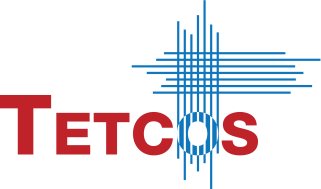


Experiments Manual



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.**

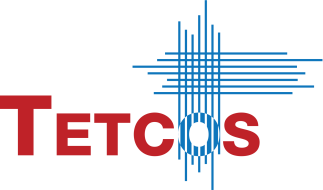
**Contact us at**

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. Performance and analyze of a set of Local Area Networks interconnected by switches and Hubs 3](#_Toc21449002)

[2. Interfacing tail with NetSim 9](#_Toc21449003)

[3. Understand the working of Slow start and Congestion Avoidance (Old Tahoe), Fast Retransmit (Tahoe) and Fast Recovery (Reno) Congestion Control Algorithms in TCP. 14](#_Toc21449004)

[4. Understand the working of “Connection Establishment” in TCP 20](#_Toc21449005)

[5. Analyze the performance Round-Robin and WFQ Queuing Disciplines 24](#_Toc21449006)

[6. Evaluate the performance of TCP and UDP protocol 27](#_Toc21449007)

[7. Evaluate the performance of AODV and DSR routing protocol 29](#_Toc21449008)

[8. Simulate and Analysis of ICMP and IGMP packets 32](#_Toc21449009)

[9. CDMA Downlink 47](#_Toc21449010)

[10. Evaluate the performance of IEEE802.11 and IEEE802.15.4 49](#_Toc21449011)

# Performance and analyze of a set of Local Area Networks interconnected by switches and Hubs

## Theory:

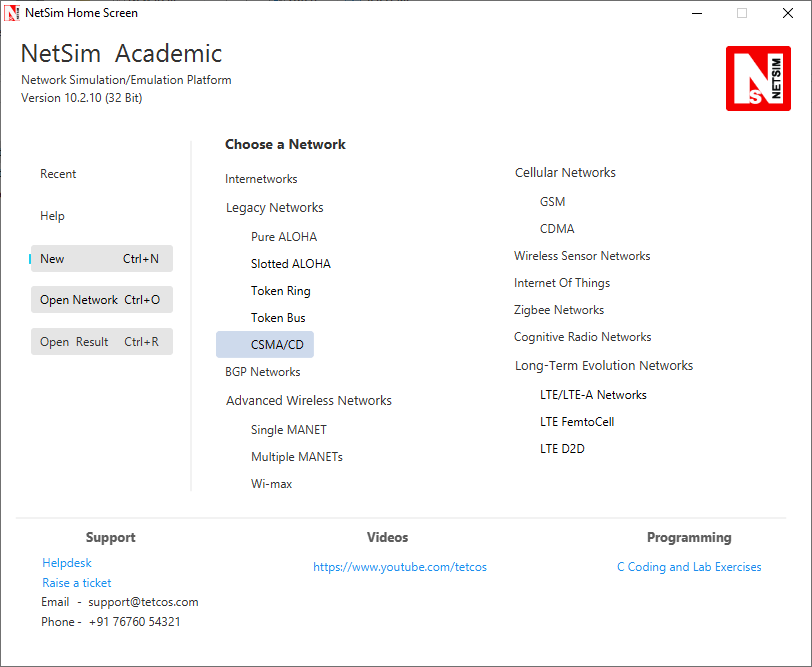
Hubs are commonly used to connect [segments](http://www.webopedia.com/TERM/S/segment.html) of a [LAN](http://www.webopedia.com/TERM/L/local_area_network_LAN.html) and contains multiple [ports](http://www.webopedia.com/TERM/P/port.html). When a [packet](http://www.webopedia.com/TERM/P/packet.html) arrives at one port, it is copied to the other ports so that all segments of the LAN can see all packets. In a hub, a frame is passed along or "broadcast" to every one of its ports. It doesn't matter that the frame is only destined for one port. The hub has no way of distinguishing which port a frame should be sent to. Passing it along to every port ensures that it will reach its intended destination. This place a lot of traffic on the network and can lead to poor network response times.

Additionally, a hub must share its [bandwidth](http://www.webopedia.com/TERM/B/bandwidth.htm) with each and every one of its ports. So, when only one PC is broadcasting, it will have access to the maximum available bandwidth. However, if multiple PCs are broadcasting, then that bandwidth will need to be divided among all of those systems, which will degrade performance.

Switch is point-to-point connection. I[n a network,](http://www.webopedia.com/TERM/N/network.html) Switches [filter](http://www.webopedia.com/TERM/F/filter.html) and forwardl [packets](http://www.webopedia.com/TERM/P/packet.html) between LAN segments. Unlike a hub, a switch will allocate a full data rate to each of its ports. So regardless of the number of PCs transmitting, users will always have access to the maximum amount of bandwidth. It's for these reasons a switch is considered to be a much better choice than a hub.

## Network Setup:

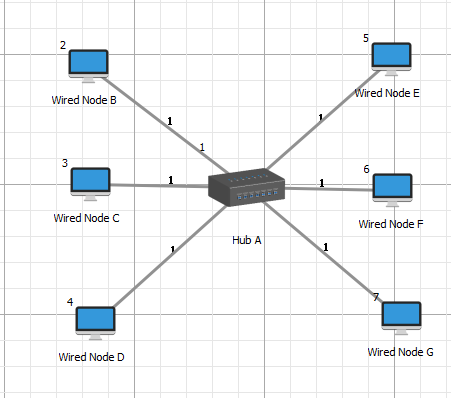
Click on **New 🡪 Legacy Network 🡪 CSMA/CD**



## Procedure:

### Part-A – HUB

**Step 1:** Create a network scenario using a hub and 6 wired nodes as shown below:



***Note: Accept the default properties for the hub and the wired nodes.***

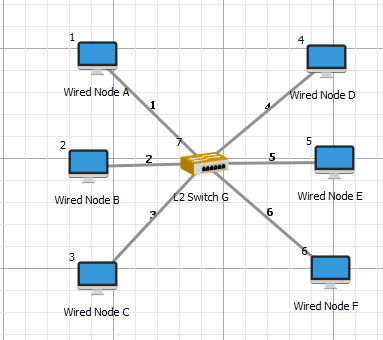
**Step 2:** To set application, click on application icon present in the top ribbon/toolbar. A configure application window will get open. Set the parameters as shown below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Application Properties** | | | | | | |
| Application Type | Custom | Custom | Custom | Custom | Custom | Custom |
| Source ID | 2 | 3 | 4 | 5 | 6 | 7 |
| Destination ID | 3 | 4 | 5 | 6 | 7 | 2 |
| **Packet Size** | | | | | | |
| Distribution | Constant | Constant | Constant | Constant | Constant | Constant |
| Value | 1000 Bytes | 1000 Bytes | 1000 Bytes | 1000 Bytes | 1000 Bytes | 1000 Bytes |
| **Inter Arrival Time** | | | | | | |
| Distribution | Constant | Constant | Constant | Constant | Constant | Constant |
| Value | 3200 µs | 3200 µs | 3200 µs | 3200 µs | 3200 µs | 3200 µs |
|  |  |  |  |  |  |  |

**Step 3:** Click on Run simulation. Set the simulation time to default 100 seconds and click on OK.

### Part-B – SWITCH

Create a network scenario using a switch and 6 wired nodes as shown below:



***Note: Accept the default properties for the switch and the wired nodes.***

**Step 2:** To set application, click on application icon present in the top ribbon/toolbar. A configure application window will get open. Set the parameters as shown below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Application Properties** | | | | | | |
| Application Type | Custom | Custom | Custom | Custom | Custom | Custom |
| Source ID | 2 | 3 | 4 | 5 | 6 | 7 |
| Destination ID | 3 | 4 | 5 | 6 | 7 | 2 |
| **Packet Size** | | | | | | |
| Distribution | Constant | Constant | Constant | Constant | Constant | Constant |
| Value | 1000 Bytes | 1000 Bytes | 1000 Bytes | 1000 Bytes | 1000 Bytes | 1000 Bytes |
| **Inter Arrival Time** | | | | | | |
| Distribution | Constant | Constant | Constant | Constant | Constant | Constant |
| Value | 3200 µs | 3200 µs | 3200 µs | 3200 µs | 3200 µs | 3200 µs |
|  |  |  |  |  |  |  |

**Step 3:** Click on Run simulation. Set the simulation time to default 100 seconds and click on OK.

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Number of Wireless Nodes | Generation Rate (Mbps) | Throughput (Mbps) | Delay (micro sec) |
| Switch | 6 | 15 | 15.00 | 466610.1 |
| Hub | 6 | 15 | 0.29 | 2864247.6 |

### Throughput comparison graph:

**HUB**

**SWITCH**

16

14

12

10

8

6

4

2

0

**Throughput (Mbps)**

## Inference:

Since hub is a broadcast domain, there are collisions during transmission. As per CSMA / CD algorithm there is node back off per the truncated binary back off algorithm. Due to collisions and backing off the throughput of a hub-based network is much lower than a switch-based network, while the delay experienced in much higher.

# Interfacing tail with NetSim

## Theory:

**What is a tail command?**

The **tail**command is a command-line utility for outputting the last part of files given to it via standard input. It writes results to standard output. By default tail returns the last ten lines of each file that it is given. It may also be used to follow a file in real-time and watch as new lines are written to it.

## Network Setup:

**PART 1:**

**Tail options**

* The following command is used to log the file

tail " path\_to\_file " -f

where -f option is used to watch a file for changes with the tail command pass the -f option. This will show the last ten lines of a file and will update when new lines are added. This is commonly used to watch log files in real-time. As new lines are written to the log the console will update will new lines.

* If users don’t want the last ten lines of the file, then use the following command

tail -n 0 " path\_to\_file " –f

where –n option is used to show the last n number of lines

* If you want to open more than 1 file then use the following command

tail –n 0 " path\_to\_file " " path\_to\_file " –f

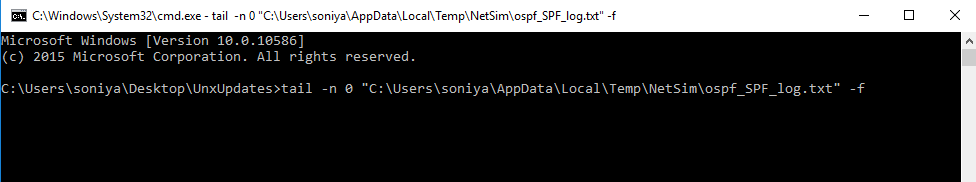
**PART 2:**

**Steps to log NetSim files using tail console**

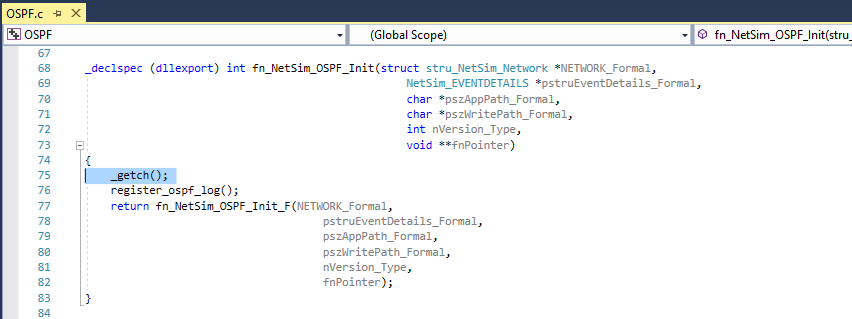
1. Open bin folder of NetSim’s current workspace path (<C:\Users\PC\Documents\NetSim\_12.0.16\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_12.0.16\_32\_std\_default\bin\bin\_x86> for 32-bit) which contains tail.exe
2. Open command window and change the directory to bin path of NetSim’s current workspace path (bin\bin\_x86 for 32-bit and bin\bin\_x64 for 64-bit)
3. Type the following command to open ospf\_log.txt file and press enter

tail -n 0 "C:\Users\PC\AppData\Local\Temp\NetSim\ospf\_SPF\_log.txt" –f

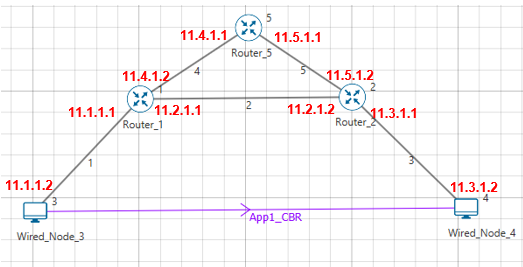
***Note:*** *Users need to change the path of the file. In this example we are using ospf\_log.txt file*



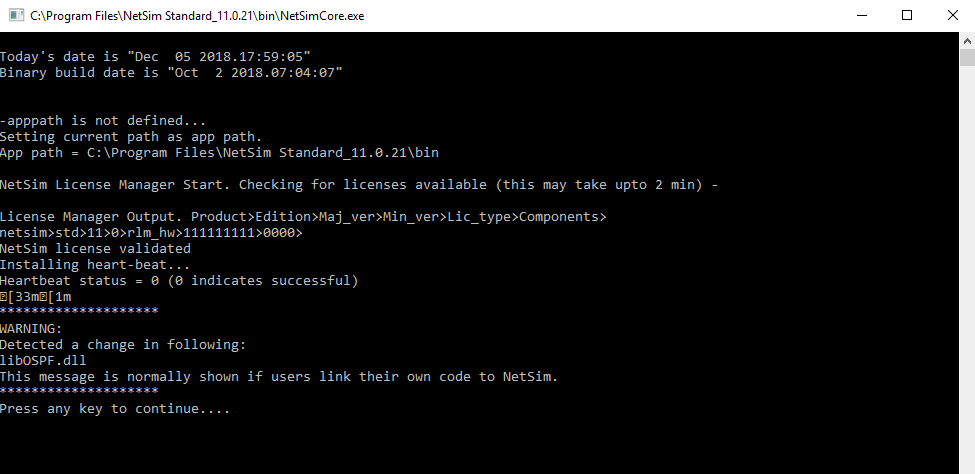
1. Open solution file and add the following line in fn\_NetSim\_OSPF\_Init() function in ospf.c file present inside OSPF project



