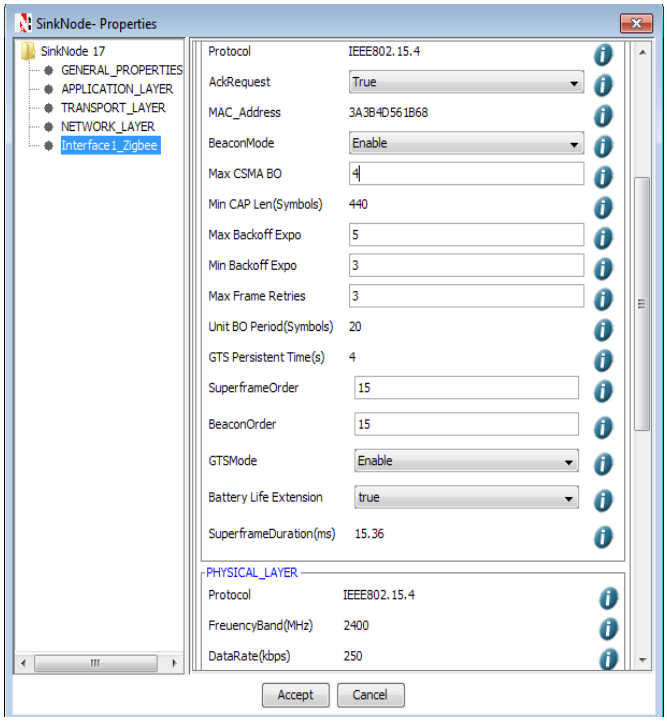
Refresh interval is the duration after which each active node periodically refresh routes to itself.

TC Interval is a Topology control messages are the link state signalling done by OLSR. These messages are sent at TC interval every time.

Zone radius: After dividing the network range of the divided network will be based on zone radius. It is the number of hop for one node to another.

4. **OLSR (Optimized link State Routing)**: Except zone radius all the parameters are similar to ZRP.

DATALINK LAYER – 802.15.4 (Zigbee Protocol) runs in MAC layer. In the sink node or pan coordinator properties users can configure the Beacon frames and the superframe structure**.**



***SuperframeOrder* –** It describes the length of the active portion of the Superframe, which includes the beacon frame. Range is from 0-15.

Beacon Order- Describes the interval at which coordinate shall transmit its beacon frames. Range is from 1-15.

GTS Mode (Guaranteed Time Slot) – If it is enabled it allows a device to operate on the channel within a portion of the super frame that is dedicated (on the PAN) exclusively to the device.

Battery life Extension subfield is 1 bit in length and shall be set to one if frames transmitted to the beaconing device.

Superframe Duration is divided into 16 equally sized time slots, during which data transmission is allowed. The value of supreframe duration by default is 15.36ms.

Max CSMA Backoff is the CSMA-CA algorithms will attempts before declaring a channel access failure. Having range 0-5.

Minimum CAP length is the minimum number of symbols forming the Contention access period. This ensure that MAC commands can still be transferred to devices when GTSs (Guaranteed time slots) are being used.

Max and Min backoff exponent values of CSMA-CA algorithms having range 3-5.

Max frame retries: is the total number of retries after failed attempts.

Unit Backoff period is the number of symbol forming the basic time period used by the CSMA-CA algorithms.

PHYSICAL LAYER- The frequency band used in NetSim WSN simulations is 2.4 GHz.

Data rate is the number if bits that are processed per unit of time. The data rate is fixed at 250 kbps per the 802.15.4 standard

Chip Rate: A chip is a pulse of direct-sequence spread spectrum code, so the chip rate is the rate at which the information signal bits are transmitted as pseudo random sequence of chips.

Modulation technique: O-QPSK (Offset quadrature phase shift keying) sometimes called as staggered quadrature phase shift keying is a variant of phase-shift keying modulation using 4 different values of the phase to transmit.

MinLIFSPeriod is minimum long inter frame spacing Period. It's a time difference between short frame and long frame in unacknowledged case and time difference between short frame and Acknowledged in acknowledge transmission.

SIFS(Short inter frame Symbol) is generally the time for which receiver wait before sending the CTS (Clear To Send) & acknowledgement package to sender, and sender waits after receiving CTS and before sending data to receiver. Its main purpose is to avoid any type of collision. Min SIFS period is the minimum number of symbols forming a SIFS period.

Phy SHR duration is the duration of the synchronization header (SHR) in symbol for the current PHY.

Phy Symbol per Octet is number of symbol per octet for the current PHY.

Turn Around Time is transmitter to receiver or receiver to transmitter turnaround time is define as the shortest time possible at the air interface from the trailing edge of the last chip(of the first symbol) of a transmitted PLCP protocol data unit to the leading edge of the first chip(of the first symbol) of the next received PPDU.

CCA (Clear Channel assessment) is carrier sensing mechanisms in Wireless Network. Here is the description:-

- **Carrier Sense Only:** It shall report a busy medium only upon the detection of a signal complaint with this standard with the same modulation and spreading characteristics of the PHY that is currently in use by the device. This signal may be above or below the ED threshold.
- **Energy Detection:** It shall report a busy medium upon detecting any energy above the ED threshold.
- **Carrier Sense with Energy Detection:** It shall report a busy medium using a logical combination of detection of a signal with the modulation and spreading characteristics of this standards and Energy above the ED threshold, where the logical operator may be AND or OR.

Receiver sensitivity is the minimum magnitude of input signal required to produce a specified output signal having a specified signal-to-noise ratio, or other specified criteria. It's up to our calculation what we want a receiver sensitivity.

Receiver ED threshold is intended for use by a network layer as part of a channel selection algorithms. It is an estimate of the received signal power within the bandwidth of the channel. No attempt is made to identify or decode signal on the channel. If the received signal power is greater than the ED threshold value then the channel selection algorithms will return false.

Transmitter Power is the signal intensity of the transmitter. The higher the power radiated by the transmitter's antenna the greater the reliability of the communication system. And connection medium is Wireless.

Power can be battery or main line. In case of battery following parameters will be considered:-

- **Recharging current** is the current flow during recharging. Range is from 0-1mA.
- **Energy Harvesting** is the Process by which energy is derived from external source, captured, and stored for small, wireless autonomous devices, like those used in wearable electronics and wireless sensor network.
- **Initial Energy** is the battery energy range is from 0 to 1000mW.
- **Transmitted current for the transmitting the power**. Range 0-20mA.
- **Idle mode** is the current flow during the ideal mode range is between 0-20mA.
- **Voltage** is a measure of the energy carried by the charge Range is from 0-10V.
- **Received current** of the current required to receive the data having range from 0-10mA.
- **Sleep mode current** is current flowing in sleep mode of battery range is from 0-20mA.

3.6.5 Set Link / Agent Properties

- **Right click** on the appropriate sensor or link and select Properties. Then modify the parameters according to the requirements. In Sensor Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer, Physical Layer and Power are Global i.e. changing properties in one node will automatically reflect in the other sensors in that network.
- Set the values according to requirement and click Accept.

The following table shows the local and global properties of sensor and Agent in NetSim.

1. Sensor Properties

Global properties	
General properties	**Default values**
Device name	Sensor ID
X / lat	Wherever placed by user
Y /long	Wherever placed by user
Velocity(m/s)	0
pause time(s)	1
Mobility model	Random_way_point
Network layer	
Routing protocol	DSR

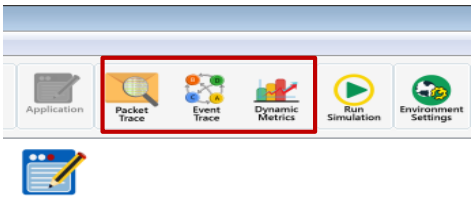
ACK _Type	LINK_LAYER_ACK
Data link layer	
ACK request	Enable
Max Csma BO	4
Max Csma Exponent	5
Min Csma Exponent	3
Max frame retires	3
Physical layer	
phySHRduration(symbols)	3
Physymbolperoctet	0.4
CCA mode	CARRIER_SENSE_ONLY
Reciever sensitivity(dbm)	-85
EDT threshold (dbm)	-95
Transmitter power(dbm)	1
Power	
Power source	Battery
Energy harvesting	ON
Recharging current (mA)	0.4
Initial energy (mw)	1000
Transmitting current(mA)	8.8
idle mode current(mA)	3.3
Voltage (v)	3.6
Receiving current(mA)	9.6
sleep mode current (mA)	0.237
Sensor	
Sensor range (m)	100
Sensor interval(ms)	1000

2. Agent properties:

Local Properties	
General Properties	**Default values**
Device name	Sensor A
X /lat	16.0
Y /long	8.0
Mobility Model	
Mobility model	Random_walk
Velocity (m/s)	10
Pause time (s)	1

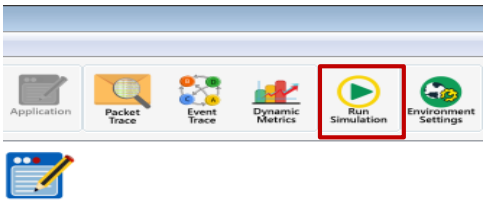
3.6.6 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.6.7 Run Simulation

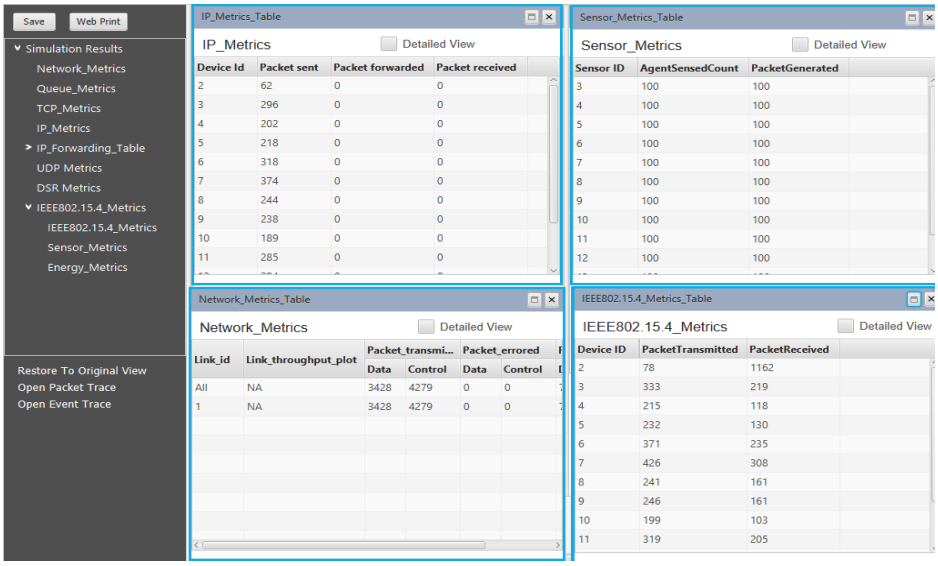
Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

3.6.8 Performance Metrics in WSN

After completion of simulation various performance metrics will be generated.



Network Metrics:

- Simulation time is the virtual time duration for the network is simulated.
- Packet Transmitted is the total number of packet transmitted from the source.
- Byte transmitted is the value increases with the number of packets transmitted on the network.
- Total number of packet error and collided also specified in the network metrics.
- Payload is the useful data and overhead are specified separately in the metrics.

Link Metrics:

- In link metrics each of the link status will be provided.
- In link metrics ID will be provided for each links here only one wireless link so only one ID.
- Total number of packets 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.
- Total number of packets error in the link inclusive of data and control packets.
- Packet Collided is the total number of packets collided in the link including data and control packets.
- Byte Transmitted 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 is the total payload transmitted in the link. The total overhead transmitted in the link. It includes the layer wise overheads and all control packets in the link.

Queue Metrics- This Metrics is for Router, Access point and other devices having buffer. Here no any devices hence no queue metrics.

IP Metrics - IP metrics provide node to node reception and transmission of the packet. This screen shot shows the IP metrics of a WSN network with 16 sensors and one sink.

IP Metrics_Table

IP Metrics Detailed View

Device Id	Packet sent	Packet forwarded	Packet received
1	232	0	0
10	198	0	0
11	1490	0	0
12	186	0	0
2	55	0	0
4	272	0	0
5	244	0	0
6	279	0	0
7	1391	0	0
8	194	0	0

DSR Metrics- In Network layer we can see various available protocol. We enabled DSR protocol and getting DSR metrics.

Users can observe the Route Error Packet send, Route reply packet, Route Request Forwarded, Route Reply Forwarded packets by intermediate nodes, Route Breaks by node, Total number of Packets originated, Transmitted, dropped and received by the node.

IEEE 802.15.4 Metrics- This is the standard for Local Personal Area Network Zigbee for MAC and Phy Layer of the sensor. Here for carrier sensing and congestion avoidance CSMA/CA is running according to that Parameters are mentioned in below figure.

IEEE802.15.4 Metrics_Table

IEEE802.15.4 Metrics ☑ Detailed View

Device ID	PacketTransmitted	PacketReceived	AckTransmitted	AckReceived	CCAAttempt	SuccessfulCCAAttempt	FailedCCA	Totalbackofftime(Micro seconds)
1	258	158	106	191	1157	635	522	2405760.0000
2	62	933	874	53	234	156	78	471680.0000
4	286	180	133	215	1217	675	542	2470080.0000
5	241	151	103	184	1089	581	508	2218880.0000
6	301	191	144	230	1263	724	539	2499840.0000
7	1392	1311	1266	1317	4167	3557	610	5321920.0000
8	194	103	55	129	1098	534	564	2432960.0000
9	275	168	118	205	1275	709	566	2490240.0000
10	196	112	60	141	985	481	504	2174720.0000
11	1454	1400	1355	1391	4179	3624	555	5140800.0000
12	213	103	52	139	1105	523	582	2496000.0000

Sensor Metrics - In sensor metrics sensor id is provided for each sensor and based on the mobility and sensing of agents timing are mentioned in this metrics.

Sensor Metrics_Table

Sensor Metrics ☑ Detailed View

Sensor ID	AgentSensedCount	PacketGenerat...	AgentId	X(m)	Y(m)	Time(Micro seconds)
1	100	100	3	26.0000	7.0000	0.0000
4	100	100	3	26.0000	7.0000	0.0000
5	100	100	3	26.0000	7.0000	0.0000
6	100	100	3	26.0000	7.0000	0.0000
7	100	100	3	26.0000	7.0000	0.0000
8	100	100	3	26.0000	7.0000	0.0000
9	100	100	3	26.0000	7.0000	0.0000
10	100	100	3	26.0000	7.0000	0.0000
11	100	100	3	26.0000	7.0000	0.0000
12	100	100	3	26.0000	7.0000	0.0000

- **Agent Sensed Count** is the total number of time sensor sensed the agent this is depending upon the sensor interval and simulation time.
- **Packet generated** is the total number of Packet generated by the sensor.
- **Agent ID** is also we can see in this metrics which indicating that which agent it is sensing because in our grid more then on i.e. 5 agent also can be possible.
- **X and Y** are the coordinated and position of Agent i.e. at which point sensor sensed the agent.
- **Time** is that which is noted at the time of sensing the agent.

Energy_Metrics – This metrics of wireless network devices where supply is not provided by the Main line.

Sensor are wireless devices equipped with a battery. This metrics gives parameters Transmitter Energy consumed, Receiver Energy consumed, Idle Energy Consumed, Sleep Energy and Total Energy Consumed.

When node is transmitting during that period energy consumed by node is transmitter Energy similarly receiving energy is the energy which is consumed by the node in receiving. In idle mode when it is not transmitting and receiving also it will consume some part of energy. Sleep Energy when sensor is doing nothing.

Total energy which is consumed by the sensor during simulation is the sum of all the consumption.

3.6.9 The IEEE 802.15.4 Protocol implementation in NetSim

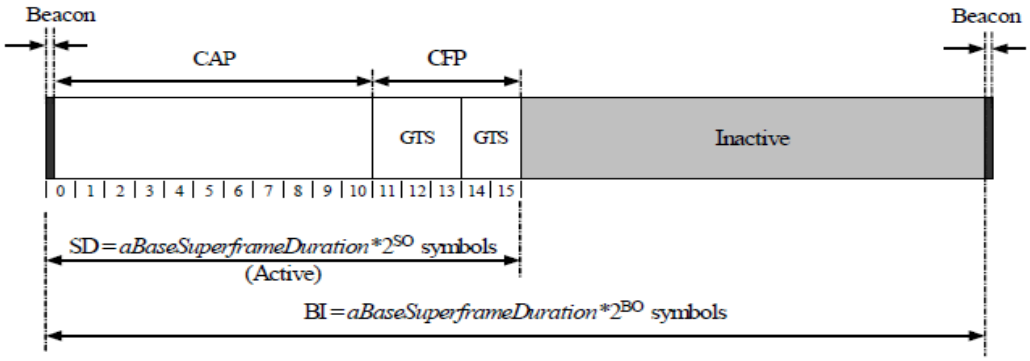
IEEE802.15/TG4 formulated the IEEE802.15.4 for lowrate wireless personal area network, i.e. LR-WPAN. The standard gives priority to low-power, low-rate and lowcost. The group devotes to the standard of the physical layer of WPAN network, i.e. PHY, and media access layer.

In NetSim 802.15.4 is implemented with the following features:

- Data rates of 250 kbps.
- Addressing modes: 16-bit IEEE addressing.
- 16 channels in the 2.45 GHz ISM band,
- Support carrier sense multiple access with collision avoidance, CSMA-CA.
- Power management to ensure low power consumption.
- GTS
- Support of contention-free and contention-based periods.
- The MAC protocol supports two operational modes that may be selected by the coordinator:
    - **Non Beacon-enabled mode:** In Non-Beacon-enabled mode, the devices can simply send their data by using Unslotted CSMA/CA. There is no use of a superframe structure in this mode.
    - **Beacon-enabled mode**: Beacons are periodically generated by the coordinator to synchronize attached devices and to identify the PAN. A beacon frame is the first part of a superframe, which also embeds all data frames exchanged between the nodes and the PAN coordinator. Data transmissions between nodes are also allowed during the superframe duration.

3.6.10 CSMA/CA Implementation in NetSim

- In both Slotted and Unslotted CSMA/CA cases, the CSMA/CA algorithm is based on backoff periods, where one backoff period is equal to aUnitBackoffPeriod Symbols = 20 symbols.
- This is the basic time unit of the MAC protocol and the access to the channel can only occur at the boundary of the backoff periods. In slotted CSMA/CA the backoff period boundaries must be aligned with the superframe slot boundaries where in unslotted CSMA/CA the backoff periods of one device are completely independent of the backoff periods of any other device in a PAN.
- The CSMA/CA mechanism uses three variables to schedule the access to the medium:
    - **NB** is the number of times the CSMA/CA algorithm was required to backoff while attempting the access to the current channel. This value is initialized to zero before each new transmission attempt.
    - **CW** is the contention windows length, which defines the number of backoff periods that need to be clear of channel activity before starting transmission. CW is only used with the slotted CSMA/CA. This value is initialized to 2 before each transmission attempt and reset to 2 each time the channel is assessed to be busy.
    - **BE** is the backoff exponent, which is related to how many backoff period a device must wait before attempting to assess the channel activity.
- In beacon-enabled mode, each node employs two system parameters: **beacon order (BO) and Superframe Order (SO)**.
- The parameter BO decides the length of beacon interval (BI), where **BI** = *aBaseSuperframeDuration* × $\mathbf{2^{BO}}$ **symbols** and $0 \leq BO \leq 14$; while the parameter SO decides the length of superframe duration (SD), where **SD** = *aBaseSuperframeDuration* × $\mathbf{2^{SO}}$ **symbols** and $0 \leq SO \leq BO \leq 14$.
- The value of *aBaseSuperframeDuration* is fixed to 960 symbols. The format of the superframe is defined as shown in the following figure:-



- Furthermore, the active portion of each superframe consists of three parts: beacon, CAP, and CFP, which is divided into 16 equal length slots. The length of one slot is equal to $aBaseSlotDuration \times 2^{SO}$ symbols, where *aBaseSlotDuration* is equal to 60 symbols.
- In CAP, each node performs the CSMA/CA algorithm before transmitting data packet or control frame. Each node maintains three parameters: the number of backoffs (NB), contention window (CW), and backoff exponent (BE).
- The initial values of NB, CW, and BE are equal to 0, 2, and *Min Backoff Expo*, respectively, where *Min Backoff Expo* is by default 3 and it can be set upto 8.
- For every backoff period, node takes a delay for random backoff between 0 and 2^{BE}-1 Unit backoff Time (UBT), where UBT is equal to 20 symbols (or 80 bits).
- A node performs clear channel assessment (CCA) to make sure whether the channel is idle or busy, when the number of random backoff periods is decreased to 0.
- The value of CW will be decreased by one if the channel is idle; and the second CCA will be performed if the value of CW is not equal to 0. If the value of CW is equal to 0, it means that the channel is idle; then the node starts data transmission.
- However, if the CCA is busy, the value of CW will be reset to 2; the value of NB is increased by 1; and the value of BE is increased by 1 up to the maximum BE (*Max Backoff Expo*), where the value *Max Backoff Expo* is by default 5 and can be set upto 8.
- The node will repeatedly take random delay if the value of NB is less than the value of Max CSMA BO (*macMaxCSMABackoff),* where the value of Max CSMA BO is equal to 4; and the transmission attempt fails if the value of NB is greater than the value of Max CSMA BO.

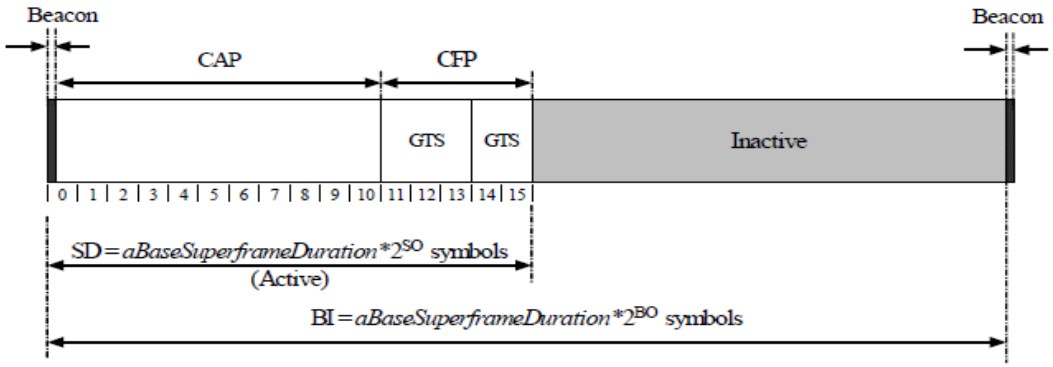
3.6.11 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

WSN – Example Simulations

Example 1: Beacon time Calculation

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \WSN\beacon_time"



Settings done in example config file

1. Beacon mode -> Enable in SinkNode , Superframe Order (SO) -> 10 and Beacon Order (BO) -> 12
2. Channel characteristics -> Path Loss only, Path Loss Model -> Log Distance and path loss exponent -> 2
3. Simulation time -> 200 sec

Steps to calculate Beacon Time theoretically:

1. Beacon Interval(BI)= Super Frame duration (Active Period) + Inactive Period(IP)
   i. BI = abasesuperframe duration*2 ^BO = 15.36ms * 2^12 = 62914.56ms ~ 62s
   ii. SD = abasesuperframe duration*2 ^SO = 15.36ms * 2^10 = 15728.64ms
   iii. To find Inactive period, IP = BI – SD = 62914.56ms - 15728.64ms = 47185.92ms
   iv. Super Frame duration is divided into 16 slots (1st slot is allocated for beacon frame)
   v. Each slot time = 15728.64ms / 16 = 983.04ms

2. In packet trace users should get four zigbee_beacon_frames at 0, 62.9, 126, 189 secs, because the time interval between two beacon frames is 62 seconds.
3. 4 beacons were transmitted, so beacon time = 983.04ms * 4 = 3932.16ms (since 1 beacon = 1 time slot). Check this beacon time in 802.15.4 metrics

Example 2: CAP Time Calculation

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \WSN\CAP_time"

Settings done in example config file

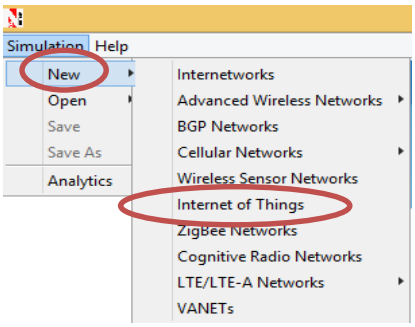
1. Beacon mode -> Enable in SinkNode , Superframe Order (SO) -> 10 and Beacon Order (BO) -> 12
2. Channel characteristics -> Path Loss only, Path Loss Model -> Log Distance and path loss exponent -> 2
3. Simulation time -> 100 sec

Steps to calculate CAP Time theoretically:

1. To find CAP time,
2. BI is 62914.56ms -> So in 100s, two beacon frames should be transmitted at 0 & 62s
3. Check no. of beacon frames transmitted in 802.15.4 metrics
4. Here CFP = 0 because there is only 1 sensor.
5. CAP Time = SD – beacon time = (15728.64ms) – (983.04ms) = 14745.6ms
6. Since two Beacon frames are transmitted, CAP time = 2 * 14745.6ms = 29491200µs
7. Check and compare the theoretical CAP time with NetSim simulation results in 802.15.4 metrics in Results Windows

3.7 Internet of Things

3.7.1 New Experiment



In the **Simulation** menu select **Simulation → New → Internet of Things**

3.7.2 Introduction

Internet of Things (IoT) is an ecosystem of connected physical objects that are accessible through the internet. It is the network of physical objects that can communicate, sense or interact with their internal states or the external environment.

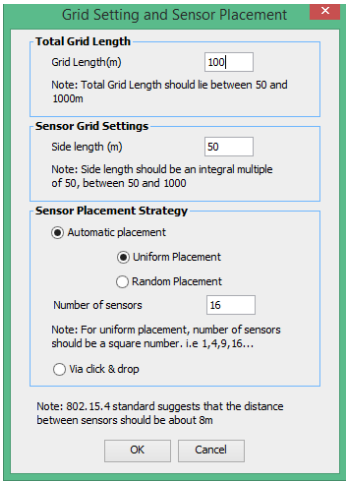
The 'thing' in IoT could be a person with a heart monitor or an automobile with built-in-sensors, i.e. objects that have been assigned an IP address and have the ability to collect and transfer data over a network without manual assistance or intervention.

NetSim IOT is modeled as a wireless sensor network that can connect to the internet via a 6LowPAN Gateway. The default protocols in the WSN section is AODV with IPv6 addressing in L3 and 802.15.4 MAC & PHY. This WSN sends data to a LowPAN Gateway which has a Zigbee (802.15.4) interface and a WAN Interface. The Zigbee interface allows wireless connectivity to the WSN while the WAN interface connects to the internet.

Any WSN comprises of two parts, the sensing part and the network communication part. NetSim is "agnostic" to the sensing and this sensing is abstracted as an Agent (sometime known as Agent based modeling). Whatever is sensed is finally converted to a "packet" and it is from this point on that NetSim simulation can be used.

NetSim does not provide any connectivity / interfacing to real hardware / sensors. However, NetSim provides interfacing to MATLAB or via text files. In case of text files, real sensors can write their data to files which can then take as input by NetSim.

Create Scenario



Total Grid Length (m) settings allows the user to set the total environment length of IOT Networks containing sensors, LoWPAN gateway, wired nodes, routers, switches, access point, wireless nodes.

Sensor Grid Settings (m) allows the user to set the environment length for placing the sensors uniformly or randomly.

Users can manually create the scenario by selecting “Via click and drop”, or place the sensors automatically in an uniform or random manner.

Adding Sensor - Click on **Sensor Node** icon in toolbar and click and drop inside the grid.

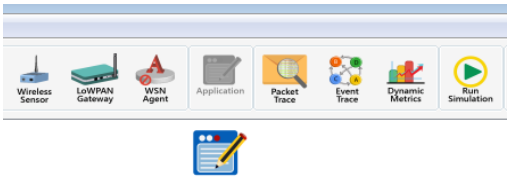
Adding LoWPAN gateway- LoWPAN is an acronym of Low power Wireless Personal Area Networks. The LoWPAN IoT gateway functions as a border router in a LoWPAN network, connecting a wireless IPv6 network to the Internet. Designed to send IPv6 packets over IEEE802.15.4-based networks and implementing open IP standards including TCP, UDP, HTTP and more, the standard offers end-to-end addressable nodes, allowing a router to connect the network to IP**.**

Click on the **LoWPAN gateway** icon in the toolbar and click and drop inside the grid.

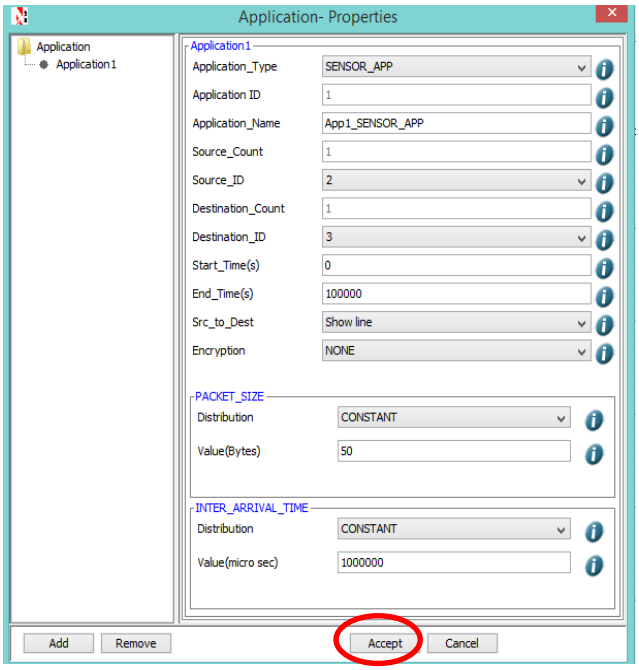
Users can also add devices as shown in Internetworks scenario.

3.7.3 Set Node, Link and Application Properties

- User need not connect the sensors with **LoWPAN gateway** using wireless links.
- Interconnection among other devices is same as in Internetworks.
- **LoWPAN gateway** can be connected with router using wired links.
- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements.
    - Routing Protocol in Application Layer of router and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global i.e. changing properties in one node will automatically reflect in the others in that network.
    - In Sensor Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer, Physical Layer and Power are Global i.e. changing properties in one node will automatically reflect in the others in that network.
- Set the values according to requirement and click Accept.
- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.

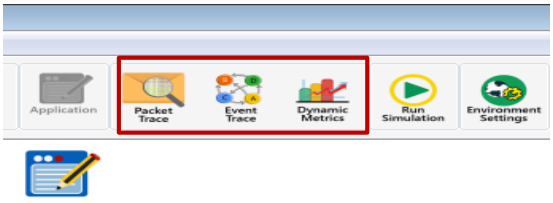


- Set the values according to requirement and click Accept.



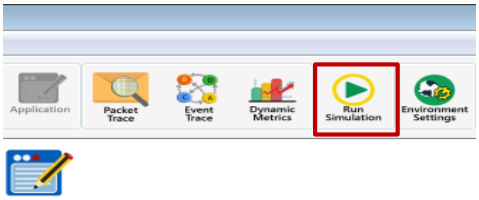
3.7.4 Enable Packet Trace, Event Trace & Dynamic Metrics(Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.7.5 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

3.7.6 RPL protocol in IOT

Routing Protocol for Low power and Lossy Nertworks (RPL) Overview

Low-power and Lossy Networks consist largely of constrained nodes (with limited processing power, memory, and sometimes energy when they are battery operated). These routers are interconnected by lossy links, typically supporting only low data rates that are usually unstable with relatively low packet delivery rates. Another characteristic of such networks is that the traffic patterns are not simply point-to-point, but in many cases point-to-multipoint or multipoint-to-point.

RPL Routing Protocol works in the Network Layer, and uses IPv6 addressing. It runs a distance vector routing protocol based on Destination Oriented Directed Acyclic Graph (DODAGs).

Teminology of RPL routing protocol:

- **DAG (Directed Acyclic Graph)**: A directed graph having the property that all edges are oriented in such a way that no cycles exist. All edges are contained in paths oriented toward and terminating at one or more root nodes.
- **DAG root**: A DAG root is a node within the DAG that has no outgoing edge. Because the graph is acyclic, by definition, all DAGs must have at least one DAG root and all paths terminate at a DAG root. In NetSim, only single root is possible. i.e. the 6LowPAN Gateway.
- **Destination-Oriented DAG (DODAG)**: A DAG rooted at a single destination, i.e., at a single DAG root (the DODAG root) with no outgoing edges.
- **Up:** Up refers to the direction from leaf nodes towards DODAG roots, following DODAG edges. This follows the common terminology used in graphs and depth-first-search, where vertices further from the root are "deeper" or "down" and vertices closer to the root are "shallower" or "up".
- **Down:** Down refers to the direction from DODAG roots towards leaf nodes, in the reverse direction of DODAG edges. This follows the common terminology used in graphs and depth-first-search, where vertices further from the root are "deeper" or "down" and mvertices closer to the root are "shallower" or "up".
- **Rank**: A node's Rank defines the node's individual position relative to other nodes with respect to a DODAG root. Rank strictly increases in the Down direction and strictly decreases in the Up direction.

- **RPLInstanceID:** A RPL Instance ID is a unique identifier within a network. DODAGs with the same RPLInstanceID share the same Objective Function.

- **RPL instance**: When we have one or more DODAG, then each DODAG is an instance. A RPL Node may belong to multiple RPL Instances, and it may act as router in some and as a leaf in others. Any sensor can be configured as a Router or Leaf. Leaf nodes doesn't take part in RPL routing.

- **DODAG ID**: Each DODAG has an IPV6 ID. This ID is given to its root only. And as the root doesn't change ID also don't change

- **Objective Function (OF):** An OF defines how routing metrics, optimization objectives, and related functions are used to compute Rank. Furthermore, the OF dictates how parents in the DODAG are selected and, thus, the DODAG formation.

  The objective function in NetSim RPL seeks to find the route with the best link quality. Link quality calculations, available in Zigbee Prorject 802.15.4 C file with function get_link_quality( ):

$$Lq = \frac{1-p}{rs}$$

  Where,

  p = received power (dBm) and

  rs = Receiver sensitivity (dBm)

  And,

$$Final\ Link\ Quality = \frac{(Sending\ Link\ Quality + Receiving\ link\ quality)}{2}$$

  The rank calculations are done in Neighbour.c in RPL project and is

  *Rank = (Max_increment - Min_Increment) * (1 - Lq) ^ 2 + Min_Increment*

  The link quality in this case is based on received power and can be modified by the user to factor in distance, delay etc.

Topology Construction:

NetSim IOT WSNs do not typically have predefined topologies, for example, those imposed by point-to-point wires, so RPL has to discover links and then select peers sparingly. RPL routes are optimized for traffic to or from one or more roots that act as sinks for the topology. As a result, RPL organizes a topology as a Directed Acyclic Graph (DAG) that is partitioned into one or more Destination Oriented DAGs (DODAGs), one DODAG per sink.

RPL identifiers:

RPL uses four values to identify and maintain a topology:

- The first is a RPLInstanceID. A RPLInstanceID identifies a set of one or more Destination Oriented DAGs (DODAGs). A network may have multiple RPLInstanceIDs, each of which defines an independent set of DODAGs, which may be optimized for different Objective Functions (OFs) and/or applications. The set of DODAGs identified by a RPLInstanceID is called a RPL Instance. All DODAGs in the same RPL Instance use the same OF.

- The second is a DODAGID. The scope of a DODAGID is a RPL Instance. The combination of RPLInstanceID and DODAGID uniquely identifies a single DODAG in the network. A RPL Instance may have multiple DODAGs, each of which has an unique DODAGID.

- The third is a DODAGVersionNumber. The scope of a DODAGVersionNumber is a DODAG. A DODAG is sometimes reconstructed from the DODAG root, by incrementing the DODAGVersionNumber. The combination of RPLInstanceID, DODAGID, and DODAGVersionNumber uniquely identifies a DODAG Version.

- The fourth is Rank. The scope of Rank is a DODAG Version. Rank establishes a partial order over a DODAG Version, defining individual node positions with respect to the DODAG root.

DIS (DODAG Information Solicitation) transmission:

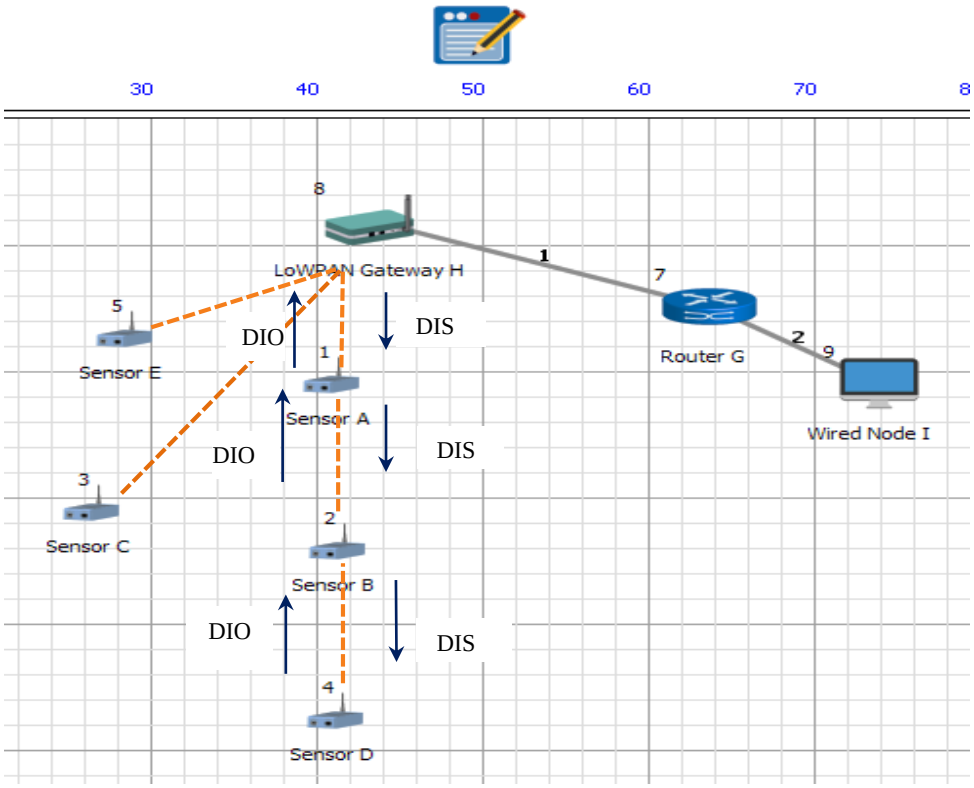
Root sends DIS message to the Router/Leaf which are in range. The Router in turn broadcasts its further and so on.

DIO (DODAG Information Object) transmission:

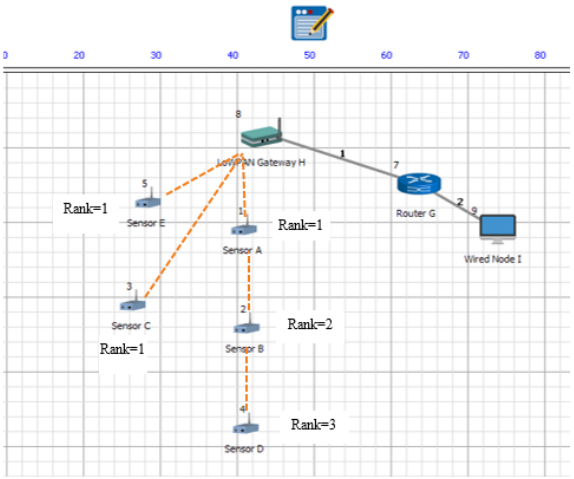
RPL nodes transmit DIOs using a Trickle Timer. This message is multicasted downwards in a DODAG. With DIO child parent relationship and siblings relationship is established.

DODAG Construction

- Nodes periodically send link-local multicast DIO messages
- Stability or detection of routing inconsistencies influence the rate of DIO messages
- Nodes listen for DIOs and use their information to join a new DODAG, or to maintain an existing DODAG
- Nodes may use a DIS message to solicit a DIO. RPL established the following graph:-



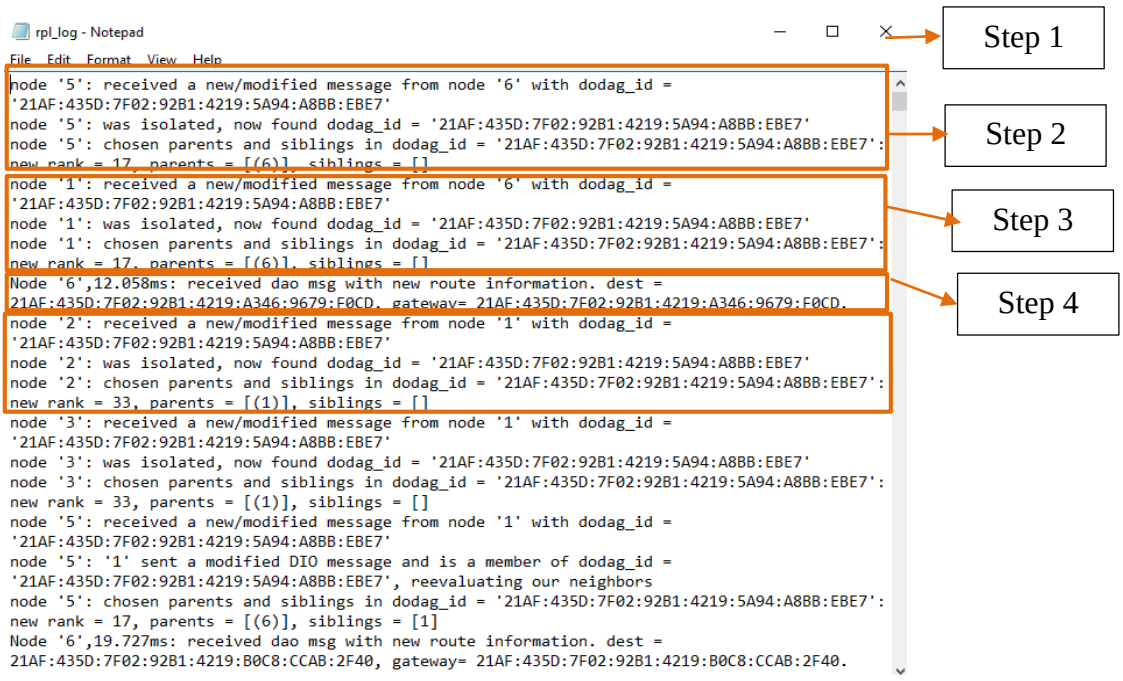
- Rank is then established. Rank is decided based on the DIS message received from the Root and the link quality.



- Based on information in the DIOs the node chooses parents to the DODAG root
- As a Result, the nodes follows the Upward routes towards the DODAG root.
- If the destination is unreachable then the root will drop the packet.

RPL_log file:

Once simulation is completed, users can find rpl_log file inside the temp path. This file contains information about the DODAG formation. An example rpl_log file is shown below:-



Explanation of the log file:

Step 1:

- Node 5 receives a DIO msg from Node 6 (i.e root).
- Node 5 found the DODAG id
- Based on the DIO message received from Node 6, Node 5 choses its "Parent as Node 6" and establishes its "New Rank = 17". It doesn't have any siblings.

Step 2:

- Node 1 receives a DIO msg from Node 6 (i.e root).
- Node 1 found the DODAG id
- Based on the DIO message received from Node 6, Node 1 choses its "Parent as Node 6" and establishes its "New Rank = 17". It doesn't have any siblings.

Step 3:

- Node 6 receives as DAO message from Node 1 with the new route information about the destination and the Gateway.

Step 4:

- Node 2 receives a DIO msg from Node 1 (i.e Sensor which is configured as Router).
- Node 2 found the DODAG id
- Based on the DIO message received from Node 1, Node 2 choses its "Parent as Node 1" and establishes its "New Rank = 33" since it is in the next Rank level. It doesn't have any siblings.

  Likewise DODAG formation throughout the simulation is logged inside the rpl_log file

Limitations:

- NetSim currently supports only single root.
- NetSim GUI supports only one RPL instance. Mutiple RPL instance can be created by manually editing the config file.

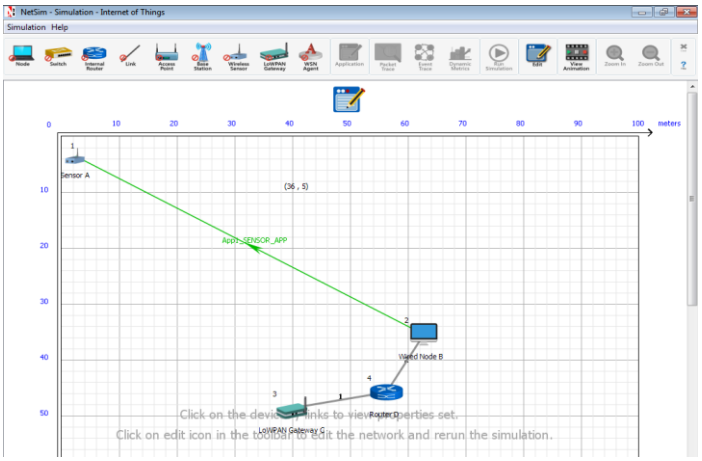
3.7.7 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

IOT – Example Simulations:

Example 1: Power Usage

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \IOT\Energy_model"



Settings done in sample network:

2. Beacon mode -> enable, Beacon Order -> 10, Superframe order -> 10
3. Simulate for 100 sec.
4. Check the Energy model metrics in metrics window, users should get zero value for Sleep energy consumed.
5. Click on edit Network.
6. Enable Beacon mode in PAN Coordinator and set Superframe Order (SO) and Beacon Order (BO) as 10 and 12 (0 <= SO <= BO <= 14).
7. Simulate for 100 sec.
8. Check the Energy model metrics in metrics window, users should get non-zero value for Sleep energy consumed.

Inference:

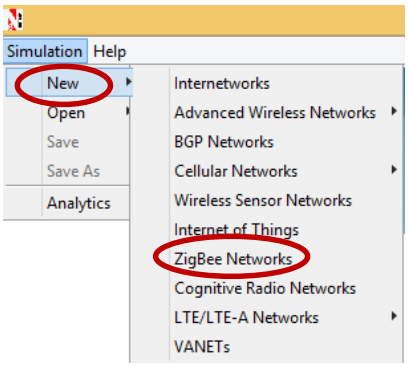
In sample 1, users can observe sleep energy consumed value will be zero in power model metrics in the Results Window since sensor is in active state all the time. i.e., if SO=10 and BO=10, there won't be any inactive portion of the Superframe.

In sample 2, sleep energy consumed will be non-zero since sensor goes to sleep mode during the inactive portion of the Superframe.

3.8 Zigbee

3.8.1 New Experiment

In the **Simulation** menu select **Simulation** → **New** → **ZigBee Networks**



3.8.2 Create Scenario

Adding Node -

- **Click** on the **ZigBee** icon in the toolbar and click and drop it inside the grid (i.e. **Visibility Range** - The systems can move and communicate in this range only).
- A Node cannot be placed on another Node. A Node cannot float outside the **grid**.

Adding PAN Coordinator - Click on the **PAN Coordinator** icon in the toolbar and click and drop inside the **grid**.

Set Environment Properties

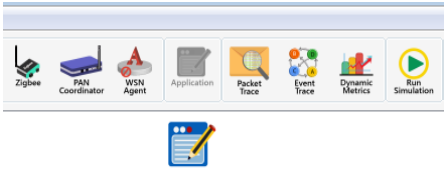
Right click in side of the on the Environment and **click Properties.**

3.8.3 Modifying/Viewing/Accepting Properties

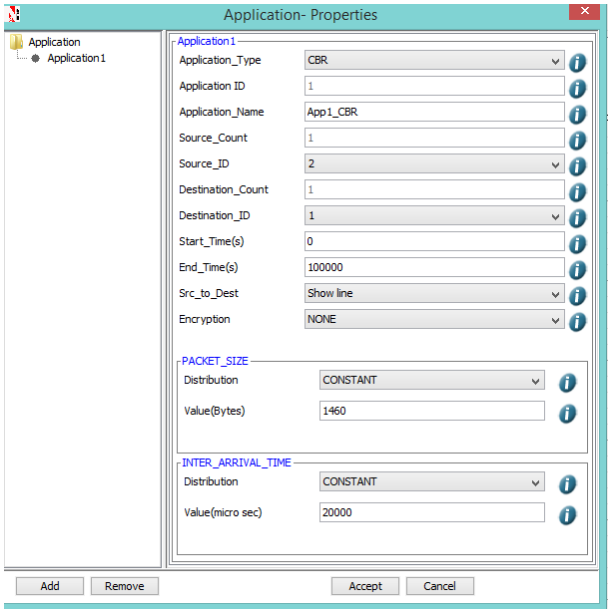
On opening an already configured properties of environment, the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

3.8.4 Set Node, Link and Application Properties

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements. In Zigbee Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer, Physical Layer and Power are Global i.e. changing properties in one node will automatically reflect in the others in that network.
- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.

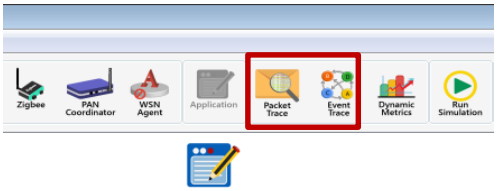


- Set the values according to requirement and click Accept.



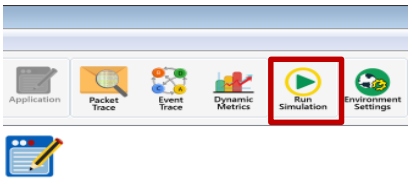
3.8.5 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.8.6 Run Simulation

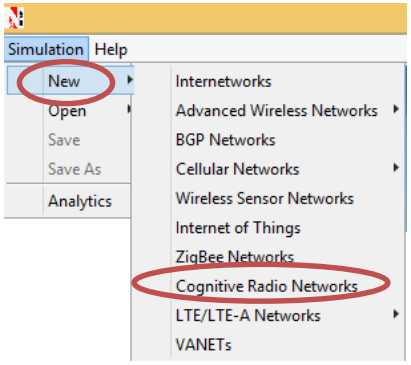
Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

3.9 Cognitive Radio

Cognitive Radio Network simulation is available from v7 of NetSim. Cognitive Radio Networks allows you to connect, if required, with Ethernet, Wireless LAN, IP Routing, TCP / UDP and allows users to log packet and event (in NetSim standard version only) traces.



3.9.1 New Experiment

In the **Simulation** menu select → **New** → **Cognitive Radio Networks.**

3.9.2 Create Scenario

Adding Devices –

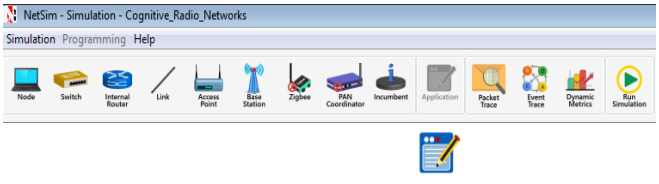
- Cognitive Radio Networks comes with the palette of various devices like Cognitive Radio CPE, Cognitive Radio Base Station, Switch, Router, Wired Node, Wireless Node, Access point, Incumbent etc.
- Select the desired devices in the toolbar and click and drop on the environment.
- To remove devices, right click on the particular device and then click Remove.

Connect the devices

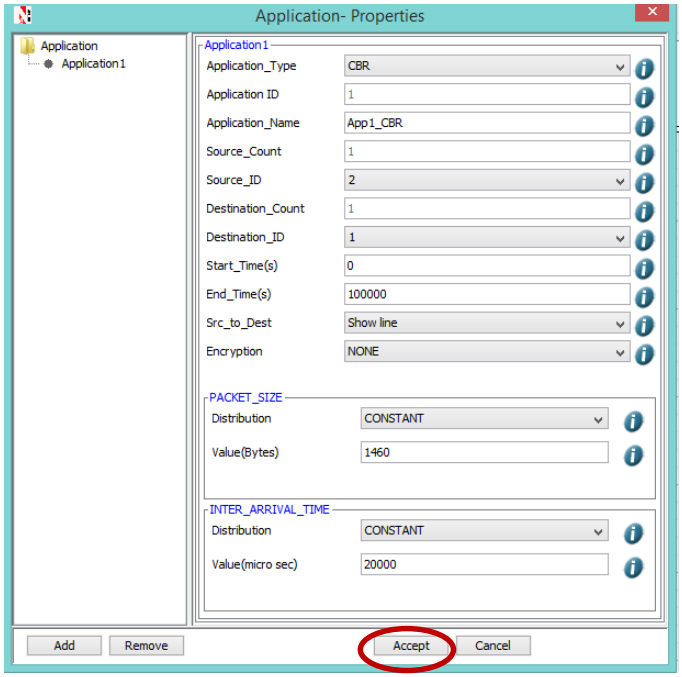
Select the appropriate link in the toolbar and connect the devices by clicking on the device 1 and device 2.

3.9.3 Set Node, Link and Application Properties

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements. Routing Protocol in Application Layer of router and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global i.e. changing properties in one node will automatically reflect in the others in that network.
- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.

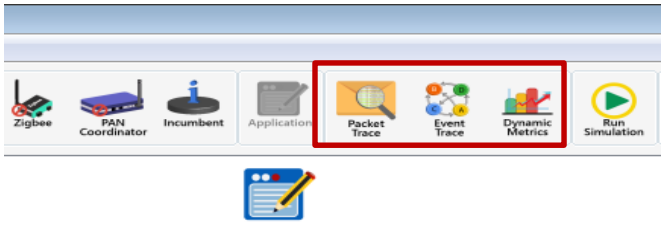


- Set the values according to requirement and click Accept.



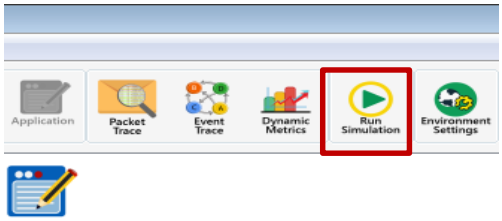
3.9.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.9.5 Run Simulation

Click on **Run Simulation** icon on the top toolbar. Set the Simulation Time and click on Simulate.



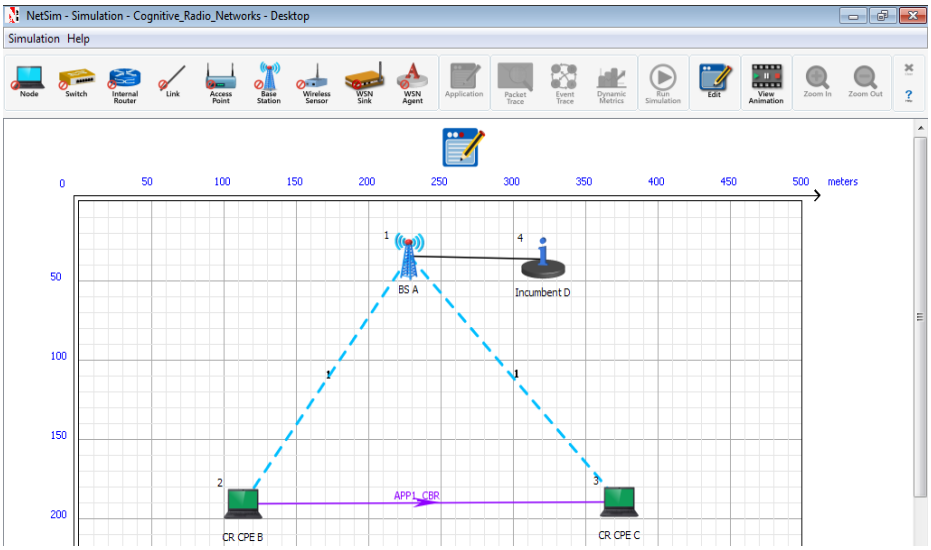
3.9.6 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

Cognitive Radio – Example Simulations:

Example 1: Keepout Distance

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \CR\Keepout_distance"



Settings done in example config file:

1. Distance between Incumbent and CRCPE should be more than 100m.
2. Base station Properties: Operational Freq Start -> 54 MHz, Operational Freq End -> 60 MHz in both Incumbent and physical layer properties.
3. Channel bandwidth -> 6
4. ON duration -> 10
5. OFF Duration -> 0
6. Keepout distance -> 100m
7. Simulation time -> 100s

Results:

1. In results windows open CR Incumbent metrics and check the operational time(100s) and idle time(0s)
2. Then open Application Metrics, you should get non-zero throughput value, because there will be no Interference if distance between incumbent and CR CPEs is more than the keepout distance.

Example 2: PU / SU spectrum usage

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \CR\PU_SU_SpectrumUsage"

Settings done in sample network:

1. Open Base Station properties -> select interface_CR -> change Operational Freq Start as 54 MHz and Operational Freq End as 60 MHz in both Incumbent and physical layer properties.
2. Channel bandwidth = 6
3. On_Duration = 10s
4. Off_Duration = 10s
5. Keepout distance = 100m
6. Disable TCP
7. Ensure that the distance between incumbent and CR CPEs should be less than 100m (keepout distance)
8. In application properties, select CBR application
9. Enable packet Trace and Simulate for 100s
10. Open packet trace and check at what time data packets are transmitting (Check PHY_Layer_START time)
11. First 10s CPE should use the channel and next 10s Incumbent should use the channel alternatively.

Time	0-10s (OI)	10-20 (OT)	20-30s (OI)	30-40s (OT)	40-50s (OI)	50-60s (OT)	60-70s (OI)	70-80s (OT)	80-90s (OI)	90-100 (OT)
User	CPE	Inc.	CPE	Inc.	CPE	Inc.	CPE	Inc.	CPE	Inc.

Notes on Cognitive Radio implementation in NetSim:

- CR BS allocates max one symbol per CPE. If the generation rate is more than the data filled in one symbol then allocation fails and it results in zero throughput.
- The first symbol is reserved for CR control frames or any broadcast PDU
- **Operational frequency**: It is the frequency band at which the incumbent operates. It ranges from 54 MHz to 862 MHz.
- **Operational interval(s)**: It is the time gap between two successive incumbent operations. It ranges from 0-10.
- **Operational time(s)**: It is the active period of the incumbent. i.e. If the operational interval is set to 5s, then incumbent operates with an interval of every 5s. If the operational interval is set to 0s, then the incumbent remains active.

For Operational Time(s) = 4, Operational interval=9

The timing diagram is as follows:

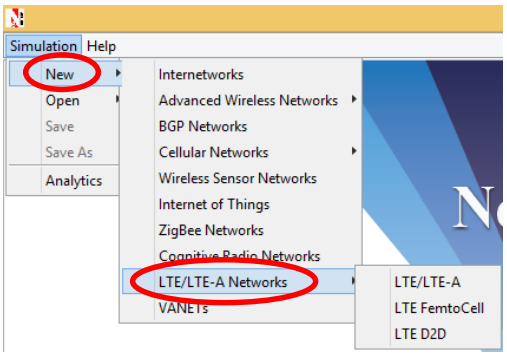
Incumbent --- 0s to 10s (OFF), 10s to 14s (ON), 14s to 24s (OFF), 24s to 28s (ON) ... and so on.

- NetSim gives App layer throughput which is lesser than that of MAC layer throughput because of
  - TCP connection establishment
  - ARP set-up
  - Service flow creation CPE-BS and BS-CPE
  - BW request
- To avoid the above effects
  - Generate custom traffic
  - Set DL/UL Ratio as 1:1 so that the BS transmits whatever it receives
  - Run UDP in transport layer
  - Use static ARP
  - Run Simulation for more than 100 sec

3.10 LTE/LTE-A

3.10.1 New Experiment

In the **Simulation** menu select → **New→ LTE/LTE-A Networks (LTE/LTE-A, Femto cell, D2D)**



3.10.2 Create Scenario

Adding MME- Click on the **Router** icon in the tool bar, click and drop the **MME (Mobility Management Entity)** onto the **environment.**

Adding ENB - Click on the **Evolved node B (ENB)** icon in the toolbar and click it onto the **environment**.

Adding Relay - Click on the **Relay** icon in the toolbar and click it onto the **environment**.

Adding UE –

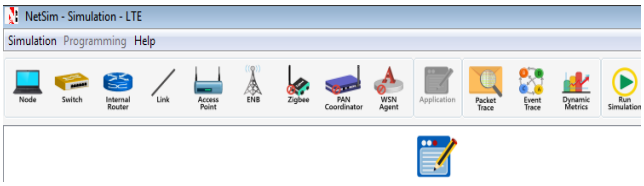
- **Click** on the **UE (User Equipment)** icon from the **Node** icon in the toolbar, click and drop it on the **ENB coverage** area.
- **UE** cannot be placed on another **UE**. It has to be clicked and placed on the **ENB node coverage** area.

Connect the devices

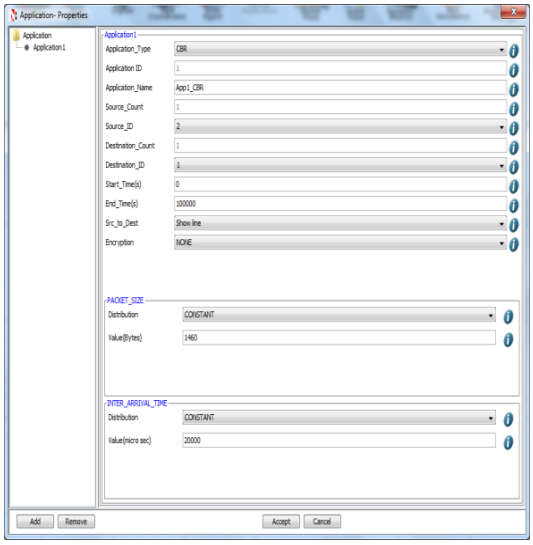
Select the appropriate link in the toolbar and connect the devices by clicking on the device 1 and device 2.

3.10.3 Set Node, Link and Application Properties

- **Right click** on the appropriate node or link and select Properties. Then modify the parameters according to the requirements. Routing Protocol in Application Layer of router and all user editable properties in DataLink Layer and Physical Layer of Access Point and Wireless Node are Global i.e. changing properties in one node will automatically reflect in the others in that network. Transmission Mode Index of ENB, Relay and UE are also global properties.
- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.

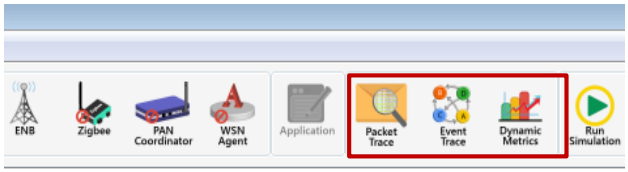


- Set the values according to requirement and click Accept.



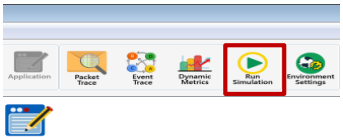
3.10.4 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.10.5 Run Simulation 

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

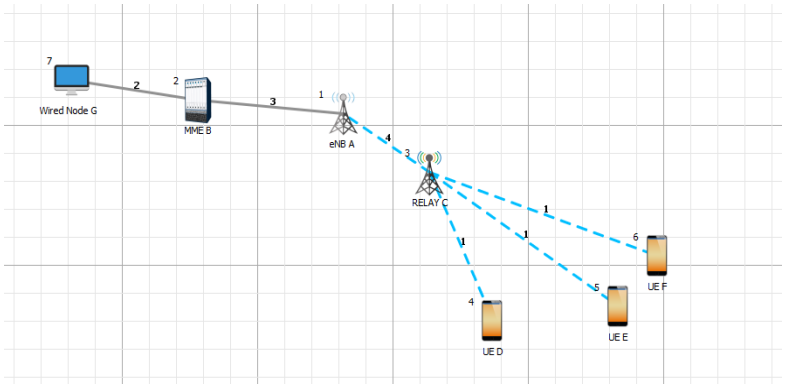
3.10.6 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

LTE-A - Example Simulations

Example 1: Transmission Mode Index

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \LTE\tx_mode_index".



Settings done in sample network:

1. Select UDP in all devices in Transport Layer
2. Change all wired link speeds by 10 Gbps
3. Create 3 CBR applications from **UEs to Wired Node** with generation rate 300Mbps(Packet size=1460, IAT=37micro sec)
4. Simulate for 10 seconds

Transmission Mode Index for ENB and Relay (Txn. Mode, Tx. Antenna Count, Rx. Antenna Count)	Throughput for 3 applications (with relay)	Throughput for 3 applications (without relay)
0 (1 1 1)	74.3, 0, 0	74.3, 0, 0
1 (2 2 2)	74.3, 0, 0	74.3, 0, 0
2 (3 2 2)	74.3, 0, 0	74.3, 0, 0
3 (3 4 2)	74.3, 0, 0	74.3, 0, 0
4 (5 4 2)	37.1, 37.1, 0	74, 74, 0
4 (5 8 2)	37.1, 37.1, 0	74, 74, 0
4 (5 16 2)	24.7, 24.7, 24.7	74, 74, 74

For 1 application

Transmission Mode Index (Txn. Mode, Tx. Antenna Count, Rx. Antenna Count)	Throughput for 1 applications (application is from node to UE - Downlink)	Throughput for 1 applications (application is from UE to node - Uplink)
0 (1 1 1)	62.8	74
1 (2 2 2)	62.8	74
2 (3 2 2)	136	74
3 (3 4 2)	136	74
4 (5 4 2)	136	74
4 (5 8 2)	136	74
4 (5 16 2)	136	74

1. **Transmission Mode = 1** uses only one transmit antenna
    1. Here we can see the difference between with relay and without relay. With relay 2nd application is getting transmitted (no zero throughput) but the application doesn't get transmitted (Zero throughput) for 2nd application in case of without relay.
2. **Transmission Mode = 2 → Transmit diversity**
    1. Transmit diversity sends the same information via various antennas
    2. Here all UEs will send the data to Relay, Relay sends the same data (this means data from only one of the UE's due to scheduling) through 2 antennas. So only one application would see a non-zero throughput.
3. **Transmission Mode = 3 → Spatial Multiplexing**
    1. Spatial Multiplexing is used to achieve high data rates
    2. Here data is divided and sends via various antennas
4. **Transmission Mode = 5 → MIMO**
    1. Here data is sent through multiple antennas and received through multiple antennas and hence we can see non-zero throughput for multiple applications

Example 2: Data rate v/s Transmission mode

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \LTE\Data_rate_vs_txn_mode"

Settings done in sample network:

1. Select UDP in all devices in Transport Layer

2. Change all wired link speeds by 10 Gbps
3. Create 1 CBR applications from Wired Node to UE
4. Accept remaining properties as default
5. Simulate for 10 seconds

Transmission mode Index	Data Rate(Mbps)
0	35.1
1	35.1
2	70.2
3	70.2
4 (txn. Antenna count = 4)	70.2
4 (txn. Antenna count = 8)	70.2
4 (txn. Antenna count = 64)	70.2

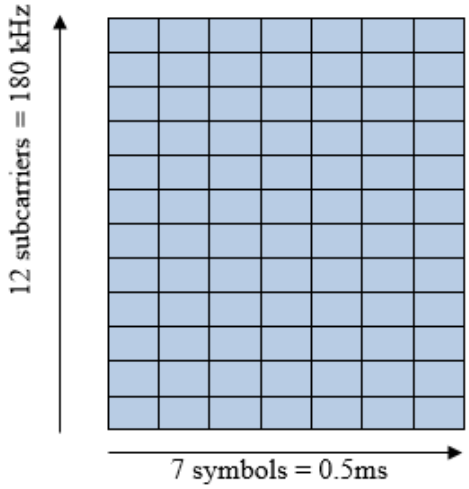
The transmission mode index 0 and 1 follows single transmit antenna and transmit diversity scheme respectively.

The transmission mode index 2, 3, and 4 follows spatial diversity scheme which provides better signal performance and higher data rates. Hence data rates for transmission mode index 2, 3 and 4 are higher compared to transmission mode index 0 and 1.

3.10.7 Physical speed of the LTE Air Interface

The theoretical calculation for the LTE Air Interface is as follows

One Resource block can have 12 subcarriers (each carrier is 15 kHz) in frequency domain and 0.5ms (7 symbols) in time domain.



Thus, total number of symbols per resource block = 12 * 7 = 84

Each symbol can accommodate certain number of bits based on the modulation scheme as per the table below

Modulation scheme	Bits per symbol
QPSK	2
16-QAM	4
64-QAM	6

The table below shows the number of resource blocks available for different LTE channel bandwidths.

Channel Bandwidth (MHz)	5	10	15	20
Resource Blocks (RB)	25	50	75	100
No. of subcarriers (RB*12)	300	600	900	1200
Occupied bandwidth (MHz)	4.5	9	13.5	18

Note: In LTE 10% of total bandwidth is used for guard band. For example if the channel bandwidth is 20MHz, then 2MHz is used for guard band. Thus, if 180 kHz has 1 RB, 18 MHz will have 100 RBs

Given below is the calculation for PHY rate:

In LTE for 20MHz, there are 100 Resource blocks and each Resource block has 12*7 = 84 symbols

Example: PHY rate calculation for 20MHz band, using 64-QAM and 4*4 Tx Rx antennae

- For 20 MHz there are 100 Resource blocks
- Each resource block has 12*7 = 84 symbols
- 100 resource blocks have 8400 symbols
- 1 sub frame = 1 ms = 2 time slots
- 16800 symbols per subframe (or per ms)
- 64-QAM can transmit 6 bits per symbol
- 1 subframe using 64-QAM modulation can transmit 100800 bits/ms

- 100.8*10^6 bits per second or 100.6 Mbps
- This is for a 1*1 Tx Rx antenna and for 4*4 Tx Rx antennae 100.8*4 = 403.2 Mbps

The above note explains the theoretical method of calculating the LTE PHY Data rate, where there are no channel (propagation) losses.

However in the real world there is signal attenuation due to propagation losses. Thus the calculation for PHY Data rate in NetSim is based on the Transport Block Size which is specified in the standard. This calculation is as follows

i. Any signal received at the receiver has a SNR (signal to noise ratio).
ii. Based on the SNR a CQI value is calculated.
iii. The SNR - CQI Table is available in LTE.h in NetSim and is per the LTE standard
iv. Based on the SNR and the CQI an MCS value is calculated
v. The SNR CQI MCS table is available in LTE.h in NetSim and is per the LTE standard
vi. Based on the MCS the TBS Index is calculated, again from a table available in LTE.h which is per the LTE Standard
vii. Based on the TBS Index the TBS Table is looked up and the transport block size is retrieved.

Approximately 25% of overhead is used for controlling and signalling. Effective PHY data rate is 300 Mbps.

3.11 VANETs

3.11.1 Introduction

802.11p has been developed as amendment to IEEE 802.11 standard specifications in order to support ad-hoc communication between vehicles and between vehicle and infrastructure network. There are changes made to PHY (Physical) and MAC layers for the same.

IEEE 802.11p is also known by names such as WAVE (Wireless Access for Vehicular Environments) and DSRC (Dedicated Short Range Communication).

The main objective of 802.11p compliant devices is to improve traffic efficiency and have safety in the traffic flow (i.e. to prevent accidents). The network formed by 802.11p compliant devices is known as VANET.

3.11.2 IEEE802.11p DSRC/WAVE Protocol Stack:

It consists of IEEE 1609 standard and IEEE 802.11p standard. 802.11p standard defines PHY and MAC layers while upper layers are defined by IEEE1609.

Following are the functions performed by each of the layers in IEEE 802.11p protocol stack:

- IEEE 1609-2 defines security services for application messages and management messages in WAVE.
- IEEE 1609-3 defines connection set up and management of WAVE compliant devices.
- IEEE 1609-4 sits on top of 802.11p layers. It enables upper layer operational aspects across multiple channels without knowledge of Physical layer parameters.
- IEEE 802.11p PHY layer takes care of modulation/demodulation, error correction technique etc. Physical layer has been changed. It supports 10 MHz bandwidth, improved performance in WAVE compliant receiver and improvement in the power transmission mask.
- IEEE 802.11p MAC layer takes care of messages to establish and maintain connection in harse vehicular environment. It also defines signalling techniques and interface functions. Stations communicate directly without need to communicate or join with BSS in 802.11p.

3.11.3 Implementation of 802.11p protocol in NetSim

- 802.11p protocol stack covering physical layer and mac layer
- 802.11p is also known as WAVE or DSRC
- FCC has allocated spectrum having 75 MHz bandwidth from 5850 to 5925 MHz for vehicle to vehicle and vehicle to infrastructure communication.
- Supports bandwidth of 10 MHz instead of 20MHz used in 802.11a.
- It supports half of the bit rates as compare to 802.11a i.e. 3/4.5/6/9/12/18/24/27 Mbps
- Transmission type - OFDM
- Slot time - 9µs
- SIFS - 16µs

There are two types of channels in DSRC: CCH and SCH

- **Control channel (CCH):** A single radio channel, not a service channel, intended for the exchange of management information, including Wireless Access in Vehicular Environments (WAVE), Service Advertisements, and WAVE Short Messages.
- **Service channel (SCH):** Any channel that is not the control channel, intended for management frames and higher layer information exchanges (Wireless Access in Vehicular Environments [WAVE] Short Message [WSMs].
- **Guard interval:** A time interval at the start of each control channel (CCH) interval and service channel (SCH) interval during which devices that are switching channels do not transmit.
- **BSM Application:**
  - DSRC protocol runs with BSM (Basic Safety Message) applications
  - BSM is a broadcast packet transmitted regularly at a regular interval, and it can be classified as a beacon style transmission.
  - Broadcast packet is a packet destined for all nodes (vehicles) to receive, a beacon is a continuous broadcast.
  - 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.

Working of 802.11p protocol:

- IEEE 802.11p uses a Medium Access Control (MAC) protocol based on the Carrier Sense Multiple Access protocol with Collision Avoidance (CSMA/CA).
- This means that when a node wants to send a message, the channel has to be idle for a duration of SIFS. If the channel is idle it starts transmission.
- When it finds the channel busy, it chooses a random backoff time from the interval [0, CW] and transmits only when the backoff timer has elapsed.
- The variable CW represents the size of the Contention Window.
- When the SCH is used and a node does not receive an acknowledgement for a message, it concludes that the message has collided and is lost, so the value of CW is doubled and it will retry transmission.
- In the CCH however, beacons are broadcast in the channel and no acknowledgments are sent. This means that the value of CW is never doubled in the CCH.

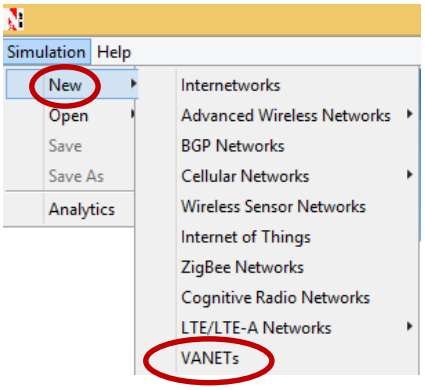
3.11.4 Introduction to NetSim – SUMO interfacing for VANET simulation

- NetSim's VANET module allows users to interface with SUMO which is an open source road traffic simulation package designed to handle vehicular & road networks.
- The road traffic simulation is done by SUMO while NetSim does the network simulation along with RF propagation modelling in the physical layer.
- While SUMO Simulates the road traffic conditions and movements, NetSim Simulates the communication occurring between the Vehicles.
- NetSim and SUMO are interfaced using 'pipes'. A pipe is a section of shared memory that processes use for communication. SUMO process writes information to pipe, then NetSim process reads the information from pipe.
- On running the Simulation, SUMO determines the positions of vehicles with respect to time as per the road conditions. NetSim reads the coordinates of vehicles from SUMO (through pipe) during runtime and uses it as input for vehicles mobility.
- Users will notice an inversion along X axis in the NetSim GUI, since origin (0, 0) in SUMO is at the left bottom, while origin is at the left-top in NetSim.

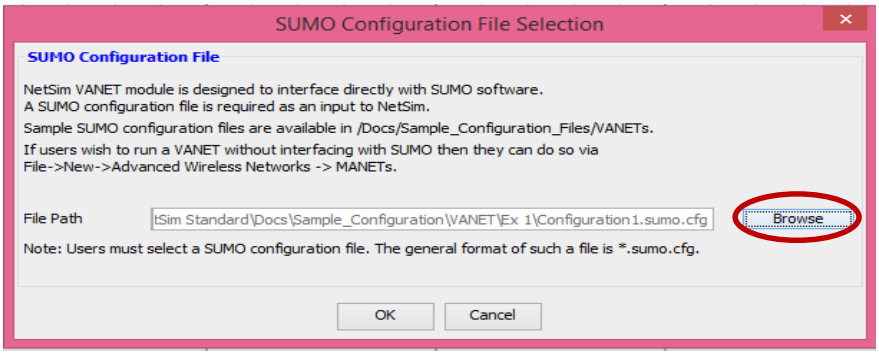
- VANET operates in wireless environment and hence RF channel loss occurs. The amount of loss can be configured by users. To modify the Wireless channel characteristics users can right click on the grid environment and choose either line of sight, fading and shadowing or fading only as per the requirement.

3.11.4.1 New Experiment

- In the **Simulation** menu select → **New →VANETs**



- A dialogue box appears as shown below, in that browse the Sumo Configuration File path. The general format of such file is "*.Sumo.cfg".



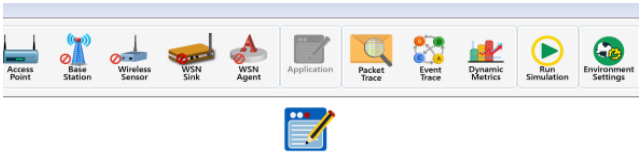
- NetSim VANET module is designed to interface directly with SUMO.
- A SUMO configuration file is required as an input to NetSim.
- Sample SUMO configuration files are available inside ""C:\Program Files\NetSim Standard\Docs\Sample_Configuration\VANET" folder.
- If users wish to run VANET without interfacing with SUMO then they can do so via File → New → Advanced Wireless networks → MANET.
- Users can either use a Sumo configuration file which is provided inside NetSim's installation directory or use a different user specified SUMO configuration file. This .cfg file contains the path of NETWORK file and VEHICLES file.
- Further help on how to create SUMO configuration files is available at http://sumo.dlr.de/wiki/Networks/Building_Networks_from_own_XML-descriptions.

3.11.4.2 Create Scenario

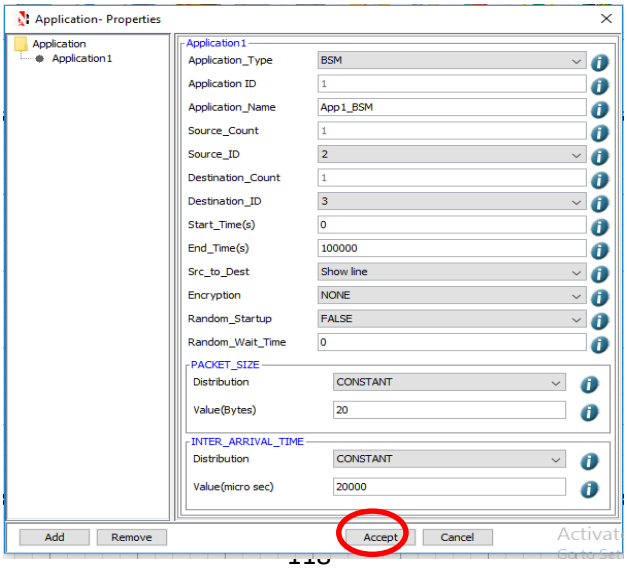
After selecting the Sumo configuration file name, the scenario is opened, with nodes placed at their respective starting positions (tracked form Sumo). Roads and Traffic Lights are also placed exactly as present in SUMO Configuration file.

3.11.4.3 Set Node, Link and Application Properties

- Right click on the appropriate node or link and select Properties. Then modify the parameters according to the requirements.
- Routing Protocol in Network Layer and all user editable properties in Data Link Layer, Physical Layer and Power are Global.
- In the Global properties, Mobility Model is set to SUMO and it is non-Editable. This signifies that the Node movements will be traced from SUMO.
- File name gives the path to Sumo Configuration file that was given by the user.
- Step Size is taken from the Sumo Configuration file specified which tells the amount of time paused in sumo corresponding to single step of SUMO Simulation.
- In interface_wireless properties, under Physical layer, by default Standard is set to IEEE 802.11p in case of VANET.
- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.



- Set the values according to requirement and click Accept.

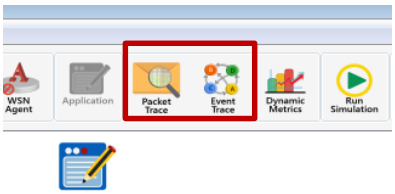


3.11.4.4 Modifying/Viewing/Accepting Properties

On opening an already configured properties of environment, the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

3.11.4.5 Enable Packet Trace, Event Trace (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively.

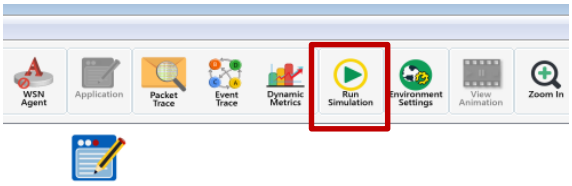


3.11.5 Run Simulation

Click on **Run Simulation** icon on the top toolbar.

Simulation Time is set from the Configuration File of Sumo. The simulation has three options

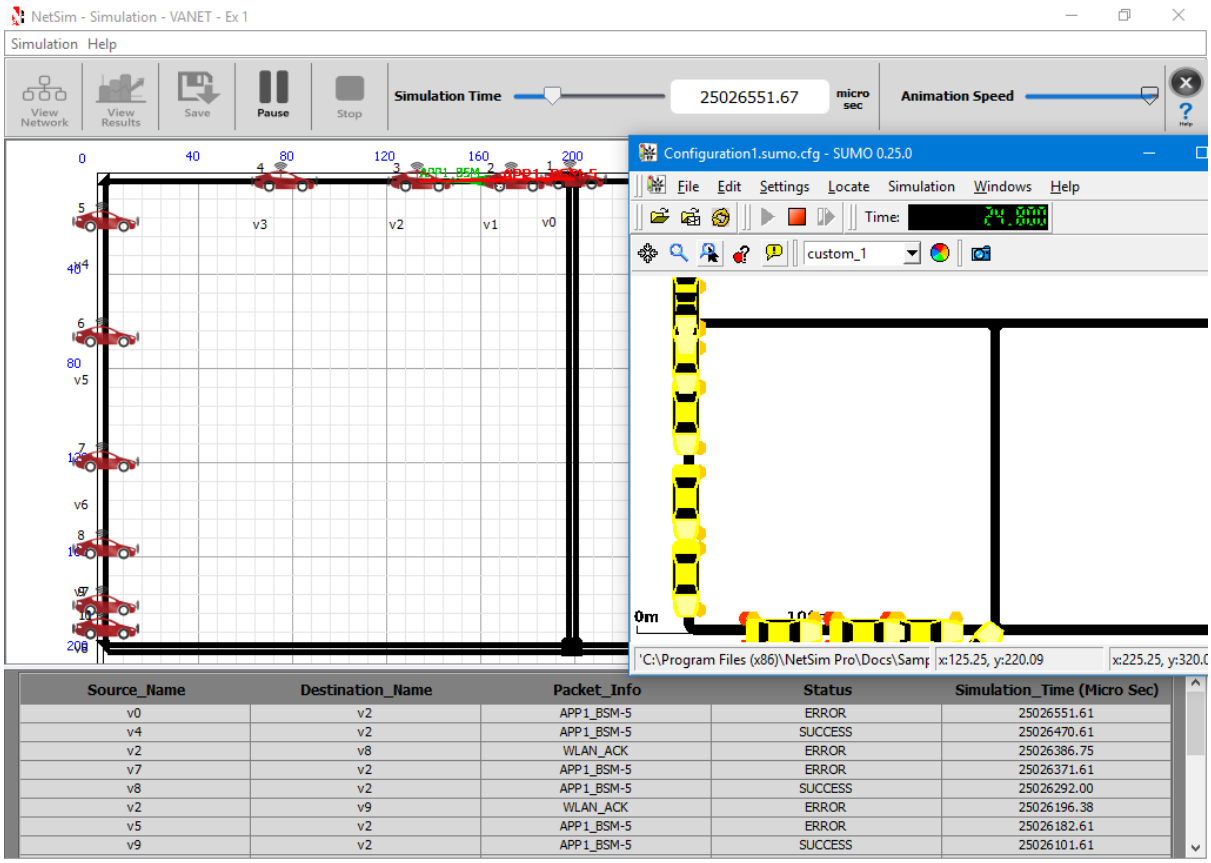
- **Record only** - which runs Sumo in background. Users can view animation after completion of Simulation.



- **Play & Record** – Opens NetSim GUI and Sumo GUI in parallel with parameters being continuously passed between the two Simulators.
- **Don't record** – runs Sumo in Backend. Animation is not recorded.

On running the Simulation by selecting **Play & Record** option, users can view NetSim Packet animation and SUMO simulation simultaneously.

SUMO determines the positions of vehicles with respect to time as per the road conditions. NetSim reads the coordinates of vehicles from SUMO (through pipe) during runtime and uses it as input for vehicles mobility.



Source_Name	Destination_Name	Packet_Info	Status	Simulation_Time (Micro Sec)
v0	v2	APP1_BSM-5	ERROR	25026551.61
v4	v2	APP1_BSM-5	SUCCESS	25026470.61
v2	v8	WLAN_ACK	ERROR	25026386.75
v7	v2	APP1_BSM-5	ERROR	25026371.61
v8	v2	APP1_BSM-5	SUCCESS	25026292.00
v2	v9	WLAN_ACK	ERROR	25026196.38
v5	v2	APP1_BSM-5	ERROR	25026182.61
v9	v2	APP1_BSM-5	SUCCESS	25026101.61

Animation is also synchronized to match the simulation speed in SUMO and NetSim as shown in above figure. Users can see the movement of vehicles in SUMO and observe equivalent movement in NetSim.

Here users can notice an inversion of nodes in the GUI, since origin (0, 0) in SUMO is at the left bottom, while origin is at the left-top in NetSim.

But when users select **Record only** option, NetSim and SUMO run separately and also users can find that the animation in SUMO is much faster than that of NetSim. This is because, NetSim has to animate the flow of packets between the vehicles in addition to the vehicle movement.

Features of VANET implematation in NetSim

- IEEE 802.11p
- Layer 3 Routing – AODV,DSR,OLSR,ZRP
- PHY Layer RF Propagation
    - Pathloss(log distance)
    - Shadowing
    - Fading
- Source C Code
- Automatic import of road network and vehicles from SUMO
- Wide range of output metrics including Delay, Throughput, Error, Retransmission, etc.
- Interfacing between SUMO & NetSim via Traffic control interface (TraCI).

Source C Code:

- Source code related to interfacing of SUMO and NetSim is available in Sumo_interface.c file inside mobility folder/project.
- VANET runs MANET routing protocols in Layer 3. These are AODV, DSR, OLSR, and ZRP.
- The following dlls are executed when VANET simulation run:-
    - libMobility.dll
    - libAODV.dll (or) libDSR.dll (or) libZRP.dll
    - libIEEE802.11.dll
- IEEE 1609
- DSRC – J2735

3.11.6 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

VANET – Example Simulations:

Example 1: CCH

Open the scenario available in "<NetSim Install Dir>\Docs\Sample_Configuration \VANET\EX2".

Settings to be done in the sample network:

- In Interface1_ wireless->Data Link Layer, CCH_TIME_INTERVEL - 30000micro seconds.
- No. of Lanes – 3
- No. of Vehicles - 3
- SCH_time – 100000 microseconds by default.
- Guard_interval – 100ms by default
- Application is set as BSM with 3Mbps generation rate.
    - Packet size – 100
    - IAT - 266
- Run simulation and note down the number of data packets and throughput for sample 1.
- Modify the CCH_TIME_INTERVEL to 50000, 70000 microseconds for sample 2 and 3.

Output		
CCH_Time (micro seconds)	**No.of Packet transmitted (Basic Safety Messages)**	**Throughput (Mbps)**
30000	198837	0.913
50000	218437	1.005
70000	225687	1.03

Inference:

As we are increase the CCH time, throughput increases. Larger the service time greater the throughput.

Here, the delivered packets are considered to be the safety type, which are simulated for the CCH interval -> 30000 µµ s, 50 ms, 70ms.

To analyse this users can open packet trace and filter the data packets. Number of safety messages. i.e BSM messages are more when the CCH time is increased.

In DSRC protocol, the duration is dynamic, and the channel can be adjusted dynamically to transmit the packets.

Example 2: Guard Interval

Guard interval: A time interval at the start of each control channel (CCH) interval and service channel (SCH) interval during which devices that are switching channels do not transmit.

- Goto the path "<NetSim Install Dir>\Docs\Sample_Configuration \VANET\EX1" and open "Configuration1.sumo" in notepad. Change the simulation time to 10 seconds and save.
- Open the scenario inside "<NetSim Install Dir>\Docs\Sample_Configuration \VANET\EX1"

Settings done in the sample network:

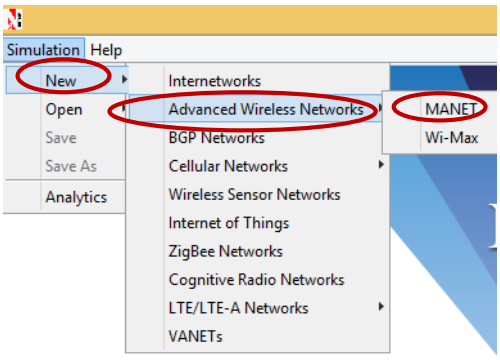
- In Interface1_ wireless ->Data Link Layer, CCH_TIME_INTERVEL - 100000micro seconds by default.
- SCH_time – 100000 microseconds by default.
- Guard_interval – 100microseconds by default.
- Generate traffic with Node 2 to Node 3 with BSM application.
- Change the CCH_TIME_INTERVEL and SCH_TIME_INTERVEL to 10 ms i.e. 10000 us in GUI.
- Click and drop application icon and have to create one application from node 2 to node 3.
- Simulate for 10 sec that is provided by user in "Configuration1.sumo" file.
- Increase the guard interval by 200, 300 and 400 micro seconds will result in decrease in throughput.

Inference:

As we are increase the Guard_interval time, throughput decreases. If Guard interval is more, service time will be less which leads to decrease in throughput.

3.12 Military Radio – TDMA link 16

3.12.1 New Experiment

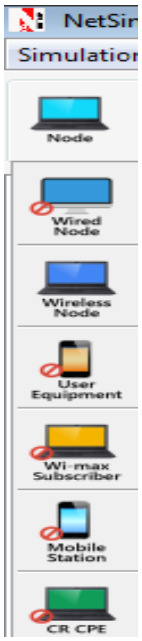


In the **Simulation** menu,

Select → **New** → **Advanced Wireless Networks** → **MANET**

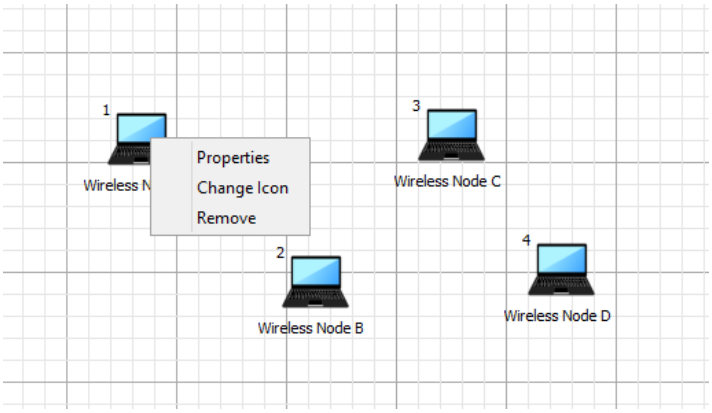
3.12.2 Create Scenario

Click on the **Node** icon in the Toolbar, and then click on **Wireless Node**. Next, click on the environment where you want to drop it inside the **grid**. (Note: A Node cannot be placed on another Node. A Node cannot float outside of the grid.)

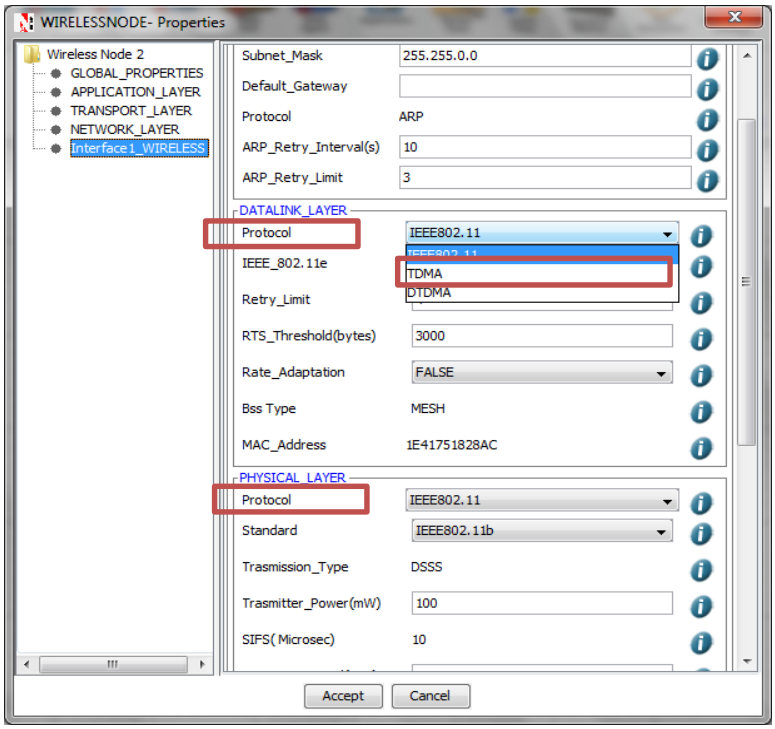


3.12.3 Set Node Properties

- **Right click** on the appropriate node to select Properties.

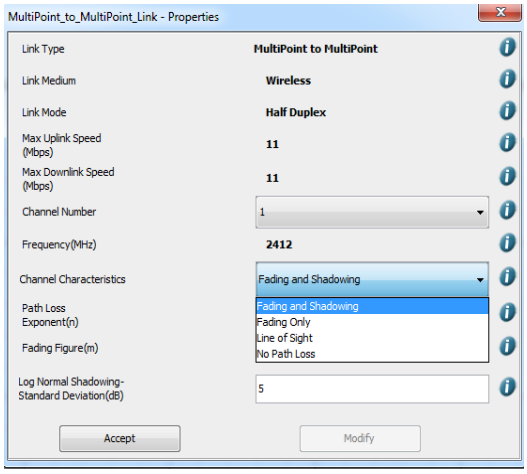


- In **Interface1_Wireless**, go to **DATALINK_LAYER** and **PHYSICAL_LAYER** section and change the **Protocol** to **TDMA.** In Wireless Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer, Physical Layer and Power are Global i.e. changing properties in one node will automatically reflect in the others in that network.
- In **Interface1_Wireless** properties, under network layer, Link layer ack should be selected as “Network LayerAck” .



3.12.4 Set Environment Properties

- **Right click** anywhere on the Environment Grid and select Properties.
- Select the **Channel Characteristics** and set the parameters accordingly.

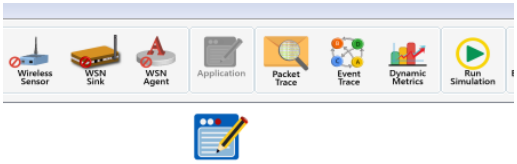


3.12.5 Modifying/Viewing/Accepting Properties

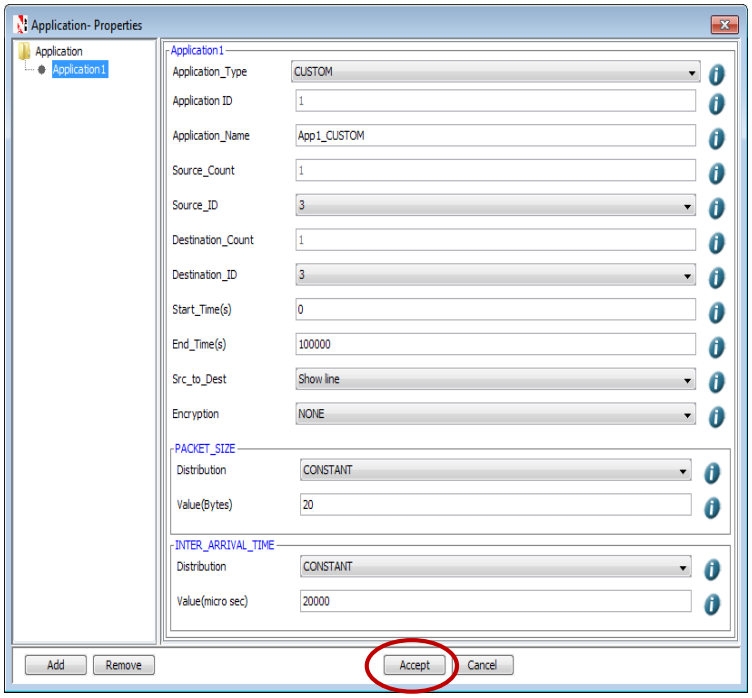
On opening an already configured properties of environment, the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

3.12.6 Set Application Properties

- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.



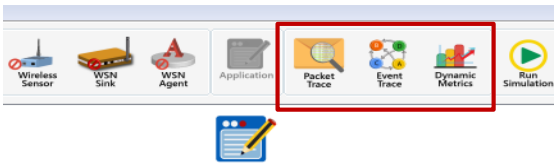
- Set the values according to requirement and click Accept.



Note: Maximum Packet Size in TDMA is 48 bytes.

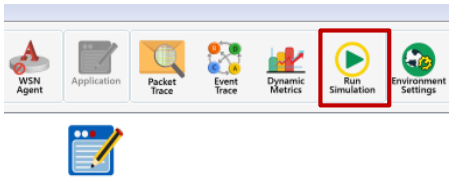
3.12.7 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.12.8 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

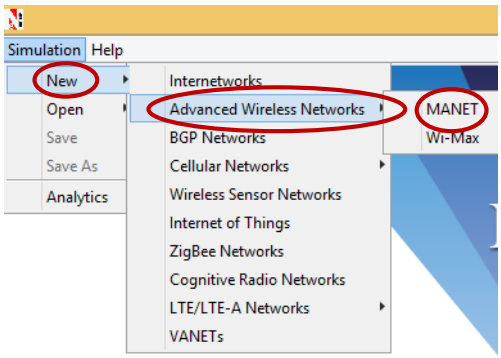
3.13 Military Radio – DTDMA

(**Note**: DTDMA is featured as part of the MAC and PHY layer of MANET radios)

3.13.1 New Experiment

In the **Simulation** menu,

Select → **New** → **Advanced Wireless Networks** → **MANET**



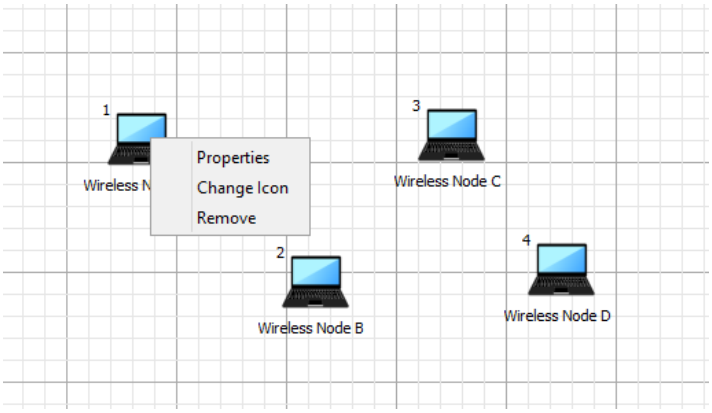
3.13.2 Create Scenario

Click on the **Node** icon in the Toolbar, and then click on **Wireless Node**. Next, click on the environment where you want to drop it inside the **grid**. (Note: A Node cannot be placed on another Node. A Node cannot float outside of the grid.)



3.13.3 Set Node Properties

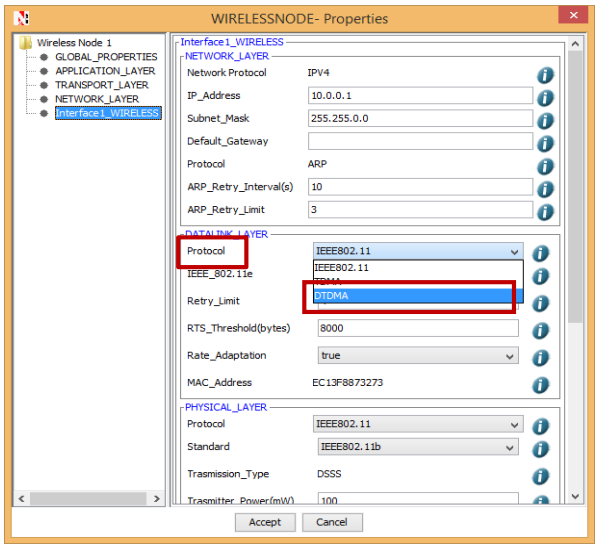
- **Right click** on the appropriate node to select Properties.



- In **Interface1_Wireless**, go to **DATALINK_LAYER** and **PHYSICAL_LAYER** section and change the **Protocol** to **DTDMA.**

- Note that in a Wireless Node, Routing Protocol in Network Layer and all user editable properties in DataLink Layer / Physical Layer / Power model are Global i.e. changing properties in one node will automatically reflect in the others in that network.

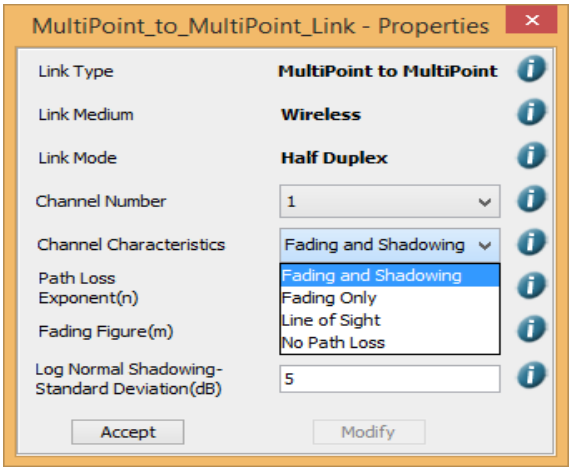
In the Datalink Layer/Physical Layer, we can select the DTDMA protocol as shown in figure below.



Furthermore, in Physical layer, we can select the frequency bands (**HF/VHF/UHF**). Users can modify the **lower frequency** range and the **Bandwidth**. The sum of the Lower frequency and Bandwidth gives the Upper frequency. Users can also select the modulation techniques such as **QPSK/16-QAM/64-QAM** and, an option to turn ON/OFF frequency hopping is also provided.

3.13.4 Set Environment Properties

- **Right click** anywhere on the Environment Grid and select Properties.
- Select the **Channel Characteristics** and set the parameters accordingly.

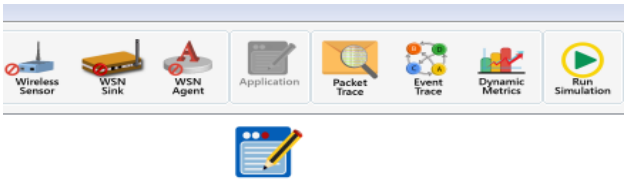


3.13.5 Modifying/Viewing/Accepting Properties

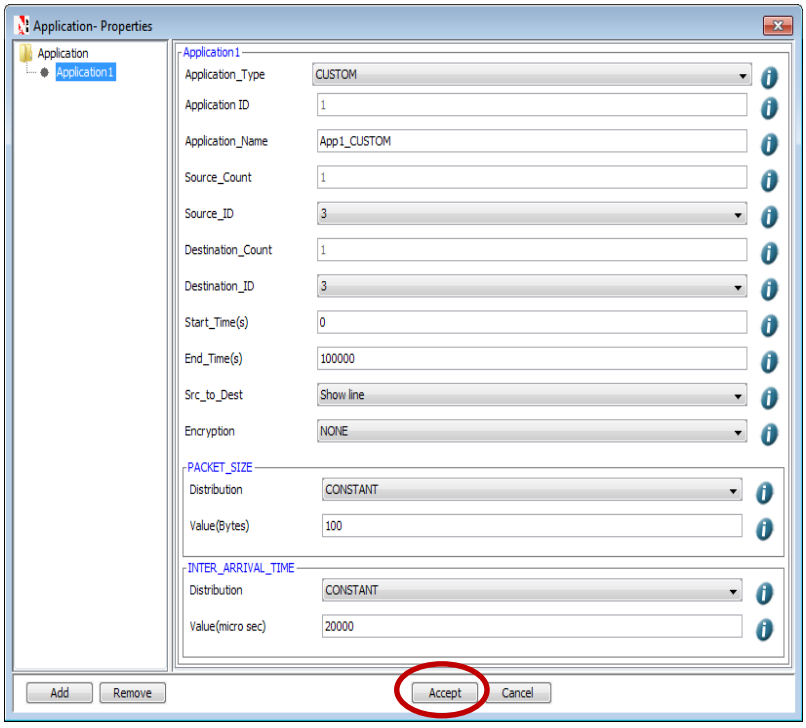
On opening an already configured properties of environment, the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

3.13.6 Set Application Properties

- Select the Application Button on the ribbon and click on the empty region between the Grid Environment and the ribbon. Now right click on Application and select Properties. Multiple applications can be generated by using add button in Application properties.

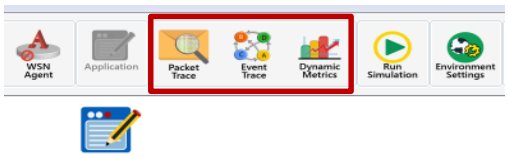


- Set the values according to requirement and click Accept.



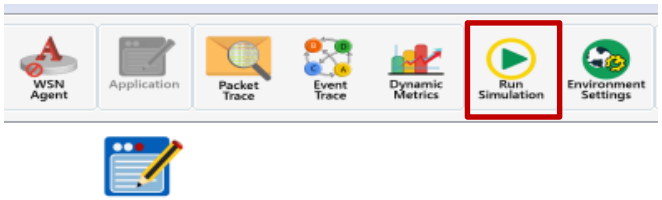
3.13.7 Enable Packet Trace, Event Trace & Dynamic Metrics (Optional)

Click Packet Trace / Event Trace icon in the tool bar. To get detailed help, please refer section 6.4 and 6.5 respectively. Select Dynamic Metrics icon for enabling Dynamic Metrics and click OK.



3.13.8 Run Simulation

Click on **Run Simulation** icon on the top toolbar.



Set the Simulation Time and click on Simulate.

3.13.1 Example Configuration files in NetSim

Sample configuration files for all networks are available inside the folder "<NetSim Install Dir>\Docs\ Sample_Configuration". These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

DTDMA – Example Simulations:

In Time Division Multiple Access (TDMA), each time interval is divided into time slots. Together, all the time slots in the interval are called a "frame". So for example if a network has 6 nodes, then each frame consists of 6 slots. The slot allocation is done in increasing order of Node ID, and in the form of a round robin.

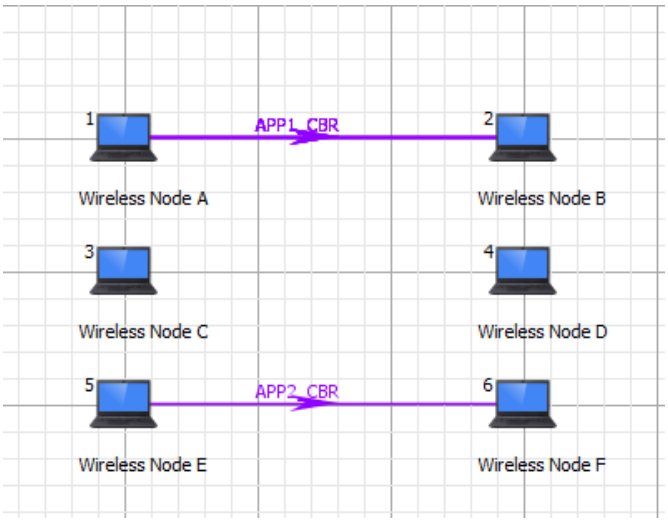
In DTDMA devices can leave / enter the network. For example a network has 5 nodes. Assume that all nodes are present in the network initially. Node 1 uses slot 1, Node 2 uses slot2 and so on till Node 5 uses slot 5. Let us say Node 2 leaves the network, then the frame is split into 4 slots. Node 1 uses slot 1, Node 3 uses slot 2, node 4 uses slot 3 and node 5 uses slot 4.

