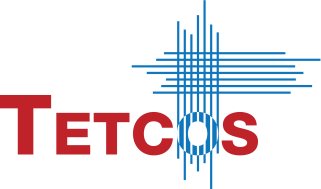


Experiments Manual



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

This manual is for informational purposes only. TETCOS MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

**Warning! DO NOT COPY**

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

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

Rev 12.0 (V), Aug 2019, TETCOS. All rights reserved.

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

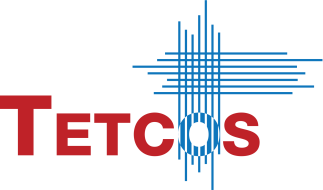
**Contact us at**

TETCOS

# 214, 39th A Cross, 7th Main, 5th Block Jayanagar,

Bangalore - 560 041, Karnataka, INDIA. Phone: +91 80 26630624

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

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

**LIST OF EXPERIMENTS**

[1. Introduction to NetSim 4](#_Toc19893114)

[2. Framing Sequence 12](#_Toc19893115)

[3. Error Detection Code 17](#_Toc19893116)

[4. Study the working and routing table formation of Interior routing protocols, i.e. Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) 26](#_Toc19893117)

[5. Study how throughput and error of a Wireless LAN network changes as the distance between the Access Point and the wireless nodes is varied 35](#_Toc19893118)

[6. How many downloads can a Wi-Fi access point simultaneously handle? 42](#_Toc19893119)

[7. Multi-AP Wi-Fi Networks: Channel Allocation 49](#_Toc19893120)

[8. WiFi: UDP Download Throughput 57](#_Toc19893121)

[9. Understand the working of basic networking commands (Ping, Route Add/Delete/Print, ACL) 67](#_Toc19893122)

[10. Working of IGMP 77](#_Toc19893123)

[11. Understand the working of “Connection Establishment” in TCP 87](#_Toc19893124)

[12. Analyze the performance of FIFO, Priority and WFQ Queuing Disciplines 91](#_Toc19893125)

[13. Understand the working of Slow start and Congestion Avoidance (Old Tahoe), Fast Retransmit (Tahoe) and Fast Recovery (Reno) Congestion Control Algorithms in TCP. 95](#_Toc19893126)

[14. Understand the working of TCP BIC Congestion control algorithm, simulate and plot the TCP congestion window 101](#_Toc19893127)

[15. Quality of Service (QoS) in 802.11e based WLANs 106](#_Toc19893128)

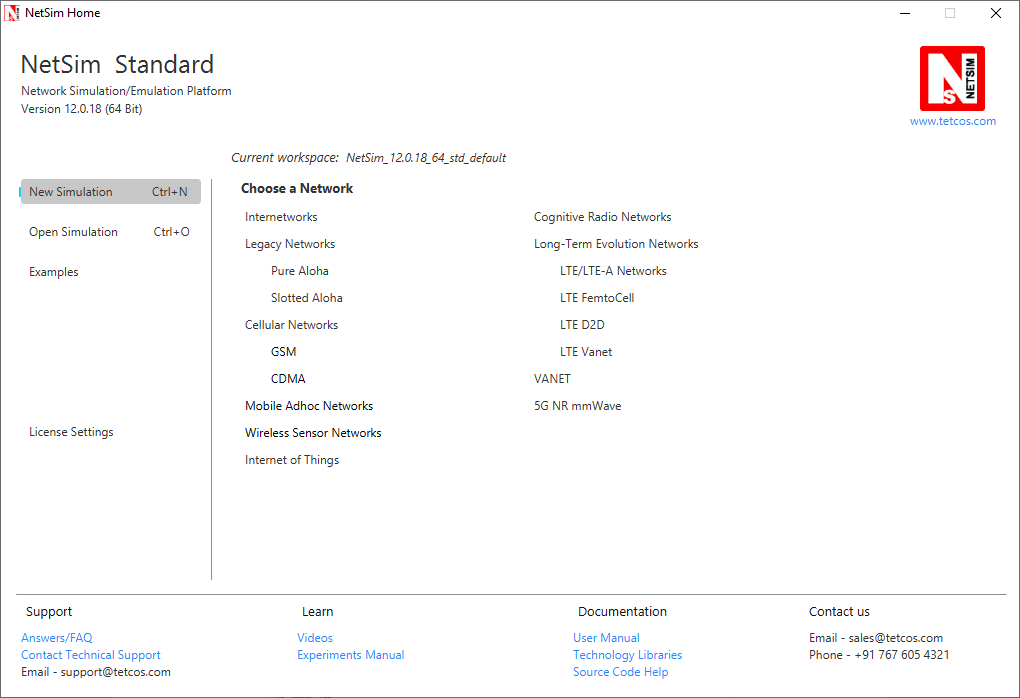
# Introduction to NetSim

## Introduction to network simulation with NetSim, NetSim feature list and NetSim Simulation environment

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

* Internetworks - Ethernet, WLAN, IP, TCP
* Legacy Networks - Aloha, Slotted Aloha
* Cellular Networks - GSM, CDMA
* Mobile Adhoc Networks - DSR, AODV, OLSR, ZRP
* Wireless Sensor Networks - 802.15.4
* Internet of Things - 6LoWPAN gateway, 802.15.4 MAC / PHY, RPL
* Cognitive Radio Networks - 802.22
* Long-Term Evolution Networks – LTE/LTE-A/LTE Femto Cell/LTE D2D/LTE Vanet
* Software Defined Networking
* Advanced Routing and Switching - VLAN, IGMP, PIM, L3 Switch, ACL and NAT
* 5G NR mmWave – LTE NR

NetSim home screen will appear as shown below:



* **Network Design Window:** NetSim design window or the GUI, enables users to model a network comprising of network devices like switches, routers, nodes, etc., connect them through links, and model application traffic to flow through the network. The network devices shown are specific to the network technologies chosen by the user.



**Description:**

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

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

1. **Settings** - Go to Settings 🡪 Environmental Settings and choose the type of environment. Here we have chosen the Environment in the form of a Grid. Map option can be used for specific cases like while designing VANET scenarios.
2. **Help** - Help option allows the users to access all the help features.

* **About NetSim** – Assists the users with basic information like,

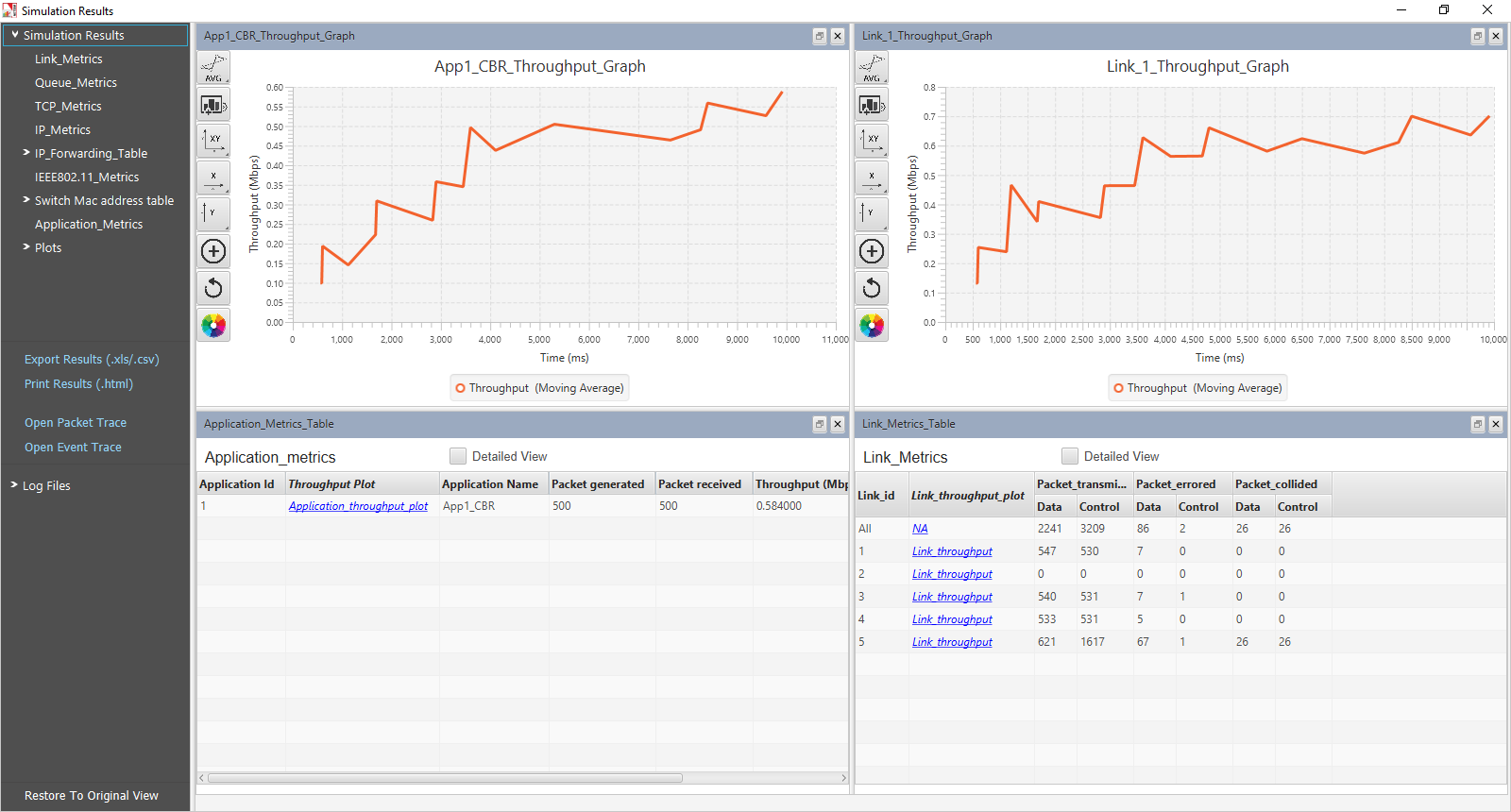
Which version of NetSim is used and whether it is a 32-bit build or 64-bit build?

What kind of License is being used? Whether Floating or Node Locked?

* **Video Tutorials** – Assists the users by directing them to our dedicated YouTube Channel **“TETCOS”**, where we have lots of video presentations ranging from short to long, covering different versions of NetSim up to the latest release.
* **Answers/FAQ** – Assists the user by directing them to our **“NetSim Support Portal”**, where one can find a well-structured **“Knowledge Base”**, consisting of answers or solutions to all the commonest queries which a new user can go through.
* **Raise a Support Ticket** – Assists the user by directing them to our **“NetSim Support Portal”**, where one can **“Submit a ticket”** or in other words raise his/her query, which reaches our dedicated Helpdesk and due support will be provided to the user.
* **User Manual** – Assists the user with the usability of the entire tool and its features. It highly facilitates a new user with lots of key information about NetSim.
* **Source Code Help** – Assists the user with a structured documentation for **“NetSim Source Code Help”**, which helps the users who are doing their R&D using NetSim with a structured code documentation consisting of more than 5000 pages with very much ease of navigation from one part of the document to another.
* **Open Source Code** – Assists the user to open the entire source codes of NetSim protocol libraries in Visual Studio, where one can start initiating the debugging process or performing modifications to existing code or adding new lines of code. Visual Studio Community Edition is a highly recommended IDE to our users who are using the R&D Version of NetSim.
* **Experiments** – Assists the user with separate links provided for 30+ different experiments covering almost all the network technologies present in NetSim.
* **Technology Libraries** – Assists the user by directing them to a folder comprising of individual technology library files comprising all the components present in NetSim.

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

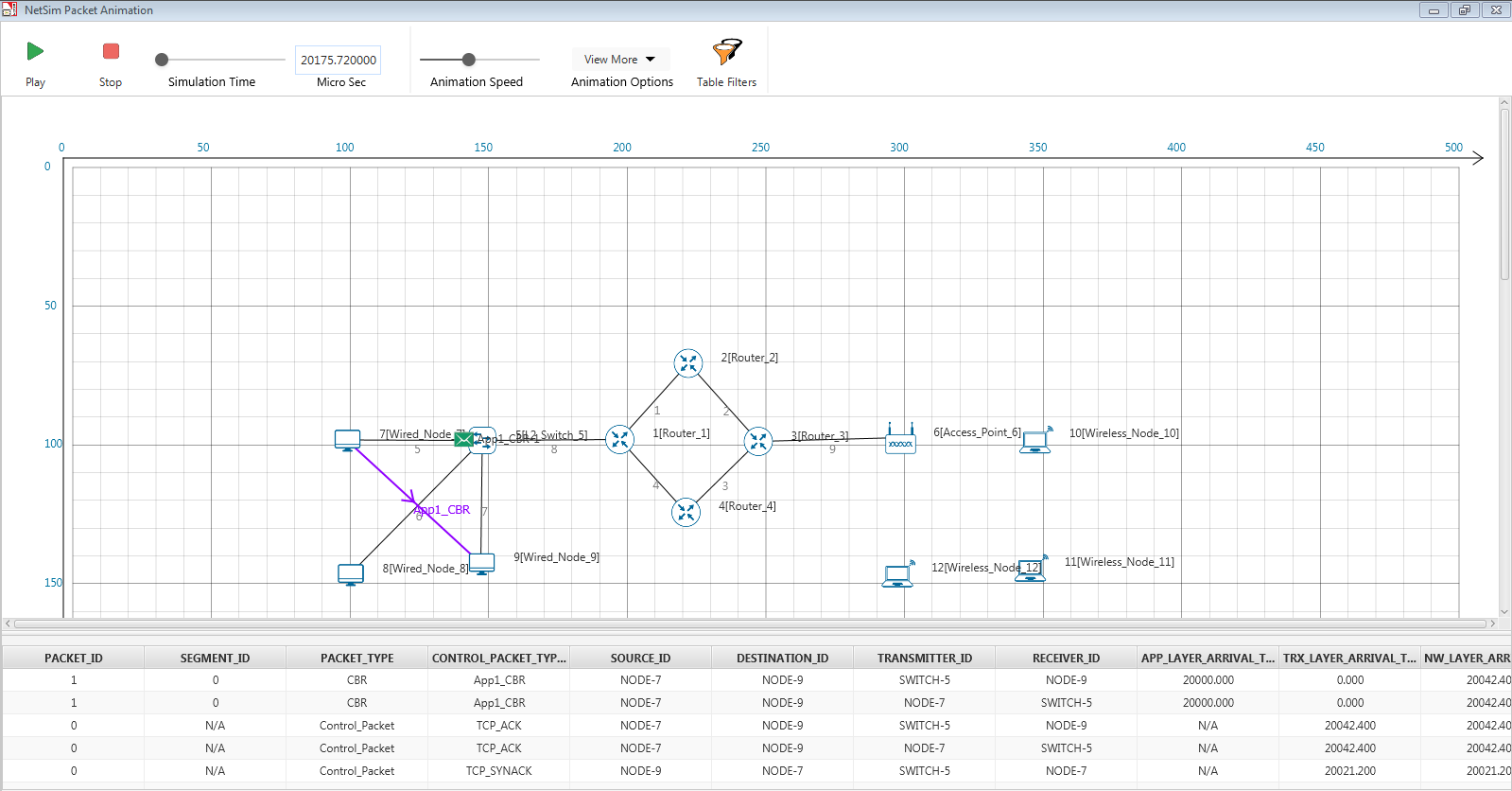
* Click and drop network devices and right click to edit properties
* Click on Wired/Wireless links to connect the devices to one another. It automatically detects whether to use a Wired/Wireless link based on the devices we are trying to connect
* Click on Application to configure different types of applications and generate traffic
* Click on Plots, Packet Trace, and Event Trace and click on the enable check box option which appears in their respective windows to generate additional metrics to further analyze the network performance.
* Click on Run to perform the simulation and specify the simulation time in seconds.
* Next to Run, we have View Animation and View Results options. Both the options remain hidden before we run the simulation or if the respective windows are already open.
* Display Settings option is mainly used to display various parameters like Device Name, IP, etc., to provide a better understanding especially during the design and animation.
* **Results Window:** Upon completion of simulation, Network statistics or network performance metrics reported in the form of graphs and tables. The report includes metrics like throughput, simulation time, packets generated, packets dropped, collision counts etc.



**Description:**

1. Below Simulation Results, clicking on a particular metrics will display the respective metrics window.
2. Clicking on links in a particular metrics will display the plot in a separate window
3. Enabling Detailed View by clicking on it will display the remaining properties
4. Clicking on Restore to Original View will get back to the original view
5. Click on Open Packet Trace / Open Event Trace to open the additional metrics which provide in depth analysis on each Packets / Events.

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

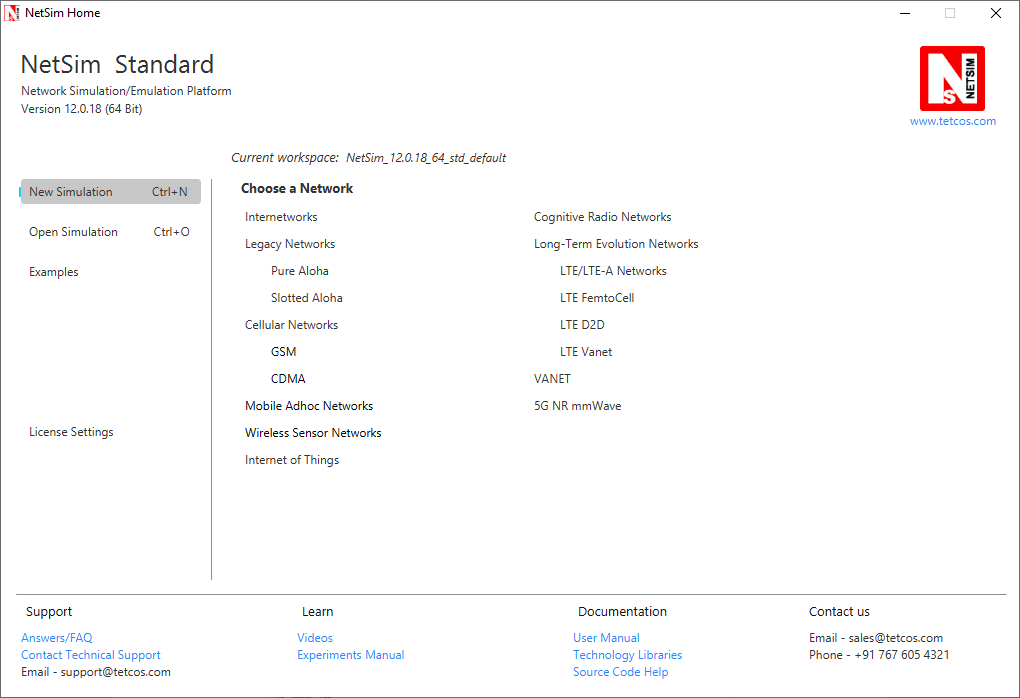


**Description:**

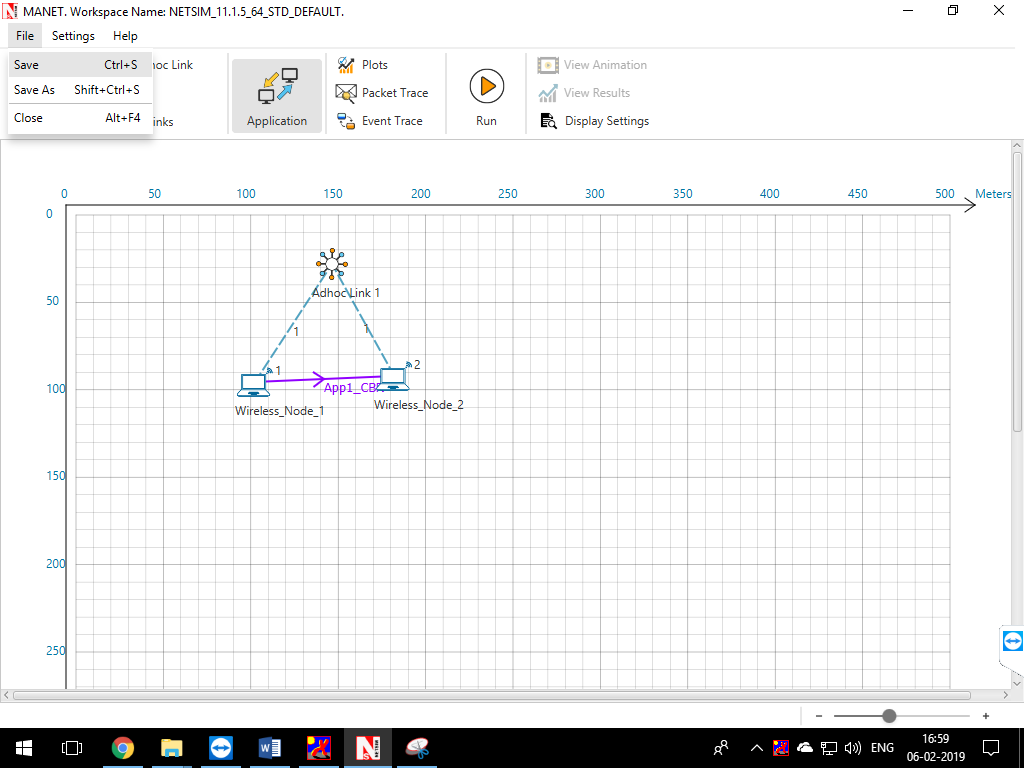
1. Click on Play to view the animation. You can Pause the animation at any interval and Play again.
2. Click on Stop to stop the animation. Now click on Play to start the animation from the beginning.
3. Next to that we also have speed controllers to increase/decrease Simulation Time and Animation Speed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

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

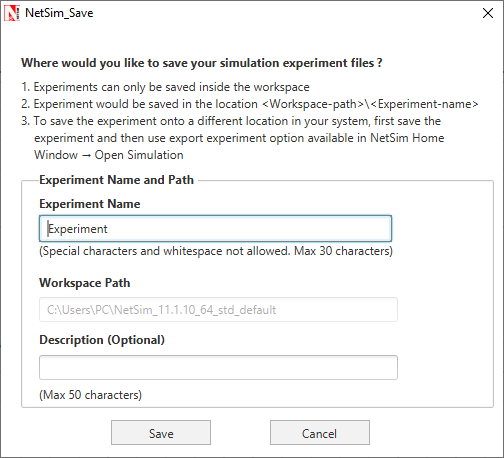
To create an experiment, select New Simulation-> <Any Network> in the NetSim Home Screen.



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

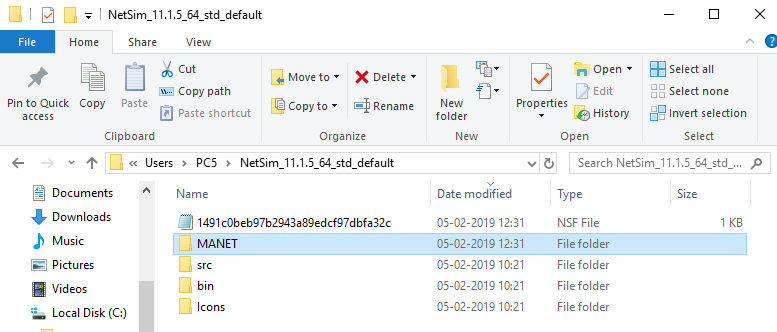


A save popup window appears which contains Experiment Name, Folder Name, Workspace path and Description.

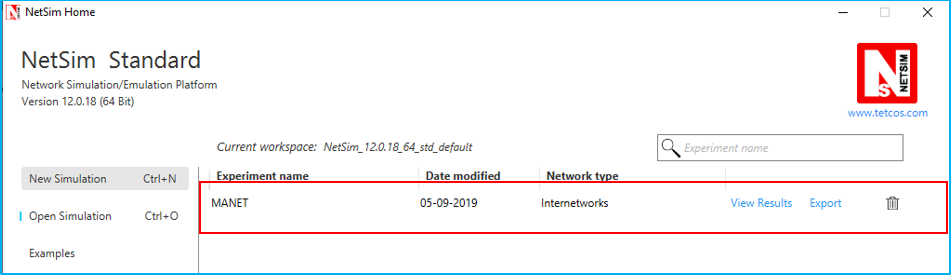


Specify the Experiment Name and Description (Optional) and then click on Save. The workspace path is non-editable. Hence all the experiments will be saved in the default workspace path. After specifying the Experiment Name click on Save.

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



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



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

## Typical sequence of steps to do experiments in this manual

The typical steps involved in doing experiments in NetSim are,

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

***NOTE: Example Configuration files for all experiments would available where NetSim has been installed. This directory is (<NetSim\_Install\_Directory>\Docs\Sample\_Configuration\NetSim\_Experiment\_Manual)***

# 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=For Random value generation  Example:  Destination Address=00011111  Source Address=00111111  Data=Hello  CRC Polynomial=10011  Error Status=0  Seed Value=0 | 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=0100100001100101011011000110110001101111>  CRC Polynomial=10011>  CheckSumSender=1110>  <Stuffing>  Destination Address=000111110>  Source Address=001111101>  Data=010010000110010101101100011011000110111110110>  <DeStuffing>  Destination Address=00011111>  Source Address=00111111>  Data=01001000011001010110110001101100011011111110>  Error Status=0>  CheckSumReceiver=0000>  BinaryValues=0100100001100101011011000110110001101111>  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.