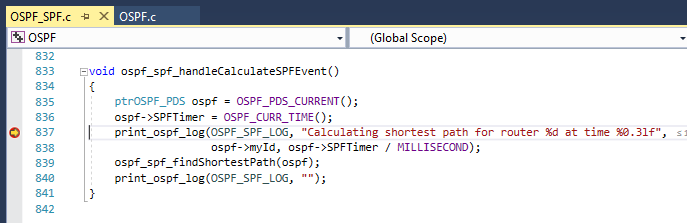
1. Rebuild the project
2. Upon rebuilding, **libOSPF.dll** will get created in the bin folder of NetSim’s current workspace path <C:\Users\PC\Documents\NetSim\_12.0.16\_64\_std\_default\bin\bin\_x64> for 64-bit and <C:\Users\PC\Documents\NetSim\_12.0.16\_32\_std\_default\bin\bin\_x86> for 32-bit.
3. Create a scenario in NetSim as per the screenshot below and run simulation



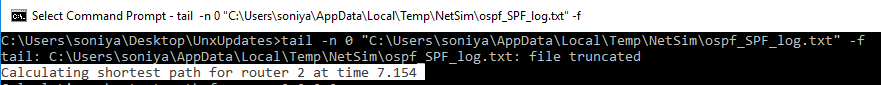
1. In the console window user would get a warning message shown in the below screenshot (because of changed DLL) and then the simulation will pause for user input (because of \_getch() added in the init function)



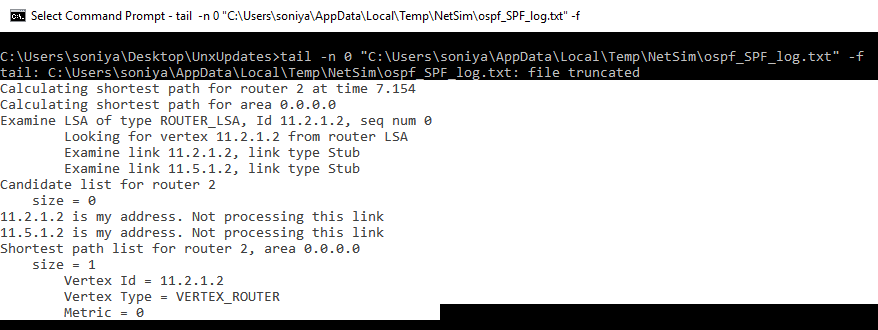
1. In Visual Studio, put break point inside all the functions in OSPF\_SPF.c file present inside OSPF project
2. Go to “Debug->Attach to Process” in Visual studio and attach to NetSimCore.exe.
3. Press enter in the command window. Then control goes to the project and stops at the break point in the source code as shown below



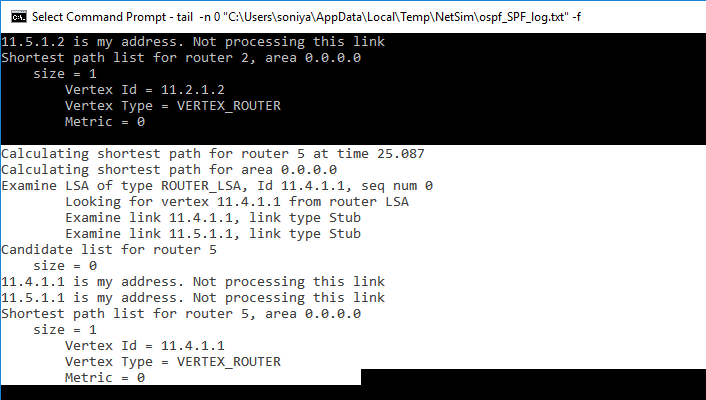
1. Press F5 and check the tail console to watch the ospf\_SPF log would look like the following screenshot which calculates the shortest path for Router2



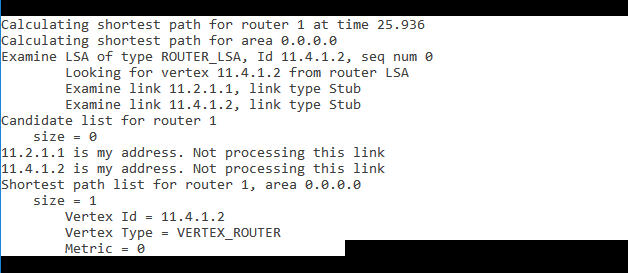
1. Keep pressing F5 will add the ospf\_SPF log to the tail console. The below screenshot examines the WAN links connected to Router2 i.e. 11.2.1.2 and 11.5.1.2



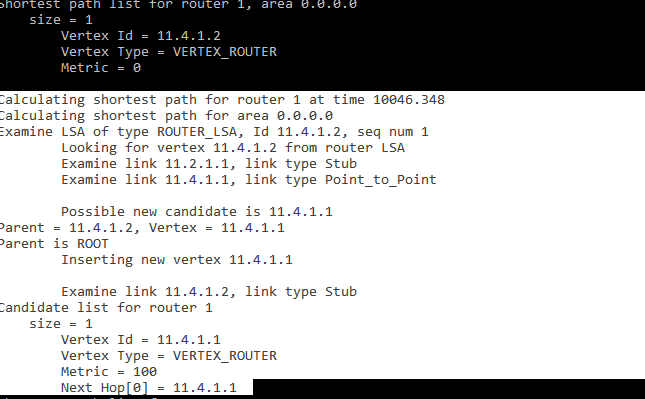
1. In the above screenshot, the shortest path for Router2 is 11.2.1.2 with Metrics 0 since it is one of the Router2’s interface
2. The below screenshot calculates the shortest path for Router5 and examines the WAN links connected to Router5 i.e. 11.4.1.1 and 11.5.1.1



1. In the above screenshot, the shortest path for Router5 is 11.4.1.1 with Metrics 0 since it is one of the Router5’s interface
2. The below screenshot calculates the shortest path for Router1 and examines the WAN links connected to Router1 i.e. 11.2.1.1 and 11.4.1.2



1. In the above screenshot, the shortest path for Router1 is 11.4.1.2 with Metrics 0 since it is one of the Router5’s interface
2. As shown in the below screenshot, the router1 calculates another new entry i.e. 11.4.1.1 with metrics 100 since it is the next hop (Router5’s 1st interface) connected to Router1



1. Similarly, users can debug the code and observe how the OSPF tables get filled
2. Users can also open multiple files by using the command given in section1

# **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.

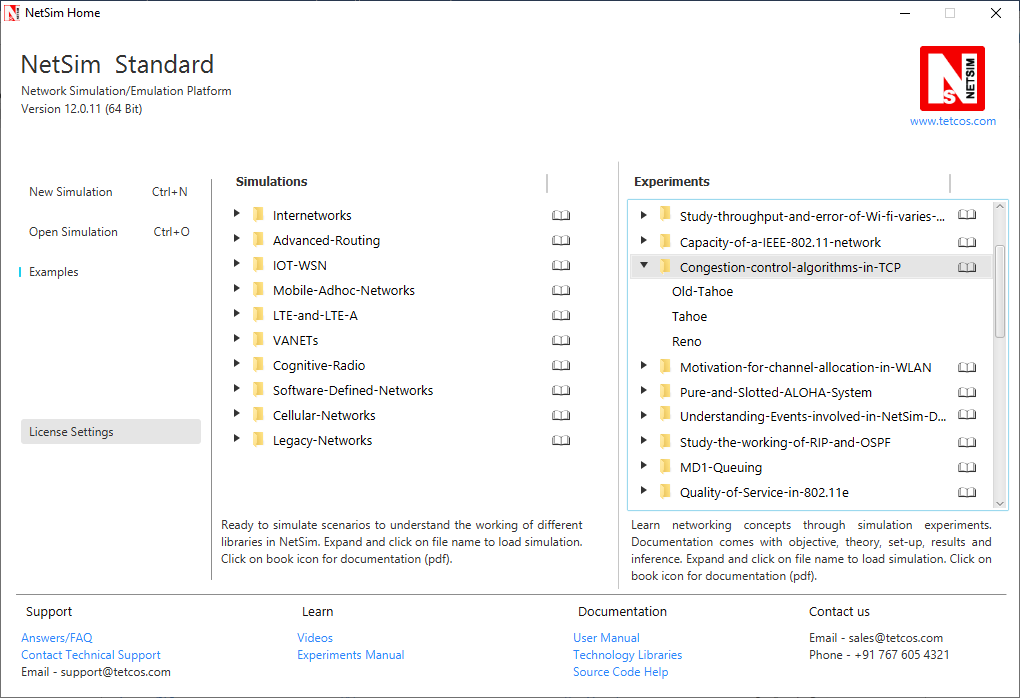
On the arrival of another duplicate ACK the congestion window increases by one MSS. This is done because an ACK signifies that a segment if out of the network and the sender can pump in another packet into the network. This is somewhat similar to slow start. 𝑐𝑤𝑛𝑑 += 𝑆𝑀𝑆𝑆.

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

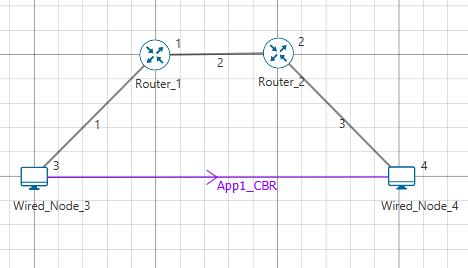
On receiving a higher ACK the congestion window is set to the threshold value. From now onwards congestion avoidance is followed. 𝑐𝑤𝑛𝑑 = 𝑠𝑠𝑡ℎ𝑟𝑒𝑠ℎ.

## 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.

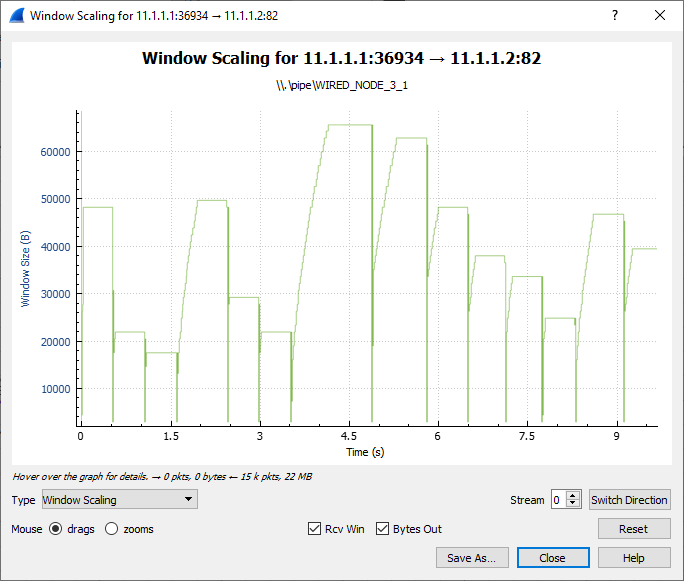
Go to the Wireshark Capture window.

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:**

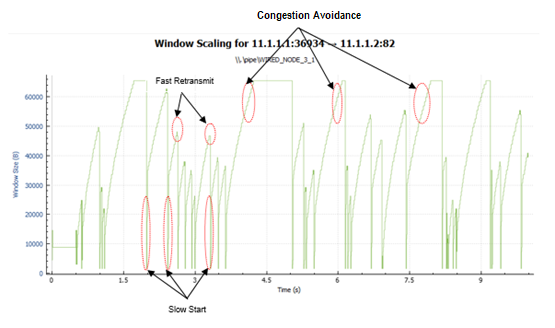


Slow Start

Congestion Avoidance

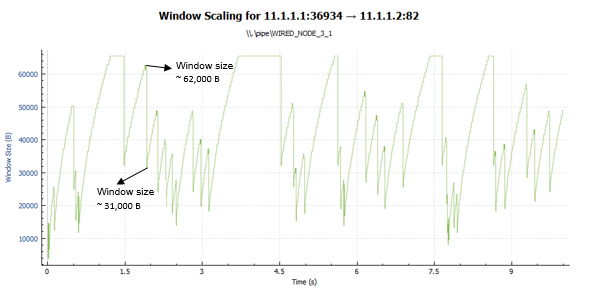
The graph shown above is a plot of Congestion window vs. Time. Each point on the graph represents the congestion window at the time when the packet is sent. You can observe that after every congestion window drop (caused by packet loss or timer expiry) Old Tahoe enters slow start, thereby increasing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

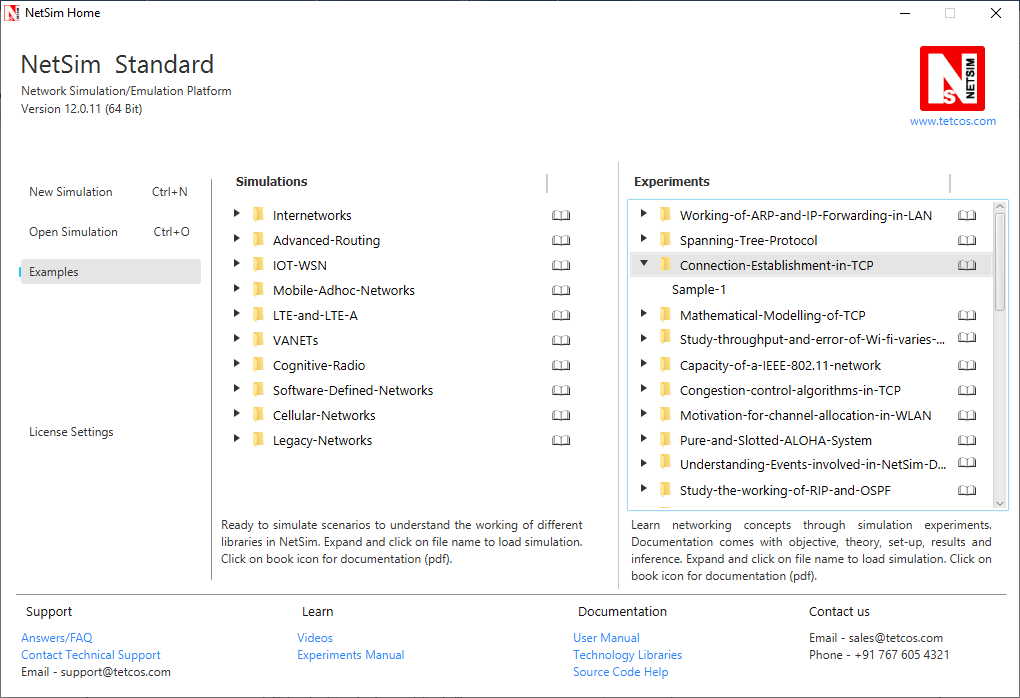
Fig: Basic 3-Way Handshake for Connection Synchronization

**Explanation:**

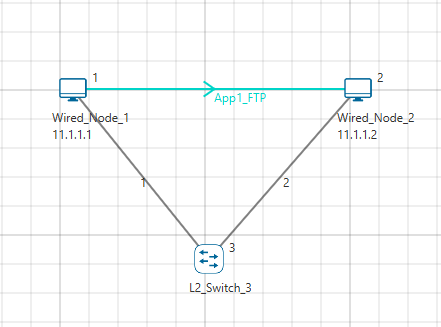
The above figure should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (🡪) indicates the departure of a TCP Segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (🡨 ) indicates the reverse. TCP states represent the state after the 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.

