MANIPAL INSTITUTE
OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that Ms/Mr. ……...……

Reg. No.: …..…… Section: …… Roll No.: ……...

has satisfactorily completed the laboratory exercises prescribed for **Communication Networks Lab** [ECE-3212] of VI Semester B. Tech. (E & C Engg.) Degree at MIT, Manipal, in the academic year 2016 - 2017.

Date: .....

Signature
Faculty in Charge

Signature
Head of the Department

CONTENTS

ECE 3212 COMMUNICATION NETWORK LAB [0 0 6 2]

Total Number of Lab Classes: 10

Course Objectives

CO1: To simulate the Wired and Wireless LANs and Wi-Max.
CO2: To Configure and test the various Data Link Layer protocols.
CO3: To simulate and verify the function of various Network Layer protocols.
CO4: To Demonstrate TCP, UDP Communication Protocols.
CO5: To Simulate the MANETs and WSNs and analyse their performance.

Course Outcomes

At the end of this course, student will be able to:

- Analyse the congestion in Point-to-Point networks and Wireless LANs.
- Configure and test the ALOHA, CSMA, CSMA/CD, and CSMA/CA protocols.
- Simulate and verify the functioning of Dijkstra's and Bellman-ford algorithms.
- Demonstrate the different applications using TCP, UDP Communication Protocols.
- Verify the performance of various Error control protocols in Data Link Layer.

Evaluation Plan

Communication Network Lab

- Internal Assessment Marks: **60 Marks**
  - ✓ Continuous evaluation component (for each experiment): **10 Marks**
  - ✓ Assessment is based on conduction of each experiment, exercise problems, answering the questions related to the experiment.
  - ✓ Total marks of the 10 experiments scaled to **60 Marks**
- End semester assessment: **40 Marks**

Communication Network Project Lab

- Internal Assessment: **60 Marks**
  - ✓ Continuous evaluation component (for each progress): **10 Marks**
  - ✓ Assessment is based on satisfactory progress in execution of project.
  - ✓ Total marks of the 6 evaluations is **60 Marks**
- End semester project demonstration: **40 Marks**

Note: Final Grading is based on the average of marks scored in Communication Networks Lab and Project Lab.

PART A

Experiment 1
Point-to-Point Networks

Objective:

A) To simulate a three point-to-point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped.

B) To simulate the transmission of ping message over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

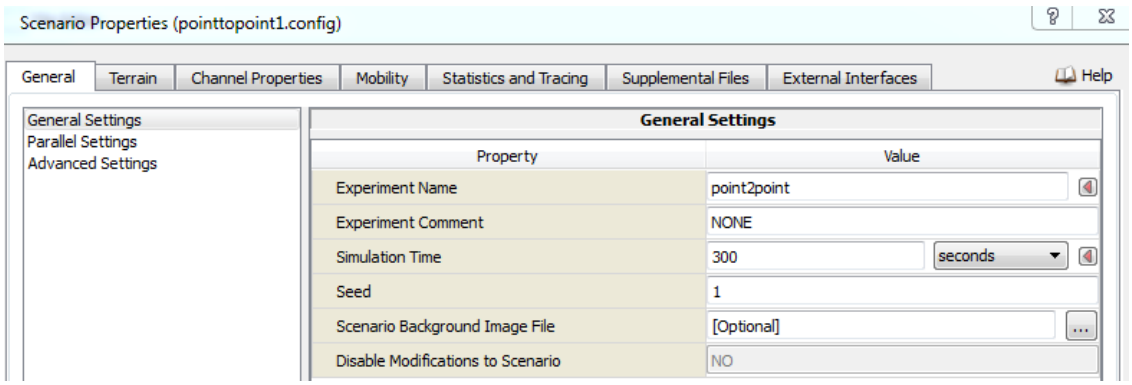
Procedure:

Create a folder with your registration number under"**C://qualnet/7.4/scenarios/user/**"

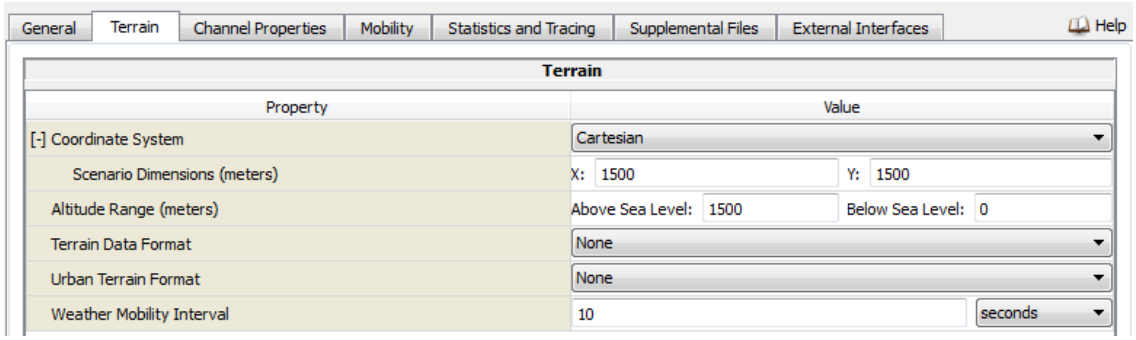
A) Go to File→new→save as→ point-to-point

Select Scenario Properties → General Settings → Give Experiment name, Simulation Time