The above example is based on demand based slots allocation with devices having traffic to send will be allocated slots. Users can also input upto 100 slots per device.

Apart from demand based allocation, round robin slot allocation can also be chosen by the user. In such cases all devices will get 1 slot.

Example 1: Packet_size & Slot allocation analysis

Open the scenario which is available in "<NetSim Install Dir>\Docs\Sample_Configuration \DTDMA\packet_size & slot_allocation"



Settings done in example config file:

In DTDMA, time is divided into slots. In between 2 slots there is a guard interval of 100µs

1) Grid length ->500m * 500m
2) DTDMA -> enabled in MAC and PHY layers
3) Packet size -> 1000Bytes
4) Inter arrival time -> 20000µs
5) Slot Duration -> 2ms
6) Guard interval -> 100µs
7) Bits per slot -> 3000 bits
8) Overhead per slot -> 600 bits
9) Maximum slots per device -> 5
10) Packet trace -> enable
11) Run simulation for 10 seconds

Output:

Users can observe how the slots are allocating for each device in detail in Packet trace. In this case, packet size is 1000 bytes = 1000*8 = 8000 bits and 1 slot is allocated for each device = 1*3000bits = 3000 bits. So, the packet size won't fit in one slot.

Thus fragmentation happens in PHY layer. Users can observe this in Packet trace by filtering the CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR. Packets of any greater size are fragmented. Fragmentation also takes into account the number of bits remaining in the allocated slot to improve slot utilization.

Example: For a slot size of 3000 bits, two 1040Bytes packets will be fragmented as

Slot 1 - 300 bytes

Slot 2 - 300 bytes

Slot 3 - 300 bytes

Slot 4 - 140 + 160 bytes(2nd packet)

Slot 5 - 300 bytes

Slot 6 - 300 bytes

Slot 7 - 280 bytes

…….

	A	B	C	D	E	F	G	H	P	Q	R	S	T	U	V
1	PACKET	SEGME	PACKET	CONTR	SOURC	DESTIN	TRANSI	RECEIV	APP_LA	TRX_LA	NW_LA	MAC_LA	PHY_LA	PHY_LA	PACKET
91	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
101	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
109	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
130	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	140	0	Successful
131	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	160	0	Successful
139	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
149	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
158	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	280	0	Successful
170	3	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
174	3	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
180	3	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	0	Successful
184	3	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	140	0	Successful

In packet trace, filter column of CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR. Node-1 and Node-5 are generating traffic, so time slots are allocated for Node-1 and Node-5as shown below

1	PACKET	SEGME	PACKET	CONTROL_PA	SOURC	DESTIN	TRANSI	RECEIV	APP_LA	TRX_LA	NW_LA	MAC_L	PHY_LA	PH
87	1	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	300	
91	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	
97	1	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	300	
101	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	
105	1	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	300	
109	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	
117	1	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	140	
118	2	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	160	
130	1	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	140	
131	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	160	
135	2	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	300	
139	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	
145	2	0	CBR	APP2_CBR	NODE-5	NODE-6	NODE-5	NODE-6	1000	1000	1008	1040	300	
149	2	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1000	1000	1008	1040	300	

Example 2: ROUND ROBIN

Open the scenario which is available in "<NetSim Install Dir>\Docs\Sample_Configuration \DTDMA\round_robin"

Settings done in example config file:

1) Grid length ->500m * 500m
2) DTDMA -> enabled in MAC and PHY layers
3) Packet size -> 1000Bytes
4) Inter arrival time -> 2000µs
5) Slot Duration -> 2ms
6) Guard interval -> 100µs
7) Bits per slot -> 3000 bits
8) Overhead per slot -> 600 bits
9) Slot allocation technique -> ROUND_ROBIN
10) Packet trace -> enable
11) Run simulation for 10 seconds

Output:

Open packet trace and filter PACKET_TYPE to CBR. ROUND_ROBIN allocates slots to each and every device present in the networkregardless of whether the nodes are generating traffic or not. The slot duration is 2ms = 2000µs. In between 2 slots there is a guard interval of 100µs.In this scenario Nodes 1, 3 and 5 are generating traffic and users can notice that the PHY_LAYER_END time between Source node-1 and Source Node-3 is 42400µs-38200µs = 4200µs = 2slots (including guard interval). So that slot1 belongs to Node-1 and slot2 belongs to Node-2. Observe this in packet trace screenshot given below.

	A	B	C	D	E	F	M	N	O	P	Q	R	S
1	PACKET	SEGME	PACKET	CONTR	SOURCI	DESTIN	PHY_LA	PHY_LAYER_START_TI	PHY_LA	APP_LA	TRX_LA	NW_LA	MAC_L
129	1	0	CBR	APP1_CBR	NODE-1	NODE-2	37800	38200	39800.5	1000	1000	1008	1040
141	1	0	CBR	APP2_CBR	NODE-3	NODE-4	42000	42400	44000.5	1000	1000	1008	1040
145	1	0	CBR	APP3_CBR	NODE-5	NODE-6	46200	46600	48200.5	1000	1000	1008	1040
149	1	0	CBR	APP1_CBR	NODE-1	NODE-2	50400	50800	52400.5	1000	1000	1008	1040
153	1	0	CBR	APP2_CBR	NODE-3	NODE-4	54600	55000	56600.5	1000	1000	1008	1040
165	1	0	CBR	APP3_CBR	NODE-5	NODE-6	58800	59200	60800.5	1000	1000	1008	1040
169	1	0	CBR	APP1_CBR	NODE-1	NODE-2	63000	63400	65000.5	1000	1000	1008	1040
173	1	0	CBR	APP2_CBR	NODE-3	NODE-4	67200	67600	69200.5	1000	1000	1008	1040
177	1	0	CBR	APP3_CBR	NODE-5	NODE-6	71400	71800	73400.5	1000	1000	1008	1040
181	1	0	CBR	APP1_CBR	NODE-1	NODE-2	75600	76000	76747.16	1000	1000	1008	1040
182	2	0	CBR	APP1_CBR	NODE-1	NODE-2	75600	[illegible]	77600.5	1000	1000	1008	1040
186	1	0	CBR	APP2_CBR	NODE-3	NODE-4	79800	80200	80947.17	1000	1000	1008	1040
187	2	0	CBR	APP2_CBR	NODE-3	NODE-4	79800	80946.667	81800.5	1000	1000	1008	1040
191	1	0	CBR	APP3_CBR	NODE-5	NODE-6	84000	84400	85147.17	1000	1000	1008	1040

Example: How Slots are allocated in between 38200µs to71800µsin ROUND_ROBIN slot allocation technique. Yellow colour represents the nodes that are not generating the traffic.

Node ID	Node-1	Node-2	Node-3	Node-4	Node-5	Node-6	Node-1	Node-2
Application ID	CBR-1	N/A	CBR-2	N/A	CBR-3	N/A	CBR-1	N/A
PHY_LAYER_END_TIME	38200 µs	40300 µs	42400 µs	44500 µs	46600 µs	48700 µs	50800 µs	52900 µs

Node-3	Node-4	Node-5	Node-6	Node-1	Node-2	Node-3	Node-4	Node-5
CBR-2	N/A	CBR-3	N/A	CBR-1	N/A	CBR-2	N/A	CBR-3
55000µs	57100µs	59200µs	61300µs	63400µs	65500µs	67600µs	69700µs	71800µs

Compare the above table with packet trace

Example 3: Node join & Node leave

Open the scenario which is available in "<NetSim Install Dir>\Docs\Sample_Configuration \DTDMA\node_join&node_leave"

Node join - It is the time at which the node join the network.

Node leave - It is the time at which the node leaves the network.

Settings done in example config file:

1) DTDMA -> enabled in MAC and PHY layers
2) TCP -> Disable
3) Packet trace -> enable
4) Packet size = 1000 Bytes
5) Inter arrival time = 100000µs
6) Simulate for 10 seconds and save the network
7) In edit and rerun, Node Leave (Node-2) -> 5s (present in global properties)
8) Simulate for 10 seconds

Output:

In case1, Node-1 transmits the packets throughout the simulation time, but in case-2, it will transmit upto 5 seconds since Node-2 left the network at 5th second. To observe this open packet trace and filter PACKET_TYPE to CBR. In the below figure, NODE-1 has transmitted the packets upto 10th second

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	PACKET	SEGME	PACKET	CONTR	SOURC	DESTIN	TRANSI	RECEIV	APP_LA	TRX_LA	NW_LA	MAC_L	PHY_LA	PHY_LA	PHY_LA
280	30	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3000000	3000000	3863107	0	9209200	9215926	9222743
281	31	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3100000	3100000	3863107	0	9209200	9222743	9309201
283	31	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3100000	3100000	3863107	0	9309300	9316026	9323202
284	32	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3200000	3200000	3863107	0	9309300	9323201	9409301
286	32	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3200000	3200000	3863107	0	9409400	9416126	9423661
287	33	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3300000	3300000	3863107	0	9409400	9423660	9509401
289	33	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3300000	3300000	3863107	0	9509500	9516226	9524119
290	34	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3400000	3400000	3863107	0	9509500	9524119	9609501
298	34	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3400000	3400000	3863107	0	9709700	9716426	9724678
299	35	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3500000	3500000	3863107	0	9709700	9724678	9809701
301	35	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3500000	3500000	3863107	0	9809800	9816526	9825137
302	36	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3600000	3600000	3863107	0	9809800	9825136	9909801
304	36	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	3600000	3600000	3863107	0	9909900	9916626	9925596

But in case2 Node-2 is leaving the network at 5th second. So packet transmission will takes place upto 5th second only.

1	PACKET	SEGME	PACKET	CONTR	SOURC	DESTIN	TRANSI	RECEIV	APP_LA	TRX_LA	NW_LA	MAC_L	PHY_LA	PHY_	LAYER_STAR
117	12	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1200000	1200000	3613107	3613107	3703700		3710426.457
119	13	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1300000	1300000	3613107	0	3803800		3810526.457
121	14	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1400000	1400000	3613107	0	3903900		3910626.457
123	15	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1500000	1500000	3613107	0	4004000		4010726.457
125	16	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1600000	1600000	3613107	0	4104100		4110826.457
133	17	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1700000	1700000	3613107	0	4304300		4311026.457
135	18	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1800000	1800000	3613107	0	4404400		4411126.457
137	19	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	1900000	1900000	3613107	0	4504500		4511226.457
139	20	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	2000000	2000000	3613107	0	4604600		4611326.457
141	21	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	2100000	2100000	3613107	0	4704700		4711426.457
149	22	0	CBR	APP1_CBR	NODE-1	NODE-2	NODE-1	NODE-2	2200000	2200000	3613107	0	4904900		4911626.457
150															
151															
152															
153															

3.13.2 DTDMA Packet size

It is important to set packet size (of any application running over DTDMA) to be lower than the Max packet Size setting indicated below. If the packet size exceeds the Max Packet Size setting then DTDMA would not be able to transmit that packet.

$$\text{Maximum Packet size (bytes)} = \frac{Bits\ per\ slot - overhead\ per\ slot - Tx\ Layer\ OH - NW\ layer\ OH}{8}$$

By default,

Bits per slot (bits) – 3000, Overhead per slot (bits) - 600

Users can also edit the values of Bits per slot and Overhead per slot in the GUI.

Assuming default values are chosen for Bits per slot and Overhead per slot, DTDMA packet size is calculated for different protocols as shown below:

i. **For DSR protocol (if TCP is enabled),**

DSR overhead (one hop) - 12 bytes which is added with Network layer overheads, plus IP overhead of 20 plus TCP Overhead of 20, totaling 52 bytes (416 bits).

$$\text{Max Packet size (bytes)} = \frac{Bits\ per\ slot - overhead\ per\ slot - 416}{8} = 248 \text{ bytes}$$

ii. **For DSR protocol (if UDP is enabled),**

DSR overhead (one hop) - 12 bytes which is added with Network layer overheads, plus IP overhead of 20 plus UDP Overhead of 8, totaling 40 bytes (320 bits).

$$\text{Max Packet size (bytes)} = \frac{Bits\ per\ slot - overhead\ per\ slot - 320}{8} = 260 \text{ bytes}$$

iii. **For AODV,ZRP,OLSR protocol (if TCP is enabled),**

Here AODV, ZRP, OLSR overhead – 0 (no overhead is added) plus IP overhead of 20 plus TCP Overhead of 20, totaling 40 bytes (320 bits)

$$\text{Max Packet size (bytes)} = \frac{Bits\ per\ slot - overhead\ per\ slot - 320}{8} = 260 \text{ bytes}$$

iv. **For AODV,ZRP,OLSR protocol (if UDP is enabled),**

Here AODV, ZRP, OLSR overhead – 0 (no overhead is added) plus IP overhead of 20 plus UDP Overhead of 8, totaling 40 bytes (224 bits)

$$\text{Max Packet size (bytes)} = \frac{Bits\ per\ slot - overhead\ per\ slot - 224}{8} = 272 \text{ bytes}$$

3.13.3 Node Join / Leave

Node join(s) - It is the time at which the node join the network and accesses the communication channel.

Node leave (s) – It is the time at which the node leaves the network.

Dynamic metrics will be shown only for the period in which the node is present in the network. For example, if node join = 0, node leave = 5, even if the simulation time = 100s, the dynamic metrics will be shown only for 5s.

Use case:

Fields can take multiple inputs separated by comma as shown below:

Node join - 0, 10

Node leave - 5, 100

In this case, the node joins the network at 0s and leaves at 5s and the node joins the network again at 10s and leaves at 100s.

3.14 Propagation models in NetSim

Propagation models are used to model signal attenuation for all wireless links. These include WLAN – 802.11, Zigbee / IOT / WSN – 802.15.4, LTE / LTE – A and Wi-Max.

3.14.1 Propagation Loss

Three different and mutually independent propagation phenomena influence the power of the received signal: ***path loss, shadowing and multipath fading.***

The different models available in NetSim are

1. Pathloss Models
    a. Friis Free Space Propagation (Default option in GUI)
    b. Log Distance
    c. HATA Suburban
    d. HATA Urban
    e. COST 231 HATA Suburban
    f. COST 231 HATA Urban
    g. Indoor Office
    h. Indoor Factory
    i. Indoor Home
    j. No Path Loss
2. Shadowing Models
    a. No shadowing
    b. Constant
    c. Lognormal
3. Fading Model
    a. No fading
    b. Rayleigh
    c. Nakagami

3.14.2 Path Loss

Path loss is the reduction in power density of an electromagnetic wave as it propagates through space. Path loss may be due to many effects, such as free reflection, aperture-medium coupling loss, and absorption.

The general formula by which pathloss is calculated is

$$RX_{power} = TX_{power} + Gain_{TX} + Gain_{RX} - PL_{1Meter} - 10 \log D^{n}$$

Where *n* is the path loss exponent, whose value is normally in the range of 2 to 5. In NetSim, the default value for path loss exponent is 2.

and D is the distance between transmitter and the receiver, usually measured in meters.

And PL_{1Meter} is the path loss at reference distance (here taken as 1m). This value varies depending on the radio and is a user input available in the PHY layer of the radios. For 802.11b this value is 30dB

And the Gain represents the transmit and receive antenna gains

An example: Let us say the transmit power of a radio is 20 mW (or 13 dBm), with zero gains for the tx and rx antenna, and let us say the distance between transmitter and receiver is 100m and the path loss exponent n = 3. Then Received power in dB is

$$\begin{aligned} RX_{power} &= 10 \log (20) + 0 + 0 - 30 - 10 \log (100)^{3} \\ &= 13 - 30 - 60 \\ &= -77 \text{ dBm} \end{aligned}$$

The default value for reference distance d0 and pathloss at reference distance PL_d0 are

1. 802.11 a / b / g / n / ac / p
    - 2.4 GHz : Default d0 = 1m and PL_D0 = 40dB
    - 5 GHz: Default d0 = 1 m and PL_D0 = 47 dB
2. 802.15.4 - Default d0 = 8m and PL_D0 = 58.5 dB
3. In LTE the calculation is done for each carrier for uplink and download. Default d0 = 1m and PL_D0 = 32 dB

The code for calculating the Path loss is included inside the function propagation_calculate_logdistancepathloss() which is available in the path **NetSim Standard\src\Simulation\IEEE802_11.**

3.14.3 Shadowing

Slow shadowing in wireless network is the attenuation caused by buildings or any obstacles between a transmitter and a receiver. In the model with shadowing, the shadowing value Xσ, typically defined in dB, is added to (or subtracted from) the average received power. *Xσ is a zero means Gaussian distributed random variable with standard deviation σ.*

The Probability Density Function (PDF) of the lognormal distribution is:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma x} \times e^{\left(\frac{(\ln x)^2}{2\sigma^2}\right)}$$

The default value for standard deviation is chosen as 5 dB.

The code for calculating the shadow loss is present inside the function propagation_calculate_lognormalshadow() and the file is available inside **NetSim Standard\src\Simulation\IEEE802_11**.

3.14.4 Fading

In wireless communications, fading is deviation of the attenuation affecting a signal over certain propagation media. The fading may vary with time, geographical position or radio frequency, and is often modelled as a random process.

In NetSim, the Rayleigh Fading, which follows Rayleigh Probability Distribution with mean of 1, is used. The code for calculating fading loss is present in the file IEEE802_11_Phy.c, and path for the file is **NetSim Standard\src\Simulation\IEEE802_11**.

3.14.5 SINR Calculation

Analogous to the SNR used often in wired communications systems, the SINR is defined as the power of a certain signal of interest divided by the sum of the interference power (from all the other interfering signals) and the power of some background noise. The interference power is the difference between the total power received by the receiver and the power received from one particular transmitter.

The background thermal noise in dBm at room temperature is given by:

$$\text{P (in dBm)} = -174 + 10 \times \log_{10}(\Delta f)$$

where Δf is the Bandwidth in Hertz. For 802.15.4, Δf = 2 MHz. For 802.11a, b, g, Δf = 20 MHz, and for 802.11n, Δf = 20 MHz or 40 MHz.

$$\text{P (in mW)} = 10^{\left(\frac{P\ (in dBm)}{10}\right)}$$

Therefore, SINR in dBm is calculated as:

$$\text{SINR (in dBm)} = \log_{10}\left(\frac{\text{Received power (in mW)}}{\text{Interference Noise (in mW)} + \text{Thermal Noise (in mW)}}\right)$$

3.14.6 Bit Error Rate (BER) Calculation

The bit error rate (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval. The BER calculation is calculated using as SNR – BER tables for different modulation schemes. These tables are closed and not available for user modification. In the case of LTE an SNR-BER table is looked up for each MCS.

4 Traffic generator in NetSim (Application Models)

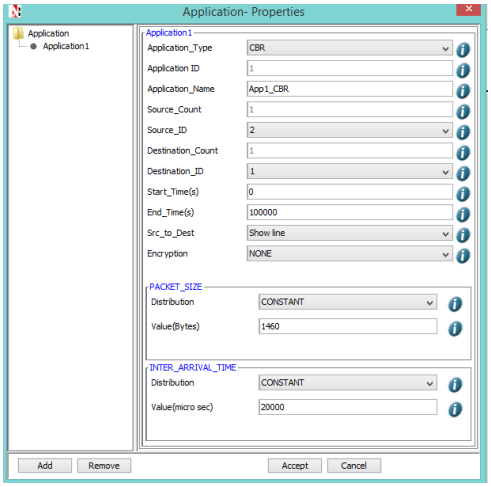
NetSim allows users to model and simulate the applications:

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. Emulator (Only if Emulator Add-on is present)

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



Right click on the application icon and select properties



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

4.1 Common properties for all the traffic types

Application ID

This property represents the unique identification number of the application.

Start time

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

End time

This property represent 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.*

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** application users can select '0' as the Destination ID.

4.2 CBR

Packet Size

Distribution: The options available for distribution is

- **Constant**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 1460 bytes is entered.

Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution is

- **Constant**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 20000 Micro Sec is entered.

4.3 Custom

Packet Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 1460 bytes is entered.

Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 20000 Micro Sec is entered.

4.4 Voice

Codec

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 are given which can be selected depending on the variations required. Packet size and Inter-arrival time value will vary depending on the codec value chosen.

Packet Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 160 bytes is entered.

Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 20000 Micro Sec is entered.

Service Type

- **CBR** - CBR stands for Constant Bit Rate. Packets of constant size are generated at constant inter arrival times.
- **VBR** - VBR stands for Variable Bit Rate. The two types of Suppression Model that can be selected are,
    - **Deterministic**
    - **Markov Chain**

4.5 Video

Video is an electronic medium for the recording, copying and broadcasting of moving visual images.

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 pixelsin 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.

- **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.
  - **Constant A**– First order autoregressive Markov processλ(n) can be generated by the recursive relation λ(n)=aλ(n-1)+bw(n).
  - **Constant B**– First order autoregressive Markov process λ(n) can be generated by the recursive relation λ(n)=a λ(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

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

  $D_k = D_{k-1} - (N^2_{k-1}/D_{k-1})$

  $\Phi_{kk} = N_k / D_k$

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

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

  $y_k = (1 - \Phi^2_{kk})\, y_{k-1}$

  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.

$Yk = F\text{-}1\beta\ (FN(Xk)),\ k>0$

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** – Refer i-button help of Simple IPB Composite Model.
- **Gamma B**– Refer i-button help of Simple IPB Composite Model.
- **Gamma P**– Refer i-button help of Simple IPB Composite Model.
- **Eta I** – Refer i-button help of Simple IPB Composite Model.
- **Eta B** – Refer i-button help of Simple IPB Composite Model.
- **Eta P**– Refer i-button help of Simple IPB Composite Model.
- **Beta I** – Refer i-button help of Simple IPB Composite Model.
- **Beta P**– Refer i-button help of Simple IPB Composite Model.
- **Beta B**– Refer i-button help of Simple IPB Composite Model.

4.6 FTP

File Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

File Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 100000 bytes is entered.

NOTE: Devices must have TCP enabled in Transport layer for implementing FTP application successfully.

File_Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 5 Sec is entered.

4.7 Database

Transaction Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet_Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 10000 bytes is entered.

Transaction Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 1000000 Micro Sec is entered.

4.8 Peer to Peer

A pure 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.

NOTE: Devices must have TCP enabled in Transport layer for implementing peer to peer application successfully.

File Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Value (Bytes): This represents the size of the file in bytes.

Piece size (Bytes): Each file is divided into equal sized pieces and then the piece of the data is transmitted. This property represents the size of each piece.

4.9 HTTP

HTTP (Hyper Text Transfer Protocol) 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.

Http_request_interarrival_time

This indicates the time gap between the pages.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time (micro sec): This represents the rate at which client sends the requests.

Page_property

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Page Count: This represents the number of pages received from the server.

Page Size (Bytes): This represents the size of each page that is received from the server.

4.10 Email

Email_Receive

This represents the rate at which emails are received.

Duration_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Duration: Time gap between two successive receiving emails in seconds.

Email_Size_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Email_Size: It represents the size of the email that is received.

Email_Send

This represents the rate at which emails are sent.

Duration_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Duration: Time gap between two successive sending emails in seconds.

Email_Size_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Email_Size: It represents the size of the email that is sent.

4.11 Sensor App

Packet Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 50 bytes is entered.

Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 1000000 Micro Sec is entered.

4.12 Erlang Call

Packet Size

Distribution: The options available for distribution is

- **Constant**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 160 bytes is entered.

Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution is

- **Constant**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and vice versa. By default 20000 Micro Sec is entered.

Call

Duration Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Duration: The options available

Inter Arrival Time: Enter the average inter-arrival time between calls. A lower inter-arrival time would lead to a higher call rate and vice versa. By default 60 Sec is entered.

IAT Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Service Type:

- **CBR** - CBR stands for Constant Bit Rate. Packets of constant size are generated at constant inter arrival times.
- **VBR** - VBR stands for Variable Bit Rate. The two types of Suppression Model that can be selected are,
    - **Deterministic**
    - **Markov Chain**

Success ratio: Sets the ratio of the packets that are not silenced during VBR calls.

4.13 BSM

NOTE- Will be present only in VANETs

Duration Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 20 bytes is entered.

Inter Arrival Time:

This indicates the time available for distribution is **Constant** gap between packets.

- **Distribution:** The options available for distribution is

4.14 Emulator

NOTE- Will be present only when Emulator Add-on is installed

Emulation

Source Real IP: Specifies the real IP Address of source device in Emulation

Source Port: Specifies the Port no used for transmission by Application running in source device

Destination Real IP: Specifies the real IP Address of destination device in Emulation

Destination Port: Specifies the Port no used for reception by Application running in destination device

4.15 Priority and QoS of Applications

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

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

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

4.16 Modelling Poisson arrivals in NetSim

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

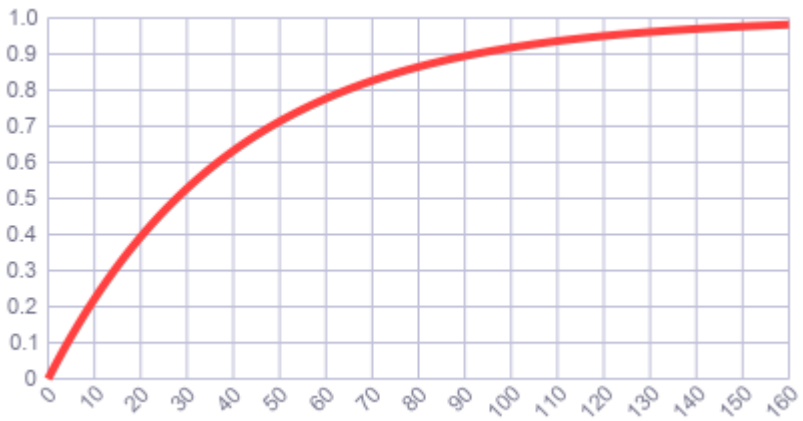
Any time you have events which occur individually at random moments, but which tend to occur at an average rate when viewed as a group, you have a Poisson process.

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 $\lambda = 1/ 0.05 = 20$ 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.

5 Running simulation via CLI

5.1 Running NetSim via CLI

Note: *This feature is available from v7 onwards. In v8, NetSim can be run via CLI for Internetworks, Advanced Wireless Networks – MANET, Wireless Sensor Networks, Internet of Things, ZigBee Networks, BGP Networks, Cellular Networks, LTE Networks and Cognitive Radio Networks only. For other networks NetSim v8 can only be run via GUI. In v9, NetSim can be run via CLI for VANET, Wi-max, LTE/LTE-A, Legacy networks.*

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 App Path

App path is the file location in the system in which NetSim has been installed.

The app path can be found out by right clicking the NetSim Shortcut in the desktop and selecting Open file location (Windows 7/8/10). The app path will be something like "C:\Program Files\NetSim Standard\bin", depending on where NetSim is installed.

Note: The path to the bin folder of the app path must be mentioned

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.xml*** (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.xml 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.xml can be placed inside that folder.

> ***Note:*** *Sample configuration.xml 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:\Program Files (x86)\NetSim Standard\bin"

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

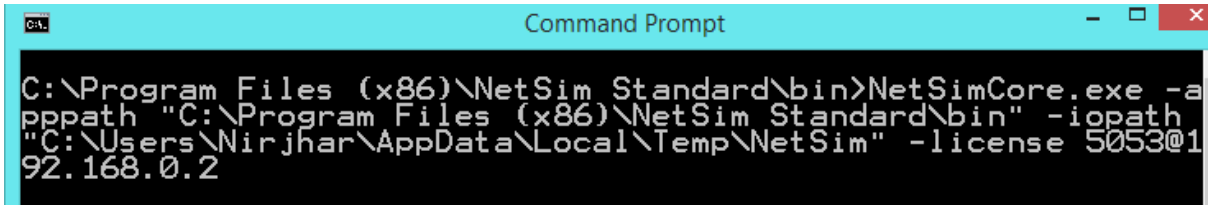
```
>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>-license<space>5053@<Server IP Address>
```

Where,

- <app path> contains all files of NetSim including NetSimCore.exe
- <iopath> contains Configuration.xml and Configuration.xsd (Configuration.xsd is available in bin folder). Refer section 5.2.4 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.xml 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.xml 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

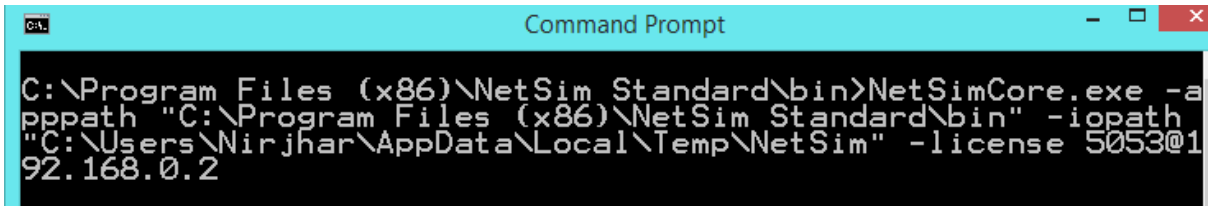
***

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>

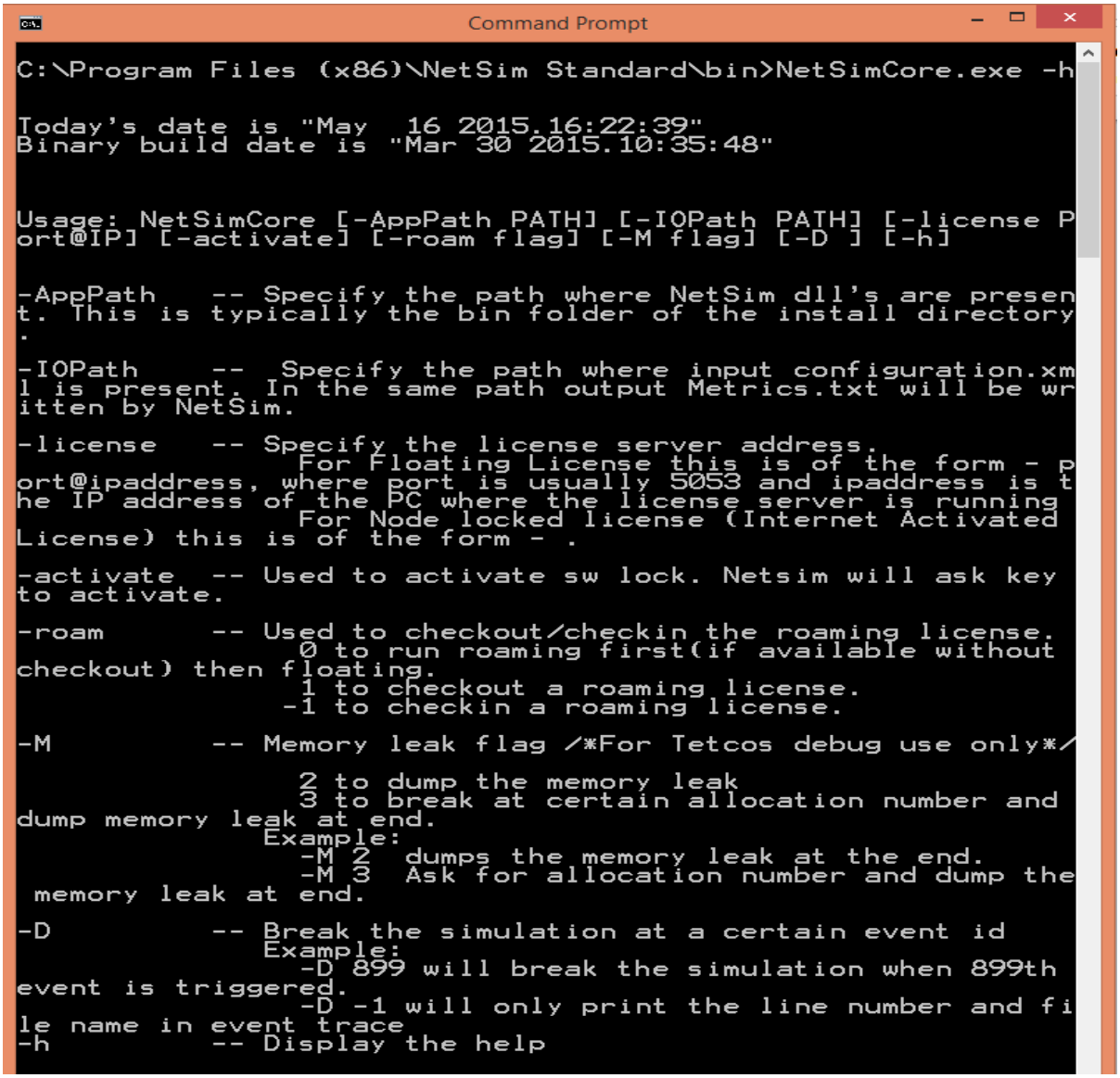
>NetSimCore.exe -apppath <"app path"> -iopath <io path>-license 5053@<License Server IP Address>



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



Note:

5.1.1 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.

5.2 Understanding Configuration.xml file

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

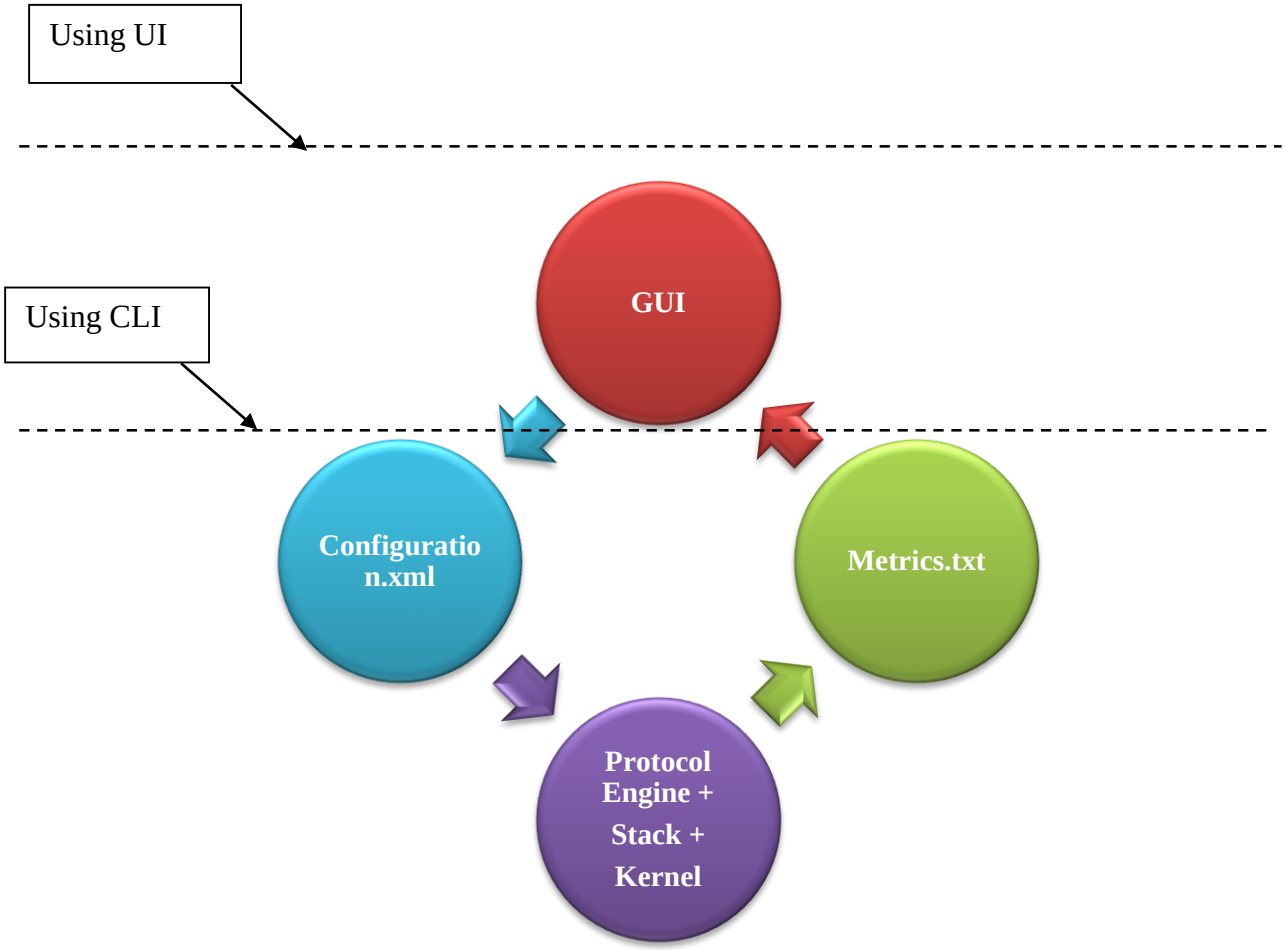


Fig.1. NetSim Architectural Overview

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.xml*** just when the user performs the simulation.

The back-end engine that contains dlls and NetSimCore.exe will read this Configuration.xml, 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.xml furnishing all the details about the devices, links and the environment of the desired scenario.

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

To edit the Configuration.xml, xml editor is required. There are various xml editors available in the market. Visual Studio 2010 is one of the xml editors that can be used to edit the Configuration.xml with efficacy.

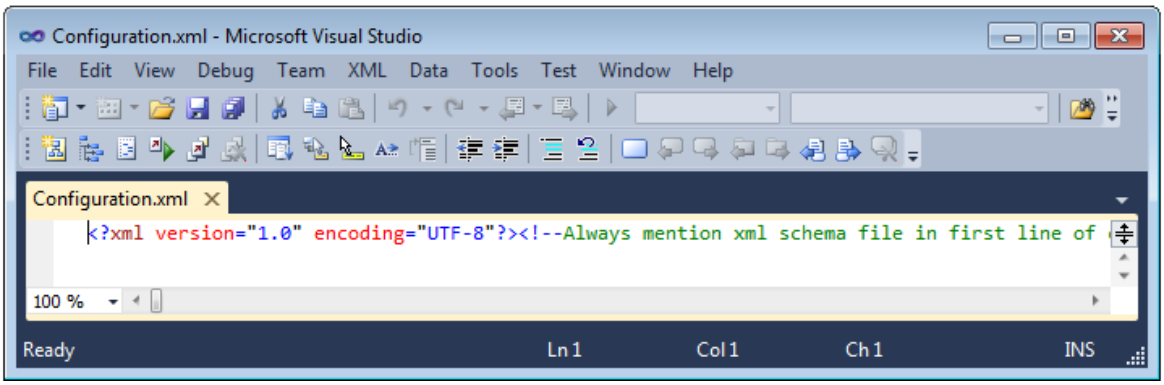
This section shows limelight on how to use Visual Studio to edit the Configuration.xml.

XML view provides an editor for editing raw XML and provides *IntelliSense* and *color coding*.

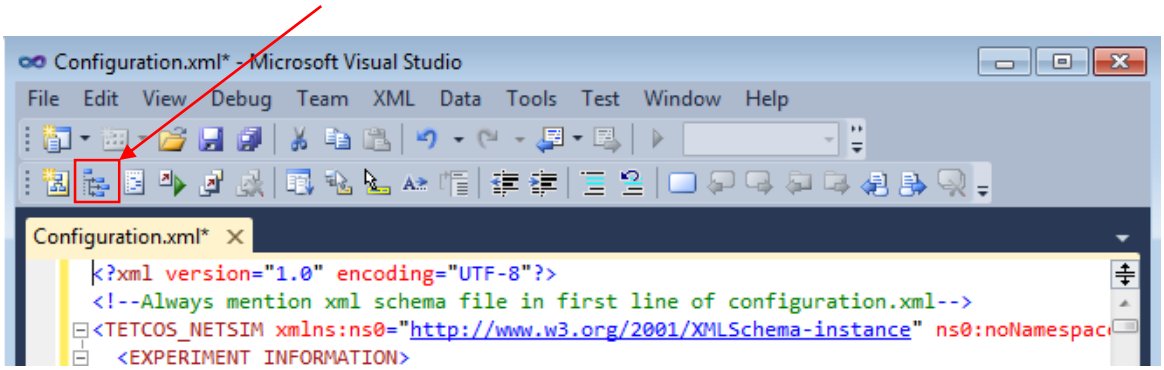
After you type the element name and press the CTRL+ SPACE, you will be presented with a list of attributes that the element supports. This is known as “IntelliSense”. Using this feature, you can select the options that are required to create the desired scenario.

Color coding is followed to indicate the elements and the attributes in a unique fashion.

The following screenshot displays the Configuration.xml which is opened through the Visual Studio.



To reformat it, click on “Reformat Selection” icon.



5.2.2 Sections of Configuration file

These are the different sections in Configuration.xml:

- 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.xml.

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.

5.2.3 Sample Configuration file

Sample "Configuration.xml" 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 or any XML editor. The purpose of providing the sample "Configuration.xml" file is to assist the user in writing a network scenario manually by analyzing the format for that specific network technology.

5.2.4 Configuration.xsd file

Configuration.xsd is a XML schema Definition file which is present in the path <NetSim_Installation_Directory>/bin

Configuration.xsd file can be placed inside the <iopath> along with the configuration.xml file to verify the contents in the configuration.xml file. This file checks and validates the structure and vocabulary of a configuration.xml document against the grammatical rules of the appropriate XML language.

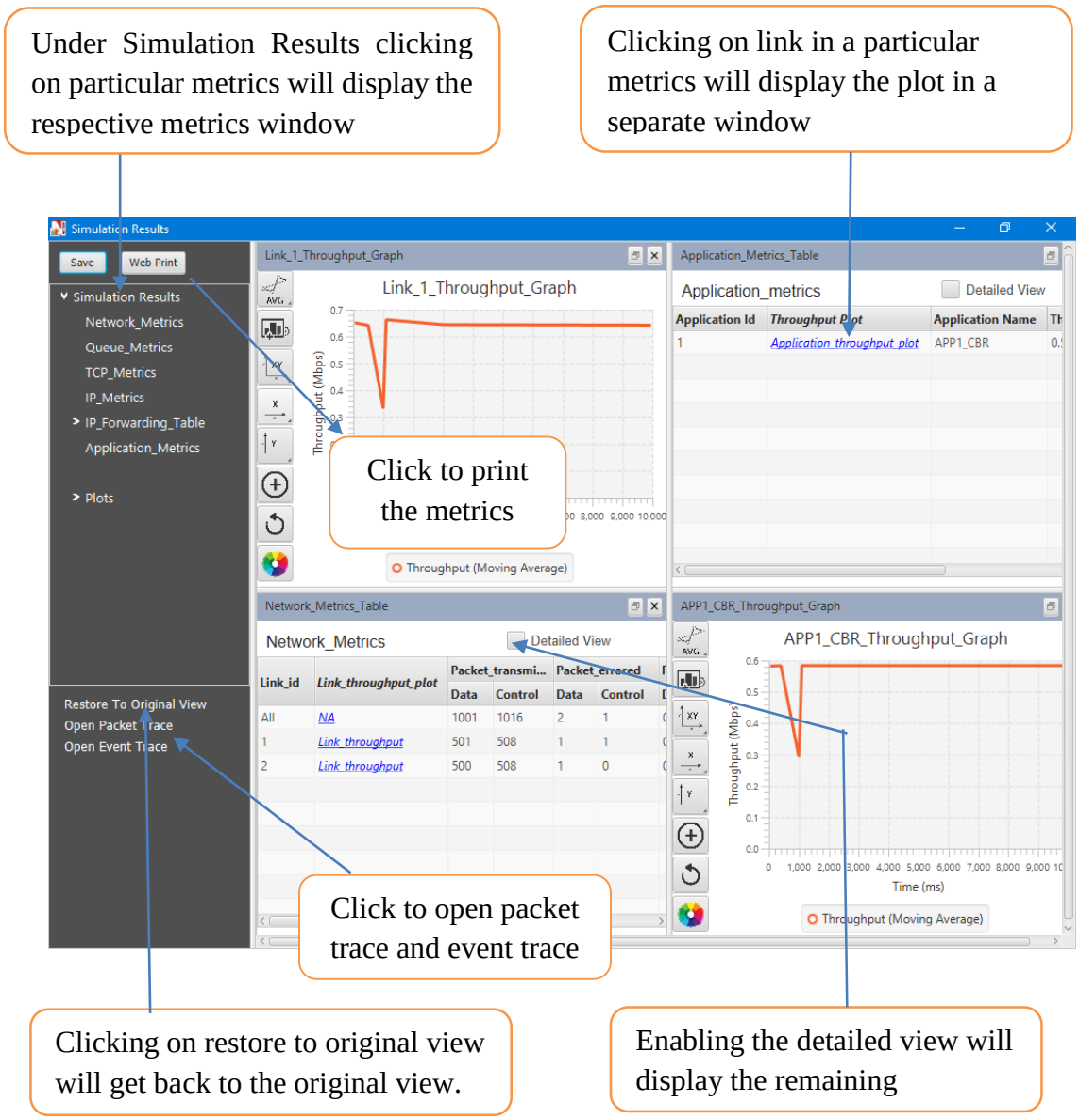
It is not mandatory to place the configuration.xsd file along with the configuration.xml file in the iopath. But if it is done, then it will be easier to check & validate changes that are done to the configuration.xml file.

6 Analysis

6.1 Performance Metrics

NetSim provides distinct quantitative 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 can be further subdivided 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 –** It is the total bytes transmitted over the link with respect to the simulation time.

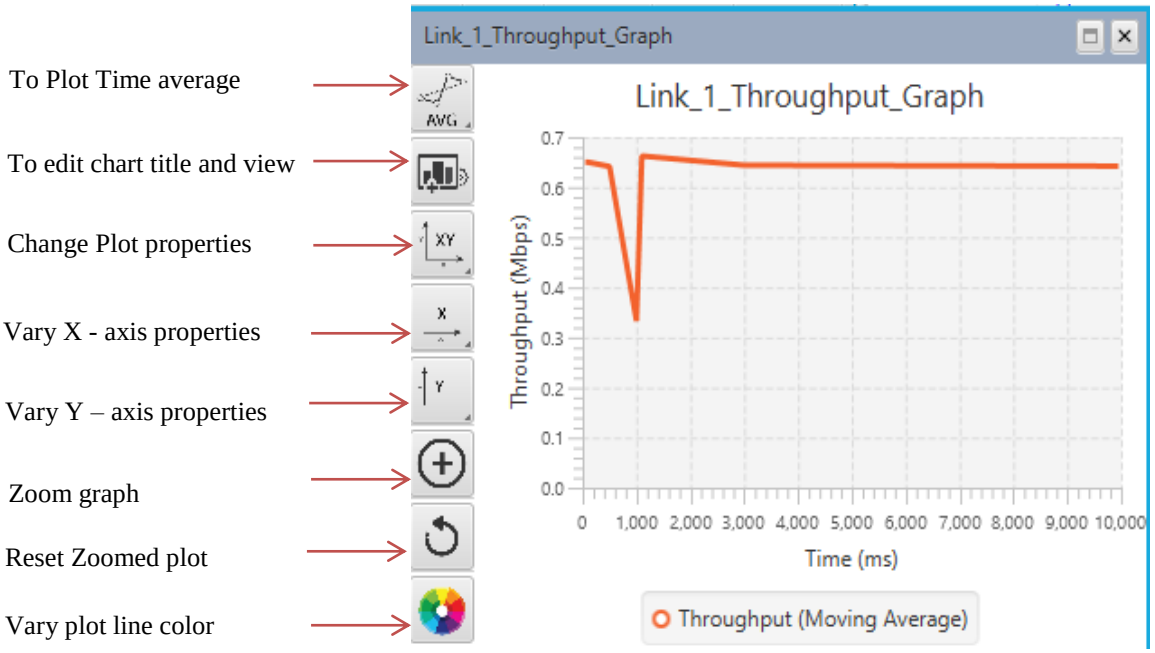
Calculation:

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

Now Total Bytes transmitted includes data packets and control packet.

By data packet, we mean the total bytes the packet has at the phy layer, which would include app layer payload plus the overheads of all layers.

Both successful, error and collision packets are included in this calculation (or) only successful packets are counted and error / collision are not counted in this metrics.



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: It is the total distance covered by a packet to the elapsed time to cover the distance. It is calculated by

$$\text{Time average} = \frac{Total\ distance\ covered}{elapsed\ time\ to\ cover\ the\ distance}$$

- **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.

- **Device metrics**: Displays device related metrics like ARP table, IP forwarding tables. This is also dependent upon the type of network

- **Application Metrics**: Displays the various metrics based on the Application running in the network scenario.
    - **Application Id-** It is the unique Id of the application running at the source.
    - **Application_ throughput_ graph:** it is the total payload delivered to destination to the simulation time.
    - **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 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'.
    - **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:

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

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

Note about metrics: 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.

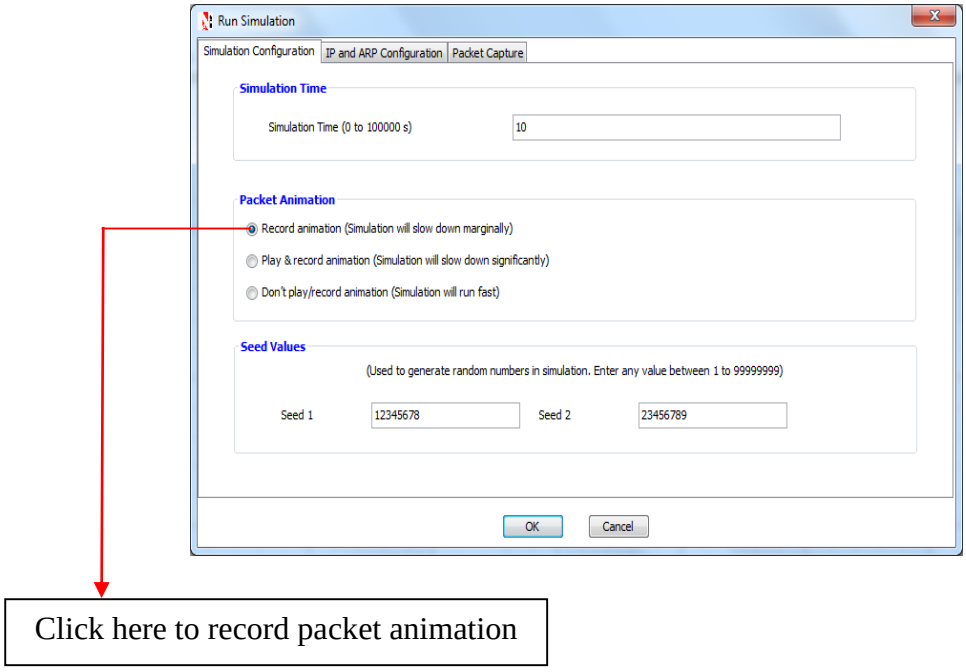
If NetSim runs via CLI, then the metrics will be written into Metrics.txt and MetricsGraph.txt.

6.2 Packet Animation

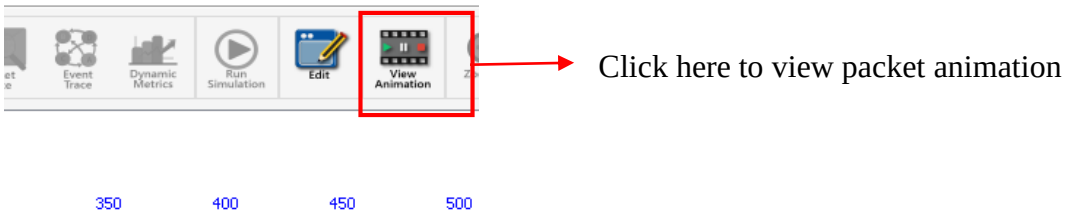
NetSim provides the feature to play and record animations to the user. Packet animation enables users to watch traffic flow through the network for in-depth visualization and analysis.

User has the following options before running simulation:

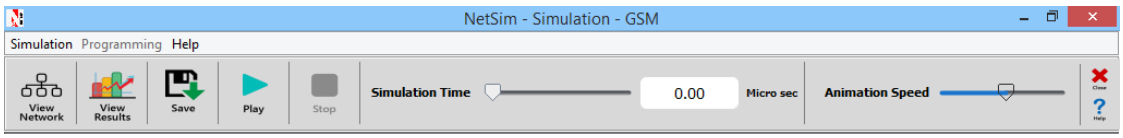
- No animation,
- Record the animation and
- Play and record animation while running simulation.



The packet animation would then be recorded and the user can view the animation from the metrics screen as shown below:

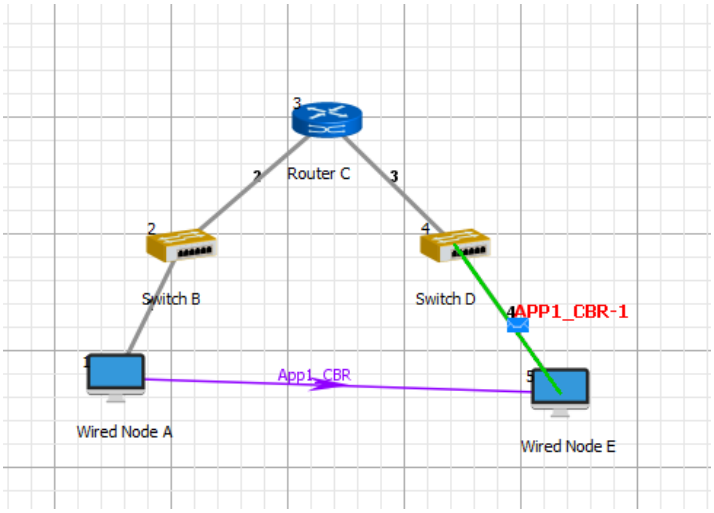


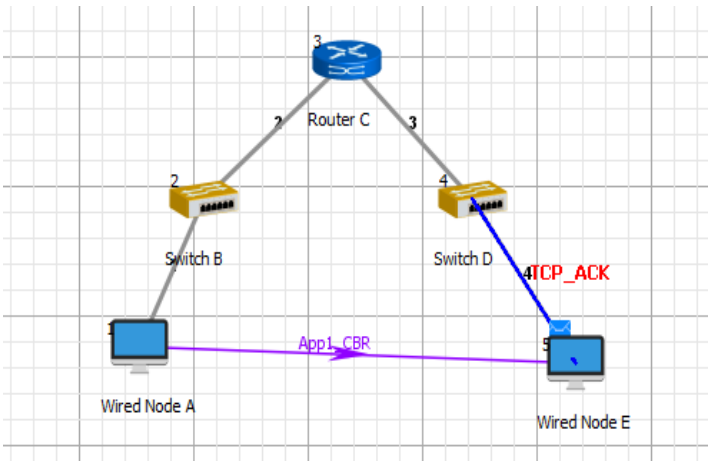
When packet animation is enabled, a screen with the following toolbar appears:



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.

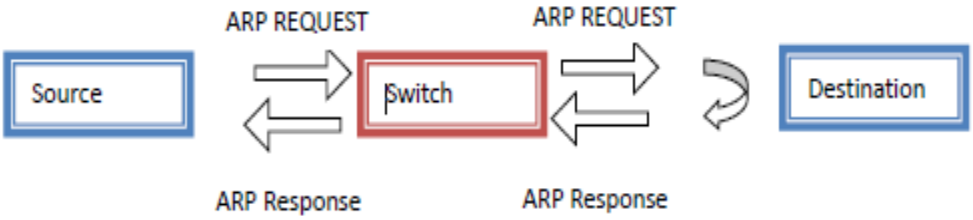
Example showing packet animation: In first figure, CBR data packet is flowing from Switch D to Node E (green color) and TCP_ACK is sent from Node E to Switch D in second figure (blue color).





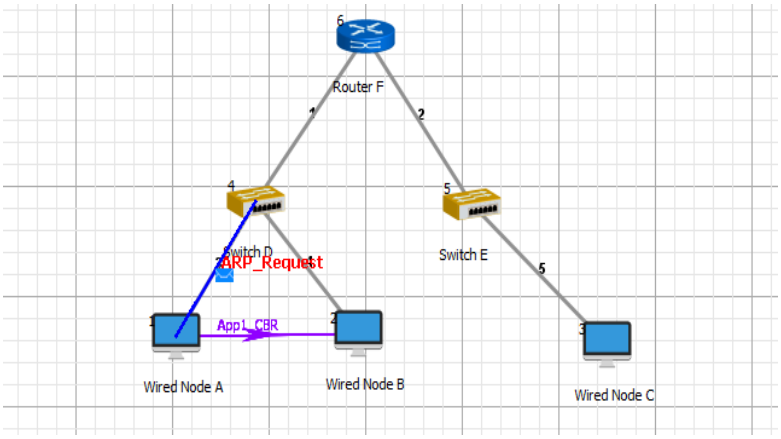
6.2.1 Example on how to use NetSim packet animation feature:

Case 1: ARP PROTOCOL - WORKING



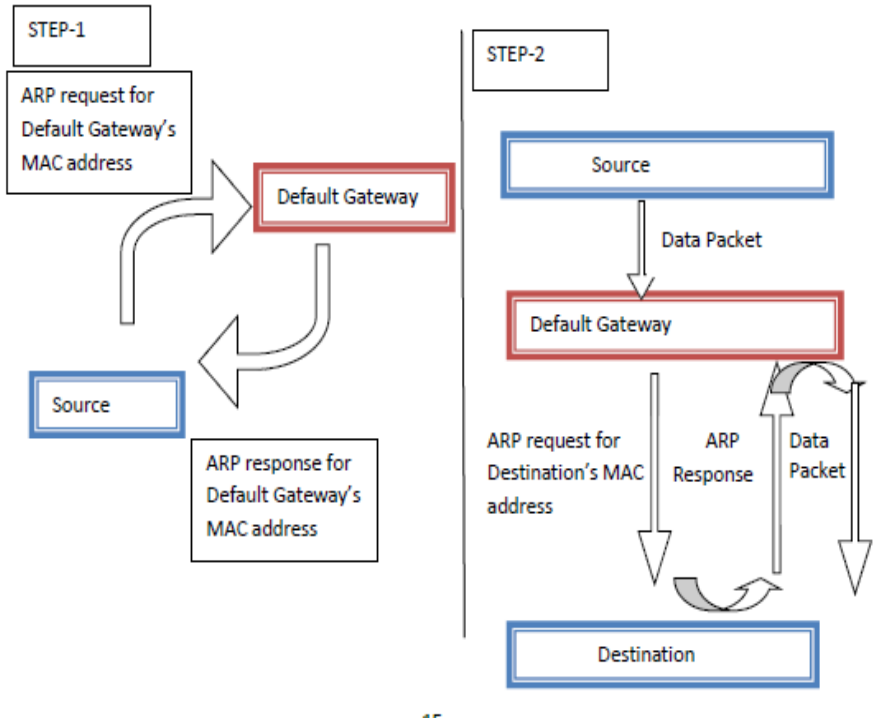
- Create a scenario with 3 wired nodes, 2 switches and 1 router and connect it based on the following scenario.
- Disable TCP in all the wired nodes.
- Select Application button and click on the gap between grid environment and ribbon. Right click on application, select properties and set Source_Id and Destination_Id as 1 and 2 respectively.
- Set Simulation time = 100s. After clicking on Run Simulation, edit **IP and ARP Configuration** tab by setting Static ARP as Disable. Click on OK button to simulate.

Now click on packet animation and analyse the following:



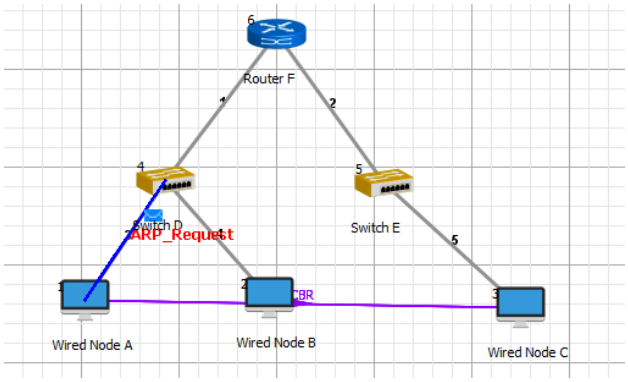
- NODE-1 sends ARP_Request which is then broadcasted by SWITCH-4.
- During the process the devices that receive the ARP_Request packet (Switch, Router, and Node-2) will update their ARP table or the switch table.
- NODE -2 sends the ARP_Reply to NODE-1 via SWITCH-4.
- Now NODE-1 updates its ARP table 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, drop 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 **IP and ARP Configuration** tab and set Static ARP as Disable. Click on OK button to simulate.

Click on packet animation to analyse the following:



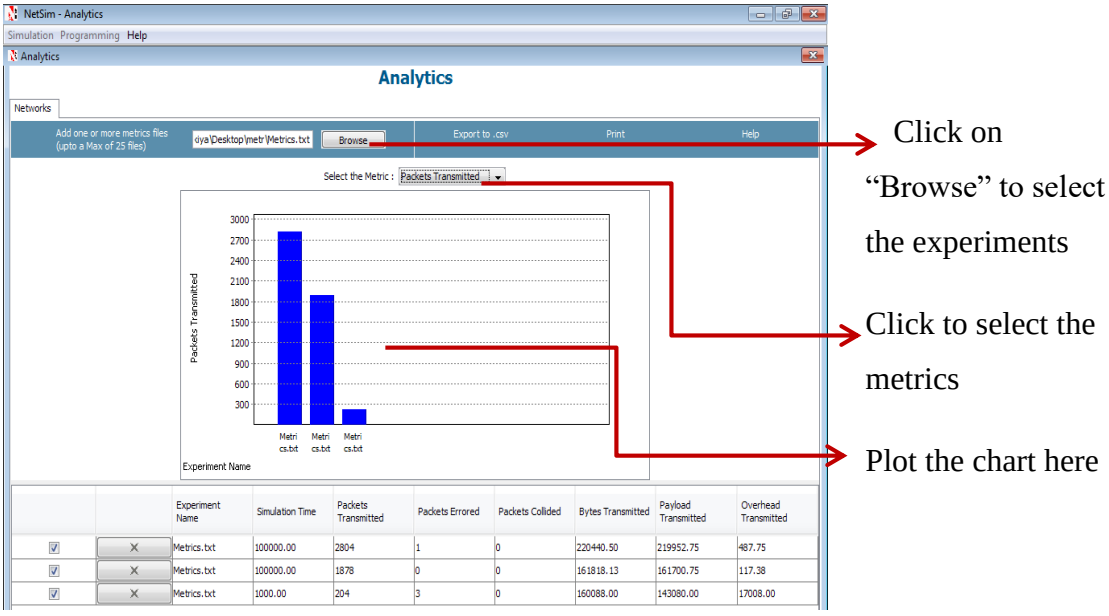
- NODE-1 transmits ARP_Request which is further broadcasted by SWITCH-4. ROUTER-6 sends ARP_Reply to NODE-1 which goes through SWITCH-4. Then NODE-1 starts to send data to NODE-3.
- If the router has the address of NODE-3 in its routing table, ARP protocol ends here and data transfer starts that is PACKET_ID 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.

6.3 Analytics Menu (Multiple Experiments)

NetSim analytics module is designed to enable comparison of performance metrics of multiple experiments. All the different networks of NetSim can be compared in this menu. Parameters like Simulation time, packets transmitted, packets errored, packets collided, Bytes transmitted etc., of different sample experiments can be visualized with the help of graphs.

To compare and analyse performance of various networks goto Netsim **Simulation** menu and click **Analytics** to view the **Analytics** screen. A graphing engine facilitates plotting of different performance metrics across the different experiments. This can be used to infer/trend how changes in input parameters affect network performance. An explanation of the use of the different fields in the **Analytics** screen is detailed below:

- Select a Metrics File (simulation output file which will be stored where the simulation experiment is saved) by choosing select the **Metrics File** to add it onto the Metrics Table by using **Browse** button.



- **The Graph Field**
  - **Select the Metrics - Select** the parameters for **Y-axis** by clicking on the dropdown menu.

- **Graph** - Based on the **X-axis** (i.e. **Metrics File**/ **Experiment** selected) and **Y-axis** (i.e. **Metrics** selected by using the dropdown menu above the **graph**), a **Bar graph** will be plotted.
- Add metrics file of different experiments to compare.

- **The Metrics Table-** This is the table that is generated when the Experiments are selected using the **Select the Metrics File**/ **Experiment** option.
  - The first column in the **Metrics Table** consists of the **Check Boxes** that are used to **select** (i.e. by default all the Experiments are selected) or **deselect** the selected **Experiments**. If an Experiment is selected, then that Experiment would be compared along with other Experiments, else the Experiment would not be compared with other Experiments.
  - The second column in the **Metrics Table** "**X**" button is available. This "**X**" button is used for deleting the **Experiment.**
- **Change Graph color**: This option is provided for users to change the color of the graph. For this, users can goto change color option → click on color→ select their desired color → click on ok.
- **Export to .csv** - **Click** on **Export to .csv** to export the **Metrics Table** to a "**.csv**" format. This action button will export the contents to a ".csv" format that is available.
- **Print** - **Click** on the **Print** option to print the **Metrics Table**. This action button will fetch the Print dialogue box.

6.4 Packet Trace

NetSim Packet trace is saved as a spread sheet. This file 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.

6.4.1 Analysing Packet trace using Pivot tables

NetSim Packet trace is saved as a spread sheet. This file 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

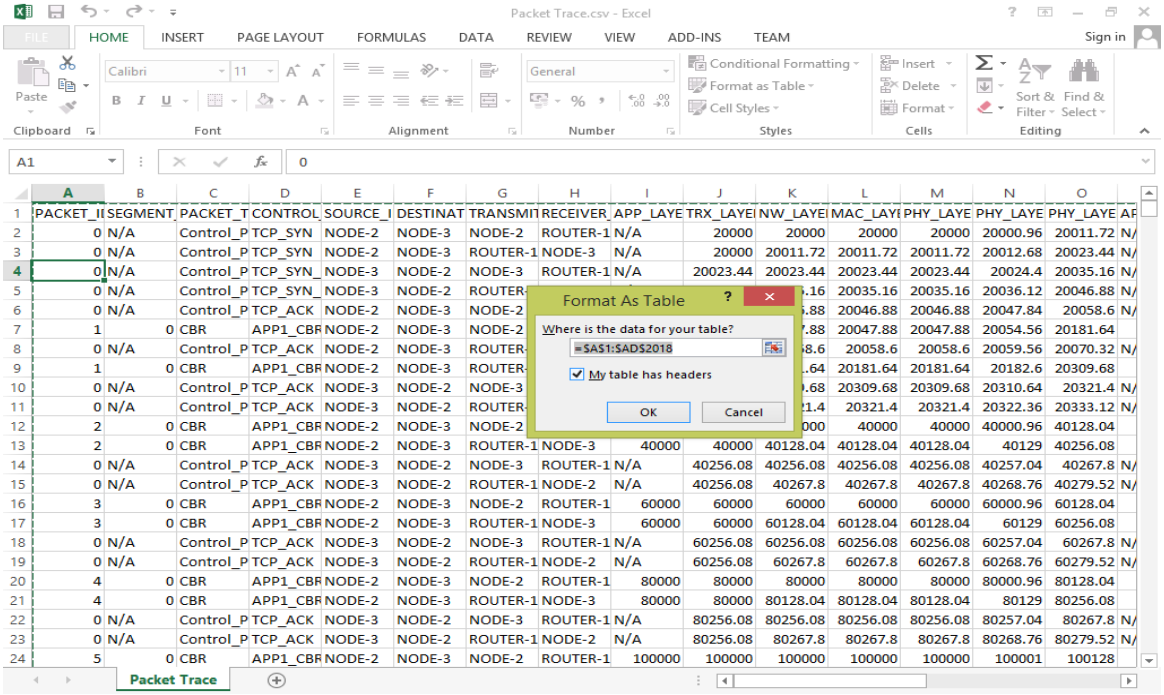
PivotTables are a great way to summarize, analyse, explore, and present your data, and you can create them with just a few clicks. PivotTables are highly flexible and can be quickly adjusted depending on how you need to display your results. You can also create Pivot Charts based on PivotTables that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

Following are the steps to analyse the packet trace using pivot tables (using Excel 2013)

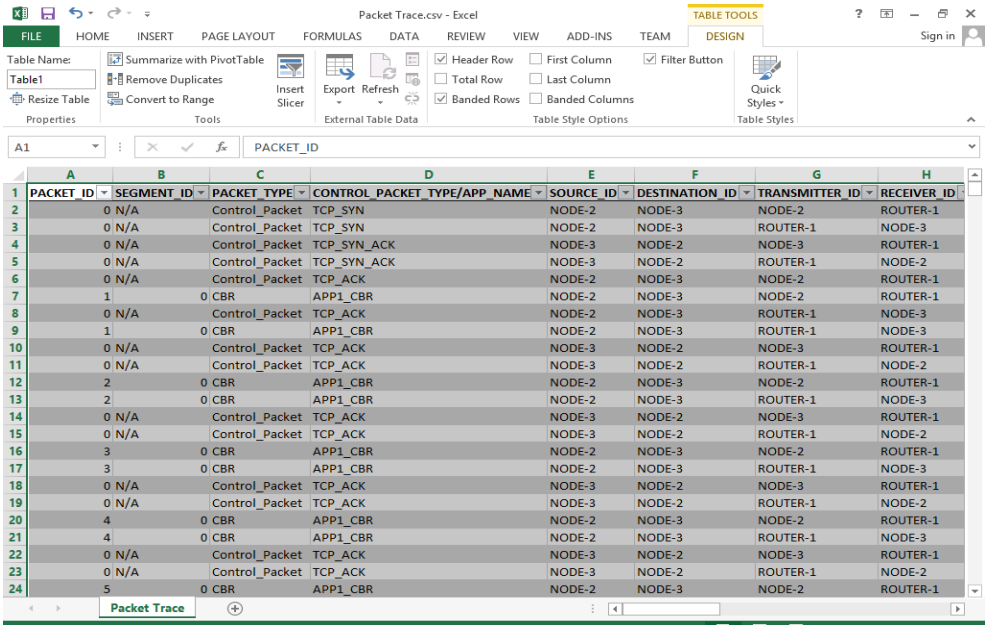
1. Click on a cell and then click on Format as Table



The Excel would ask you for the range of the data set and in general Excel would automatically choose till the last row, as shown below:

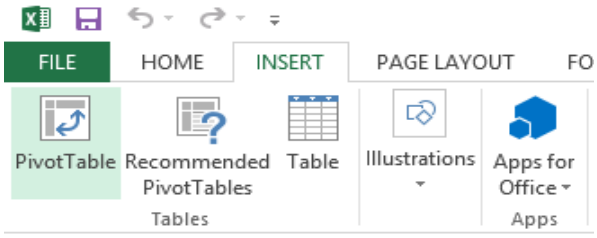


Click Ok and then the spread sheet will be converted to a table as shown below

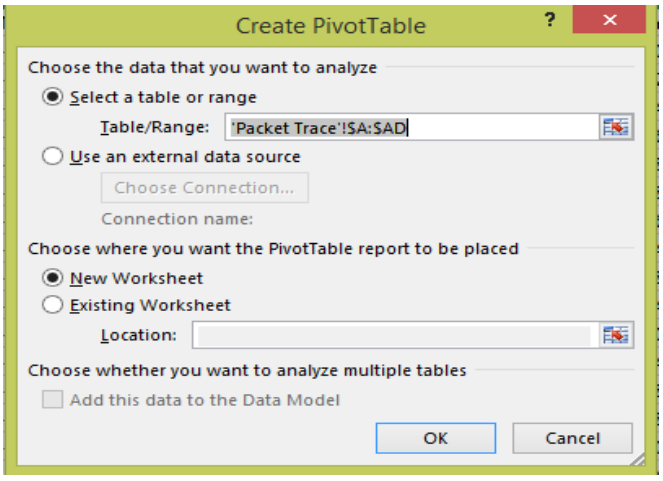


	A	B	C	D	E	F	G	H
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
2	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1
3	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3
4	0	N/A	Control_Packet	TCP_SYN_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1
5	0	N/A	Control_Packet	TCP_SYN_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2
6	0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1
7	1	0	CBR	APP1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1
8	0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3
9	1	0	CBR	APP1_CBR	NODE-2	NODE-3	ROUTER-1	NODE-3
10	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1
11	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2
12	2	0	CBR	APP1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1
13	2	0	CBR	APP1_CBR	NODE-2	NODE-3	ROUTER-1	NODE-3
14	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1
15	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2
16	3	0	CBR	APP1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1
17	3	0	CBR	APP1_CBR	NODE-2	NODE-3	ROUTER-1	NODE-3
18	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1
19	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2
20	4	0	CBR	APP1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1
21	4	0	CBR	APP1_CBR	NODE-2	NODE-3	ROUTER-1	NODE-3
22	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1
23	0	N/A	Control_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2
24	5	0	CBR	APP1_CBR	NODE-2	NODE-3	NODE-2	ROUTER-1

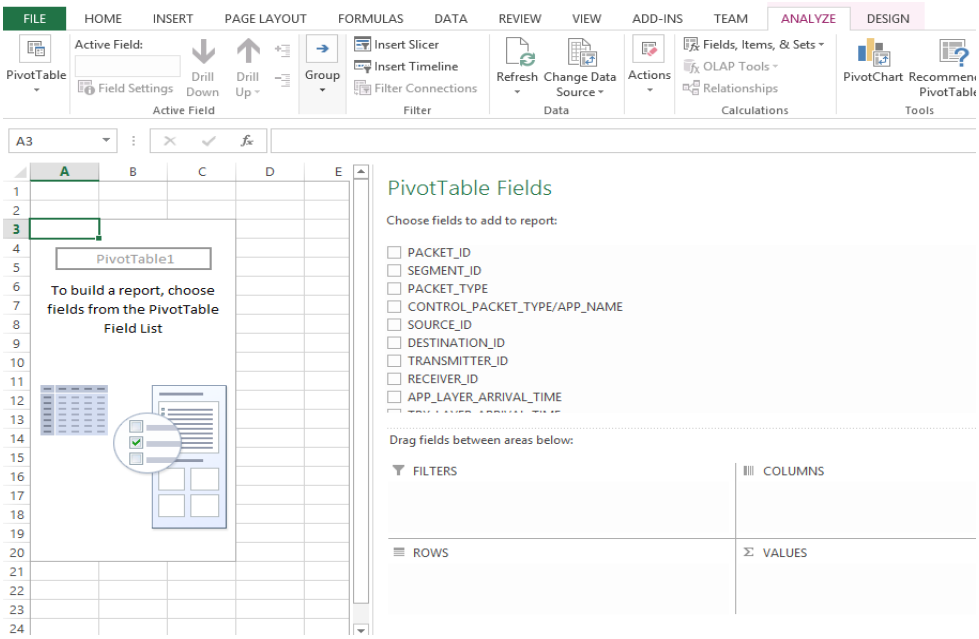
2. Next, from the **Insert** tab, click the **PivotTable** command.



3. The **Create PivotTable** dialog box will appear. Click on OK



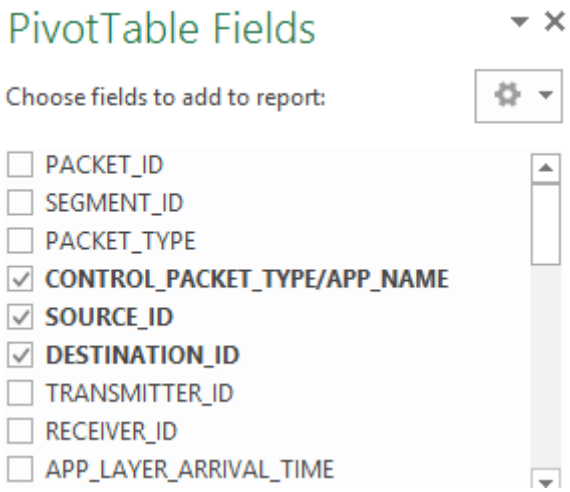
4. A blank **PivotTable** and **Field List** will appear on a new worksheet.



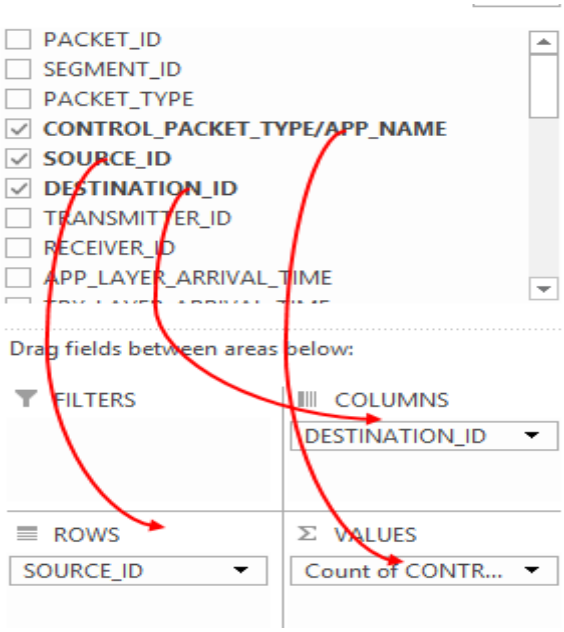
5. Once you create a PivotTable, you'll need to decide which **fields** to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

Packet Transmitted / Received Analysis

6. 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.



7. 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

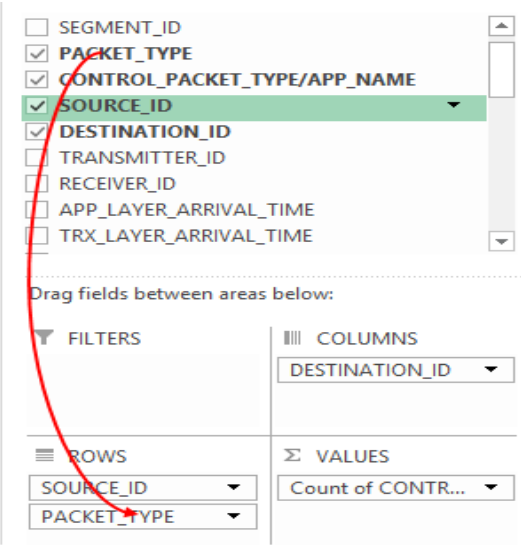


8. The PivotTable will calculate and summarize the selected fields. In this example, the PivotTable shows the packets sent from all sources to all destinations.

Count of CONTROL_PACKET_TYPE/APP_NAME	Column Labels				
Row Labels	NODE-2	NODE-3	ROUTER-1	(blank)	Grand Total
NODE-2		10018	33		10051
NODE-3	10000		33		10033
ROUTER-1	33	33			66
(blank)					
Grand Total	**10033**	**10051**	**66**		**20150**

9. The above example shows all the packets which including data packets and control packets.

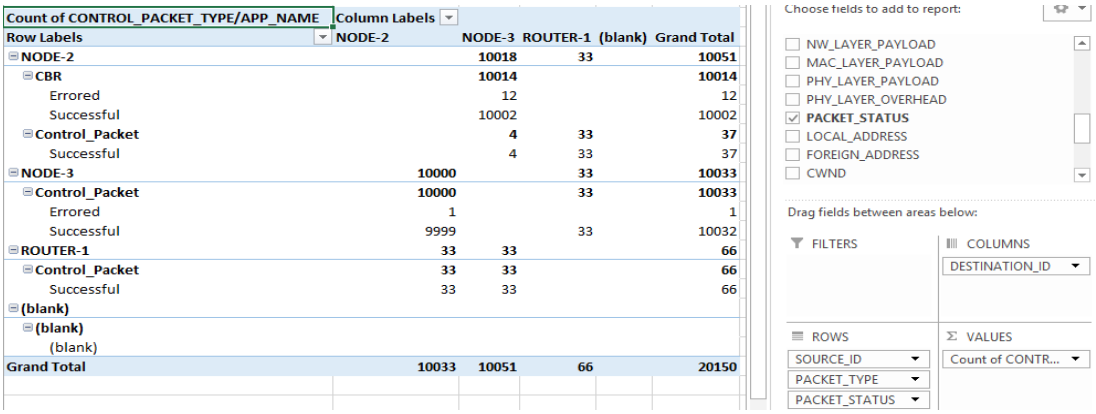
10. If you wish to know how many Data and how many were control packets then, check the PACKET_TYPE and drag it to the ROWS field.



11. This will look like

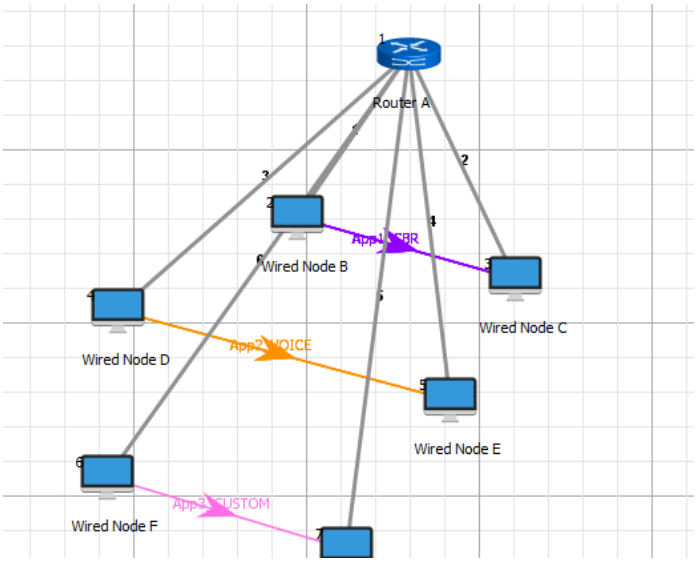
Count of CONTROL_PACKET_TYPE/APP_NAME	Column Labels				
Row Labels	NODE-2	NODE-3	ROUTER-1	(blank)	Grand Total
NODE-2		**10018**	**33**		**10051**
CBR		10014			10014
Control_Packet		4	33		37
NODE-3	**10000**		**33**		**10033**
Control_Packet	10000		33		10033
ROUTER-1	**33**	**33**			**66**
Control_Packet	33	33			66
(blank)					
(blank)					
Grand Total	**10033**	**10051**	**66**		**20150**

12. 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.



Delay analysis:

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

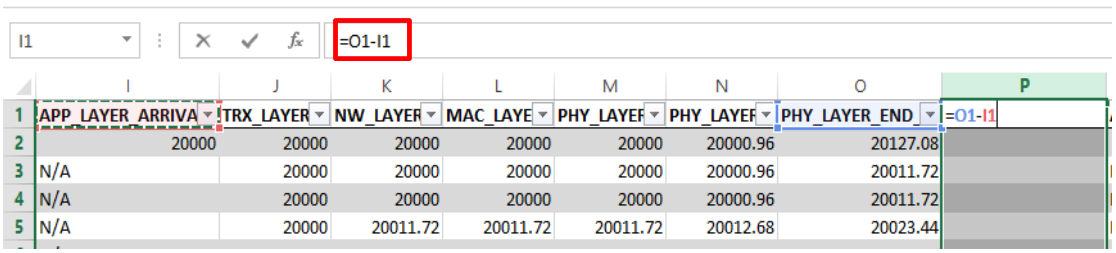


Create a network scenario with 1 router and 6 wired nodes. Create 3 applications as per the following

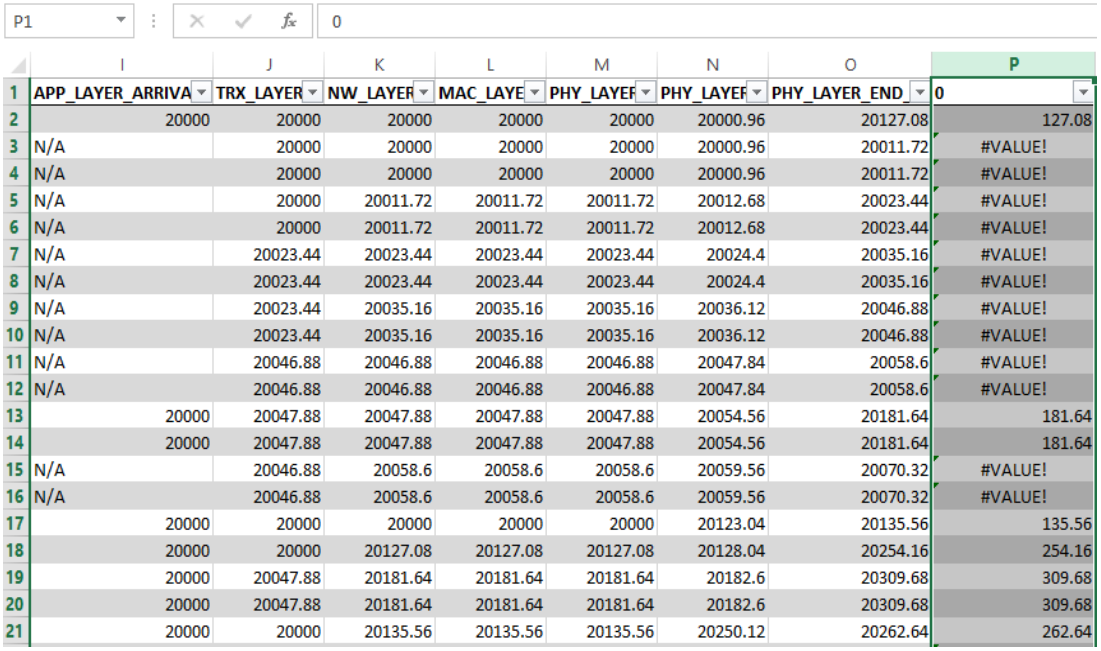
Application Type	Source Id	Destination Id	Packet Size (Bytes)	Inter arrival time (µs)
CBR	2	3	1460	20000
VOICE	4	5	1500	20000
CUSTOM	6	7	1200	20000

Enable Packet Trace and simulate the scenario for 10 seconds. Open packet trace and convert the spread sheet to table (explained earlier):-

1. After creating table, Insert 1 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



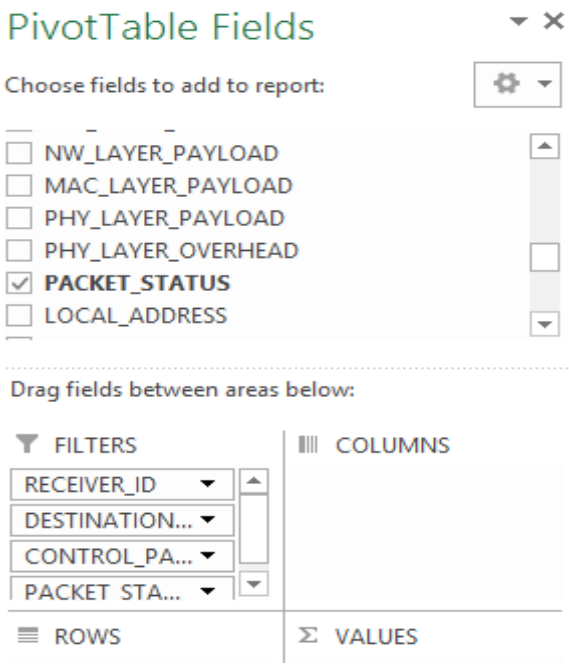
2. Then Press CTRL + ENTER. This will calculate delay for the whole column shown below



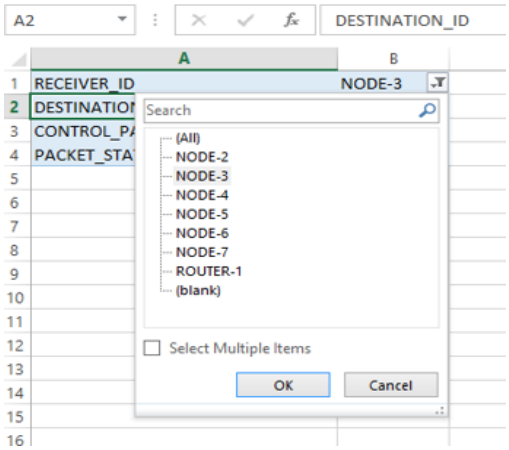
P1 | 0

	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

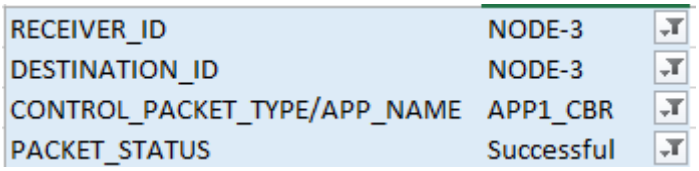
3. Name the column as DELAY
4. Drag and drop DESTINATION_ID, RECEIVER_ID, PACKET_STATUS and CONTROL_PACKET_TYPE/APP_NAME to FILTERS field shown below



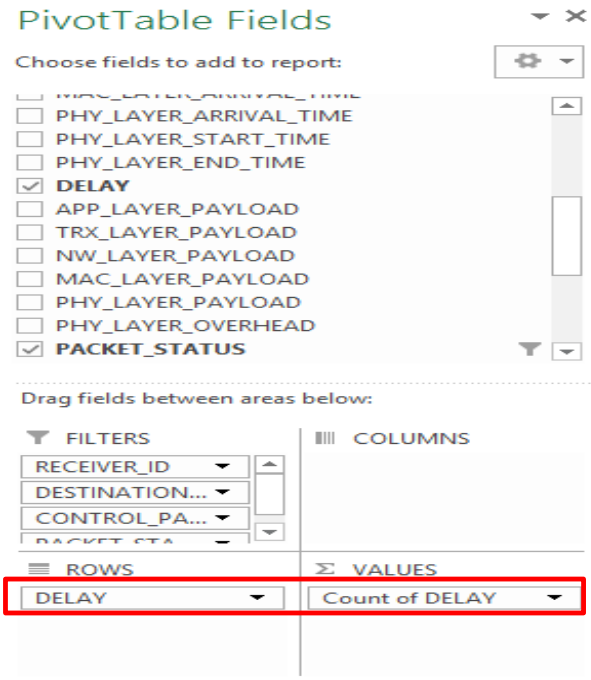
5. Filter RECEIVER_ID to Node-3 by clicking on the drop down and select OK



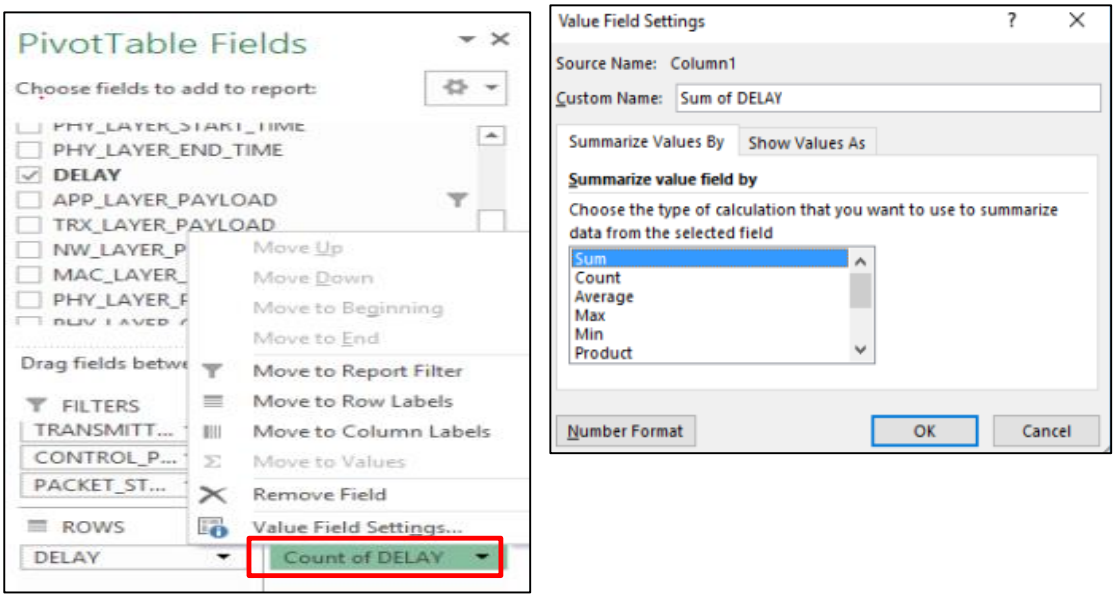
6. Similarly filter CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR, DESTINATION_ID to NODE-3 and PACKET_STATUS to Successful



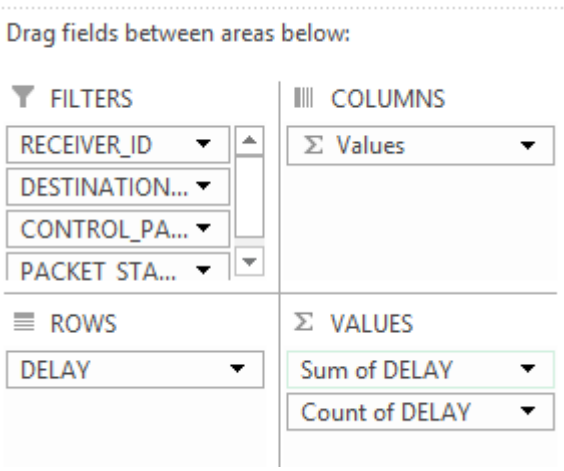
7. Drag and drop DELAY value that we have calculated earlier to ROWS and VALUES field



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

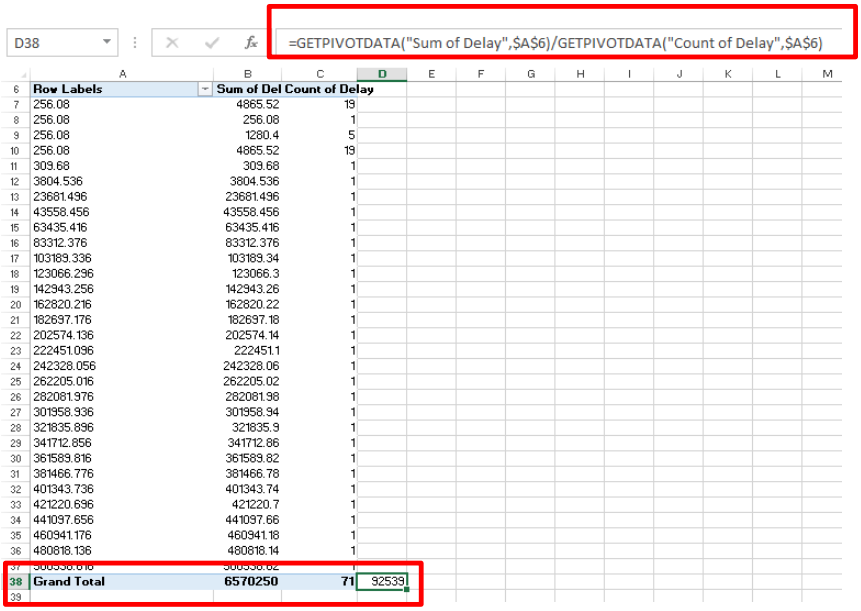


9. Again Drag and drop DELAY to VALUES field.



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

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



11. 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

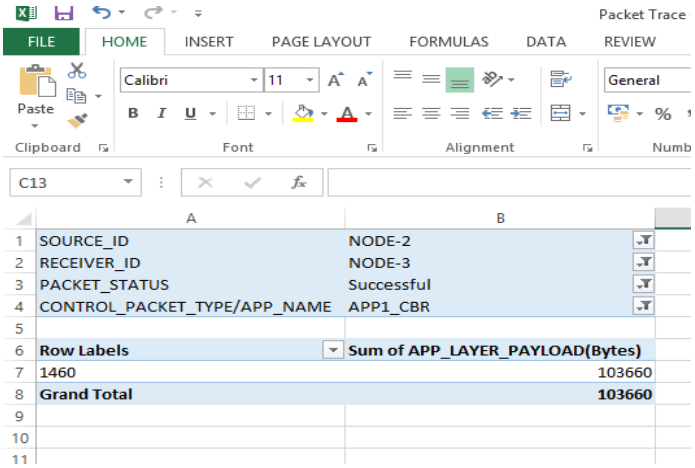
12. To calculate DELAY for VOICE application, filter DESTINATION_ID to Node-5, RECEIVER_ID to Node-5, CONTROL_PACKET_TYPE/APP_NAME to APP2_VOICE and PACKET_STATUS to Successful
13. Similarly calculate and compare DELAY for other applications by following the above procedure

Throughput analysis:

To explain how users can perform Throughput Analysis, we have used same network design example as was used for Delay analysis above.

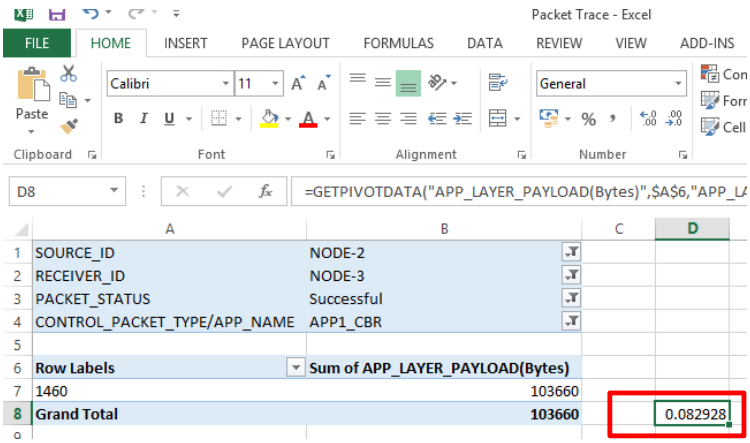
1. After creating Pivot table, Drag and drop SOURCE_ID, RECEIVER_ID, CONTROL_PACKET_TYPE / APP_NAME and PACKET_STATUS to FILTERS field.
2. Similarly drag and drop APP_LAYER_PAYLOAD to ROWS field and VALUES field

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



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

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



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

Application_Metrics_Table

Application_metrics Detailed View

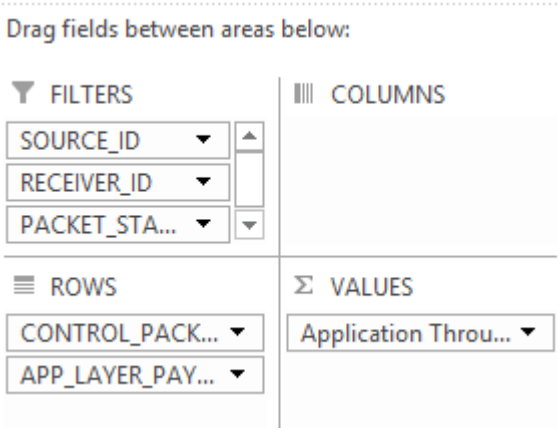
Application Id	Application Name	Throughput (Mbps)	Delay(microsec)
1	APP1_CBR	0.082928	92538.737092
2	APP2_VOICE	0.597600	258.440482
3	APP3_CUSTOM	0.048960	215.530980

8. 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
9. Similarly calculate and compare THROUGHPUT for other applications by following the above procedure.

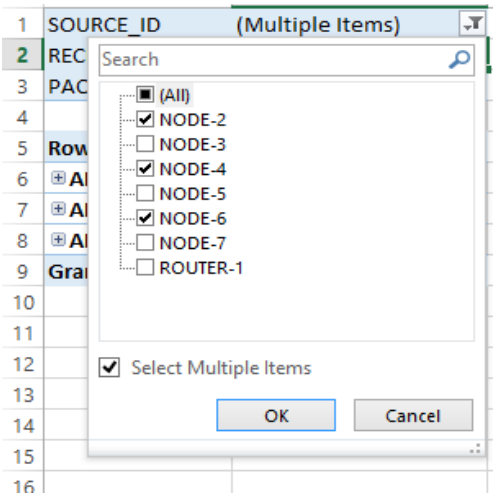
Plotting with Pivot Charts

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

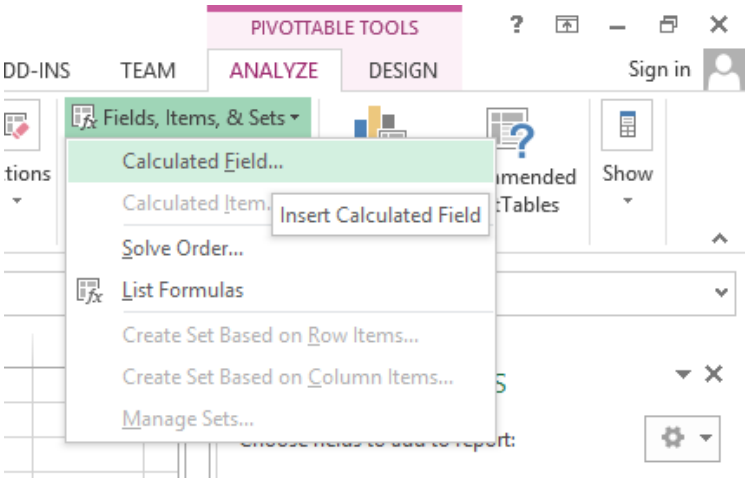
1. Drag and drop SOURCE_ID, RECEIVER_ID and PACKET_STATUS to FILTERS field, then CONTROL_PACKET_TYPE/APP_NAME, APP_LAYER_PAYLOAD to ROWS field and APP_LAYER_PAYLOAD to VALUES field as shown below



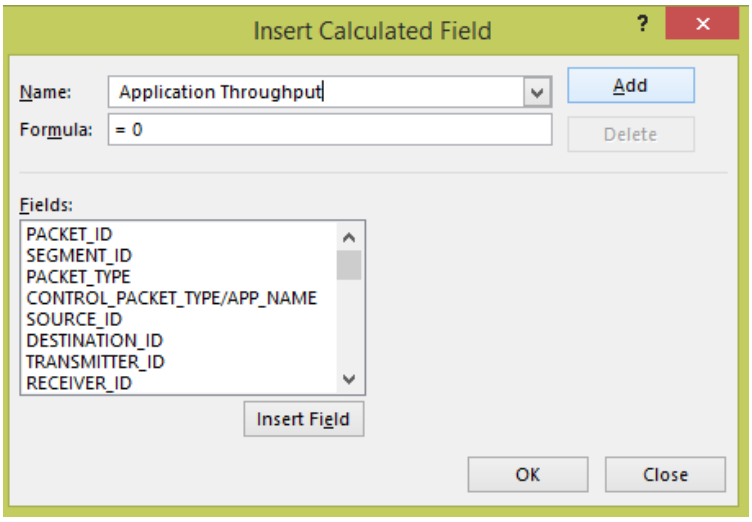
2. Filter SOURCE_ID to Node 2, Node 4 and Node 6, then RECEIVER_ID to Node 3, Node 5 and Node 7 and PACKET_STATUS to successful



3. Filter CONTROL_PACKET_TYPE/APP_NAME to APP1_CBR, APP2_VOICE and APP3_CUSTOM
4. 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).
5. In the Calculations group, click Fields, Items, & Sets, and then click Calculated Field.

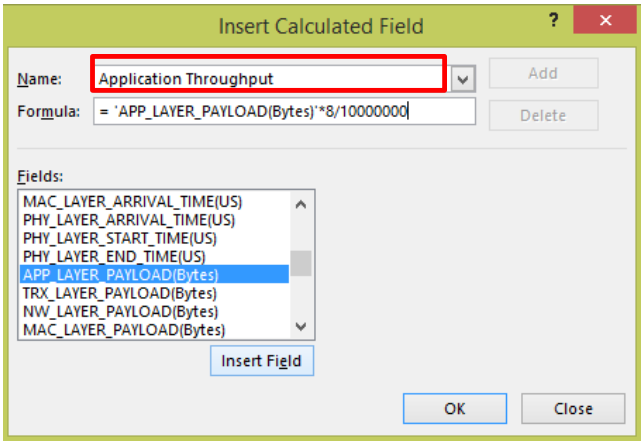


6. Type a name for the calculated field, Application Throughput
7. Then click on ADD to save the calculated field.



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

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

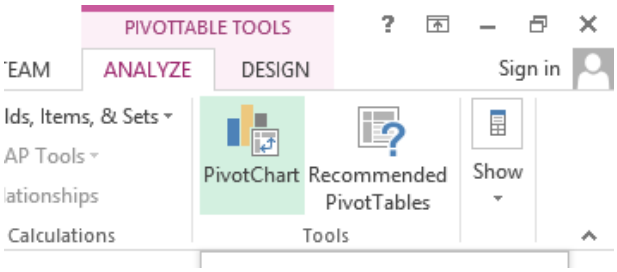


10. This would look like

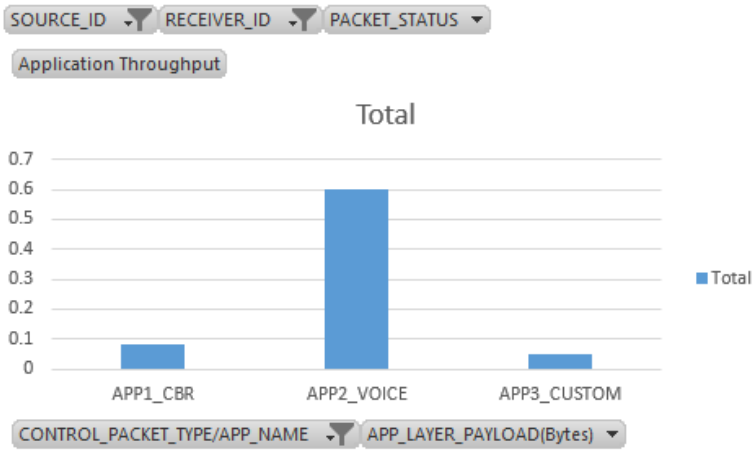
Row Labels	Application Throughput
APP1_CBR	0.084096
APP2_VOICE	0.5988
APP3_CUSTOM	0.04896
Grand Total	0.731856

SCREEN SHOT FOR COMPARISON

11. 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).
12. In the Tools group, click Pivot chart and select OK.



13. This will display a pivot chart shown below



Packet Trace Metrics:

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 **BGP:** BGP_KEEPALIVE, BGP_OPEN, BGP_UPDATE **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.