## 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.

# Error Detection Code

## 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,
  1. **SampleMode** should be selected.
  2. **SelectCRC12** as **Algorithm** from the list available.
  3. Under **Condition**, “**NoError**” should be selected.
  4. 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.
  5. **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,
  1. **SampleMode** should be selected.
  2. **SelectCRC12** as **Algorithm** from the list available.
  3. Under **Condition**, “**Error**” should be selected.
  4. 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.
  5. **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**”.

## 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,
  + **SampleMode** should be selected.
  + **SelectCRC16** as **Algorithm** from the list available.
  + Under **Condition**, “**No Error**” should be selected.
  + 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.
  + **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,
  + **SampleMode** should be selected.
  + **SelectCRC16** as **Algorithm** from the list available.
  + Under **Condition**, “**Error**” should be selected.
  + 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.
  + **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**”.

## 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,
  1. **SampleMode** should be selected.
  2. **SelectCRC32** as **Algorithm** from the list available.
  3. Under **Condition**, “**NoError**” should be selected.
  4. 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.
  5. **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,
  1. **SampleMode** should be selected.
  2. **SelectCRC32** as **Algorithm** from the list available.
  3. Under **Condition**, “**Error**” should be selected.
  4. 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.
  5. **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**”.

## 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,
  1. **SampleMode** should be selected.
  2. **SelectCRCCCITT** as **Algorithm** from the list available.
  3. Under **Condition**, “**NoError**” should be selected.
  4. 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.
  5. **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,
  1. **SampleMode** should be selected.
  2. **SelectCRCCCITT** as **Algorithm** from the list available.
  3. Under **Condition**, “**Error**” should be selected.
  4. 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.
  5. **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**”.

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

## Introduction:

**RIP**

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

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

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

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

**OSPF**

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

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

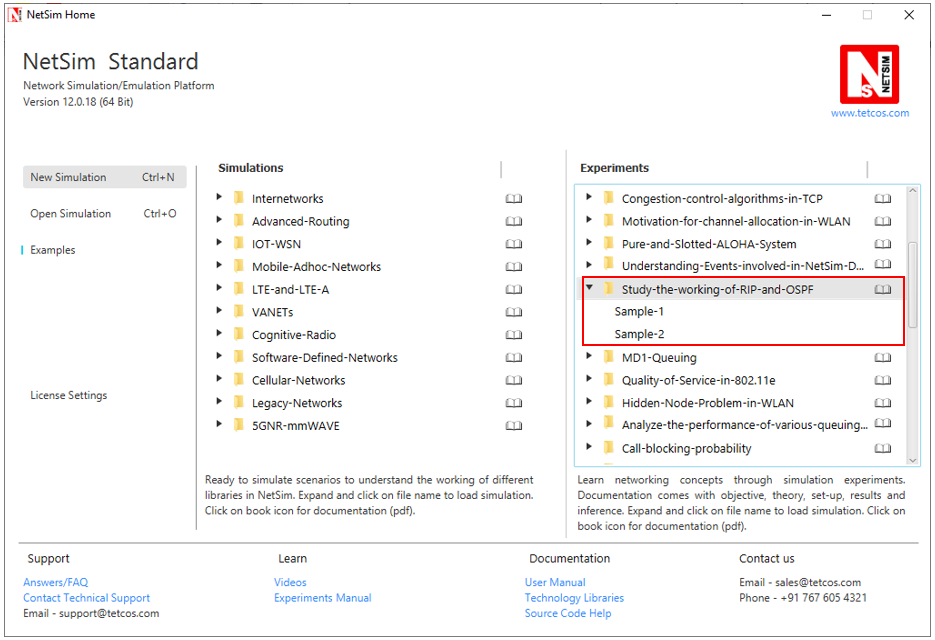
In OSPF, each router has a link state database which is tabular representation of the topology of the network (including cost). Using Dijkstra algorithm each router finds the shortest path between source and destination.

**Formation of OSPF Routing Table**

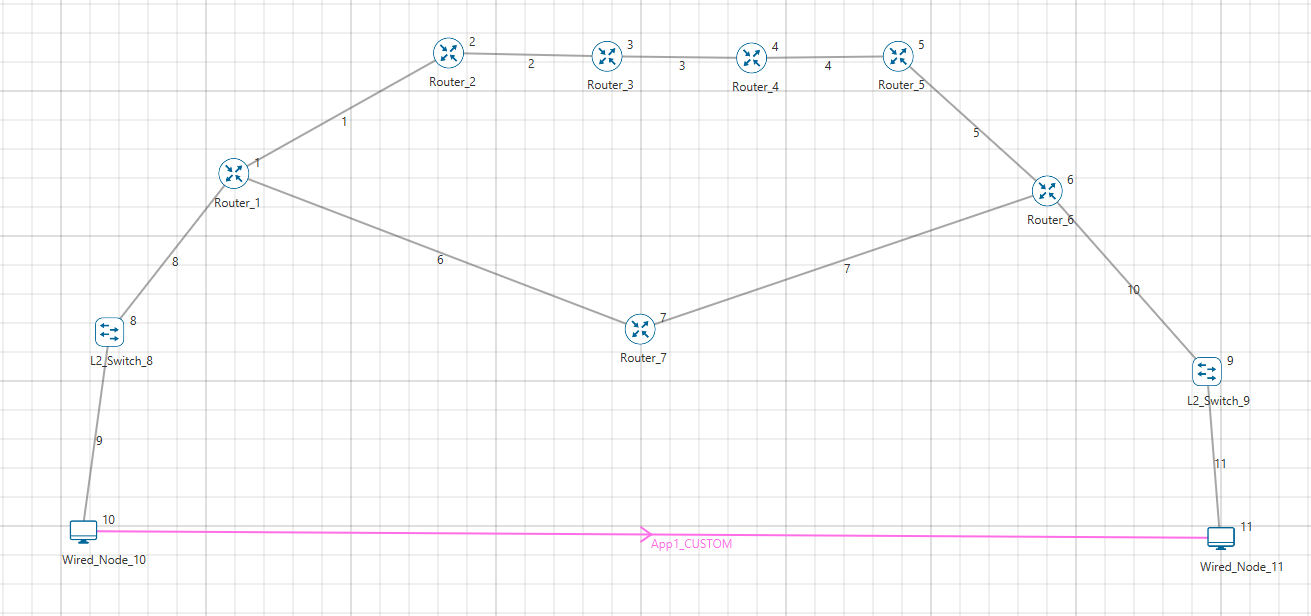
1. OSPF-speaking routers send Hello packets out all OSPF-enabled interfaces. If two routers sharing a common data link agree on certain parameters specified in their respective Hello packets, they will become neighbors.
2. Adjacencies, which can be thought of as virtual point-to-point links, are formed between some neighbors. OSPF defines several network types and several router types. The establishment of an adjacency is determined by the types of routers exchanging Hellos and the type of network over which the Hellos are exchanged.
3. Each router sends link-state advertisements (LSAs) over all adjacencies. The LSAs describe all of the router's links, or interfaces, the router's neighbors, and the state of the links. These links might be to stub networks (networks with no other router attached), to other OSPF routers, or to external networks (networks learned from another routing process). Because of the varying types of link-state information, OSPF defines multiple LSA types.
4. Each router receiving an LSA from a neighbor records the LSA in its link-state database and sends a copy of the LSA to all of its other neighbors.
5. By flooding LSAs throughout an area, all routers will build identical link-state databases.
6. When the databases are complete, each router uses the SPF algorithm to calculate a loop-free graph describing the shortest (lowest cost) path to every known destination, with itself as the root. This graph is the SPF tree.
7. Each router builds its route table from its SPF tree

## Network Setup:

Open NetSim and click **Examples > Experiments > Study-the-working-of-RIP-and-OSPF > Sample-1** as shown below:



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



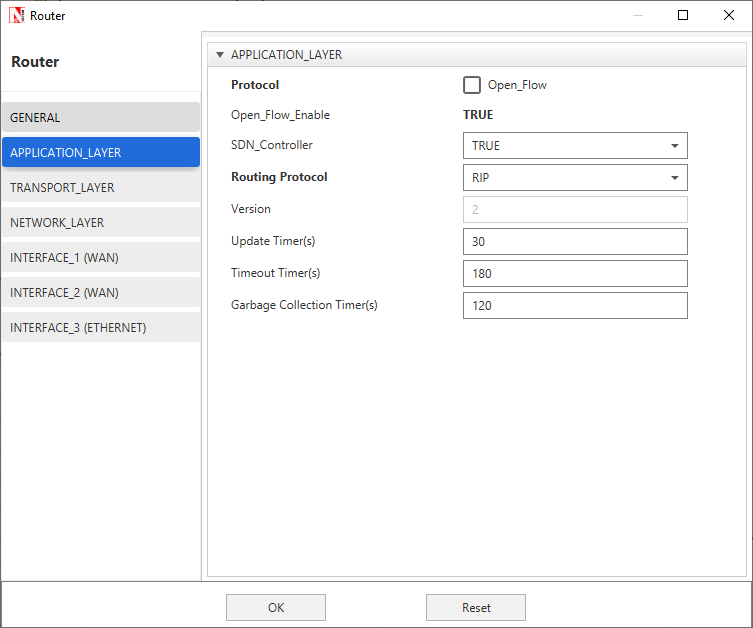
## Procedure:

**Sample 1:**

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

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

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



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

**Step 3:** In the Source Node, i.e. Wired Node 10, TCP Protocol in the Transport Layer is disabled.

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

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

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

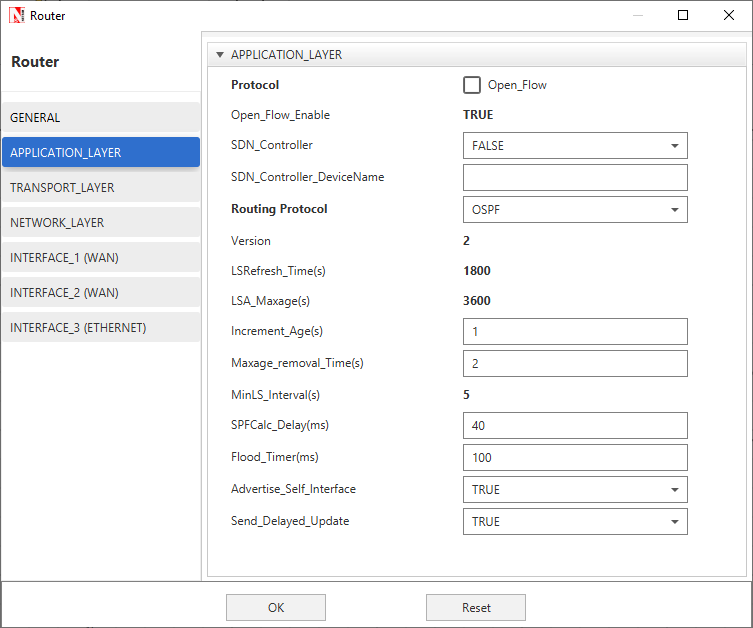
**Step 6:** Run the Simulation for 100 Seconds.

**Sample 2:**

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

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

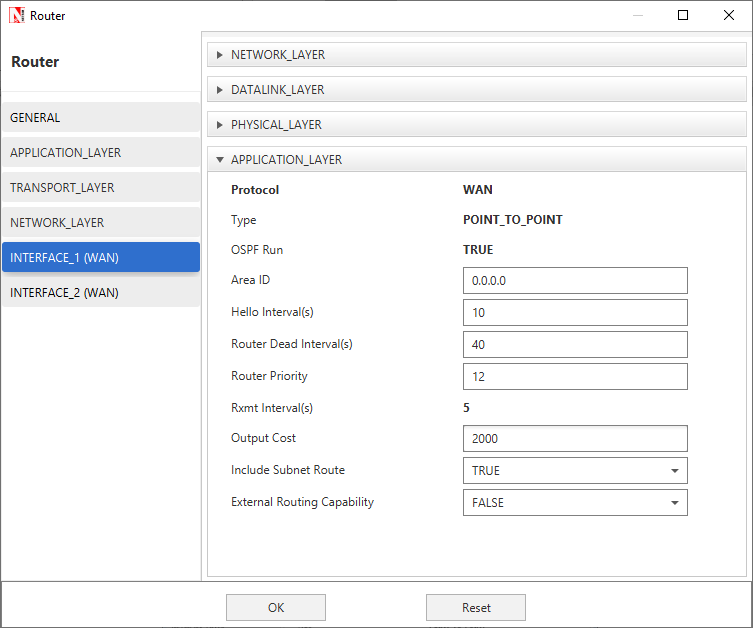
**Step 2:** Go to Router 1 Properties. In the Application Layer, Routing Protocol is set as OSPF.



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

**Step 3:** In the Source Node, i.e. Wired Node 10, TCP Protocol in the Transport Layer is disabled.

**Step 4:** Go to Router 7 Properties. In both the WAN Interfaces, the Output Cost is set to 2000.



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

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

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

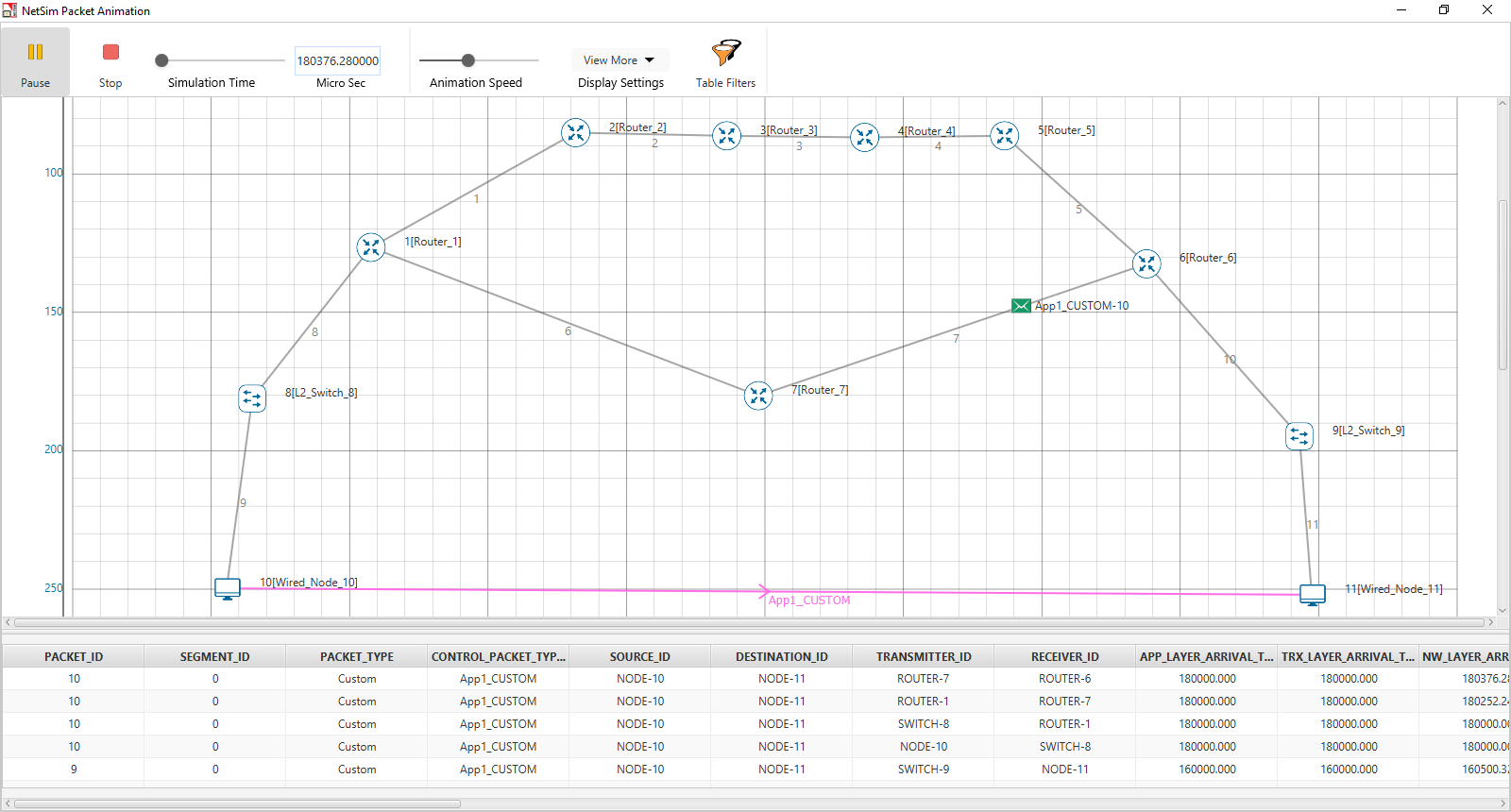
Additionally, the “**Start Time (s)**” parameter is set to 30, while configuring the application. This time is usually set to be greater than the time taken for OSPF Convergence (i.e. Exchange of OSPF information between all the routers), and it increases as the size of the network increases.

**Step 6:** Packet Trace is enabled, and hence we are able to track the route which the packets have chosen to reach the destination based on the Open Shortest Path First Routing Protocol that is set.

**Step 7:** Run the Simulation for 100 Seconds.

## Output I:

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

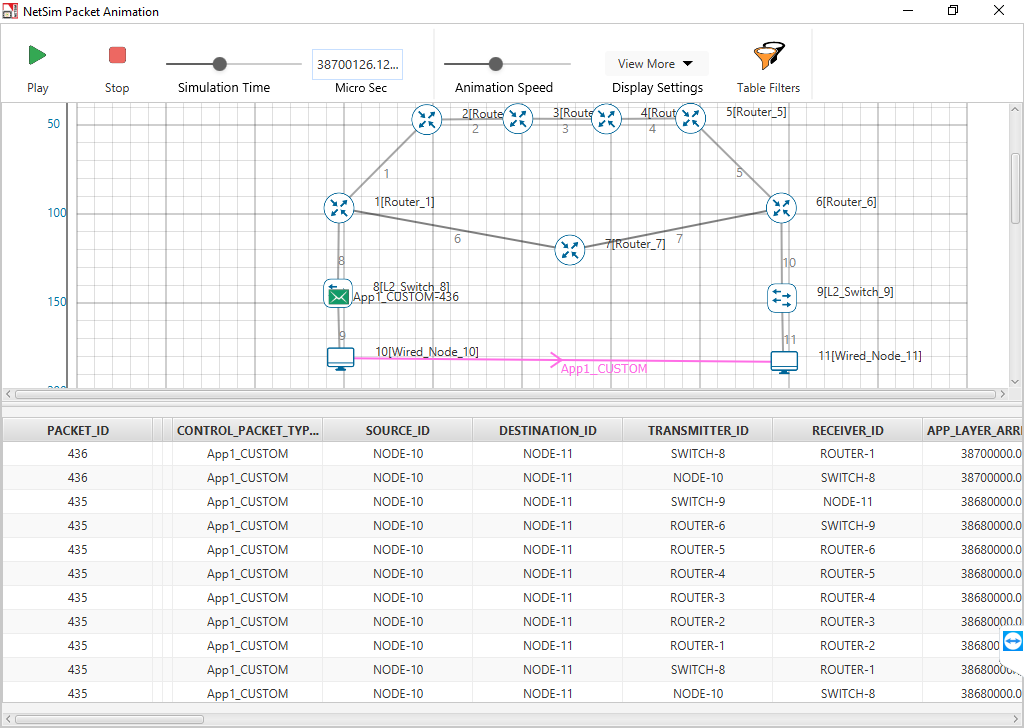


Users can view the same in Packet Trace.

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

## Output II:

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



Users can view the same in Packet Trace.

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

## Inference:

**RIP**

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

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

In NetSim the Routing Table Formation has 3 stages,

1. **Initial Table:** The Initial Table will show the direct connections made by each Router.
2. **Intermediate Table:** The Intermediate Table will have the updates of the Network in every 30 seconds
3. **Final Table:** The Final Table is formed when there is no update in the Network.

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

**OSPF**

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

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

**Compiling the LSDB**

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

**Calculating the SPF Tree Using Dijkstra's Algorithm**

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

**Calculating the Routing Table Entries from the SPF Tree**

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

If the application start time isn't changed then,

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

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

# Study how throughput and error of a Wireless LAN network changes as the distance between the Access Point and the wireless nodes is varied

## Introduction

In this experiment we will study the physical layer standard for IEEE 802.11b WiFi. A physical layer standard (abbreviated as PHY standard) defines the mechanism by which logical information bits are transmitted over the wireless channel that has been allotted to the WiFi system. WiFi systems are confined to working in an approximately 80MHz bandwidth in the 2.4GHz ISM band. Within this bandwidth, any particular WiFi Access Point (AP) must choose to work in one of 13 channels, each of nominal bandwidth 22MHz. In this experiment, we aim to study how the packet error performance of an IEEE 802.11b AP-STA connection varies as the distance between the AP and the STA varies.

## Background

The IEEE 802.11b standard defines 4 digital modulation schemes for such channels. All are based on Direct Sequence Spread Spectrum (DSSS) with a chipping rate of 11 million chips per second (11Mcps). An 11 chip Barker code yields 1 million symbols per second (1Msps). These symbols are Differential Phase Shift Keying modulated to get 1 bit per symbol, thereby yielding 1Mbps, and Quarternary Differential Phased Keying modulated to get 2 bits per symbol, thereby yielding 2Mbps. In order to get 5.5Mbps and 11Mbps, each symbol is made from 8 chips, so that the symbol rate is 1.375Msps. A technique called Complementary Code Keying (CCK) then provides 4 bits per symbol, yielding 5.5Mbps, and 8 bits per symbol, which yields 11Mbps.

A simple qualitative fact is that, for a given signal to noise ratio at the receiver, as the modulation scheme attempts to send more bits per second, the bit error probability increases. This happens because, as the bit rate increases, the bit sequences that the receiver needs to distinguish between become closer packed, so that bit errors become harder to resolve. The signal to noise ratio (SNR) at the receiver depends on the transmission power, the attenuation of power from the transmitter to the receiver, and noise power.

Where is the noise power spectral density (W/Hz) and is the system bandwidth (nominally 22Mhz). The noise power works out to approximately dBm. The received power is obtained by subtracting the *path-loss,* between the transmitter and the receiver, from the transmitted power (e.g., dBm, would arise from a transmit power of 1mW). A simple expression for path-loss is given by

where is the path loss at the “reference” distance of 1m, is the path-loss exponent and is the distance between the transmitter and the receiver. It may be noted that this deterministic expression ignores random phenomena such as “shadowing” and “fading.”

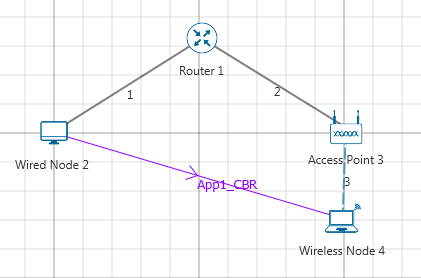
As increases, the received power decreases; e.g., doubling the distance reduced the received power by approximately , since . Typical values of , indoors, could be 3 to 5, resulting in 9dB to 15dB additional path loss for doubling the value of .

## Network Setup

Open NetSim and click **Examples > Experiments > Study-throughput-and-error-of-a-WLAN-varies-with-distance > Sample-1** as shown below:



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



## Procedure:

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

**Step 1:** A network scenario is created in NetSim GUI comprising of 1 Wired Node, 1 Router, 1 Access Point and 1 Wireless Node.

**Step 2:** In the Destination Node, i.e. Wireless Node 4, the Interface 1 (WIRELESS) > Physical Layer, Protocol Standard is set to IEEE802.11b and in the Datalink Layer, Rate Adaptation is set to False.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Wireless Node 4 Properties | | Access Point |
| X/Lat | | 200 | 200 |
| Y/Lon | | 30 | 0 |
| Interface\_Wireless properties | | | |
| IEEE Standard | | 802.11b | 802.11b |
| Rate \_Adaptation | | False | False |

**Step 3:** In the Source Node, i.e. Wired Node 2 and in the Destination Node, i.e. Wireless Node 4, TCP Protocol in the Transport Layer is disabled.

**Step 4:** Wired and Wireless links properties are set as follows:

|  |  |
| --- | --- |
| Wireless Link Properties | |
| Channel Characteristics | Path loss only |
| Path Loss Model | Log\_Distance |
| Path Loss Exponent | 3 |

|  |  |
| --- | --- |
| Wired Link Properties | |
| Uplink Speed (Mbps) | 100 |
| Downlink Speed (Mbps) | 100 |
| Uplink BER | 0.0000001 |
| Downlink BER | 0.0000001 |

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

A CBR Application is generated from Wired Node 2 i.e. Source to Wireless Node 4 i.e. Destination with Packet Size set to 1450 Bytes and Inter Arrival Time set to 770 µs. It is set such that, the **Generation Rate** equals to 15 Mbps.