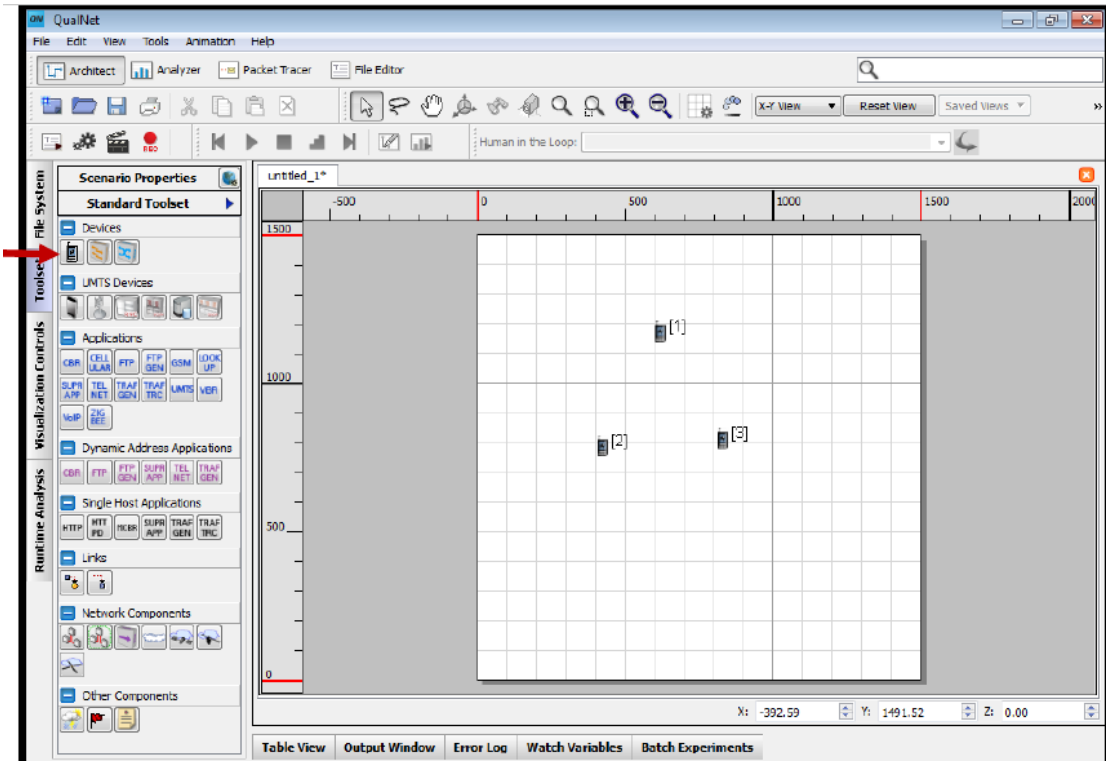
Click Apply, Ok



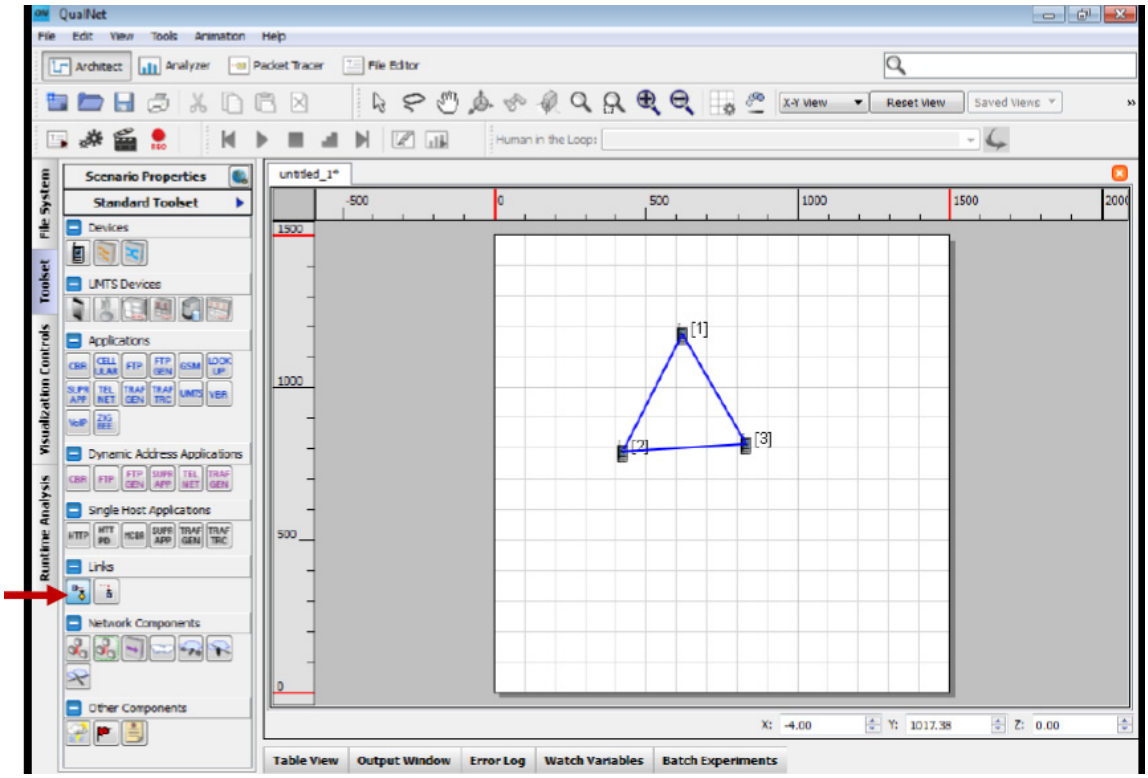
Set the Terrain Properties



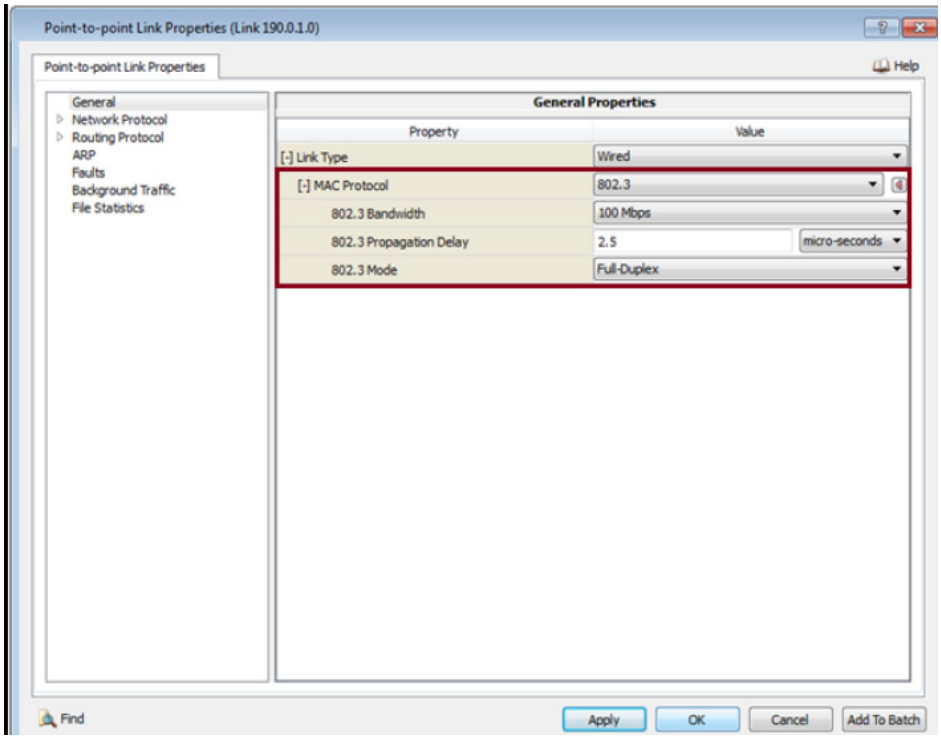
Step 1: Select Default icon from **Standard Toolset window→ Devices** and Place the three nodes on the canvas.



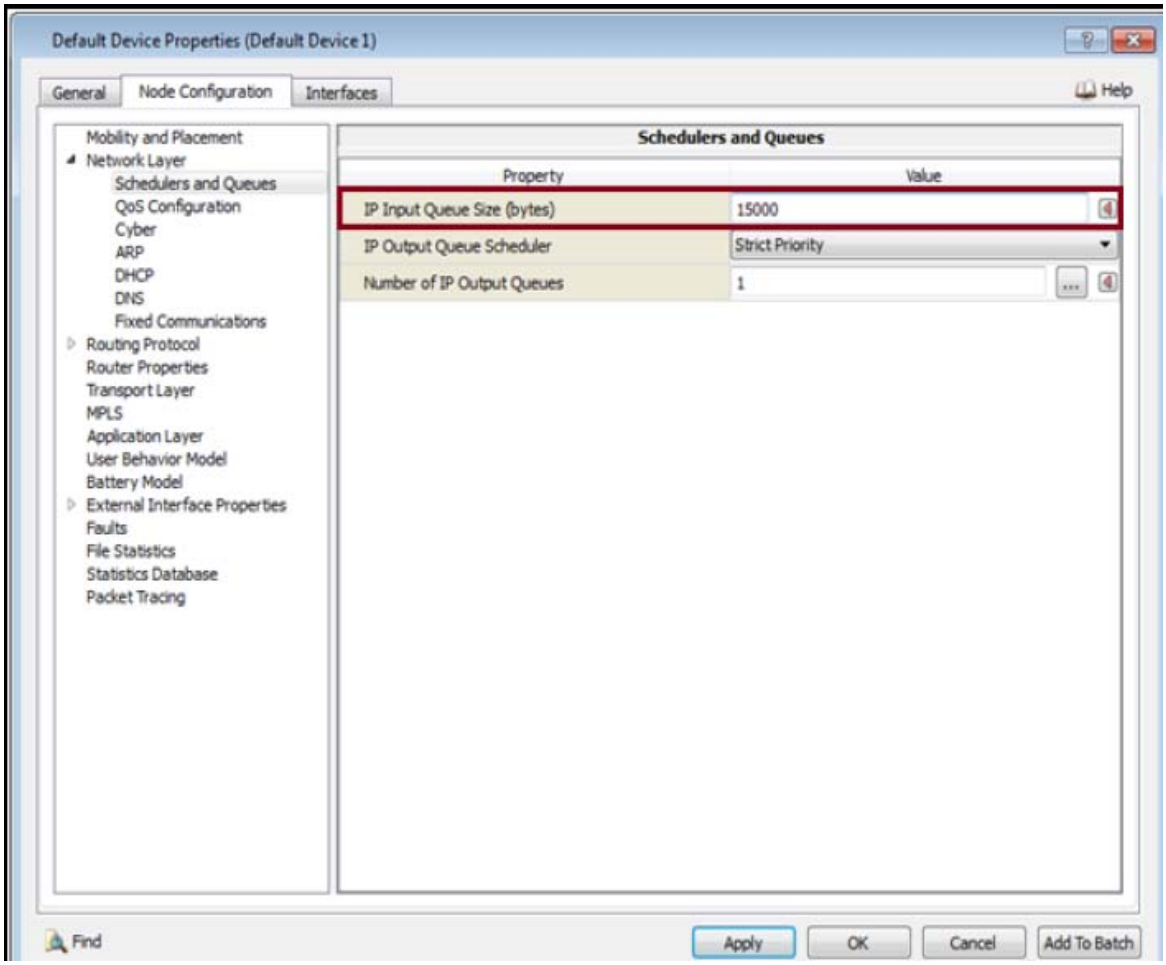
Step 2: Connect point-to-point link between three nodes using **link** from **Standard Toolset window.**



Step 3: Double click on the links to set duplex link and bandwidth between the nodes by configure 802.3 as MAC Protocol.

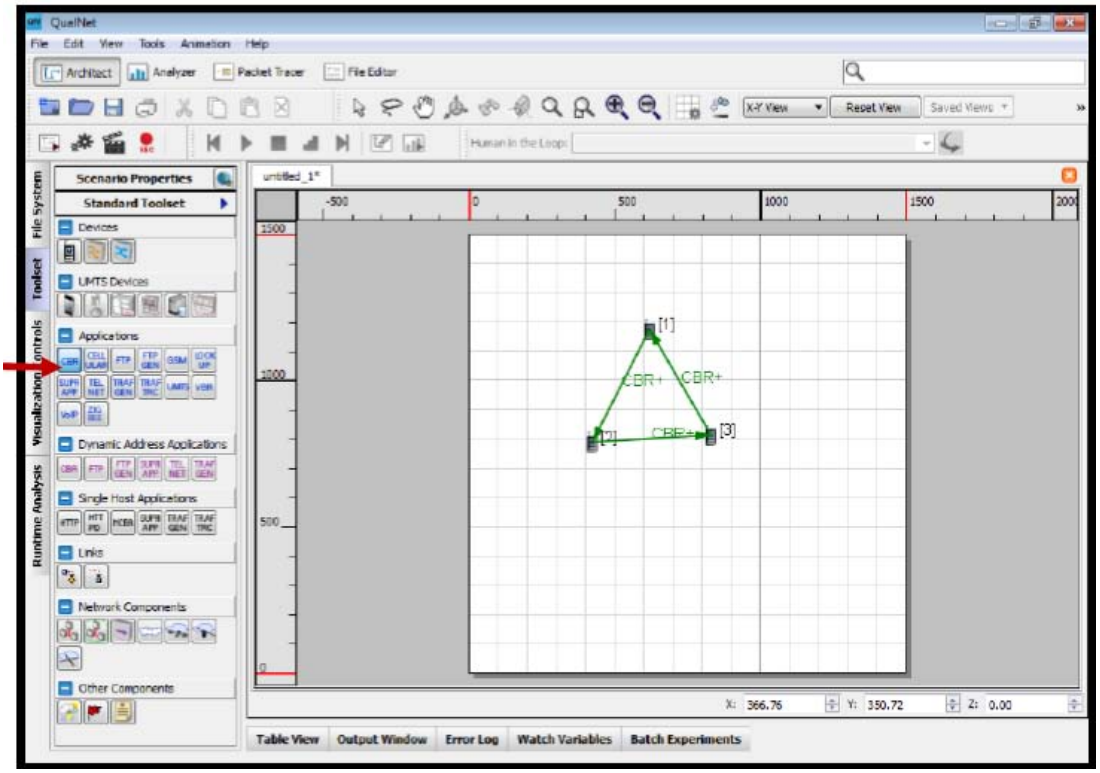


Step 4: To set queue size of each node in the scenario double click on the nodes, go to **Node Configuration tab→Network Layer→Schedulers and Queues.**