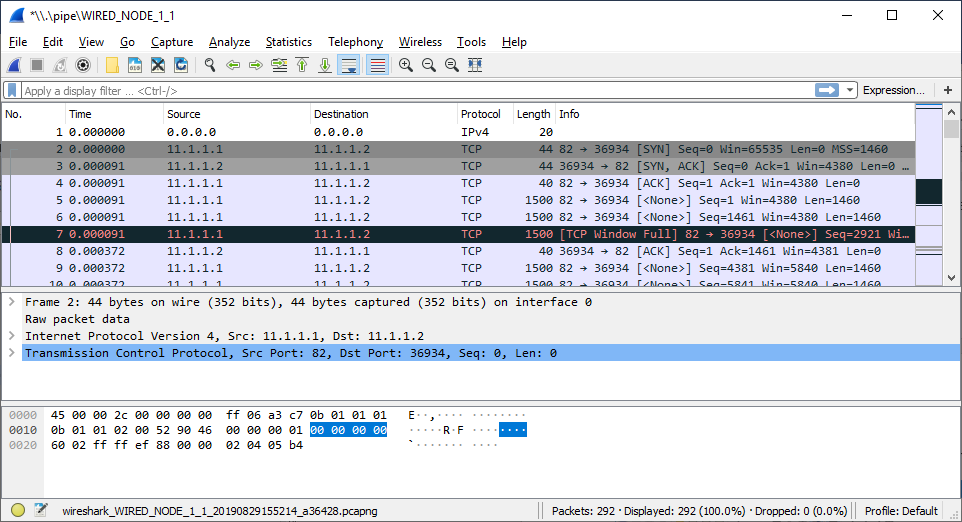
## Output:

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.

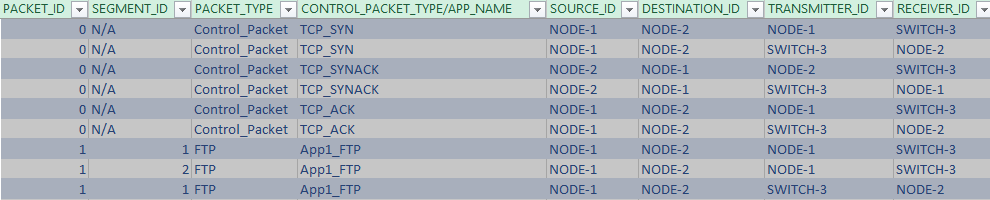
Once the connection is established, data transmission starts and we can see the data packets (of size 1500 bytes) sent from NODE-1 to NODE-2.

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.

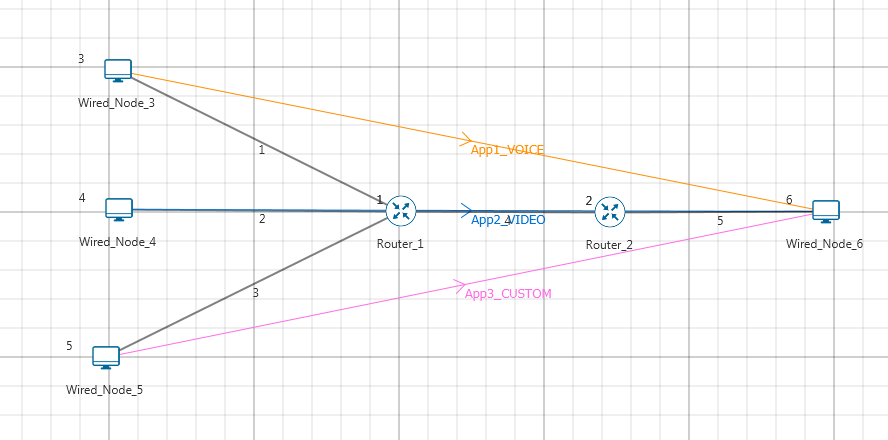
# Analyze the performance Round-Robin and WFQ Queuing Disciplines

## 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:

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



## Procedure:

**Sample 1: (Round-Robin)**

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 ROUND\_ROBIN. Similarly, Scheduling Type is set as ROUND\_ROBIN 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: (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)\* | Round-Robin-Sample-1 Throughput (Mbps) | WFQ-Sample-3 Throughput (Mbps) |
| Voice | 5 | 1.76  ~ (5 / 13.6) \*5 | 1.89 |
| Video | 2.6 | 0.90  ~ (2.6/13.6) \*5 | 0.85 |
| Custom | 6 | 2.12  ~ (6/13.6) \*5 | 2.02 |
| Total | 13.6 | 4.78  ~ 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 Round-Robin, 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.

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

# **Evaluate the performance of TCP and UDP protocol**

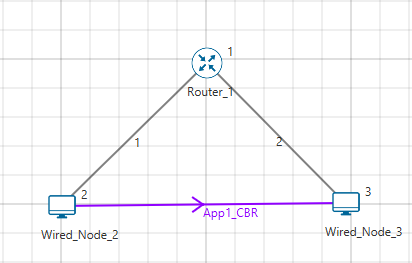
## Theory:

**TCP:** 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”.

**UDP:** The User Datagram Protocol (UDP) is simplest Transport Layer communication protocol available of the TCP/IP protocol suite. It involves minimum amount of communication mechanism. UDP is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism. In UDP, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This shortcoming makes this protocol unreliable as well as easier on processing.

## Network Set Up:

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



## Procedure:

**TCP:**

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 Routers in the **“Internetworks”** Network Library.

**Step 2:** The Link Properties are set according to the table given below:

|  |  |  |
| --- | --- | --- |
| Link Properties | Wired Link 1 | Wired Link 2 |
| Uplink Speed (Mbps) | 100 | 100 |
| Downlink Speed (Mbps) | 100 | 100 |
| Uplink propagation delay (µs) | 5 | 5 |
| Downlink propagation delay (µs) | 5 | 5 |
| Uplink BER | 0.00001 | 0.00001 |
| Downlink BER | 0.00001 | 0.00001 |

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

**Step 5:** Click on Run simulation. The simulation time is set to 100 seconds.

**UDP:**

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

**Step 1:** In the Source Node, i.e. Wired Node 2, in the TRANSPORT LAYER Properties, uncheck TCP checkbox for disabling TCP.

**Step 2:** Click on Run simulation. The simulation time is set to 100 seconds.

## Output

**Comparison Table:**

|  |  |
| --- | --- |
|  | Throughput (Mbps) |
| TCP | 0.453 |
| UDP | 0.005 |

# **Evaluate the performance of AODV and DSR routing protocol**

## Theory:

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

**Using Link-Layer Acknowledgements**

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

**Using Network-Layer Acknowledgements**

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

While simulating certain network configurations, users may see that packets received are more than packets sent. This is because:

* This is being measured as part of our UDP protocol metrics in layer 4 in the source and in the destination
* Let us say UDP protocol at source node A sends a datagram. At the MAC - WLAN send the frame and starts a re-transmission timer.
* If no Ack is received within this timer period it would initiate a re-transmission (consider cases where the WLAN Ack has a collision or is errored)
* As the destination the MAC (WLAN) layer would send up to UDP both the first packet it received and the re-transmitted packet it received.

UDP protocol in the destination would count both the packets received

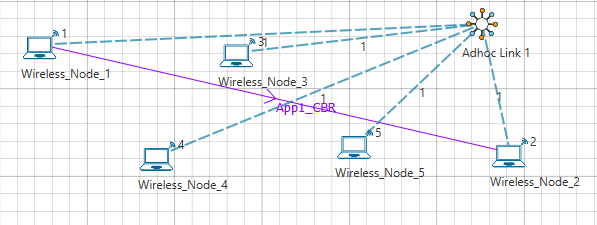
**AODV:** Adhoc On Demand Distance Vector Routing (AODV) protocol defines 3 message types:

* **Route Requests (RREQs**) - RREQ messages are used to initiate the route finding process.
* **Route Replies (RREPs)** - RREP messages are used to finalize the routes
* **Route Errors (RERRs)** - RERR messages are used to notify the network of a link breakage in an active route.

The route discovery process involves ROUTE REQUEST (RREQ) and ROUTE REPLY (RREP) packets. The source node initiates the route discovery process using RREQ packets. The generated route request is forwarded to the neighbors of the source node and this process is repeated till it reaches the destination. On receiving a RREQ packet, an intermediate node with route to destination or the destination node generates a RREP containing the number of hops required to reach the destination. All intermediate nodes that participates in relaying this reply to the source node creates a forward route to destination.

RERR message processing is initiated when: – Node detects a link break for the next hop of an active route, or receives a data packet destined for a node for which it has no (active) route.

## Network Set Up:



## Procedure:

**DSR:**

**Settings done in the Network:**

**Step 1:** Grid length: 500m\*500m

**Step 2:** Disable TCP in all nodes and disable mobility in all nodes

**Step 3:** Configure CBR application with default properties

**Step 4:** Channel characteristics: Path loss only, Path loss model: Log Distance, Path loss exponent:4

**Step 5:** Set DSR routing protocol and Network Layer ACK as ACK Type under network layer properties in all nodes

**Step 6:** Enable Packet Trace and run simulation for 10 seconds

**AODV:**

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

**Step 1:** Set AODV routing protocol under network layer properties in all nodes

**Step 2:** Click on Run simulation. The simulation time is set to 10 seconds.

## Output

**Comparison Table:**

|  |  |
| --- | --- |
|  | Throughput (Mbps) |
| DSR | 0.077 |
| AODV | 0.264 |

# **Simulate and Analysis of ICMP and IGMP packets**

## Introduction

**About Multicasting**

Multicasting is a data delivery method where one sender sends data to thousands of recipients across a routed network. Multicasting is controlled-broadcasting; the sender transmits data to specific recipients only.

With IP multicasting, a host sends packets to a multicast group of hosts anywhere within the IP network by using a special form of IP address called the IP multicast group address. A multicast group is made of an arbitrary number of hosts who join a group to receive packets from the source. To ensure that a host receives data, the host must join the multicast group to which the sender is sending data.

***Note: You can configure and simulate multicast routing protocols such as IGMP and PIM, only if you have licenses for component 3 (advanced routing).***

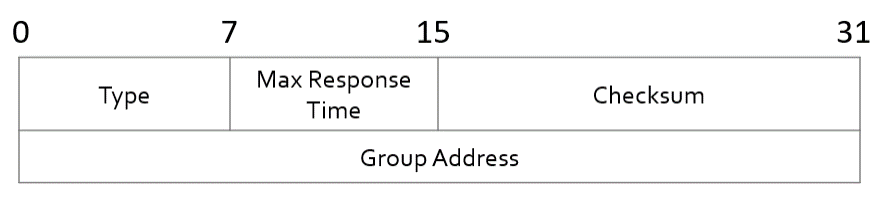
**About IGMP**

The Internet Group Management Protocol (IGMP) is a communication protocol that hosts and adjacent multicast routers on IPv4 networks use, to establish and manage the membership of hosts and routing devices in multicast groups. Hosts and multicast routers use IGMP as follows:

* The hosts use IGMP to report their multicast group memberships to neighboring multicast routers.
* The multicast routers use IGMP to know the members in multicast groups, for every physical network the multicast router is connected.

The multicast routers maintain a list of multicast group memberships for every network to which the multicast routers are connected, and a timer for each membership.

The messages that IGMP uses are encapsulated in IP datagrams, with an IP protocol number of 2. All IGMP messages are sent with an IP TTL of 1 and contain the IP Router Alert option in their IP header. All IGMP messages sent between a host and the multicast router use the following format:



## Network Set Up:

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.