**Step 6:** Packet Trace is enabled in NetSim GUI. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis.

Similarly do the other samples by varying the distance between Access Point and Wireless Node as 60, 85, 90, 100, 110, 115, 180, 260, 360, 400, 420, 440, 460, 480, and 500 m.

## Output:

Note down the values of Data rate and Throughput for all the samples and compare with IEEE standards

Phy rate can be calculated using packet trace by using the formula shown below:

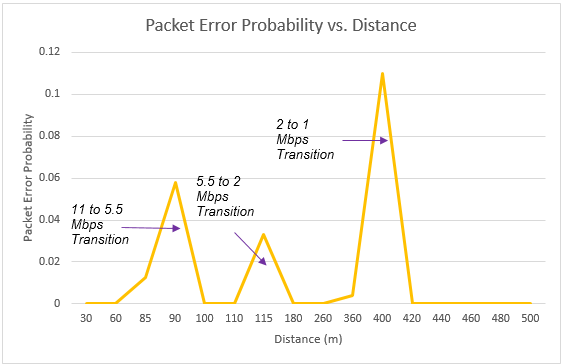
192 micro seconds is the approximate preamble time for 802.11b

Calculate PHY rate for all the data packets coming from Access Point to Wireless node. For doing this please refer section 7.5.1 How to set filters to NetSim Packet Trace file from NetSim’s User Manual. Filter Packet Type to CBR, Transmitter to Access Point and Receiver to Wireless node.

Since where PER is packet error rate, PL is packet length in bits and BER is bit error rate, we get

On tabulating the results, you would see

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Distance (m) | 802.11b | | | | |
| **PHY rate in Mbps**  **(Channel capacity)** | **Application Throughput (Mbps)** | **Packets Transmitted** | **Packets Errored** | **Packet error probability** |
| 30 | 11 | 5.9276 | 5110 | 0 | 0 |
| 60 | 11 | 5.9276 | 5110 | 0 | 0 |
| 85 | 11 | 5.842920 | 5101 | 64 | 0.0125 |
| 90 | 11 | 5.53204 | 5063 | 294 | 0.058 |
| 100 | 5.5 | 3.78856 | 3266 | 0 | 0 |
| 110 | 5.5 | 3.78856 | 3266 | 0 | 0 |
| 115 | 5.5 | 3.64588 | 3253 | 110 | 0.033 |
| 180 | 2 | 1.6762 | 1445 | 0 | 0 |
| 260 | 2 | 1.6762 | 1445 | 0 | 0 |
| 360 | 2 | 1.66808 | 1445 | 7 | 0.004 |
| 400 | 2 | 1.48016 | 1436 | 160 | 0.110 |
| 420 | 1 | 0.89204 | 769 | 0 | 0 |
| 440 | 1 | 0.89204 | 769 | 0 | 0 |
| 460 | 1 | 0.89204 | 769 | 0 | 0 |
| 480 | 1 | 0.89204 | 769 | 0 | 0 |
| 500 | 0 | 0 | 0 | 0 | 0 |



***Note: All the above plots highly depend upon the placement of nodes in the simulation environment. So, note that even if the placement is slightly different, the same set of values will not be got but one would notice a similar trend.***

## Inference:

We notice that as the distance increases, the 802.11b PHY rate (channel capacity decreases) decreases. This is because the underlying data rate depends on the received power at the receiver.

RF losses are directly proportional to distance to the power of path loss exponent. As RF propagation losses increase, the received power decreases.

We can see that the rate drops from 11 Mbps to 5.5 Mbps at around 95m, and then to 2 Mbps at 175m and to 1 Mbps at 415m (in this case the path loss exponent is set to 3.0). We also notice how the packet error rate increases with distance, then when the data rate changes (a lower modulation scheme is chosen), the error rate drops. This happens for all the transitions i.e. 11 to 5.5, 5.5 to 2 and from 2 to 1 Mbps. One must note that WLAN involves ACK packets after data transmission. These additional packet transmission lead to reduced Application throughput of 5.9 Mbps (at lower distances) even though the PHY layer data rate is 11 Mbps and the error rates is almost NIL. The application throughput is dependent on the PHY rate and the channel error rate, and one can notice it drops / rise accordingly.

# How many downloads can a Wi-Fi access point simultaneously handle?

## Motivation

Wi-Fi has become the system of choice for access to Internet services inside buildings, offices, malls, airports, etc. In order to obtain access to the Internet over Wi-Fi a user connects his/her mobile device (a laptop or a cellphone, for example) to a nearby Wi-Fi access point (AP). A popular use of such a connection is to download a document, or a music file; in such an application, the user’s desire is to download the file as quickly as possible, i.e., to get a high throughput during the download. It is a common experience that as the number of users connected to an AP increases, the throughput obtained by all the users decreases, thereby increasing the time taken to download their files. The following question can be asked in this context.

If during the download, a user expects to get a throughput of at least bytes per second, what is the maximum number of users (say, up to which the throughput obtained by every user is at least We can say that is the *capacity* of this simple Wi-Fi network for the *Quality of Service (QoS)* objective . [[1]](#footnote-2)

## Objective

In this experiment we will learn how to obtain in a simple WiFi network where the packet loss due to channel errors is 0. In this process we will understand some interesting facts about how WiFi networks perform when doing file transfers.

## Theory

In NetSim, we will set up a network comprising a server that carries a large number of large files that the users would like to download into their mobile devices. The server is connected to a Wi-Fi AP, with the IEEE 802.11b version of the protocol, via an Ethernet switch. Several mobile devices (say, ) are associated with the AP, each downloading one of the files in the server. The Ethernet speed is 100Mbps, whereas the mobile devices are connected to the AP at 11Mbps, which is one of the IEEE 802.11b speeds.

We observe, from the above description, that the file transfer throughputs will be limited by the wireless links between the AP and the mobile devices (since the Ethernet speed is much larger than the Wi-Fi channel speed). There are two interacting mechanisms that will govern the throughputs that the individual users will get:

1. The Wi-Fi medium access control (MAC) determines how the mobile devices obtain access to the wireless medium. There is one instance of the WiFi MAC at each of the mobile devices.
2. The end-to-end protocol, TCP, controls the sharing of the wireless bandwidth between the ongoing file transfers. In our experiment, there will be one instance of TCP between the server and each of the mobile devices.

For simplicity, the default implementation of TCP in NetSim does not implement the delayed ACK mechanism. This implies that a TCP receiver returns an ACK for every received packet. In the system that we are simulating, the server is the transmitter for all the TCP connections, and each user’s mobile device is the corresponding receiver.

Suppose, each of the TCP connection transmits one packet to its corresponding mobile device; then each mobile device will have to return an ACK. For this to happen, the AP must send packets, and each of the mobile devices must send back ACKs. Thus, for the file transfers to progress, the AP needs to packets for each packet (i.e., ACK) returned by each mobile device. We conclude that, in steady state, the AP must send as many packets as all the mobile devices send, thus requiring equal channel access to the AP as to all the mobile devices together.

At this point, it is important to recall that when several nodes (say, an AP and associated mobile devices) contend for the channel, the WiFi medium access control provides fair access at the packet level, i.e., each contending device has an equal chance of succeeding in transmitting a packet over the channel. Now consider the system that we have set up in this present experiment. There are mobile devices associated with one AP. Suppose, for example, of them ( all have a packet to transmit (and none other has a packet). By the fair access property of the WiFi MAC, each of these nodes, along with the AP, has an equal probability of successfully transmitting. It follows, by the packet level fair access property, that each node will have a probability of of succeeding in transmitting its packet. If this situation continues, the channel access ratio to the AP will be inadequate and the equal channel access argued in the previous paragraph will be violated. It follows from this that, on the average, roughly only one mobile device will have an ACK packet in it; the AP will contend with one other node, thus getting half the packet transmission opportunities.

With the just two nodes contending, the collision probability is small (~ 0.06) and the probability of packet discard is negligibly small. Thus, the TCP window for every transfer will grow to the maximum window size. The entire window worth of TCP data packets for the sessions will be in the AP buffer, except for a very small number of packets (averaging to about 1) which will appear as ACKs in the mobile devices.

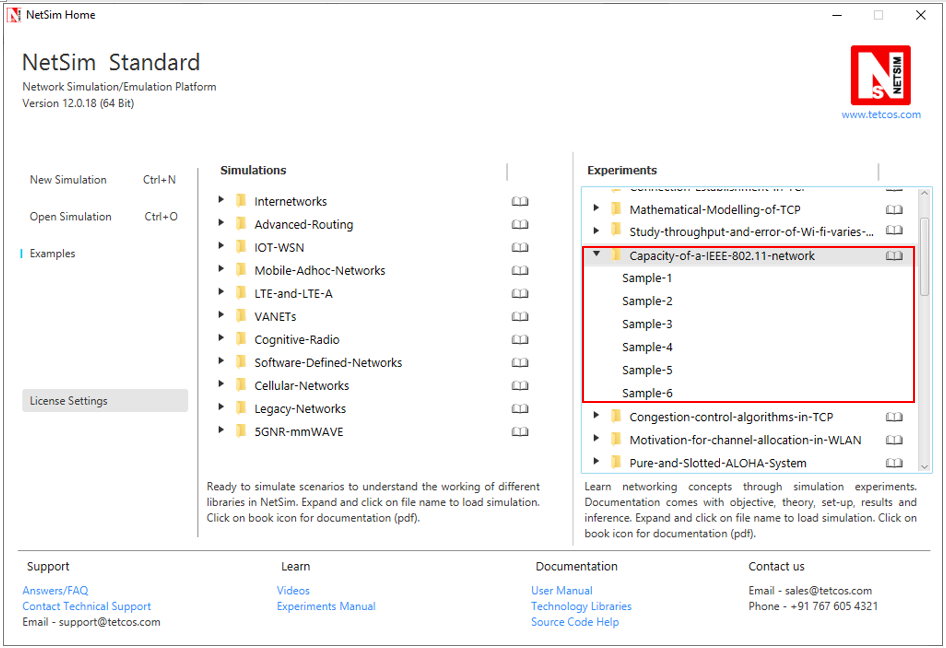
It follows that, in steady state, the system will look like two contending WiFi nodes, one with TCP data packets and the other with TCP ACK packets. This will be the case no matter how many downloading mobile devices there are. The total throughput can be obtained by setting up the model of two saturated nodes, one with TCP data packets, and the other with TCP ACK packets. The data packets of all the TCP connections will be randomly ordered in the AP buffer, so that the head-of-the-line packet will belong to any particular mobile device with probability This throughput is shared equally between the mobile devices.

Now suppose that the TCP data packet throughput with the two-node model is . Then

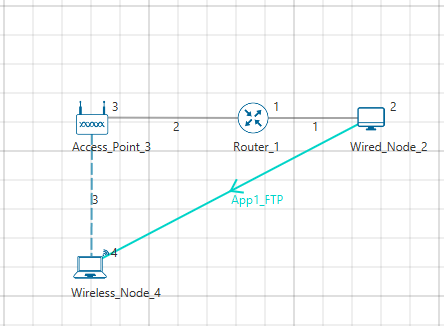
where the denotes the largest integer less than or equal to Use NetSim to verify that for an 11Mbps Wi-Fi speed, with RTS/CTS enabled the total TCP throughput is 3.4 Mbps. If , then In this example, if the download throughput obtained by each of them will be but if one more downloading device is added then each will get a throughput less than . We say that the capacity of this network for a target throughput of is 5.

## Procedure

Open NetSim and click **Examples > Experiments > Capacity-of-a-IEEE-802.11** **> Sample-1** as shown below:



NetSim UI displays the configuration file corresponding to this experiment.

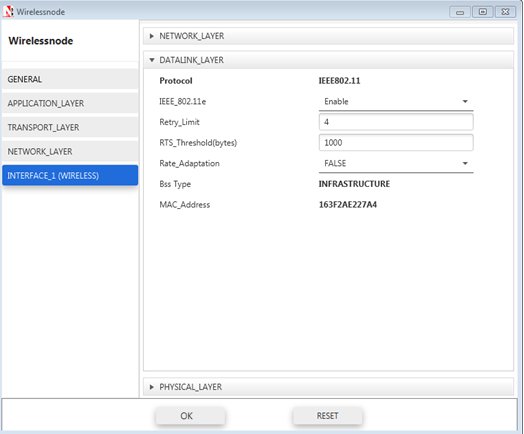


## Procedure:

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

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

**Step 2:** In the Interface (WIRELESS) > Data Link Layer Properties of Wireless Node 4, Retry Limit is set to 4 and RTS Threshold is set to 1000.

****

**Step 3:** The Link Properties is set as per the table given below:

|  |  |
| --- | --- |
| Wired Link | |
| Uplink BER rate | 0 |
| Downlink BER rate | 0 |
| Uplink Delay | 0µs |
| Downlink Delay | 0µs |

|  |  |
| --- | --- |
| Wireless Link | |
| Channel Characteristics | No path loss |

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

An FTP Application is generated from Wired Node 2 i.e. Source to Wireless Node 4 i.e. Destination with File Size set to 100000000 Bytes and Inter Arrival Time set to 15µs.

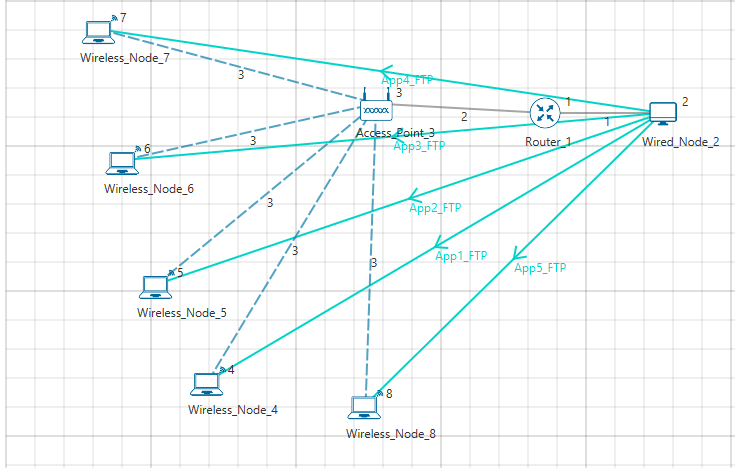
**Step 4:** Run the Simulation for 15 Seconds and note down the throughput.

**Sample 2:**

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

**Step 1:** The number of Wireless Nodes is increased to 5 and FTP applications are generated from Wired Node 2 to each of the Wireless Nodes as shown below:

**No. of wireless nodes = 5**



**Application Properties**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Properties | App1 | App2 | App3 | App4 | App5 |
| Application Type | FTP | FTP | FTP | FTP | FTP |
| Source Id | 2 | 2 | 2 | 2 | 2 |
| Destination Id | 4 | 5 | 6 | 7 | 8 |
| File size (Bytes) | 100000000 | 100000000 | 100000000 | 100000000 | 100000000 |
| File Inter arrival time | 15s | 15s | 15s | 15s | 15s |

**Step 4:** Run the Simulation for 15 Seconds and note down the throughput.

***NOTE: Follow the same procedure for next samples with wireless nodes 10, 15, 20, 25 and note down the sum of throughputs for all applications.***

## Measurements and Output:

Aggregated download throughput with different values of N (wireless nodes) is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Number | Number of Devices | Sum of throughputs (Mbps) | Throughput Per Device (Mbps) |
| 1 | 1 | 3.38 | 3.38 |
| 2 | 5 | 3.40 | 0.68 |
| 3 | 10 | 3.19 | 0.318 |
| 4 | 15 | 3.25 | 0.216 |
| 5 | 20 | 3.22 | 0.161 |
| 6 | 25 | 3.25 | 0.129 |

***NOTE: In the referred paper we see that, the throughput value for 11 Mbps WLAN is 3.8 Mbps. Please note that this is the aggregate PHY throughput of the AP. However, in NetSim, we are calculating the total Application throughput.***

To derive the PHY layer throughput from the APP layer throughput, we need to add overheads of all layers

|  |  |
| --- | --- |
| Layer | Overhead (Bytes) |
| Transport Layer | 20 |
| Network Layer | 20 |
| MAC Layer | 40 |
| PHY layer | 48µs = (11\*48)/8 = 66 |
| Total Overhead | 146 |

*PHY\_Throughput = APP\_Throughput \* 1606/1460 = 3.41\*1606/1460 = 3.79 Mbps)*

## Inference:

We see that as the number of devices increase the aggregate (combined) throughput remains constant whereas the throughput per user decreases.

As discussed earlier, our goal was to identify that if during the download, a user expects to get a throughput of at least bytes per second, what is the maximum number of users (say, ?

If we set to be 650 Kbps, then we see that from the output table that the maximum number of users who can simultaneously download files is 5

## Reference Documents:

1. *Analytical models for capacity estimation of IEEE 802.11 WLANs using DCF for internet applications. George Kuriakose, Sri Harsha, Anurag Kumar, Vinod Sharma*

# Multi-AP Wi-Fi Networks: Channel Allocation

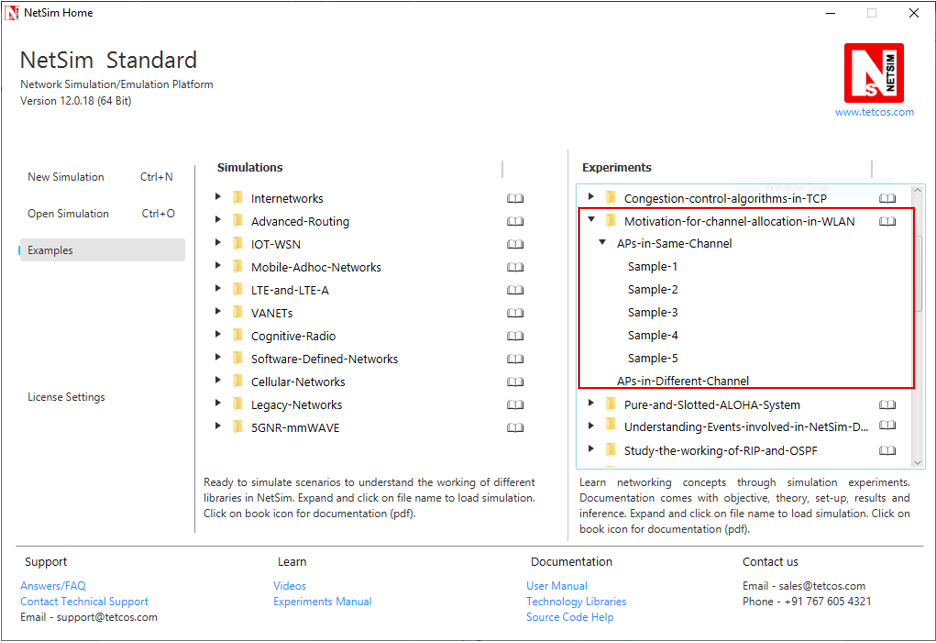
## Introduction

A single Wi-Fi Access Point (AP) can connect laptops and other devices that are a few 10s of meters distance from the AP, the actual coverage depending on the propagation characteristics of the building in which the Wi-Fi network is deployed. Thus, for large office buildings, apartment complexes, etc., a single AP does not suffice, and multiple APs need to be installed, each covering a part of the building. We will focus on 2.4GHz and 5GHz systems. In each of these systems the available bandwidth is organized into channels, with each AP being assigned to one of the channels. For example, 2.4GHz Wi-Fi systems operate in the band 2401MHz to 2495MHz, which has 14 overlapping channels each of 22MHz. There are 3 nonoverlapping channels, namely, Channels 1, 6, and 11, which are centered at 2412MHz, 2437MHz, and 2462MHz. Evidently, if neighboring APs are assigned to the same channel or overlapping channels they will interfere, thereby leading to poor performance. On the other hand, since there are only three nonoverlapping channels, some care must be taken in assigning channels to APs so that nearby APs have nonoverlapping channels, whereas APs that are far apart can use the same or overlapping channels.

In this experiment we will understand some basic issues that arise in multi-AP networks, particularly with attention to channel allocation to the APs.

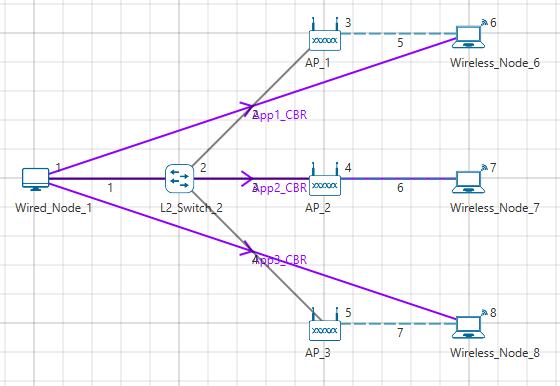
## Network Setup:

Open NetSim and click **Examples > Experiments > Motivation-for-channel-allocation-in-WLAN** as shown below:



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

**APs on the same channel:**



**Sample1:**

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

**Step 1:** A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 1 L2 Switch, 3 Wireless Nodes and 3 Access Points in the **“Internetworks”** Network Library.

**Step 2:** The device positions are set as per the table given below:

|  |  |  |
| --- | --- | --- |
| General Properties | | |
| Device Name | **X** | **Y** |
| AP\_1 | 15 | 5 |
| AP\_2 | 15 | 10 |
| AP\_3 | 15 | 15 |
| Wireless\_Node\_6 | 20 | 5 |
| Wireless\_Node\_7 | 20 | 10 |
| Wireless\_Node\_8 | 20 | 20 |

**Step 3:** TCP is disabled in Wired Node 1.

**Step 4:** In the INTERFACE (WIRELESS) > PHYSICAL LAYER Properties of all the Wireless Nodes and Access Points, the Protocol Standard is set to IEEE 802.11 b.

**Step 5:** In all the Wired Link Properties, Bit Error Rate and Propagation Delay is set to 0.

**Step 6:** The Wireless Link Properties are set as follows:

|  |  |
| --- | --- |
| Channel Characteristics | PATH LOSS ONLY |
| Path Loss Model | LOG DISTANCE |
| Path Loss Exponent | 3.5 |

**Step 7:** Right click on the Application Flow App1 CBR and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 1 i.e. Source to Wireless Node 6 i.e. Destination with Packet Size set to 1460 Bytes and Inter Arrival Time set to 1168µs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 10 Mbps. Generation Rate can be calculated using the formula:

Similarly, two more CBR applications are generated.

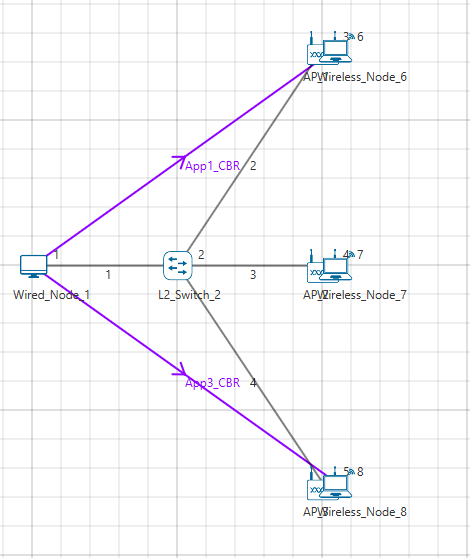
**Step 8:** Run the Simulation for 10 Seconds and note down the throughput.

**Sample2:**

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

**Step 1:** Before we start designing the network scenario, the Grid Length is set to 1000 meters. This can be set by choosing the Menu Option Settings > Environment Settings > Grid from the GUI.