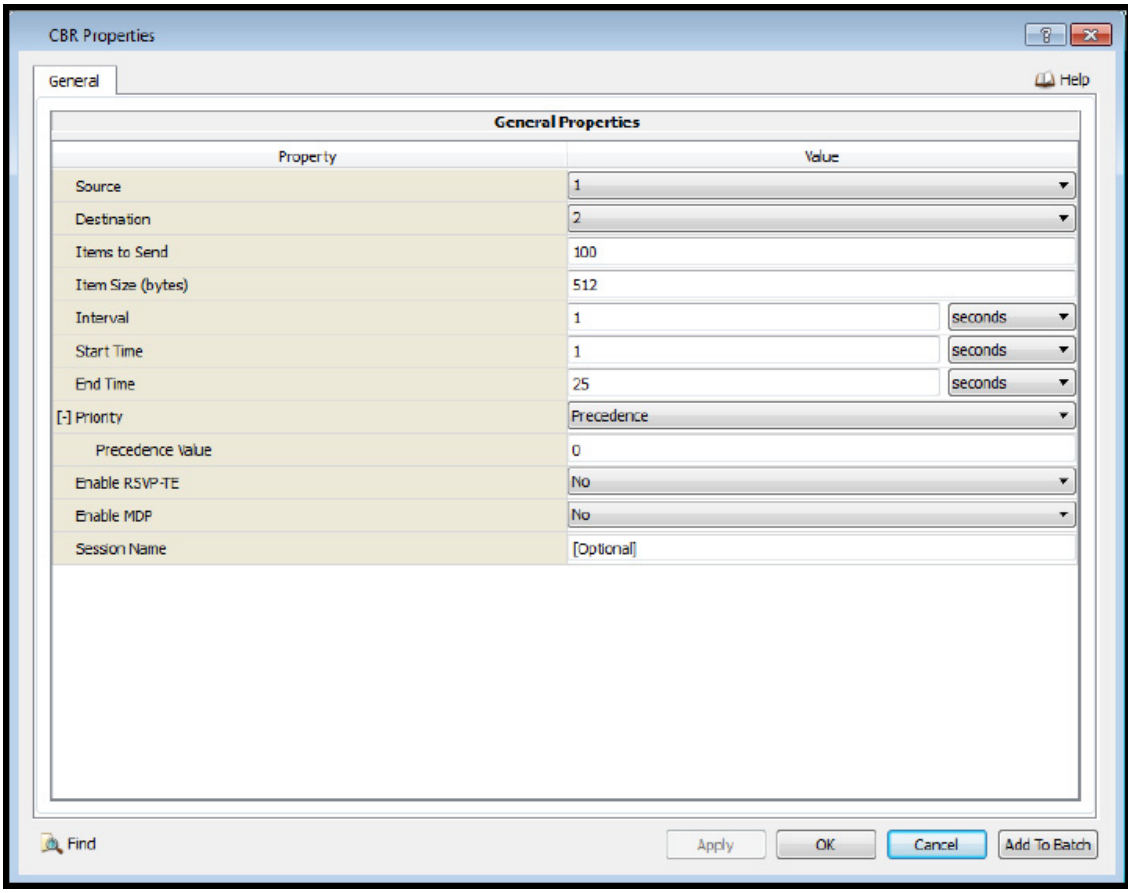
Step 5: Select the **Applications** tab of **Standard Toolset window**

Select **CBR** and draw the application between Node 1 and Node 2. Similarly Node 2 to Node 3 and Node 3 Node 1.



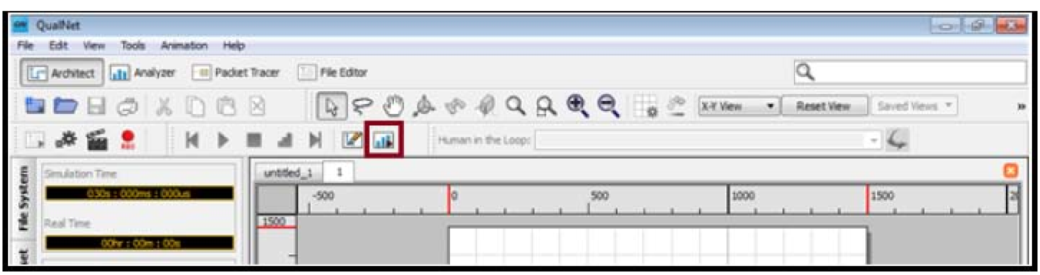
Step 6: To set CBR Application parameters, go to Table View (bottom panel of GUI) →Application Tab→double click on CBR Application

Set the number of CBR packets (Items to send) as per the user requirement.



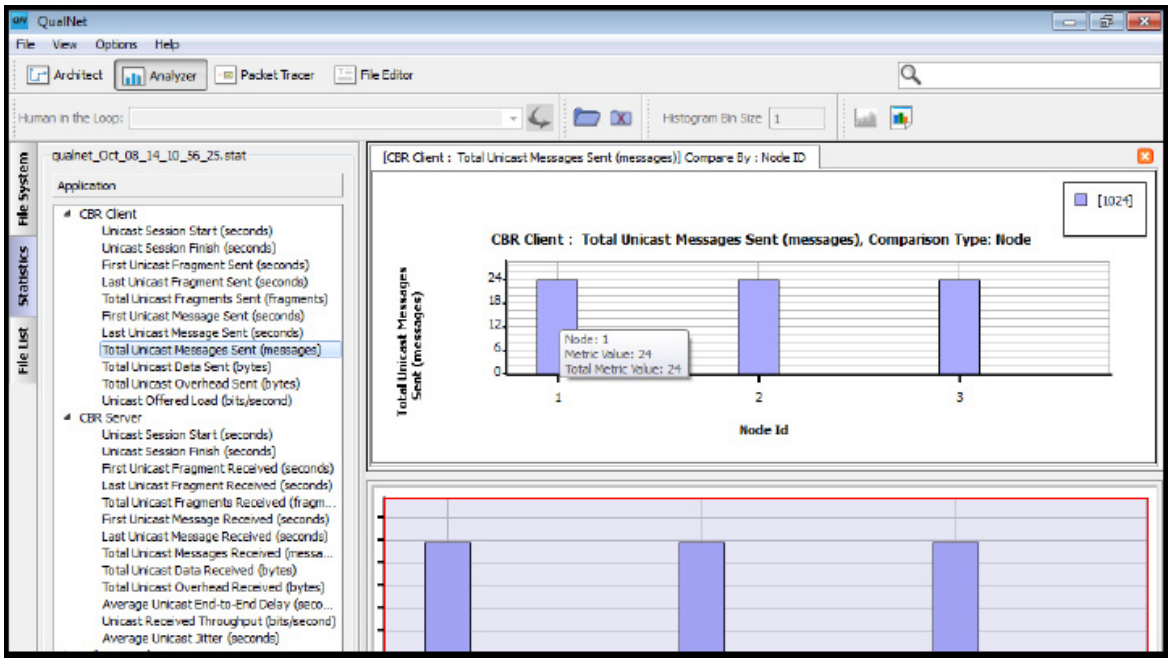
Step 7: Save the changes by clicking on the **Save** button. Click on the **Run** button. Click on the **Play** button to execute the scenario.

Step 8: On completion of the Scenario execution, check the corresponding Statistics for the desired results click on **Analyze Statistics of Current Scenario.**



Step 9: To check the no. of packets at sender, go to Application tab →CBR Client→Total Unicast Messages Sent

To check the no. of packets at receiver, go to Application tab→CBR Server →Total Unicast Messages Received



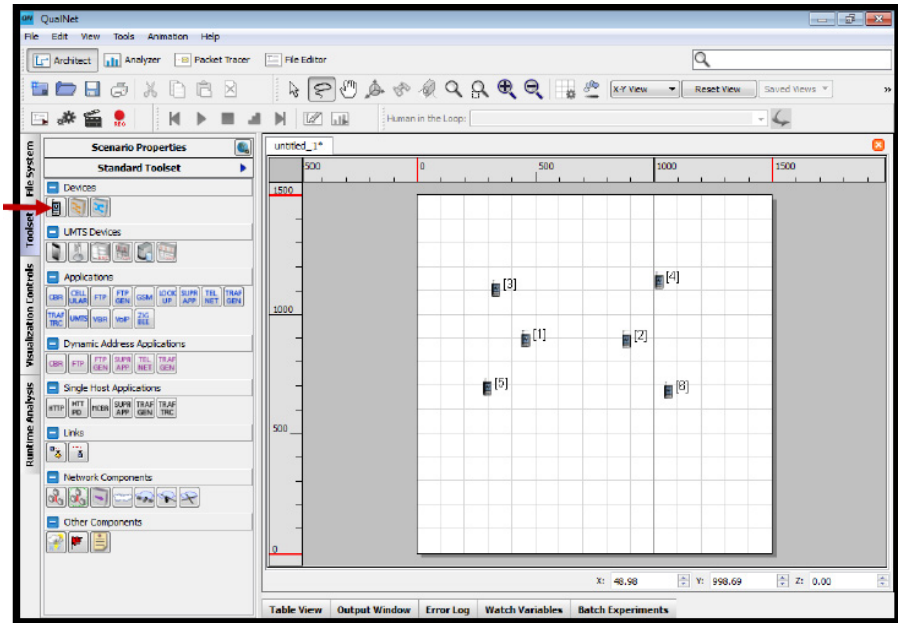
Note: The packet drop is calculated as follows:

Packet drop = Total no. of packets sent – Total no. of packets received

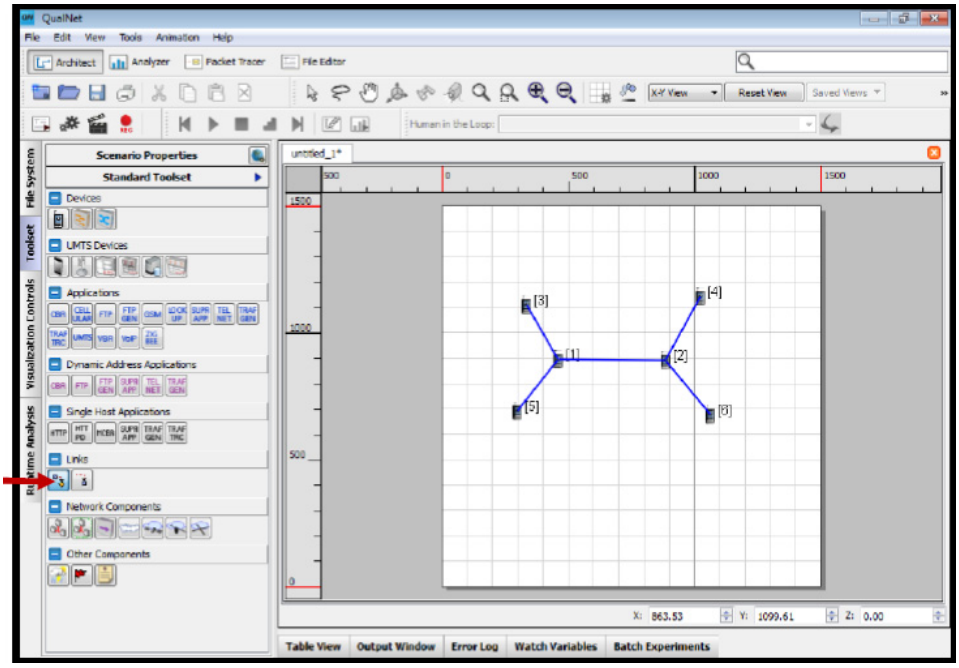
B) Procedure: Go to File→new→save as→congestion