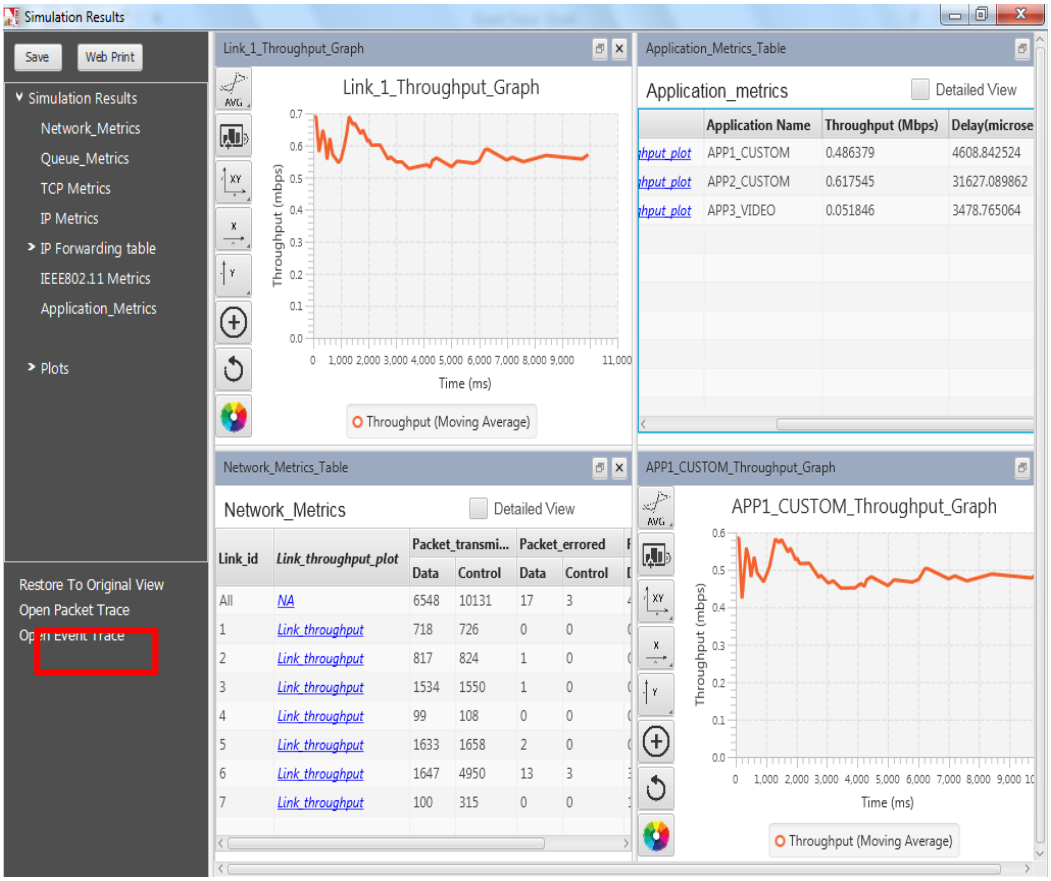
FOREIGN_ADDRESS	Specifies the Port Number at Destination Node. Port Numbers are chosen randomly by NetSim.
CWND (bytes)	Specifies the current size of the TCP congestion window
SEQ_NO	If TCP is enabled, it specifies the TCP Sequence number of the packet
ACK_NO	If TCP is enabled, it specifies the TCP Acknowledgement 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

Note:

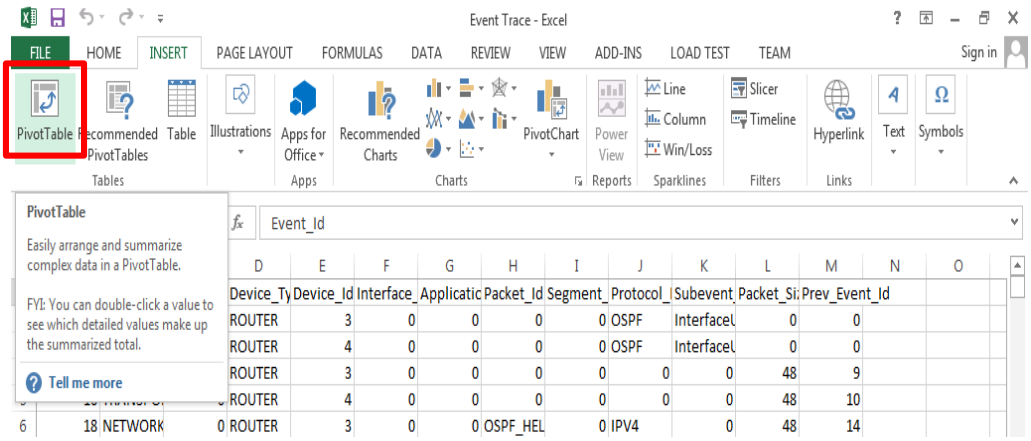
- Each line in the packet trace represents one hop of one packet.
- The packet trace is logged in ascending order of time as measured in Phy_Layer_End_Time.

6.4.2 Calculation of Delay, Jitter and Application through put using pivot tables in NetSim event trace

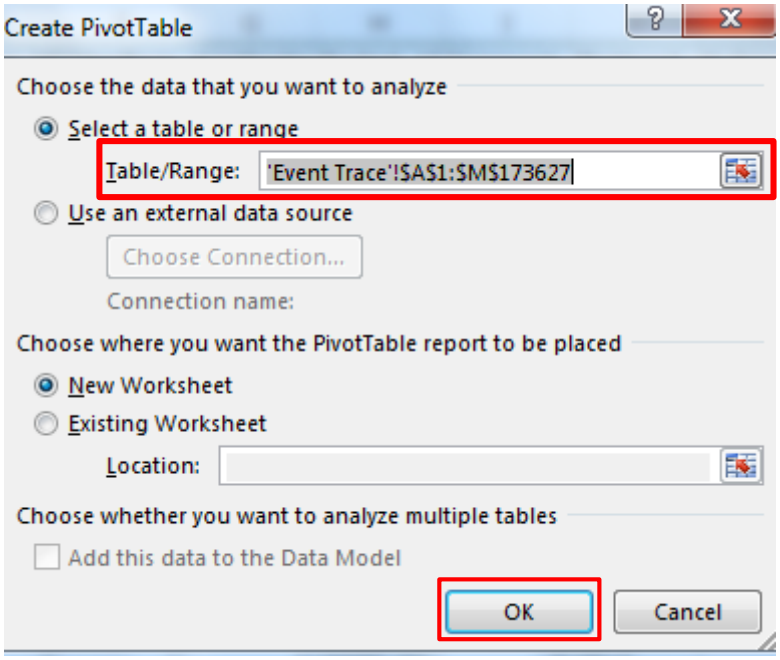
1. Enable Event trace and after simulation select Event Trace in the Simulation results window. Event tracing is available only in NetSim standard and Pro versions as of v10.



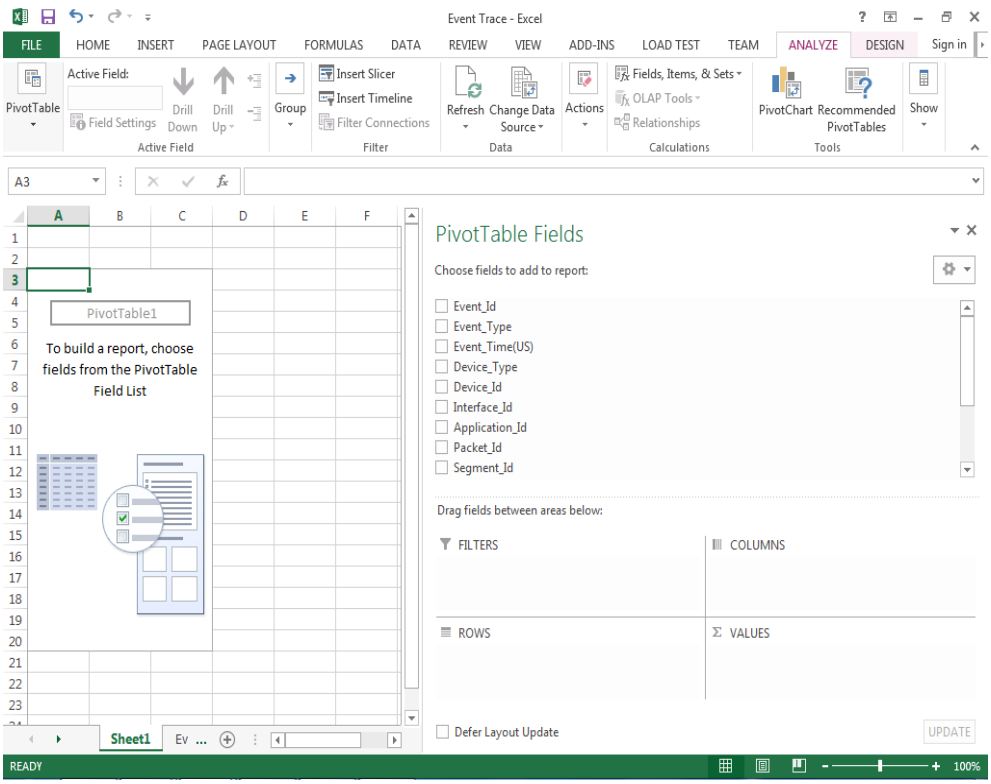
2. Click on **Pivot Table** in **INSERT** tab



3. Then a window named **Create Pivot Table** pops up which automatically selects the entire table, then click **OK** button. In case the entire table isn't selected please enter the range such that all rows are selected.



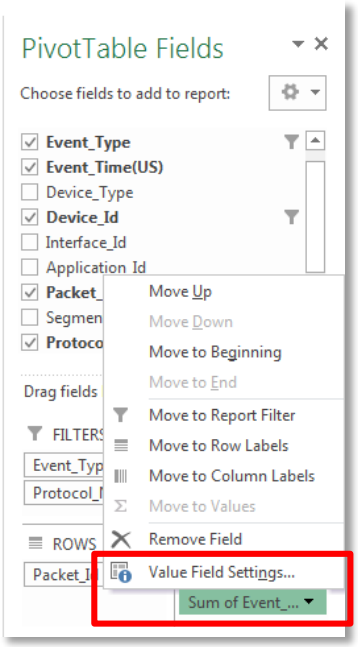
4. A blank **PivotTable** and **Field List** will appear on a new worksheet.



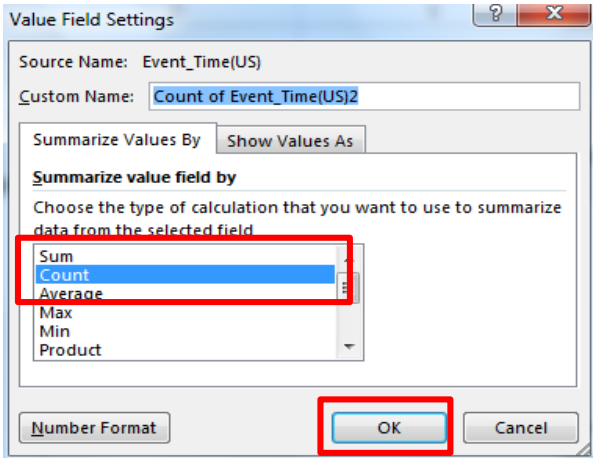
5. Once you create a PivotTable, you'll need to decide which **fields** to add. Each field is simply a **column header** from the source data. In the **PivotTable Field List**, check the box for each field you want to add.

6.4.2.1 Application Delay Analysis:

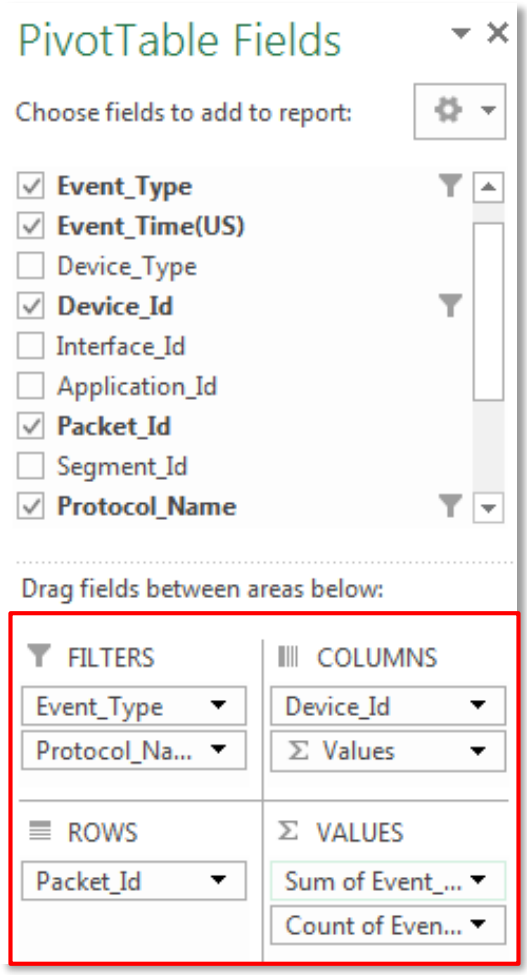
6. Drag and drop the **Event_Type**, **Protocol_Name** Fields into **FILTERS, Packet_Id** into **ROWS** and **Device_Id** into **COLUMNS**.
7. Drag and Drop **Event_Time** Field into **VALUES twice**, then both will show Sum of Event_Time. Recheck that you have dropped the Event_Time field twice.
8. Click on the **second Event_Time** field in the VALUES and select the **Value Field Settings.**



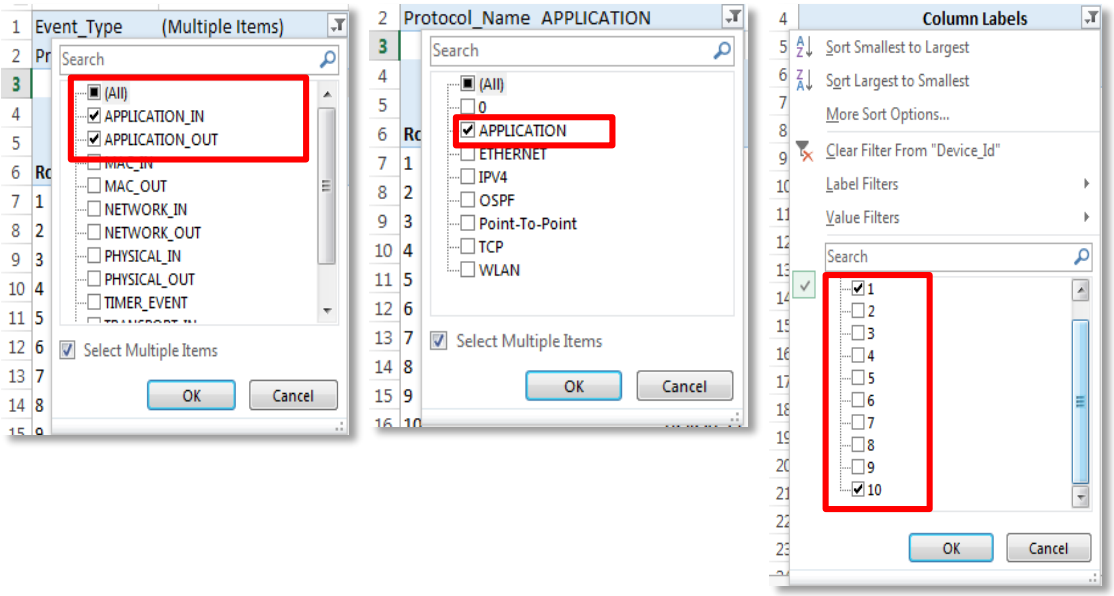
9. A window named **Value Field Settings** opens then select **Count** option and click **OK** button



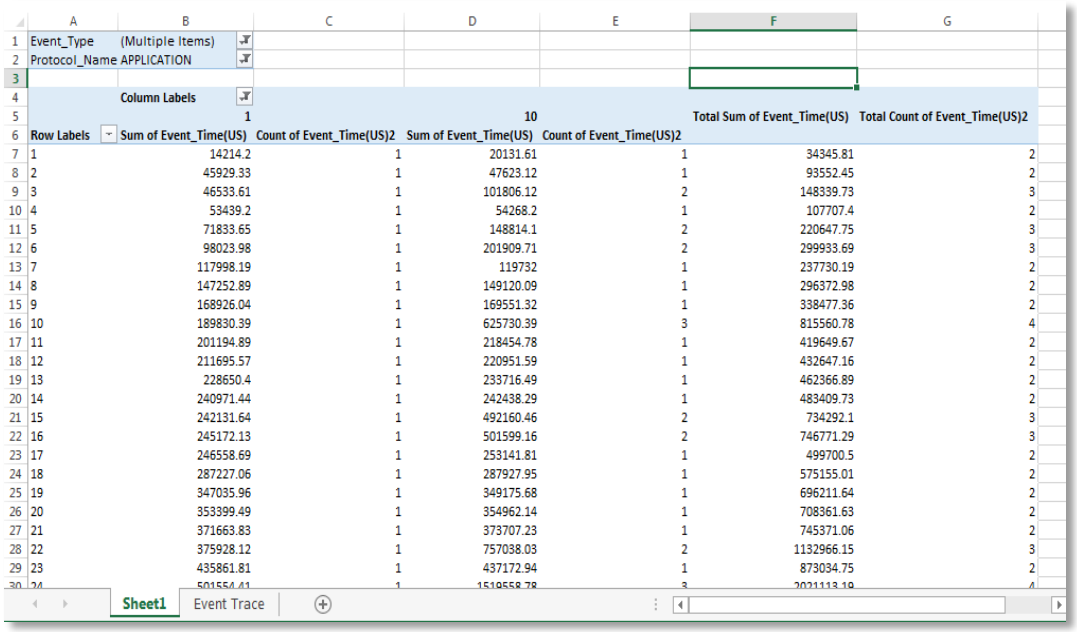
10. Then finally the **Pivot Table Fields** will be as shown below.



11. 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)

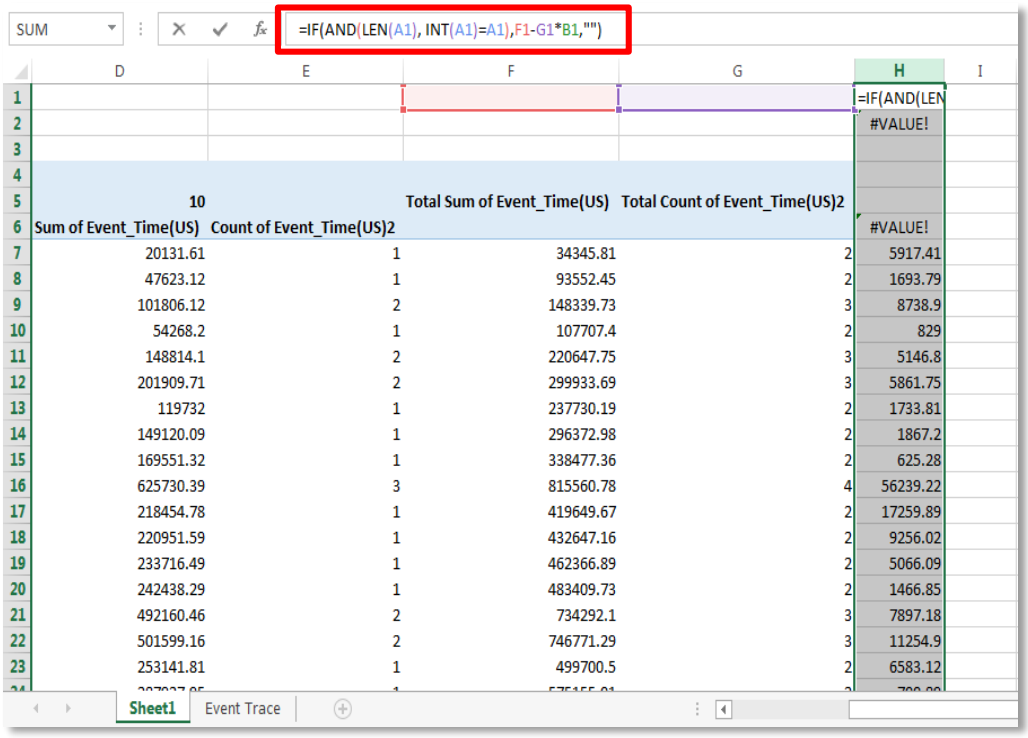


Event_Type	(Multiple Items)					
Protocol_Name	APPLICATION					
	Column Labels					
	1		10		Total Sum of Event_Time(US)	Total Count of Event_Time(US)2
Row Labels	Sum of Event_Time(US)	Count of Event_Time(US)2	Sum of Event_Time(US)	Count of Event_Time(US)2		
1	14214.2	1	20131.61	1	34345.81	2
2	45929.33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

Sheet1 Event Trace

12. Select the entire empty column H then and enter the formular **=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,"")

					=IF(AND(LEN
					#VALUE!
	10		Total Sum of Event_Time(US)	Total Count of Event_Time(US)2	
Sum of Event_Time(US)	Count of Event_Time(US)2				#VALUE!
20131.61	1		34345.81	2	5917.41
47623.12	1		93552.45	2	1693.79
101806.12	2		148339.73	3	8738.9
54268.2	1		107707.4	2	829
148814.1	2		220647.75	3	5146.8
201909.71	2		299933.69	3	5861.75
119732	1		237730.19	2	1733.81
149120.09	1		296372.98	2	1867.2
169551.32	1		338477.36	2	625.28
625730.39	3		815560.78	4	56239.22
218454.78	1		419649.67	2	17259.89
220951.59	1		432647.16	2	9256.02
233716.49	1		462366.89	2	5066.09
242438.29	1		483409.73	2	1466.85
492160.46	2		734292.1	3	7897.18
501599.16	2		746771.29	3	11254.9
253141.81	1		499700.5	2	6583.12

Sheet1 Event Trace

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

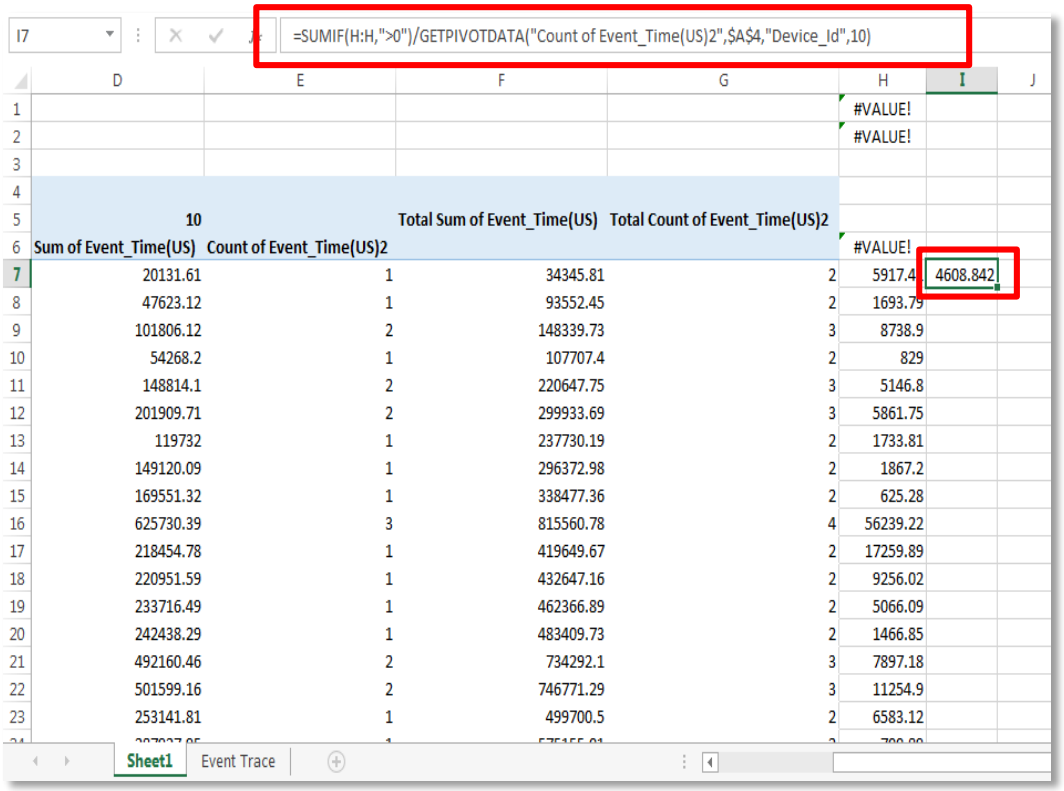
Note: If the packet size is > 1500 then fragmentation occurs and the packet is received as multiple segments. In NetSim the destination counts each segment as different packet.

Then in an empty cell enter

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

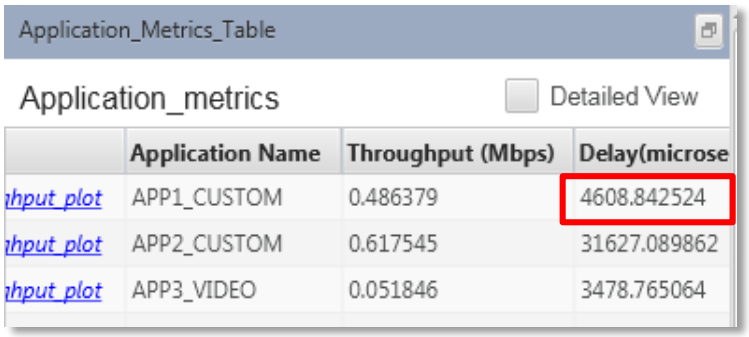
GETPIVOTDATA ("Count of Event_Time(US)2",$A$4,"Device_Id",10) gives the total number of packets received by the destination(in this case 10).

This will give the exact Application Delay



Sum of Event_Time(US)	Count of Event_Time(US)2	Total Sum of Event_Time(US)	Total Count of Event_Time(US)2	
20131.61	1	34345.81	2	5917.4
47623.12	1	93552.45	2	1693.79
101806.12	2	148339.73	3	8738.9
54268.2	1	107707.4	2	829
148814.1	2	220647.75	3	5146.8
201909.71	2	299933.69	3	5861.75
119732	1	237730.19	2	1733.81
149120.09	1	296372.98	2	1867.2
169551.32	1	338477.36	2	625.28
625730.39	3	815560.78	4	56239.22
218454.78	1	419649.67	2	17259.89
220951.59	1	432647.16	2	9256.02
233716.49	1	462366.89	2	5066.09
242438.29	1	483409.73	2	1466.85
492160.46	2	734292.1	3	7897.18
501599.16	2	746771.29	3	11254.9
253141.81	1	499700.5	2	6583.12

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.

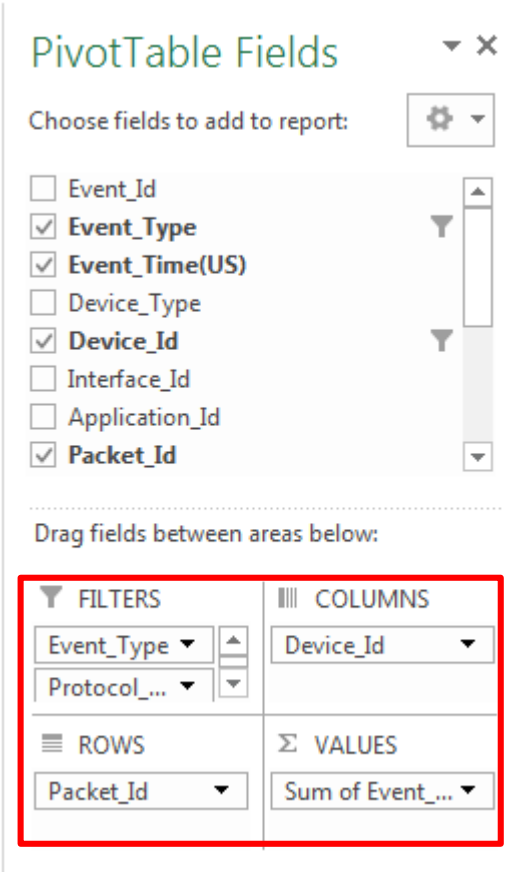


6.4.2.2 Application Jitter Analysis

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

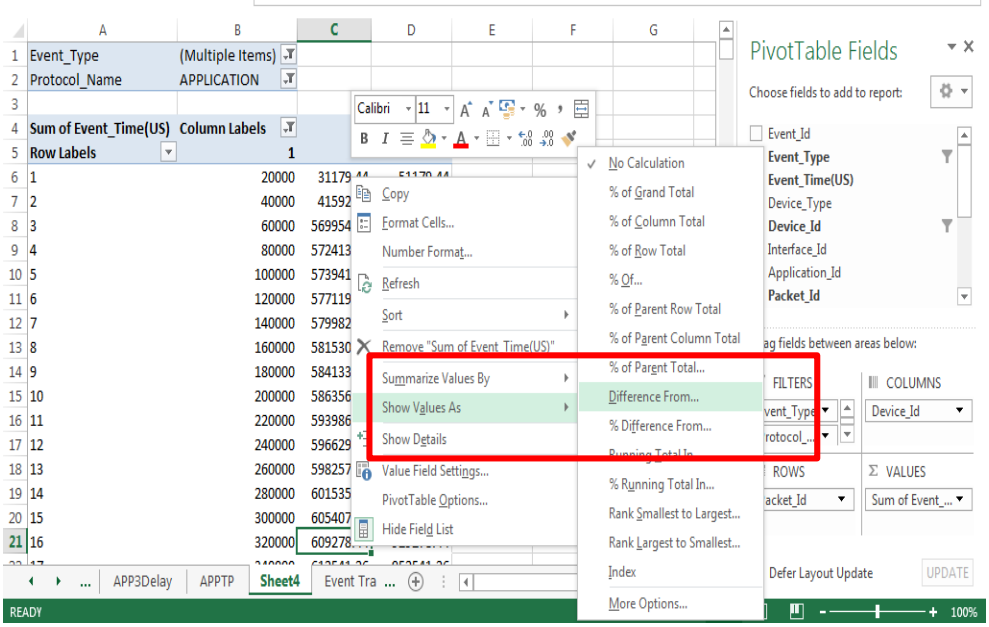
(Note: This calculation is only valid only when there is with no packet segmentation. This means that all Packet Sizes should be less than 1500B)

13. For Jitter Calculation Pivot table Fields should be as shown below

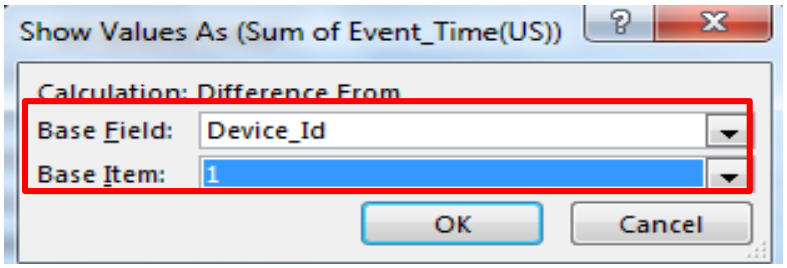


FILTERS to have of **Event_Type**, **Protocol_Name**, COLUMNS to have **Device_Id**, ROWS to have **Packet_Id** and VALUES to have **Event_Time** (Note that **Event_Time** is added only once and not twice as was done for delay calculation)

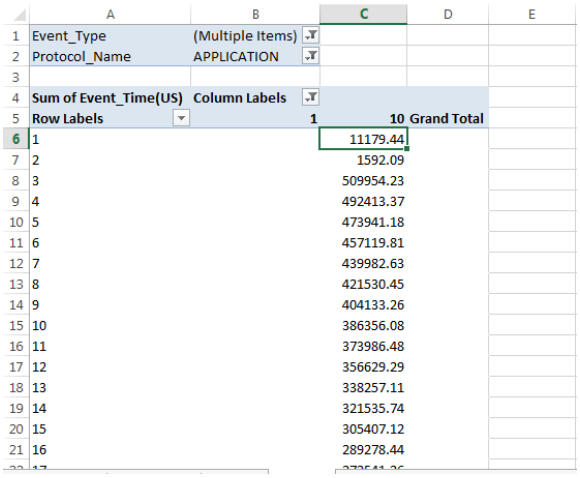
14. Then select any cell of **Destination node** column and **right click** and select **Show Values As** option, it dropdown a list in which you select **Difference From** option.



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



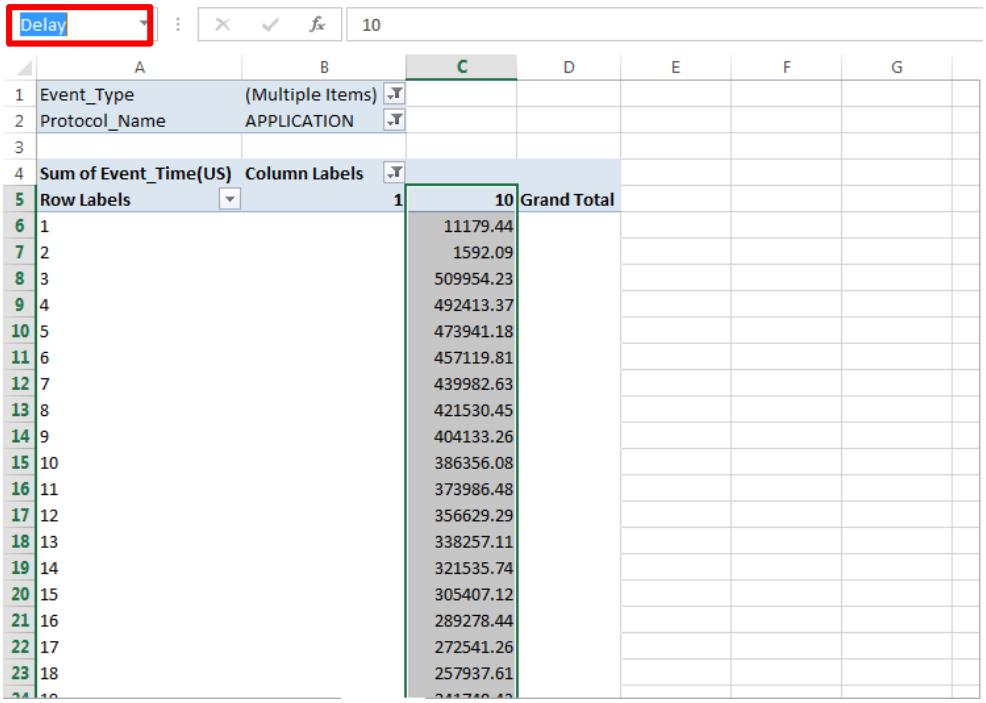
Then the Pivot Table will look like this



	A	B	C	D	E
1	Event_Type	(Multiple Items)			
2	Protocol_Name	APPLICATION			
3					
4	**Sum of Event_Time(US)**	**Column Labels**			
5	**Row Labels**	**1**	**10**	**Grand Total**	
6	1		11179.44		
7	2		1592.09		
8	3		509954.23		
9	4		492413.37		
10	5		473941.18		
11	6		457119.81		
12	7		439982.63		
13	8		421530.45		
14	9		404133.26		
15	10		386356.08		
16	11		373986.48		
17	12		356629.29		
18	13		338257.11		
19	14		321535.74		
20	15		305407.12		
21	16		289278.44		

And the destination node column shows the delay of each packet.

15. Place the mouse cursor on the top of Destination Id then left click it will automatically selects the rows of the destination Id, then select the Name Box and name the row.

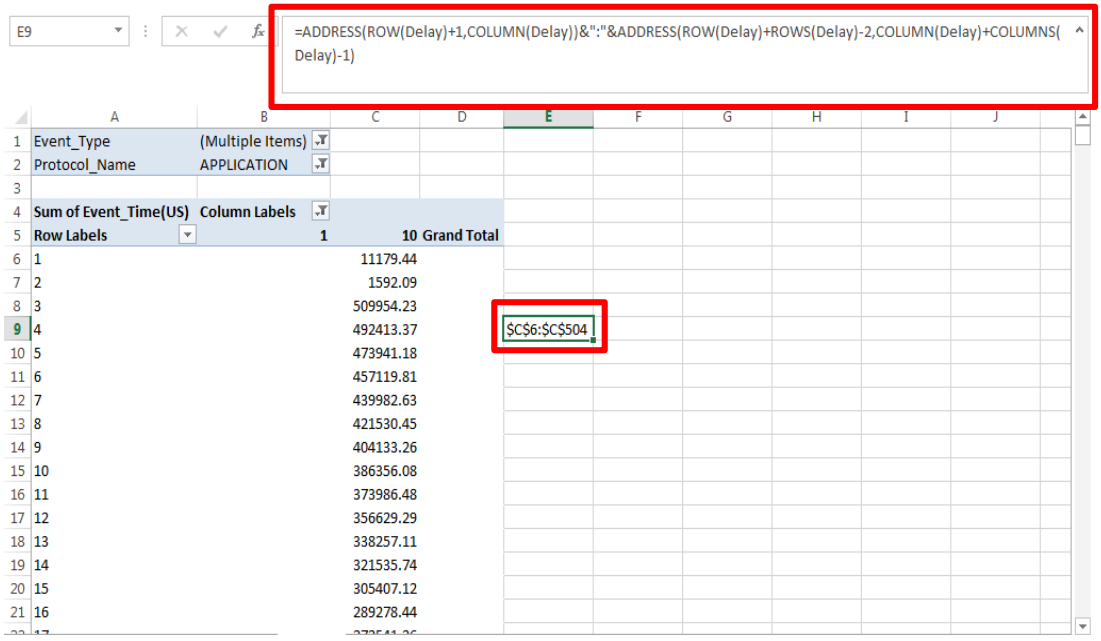


	A	B	C	D	E	F	G
1	Event_Type	(Multiple Items)					
2	Protocol_Name	APPLICATION					
3							
4	Sum of Event_Time(US)	Column Labels					
5	Row Labels	1	10	Grand Total			
6	1		11179.44				
7	2		1592.09				
8	3		509954.23				
9	4		492413.37				
10	5		473941.18				
11	6		457119.81				
12	7		439982.63				
13	8		421530.45				
14	9		404133.26				
15	10		386356.08				
16	11		373986.48				
17	12		356629.29				
18	13		338257.11				
19	14		321535.74				
20	15		305407.12				
21	16		289278.44				
22	17		272541.26				
23	18		257937.61				

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)

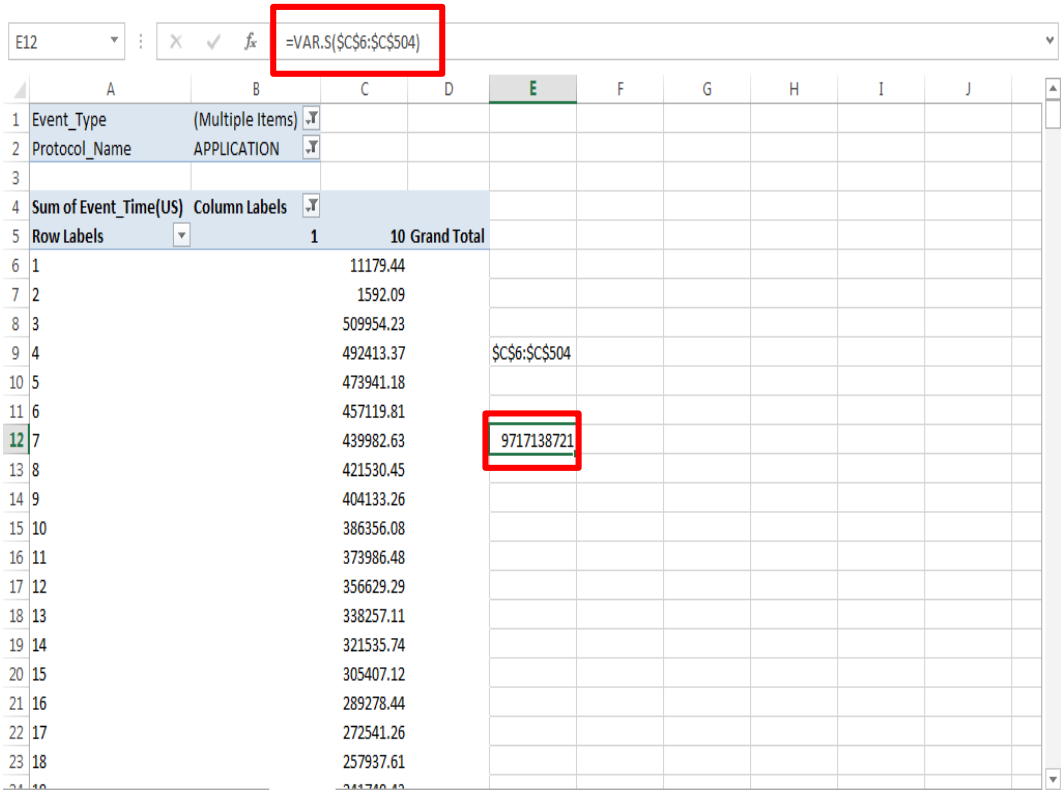
This will give you the **Row Address**



	A	B	C	D	E	F	G	H	I	J
1	Event_Type	(Multiple Items)								
2	Protocol_Name	APPLICATION								
3										
4	Sum of Event_Time(US)	Column Labels								
5	Row Labels	1	10	Grand Total						
6	1		11179.44							
7	2		1592.09							
8	3		509954.23							
9	4		492413.37		$C$6:$C$504					
10	5		473941.18							
11	6		457119.81							
12	7		439982.63							
13	8		421530.45							
14	9		404133.26							
15	10		386356.08							
16	11		373986.48							
17	12		356629.29							
18	13		338257.11							
19	14		321535.74							
20	15		305407.12							
21	16		289278.44							

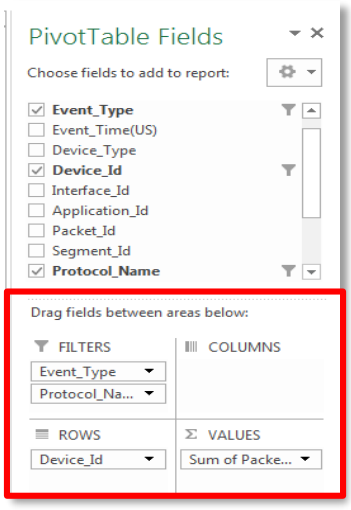
Then enter **=VAR.S(Row Address)**, where the **Row Address** is one that you got in the previous step.

This will give you the **Application Jitter.**

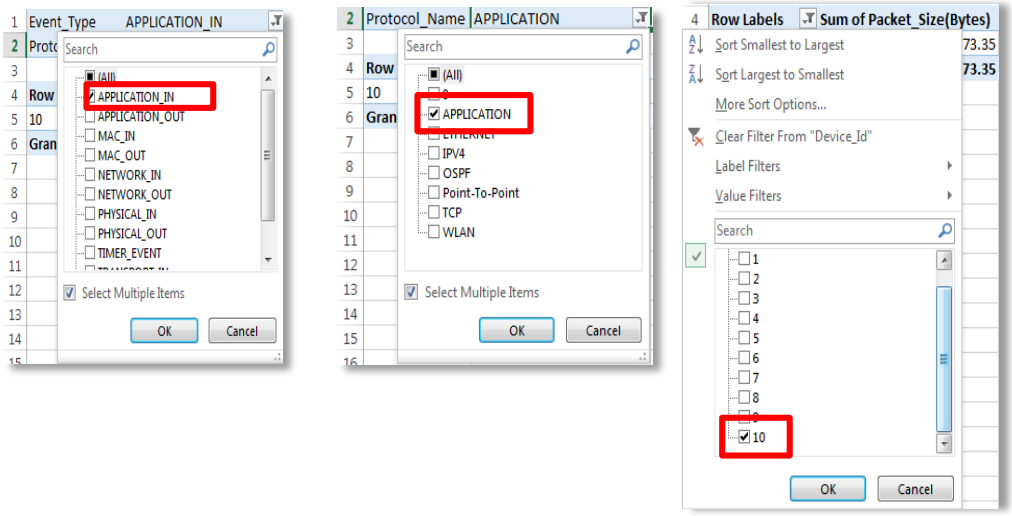


6.4.2.3 Application Throughput Analysis:

16. 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).**

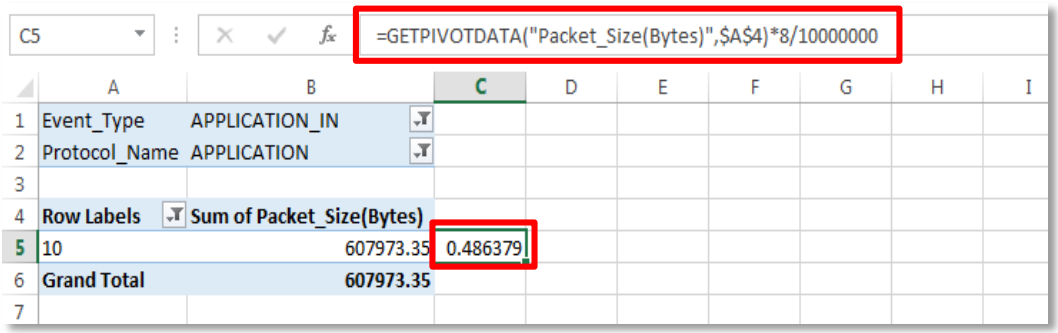


17. App Throughput = $\frac{Total\ Payload\ application\ succesfully\ received\ by\ the\ destination}{Simulation\ time}$

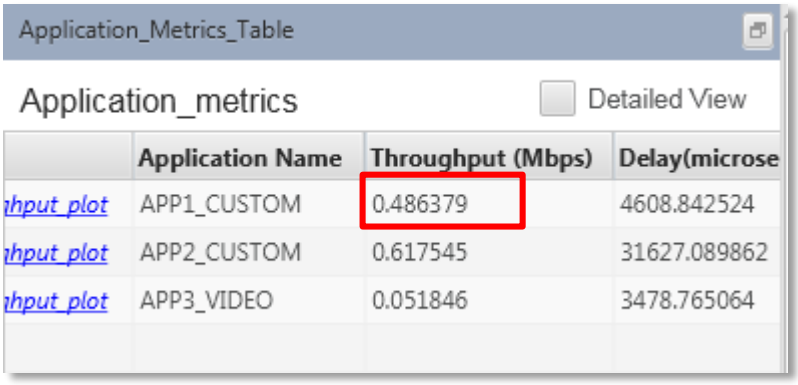
18. Then in an empty cell type

    =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**

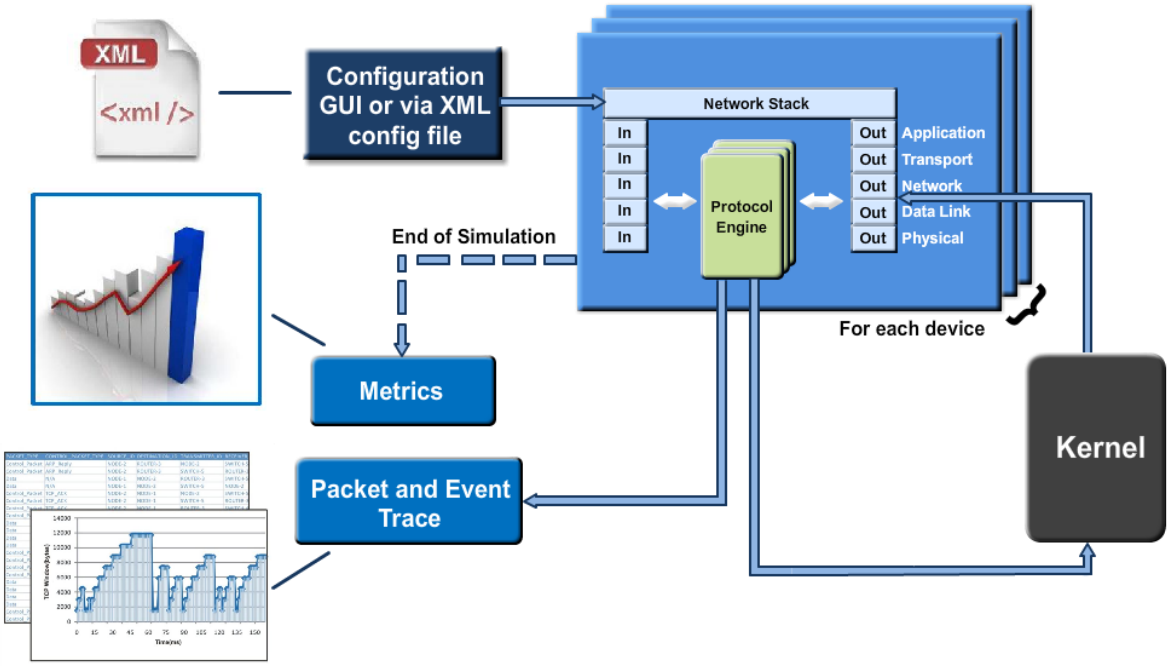


6.5 Event Trace (only in Standard/Pro Version)

6.5.1 NetSim Network Stack

NetSim's Network Stack forms the core of NetSim and its architectural aspects are diagrammatically explained below. Network Stack accepts inputs from the end-user in the form of Configuration file and the data flows as packets from one layer to another layer in the Network Stack.

All packets, when transferred between devices move up and down the stack, and all events in NetSim fall under one of these ten categories of events, namely, ***Physical IN, Data Link IN, Network IN, Transport IN, Application IN, Application Out, Transport OUT, Network OUT, Data Link OUT*** and ***Physical OUT***. The IN events occur when the packets are entering a device while the OUT events occur while the packet is leaving a device.



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

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

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

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

Event Trace:

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

Apart from a host of information, the event trace has two special information fields for diagnostics

a) A log of the file name and line number from where the event was generated (Please refer "**Custom Code in NetSim → Debugging your code → Via CLI**") and

b) Previous event which triggered the current event.

> ***Note:*** *Turning on Event Trace will slow down the simulation significantly*

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.

How to import Event Trace to Excel?

Refer Help on "**Generating Packet Trace How to import Packet Trace to Excel?"**

Event Trace Metrics

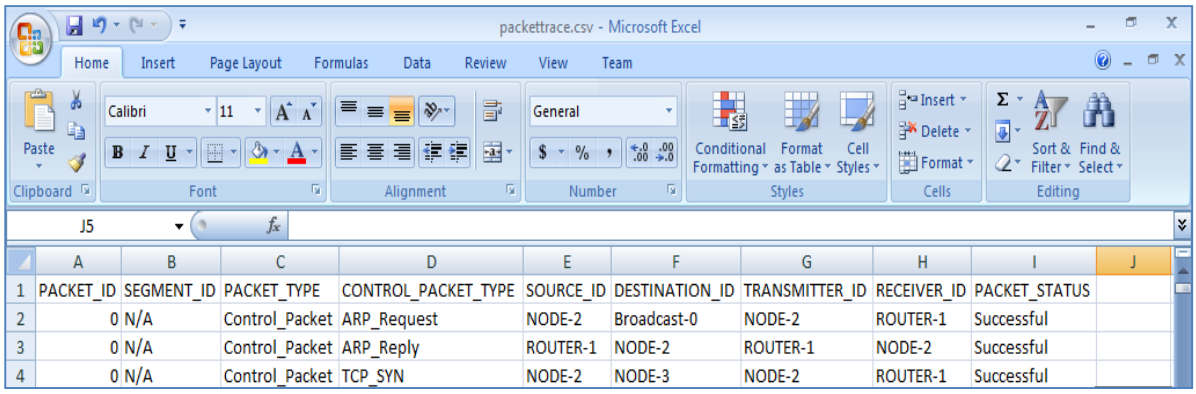
Event_Id	Specifies the ID of the Event
Event_Type	Specifies the type of event being performed, for eg - APPLICATION_IN, APPLICATION_OUT, MAC_OUT, MAC_IN, PHYSICAL_OUT, PHYSICAL_IN,etc
Event_Time	Specifies the time(in microseconds) at which the event is being executed
Device_Type	Specifies the type of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

6.6 Trace Data Analysis (only in Standard/Pro Version)

This section explains about how to analyze the data of a NetSim trace file.

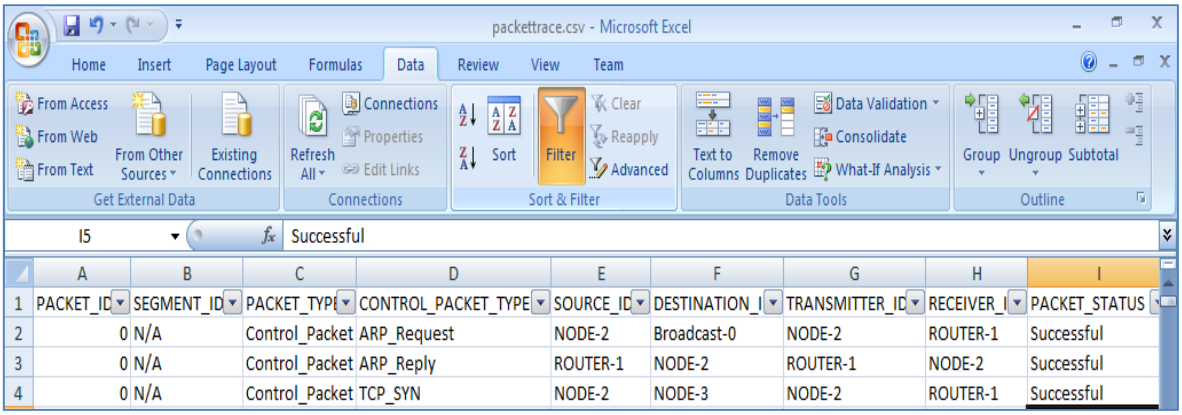
How to set filters to NetSim trace file

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



	A	B	C	D	E	F	G	H	I	J
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	PACKET_STATUS	
2	0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Successful	
3	0	N/A	Control_Packet	ARP_Reply	ROUTER-1	NODE-2	ROUTER-1	NODE-2	Successful	
4	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful	

Step 2: Go to Data menu and click on filter icon to enable the auto filter.

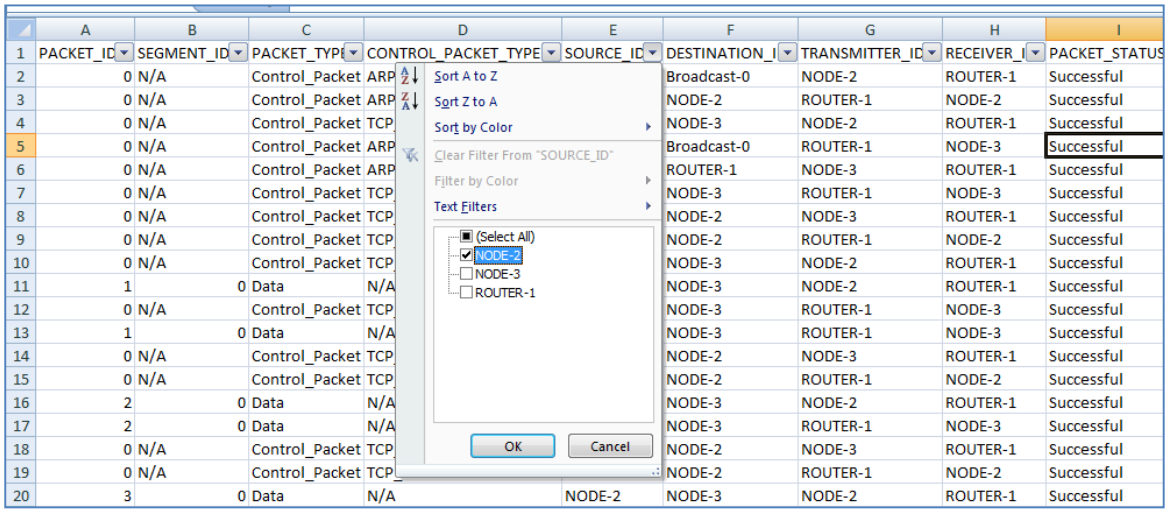


	A	B	C	D	E	F	G	H	I
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_I	TRANSMITTER_ID	RECEIVER_I	PACKET_STATUS
2	0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Successful
3	0	N/A	Control_Packet	ARP_Reply	ROUTER-1	NODE-2	ROUTER-1	NODE-2	Successful
4	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful

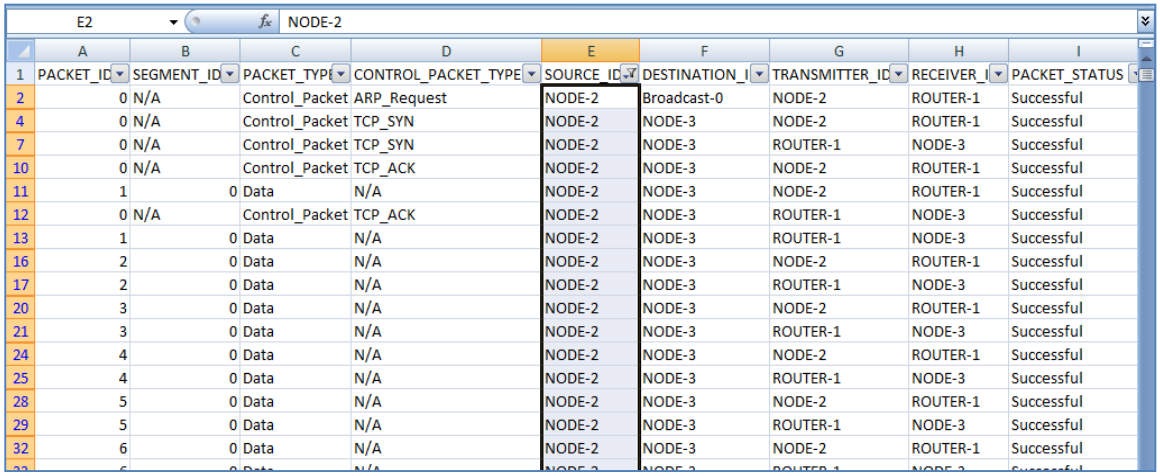
Step 3: Click the arrow in the header of the column you want to filter. In the list of text or numbers, uncheck the (Select All) box at the top of the list, and then check the boxes of the items you want to show.

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.



	A	B	C	D	E	F	G	H	I
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_I	TRANSMITTER_ID	RECEIVER_I	PACKET_STATUS
2	0	N/A	Control_Packet	ARP		Broadcast-0	NODE-2	ROUTER-1	Successful
3	0	N/A	Control_Packet	ARP		NODE-2	ROUTER-1	NODE-2	Successful
4	0	N/A	Control_Packet	TCP		NODE-3	NODE-2	ROUTER-1	Successful
5	0	N/A	Control_Packet	ARP		Broadcast-0	ROUTER-1	NODE-3	Successful
6	0	N/A	Control_Packet	ARP		ROUTER-1	NODE-3	ROUTER-1	Successful
7	0	N/A	Control_Packet	TCP		NODE-3	ROUTER-1	NODE-3	Successful
8	0	N/A	Control_Packet	TCP		NODE-2	NODE-3	ROUTER-1	Successful
9	0	N/A	Control_Packet	TCP		NODE-2	ROUTER-1	NODE-2	Successful
10	0	N/A	Control_Packet	TCP		NODE-3	NODE-2	ROUTER-1	Successful
11	1	0	Data	N/A		NODE-3	NODE-2	ROUTER-1	Successful
12	0	N/A	Control_Packet	TCP		NODE-3	ROUTER-1	NODE-3	Successful
13	1	0	Data	N/A		NODE-3	ROUTER-1	NODE-3	Successful
14	0	N/A	Control_Packet	TCP		NODE-2	NODE-3	ROUTER-1	Successful
15	0	N/A	Control_Packet	TCP		NODE-2	ROUTER-1	NODE-2	Successful
16	2	0	Data	N/A		NODE-3	NODE-2	ROUTER-1	Successful
17	2	0	Data	N/A		NODE-3	ROUTER-1	NODE-3	Successful
18	0	N/A	Control_Packet	TCP		NODE-2	NODE-3	ROUTER-1	Successful
19	0	N/A	Control_Packet	TCP		NODE-2	ROUTER-1	NODE-2	Successful
20	3	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful



E2 NODE-2

	A	B	C	D	E	F	G	H	I
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_I	TRANSMITTER_ID	RECEIVER_I	PACKET_STATUS
2	0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Successful
4	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
7	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
10	0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
11	1	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
12	0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
13	1	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
16	2	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
17	2	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
20	3	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
21	3	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
24	4	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
25	4	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
28	5	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
29	5	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
32	6	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful

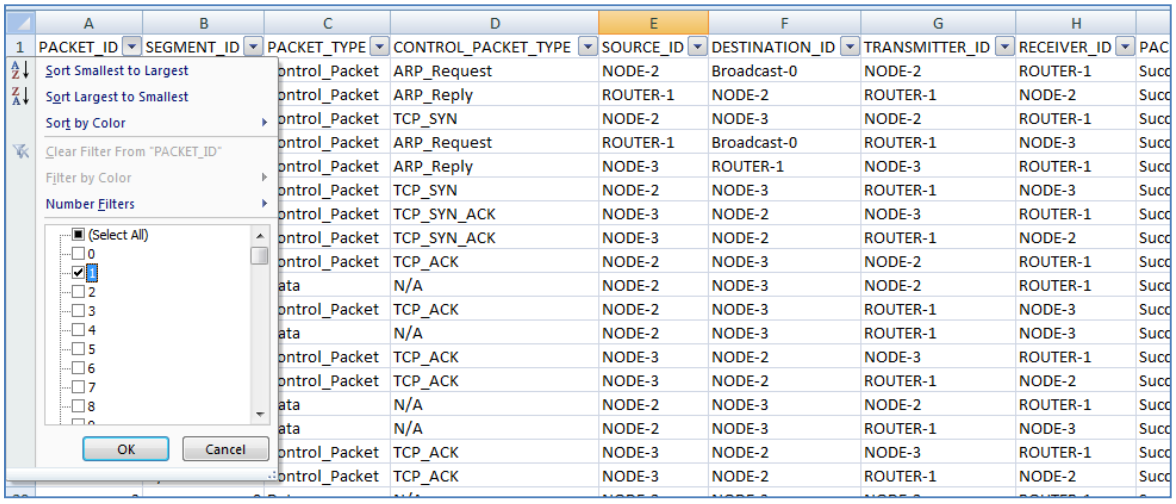
Typically filters can be set to observe “Errored/Collided/Successful “packets, packets of destination and packets of source.

Observing packet flow in the Network through packet trace file

Step 1: Open the packet trace file.

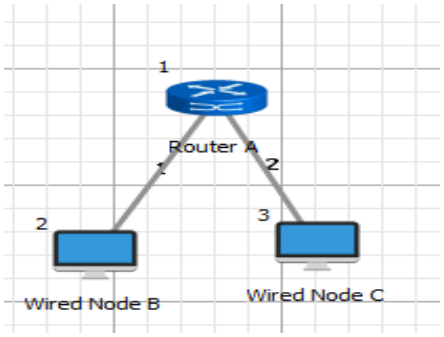
Step 2: Set the filter.

Step 3: 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.

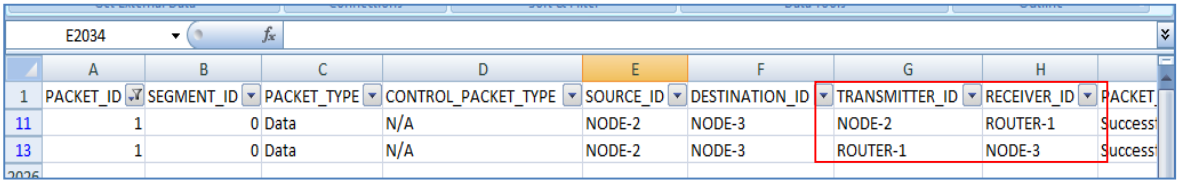


PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	PAC
1	0	ontrol_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Succ
1	0	ontrol_Packet	ARP_Reply	ROUTER-1	NODE-2	ROUTER-1	NODE-2	Succ
1	0	ontrol_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Succ
1	0	ontrol_Packet	ARP_Request	ROUTER-1	Broadcast-0	ROUTER-1	NODE-3	Succ
1	0	ontrol_Packet	ARP_Reply	NODE-3	ROUTER-1	NODE-3	ROUTER-1	Succ
1	0	ontrol_Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3	Succ
1	0	ontrol_Packet	TCP_SYN_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	Succ
1	0	ontrol_Packet	TCP_SYN_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	Succ
1	0	ontrol_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	Succ
1	0	ata	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Succ
1	0	ontrol_Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3	Succ
1	0	ata	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Succ
1	0	ontrol_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	Succ
1	0	ontrol_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	Succ
1	0	ata	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Succ
1	0	ata	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Succ
1	0	ontrol_Packet	TCP_ACK	NODE-3	NODE-2	NODE-3	ROUTER-1	Succ
1	0	ontrol_Packet	TCP_ACK	NODE-3	NODE-2	ROUTER-1	NODE-2	Succ

Scenario is as shown below and traffic flow is from Wired Node 1 to Wired Node 2.



Flow of packet 1 can be observed from the packet trace as shown below.



E2034

	A	B	C	D	E	F	G	H	
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	PACKET_
11	1	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Success
13	1	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Success

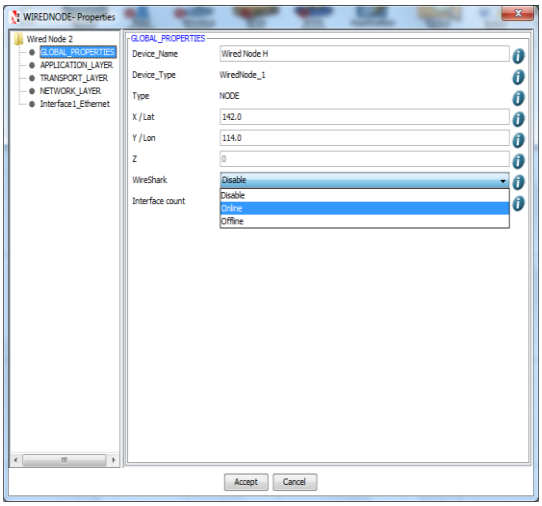
Note: *In the trace file device IDs are shown not device names. Wired Node 1's ID is 2 so it is Shown as NODE-2, Wired Node 2's ID is 3 so it is shown as NODE -3, Router-1' ID is 1 so it is shown as ROUTER-1. Device IDs are shown on the top of the device icon in the above scenario.*

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.

6.7 Packet Capture & analysis using Wireshark (www.wireshark.com) (only in Standard/Pro Version)

6.7.1 Enabling Wireshark in the network scenario

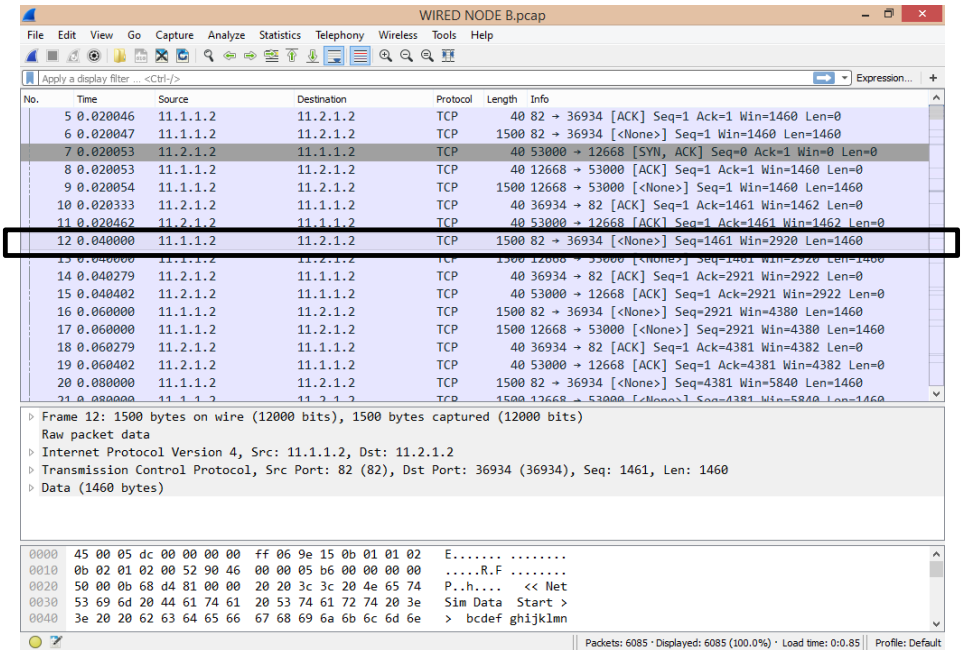
In NetSim, to enable packet capture in Wireshark, **Right Click** on the device where Wireshark will capture packets. In the properties, go to Global_Properties and set the Wireshark parameter as Online



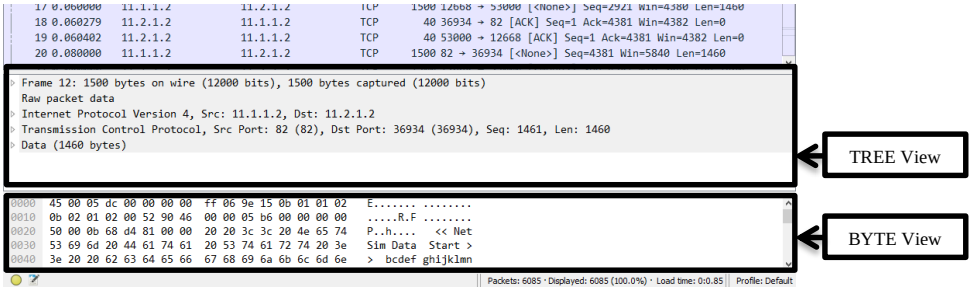
6.7.2 Viewing captured packets

If enabled, Wireshark 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.

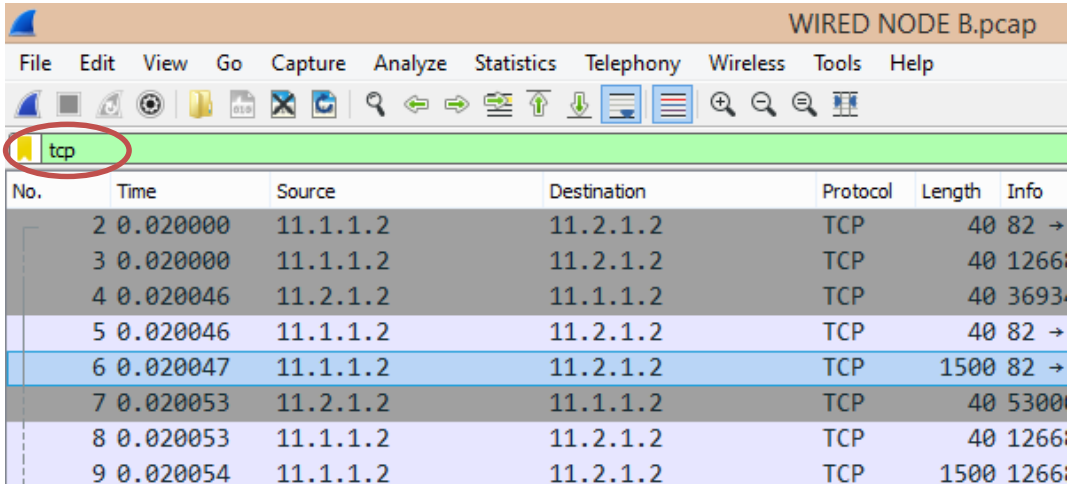


In the above figure, the details of the packet are displayed in both tree form and bytes form. In the tree form, user can expand the data by clicking on the part of the tree and view detailed information about each protocol in each packet.

6.7.3 Filtering captured packets

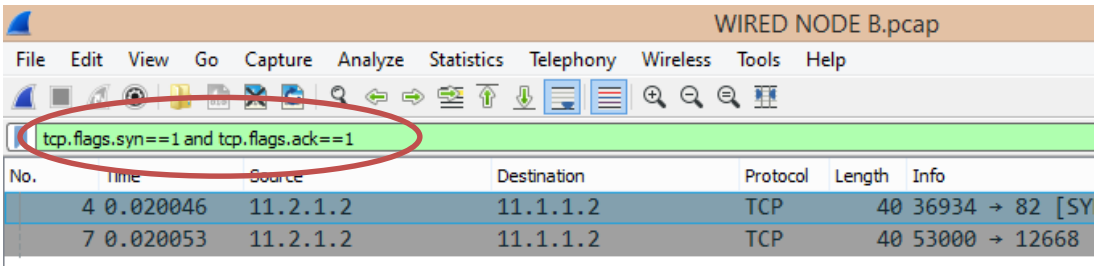
Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. Packets can be filtered by protocol, presence of a field, values of fields etc.

To select packets based on protocol, type the protocol in which you are interested in the ***Filter:*** field of the Wireshark window and press enter to initiate the filter. In the figure below, *tcp* protocol is filtered.



You can also build display filters that compare values using a number of different comparison operators like ==, != , >, <, <=, etc.

Following is an example displaying filtered packets whose SYN Flag and ACK Flag are set to 1 in a TCP Stream.



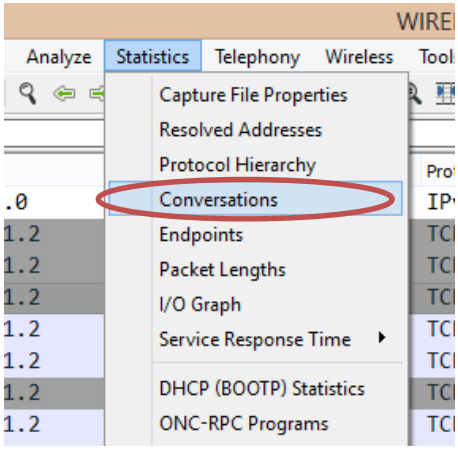
6.7.4 Analyzing packets in Wireshark

6.7.4.1 Analyzing Conversation using graphs

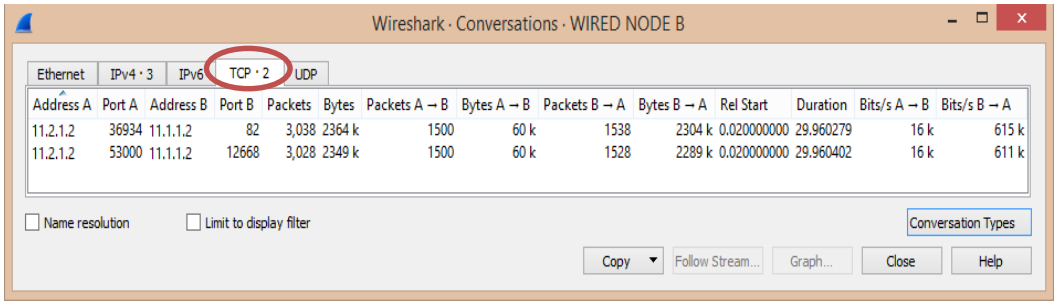
A network conversation is the traffic between two specific endpoints. For example, an IP conversation is all the traffic between two IP addresses.

In Wireshark,

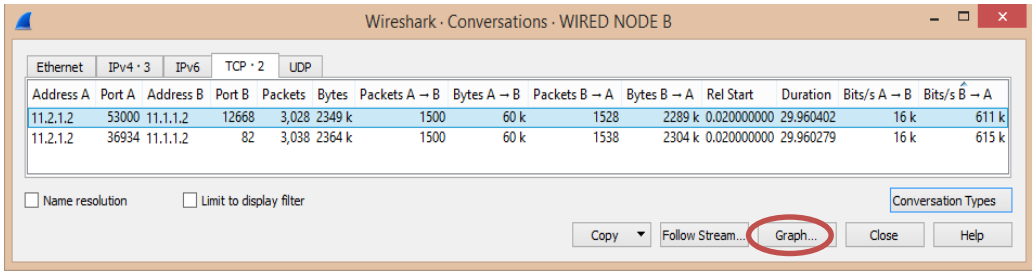
Go to Statistics Menu→ Conversations



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 and can 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

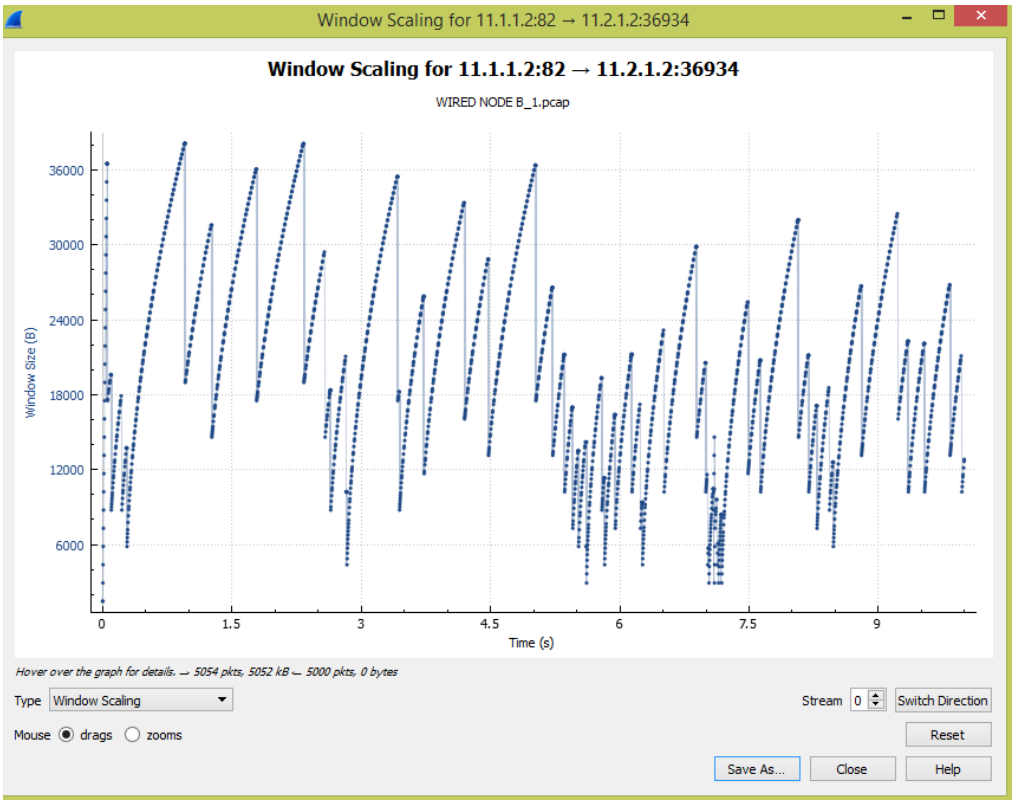
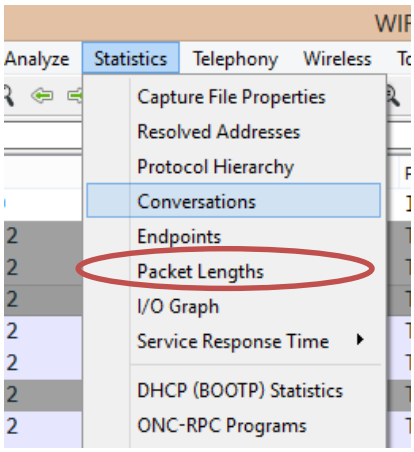


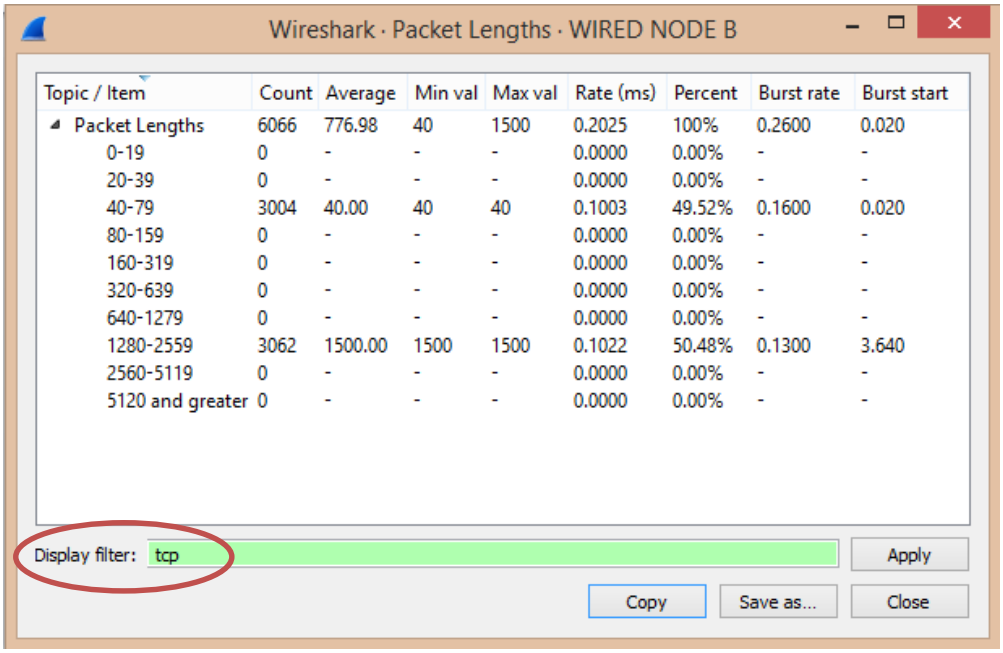
Fig: TCP congestion window Plot

6.7.4.2 Comparing the packet lengths

To analyze the packet sizes of all packets transmitted, go to Statistics Menu→Packet lengths.

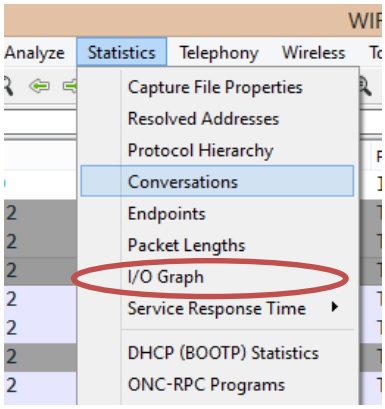


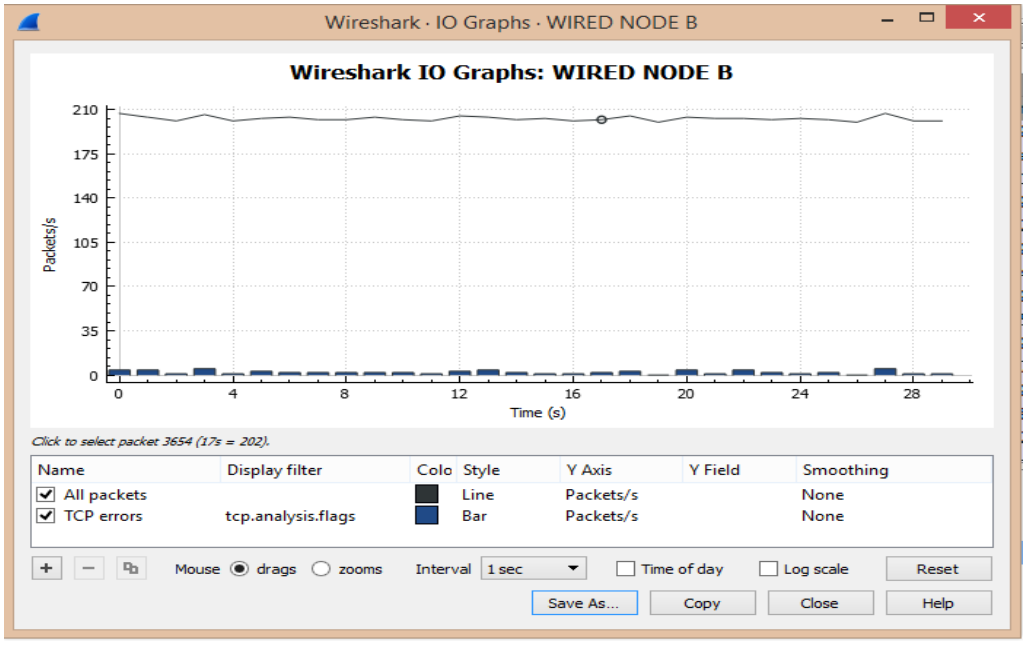
Users can also set filter to analyze a collection of specific packets only. For example, *tcp* filter is set to obtain the packet length below:



6.7.4.3 Creating IO graphs

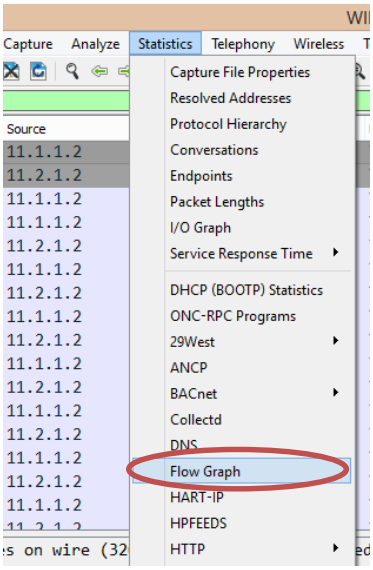
To get the graph, go to Statistics Menu →IO Graph.





6.7.4.4 Creating Flow graphs

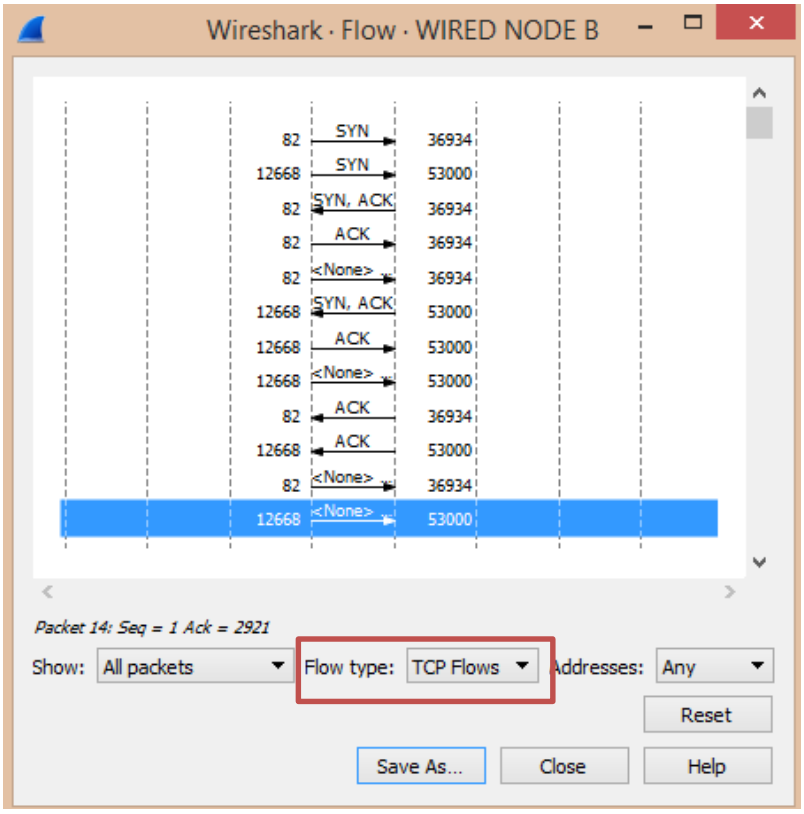
The flow graph feature provides a quick and easy to use way of checking connections between a client and a server. It can show where there might be issues with a TCP connection, such as timeouts, re-transmitted frames, or dropped connections.



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:



7 Custom code in NetSim

7.1 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 and replacing NetSim's default DLL with the user's custom DLL.

There are various important steps in this process, and each of these steps has various options as explained in the subsequent pages:

7.1.1 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.

> ***Note:*** *If the DLL needs to be debugged then build the DLL in "debug" mode or else build in "release" mode.*

Dynamic link libraries (DLLs) can be built using different compilers:

(a) Using Microsoft Visual Studio Compiler

a) Compiler: Microsoft Visual Studio

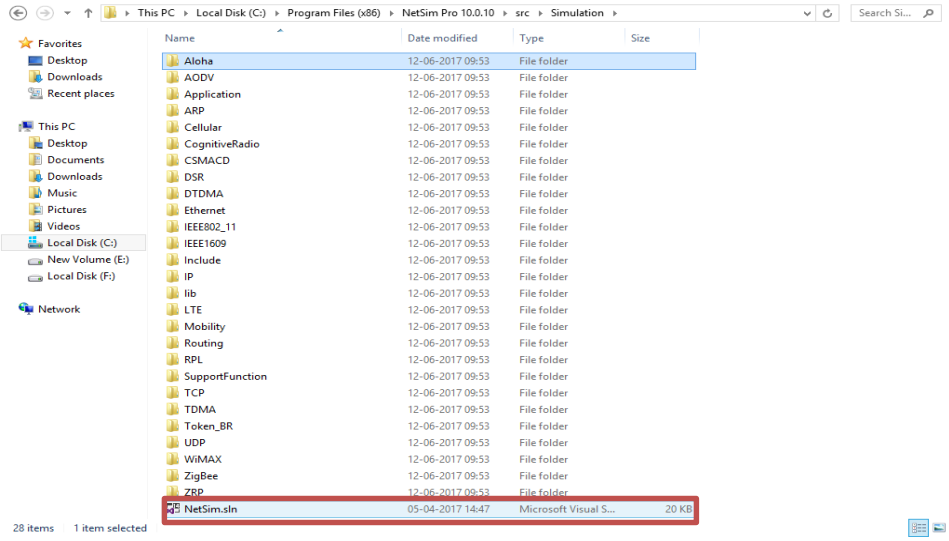
To create the DLL follow the steps given below.

> ***Note:*** *Make sure that Visual Studio 2015 is installed in your computer*

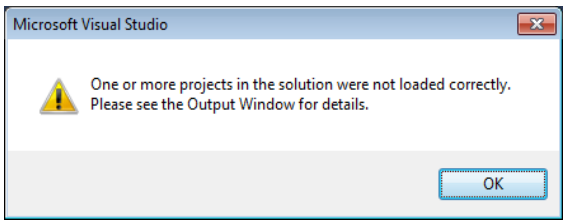
Step 1: Go to *"../NetSim Standard/src/Simulation"* folder and double click on "NetSim.sln" file to open the solution in visual studio.

> ***Note:*** *It is recommended to keep a separate copy of the "../NetSim Standard/src/Simulation" folder to have the backup of the original source code.*

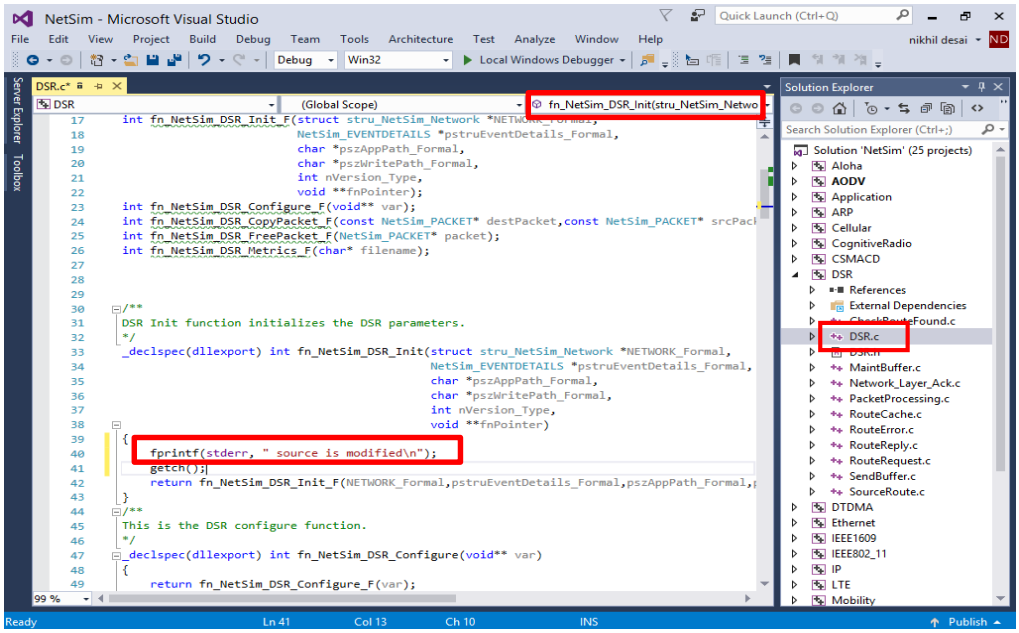
b)



Note: *Sometimes you may get the following warning message. Then click on ok and proceed.*



Step 2: Inside **Solution Explorer** pane in Visual Studio 2015, double click on DSR. Then open DSR.C file by double clicking on it. To the right of global space ,we have several functions ,we have to choose fn_Netsim_DSR_Init(stru_Netsim_Network*NETWORK) Simalarly for all protocols choose **Init( )** fn to add the custom code.



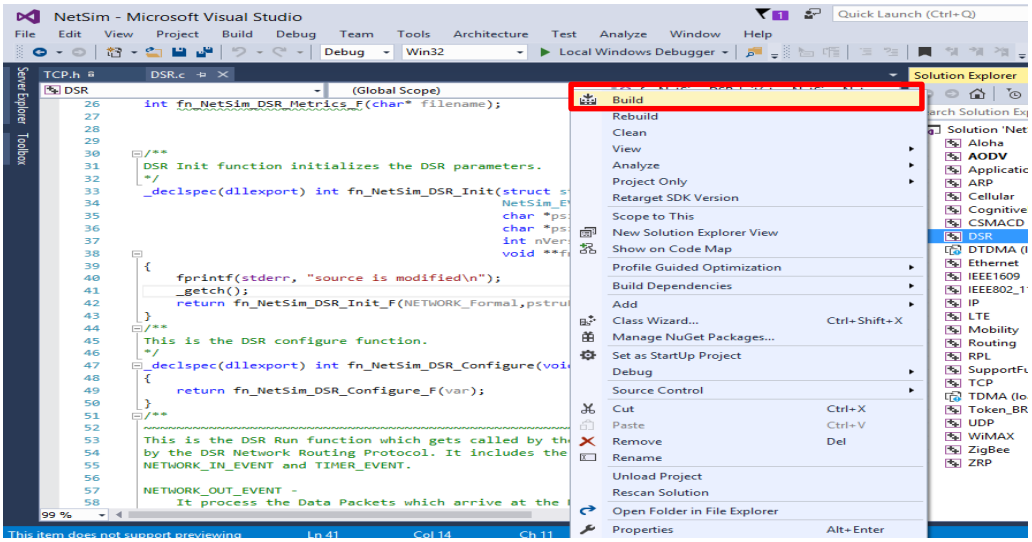
Step 3: Add ***fprintf (stderr, "Source is modified\n");*** statement inside the source code of DSR as shown below to print "Source is modified".

> ***Note:*** *- Use fprintf (stderr,"Hello World") instead of printf ( ) to write to standard out. stdout is redirected to LogFile.txt which is present in <iopath>. So you can see "Hello World" Statement (or whatever you write inside the printf) inside the LogFile.txt.*

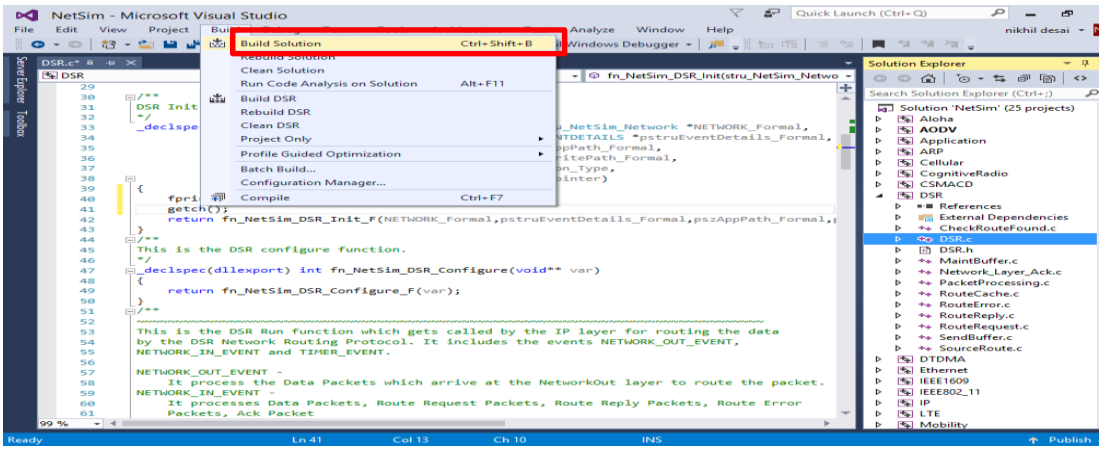
Once this is done click to save the changes and overwrite the file (in case of write protection).

7.1.2 Building Dlls

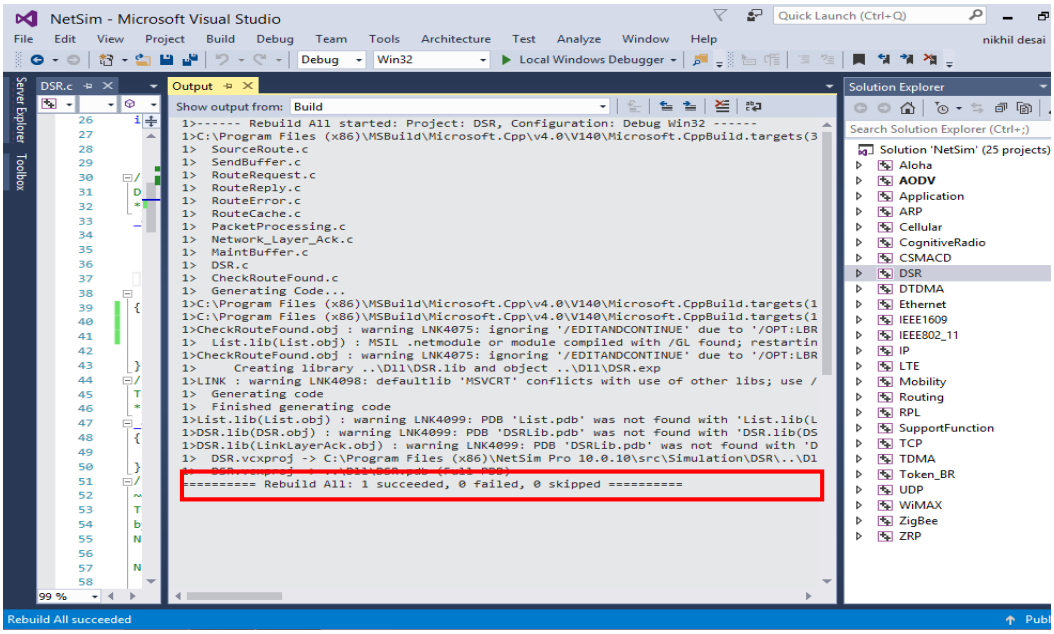
Step 1: Right Click on the Project (in this case DSR) inside the Solution Explorer pane and select Build to create DLL file of that specific protocol only



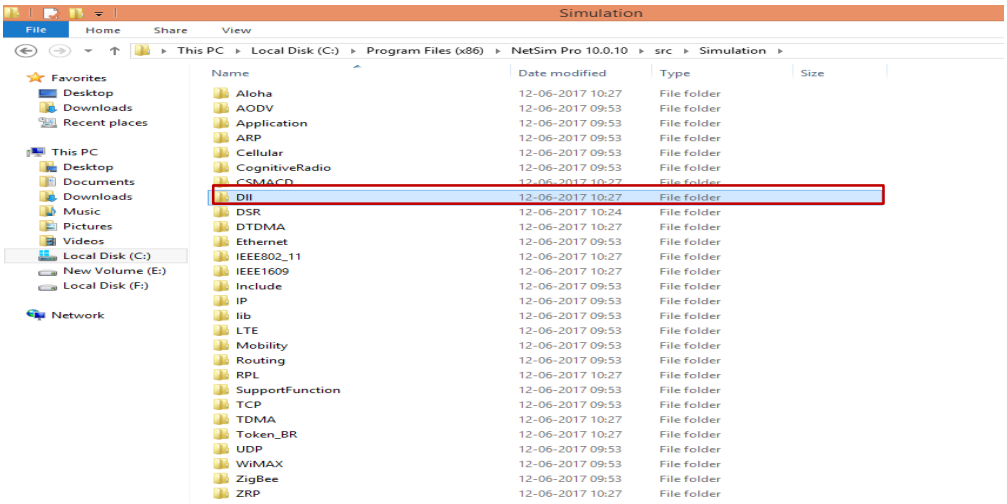
NOTE: To create DLL file of all the protocols actively displayed inside solution pane, click on "Build→Build Solution"



Check that the build is successful and necessary DLL files have been built.



Step 2: Inside the simulation folder (present on the desktop) a Dll folder will be created and inside, the DLL's are built.

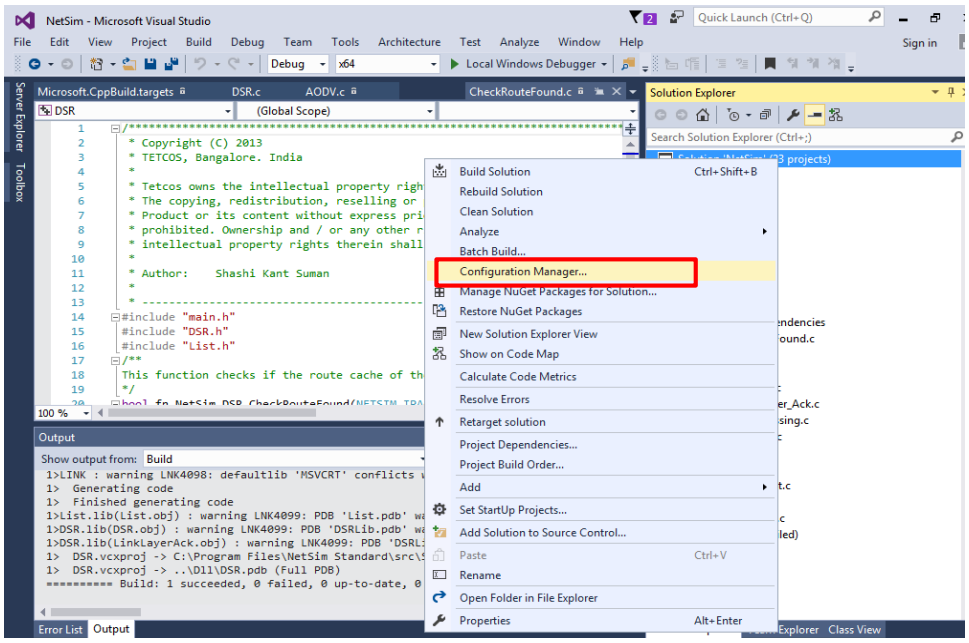


It will contain custom DLL's which have been built per the modifications done.

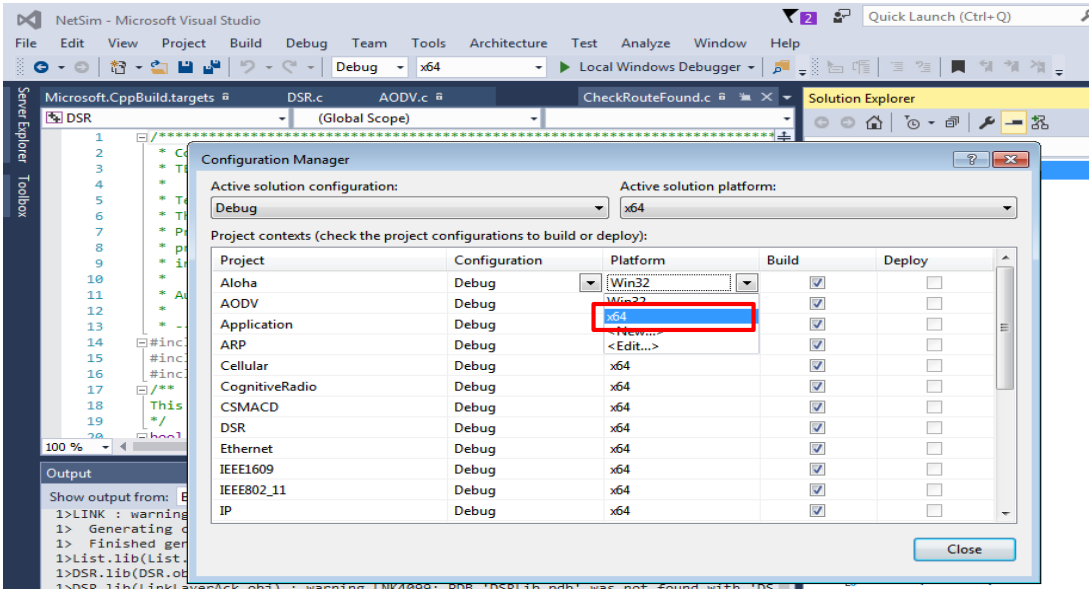
7.1.3 Building 64 Bit DLL in NetSim

If you are using 64 bit windows OS with a 64 bit build of NetSim installed, then the following steps explain to build a 64 bit custom DLL in NetSim.

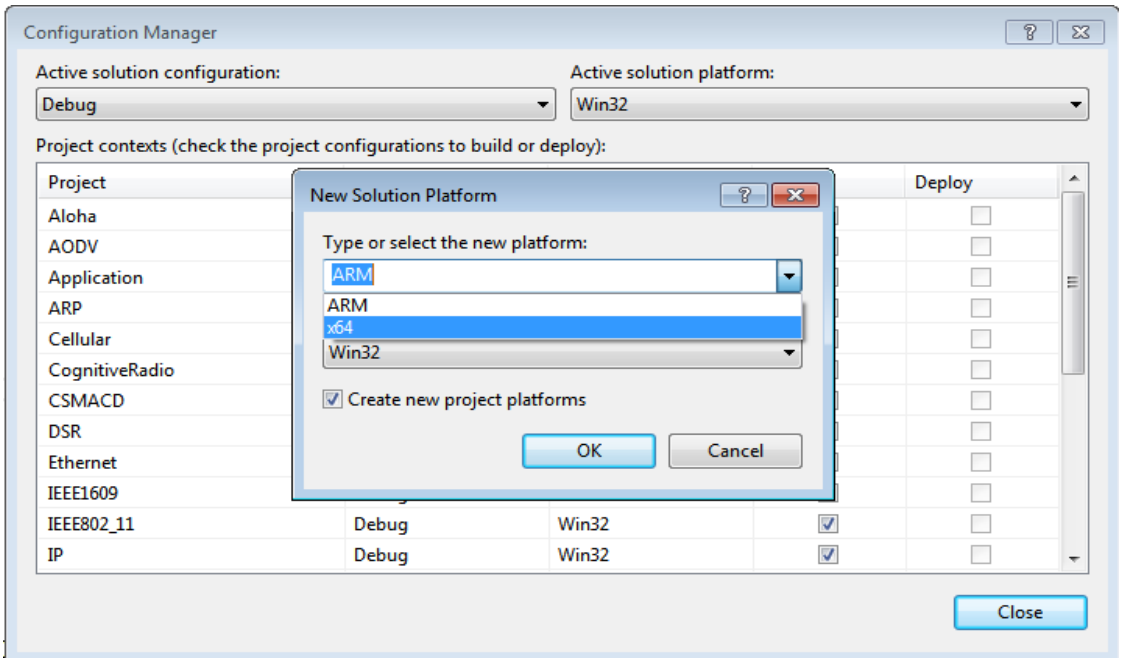
Step 1: Right Click on the Project (in this case DSR) inside the Solution Explorer pane and select Build to create DLL file of that specific protocol only. Right click on the **Solution Explorer** pane & Go to "**Configuration Manager**".



Step 2: In "project contexts", select the "platform" for each protocol as "x64".