**Step 2:** From the previous sample, we have removed App2 CBR (i.e. from Wired Node1 to Wireless Node7), set distance between the other 2 Access Points (AP 1 and AP 3) as 300m and distance between APs and Wireless nodes as 10m as shown below:



**Step 3:** The device positions are set according to the table given below:

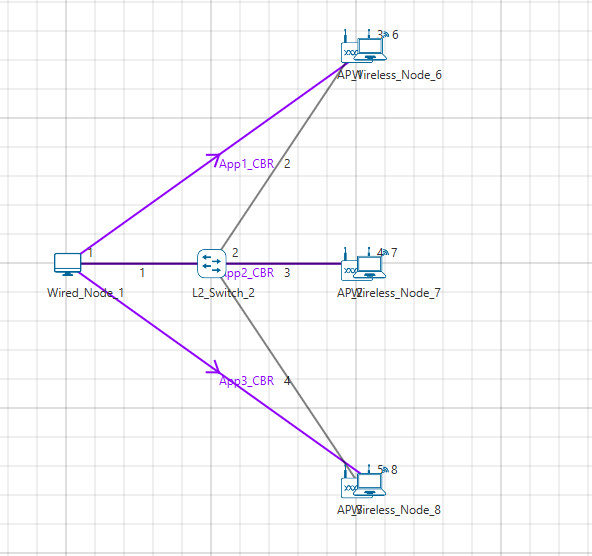
|  |  |  |
| --- | --- | --- |
| General Properties | | |
| Device Name | **X** | **Y** |
| AP\_1 | 400 | 0 |
| AP\_2 | 400 | 200 |
| AP\_3 | 400 | 400 |
| Wireless\_Node\_6 | 410 | 0 |
| Wireless\_Node\_7 | 410 | 200 |
| Wireless\_Node\_8 | 410 | 400 |

**Step 4:** Run the Simulation for 10 Seconds and note down the throughput.

**Sample3:**

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

**Step 1:** Thedistance between the Access Points (AP 1 and AP 3) is set to 150m and distance between APs and Wireless nodes as 10m as shown below:



**Step 2:** The device positions are set according to the table given below:

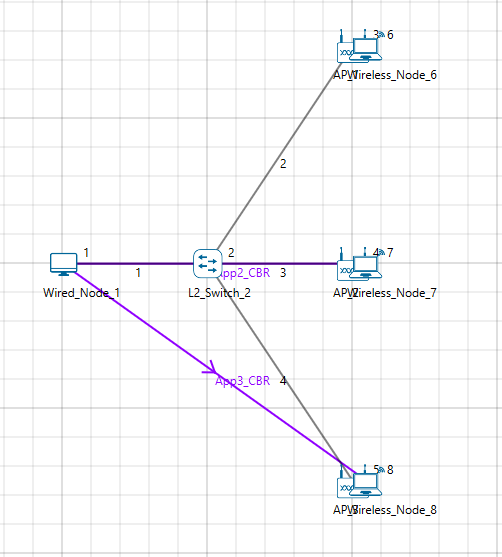
|  |  |  |
| --- | --- | --- |
| General Properties | | |
| Device Name | **X** | **Y** |
| AP\_1 | 400 | 0 |
| AP\_2 | 400 | 200 |
| AP\_3 | 400 | 400 |
| Wireless\_Node\_6 | 410 | 0 |
| Wireless\_Node\_7 | 410 | 200 |
| Wireless\_Node\_8 | 410 | 400 |

**Step 3:** Run the Simulation for 10 Seconds and note down the throughput.

**Sample4:**

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

**Step 1:** From the previous sample, we have removed App1 CBR (i.e. from Wired Node 1 to Wireless Node 6), set distance between the other 2 Access Points (AP 2 and AP 3) as 1500m and distance between APs and Wireless nodes as 10m as shown below:



**Step 2:** The device positions are set according to the table given below:

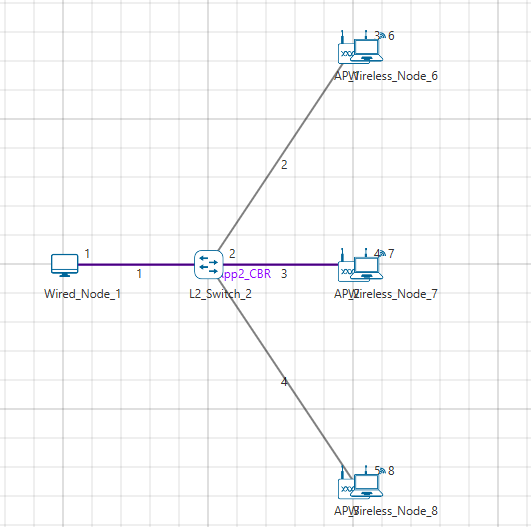
|  |  |  |
| --- | --- | --- |
| General Properties | | |
| Device Name | **X** | **Y** |
| AP\_1 | 400 | 0 |
| AP\_2 | 400 | 200 |
| AP\_3 | 400 | 400 |
| Wireless\_Node\_6 | 410 | 0 |
| Wireless\_Node\_7 | 410 | 200 |
| Wireless\_Node\_8 | 410 | 400 |

**Step 3:** Run the Simulation for 10 Seconds and note down the throughput.

**Sample5:**

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

**Step 1:** From Sample 3,we have removed first and third applications as shown below:



**Step 2:** The device positions are set according to the table given below:

|  |  |  |
| --- | --- | --- |
| General Properties | | |
| Device Name | **X** | **Y** |
| AP\_1 | 400 | 0 |
| AP\_2 | 400 | 200 |
| AP\_3 | 400 | 400 |
| Wireless\_Node\_6 | 410 | 0 |
| Wireless\_Node\_7 | 410 | 200 |
| Wireless\_Node\_8 | 410 | 400 |

**Step 3:** Run the Simulation for 10 Seconds and note down the throughput.

**APs in different channel:**

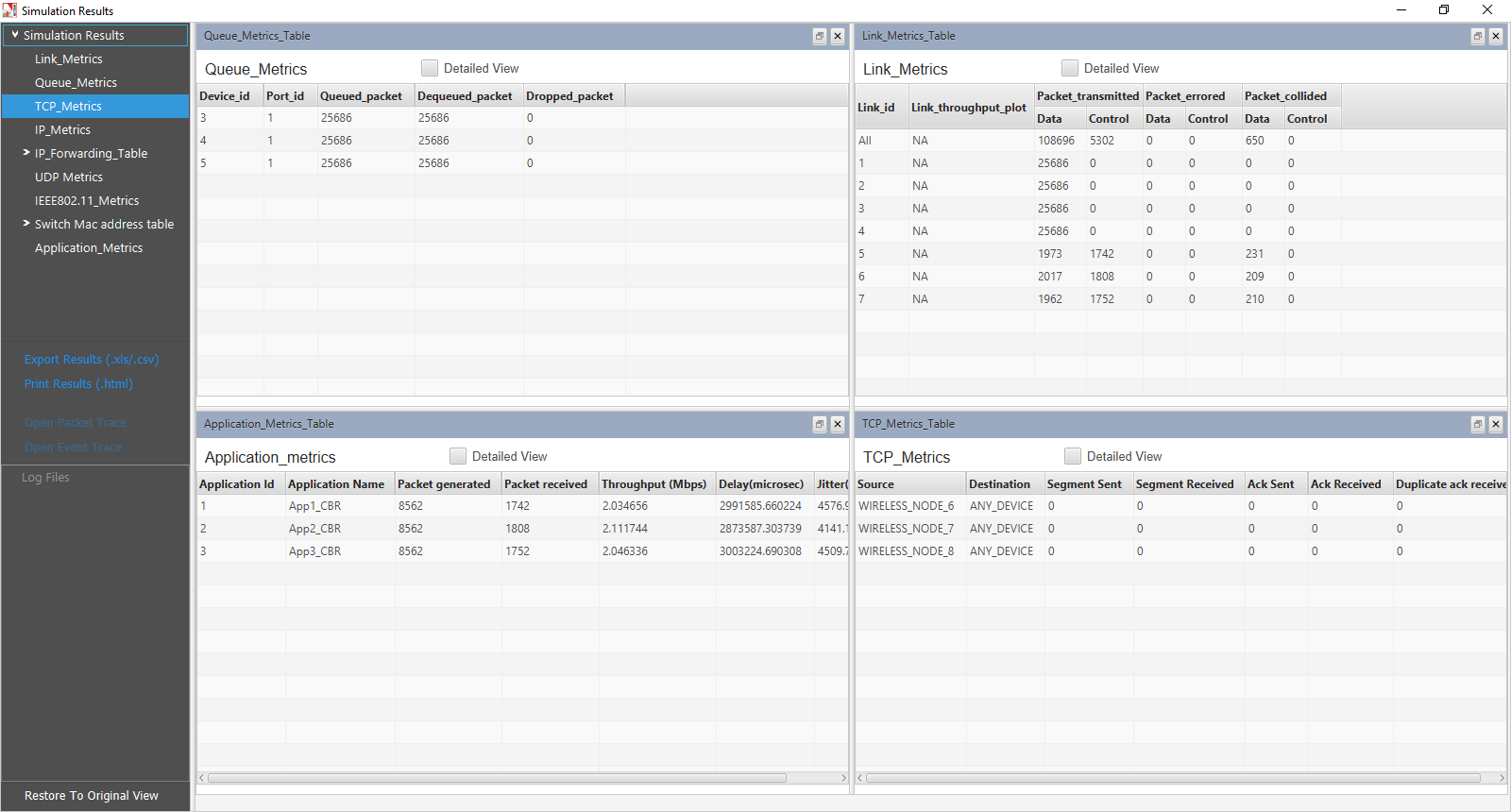
The following changes in settings are done from the previous sample:

**Step 1:** In Sample 3, we have changed standard channel to 11\_2462 under INTERFACE (WIRELESS) > DATALINK LAYER Properties of AP 2.

**Step 2:** Run the Simulation for 10 Seconds and note down the throughput.

## Output:

After running simulation, check throughput in Application metrics as shown in the below screenshot:



|  |  |  |  |
| --- | --- | --- | --- |
| Sample | Throughput (Mbps) | | |
| **AP\_1** | **AP\_2** | **AP\_3** |
| All APs on the same channel | | | |
| 1 | 2.03 | 2.11 | 2.04 |
| 2 | 5.94 | N/A | 5.92 |
| 3 | 5.42 | 0.63 | 5.41 |
| 4 | N/A | 3.29 | 3.26 |
| 5 | N/A | 5.92 | N/A |
| Each AP on a different nonoverlapping channel | | | |
| 1 | 5.94 | 5.92 | 5.92 |

***NOTE****:* ***Please refer “Wi-Fi UDP Download Throughput” experiment for theoretical WLAN throughput calculations in NetSim Experiment Manual.***

## Discussion

We recall that each AP is associated with one station (STA; e.g., a laptop). All the APs are connected to the same server which is sending separate UDP packet streams to each of the STAs via the corresponding AP. The packet transmission rate from the server is large enough so that the AP queue in permanently backlogged, i.e., the rate at which the server transmits packets is larger than the rate at which the AP can empty the packet queue.

### All APs on the same channel

* **Case 1:** All the APs and their associated STAs are close together, so that all devices (APs and STAs) can sense every other device.
  + The table shows that all the AP-STA links achieve the same UDP throughput. This is because all the AP-STA links are equivalent (since all interfere with each other), and only one can be active at one time. The throughput for this scenario can be predicted from the analysis in Section 7.4 of the book *Wireless Networking* by *Anurag Kumar, D. Manjunath and Joy Kuri*
* **Case 2:** AP1 and AP3 are close to their associated STAs but are 400m apart. The link from AP2 to its STA is half-way between the other two APs, and is notcarrying any traffic.
  + The table shows that both the links from AP1 and AP3 to their respective STAs carry the same throughput, of 5.94Mbps and 5.92Mbps. These are also the throughputs that each link would have if the other was not present, indicating that the two links are far enough apart that they do not interfere.
* **Case 3:** This is the same scenario as Case 2, but the AP2-STA link is now carrying traffic
  + We find that, in comparison with Case2, the AP1-STA and AP3-STA carry slightly lower throughputs of about 5.4Mbps, whereas the AP2-STA link carries a small throughput of 0.63Mbps. Comparing Cases 1 and 3 we conclude that in these networks there can be severe unfairness depending on the relative placement of the AP-STA links. In Case 1, all the links could sense each other, and each got a fair chance. In Case 3, we have what is called the “link-in-the-middle problem.” The AP2-STA link is close enough to interfere with the AP1-STA link and the AP3-STA link, whereas the AP1-STA link and the AP3-STA link do not “see” each other. The AP2-STA link competes with the links on either side, whereas the other links compete only with the link in the centre, which thereby gets suppressed in favour of the outer links.
* **Case 4:** Here we stop the traffic to AP1 but send the traffic to the AP2-STA link and the AP3-STA link.
  + The two active links interfere with each other, but the situation is symmetric between them (unlike in Case 3), and they obtain equal throughput. Again, the throughput obtained by these two links can be predicted by the analysis mentioned earlier in this section.
* **Case 5:** Now we send traffic only to AP2
  + The throughput is now 5.92Mbps, since the AP2-STA link can transmit without interference; there are no collisions. The reason that this throughput is less than the sum of the two throughputs in Case 4 is that the single link acting by itself, with all the attendant overheads, is unable to occupy the channel fully.

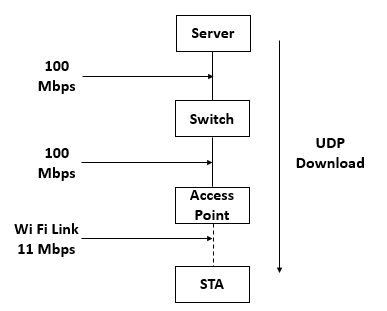
### Each AP on a different nonoverlapping channel

There is only one case here. Having observed the various situations that arose in the previous subsection when all the APs are on the same channel, now we consider the case where all the AP-STA pairs are each on a different nonoverlapping channel. As expected, every AP-STA pair gets the same throughput as when they are alone on the network.

# WiFi: UDP Download Throughput

## The Setup and Motivation

The most basic packet transfer service offered by the Internet is called the “datagram” service, in which a series of packets are transmitted to a receiver without any packet loss recovery, flow control, or congestion control. The Internet’s UDP protocol implements the datagram service. In this experiment we will study the performance of UDP transfers from a server on a wireline local area network to WiFi Stations (STA), via WiFi Access Points (AP). The schematic of the network that we will be simulating in NetSim is shown in the figure below:



The server, which contains the data that needs to be transferred to the STAs (say, a laptops), is connected by a 100 Mbps switched Ethernet link to an Ethernet switch, which is, in turn, connected to the WiFi APs. Each AP is associated (i.e., connected) at 11 Mbps to a single STA. The objective is to transfer a large number of packets (say, constituting a video) from the server to each of the STAs, the packet stream to each of the STAs being different (e.g., each STA is receiving a different video from the server). In this experiment, we are interested in studying the limitation that the WiFi link places on the data transfers. We assume that the server transmits the packets at a high enough rate so that the queues at the APs fill up, and the rate of the UDP transfers is, therefore, governed by the WiFi link. It may be noted that, in practice, there will be a flow control mechanism between each STA and the server, that will control the rate at which the server releases packets, in order to prevent buffer overflow at the APs.

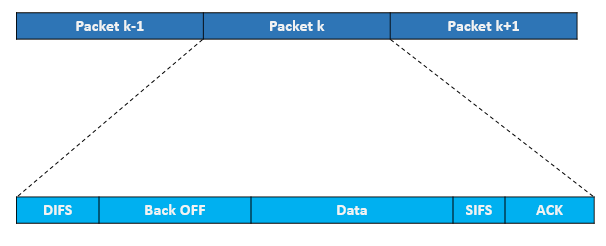
In this setting, this experiment will ask one precise question. With the buffers at the AP full, at what rate will the WiFi protocol transfer the packets from the APs to the STAs over the wireless link. We will study two cases:

1. A single AP and a single STA: Since there is only one transmitter in this wireless network (namely, the AP), there is no contention, and the rate of packet transfer over the link will be governed by the basic overheads in the protocol, such as the interframe spacings, packet header overheads, transmit-receive turn-around times, and acknowledgement times. We will begin by a simple calculation (essentially timing book-keeping) that will predict the UDP throughput, and then we will verify our calculation using the NetSim simulator.
2. Multiple APs and one STA for each AP: This is the more common situation (for example neighboring apartments in a building, each with one AP and one laptop, all drawing data from the Internet service provider). The performance of such a system depends on the wireless propagation path-loss between the various APs. A predictive analysis is difficult in the general case. For deriving some insight, we will study the case where all the APs are close to each other, and thus exactly one transmission from AP to an STA can be successful at any time. If two or more APs transmit together, then all the transmissions are not successful. Even in this case, the analysis mathematically complex and is available in, Anurag Kumar, D. Manjunath and Joy Kuri. 2008: Wireless Networking. Sec 7.4

## Predicting the UDP Throughput

### One AP and one STA

As stated above, in the setup described, the AP queue is full. Thus, after a packet is completely transmitted over the wireless link, immediately the process for transmitting the next packet starts. This is illustrated by the upper part of the figure below, where the successive packets from the AP are shown as being sent back-to-back. The time taken to send a packet is, however, not just the time to clock out the physical bits corresponding to the packet over the Wi-Fi medium. After the completion of a packet transfer, the AP’s Wi-Fi transmitter waits for a Distributed Coordination Function Inter-Frame Space (DIFS), followed by a backoff that is chosen randomly between 1 and 32 slots. Upon the completion of the backoff, the packet transmission starts. Each packet carries physical layer overheads, MAC layer overheads, and IP overheads. After the transmission of the packet, there is a Short Inter-Frame Space (SIFS), which gives time to the receiver (namely, the STA) to transition from the listening mode to the transmit mode. After the SIFS, the STA sends back a MAC acknowledgement (ACK). This completes the transmission of one UDP packet from the AP to the STA. Immediately, the process for sending the next packet can start. The details of the various timings involved in sending a single UDP packet are shown in the lower part of the figure below.



In this experiment, the payload in each packet is the same (1450 Bytes). Since the packets are sent back to back, and the state of the system is identical at the beginning of each packet transmission, the throughput (in Mbps) is computed by the following simple (and intuitive) relation.

#### Without RTS / CTS

where

Application payload = 1450 Bytes

Average time per packet = 50 + 310 + 1296 + 10 + 304 = 1970 µs

SIFS = 10 µs

Slot time = 20 µs

CWmin = 31 slots for 802.11b

DIFS = SIFS + 2 \* Slot Time = 10 µs + 2 \* 20 µs = 50 µs

Average Backoff time = 310 µs

Packet Transmission Time = 192 µs + (1518 \* 8/11 Mbps) = 1296 µs

Preamble time = 192 µs for 802.11b standard

MPDU Size = 1450 + 8 + 20 + 40 = 1518 Bytes

Ack Transmission Time = 192 µs + (14 Bytes \* 8 / 1Mbps) = 304 µs

UDP throughput = 1450\*8/ (1970) = 5.92 Mbps

#### With RTS/CTS

RTS packet transmission time = Preamble time + (RTS Packet payload/Data rate) = 192 + 20 \* 8 / 1 = 352 µs

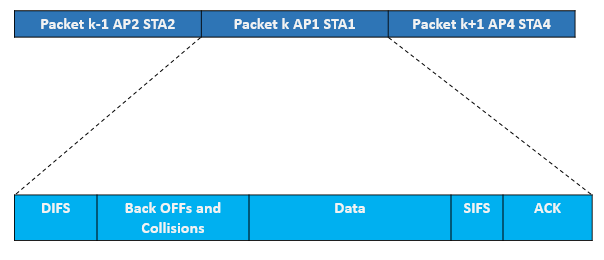
CTS packet transmission time = Preamble time + (CTS Packet payload/Data rate) = 192 + 14 \* 8 / 1 = 304 µs

Average time per packet = 50 + 352 + 304 + 310 + 1296 + 10 + 304 = 2626 µs

UDP throughput = 1450\*8/ (2626) = 4.44 Mbps

### Multiple APs (near each other) and one STA per AP

Since the AP queues are full, on the WiFi medium the packet transmission can still be viewed as being back-to-back as shown in the upper part of the figure below. However, since there are multiple contending AP-STA links, there are two differences between this figure and the one shown above (for the single AP and single STA case).



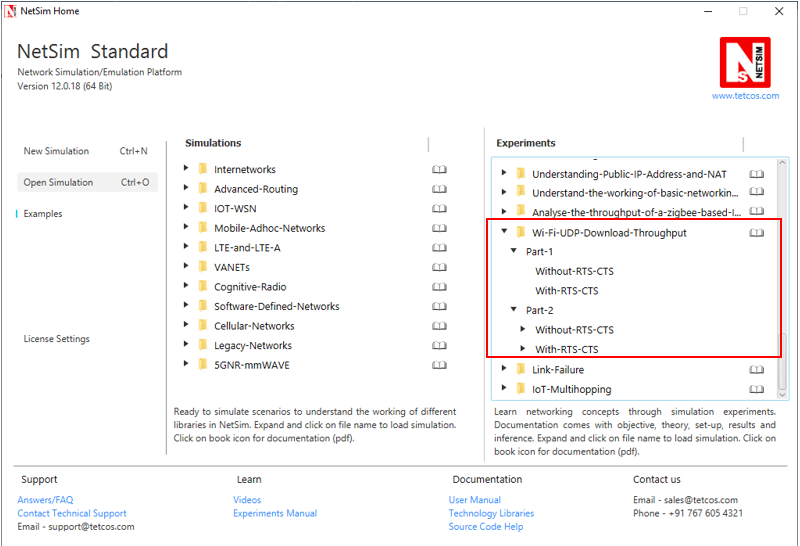
1. Within each transmission period, there is now a “backoffs and collisions” period, where in the figure above we only showed a “backoff” period. Access to the channel is be contention, collision, and backoff, and this “backoffs and collisions” duration is the time taken to select one transmitting AP.
2. The other difference is that, after each “backoffs and collisions” period, any one AP-STA pair “wins” the contention, and the corresponding AP can then send a packet. It turns out that the contention mechanism is such that each of the AP-STA pairs can succeed with equal probability, independent of the pair that has previously been successful. Thus, if there are, say, 5 AP-STA pairs, then each successful packet transmission will be from any of these pairs with a probability of 0.2.

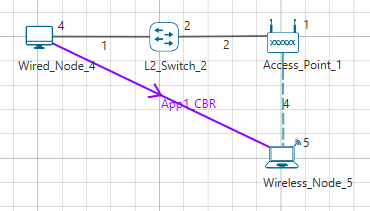
With this discussion, and the upper part of the figure above, it follows that the following expression still holds

Having obtained the total throughput over all the AP-STA pairs in this manner, by the fact that each packet transmission is with equal probability from any of the AP-STA pairs, the UDP throughput for each AP-STA pair (for pairs) is just of the total throughput.

## Network Setup:

Open NetSim and click **Examples > Experiments > Wi-Fi-UDP-Download-Throughput** **> Part-1 > Without-RTS-CTS** as shown below:



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

## Procedure:

### Part-1: Without RTS/CTS

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

**Step 1:** A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 1 Wireless Node, 1 L2 Switch, and 1 Access Point in the **“Internetworks”** Network Library.

**Step 2:** TCP Protocol is disabled in Wired Node 4.

**Step 3:** In the Interface Wireless > Physical Layer Properties of Wireless Node 5, Protocol Standard is set to IEEE 802.11b.

In the Interface Wireless > Data Link Layer Properties of Wireless Node 5, RTS Threshold is set to 3000.

It will automatically set the same in the Access Point, since the above parameters are Global.

**Step 4:** In the Wired Link Properties, Bit Error Rate and Propagation Delay is set to 0 for both the links.

**Step 5:** In the Wireless Link Properties, Channel Characteristics is set to NO PATH LOSS.

**Step 6:** Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 4 i.e. Source to Wireless Node 5 i.e. Destination with Packet Size set to 1450 Bytes and Inter Arrival Time set to 116µs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 100 Mbps. Generation Rate can be calculated using the formula:

**Step 7:** Run the Simulation for 10 Seconds and note down the throughput.

### Part-1: With RTS/CTS

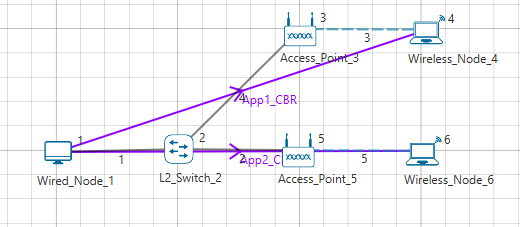
The following changes in settings are done from the previous sample:

**Step 1:** In the Interface Wireless > Data Link Layer Properties of Wireless Node 5, RTS Threshold is set to 1000.

**Step 2:** Run the Simulation for 10 Seconds and note down the throughput.

### Part-2: Without RTS/CTS: 2APs

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



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

**Step 2:** In the Interface Wireless > Data Link Layer Properties of Wireless Node 4, RTS Threshold is set to 3000.

It will automatically be set for Wireless Node 6, since the above parameter is Global.

**Step 3:** Two CBR applications are generated from Wired Node 1 i.e. Source to Wireless Node 4 and Wireless Node 6 i.e. Destination with a Generation Rate of 10 Mbps.

**Step 4:** Run the Simulation for 10 Seconds and note down the throughput.

Similarly, the subsequent samples are carried out with 3, 4, and 5 Access Points and Wireless Nodes.