Select Scenario Properties → General Settings → Give Experiment Name, Simulation Time Click Apply, Ok

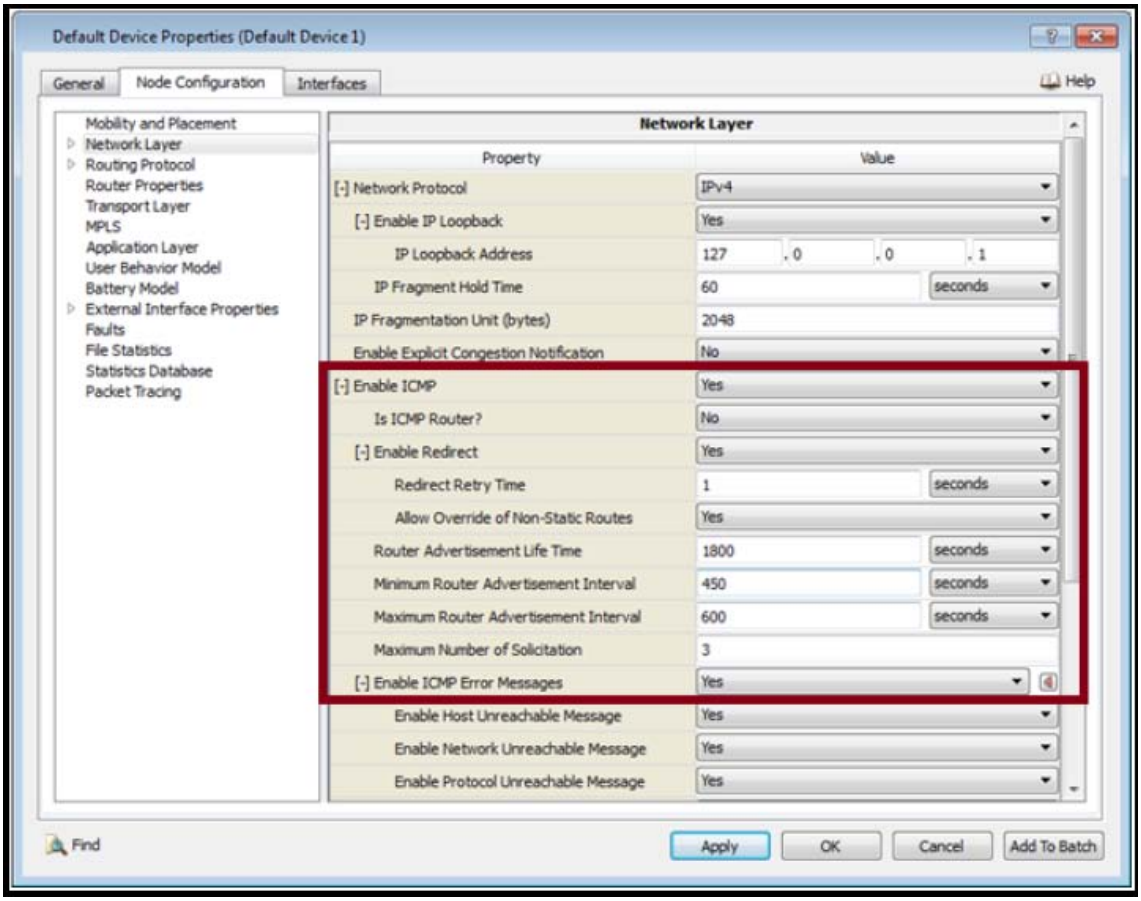
Step 1: Select Default icon from **Standard Toolset window** and Place the Six nodes on the canvas.



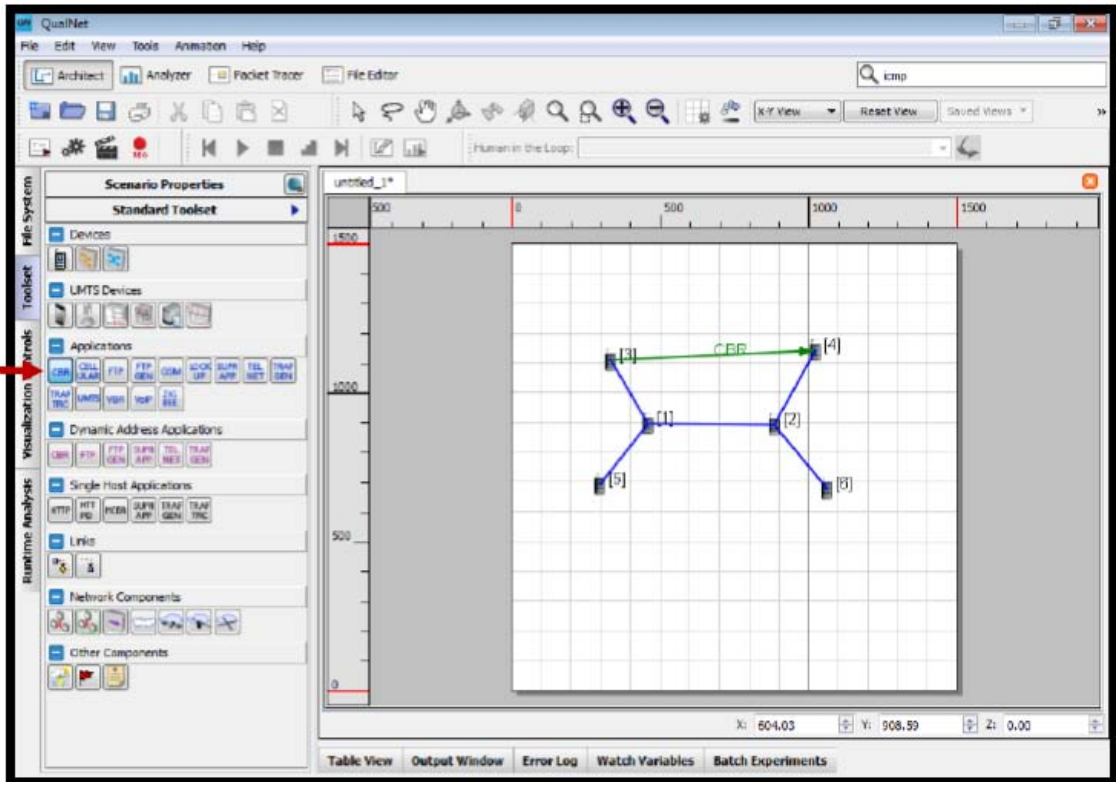
Step 2: Connect point-to-point link between three nodes using **link** from **Standard Toolset window.**



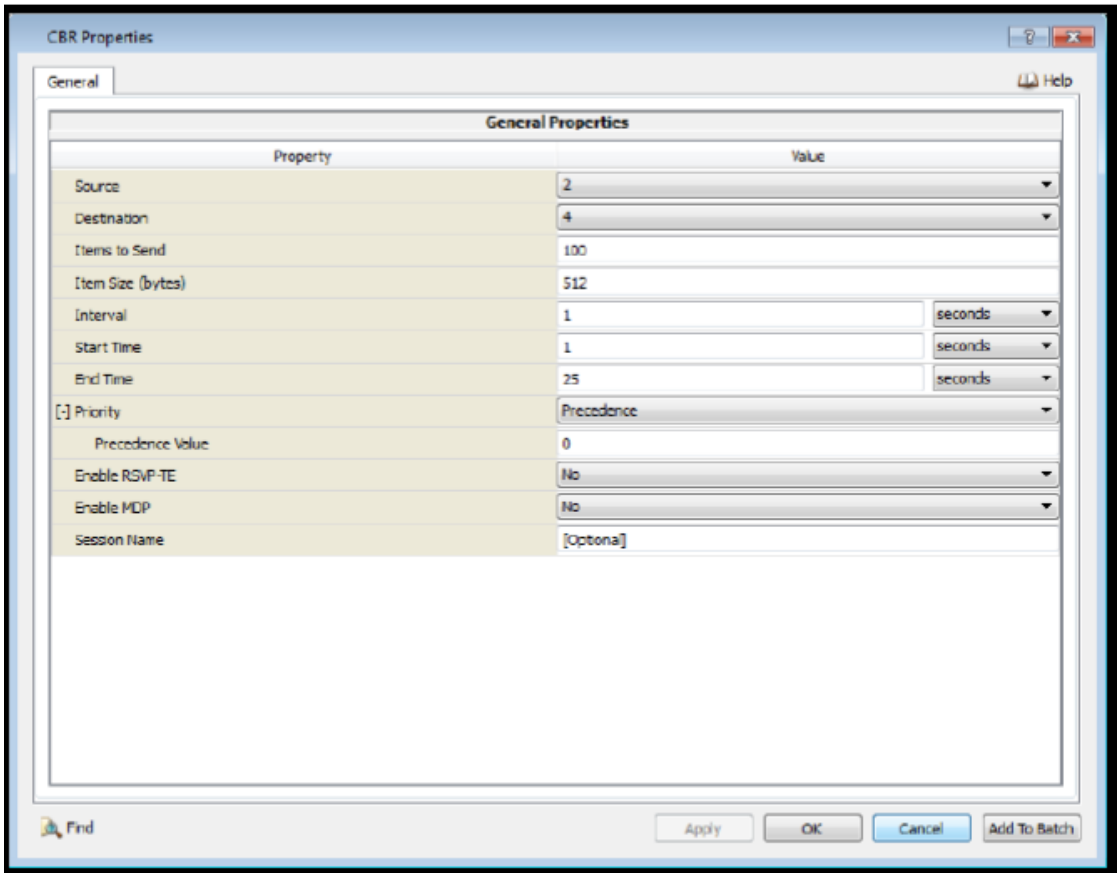
Step 3: To set ICMP of each node in the scenario double click on the nodes, go to Node Configuration tab→Network Layer→Enable ICMP.



Step 4: Select the **Applications** tab of **Standard Toolset window**
Select **CBR** and draw the application between Node 3 and Node 4.