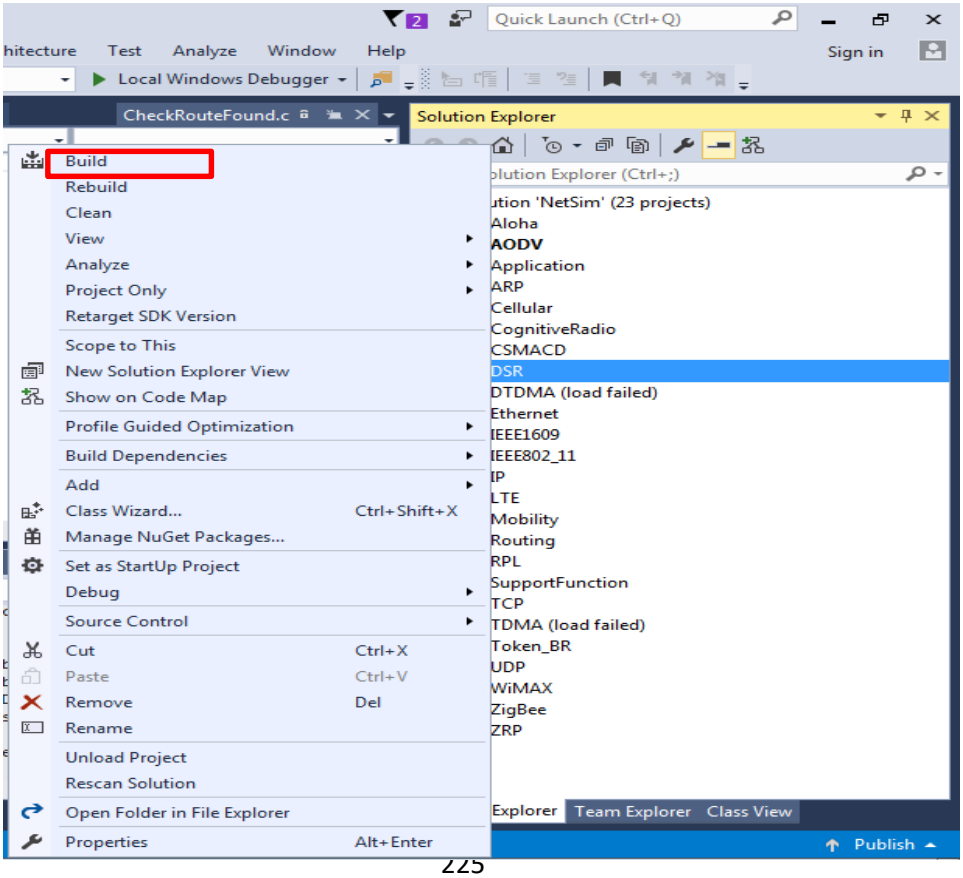


Step 2: In “configuration Manager…” under “Active solution platform” select new select x64

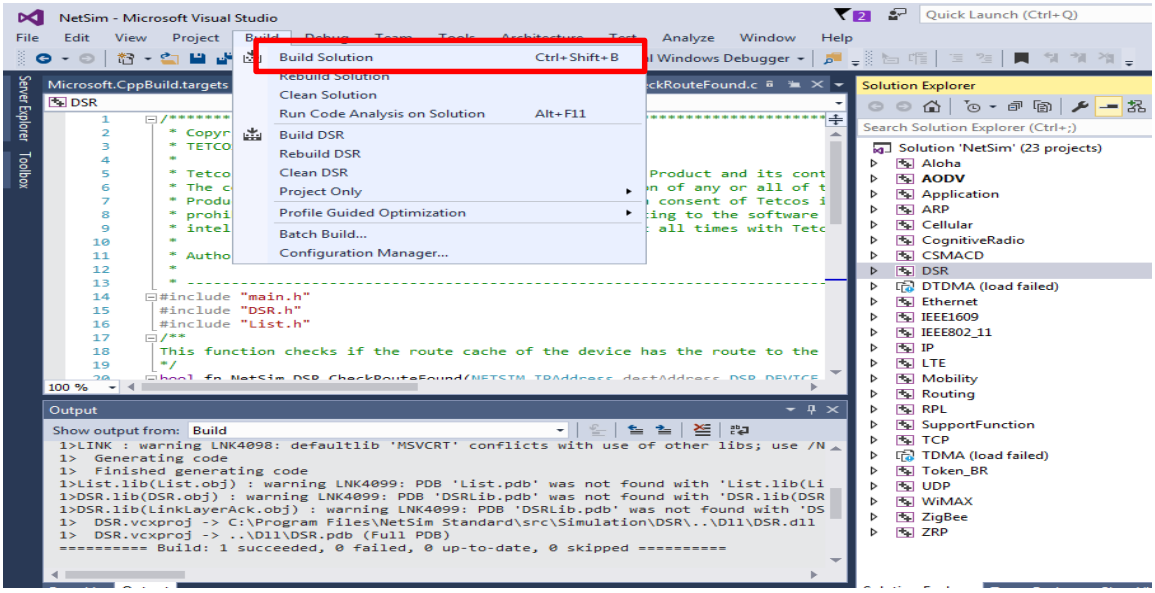


Step 3: After changing the platform to x64 for each protocol, then click on “close” button.

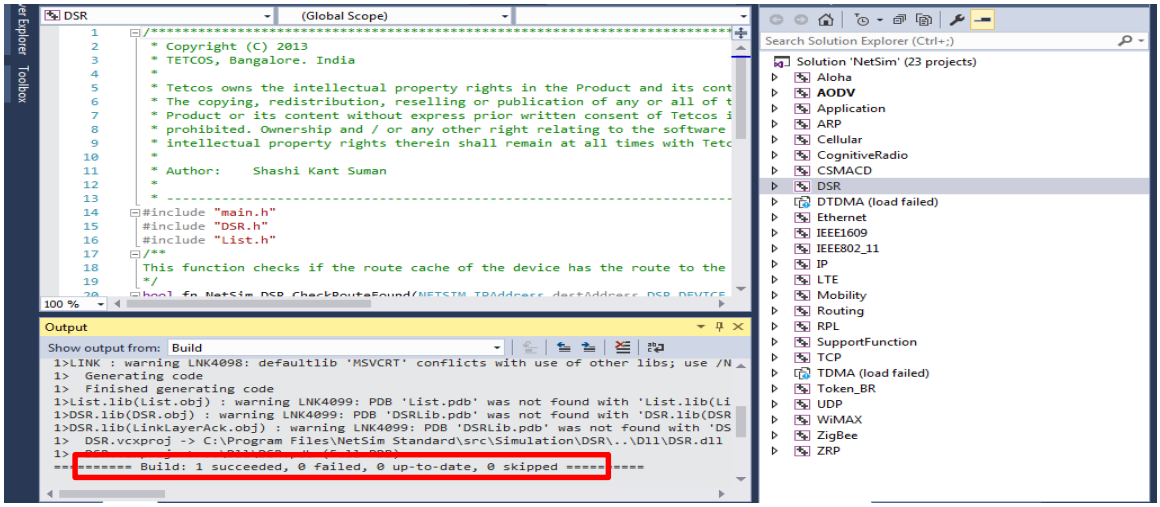
Step 4: Right Click on the Project (in this case DSR) inside the Solution Explorer pane and select “Build” to create DLL file of that specific protocol only.



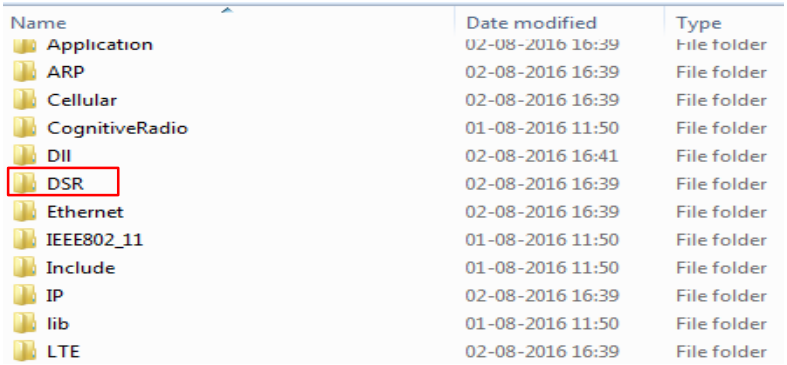
NOTE: To create DLL file for all the protocols actively displayed inside solution panel, "Debug → Build Solution"



Check that the build is successful and necessary DLL files have been built.



Step 7: Inside the simulation folder (present on the desktop) a Dll folder will be created and inside, the newly built DLL's would be present.



7.1.4 Linking Dlls

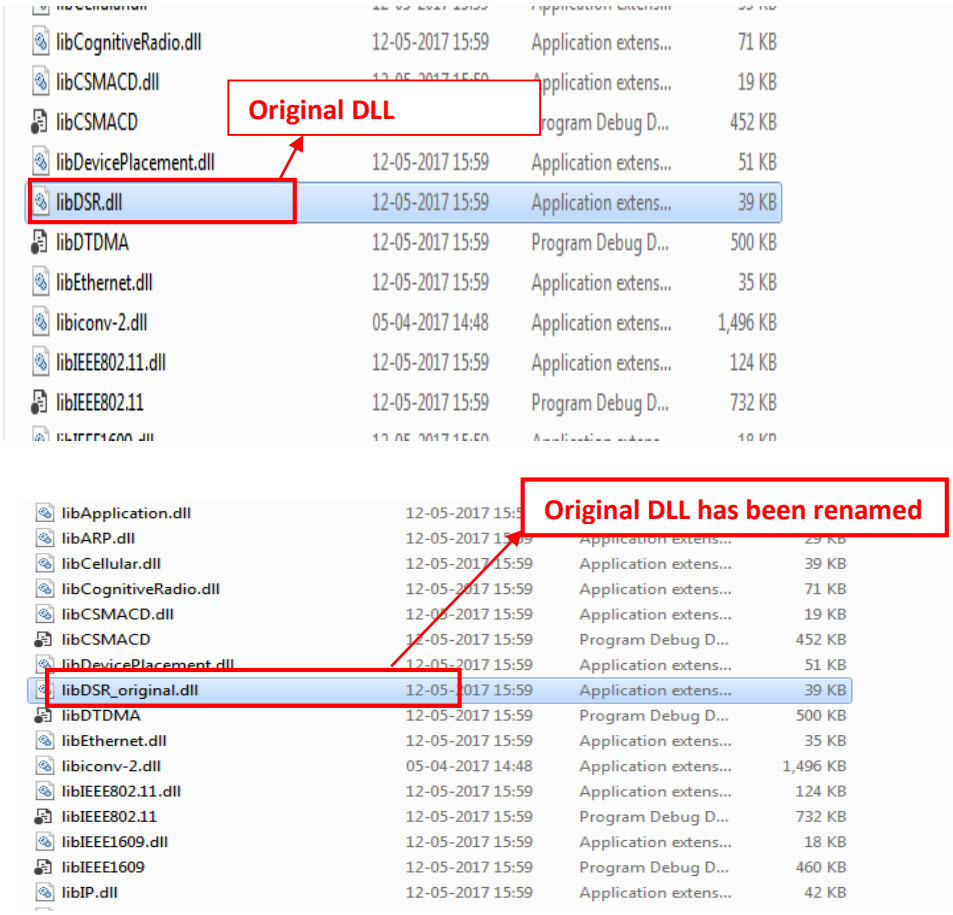
The procedure to link custom dynamic link libraries (DLLs) is explained below. This procedure is common for any linking of any custom DLL to NetSim.

Now we will link the customized libDSR.dll to the NetSim.

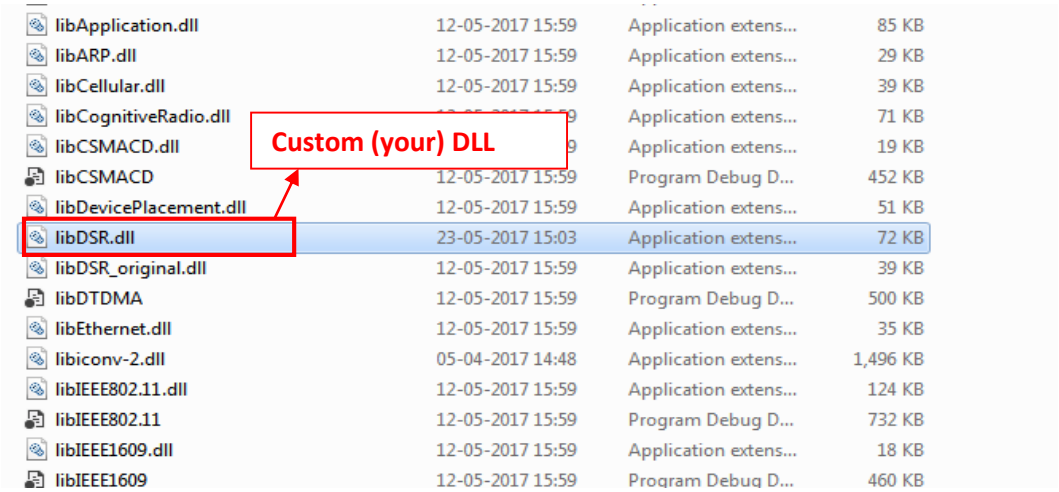
Step 1: Copy the newly created DLL file of customized DSR from "../NetSim Standard/src/Simulation" folder

Step 2: Go to "**NetSim Standard→bin**" folder. Here one would find default DLLs for respective protocols as shown in next figure.

Step 3: Rename the default DLL: It is recommended to rename the default DLL as original dll thereby saving a copy of DLL (simulation folder) in bin path . For example libDSR.dll has been renamed as libDSR_original.dll as shown below.



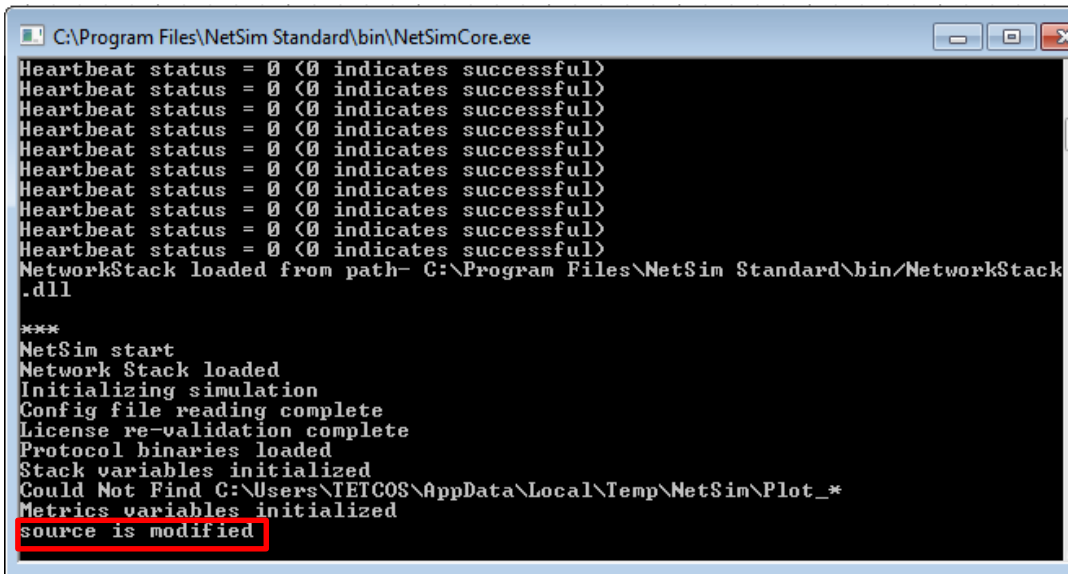
Step 4: Place the custom DLL obtained from Step 1 inside the "NetSim Standard → bin" folder.



> ***Note**: **Ensure that the name of the DLL which is pasted is exactly the same as the original DLL (before it was renamed)**. For the example case of DSR, the DLL must be libDSR.dll.*

7.1.5 Running Simulation

User can run the simulation via GUI (Please refer section 3). In this case, user can create a scenario in any network which involves DSR protocol. Running the simulation with the custom DLL will give desired output to the user as shown below.



Press enter then simulation will continue.

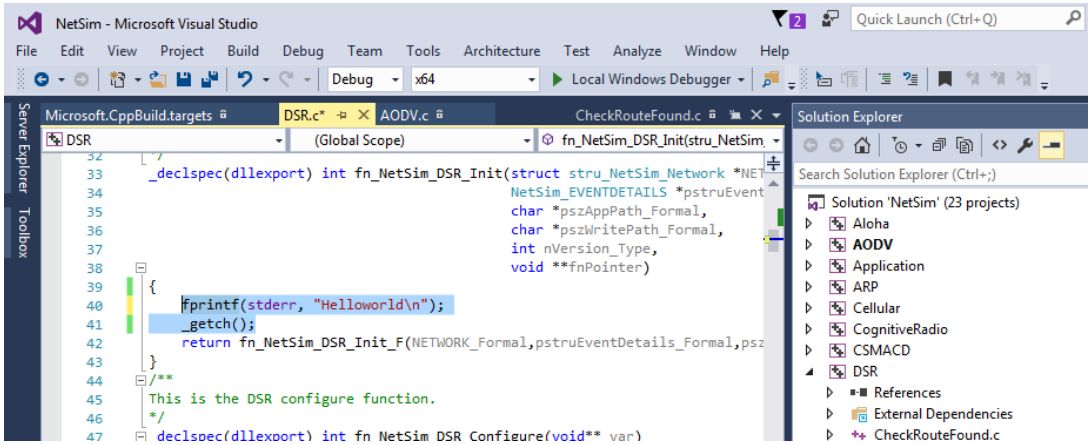
7.2 Implementing your code - Examples

7.2.1 Hello World Program

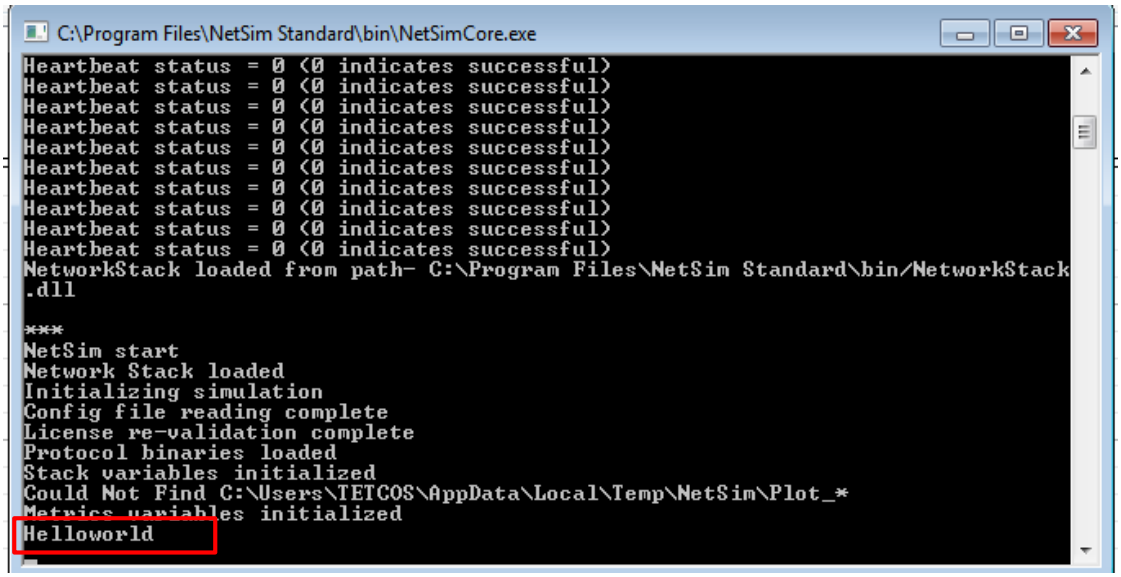
Objective: Print Hello World from DSR protocol.

Implementation: Add fprintf (stderr, "<MESSAGE>") statement inside the source code of DSR as shown below to print "Hello World" when custom built dll is executing.

> ***Note:*** *- Use fprintf (stderr,"Hello World") instead of printf ( ) to write to standard out. stdout is redirected to LogFile.txt which is present in <iopath>. So you can see "Hello World" Statement (or whatever you write inside the printf) inside the LogFile.txt.*



Create DLL and Link the DLL to the NetSim as explained in Section 7.1. And run the simulation, you can see the following output on the console.



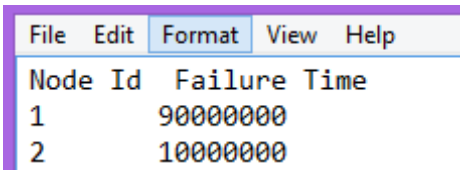
Press enter then simulation will continue.

7.2.2 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 installation directory. The file will contain two columns: one being the Node ID of the device to be failed and other being the failure time (in microseconds).
For example, to fail Node Id 2 from 10th 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: Create DLL and Link the DLL to the NetSim as explained in Section 7.1.

Step 7: Create a scenario in MANET and run the simulation. User can utilize Packet Animation to check the node failure (i.e. no packets are forwarded by failed nodes) after the mentioned time

7.2.3 Transferring file from source to destination in WSN

Objective: Transferring a real file from source node to destination node in WSN

Implementation: The code modifications to transfer file from Sensor to Sink node are described here:

1. Open NetSim.sln in Visual Studio and add the following modifications.
2. The modified files are in Zigbee: Sensor.c and 802_15_4.h
3. In 802_15_4.h add the following line of code

```
#ifndef _NETSIM_802_15_4_H_
#define _NETSIM_802_15_4_H_
#define _FILE_SEND_  //Uncomment to transfer file
```

4. In Sensor.c, the code must be modified at specified places in red color. Add the modified code:

```
#include "main.h"
#include "List.h"
#include "802_15_4.h"
#define MAX_PAYLOAD 70

int fn_NetSim_FindAgentPos(double* dXPos, double* dYPos, int nAgentId,double dSensingTime,AGENT** pstruAgent);
double fn_Sensor_CalculateDistance(POS_2D* pstruPos1, POS_2D* pstruPos2);

#ifdef _FILE_SEND_
typedef struct file_info
	{
		char Packet[100];
		long len;
		int Packet_Id;
		_ele* ele;
	}FILE_INFO,*PFILE_INFO;
#define FILE_INFO_ALLOC() (PFILE_INFO)list_alloc(sizeof(FILE_INFO),offsetof(FILE_INFO,ele))
PFILE_INFO fileinfo=NULL;
static int nPacketId=0;
char file_name[100][50] = {"send.txt"};
char outfile_name[100][50] = {"receive.txt"};

int fnWriteFile(int PacketId)
{
	char *packet;
	static FILE *file_receive=NULL;
	PFILE_INFO file_rec=fileinfo;
	size_t siz;

	if(!file_receive)
		file_receive = fopen(outfile_name[0],"wb");

	while(file_rec)
	{
```

```
                if(file_rec->Packet_Id == PacketId)
                {
                        //fprintf(stderr,"file written. size = %d\n",file_rec->len);
                        packet = file_rec->Packet;
                        siz = file_rec->len;
                        fwrite(packet,sizeof(char),siz,file_receive);
                }
                file_rec=LIST_NEXT(file_rec);
        }
        fflush(file_receive);
        return 0;
}

int fnsendfile(NETSIM_ID nSensorLoop)
{
        NetSim_PACKET *PstruPacket;
        FILE *file_transfer;
        size_t siz;
        long file_size;
        long n;
        PFILE_INFO file;

        if(file_name[nSensorLoop-1] && *file_name[nSensorLoop-1])
                file_transfer = fopen(file_name[nSensorLoop-1],"rb");
        else
                return -1;
        if(!file_transfer)
        {
                perror(file_name[nSensorLoop-1]);
                return -1;
        }
        fseek(file_transfer,0,SEEK_END);
        file_size = ftell(file_transfer);
        rewind(file_transfer);
        n = file_size;
        while(n>0)
        {
                char str[MAX_PAYLOAD+10];
                fprintf(stderr,"Size left = %d\n",n);
                if(n>=MAX_PAYLOAD)
                {
                        siz = fread(str,sizeof(char),MAX_PAYLOAD,file_transfer);

                }
                else
                {
                        siz = fread(str,sizeof(char),n,file_transfer);
                }
                n-=siz;

                //Add application out to tramit the position
                //Generate the packet
                PstruPacket = fn_NetSim_Packet_CreatePacket(5);
                PstruPacket->dEventTime = pstruEventDetails->dEventTime;
                PstruPacket->nDestinationId = nGlobalPANCoordinatorId;
                PstruPacket->nPacketId = ++nPacketId;
                PstruPacket->nPacketStatus = 0;
```

```
		PstruPacket->nPacketType = PacketType_Custom;
		PstruPacket->nPacketPriority = Priority_Low;
		PstruPacket->nQOS =(NETSIM_ID)QOS_BE;
		PstruPacket->nSourceId = (NETSIM_ID)nSensorLoop;
		//Update the Transport layer information
		PstruPacket->pstruTransportData->nSourcePort = SOURCEPORT;
		PstruPacket->pstruTransportData->nDestinationPort = DESTINATIONPORT;
		//Update the Network layer information
		PstruPacket->pstruNetworkData->szSourceIP =
IP_COPY(fn_NetSim_Stack_GetFirstIPAddressAsId((NETSIM_ID)nSensorLoop,0));
		PstruPacket->pstruNetworkData->szDestIP =
IP_COPY(fn_NetSim_Stack_GetFirstIPAddressAsId(PstruPacket->nDestinationId,0));

		PstruPacket->pstruNetworkData->nTTL = MAX_TTL;
		//Update the Application layer information
		//For transferring file from Sensor to sink node
		//70 bytes at a time
		file =FILE_INFO_ALLOC();
		memcpy(file->Packet,str,siz);
		file->Packet_Id = PstruPacket->nPacketId;
		file->len = siz;
		LIST_ADD_LAST((void**)&fileinfo,file);

		PstruPacket->szPayload = NULL;
		PstruPacket->pstruAppData->dPayload = siz;
		PstruPacket->pstruAppData->dOverhead = 0;
		PstruPacket->pstruAppData->dPacketSize  =  PstruPacket->pstruAppData-
>dPayload + PstruPacket->pstruAppData->dOverhead;

		PstruPacket->pstruAppData->dArrivalTime     =     pstruEventDetails-
>dEventTime;
		PstruPacket->pstruAppData->dEndTime = pstruEventDetails->dEventTime;
		PstruPacket->pstruAppData->dStartTime    =        pstruEventDetails-
>dEventTime;

		if(NETWORK->ppstruDeviceList[nSensorLoop-1]->pstruTransportLayer-
>isUDP)
			PstruPacket->pstruTransportData-
>nTransportProtocol=TX_PROTOCOL_UDP;
		else                    if(NETWORK->ppstruDeviceList[nSensorLoop-1]-
>pstruTransportLayer->isTCP)
			PstruPacket->pstruTransportData-
>nTransportProtocol=TX_PROTOCOL_TCP;
		else
			PstruPacket->pstruTransportData->nTransportProtocol=0;
		if(NETWORK->ppstruDeviceList[nSensorLoop-1]->pstruSocketInterface-
>pstruSocketBuffer[0]->pstruPacketlist==NULL)
		{
			fn_NetSim_Packet_AddPacketToList((NETWORK-
>ppstruDeviceList[nSensorLoop-1]->pstruSocketInterface-
>pstruSocketBuffer[0]),PstruPacket,3);
			pstruEventDetails->dPacketSize=PstruPacket->pstruAppData-
>dPacketSize;
			pstruEventDetails->nDeviceType = SENSOR;
			pstruEventDetails->nApplicationId=0;
			pstruEventDetails->nProtocolId=PstruPacket-
>pstruTransportData->nTransportProtocol;
```

```
                        pstruEventDetails->nDeviceId=(NETSIM_ID)nSensorLoop;
                        pstruEventDetails->nInterfaceId=0;
                        pstruEventDetails->nEventType=TRANSPORT_OUT_EVENT;
                        pstruEventDetails->nSubEventType=0;
                        pstruEventDetails->pPacket=NULL;
                        fnpAddEvent(pstruEventDetails);
                }
                else
                {
                        fn_NetSim_Packet_AddPacketToList((NETWORK-
>ppstruDeviceList[nSensorLoop-1]->pstruSocketInterface-
>pstruSocketBuffer[0]),PstruPacket,2);
                }
        }
        fclose(file_transfer);
        return 0;
}
#endif
/** In this function the sensors sense the agent, creates a packet and forwards it
to sink node.*/
int              fn_NetSim_Zigbee_SensorEvent(int              nSensorLoop,NETSIM_ID
nGlobalPANCoordinatorId,AGENT**      pstruAgent,SENSORS*      pstru_Sensor,METRICS**
pstruMetrics,NetSim_EVENTDETAILS* pstruEventDetails)
{
        int nFlag = 0;
        static int nPacketId;
        char str[500];
        int nAgentLoop;
        POS_2D* pstruPos;
        double dDistance;
        POS_2D* pstruTemppos;
        NetSim_PACKET *PstruPacket;

        #ifdef _FILE_SEND_
                fnsendfile((NETSIM_ID)nSensorLoop);
                return 0;
        #endif
        pstruPos = (POS_2D*)fnpAllocateMemory(sizeof(POS_2D),1);
        pstruTemppos = (POS_2D*)fnpAllocateMemory(sizeof(POS_2D),1);
        for(nAgentLoop =0;nAgentLoop<MAXAGENT;nAgentLoop++)
        {
                if(pstruAgent[nAgentLoop] == NULL)
                        continue;
```

In 802_15_4.c , the code must be modified at specified places in red color. Add the modified code:

```
case MAC_IN_EVENT:
{
        .

        .

        .

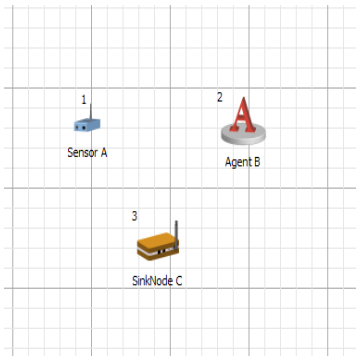
        if(pstruPacket->nControlDataType/100 != MAC_PROTOCOL_IEEE802_15_4)
        {
                //Prepare the Network in event details
```

```
                pstruPacket->pstruMacData->dOverhead -= 5;
                pstruPacket->pstruMacData->dPacketSize  =  pstruPacket->pstruMacData-
>dPayload + pstruPacket->pstruMacData->dOverhead;
                pstruEventDetails->dPacketSize      =      pstruPacket->pstruMacData-
>dPacketSize;
                pstruEventDetails->pPacket = pstruPacket;
                pstruEventDetails->nEventType = NETWORK_IN_EVENT;
                pstruEventDetails->nSubEventType = 0;
                pstruEventDetails->nProtocolId                                          =
fn_NetSim_Stack_GetNWProtocol(pstruEventDetails->nDeviceId);
                //Add Network in event
                fnpAddEvent(pstruEventDetails);
        #ifdef _FILE_SEND_
                if(pstruPacket->nPacketType == PacketType_Custom)
                fnWriteFile(pstruPacket->nPacketId);
        #endif
                        }
                        else if(pstruPacket->nControlDataType == BEACON_FRAME)
                        { ...
```

5. Copy the input file (**file to be transferred**) in NetSim bin folder ("C:\Program Files\NetSim Standard\bin") and rename it as **send.txt**.
6. In Sensor.c, user can optionally edit the name of the input file in file_name[] and output file in outfile_name[] in the code. For example, currently it is **receive.txt** for output file.
7. Build Zigbee (Please refer section 7.1) and link the dll to bin folder of NetSim. Take care to rename the original libZigbee.dll so as to preserve the original binaries of NetSim
8. Next, to run the code, follow these steps:

In this section we create a sample scenario to transfer file from Sensor to Sink Node in WSN:

Step 1: Create a scenario in NetSim as follows. Make sure the sensor is dropped first on the environment



Step 2: Run the simulation. (Make sure the input file to be transferred is present in bin folder of NetSim).

Step 3: Output file should be present in bin folder of NetSim with the name **receive.txt** defined earlier in outfile_name[] in Sensor.c.

Note: Due to retransmissions and errors, sometimes the output file is not reproduced correctly. To get exact file, user has to enable TCP (WSN works on UDP).

7.3 Debugging your code

This section is helpful to debug the code which user has written. To write your own code please refer Section 7.1

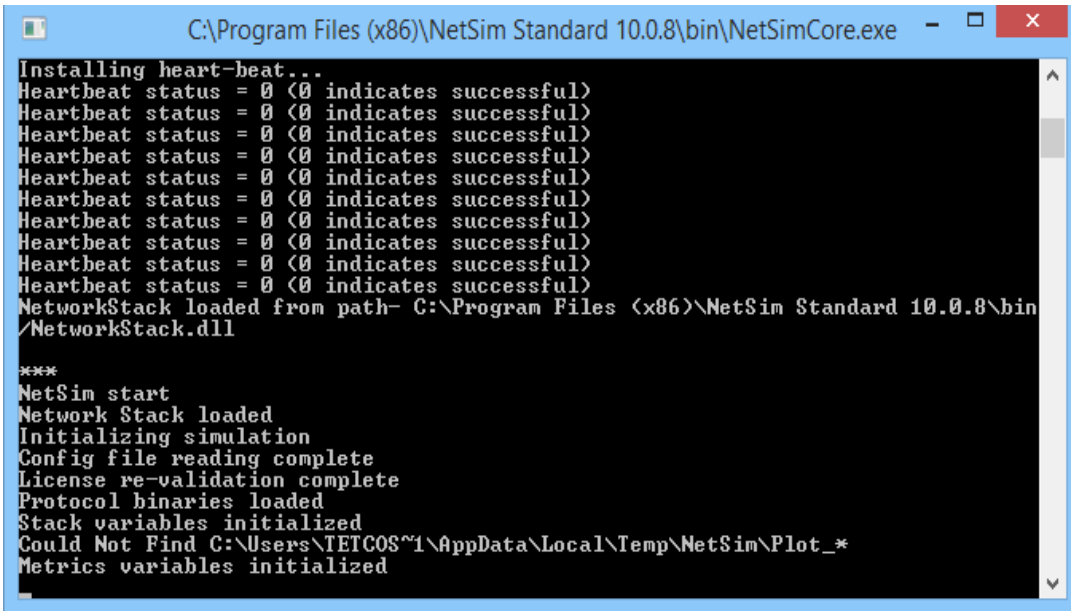
7.3.1 Via GUI

Step 1:- Perform the required modification of the protocol source code. Also add _getch() statement inside init function of the modified protocol.

```
/**
DSR Init function initializes the DSR parameters.
*/
 _declspec(dllexport) int fn_NetSim_DSR_Init(struct stru_NetSim_Network *NETWORK_Formal,
                                             NetSim_EVENTDETAILS *pstruEventDetails_Formal,
                                             char *pszAppPath_Formal,
                                             char *pszWritePath_Formal,
                                             int nVersion_Type,
                                             void **fnPointer)
{
    _getch();
    return fn_NetSim_DSR_Init_F(NETWORK_Formal,pstruEventDetails_Formal,pszAppPath_Formal,pszW
}
/**
```

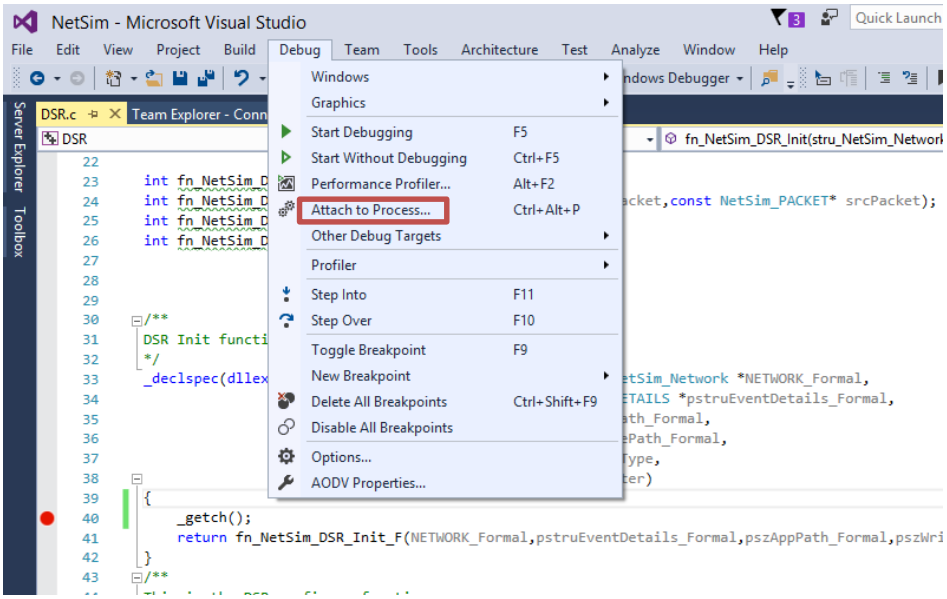
Step 2:- Build the protocol and replace the dll in bin folder in NetSim. Do not close Visual Studio.

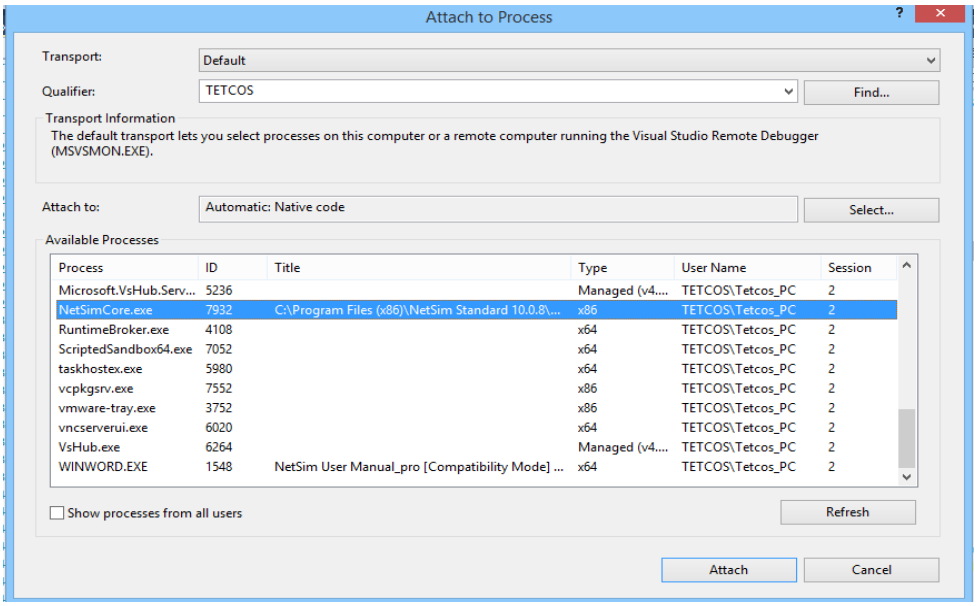
Step 3:- In NetSim, create a network scenario where the protocol is being used. The simulation won't start and will pause (because of _getch())



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.

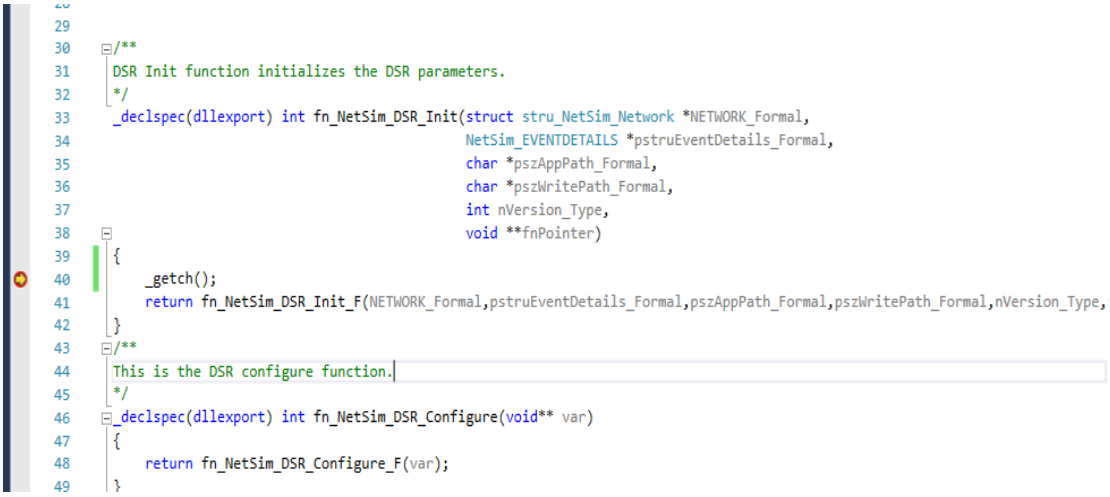




Click on Attach.

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.



```
29
30  /**
31   DSR Init function initializes the DSR parameters.
32   */
33   _declspec(dllexport) int fn_NetSim_DSR_Init(struct stru_NetSim_Network *NETWORK_Formal,
34                                               NetSim_EVENTDETAILS *pstruEventDetails_Formal,
35                                               char *pszAppPath_Formal,
36                                               char *pszWritePath_Formal,
37                                               int nVersion_Type,
38                                               void **fnPointer)
39   {
40       _getch();
41       return fn_NetSim_DSR_Init_F(NETWORK_Formal,pstruEventDetails_Formal,pszAppPath_Formal,pszWritePath_Formal,nVersion_Type,
42   }
43  /**
44   This is the DSR configure function.
45   */
46  _declspec(dllexport) int fn_NetSim_DSR_Configure(void** var)
47   {
48       return fn_NetSim_DSR_Configure_F(var);
49   }
```

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

7.3.2 How to debug and visualize animation simultaneously

- First follow all the 3 steps mentioned in the section 7.3.1 in User Manual but except in “Step 3” check play and record option in Run and Simulate
- After attaching the Netsimecore.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.
- In watch window add {,,Networkstack.dll}pstruEventDetails and drop down to see the dEventTime, dEventType. Note that dEventTime is the simulation time.
- Animation is called only at PHYSICAL_OUT_EVENTs and PHYSICAL_IN_EVENTs and not at any of the other events. Hence users would notice a lag between dEventTime shown when debugging and Simulation Time shown in animation window

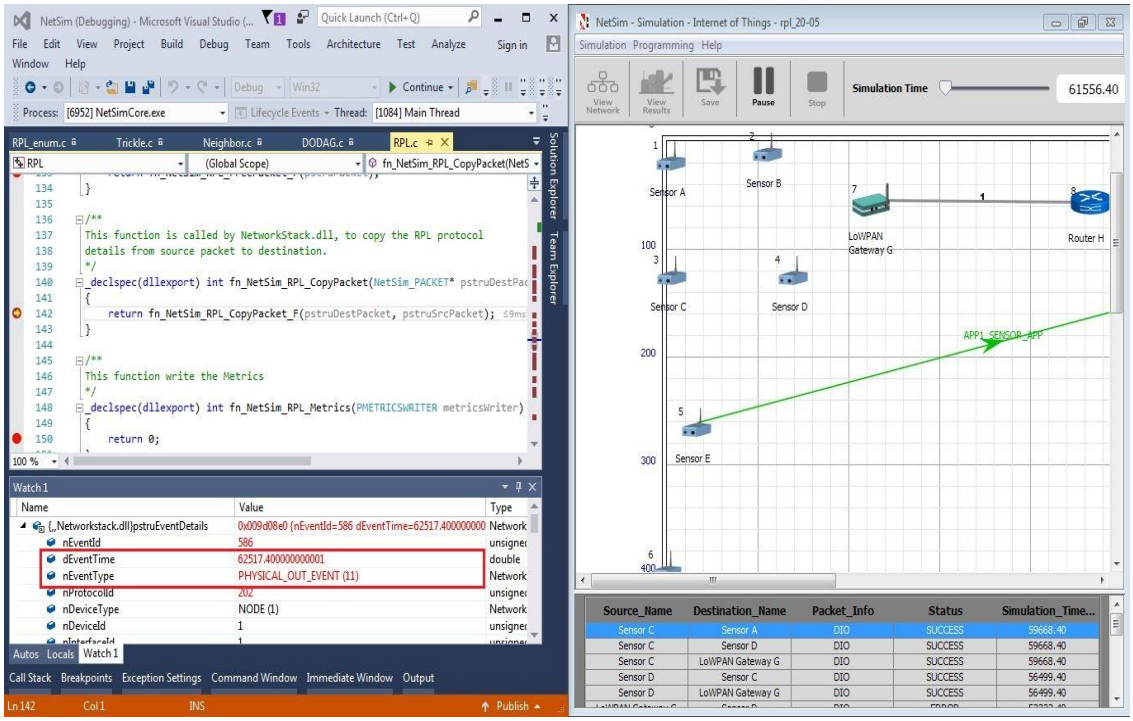


FIGURE: 1

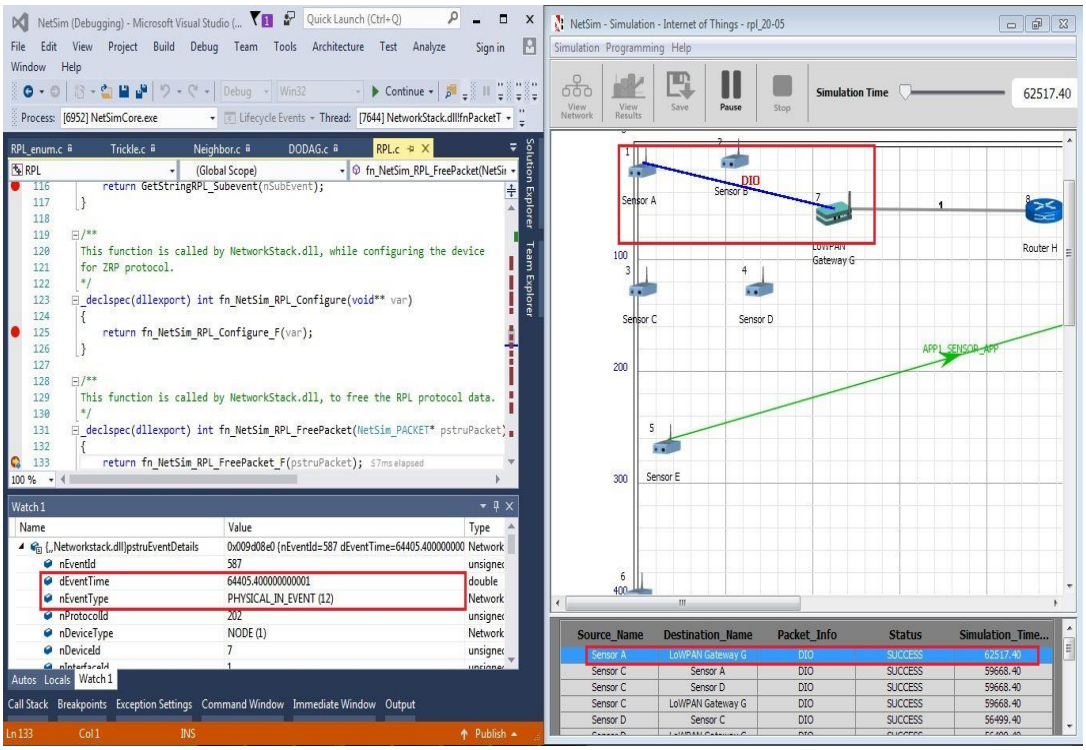


FIGURE: 2

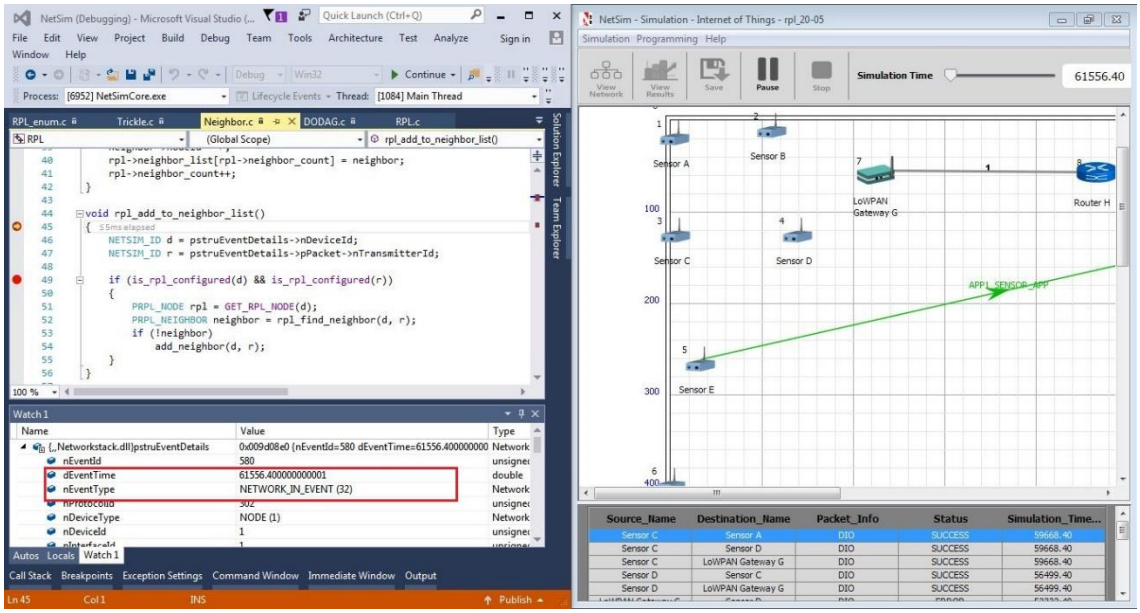


FIGURE: 3

- All debugging options like step over (F10), step into (F11), step out (Shift + F11), continue (F5) are available.
- In the example chosen (IOT simulation with RPL protocol) when you can see on the right window on the image DIO message is copied from Source (Sensor A) to the Destination (LoWPAN Gateway7) and the details of source name, destination, packet info, status and simulation time are seen on the bottom of the NetSim Animation window.
- In **FIG: 1** as you can see on the LEFT WINDOW dEventTime = 62517.4 and dEventType = PHYSICAL_OUT_EVENT
- In **FIG: 2** dEventType = PHYSICAL_IN_EVENT on the left window and the animation is shown for PHYSICAL_OUT of Sensor A and PHYSICAL_IN of LoWPAN Gateway on the right window and the simulation time is 62517.4 which is equal to the dEventTime of PHYSICAL_OUT_EVENT.
- In **FIG: 3** no animation is seen because another event (NETWORK_OUT_EVENT) is being processed
- During debugging every time a new packet is transferred the details of the previous packets will scroll down automatically and the details of the current packet is displayed on the top.
- To stop debugging remove all the breakpoints and press F5 then the control goes over to command window again.

7.3.3 Via CLI and co-relating with event trace

Step 1:- Open the Command prompt.

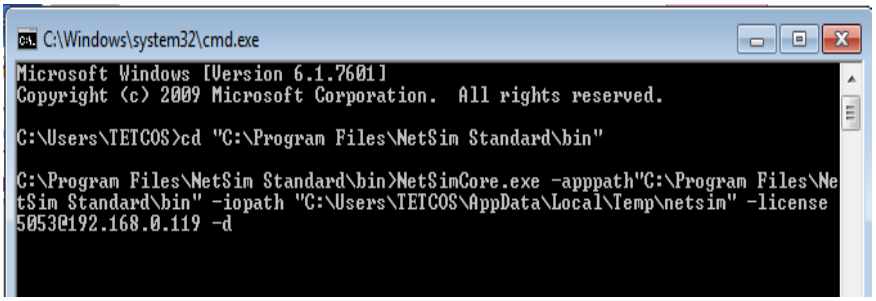
Press “windows+R” and type “cmd”.

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

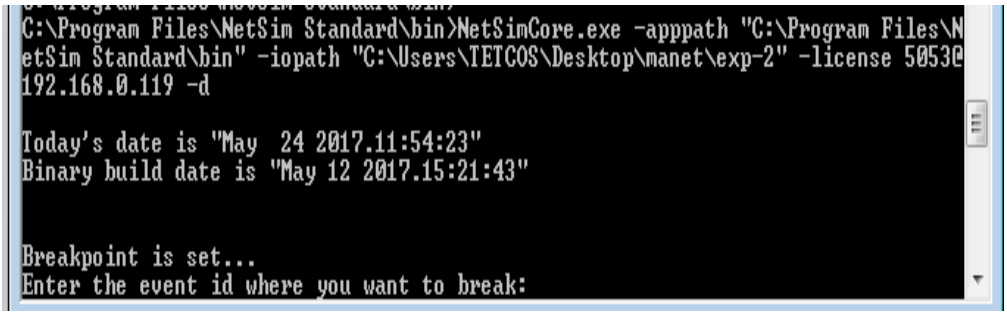
```
>cd <apppath>
```

Step 3:- Type the following command.

```
>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>-license<space>5053@<Server IP Address><space> -d
```

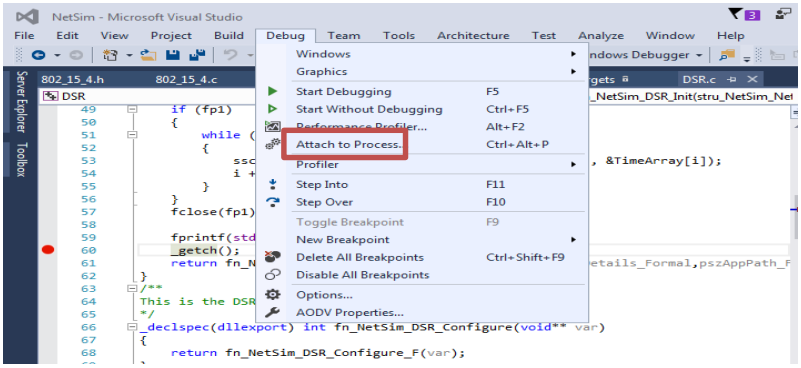


Press enter. Now you can see the following screen.

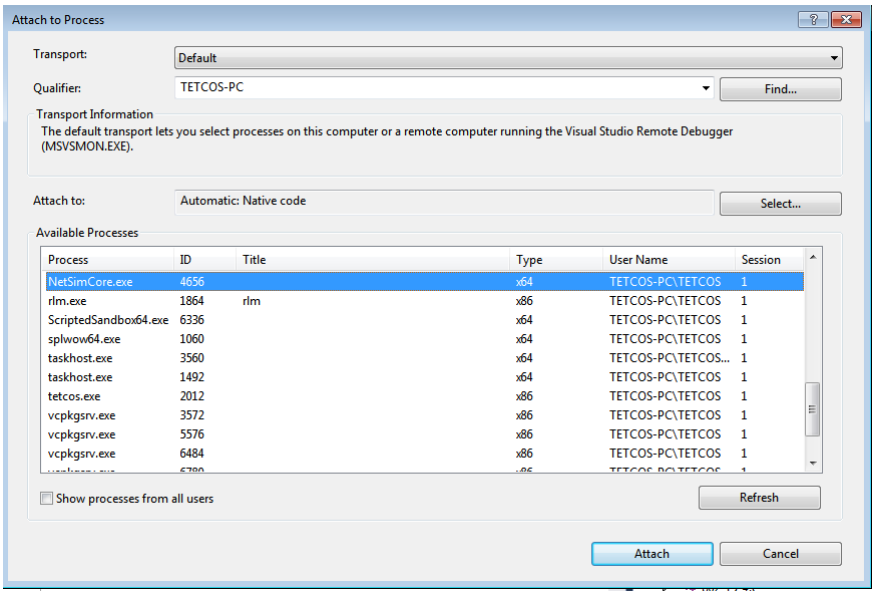


Step 4:- Open/Create the Project in Visual Studio and put break point inside the source code.

Step 5:- Go to “Debug → Attach to Process”



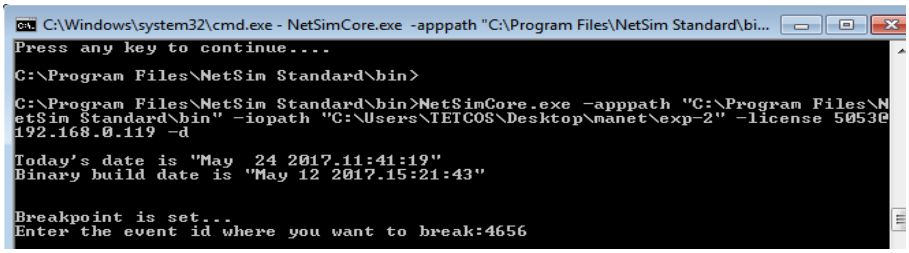
Attach to NetSimCore.exe.



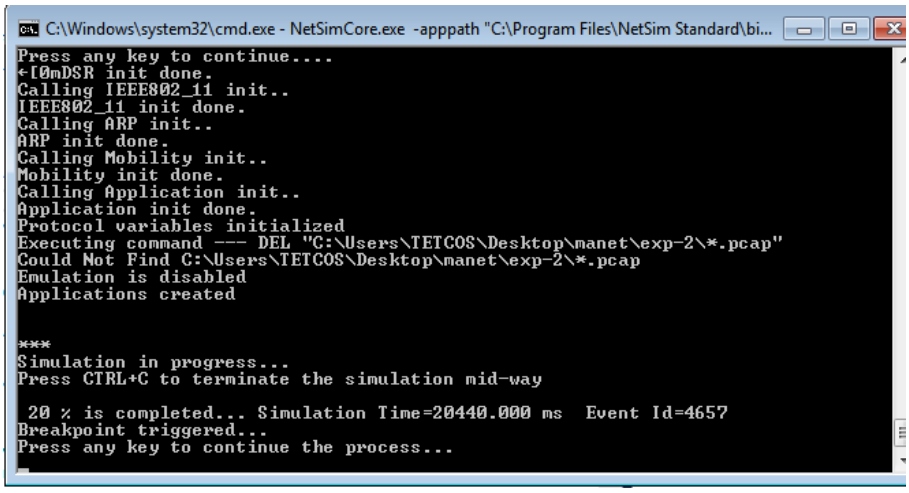
Click on Attach.

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

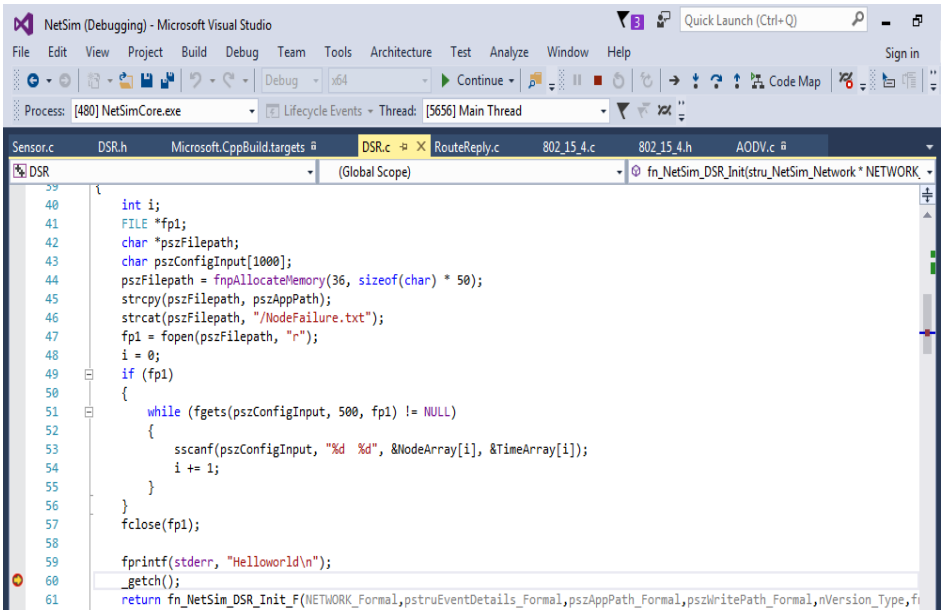
> ***Note:*** *If you don't want to stop at any event you can specify 0 as event id.*



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.

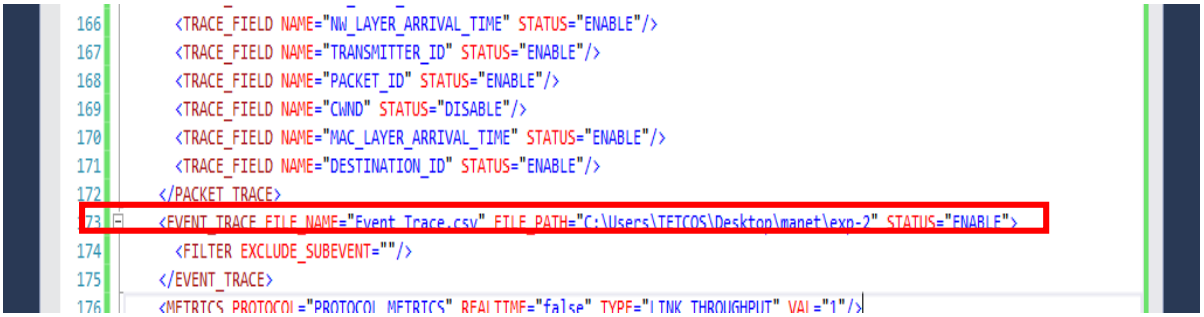
To stop debugging and continue execution, remove all breakpoint and press F5 (key). This then gives the control back to NetSim, for normal execution to continue.

Co-relating with Event Generated

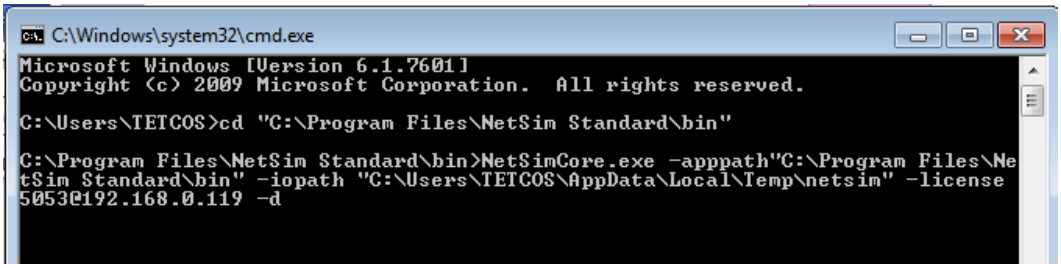
To debug your own (custom) code, it is often helpful to know which section of the code (file name & line number) generated the event under study. There are 2 ways to enable this feature.

Procedure 1

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

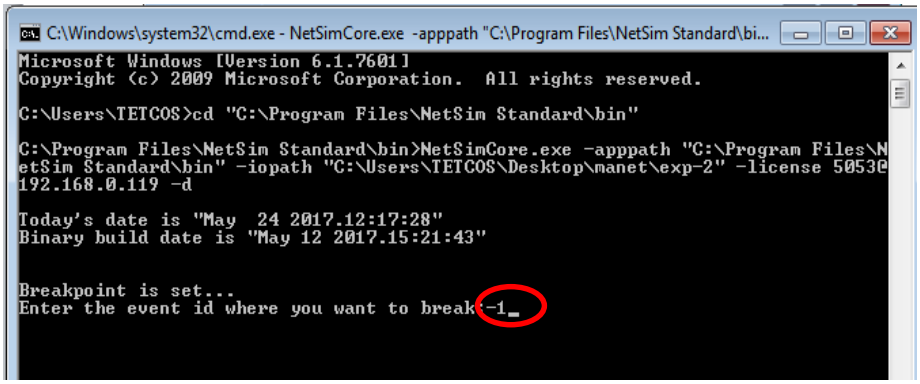


Step 2: Run the NetSim via CLI in debug mode (Refer NetSim Help in chapter 5 → Running Netsim via CLI) with –d as the fourth parameters

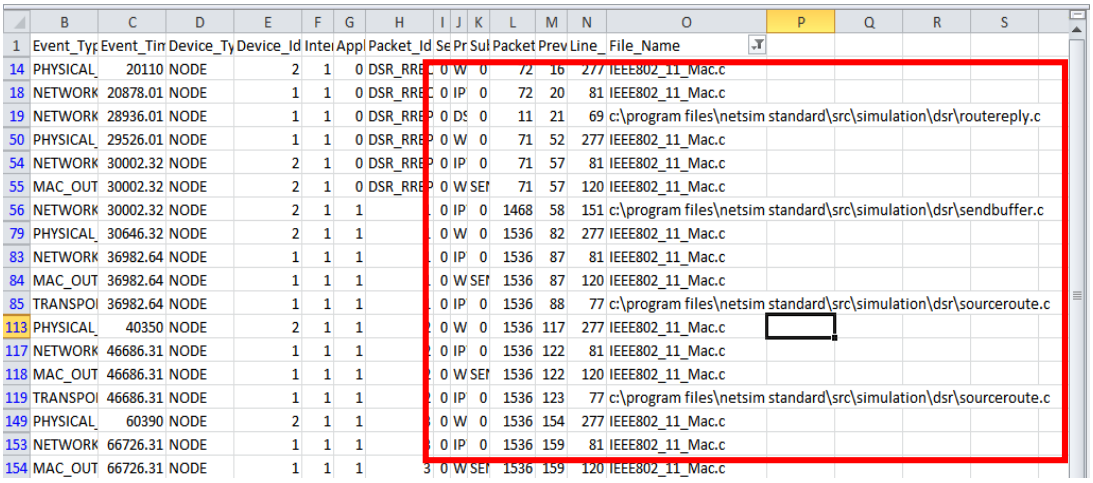


Press enter

Step 3: Enter -1 as the event ID

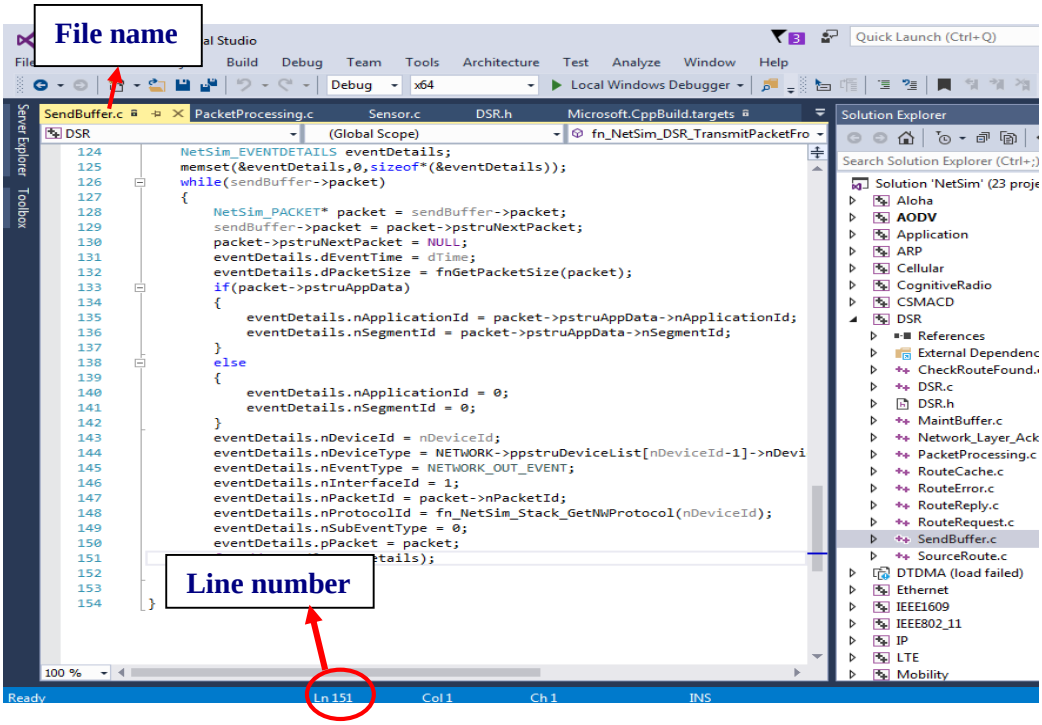


Upon running, NetSim will write the file name and line number of the source code that generated each event.



	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Event_Typ	Event_Tin	Device_Ty	Device_Id	Inter	Appl	Packet_Id	Se	Pr	Sul	Packet	Prev	Line_	File_Name
14	PHYSICAL_	20110	NODE	2	1	0	DSR_RREI	0	W	0	72	16	277	IEEE802_11_Mac.c
18	NETWORK	20878.01	NODE	1	1	0	DSR_RREI	0	IP	0	72	20	81	IEEE802_11_Mac.c
19	NETWORK	28936.01	NODE	1	1	0	DSR_RREI	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_RREI	0	W	0	71	52	277	IEEE802_11_Mac.c
54	NETWORK	30002.32	NODE	2	1	0	DSR_RREI	0	IP	0	71	57	81	IEEE802_11_Mac.c
55	MAC_OUT	30002.32	NODE	2	1	0	DSR_RREI	0	W	SEI	71	57	120	IEEE802_11_Mac.c
56	NETWORK	30002.32	NODE	2	1	1		0	IP	0	1468	58	151	c:\program files\netsim standard\src\simulation\dsr\sendbuffer.c
79	PHYSICAL_	30646.32	NODE	2	1	1		0	W	0	1536	82	277	IEEE802_11_Mac.c
83	NETWORK	36982.64	NODE	1	1	1		0	IP	0	1536	87	81	IEEE802_11_Mac.c
84	MAC_OUT	36982.64	NODE	1	1	1		0	W	SEI	1536	87	120	IEEE802_11_Mac.c
85	TRANSPOI	36982.64	NODE	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		0	W	0	1536	117	277	IEEE802_11_Mac.c
117	NETWORK	46686.31	NODE	1	1	1		0	IP	0	1536	122	81	IEEE802_11_Mac.c
118	MAC_OUT	46686.31	NODE	1	1	1		0	W	SEI	1536	122	120	IEEE802_11_Mac.c
119	TRANSPOI	46686.31	NODE	1	1	1		0	IP	0	1536	123	77	c:\program files\netsim standard\src\simulation\dsr\sourceroute.c
149	PHYSICAL_	60390	NODE	2	1	1		0	W	0	1536	154	277	IEEE802_11_Mac.c
153	NETWORK	66726.31	NODE	1	1	1		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

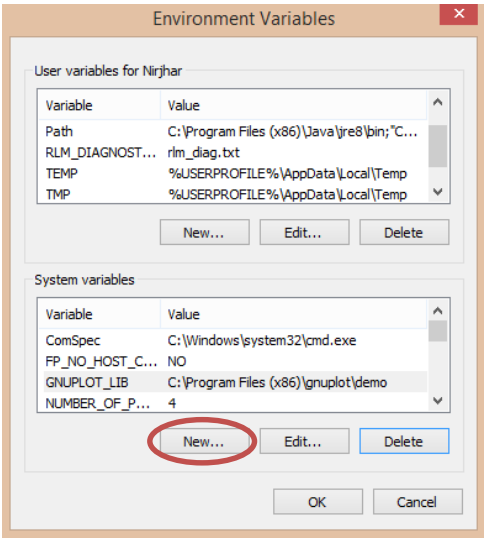
Note: In the above trace file Event Id 56 is triggered inside the sendbuffer.c file which is present in DSR.c. Since all the lib files are opaque to the end user, you cannot see the source code of the lib file. However, Event Id 9 is triggered at line number 69 of routereply.c file and you can find the location of the event by opening the routereply.c file as shown below.



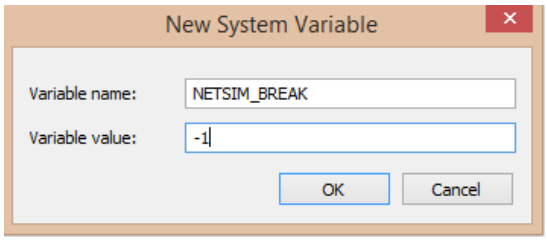
Procedure 2:

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.



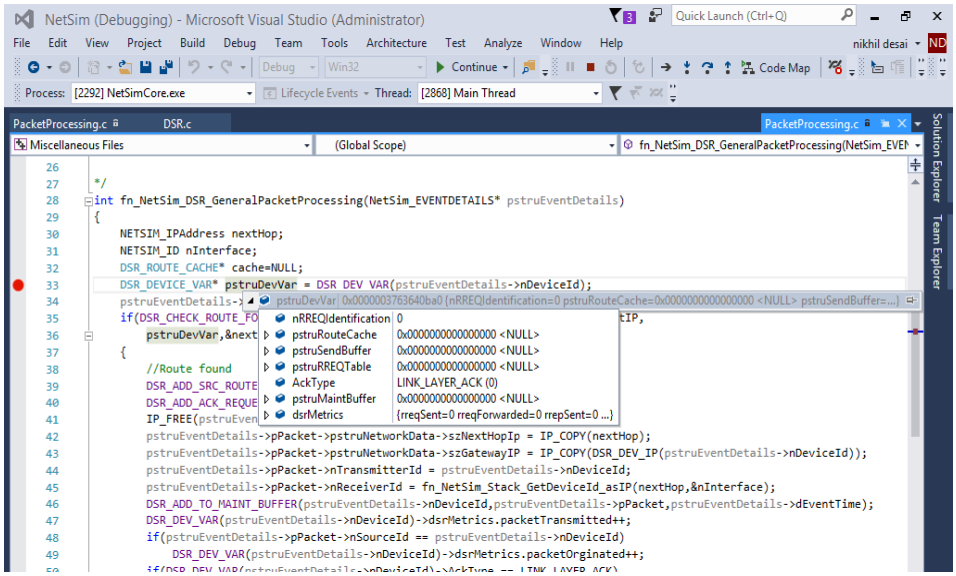
Step 4: Now perform simulation in NetSim (Enable event trace in GUI) . Upon running, NetSim will write the file name and line number of the source code that generated each event.

	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	App	Packet_Id	Se	Pr	Sul	Packet	Prev	Line_	File_Name				
14	PHYSICAL_	20110	NODE	2	1	0	DSR_RRE	0	W	0	72	16	277	IEEE802_11_Mac.c				
18	NETWORK	20878.01	NODE	1	1	0	DSR_RRE	0	IP	0	72	20	81	IEEE802_11_Mac.c				
19	NETWORK	28936.01	NODE	1	1	0	DSR_RRE	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_RRE	0	W	0	71	52	277	IEEE802_11_Mac.c				
54	NETWORK	30002.32	NODE	2	1	0	DSR_RRE	0	IP	0	71	57	81	IEEE802_11_Mac.c				
55	MAC_OUT	30002.32	NODE	2	1	0	DSR_RRE	0	W	SEI	71	57	120	IEEE802_11_Mac.c				
56	NETWORK	30002.32	NODE	2	1	1		0	IP	0	1468	58	151	c:\program files\netsim standard\src\simulation\dsr\sendbuffer.c				
79	PHYSICAL_	30646.32	NODE	2	1	1		0	W	0	1536	82	277	IEEE802_11_Mac.c				
83	NETWORK	36982.64	NODE	1	1	1		0	IP	0	1536	87	81	IEEE802_11_Mac.c				
84	MAC_OUT	36982.64	NODE	1	1	1		0	W	SEI	1536	87	120	IEEE802_11_Mac.c				
85	TRANSPOI	36982.64	NODE	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		0	W	0	1536	117	277	IEEE802_11_Mac.c				
117	NETWORK	46686.31	NODE	1	1	1		0	IP	0	1536	122	81	IEEE802_11_Mac.c				
118	MAC_OUT	46686.31	NODE	1	1	1		0	W	SEI	1536	122	120	IEEE802_11_Mac.c				
119	TRANSPOI	46686.31	NODE	1	1	1		0	IP	0	1536	123	77	c:\program files\netsim standard\src\simulation\dsr\sourceroute.c				
149	PHYSICAL_	60390	NODE	2	1	1		0	W	0	1536	154	277	IEEE802_11_Mac.c				
153	NETWORK	66726.31	NODE	1	1	1		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				

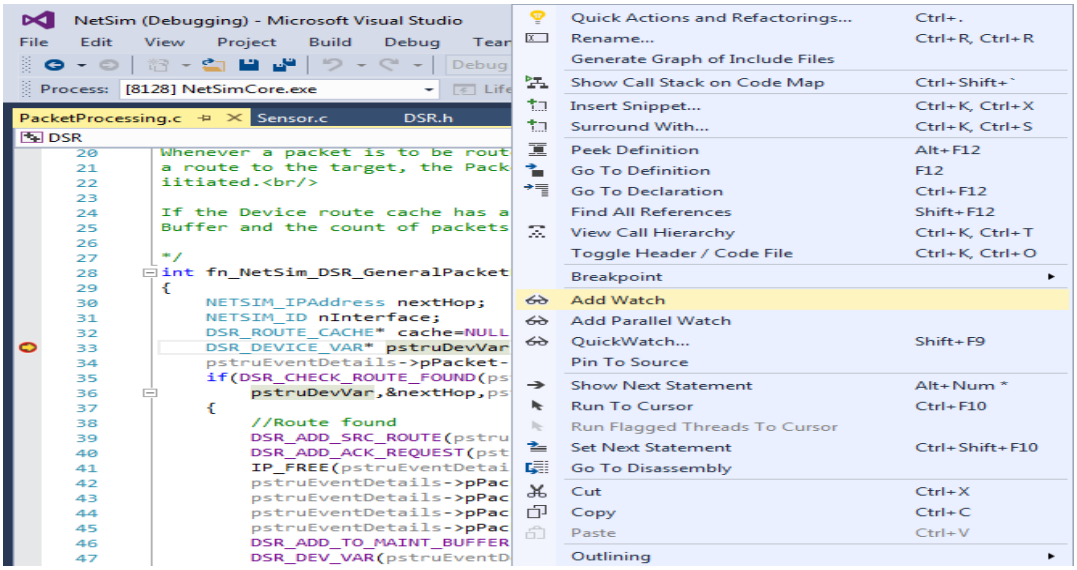
7.3.4 Viewing & Accessing variables

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 –

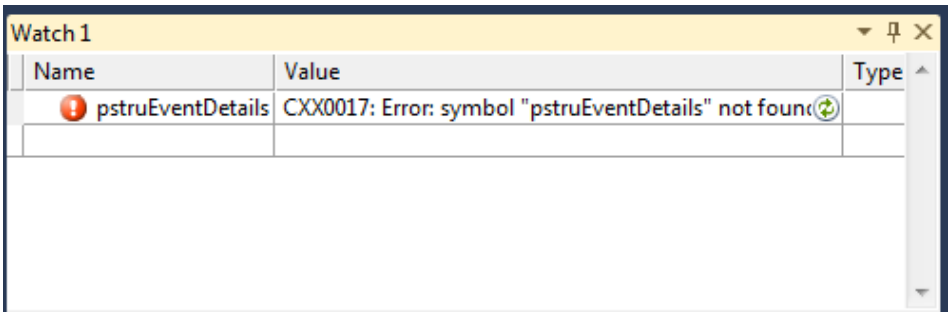


Watch the change in the variable as the code progress by right clicking on the variable & clicking on "add watch" tab. This is useful if to continuously monitor the change in the variable as the code progresses.

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.

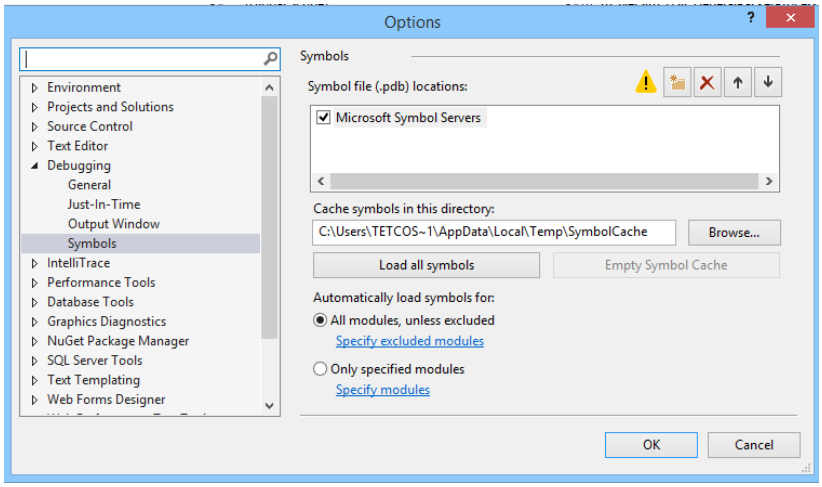


In the call stack window one can find the file in which that variable is situated. Right click on the dll file name in the call stack window, in this case NetworkStack.dll. Then in the pull down menu which appears, select "load symbols from" and give the path of the pdb(program database) file.

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 executablesof 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.

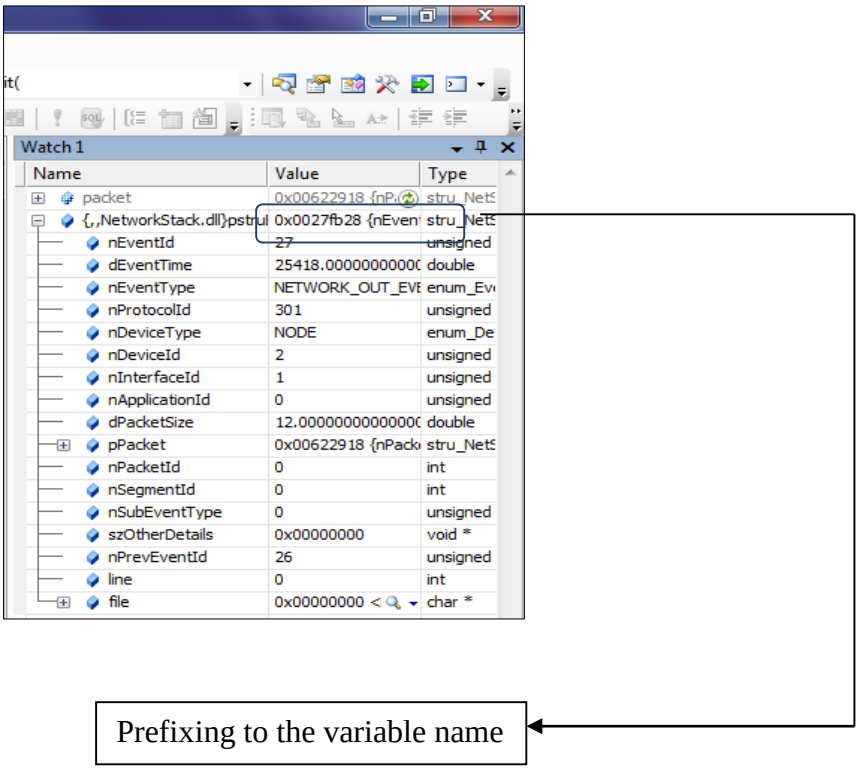
When a user debugs a project in the Visual Studio IDE, the debugger knows exactly where to find the .pdb and source files for the code. If the user wants to debug code outside their project source code, such as the Windows or third-party code the project calls, the user has to specify the location of 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 –

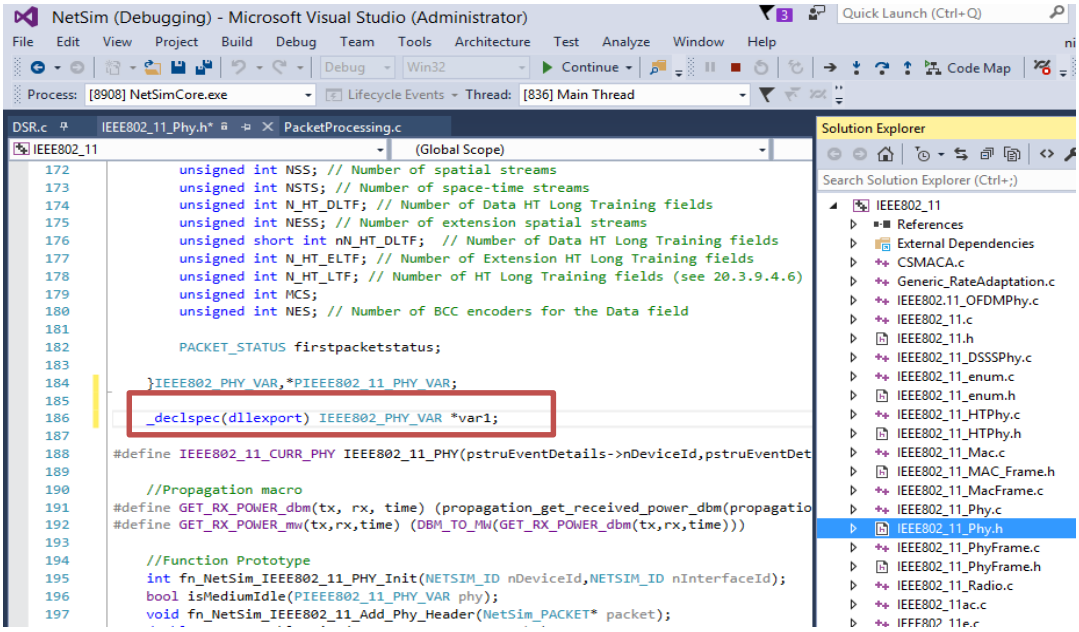
In NetSim, while a scenario is simulated, it is possible to access variables which are defined in one .dll file from another .dll file. An example is given below showing how a IEEE802.11 file variable dReceivedPower_mw can be accessed in the DSR file.

The user must modify the files present in <Installed Directory>\src\Simulation as specified below.

The variable dReceivedPower_mw is defined in a structure **stru_802_11_Phy_Var.** 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 –

_declspec(dllexport) IEEE802_PHY_VAR *var1;

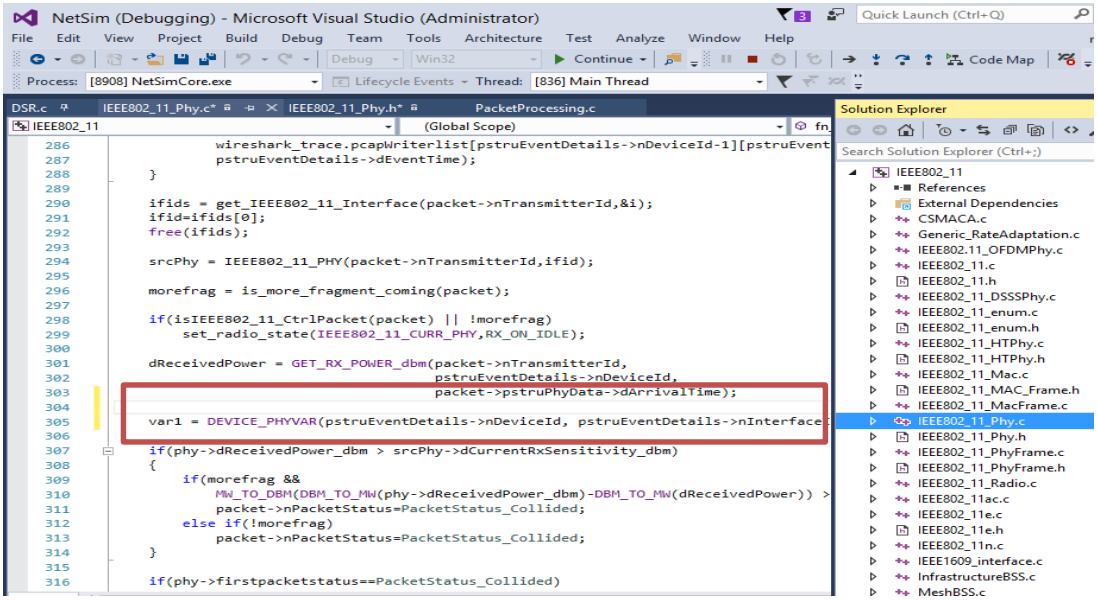
In the example, the code line must be written in IEEE802_11_Phy.h file present inside IEEE802_11 folder.



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.

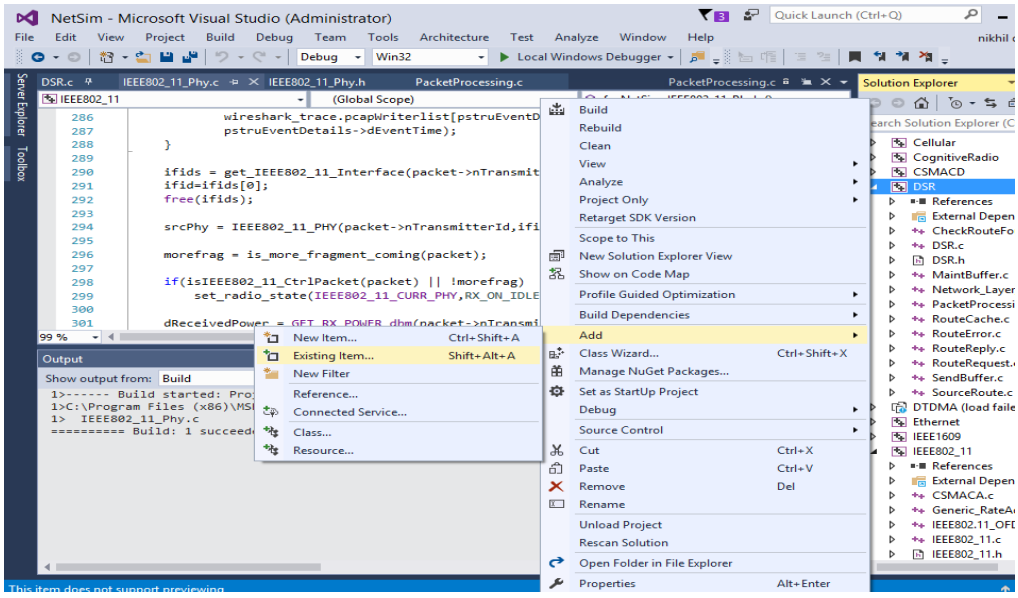
var1 = DEVICE_PHYVAR(pstruEventDetails->nDeviceId,pstruEventDetails->nInterfaceId);

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.

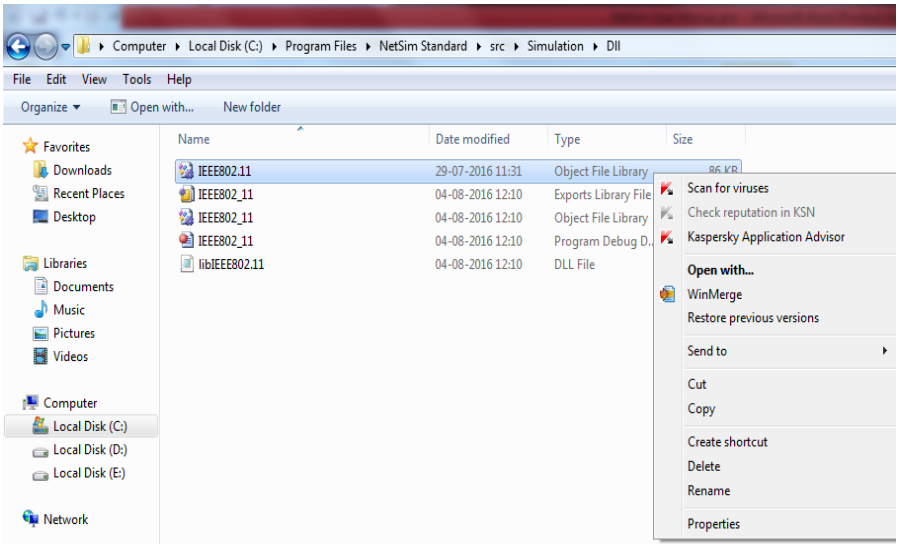


The solution must be built and the resulting <Installed Directory>\src\Simulation \Dll\libIEEE802.11.dll file which gets created must be copied in to the NetSim standard bin folder.

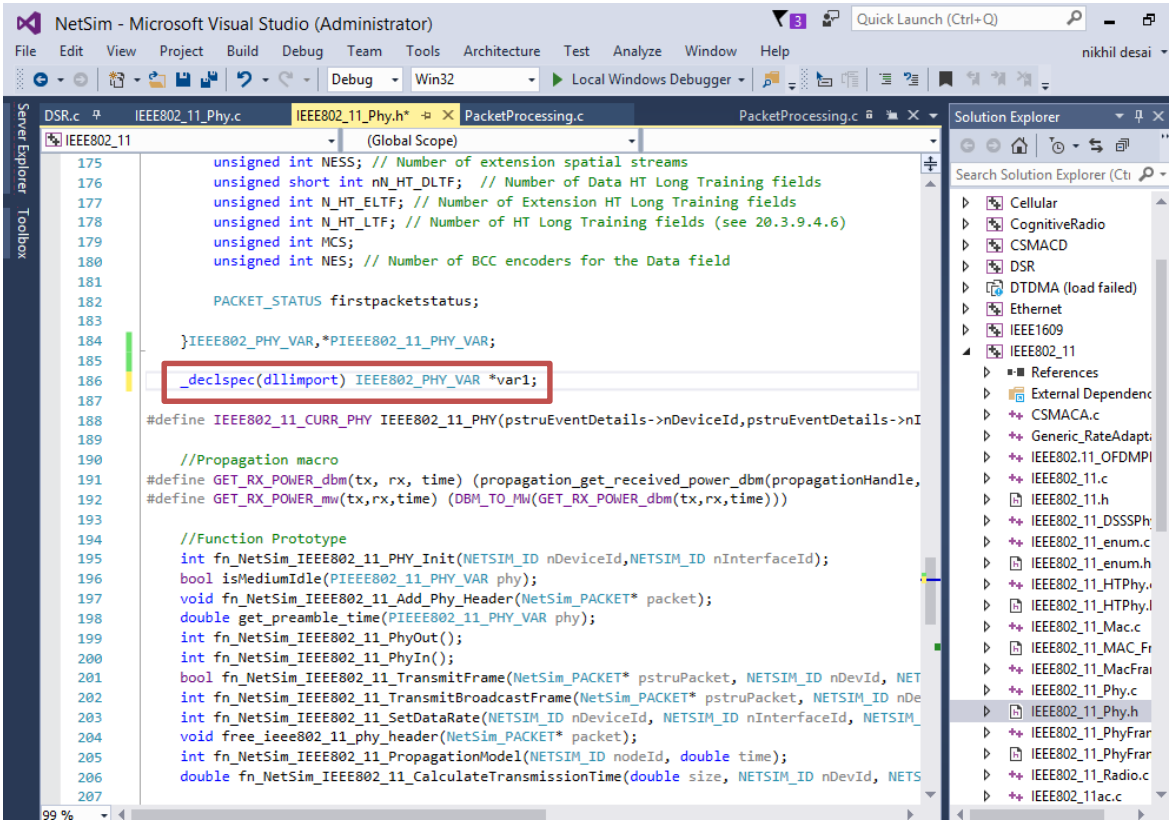
The modified IEEE802_11.h file along with other header files on which it depends (IEEE802_11.h, IEEE802_11_enum.h, IEEE802_11_HTPhy.h, IEEE802_11_Phy.h, IEEE802_11_PhyFrame.h,and IEEE802_11e.h), present inside <Installed Directory> \src \Simulation \IEEE802_11\ folder must be copied and pasted in the DSR solution folder <Installed Directory>\src\Simulation\DSR and must be included in the DSR solution in visual studio.



The Object file library <Installed Directory>\src\Simulation \Dll\IEEE802_11.lib file which got created must be copied and pasted in the lib folder located at <Installed Directory>\src\Simulation.



The IEEE802_11.h header file which was included in DSR solution must be edited and the line where _declspec(dllexport) IEEE802_11_PHY_VAR *var1; was written must be replaced with _declspec(dllimport) IEEE802_11_PHY_VAR *var1;

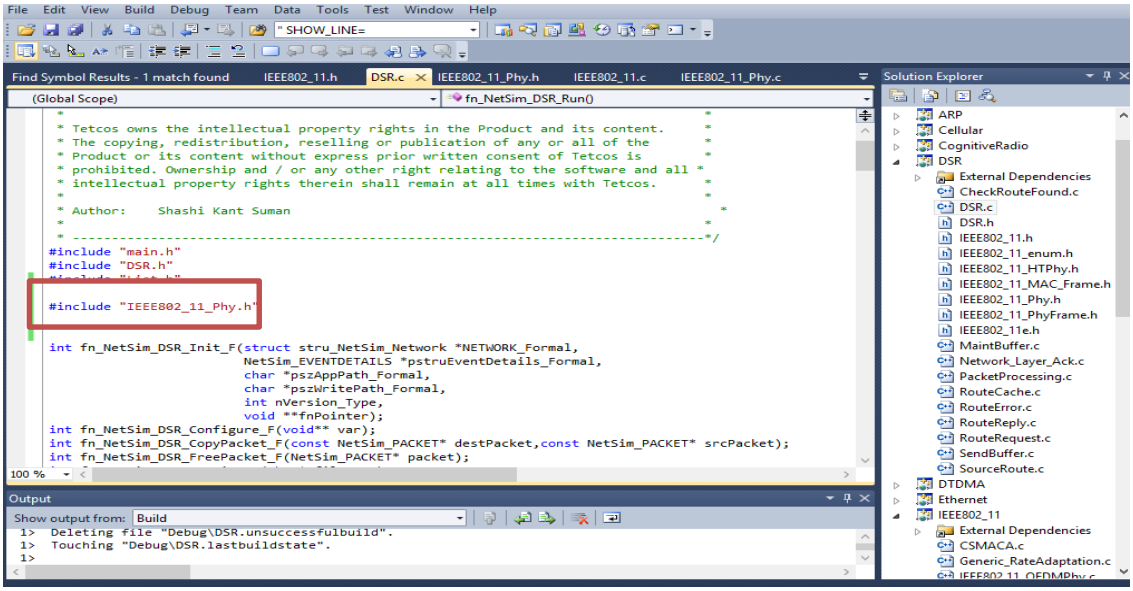


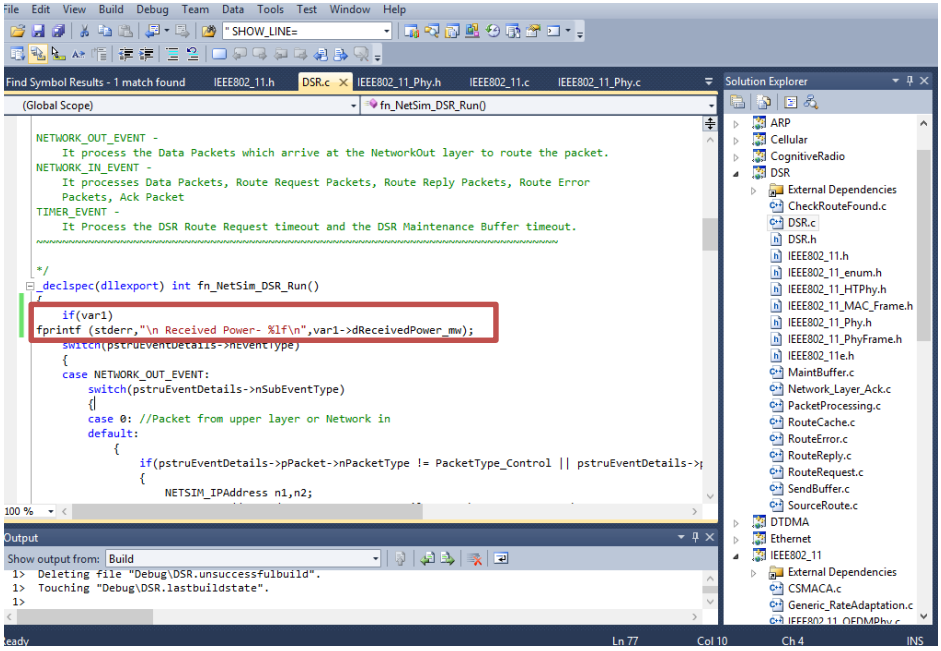
For viewing the variable value in the command prompt, the following lines must be added in DSR.c.

#include “IEEE802_11_Phy.h”

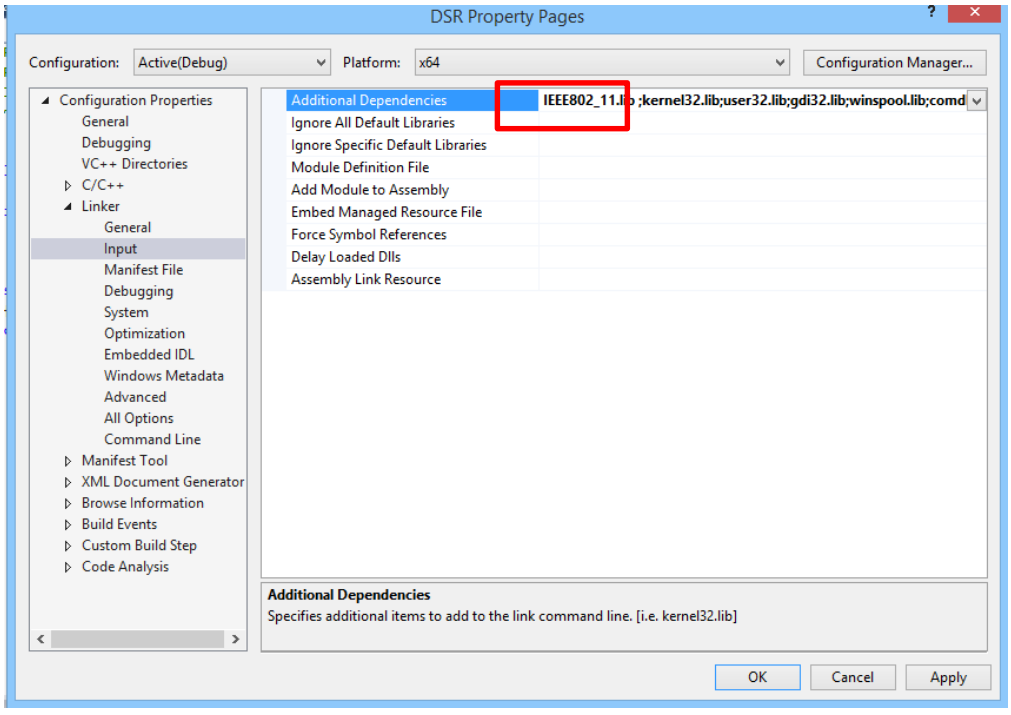
if(var1)

fprintf (stderr,"\n Received Power- %lf\n",var1->dReceivedPower_mw);





Now Right click on DSR project → Properties → Linker →Input → add **IEEE802_11.lib** to the Additional Dependencies.



The solution must be built and the resulting dll file <Installed Directory>\src\Simulation \Dll\libDSR.dll must be copied and replaced in the NetSim Standard bin path. When a scenario is run, the Received Power can be seen in the command prompt.

7.3.5 Print to console window in NetSim

Users can try printing the Device ID, Application ID, Duplicate Ack Count, Congestion Window Size and the RTT time from the functions

fn_NetSim_TCP_Window_Expansion(TCB *pstruTCP_Connection_Var)

(or)

fn_NetSim_TCP_Window_Shrinkage(TCB *pstruTCP_Connection_Var)

which are part of the **Window_Expansion.c** and **Window_Shrinkage.c** files.

Users can use the TCB connection variable **pstruTCP_Connection_Var** to print the required parameters.

The lines of code can be something like:

To print to a file:

```
fprintf (fp,"Device ID\tAppln ID\tDuplicate ACKs\tCongestion Window Size\tRTT\n");
fprintf(fp,"%d\t%d\t%d\t%f\n", pstruTCP_Connection_Var->nDevice_ID, pstruTCP_Connection_Var->nAppId, pstruTCP_Connection_Var->nDup_ACK_Count, pstruTCP_Connection_Var->un_cwnd, pstruTCP_Connection_Var->dRTT_TIME);
```

Where **fp** can be a file pointer to your log file.

To print to console:

```
fprintf (stderr,"Device ID\tAppln ID\tDuplicate ACKs\tCongestion Window Size\tRTT\n");
fprintf(stderr,"%d\t%d\t%d\t%f\n", pstruTCP_Connection_Var->nDevice_ID, pstruTCP_Connection_Var->nAppId, pstruTCP_Connection_Var->nDup_ACK_Count, pstruTCP_Connection_Var->un_cwnd, pstruTCP_Connection_Var->dRTT_TIME);
```

This will give you the values of these parameters at the current instant of "simulation time", whereas printing via custom the metrics will give you the values of these parameters at the last instant of the simulation.

Note:

Whenever Device Id is printed to console or to a file it has to be invoked from the **pstruEventDetails->deviceid** variable. This is because, it keeps changing every time a new event is called.

7.4 NetSim API's

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

1. packet.h – Packet related APIs (Eg: Create_Packet ( ), Copy_Packet ( ), Free_Packet ( ) etc.)
2. stack.h – Network / device / link and event related APIs (Eg: Get_Device_IP ( ), Get_Connected_Link( ), Add_Event ( ) etc.)
3. list.h, -- Optimized list operation calls since NetSim uses lists extensively (Eg: Add_to_list ( ), Sort_list ( ) etc.)
4. NetSim_Graph.h – This is used for plotting graphs using GNU plot
5. IP_Addressing.h – For setting & getting IP address per the appropriate format (Eg: Convert_IP_to_string ( ), Compare_IP( ) etc.)

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.

8 Advanced Features

8.1 Random number Generator and Seed Values

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

a. 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.
b. Node back-off's after collisions are random to resolve contention issues
c. 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.

8.2 Static Routing

Static routing is a form of routing that occurs when a router uses a manually-configured routing entry, rather than information from a dynamic routing traffic. In many cases, static routes are manually configured by a network administrator by adding in entries into a routing table, though this may not always be the case. Unlike dynamic routing, static routes are fixed and do not change if the network is changed or reconfigured.

RIP

RIP (Routing Information Protocol) is a well-known and commonly used distance-vector routing protocol which employs the hop count as a routing metric. Although originally developed for LANs, RIP can also be used in WANs. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from source to destination. The maximum number of hops allowed for RIP is 15, which limits the size of networks that RIP can support. A hop count of 16 is considered an infinite distance and the route is considered unreachable. RIP generally uses the User Datagram Protocol (UDP) as its transport protocol.

OSPF

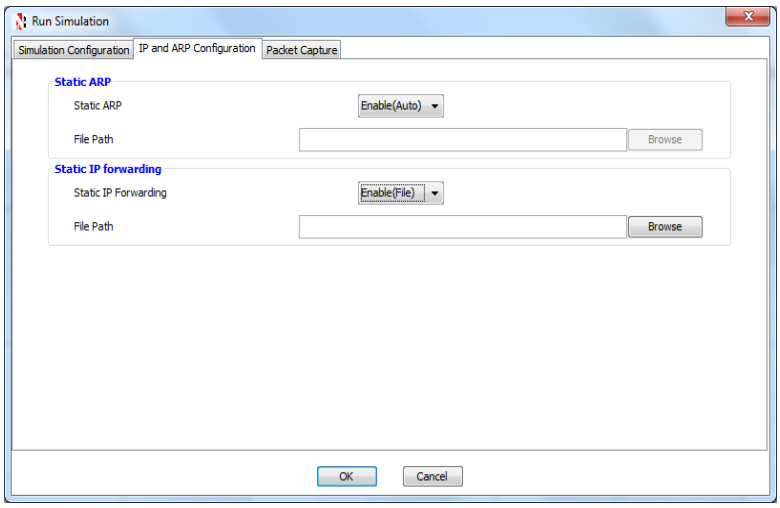
Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing algorithm and falls into the group of interior routing protocols, operating within a single autonomous system. OSPF routes are typically more reliable than RIP routes. RIP only takes the number of hops into account when computing a route's cost, but OSPF also considers the relative cost of each link used in the route. OSPF usually computes the cost of a link relative to that link's inverse bandwidth. In addition, RIP networks are limited to 15 hops whereas OSPF allows networks to expand beyond this limit.

How to Setup Static Routes in RIP and OSPF

Using GUI

Step1: Create a scenario in internetworks and set properties of the devices. Configure the applications.

Step 2: While running the simulation, go to IP and ARP configuration tab and enable Static IP forwarding and browse the path where it has to be saved.



Using CLI

Step1: In internetworks, Static Routes can be set for any scenario having a minimum of 3 routers, 2 switches and 3 wired nodes. The easiest way to do this is to first run the scenario with routing protocol set as RIP/OSPF. For that, users can go to Router properties → Application Layer properties → set RIP/OSPF protocols and save the Configuration file.

Step 2: Open the Configuration file with Visual studio. Expand the Protocol configuration and set the Static Routing information in Application Layer property of the device, by enabling the Static routing status. Then set the appropriate Static routing file name and file path.

By default:

```
<PROTOCOL_CONFIGURATION>
  <PROTOCOL NAME="IPV4">
    <STATIC_IP_FORWARDING_TABLE FILE="" STATUS="DISABLE"/>
  </PROTOCOL>
  <PROTOCOL NAME="ARP">
    <STATIC_ARP FILE="" STATUS="ENABLE"/>
  </PROTOCOL>
 </PROTOCOL_CONFIGURATION>
```

Change to:

```
<PROTOCOL_CONFIGURATION>
  <PROTOCOL NAME="IPV4">
    <STATIC_IP_FORWARDING_TABLE FILE="C:\Program Files (x86)\NetSim
Pro\Docs\Sample_Configuration\Internetworks\Static Routing Scenario\StaticRouting_TCP.txt"
STATUS="ENABLE"/>
  </PROTOCOL>
  <PROTOCOL NAME="ARP">
    <STATIC_ARP FILE="" STATUS="ENABLE"/>
  </PROTOCOL>
 </PROTOCOL_CONFIGURATION>
```

Note:

1. Update this information in any one of the router
2. A sample StaticRouting.txt file will be available inside "C:\Program Files\NetSim standard\Docs\Sample_Configuration\Internetworks". Appropriately modify it for the scenario.

The StaticRouting.txt file contains

1. Device Id
2. List of entries to add in the routing table

DEVICE_ID:1

ROUTE ADD 192.168.0.0 MASK 255.255.255.0 192.168.0.1 METRICS 1 IF 1

ROUTE ADD 192.168.1.0 MASK 255.255.255.0 192.168.1.1 METRICS 1 IF 2

DEVICE_ID:2

ROUTE ADD 192.168.2.0 MASK 255.255.255.0 192.168.2.1 METRICS 1 IF 1

ROUTE ADD 192.168.3.0 MASK 255.255.255.0 192.168.3.1 METRICS 1 IF 2

ROUTE ADD dest_ip MASK subnet_mask gateway_ip METRICS metric_value IF Interface_Id

1. The ROUTE ADD command to add the static route
2. The dest_ip is the Network address for the destination network
3. The MASK is the Subnet mask for the destination network
4. The gateway_ip is the IP address of the next-hop router
5. The METRICS is the value used to choose between two routes
6. The IF is the Interface to which the gateway_ip is connected. The default is 1.

Step 3: After this, run this configuration file through CLI and static routes will be used for routing.