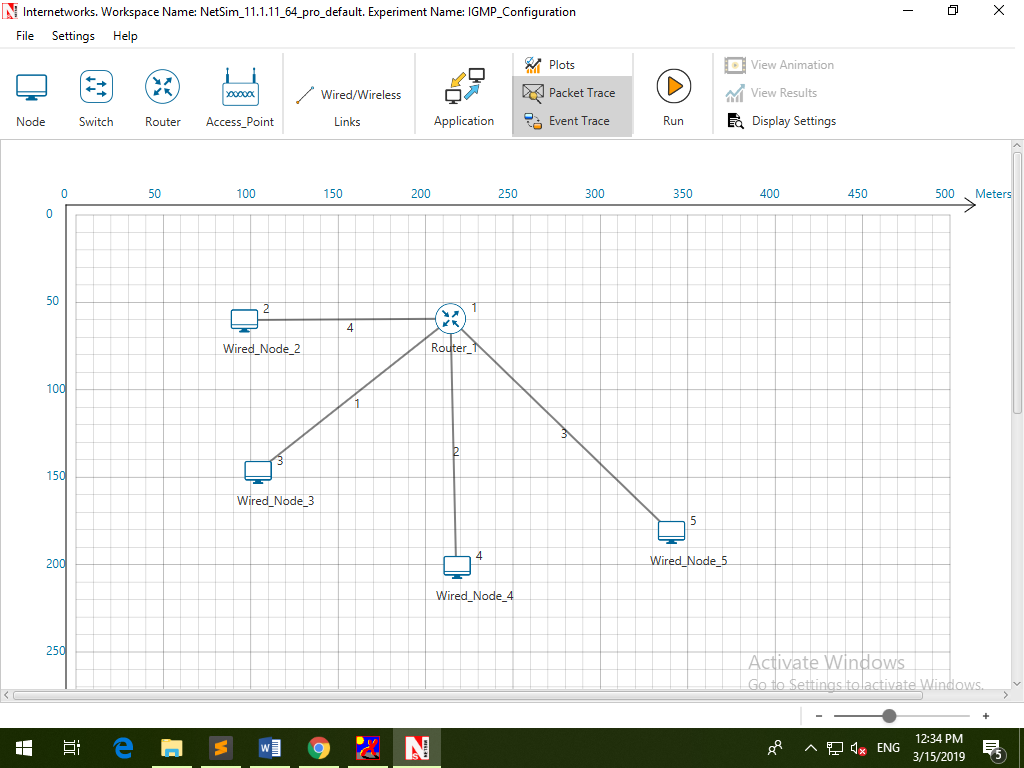
NetSim uses the following defaults for IGMP simulations:

* 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

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



## Procedure

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**.

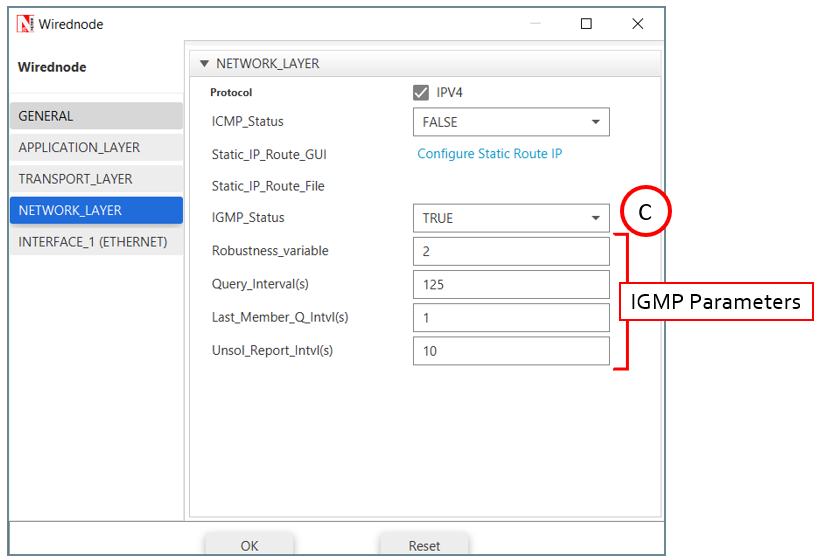
The Wired node pop-up window appears.

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.

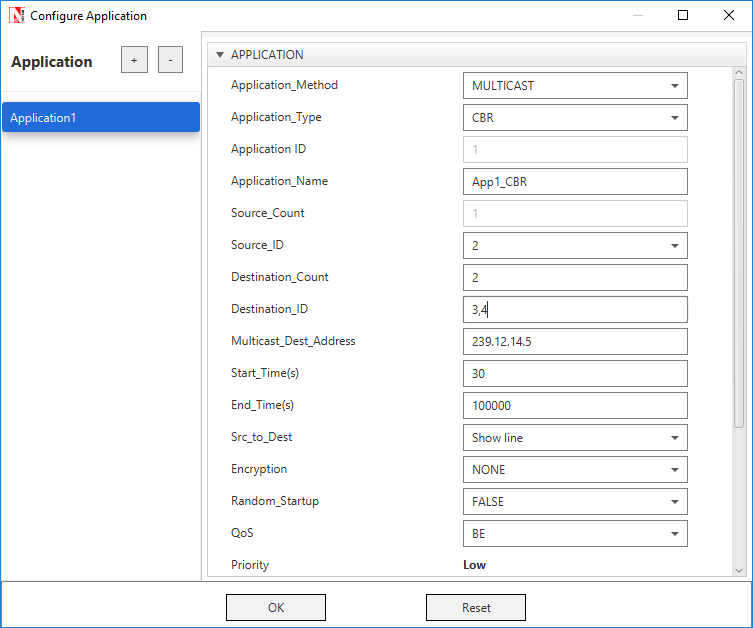
You can enter a value between 1 and 10,000 seconds.

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.

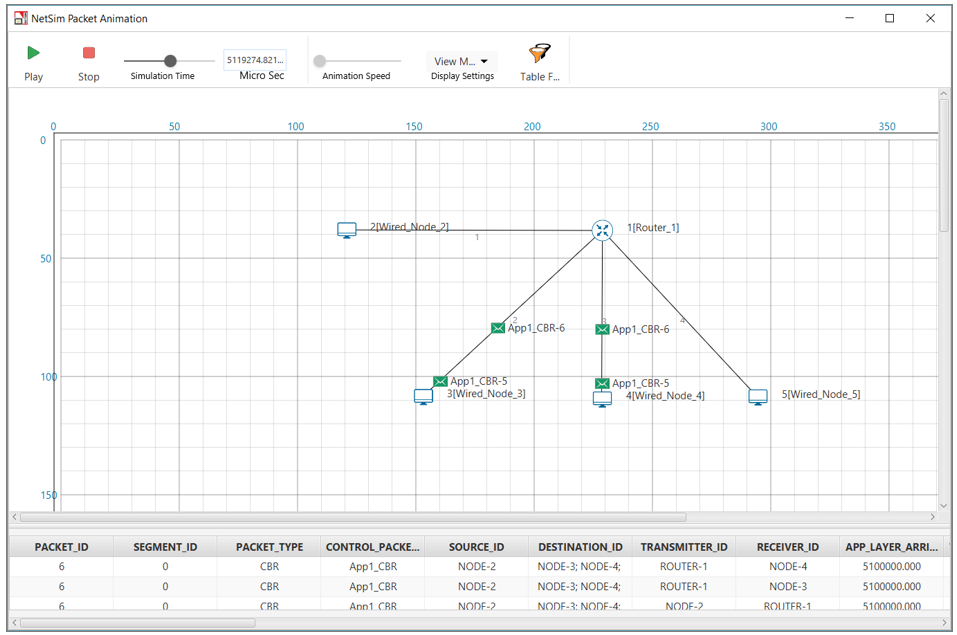
You will see the following happen in the animation:

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.

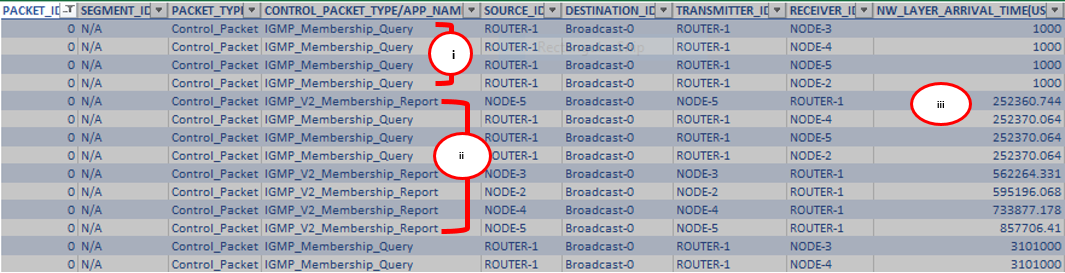
A *.CSV* appears.

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).

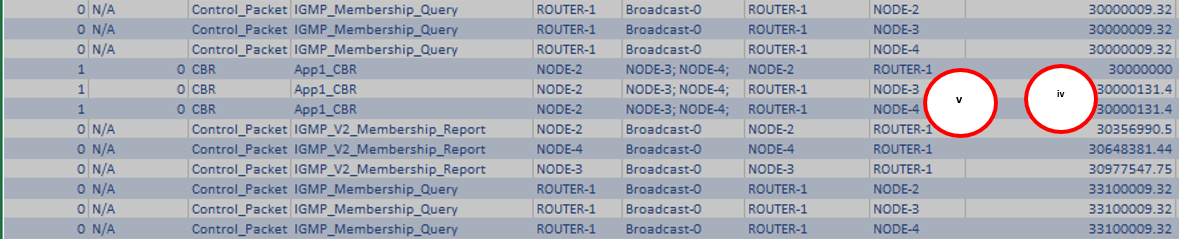


1. The multicast application Wired\_Node\_2 starts to send multicast traffic to Wired\_Node\_3 and Wired\_Node\_4 only after 5 seconds in to the simulation.

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

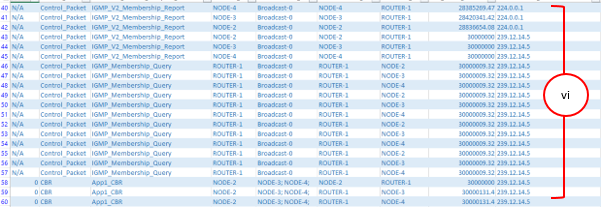
1. Wired\_Node\_2 multicasts Constant Bit Rate (CBR) packets only to Wired\_Node\_3 and Wired\_Node\_4.

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.

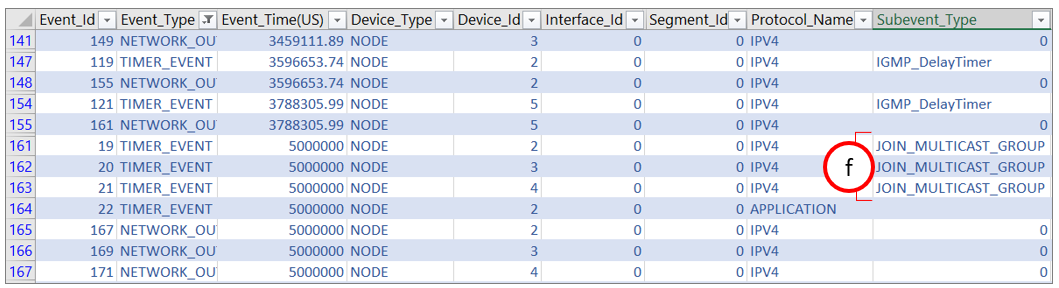
1. Open the .*CSV* file and filter the Event\_Type column by **NETWORK\_OUT** and **TIMER\_EVENT**.

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.



## ICMP:

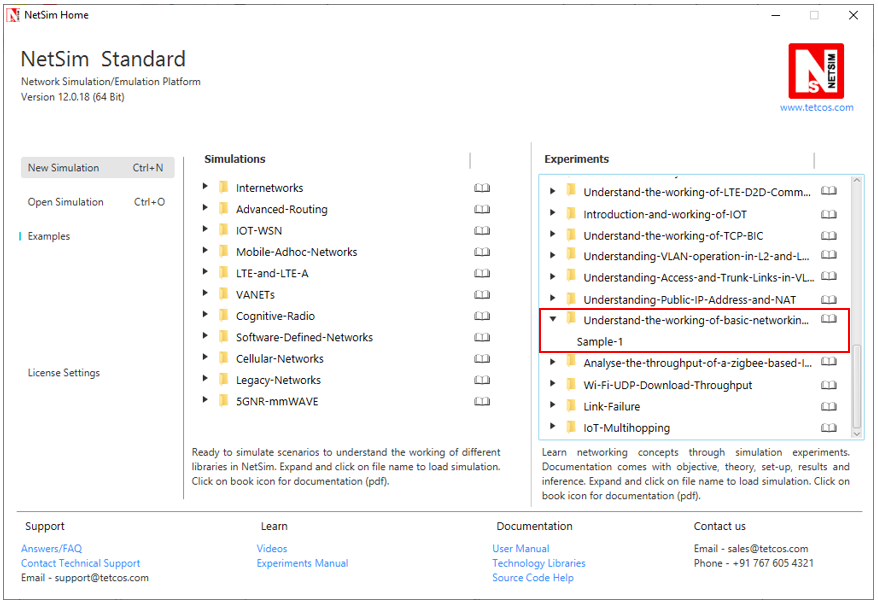
Since IP does not have an inbuilt mechanism for sending error and control messages. It depends on **Internet Control Message Protocol (ICMP)** to provide an error control. It is used for reporting errors and management queries. It is a supporting protocol and used by networks devices like routers for sending the error messages and operations information. Ping operates by sending ICMP echo request packets to the target host and waiting for an ICMP echo reply.

**About Ping Command:**

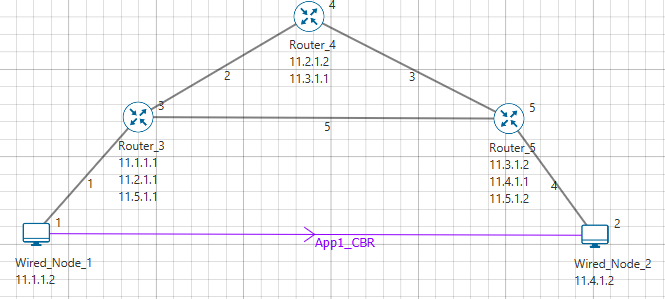
* The ping command is one of the most often used networking utilities for troubleshooting network problems
* You can use the ping command to test the availability of a networking device (usually a computer) on a network
* When you ping a device, you send that device a short message, which it then sends 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.