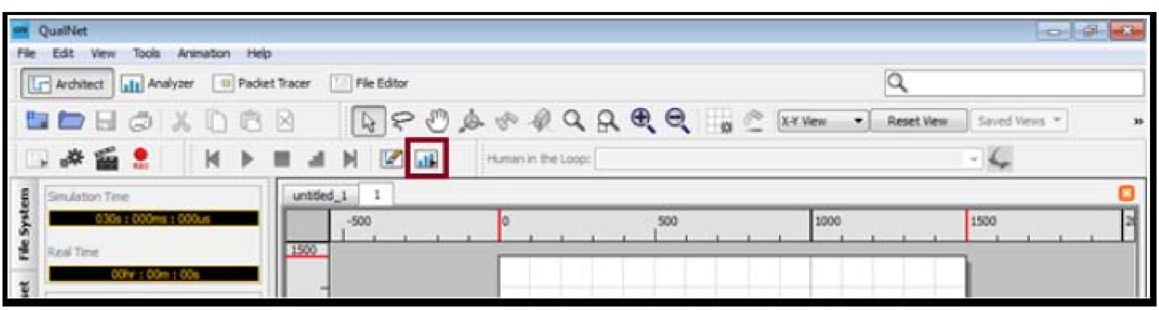


Step 5: To set CBR Application parameters, go to Table View (bottom panel of GUI) →Application Tab→ double click on CBR Application.
Set the number of CBR packets (Items to send) as per the user requirement.



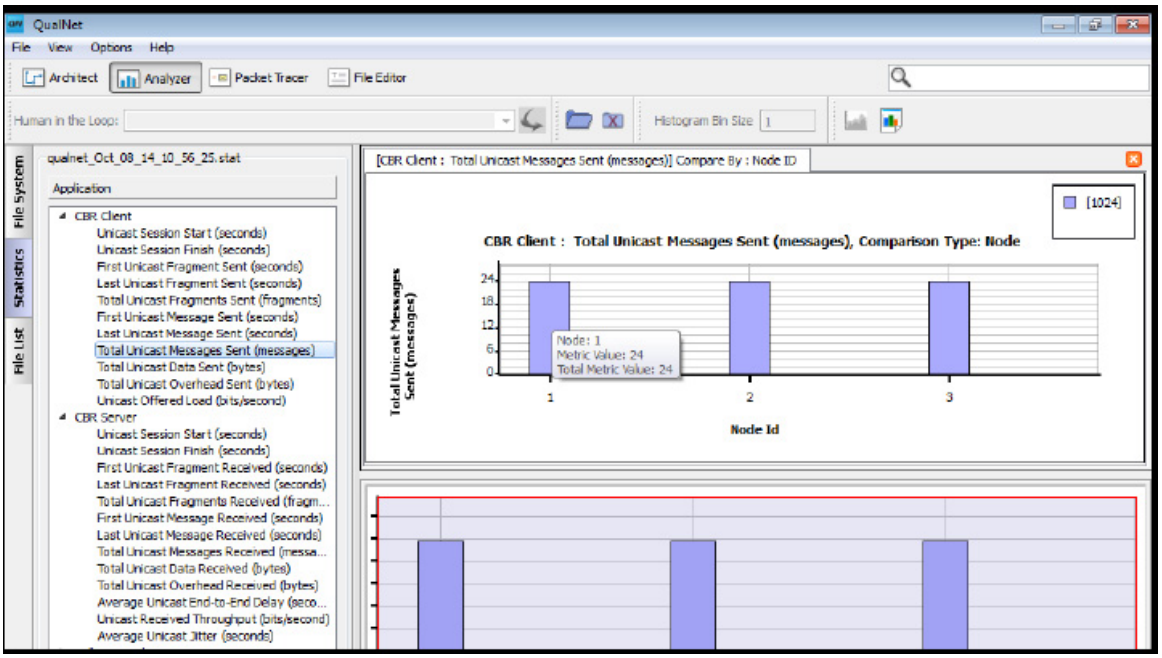
Step 6: Save the changes by clicking on the **Save** button. Click on the **Run** button. Click on the **Play** button to execute the scenario.

Step 7: On completion of the Scenario execution, check the corresponding Statistics for the desired results click on **Analyze statistics of Current Scenario.**



Step 8: To check the no. of packets at sender, go to Application tab→CBR Client →Total Unicast Messages Sent

To check the no. of packets at receiver, go to Application tab→CBR Server→Total Unicast Messages Received



Similarly, observe Average End-to-End Delay, Throughput, and Jitter.

Note: The packet drop is calculated as follows
Packet drop =Total no. of packets sent – Total no. of packets received

Exercise 1: Simulate a four node point-to-point network with the links connected as follows: n1-n3, n2-n3 and n3-n4, Apply TCP agent between n1-n4 and UDP between n2-n4. Apply relevant application over TCP and UDP agents changing the parameters and determine the number of packets sent by TCP/UDP.

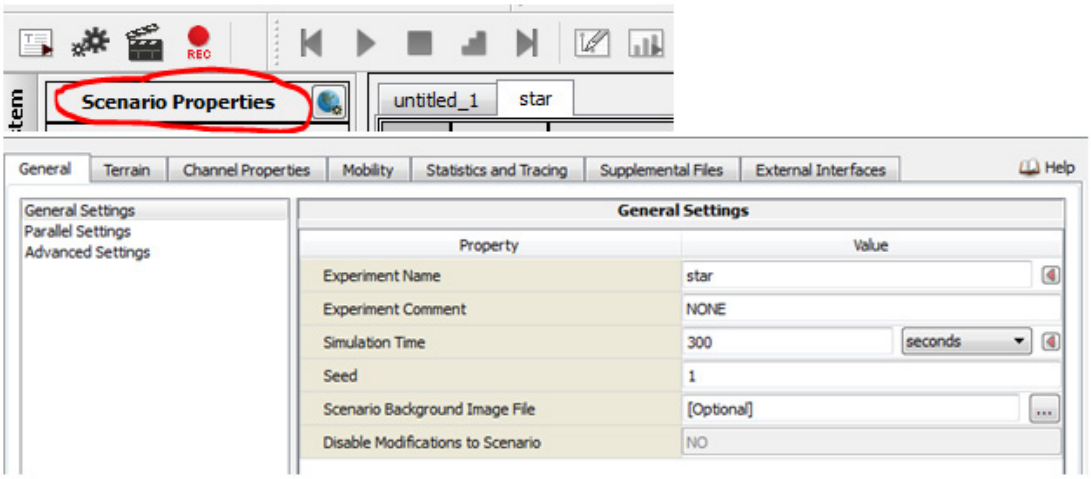
Experiment 2
Network Topologies

Objective: To Simulate and compare the performance of network with different topologies such as Star, Ring and Mesh.

A) Star Topology
Procedure: Go to file → New → Save as → star

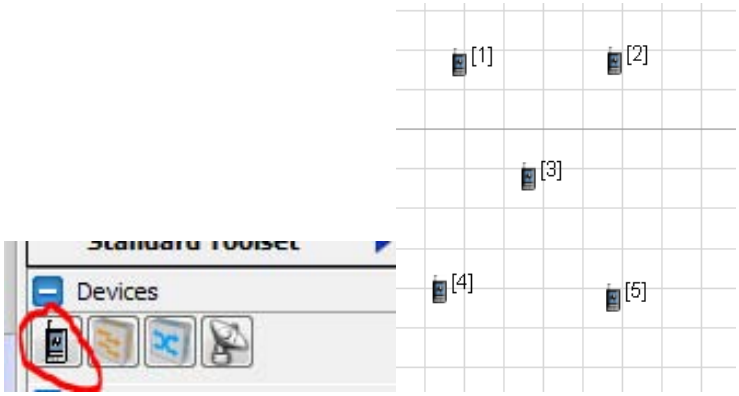
Go to Scenario Properties → General Setting → Give experiment name and simulation time

Click Apply, Ok

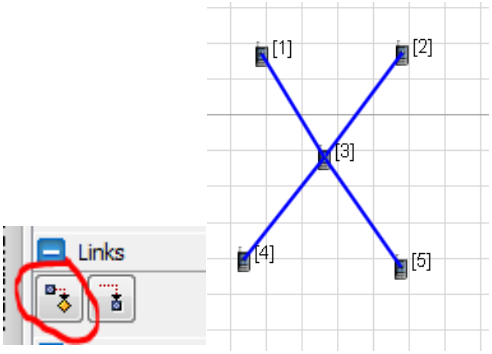


Keep the default value in other fields
Apply and click ok.

Step 1: Placing the nodes, click on the device and place the nodes in the form of star

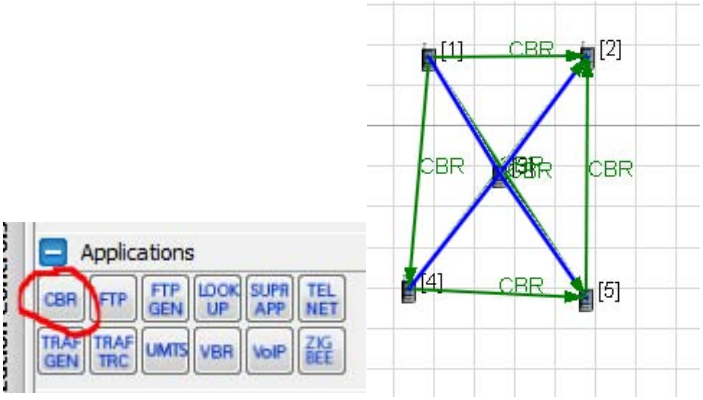


Step 2: Link to all nodes: click on the link and link all the nodes

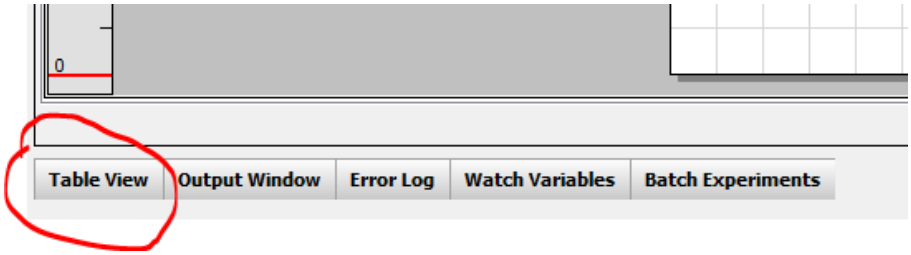


Save the scenarios: click save

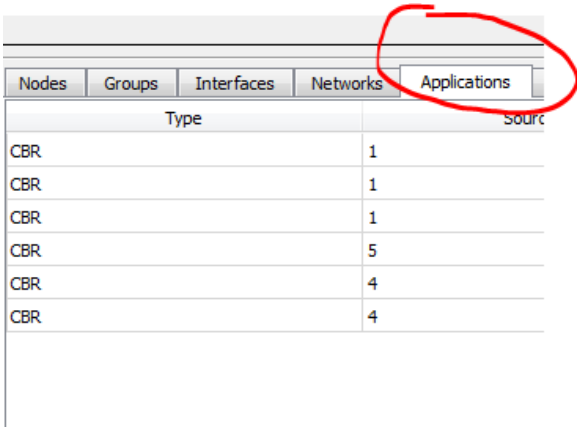
Step 3: Click on the CBR to give traffic between the nodes using CBR application under applications tab