### Part-2: With RTS/CTS: 2APs

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

**Step 1:** In the Interface Wireless > Data Link Layer Properties of Wireless Node 4, RTS Threshold is set to 1000.

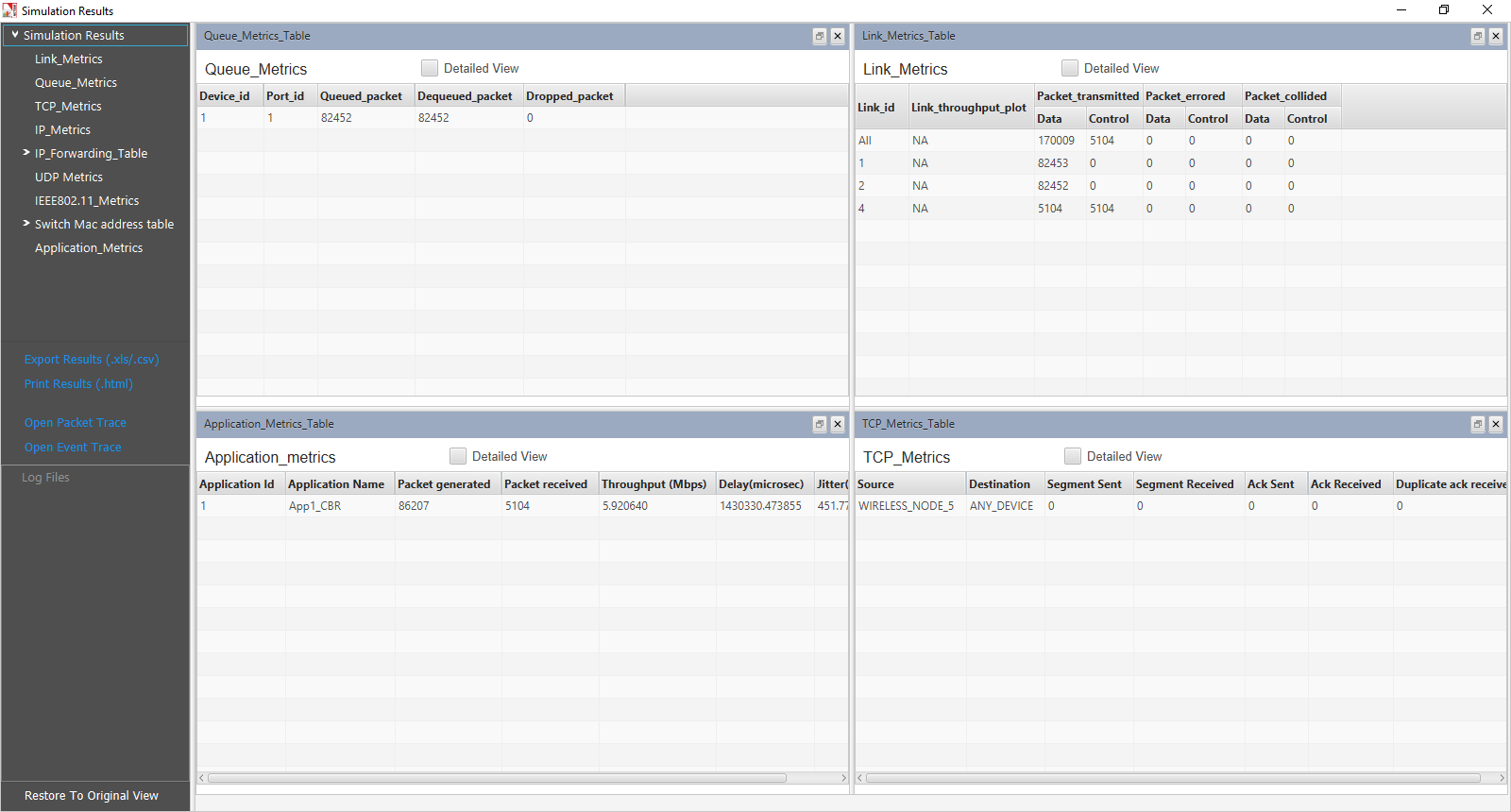
It will automatically be set for Wireless Node 6, since the above parameter is Global.

**Step 2:** Run the Simulation for 10 Seconds and note down the throughput.

Similarly, the subsequent samples are carried out with 3, 4, and 5 Access Points and Wireless Nodes.

## Output I:

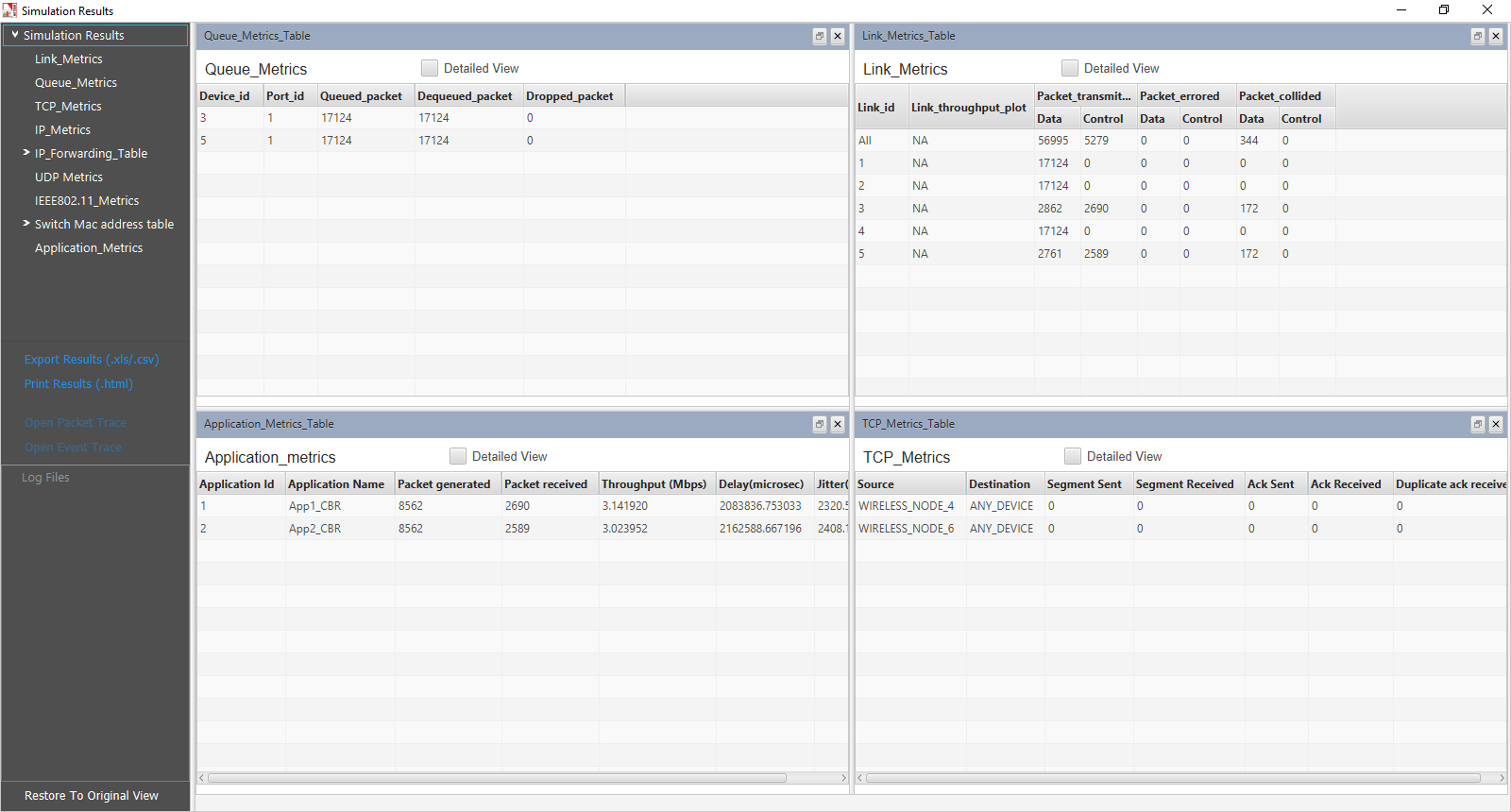
After running simulation, check throughput in Application metrics as shown in the below screenshot:



|  |  |  |
| --- | --- | --- |
| Sample | Predicted throughput (Mbps) | Simulated  Throughput (Mbps) |
| 1 (Without RTS/CTS) | 5.92 | 5.92 |
| 2 (With RTS/CTS) | 4.44 | 4.39 |

## Output II:

After running simulation, check throughput in Application metrics as shown in the below screenshot:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sample | Throughput (Mbps) with 2 APs | Throughput (Mbps) with 3 APs | Throughput (Mbps) with 4 APs | Throughput (Mbps) with 5 APs |
| 1 (Without RTS/CTS) | App 1: 3.14  App 2: 3.02  **Total: 6.16** | App 1: 2.12  App 2: 1.97  App 3: 2.09  **Total: 6.18** | App 1:1.58  App 2:1.57  App 3:1.52  App 4:1.46  **Total: 6.13** | App 1: 1.25  App 2: 1.20  App 3: 1.30  App 4: 1.20  App 5: 1.12  **Total: 6.07** |
| 2 (With RTS/CTS) | App 1: 2.34  App 2: 2.25  **Total: 4.59** | App 1: 1.58  App 2: 1.49  App 3: 1.58  **Total: 4.65** | App 1: 1.21  App 2: 1.18  App 3: 1.15  App 4: 1.12  **Total: 4.66** | App 1: 0.95  App 2: 0.90  App 3: 1.00  App 4: 0.92  App 5: 0.89  **Total: 4.66** |

# Understand the working of basic networking commands (Ping, Route Add/Delete/Print, ACL)

## Theory:

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

**Ping Command**

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

**Route Commands**

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

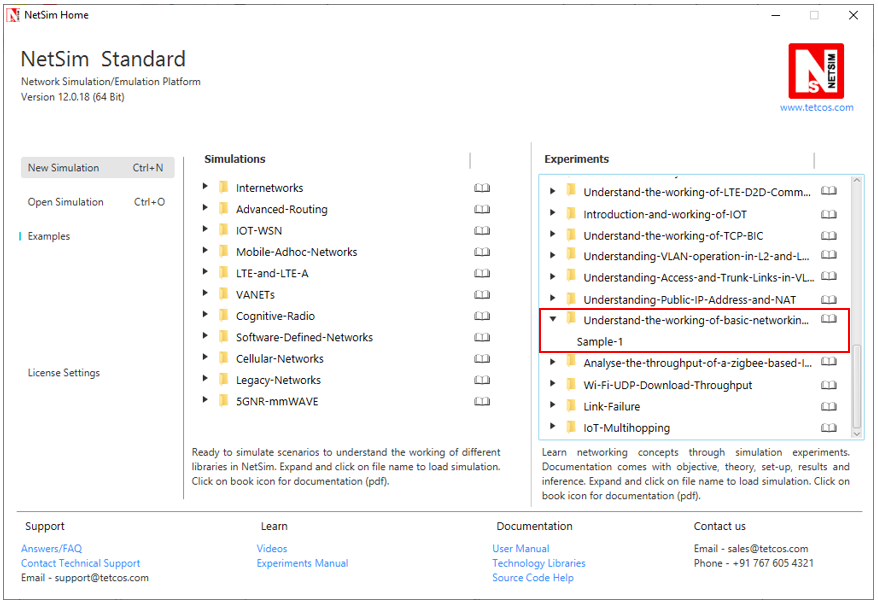
* **route print:** In order to view the entire contents of the IP routing table
* **route delete:** In order to delete all routes in the IP routing table
* **route add:** In order to add a static TCP/IP route to the IP routing table

**ACL Configuration**

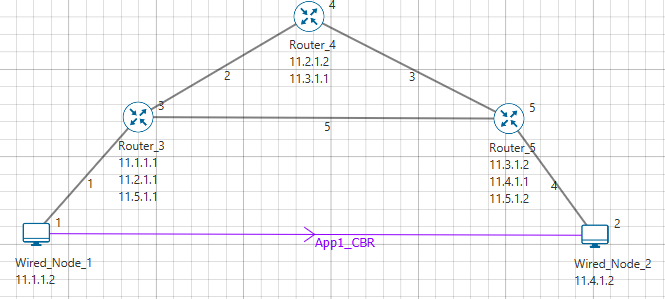
Routers provide basic traffic filtering capabilities, such as blocking the Internet traffic with access control lists (ACLs). An ACL is a sequential list of **Permit** or **Deny** statements that apply to addresses or upper-layer protocols. These lists tell the router what types of packets to: **PERMIT**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

## Network setup:

Open NetSim and click **Examples > Experiments > Understand-the-working-of-basic-networking-commands** **> Sample-1** as shown below:



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



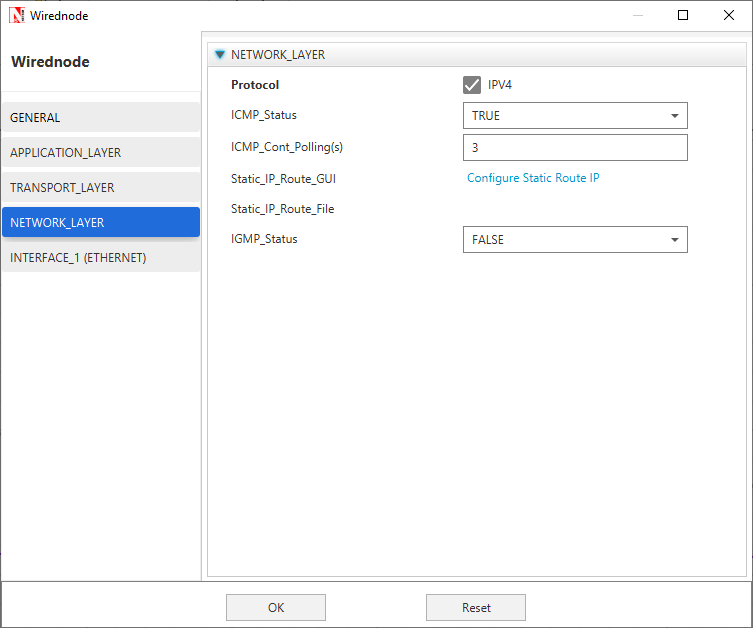
## Procedure:

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

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

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

Similarly, ICMP Status is set as TRUE for all the devices.



**Step 3:** TCP Protocol in Transport Layer is disabled in all the devices.

**Step 4:** In the General properties of Wired Node 1, **Wireshark Capture** is set as Online.

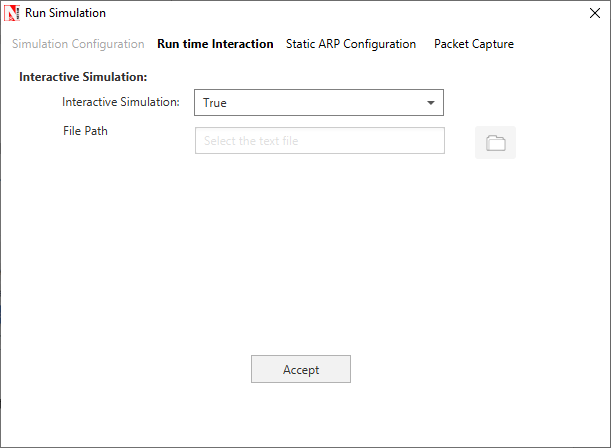
**Step 5:** Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

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

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

**Step 6:** Packet Trace is enabled in NetSim GUI. At the end of the simulation, a very large .csv file is containing all the packet information is available for the users to perform packet level analysis.

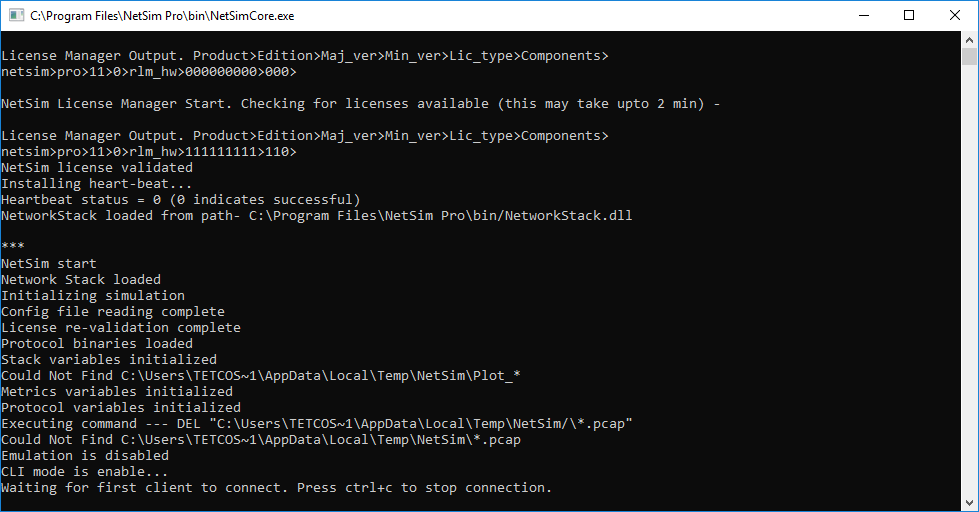
**Step 7:** Click on Run Simulation. Simulation Time is set to 300 Seconds and in the **Runtime Interaction** tab, Interactive Simulation is set to True.



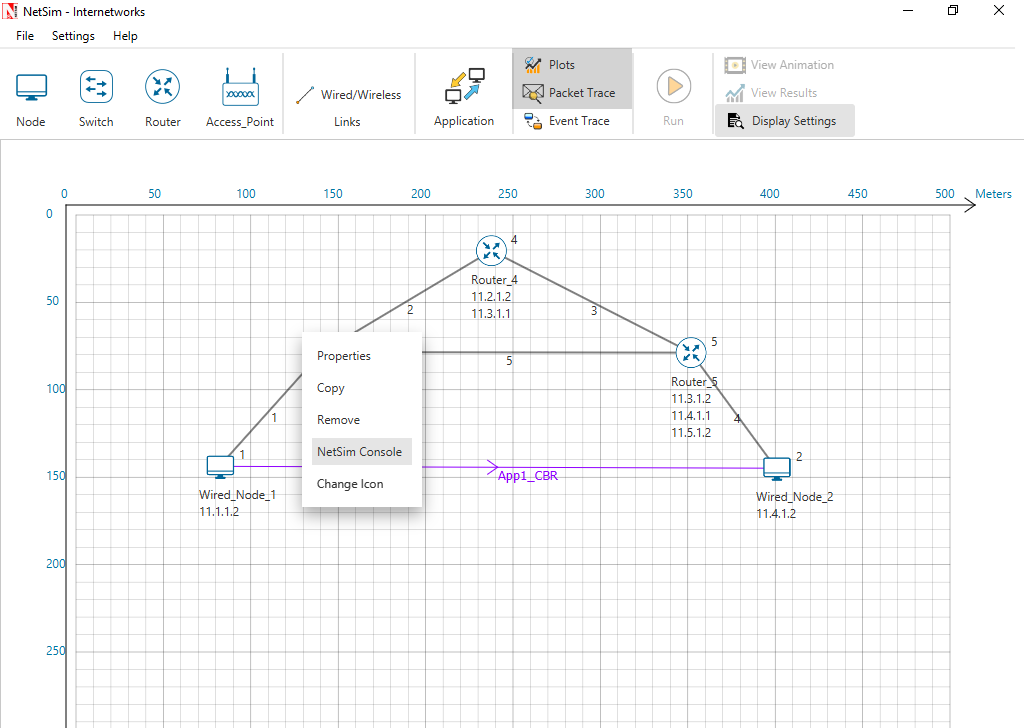
***NOTE:******It is recommended to specify a longer simulation time to ensure that there is sufficient time for the user to execute the various commands and see the effect of that before the Simulation ends.***

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

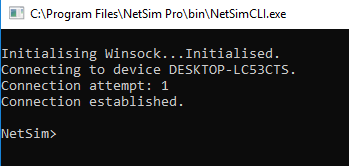
* Simulation (NetSimCore.exe) will start running and will display a message **“waiting for first client to connect”** as shown below:



* Go back to the network scenario. Click on **“Display Settings”** in the top ribbon/toolbar and select the **“Device IP”** checkbox inorder to display the IP address of all the devices. Now, Right click on Router 3 or any other Router and select **“NetSim Console”** option.



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



* After this the command line interface can be used to execute all the supported commands.

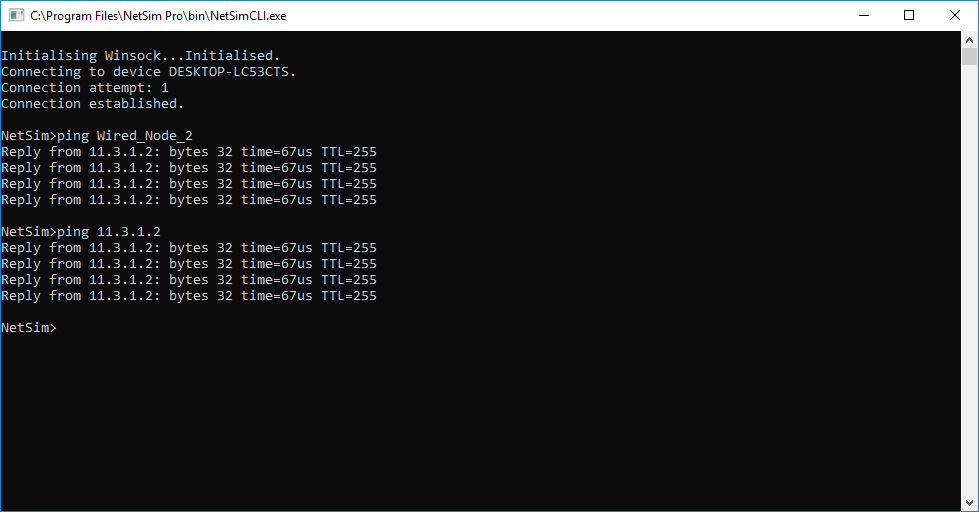
## Network Commands

**Ping Command:**

* You can use the ***ping*** command with an IP address or Device name
* ICMP\_Status should be set as True in all nodes for ping to work

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

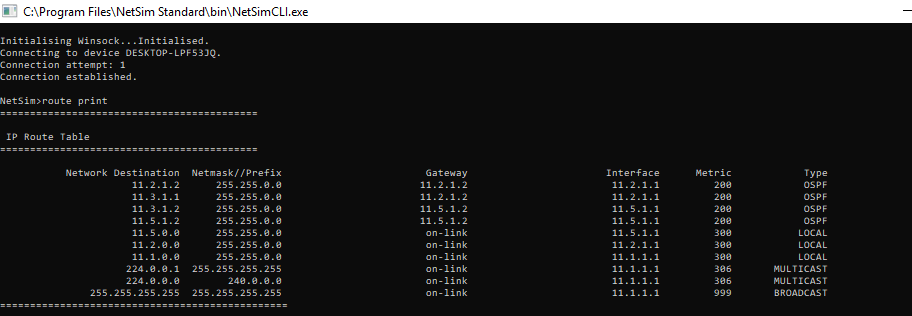
**Ping <Node Name> e.g. ping Wired\_Node\_2**



**Route Commands:**

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

**route print**



* You’ll see the routing table entries with network destinations and the gateways to which packets are forwarded, when they are headed to that destination. Unless you’ve already added static routes to the table, everything you see here is dynamically generated
* In order to delete a route in the IP routing table you’ll type a command using the following syntax

**route delete *destination\_network***

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

**route delete 11.5.1.2**

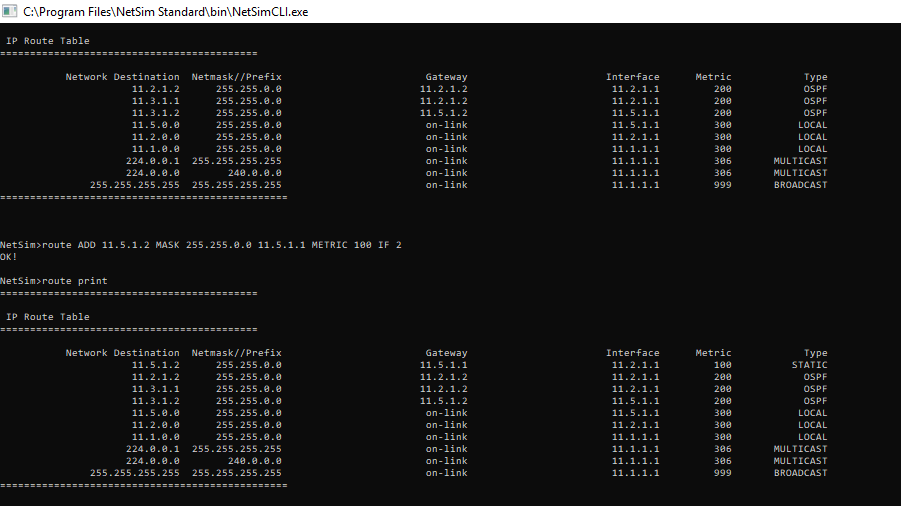
* To check whether route has been deleted or not check again using **route print** command
* To add a static route to the table, you’ll type a command using the following syntax

**route ADD destination\_network MASK subnet\_mask  gateway\_ip metric\_cost interface**

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

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

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



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

**ACL Configuration:**

**Commands to configure ACL:**

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

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

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

**Step to Configure ACL:**

* To create a new rule in the ACL use command as shown below to block UDP packet in Interface 2 and Interface 3 of Router 3.
* TCP Protocol in Transport Layer is disabled in all the devices.
* Use the command as follows:

*NetSim>****acl enable***

*ACL is enable*

*NetSim>****aclconfig***

*ROUTER\_3/ACLCONFIG>****acl print***

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

*ROUTER\_3/ACLCONFIG>****DENY BOTH UDP ANY ANY 0 0 2***

*OK!*

*ROUTER\_3/ACLCONFIG>****DENY BOTH UDP ANY ANY 0 0 3***

*OK!*

*ROUTER\_3/ACLCONFIG>****print***

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

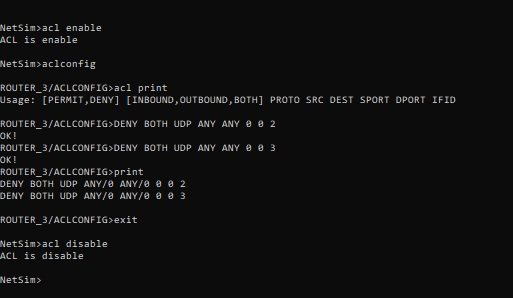
*DENY BOTH UDP ANY/0 ANY/0 0 0 3*

*ROUTER\_3/ACLCONFIG>****exit***

*NetSim>****acl disable***

*ACL is disable*

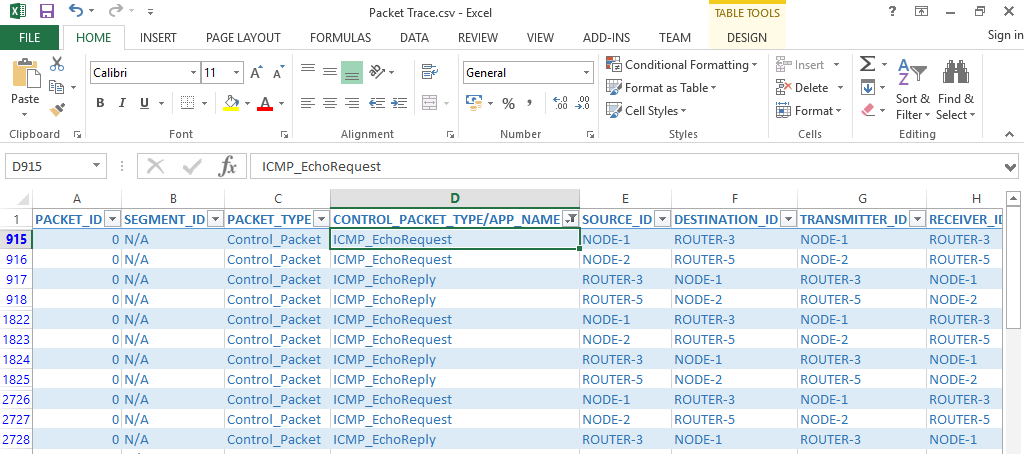
*NetSim>*



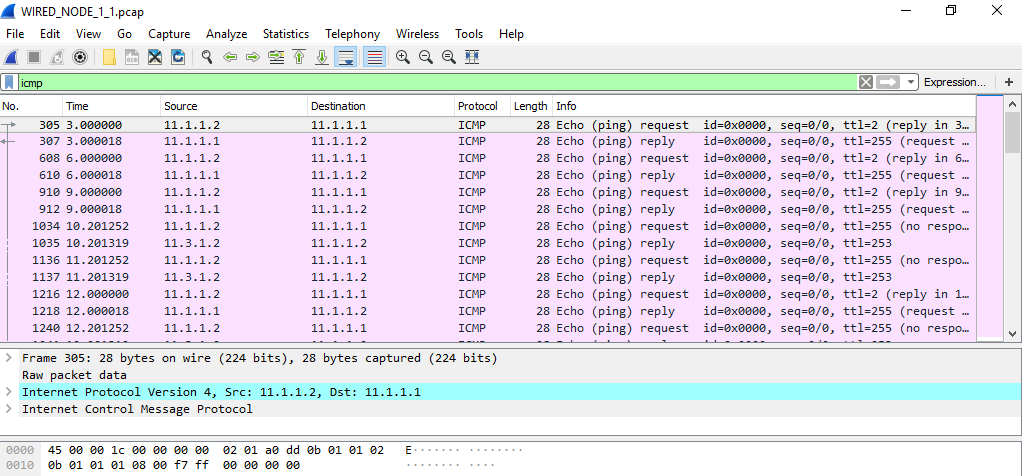
**Ping Command Results:**

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

Filter Control Packet Type/App Name to **ICMP EchoRequest** and **ICMP EchoReply**.