8.3 Mobility Models in NetSim

Mobility models represent the movement of mobile user, and how their location, velocity and acceleration change over time. Such models are frequently used for simulation purposes when new communication or navigation techniques are investigated, or to evaluate the performance of mobile wireless systems and the algorithms and protocols at the basis of them. Typical mobility models provided in NetSim are as follows:

8.3.1 Random Walk mobility model

It is a simple mobility model based on random directions and speeds. In this mobility model, a mobile node moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and direction are both chosen from pre-defined ranges. Each movement in the Random Walk Mobility Model occurs in either a constant time interval or a constant distance traveled, at the end of which a new direction and speed are calculated.

8.3.2 Random Waypoint Mobility Model

It includes pause time between changes in direction and/or speed. A mobile node begins by staying in one location for a certain period of time (i.e., a pause time). Once this time expires, the mobile node chooses a random destination in the simulation area and a speed that is uniformly distributed between [minspeed, maxspeed]. The mobile node then travels toward the newly chosen destination at the selected speed. Upon arrival, the mobile node pauses for a specified time period before starting the process again.

8.3.3 Group mobility

It is a model which describes the behavior of mobile nodes as they move together. i.e. the sensors having common group id will move together.

8.3.4 File Based Mobility

In File Based Mobility, users can write their own custom mobility models and define the movement of the mobile users. The name of the trace file generated should be kept as mobility.txt and it should be in the **NetSim Mobility File format**.

The user can also generate the mobility files using external tools like SUMO (Simulation of Urban MObility), Vanet MobiSim etc.

The NetSim Mobility File format is as follows

Step 1: Create a text file inside <NetSim Installation Directory>\bin and rename it as "mobility.txt"

Step 2: Open the text file and write the code in format shown below

To write comments, use # tag

User needs to specify the total number of Nodes and Environment size as shown below
#nodes: <No. of Nodes> max x = <X_Environment_Size>, max y: <Y_Environment_Size>

#First specify the location of all the devices as per their X, Y and Z Axis
$node_(<Node _ID - 1>) set X_ <Initial X_Coordinate>
$node_(<Node _ ID - 1>) set Y_ <Initial Y_ Coordinate >
$node_(<Node _ ID - 1>) set Z_ <Initial Z_ Coordinate >

#Specify the new location of the specific device at the specific time
$time <Time_in_Secs> "$node_(<Node_ ID - 1>) <X_Coordinate> <Y_ Coordinate > <Z_ Coordinate >"

Step 3: In NetSim, go to MANET and create a Network scenario. Click on **Node properties** and in **Global properties**, set the Mobility Model as "File Based Mobility" and simulate.

A sample file based mobility experiment is present at <NetSim Installed Directory> \Docs\ Sample_Configuration\ MANET.

A sample mobility.txt file for a MANET network containing 2 nodes is shown below
#
#nodes: 2 max x = 500.0, max y: 500.0
#
$node_(0) set X_ 70.0
$node_(0) set Y_ 70.0

```
$node_(0) set Z_ 0.0
$node_(1) set X_ 150.0
$node_(1) set Y_ 150.0
$node_(1) set Z_ 0.0
$time 0.0 "$node_(0) 70.00 70.00 0.00"
$time 0.0 "$node_(1) 150.0 150.0 0.0"
$time 5.0 "$node_(0) 100.00 70.00 0.00"
$time 5.0 "$node_(1) 150.0 160.0 0.0"
$time 10.0 "$node_(0) 130.00 70.00 0.00"
$time 10.0 "$node_(1) 150.0 170.0 0.0"
$time 15.0 "$node_(0) 160.00 70.00 0.00"
$time 15.0 "$node_(1) 150.0 180.0 0.0"
$time 20.0 "$node_(0) 190.00 70.00 0.00"
$time 20.0 "$node_(1) 150.0 190.0 0.0"
$time 25.0 "$node_(0) 220.00 70.00 0.00"
$time 25.0 "$node_(1) 150.0 200.0 0.0"
$time 30.0 "$node_(0) 250.00 70.00 0.00"
$time 30.0 "$node_(1) 150.0 210.0 0.0"
$time 35.0 "$node_(0) 280.00 70.00 0.00"
$time 35.0 "$node_(1) 150.0 220.0 0.0"
$time 40.0 "$node_(0) 310.00 70.00 0.00"
$time 40.0 "$node_(1) 150.0 230.0 0.0"
$time 45.0 "$node_(0) 340.00 70.00 0.00"
$time 45.0 "$node_(1) 150.0 240.0 0.0"
$time 50.0 "$node_(0) 370.00 70.00 0.00"
$time 50.0 "$node_(1) 150.0 250.0 0.0"
```

8.4 Interfacing MATLAB with NetSim

8.4.1 Implement Nakagami 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 Nakagami Distribution from MATLAB

Procedure:

1) Create a MATLAB_Interface.c file inside the IEEE802_11 folder which can be found in the path <NetSim_Install_Direcotry>/src/Simulation/. 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
     */
    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 nakagami_shape=5,nakagami_scale=2;

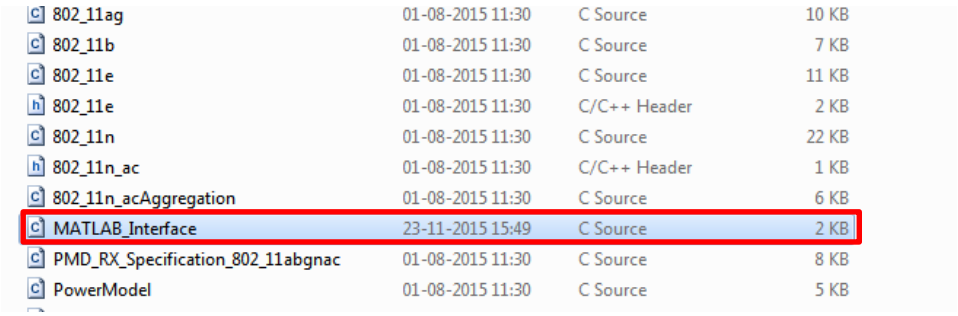
    engPutVariable(ep,"h",h);
    sprintf(buf,"h=ProbDistUnivParam('nakagami',[%d
%d])",nakagami_shape,nakagami_scale);
    status=engEvalString(ep,buf);

    engPutVariable(ep,"i",i);
    sprintf(buf,"i=random(h,1)");

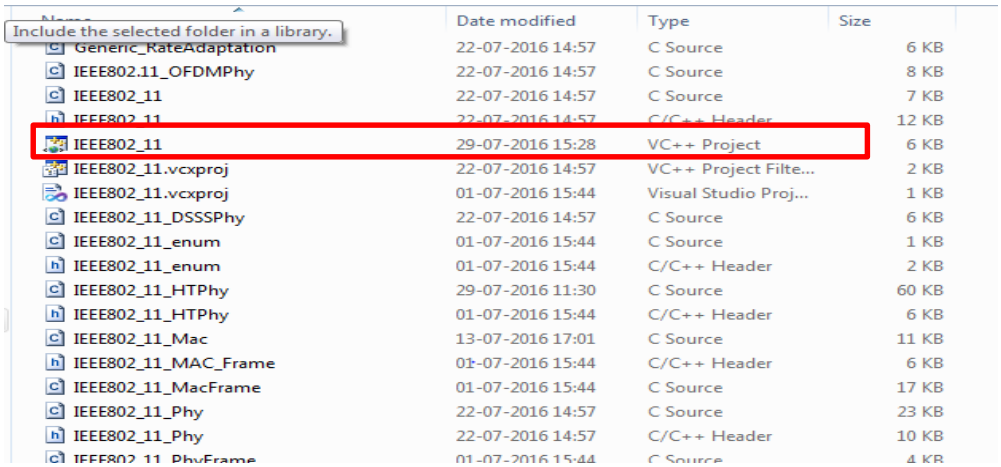
    status=engEvalString(ep,buf);
    out=engGetVariable(ep,"i");
    result=mxGetPr(out);

    return *result;
}

double fn_netsim_matlab_finish()
{
    status=engEvalString(ep,"exit");
    return 0;
}
```



2) Now open IEEE802_11 project file, inside the IEEE802_11 folder.



3) Right click on “IEEE802_11 Project” present in “Solution Explorer” window and select Add → Existing Item and select the MATLAB_Interface.c file.

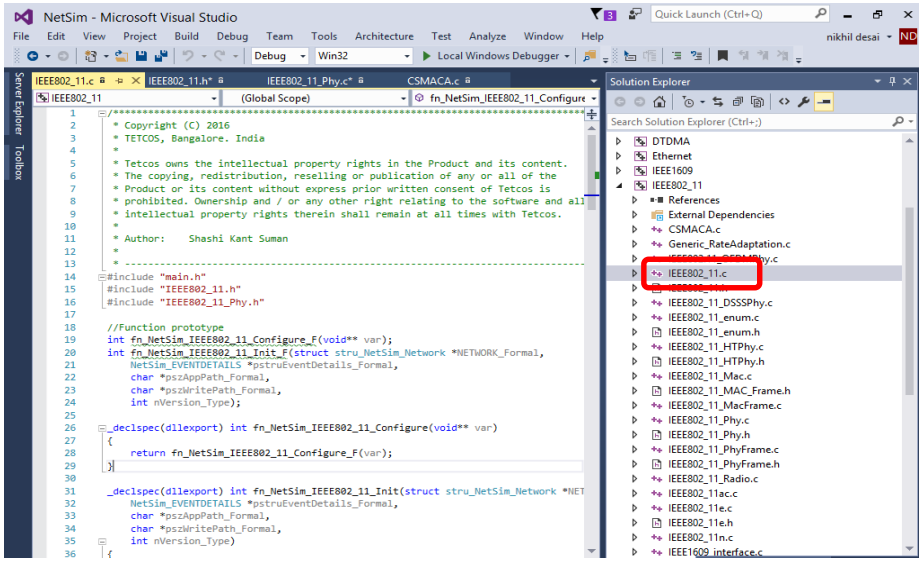
4) MATLAB_Interface.c file contains the following functions

a) fn_netsim_matlab_init() - Opens the MATLAB Engine

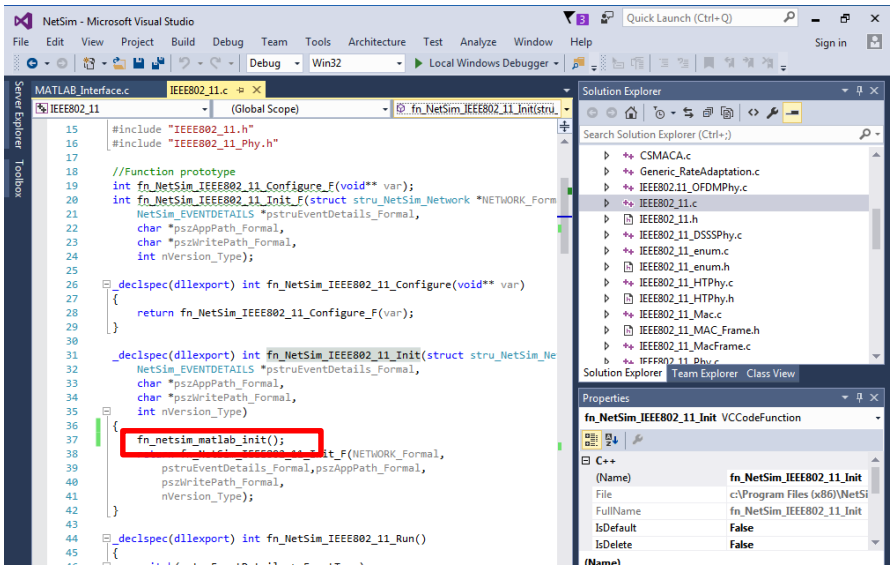
b) fn_netsim_matlab_run() - Communicates with MATLAB Command Window

c) fn_netsim_matlab_finish() - Closes the MATLAB Engine

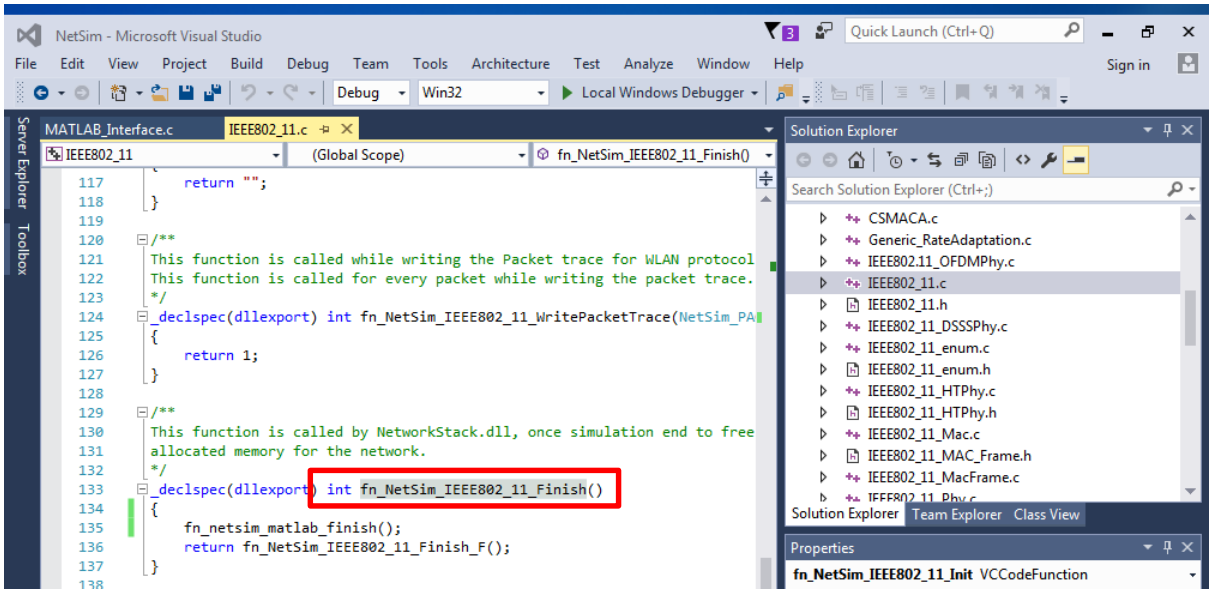
5) In the Solution Explorer double click on the IEEE802_11.c file.



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



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

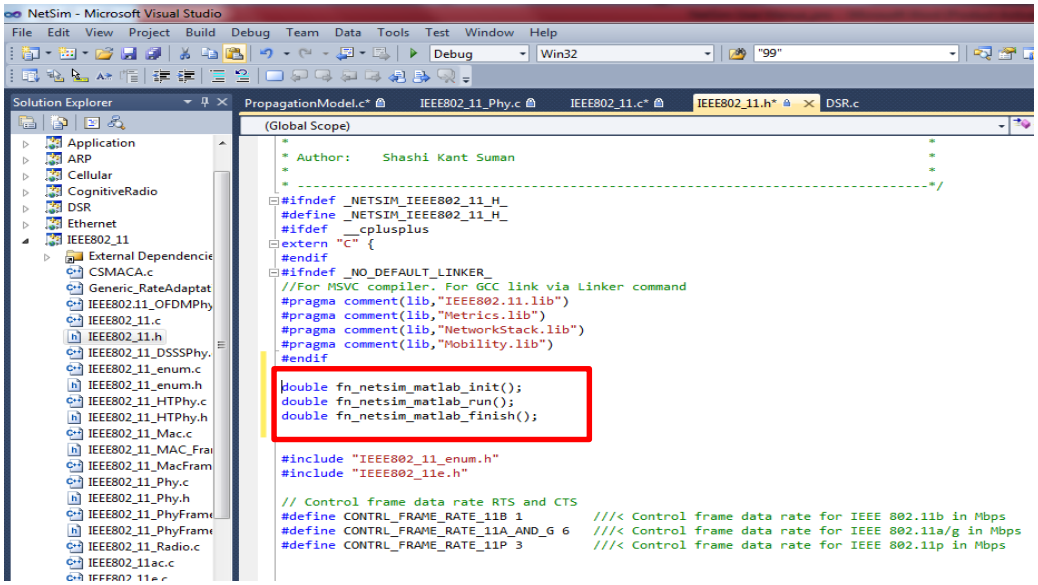


8) In the Solution Explorer double click on the IEEE802_11.h file. Add definitions of the following functions

double fn_netsim_matlab_init();

double fn_netsim_matlab_run();

double fn_netsim_matlab_finish();



9) In the Solution Explorer double click on the IEE802_11_PHY.c file.

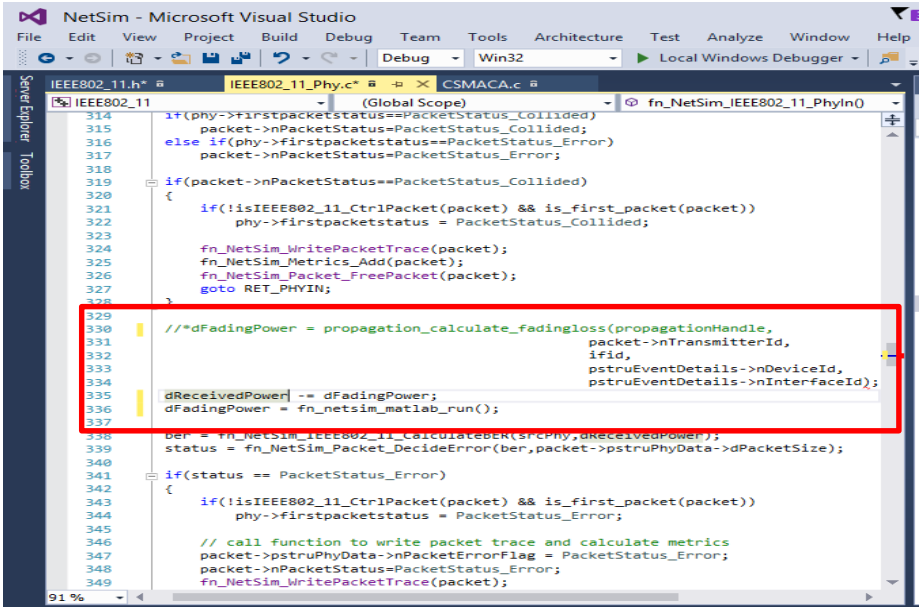
10) Inside fn_Netsim_IEEE802.11_PHYIn() function comment the lines,

dFadingPower = propagation_calculate_fadingloss(propagationHandle,

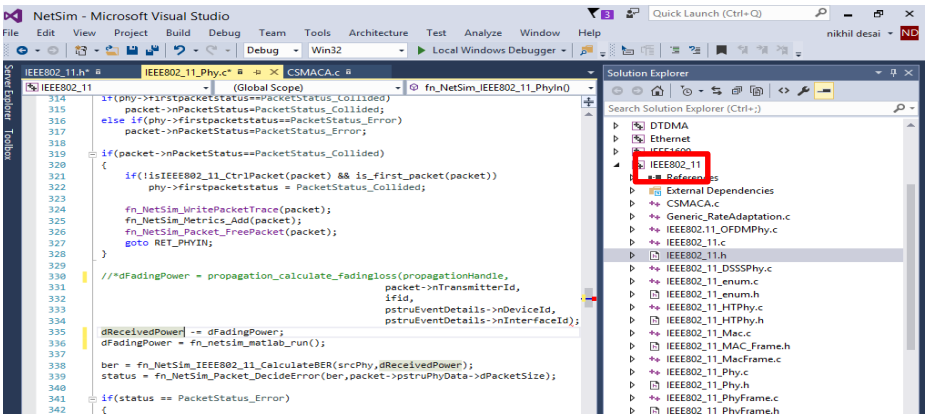
packet->nTransmitterId,ifid,pstruEventDetails->nDeviceId,
pstruEventDetails->nInterfaceId);

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

dFadingPower = fn_netsim_matlab_run();



11) To compile a MATLAB engine application in the Microsoft Visual Studio 15.0 (2015) environment, Right click on the IEEE802_11 project and select PROPERTIES in the solution explorer. Once this window has opened, make the following changes:

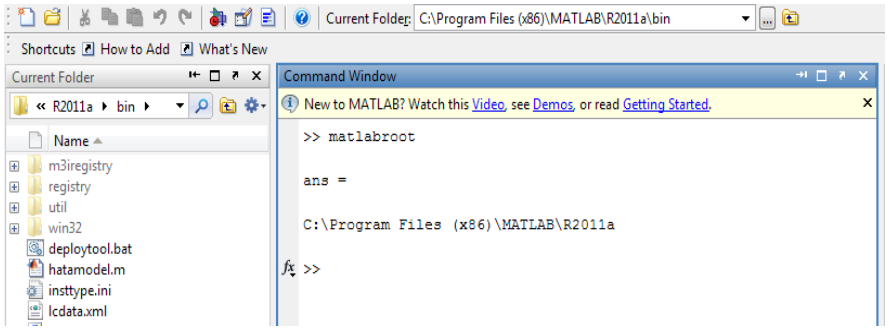


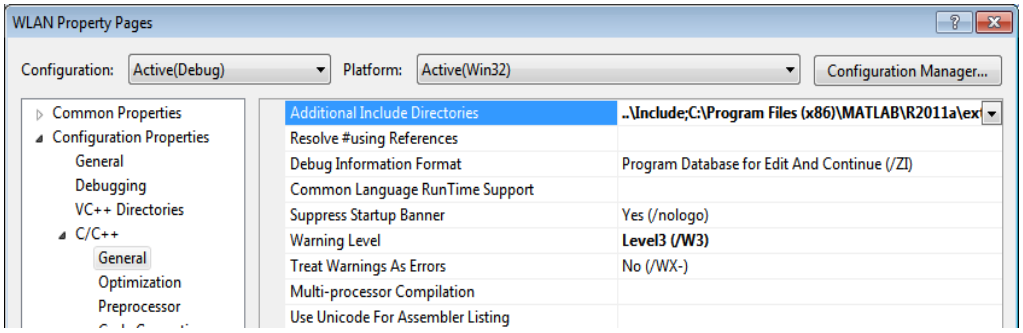
a) Under C/C++ → General, add the following directory to the field ADDITIONAL INCLUDE DIRECTORIES:

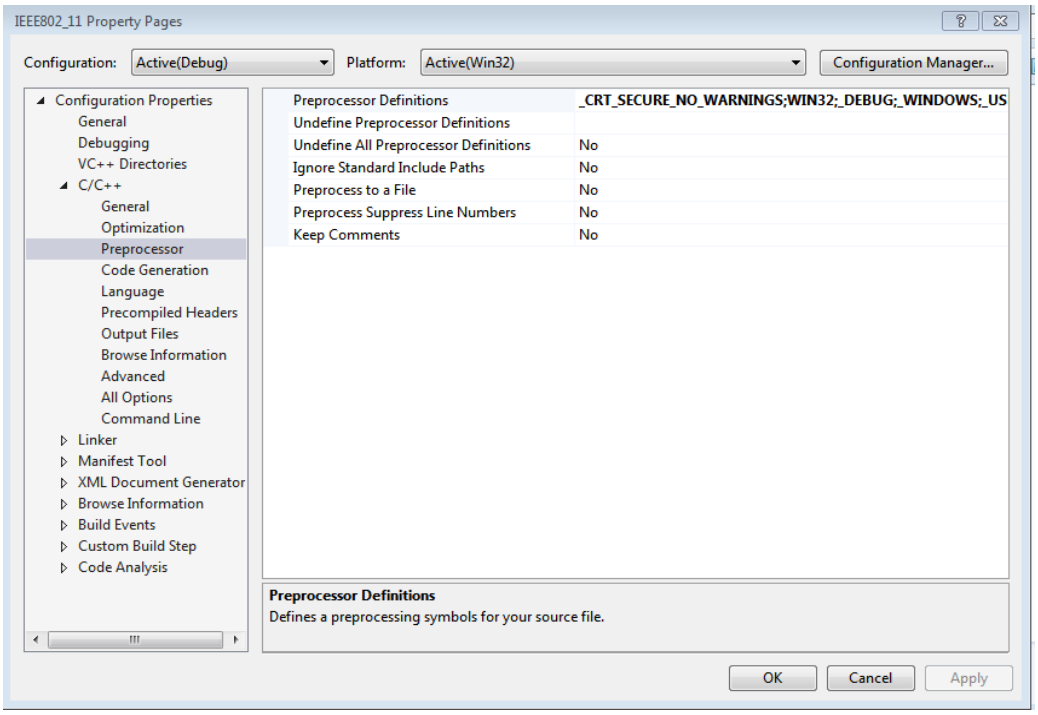
<Path where MATLAB is installed>\extern\include

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

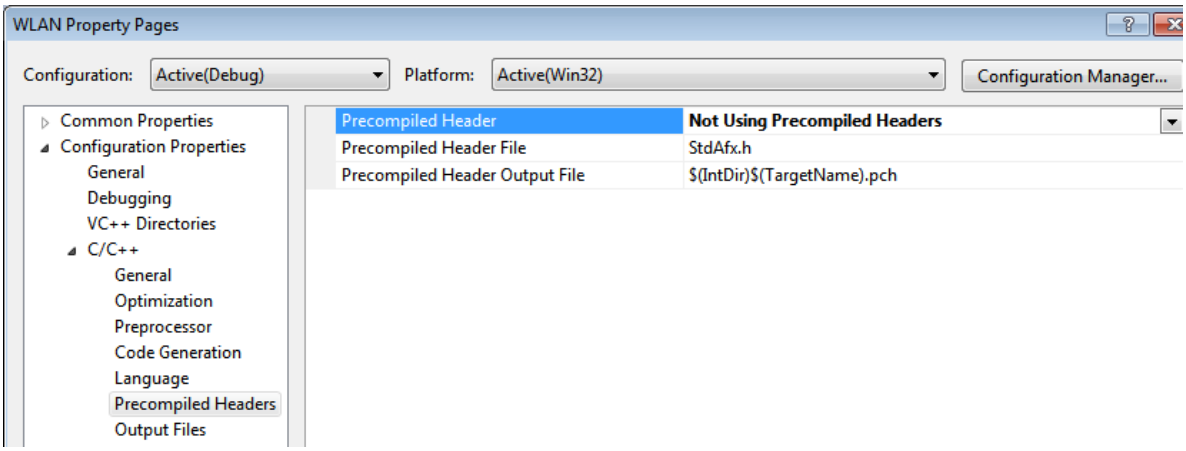
matlabroot





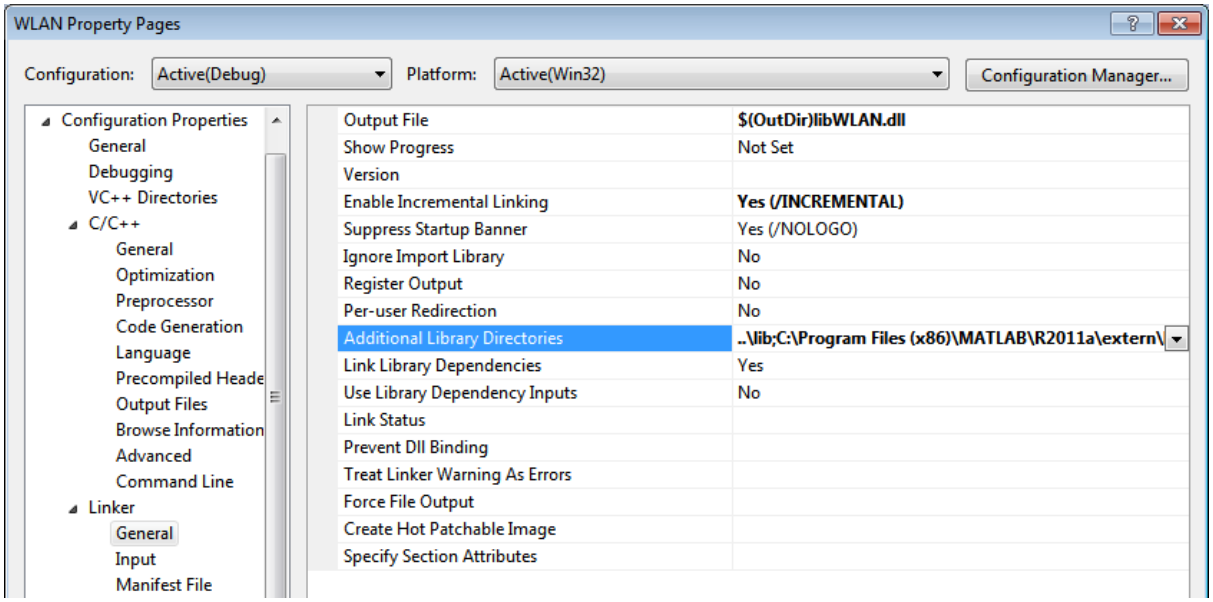


b) Under C/C++ → Precompiled Headers, set PRECOMPILED HEADERS as "Not Using Precompiled Headers".



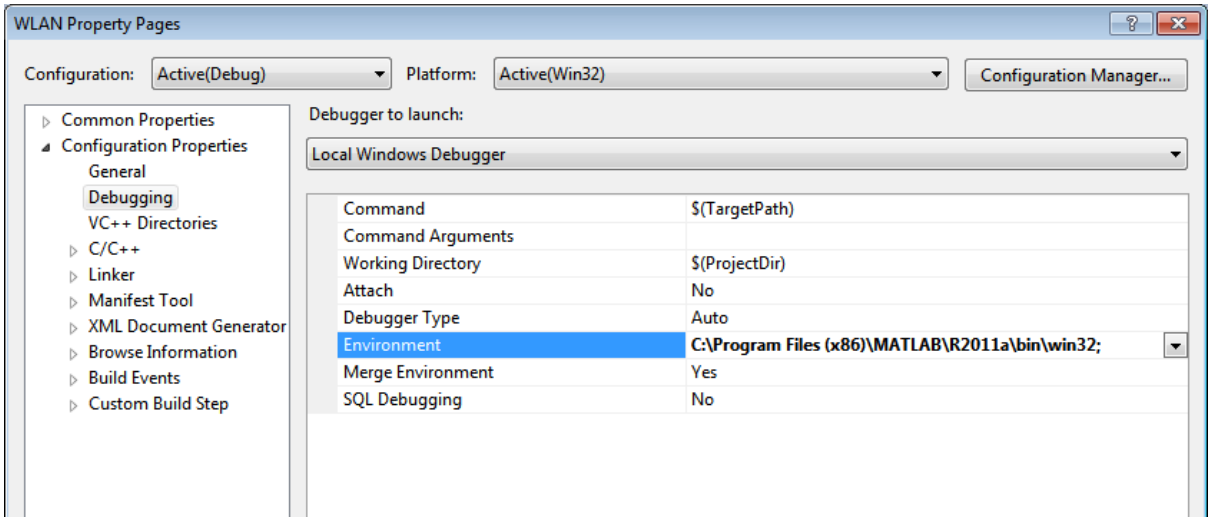
c) Under Linker → General, add the directory to the field ADDITIONAL LIBRARY DIRECTORIES:

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



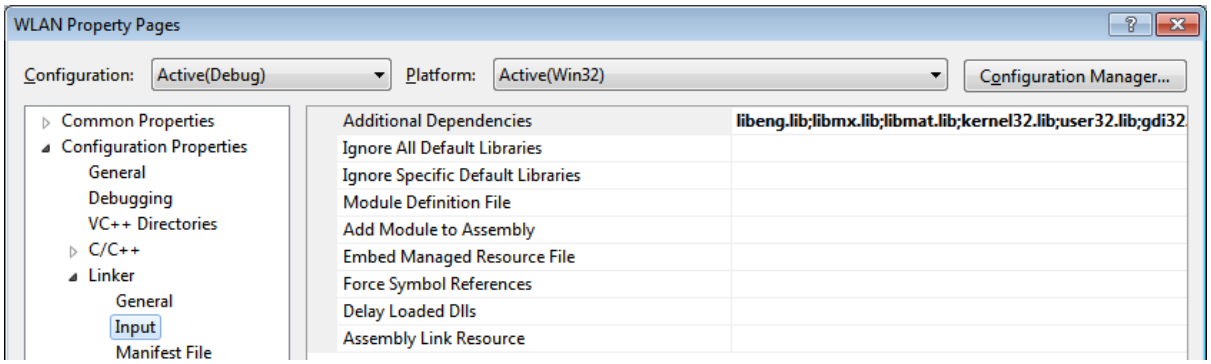
d) Under Configuration Properties →Debugging, Add the following Target path in the ENVIRONMENT:

<Path where MATLAB is installed>\bin\win32



12) Under Linker → Input, add the following names to the field marked ADDITIONAL DEPENDENCIES:

libeng.lib, libmx.lib, libmat.lib;



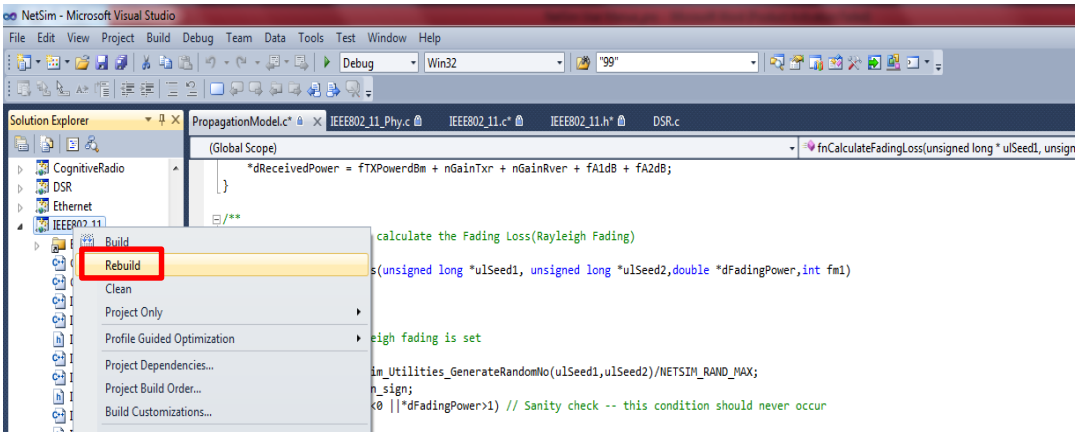
13) Make sure that the following directory is in the environment variable PATH:

<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).

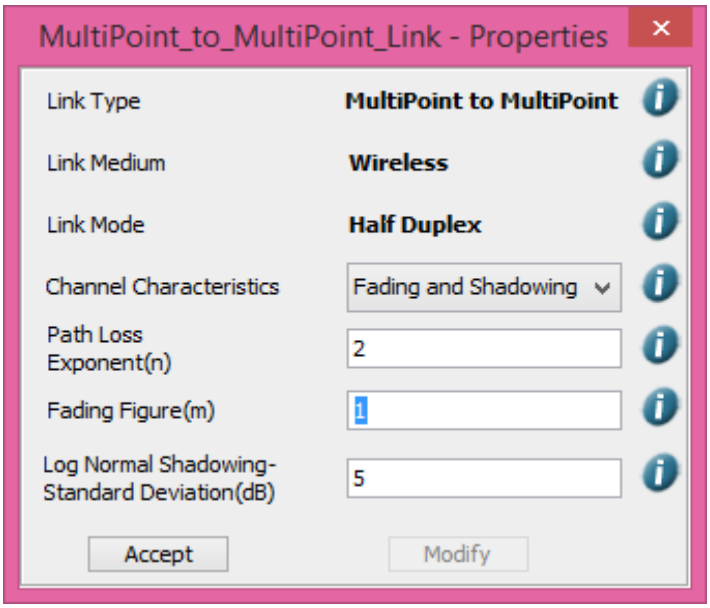
14) Now Right Click on IEEE802_11 project and select Rebuild.



15) Now replace the newly built libIEEE802.11.dll from the DLL folder, into the NetSim bin folder. Please ensure you rename the original libIEEE802.11.dll file to retain a copy of the original file.

[For more information, follow steps provided in “Writing your own code: Linking Dlls” under “Custom Code in NetSim” chapter]

16) Run NetSim in Administrative mode. Create a Network scenario involving IEEE802_11 say MANET, and set the Fading Figure value in the Multipoint to Multipoint Link properties to 1, to ensure that Rayleigh fading is set.



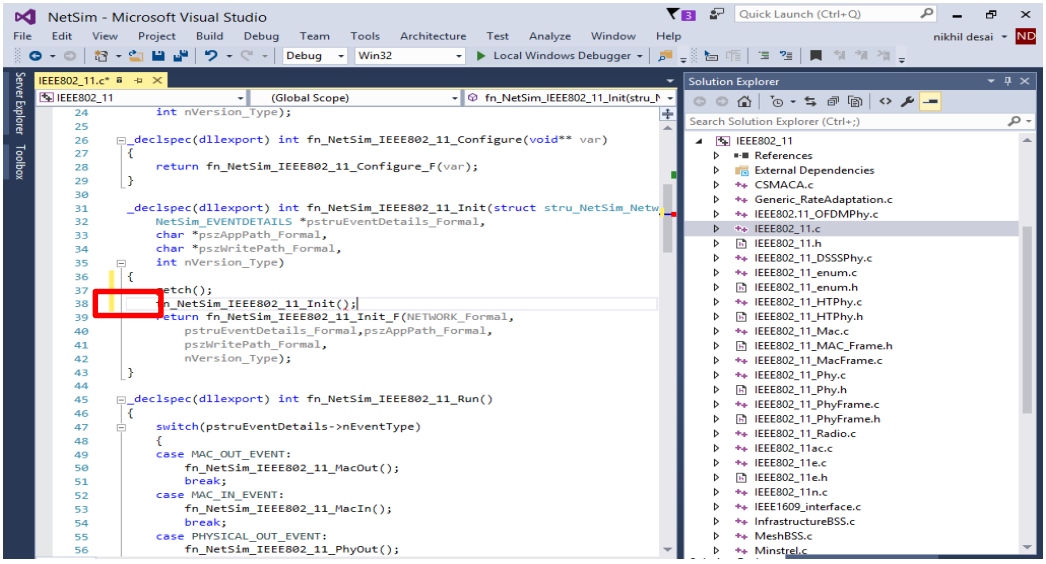
17) Perform Simulation. You will find that once the Simulation starts MATLAB command window starts and gets closed once the simulation is over.

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:

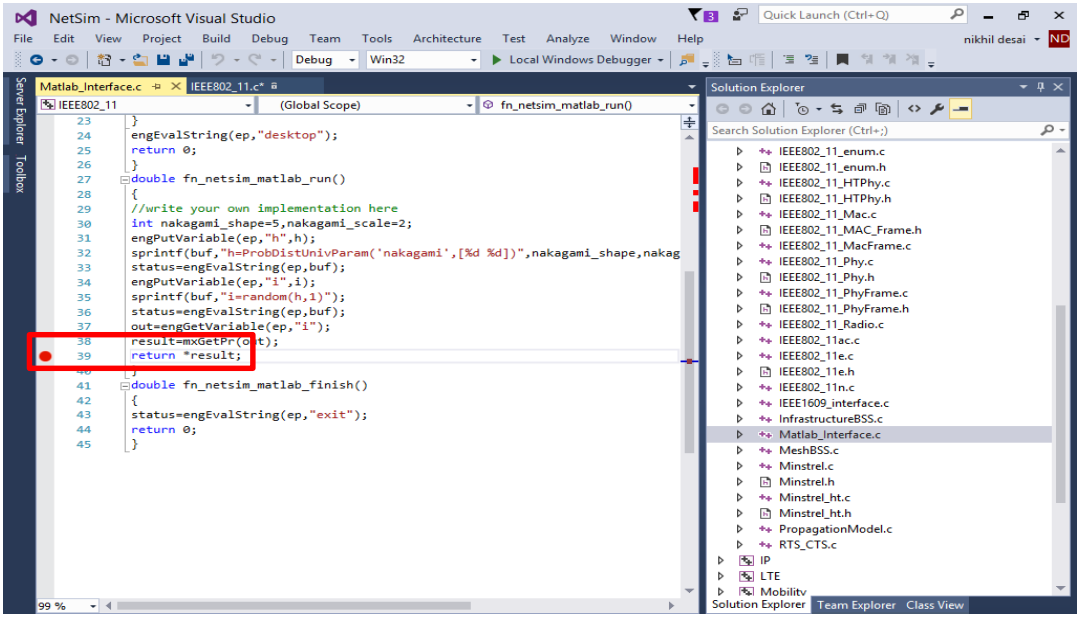
```
!matlab -regserver
```

8.4.2 Debug and understand communication between NetSim and MATLAB

1. In the Solution Explorer of Visual Studio double click on IEEE802_11.c and Add a getch() inside the fn_NetSim_IEEE802_11_Init() function.



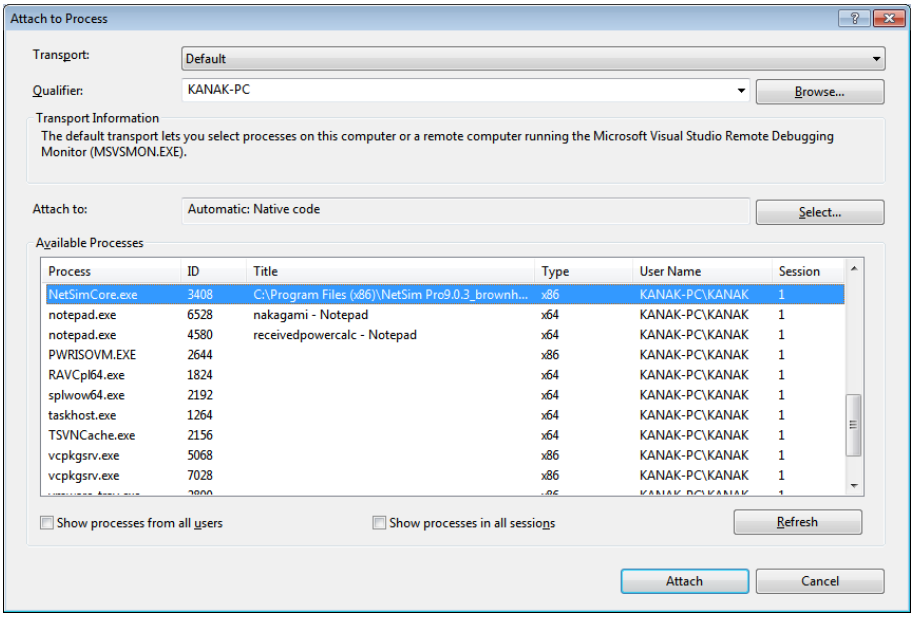
2. In the Solution Explorer double click on MATLAB_Interface.c file and place a breakpoint inside the fn_netsim_matlab_run() function before the return statement.



3. Rebuild the Dll and replace in bin path.
4. Now run the NetSim Scenario. The simulation window stops for user interrupt.

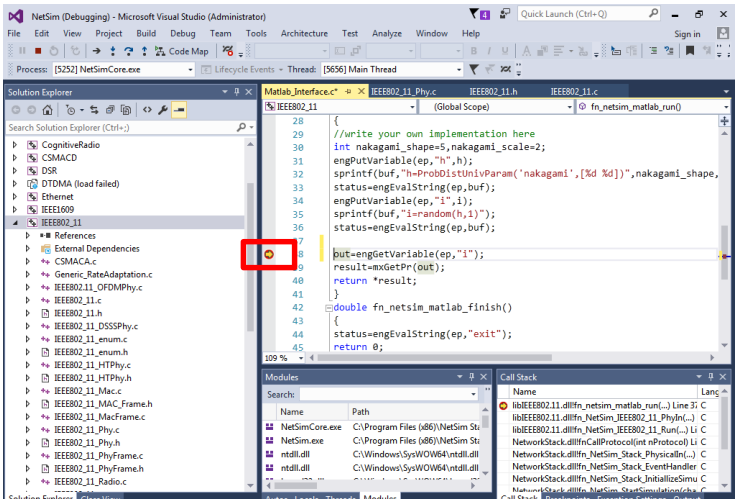
5. In Visual studio, go to Debug → Attach to Process.

6. From the list of Processes select NetSimCore.exe and click on Attach.

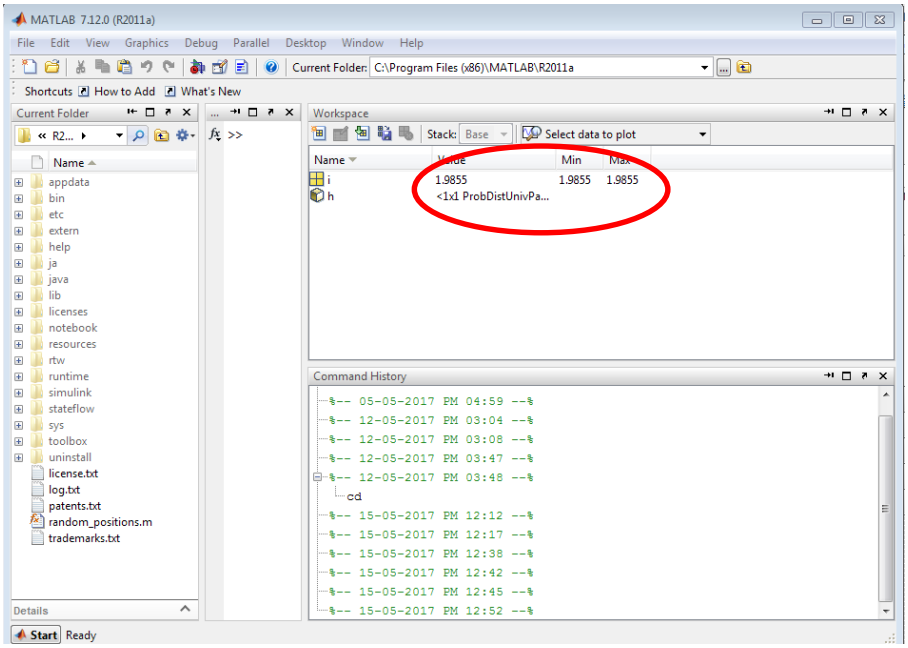


7. Now go to the Simulation window and press Enter.

8. MATLAB Command Window and MATLAB Desktop Window will start and breakpoint in Visual Studio gets triggered.

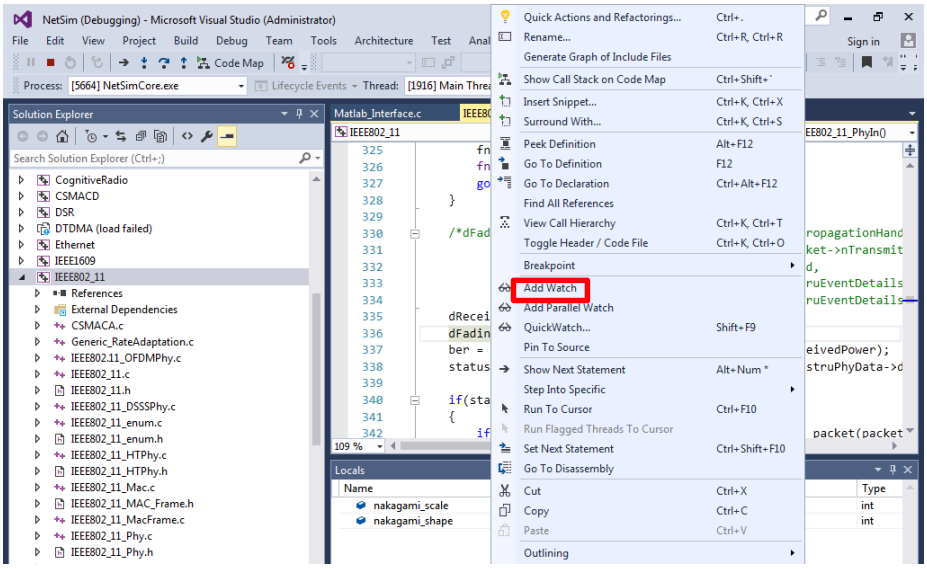


9. Now when debugging (say, by pressing F5 each time) you will find the computation taking place in the MATLAB Workspace.

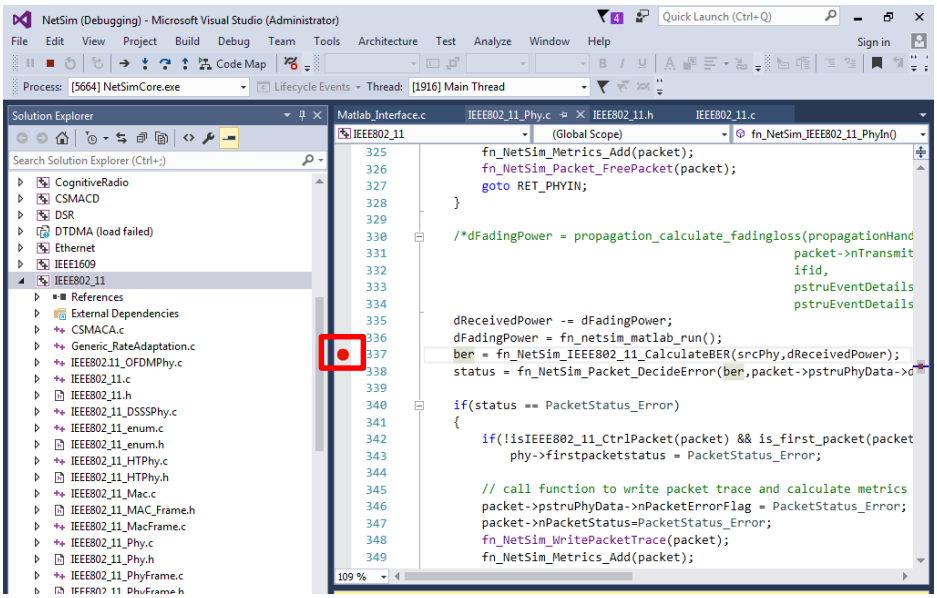


10. This value of i obtained from MATLAB is used to calculate fading power instead of the Rayleigh Fading Model.

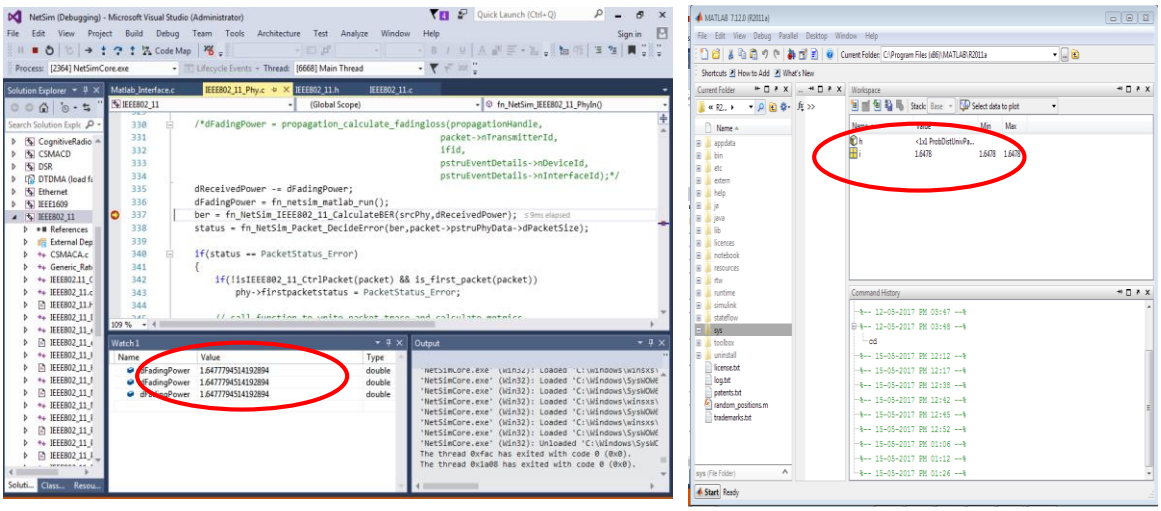
11. Add a watch to the *dFadingPower variable in the PropagationModel.c file. For this, right click on the variable *dFadingPower and select “Add Watch” option. You will find a watch window containing the variable name and its value in the bottom left corner.



12. Now place another breakpoint after the line dFadingPower = fn_netsim_matlab_run()



13. 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.



8.4.3 Implement Nakagami Distribution of MATLAB in NetSim using .m file:

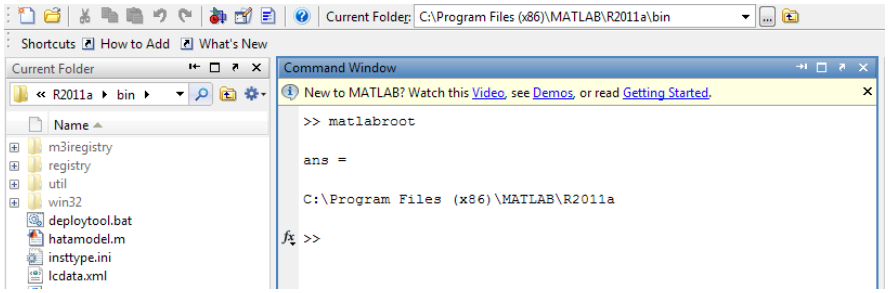
Procedure:

1. From the NetSim_MATLAB_Interface.zip, Copy and paste NetSim_MATLAB_Interface /Example1a /nakagami.m file inside <Path where MATLAB is installed>. The nakagami.m file contains the following code:

```
function WLAN=nakagami(scale,shape)
h=ProbDistUnivParam('nakagami',[scale,shape]);
i=random(h,1);
WLAN=i;
```

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

matlabroot



2. From the NetSim_MATLAB_Interface.zip, Copy and Paste NetSim_MATLAB_Interface /Example1a/ MATLAB_Interface.c file inside the IEEE802_11 folder.

Replace the MATLAB_Interface.c file used for "Implement Nakagami Distribution of MATLAB in NetSim without using .m file" (incase if you have performed that section before).

3. Follow steps 2 to 14 as given in "Implement Nakagami Distribution of MATLAB in NetSim without using .m file".

4. Now double click on MATLAB_Interface.c file in the Solution Explorer if a link exists, else right click on IEEE802_11 project and select add existing item to add the file to the project.

5. Go to the fn_netsim_matlab_run() function

```
double fn_netsim_matlab_run()
{
	//write your own implementation here
	int nakagami_shape=5,nakagami_scale=2;

	sprintf(buf,"k=nakagami(%d,%d)",nakagami_shape,nakagami_scale);
	status=engEvalString(ep,buf);
	out=engGetVariable(ep,"k");
	result=mxGetPr(out);
	return *result;
}
```

6. A call to the nakagami() function inside the nakagami.m file is made, and nakagami_shape and nakagami_scale parameters are passed from NetSim.

7. Right Click on IEEE802_11 project and select Rebuild.

8. Replace the newly build libIEEE802.11.dll from the DLL folder in the NetSim bin folder after renaming the original libIEEE802.11.dll file.

9. Create a Network scenario involving IEEE802_11 say MANET, and set the Fading Figure value in the Multipoint to Multipoint Link properties to 1 to ensure that Rayleigh fading is set.

10. 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.

8.4.4 Plot a histogram in MATLAB using the values generated by Nakagami distribution for NetSim (using .m file)

Procedure:

1. From the NetSim_MATLAB_Interface.zip, Copy and paste NetSim_MATLAB_Interface/Example1b/NetSim_MATLAB.m file inside $MATLABROOT.

The NETSIM_MATLAB.m file contains the following code:

```
function WLAN=NETSIM_MATLAB(choice,varargin)

switch(choice)

        case 'nakagami'

        h=ProbDistUnivParam('nakagami',[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. From the NetSim_MATLAB_Interface.zip, Copy and Paste NetSim_MATLAB_Interface/Example1b/ MATLAB_Interface.c file inside the IEEE802_11 folder. Replace the MATLAB_Interface.c file if it already exists.

3. Follow steps 2 to 14 as given in Example 1

4. Now double click on MATLAB_Interface.c file in the Solution Explorer if a link exists, else right click on IEEE802_11 project and select add existing item to add the file to the project.

5. Go to the fn_netsim_matlab() function

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

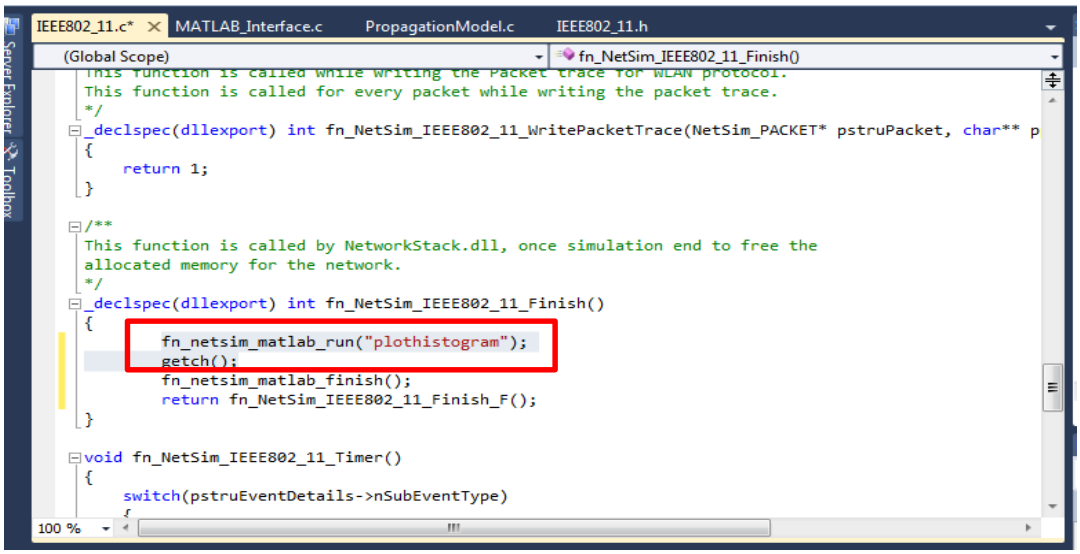
        int nakagami_shape=5,nakagami_scale=2;

        if(strcmp(arr,"nakagami")==0)
        {
        engPutVariable(ep,"h",h);
        sprintf(buf,"h=NETSIM_MATLAB('nakagami',%d,%d)",nakagami_shape,nakagami_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;
}
```

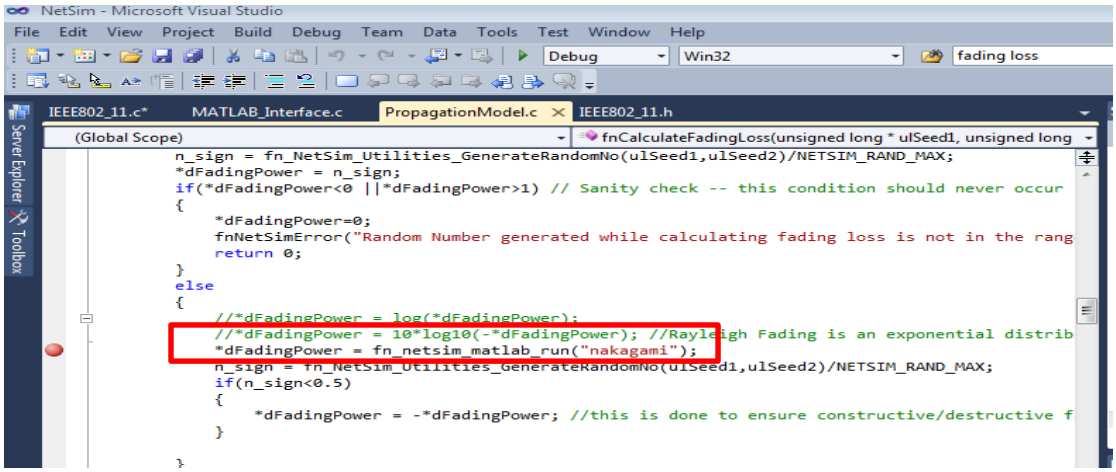
6. A call to the NetSim_MATLAB() function inside the NetSim_MATLAB.m file is made, for fading power calculation with parameters distribution(‘nakagami’), nakagami_shape and nakagami_scale parameters are passed from NetSim.

7. A call to the NetSim_MATLAB() function inside the NetSim_MATLAB.m file is made, for plotting histogram for the values generated by MATLAB..

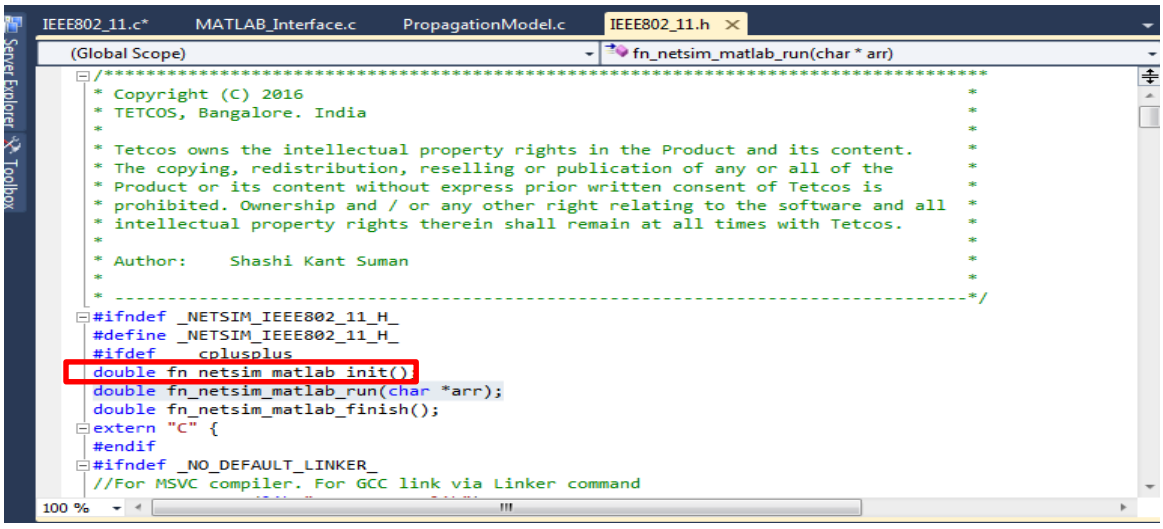
8. Also add the following call to fn_netsim_matlab_run() function along with a getch() to plot the histogram before closing the MATLAB Engine.



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



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

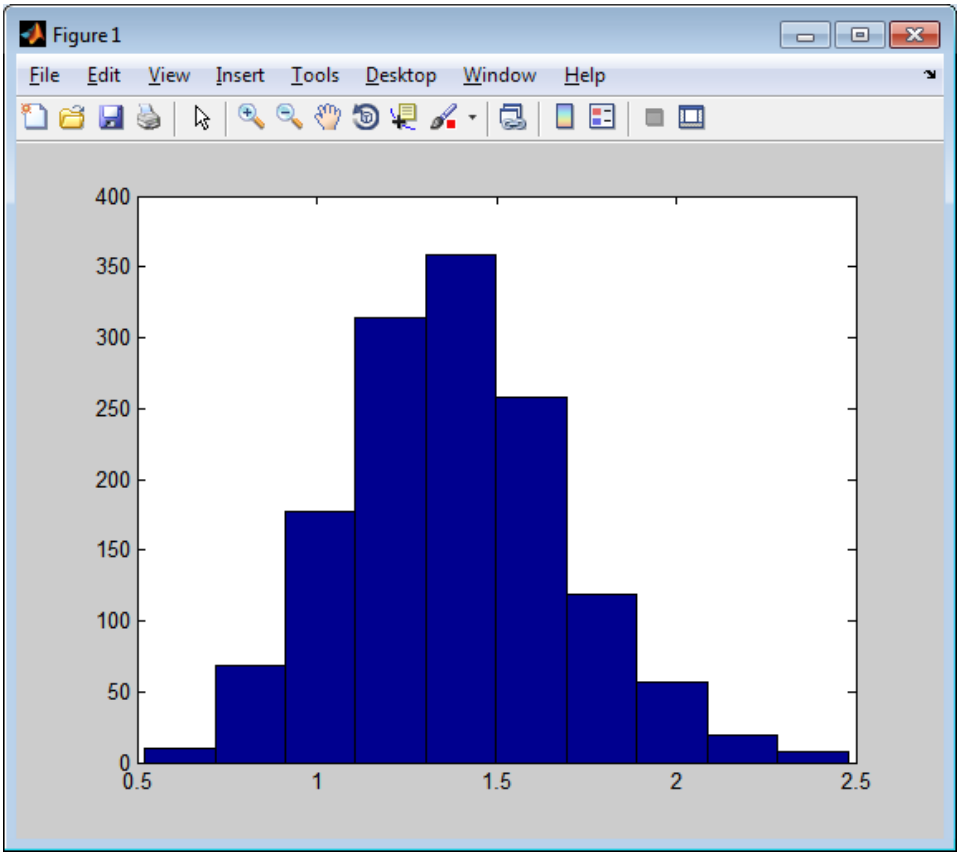


11. Right Click on IEEE802_11 project and select Rebuild.

12. Now Replace the newly build libIEEE802.11.dll from the DLL folder in the NetSim bin folder after renaming the original libIEEE802.11.dll file.

13. Create a Network scenario involving IEEE802_11 say MANET, and set the Fading Figure value in the Multipoint to Multipoint Link properties to 1 to ensure that Rayleigh fading is set.

14. 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 nakagami distribution.



15. 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.

8.5 Adding Custom Performance Metrics

In Performance Metrics, users have the latitude of adding their own customized metric variables by editing the source code of the protocol of that specific networking technology. The following example provides a better understanding of how to implement it.

For illustration, an example regarding Wireless Sensor Network is provided. In this example, users will print Sensor Node Name, Residual Energy, State (On/Off) and turn–off time in the performance metrics

STEP 1: Copy the provided code at the top in 802_15_4.h file

```
#include "string.h"

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

STEP 2:

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(PMETRICSWRITER metricsWriter)
{
        //CUSTOM METRICS
        PMETRICSNODE table = init_metrics_node(MetricsNode_Table,
"Custom_WSN_Metrics", NULL);
        NETSIM_ID nDeviceCount = NETWORK->nDeviceCount;
        add_table_heading(table, "Node Name", true, 0);
        add_table_heading(table, "Status", true, 0);
        add_table_heading(table, "Time", true, 0);
        add_table_heading(table, "Residual_Energy", true, 0);
        int i;
        for (i = 1; i <= nDeviceCount; i++)
        {
                add_table_row_formatted(false, table, "%s,%s,%.2lf,%.2lf,",
```

```
                    NetSim_Node_name[i - 1], NetSim_Node_state[i - 1],
NetSim_Off_Time[i - 1], NetSim_Residual_Energy[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 3:

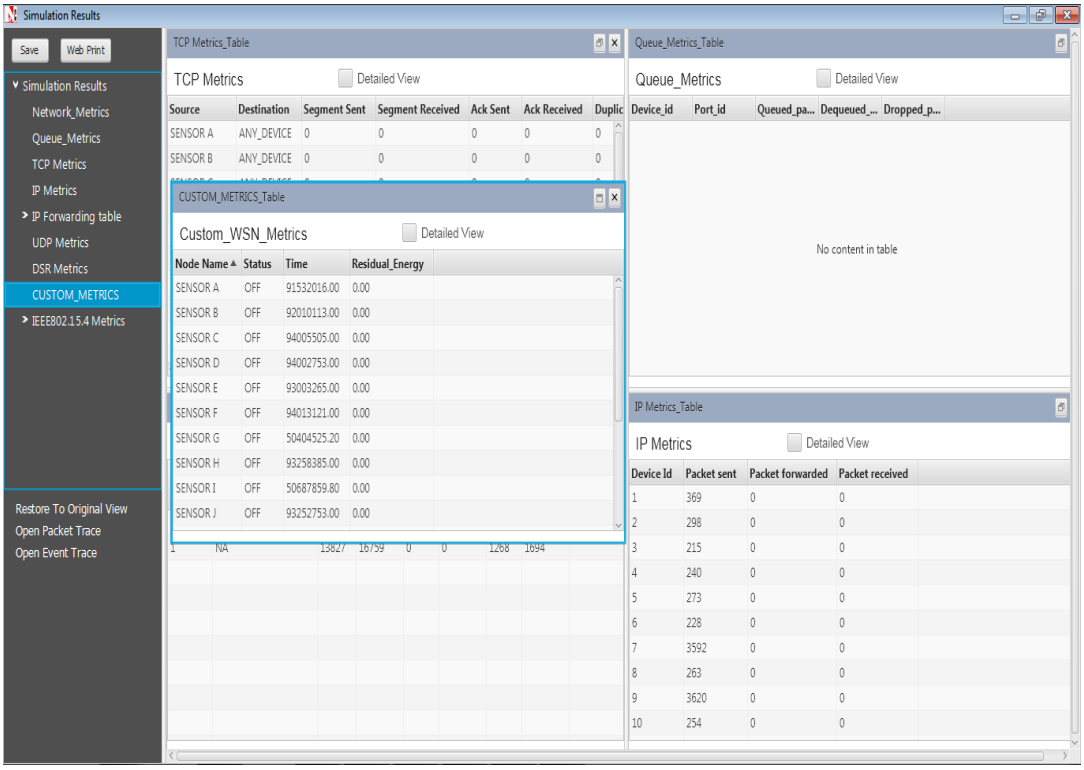
Copy the below code (in red colour) at the end of ChangeRadioState.c file (inside IF(nStatus) loop)

```
if(nStatus)
 {
        WSN_PHY(nDeviceId)->nOldState = nOldState;
        WSN_PHY(nDeviceId)->nRadioState = nNewState;
        NetSim_Node_state[nDeviceId-1]= "ON";
        NetSim_Node_name[nDeviceId-1]=    NETWORK->ppstruDeviceList[nDeviceId-1]-
>szDeviceName;
        return nStatus;
 }
        else
 {
        WSN_PHY(nDeviceId)->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 nStatus;
  }
```

STEP 4:

Build DLL with the modified code and run a Wireless Sensor Network scenario.

After Simulation, user will notice a new Performance metrics named **"Custom WSN Metrics"** is added.



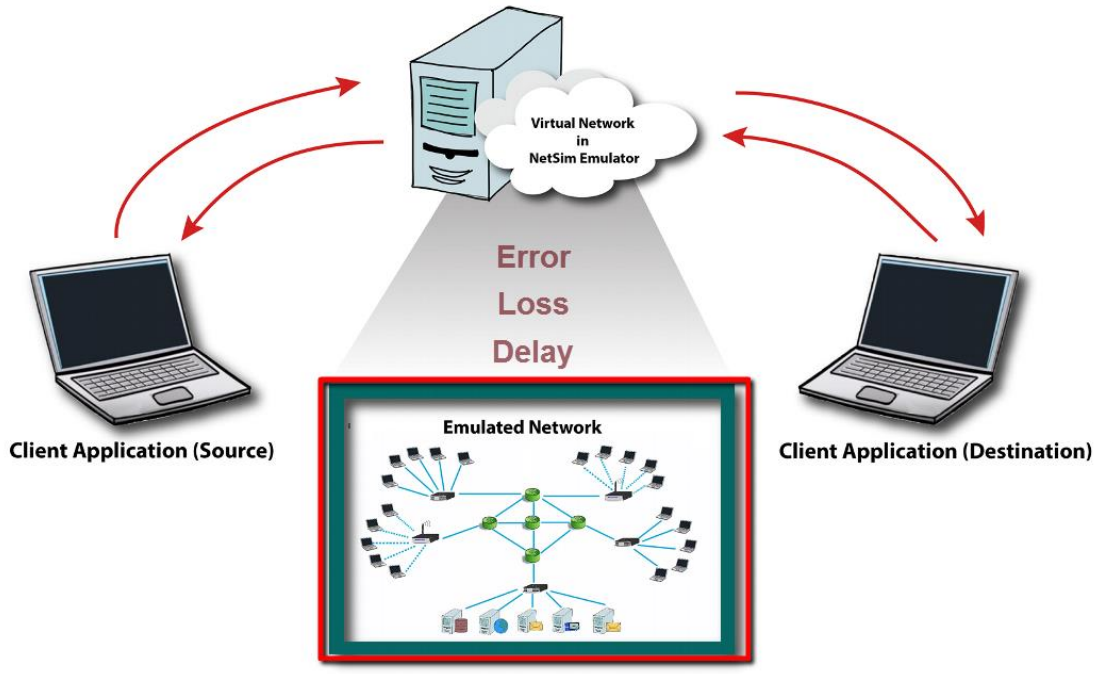
9 NetSim Emulator

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

9.1 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.

9.1.1 Emulation: How Simulation interacts with the real world



A real PC (running NetSim Emulation Client) sends live traffic to the PC (running NetSim Emulation Server). Whenever a packet arrives at the interface of server, this packet is "modulated" into a simulation packet and sent from a source node (user selectable) in the simulated network (user configurable) to a destination node (again user selectable). Upon receipt of this packet at the destination, the packet is then "de-modulated" and sent back to a real PC destination node (running NetSim Emulation Client). The real packet thus undergoes network effects such as delay, loss, error etc. created virtually by NetSim Simulator.

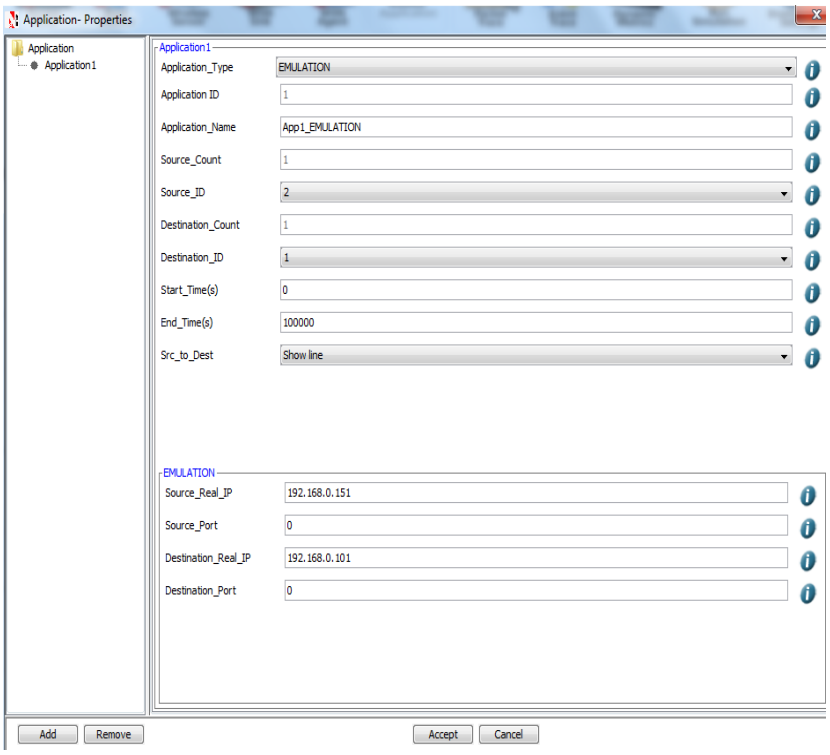
9.2 Emulation Set-up:

The ideal set-up to run emulation would be to have a minimum of three (3) PC's. One would be the real source, the second would run NetSim emulation server, and the third would be the real destination.

Alternately, this set-up can also be managed where the two (2) PC's are running client applications that communicate with the central server, where NetSim Emulation server is running.

9.2.1 Setting up the NetSim Server:

- Run NetSim in Administrative Mode (Right Click on NetSim.exe → Run as Administrator).
- User has to open any Stack based Network (Any network except Legacy Networks, Wireless Sensor Network, Zigbee Network and Cellular Network) in NetSim with Emulation.
- Create a network scenario of your choice (refer application examples provided) and set the Application properties.

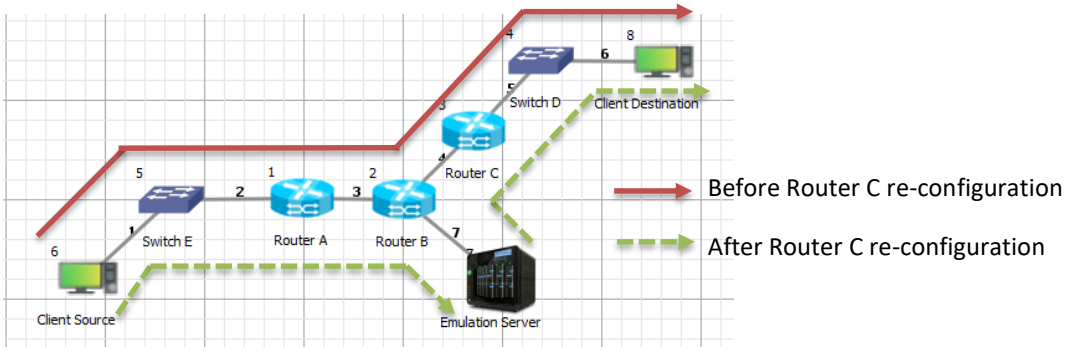


- In the Application Properties, set Application Type as “EMULATION”. Assign real Source IP address and Destination IP address in the respective fields. Then Click Accept.
- Set the Simulation Time as how long you want to perform the Emulation in Real World. **Do not run the simulation until setting up Emulation in the Client system**.

NOTE: If the Emulation Server is located in a different subnet from clients

- User has to configure the router settings of the real-world network so as to allow the packets to be transmitted to the Emulation Server

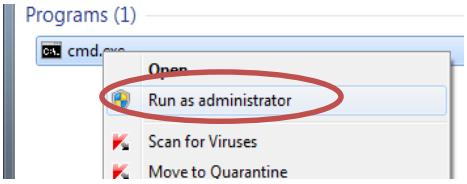
  For Example, if we consider a sample real world network scenario where the Emulation clients and server are located in different subnets



- Routing table of router C needs to be configured such that any packet having Source Address as IP Address of Node 6 (Client Source) and Destination Address as IP Address of Node 8 (Client Destination) must be routed to Emulation Server. NetSim configuration will the ensure that the packet is re-injected with destination set to the appropriate IP Address (set in the application properties)

9.2.2 Setting up the Client systems (Real Source and Destination system)

- Open command prompt in administrative mode.



- Type command,

  route delete <Network Address>

  then press Enter key. You will get “OK”. For example if your IP address is 192.168.0.4 and the subnet mask is 255.255.255.0 then the network address is 192.168.0.0 (Got by performing a bitwise AND of the IP Address and the subnet mask)

- Type command

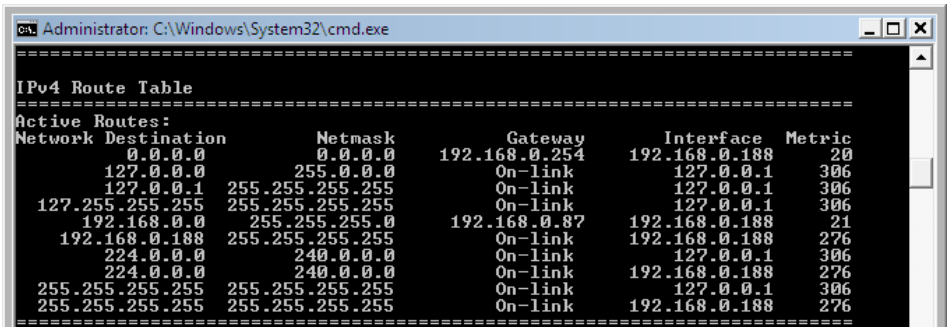
  route add <Network Address>**mask 255.255.255.0** <IP Address where NetSim Emulation server is running> **metric 1**

  here the subnet mask is taken as 255.255.255.0). After execution, you will get “OK”.

- Type command

  netstat -r

  To check if the IP configuration is done or not.

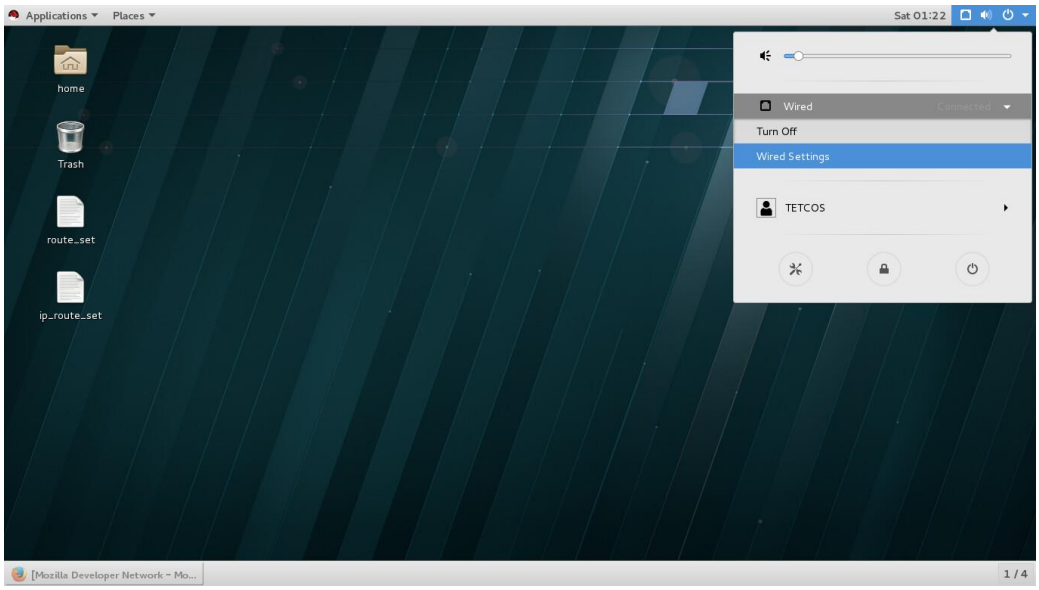


Note that in the above screenshot, for the network 192.168.0.0, the gateway address assigned is 192.168.0.87 (Address of the system where NetSim Emulation Server is running).

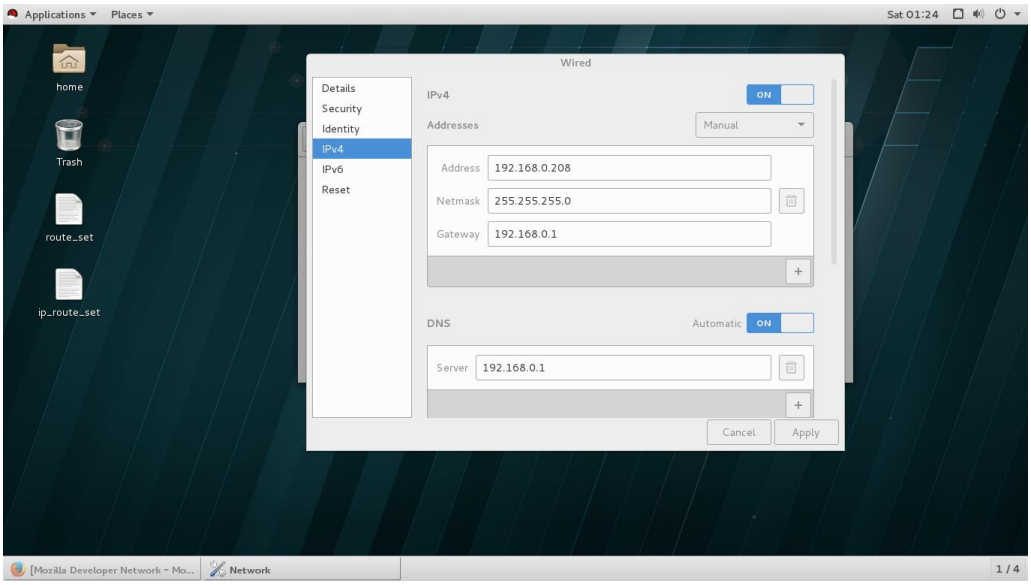
9.2.3 Setting up the Linux Client Systems (Real Source and Destination system) for Emulation

Note: Following Screenshots are relevant to RHEL 7. Equivalent settings has to be done in case of other Linux Variants.

Go to the Wired Settings option in the Network Adapter Icon.

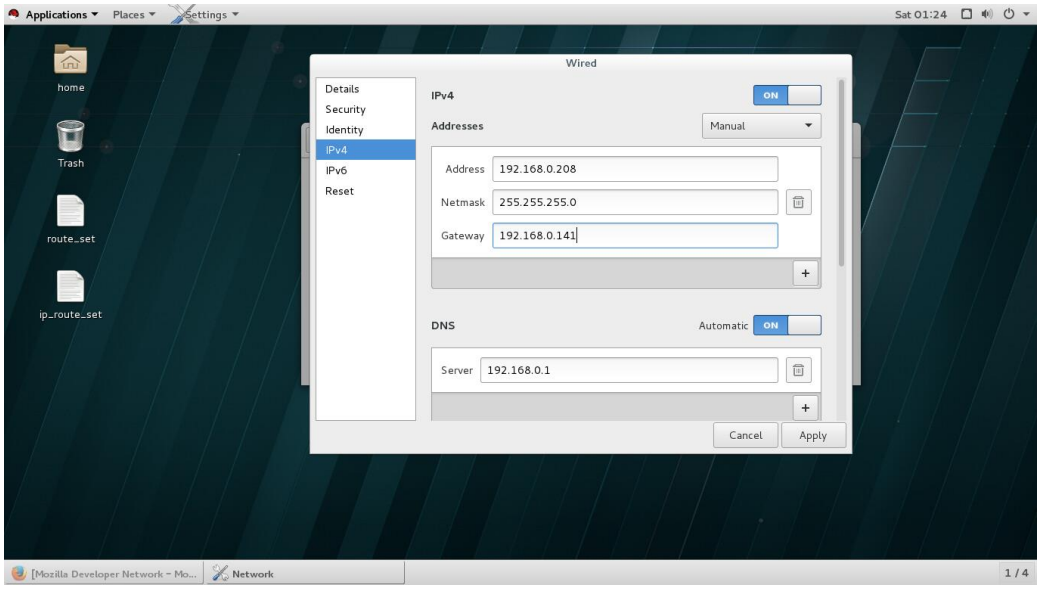


In the IPV4 settings, set static IP Address to the machine and specify the Emulation Server IP as the Gateway IP.

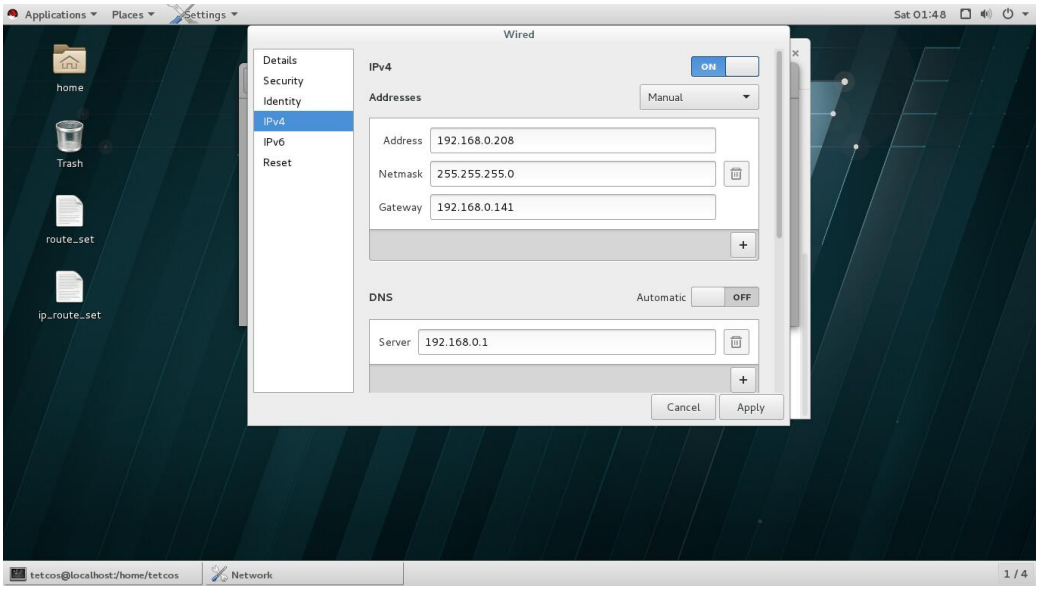


Eg:

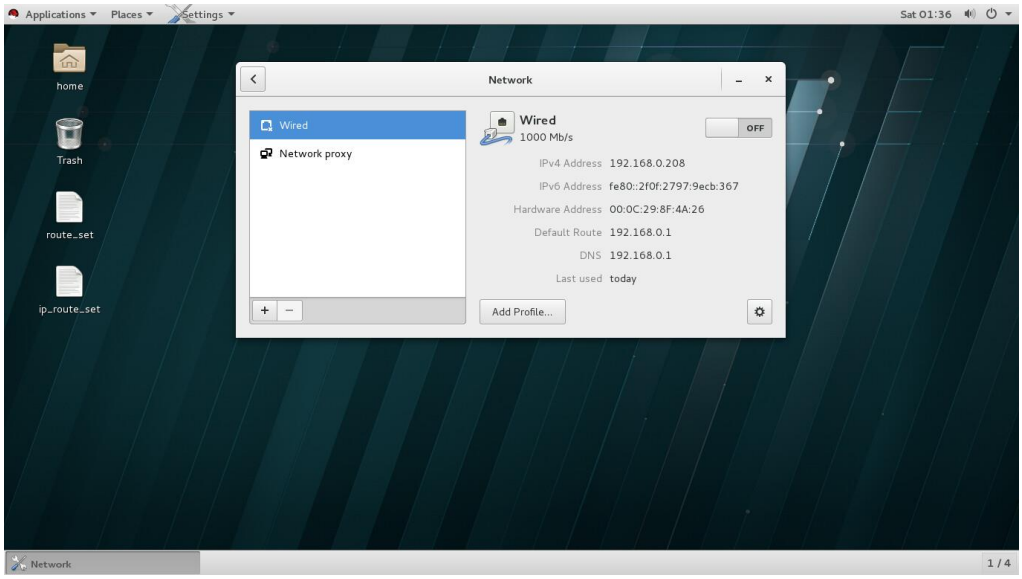
If 192.168.0.141 is the IP of the system where Emulation Server is running. This is specified as the gateway IP.

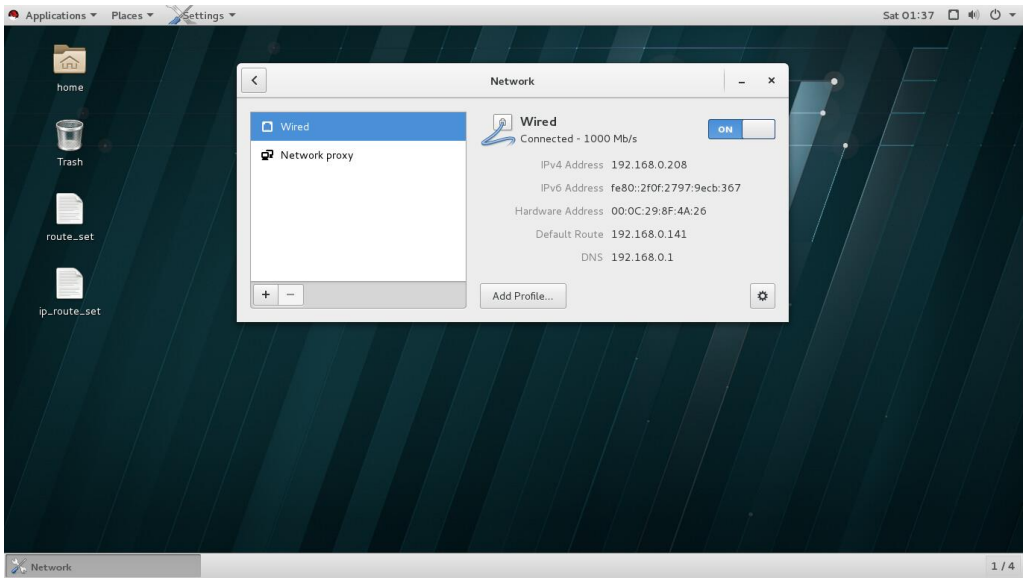


Turn off Automatic DNS.



Turn off and on the Network Adapter





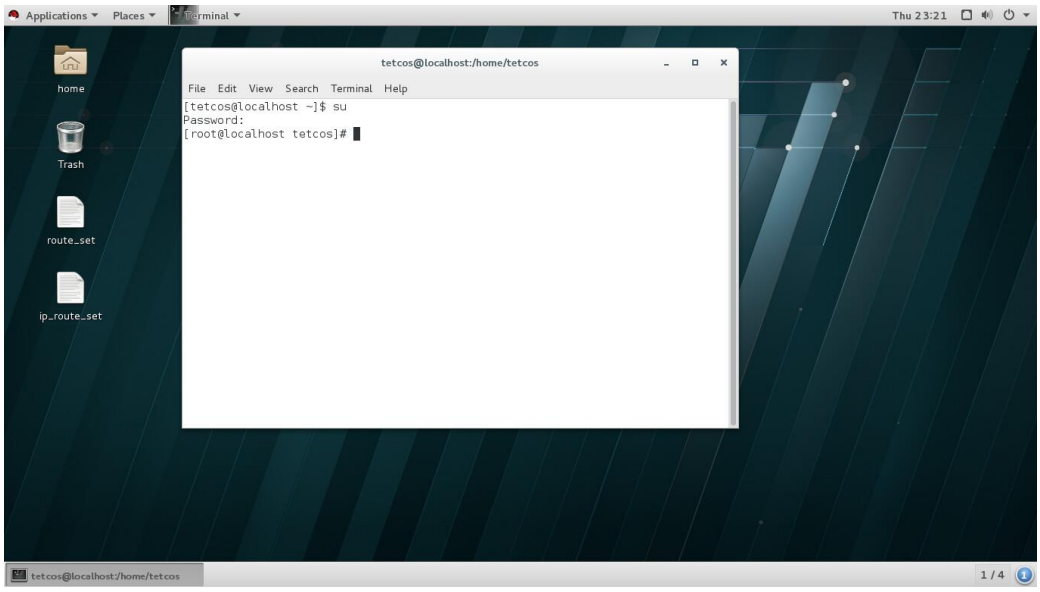
Open terminal window

Type command

su

This is to switch to root user.

Enter the root password

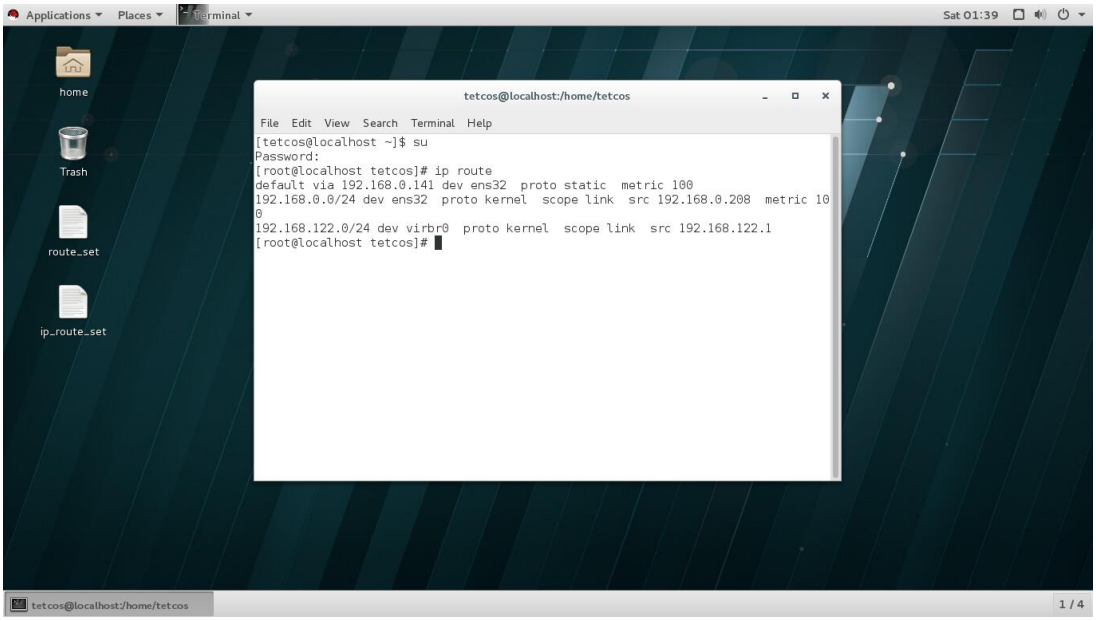


Type command

ip route

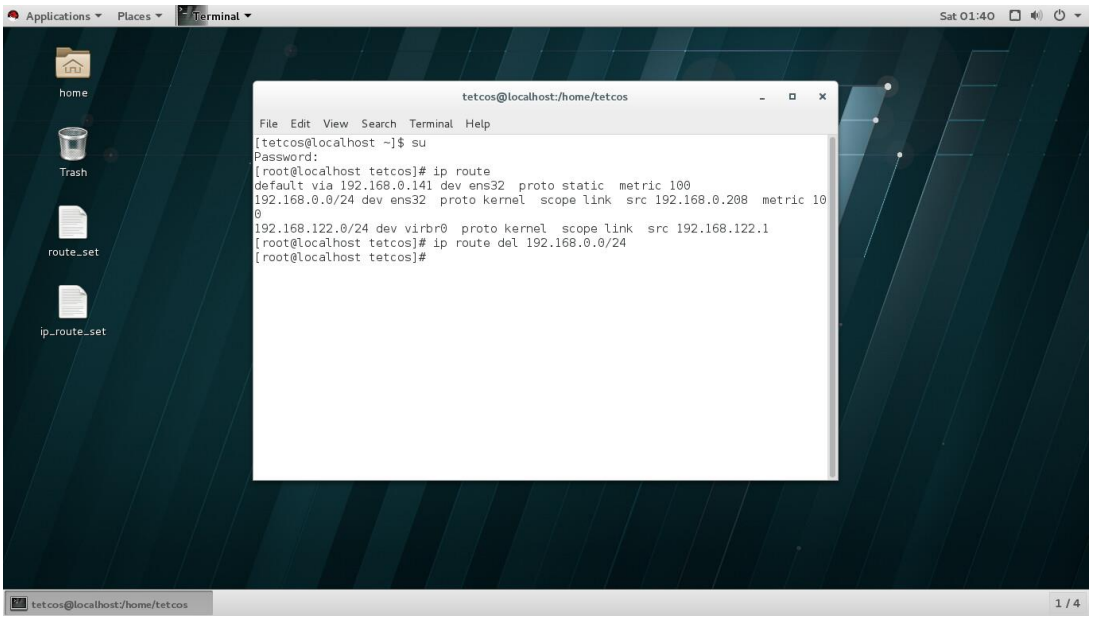
This is to check the default route

It will now show the default via <Emulation server IP>



Type command

ip route del <Network Address>



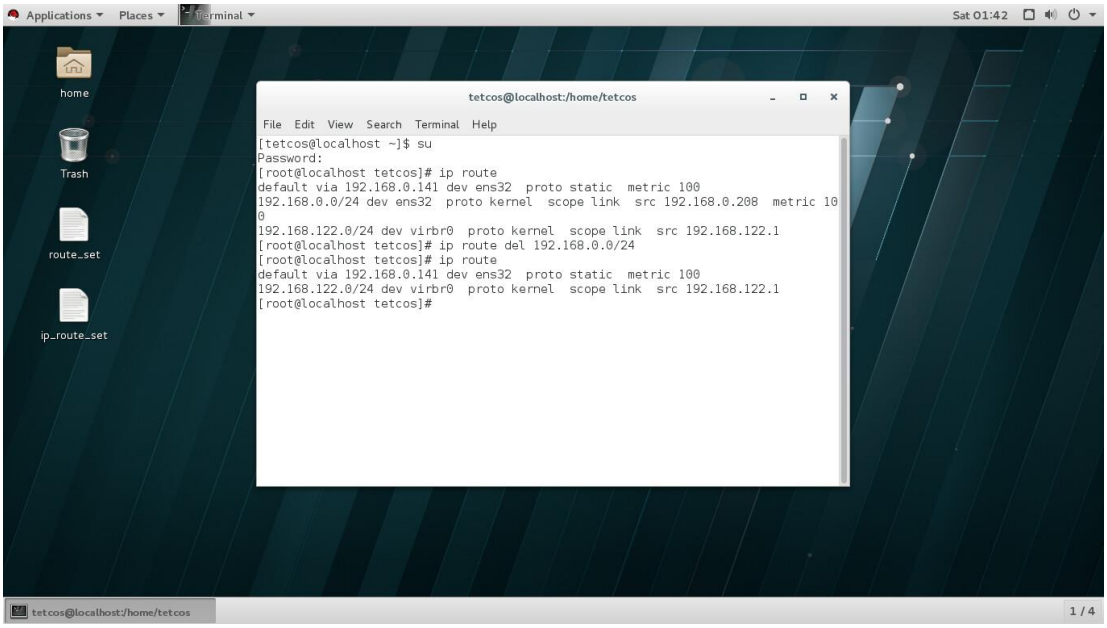
Eg:

ip route del 192.168.0.0/24

Type command

ip route

This is to check if the IP configuration is done.

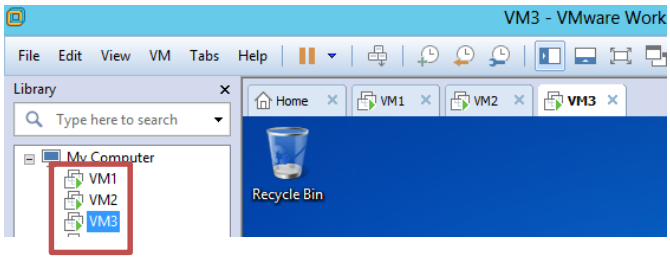


9.2.4 Setting multiple Virtual Machines (VM) to act as Nodes for Emulation

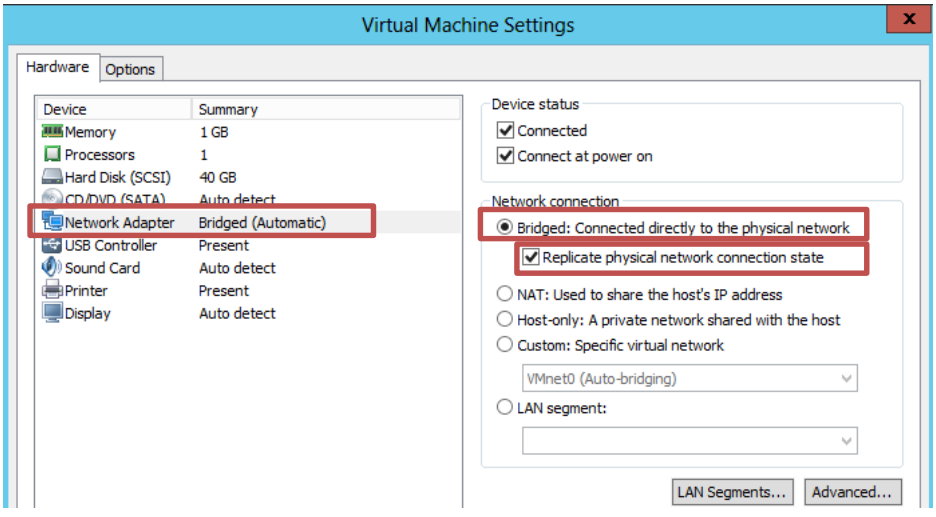
NOTE - VM enabled license are required

9.2.4.1 VMs sharing the same network as the host.

A computer on which one or more virtual machines are running is defined as a Host Machine. Each virtual machine is called a Guest Machine. In this scenario, we have 3 VMs running in a Host Machine – VM1, VM2 and VM3. Users can run NetSim License server in any system connected to the network in which Host Machine is running.



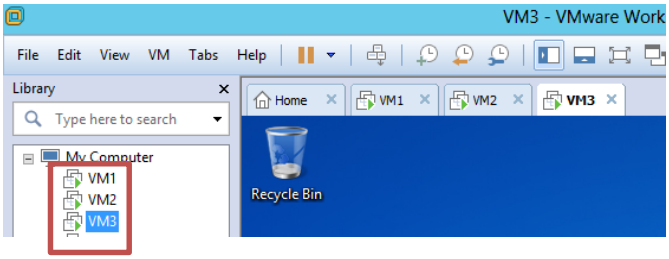
Now right click on each VM and select Settings. Click on Network Adapter, and select "**Bridged: Connected directly to the physical network**". Also enable the "**Replicate Physical network connection state**".



An advantage of this technique is that, if the license server is running in another system, connected to the same network as the original host, then NetSim running in the VM can obtain the licenses.

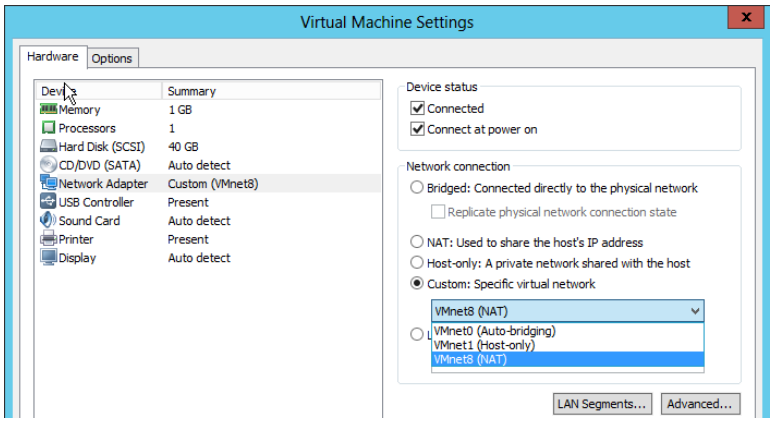
9.2.4.2 VMs sharing a network but insulated from the host network.

A computer on which one or more virtual machines are running is defined as a Host Machine. Each virtual machine is called a Guest Machine. In this scenario, we have 3 VMs running in a Host Machine – VM1, VM2 and VM3. NetSim License server is running in one of these 3 VMs.

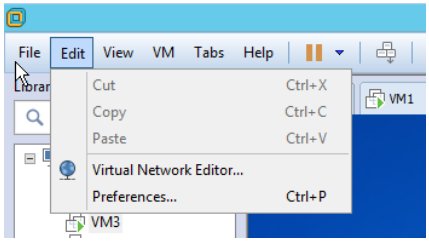


If user needs to create an internal network which is segregated from host network, follow the steps

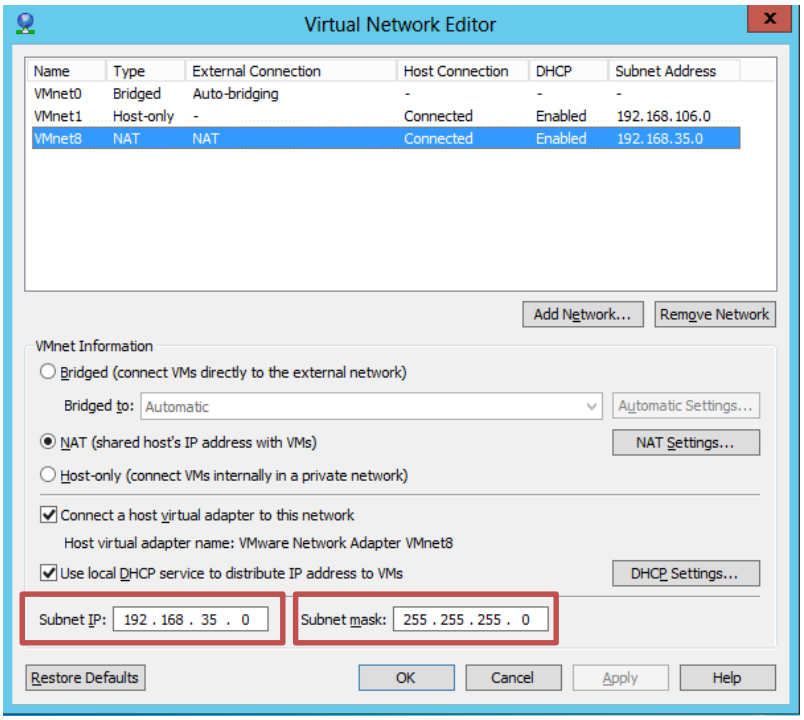
1. Right click on each VM and select Settings
2. Click on Network Adapter, and select “Custom: Specific Virtual network”
3. Select “VMnet8 (NAT)”



By default, a network address is assigned to this segregated network by VMware. To configure this IP address, go to EDIT → Virtual Network Editor



User can modify the Subnet IP and Subnet Mask to suit their own preference.



The disadvantage of this technique is that, if the license server must compulsorily run in the VM for NetSim to obtain the licenses.

9.2.5 Multicasting in NetSim Emulator using JPERF

In computer networking, **multicast** (one-to-many or many-to-many distribution) is group communication where information is addressed to a group of destination computers simultaneously.