## 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:



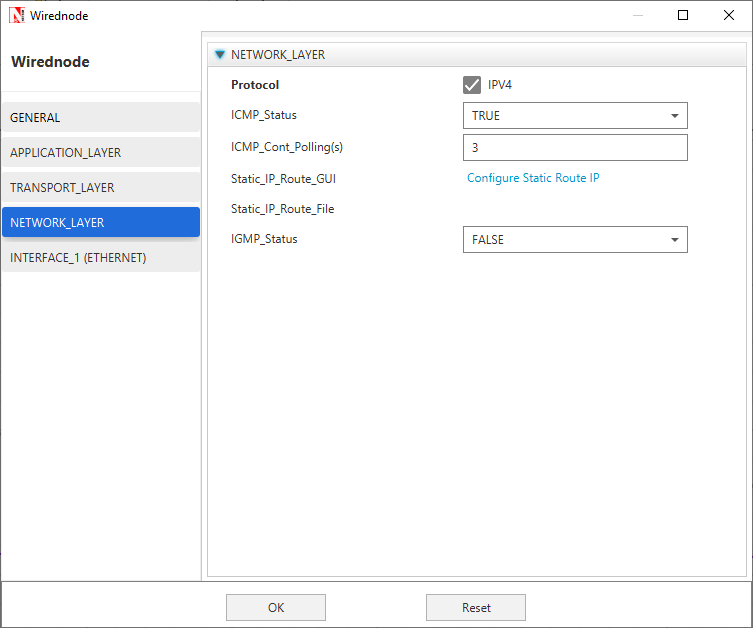
## Procedure:

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.

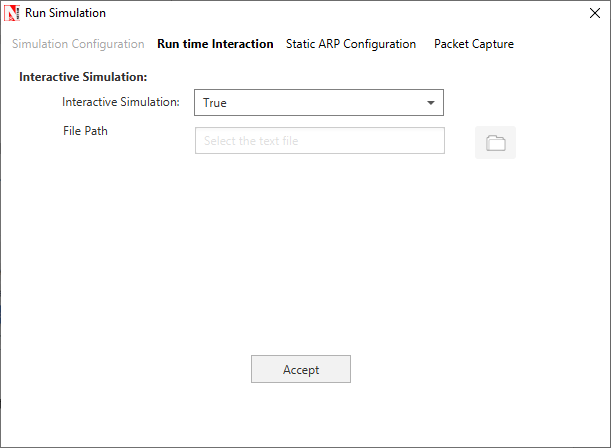
**Step 5:** Right click on the Application Flow **App1 CBR** and select Properties or click on the Application icon present in the top ribbon/toolbar.

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.

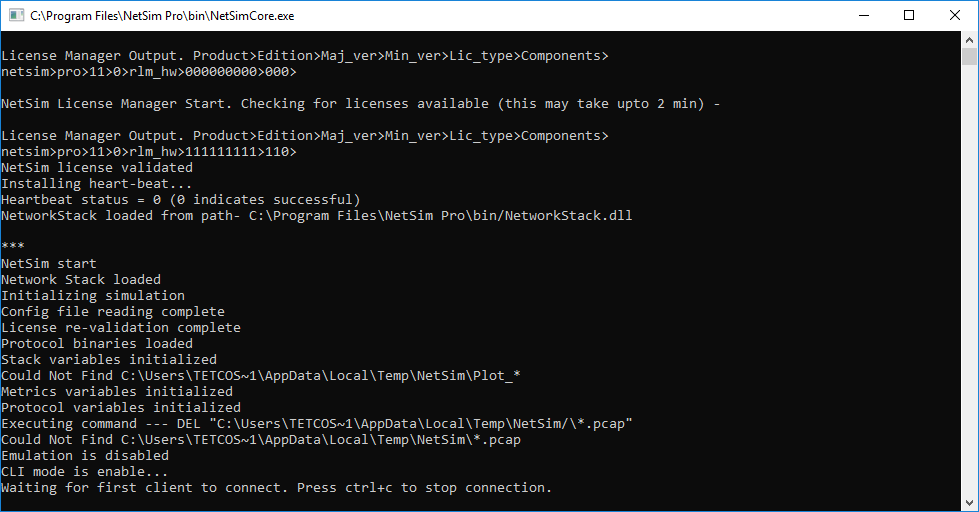
**Step 7:** Click on Run Simulation. Simulation Time is set to 300 Seconds and in the **Runtime Interaction** tab, Interactive Simulation is set to True.



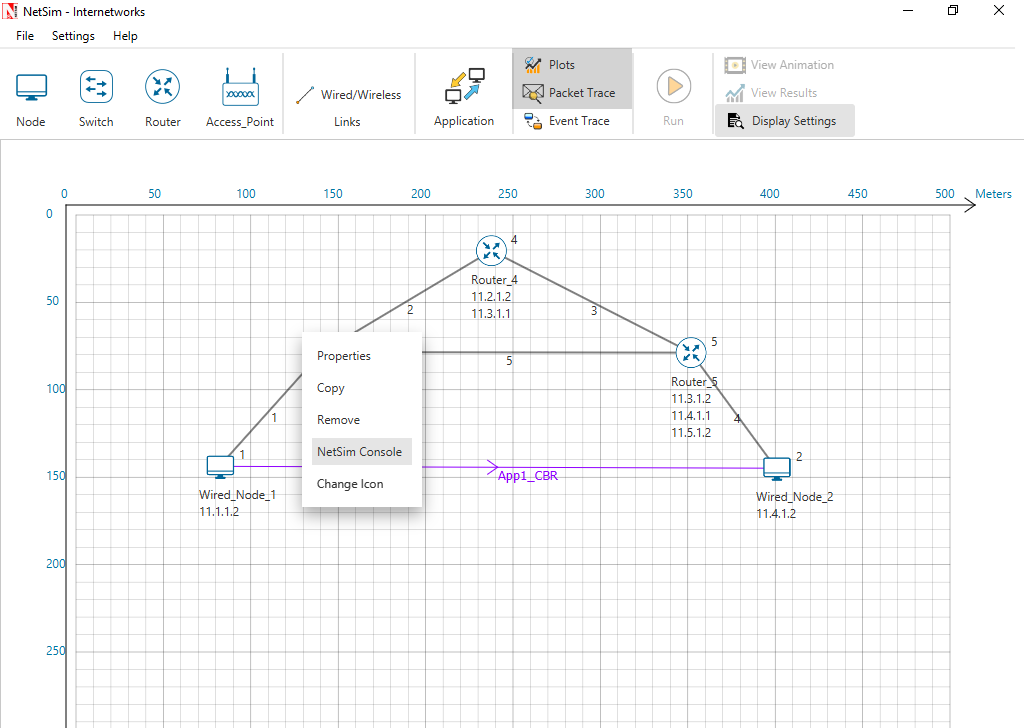
***NOTE:******It is recommended to specify a longer simulation time to ensure that there is sufficient time for the user to execute the various commands and see the effect of that before the Simulation ends.***

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

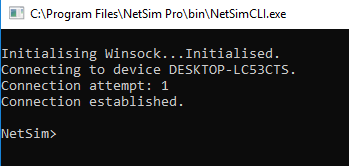
* Simulation (NetSimCore.exe) will start running and will display a message **“waiting for first client to connect”** as shown below:



* 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.

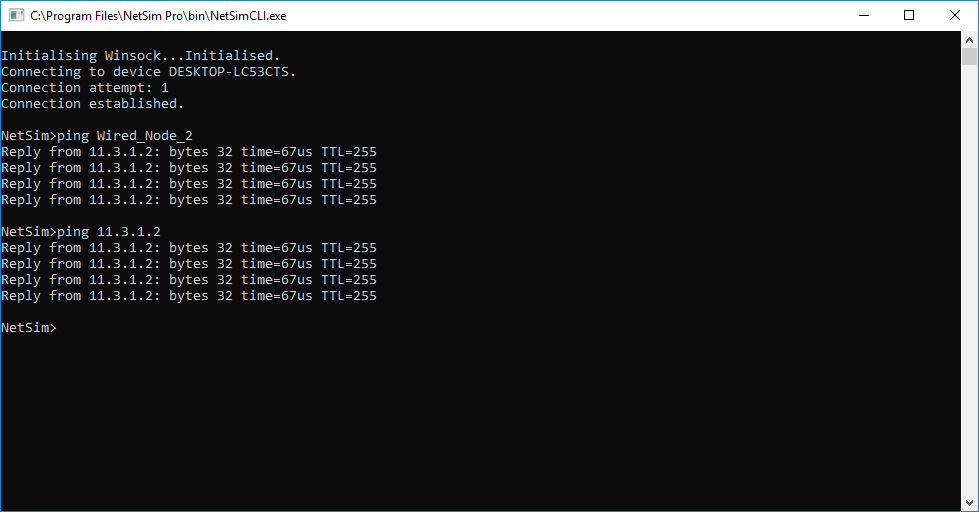
## Network Commands

**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**

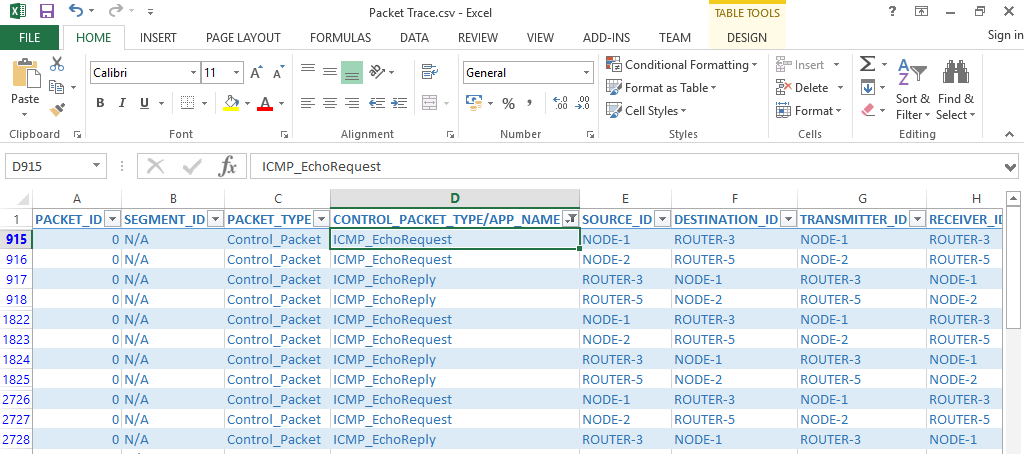
**Ping <Node Name> e.g. ping Wired\_Node\_2**



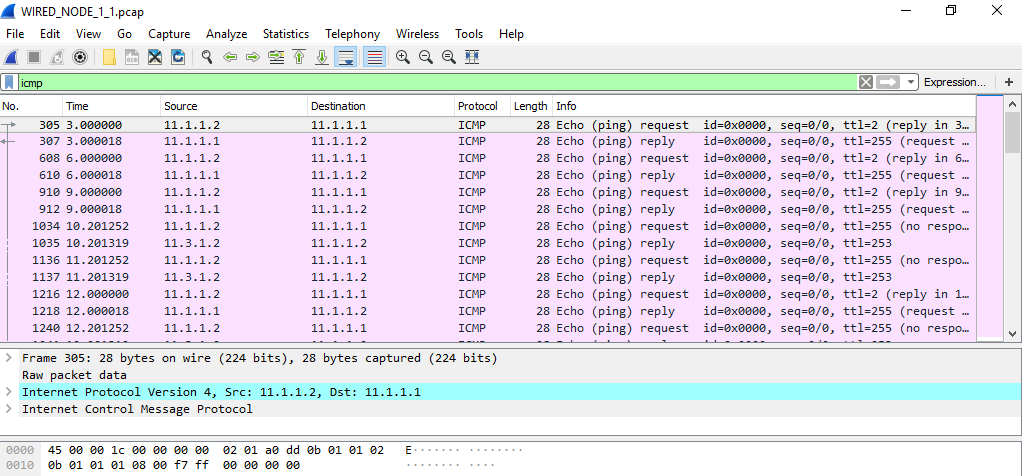
**Ping Command Results:**

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.



# CDMA Downlink

## Theory:

A Cellular Network (also known as a mobile network) is a communication network where the last link is wireless. The Cellular Network is distributed over land areas called cells.

Every cell contains at least one fixed-location transceiver known as a base station. The cells altogether provide Cellular Network coverage over larger geographical areas.

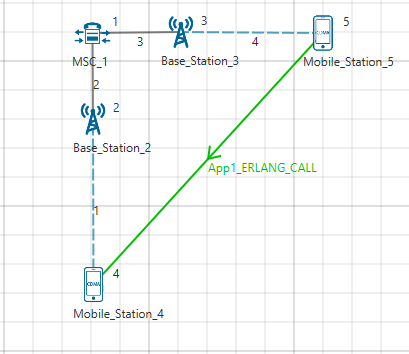
In a Cellular Network, handover is a process of transferring an ongoing call or data session from one channel to another channel, where both are connected to the same core network. This way, user equipment such as mobile phones, can communicate even if the user is moving across different cells.

Cellular Networks use non-IP protocols and run standalone. So, you cannot connect Cellular Networks to Internetworks.

*Note: NetSim Cellular Networks library contains two such Cellular Network standards: Global System for Mobile communication (GSM) and Code-Division Multiple Access (CDMA).*

## Network Set Up:

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 Base station, 2 Mobile station, and 1 MSC in the **“CDMA”** Network Library.

**Step 2:** Create ERLANG\_CALL application from Mobile\_Station\_5 to Mobile\_Station\_4. Set generation rate to

**Step 2:** Click on Run simulation. The simulation time is set to 100 seconds.

## Output

|  |  |
| --- | --- |
|  | Throughput (Mbps) |
| Downlink | 0.026 |

# Evaluate the performance of IEEE802.11 and IEEE802.15.4

## Theory:

**IEEE802.11 -** Performance of a WiFi link and system depends on signal quality, interference and communication overheads.