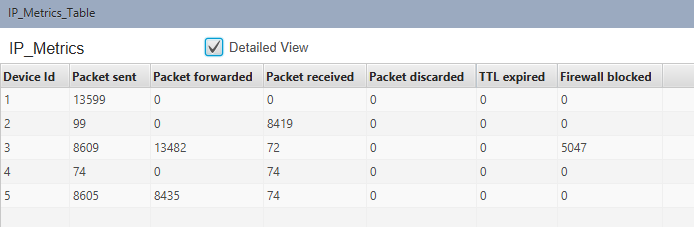


In Wireshark, apply filter as ICMP. we can see the ping request and reply packets in Wireshark.



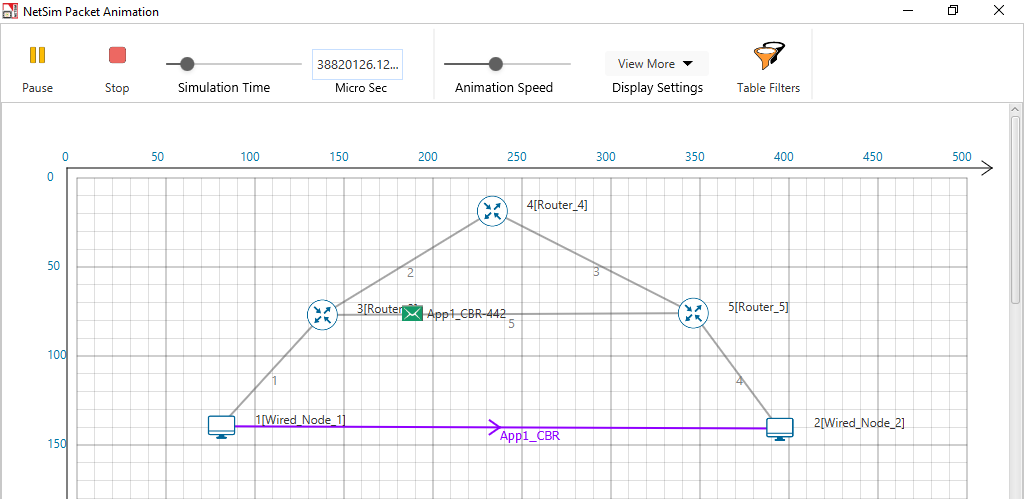
**ACL Results:**

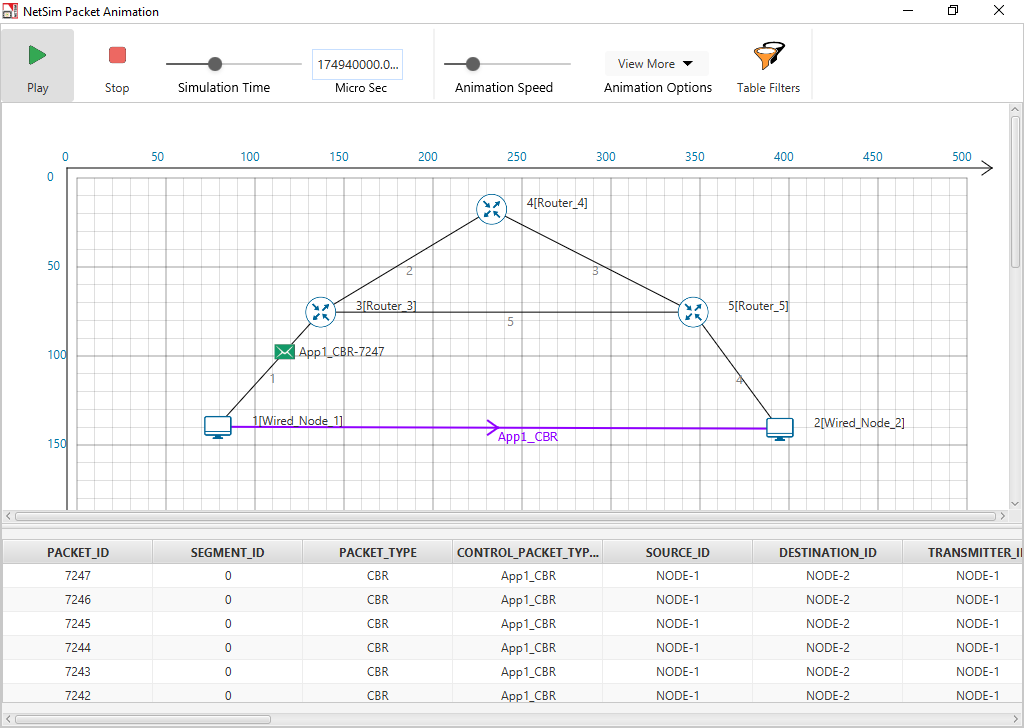
The impact of ACL rule applied over the simulation traffic can be observed in the IP Metrics Table in the simulation results window. In Router 3, the number of packets blocked by firewall has been shown below:



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

Users can also observe this in Packet Animation before and after the Packets are blocked as shown below:





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

# Working of IGMP

This example explains how IGMP works to multicast data in interconnected networks.

The network modelled consists of:

* A subnet with 4 wired nodes, a multicast router, and a multicast application running on one of the wired nodes.
* IGMP is running on all the wired nodes.
* IGMP is running on the multicast router.
* Only a few nodes receive multicast traffic.

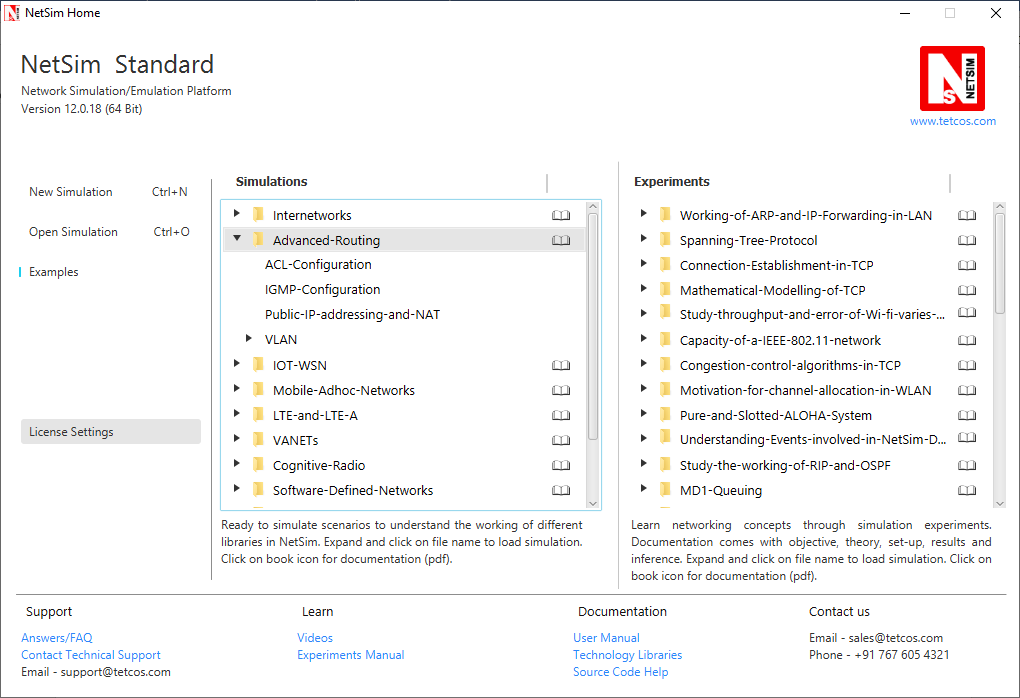
NetSim uses the following defaults for IGMP simulations:

* The multicast destination address is set to 239.12.14.5.
* The IGMP protocol starts only after 1 second in to the simulation.
* The multicast application starts only after 5 seconds in to the simulation.

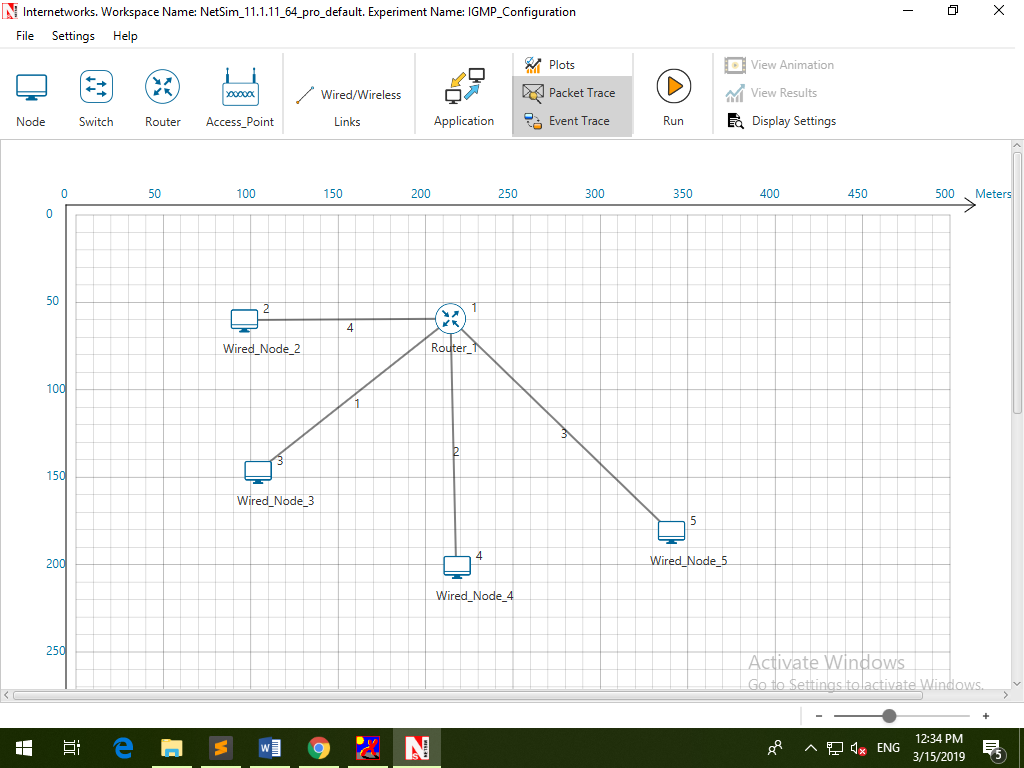
Note that NetSim does not support the following in IGMP:

* Leave Group message
* IGMP v1 compatibility

Open NetSim, Select Examples->Advanced-Routing->IGMP-Configuration as shown below:



The following network diagram illustrates what the NetSim UI displays when you open the example configuration file for IGMP.



1. See that by default, NetSim has enabled IGMP on the router, as follows:
   1. Right-click the router and click **Properties**.

The Router pop-up window appears.

* 1. Click **NETWORK LAYER** in the left area.
  2. **IGMP\_Status** drop-down list is set to **TRUE**.
  3. Click **OK**.

1. See that by default, NetSim has enabled IGMP on a node, as follows:
2. Right-click a wired node (say Wired\_Node\_2) and click **Properties**.

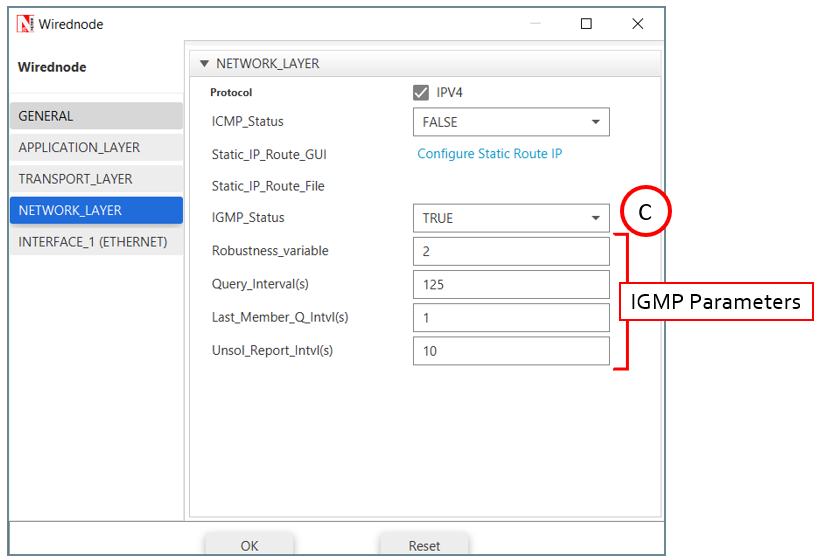
The Wired node pop-up window appears.

1. Click **NETWORK LAYER** in the left area.
2. **IGMP\_Status** drop-down list is set to **TRUE**.

The Wired node pop-up window displays the following parameters you can configure for IGMP, on the node:

* + - **Robustness\_variable**: The Robustness\_variable parameter allows you tune your subnet to a specific number of lost packets (packet loss) in the subnet.
    - Query\_Interval(s): The Query\_Interval(s) parameter allows you to specify the interval (in seconds) between two successive General Queries that a Querier multicast router sends.
    - Last\_Member\_Q\_Intvl(s): The Last\_Member\_Q\_Intvl(s) parameter allows you specify the interval (in seconds) between two successive Group-Specific Query messages that a multicast router sends to hosts.
    - Unsol\_Report\_Intvl(s): The Unsol\_Report\_Intvl(s) is the time between repetitions of a host’s initial report of membership in a group.

The following image illustrates the wired node pop-up window and the parameters you can configure for IGMP, on the node.



1. Click **OK**.
2. (Optional) Do the following to modify the parameters of IGMP.

* To modify the value of the Robustness\_variable, enter a value in the **Robustness\_variable** text box.

The default value of the Robustness\_variable parameter is 2.

You can enter a value between 2 and 10.

NetSim does not allow you to enter a value that is less than 2. If you enter a value that is less than 2, NetSim resets the value to 2.

Increase the value of the Robustness\_variable to more than 2, if you want to simulate a subnet that must lose more packets.

By default, IGMP is robust to (Robustness Variable-1) packet losses.

* To modify the value of the Query\_Interval(s), enter a value in seconds, in the **Query\_Interval(s)** text box.

The default value of the Query\_Interval(s) parameter is 125 seconds.

You can enter a value between 1 and 3600 seconds.

Fine-tune the Query\_Interval(s) parameter to control the number of IGMP messages on the subnet.

* To modify the value of the Query\_Interval(s), enter a value in seconds, in the Last\_Member\_Q\_Intvl(s) text box.

The default value of the Last\_Member\_Q\_Intvl(s) parameter is 1 second.

You can enter a value between 1 and 25 seconds.

Fine-tune the Last\_Member\_Q\_Intvl(s) parameter to make your subnet less or more bursty of IGMP messages.

* To modify the value of the Query\_Interval(s), enter a value in seconds, in the Unsolicited\_Report\_Interval(s) text box.

The default value of the Last\_Member\_Q\_Intvl(s) parameter is 10 seconds.

Fine-tune the Unsolicited\_Report\_Interval(s) parameter to make your subnet less or more bursty of IGMP messages.

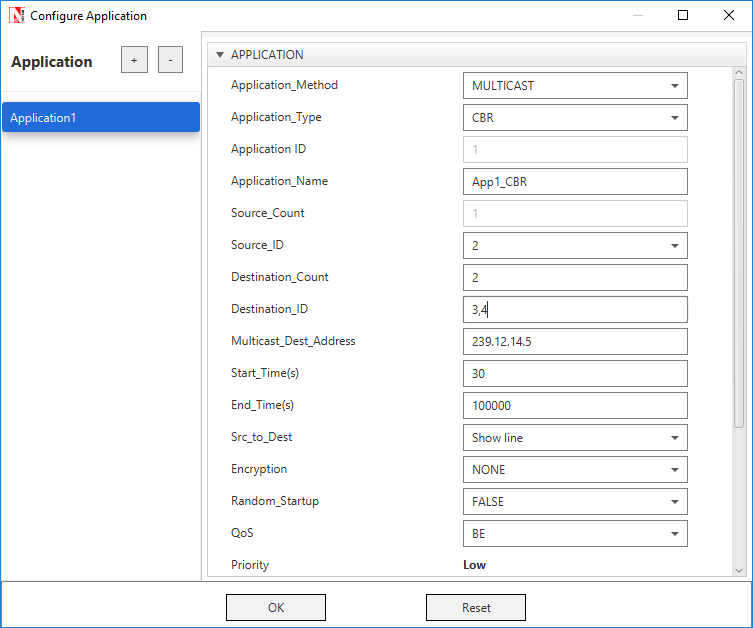
You can enter a value between 1 and 10,000 seconds.

1. Repeat steps 3 on other nodes to see that NetSim has enabled IGMP and step 4 on other nodes, if you to modify the IGMP parameters.
2. To configure a multicast application:
3. Click the Application icon located in the toolbar.

The Application pop-up window appears.

1. See that by default, NetSim has set the following properties for the multicast application:
2. **Application\_Method** = **MULTICAST**.
3. **Source\_ID** = **2**, which means **Wired\_Node\_2** node is the source of the application and the multicast traffic.
4. **Destination\_Count** = **2**, which means two nodes will receive multicast traffic from the multicast application.
5. **Destination\_ID** = **3, 4**, which means, W**ired\_Node\_3 and Wired\_Node\_4** nodes must receive multicast traffic from the multicast application.
6. Set application start time to 30s.
7. (Optional) Modify the properties except (i).

*Note:* You add more than one destination IDs, by separating two successive numbers by a “,” (comma). The following image illustrates the properties of the multicast application.



1. Click **OK**.
2. See that by default, NetSim has enabled the **Packet Trace** and **Event Trace** icons located in the toolbar.
3. To start and run the simulation:
4. Click the **Run** icon located in the toolbar.
5. Enter a numerical value in the **Simulation Time** text box, say 50s.
6. Click **OK**.

NetSim simulates IGMP for the time set

### Results

After NetSim simulates IGMP, a Simulation Results window appears.

You can do the following on this window:

* Print the results that NetSim displays in the Simulation Results window.
* View the packet trace details in a *.CSV* file and save the *.CSV* file to your computer.
* View the event trace details in a *.CSV* file and save the *.CSV* file to your computer.
* Export the results that NetSim displays in the Simulation Results window, in a spreadsheet.
* Close the Simulation Results window and return to your simulation.

NetSim also saves the last instance of your simulation for you to view, analyse, and download the results.

**Interpreting the IGMP Simulation**

Before you analyse the packet trace and event trace results, we recommend that you first interpret how IGMP worked with the parameters you specified. So, you must first view the simulation.

To view and interpret the simulation:

1. Close the **Simulation Results** window and return to your simulation.
2. Click the **View Animation** icon located on the toolbar.

The NetSim Packet Animation window appears.

1. Click the **Play** icon located on the toolbar.

You will see that the simulation runs IGMP.

The details of the packet traversing in your network appear as table located below the simulation window.

1. (Optional) To fine-tune the speed of the animation, use the **Animation Speed** slider located on the toolbar.

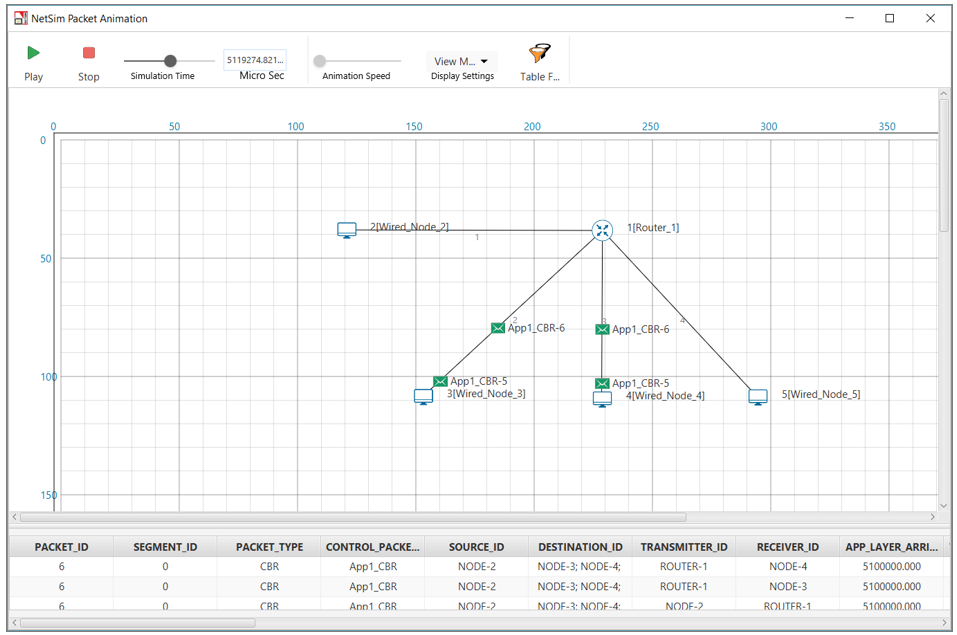
You will see the following happen in the animation:

1. Initially, all nodes (Wired\_Node\_2, 3, 4 and 5) receive the IGMP\_Memebership\_Query message from Router\_1.
2. When a node receives the IGMP\_Memebership\_Query message, the node sends the IGMP\_V2\_Membership\_Report to Router\_1 indicating that it is interested to join the multicast group.

You can see that Wired\_Node\_3 sends the IGMP\_V2\_Membership\_Report message to Router\_1. Wired\_Node 2, 4 and 5 also send the IGMP\_V2\_Membership\_Report message to Router\_1.

1. Router\_1 makes an entry for the membership in its routing table.

The following image illustrates IGMP at work.

****

When NetSim completes the simulation, the Simulation Results window appears.