Server Setup (receiver):

1. Open command window in administrator mode.
2. Copy Jperf bin path (it would look like C:\Users\Sony\Desktop\jperf-2.0.2\bin) and change the command window directory to the bin path of Jperf shown below:-



3. Enter the following command in the command window

   iperf -s -u -B 224.0.67.67 -i 1

   The options

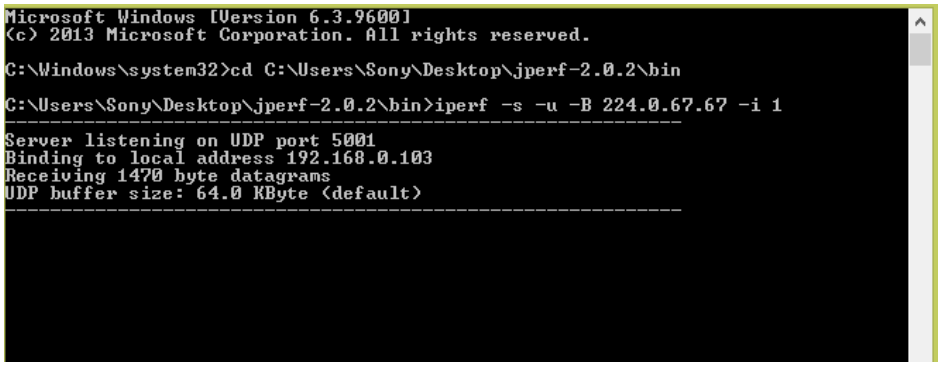
   -s specifies server

   -i specifies report interval

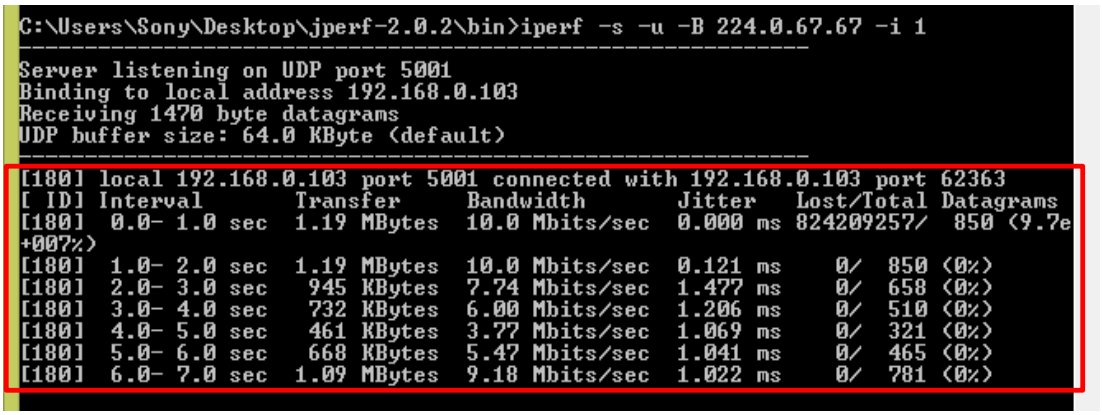
   -u specifies UDP

   -B used to bind and join to a multicast group

4. Now server is listening to the traffic



5. If any client is multicasting traffic, then the server would receive the traffic shown below:-



Client Setup (sender)

1. Open command window in administrator mode
2. Copy Jperf bin path (it would look like C:\Users\Sony\Desktop\jperf-2.0.2\bin) and change the command window directory to Jperf bin path shown below



3. Enter the following command in the command window

   iperf -c 224.0.67.67 -u -b 10m --ttl 5 -i 1 -t 50

   The options

   -c specifies client

   -i specifies report interval

   -t specifies transmit time

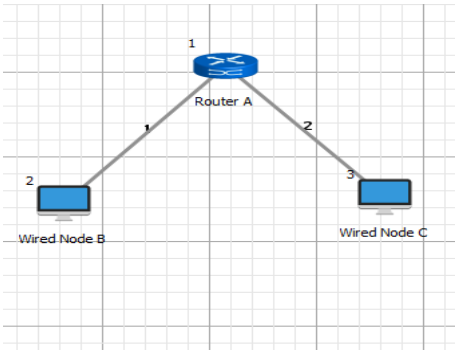
   -u specifies UDP

   --ttl is time-to-live for outgoing multicast packets

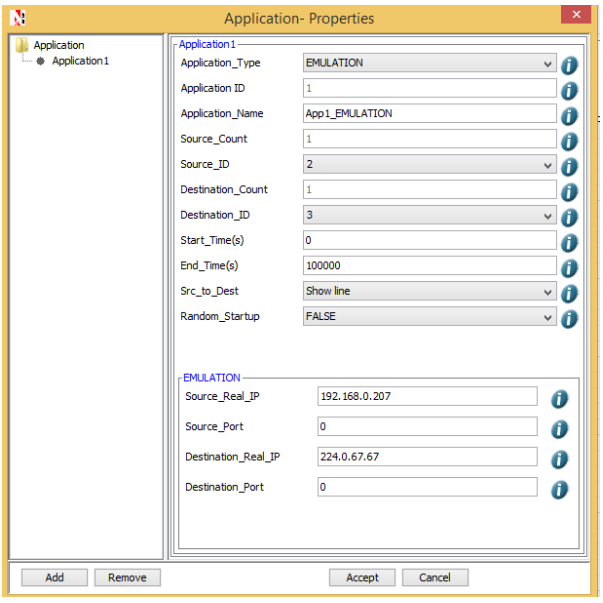
   -b specifies UDP bandwidth. Here it is 10 mbits/sec

```
C:\Users\Sony\Desktop\jperf-2.0.2\bin>iperf -c 224.0.67.67 -u -b 10m --ttl 5 -i
1 -t 50
------------------------------------------------------------
Client connecting to 224.0.67.67, UDP port 5001
Sending 1470 byte datagrams
Setting multicast TTL to 5
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[196] local 192.168.0.103 port 52432 connected with 224.0.67.67 port 5001
[ ID] Interval       Transfer     Bandwidth
[196]  0.0- 1.0 sec  1.19 MBytes  10.0 Mbits/sec
[196]  1.0- 2.0 sec  1.19 MBytes  10.0 Mbits/sec
[196]  2.0- 3.0 sec   805 KBytes  6.60 Mbits/sec
[196]  3.0- 4.0 sec   801 KBytes  6.56 Mbits/sec
[196]  4.0- 5.0 sec   490 KBytes  4.01 Mbits/sec
[196]  5.0- 6.0 sec   504 KBytes  4.13 Mbits/sec
[196]  6.0- 7.0 sec  1.05 MBytes  8.80 Mbits/sec
[196]  7.0- 8.0 sec   741 KBytes  6.07 Mbits/sec
```

4. We can observe the traffic generated by client in above figure.
5. Now open NetSim in Administrator mode and create a simple scenario with one router and 2 wired nodes in internetworks shown below



6. Open application properties, create an EMULATION application and set client ip address in Source_Real_Ip text box and 224.0.67.67 (multicast ip address) in Destination_Real_Ip text box shown below:



7. Run simulation for 50 secs and observe the jitter values in the server.
8. You will find the variation in jitter values with and without running simulation in NetSim
9. And also change the speed, error rates and delay in the link properties and observe the variation
10. If you wish to run more than 1 server, then setup another server by following the steps explained above

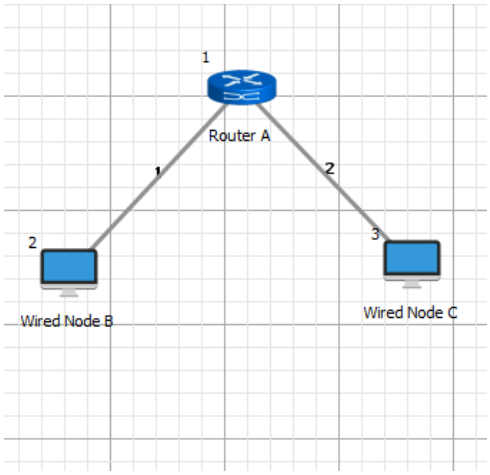
Note: Users need to run simulation (emulation) and the jperf client simultaneously

9.3 Emulation examples in NetSim

9.3.1 Example Application 1 – PING (One way Communication)

9.3.1.1 Steps at Emulation Server:

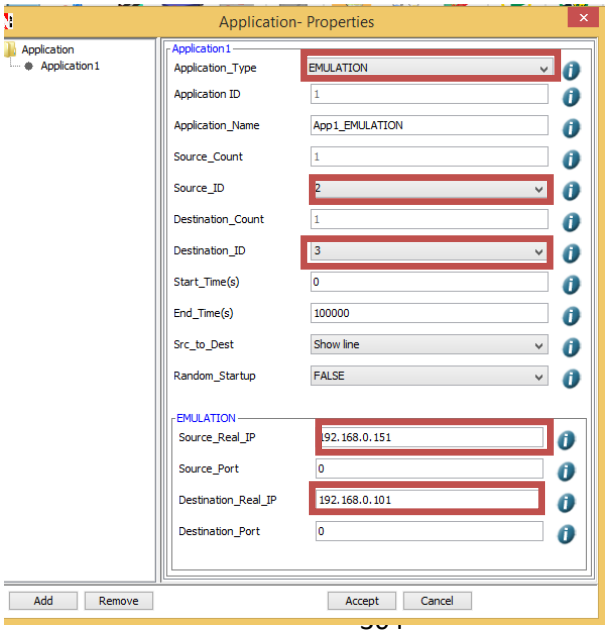
I. Run NetSim in Administrative Mode and create a basic network Scenario in any stack based protocol (Any network except Legacy Networks, Wireless Sensor Network, Zigbee Network and Cellular Network) in NetSim. Screenshot of a sample scenario in Internetworks is shown below



II. Go to Properties of Link1 and Link2 and set Uplink and Downlink Delay to 5000. Click and drop the Application. Right click Application → select Properties.

III. In the Application Type select Emulation.

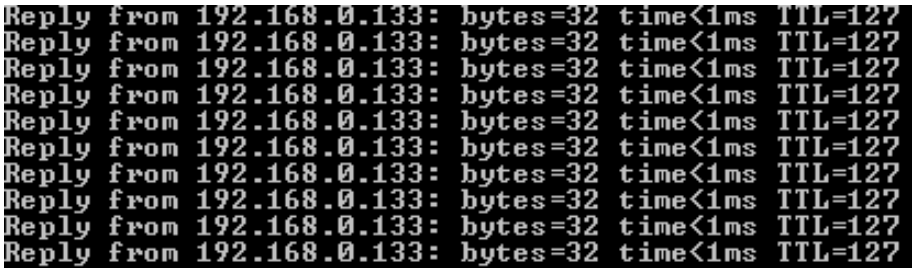
IV. Select Source and Destination ID according to the network scenario and change the Source and Destination IP address according to the IP address of the real system.



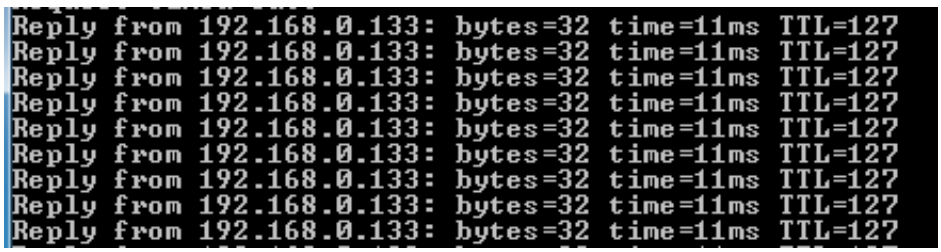
V. Provide the Simulation Time as how long you want the Emulation to be performed. Make sure client system(s) are ready and then click Run Simulation.

9.3.1.2 Steps at Source PC:

1. Before running simulation, start pinging the Destination from Source using command "ping <Destination_IP> –t" and note down the time duration.



2. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client"**.
3. Perform the steps at Emulation Server as provided and simulate. During simulation, ping the destination system. You will notice that the present time duration is higher than the earlier ping results. This is because the network created in NetSim has link propagation delay. Also Wireshark (if installed) will automatically start capturing the packets as soon as Emulation Server starts simulation.



(NOTE: In case if no ping messages can be sent from source to destination, disable windows firewall and try again.)

4. The impact of the link propagation delay in NetSim Emulator is seen on a real packet.

9.3.2 Example Application 1 – PING (Two way Communication)

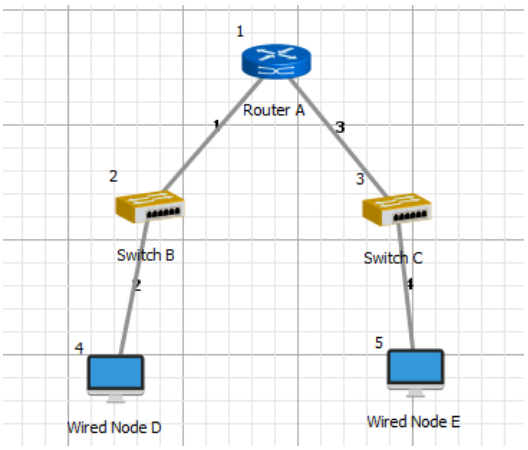
In PING (Two way communication), almost all the steps are same as PING (One way communication), except that in NetSim Emulation server there will be two application instead of one. One Application will be directed from Source to Destination node, while the other application will be directed from Destination to Source node.

The difference caused in the network behavior is that in the first case (PING -One way communication), the PING reply packets were not routed via NetSim Emulator. But in the second case (PING -Two way communication), the PING reply packets will be routed via NetSim Emulator, thereby the total delay will be approximately 21millisecond.

9.3.3 Example Application 2 – Video (One way Communication)

9.3.3.1 Steps at NetSim Emulation Server:

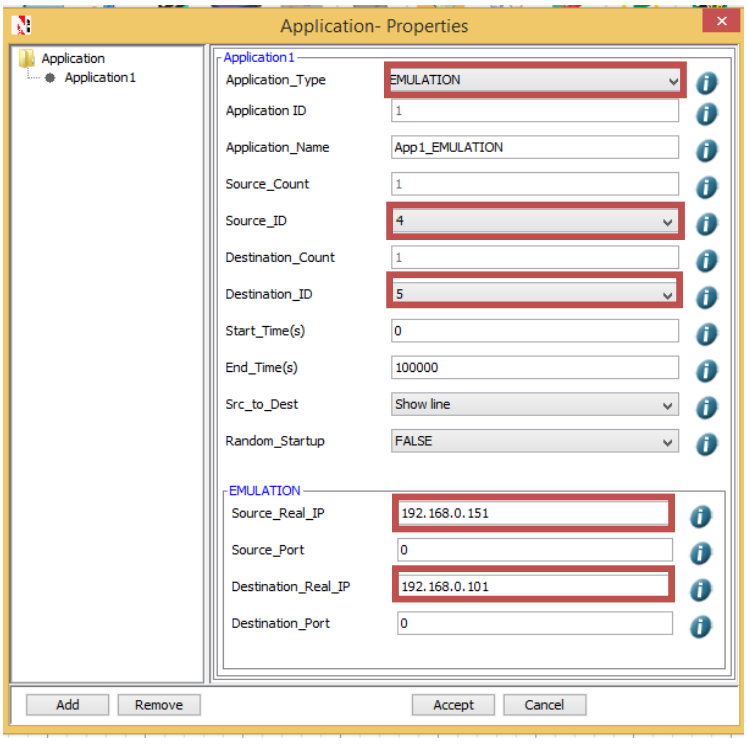
I. Run NetSim in Administrative Mode and create a basic network Scenario in any stack based protocol (Any network except Legacy Networks, Wireless Sensor Network, Zigbee Network and Cellular Network) in NetSim. Screenshot of a sample scenario in Internetworks is shown below



II. Click and drop the Application. Right click Application → select Properties.

III. In the Application Type select Emulation.

IV. Select Source and Destination ID according to the network scenario and change the Source and Destination IP address according to the IP Address of the real system and click accept.



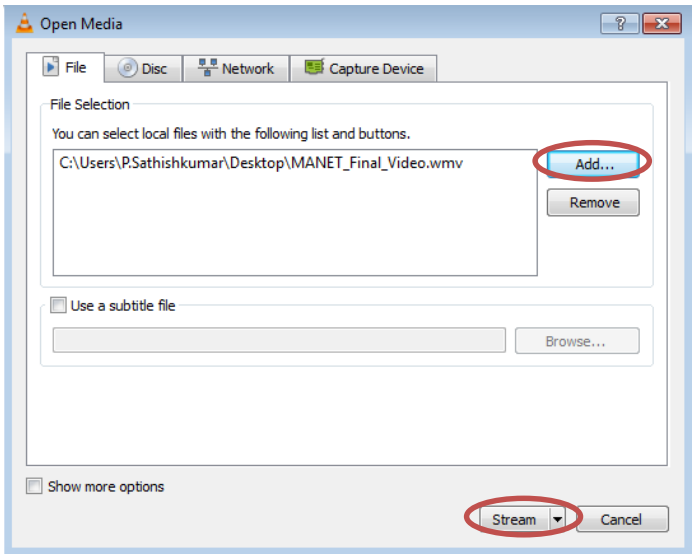
V. Provide the Simulation Time as how long you want the Emulation to be performed. Make sure client system(s) are ready and then click Run Simulation.

During Simulation you will notice a change in the quality of the video being played in the destination PC. This is because the network created in NetSim has errors / delays etc in the links. The impact of this loss / jitter / delay etc in NetSim Emulator is seen on a real video stream.

9.3.3.2 Steps at Source PC:

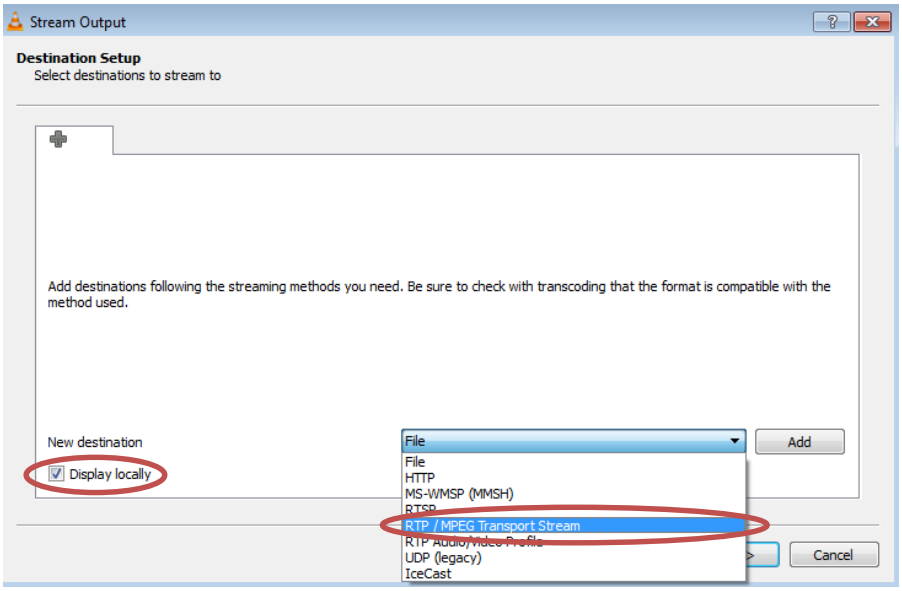
1. Follow steps as provided before in "**Running Emulation via GUI → Setting up the NetSim Client"**. Then open VLC Media player → Click Media menu → Select Stream Option.

2. Click add button then select the video which you want to play

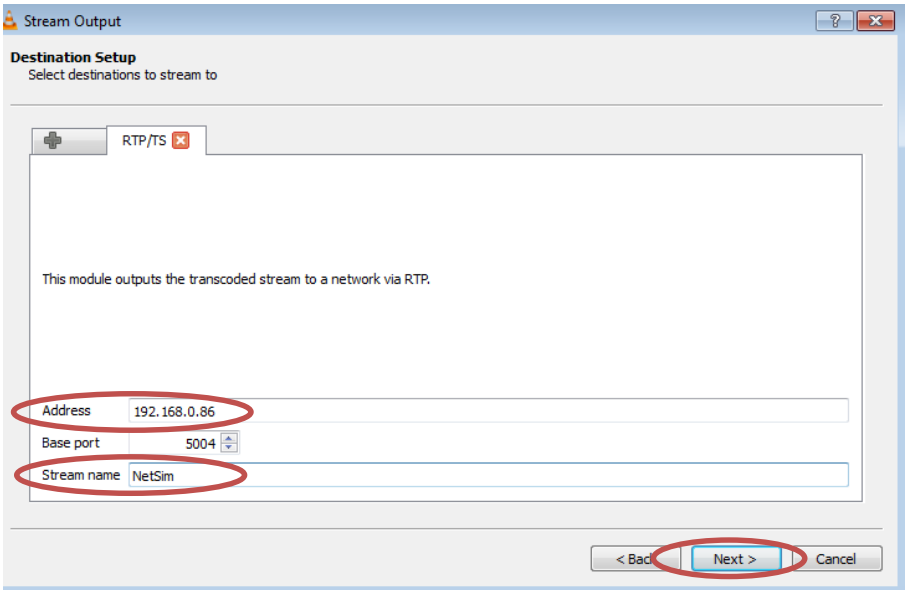


3. Click on Stream Option. Then click next button

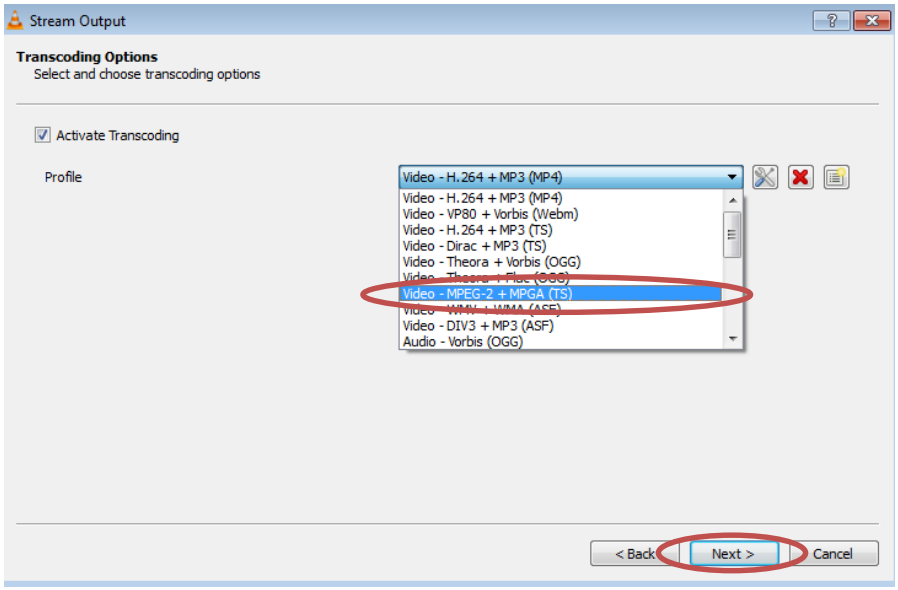
4. Enable the display locally checkbox. Then select the RTP / MPEG Transport Stream from the drop down list as shown in the below screen shot



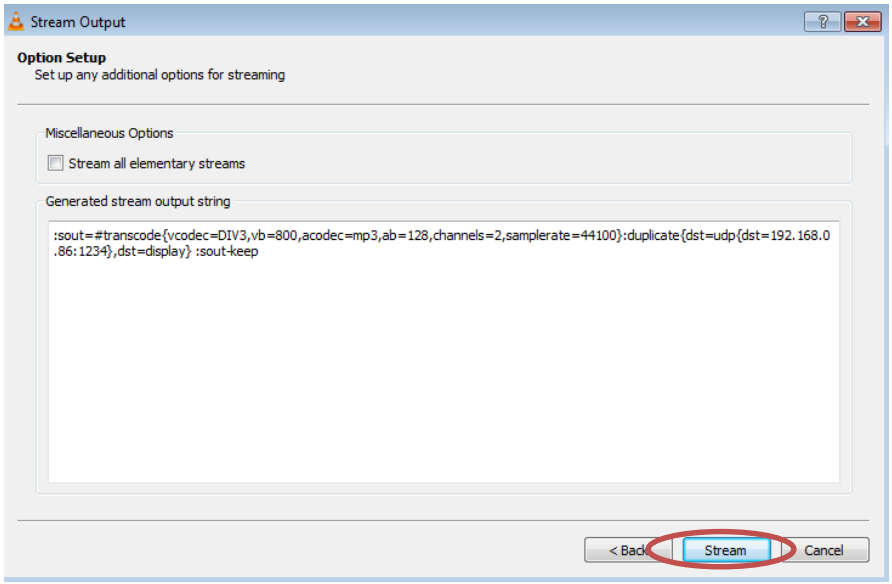
5. Click on Add Button. Then enter the Destination IP address in the Address field and enter a stream name (user defined) and click next button.



6. Select Video –MPEG-2 + MPGA (TS) option from the drop down list as shown in the below screen shot. Then click next button

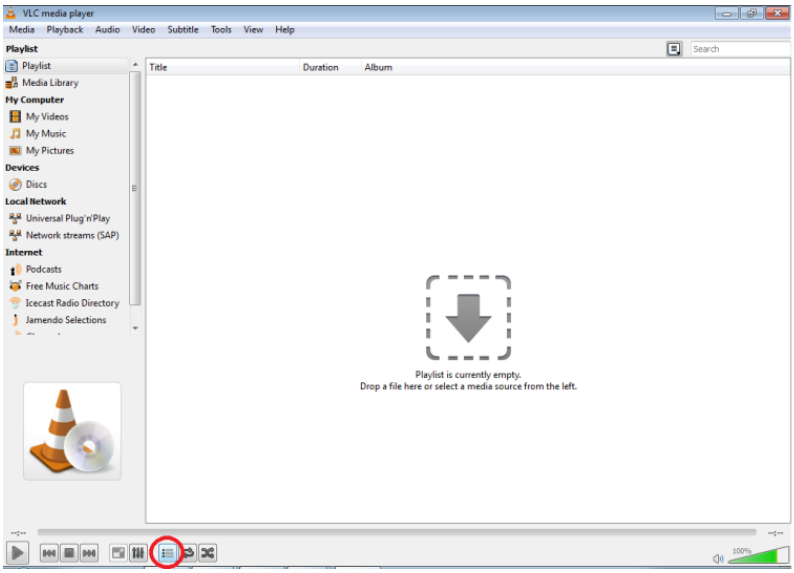


7. Perform all the steps at Emulation Server and then click on Stream button. Also Wireshark (if installed) will automatically start capturing the packets as soon as Emulation Server starts simulation.



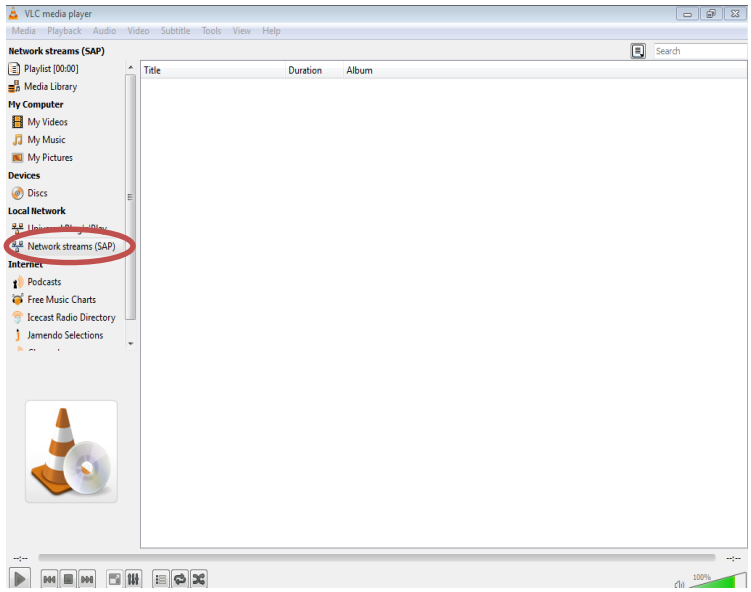
9.3.3.3 Steps at Destination PC:

1. Follow steps as provided before in "**Running Emulation via GUI–Setting up the NetSim Client"**. After performing all the steps at Source PC and NetSim Emulation Server, open VLC Media Player→ Click on Toggle Playlist icon as shown in the below screenshot.

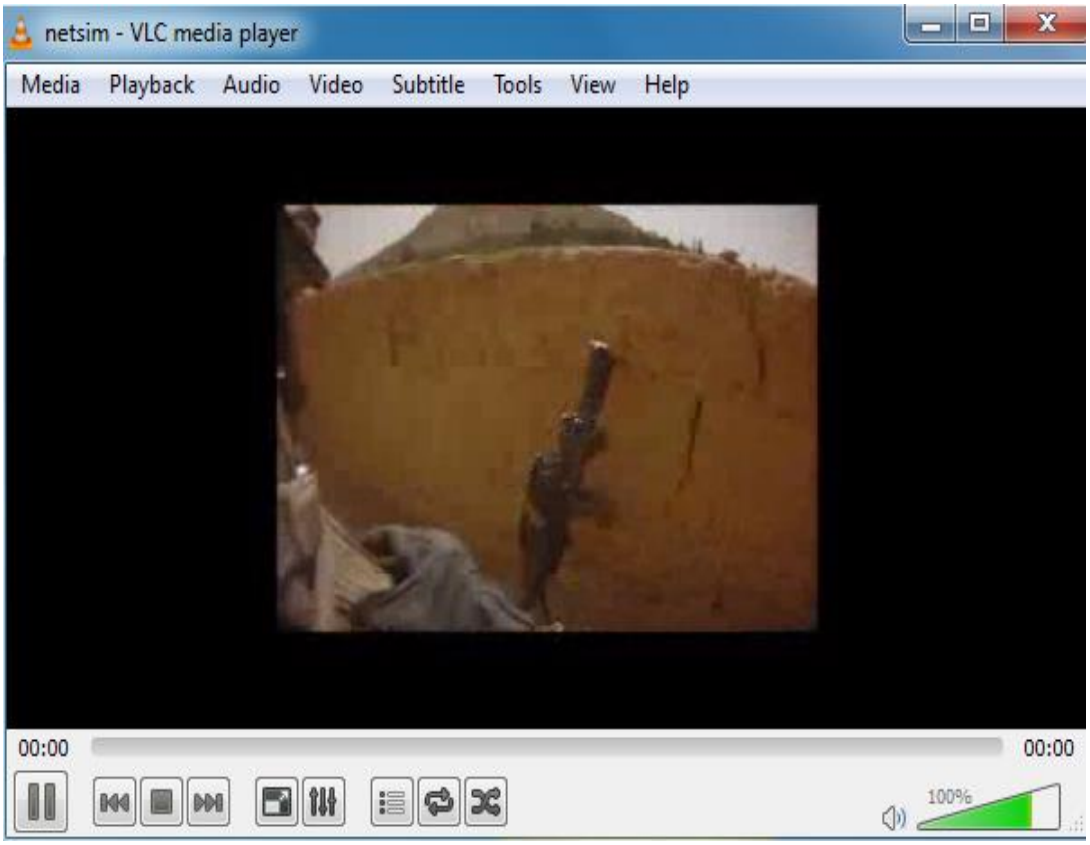


Toggle button is circled in red at the bottom of the screen shot

2. Double click on Network Stream (SAP) under local network. Then right click and play on the stream name that appears on the screen.



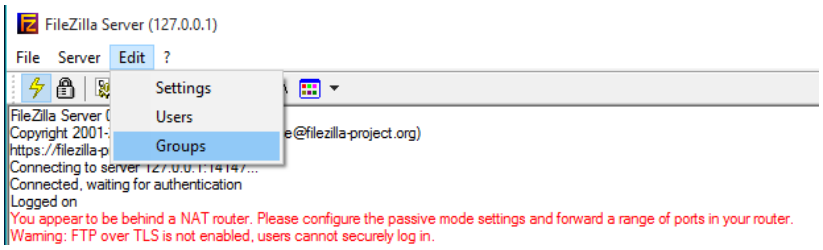
3. In the streamed video, you will notice a change in the quality of the video being played in the destination PC. Also Wireshark (if installed) will automatically start capturing the packets as soon as Emulation Server starts simulation.


netsim - VLC media player
Media
Playback
Audio
Video
Subtitle
Tools
View
Help
00:00
00:00
100%

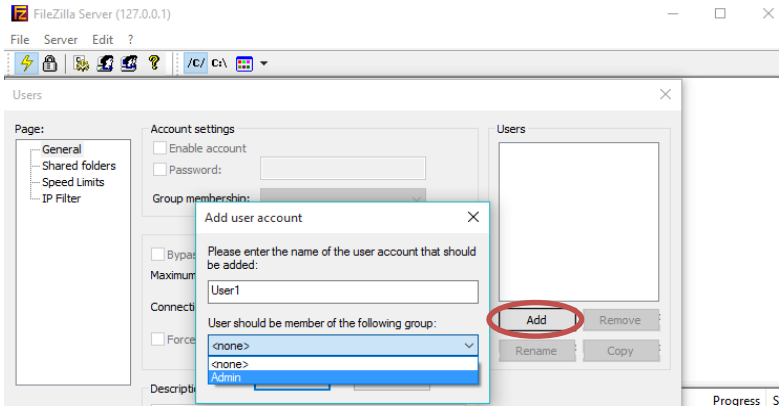
9.3.4 Example Application 3 – File Transfer using FileZilla (One way Communication)

9.3.4.1 Steps at Destination PC:

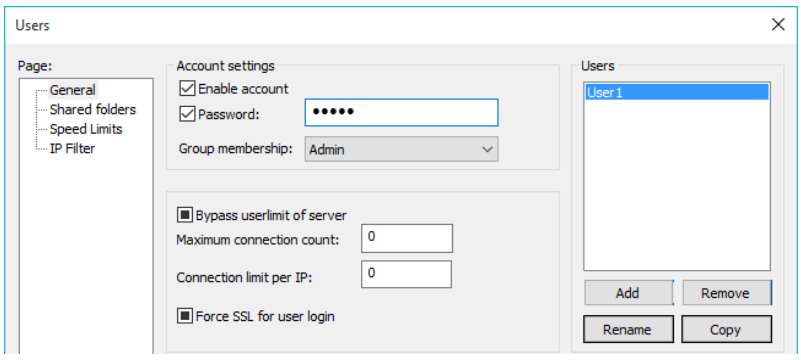
1. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client"**. Run FileZilla Server software. Create a group by going to Edit → Groups → Select "General" under Page: → Click Add in Groups → Give Any Name (Ex: Admin) and click ok.



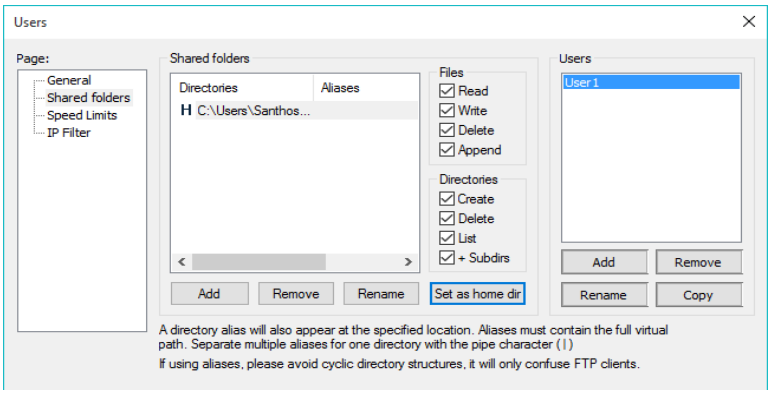
2. Go to Edit → User → General → Click Add in User → Give Any Name (Ex: User1) and Select Group what you given in Group Setting (In this case, we provide "Admin") and click ok.



3. In Account Setting, select Enable account and set password and click ok.



4. Go to Shared folder → Add Folder to share (EX: FTP_FILES from Desktop) → Select all the Files and Directories Permissions and set that folder as Home Directory by selecting "Set as Home Dir". Click Ok.

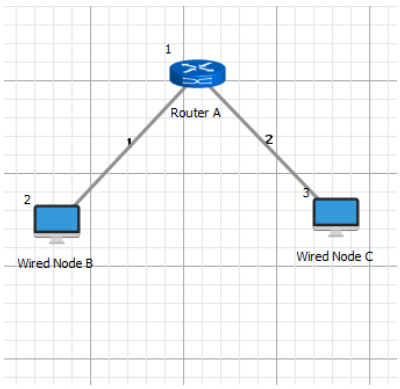


9.3.4.2 Steps at Source PC:

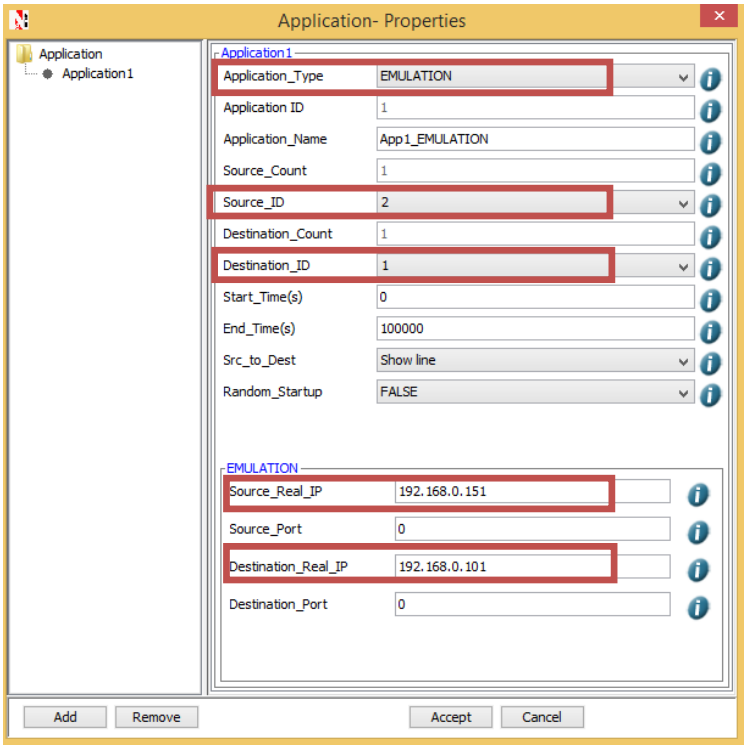
1. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client"**. Run FileZilla Client software.
2. Enter the Host Name(Server System ip (EX: 192.168.0.133)) and Give the User, Password that we created in Server side and give Port No = 21. Run Emulation server and click Quick Connect. Drag and drop files from Local Site to Remote Site.

9.3.4.3 Steps at NetSim Emulation Server:

1. Run NetSim in Administrative Mode and create a basic network Scenario in any stack based protocol (Any network except Legacy Networks) in NetSim. A sample scenario in Internetworks is performed as shown with link speed set to 1 Mbps.



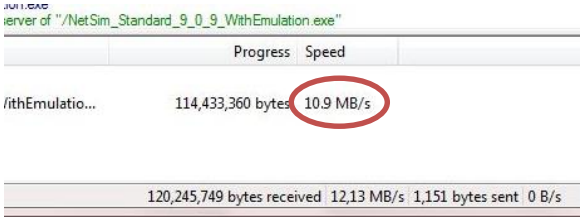
2. Click and drop the Application. Right click Application → select Properties.
3. In the Application Type select Emulation.
4. Select Source and Destination ID according to the network scenario and change the Source and Destination IP address according to the IP Address of the real system and click accept.



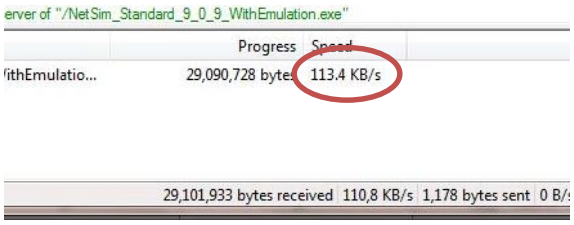
5. Provide the Simulation Time as how long you want the Emulation to be performed. Make sure client system(s) are ready and then click Run Simulation.

9.3.4.4 Results:

Transfer speed from client without emulation:



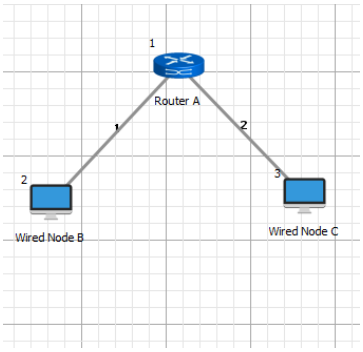
Transfer speed from client with emulation:



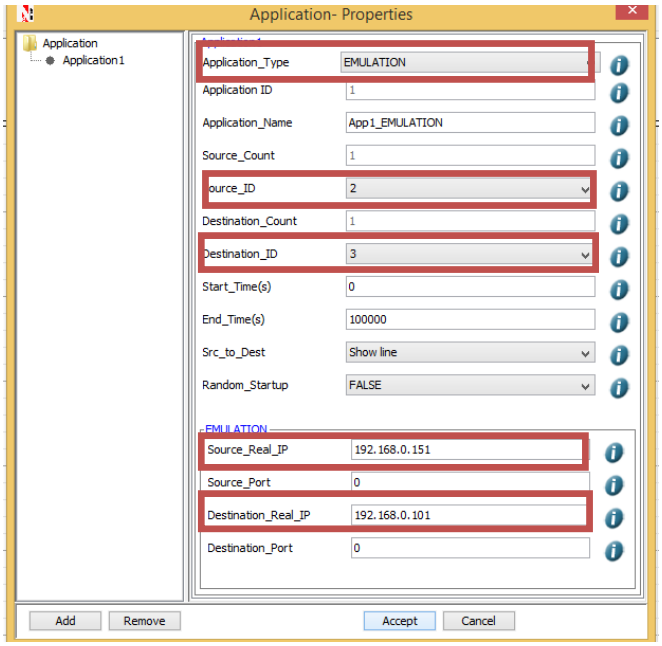
9.3.5 Example Application 4 –Skype (Two way Communication)

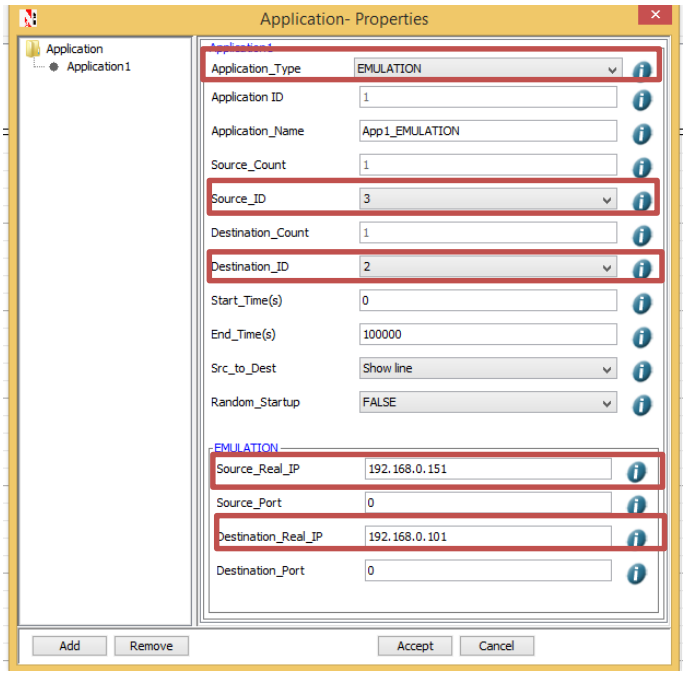
9.3.5.1 Steps at NetSim Emulation Server:

1. Run NetSim in Administrative Mode and create a basic network Scenario in any stack based protocol (Any network except Legacy Networks, Wireless Sensor Network, Zigbee Network and Cellular Network) in NetSim. Screenshot of a sample scenario in Internetworks is shown below.



2. Click and drop Application button. Right click Application → select Properties. As it is two way communication, add and create two applications.
3. In both the Application Type select Emulation.
4. In one Application, select Source ID and Destination ID according to the network scenario and change the Source and Destination IP address according to the IP Address of the real system. In the second application, set the opposite of first application, i.e. Source ID and IP address will be exchanged with Destination ID and IP address. (Refer the IP settings in the screen-shot to get a clear picture)





5. Provide the Simulation Time as how long you want the Emulation to be performed. Make sure client system(s) are ready and then click Run Simulation.

9.3.5.2 Steps at Source PC:

1. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client"**.

2. Run Skype and make a call to the destination system (Make sure that Skype is running in Destination PC).

3. Wireshark (if installed) will automatically start capturing the packets as soon as Emulation Server starts simulation.

9.3.5.3 Steps at Destination PC:

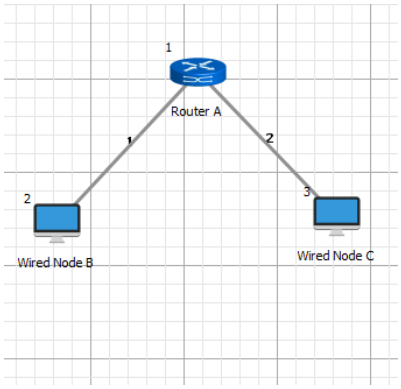
1. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client"**. After performing all the steps at Source PC and NetSim Emulation Server, open Skype.

2. Wireshark (if installed) will automatically start capturing the packets as soon as Emulation Server starts simulation.

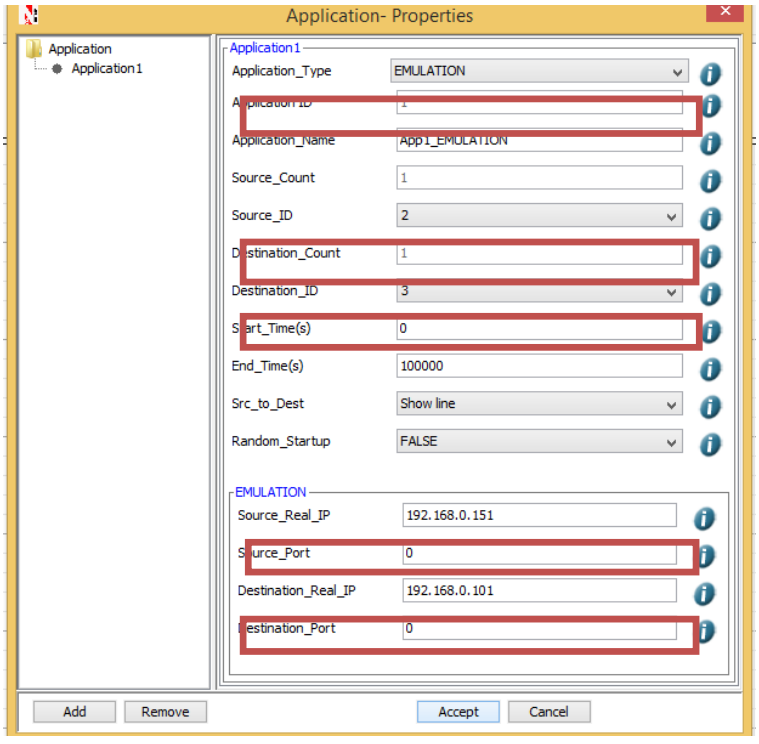
9.3.6 Example Application 5 – JPerf Network performance measurement graphical tool (One way Communication)

9.3.6.1 Steps at NetSim Emulation Server:

1. Run NetSim in Administrative Mode and create a basic network Scenario in any stack based protocol (Any network except Legacy Networks, Wireless Sensor Network, Zigbee Network and Cellular Network) in NetSim. Screenshot of a sample scenario in Internetworks is shown below



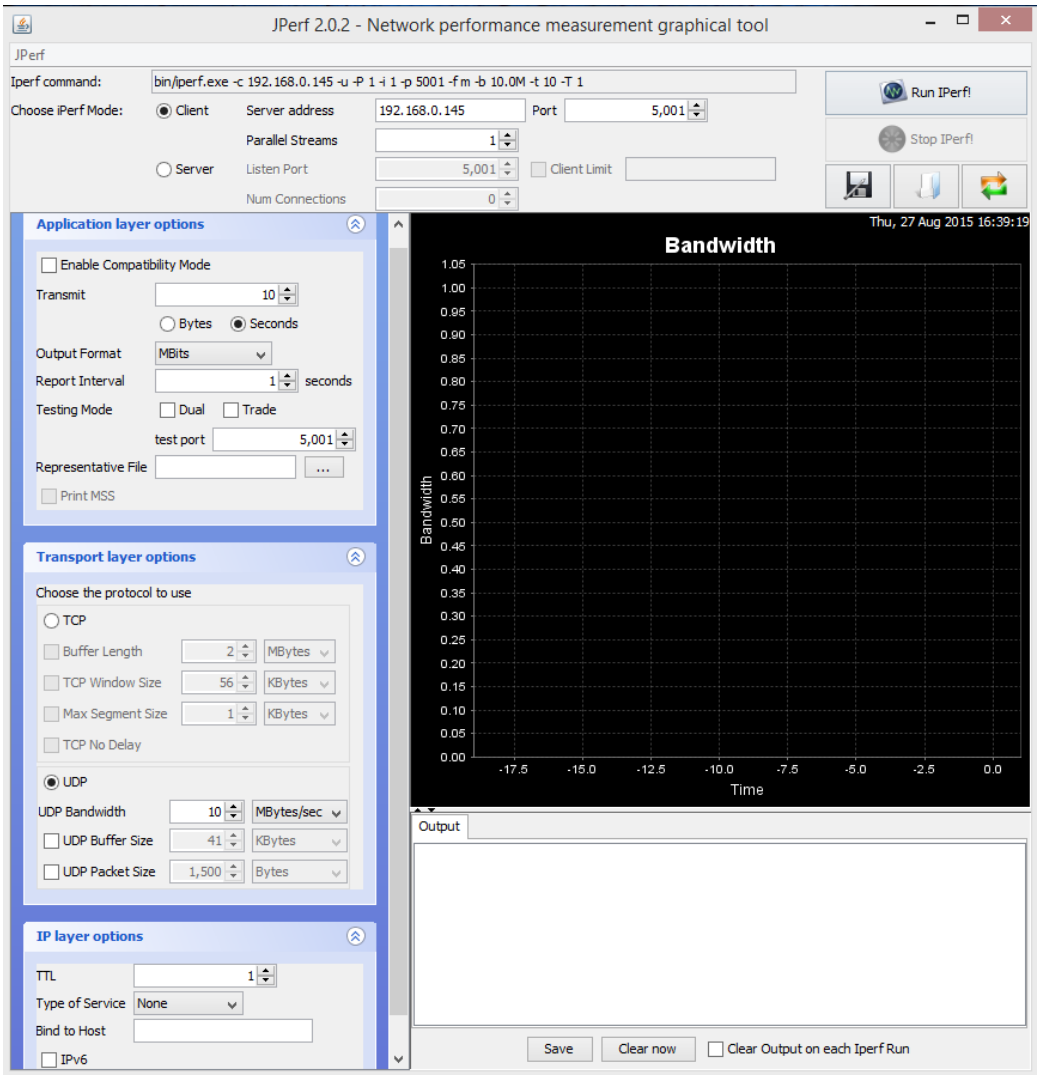
2. Click and drop the Application. Right click Application → select Properties.
3. In the Application Type select Emulation.
4. Select Source and Destination ID according to the network scenario and change the Source and Destination IP address according to the IP Address of the real system and click accept.



5. Provide the Simulation Time as how long you want the Emulation to be performed. Make sure client system(s) are ready and then click Run Simulation.

9.3.6.2 Steps at Source PC:

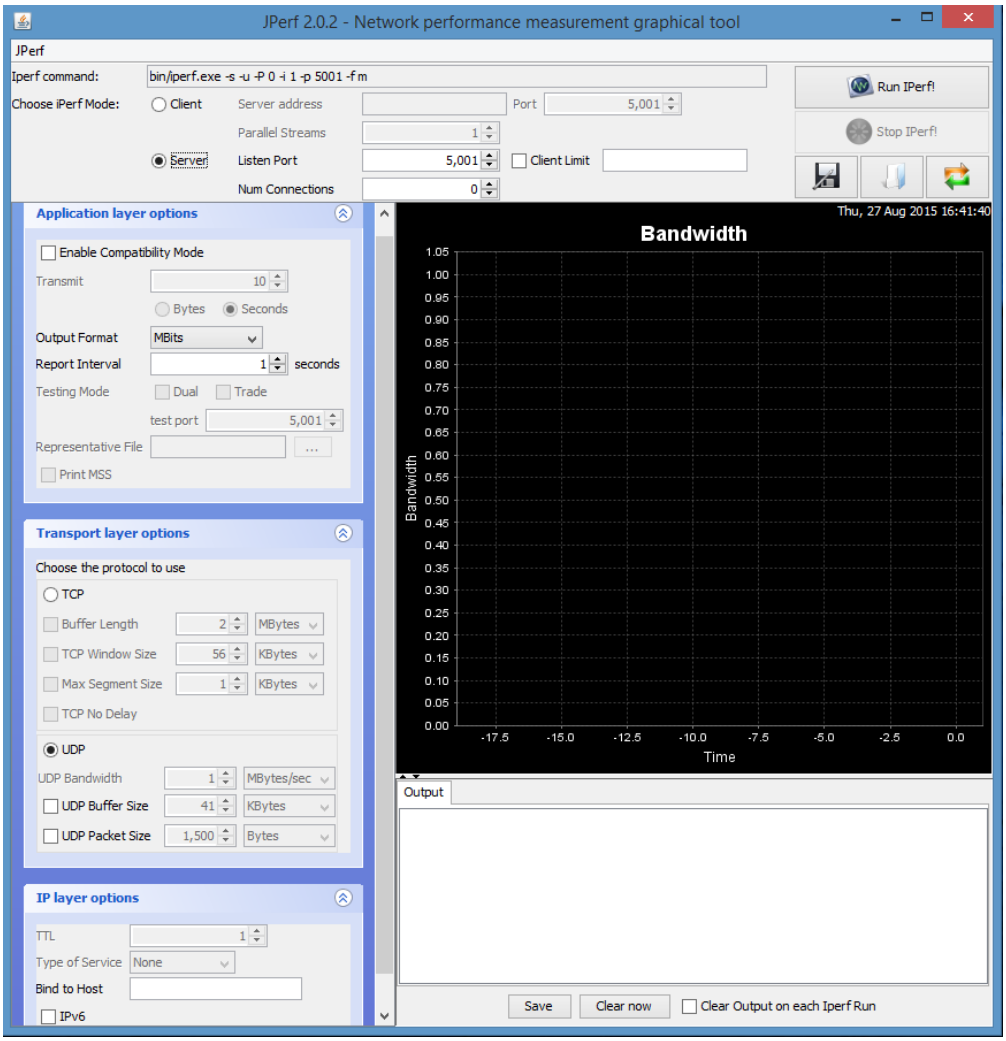
1. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client"**. Run JPerf and select Client and set Server Address as **192.168.0.145**. User can edit the Application Layer options, Transport Layer options and IP Layer options depending on the type of data they want to transmit in the network.



2. Do not click "**Run IPerf"** until all the steps at NetSim Emulation Server are done. Also Wireshark (if installed) will automatically start capturing the packets as soon as Emulation Server starts simulation.

9.3.6.3 Steps at Destination PC:

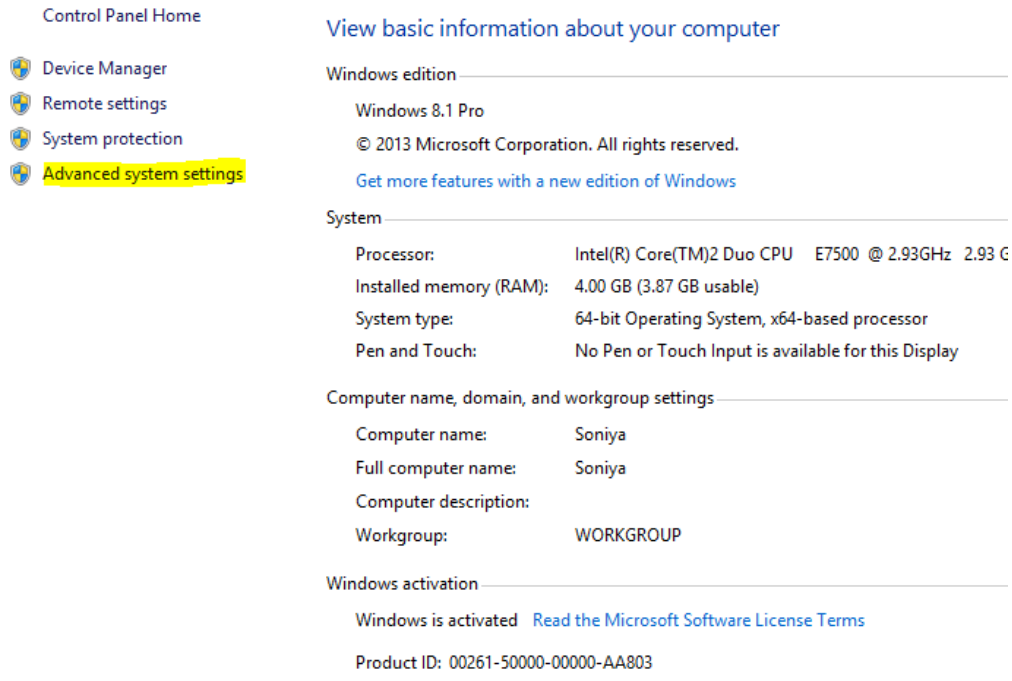
1. Follow steps as provided before in "**Emulation Set-up: Setting up the NetSim Client**". Run JPerf and select Server.

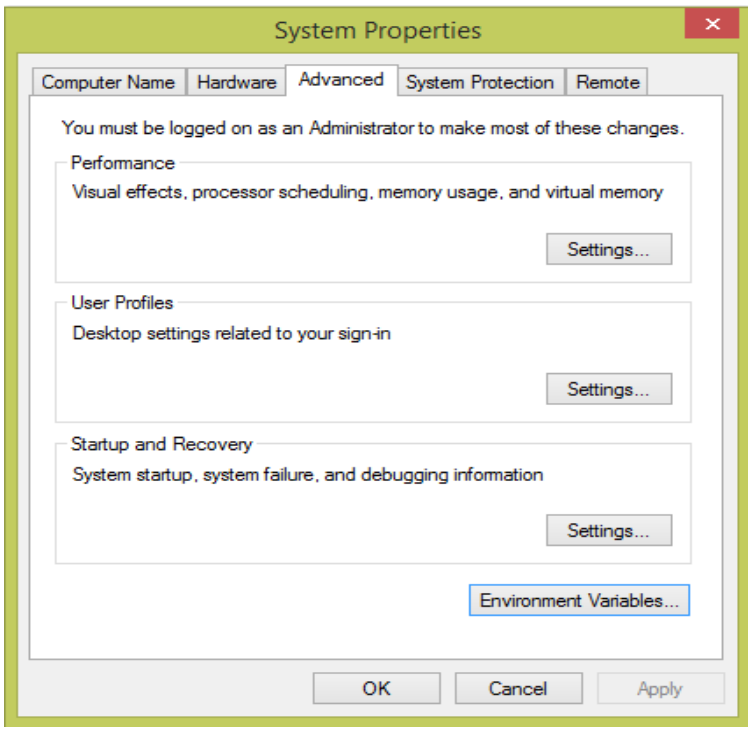


2. Click on "**Run IPerf**" after the Source PC starts running JPerf.

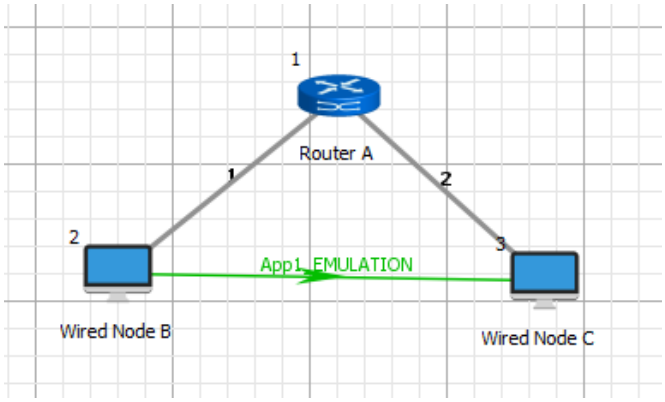
9.3.7 Steps to follow for providing pcap file as input to NetSim Emulator

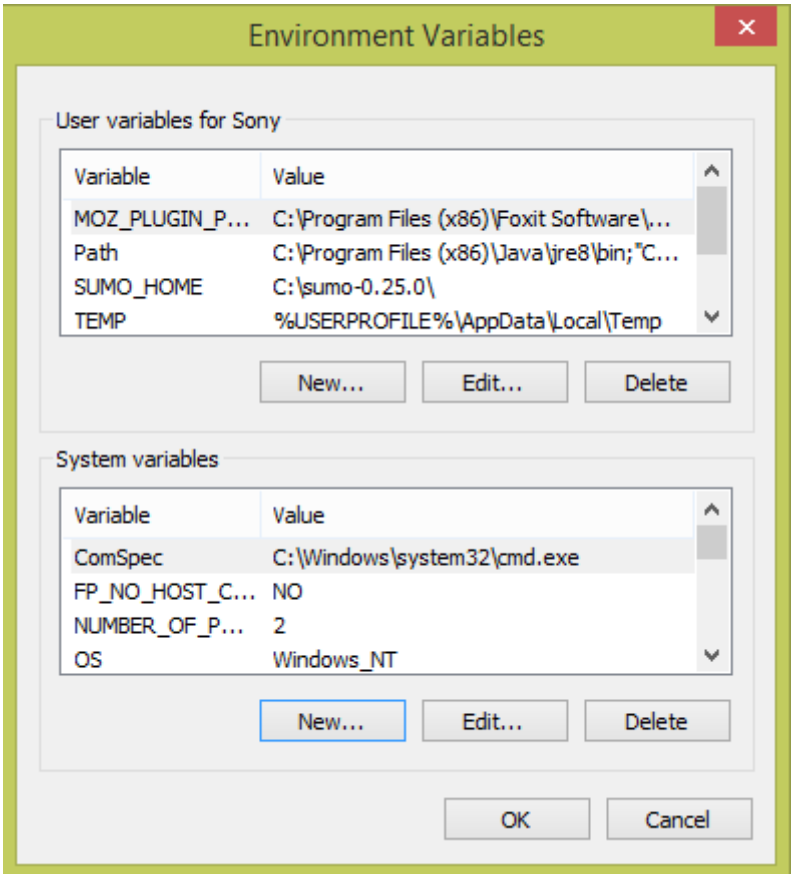
1. Create a pcap file using wireshark
2. Go to Computer->Properties->Advanced system settings->Environment variables

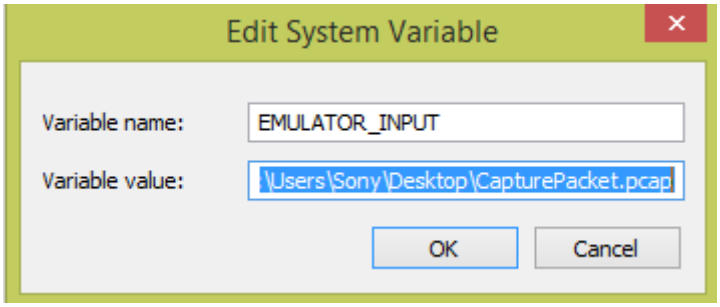




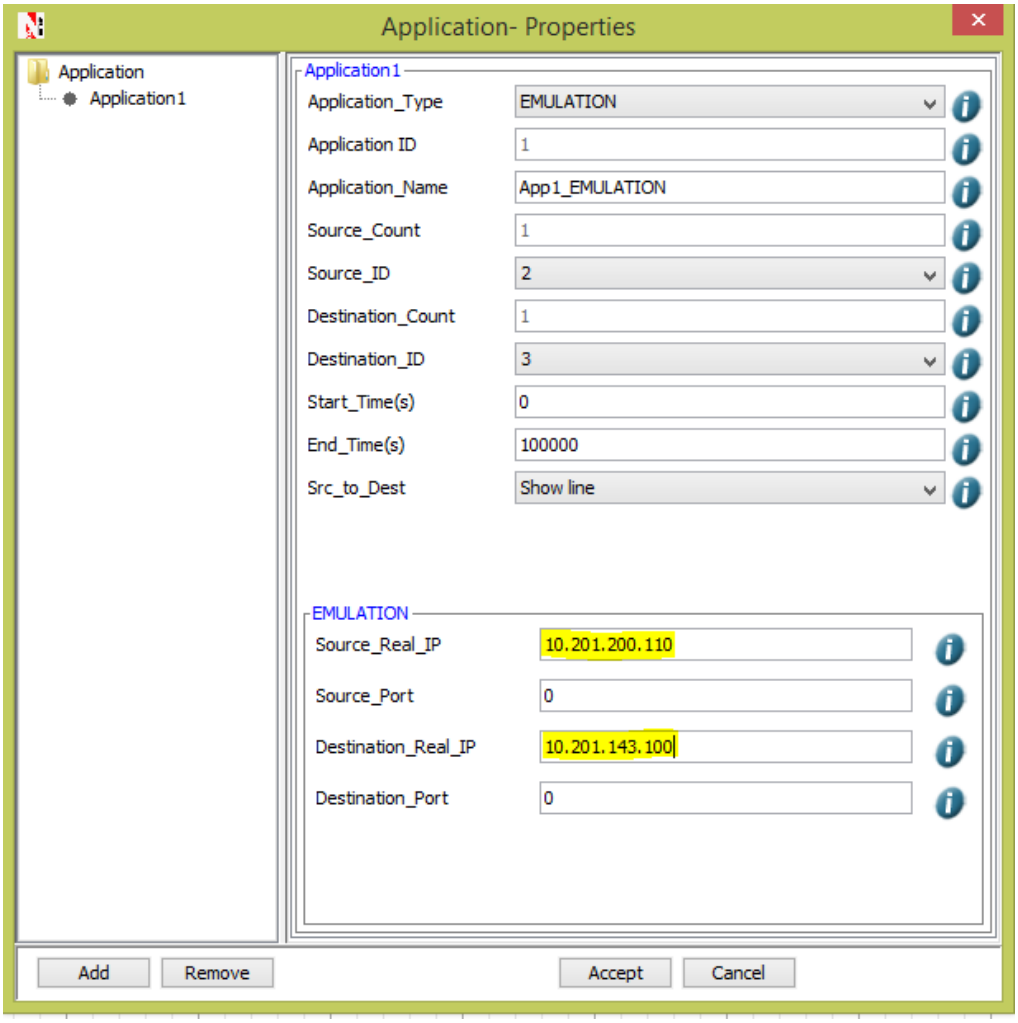
3. In System variables pane, select new and give EMULATOR_INPUT in the variable name text box and give the path where pcap file is present in the variable value text box (Ex: C:\Users\Sony\Desktop\CapturePacket.pcap) and then click on OK.

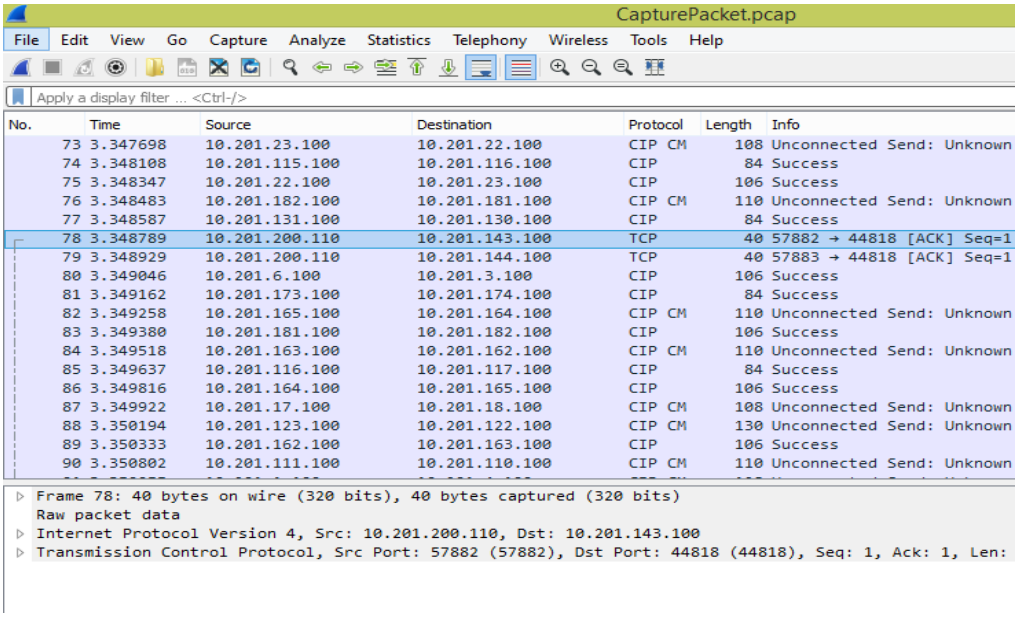




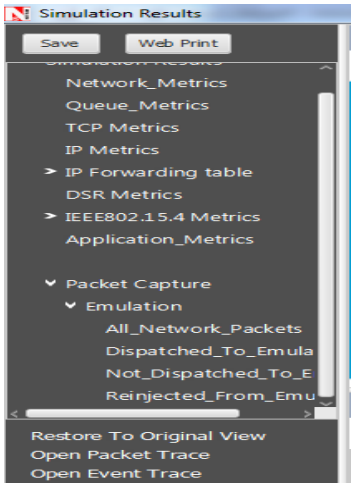


4. Run NetSim in administrator mode for Emulator application.
5. Now create a simple scenario in NetSim. For example create a scenario in Internetworks with 1 router and 2 wired nodes.
6. Create an Emulator application by giving the real source and destination IP's present in the pcap file.





7. We can give more than one application also.
8. NetSim Emulator will read the packets from pcap file as per the source and destination that we are giving in the application properties.
9. After simulation, NetSim provides Packet Capture Metrics. Here users can observe 4 different types of packets
    a. All Network Packets (contains all the Packets present in the pcap file)
    b. Dispatched to Emulator (contains only the packets that are going through the Network Emulator)
    c. Non Dispatched To Emulator (contains packets that are not going to the Network Emulator)
    d. Reinject from Emulator (contains the packets that are coming out from the Network Emulator).



10. Open Dispatch to Emulator packets, it contains only the packets whose source and destination IP addresses match with the source and destination IP addresses that we have configured in the application properties.

9.4 Working of an Emulation Application in NetSim:

Note: The following explanation is provided assuming that you have performed all necessary configuration required to divert network traffic via the system running NetSim Emulator. (This is explained in section 9 of the User Manual).

The following parameters are specific to Emulation Application in NetSim:

1. Source_Real_IP
2. Source_Port
3. Destination_Real_IP
4. Destination_Port

Unlike Simulation, if users want to connect real devices running live applications to the simulator, then Emulation component is required. The Emulation Application in the traffic generator allows users to pump in real traffic into the Simulator.

The real application is mapped using the source and destination IP addresses that we set in the Emulation Application.

Various combination of Emulation Parameters are as follows:

1. Device Specific Emulation:

Example 1:

SOURCE_REAL_IP = 192.168.0.151

SOURCE_PORT = 0

DESTINATION_REAL_IP = 192.168.0.202

DESTINATION_PORT = 0

Dispatches all packets with the source real IP 192.168.0.151 and destination real IP as 192.168.0.202, into the Simulator.

Example 2:

SOURCE_REAL_IP = 192.168.0.151

SOURCE_PORT = 0

DESTINATION_REAL_IP = 0.0.0.0

DESTINATION_PORT = 0

Dispatches all packets from source real IP 192.168.0.151 regardless of whatever is the destination real IP, into the Simulator.

Example 3:

SOURCE_REAL_IP = 0.0.0.0

SOURCE_PORT = 0

DESTINATION_REAL_IP = 192.168.0.202

DESTINATION_PORT = 0

Dispatches all packets to destination real IP 192.168.0.202 regardless of whatever is the source real IP, into the Simulator.

Example 4:

SOURCE_REAL_IP = 0.0.0.0

SOURCE_PORT = 0

DESTINATION_REAL_IP = 0.0.0.0

DESTINATION_PORT = 0

Dispatches all packets that are reaching the Emulator Device regardless of whatever is the source or destination, into the Simulator.

2. Application Specific Emulation

Example 1:

SOURCE_REAL_IP = 192.168.0.151

SOURCE_PORT = 5004

DESTINATION_REAL_IP = 192.168.0.202

DESTINATION_PORT = 6245

Dispatches all packets with the source real IP 192.168.0.151, source Port No 5004, destination real IP as 192.168.0.202 and destination Port No 6245 into the Simulator.

Emulation Specific Metrics:

On running an Emulation Application Users can optionally obtain the following log files which are WIRESHARK compatible .pcap files:

All_Network_Packets - Log of all network packets flowing via the system running NetSim Emulator.

Dispatched_To_Emulator - Log of network packets for which Emulation Application is configured in NetSim.

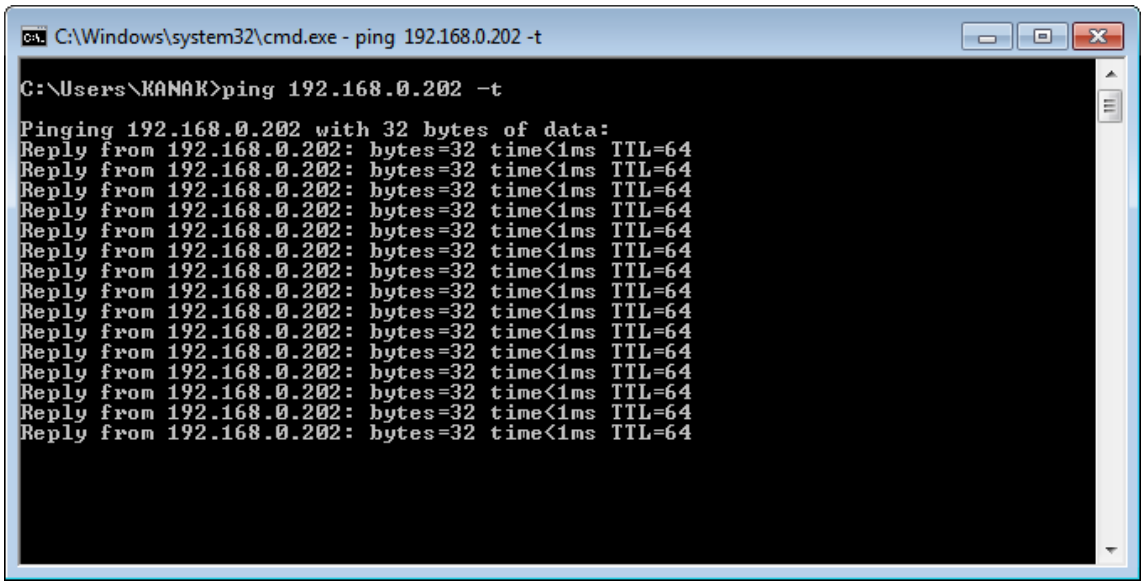
Reinjected_To_Emulator - Log of network packets that has successfully reached the virtual destination node in NetSim Simulator.

Non_Dispatched_To_Emulator - Log of network packets for which we have not configured any Emulation Application.

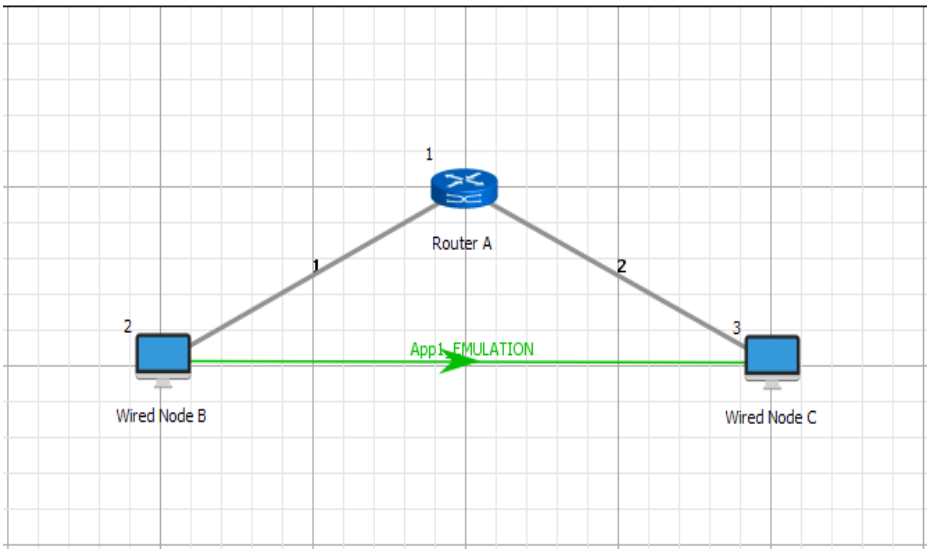
9.4.1 When I run ping in emulation why is it showing round trip delay but only one direction delay?

Ping in emulation takes only one direction delay, if you have set only one application with Ping Source IP and ping Destination IP. This is because PING is a two way application and constitutes PING_REQUEST and PING_REPLY. For ping to take round trip delay users must configure two Emulation Applications, one for forward PING_REQUEST and other for the reverse PING_REPLY.

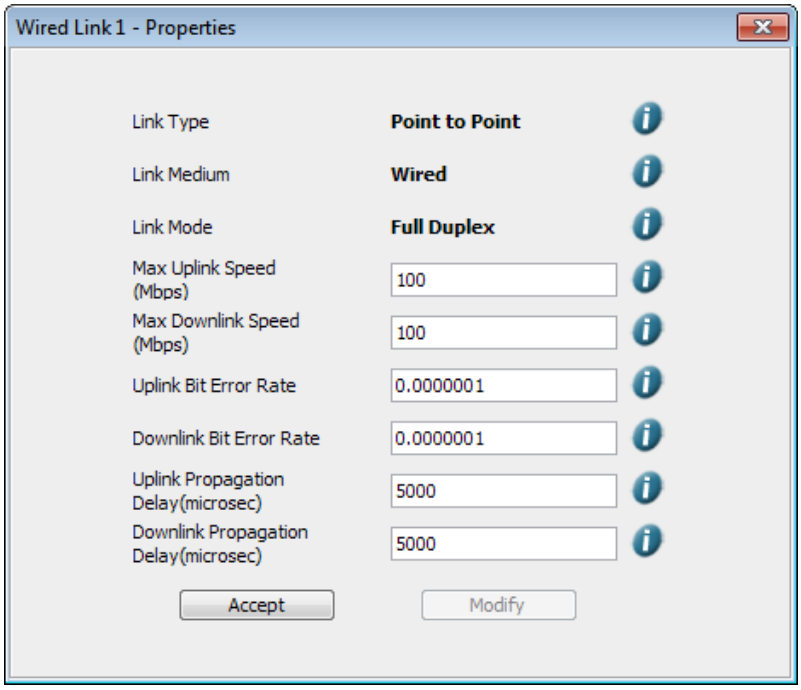
For Eg: If you are running a ping from the IP 192.168.0.151 to an IP 192.168.0.202 the time take will normally be around 1ms.



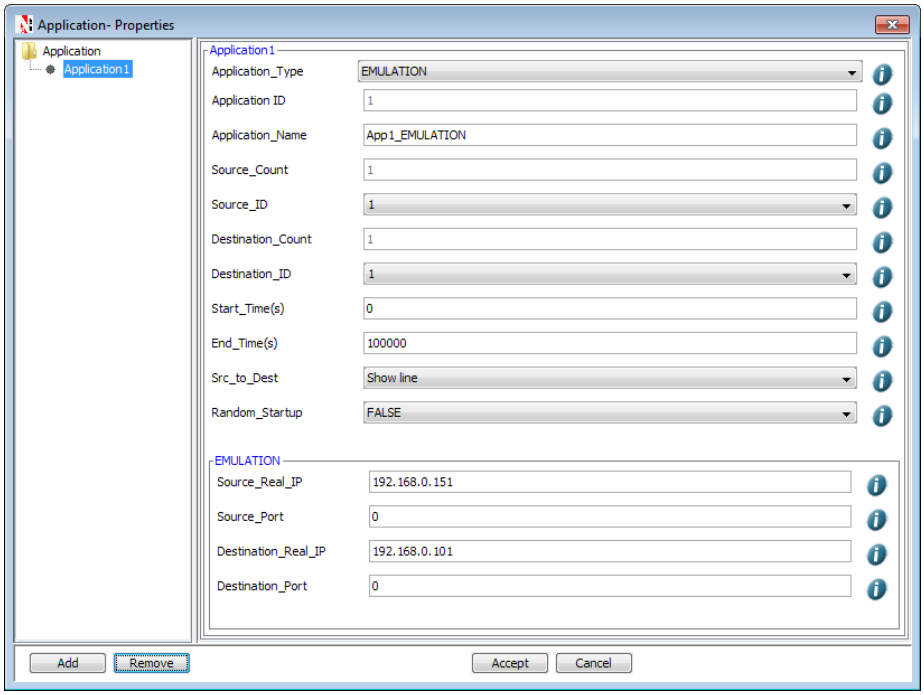
Now we create a network scenario in NetSim similar to the screenshot shown below,



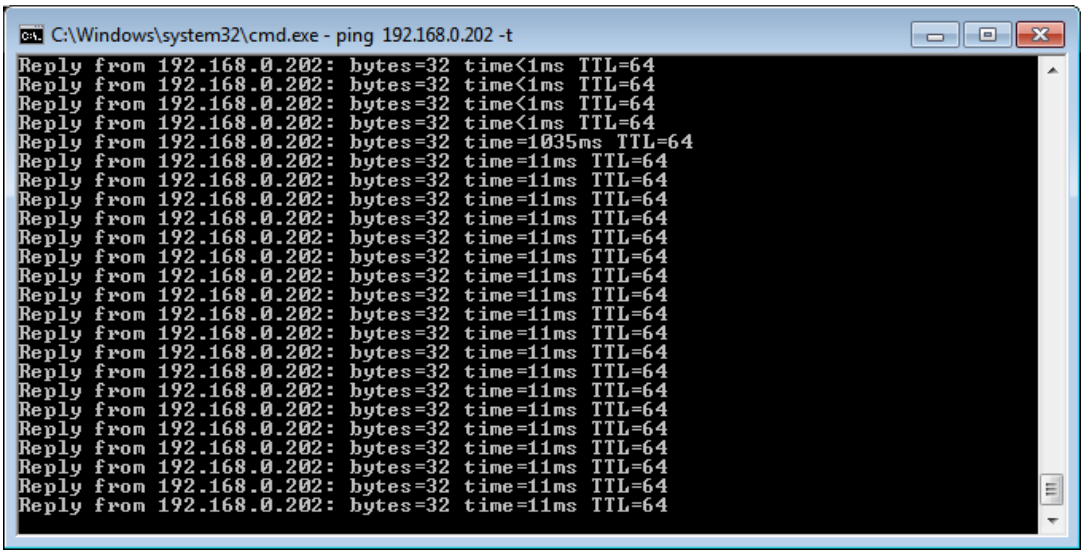
We reset the propagation delay in both the wired links to 5 ms.



We configure an Emulation application between the wired nodes with the source and destination real IP specified, as shown below:



On running the simulation, you will observe the variation in the time taken to get the ping reply in the source system, as shown below:

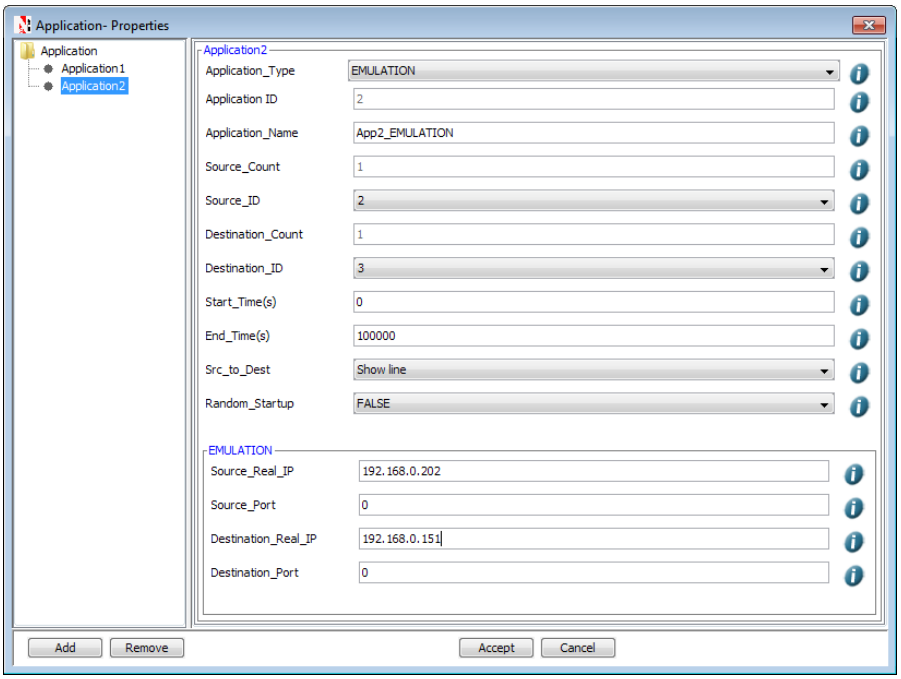


Ping packets has experienced an additional delay of 10ms which is a sum of the delay in both the links.

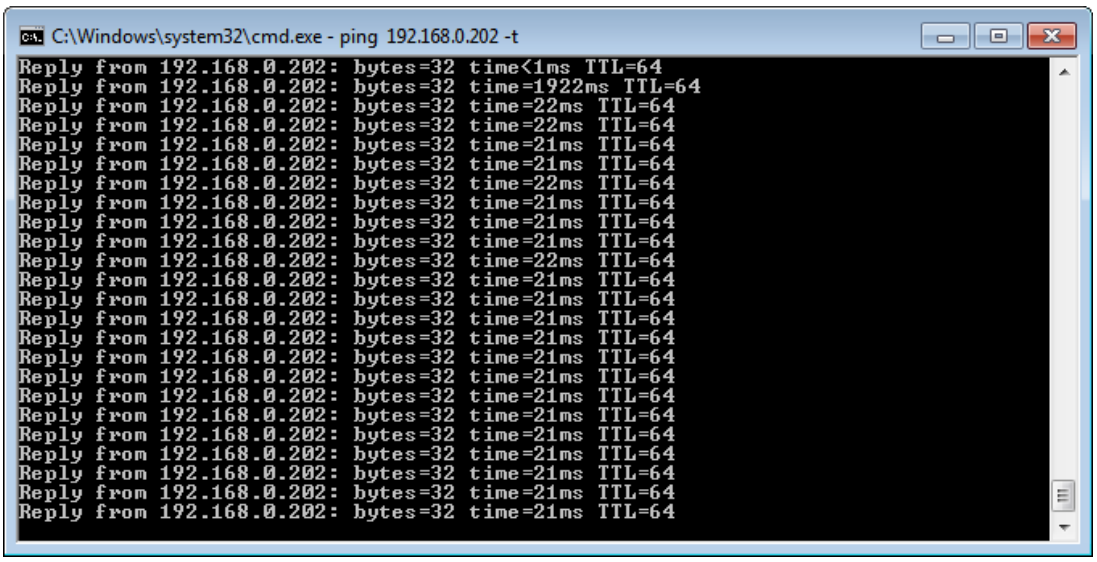
The additional delay experienced by ping packets is not 20ms because, the application that we have configured applies to only the Ping Reuqest packets which has the Source IP as 192.168.0.151 and Destination IP as 192.168.0.202.

The Ping Reply Packets has the Source IP as 192.168.0.202 and Destination IP as 192.168.0.151, for which we have not configured any application.

For the ping to take the round trip delay, we will have to configure one more application for the reverse traffic. On adding an application for the reverse traffic as shown below:



We will now be able to see round trip delay being experienced by the PING application, as shown below:



Ping experiences an additional overall delay of 20ms, which is the sum of the delay's experienced by Ping Request and Ping Reply (10ms + 10ms).

10 Programming Exercises

This menu contains network programming exercises and **is not available in Pro version**. Run down the menu and select the desired programming exercise. The programs available are as follows,

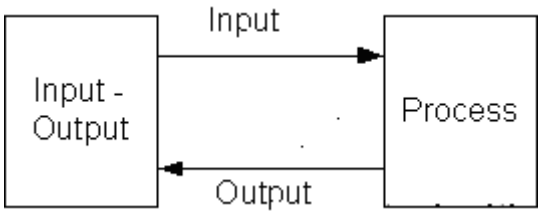
- Assignments of Sites to Concentrator
- Address Resolution Protocol
- Cryptography
    - Substitution
    - Transposition
    - XOR
    - Advanced
        - Data Encryption Standard
        - RSA
        - Wired Equivalent Privacy
- Distance Vector Routing
- Dynamic Host Configuration Protocol
- Error Correcting Code
    - Hamming Code
- Error Detecting Codes
    - Cyclic Redundancy Check
    - Longitudinal Redundancy Check
- Framing Sequence
    - Bit Stuffing
    - Character Stuffing
- Generic Cell Rate Algorithm
    - Virtual Scheduling Algorithm
- IPV4 Addressing
    - Address Mask
    - Binary Conversion
    - Classless InterDomain Routing
    - Network Address
    - Special Addresses

    - Subnetting
- IPV6 Addressing
    - EUI - 64 Interface Identifier
    - Host Addresses
    - Subnetting
- Leaky Bucket Algorithm
- Multi Level Multi Access
- Multiple Access Technology
    - CDMA
    - TDMA
    - OFDMA
- PC to PC Communication
    - Socket Programming
    - Chat Application
- Scheduling
- Shortest Path
- Sliding Window Protocol
- Sorting Techniques
- Spanning Tree
- Transmission Flow Control

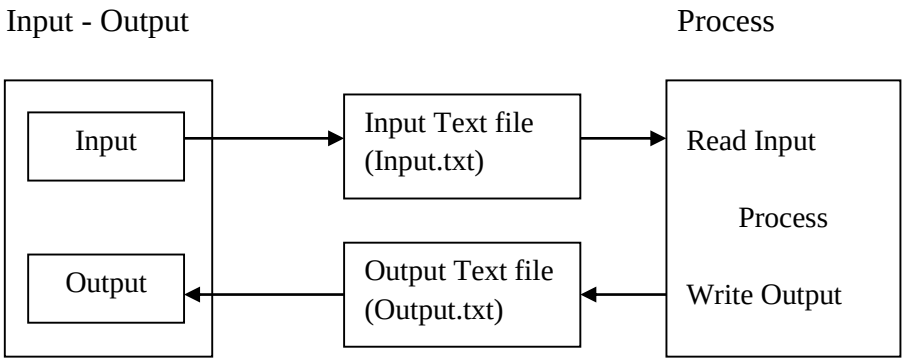
NetSim's Programming Lab has been designed to provide hands - on network programming skills to students. The labs come with a GUI where students can first run the experiment in "**Sample mode**" and subsequently link their own code and visualize it's working. Programs can be written in C and the executable can be linked to NetSim.

10.1 Architecture

The following **Architecture** is applicable for all the exercises under the **Programming** menu. Each exercise has two modules



Using the **Input - Output** module inputs are given and output is viewed. The working of the concept/algorithm is done in the process module. The link between the **Input - Output** module and process module is as follows



Process is the module for which the user has to write and link the written code when using the user mode.

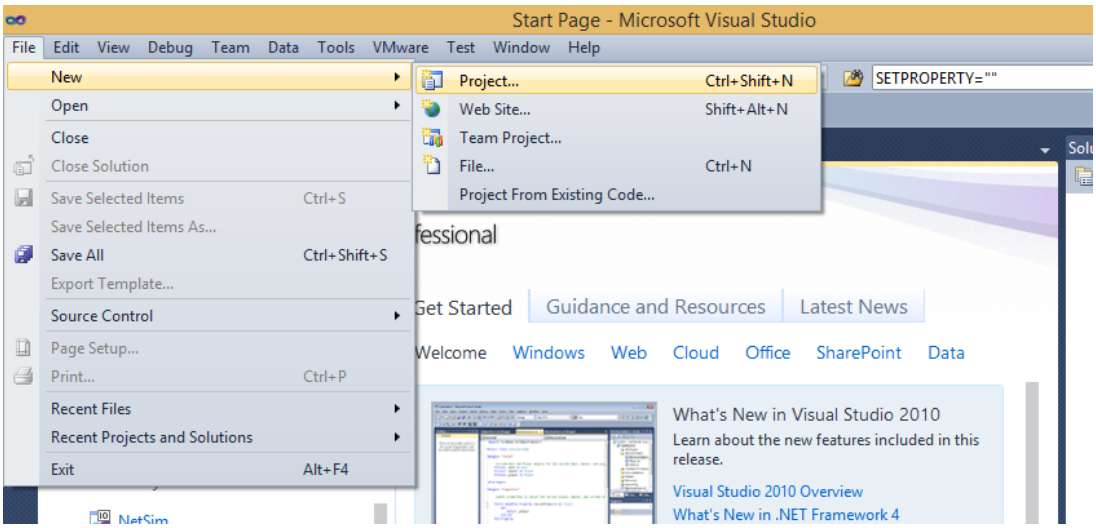
The code can be written either by C or C++, the executable file created should be linked to the software.

The input and output file format should be as required, as they form the link between the software and the user executable file.

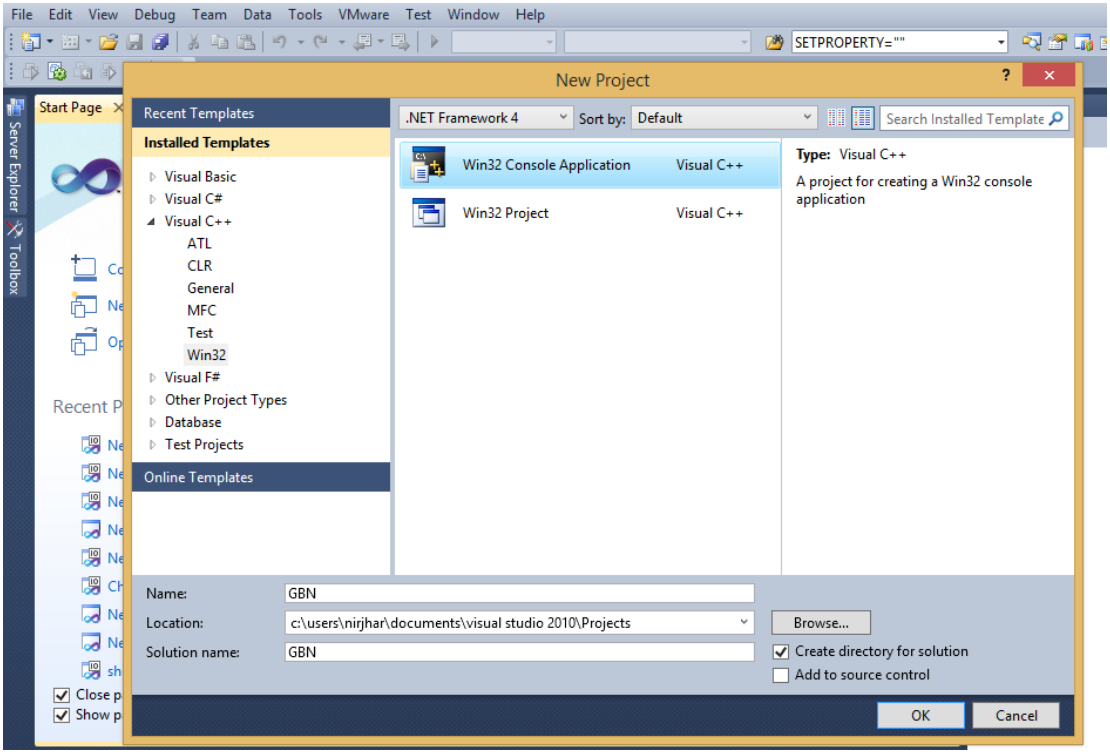
10.2 Creating .exe file for Programming Exercise

10.2.1 Using Visual Studio

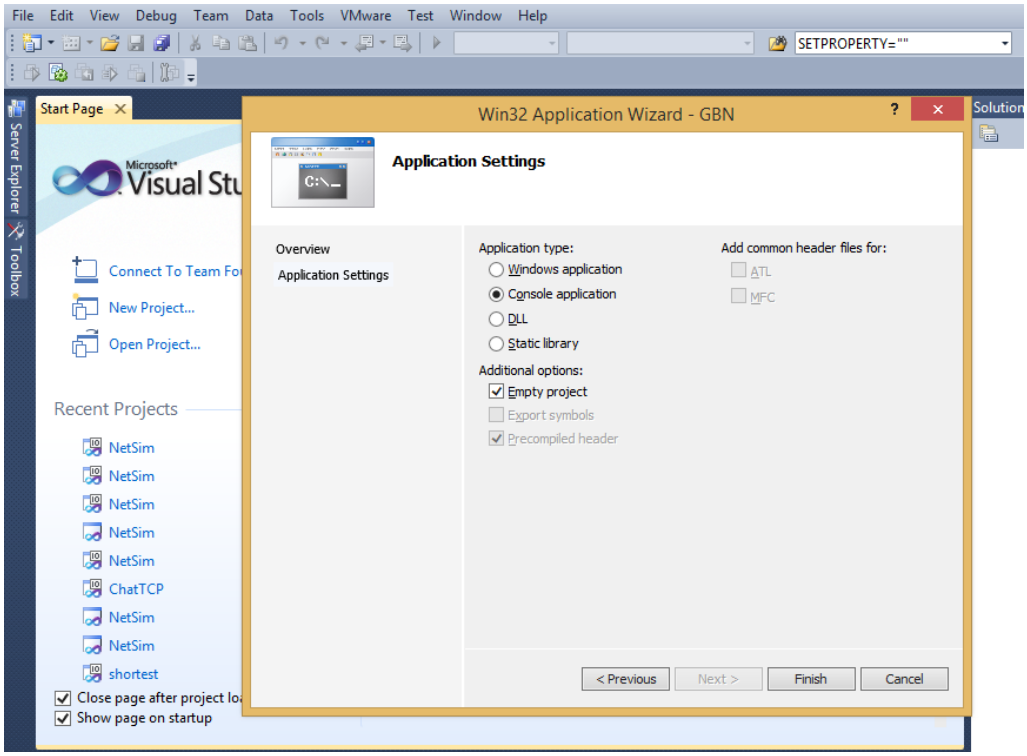
1. Select File→New→Project



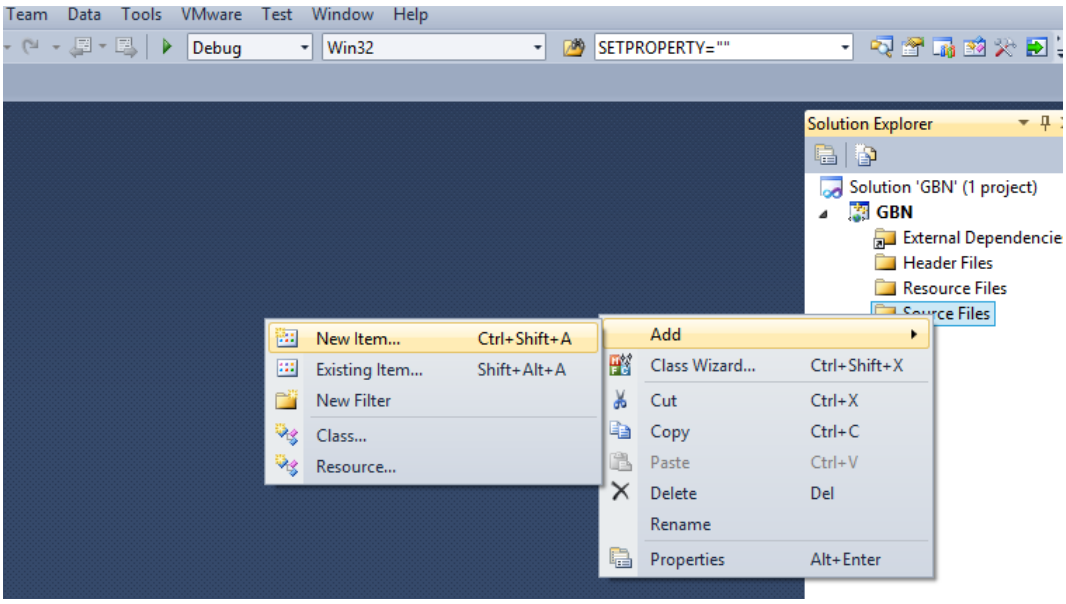
2. Select **Win32Console Application.** Name the project and select location to save the project and then click OK button.



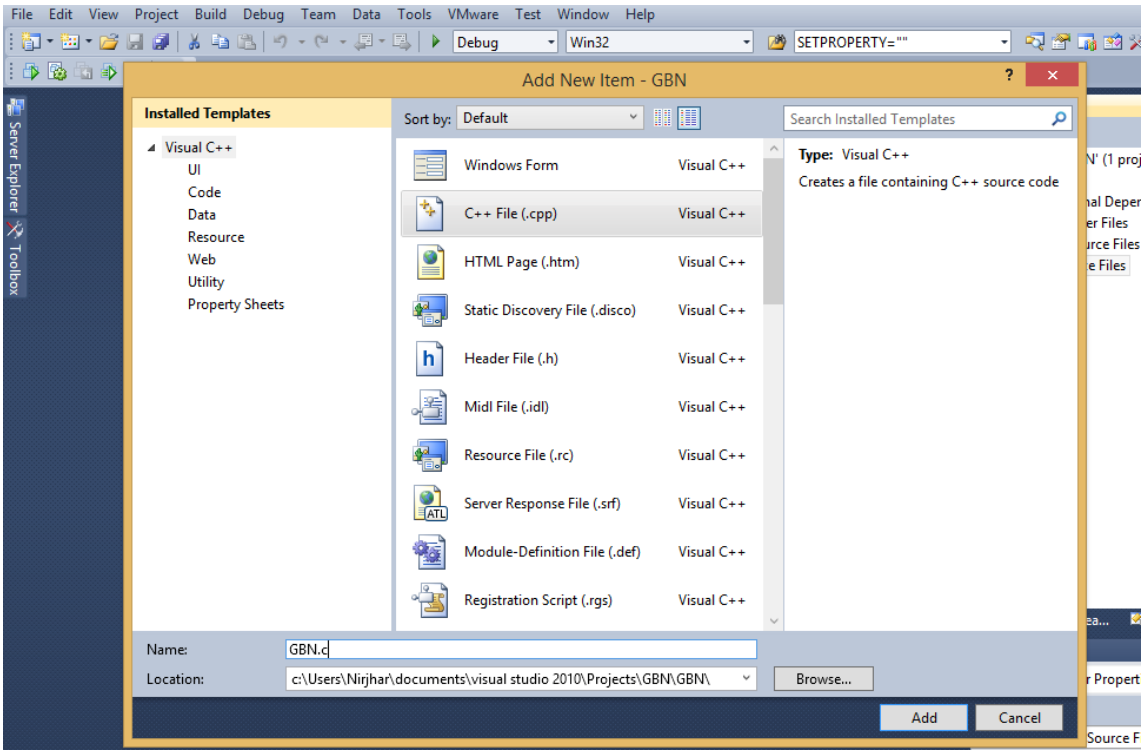
3. Click Next→ Check Whether Console application is selected or not. If selected, then select Empty Project, otherwise select Console application and Empty Project and finally click Finish button.