1. **Signal quality** function of transmit power, path loss, shadowing, etc poor signal quality leads to packet errors and losses mechanisms in WiFi - rate adaptation, dynamic association
2. **Interference** both from WiFi and non-WiFi radios interference leads to poor channel utilization, packet errors and losses mechanisms in WiFi - random access, channel selection
3. **Communication overheads** at PHY, MAC and higher layers overheads reduce effective utilization and throughput mechanisms in WiFi - packet aggregation, MU-MIMO

**IEEE802.15.4 - Internet of Things (IoT)** is an ecosystem of devices that connect to and communicate via the internet. A typical IOT deployment consists of

1. Embedded devices / sensors
2. Communication over an IP network (between the devices and to/from cloud servers)
3. Cloud services, Big Data, Analytics / Machine learning on the cloud

NetSim can be used to simulate the communication network. The sensors used in NetSim are abstract, which means that they could be any kind of sensor or embedded device. Any data transmitted by these devices have to be in the form of ‘IP Packets’. NetSim simulates the transmission of these IP packets. It does not focus on ‘what is the application payload’ being sent and is not concerned with data storage & analytics of this payload.

NetSim’sInternet of Things (IoT) and Wireless Sensor Network (WSN)libraries stack comprises of

1. Application Layer: Sensor App as well as applications such as Voice, Video, CBR etc.
2. Transport Layer: UDP
3. Network layer: AODV and RPL
4. MAC and PHY layers: 802.15.4 Zigbee

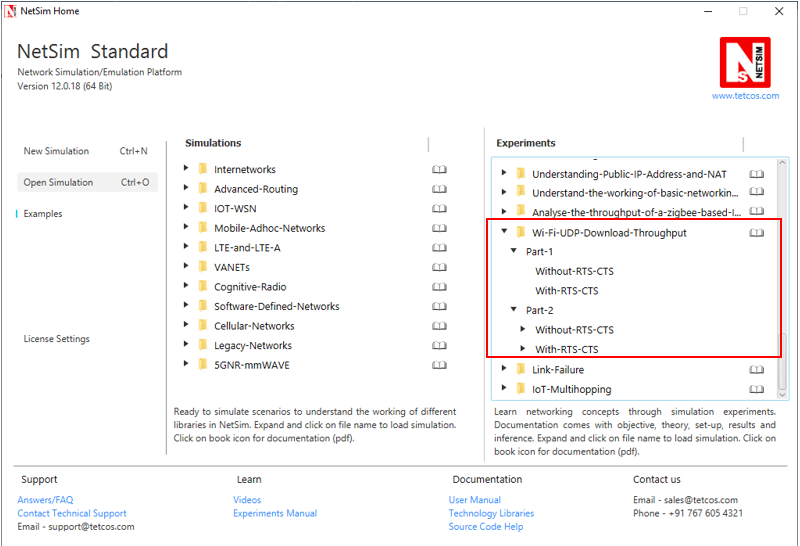
NetSim models IoT as a WSN that connects to an Internetwork through a 6LowPAN Gateway. The 6LowPAN Gateway uses two interfaces: a Zigbee (802.15.4) interface and a WAN Interface. This WSN sends data to a 6LowPAN Gateway. The Zigbee interface allows wireless connectivity to the WSN while the WAN interface connects to an Internetnetwork.

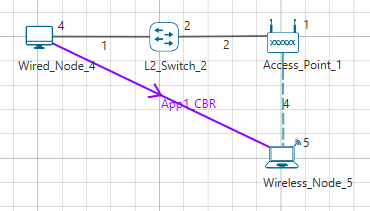
IEEE 802.15.4 uses either Beacon Enabled or Disabled Mode for packet transmission. In Beacon Enabled Mode, nodes use slotted CSMA/CA algorithm for transmitting packets else they use Unslotted CSMA/CA.

## IEEE 802.11

### Network Set Up:

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

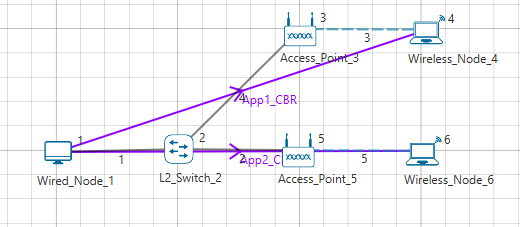
The following changes in settings are done from the previous sample:

**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.

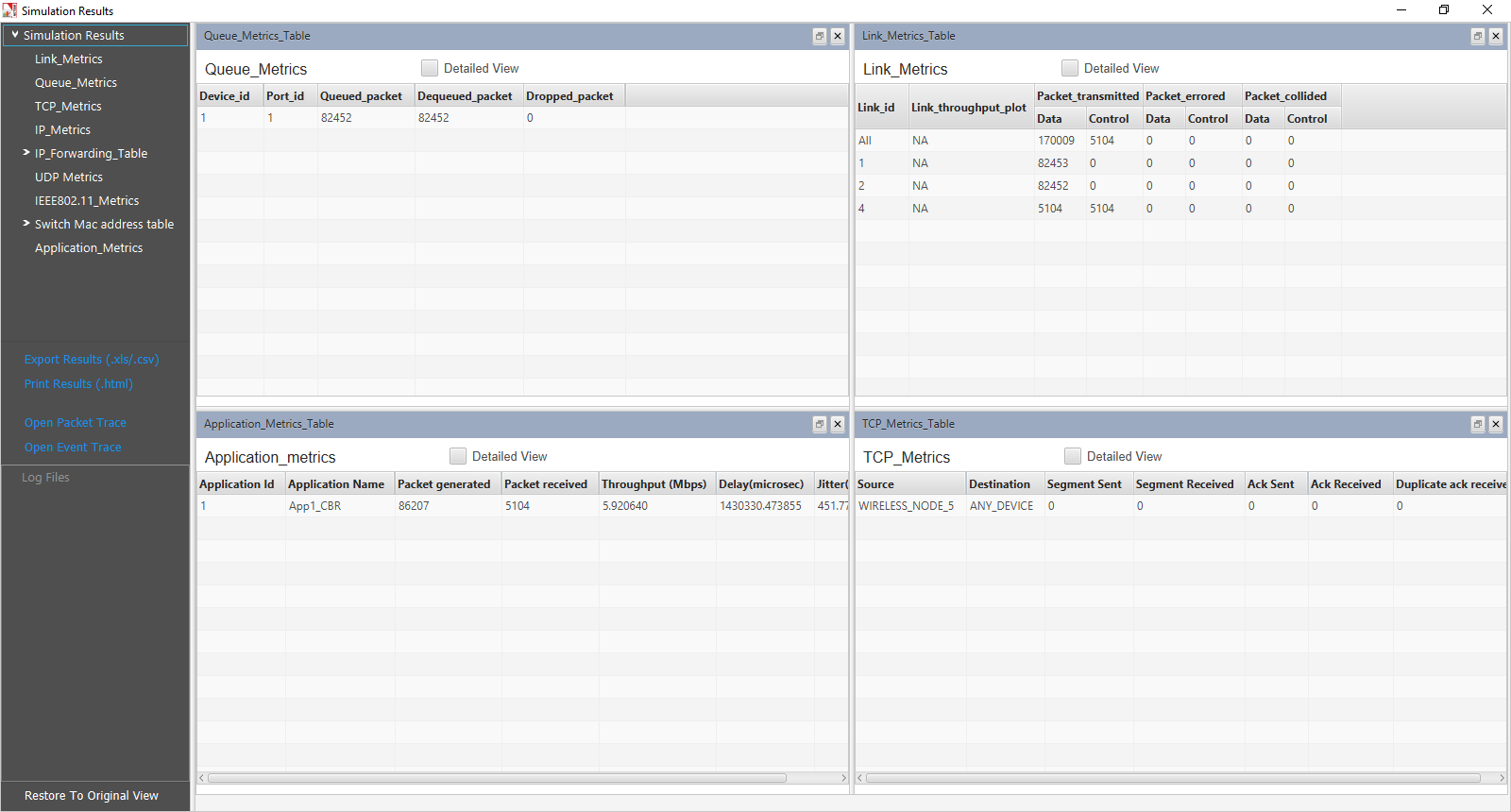
It will automatically be set for Wireless Node 6, since the above parameter is Global.

**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:

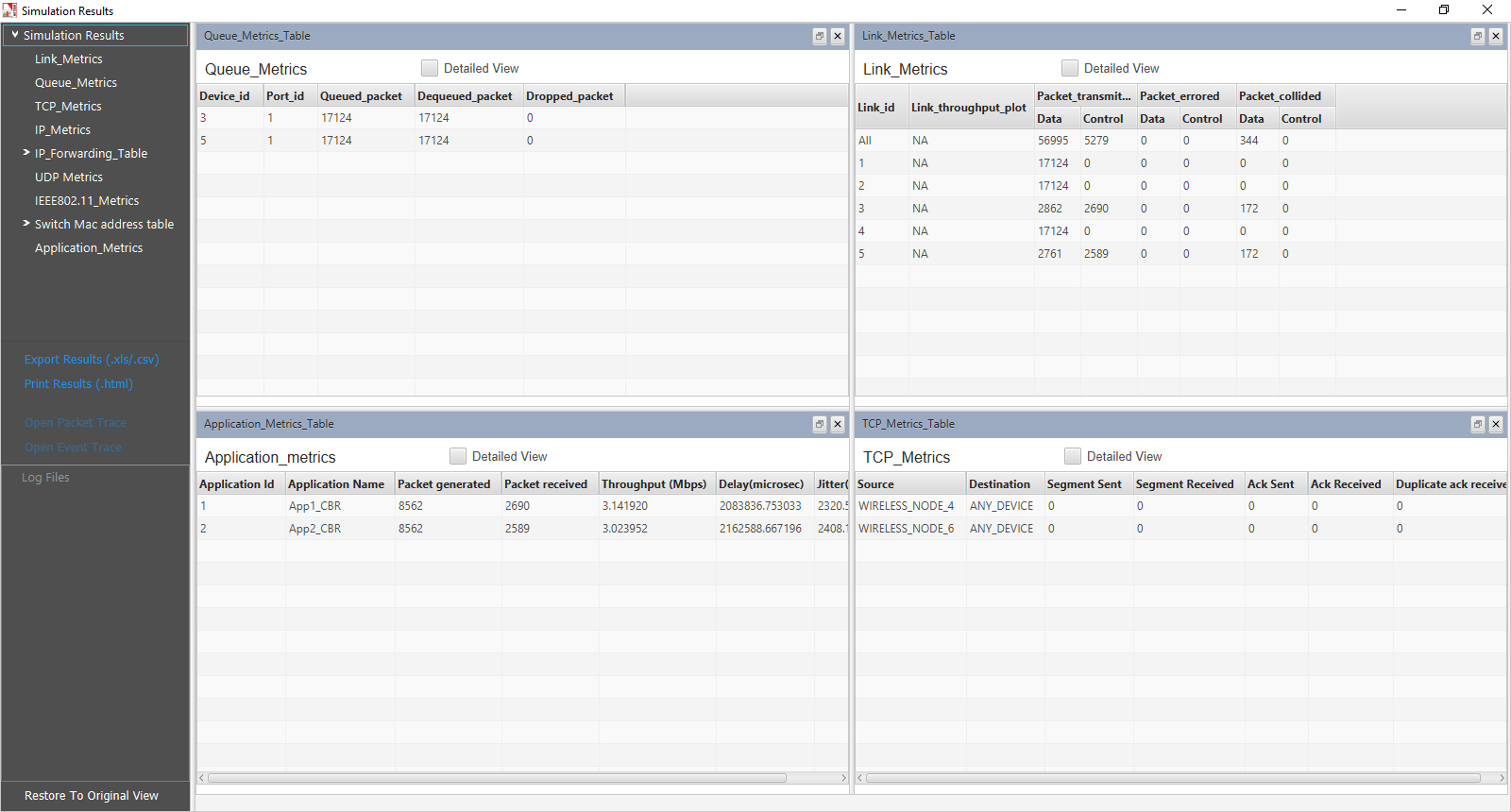
After running simulation, check throughput in Application metrics as shown in the below screenshot:



|  |  |  |
| --- | --- | --- |
| 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:

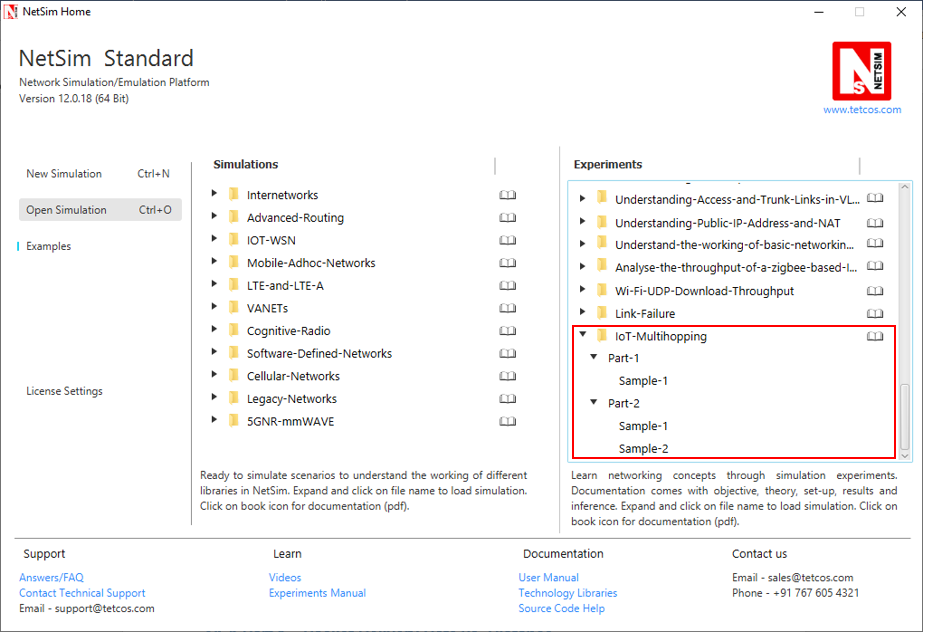


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sample | Throughput (Mbps) with 2 APs | Throughput (Mbps) with 3 APs | Throughput (Mbps) with 4 APs | Throughput (Mbps) with 5 APs |
| 1 (Without RTS/CTS) | App 1: 3.14  App 2: 3.02  **Total: 6.16** | App 1: 2.12  App 2: 1.97  App 3: 2.09  **Total: 6.18** | 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** |

## IEEE802.15.4

### Network Setup:

Open NetSim and click **Examples > Experiments > IoT-Multihopping** **> Part-1 > Sample-1** as shown below:

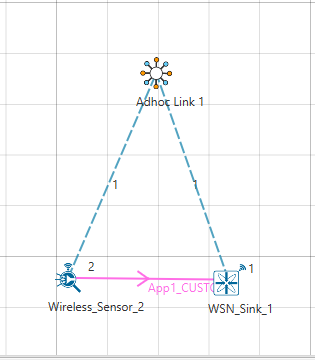


### Part 1 – Packet Delivery Rate vs. Distance

In this part, we perform a simulation to understand, **“How the distance between the source and sink impacts the received signal strength (at the destination) and in turn the packet error rate?”** We will assume a well-established path-loss model under which, as the distance varies, the received signal strength (in dBm) varies linearly. For a given transmit power (say 0dBm), at a certain reference distance (say 1m) the received power is dBm, and decreases beyond this point as for a transmitter-receiver distance of . This is called a *power-law* path loss model, since in mW the power decreases as the power of the distance . The value of is 2 for free space path loss and varies from 2 to 5 in the case of outdoor or indoor propagation. Values of are obtained by carrying out experimental propagation studies.