**Analyzing the Packet Trace Results**

Now that you have seen the simulation for IGMP, we will analyze the communication between the nodes and the router.

To view and analyze the packet trace results:

1. On the Simulation Results window, click **Open Packet Trace** located in the left area.

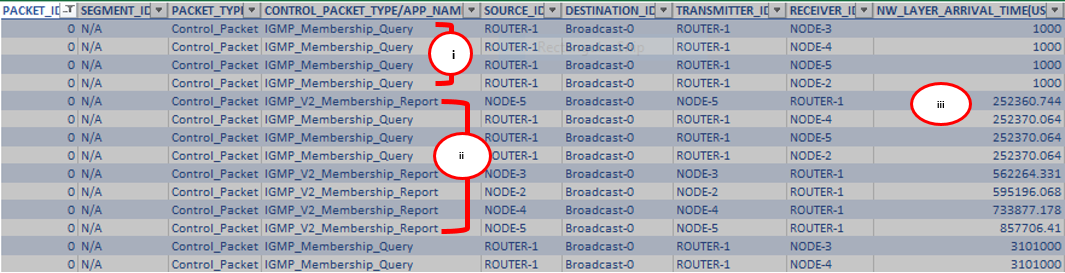
A *.CSV* appears.

1. Open the .*CSV* file and filter the **PACKET\_ID** column by **0** and **1**.

You will see the following in the *.CSV* file.

1. Router\_1 broadcasts the IGMP\_Memebership\_Query message to all the nodes.
2. When a node receives the IGMP\_Memebership\_Query message, the node sends the IGMP\_V2\_Membership\_Report message to the Router\_1.
3. The IGMP protocol starts to work only after 1 second in to the simulation.

The following image illustrates (i), (ii), and (iii).

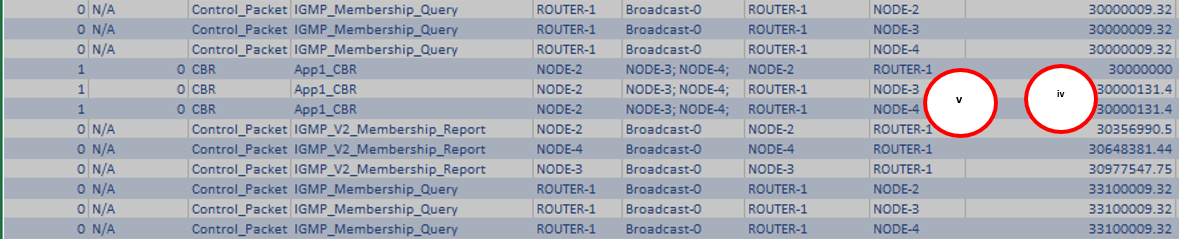


1. The multicast application Wired\_Node\_2 starts to send multicast traffic to Wired\_Node\_3 and Wired\_Node\_4 only after 5 seconds in to the simulation.

This is because, in NetSim, the multicast application starts after 5 seconds by default.

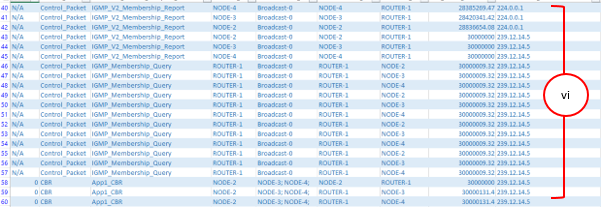
1. Wired\_Node\_2 multicasts Constant Bit Rate (CBR) packets only to Wired\_Node\_3 and Wired\_Node\_4.

The following image illustrates (iv), and (v).



1. Hosts send the IGMP\_V2\_Membership\_Report to 224.0.0.1 to the multicast application sends multicast traffic to 239.12.14.5.

The following image illustrates (vi).



**IGMP Event Trace Analysis:**

Now that you have seen the results of packet trace, we will analyze the event trace for this IGMP simulation.

To view the event trace results:

1. On the Simulation Results window, click **Open Event Trace** located in the left area.

A *.CSV* appears.

1. Open the .*CSV* file and filter the Event\_Type column by **NETWORK\_OUT** and **TIMER\_EVENT**.

You will see the following sub-events in the **Subevent\_Type** column:

1. **IGMP\_DelayTimer**: This sub-event occurs when a node sets the delay timers for every group (excluding the all-systems group) to which the node belongs, on the interface on which it received the query, after the node receives a General Query from the multicast router.
2. **IGMP\_GroupMembershipTimer:** This sub-event occurs when the multicast router refreshes the group membership interval timer, every time it receives a membership report for a multicast group. If this timer expires, the multicast router removes this group from the list of destinations for multicast traffic.
3. **IGMP\_SendQuery**: This sub-event occurs when the multicast router periodically (based on Query Interval) sends a Query message on every network to which the multicast router is connected, to solicit multicast group membership information.
4. **IGMP\_SendStartupQuery**: This subevent occurs when the multicast router sends the Startup query count to quickly and reliably determine the multicast group membership information, at startup.
5. **IGMP\_UnsolicitedReportTimer**: If the initial membership report is lost or damaged, this timer repeats once or twice after short delays, after every Unsolicited Report Interval.
6. **JOIN\_MULTICAST\_GROUP:** This sub-event occurs when a node sends the join multicast group message, when the node decides to join a multicast group on an interface.

In NetSim, a node joins a multicast group only after 5 seconds in to the simulation.

The following image illustrates that hosts join the multicast group after 5 seconds.



# Understand the working of “Connection Establishment” in TCP

## Introduction:

When two processes wish to communicate, their TCP’s must first establish a connection i.e. initialize the status information on each side. Since connections must be established between unreliable hosts and over the unreliable internet communication system, a “three-way handshake” with clock-based sequence numbers is the procedure used to establish a Connection. This procedure normally is initiated by one TCP and responded by another TCP. The procedure also works if two TCPs simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a “SYN” segment which carries no acknowledgement after it has sent a “SYN”.

The simplest three-way handshake is shown in the following figure:

**TCP A TCP B**

1. CLOSED LISTEN

2. SYN-SENT 🡪 <A: SEQ=100><CTL=SYN> 🡪SYN-RECEIVED

3. ESTABLISHED 🡨 <B: SEQ=300><ACK=101><CTL=SYN, ACK> 🡨SYN-RECEIVED

4. ESTABLISHED 🡪 <A: SEQ=101><ACK=301><CTL=ACK> 🡪ESTABLISHED

5. ESTABLISHED 🡪 <A: SEQ=101><ACK=301><CTL=ACK><DATA> 🡪ESTABLISHED

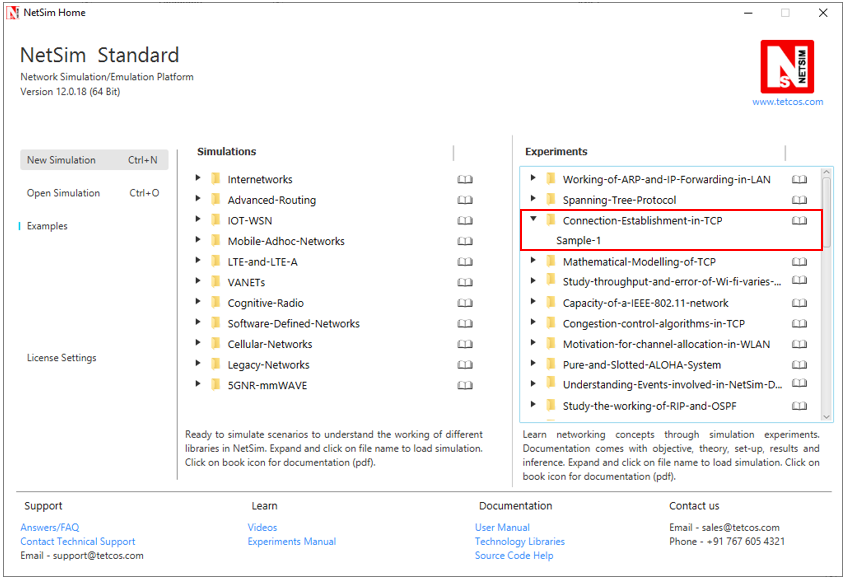
Fig: Basic 3-Way Handshake for Connection Synchronization

**Explanation:**

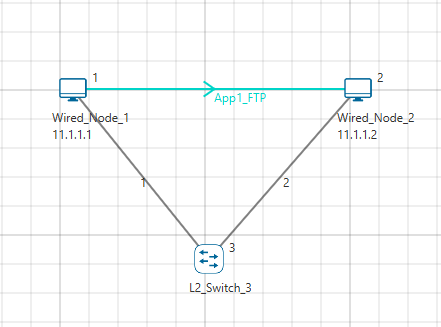
The above figure should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (🡪) indicates the departure of a TCP Segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (🡨 ) indicates the reverse. TCP states represent the state after the departure or arrival of the segment (whose contents are shown in the center of each line). Segment contents are shown in abbreviated form, with sequence number, control flags, and ACK field. In line 2 of the above figure, TCP A begins by sending a SYN segment indicating that it will use sequence numbers starting with sequence number 100. In line 3, TCP B sends a SYN and acknowledges the SYN it received from TCP A. Note that the acknowledgment field indicates, TCP B is now expecting to hear sequence 101, acknowledging the SYN which occupied sequence 100. At line 4, TCP A responds with an empty segment containing an ACK for TCP B's SYN and in line 5, TCP A sends some data.

## Network Setup:

Open NetSim and click **Examples > Experiments > Connection-Establishment-in-TCP > Sample-1** as shown below:



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



## Procedure:

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

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

**Step 2:** In the General Properties of Wired Node 1 i.e. Source, Wireshark Capture is set to Online.

***Note:*** *Accept default properties for L2 Switch as well as the Links.*

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

An FTP Application is generated from Wired Node 1 i.e. Source to Wired Node 2 i.e. Destination with File Size remaining 100000 Bytes and File Inter Arrival Time remaining 5 Seconds.

**Step 4:** Packet Trace is enabled in the NetSim GUI and hence we can observe the TCP Three-Way Handshake that occurs before the data packet transmissions.

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

**Step 6:** Click on Run simulation. The simulation time is set to 10 seconds. In the **Static ARP Configuration** tab, Static ARP is set to Disable.

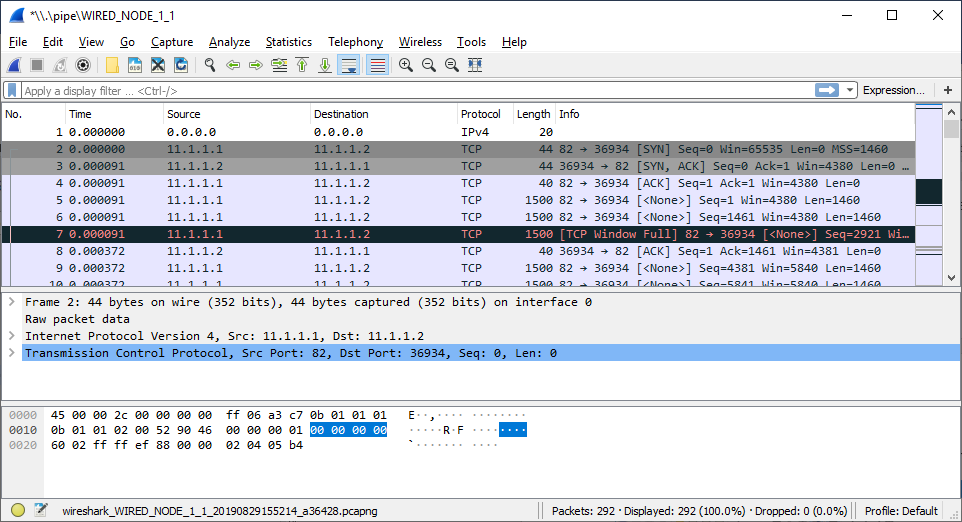
## Output:

Once the simulation begins, Wireshark starts to capture the packets.

1. NODE-1 (11.1.1.1 as per this scenario) sends a control packet of type TCP\_SYN requesting a connection with NODE-2 (11.1.1.2 as per this scenario).
2. NODE-2 responds with the control packet of type TCP\_SYN\_ACK to NODE-1. This TCP\_SYN\_ACK is the ACK packet sent for the TCP\_SYN packet.
3. NODE-1 then sends the TCP\_ACK to NODE-2 and internally sets its CONNECTION\_STATE as TCP\_ESTABLISHED.

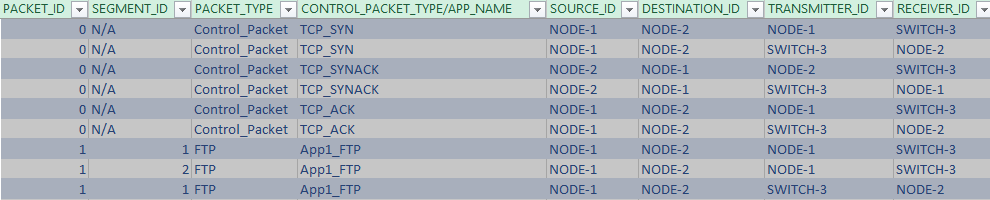
Once the connection is established, data transmission starts and we can see the data packets (of size 1500 bytes) sent from NODE-1 to NODE-2.

TCP Three-way Handshake can be observed in Wireshark as shown below:



Fig**:** 3-way handshake captured in Wireshark

Similarly, users can also see the TCP 3-Way-Handshake using the Packet Trace. To view the packet trace file, click on **“Open Packet Trace”** option present in the left-hand-side of the Results Dashboard.



Fig**:** 3-way handshake captured in NetSim Packet Trace

From the Packet Trace file, we can observe the following:

1. NODE-1 sends TCP\_SYN to NODE-2 via SWITCH-3.
2. NODE-2 responds by sending back TCP\_SYNACK to NODE-2 via SWITCH-3.
3. Then NODE-1 sends TCP\_ACK to NODE-2 via SWITCH-3 and then starts sending the data packets to NODE-2.

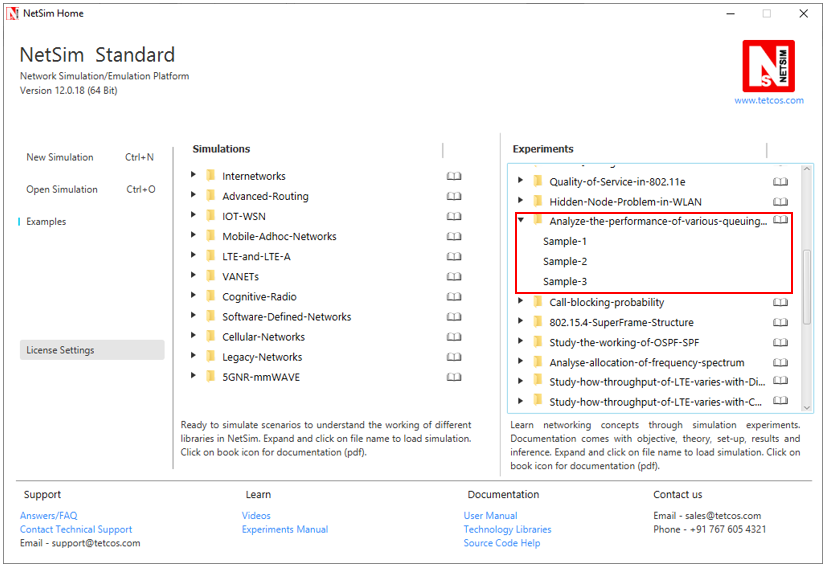
# Analyze the performance of FIFO, Priority and WFQ Queuing Disciplines

## Introduction

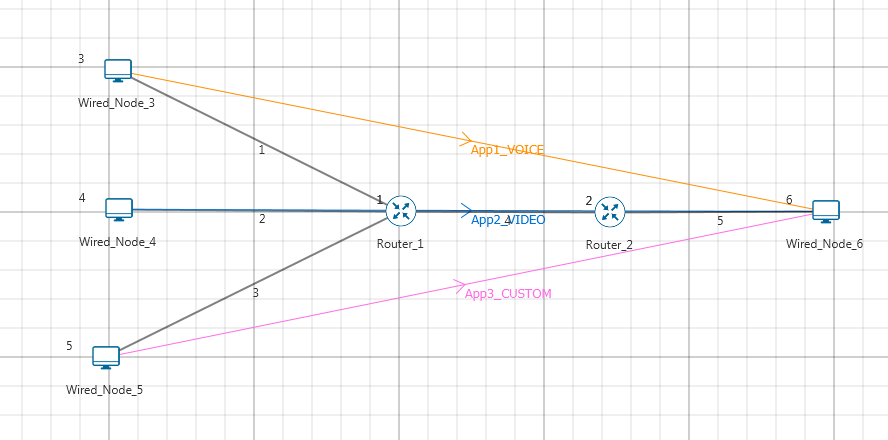
As part of the resource allocation mechanisms, each router must implement some queuing discipline that governs how packets are buffered while waiting to be transmitted. Various queuing disciplines can be used to control which packets get transmitted (based on bandwidth allocation) and which packets get dropped (based on buffer space). The queuing discipline also affects the latency experienced by a packet, by determining how long a packet waits to be transmitted. Examples of the common queuing disciplines are first-in-first-out (FIFO) queuing, priority queuing (PQ), and weighted-fair queuing (WFQ).

## Network Setup:

Open NetSim and click on **Examples > Experiments > Analyze-the-performance-of various-queuing-disciplines > Sample-1** as shown below:



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



## Procedure:

**Sample 1: (FIFO)**

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

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

**Step 2:** TCP Protocol is set to Disable in all the devices.

**Step 3:** Wired Link Properties is set as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Link Properties | Link 1 | Link 2 | Link 3 | Link 4 | Link 5 |
| Max Uplink Speed (Mbps) | 10 | 10 | 10 | 5 | 5 |
| Max Downlink Speed (Mbps) | 10 | 10 | 10 | 5 | 5 |

**Step 4:** In the **Interface WAN > Network Layer** Properties of Router 1, Scheduling Type is set as FIFO. Similarly, Scheduling Type is set as FIFO for Router 2.

**Step 5:** Three different applications are generated as per the table given below:

***NOTE: For Voice application set codec as Custom.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Application Properties | Application 1 | Application 2 | Application 3 | |
| Application Type | **Voice**  **(Codec-Custom)** | **Video** | | Custom |
| Source\_Id | **3** | **4** | | 5 |
| Destination\_Id | **6** | **6** | | 6 |
| QoS | **RTPS** | **NRTPS** | | BE |
| Packet Size | | | | | |
| Distribution | **Constant** | **Frame\_Per\_Sec** | | Constant |
| Value (bytes) | **1460** | **50** | | 1000 |
| Inter Arrival Time | | | | | |
| Distribution | **Constant** | **Pixel\_Per\_Frame** | | Constant |
| Value (micro secs) | 2336 | 100000 | | 1333 |

**Step 6:** Run the Simulation for 10 Seconds. Note down the Application Throughput.

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

**Sample 2: (Priority)**

**Step 1:** In the **Interface WAN > Network Layer** Properties of Router 1, Scheduling Type is set as PRIORITY. Similarly, Scheduling Type is set as PRIORITY for Router 2.

**Step 2:** Run the Simulation for 10 Seconds. Note down the Application Throughput.

**Sample 3: (WFQ)**

**Step 1:** In the **Interface WAN > Network Layer** Properties of Router 1, Scheduling Type is set as WFQ. Similarly, Scheduling Type is set as WFQ for Router 2.

**Step 2:** Run the Simulation for 10 Seconds. Note down the Application Throughput.

## Measurements and Outputs:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Application | Traffic Generation Rate (Mbps)\* | FIFO-Sample-1 Throughput (Mbps) | Priority-Sample-2 Throughput (Mbps) | WFQ-Sample-3 Throughput (Mbps) |
| Voice | 5 | 1.75  ~ (5 / 13.6) \*5 | 3.81 | 1.89 |
| Video | 2.6 | 0.89  ~ (2.6/13.6) \*5 | 0.28 | 0.85 |
| Custom | 6 | 2.12  ~ (6/13.6) \*5 | 0.70 | 2.02 |
| Total | 13.6 | 4.76  ~ 5 | 4.79  ~ 5 | 4.76  ~ 5 |

***NOTE: For Traffic Generation Rate calculation please refer user manual section 5.3***

\*The traffic generation rate is based on settings done in step 5.

The 5 mentioned above refers to 5 Mbps which is the data rate of link 4.

## Inference

In FIFO, packets will get served based on their packet arrival time to router. Therefore, since link 4 is a 5 Mbps link, the throughputs of Voice, Video and Custom applications is equal to the ratio of their generation rates.

Priority scheduling technique processes packets based on their priority. Hence voice and video which have higher priority take up the complete bandwidth available.

Weighted fair queuing (WFQ) assigns a weight to each application and hence gives a result between that is in between priority and FIFO.

# **Understand the working of Slow start and Congestion Avoidance (Old Tahoe), Fast Retransmit (Tahoe) and Fast Recovery (Reno) Congestion Control Algorithms in TCP.**

## Theory:

One of the important functions of a TCP Protocol is congestion control in the network. Given below is a description of the working of Old Tahoe and Tahoe variants (of TCP) control congestion.

**Old Tahoe:** Old Tahoe is one of the earliest variants of TCP. It implements two algorithms called slow start and congestion avoidance to update the congestion window.

**Slow Start:** At the start of data transmission the size of congestion window is one. This means TCP can send only one packet until it receives an acknowledgement. When the ACK is received by the sender the congestion window increases to two. Now the sender can send two data packets. Upon the arrival of every new ACK the sender increases its congestion window by one. This phase is known as the slow start phase where the congestion window increases exponentially. So on the arrival of a new ACK, 𝑐𝑤𝑛𝑑 + = 𝑀𝑆𝑆;

**Congestion Avoidance:** TCP will continue the slow start phase until it reaches a certain threshold, or if packet loss occurs. Now it enters in to a phase called congestion avoidance. Here the congestion window grows linearly. This means that the congestion window increases from ‘n’ to ‘n+1’ only when it has received ‘n’ new ACKs. The rate of growth of congestion window slows down because this is the stage where TCP is susceptible to packet loss. The formula used here is 𝑐𝑤𝑛𝑑 += (𝑆𝑀𝑆𝑆 ∗ 𝑆𝑀𝑆𝑆)/𝑐𝑤𝑛d

**Tahoe (Fast Retransmit):** TCP Tahoe implements all the above mentioned algorithms used by Old Tahoe. The Fast Retransmit algorithm was included in Tahoe to improve the response time of TCP.

**Fast Retransmit:** One of the major drawbacks of Old Tahoe is that it depends on the timer to expire before it can retransmit a packet. TCP Tahoe tries to improve upon Old Tahoe by implementing the Fast Retransmit algorithm. Fast Retransmit takes advantage of the fact that duplicate ACKs can be an indication that a packet loss has occurred. So, whenever it receives 3 duplicate ACKs it assumes that a packet loss has occurred and retransmits the packet.

**Reno (Fast Recovery):** TCP Reno retains the basic principles of Tahoe such as Slow Start, Congestion Avoidance and Fast Retransmit. However it is not as aggressive as Tahoe in the reduction of the congestion window. Reno implements the Fast Recovery algorithm which is described below.

**Fast Recovery:** The congestion window drop to one on the arrival of a 3 duplicate ACK can be considered as an extreme precaution. Arrival of 3 duplicate ACKs corresponds to light congestion in the network and there is no need for the congestion window to drop down drastically. The Fast Recovery algorithm does the following on the arrival of a third duplicate ACK:

The threshold value is set to half of the congestion window. 𝑠𝑠𝑡ℎ𝑟𝑒𝑠ℎ = 𝑐𝑤𝑛𝑑/2.

The congestion window is now set to be threshold plus three times the MSS. 𝑐𝑤𝑛𝑑 = 𝑠𝑠𝑡ℎ𝑟𝑒𝑠ℎ + 3 ∗ 𝑆𝑀𝑆𝑆.

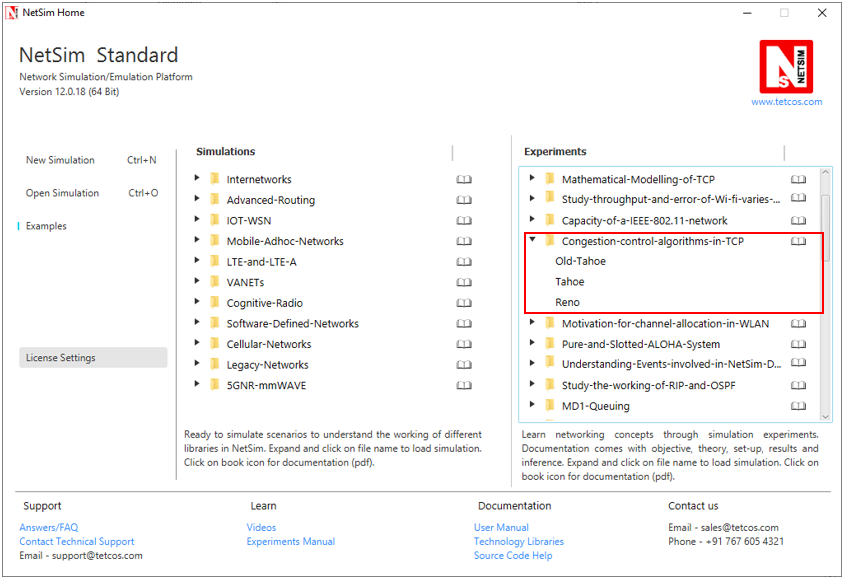
On the arrival of another duplicate ACK the congestion window increases by one MSS. This is done because an ACK signifies that a segment if out of the network and the sender can pump in another packet into the network. This is somewhat similar to slow start. 𝑐𝑤𝑛𝑑 += 𝑆𝑀𝑆𝑆.