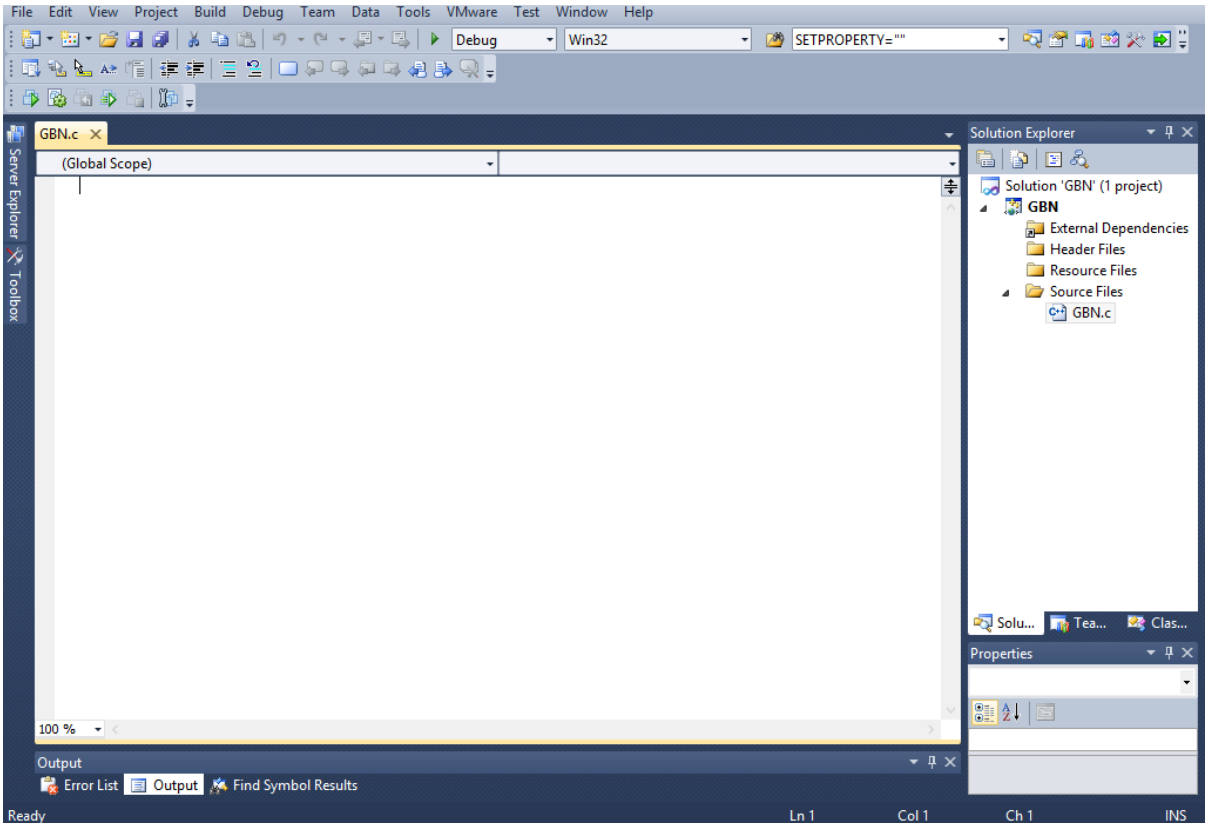
4. Add source codes to the project. Right click on the Source Files→Add→New Item



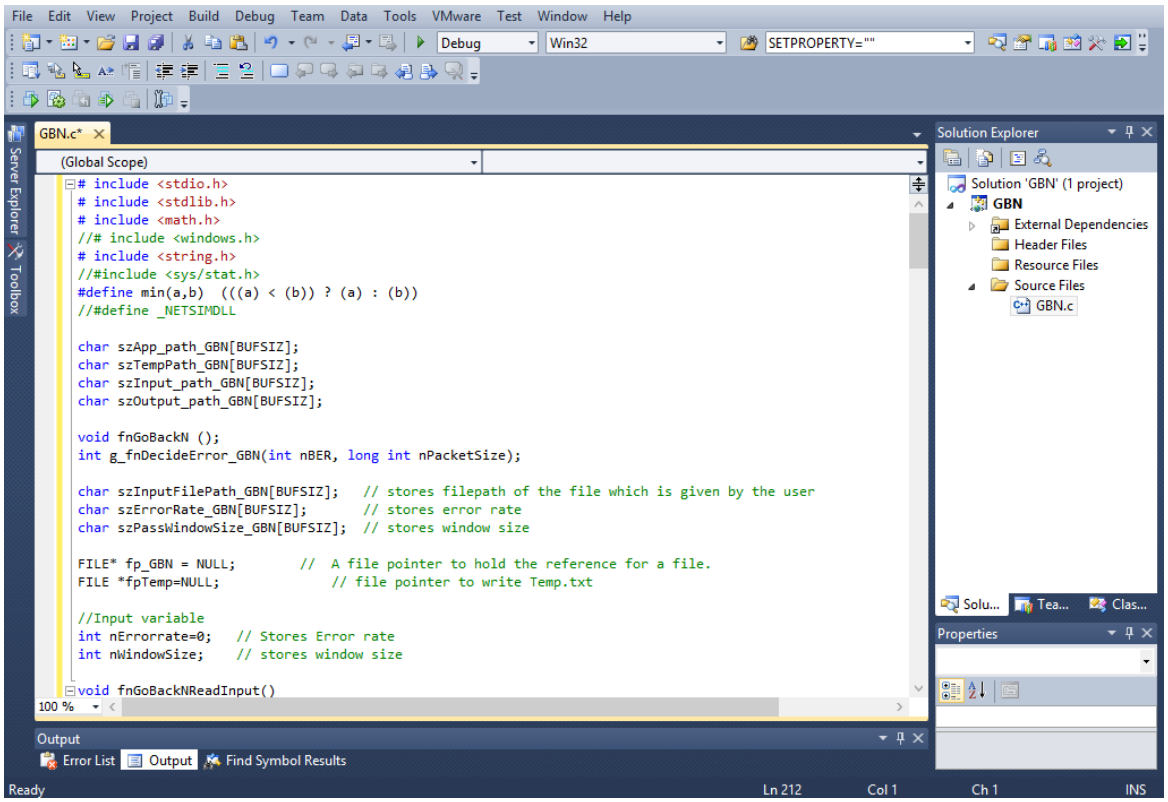
5. Select C++ File (.cpp) and name the file with extension of .c



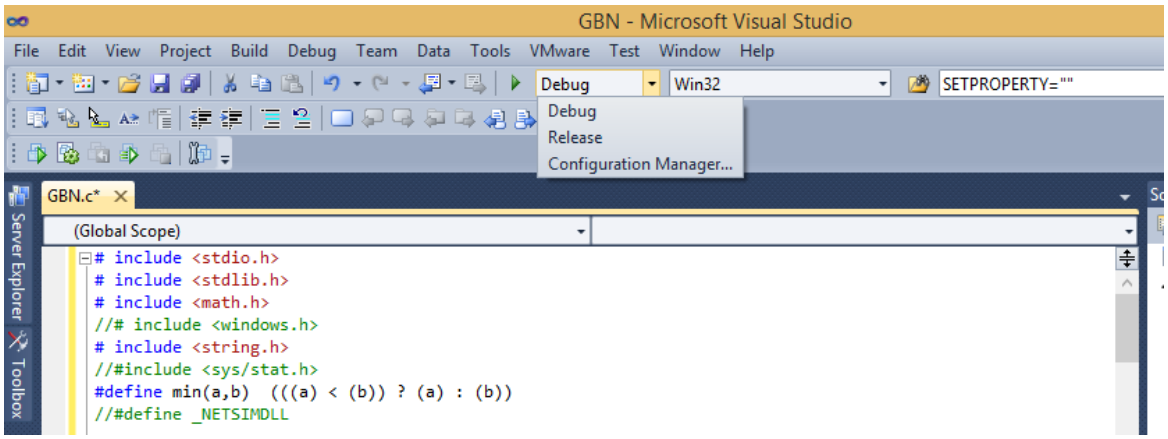
6. Now Source file is created



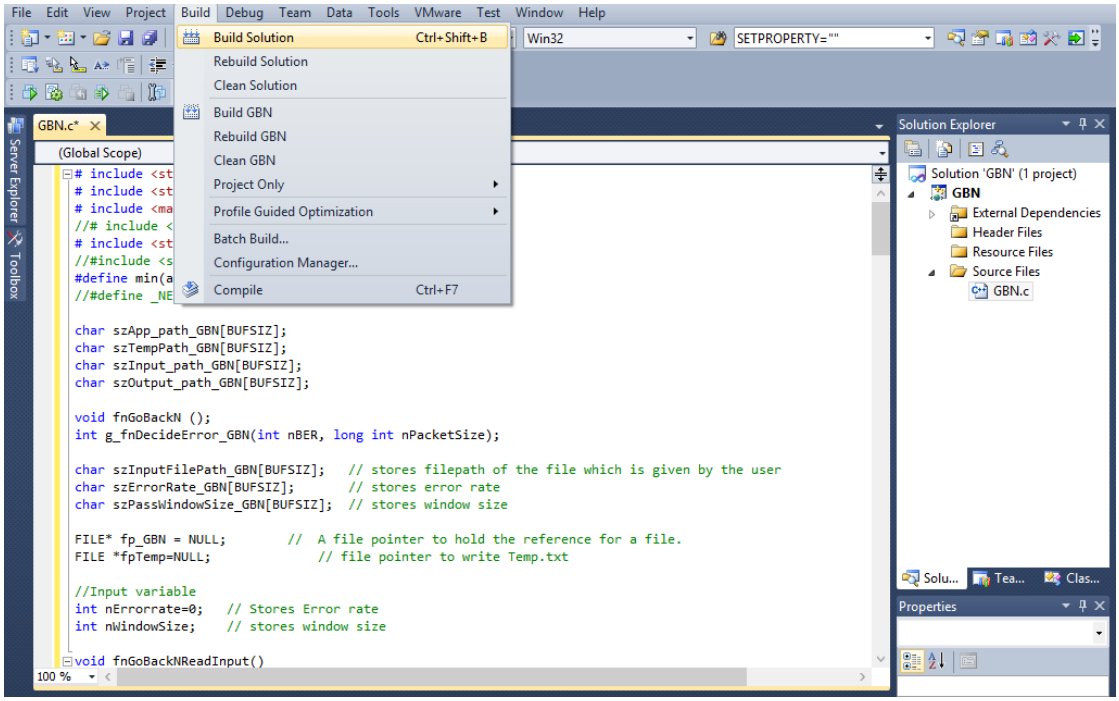
7. Copy and paste the source code.



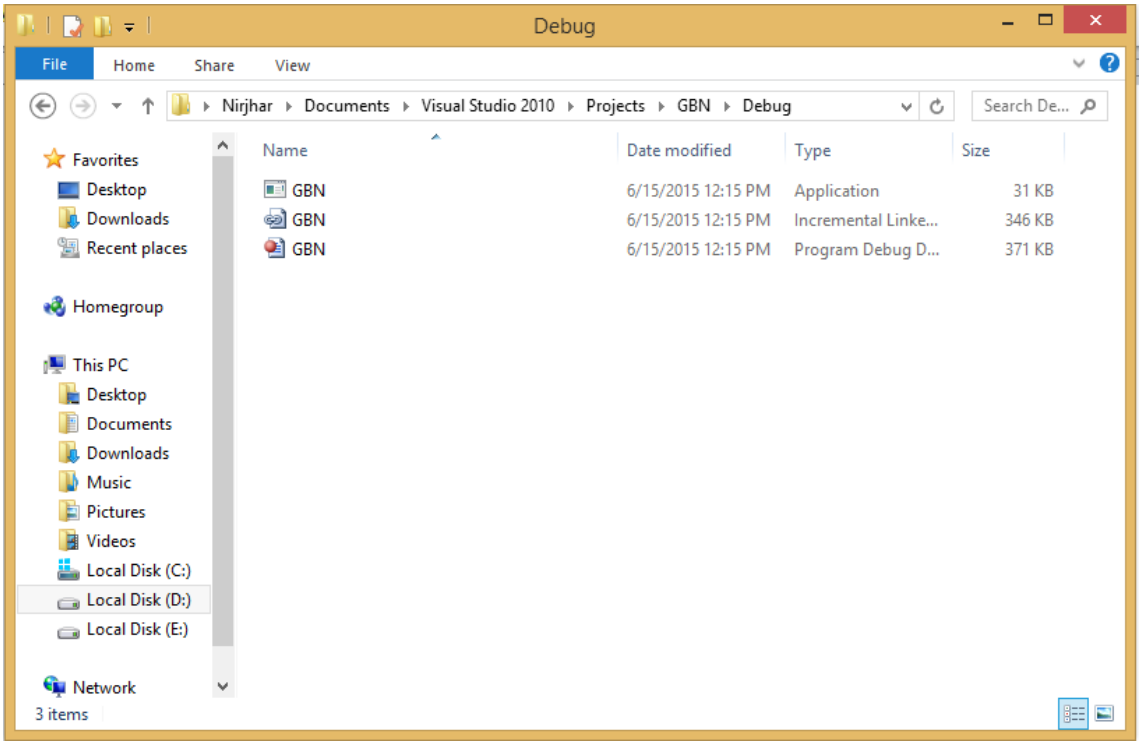
8. Select the mode for creating the exe i.e., debug or release mode by choosing the required option as shown in the figure. The preferred setting is debug mode.



9. To build the Solution,Select Build Menu →Build Solution



10. Now, Exe is created in the Project Folder as shown below.

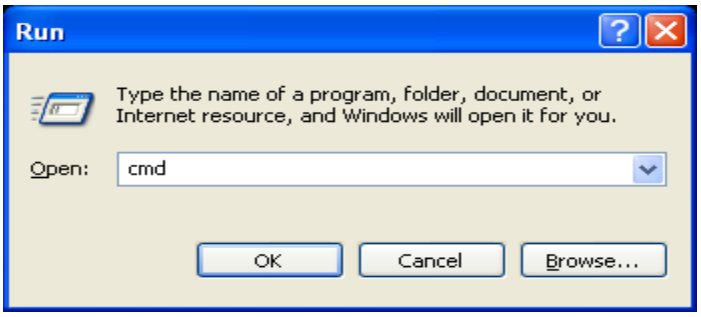


10.2.2 Using GCC

- **By using Command Prompt:**

  C / C++ files can be created using any editor. Ex: Notepad

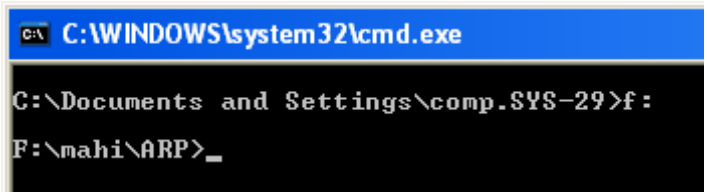
  Once C / C++ file is ready go to command prompt using Start → Run.



There are three cases for creating Exe (.exe) file using GCC.

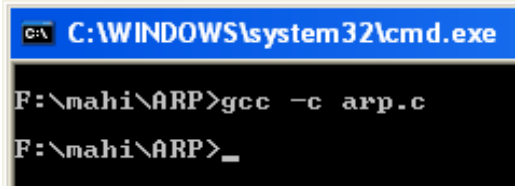
Case 1: C programs

1. Set the path as C program path



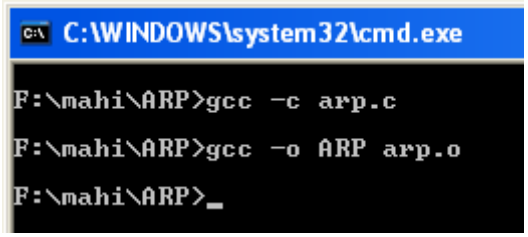
2. Create Output file (.o file) using the command

   gcc - C Filename.c



3. Create exe file (.exe file) using the command

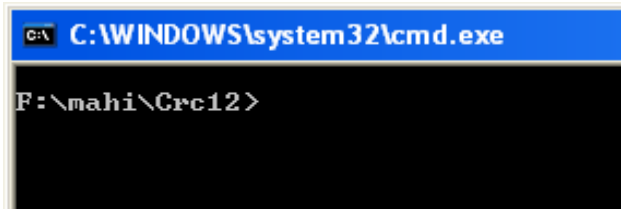
   gcc -o filename filename.o



4. Once exe file is created link that exe file with NetSim → Programming → User mode

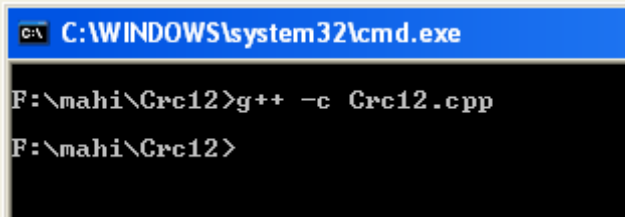
Case 2: C++ programs

1. Set the path as C++ program path.



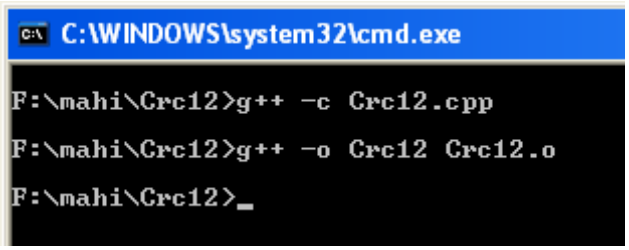
2. Create output file (.o file) using the command.

g++ -C Filename.c



3. Create exe file (.exe file) using the command

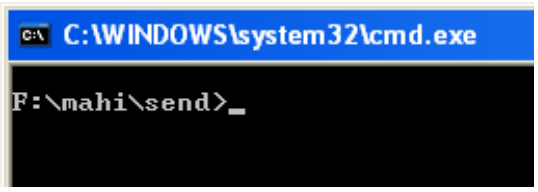
g++ -o filename filename.o



4. Once exe file is created link that exe file with NetSim → Programming → User mode

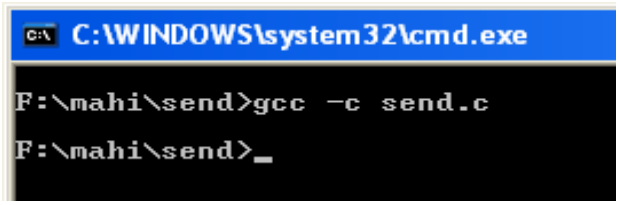
Case 3: Socket programs

1. Set the path to Socket program location.



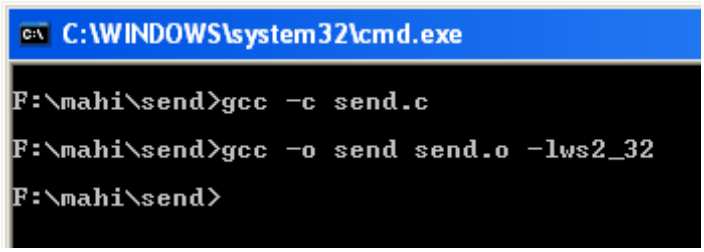
2. Create output file (.o file) using below command.

gcc -C Filename.c



3. Create exe file (.exe file) using below command

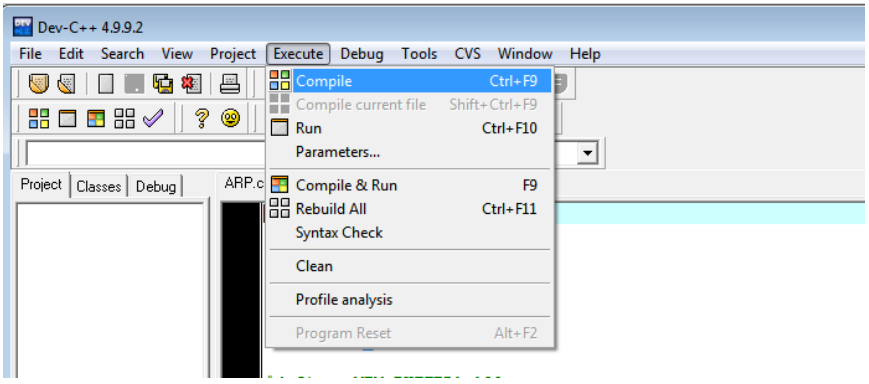
gcc -o send send.o -lws2_32



4. Once exe file is created link that exe file with NetSim → Programming → User mode

10.2.3 Using Dev C++

1. Open the C source code file in Dev C++. Then go to Execute → Compile.



Dev C++ will create the exe file in the same location where the C source code file is located

2. Once exe file is created , link that exe file with NetSim → Programming → User mode

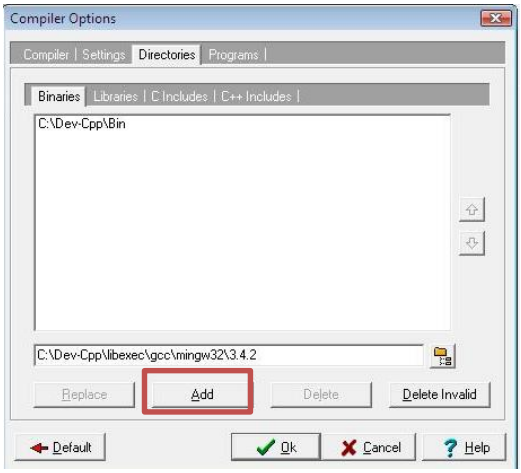
Note: For Windows 8 and above, use Dev C++ v 5.11

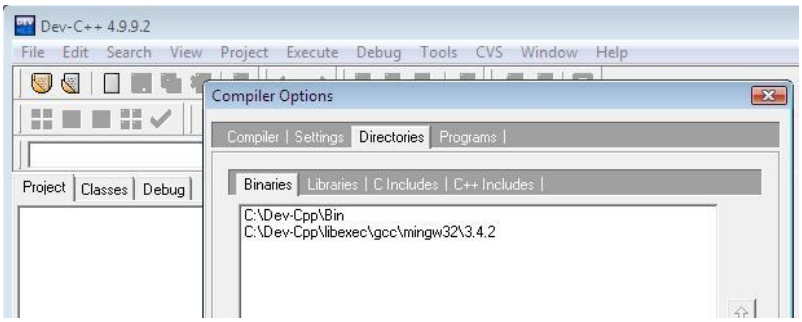
Note: For Windows Vista, set up Dev C++ as shown below

In Dev C++ 4.9.9.2.exe, go to Tools→ Compiler Options → Directories tab → Binaries tab

Add the following path as per your installation directory in the text box provided and click on add button.

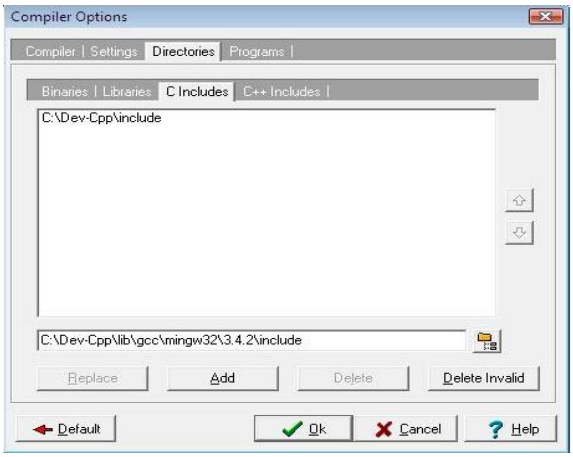
C:\Dev-Cpp\libexec\gcc\mingw32\3.4.2

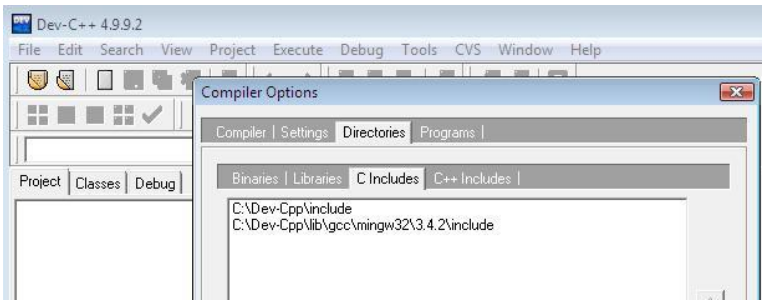




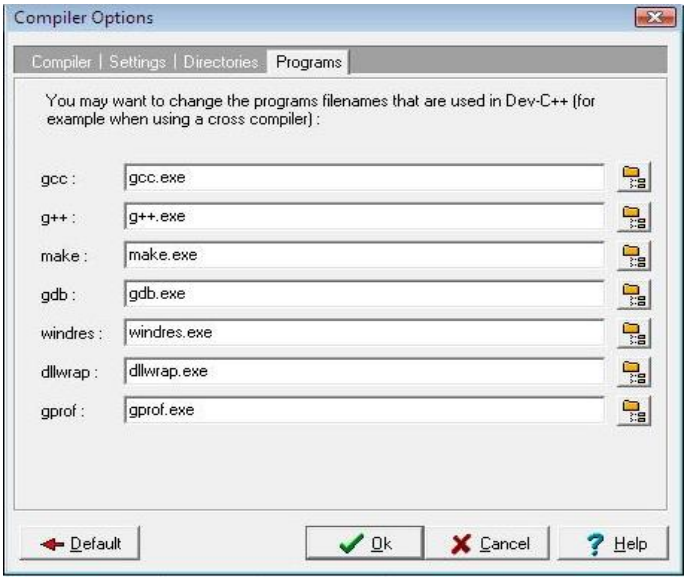
In the C Includes tab add the follwing path in the text box provided and click on add button.

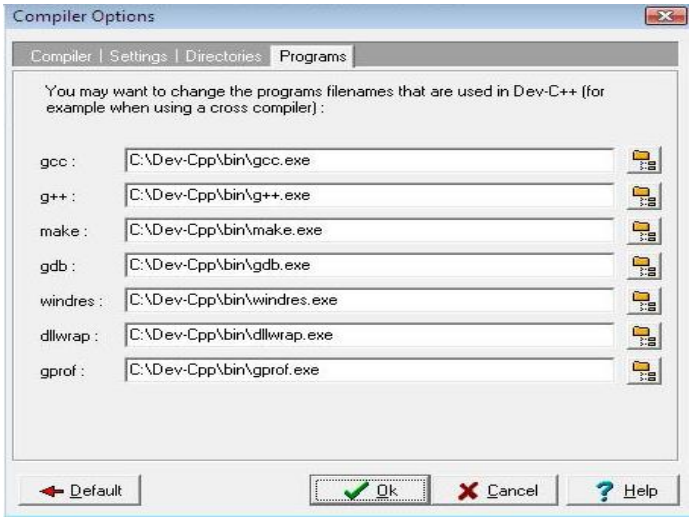
C:\Dev-Cpp\lib\gcc\mingw32\3.4.2\include





In the programs tab edit the following paths as shown in screenshot.





10.3 Steps to perform Programming Exercise in NetSim

In NetSim, select “Programming → < The exercise you want > ”.

Step 1:

In the left panel, select the Mode as **Sample**. Provide the required inputs in the panel and click on Run.

So presently NetSim will run that algorithm code already present in the software and will display the result graphically.

Step 2:

For user to write their own C Code in NetSim and check the result, click on “Concept Algorithm Pseudo Code and Flowchart” (present in Help in the left pane) and understand the working of the algorithm.

Step 3:

Then click on **Interface Source Code** (present in Help in the left pane).

Open Dev C++ or any GNU C compiler based IDK and copy the code from the Interface Source Code.

The user needs to edit the Interface Source Code at the following location. For Example,

```
int fnCrc12(char* pszString)
{
// Write your own code here
}
```

So the user needs to write the code, create exe and attach it with NetSim to run.

Step 4:

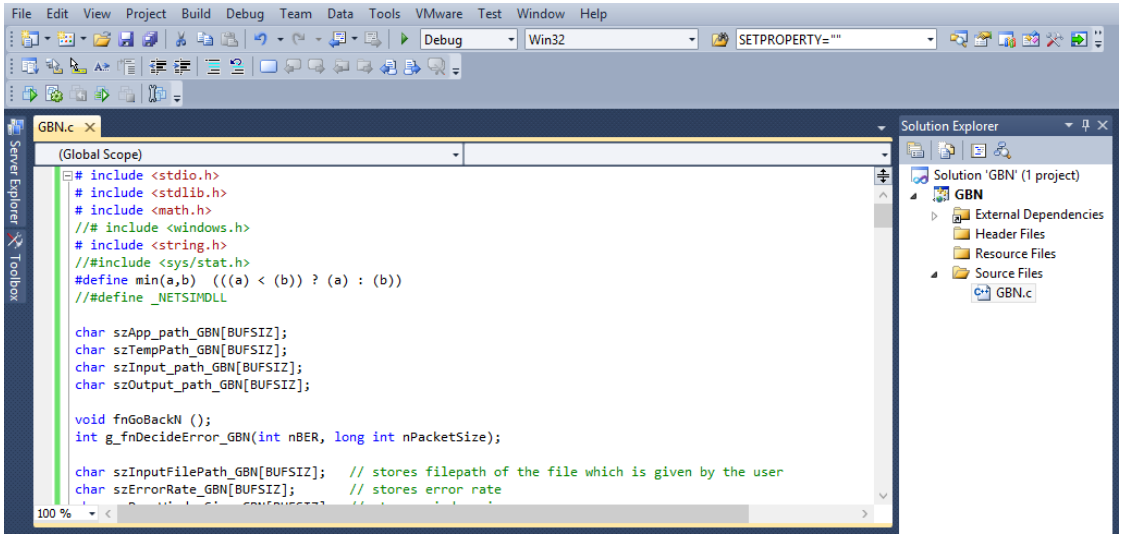
Create .exe file. In the left panel, select the Mode as **User.** Select the **.exe** file created above. Set the input and click Run. So presently NetSim will run code which is written by the user and will display the result graphically.

In case of any error, “ERROR IN USER CODE” message will be displayed.

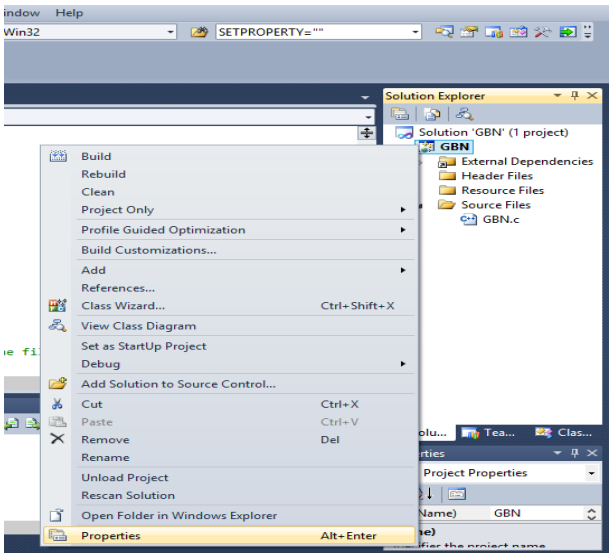
10.4 How to De-bug your code linked to NetSim's Programming Exercise

In NetSim, programming exercise menu, users can write and link their code as executables (*.exe). If the user's *.exe file does not provide the correct output, NetSim UI will display the message "Error in User Code". To de-bug your code on getting this message, follow the steps given below:

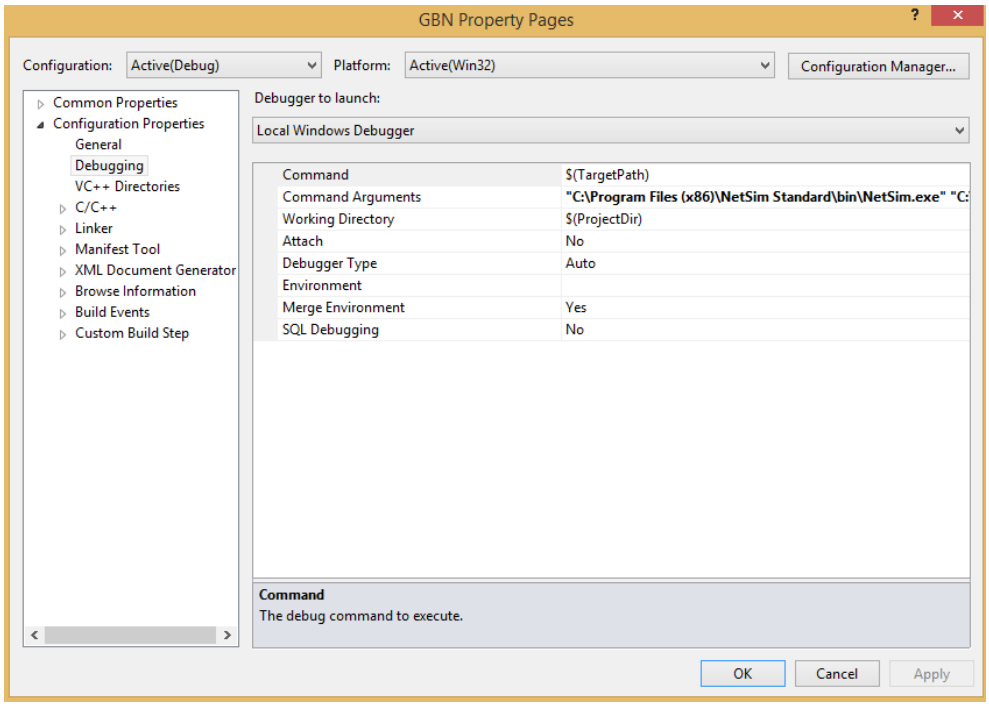
1. Run the scenario again in sample mode. On completion, minimize NetSim and keep it running without closing the programming exercise.
2. Open your code in Visual studio 2010.The procedure is explained above in "*Creating.exe file for Programming Exercises*".



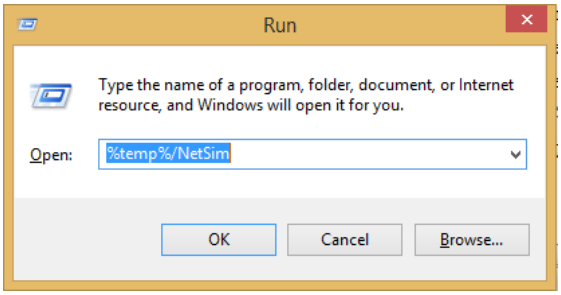
3. Right click on your project in the Solution Explorer pane and select properties as shown below



4. Inside the properties window select Debugging and edit the Command Arguments as shown



5. Inside command argument add the following two paths
   a. Path to where NetSim is installed within double quotes " ". This is usually C:\Program Files\NetSim Standard. This can be got by right clicking on the NetSim icon and selecting Find target or open file location.
   b. The windows temporary path which has the NetSim folder for temporary data. This can be got by Start → Run and typing %temp%/NetSim



   c. On clicking this, you will get a path similar to C:\Users\George\Local Settings \Temp\NetSim. As mentioned above, this should also be in double quotes " ", and there should be a single space between the first path and the second path.For example: "C:\Program Files (x86)\NetSim Standard\bin\NetSim.exe" "C:\Users\George \AppData \Local \Temp\NetSim"

6. Now add a breakpoint to your code in your function or in the main ( ), and proceed with de-bugging
7. At the end check if the output.txt present in the %temp%\NetSim path and the temp.txt present in the %temp%\NetSim path are exactly similar. Exact similarity would indicate that your code will work fine when you run it in use mode the next time.

10.5 Programming Exercises

10.5.1 Address Resolution Protocol

Programming Guidelines

This section guides the user to link his/her own code for Address Resolution Protocol to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
The first line of the ***Input.txt*** contains the number of **nodes** in the network, following which the IP Addresses are assigned to each node (ascending order). The last two lines are the Source node IP Address and the Destination node IP Address respectively. An example format of ***input.txt***is given below, Number_of_Nodes=3 Node1_IP_Address=192.168.1.1 Node2_IP_Address=192.168.1.2 Node3_IP_Address=192.168.1.3 Source_Node_IP_ Address=192.168.1.3 Destination_Node_IP_Address=192.168.1.1	The output file should contain two lines, The first line has the details of the Source (i.e.) in which class it is present. The second line has the result of the ARP i.e., whether destination is present in the class. The second line has the flag value (**important)** that is used for the animation. The result for the above is: Source is Class C.Destination is present in that class 2

Interface Source Code

Interface Source code written in C is given and using this the user can write only the Address Resolution Protocol inside the function fnARP () using the variables already declared.

To view the interface source code, go to **NetSim Installation path / src / Programming / ARP.c**

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenario:

Objective - To find the Medium Access Control (MAC**) Address of the Destination Node using** Address Resolution Protocol (ARP).

How to Proceed? - The objective can be executed in NetSim by using the programming exercise available. In the **Programming** menu select **Address Resolution Protocol(ARP)**.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- **Click** on the drop down button and select 6 **Nodes**.
- List of **NodesNumbers** along with their **IPAddresses** that would be availed are,

Node Numbers	IP Address
1	192.168.1.1
2	192.168.1.2
3	192.168.1.3
4	192.168.1.4
5	192.168.1.5
6	192.168.1.6

- **ARP Request System** and **ARP Reply System** needs to be selected. That is,
  1. **ARP Request System→** 192.168.1. "1" (Any one **Node Number** to be selected)
  2. **ARP Reply System→** 192.168.1. "6" (Any one **Node Number** to be selected)
- Then **Run** button need to be **clicked**.

Output - **Output** for the above **Sample** is as follows,

- The **Source Node** (i.e. **Node Number** 1 in the above example) sends the **ARP Request** to the **Connecting Device**.
- The **Connecting Device** then broadcasts the **ARP Request** to all the **Nodes** available in the network.
- The **Destination Node** (i.e. **Node Number** 6 in the above example) sends an acknowledgement in the form of **ARP Reply** (i.e. The **Destination MAC - MAC Address** of the **Destination Node**) to the **Connecting Device**.
- The **Device** then transmits the **ARP Reply** (i.e. The **Destination MAC - MAC Address** of the **Destination Node**) only to the **Source Node**.
- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.2 Assignment of Sites to Concentrator

Programming Guidelines

This section guides the user to link his/her own code for Assignment of Sites to Concentrator to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input file format	Output file format
Input.txt contains the number of sites, number of concentrators, and number of sites that can be assigned to a concentrator, the order in which sites are to be assigned and the distance matrix given by the user. The format of input.txtis: Number_of_Sites=2 Number_of_Concentrators=2 Sites_per_Concentrator=1 Selected_Priority=1>0> Distance 1>2> 3>4> Note: '>' is the delimiter symbol, which is used to separate each input.	In the output file (i.e. 'Output.txt') each line should indicate one complete traverse. For example, for the above input, the output file should be: 2>1> 1>2> In each line, the first character indicates the site and the rest indicates the concentrators. The second character (excluding the '>' symbol) is the first concentrator found with minimum distance/cost. The third character is a concentrator, which has the minimum distance/cost compared to the previous one and so on. Note:'>' is the delimiter symbol, which is used to separate each input.

Interface Source Code

Interface Source code code written in C is given using this the user can write only the Assignments of Sites to Concentrators inside the function fnAssignmentsofSites() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ AssignSites.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To allocate the resource which is concentrator to the require system which is sites based on the distance for each sites to concentrator.

How to proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Assignment of Sites to Concentrator.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- Number of Sites need to be **selected**. The value that has selected ranges from **1** to **7**.
- Number of Concentrators need to be **selected**. The values available for selection are **2**, **3** and **4**.
- Number of Sites / Concentrators need to be **selected**. The value that has selected ranges from **1** to **7**.
- **Click** on the image to **select** Priority. **Click** on Change Priority to reset the Priority of the Sites.
- **Enter** the Distance in the given table. Distance should be in the range, **1** to **99** km.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.
- **Click** on Concept, Algorithm, Pseudo Code & Flow Chart to get help on it.

Output - The following steps are under gone internally,

- The site which has the highest priority searches for a concentrator which is the nearest.
- A red line appears between the site and a concentrator. This indicates the only shortest path available between the site and the concentrators. Hence, it indicates the site has been allocated to that concentrator.
- A site can have only one concentrator, whereas a concentrator can have many sites linked to it.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.3 Cryptography - Substitution - Encryption

Programming Guidelines

This section guides the user to link his/her own code for Substitution to NetSim.

Pre-conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
There are six lines in Input.txt file. Cryptographic_Technique=Substitution Cryptographic_Method=Encryption Plain_Text: tetcos Key_Text: 2	Plain letter>encrypted letter for the corresponding plain letter> t>u>v> e>f>g> t>u>v> c>d>e> o>p>q> s>t>u>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Substitution-Encryption inside the function fnSubstutionEncryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ SubstEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.4 Cryptography - Substitution - Decryption

Programming Guidelines

This section guides the user to link his/her own code for Substitution to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	**Output File Format**
There are six lines in Input.txt file. Cryptographic_Technique=Substitution Cryptographic_Method=Decryption Cipher_Text: vgvequ Key_Text: 2	Cipher letter>decrypted letter for the corresponding cipher letter> v>u>t> g>f>e> v>u>t> e>d>c> q>p>o> u>t>s>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Substitution-Decryption inside the function SubstutionDecryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ SubstDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Encrypt and decrypt the message with the same key value using Substitution.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography → Substitution.**

❖ **Encryption:**

Sample Input: In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Encryption need to be selected.
- Enter the Plain Text that needs to be encrypted. Maximum of 8 alphabets need to be entered.
- Enter the Key Value. This is an Integer which is within the range **0** to **26**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The Plain Text and Key Value entered would be displayed in red color.
- The corresponding Cipher Text would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

❖ **Decryption:**

Sample Input: In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Decryption need to be selected.

- Cipher Text obtained while encryption is filled in the Plain Text and also the Key Text is same as that entered while encrypting.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The Cipher Text and Key Value entered would be displayed in red color.
- The corresponding Plain Text that had been entered at the time of encrypting is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.5 Cryptography - Transposition - Encryption

Programming Guidelines

This section guides the user to link his/her own code for Transposition Encryption to NetSim.

Pre-conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt. The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Technique=Transposition Cryptographic_Method=Encryption Plain_Text: TETCOS Key_Text: BLORE	1>3>4>5>2> T>E>T>C>O>S>T>E>T>C> T>S>O>C>E>T>T>E>C>T>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Transposition-Encryption inside the function fnTranspositionEncryption() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ TranspEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.6 Cryptography - Transposition – Decryption

Programming Guidelines

This section guides the user to link his/her own code for Transposition Decryption to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	**Output File Format**
Cryptographic_Technique=Transposition Cryptographic_Method=Decryption Cipher_Text: TSOCETTECT Key_Text: BLORE	BELOR> 1>5>2>3>4> T>O>E>T>C>S>C>T>E>T>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Transposition-Decryption inside the function fnTranspositionDecryption() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ TranspDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Encrypt and decrypt the message with the same key text using Transposition.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography → Transposition**.

❖ **Encryption:**

Sample Input: In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Encryption need to be selected.
- Enter the Cipher Text. Maximum of 14 Characters can be entered.
- Enter the Key Value. Maximum of 8 alphabets can be entered.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The entered Key Value would be displayed first.
- The order of the Key Value is internally sensed.
- The corresponding Cipher Text is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

❖ **Decryption:**

Sample Input: In the Input panel the following steps need to be done,

- Once Encryption is done Decryption has to be selected.
- The Cipher Text obtained for Encryption is the Cipher Text for Decryption. This would be automatically taken. Maximum of 14 Characters can be entered.
- Enter the Key Value will also be taken internally. Maximum of 8 alphabets can be entered.

- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The entered Key Value would be displayed first. This is arranged in such a way that the order is changed when compared to Encryption.
- The order of the Key Value is internally sensed.
- The corresponding Plain Text is obtained. This Plain Text would be similar to the Plain Text entered at the time of Encryption.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.7 Cryptography - XOR - Encryption

Programming Guidelines

This section guides the user to link his/her own code for XOR Encryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	**Output File Format**
Cryptographic_Technique=XOR Cryptographic_Method=Encryption Plain_Text: TETCOS Key_Text: BNLORE	01010100>01000010>00010110> 01000101>01001110>00001011> 01010100>01001100>00011000> 01000011>01001111>00001100> 01001111>01010010>00011101> 01010011>01000101>00010110>

Interface Source Code

Interface Source code written in C is given using this the user can write only the XOR-Encryption inside the function fnXOREncryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ XorEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.8 Cryptography - XOR - Decryption

Programming Guidelines

This section guides the user to link his/her own code for XOR Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type**"%temp%"→NetSim→** "**Input.txt**" and "**Output.txt**"

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Technique=XOR Cryptographic_Method=Decryption Cipher_Text: 000101100000101100011000000011000001110100010110 Key_Text: BNLORE	00010110>01000010>01010100> 00001011>01001110>01000101> 00011000>01001100>01010100> 00001100>01001111>01000011> 00011101>01010010>01001111> 00010110>01000101>01010011>

Interface Source Code

Here a skeleton code written in C is given using this the user can write only the XOR-Decryption inside the function fnXORDecryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ XorDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Encrypt and Decrypt the message by using the same KeyText in XOR.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select **Cryptography →XOR**.

Note:Encryption should be done first and then **Decryption** should be done.

- **Encryption:**

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **EncryptionMode** needs to be selected.
- **PlainText** and **KeyValue** need to be entered in the fields available. That is,
  - **Plain Text→** tetcos (Maximum of 8 Characters)
  - **Key Text** → 123456 (Maximum of 8 Characters)

  Note: If the length of the **PlainText** and **Key Value** differs then an error would pop out.
- Then **Run** button need to be clicked.

Output - Output for the above Sample is as follows,

- **Plain Text** → t e t c o s
- **Key Text** → 1 2 3 4 5 6
- **Binary of plain text** → 01110011
- **Binary of key text** → 00110110
- **XOR Value** → 01000101
- **ASCII Equivalent** → 69 87 71 87 90 69
- **Cipher Text** →E W G W Z E

❖ Decryption:

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- After completing the **Encryption** part, **DecryptionMode** needs to be selected.
- **CipherText** (this field is automatically filled) and **KeyValue** (this should be same as the one that is entered in Encryption) need to be entered in the fields available. That is,
  - **CipherText→** 0100010101010111010001110101011101011010001000101 (Cipher Text in Binary Format Maximum 64 bits)
  - **KeyText→** 123456 (Maximum of 8 Characters)

  Note - Both **Cipher Text** and **Key Text** is automatically taken when **Decryption** button is selected (i.e. after undergoing **Encryption**).
- Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

- **Cipher Text** →E W G W Z E
- **Key Text** → 1 2 3 4 5 6
- **Binary of cipher text** → 01000101
- **Binary of key text** → 00110110
- **XOR Value** → 01110011
- **ASCII Equivalent** → 116 101 116 99 111 115
- **Plain Text** → t e t c o s

Note -

- **Text** in the **Input** of the **Encryption** and **Output** of the **Decryption** should be the same if the **Key Value** is same.
- The **Cipher Text** in case of **Encryption** is **Alpha Numeric** (i.e. for the user to understand in a better manner)and in case of **Decryption** the tool converts it to **Binary form** (i.e. since, the tool doesn't recognize **Alpha Numeric**).

10.5.9 Cryptography - Data Encryption Standard (DES) - Encryption

Programming Guidelines

This section guides the user to link his/her own code for DES Encryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

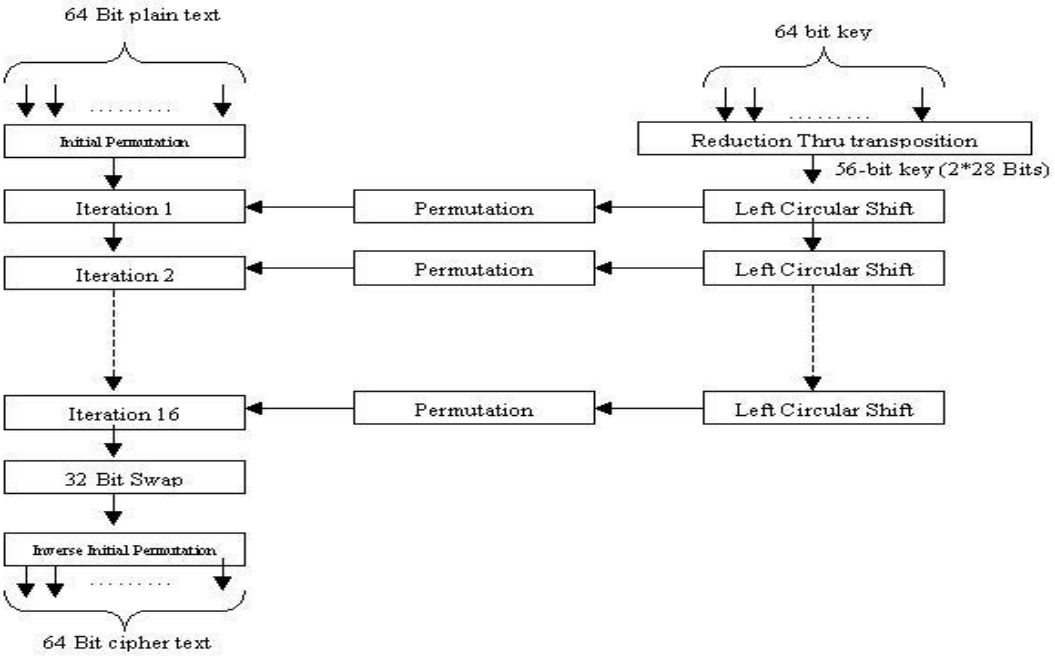
Note: The temporary directory is navigated through the following step.

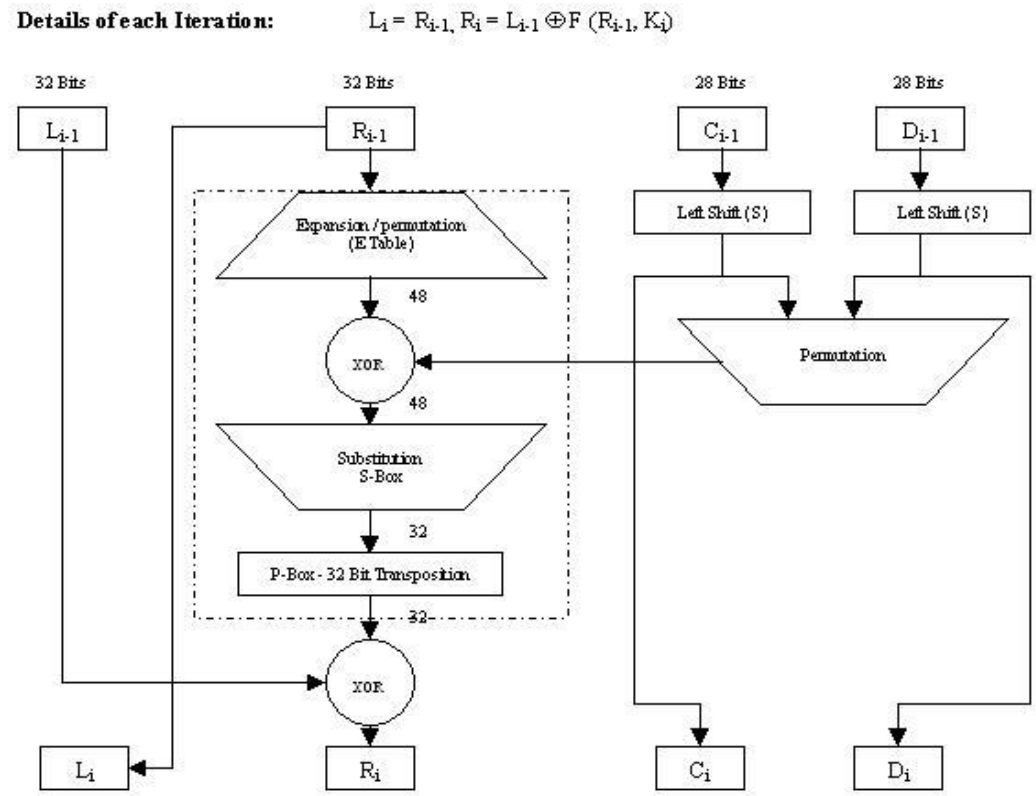
Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

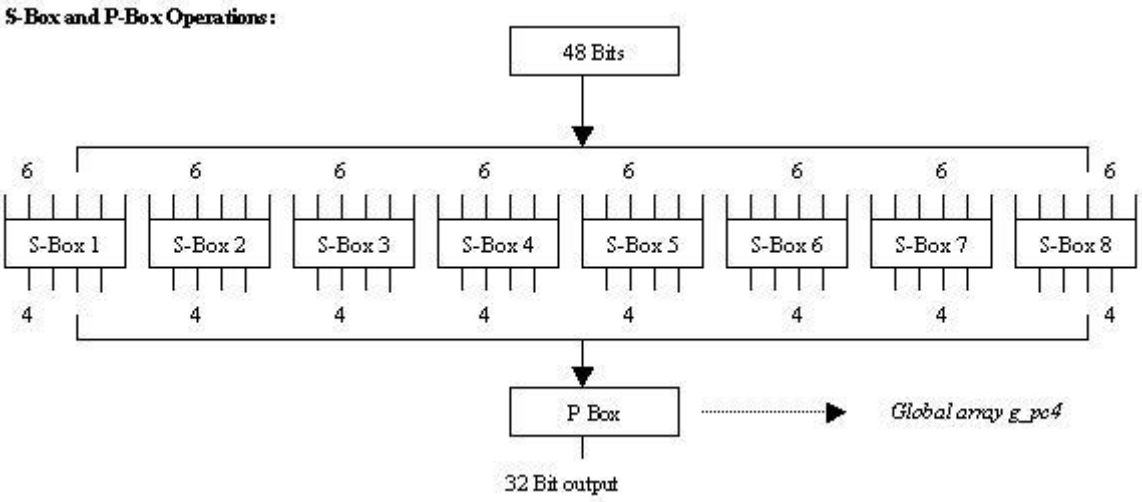
General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,The results of the program should be written into the output file **Output.txt**.







6bit - 4 Bit conversion logic (in each S-Box):

Of the 6 input bits, 4 bits (1, 2, 3, and 4) are used for identifying the S-Box column (4 bits representing 16 columns) and the remaining 2 bits (0, 5) are used for identifying the S-Box row (2 bits representing 4 rows). The corresponding hexadecimal number is chosen from the S-box. Since each Hexadecimal number represents 4 bits in binary form, the total output is of 32 bits.

Input File Format	Output File Format
Cryptographic_Method=Encryption Key_Text=abcdef1234567890 No_of_Iterations=1 Data=tetcos	The number of lines present depends on the number of iterations chosen. The number of lines equals the number of iterations, plus one. The last line of inputs gives the data (encrypted data if encryption has been chosen, else the decrypted data if decryption has been chosen). The previous lines give the DES Key generated for encryption or decryption. 1101010101111100100110101100001011100110000110 01616075733F724B40

Interface Source Code

Interface Source code written in C is given using this the user can write only the DES-Encryption inside the function fnDESEncryption() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ DesEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.10 Cryptography - Data Encryption Standard (DES) - Decryption

Programming Guidelines

This section guides the user to link his/her own code for DES Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

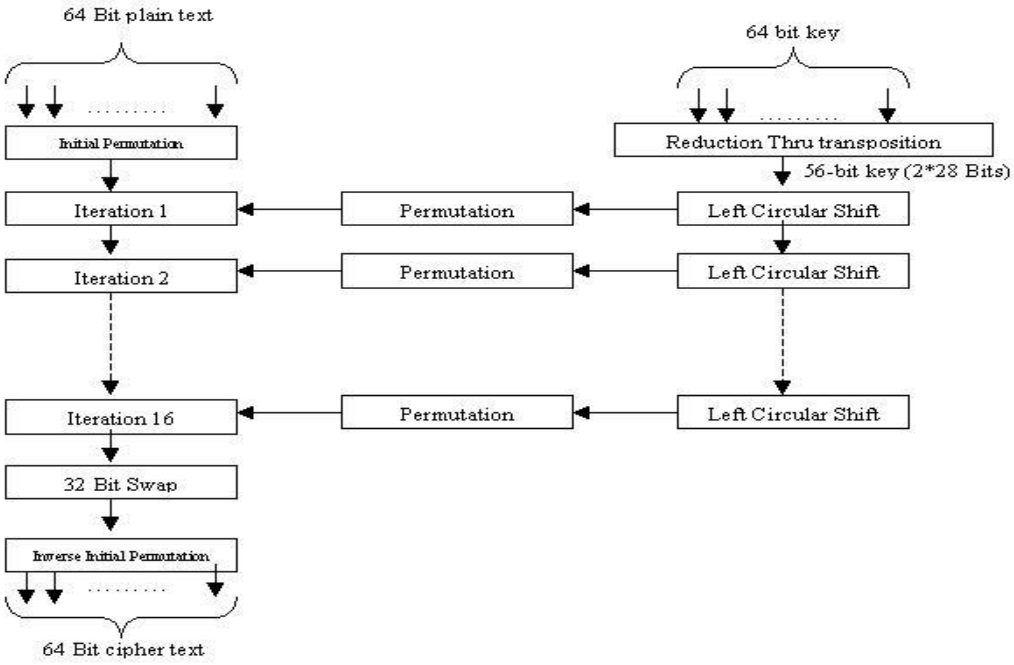
Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

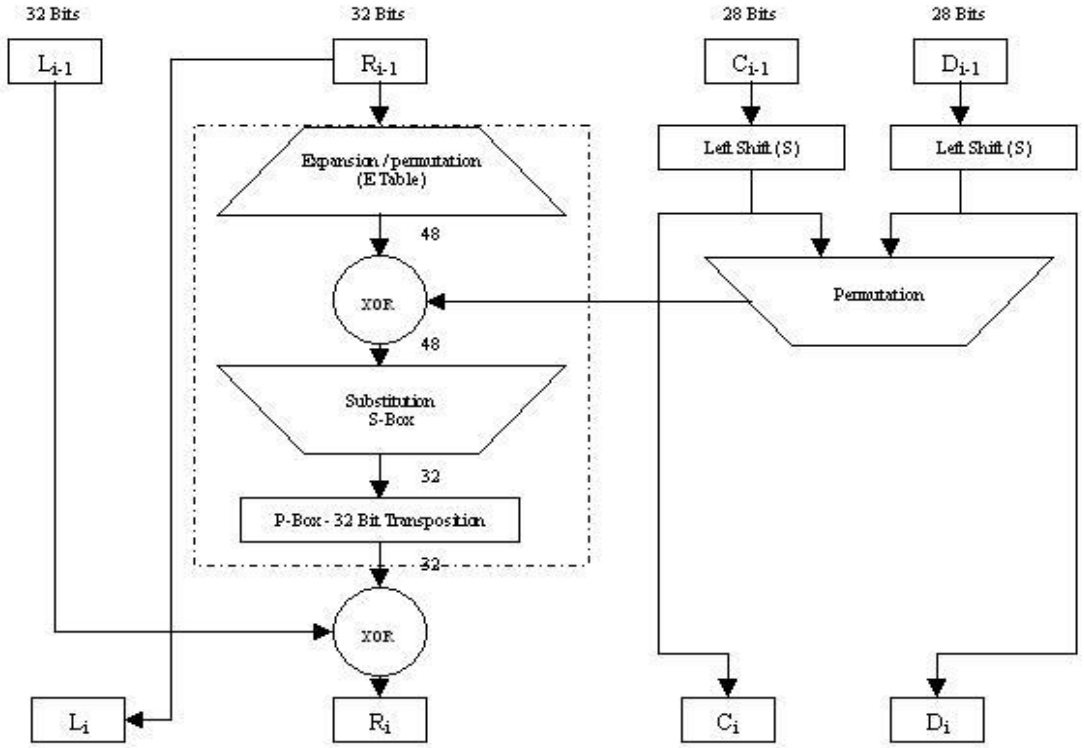
The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

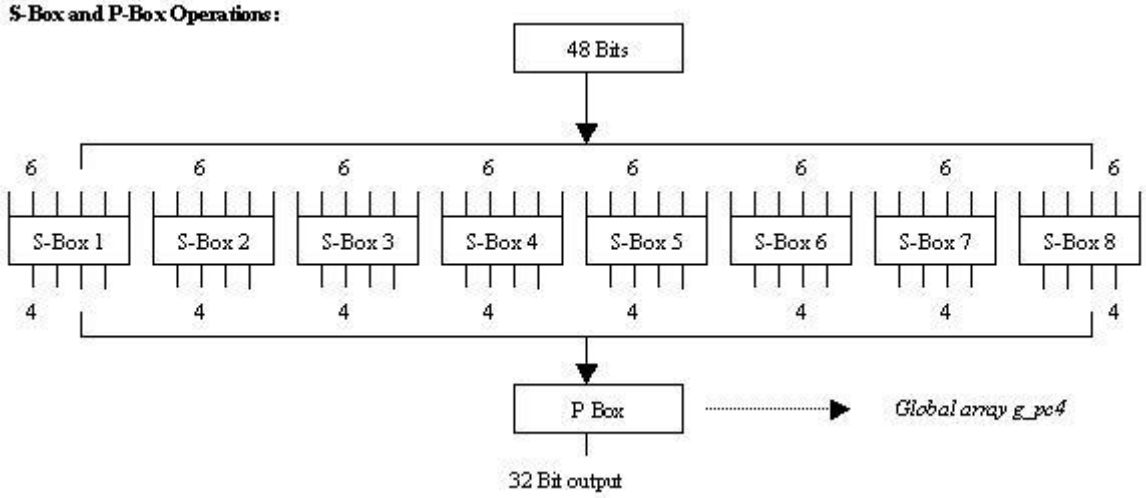
The results of the program should be written into the output file **Output.txt**.



Details of each Iteration: $L_i = R_{i-1}, R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$



S-Box and P-Box Operations:



6bit - 4 Bit conversion logic (in each S-Box):

Of the 6 input bits, 4 bits (1, 2, 3, 4) are used for identifying the S-Box column (4 bits representing 16 columns) and the remaining 2 bits (0, 5) are used for identifying the S-

Box row (2 bits representing 4 rows). The corresponding hexadecimal number is chosen from the S-box. Since each Hexadecimal number represents 4 bits in binary form, the total output is of 32 bits.

Input File Format	Output File Format
Cryptographic_Method=Decryption Key_Text=abcdef1234567890 No_of_Iterations=1 Data=616075733F724B40	The number of lines present depends on the number of iterations chosen. The number of lines equals the number of iterations, plus one. The last line of inputs gives the data (encrypted data if encryption has been chosen, else the decrypted data if decryption has been chosen). The previous lines give the DES Key generated for encryption or decryption. 1101010101111100100110101100001011100110000110 01 tetcos

Interface Source Code

Interface Source code written in C is given using this the user can write only the DES-Decryption inside the function fnDESDecryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ DesDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Encrypt and decrypt the message with the using DES.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography→Advanced→DataEncryptionStandard**.

❖ Encryption:

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Encryption need to be **selected**.
- Enter the Key Text in the provided field. Only hexadecimal characters have to be entered. Maximum of 16 characters can be entered in this field. Characters more than 16 will be filled in the Data field.

- Number of Iterations need to be selected. The value ranges from **1** to **16**.
- Data needs to be entered in this field that has to be encrypted. Data can be entered only when the Enter Key Text field is filed with 16 hexadecimal characters. Maximum of 1500 characters can be stuffed into this field.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output -

- Key1 (i.e. if Iteration is 1) in binary format is obtained.
- The corresponding Cipher text is obtained.
- Click on Copy button and then on the Paste button to make use of the Cipher text for Decryption.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

❖ **Decryption:**

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Once the Encryption is done, click on Copy and Paste buttons. The encrypted data is filled in the data field available in the Decryption view.
- Fields such as Enter the Key Text, Number of Iteration and Data is filled in automatically when Copy and Paste button is clicked.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Key1 (i.e. if Iteration is 1) in binary format is obtained.
- The Data that was entered in the Encryption view would be encrypted and displayed.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.11 Rivest-Shamir - Adleman Algorithm (RSA)

Programming Guidelines

This section guides the user to link his/her own code for RSAAlgorithmto NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input file format	Output file format
Input.txt file has the input data. The format is: Plain_Text=T	dt1>dt2>dt3>dt4>dt5>dt6>dt7>dt8> dt 1: specifies the prime P dt 2: specifies the prime Q dt 3: specifies the value of N dt 4: specifies the value of Z dt 5: specifies the value of Kp dt 6: specifies the value of Ks dt 7: specifies the value of the Cipher Text dt 8: specifies the value of the Plain Text Example: For input: 'T' Output: 11>3>33>20>7>3>14>20>

Interface Source Code

To view the interface source code, go to

NetSim Installation path / src / Programming/ RSA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenario:

Objective - Encrypt and decrypt the message with the using DES.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography→Advanced→RSA**.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Plain Text of only one character has to be entered in the field available.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Value of the prime numbers P and Q are obtained.
- Value of N[p*q] is obtained.
- Value of Z [(p-1)*(q-1)] is obtained.
- Value of Kp and Ks is obtained
- Plain Text which is the actual count of the alphabet is obtained.
- The corresponding Cipher Text is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.12 Cryptography - Wired Equivalent Privacy (WEP) – Encryption

Programming Guidelines

This section guides the user to link his/her own code for WEP Encryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Method=Encryption Data: TETCOS	There are 3 lines in output.txt file Initialization Vector Key Text Cipher Text Ex: 8A699C 0123456789 364130B9CA018D5B760383CD85528751

Interface Source Code

Interface Source code written in C is given using this the user can write only the WEP-Encryption inside the function fnWEPEncrypt () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ WEPEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.13 Cryptography - Wired Equivalent Privacy (WEP) - Decryption

Programming Guidelines

This section guides the user to link his/her own code for WEP Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note:

The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Method=Decryption Initialization_Vector=8A699C Key=0123456789 Data: 364130B9CA018D5B760383CD85528751	Plain text Ex: TETCOS

Interface Source Code

Interface Source code written in C is given using this the user can write only the WEP-Decryption inside the function fnWEPDecrypt () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ WEPDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Encrypt and Decrypt the message by using WEP.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In **Programming** menu select **Cryptography→Advanced→Wired Equivalent Privacy**.

Note: **Encryption** should be done first and then **Decryption** should be done.

❖ **Encryption**

Sample Inputs - By using the **Input Panel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- Under **CryptographicMethod**, **Encryption Mode** needs to be selected.
- **Input→Data** (Maximum of 1500 Characters) - **NetSim TetCos Bangalore**
- Then **Run** button need to be clicked.

Sample Output - Output for the above **Sample** is as follows,

- **Initialization Vector** : 3E41C6
- **Key** : ABCDEF0123
- **The Encrypted Text obtained** - "786441A955C419F9537576954F897BCC5866549653DB5CA73E1B4D F63CA939AD"
- **Click** on **Copy** button and then on **Paste** button. This would **Copy** the **EncryptedText** onto the **DecryptionInputDatafield**.

❖ **Decryption**

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected. (This is **Automatically** selected once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- Under **Cryptographic Method**, **Encryption Mode** needs to be selected. (This is **Automatically** selected once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- **Enter Key Value→** 3E41C6ABCDEF0123 (16Hexadecimal Characters). (This field is **Automatically** filled once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- **Number of Iterations→** 1.
- **Input→Data** - "786441A955C419F9537576954F897BCC5866549653DB5CA73E1B4DF63CA939AD" (This field is **Automatically** filled once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- Then **Run** button need to be clicked.

Sample Output - Output for the above **Sample** is as follows,

- **Initialization Vector** : 3E41C6
- **Key** : ABCDEF0123
- The **EncryptedText** obtained - **NetSim TetCos Bangalore**

10.5.14 Distance Vector Routing

Programming Guidelines

This section guides the user to link his/her own code for Distance Vector Routing to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Router_ID=1>Router_Name=Router_1>No_Of_Neighbour=2>Neighbours_ID=2>5 Router_ID=2>Router_Name=Router_2>No_Of_Neighbour=1>Neighbours_ID=1 Router_ID=3>Router_Name=Router_3>No_Of_Neighbour=0 Router_ID=4>Router_Name=Router_4>No_Of_Neighbour=0 Router_ID=5>Router_Name=Router_5>No_Of_Neighbour=1>Neighbours_ID=1 Router_ID=6>Router_Name=Router_6>No_Of_Neighbour=0	Initial Stage>Source Router ID - 1>Destination RouterID>CostID>Intermediate RouterID Ex: 0>1>2>1>0 0>1>5>1>0 0>2>1>1>0 0>5>1>1>0 1>1>2>1>0 1>1>5>1>0 1>2>1>1>0 1>2>5>2>1 1>5>1>1>0 1>5>2>2>1

	2>1>2>1>0 2>1>5>1>0 2>2>1>1>0 2>2>5>2>1 2>5>1>1>0 2>5>2>2>1

Interface Source Code

Interface Source code written in C is given using this the user can write only the Distance Vector Routing inside the function fnDistVectAlgorithm() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ DVR.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

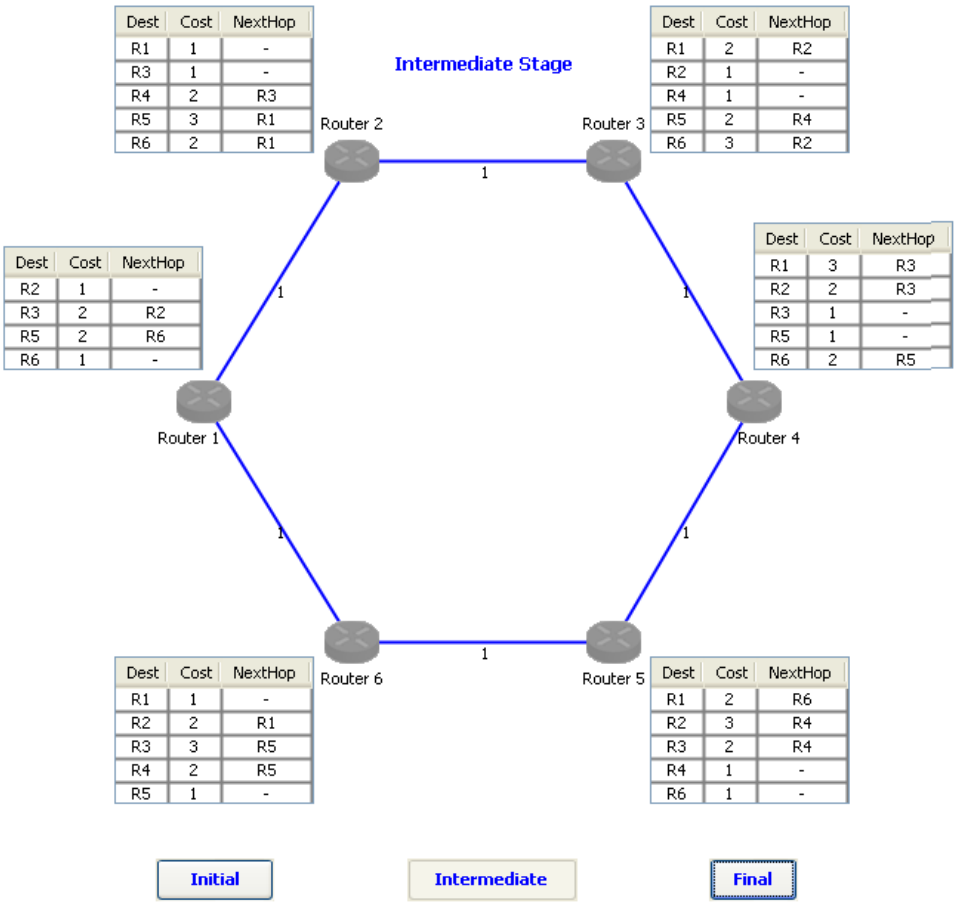
Objective - Find the Shortest Path using Distance Vector Routing.

How to Proceed? - The objective can be executed in NetSim by using the programming exercise available. In the **Programming** menu select**Distance Vector Routing.**

Sample Input -Follow the below given steps,

- Click on two Routers to establish a Path (i.e. 1st Click on Router number 1 and 2, then similarly on Router number 2 and 3, 3 and 4, 4 and 5, 5 and 6, & 6 and 1).
- Router 1, 2, 3, 4, 5, 6 are connected.
- Click on **Initial, Intermediate** and **Final** button to execute.

Output - The Output that is obtained is given below,



Initial button can be clicked to view the initial table of the all router.

Intermediate button can be clicked to view the intermediate table of the all router.

Final button can be clicked to view the final table of the all router.

10.5.15 Distance Host Configuration Protocol

Programming Guidelines

This section guides the user to link his/her own code for Dynamic Host Configuration Protocol to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Number_of_Clients = 4 Start_Address = 192.168.0.105 End_Address = 192.168.0.109 Mask = 255.255.255.0	DHCP_DISCOVER> Node1>Node2> Node1>Node3> Node1>Node4> Node1>Server DHCP_OFFER> Server>Node1 DHCP_REQUEST> Node1>Server DHCP_ACK> Server>192.168.0.105 DHCP_DISCOVER> Node2>Node1> Node2>Node3> Node2>Node4> Node2>Server DHCP_OFFER> Server>Node2 DHCP_REQUEST> Node2>Server

	DHCP_ACK> Server>192.168.0.106 DHCP_DISCOVER> Node3>Node1> Node3>Node2> Node3>Node4> Node3>Server DHCP_OFFER> Server>Node3 DHCP_REQUEST> Node3>Server DHCP_ACK> Server>192.168.0.107 DHCP_DISCOVER> Node4>Node1> Node4>Node2> Node4>Node3> Node4>Server DHCP_OFFER> Server>Node4 DHCP_REQUEST> Node4>Server DHCP_ACK> Server>192.168.0.108

Interface Source Code

Interface Source code written in C is given using this the user can write only the Dynamic Host Configuration Protocol inside the function fnDHCPServer() and fnDHCPClient() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming / DHCP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

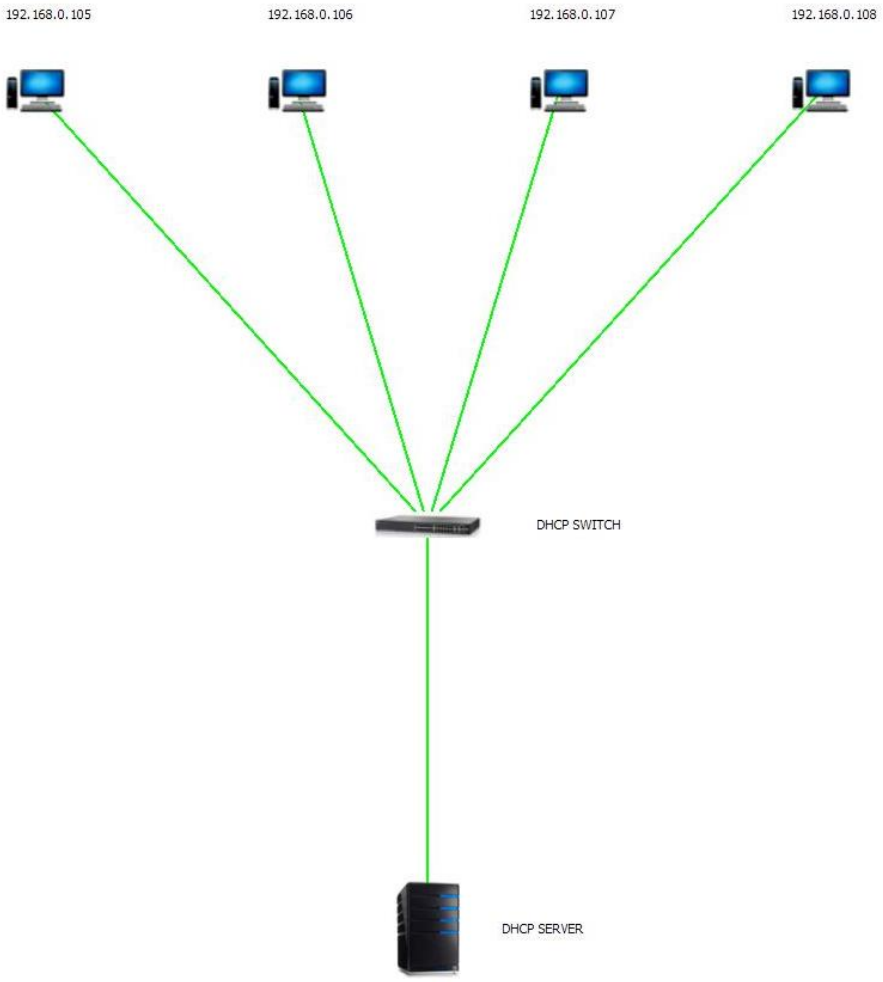
Objective - Find the Shortest Path using Dynamic Host Configuration Protocol.

How to Proceed? - The objective can be executed in NetSim by using the programming exercise available. In the **Programming** menu select**Dynamic Host Configuration Protocol.**

Sample Input -Follow the below given steps,

- Select number of clients.
- Set the Start Address and the End Address
- Click on **Run** button to execute.

Output - The Output that is obtained is given below,



10.5.16 Error Correcting Code - Hamming Code

Programming Guidelines

This section guides the user to link his/her own code for Hamming Code Generator to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input file format	Output file format
Parity=Odd Data=a Error_Position=7 Data_Bits_Original=01100001 Data_Bits_Error=00100001	'Output.txt ' file, the first line is the count of check bits to be introduced in the current hamming code application. The next 8 lines is the check bit calculation process, the first four lines are for input data and the next four lines are for the error data. 1 and 2 are the check bits that vary in the Hamming String. The last value is the actual position in the Hamming Code generated where the data bits has been changed. 4 - The number of check bits introduced

	3>5>7>9>11>13>0>0>- the check bit of position 1 of input data 3>6>7>10>11>1>1>- the check bit of position 2 of input data 5>6>7>12>13>0>0>- the check bit of position 4 of input data 9>10>11>12>13>0>0>- the check bit of position 8 of input data 3>5>7>9>11>13>0>0>- the check bit of position 1 of error data 3>6>7>10>11>1>1>- the check bit of position 2 of error data 5>6>7>12>13>1>1>- the check bit of position 4 of error data 9>10>11>12>13>1>1>- the check bit of position 8 of error data 4>8>12>- the check bit position whose value has been changed and the final error position.

Interface Source Code

To view the interface source code, go to

NetSim Installation path / src / Programming/ HammingCode.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To detect and correct the single bit error occurs in the transmission of data.

How to proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select **Error Correcting Code → Hamming Code**

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Parity need to be selected. Either Odd or Even can be selected.
- Data need to be entered. Maximum of 8 alphabets can be entered.
- Error Position needs to be selected. Based on the input, its values ranges.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Data Bits which if in the binary form is obtained. This Data Bits is obtained for Original Data entered as well as for the Error selected.
- Depending on the Parity selected, tables of Hamming String, Original Data and Error Data will be obtained.
- For the Error Data table to be obtained click on the Next button available in the output panel. When the Error Data table is obtained Error Position value is also obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.17 Error Detection Code - Cyclic Redundancy Check (CRC) - 12

Programming Guidelines

This section guides the user to Run his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_12 Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: 8CB 000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ Crc12.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC12.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming Menu** select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC12** as **Algorithm** from the list available.
    2. Under **Condition**, "**NoError**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.
- **For Error Case -** Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC12** as **Algorithm** from the list available.

2. Under **Condition**, "**Error**" should be selected.
3. Under **Input**, **Enter** the path of the file name to get its **CRC**.The file should be in "**.txt**" format which should not exceed 5000bytes.
4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txt file**" is received by the **Node2**. The message "**Data Frame** is **Flowing** from **Node1** to **Node2** with **No Error**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the **".txt file**" is received by the **Node2**. The message "**Data Frame** is **Flowing** from **Node1** to **Node2** with **Error**".

10.5.18 Error Detection Code - Cyclic Redundancy Check (CRC) – 16

Programming Guidelines

This section guides the user to link his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_16 Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: 0FCF 0000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ Crc16.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC16.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    5. **SampleMode** should be selected.
    6. **SelectCRC16** as **Algorithm** from the list available.
    7. Under **Condition**, "**No Error**" should be selected.
    8. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    9. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

- **For Error Case -** Follow the below given steps,
    10. **SampleMode** should be selected.
    11. **SelectCRC16** as **Algorithm** from the list available.
    12. Under **Condition**, "**Error**" should be selected.
    13. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    14. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **NoError**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **Error**".

10.5.19 Error Detection Code - Cyclic Redundancy Check (CRC) - 32

Programming Guidelines

This section guides the user to link his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_32 Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: DD8F598B 00000000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ Crc32.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC32.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC32** as **Algorithm** from the list available.
    2. Under **Condition**, "**NoError**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

- **For Error Case -** Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC32** as **Algorithm** from the list available.
    2. Under **Condition**, "**Error**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.

4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **NoError**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **Error**".

10.5.20 Error Detection Code - Cyclic Redundancy Check (CRC) – CCITT

Programming Guidelines

This section guides the user to link his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_CCITT Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: 92BF 0000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ CrcCcitt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC CCITT.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRCCCITT** as **Algorithm** from the list available.
    2. Under **Condition**, "**NoError**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

- **For Error Case -** Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRCCCITT** as **Algorithm** from the list available.
    2. Under **Condition**, "**Error**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.

4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **NoError**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the ".txt file" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **Error**".

10.5.21 Error Detection Code - Longitudinal Redundancy Check (LRC)

Programming Guidelines

This section guides the user to link his/her own code for Longitudinal Redundancy Check to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
'Input.txt' file has the original data bits, the type of parity (odd or even) and the error data bits. Parity=Odd (or Even) Data=Tetcos Data_Bits_Original=0101010001100101011101000110001 10110111101110011 Data_Bits_Error=0010101101000101010101000110001101 10111101110011	'Output.txt ' file, the First line is the LRC bits value for the original data bits. The Second line is the LRC bits value for the error data bits. 00111010 - LRC bits of the original data 01000101 - LRC bits of the error data

Interface Source Code

Interface source code written in C is given using this the user can write only the Longitudinal Redundancy Check inside the function fnLRC () using the variables already declared.To view the interface source code, go to

NetSim Installation path / src / Programming/ LRC.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Longitudinal Redundancy Check (LRC)

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Error Detecting Codes in that Longitudinal redundancy Check,

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Parity need to be selected. Either Odd or Even can be selected.
- Data need to be entered. Maximum of 8 alphabets can be entered.
- Error Position needs to be selected. The value ranges from **1** to **48**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Data Bits which if in the binary form is obtained. This Data Bits is obtained for Original Data entered as well as for the Error selected.
- Depending on the Parity selected, tables of LRC of Original Data Bits and LRC of Error Data Bits are obtained.
- In each of the above mentioned tables last row would contain the Parity.
- Error In Column(s) is obtained.

10.5.22 Framing Sequence – Bit Stuffing

Programming Guidelines

This section guides the user to link his/her own code for Bit Stuffing to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory.** The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File	Output File
The "Input.txt" file contains, Destination Address=Value Source Address=Value Data=Message CRC Polynomial=Value Error Status=0 or 1 Seed Value=Value Example: Destination Address=00011111 Source Address=00111111 Data=Hello CRC Polynomial=10010 Error Status=0 Seed Value=45	The "Output.txt" file contains, Message=Value> H=Value>e=Value>l=Value>l=Value>o=Value> Binary Values= Value > CRC Polynomial= Value > CheckSumSender= Value > <Stuffing> Destination Address= Value > Source Address= Value > Data= Value > <DeStuffing> Destination Address= Value > Source Address= Value > Data= Value > Error Status= Value > CheckSumReceiver= Value > Binary Values= Value > H= Value >e= Value >l= Value >l= Value >o= Value > Message= Value > Example:

	Message=Hello> H=72>e=101>l=108>l=108>o=111> Binary Values=01001000011001010110110001101100011011111> CRC Polynomial=10011> CheckSumSender=1110> <Stuffing> Destination Address=000111110> Source Address=001111101> Data=01001000011001010110110001101100011011111110> <DeStuffing> Destination Address=00011111> Source Address=00111111> Data=01001000011001010110110001101100011011111110> Error Status=0> CheckSumReceiver=0000> BinaryValues=01001000011001010110110001101100011011111> H=72>e=101>l=108>l=108>o=111> Message=Hello>

Interface Source Code

Interface source code written in C is given. Using this, the user can write only the functions **fnBitStuffing () and fnDeStuffing ()**, using the variables already declared.

To view the interface source code, go to **NetSim Installation path / src / Programming/ BitStuffing.c**

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Bit Stuffing technique.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Framing Sequence → Bit Stuffing.

Sample Inputs - In the Input panel,

- Sample mode should be selected.
- Fill in the HDLC Frame fields available

- Enter the 8 binary digits in the **Source Address** field.
- Enter the 8 binary digits in the **Destination Address** field.
- Enter data, with a maximum of 5 characters
- CRC polynomial will be chosen by default and select either **Error** or **No Error.**

- Then **Run** button needs to be clicked. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally by NetSim -

- Data will be converted into ASCII Values. ASCII Values will be converted into Binary Values.
- The Binary Value for CRC polynomial will be shown.
- Checksum will be calculated for the user data in **Sender** side.
- HDLC frame will be formed in Sender side and Bit Stuffing process is animated (Adding '0' for every consecutive **five** 1's).
- Then Destuffing process will be animated in **Receiver** side.
- Checksum will be calculated in receiver side.
- Again Binary Values will be converted into ASCII values.
- Finally the ASCII values will be converted into Data which the user entered.

.

10.5.23 Framing Sequence – Character Stuffing

Programming Guidelines

This section guides the user to link his/her own code for Character Stuffing to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File	Output File
The "Input.txt" file contains, Starting Delimiter=Value Destination Address=Value Source Address=Value Data=Value Checksum=Value Ending Delimiter=Value Example: Starting Delimiter=a Destination Address=eraerwbr Source Address=asdasdas Data=sdfgf Checksum=shfsdfsd Ending Delimiter=h	The "Output.txt" file contains, Stuffing> Destination Address=Value> Source Address= Value> Data= Value> Checksum= Value> DeStuffing> Destination Address= Value> Source Address= Value> Data= Value> Checksum= Value> Example: Stuffing> Destination Address=eraaerwbr> Source Address=aasdaasdaas>

	Data=sdfgf> Checksum=shhfsdfsd> DeStuffing> Destination Address=eraerwbr> Source Address=asdasdas> Data=sdfgf> Checksum=shfsdfsd>

Interface Source Code

Interface source code written in C is given. Using this, the user can write only the functions **fnCharacterStuffing () and fnDeStuffing ()**, using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ CharacterStuffing.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Character Stuffing technique.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Framing Sequence → Character Stuffing.

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Fill in the HDLC Frame fields available.
  - Starting Delimiter has to be filled in.
  - Enter the 8 characters in the Destination Address field.
  - Enter the 8 characters in the Source Address field.
  - Enter in Data field with a maximum of 8 characters.
  - Enter the 8 characters in the Check Sum field.
  - Ending Delimiter has to be filled in.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- HDLC Frame will be formed in **Sender** side.
- Character stuffing process will be animated in Sender Side.
- Then destuffing process will be animated in Receiver side.
- Once the sample experiment is done **Refresh** button can be **clicked** to create new samples.

10.5.24 Virtual Scheduling Algorithm

Programming Guidelines

This section guides the user to link his/her own code for Virtual Scheduling Algorithm to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Input File Content PCR=1 CDVT=0.200 Actual_Arriva_Time=0.100 Previous_Theoratical_Time=0.000	This gives us the format of 'Output. Txt' to which the user has to write his program Each value should be separated by a delimiter'>'. There should be no extra white spaces or blank lines. Example: Value1>Value2>Value3> Value1 – Cell conformation Flag, Value 2 – Next Cells Expected arrival Time Value 3 – Current cell's Actual arrival time. Note: The above convention to write into the 'Output.Txt' are mandatory.

	A sample file format is as follows:3>2.000>1.000>

Interface Source Code

Interface source code written in C is given using this the user can write only the Longitudinal Redundancy Check inside the **void main ()** using the variables already declared.To view the interface source code, go to

NetSim Installation path / src / Programming/ VSA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Virtual Scheduling Algorithm.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Virtual Scheduling Algorithm,

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Peak Cell Rate (PCR). Its value ranges from **1** to **10.**
- Select the Cell Delay Variation Tolerance (CDVT). Its value ranges from **0.1** to **0.9.**
- Enter in the Actual Arrival Time.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- A graph with the following information is obtained,
  - Cell Confirmation,
  - Actual Arrival Time,
  - Expected Time (TAT)

- Whether the Cell has been discarded or successfully received. This discarding or receiving of the cell depends on the Actual Arrival Time entered. Go through the 3 rules given in this program.

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.25 Address Mask

Programming Guidelines

This section guides the user to link his/her own code for Address mask to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
This gives us the contents of the 'Input. Txt' from which the user has to get the values for his program The Input File format IP Address> Sample Input text Format Let us consider how a given input (Data file and error index) is stored in the text and read. IP_Address=192.169.0.150 Prefix_Value=1	The Output File format Binary value of IP Address> Prefix value>Suffix value> Binary value of Address mask> Decimal value of Address mask>. Sample Output text Format 11000000 10101001 00000000 10010110 > 1>31> 10000000 00000000 00000000 0000000>128>0>0>0>

Interface Source Code

Interface source code written in C is given using this the user can write only thefnAddressMAsk () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ AddressMask.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of finding Address mask through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address
- Select the Prefix value
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Binary value of IP Address is obtained.
- Prefix, Suffix values are obtained
- Binary value of Address Mask is obtained
- Decimal value of Address Mask is obtained
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.26 Binary Conversion

Programming Guidelines

This section guides the user to link his/her own code for Binary conversion to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
This gives us the contents of the 'Input. Txt' from which the user has to get the values for his program The Input File format IP Address> Sample Input text Format IP_Address=192.168.0.100	The Output File format First Byte value>128>Quotient of First Byte/128>Remainder of First Byte/128> Previous remainder>64>Quotient of Previous remainder/64>Remainder of Previous remainder /64> Previous remainder>32>Quotient of Previous remainder/32>Remainder of Previous remainder /32> Previous remainder>16>Quotient of Previous remainder/16>Remainder of Previous remainder /16> Previous remainder>8>Quotient of Previous remainder/8>Remainder of Previous remainder /8>

	Previous remainder>4>Quotient of Previous remainder/4>Remainder of Previous remainder /4> Previous remainder>2>Quotient of Previous remainder/2>Remainder of Previous remainder /2> Previous remainder>1>Quotient of Previous remainder/1>Remainder of Previous remainder /1>. .same procedure for all bytes Binary value> Sample Output text Format 192>128>1>64> 64>64>1>0> 0>32>0>0> 0>16>0>0> 0>8>0>0> 0>4>0>0> 0>2>0>0> 0>1>0>0> 168>128>1>40> 40>64>0>40> 40>32>1>8> 8>16>0>8> 8>8>1>0> 0>4>0>0> 0>2>0>0> 0>1>0>0> 0>128>0>0> 0>64>0>0> 0>32>0>0> 0>16>0>0> 0>8>0>0> 0>4>0>0> 0>2>0>0> 0>1>0>0> 100>128>0>100>

	100>64>1>36> 36>32>1>4> 4>16>0>4> 4>8>0>4> 4>4>1>0> 0>2>0>0> 0>1>0>0> 11000000 10101000 00000000 01100100 >

Interface Source Code

Interface source code written in C is given using this the user can write only the fnBinaryConversion () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ BinaryConversion.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of Binary conversion through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Each bit is obtained one by one.
- Binary value of IP Address is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.27 Classless InterDomain Routing

Programming Guidelines

This section guides the user to link his/her own code for Classless Inter Domain Routing to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note:

The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
This gives the contents of the 'Input. Txt' from which the user has to get the values. The Input File contains: Starting_IP_Address=150.0.0.0 No_of_Networks=3 No_of_Hosts_in_Network_1=512 No_of_Hosts_in_Network_2=1024 No_of_Hosts_in_Network_3=2048	The Output file contains the First address, Last address, Mask, Value and Network Address of all the networks. First Address: The Starting Address of the Network Last Address: The last address in the Network. Mask: The Mask address (Mask should be 4 byte format). Number of Node: The number of nodes in the network.

Host_IP_Address=150.0.1.2	Value: The value will be either 0 or 1. 0 - The next line has the network address to which the given host belongs. 1 - Means that the host does not belong to any network. Network Address: If the Value is 1 then the network to which the host belongs is given. Note: Each data will have a separate line and each line will end with ">" A Sample Output File Format 150.0.0.0> 150.0.1.255> 255.255.254.0> 512> 150.0.4.0> 150.0.7.255> 255.255.252.0> 1024> 150.0.8.0> 150.0.15.255> 255.255.248.0> 2048> 0> 150.0.0.0> For each network there will be first address, last address, mask and number of hosts. Since the input is 3 networks, there are 3 set of outputs. The last two lines give the details about whether the host is present in any of the network.

Interface Source Code

Interface source code written in C is given using this the user can write only the CIDR inside the function fnCIDR () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ CIDR.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/1

Sample Scenarios:

Objective - To Implement Classless InterDomain Routing (CIDR).

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Classless InterDomain Routing,

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be **selected**.
- **Select** the **Starting IP Address** from the given range.
- **Select** the **No. of Networks** from the given range. Maximum of **6** networks can be selected.
- **Select** the **No. of Hosts** from the given range.
- **Click** on **Add** button to add the **No.ofHosts** onto the Hosts Field. The use of **Add** button depends on the **No. of Networks** selected. If a new No. of Hosts has to be added then remove button can be used.
- Select the **Host IP Address**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Based on the Starting IP Address a table is obtained.
- There are 4 columns available in the output table, i.e. First Address, Last Address, CIDR Mask and No. of Hosts.
- The First Address of the first network would be nothing but the Starting IP Address which is selected. No. of rows in the table depends on the No. of Networks selected.

- The Last Address depends on depends on the No. of Host selected.
- CIDR Mask is obtained internally by using the following formula,

  CIDR Mask = 32 - (log (No. of Host) / log (2))

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.28 Network Address

Programming Guidelines

This section guides the user to link his/her own code for Network Addresses to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
IP_Address=192.168.0.140 Prefix_Value=1	The Output File format Binary value of IP Address> Prefix value> Binary value of Address mask> Binary value of Network Address> Decimal value of Network Address>. Sample Output text Format 11000000 10101000 00000000 10001100 >1> 10000000 00000000 00000000 00000000> 10000000 00000000 00000000 00000000> 128>0>0>0>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnNetworkAddress () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ NetworkAddress.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of finding Network Address through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address
- Select the Prefix value
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Binary value of IP Address is obtained.
- Binary value of Address Mask is obtained
- Binary value of Network Address is obtained
- Decimal value of Network Address is obtained
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.29 Special Addresses

Programming Guidelines

This section guides the user to link his/her own code for Special Addresses to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
IP_Address=192.168.0.160 Prefix_Value=5	The Output File format Binary value of IP Address> 32>Prefix value>Suffix value> Binary value of Prefix Part> Binary value of Suffix Part> Condition number in the table>condition> Type of address>. Sample Output text Format 11000000 10101000 00000000 10100000 > 32>5>27> 11000>000 10101000 00000000 10100000> It doesn't Satisfy the four conditions> Not a special address>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnSpecialAddress () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ HammingCode.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of Special Addresses through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address in slash notation
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Binary value of IP Address is obtained.
- Binary value of Address Mask is obtained
- Prefix, Suffix values are obtained
- Binary value of Prefix part and Suffix part are obtained
- Type of address is obtained
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples

10.5.30 Subnetting

Programming Guidelines

This section guides the user to link his/her own code for Sub-netting to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
Class=B Network_Address=128.0.0.0 No_of_Subnets=2 Host_IP_Address=128.0.0.89	The Output file contains the Default mask (in decimal), Network portion of Default mask (in binary), Host portion of Default mask (in binary), Subnet mask bit value, Network portion of Subnet mask (in binary), Host portion of Subnet mask (in binary), Subnet mask (in decimal), number of zero in host portion, number of host in each subnet, Host address, subnet number of the given host, subnet address of the given host and Subnet address, Starting address, Ending address, Broadcast address of each subnet. The Output File format Default mask (in decimal)>

	Network portion of Default mask (in binary)>Host portion of Default mask (in binary)> Subnet mask bit value> Subnet mask (in decimal)> Network portion of Subnet mask (in binary)>Host portion of Subnet mask (in binary)> Number of zero in host portion,>Number of host in each subnet> Host address>Subnet number of the given host >Subnet address of the given host> Subnet address>Starting address>Ending address>Broadcast address > . . .until number of subnet is reached Sample Output text Format 255.255.0.0 11111111 11111111>00000000 00000000> 1> 255.255.128.0> 11111111 11111111 1>0000000 00000000> 15>32766> 128.0.0.89>1>128.0.0.0> 1>128.0.0.0>128.0.0.1>128.0.127.254>128.0.127.255> 2>128.0.128.0>128.0.128.1>128.0.255.254>128.0.255.255>

Interface Source Code

Interface source code written in C is given using this the user can write only the Leaky bucket algorithm inside the function fnSubnet () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Subnet.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of Subnetting through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Subnetting,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Class Name
- Select the Network Address from the given list.
- Select the Number of Subnets from the given list.
- Select the Host IP Address.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Subnet mask value is obtained.
- Number of host in the each subnet is obtained
- Subnet address, Starting address, Ending address and Broadcast address are obtained.
- Subnet address of the given Host IP address is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.31 EUI-64 Interface Identifier

Programming Guidelines

This section guides the user to link his/her own code for EUI 64 Interface Identifier to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
IPV6_Network_Id=2000:FE21:5931:72C3 MAC_Address=11-11-FF-FF-12-34	The Output File format MAC Address> First part of MAC Address>Second part of MAC Address FF-EE appended Address> First byte value>Binary value of First byte> 7th bit value>Complement value of 7th bit Complemented binary value of First byte>Hexa decimal value of complemented binary value> Interface Id value> Interface Id in colon notation> IPV6 prefix value> IPV6 Address>

	Sample Output text Format 11-11-FF-FF-12-34> 11-11-FF>FF-12-34> 11-11-FF-FF-FE-FF-12-34> 11>00010001> 0>1> 00010011>13> 13-11-FF-FF-FE-FF-12-34>1311:FFFF:FEFF:1234> 2000:FE21:5931:72C3> 2000:FE21:5931:72C3:1311:FFFF:FEFF:1234>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnEUI64 () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ IPV6EUI64.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of EUI 64 Interface Identifier through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select IPV6 Addressing → EUI 64 Interface Identifier,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IPV6 Network Id (IPV6 Prefix)
- Enter the MAC Address.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- MAC Address is divided into two parts
- FF-FE value is appended between the two parts
- Interface Identifier is found by complementing 7th bit of firs byte
- IPV6 Network Id (IPV6 prefix) and Interface Identifier are combined to produce IPV6 address
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.32 IPV6 Host Addresses

Programming Guidelines

This section guides the user to link his/her own code for IPV6 Host Addresses to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
IPV6_Address=1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD Prefix_Length=12	The Output File format IPV6 Address> Binary value of IPV6 Address> Prefix part of IPV6 Address>Suffix part of IPV6 Address> Prefix part of Starting Address>Suffix part of Starting Address> Starting Address in hexa decimal notation> Prefix part of Ending Address>Suffix part of Ending Address> Ending Address in hexa decimal notation> Sample Output text Format 1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>

	000100010001000100100010001000100011001100110011 010001000100010010101010101010101011101110111011 110011001100110011011101110111011>12>116> 000100010001>000100100010001000100011001100110011001 101000100010001001010101010101010101110111011101 111001100110011001101110111011101> 000100010001>000000000000000000000000000000000000 000000000000000000000000000000000000000000000000 000000000000000000000000000000000000> 1110:0000:0000:0000:0000:0000:0000:0000> 000100010001>111111111111111111111111111111111111 111111111111111111111111111111111111111111111111 111111111111111111111111111111111111> 111F:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnHostAddresses () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ IPV6HostAddress.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of IPV6 Host Addresses through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select IPV6 Addressing →IPV6 Host Addresses,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Enter the IPV6 Address.
- Select the Prefix length.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- IPV6 is separated into two parts: Network Id bits (prefix) and Host Id bits (suffix)
- Starting address of the network is found by replacing each bits of suffix part with zero
- Ending address of the network is found by replacing each bits of suffix part with one
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.33 IPV6 Subnetting

Programming Guidelines

This section guides the user to link his/her own code for IPV6 Subnetting to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
IPV6_Address=1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD Prefix_Length=12 Number_of_Subnets=2 Number_of_Level=2 Subnets_in_Level1=2 Subnets_in_Level2=4	The Output File format Binary value of IPV6 Address> Prefix part of IPV6 Address>Suffix part of IPV6 Address> Number of subnets>number of subnet msk bit> Prefix part of IPV6 Address>Subnet Id part of IPV6 address>Suffix part of IPV6 Address> Prefix part of IPV6 Address>Subnet Id part of IPV6 address>Suffix part of IPV6 Address> Prefix part of Level 1 's first subnet Address> Subnet Id part of Level 1 's first subnet Address> Suffix part of Level 1 's first subnet Address> Hexa decimal value of Level 1 's first subnet address>prefix length of Level 1 's first subnet> . .

	until number of subnet reached in the first level First level's subnet number>Prefix part of Level 2 's first subnet Address> Subnet Id part of Level 2 's first subnet Address> Suffix part of Level 2 's first subnet Address> Hexa decimal value of Level 2 's first subnet address>prefix length of Level 2 's first subnet> . . until the number of subnets reached in the second level of first level's subnet . . until the number of subnets reached in the first level Sample Output text Format 1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD> 000100010001100100100010001000100011001100110100010001000100101010101010101010111011101111001100110011001101110111011101> 000100010001>000100100010001000100011001100110011010001000100010010101010101010101011101110111011100110011001100110111011101110111>2>1> 000100010001>0>0010010001000100010001100110011001101000100010001001010101010101010101110111011101111001100110011001101110111011101>> 000100010001>0>0010010001000100010001100110011001101000100010001001010101010101010101110111011101111001100110011001101110111011101>1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>13> 000100010001>1>0010010001000100010001100110011001101000100010001001010101010101010101110111011101111001100110011001101110111011101>1119:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>13> 0>1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 0>1113:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 0>1115:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 0>1117:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>1119:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>111B:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>111D:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>111F:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnIPV6Subnetting () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ IPV6Subnetting.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of IPV6 Subnetting through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select IPV6 Addressing →IPV6 Subnetting,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Enter the IPV6 Address.
- Select the Prefix length.
- Select the number of subnets value
- Click the Add value button to add the levels
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- IPV6 is separated into two parts: prefix and suffix
- Number of subnet mask id is calculated
- Then Subnet Id portion is derived from suffix part
- Using the subnet id portion, subnets are calculated
- Like that, subnets in the all levels are calculates by using the above steps
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.34 Leaky Bucket Algorithm

Programming Guidelines

This section guides the user to link his/her own code for Leaky Bucket Algorithm to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
Output_Capacity=100 Buffer_Size=100 Input_Capacity=100,100,500,100,100,100,100,100,100,100,	'Output.txt' file contains three lines of data. The data format in the output file should be as follows: Output rate at 1st second >Output rate at 2nd second>……..> Discard rate at 1st second >Discard rate at 2nd second>……..> Total number of seconds taken > The data should be stored in the file with a delimiter ">" in between. Sample File data

	100>100>100>100>100>100>100>100>100>100>100> 0>0>300>0>0>0>0>0>0>0>0> 11>

Interface Source Code

Interface source code written in C is given using this the user can write only the Leaky bucket algorithm inside the function fnLBA() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ LBA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To understand the concept of Leaky Bucket Algorithm(LBA) through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming menu user has to select Leaky Bucket Algorithm.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Output Capacity from the given list of values. The value ranges from **100** to **1000.**
- Select the Size of Buffer from the given list of values. The value ranges from **100** to **1000.**
- Enter in the Input Capacity in the fields provided. The range that can be entered is from **100** to **1000.** These values will be plotted in the graph on the right hand side panel.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Once the **Run** button is clicked a graph with Output and Discard Rate is obtained.
- The graph explains in detail as in what is the Output and Discard Packet Rate for the given inputs.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.35 Multi Level Multi Access

Programming Guidelines

This section guides the user to link his/her own code for Multi-Level Multi Access Protocol to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File	Output File
Number_of_Nodes=2 Node1_Address=250 Node2_Address=500	'Output.txt' file has the decade's number, decade's type, and actual decades, node addresses Decade No>Decade Type>Decade>aderess1>adress2>… Sample Output: 0>0>0000100100> 1>1>0000000001> 2>2>0000000001>500> 3>1>0000100000> 4>2>0000000001>250>

Interface Source Code

Interface source code written in C is given using this the user can write only the Multi-Level Multi Access Protocol inside the function fnMLMA () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ MLMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Multi Level Multi Access (MLMA).

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Multi-Level Multi -Access Protocol.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Number of Nodes need to be **selected**. The value ranges from **2** to **15**.
- Node Address has to be entered and added onto the Node Address field. The Number of Nodes selected and the Number of Addresses added into the Address field should match.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- An Output table is formed.
- On the right hand panel an animation would play. This shows that higher the address given to the node higher the priority that node gets. Hence the node with the higher address would transmit the data first.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.36 Code Division Multiple Access

Programming Guidelines

This section guides the user to link his/her own code for Code Division Multiple Access to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory.**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The Temporary Directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
The "Input.txt" file contains, NumberofTransmittingMobilestations=Value> MobileStation=Value>Data=Value>Code=Value> MobileStation=Value>Data=Value>Code=Value> MobileStation=Value>Data=Value>Code=Value> Example: NumberOfTransmittingMobileStations=2>	No. of Transmitting MobileStations=Value> MobileStation=Value>Data=Value>code=Value>Encode=Value>Signal=Value> MobileStation=Value>Data=Value>code=Value>Encode=Value>Signal=Value> MobileStation=Value>Data=Value>code=Value>Encode=Value>Signal=Value Interference Pattern=Value> MobileStation=Value>code=Value>Decode=Value>Data=Value> MobileStation=Value>code=Value>Decode=Value>Data=Value>

MobileStation=1>Data=10101010> Code=1 1 1 1 1 1 1 1> MobileStation=2>Data=11110000> Code=1 -1 1 -1 1 -1 1 -1>	MobileStation=Value>code=Value>Decode=Value> Data=Value> Example: NumberofMobileStations=2> MobileStation=1>Data=1 0 1 0 1 0 1 0 >Code=1 1 1 1 1 1 1 1 >Encode=1 -1 1 -1 1 -1 1 -1 >Signal=1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 > MobileStation=2>Data=1 1 1 1 0 0 0 0 >Code=1 -1 1 -1 1 -1 1 -1 >Encode=1 1 1 1 -1 -1 -1 -1 >Signal=1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 > InterferencePattern=2 0 2 0 2 0 2 0 0 -2 0 -2 0 -2 0 -2 2 0 2 0 2 0 2 0 0 -2 0 -2 0 -2 0 -2 0 2 0 2 0 2 0 2 -2 0 -2 0 -2 0 -2 0 0 2 0 2 0 2 0 2 -2 0 -2 0 -2 0 -2 0 > MobileStation=1>Code=1 1 1 1 1 1 1 1 >Decode=8 -8 8 -8 8 -8 8 -8 >Data=1 0 1 0 1 0 1 0 > MobileStation=2>Code=1 1 1 1 1 1 1 1 >Decode=8 8 8 8 -8 -8 -8 -8 >Data=1 1 1 1 0 0 0 0 >

Interface Source Code

Interface source code written in C is given.Using this, the user can write only the function **fnEncode ()** and **fnDecode ()**, using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ CDMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Code Division Multiple Access technique

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming select Multiple Access Technology → Code Division Multiple Access.

Sample Inputs - In the Input panel the following steps need to be done,

- **Sample Mode** should be selected
- In default, the **Encode** operation will be enabled.
- Choose the **Number of Transmitting Mobile Stations.**
- Enter the **Binary Datas** as per the limits.
- Choose the respective **Codes** for each Mobile Station.

Number of Transmitting Mobile Stations →3 (Maximum of 8 Mobile Stations)

Mobile Station	Binary Data(1Byte)	Code
1	01110111	1 1 1 1 1 1 1 1
2	00110011	1 -1 1 -1 1 -1 1 -1
3	11011101	1 1 -1 -1 1 1 -1 -1

- Then **Run** button needs to be **clicked**. **Refresh** button can be used, if new inputs have to be given.

Output -The Output for the above sample is as follows,

- In **Transmitter side**, the values of each Mobile Station's **Binary Data, Code, Encode and Signal** are shown, if the user clicks the Mobile Station.
- The **Signals** will be broadcast from each Mobile Stations.
- The **Interference Pattern** will be formed in **Base Station** and if the user clicks the Base Station, interference Pattern will be displayed.
- **Decode mode** needs to be **selected.**
- In **Receiver side**, the **Interference Pattern** will be broadcasted to each **Mobile Station** from the **Base Station**.
- In **Receiver side,** the values of each Mobile Station's **Interference Pattern**, **Code, Decode and Binary Data** are shown, if the user clicks the Mobile Station.
- Once the sample experiment is done **Refresh** button can be **clicked** to create new samples.

The Values are showed in Table.

Transmitter Side

Mobile Station	Binary Data(1Byte)	Code	Encode	Signal
1	01110111	1 1 1 1 1 1 1 1	-1, 1, 1, 1, -1, 1, 1, 1	-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	00110011	1 -1 1 -1 1 -1 1 -1	-1, -1, 1, 1, -1, -1, 1, 1	-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1
3	11011101	1 1 -1 -1 1 1 -1 -1	1, 1, -1, 1, 1, 1, -1, 1	1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 1 1 -1 -1 1 1 -1 -1

In Base Station

Mobile Station	Signal
1	-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1
3	1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1

Interference Pattern	1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1

Receiver Side

Mobile Station	Interference Pattern	Code	Decode	Binary Data(1Byte)
1	1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1	1 1 1 1 1 1 1 1	8 8 8 8 -8 8 8 8	01110111
2	1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1	1 -1 1 -1 1 -1 1 -1	-8 -8 8 8 -8 -8 8 8	00110011
3	1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1	1 1 -1 -1 1 1 -1 -1	8 8 -8 8 8 8 -8 8	11011101

10.5.37 Time Division Multiple Access

Programming Guidelines

This section guides the user to link his/her own code for Time Division Multiple Access to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory.**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The Temporary Directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
The "Input.txt" file contains, Bandwidth=Value>No. of time slots=Value>Time slot Length=Value>Guard Interval=Value> Example: BandWidth=100>TimeSlot=1>TimeSlotLength=100>GuardInterval=0.0>	Guard Interval =Value>Bandwidth=Value>start time=Value >end time=Value>nouser=Value> channel no=Value>Bandwidth=Value>start time=Value>End time=Value>user=Value> Guard Interval =Value>Bandwidth=Value>start time=Value >end time=Value>nouser=Value> Guard Interval =Value>Bandwidth=Value>start time=Value >end time=Value>nouser=Value> channel no=Value>Bandwidth=Value>start time=Value>End time=Value>user=Value> . . and so on.

	The value of the GuardInterval is -1 and the value of the no user is 0 Example: GuardInterval=-1>Bandwidth=100>StartTime=0.0>EndTime=0.0>nouser=0> Channelno=1>Bandwidth=100>StartTime=0.0>EndTime=100.0>user=1> GuardInterval=-1>Bandwidth=100>StartTime=100.0>EndTime=100.0>no user=0>

Interface Source Code

Interface source code written in C is given.Using this, the user can write only the function **fnTDMA()**, using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ TDMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Time Division Multiple Access technique.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Multiple Access Technology → Time Division Multiple Access.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Enter the Value of the **Bandwidth**.
- Enter the Value of the **No. of Time Slots.**
- Enter the Value of the **TimeSlotLength.**
- Enter the Value of the **GuardInterval**

Note:

The Values entered should be within the range.

Bandwidth(kHz)	200
No. of TimeSlots	3
TimeSlotLength(µs)	100
GuardInterval(µs)	10

- Then **Run** button need to be **clicked**. **Refresh** button can be used, if new Inputs have to be given.

Output -The Output for the above sample is as follows,

- Bandwidth is divided into Time slots called channel.
- The first channel is allocated to first mobile station, second channel is allocated to second mobile station and so on.
- The first Mobile Station will access the medium in first channel time slot.
- The Second Mobile Station will access the medium in second channel time slot, and so on.
- During the guard interval, No mobile station will access the medium. This is used to avoid the collision or interference.

Finally the table will be showed.

- Channel no. is obtained.
- Bandwidth value is obtained.
- Start Time and End Time of the Guard Interval is obtained
- Start Time and End Time of the channel is obtained.
- Which Mobile Station is accessing the allocated time slot is obtained.