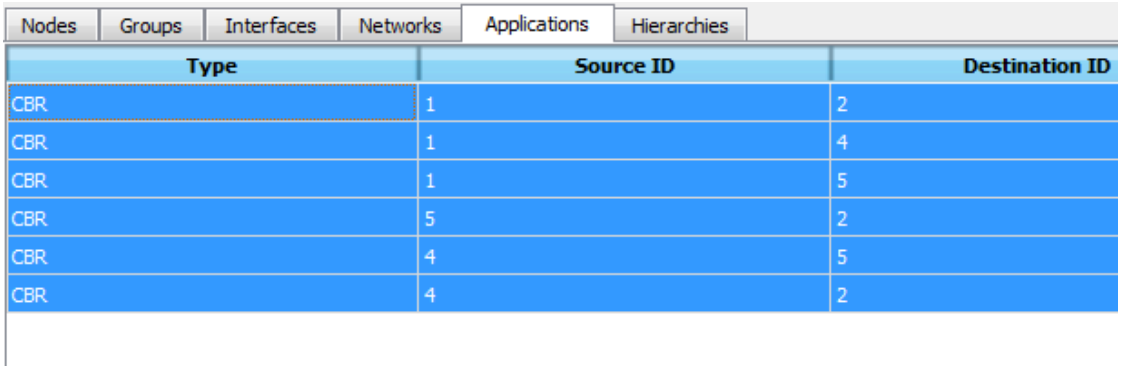
Configure CBR applications: click table view