**Sample 1:**

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 the NetSim GUI comprising of a WSN Sink and 1 Wireless Sensor in **Wireless Sensor Networks**.

***Note:*** *NetSim currently supports a maximum of only one device as WSN Sink.*

**Step 2:** Before we actually designed this network, in the Fast Config Window containing inputs for **Grid Settings and Sensor Placement**, the Grid Length and Side Length were set to 500 meters respectively, instead of the default 50 meters and we have chosen **Manually Via Click and Drop** option.

**Step 3:** The distance between the WSN Sink and Wireless Sensor is 5 meters.

***Note:*** *By default, TCP is disabled in all the devices.*

**Step 4:** Go to Network Layer properties of Wireless Sensor 2, the Routing Protocol is set as **AODV**.

***Note:*** *The Routing Protocol parameter is Global. i.e. It will automatically be set to AODV in WSN Sink.*

**Step 5:** In the Interface Zigbee > Data Link Layer of Wireless Sensor 2, **Ack Request** is set to Enable and **Max Frame Retries** is set to 4. Similarly, it is set for WSN Sink 1.

**Step 6:** In the Interface Zigbee > Physical Layer of Wireless Sensor 2, **Transmitter Power** is set to 1mW, **Reference Distance** is set to 1m, **Receiver Sensitivity** is set to -105dBm, and **ED Threshold** is set to -115dBm.

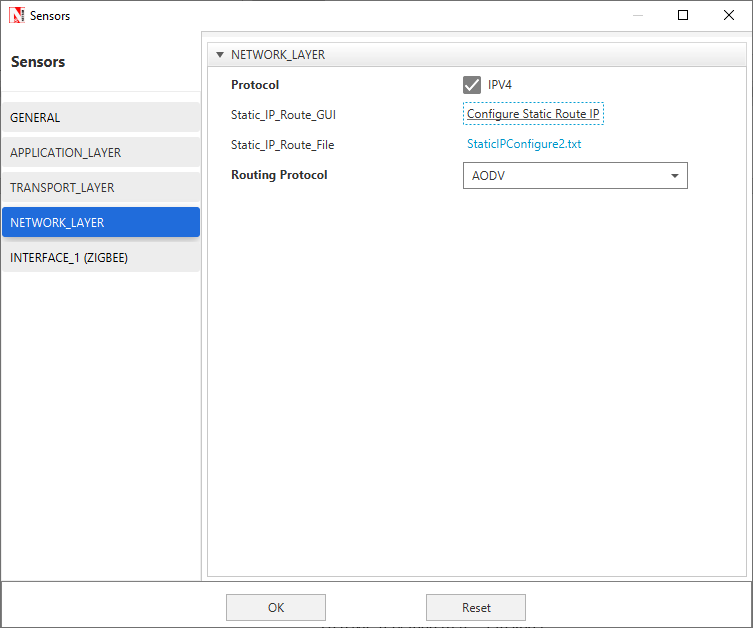
**Step 7:** Right click on the Application Flow **App1 CUSTOM** and select Properties or click on the Application icon present in the top ribbon/toolbar.

A CUSTOM Application is generated from Wireless Sensor 2 i.e. Source to WSN Sink 1 i.e. Destination with Packet Size set to 70 Bytes and Inter Arrival Time set to 4000 µs.

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 8:** The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2, Click on **Configure Static Route IP**.

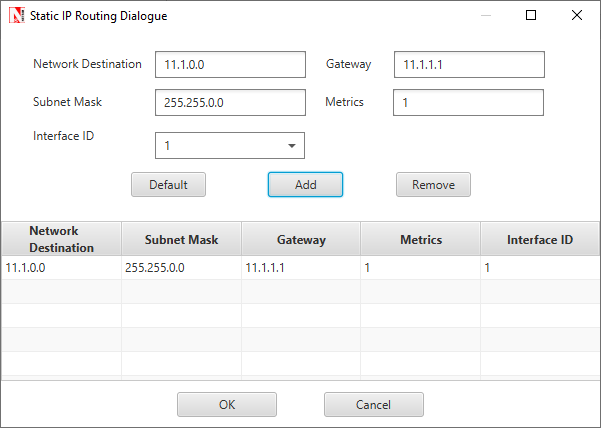


Static IP Routing Dialogue box gets open.

Enter the Network Destination, Gateway, Subnet Mask, Metrics, and Interface ID. Click on **Add**.

You will find the entry added to the below Static IP Routing Table as shown below.

Click on **OK**.



**Step 9:** **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.

***Note:*** *Before we click on* ***Run*** *simulation, user need to modify the code as per the* ***“Procedure to log RSSI and BER”*** *given below.*

***NOTE:******The following changes need to be done manually by the user inorder to carry out this experiment.***

**Procedure to log RSSI and BER (Possible in Standard / Pro Versions only):**

RSSI and BER in ZigBee project can be logged into a text file. The following code changes are required to log these parameters into a txt file.

* Go to NetSim Home page and click on **Open Simulation**.
* Click on Workspace Options and then click on **Open Code** and open the codes in Visual Studio. Set **Win32** or **x64** according to the NetSim build which you are using.

***NOTE: We recommend Visual Studio Community Edition 2017 or Higher.***

* Go to the Zigbee Project in the Solution Explorer. Open 802\_15\_4.c file and add the follwing lines of code highlighted in red, inside the **fn\_NetSim\_Zigbee\_init()** function as shown below:

**\_declspec (dllexport) int fn\_NetSim\_Zigbee\_Init(struct stru\_NetSim\_Network \*NETWORK\_Formal, \  
NetSim\_EVENTDETAILS \*pstruEventDetails\_Formal, char \*pszAppPath\_Formal, \  
char \*pszWritePath\_Formal, int nVersion\_Type, void \*\*fnPointer)  
{  
FILE\* fp;  
  
pstruEventDetails = pstruEventDetails\_Formal;  
NETWORK = NETWORK\_Formal;  
pszAppPath = pszAppPath\_Formal;  
pszIOPath = pszWritePath\_Formal;  
  
//RSSI BER SNR LOG  
fp = fopen("ZIGBEE\_BER\_LOG.txt", "w+");  
if (fp)  
{  
fprintf(fp, "PACKET\_ID,\tTRANSMITTER,\t\tRECEIVER,\tRX\_POWER(dBm),\tTOTAL\_RX\_POWER(dBm),\tBER");  
fclose(fp);  
}  
//RSSI BER SNR LOG  
  
fn\_NetSim\_Zigbee\_Init\_F(NETWORK\_Formal, pstruEventDetails\_Formal, pszAppPath\_Formal, \  
pszWritePath\_Formal, nVersion\_Type, fnPointer);  
return 0;  
}**

* Add the lines of code highlighted in red inside the **fn\_NetSim\_Zigbee\_Run()** function under **PHYSICAL\_IN\_EVENT**as shown below:

**case PHYSICAL\_IN\_EVENT:  
{  
NetSim\_PACKET \*pstruPacket;  
PACKET\_STATUS nPacketStatus;  
double SNR;  
double dBER;  
FILE\* fp;  
  
pstruPacket = pstruEventDetails->pPacket;  
if (pstruPacket->nReceiverId && pstruPacket->nReceiverId != pstruEventDetails->nDeviceId)  
{  
fnNetSimError("Different device packet received..");  
assert(false);  
return 0;  
}  
  
  
if (!ZIGBEE\_CHANGERADIOSTATE(pstruEventDetails->nDeviceId, WSN\_PHY(pstruEventDetails->nDeviceId)->nRadioState, RX\_ON\_IDLE))  
return 0;  
  
if (WSN\_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower - GET\_RX\_POWER\_mw(pstruPacket->nTransmitterId, pstruPacket->nReceiverId, pstruEventDetails->dEventTime) >= WSN\_PHY(pstruEventDetails->nDeviceId)->dReceiverSensivity)  
pstruPacket->nPacketStatus = PacketStatus\_Collided;  
nPacketStatus = pstruPacket->nPacketStatus;  
ZIGBEE\_SINR(&SNR,  
WSN\_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower,  
GET\_RX\_POWER\_mw(pstruPacket->nTransmitterId, pstruPacket->nReceiverId, pstruEventDetails->dEventTime));  
  
dBER = fn\_NetSim\_Zigbee\_CalculateBER(SNR);  
  
//RSSI BER SNR LOG  
double rxpwr = MW\_TO\_DBM(WSN\_PHY(pstruEventDetails->nDeviceId)->dTotalReceivedPower);  
double total\_rxpwr = GET\_RX\_POWER\_dbm(pstruPacket->nTransmitterId, pstruPacket->nReceiverId, pstruEventDetails->dEventTime);**

**fp = fopen("ZIGBEE\_BER\_LOG.txt", "a+");  
if (fp)  
{  
fprintf(fp, "\n%lld,\t\t%s,\t%s,\t%lf,\t%lf,\t\t%lf", pstruPacket->nPacketId,  
DEVICE\_NAME(pstruPacket->nTransmitterId),  
DEVICE\_NAME(pstruPacket->nReceiverId),  
rxpwr, total\_rxpwr, dBER);  
fclose(fp);  
}  
//RSSI BER SNR LOG  
  
if (fn\_NetSim\_Packet\_DecideError(dBER, pstruEventDetails->dPacketSize))**

* Right click on the **ZigBee** project in the solution explorer and click on rebuild.
* After the Zigbee project is rebuild successful, go back to the network scenario.

**Step 10:** Run the simulation for 10 Seconds. Once the simulation is complete, it will generate a text file named **ZIGBEE\_BER\_LOG.txt** containing **RSSI** and **BER** in the binary folder of NetSim. i.e. **<NetSim Install Directory>/bin**.

## Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RSSI, PER, BER vs. Distance (path-loss: linear in log-distance, with ) | | | | |
| Distance(m) | **RSSI (dBm)**  (pathloss model) | **BER** | **PER** | **PLR**  (after MAC retransmissions\*) |
| 5 | -64.51 | 0.00 | 0 | 0 |
| 10 | -75.04 | 0.00 | 0 | 0 |
| 15 | -81.20 | 0.00 | 0 | 0 |
| 20 | -85.58 | 0.00 | 0 | 0 |
| 25 | -88.97 | 0.00 | 0 | 0 |
| 30 | -91.74 | 0.00 | 0 | 0 |
| 35 | -94.08 | 0.000005 | 0.0051 | 0 |
| 40 | -96.11 | 0.000229 | 0.2076 | 0 |
| 45 | -97.90 | 0.002175 | 0.8905 | 0.447 |
| 50 | -99.51 | 0.008861 | 0.9999 | 1 |
| 55 | -100.95 | 0.022370 | 1 | 1 |
| 60 | -102.28 | 0.042390 | 1 | 1 |
| 65 | -103.49 | 0.067026 | 1 | 1 |
| 70 | -104.62 | 0.094075 | 1 | 1 |
| 75 | - | - | - | - |
| 80 | - | - | - | - |

\* The IEEE 802.15.4 MAC implements a retransmission scheme that attempts to recover errored packets by retransmission. If all the retransmission attempts are also errored, the packet is lost.

The table above reports the RSSI (Received Signal Strength), BER (Bit Error Rate), and Packet Error Rate (PER), and the Packet Loss Rate (PLR) as the distance between the sensor to the sink is increased from 5m to 50m with path loss exponent . We see that the BER is 0 until a received power of about -92dBm. At a distance of 35m the received power is -94 dBm, and we notice a small BER of . As the distance is increased further the BER continues to grow and at 45m the BER is about , yielding and Here is obtained from the following formula (which assumes independent bit errors across a packet)

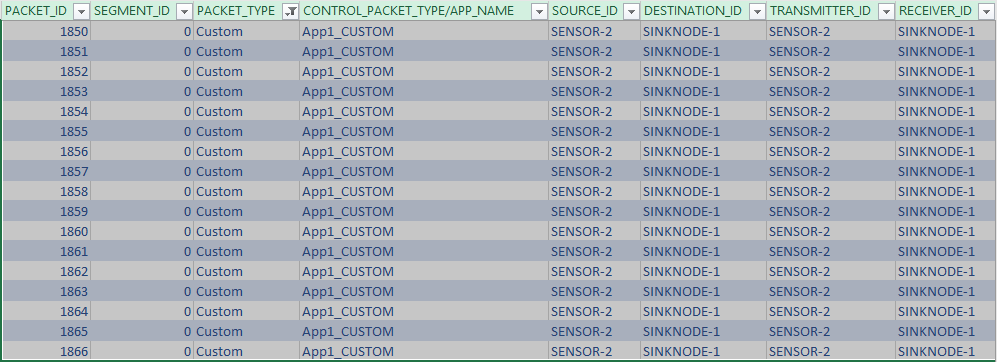
,

Where,

The in the above table has been obtained from NetSim, which implements the details of the IEEE 802.15.4 MAC acknowledgement and reattempt mechanism. This mechanism is complex, involving a MAC acknowledgement, time-outs, and multiple reattempts. Analysis of the therefore, is not straightforward. Assuming that the probability of MAC acknowledgement error is small (since it is a small packet), the can be approximated as , where is the maximum number of times a packet can be retransmitted.