TCP remains in fast recovery phase until it receives a higher ACK from the receiver.

On receiving a higher ACK the congestion window is set to the threshold value. From now onwards congestion avoidance is followed. 𝑐𝑤𝑛𝑑 = 𝑠𝑠𝑡ℎ𝑟𝑒𝑠ℎ.

## Network Set Up:

Open NetSim and click **Examples > Experiments > Congestion-control-algorithms-in- TCP** **> Old-Tahoe** as shown below:



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



## Procedure:

**Old Tahoe:**

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

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

**Step 2:** In the Source Node, i.e. Wired Node 3, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to OLD TAHOE.

***Note:*** *Accept default properties for the Router.*

**Step 3:** The Link Properties are set according to the table given below:

|  |  |  |  |
| --- | --- | --- | --- |
| Link Properties | Wired Link 1 | Wired Link 2 | Wired Link 3 |
| Uplink Speed (Mbps) | 20 | 100 | 20 |
| Downlink Speed (Mbps) | 20 | 100 | 20 |
| Uplink propagation delay (µs) | 5 | 100 | 5 |
| Downlink propagation delay (µs) | 5 | 100 | 5 |
| Uplink BER | 0.0000001 | 0.0000001 | 0.0000001 |
| Downlink BER | 0.0000001 | 0.0000001 | 0.0000001 |

Wired links between the Node and a Router are configured with a rate lower than that of the packet generation rate to create bottle necks.

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

A CBR Application is generated from Wired Node 3 i.e. Source to Wired Node 4 i.e. Destination with Packet Size remaining 1460Bytes and Inter Arrival Time set to 400 µs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 30 Mbps (Approx.). Generation Rate can be calculated using the formula:

**Step 5:** Click on Run simulation. The simulation time is set to 30 seconds.

**Tahoe:**

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

**Step 1:** In the Source Node, i.e. Wired Node 3, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to TAHOE.

**Step 2:** Click on Run simulation. The simulation time is set to 30 seconds.

**Reno:**

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

**Step 1:** In the Source Node, i.e. Wired Node 3, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to RENO.

**Step 2:** Click on Run simulation. The simulation time is set to 30 seconds.

## Output

**Comparison Table:**

|  |  |
| --- | --- |
| Congestion Control Algorithm | Throughput (Mbps) |
| Old Tahoe | 7.26 |
| Tahoe | 16.10 |
| Reno | 17.76 |

From the above table, throughput for Tahoe is high since Tahoe retransmits the packets faster than Old Tahoe. Throughput for Reno is higher compared to Tahoe since *cwnd* is set to *ssthresh* instead of setting to 1 SMSS.

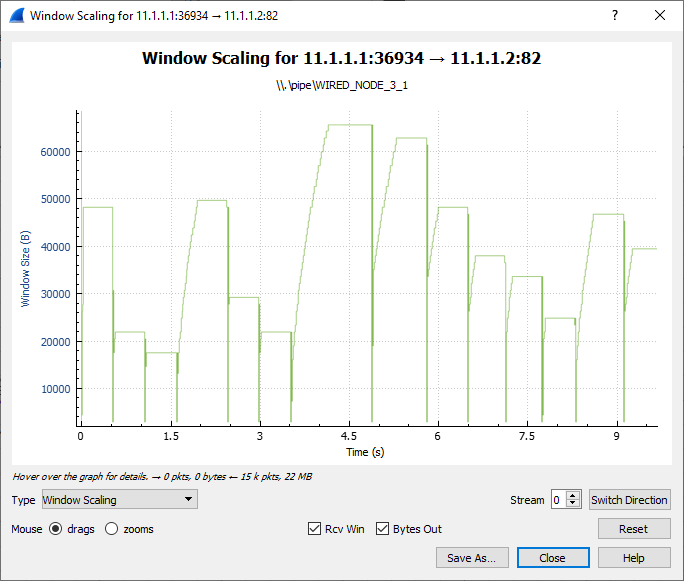
Go to the Wireshark Capture window.

Clickon data packet i.e. <None>. Go to Statistics 🡪 TCP Stream Graphs 🡪 Window Scaling.

Click on Switch Direction in the window scaling graph window to view the graph.

(For more guidance, refer to section - 7.7.5 Window Scaling in user manual)

**Old Tahoe:**

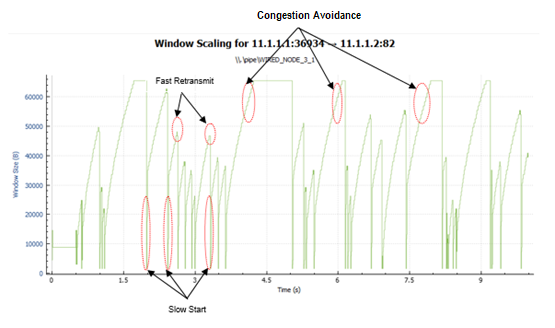


Slow Start

Congestion Avoidance

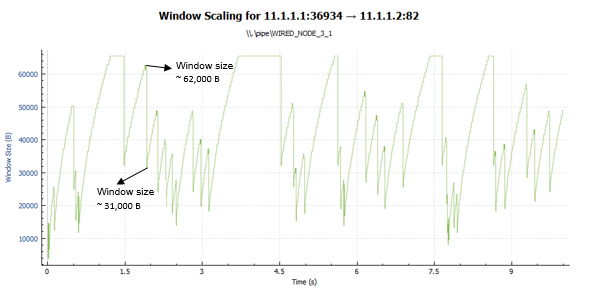
The graph shown above is a plot of Congestion window vs. Time. Each point on the graph represents the congestion window at the time when the packet is sent. You can observe that after every congestion window drop (caused by packet loss or timer expiry) Old Tahoe enters slow start, thereby increasing its window exponentially and then enters congestion avoidance where it increases the congestion window linearly.

**Tahoe:**

****

Tahoe uses the fast retransmit algorithm with which it responds to packet errors faster than Old Tahoe. Comparing the graphs of Old Tahoe and Tahoe one can observe that the latter drops the congestion window and retransmits faster than Old Tahoe (when three duplicate ack’s are received).

**Reno:**



TCP Reno upon receiving the third duplicate ACK sets its congestion window according to the formula cwnd = cwnd/2 + 3\*MSS. This can be observed in the above graph. On further arrival of duplicate ACKs it increases its congestion window by one MSS. If it receives a higher ACK then it drops the congestion window to the new threshold value. This mechanism is known as fast recovery.

## Inference:

From this experiment we were able to understand how selection of TCP congestion control algorithm can have an impact on the throughput experienced by the applications. The major difference between Old Tahoe, Tahoe and Reno algorithms is the time taken to retransmit packets and the way congestion window size is reduced. There is a considerable improvement in the performance as we go from Old Tahoe to Tahoe and then to Reno. This is evident from the throughput readings obtained from the simulations performed.

# Understand the working of TCP BIC Congestion control algorithm, simulate and plot the TCP congestion window

## Theory

In BIC congestion control is viewed as a searching problem in which the system can give yes/no feedback through packet loss as to whether the current sending rate (or window) is larger than the network capacity. The current minimum window can be estimated as the window size at which the flow does not see any packet loss. If the maximum window size is known, we can apply a binary search technique to set the target window size to the midpoint of the maximum and minimum. As increasing to the target, if it gives any packet loss, the current window can be treated as a new maximum and the reduced window size after the packet loss can be the new minimum. The midpoint between these new values becomes a new target. Since the network incurs loss around the new maximum but did not do so around the new minimum, the target window size must be in the middle of the two values. After reaching the target and if it gives no packet loss, then the current window size becomes a new minimum, and a new target is calculated. This process is repeated with the updated minimum and maximum until the difference between the maximum and the minimum falls below a preset threshold, called the minimum increment (Smin). This technique is called binary search increase.

**Additive Increase:**

In order to ensure faster convergence and RTT-fairness, binary search increase is combined with an additive increase strategy. When the distance to the midpoint from the current minimum is too large, increasing the window size directly to that midpoint might add too much stress to the network. When the distance from the current window size to the target in binary search increase is larger than a prescribed maximum step, called the maximum increment (Smax) instead of increasing window directly to that midpoint in the next RTT, we increase it by Smax until the distance becomes less than Smax, at which time window increases directly to the target. Thus, after a large window reduction, the strategy initially increases the window linearly, and then increases logarithmically. This combination of binary search increase and additive increase is called as binary increase. Combined with a multiplicative decrease strategy, binary increase becomes close to pure additive increase under large windows. This is because a larger window results in a larger reduction by multiplicative decrease and therefore, a longer additive increase period. When the window size is small, it becomes close to pure binary search increase – a shorter additive increase period.

**Slow Start:**

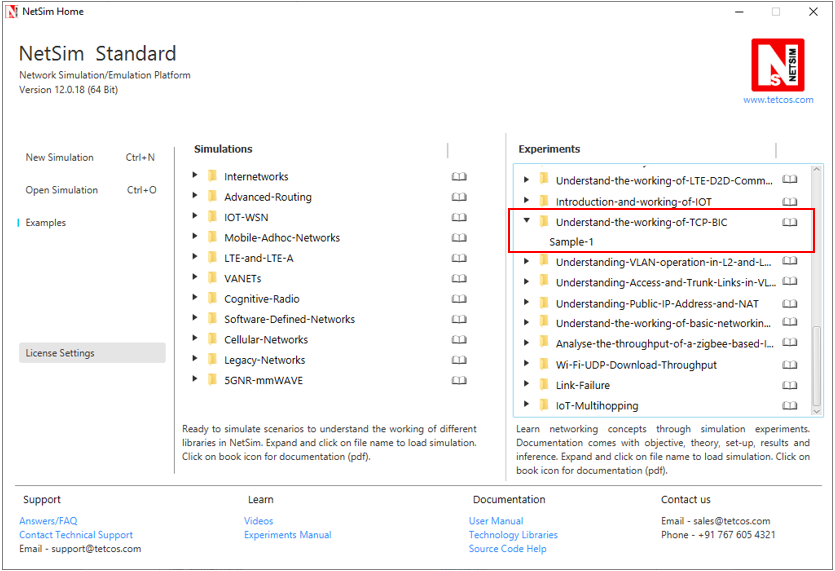
After the window grows past the current maximum, the maximum is unknown. At this time, binary search sets its maximum to be a default maximum (a large constant) and the current window size to be the minimum. So, the target midpoint can be very far. According to binary increase, if the target midpoint is very large, it increases linearly by the maximum increment. Instead, run a “slow start” strategy to probe for a new maximum up to Smax. So if cwnd is the current window and the maximum increment is Smax, then it increases in each RTT round in steps cwnd+1, cwnd+2, cwnd+4,…, cwnd+Smax. The rationale is that since it is likely to be at the saturation point and also the maximum is unknown, it probes for available bandwidth in a “slow start” until it is safe to increase the window by Smax. After slow start, it switches to binary increase.

**Fast Convergence:**

It can be shown that under a completely synchronized loss model, binary search increase combined with multiplicative decrease converges to a fair share. Suppose there are two flows with different window sizes, but with the same RTT. Since the larger window reduces more in multiplicative decrease (with a fixed factor β), the time to reach the target is longer for a larger window. However, its convergence time can be very long. In binary search increase, it takes RTT rounds to reach the maximum window after a window reduction of d. Since the window increases in a log step, the larger window and smaller window can reach back to their respective maxima very fast almost at the same time (although the smaller window flow gets to its maximum slightly faster). Thus, the smaller window flow ends up taking away only a small amount of bandwidth from the larger flow before the next window reduction. To remedy this behaviour, binary search increase is modified as follows. After a window reduction, new maximum and minimum are set. Suppose these values are max\_wini and min\_wini for flow i (i =1, 2). If the new maximum is less than the previous, this window is in a downward trend. Then, readjust the new maximum to be the same as the new target window (i.e. max\_wini = (max\_wini-min\_wini)/2), and then readjust the target. After that apply the normal binary increase. This strategy is called fast convergence.

## Network setup:

Open NetSim and click **Examples > Experiments > Understand-the-working-of-TCP-BIC > Sample-1** as shown below:



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



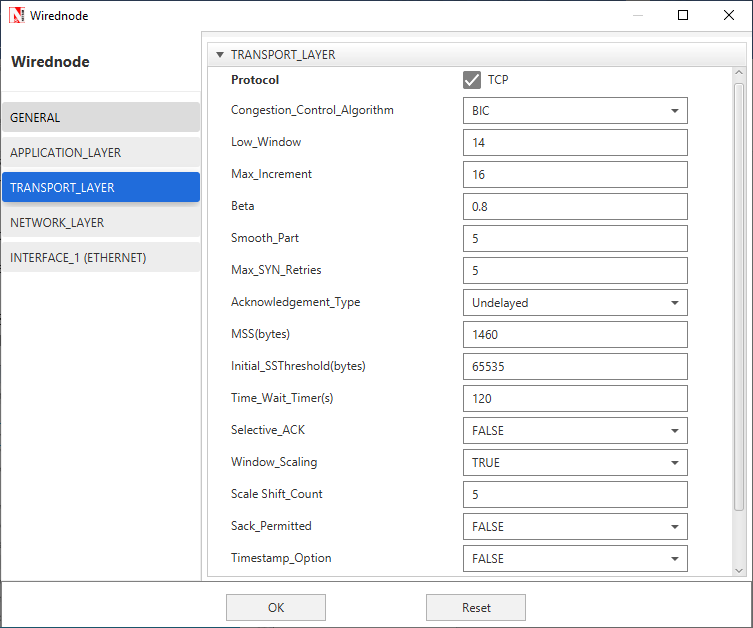
## Procedure:

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

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

**Step 2:** In the General Properties of Wired Node 3 i.e. Source, Wireshark Capture is set to Online and in the TRANSPORT LAYER Properties, Window Scaling is set as TRUE.

**Step 3:** For all the devices, in the TRANSPORT LAYER Properties, Congestion Control Algorithm is set to BIC.



**Step 4:** The Link Properties are set according to the table given below:

|  |  |  |  |
| --- | --- | --- | --- |
| Link Properties | Wired Link 1 | Wired Link 2 | Wired Link 3 |
| Uplink Speed (Mbps) | 20 | 100 | 20 |
| Downlink Speed (Mbps) | 20 | 100 | 20 |
| Uplink propagation delay (µs) | 5 | 1000 | 5 |
| Downlink propagation delay (µs) | 5 | 1000 | 5 |
| Uplink BER | 0.00000001 | 0.00000001 | 0.00000001 |
| Downlink BER | 0.00000001 | 0.00000001 | 0.00000001 |

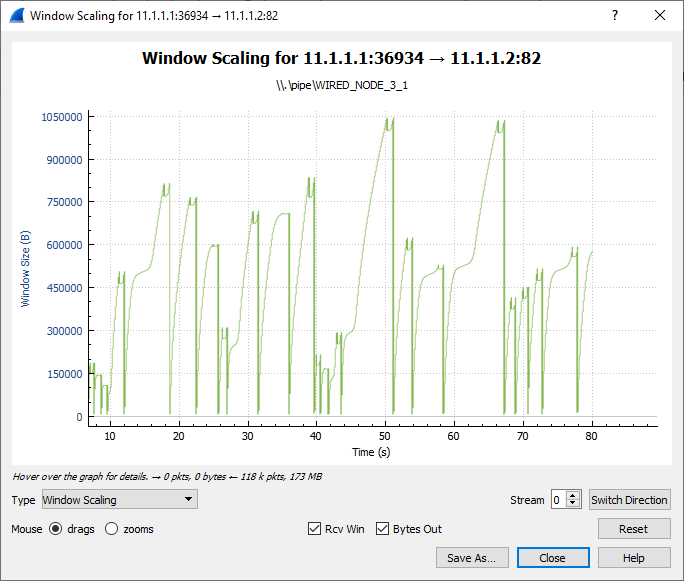
**Step 5:** Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CBR Application is generated from Wired Node 3 i.e. Source to Wired Node 4 i.e. Destination with Packet Size set to 70 Bytes and Inter Arrival Time set to 400 µs. Additionally, the **“Start Time”** parameter is set to 20 Seconds.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 140 Kbps. Generation Rate can be calculated using the formula:

**Step 6:** Click on Run simulation. The simulation time is set to 100 seconds.

## Output:



Fast Convergence

Slow Start

Binary Search

Additive Increase

Go to the Wireshark Capture window.

Clickon data packet i.e. <None>. Go to Statistics 🡪 TCP Stream Graphs 🡪 Window Scaling.

Click on Switch Direction in the window scaling graph window to view the graph.

(For more guidance, refer to section - 7.7.5 Window Scaling” in user manual)

The graph shown above is a plot of Congestion Window vs Time of BIC for the scenario shown above. Each point on the graph represents the congestion window at the time when the packet is sent. You can observe Binary Search, Additive Increase, Fast Convergence, Slow Start phases in the above graph.

# Quality of Service (QoS) in 802.11e based WLANs

## Theory:

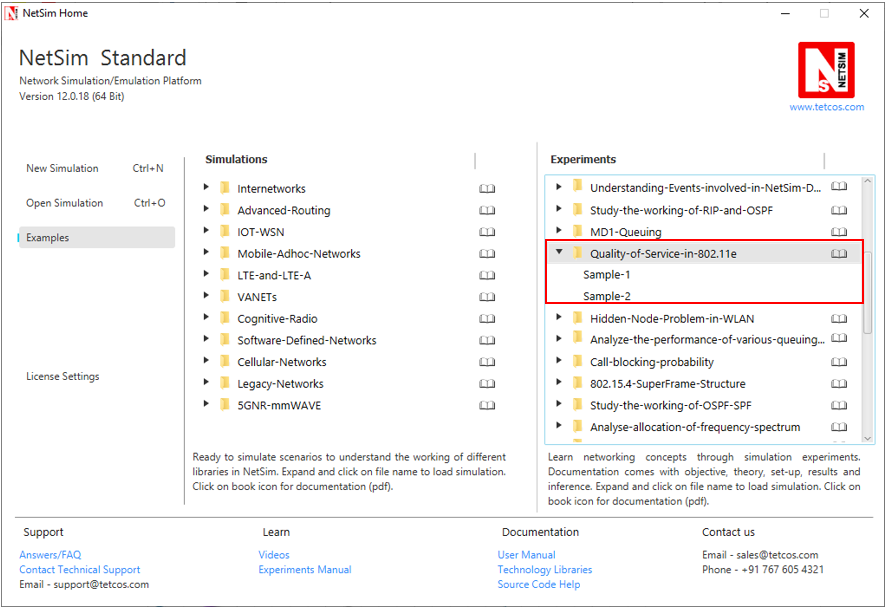
IEEE 802.11e Medium Access Control (MAC) is a supplement to the IEEE 802.11 Wireless Local Area Network (WLAN) standard to support Quality-of-Service (QoS). When 802.11e is enabled high-priority traffic has a higher chance of being sent than low-priority traffic: an application with high priority traffic waits a little less before its packet is processed and compared to an application with low priority traffic. The various application traffic generated in NetSim have the following priority and QoS values:

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

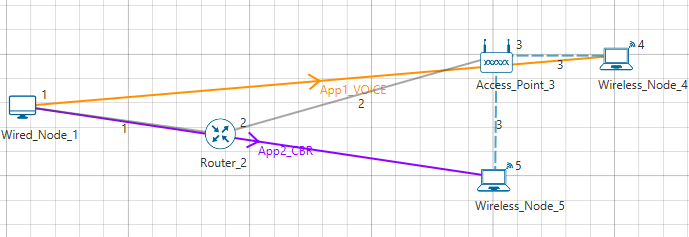
eRTPS QoS class is available in NetSim which has a priority value of 4. The QoS class for each application mentioned in the table above is fixed and can be changed by the user.

## Network Setup:

Open NetSim and click on **Examples > Experiments > Quality-of-Service-in-802.11e > Sample-1** as shown below:



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



## Procedure:

**Sample 1:**

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

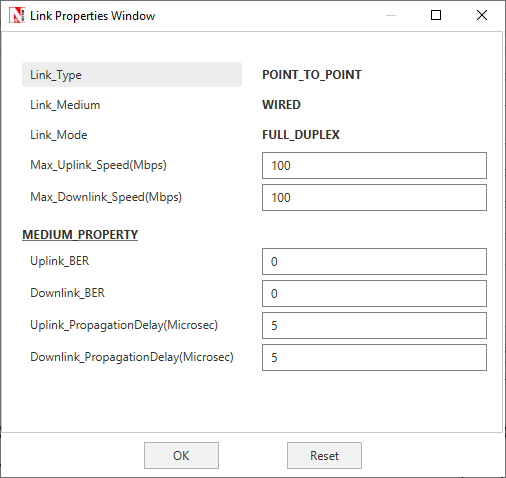
**Step 1:** A network scenario is designed in NetSim GUI comprising of 1 Wired Node, 2 Wireless Nodes, 1 Router, and 1 Access Point in the **“Internetworks”** Network Library.

**Step 2:** The device positions are set as per the below table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Access Point 2 | Wireless Node 4 | Wireless Node 5 |
| X/Lat | 250 | 300 | 250 |
| Y/Lon | 100 | 100 | 150 |

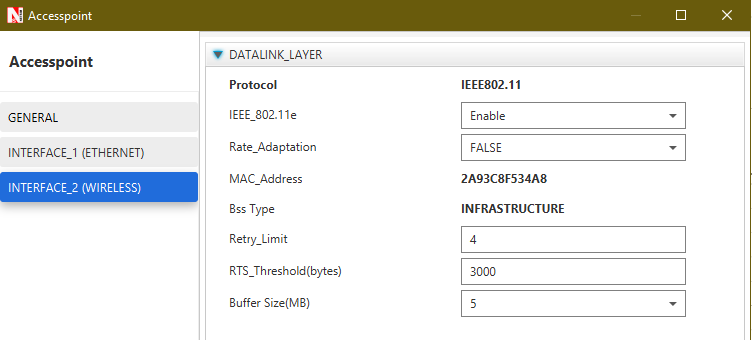
**Step 3:** TCP Protocol is set to Disable in all the devices.

**Step 4:** Wired Link Properties is set as follows:



**Step 5:** Go to Wireless Link Properties, the **“Channel Characteristics”** is set to NO PATHLOSS.

**Step 6:** In the **Interface Wireless > Data Link Layer** Properties of the Access Point, IEEE 802.11e is set to Enable and Buffer Size is set to 5MB.



–

–

**Step 7:** Right click on the Application Flow **App1 VOICE** or **App2 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A VOICE Application is generated from Wired Node 1 i.e. Source to Wireless Node 4 i.e. Destination with Packet Size set to 1000 Bytes and Inter Arrival Time set to 800µs. The **“Codec”** parameter is set to Custom.

A CBR Application is generated from Wired Node 1 i.e. Source to Wireless Node 5 i.e. Destination with Packet Size set to 1000 Bytes and Inter Arrival Time set to 800µs.

The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 10 Mbps. Generation Rate can be calculated using the formula:

**Step 8:** Run the Simulation for 10 Seconds. Note down the Application Throughput.

**Sample 2:**

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

**Step 1:** In the **Interface Wireless > Datalink Layer** Properties of the Wireless Node 5 and Access Point, IEEE 802.11e is set to Disable.

**Step 2:** Run the Simulation for 10 Seconds. Note down the Application Throughput.

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IEEE 802.11e | Application | Generation rate (Mbps) | Throughput (Mbps) | Delay (Micro. Sec.) |
| Enable  (Sample 1) | Voice | 10 | 3.22 | 945561.8 |
| CBR | 10 | 2.14 | 6466262.9 |
| Disable  (Sample 2) | Voice | 10 | 2.64 | 3672706.7 |
| CBR | 10 | 2.64 | 3671315.4 |

## Inference:

In sample 1, since QoS is enabled voice sees a higher priority than CBR. Hence voice packets in the queue are first transmitted before CBR packets are transmitted. In sample 2, since QoS has been disabled, priority is not considered for the applications. Hence they both see the same throughput.

As an additional note, when QoS is enabled the throughput for voice is 3.22 Mbps and for CBR it is 2.14, and when QoS is disabled the throughput for both is 2.64 Mbps per application or 5.28 Mbps for both applications put together. This value of around 5.5 Mbps is the maximum throughput an 802.11b access point can support. There is a slight drop in overall throughput when stations are present due to contention between the two stations.

1. It may be noted that the term *capacity* has several connotations in communications. Our use of the word here must not be confused with the notion of *information theoretic capacity* of a communication channel. [↑](#footnote-ref-2)