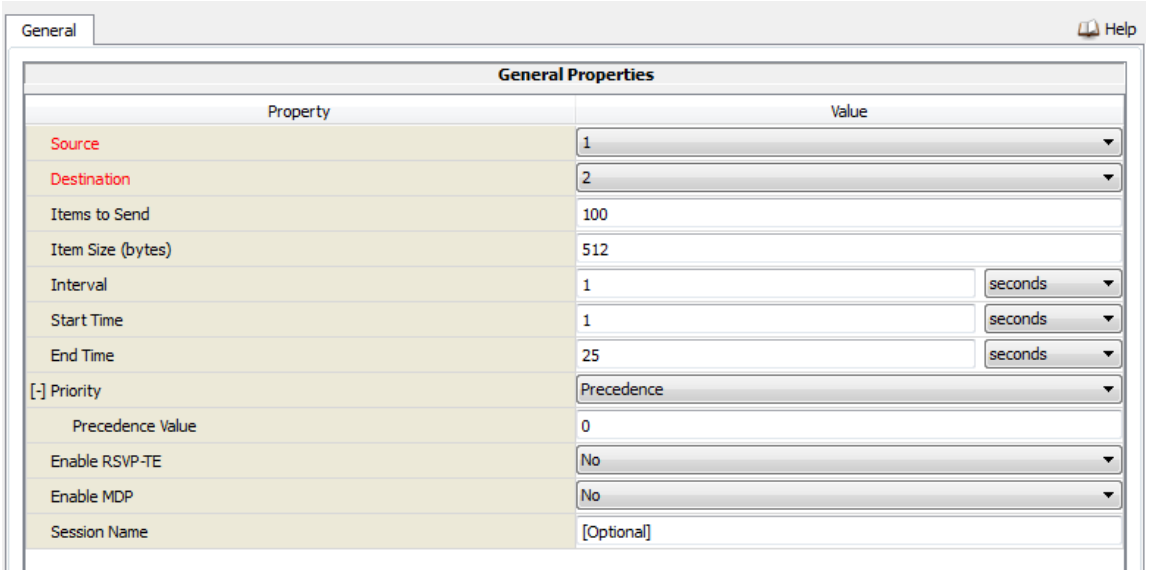
Click application Tab



Select all the CBR

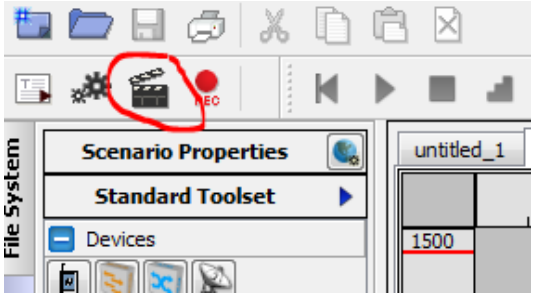


Right click go to properties

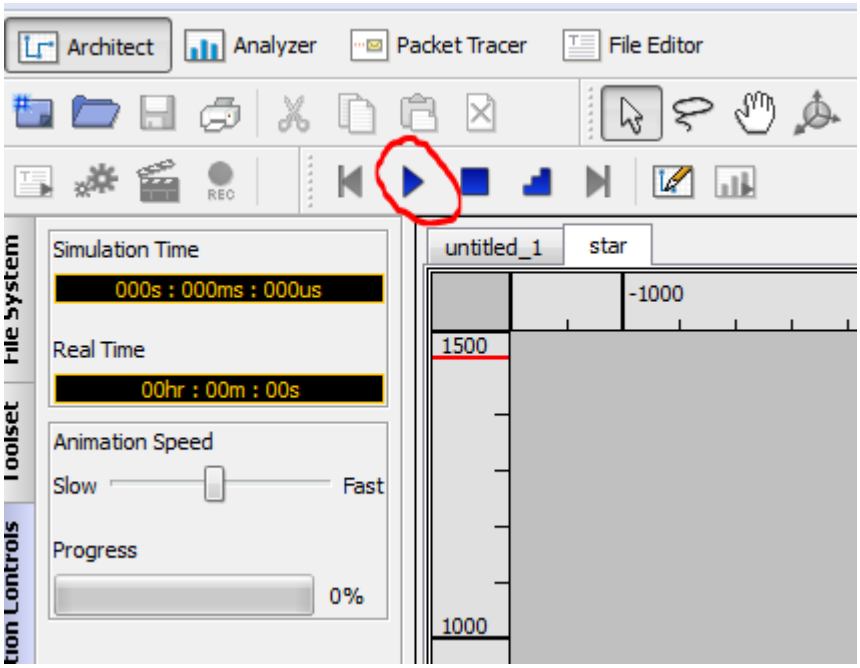


Click ok and save the scenario

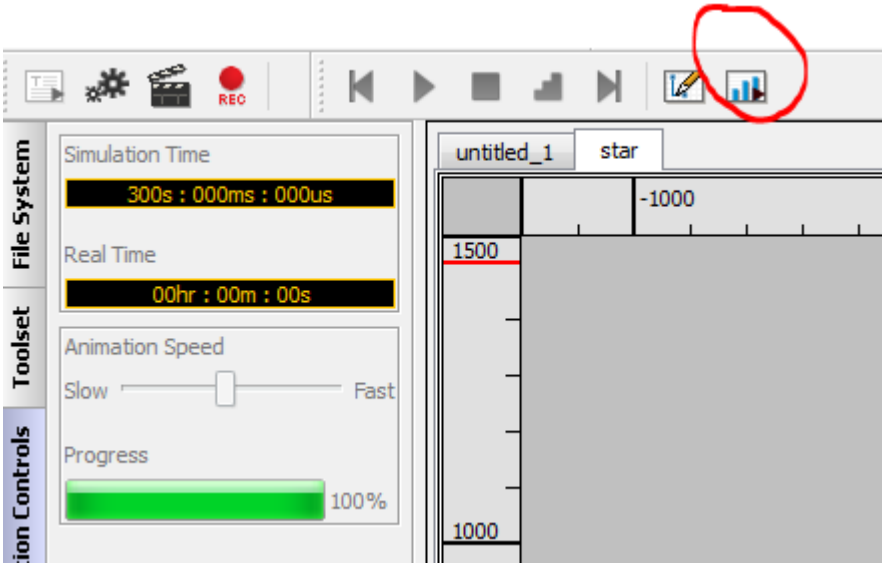
Step 4: Run the simulation

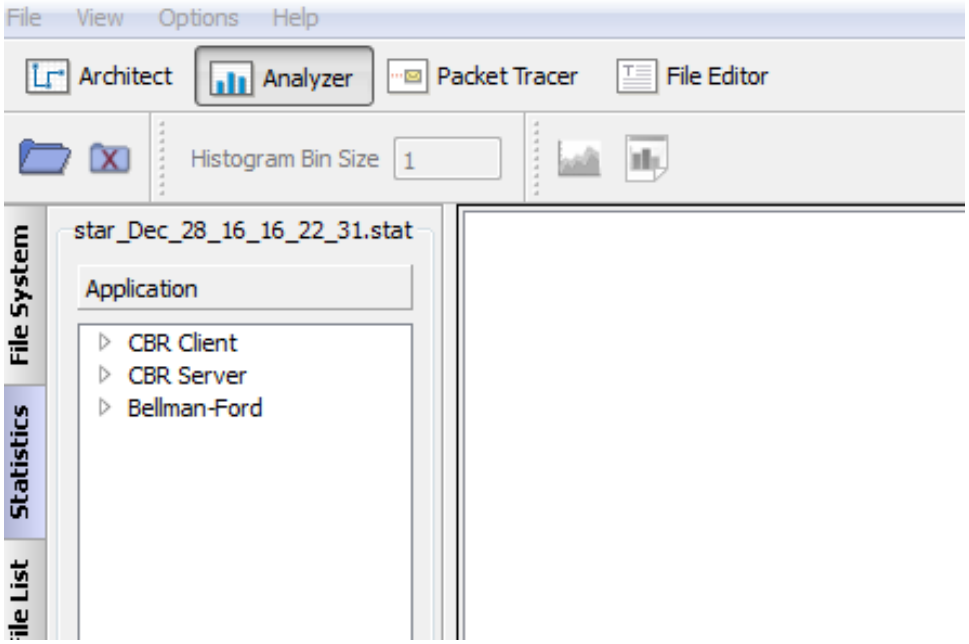


Play the scenario

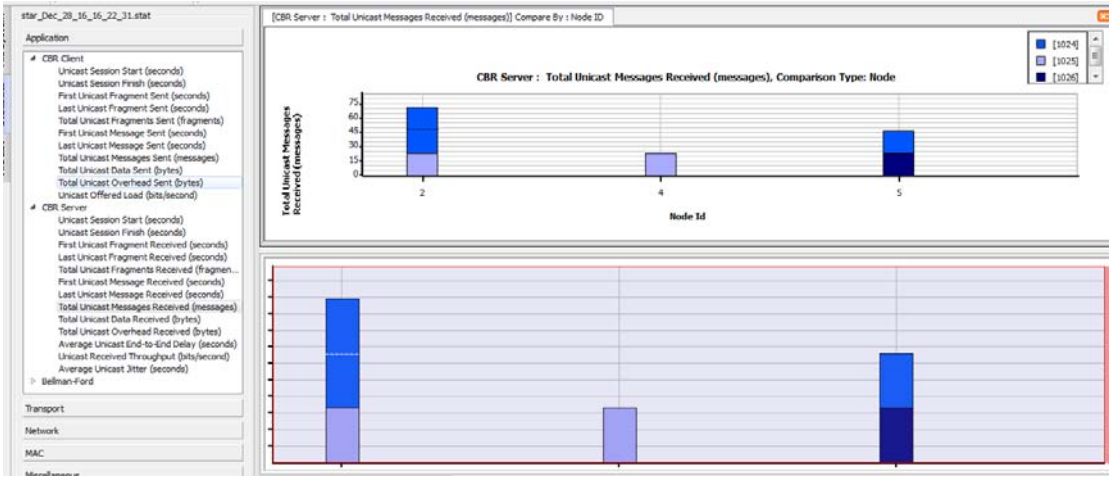


Step 5: Go to analyser mode to analyse the statistics




File
View
Options
Help
Architect
Analyzer
Packet Tracer
File Editor
Histogram Bin Size
1
File System
Statistics
star_Dec_28_16_16_22_31.stat
Application
CBR Client
CBR Server
Bellman-Ford


star_Dec_28_16_16_22_31.stat
Application
CBR Client
Unicast Session Start (seconds)
Unicast Session Finish (seconds)
First Unicast Fragment Sent (seconds)
Last Unicast Fragment Sent (seconds)
Total Unicast Fragments Sent (fragments)
First Unicast Message Sent (seconds)
Last Unicast Message Sent (seconds)
Total Unicast Messages Sent (messages)
Total Unicast Data Sent (bytes)
Total Unicast Overhead Sent (bytes)
Unicast Offered Load (bits/second)
CBR Server
Unicast Session Start (seconds)
Unicast Session Finish (seconds)
First Unicast Fragment Received (seconds)
Last Unicast Fragment Received (seconds)
Total Unicast Fragments Received (fragmen...
First Unicast Message Received (seconds)
Last Unicast Message Received (seconds)
Total Unicast Messages Received (messages)
Total Unicast Data Received (bytes)
Total Unicast Overhead Received (bytes)
Average Unicast End-to-End Delay (seconds)
Unicast Received Throughput (bits/second)
Average Unicast Jitter (seconds)
Bellman-Ford
Transport
Network
MAC
[CBR Client : Total Unicast Messages Sent (messages)] Compare By : Node ID
[1024]
[1025]
[1026]
CBR Client : Total Unicast Messages Sent (messages), Comparison Type: Node
Total Unicast Messages Sent (messages)
75
60
45
30
15
0
1
4
5
Node Id

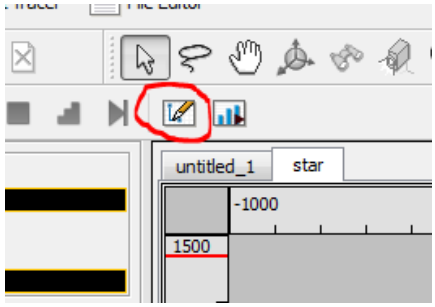


Repeat the same procedure for Ring and Mesh Topology

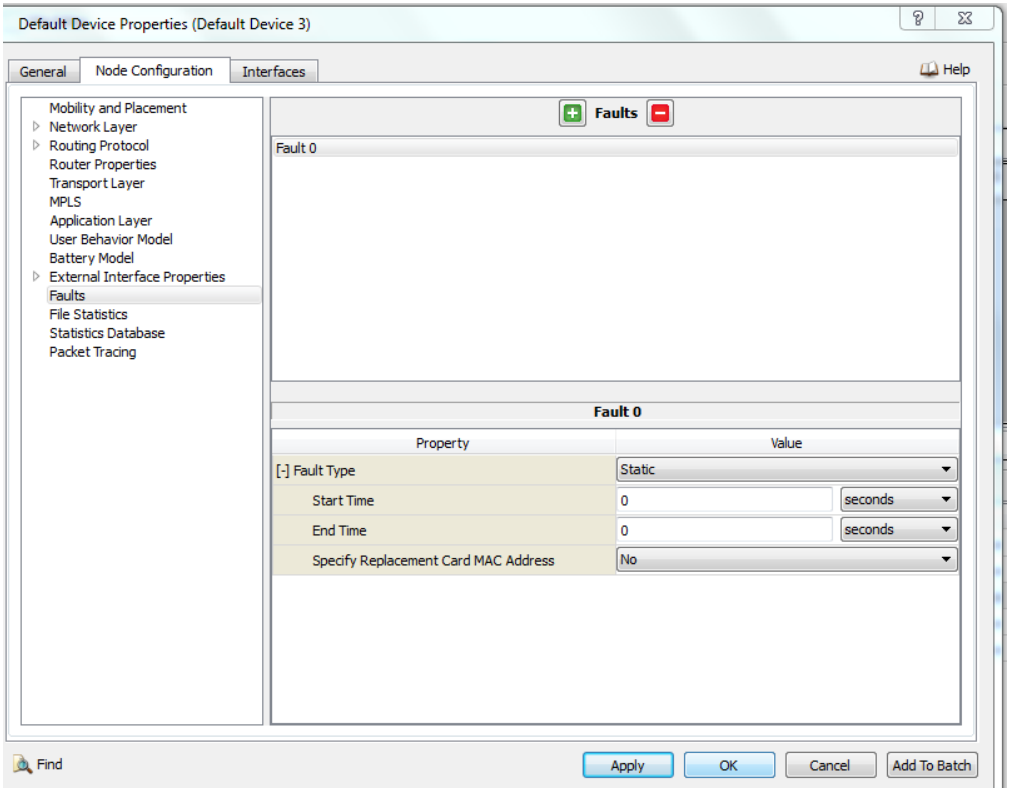
Note: Deactivate the Centre node and Analyze the statistics for all-star topology

Procedure to deactivate the Centre node

Step 6: Go to design mode

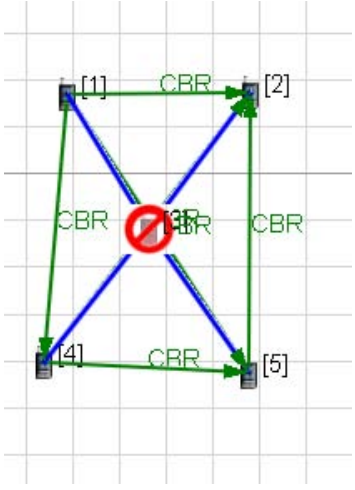


Step 7: Go to table view →Select node tab →select the node id of centre node →Edit properties

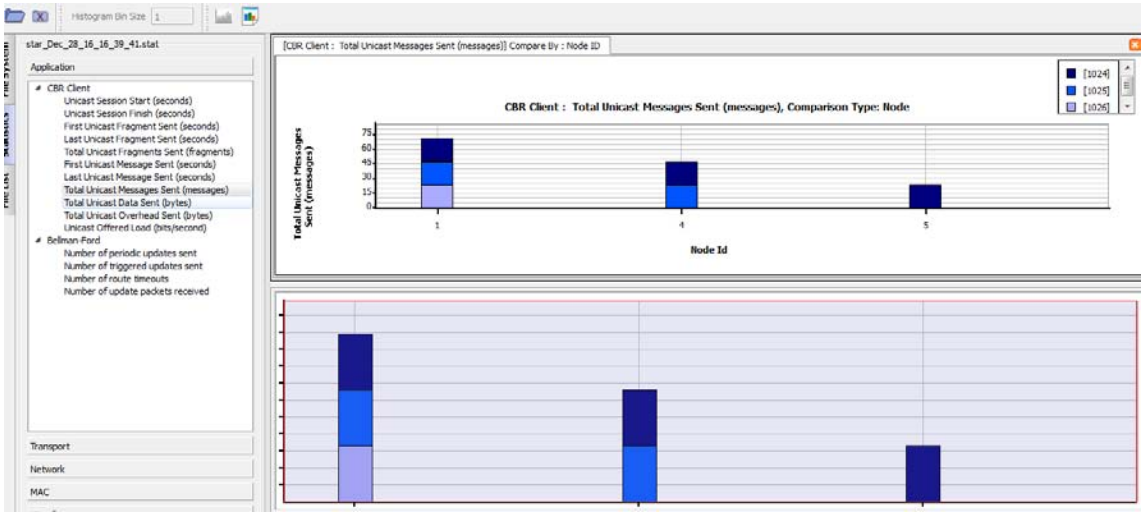


Click Apply and Ok

Step 8: Save the scenario and simulate



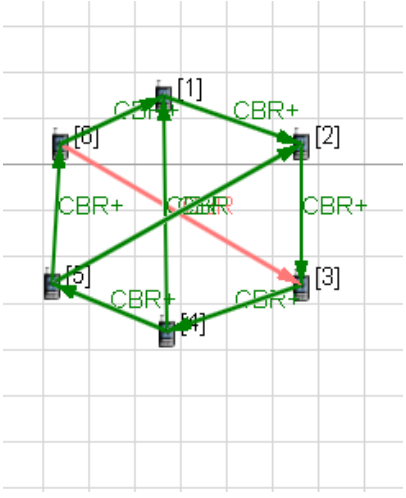
Step 9: Play the scenario→Analyze the statistics



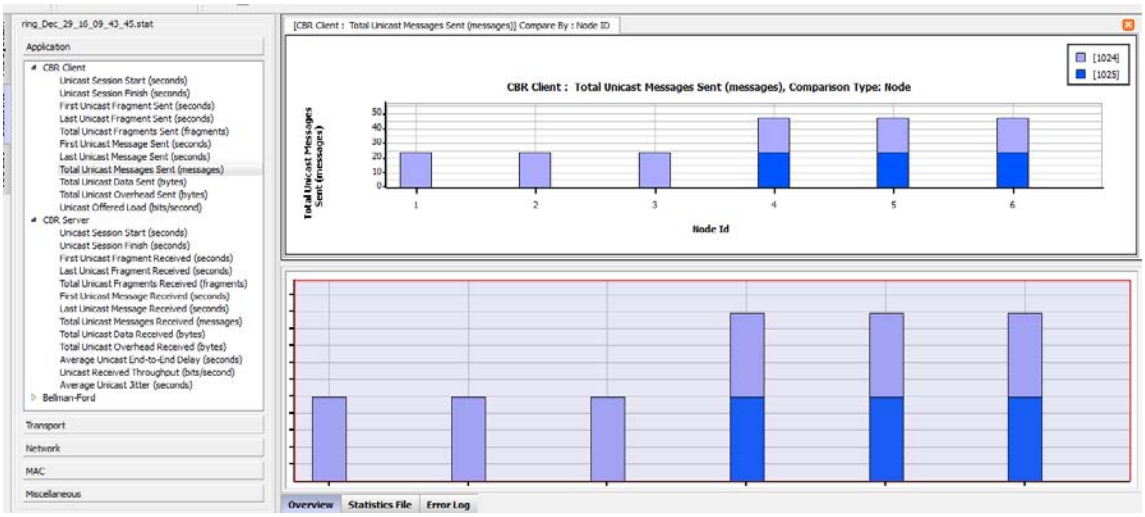
Receiver is unable to receive the data due to deactivation of Centre node.