Channel number	Bandwidth(kHz)	Time(µs)	Mobile Station
0	200	0.0 – 5.0	0
1	200	5.0 – 95.0	1
0	200	95.0 – 100.0	0
0	200	100.0 – 105.0	0
2	200	105.0 – 195.0	2
0	200	195.0 – 200.0	0

0	200	200.0 – 205.0	0
3	200	205.0 – 295.0	3
0	200	295.0 – 300.0	0
0	200	300.0 – 305.0	0
1	200	305.0 – 395.0	1
.			
.			
.			

10.5.38 Orthogonal Frequency Division Multiple Access

Programming Guidelines

This section guides the user to link his/her own code for Orthogonal Frequency Division Multiple Access to NetSim.

Pre-conditions

The user program should read the input scenario from text file named '**Input**' with extension txtwhich is in **Temporary Directory.**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txtin **Temporary Directory.**

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
The following lines will be in Input.txt file. Number of users = Value User1 =Value User2 =Value User3 =Value User4 =Value Number of sub carrier per user = Value Example: Number of users = 4 User1 =111111111111 User2 =010101010101	Parallel Data Conversion User 1 Value User 2 Value User 3 Value User 4 Value Bit Mapping User 1 Value User 2 Value

User3 =101010101010 User4 =000000000000 Number of sub carrier per user = 4	User 3 Value User 4 Value SubCarrier Mapping User 1 Value User 2 -Value User 3 Value User 4 Value SubCarrier Addition User 1 Value User 2 Value User 3 Value User 4 Value Example: Parallel Data Conversion User 1 1 1 1 1 1 1 1 1 1 1 1 1 User 2 0 1 0 1 0 1 0 1 0 1 0 1 User 3 1 0 1 0

	1	0	1	0		
	1	0	1	0		
	User 4					
	0	0	0	0		
	0	0	0	0		
	0	0	0	0		
	Bit Mapping					
	User 1					
	1	1	1	1		
	1	1	1	1		
	1	1	1	1		
	User 2					
	-1	1	-1	1		
	-1	1	-1	1		
	-1	1	-1	1		
	User 3					
	1	-1	1	-1		
	1	-1	1	-1		
	1	-1	1	-1		
	User 4					
	-1	-1	-1	-1		
	-1	-1	-1	-1		
	-1	-1	-1	-1		
	SubCarrier Mapping					
	User 1					
	sin(2PIf0t)		sin(2PIf1t)		sin(2PIf2t)	sin(2PIf3t)
	sin(2PIf0t)		sin(2PIf1t)		sin(2PIf2t)	sin(2PIf3t)
	sin(2PIf0t)		sin(2PIf1t)		sin(2PIf2t)	sin(2PIf3t)
	User 2					
	-sin(2PIf4t)		sin(2PIf5t)		-sin(2PIf6t)	sin(2PIf7t)
	-sin(2PIf4t)		sin(2PIf5t)		-sin(2PIf6t)	sin(2PIf7t)
	-sin(2PIf4t)		sin(2PIf5t)		-sin(2PIf6t)	sin(2PIf7t)
	User 3					
	sin(2PIf8t)		-sin(2PIf9t)		sin(2PIf10t)	-sin(2PIf11t)
	sin(2PIf8t)		-sin(2PIf9t)		sin(2PIf10t)	-sin(2PIf11t)

	sin(2PIf8t) -sin(2PIf9t) sin(2PIf10t) -sin(2PIf11t) User 4 -sin(2PIf12t) -sin(2PIf13t) -sin(2PIf14t) -sin(2PIf15t) -sin(2PIf12t) -sin(2PIf13t) -sin(2PIf14t) -sin(2PIf15t) -sin(2PIf12t) -sin(2PIf13t) -sin(2PIf14t) -sin(2PIf15t) SubCarrier Addition User 1 sin 2PI(f0+f1+f2+f3)t sin 2PI(f0+f1+f2+f3)t sin 2PI(f0+f1+f2+f3)t User 2 sin 2PI(-f4+f5-f6+f7)t sin 2PI(-f4+f5-f6+f7)t sin 2PI(-f4+f5-f6+f7)t User 3 sin 2PI(f8-f9+f10-f11)t sin 2PI(f8-f9+f10-f11)t sin 2PI(f8-f9+f10-f11)t User 4 sin 2PI(-f12-f13-f14-f15)t sin 2PI(-f12-f13-f14-f15)t sin 2PI(-f12-f13-f14-f15)t

Interface Source Code

Interface Source code written in C is given. Using this, the user can writeonly the fnOFDMA() and fnSubcarrierAddition() functions, using the variables already declared.To view the interface source code, go to

NetSim Installation path/src/Programming/OFDMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Orthogonal Frequency Division Multiple Access technique

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming select Multiple Access Technology →OFDMA

Sample Inputs - In the Input panel,

- Sample Mode should be selected.
- The value of **Number of Users** needs to be selected.
- Enter the **Binary Data**for each user.
- **Number of SubCarriers per User** needs to be selected.
- Then **Run** button needs to be clicked. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally by NetSim

- The values of each user**Binary Data, Parallel Conversion, Bit Mapping, SubCarrier Mapping**&**SubCarrier Addition and Transmission** will be shown in the right hand side of the screen by animation.
- Once the sample experiment is done **Refresh** button can be **clicked** to create new samples.

10.5.39 PC to PC Communication - Socket Programming TCP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

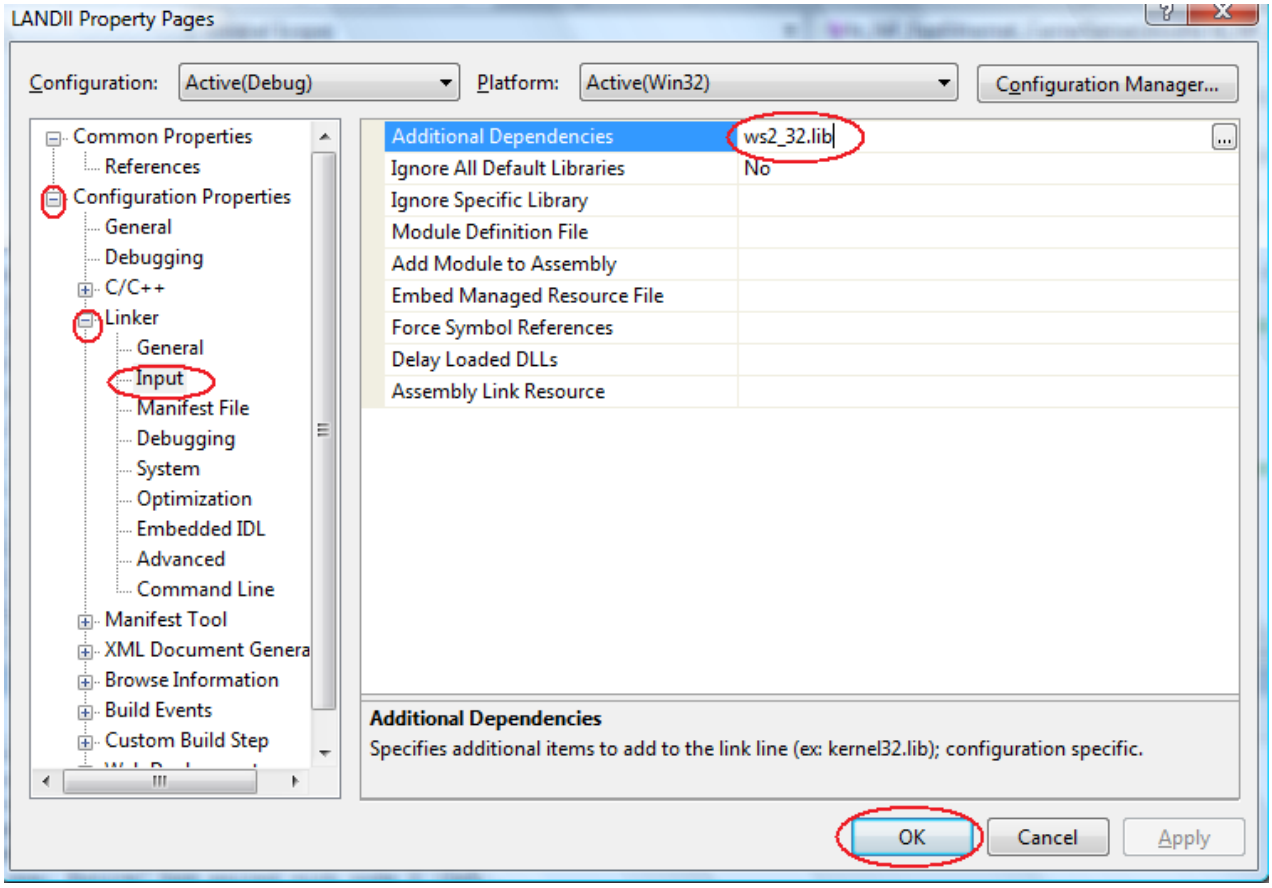
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

- For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

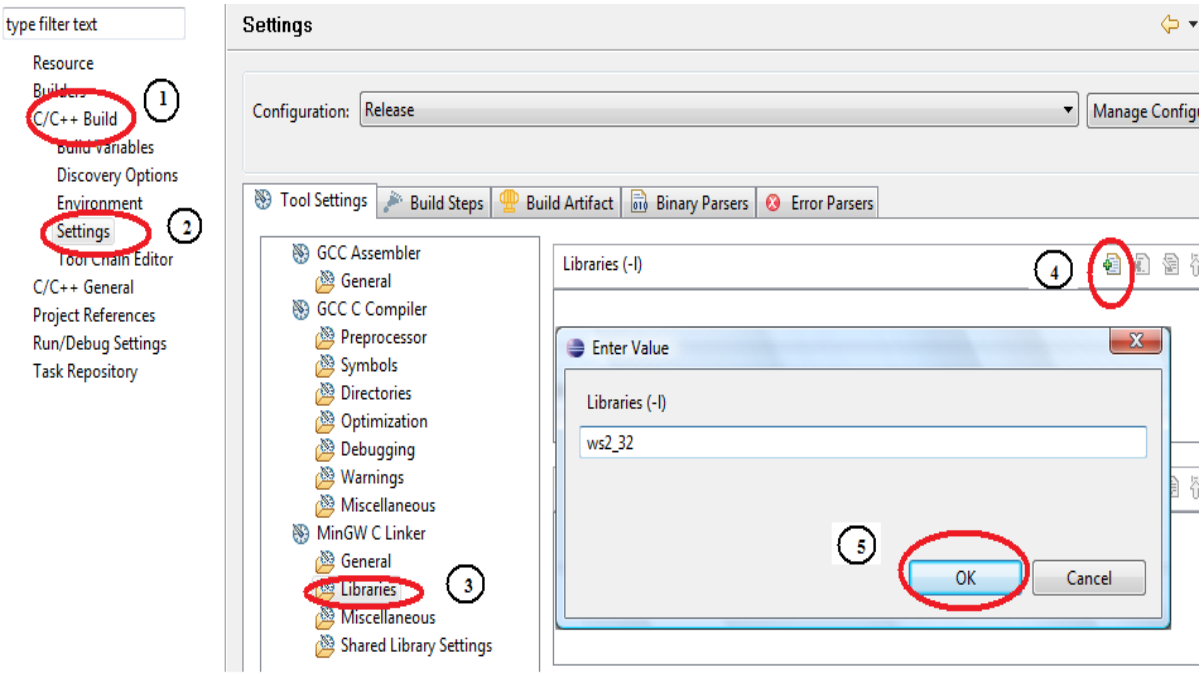
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

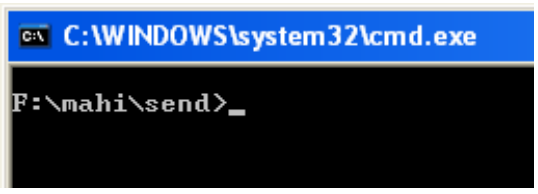
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

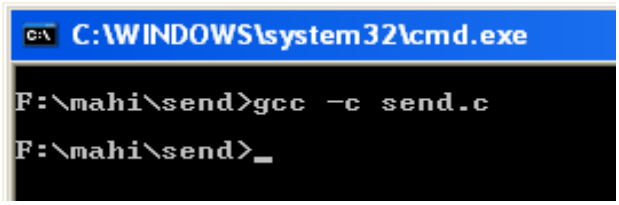
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

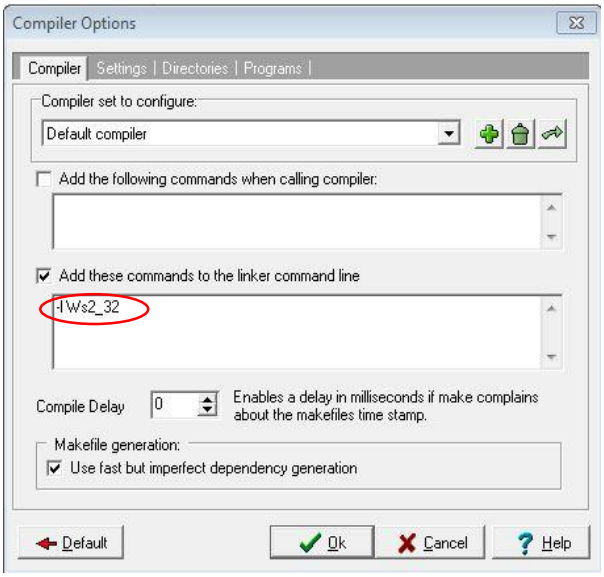
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools→compiler options and add -lws2_32 as follows before executing:



Input File Format	**Output File Format**
The data format in the input file is as follows, **Server Input** Protocol=TCP Operation=Server **Client** Protocol=TCP Operation=Client Destination IP=192.168.0.132 <Client IP:-Message> Ex: 192.168.0.2:- Requesting for the connection	Write the received message Ex:192.168.0.2:- Requesting for the connection

Interface Source Code

Interface source code written in C is given using this the user can write only the TCP Client () function. To view the interface source code, go to

NetSim Installation path / src / Programming/ SocketTCP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.40 PC to PC Communication - Socket Programming UDP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions`

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

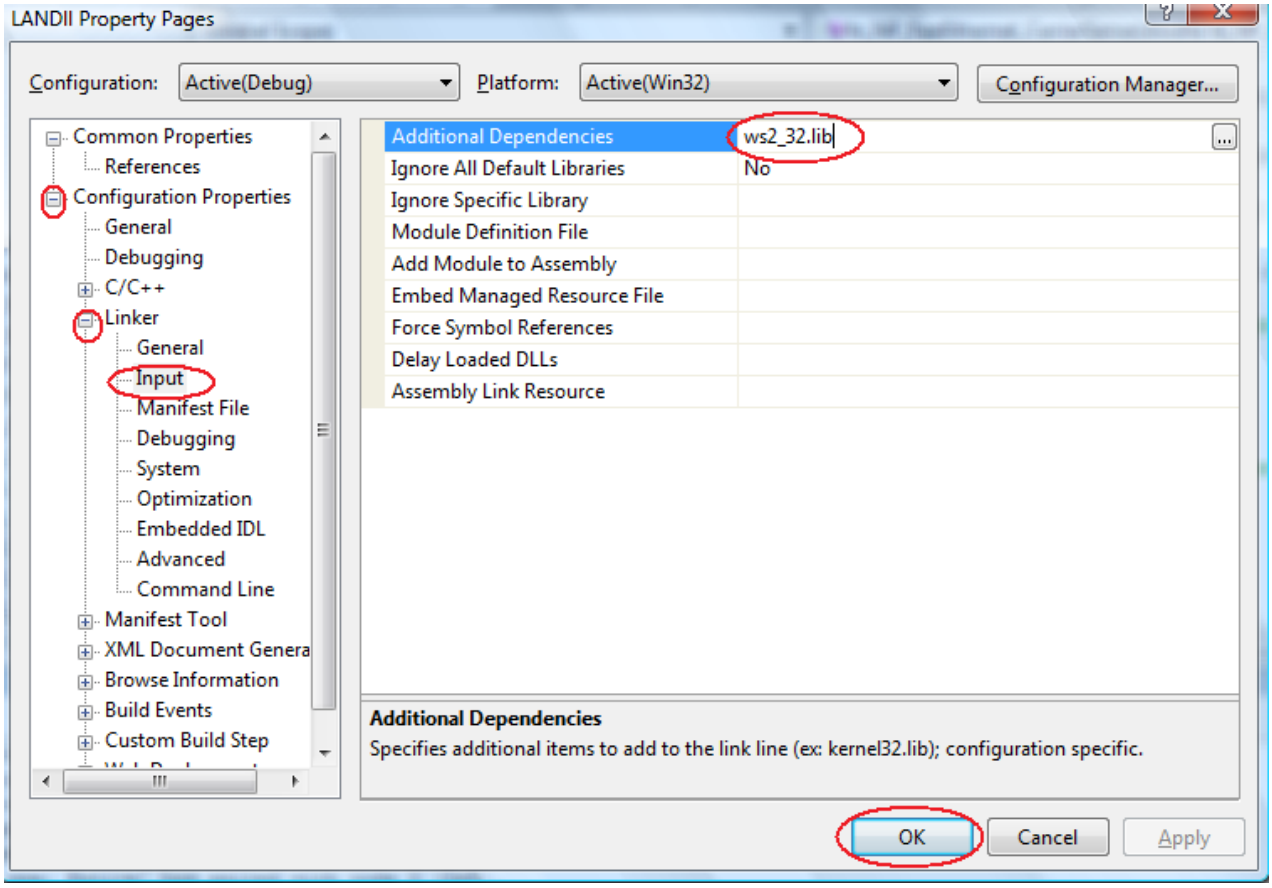
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

- For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

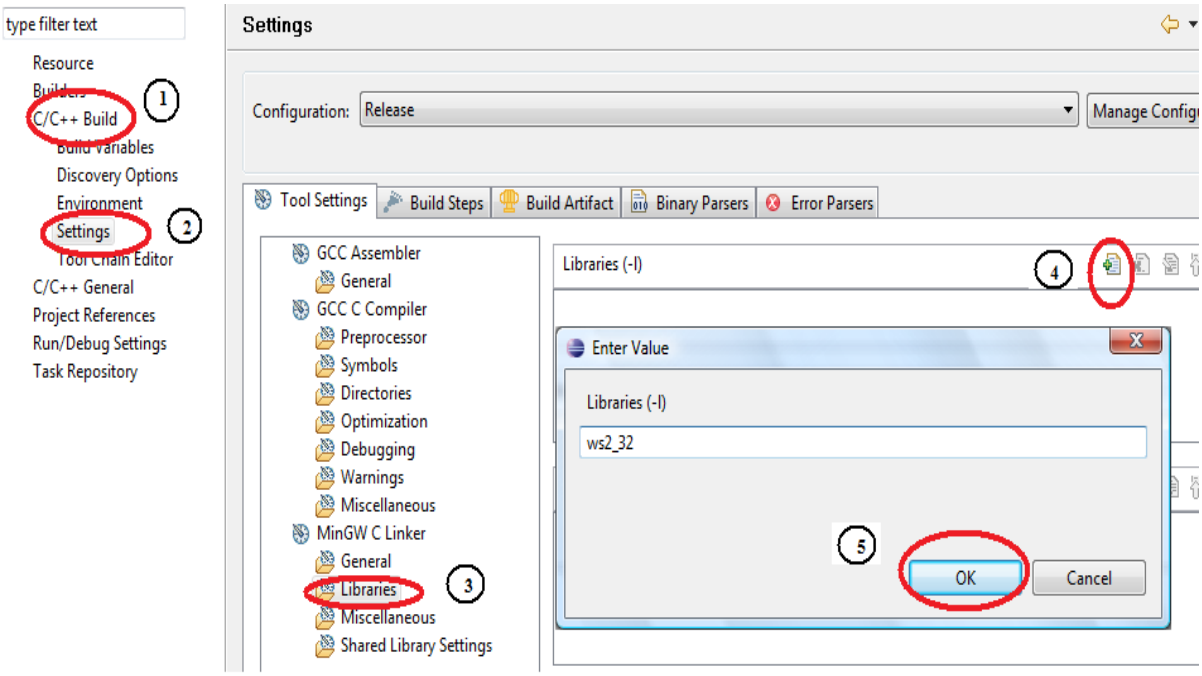
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

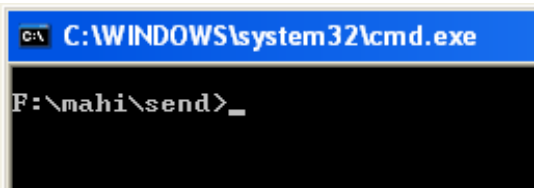
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

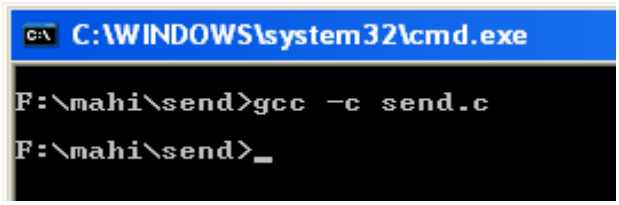
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

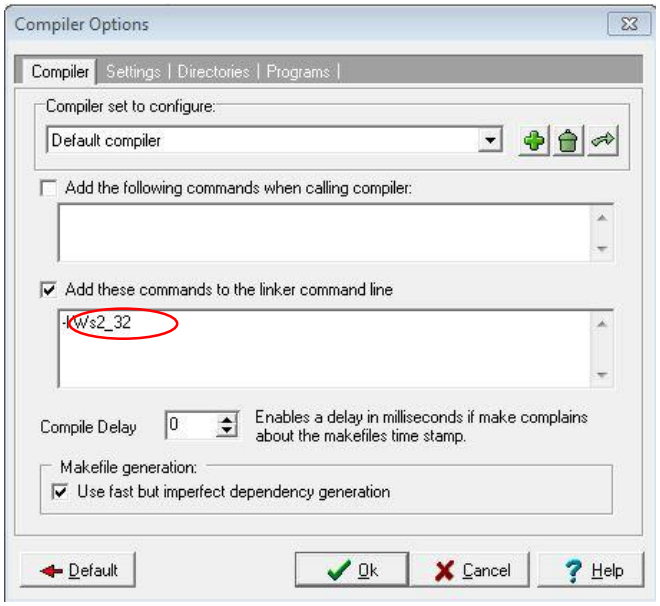
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools→compiler options and add -lws2_32 as follows before executing:



Input File Format	**Output File Format**
The data format in the input file is as follows,. **Server Input** Protocol= UDP Operation=Server **Client** Protocol=UDP Operation=Client Destination IP=192.168.0.132 <Client IP:-Message>	Write the received message Ex:192.168.0.2:- Requesting for the connection

Ex: 192.168.0.2:- Requesting for the connection	

Interface Source Code

Interface source code written in C is given using this the user can write only the UDP () functions. To view the interface source code, go to

NetSim Installation path / src / Programming/ SocketUDP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and Receive the data using PC to PCCommunication - SocketProgramming

How to Proceed? -Two Systems are required to perform this experiment. When one System is in the **ServerMode** other should be in the **Client Mode**. The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**PC to PC Communication → Socket Programming**.

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** a **Protocol** from the following,
  - **TCP**
  - **UDP**
- Under **Operation**, **Server** or **Client** should be selected.
- On the Right hand side panel there is an **InputField** "**Enter Data to be Transmitted**", where **Data** needs to be typed in. This **Data** is later sent to the **ReceiverSystem**. Here **Type** "**Hello**".
- Under **Input** there are two things,
  - When **Operation** is **Client**, then the **Server'sIPAddress** (192.168.1.2) should be given in the **Server** IP **Address** field.
  - When **Operation** is **Server**, then the **Server'sIPAddress** (192.168.1.1) would be automatically filled in the **Local IP Address** field.

Sample Output:

- On the Right hand side panel there is an **InputField** "**ReceivedData**", where **Data** would get displayed. "**Hello**" is the **Data** that is received from the **ClientSystem**.

TCP

- First the **Server** should **click** on the **Run** button after which the **Client** should click on the **Run** button to **Create the socket**
- **Client** should click on the **Connect** button to **establish the connection with server**.
- The **Client** should click on the **Send** button to transmit the data to the **Server.**
- The **Client** should click on the **Close** button to terminate the Connection with **Server.**
- If the **Data** is successfully transmitted then the **SentData** would be **Received** in the **ServerSystem**.

UDP

- First the **Server** should **click** on the **Run** button after which the **Client** should click on the **Run** button to **Create the socket**
- The **Client** should click on the **Send** button to transmit the data to the **Server.**
- The **Client** should click on the **Close** button to terminate the Connection with **Server.**
- If the **Data** is successfully transmitted then the **SentData** would be **Received** in the **ServerSystem**.

10.5.41 PC to PC Communication – Chat Application TCP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

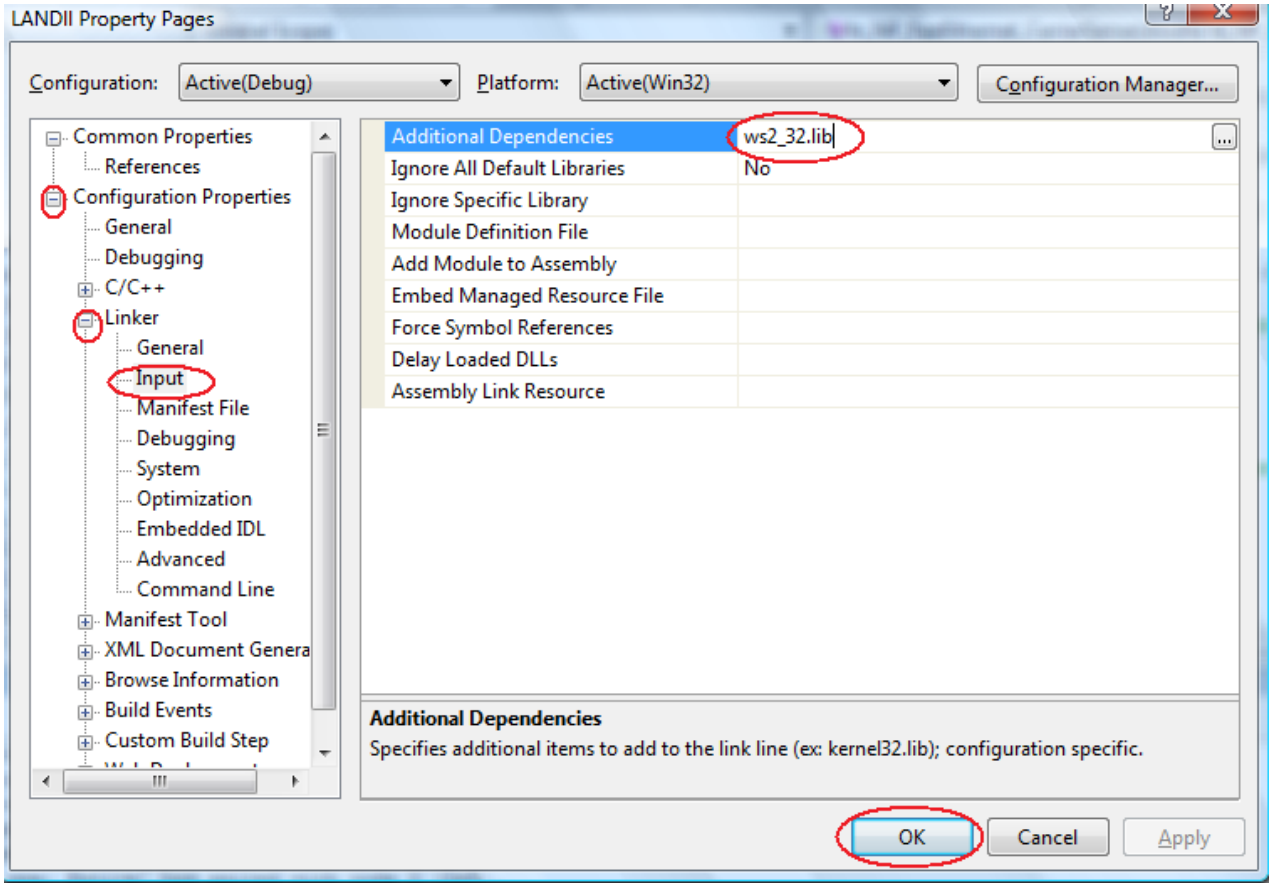
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

- For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

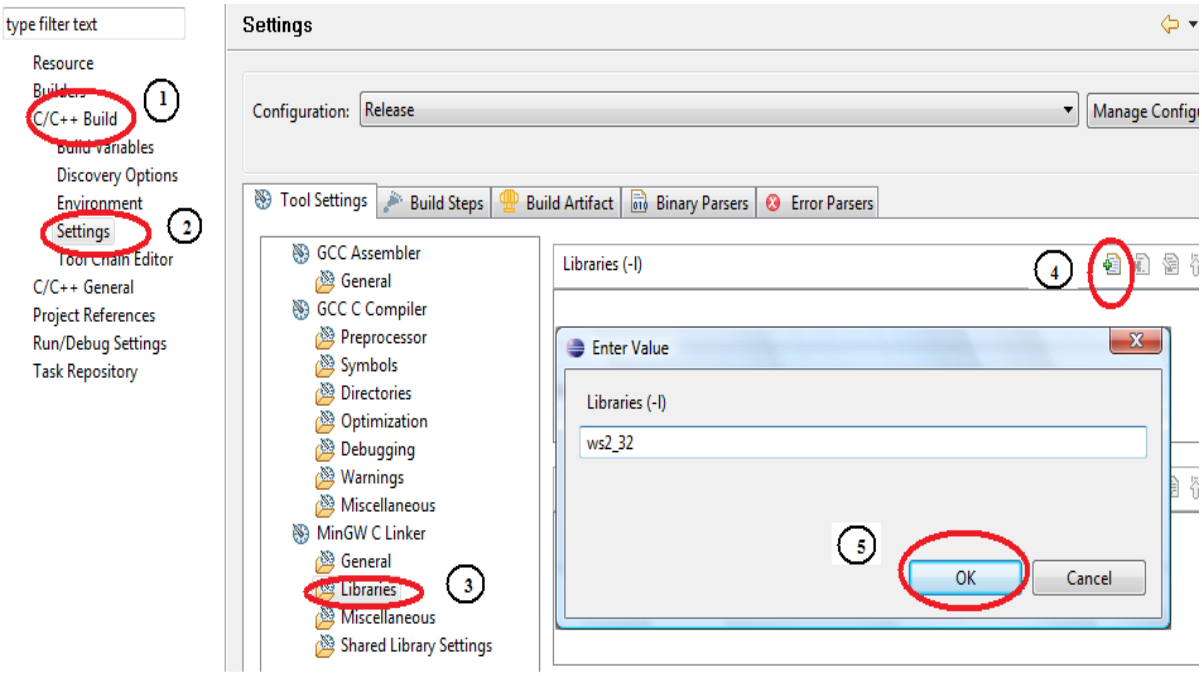
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

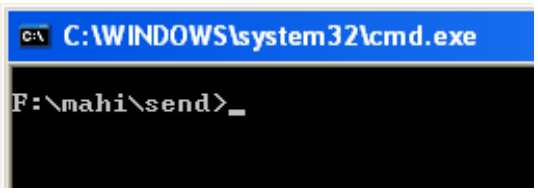
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

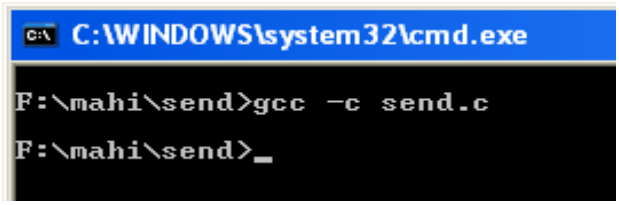
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

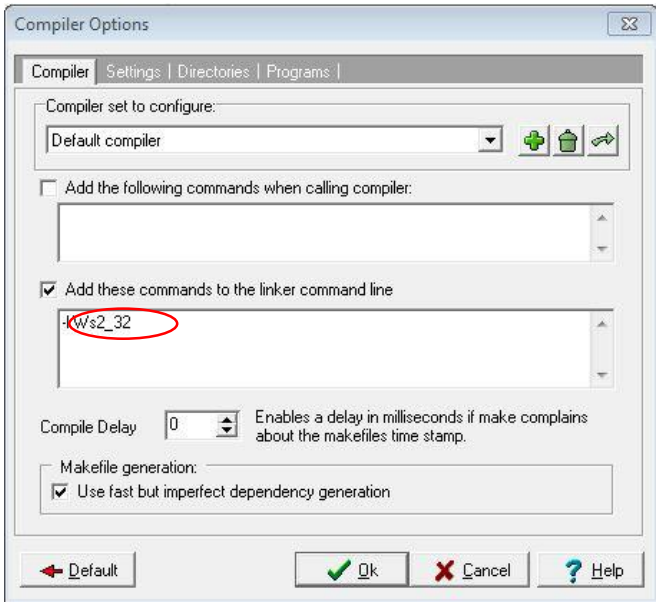
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools→compiler options and add -lws2_32 as follows before executing:



Input File Format	Output File Format
The data format in the input file is as follows, Receive Input Protocol=TCP Local User Name=Tetcos Send Protocol=TCP Number of Users=2 1=192.168.0.1 2=192.168.0.2 Tetcos:-Hello!!!!!!!!!!	Write the received message Ex: Tetcos:-Hello!!!!!!!!!!!!!!

Interface Source Code

Interface source code written in C is given using this the user can write the TCP chat functions. To view the interface source code, go to

NetSim Installation path / src / Programming/ ChatTCP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

10.5.42 PC to PC Communication – Chat Application UDP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions`

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

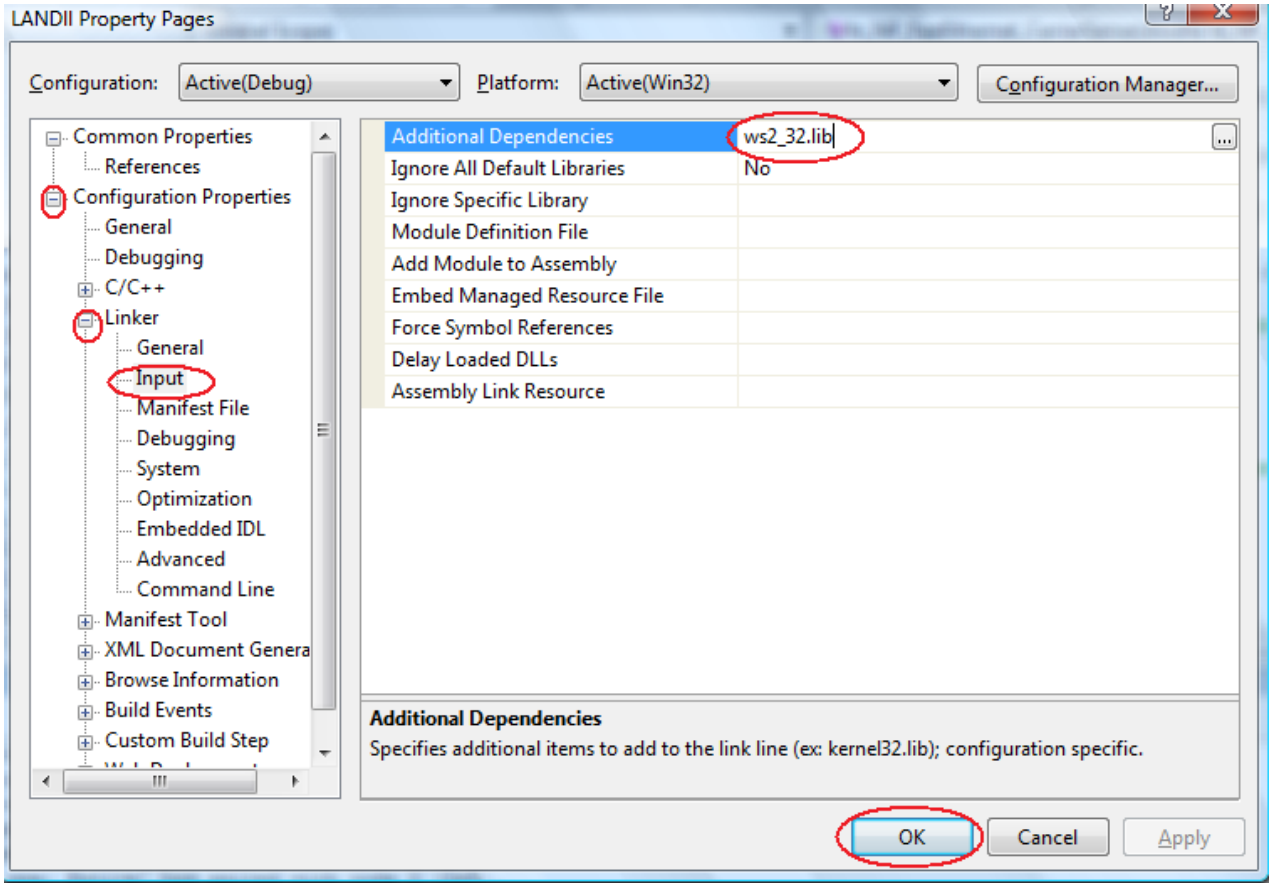
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

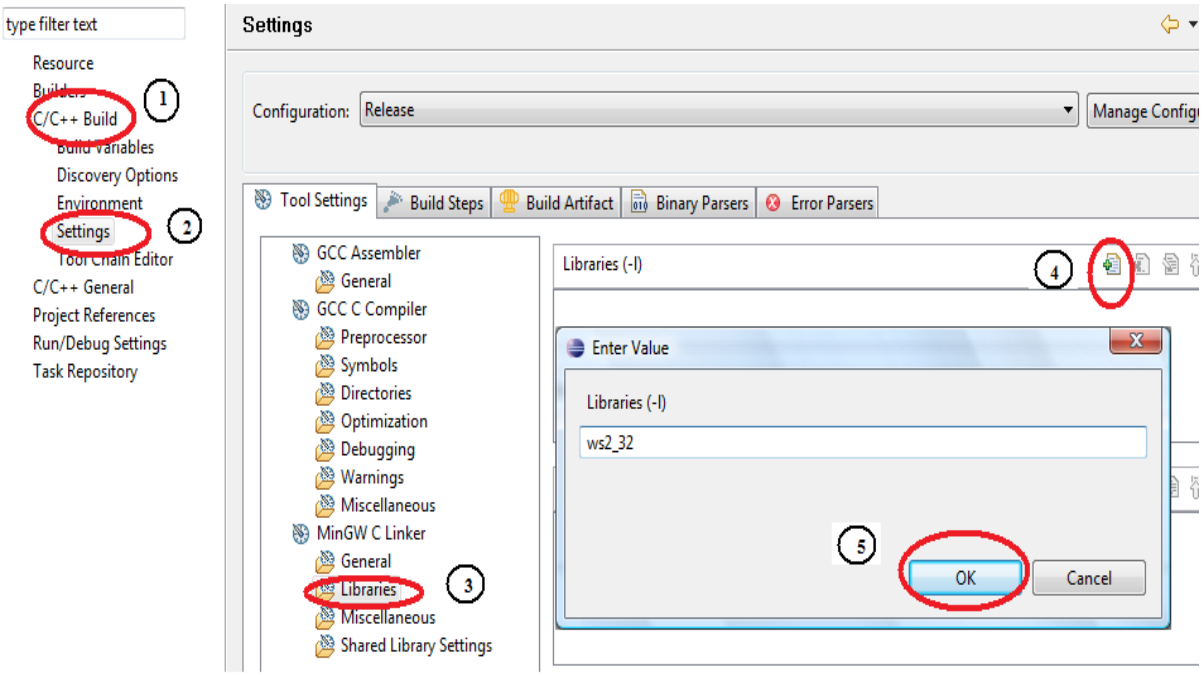
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

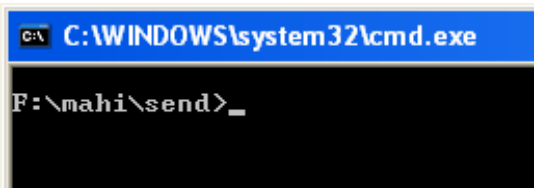
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

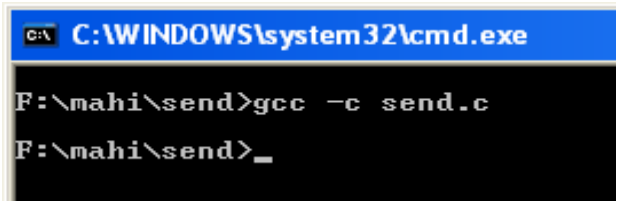
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

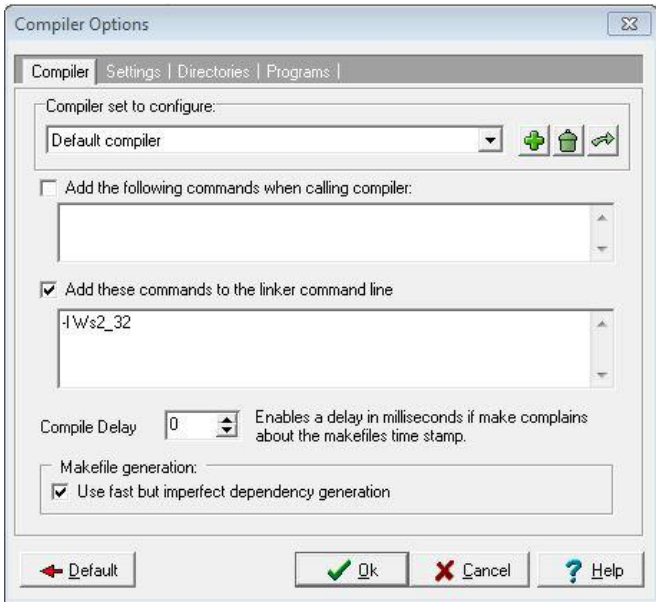
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools→compiler options and add -lws2_32 as follows before executing:



Input File Format	Output File Format
The data format in the input file is as follows,. Receive Input Protocol=UDP Local User Name=Tetcos Send Protocol=TCP Number of Users=2 1=192.168.0.1 2=192.168.0.2 Tetcos:-Hello!!!!!!!!!!	Write the received message Ex: 192.168.0.2:- Requesting for the connection

Interface Source Code

Interface source code written in C is given using this the user can write UDP Chat functions. To view the interface source code, go to

NetSim Installation path / src / Programming/ ChatUDP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and Receive the data using PC to PCCommunication – Chat Application

How to Proceed? -Two to Ten Systems are required to perform this experiment.

The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**PC to PC Communication → Chat Application**.

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** a **Protocol** from the following,
  - **TCP**
  - **UDP**
- On the Right hand side panel there is an **InputField** "**Send Data**", where **Data** needs to be typed in. This **Data** is later sent to the **ReceiverSystem**. Here **Type** "**Hello**".
- On the Right hand side panel there is "**Receive Data**", where data is received from user. Ex. **"Hello"**
- Under **Input** there are two things,
  - Number of users must be select between one to ten
  - Enter the IP address of the data where needs to be in all the column in the input pane.

Sample Output:

- On the Right hand side panel there is an **InputField** "**ReceivedData**", where **Data** would get displayed. "**Hello**" is the **Data** that is received from the **users**.

TCP

- Click **Run** button to **create the socket** and start Receive Data from the users.
- Click **Send** button to transmit the data to the users**.**
- Click **Refresh** button to terminate the connection.

UDP

- Click **Run** button to **create the socket** and start Receive Data from the users.
- Click **Send** button to transmit the data to the users**.**

10.5.43 Scheduling - First In First Out (FIFO)

Programming Guidelines

This section guides the user to link his/her own code for Scheduling - FIFO to NetSim.

Pre - Conditions

The program should read the data from '**Input.txt**' file that is available in the temporary directory.

The program should write the output data to '**Output.txt'** file that is available in the temporary directory.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Output_Link_Capacity=100 Node's_Priority=8,7,6,5,4,3,2,1, Node's_Capacity=0,0,500,0,0,0,0,0,	Output.txt file format is as follows, Seconds>no of bits transmitted in node 1>Transmission sequence of node 1>... no of bits transmitted in node 8>Transmission sequence of node 8> Sample Output File 1>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 2>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 3>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 4>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 5>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0>

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnFIFO() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ FIFO.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - The objective is to allocate the output capacity among the various input capacities using **First In First Out (FIFO) Scheduling** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Scheduling**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Algorithm** should be selected as **FirstIn First Out (FIFO).**
- Under **Input** field,
    0. **EnterOutputLinkCapacity** which is in **MBPS**. The options that are available for selecting are 100, 200, 300, …… 1000 MBPS. Select **400 MBPS**.
    1. **EnterBits** / **Node** for one second (limit 0 - 9999) in the fields available. i.e. Node1 - "100" , Node2 - "100", …….., Node8 - "100".
    - The **Priority** for the **Node** needs to be given. Click to Select or Deselect the Priority of a Node. In this **Sample** the **Priorities** are as follows,

Node Number	Priority Given
1	7
2	6
3	8
4	3

5	2
6	1
7	5
8	4

- **Click** on the **Run** button.

Output

Note: The Transmission of the data takes place in the following sequence. The Bits / Node are represented in different colors. These colors indicate the order in which the transmission takes place.

The Transmission Sequence →  **1 2 3 4 5 6 7 8.**

Seconds	1	2
Node1	0	100
Node2	0	100
Node3	100	0
Node4	0	100
Node5	0	100
Node6	100	0
Node7	100	0
Node8	100	0

10.5.44 Scheduling - Max - Min Fair (MMF)

Programming Guidelines

This section guides the user to link his/her own code for Scheduling - MMF to NetSim.

Pre - Conditions

The program should read the data from '**Input.txt**' file that is available in the application path.

The program should write the output data to '**Output.txt'** file that is available in the application path.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Output_Link_Capacity =100 Node's_Capacity=0,0, 300,0,0,0,0,0,	'Output.txt' file format is as follows. 1st second >Output rate of node 1 >Priority of Node 1>……..>Output rate of node 8.Priority of Node8 2nd second >Output rate of node 1 >Priority of Node 1>……..>Output rate of node 8.Priority of Node8 Nth second >Output rate of node 1 >Priority of Node 1>……..>Output rate of node 8.Priority of Node8 Sample File data 1>0.00>0>0.00>0>100.00>1>0.00>0>0.00>0>0.00>0>0.00>0>0.00>0> 2>0.00>0>0.00>0>100.00>1>0.00>0>0.00>0>0.00>0>0.00>0>0.00>0> 3>0.00>0>0.00>0>100.00>1>0.00>0>0.00>0>0.00>0>0.00>0>0.00>0>

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnMMFA () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ MMF.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - The objective is to allocate the output capacity among the various input capacities using **Max-Min Fare(MMF) Scheduling** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Scheduling**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Algorithm** should be selected as **Max - Min Fair (MMF).**
- Under **Input** field,
    0. **EnterOutputLinkCapacity** which is in **MBPS**. The options that are available for selecting are 100, 200, 300, …… 1000 MBPS. **Select100MBPS**.
    1. **EnterBits** / **Node** for one second (limit 0 - 9999) in the fields available. i.e. Node1 - "100" , Node2 - "100", …….., Node8 - "100".
- **Click** on the **Run** button.

Output

The Transmission of the data takes place in the following sequence. The **Bits** / **Node** are represented in different colors. These colors indicate the order in which the transmission takes place.

The Transmission Sequence →  **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8.**

Seconds	1	2	3	4	5	6	7	8
Node1	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node2	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node3	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node4	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node5	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node6	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node7	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node8	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50

10.5.45 Shortest Path - Floyd's

Programming Guidelines

This section guides the user to link his/her own code for shortest path – Floyd's to NetSim.

Pre - Conditions

The program should read the data from 'Input.txt' file that is available in the temporary directory.

The program should write the output data to 'Output.txt' file that is available in the temporary directory.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file Input.txt.

Executing the required concept and, the results of the program should be written into the output file Output.txt.

File Format

Input file format	Output file format
Algorithm=Floyd's No_of_Router=3 Distance: 999>5>1> 5>999>10> 1>10>999> Source_Router=1 Destination_Router=3 Note: '>' is the delimiter symbol, which is used to separate each input.	Consider the source node is 1 Distance>0>1>2>1>2>1> Path>0>1>2>1>4>1> The first line has the all the distance values from the source node to all the other connected nodes. The second line has the path values from the source node to all the other nodes. The first line contains the values stored in the distance array (single dimensional array) used in the algorithm. The second line contains the values stored in the path array (single dimensional array) used in the algorithm.

	Sample Output: 1>2>3> 1>2> 1>3> 1> >>> path_FS> 0>0>0> 0>0>1> 0>1>0> distance_FS> 0>5>1> 5>0>6> 1>6>0> Note: The string Distance and Path in the file is compulsory. The output must be stored in the same format

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnFloyd() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Floyds.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - The objective is to find the **Shortest Path** between the two **Routers** using the **Floyds** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the **Programming** exercise available. Under **Programming Menu** select **Shortest Path**.

Sample Input - By using the **Input** Panel that is available on the left hand side a **Sample Scenario** is created. The **Steps** involved in creating are as follows,

- **Sample Mode** should be selected.
- **Select** the **Algorithm** as **Floyd's**.
- **Number** of **Routers** needs to be selected. **Minimum** of **3Routers** and **Maximum** of **8Routers** can be selected. Let us consider 4 **Routers** are selected.
- **Source Router** needs to be given. Let **Router** 4 be the **Source Router**.
- **Destination Router** also needs to be given. Let **Router** 1 be the **Destination Router**.
- **Click** on 2 **Routers** to give the **distance** between those 2 **Routers**. A **blue arrow** would be pointing from the **Source Node** to the **Destination Node**. The **Distances** between the **Routers** that needs to be given are as follows,
    1. Router 4 and Router 1 → 67 Km,
    2. Router 4 and Router 3 → 46 Km,
    3. Router 3 and Router 1 → 18 Km,
    4. Router 3 and Router 2 → 56 Km, and
    5. Router 1 and Router 2 → 70 Km.
- **Click** on **Run** button to view the output in the **Routing Table**.

Output - The **Output** for the above **Scenario** is as follows,

- **Click** the **Run** button to view program output.
- The **Output** is obtained in a **Routing Table**. All possible **Routes** and the corresponding **Distance** are obtained. Below is a **Routing Table** for the above Inputs,

Routing Table	
Route	**Distance[KM]**
4>1>	67
4>3>1>	64
4>3>2>	102
4>3>	46
4>	0

- The **ShortestPath** would be highlighted in **green color**.
- **Click** on **Refresh Button** to **Refresh** the screen and create fresh **Scenarios**.

10.5.46 Shortest Path - Link State

Programming Guidelines

This section guides the user to link his/her own code for shortest path - Link State Routing to NetSim.

Pre - Conditions

The program should read the data from '**Input.txt**' file that is available in the temporary directory.

The program should write the output data to '**Output.txt'** file that is available in the temporary directory.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Input file format	**Output file format**
Input.txt contains the distance matrix given by the user, along with the source node, destination node and the total number of nodes. Here, we have the input as a SYMMETRIC matrix. The format of input.txt is, for example, Algorithm=Link_State No_of_Router=3 Distance: 999>6>5> 6>999>7> 5>7>999> Source_Router=1 Destination_Router=2 Note:	Consider the source node is 1 Distance>0>1>2>1>2>1> Path>0>1>2>1>4>1> The first line has the all the distance values from the source node to all the other connected nodes. The second line has the path values from the source node to all the other nodes. The first line contains the values stored in the distance array (single dimensional array) used in the algorithm. The second line contains the values stored in the path array (single dimensional array) used in the algorithm. Sample Output:

'>' is the delimiter symbol, which is used to separate each input.	1>2> 1>3>2> 1>3> 1> >>> Distance>0>6>5> Path>1000>1>1> Note: The string Distance and Path in the file is compulsory. The output must be stored in the same format.

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnDijkstra() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ LinkState.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - The objective is to find the **Shortest Path** between the two **routers** using the **Link State** algorithm.

How to Proceed? - Under **ProgrammingMenu** select **ShortestPath**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** the **Algorithm** as **RunedState**.
- **Number** of **Routers** needs to be selected. **Minimum** of **3Routers** and **Maximum** of **8Routers** can be selected. Select 5 **Routers**.
- **SourceRouter** needs to be given. Let **Router** 5 be the **SourceRouter**.
- **DestinationRouter** also needs to be given. Let **Router** 3 be the **DestinationRouter**.

- **Click** on 2 **Routers** to give the distance between those 2 **Routers**. A blue Line would be pointing from the **SourceNode** to the **DestinationNode**. The **Distances** between the **Routers** that needs to be given are as follows,
    1. Router 1 and Router 2 → 35,
    2. Router 2 and Router 3 → 30,
    3. Router 3 and Router 4 → 60,
    4. Router 4 and Router 1 → 20,
    5. Router 4 and Router 5 → 67, and
    6. Router 5 and Router 2 → 58.
- **Click** on **Run** button to view the output in the **RoutingTable**.

Output - The **Output** for the above **Scenario** is as follows,

- **Click** the **Run** button to view program output.
- The **Output** is obtained in a **RoutingTable**. All possible **Routes** and the corresponding **Distance** are obtained. Below is a **RoutingTable** for the above Inputs,
- The **ShortestPath** would be highlighted in **Green** color in the **RoutingTable**.
- **Click** on **RefreshButton** to **Refresh** the screen and create fresh **Scenarios**.

Routing Table	
Route	**Distance[KM]**
5>2>1>4>3>	173
5>2>1>4>	113
5>2>1>	93
5>2>3>	88
5>2>	58
5>4>1>2>3>	152
5>4>1>2>	122
5>4>1>	87
5>4>3>	127
5>4>	67
5>	0

10.5.47 Sliding Window Protocol - Go Back N

Programming Guidelines

This section guides the user to link his/her own code for Go Back N to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for **Go Back N** is **7**

Input File Format	Output File Format
Algorithm=Go_Back_N Data_File=C:\Users\P.Sathishkumar\Documents\1 Th.txt> Bit_Error_Rate=6 Sequence_Number=3 Window_Size=7	Value1>Value2>Value3>Value4> Types: There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. Type1: Value1 - "CNT", Value 2 -output of the slidingcount function Value 3 - "FRAMES"

	Value 4 - "TRANSMIT" Type2: Value1 - "DT", Value 2 -Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. Type3: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. Type4: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. Type5: Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" Note: The above convention to write into the 'Output.Txt' ismandatory. Values in Quotes"" to be written into file 'Output.Txt' as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: CNT>1>FRAMES>TRANSMIT>

	DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node1>node2>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sliding Window Protocols - Go Back N inside the function GoBackN () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ GobackN_SW.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and receive the data using **Sliding Window Protocol - Go BackN**.
How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **ProgrammingMenu** select **Sliding WindowProtocol→Go Back N**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **SelectGoBackN** as **Algorithm**.
- **Create** a "**Datafile** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Datafile** (**.txt**)" which could be **Maximum** of **100000bytes**.
- **SelectBit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.
- **SelectSequencenumber** (**Bits**) as "**3**" from the drop down menu.
- A default value is entered for **WindowSize** as "**7**".

1. Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

2. An **OutputTable** is obtained when **Run** button is clicked. In the **OutputTable**, **Transmittedframe** (**Data** and **Acknowledgement**) and their corresponding Counts is obtained. The **TotalCount** is also obtained. The table is given below,

Output

Total data frames to be transmitted - 24

Transmitted frame	Count
Data	30
Acknowledgement	5
	Total = 35

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

3. The details of the **DataFrames** flowing from **Node1** to **Node 2** are obtained on the right hand side panel. Below are the details that is obtained in the tool,
    - Data Frame 1 is flowing from Node1 to Node2 with no error.
    - Data Frame 2 is flowing from Node1 to Node2 with no error.
    - Data Frame 3 is flowing from Node1 to Node2 with no error.
    - Data Frame 4 is flowing from Node1 to Node2 with error.
    - Data Frame 5 is flowing from Node1 to Node2 with no error.
    - Data Frame 6 is flowing from Node1 to Node2 with no error.
    - Data Frame 7 is flowing from Node1 to Node2 with error.
    - Acknowledgement from Node2 to Node1.
    - Data Frame 4 is flowing from Node1 to Node2 with no error.
    - Data Frame 5 is flowing from Node1 to Node2 with no error.
    - Data Frame 6 is flowing from Node1 to Node2 with no error.
    - Data Frame 7 is flowing from Node1 to Node2 with no error.
    - Data Frame 8 is flowing from Node1 to Node2 with no error.
    - Data Frame 9 is flowing from Node1 to Node2 with no error.
    - Data Frame 10 is flowing from Node1 to Node2 with no error.
    - Acknowledgement from Node2 to Node1.
    - Data Frame 11 is flowing from Node1 to Node2 with no error.
    - Data Frame 12 is flowing from Node1 to Node2 with no error.
    - Data Frame 13 is flowing from Node1 to Node2 with no error.

- Data Frame 14 is flowing from Node1 to Node2 with no error.
- Data Frame 15 is flowing from Node1 to Node2 with no error.
- Data Frame 16 is flowing from Node1 to Node2 with no error.
- Data Frame 17 is flowing from Node1 to Node2 with no error.
- Acknowledgement from Node2 to Node1.
- Data Frame 18 is flowing from Node1 to Node2 with no error.
- Data Frame 19 is flowing from Node1 to Node2 with no error.
- Data Frame 20 is flowing from Node1 to Node2 with no error.
- Data Frame 21 is flowing from Node1 to Node2 with no error.
- Data Frame 22 is flowing from Node1 to Node2 with no error.
- Data Frame 23 is flowing from Node1 to Node2 with error.
- Data Frame 24 is flowing from Node1 to Node2 with no error.
- Acknowledgement from Node2 to Node1.
- Data Frame 23 is flowing from Node1 to Node2 with no error.
- Data Frame 24 is flowing from Node1 to Node2 with no error.
- Acknowledgement from Node2 to Node1.

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.48 Sliding Window Protocol - Selective Repeat

Programming Guidelines

This section guides the user to link his/her own code for Selective Repeat to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for Selective Repeat is **7**

Input File Format	Output File Format
Algorithm=Selective_Repeat Data_File=C:\Users\P.Sathishkumar \Documents\1 Th.txt> Bit_Error_Rate=5 Sequence_Number=3 Window_Size=4	Value1>Value2>Value3>Value4> Types: There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. Type1: Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES"

	Value 4 - "TRANSMIT" Type2: Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. Type3: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. Type4: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. Type5: Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" Note: The above convention to write into the 'Output.Txt' ismandatory. Values in Quotes"" to be written into file 'Output.Txt' as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex:

	CNT>1>FRAMES>TRANSMIT> DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node1>node2>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sliding Window Protocol - Selective Repeat inside the function SelectiveRepeat () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ SelectiveRepeat_SW.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and receive the data using **Sliding Window Protocol** - **Selective Repeat**.
How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **Programming Menu** select **Sliding WindowProtocol→Selective Repeat**.
Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- **Select Selective Repeat** as **Algorithm**.
- **Create** a "**Data file** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Data file** (**.txt**)" which could be **Maximum** of **100000 bytes**.
- **Select Bit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.
- **Select Sequence number** (**Bits**) as "**3**" from the drop down menu.
- A default value is entered for **Window Size** as "**4**".

1. Then **Run** button need to be clicked.

Output - **Output** for the above **Sample** is as follows,

2. An **Output Table** is obtained when **Run** button is clicked. In the **Output Table**, **Transmitted frame** (**Data** and **Acknowledgement**) and their corresponding **Counts** is obtained. The **Total Count** is also obtained. The table is given below,

Output

Total data frames to be transmitted - 24

Transmitted frame	Count
Data	26
Acknowledgement	7
	Total = 33

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

3. The details of the **Data Frames** flowing from **Node 1** to **Node 2** are obtained on the right hand side panel. Below are the details that is obtained in the tool,
    - Data Frame 1 is flowing from Node1 to Node2 with no error
    - Data Frame 2 is flowing from Node1 to Node2 with no error
    - Data Frame 3 is flowing from Node1 to Node2 with no error
    - Data Frame 4 is flowing from Node1 to Node2 with error
    - Acknowledgement from Node2 to Node1
    - Data Frame 4 is flowing from Node1 to Node2 with no error
    - Data Frame 5 is flowing from Node1 to Node2 with no error
    - Data Frame 6 is flowing from Node1 to Node2 with error
    - Data Frame 7 is flowing from Node1 to Node2 with no error
    - Acknowledgement from Node2 to Node1
    - Data Frame 6 is flowing from Node1 to Node2 with no error
    - Data Frame 8 is flowing from Node1 to Node2 with no error
    - Data Frame 9 is flowing from Node1 to Node2 with no error
    - Data Frame 10 is flowing from Node1 to Node2 with no error
    - Acknowledgement from Node2 to Node1
    - Data Frame 11 is flowing from Node1 to Node2 with no error
    - Data Frame 12 is flowing from Node1 to Node2 with no error

- Data Frame 13 is flowing from Node1 to Node2 with no error
- Data Frame 14 is flowing from Node1 to Node2 with no error
- Acknowledgement from Node2 to Node1
- Data Frame 15 is flowing from Node1 to Node2 with no error
- Data Frame 16 is flowing from Node1 to Node2 with no error
- Data Frame 17 is flowing from Node1 to Node2 with no error
- Data Frame 18 is flowing from Node1 to Node2 with no error
- Acknowledgement from Node2 to Node1
- Data Frame 19 is flowing from Node1 to Node2 with no error
- Data Frame 20 is flowing from Node1 to Node2 with no error
- Data Frame 21 is flowing from Node1 to Node2 with no error
- Data Frame 22 is flowing from Node1 to Node2 with no error
- Acknowledgement from Node2 to Node1
- Data Frame 23 is flowing from Node1 to Node2 with no error
- Data Frame 24 is flowing from Node1 to Node2 with no error
- Acknowledgement from Node2 to Node1

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.49 Sorting Technique - Bubble Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Bubble Sorting_Order=Ascending Total_Number=3 Number_to_Sort=5,4,3 Descending Order Sorting_Type=Bubble Sorting_Order=Descending Total_Number=3 Number_to_Sort=4,5,6	type>index 1>index 2> type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared type 2 : specifies the two index following it are being swapped The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value.

	type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. Sample Output: Ascending Order: 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1> 2>0>1> Descending Order: 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1> 2>0>1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function fnBubblesort () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ BubbleSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Bubble Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.
- A table with the values tabulated would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.50 Sorting Technique - Insertion Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Insertion Sorting_Order=Ascending Total_Number=3 Number_to_Sort=9,8,7 Descending Order Sorting_Type=Insertion Sorting_Order=Descending	type>index 1>index 2> type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared type 2 : specifies the two index following it are being swapped The following types are used in Quick Sort :

Total_Number=3 Number_to_Sort=1,9,3	type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. Sample Output: Ascending Order: 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1> 2>0>1> Descending Order: 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function fnInsertsort () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ InsertionSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Project:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Insertion Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.
- A table with the values tabulated would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.51 Sorting Technique - Quick Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Quick Sorting_Order=Ascending Total_Number=3 Number_to_Sort=33,22,66 Descending Order	type>index 1>index 2> type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared type 2 : specifies the two index following it are being swapped

Sorting_Type=Quick Sorting_Order=Descending Total_Number=3 Number_to_Sort=22,33,44	The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. Sample Output: Ascending: 0>0>0> 1>0>2> 3>1>0> 4>0>1> 0>2>2> 0>0>0> Descending: 0>0>0> 3>2>0> 1>0>1> 4>0>2> 0>0>0> 1>0>1> 4>0>0> 0>1>1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function fnQuicksort () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ QuickSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Quick Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.
- A table with the values tabulated would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.52 Sorting Technique - Selection Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and the results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	**Output File Format**
Ascending Order Sorting_Type=Selection Sorting_Order=Ascending Total_Number=3 Number_to_Sort=88,77,66	type>index 1>index 2> type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared type 2 : specifies the two index following it are being swapped

Descending Order Sorting_Type=Selection Sorting_Order=Descending Total_Number=3 Number_to_Sort=55,11,22	The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. Sample Output: Ascending: 0>0>0> 1>1>0> 0>1>1> 1>2>1> 0>2>2> 2>0>2> 0>1>1> 1>2>1> Descending: 0>0>0> 1>1>0> 1>2>0> 0>1>1> 1>2>1> 0>2>2> 2>1>2>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function select() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ SelectionSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties → find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Selection Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.
- A table with the values tabulated would be obtained.

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.5.53 Spanning Tree – Borovska

Programming Guidelines

This section guides the user to link his/her own code for Spanning Tree using Borovska algorithm to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the Reading of the Inputs from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Borovska No_of_Switches=3 No_of_Edges=3 Source_Switch=3,Destination_Switch=1, Distance=11, Source_Switch=1,Destination_Switch=2, Distance=22, Source_Switch=3,Destination_Switch=2, Distance=33,	'Output.txt ' file has the edges in the spanning tree, which is selected from the input edges. Node1>Node2>Cost> Example: 1>2>22> 1>3>11>

Interface Source Code

Interface source code written in C is given .Using this the user can write only the Borovska algorithm inside the function fnBorovska() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Boruvska.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To find a **SpanningTree** for a network using **Borovska's** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** Menu select**Spanning Tree**.

Sample Input

- **SampleMode** should be selected.
- **SelectBorovska** as **Algorithm** from the list available.
- **Select** the **Number** of **Switches**. **Select** 4 **Switches** as an **Input**.
- **Click** on 2 **Switches** to give the **distance** between them. Similarly connect all the **Switches** that are available in the network. The **Distances** between the **Routers** that needs to be given are as follows,
    - Switch 1 and Switch 2 → 67 Km
    - Switch 1 and Switch 4 → 46 Km
    - Switch 1 and Switch 3 → 89 Km
    - Switch 2 and Switch 3 → 78 Km
    - Switch 2 and Switch 4 → 99 Km
    - Switch 3 and Switch 4 → 56 Km
- **Click** on **Run** button to execute. **Refresh** button can be used if new **Inputs** have to be given.

Sample Output

- The **SpanningTreeTable** with the **Path** (**Source Switch** and **Destination Switch**) and **Distance** is obtained. Below is the Table that is obtained for the above inputs.

Spanning Tree Table

Path		Distance[KM]
1	2	67
3	4	56
1	4	46

- The "**Length** of the **SpanningTree** (**KM**)" would be given below in the output panel. The **SpanningTreePath** consists of **green** lines, whereas the **Non-SpanningTree** consists of **red** lines. Here in this Sample, "**Length** of the **Spanning Tree** (**KM**) → 169".

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.54 Spanning Tree – Kruskal

Programming Guidelines

This section guides the user to link his/her own code for Spanning Tree using Kruskal algorithm to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the Reading of the Inputs from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Kruskal's No_of_Switches=3 No_of_Edges=3 Source_Switch=3,Destination_Switch=1,Distance=23, Source_Switch=1,Destination_Switch=2,Distance=34, Source_Switch=3,Destination_Switch=2,Distance=45,	'Output.txt ' file has the edges in the spanning tree, which is selected from the input edges. Node1>Node2>Cost> Example: 3>1>23> 1>2>34>

Interface Source Code

Interface source code written in C is given .Using this the user can write only the Kruskal algorithm inside the function fnKruskal() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Kruskal.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To find the **SpanningTree** for a network by using **Kruskal** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** Menu select**Spanning Tree**.

Sample Input

- **SampleMode** should be selected.
- **SelectKruskal** as **Algorithm** from the list available.
- **Select** the **Number** of **Switches**. **Select** 4 **Switches** as an **Input**.
- **Click** on 2 **Switches** to give the distance between them. Similarly connect all the **Switches** that are available in the network. The **Distances** between the **Routers** that needs to be given are as follows,
    - Switch 1 and Switch 2 → 67 Km
    - Switch 1 and Switch 4 → 46 Km
    - Switch 1 and Switch 3 → 89 Km
    - Switch 2 and Switch 3 → 78 Km
    - Switch 2 and Switch 4 → 99 Km
    - Switch 3 and Switch 4 → 56 Km
- **Click** on **Run** button to execute. **Refresh** button can be used if new **Inputs** have to be given.

Sample Output

- The **SpanningTreeTable** with the **Path** (**Source Switch** and **Destination Switch**) and **Distance** is obtained. Below is the **Table** that is obtained for the above inputs,

Spanning Tree Table

Path		Distance[KM]
1	4	46
4	3	56
1	2	67

- The "**Length** of the **SpanningTree** (**KM**)" would be given below in the output panel. The **SpanningTree Path** consists of **green** lines, whereas the **Non-SpanningTree** consists of **red** lines. Here in this Sample, "**Length** of the **SpanningTree** (**KM**) → 169"

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.55 Spanning Tree – Prims

Programming Guidelines

This section guides the user to link his/her own code for Spanning Tree using Prims algorithm to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the Reading of the Inputs from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Prim's No_of_Switches=3 No_of_Edges=3 Source_Switch=1 Source_Switch=3,Destination_Switch=1,Distance=44, Source_Switch=1,Destination_Switch=2,Distance=55, Source_Switch=2,Destination_Switch=3,Distance=66,	'Output.txt' file has the edges in the spanning tree, which is selected from the input edges. Node1>Node2>Cost> Example: 1>3>44> 1>2>55>

Interface Source Code

Interface source code written in C is given .Using this the user can write only the Prim's algorithm inside the function fnPrims() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Prims.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - To find a **SpanningTree** for a network using **Prims** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** Menu select**Spanning Tree**.

Sample Input

- **SampleMode** should be selected.
- **SelectPrim's** as **Algorithm** from the list available.
- **Select** the **Number** of **Switches**. **Select** 4 **Switches** as an **Input**.
- **Click** on 2 **Switches** to give the **distance** between them. Similarly connect all the **Switches** that are available in the network. The **Distances** between the **Routers** that needs to be given are as follows,
    - Switch 1 and Switch 2 → 67 Km
    - Switch 1 and Switch 4 → 46 Km
    - Switch 1 and Switch 3 → 89 Km
    - Switch 2 and Switch 3 → 78 Km
    - Switch 2 and Switch 4 → 99 Km
    - Switch 3 and Switch 4 → 56 Km
- **SourceSwitch** can be any 1 from list.Here, **SourceSwitch** is selected as"3".
- **Click** on **Run** button to execute. **Refresh** button can be used if new **Inputs** have to be given.

Sample Output

- The **SpanningTreeTable** with the **Path** (**SourceSwitch** and **Destination Switch**) and **Distance** is obtained. Below is the **Table** that is obtained for the above inputs,

SpanningTreeTable

Path		Distance[KM]
3	4	56
4	1	46
1	2	67

  - The "**Length** of the **SpanningTree** (**KM**)" would be given below in the output panel. The **SpanningTree Path** consists of **green** lines, whereas the **Non-SpanningTree** consists of **red** lines. Here in this Sample, "**Length** of the **Spanning Tree** (**KM**) → 169".

- Once the sample experiment is done, then **Refresh** button can be clicked to create New Samples.

10.5.56 Transmission Flow Control - Go Back N

Programming Guidelines

This section guides the user to link his/her own code for Go Back N to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for Go Back N is **7**

Input File Format	Output File Format
Algorithm=Go_Back_N Data_File=C:\Users\P.Sathishkumar\Documents\1 Th.txt> BER=0	Value1>Value2>Value3>Value4> Types: There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. Type1: Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES"

	Value 4 - "TRANSMIT" Type2: Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. Type3: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. Type4: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. Type5: Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" Note: The above convention to write into the 'Output.Txt' is mandatory. Values in Quotes"" to be written into file 'Output.Txt' as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: CNT>1>FRAMES>TRANSMIT> DT>1>node1>node2>

	EV>0>node1>node2> ACK>POS>node2>node1> DEL>1>FRAME>DELETED>

Interface Source Code

Interface source code written in C is given using this the user can write only the Transmission Flow Control - Go Back N inside the function GoBackN () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ GoBackN_TFC.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and receive the data using **Transmission Flow Control** - **Go BackN**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **ProgrammingMenu** select **Transmission Flow Control→Go Back N**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **SelectGoBackN** as **Algorithm**.
- **Create** a "**Datafile** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Datafile** (**.txt**)" which could be **Maximum** of **100000bytes**.
- **SelectBit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.

1. Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

2. An **OutputTable** is obtained when **Run** button is clicked. In the **OutputTable**, **Transmittedframe** (**Data** and **Acknowledgement**) and their corresponding Counts is obtained. The **TotalCount** is also obtained. The table is given below,

Output

Total data frames to be transmitted - 24

Transmitted frame	Count
Data	30
Acknowledgement	5
	Total = 35

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.57 Transmission Flow Control - Selective Repeat

Programming Guidelines

This section guides the user to link his/her own code for Selective Repeat to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for Selective Repeat is **7**

Input File Format	Output File Format
Algorithm=Selective_Repeat Data_File=C:\Users\P.Sathishkumar\ Documents\1 Th.txt> BER=0	Value1>Value2>Value3>Value4> Types: There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. Type1: Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES"

	Value 4 - "TRANSMIT" Type2: Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. Type3: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. Type4: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. Type5: Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" Note: The above convention to write into the 'Output.Txt' is mandatory. Values in Quotes"" to be written into file 'Output.Txt' as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: CNT>1>FRAMES>TRANSMIT>

	DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node2>node1> DEL>1>FRAME>DELETED>

Interface Source Code

Interface source code written in C is given using this the user can write only the Transmission Flow Control -Selective Repeat inside the function SelectiveRepeat () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ SelectiveRepeat_TFC.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and receive the data using **Transmission Flow Control - Selective Repeat**.
How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **Programming Menu** select **Transmission Flow Control→Selective Repeat**.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- **Select Selective Repeat** as **Algorithm**.
- **Create** a "**Data file** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Data file** (**.txt**)" which could be **Maximum** of **100000 bytes**.
- **Select Bit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.

1. Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

2. An **Output Table** is obtained when **Run** button is clicked. In the **Output Table**, **Transmitted frame** (**Data** and **Acknowledgement**) and their corresponding **Counts** is obtained. The **Total Count** is also obtained. The table is given below,

Output

Total data frames to be transmitted -

Transmitted frame	Count
Data	
Acknowledgement	
	Total =

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

10.5.58 Transmission Flow Control - Stop and Wait

Programming Guidelines

This section guides the user to link his/her own code for Stop and Wait to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Stop_and_Wait Data_File=C:\Users\P.Sathishkumar\ Documents\1 Th.txt> BER=0	Value1>Value2>Value3>Value4> Types: There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. Type1: Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES" Value 4 - "TRANSMIT" Type2: Value1 - "DT",

	Value 2 - Frame number, Value 3 - Frame’s Source address, Value 4 - Frame’s Destination address. Type3: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame’ Source address, Value 4 - Frame’s Destination address. Type4: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame’s Source Address, Value 4 - Acknowledgement frame’s Destination Address. Type5: Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" Note: The above convention to write into the ‘Output.Txt’ is mandatory. Values in Quotes"" to be written into file ‘Output.Txt’ as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node2>node1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Transmission Flow Control - Stop and Wait inside the function stopandwait() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ StopandWait.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7/8/10

Sample Scenarios:

Objective - Send and receive the data using **Transmission Flow Control**– **Stop and wait**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **Programming Menu** select **Transmission Flow Control →Stop and wait**.

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- **Stop and Wait** needs to be **selected** for Algorithm.
- The path of the "**Data file (.txt)**" should be **entered**.
- **Bit Error Rate (BER)** should be **selected**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- When the **Run** Button is **clicked** the Tool generates the Output in the following format,

Output

Total data frames to be transmitted

Transmitted frame	Count
Data	
Acknowledgement	

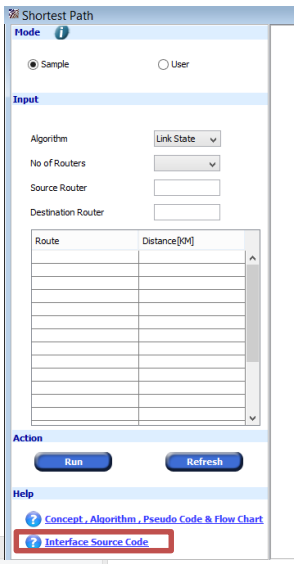
Total =

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

10.6 Programming exercise - How to practice without NetSim

Step 1: Copy Interface Source Code

Click on Interface Source Code and save the code in a text file.

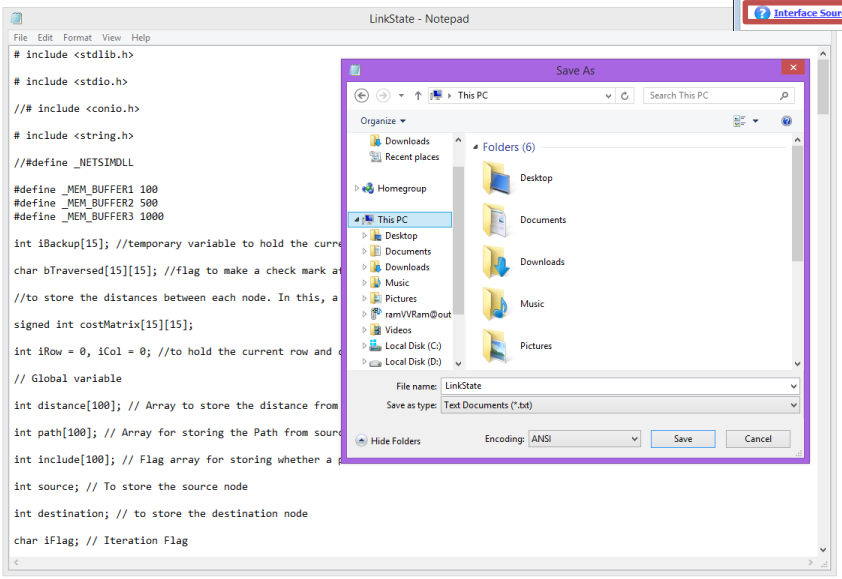


For Example:

As an example, Shortest Path programming exercise is being considered.

Open the programming exercise and select **Interface Source Code.**

Save the Interface source code in another txt file.



Step 2: Create Input.txt file

Create a new text file with the name Input.txt.

The contents of the file will vary for each programming exercises and are specified in NetSim User Manual under programming topic.

For Example:

Copy the input file contents from NetSim user manual and save it in a file called Input.txt

Sample:

Algorithm=Link_State
No_of_Router=3
Distance:
999>6>5>
6>999>7>
5>7>999>
Source_Router=1
Destination_Router=2

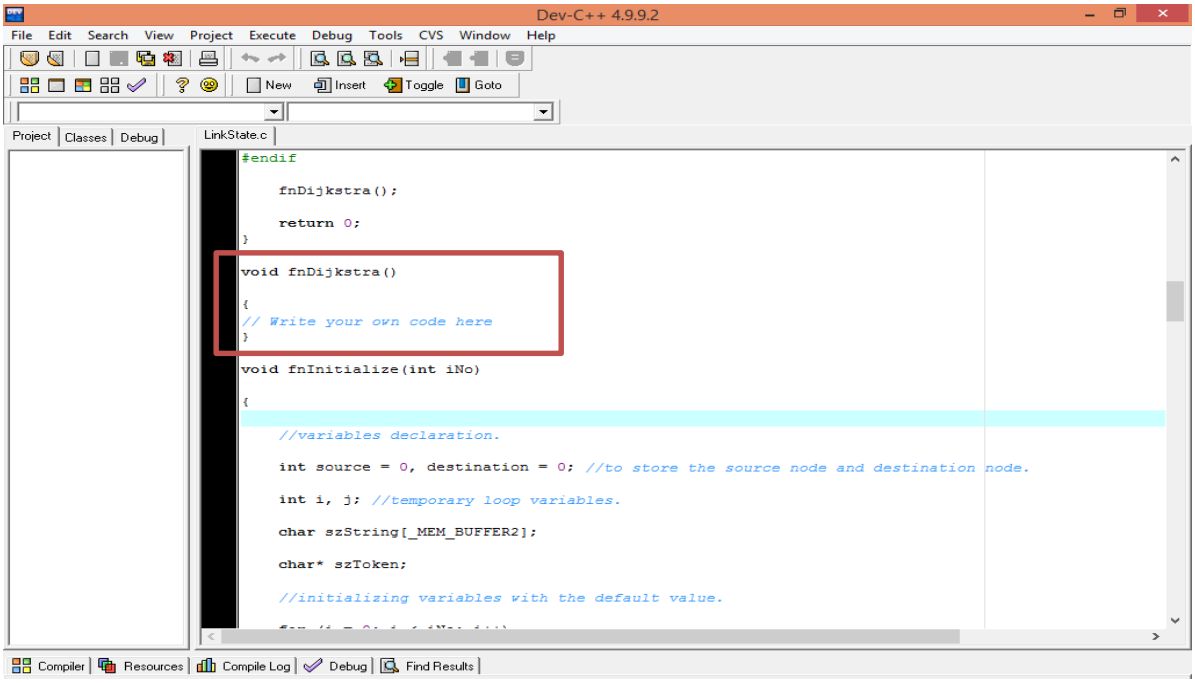
Save it in some other location. In this example, it is saved at D:\New.

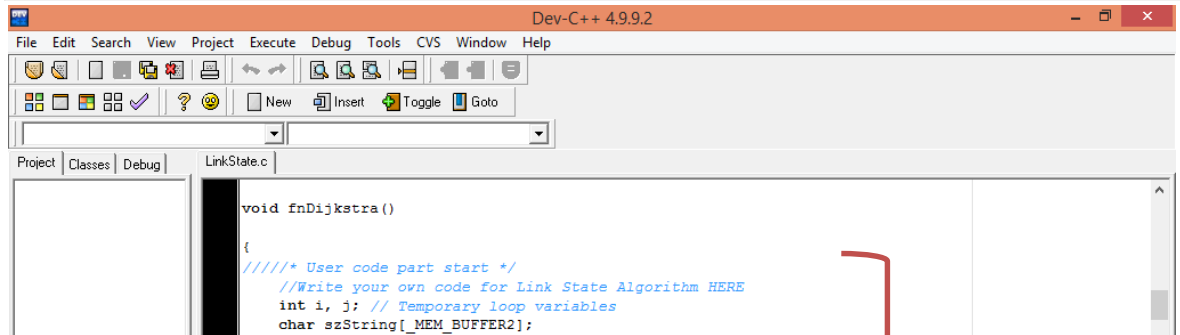
Step 3: Create .exe file

Open C editor and write the Interface source code which was saved in the first text file. Also add the code to be written by the user in the specific sections and compile it. Keep the .exe file.

For Example:

Now in C editor (Dev C++ IDE is used here), copy the interface source code. In the interface source code, void fnDijkstra() is not implemented.





After writing the code for the function void fnDijkstra(), create .exe file. The procedure to create .exe file is explained in Section 9.2 of NetSim User Manual.

In this example, the exe file created is **LinkState.exe**

Step 4: Run in command prompt

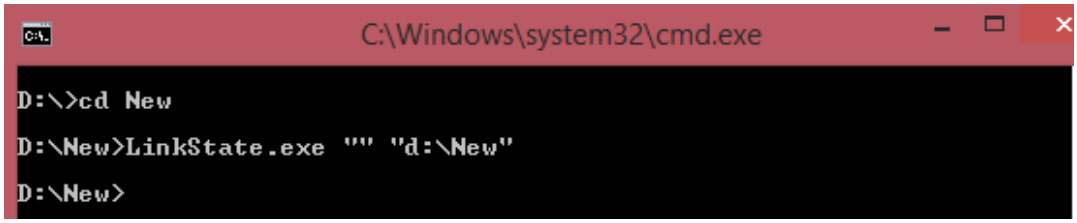
Open command prompt, and go to the folder where the .exe file is saved.

There type

<EXE Name> "<blank>" "<Path where Input.txt is saved>";

For Example:

In this example, the exe file **LinkState.exe** and **Input.txt** is present at "D:\New". So in command prompt go to the folder where exe file is saved.



NOTE: For Chat programming exercise, type

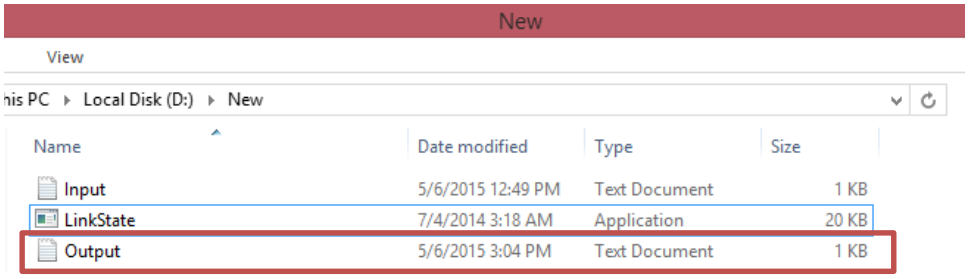
For Sender: <EXE Name> "<blank>" "<Path where Input.txt is saved>" "send"

For Receiver: <EXE Name> “<blank>“ “<Path where Input.txt is saved>” “receive”

Step 5: Check the output.txt file

After running in the command prompt, Output.txt file will be created at the same location where Input.txt file is saved. User can compare and check the content of Output.txt file with the NetSim user manual.

For Example: Here according to this example, the Output.txt file will be created at “D:\New” folder. User can open the file and compare the content with NetSim user manual.



11 Troubleshooting in NetSim

11.1 CLI mode

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

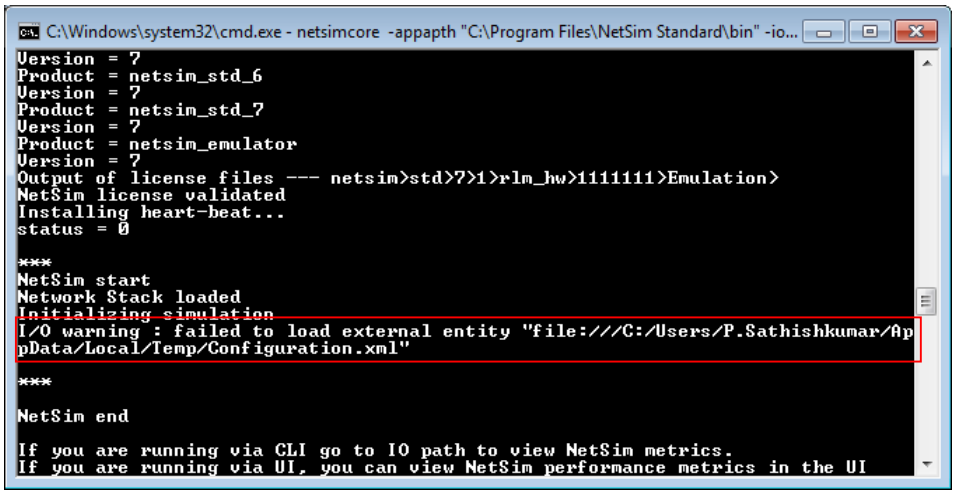
> ***Note:*** *While running NetSim via CLI, try to ensure that there are no errors in the Configuration.xml file. The file, ConfigLog.txt, written to the windows temp path would show errors, if any, found by NetSim's config parser.*

This section discusses some common issues and solutions:

Possible errors when running NetSim via CLI

11.1.1 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.xml file then the following error is thrown

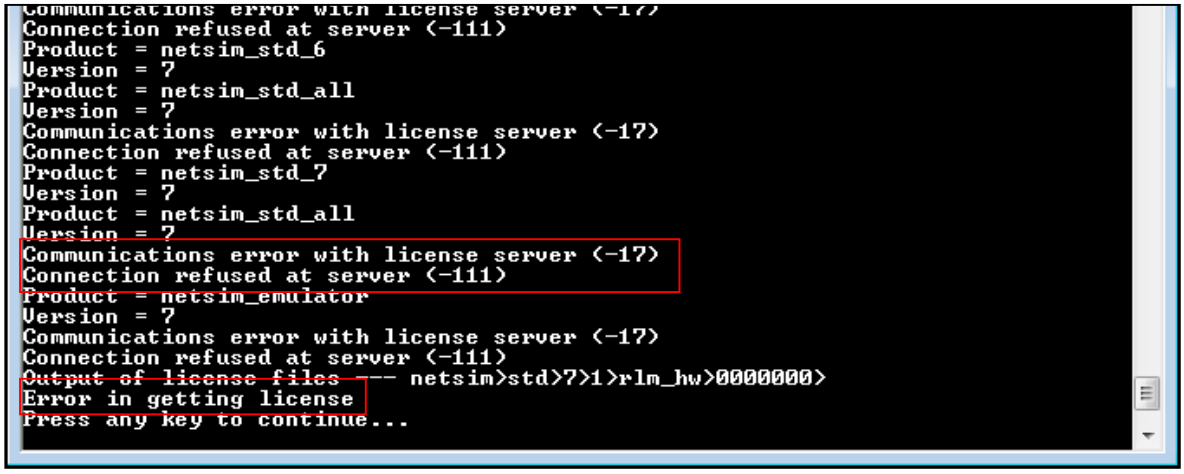


Solution: Please check the I/O path.

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

Wrong License information

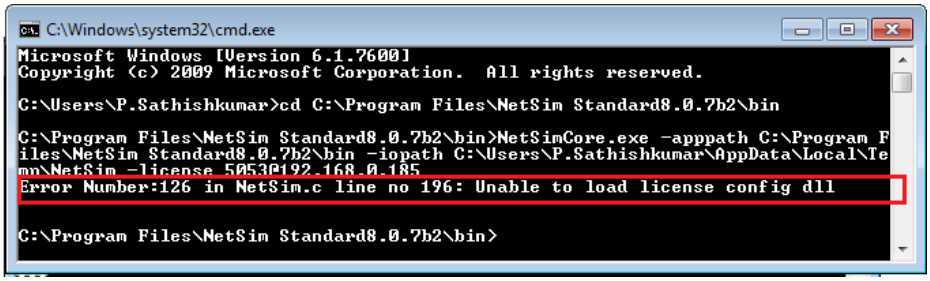
Reason: If the license information is wrong then the following message will be shown.



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.

11.1.3 Unable to load license config dll(126) problem:

Apppath and I/O path have white spaces

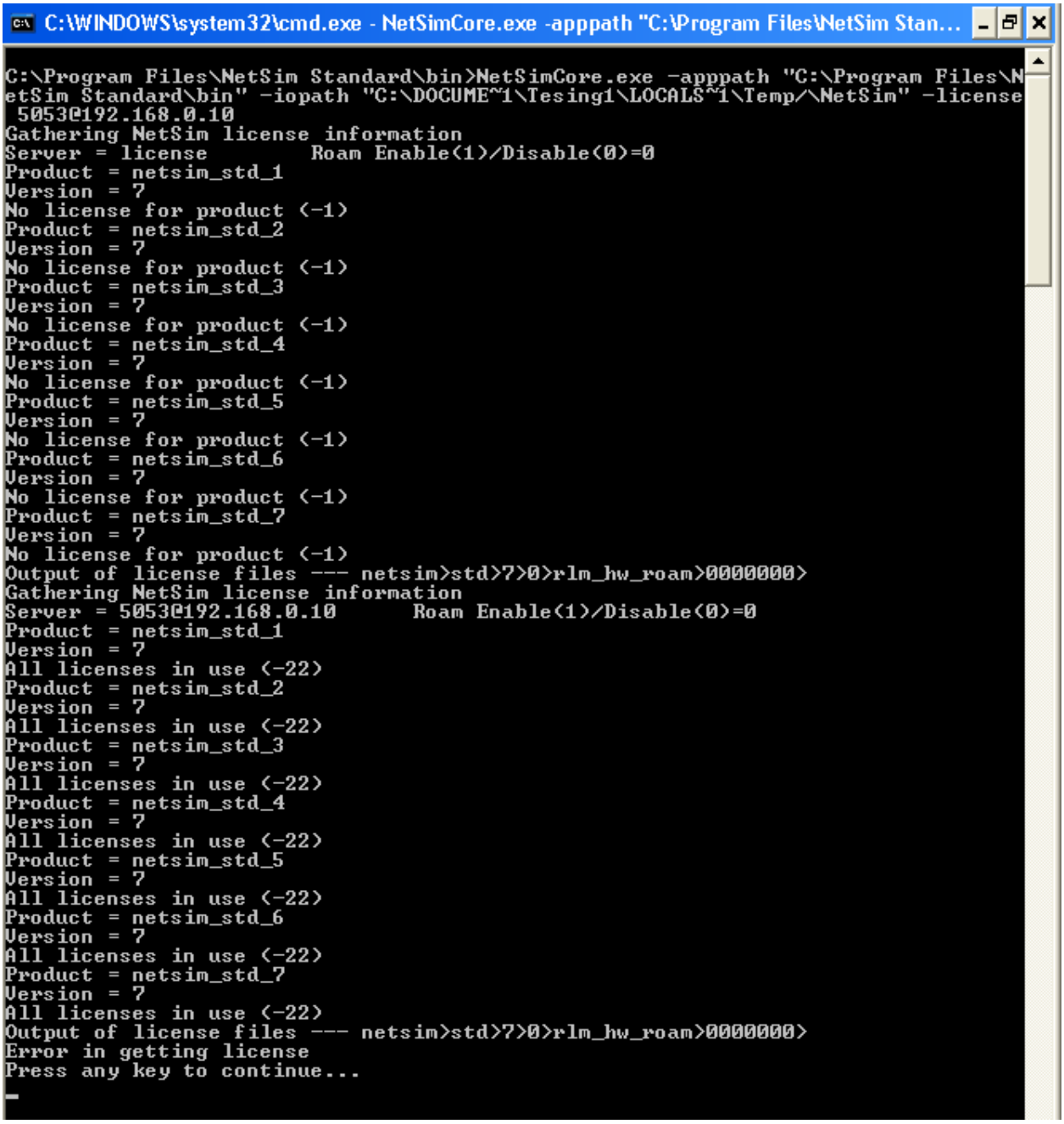


Solution: If the folder name contains white space, then mention the folder path within double quotes while specifying the folder name in the command prompt. For example, if app path contains white space, then the app path must be mentioned within double quotes in the command prompt.

11.1.4 “Error in getting License” error in CLI mode:

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

Example:



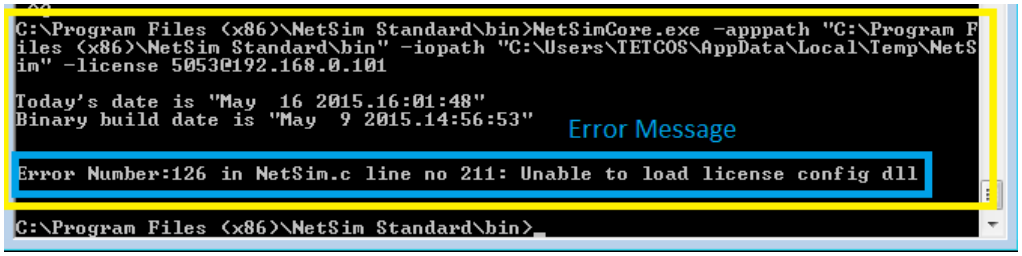
Solution:

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.

So, 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.

11.1.5 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.



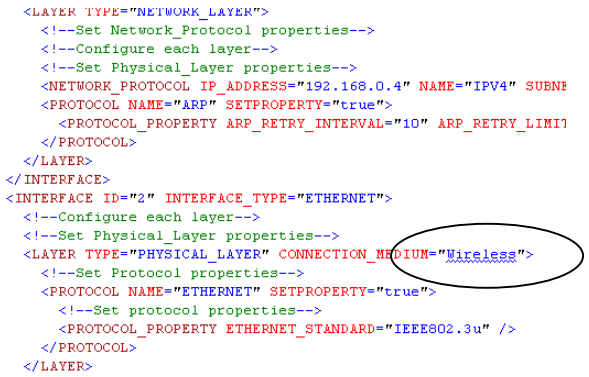
Solution: Type the command manually or copy paste the command/iopath from notepad.

11.2 Configuration.xml

11.2.1 Blue zigzag lines 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 in zigzag 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.

11.2.2 Red zigzag lines in configuration file attributes:

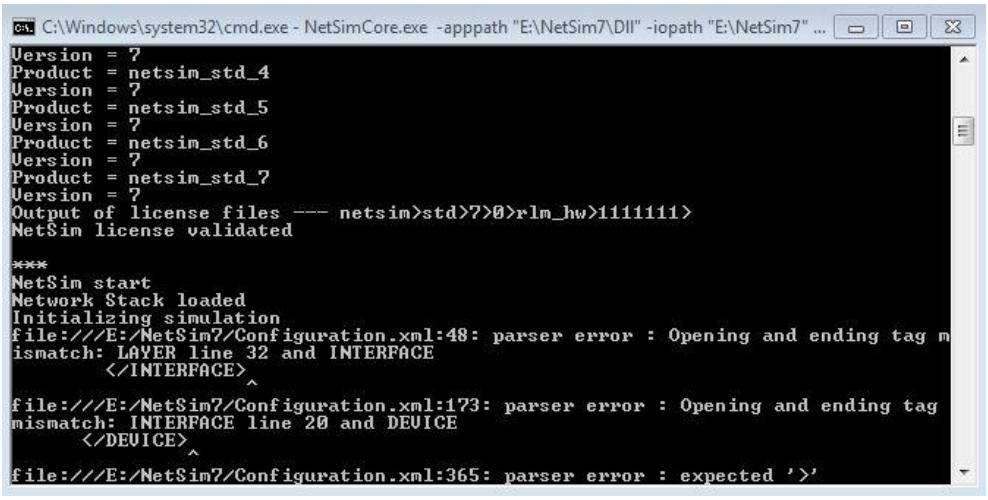
Simulation does not commence and error is displayed at the command prompt. Also, Zigzag lines appearing at the tag specifying the Layer in the Configuration file

***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

11.2.3 Zigzag lines appearing at configuration.xsd in the Configuration file:

Zigzag lines appearing at configuration.xsd in the Configuration file

Reason*:* This issue arises when the schema and the configuration file are not in the same folder.

```
--Config file-->
--Always mention xml schema file in first line of configuration.xml-->
ETCOS_NETSIM xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance" ns0:noNamespaceSchemaLocation="Configuration.xsd"
<EXPERIMENT_INFORMATION>
  <VERSION>NetSim Standard 7.0</VERSION>
  <USER>Admin</USER>
  <NAME>NetSim</NAME>
  <DATE>Thu Nov 05 14:37:02 IST 2012</DATE>
  <COMMENT>NetSim Configuration File</COMMENT>
</EXPERIMENT_INFORMATION>
<NETWORK_CONFIGURATION>
  <!--The following section of config file is used to set the device properties-->
  <DEVICE_CONFIGURATION DEVICE_COUNT="8">
    <!--Set Device properties-->
    <DEVICE DEVICE_NAME="MS1" DEVICE_ID="1" INTERFACE_COUNT="1" TYPE="MOBILE_STATION">

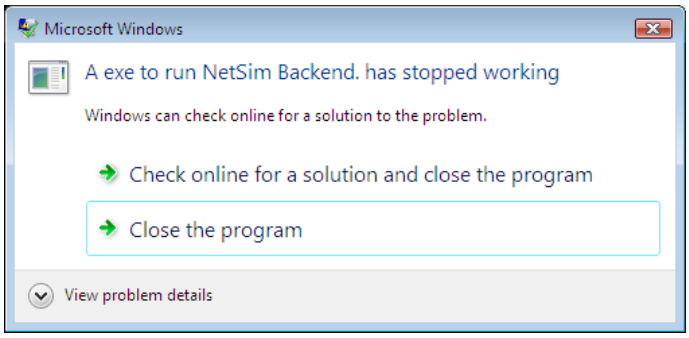
      <!--Set Pos_3D properties-->
      <POS_3D X-POS="157" Y-POS="110" Z-POS="0" />
      <!--Set Interface properties-->
      <INTERFACE ID="1" INTERFACE_TYPE="CDMA">
        <!--Configure each layer-->
        <!--Set Physical_Layer properties-->
        <LAYER TYPE="PHYSICAL_LAYER" CONNECTION_MEDIUM="WIRELESS">
          <!--Set Protocol properties-->
          <PROTOCOL NAME="CDMA"  SETPROPERTY="true">
            <!--Set Protocol properties-->
            <!--
    Unit-
      TRANSMITTED POWER = W
```

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

11.2.4 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.xml or in the dll.

Solution: Check whether the desired scenario has been configured properly in the Configuration.xml.

11.2.5 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.

11.3 Licensing

11.3.1 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.

11.4 Troubleshooting for VANET Simulation

11.4.1 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.

11.4.2 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()
```

11.4.3 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. Backend waits for the Pipes connection to be established.

11.4.4 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.

11.4.5 Backend

- After establishing the connection, Backend checks for GUI flag, send ‘1’ if animation status is online.
- Simulation proceeds. Each time, backend 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 v10

12.1 User modified parameters in device properties will not reflect in newly dropped devices

After dropping nodes on the environment, modification of protocols/ global properties (like Physical Layer parameters, Data Link Layer parameters and others) in one node will reflect in all other nodes. But after modification, if any new node is dropped, the modifications will not be reflected in the newly dropped nodes

Solution: After dropping new nodes, open the properties of any old wireless node and click accept.

12.2 Device properties does not revert to default values

User modifies the default values in a parameter, which is exclusive to a specific protocol/codec, and then changes the protocol/codec using the provided combo box. If the user reverts the combo box value again to the old protocol/codec, then the modified parameter values will be shown instead of the default ones.

12.3 Dynamic Metrics limit is 500 graphs

If Dynamic Metrics is enabled for a network scenario containing more than 500 links or application, then all the graphs may not be displayed in the Performance Metrics Window.

Reason: This is because windows has a limit of 512 on the max number of files that can be open simultaneously

12.4 Running Application between unconnected nodes

Users can create multiple isolated networks in NetSim. But if an application is set having source as a node of an isolated network and destination as a node of another isolated network, NetSim may crash or display zero application throughput.

12.5 RIP Hop count

As per RIP routing protocol, the maximum number of hops/routers it can work from one end to another is 16. But in NetSim, RIP protocol can work across more than 16 routers.

12.6 Default gateway can't be empty

When user manually provides a value in a blank "Default Gateway" parameter in any device, then it cannot be made blank again, which in turn may lead NetSim to crash.

Solution: Users should not enter any value in a blank "default gateway" field in UI.

12.7 Packet size limit in TDMA (Military Radio)

High packet sizes will lead to zero throughputs.

$$\text{Maximum packet size} = \frac{\text{TIME\_SLOT\_BLOCK\_SIZE} * \text{SLOT\_DURATION} * \text{DATA\_RATE}}{8}$$

where SLOT_DURATION is in millisecond and DATA_RATE = 25kbps.

12.8 Packet animation simulation time bar is not functioning properly in WSN, IOT

During Packet animation of Wireless Sensor Network and Internet of Things, the simulation time bar, when dragged to a longer time duration, automatically returns back to a lower value.

12.9 Broadcast application with RTS threshold value less than 1480

If a Network Scenario is created with Broadcast application and RTS threshold value is set less than 1480, the simulation will crash.

12.10 No Radio Range in map view

Users cannot enable radio range when the network is created in map view.

12.11 Scenario crashes if link name field is left empty

If the link name is left empty, then the simulation will crash.

12.12 Incumbent placement at the end of the grid

In CR network, if incumbent is placed at the end of the grid, in packet animation incumbent may move out of the grid.

12.13 Sensors at grid edge with high velocity for agents leads to crash

In WSN, if sensors are placed at the edge of the grid environment and multiple agents are configured with high velocity (> 50 m/s), then simulation may crash.

12.14 IP address issue with connection removal between switch and wired node

If you remove connections between a Wired node to Switch and a Switch to Router, and establish the link between Wired Node to Switch again, IP address of the Wired node may be wrongly configured as to 300.300.300.1 which leads simulation to crash.

12.15 Network layer ACK type will need to be reset in DTDMA

If an existing scenario is opened in DTDMA and new nodes are dropped, parameters will be changed. Hence users must take care to reset Link Layer Ack to Network Layer Ack.

12.16 Wireless link is wrongly established in MANET

When you open a network scenario which is created in MANET with map view, Wireless link will be wrongly established between Wireless nodes.

12.17 In web print we won't get axis value in the plot and in the network metrics table some table headings may be not printed

12.18 VANET_EX1-sumo config file, Throwing error in run simulation console during simulation

In VANET → select the sumo configuration file of EX1 and set the application properties as:

Select BSM for all 5 applications.

Application 1 = Source_id : 2, Destination_id : 3

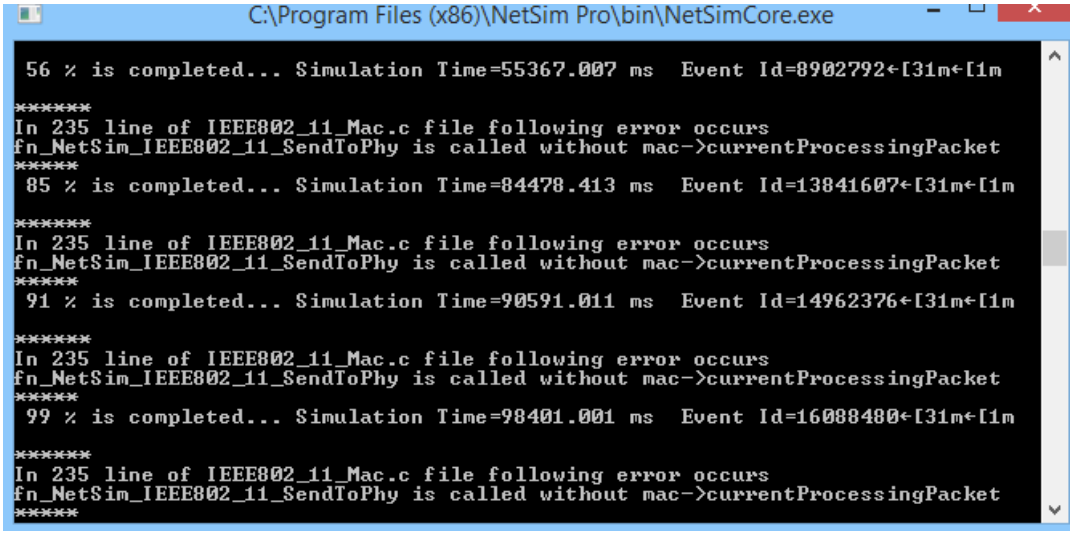
Application 1 = Source_id : 4, Destination_id : 5

Application 1 = Source_id : 6, Destination_id : 7

Application 1 = Source_id : 8, Destination_id : 9

Application 1 = Source_id : 10, Destination_id : 11

After configuring application properties if you run simulation for 100sec it will throw following error in the console.



12.19 In Map View print option is not printing the scenario

In NetSim, if we create a scenario in Map view, after simulation click the print option in animation window. In print, scenario is not available (only the metrics table are available in print).

13 What's new in NetSim v10?

- The Road to 5G - Simulation of LTE covering
    - LTE Advanced
    - LTE Device to Device (D2D)
    - LTE Femto Cell
- Intelligent Transportation Systems — Simulation of Vehicular Adhoc Networks (VANETs) covering
    - IEEE 1609 WAVE
    - Basic Safety Message (BSM) Protocol as per standard J2735 Dedicated Short Range Communications (DSRC) Message Set Dictionary
    - Interface with SUMO for Road Traffic Simulation using TraCI API's
- Accelerate IOT Research - World's only simulator to feature RPL protocol for IOT
    - Routing Protocol for low power and lossy links (RPL) per RFC 6550
- Updated RF Propagation Models
    - Path Loss
        - Friis Free Space Propagation
        - Log Distance
        - HATA Suburban, HATA Urban
        - COST 231 HATA Suburban, COST 231 HATA Urban
        - Indoor Office, Indoor Factory, Indoor Home
    - Shadowing Models: Constant, Lognormal
    - Fading Models: Rayleigh, Nakagami
- TCP enhancements - Optimized congestion control algorithms for high bandwidth networks
    - BIC
    - CUBIC (used in the latest linux kernel)
- Integrated Analytics and Visualization
    - In-built plots for link & applications throughputs
    - Graph formatting - layout, colors, axes, zoom etc
    - Pivot table operations on packet trace

14 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

14.1 R&D projects in NetSim

Example R & D projects in NetSim is available in www.tetcos.com/file-exchange.