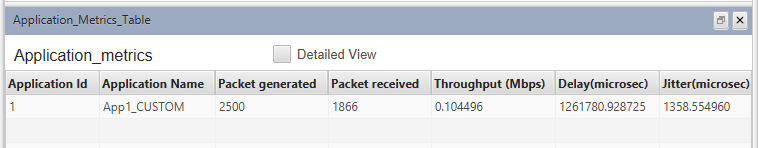
**Steps to calculate Packet Loss Rate:**

* Open Packet Trace from the Results Dashboard. Filter the PACKET TYPE column as Custom and note down the packet id of the last packet sent from the PACKET ID column.



This represents the total number of packets sent by the source.

* Note down the Packets Received from the Application Metrics in the Results Dashboard.



This represents the total number of packets received at the destination.

* Calculate the total number of Lost Packets and PLR as follows:

For the above case,

## Inference:

It is clear that Internet applications, such as banking and reliable file transfer, require that all the transmitted data is received with accuracy. The Internet achieves this, in spite of unreliable communication media (no medium is reliable) by various protocols above the network layer. Many IoT applications, however, can work with less than packet delivery without affecting the application. Take, for example, the farm moisture sensing application mentioned in the introduction. The moisture levels vary slowly; if one measurement is lost, the next received measurement might suffice for the decision-making algorithm. This sort of thinking also permits the IoT applications to utilize cheap, low power devices, making the IoT concept practical and affordable.

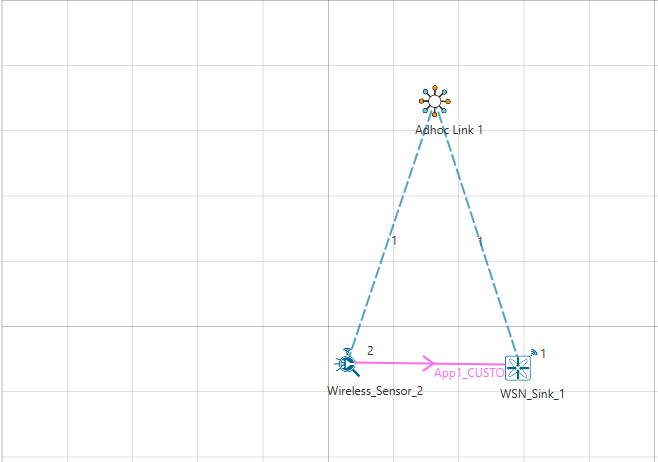
With the above discussion in mind, let us say that the application under consideration requires a measurement delivery rate of at least . Examining the table above, we conclude that the sensor-sink distance must not be more than meters. Thus, even a acre farm would require multi-hopping to connect sensors to a sink at the edge of the farm.

In Part 2 of this experiment we will study the placement of a single router between the sensor and the sink, so as to increase the sensor-sink distance beyond meters.

## Part 2 – Reaching a Longer Distance by Multihopping

**Sample1:**

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



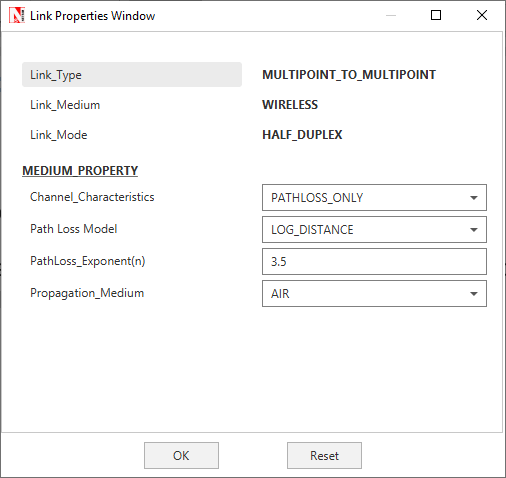
## Procedure:

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

**Step 1:** The distance between the WSN Sink and Wireless Sensor is 40 meters.

**Step 2:** In the Interface Zigbee > Data Link Layer of Wireless Sensor 2, **Ack Request** is set to Enable and **Max Frame Retries** is set to 3.

**Step 3:** The **Ad hoc Link** properties are set as follows:



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

A CUSTOM Application is generated from Wireless Sensor 2 i.e. Source to WSN Sink 1 i.e. Destination with Packet Size set to 70 Bytes and Inter Arrival Time set to 100000 µs.

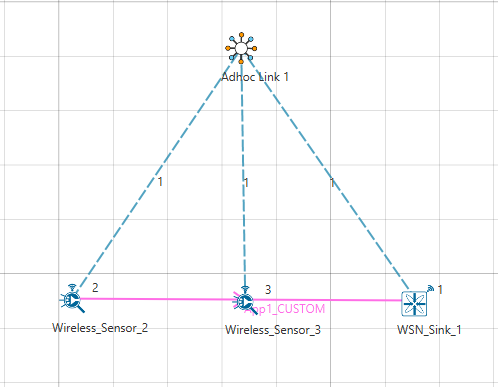
The Packet Size and Inter Arrival Time parameters are set such that the Generation Rate equals 5.6 Kbps. Generation Rate can be calculated using the formula:

**Step 5:** Run the Simulation for 100 Seconds. Once the simulation is complete, note down the Packet Generated value and Throughput value from the **Application Metrics**.

Note down the Packet Received, Packet Errored, and Packet Collided from the **Link Metrics**.

**Sample-2:**

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



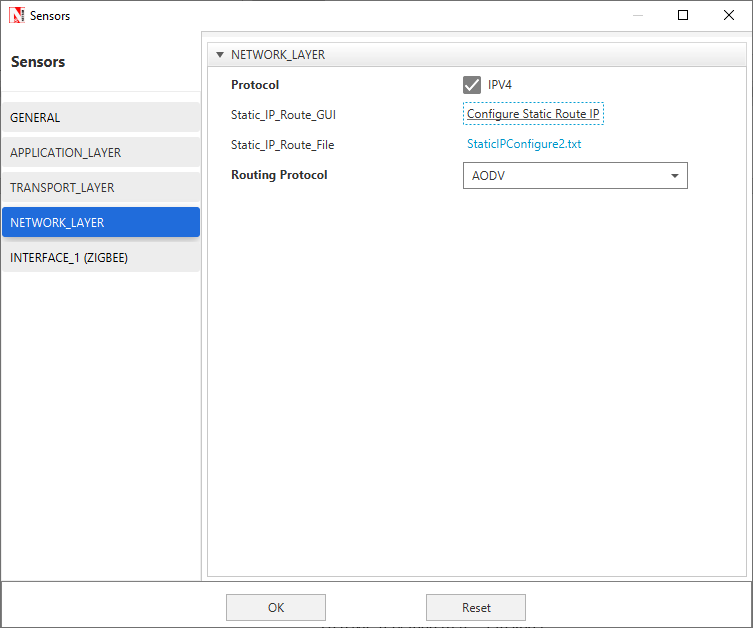
## Procedure:

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

**Step 1:** One more Wireless Sensor is added to this network. The distance between Wireless Sensor 2 and Wireless Sensor 3 is 40 meters and the distance between Wireless Sensor 3 and the WSN Sink is 40 meters.

**Step 2:** The following procedures were followed to set Static IP:

Go to Network Layer properties of Wireless Sensor 2, Click on **Configure Static Route IP**.

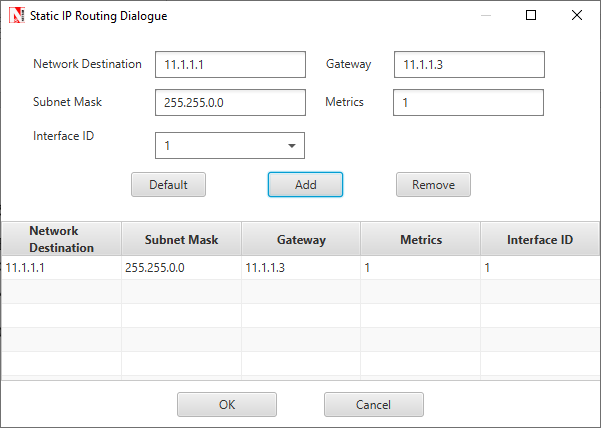


Static IP Routing Dialogue box gets open.

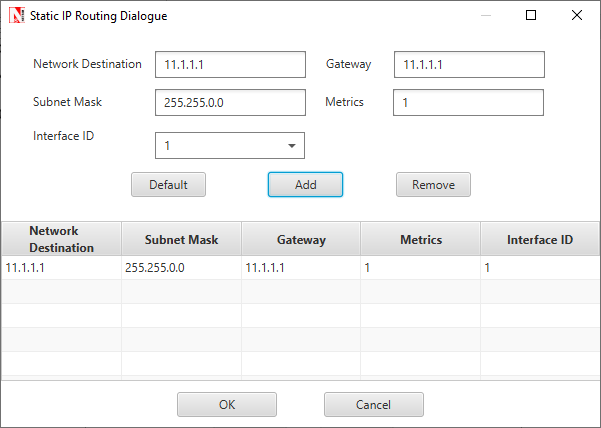
Enter the Network Destination, Gateway, Subnet Mask, Metrics, and Interface ID. Click on **Add**.

You will find the entry added to the below Static IP Routing Table as shown below:

Click on **OK**.



Similarly, Static IP is set for Wireless Sensor 3 as shown below:



**Step 3:** Run the Simulation for 100 Seconds. Once the simulation is complete, note down the Packet Generated value and Throughput value from the **Application Metrics**.

Note down the Packet Received, Packet Errored, and Packet Collided from the **Link Metrics**.

## Output:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Source-Sink Distance (m) | Packets Generated | Packets Received | Packets Errored (PHY) | Packets Collided | Packet Loss (MAC) | PLR | Mean Delay |
| Direct sensor-sink link | 40 | 1000 | 1012 | 244 | 0 | 0 | 0 | 6394.06 |
| Router between sensor and sink | 80  (router at midpoint) | 1000 | 1016 | 540 | 0 | 0 | 0 | 14075.90 |

***NOTE:******Packet loss (PHY) is the number of packets that were received in error and then recovered by retransmission. Packets received is slightly higher than packets generated on account of retransmissions of successful packets in case of ACK errors.***

## Inference:

In Part 1 of this experiment we learnt that if the sensor device uses a transmit power of 0dBm, then for one-hop communication to the sink, the sensor-sink distance cannot exceed 40m. If the sensor-sink distance needs to exceed 40m (see the example discussed earlier), there are two options:

1. The transmit power can be increased. There is, however, a maximum transmit power for a given device. Wireless transceivers based on the CC 2420 have a maximum power of 0dBm (i.e., about 1 mW), whereas the CC 2520 IEEE 802.15.4 transceiver provides maximum transmit power of 5dBm (i.e., about 3 mW). Thus, given that there is always a maximum transmit power, there will always be a limit on the maximum sensor-sink distance.
2. Routers can be introduced between the sensor and the sink, so that packets from the sensor to the sink follow a *multihop* path. A router is a device that will have the same transceiver as a sensor but its microcontroller will run a program that will allow it to forward packets that it receives. Note that a sensor device can also be programmed to serve as a router. Thus, in IOT networks, sensor devices themselves serve as routers.

In this part of the experiment we study the option of introducing a router between a sensor and the sink to increase the sensor-sink distance. We will compare the performance of two networks, one with the sensor communicating with a sink at the distance of 40m, and another with the sensor-sink distance being 80m, with a sensor at the mid-point between the sensor and the sink.

Part 2, Sample 1 simulates a one hop network with a sensor-sink distance of 40m. We recall from Part 1 that, with the transceiver model implemented in NetSim, 40m is the longest one hop distance possible for 100% packet delivery rate. In sample 2, To study the usefulness of routing we will set up network with a sensor-sink distance of 80m with a packet router at the midpoint between the sensor and the sink.

The measurement process at the sensor is such that one measurement (i.e., one packet) is generated every 100ms. The objective is to deliver these measurements to the sink with 100% delivery probability. From Part 1 of this experiment we know that a single hop of 80m will not provide the desired packet delivery performance.

The Table at the beginning of this section shows the results. We see that both networks are able to provide a packet delivery probability of 100%. It is clear, however, that since the second network has two hops, each packet needs to be transmitted twice, hence the mean delay between a measurement being generated and it being received at the sink is doubled. Thus, the longer sensor-sink distance is being achieved, for the same delivery rate, at an increased delivery delay.

The following points may be noted from the table:

1. The number of packets lost due to PHY errors. The packet delivery rate is 100% despite these losses since the MAC layer re-transmission mechanism is able to recover all lost packets.
2. There are no collisions. Since both the links (sensor-router and router-sink) use the same channel and there is no co-ordination between them, it is possible, in general for sensor-router and router-sink transmissions to collide. This is probable when the measurement rate is large, leading to simultaneously nonempty queues at the sensor and router. In this experiment we kept the measurement rate small such that the sensor queue is empty when the router is transmitting and vice versa. This avoids any collisions.