B) Ring Topology

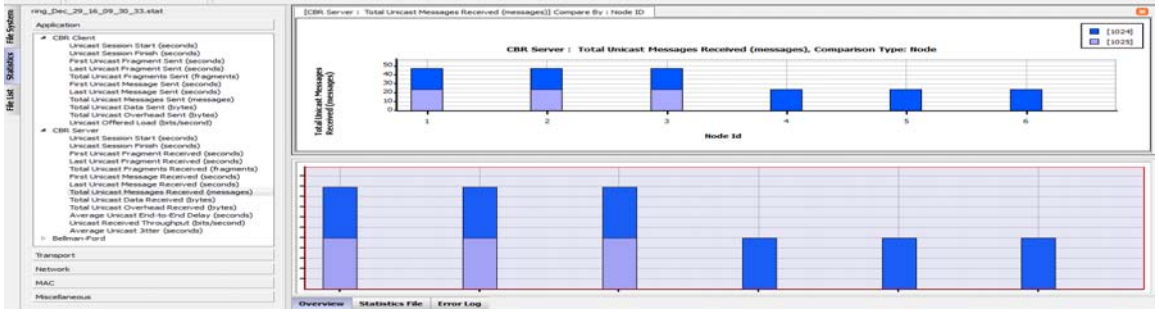
Procedure: Repeat the steps described in star topology.



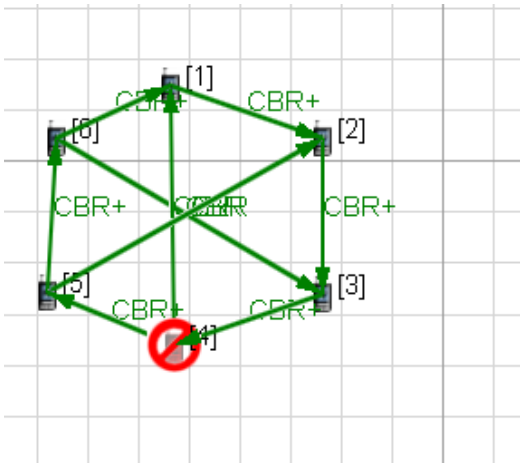
Total Unicast message sent



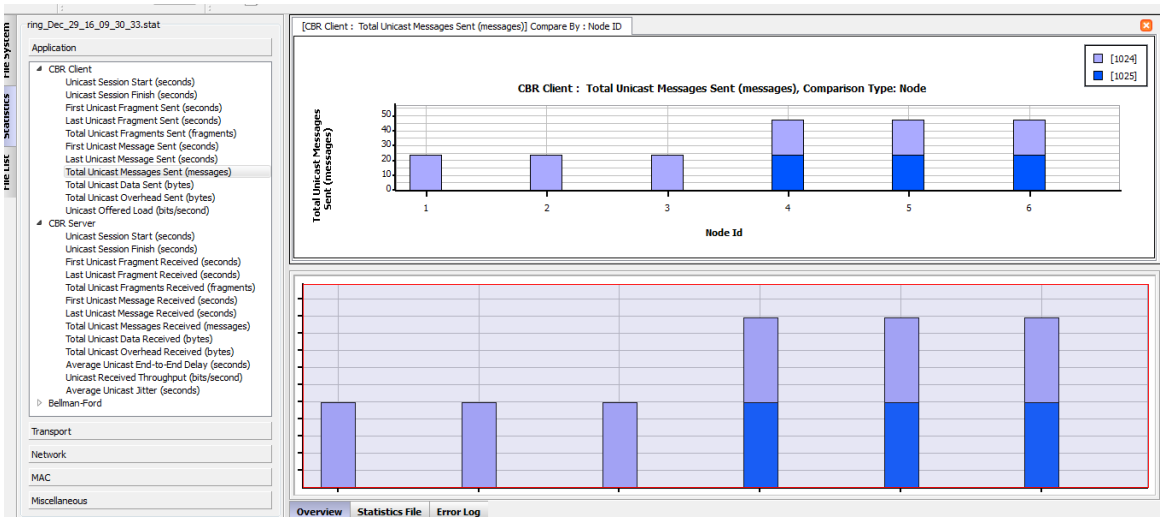
Total unicast message received



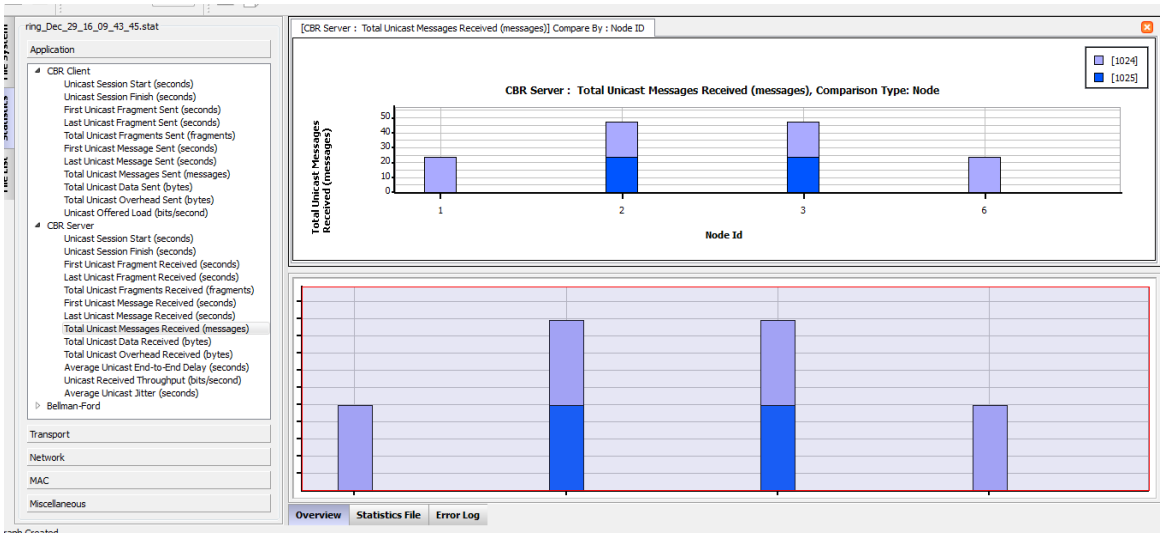
If any node is faulty in topology (4th node)



Total unicast message sent

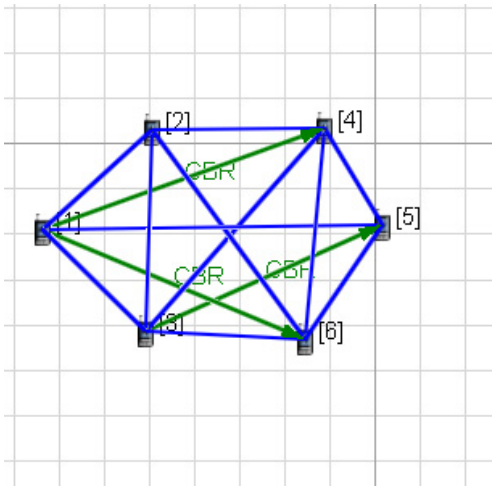


Total unicast message received

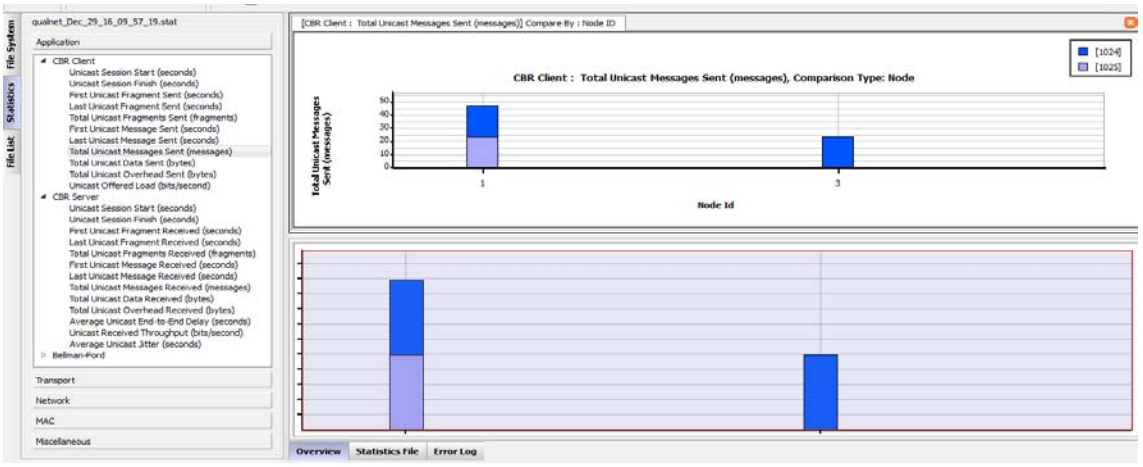


C) Mesh Topology

Procedure: Repeat the steps described in star topology.

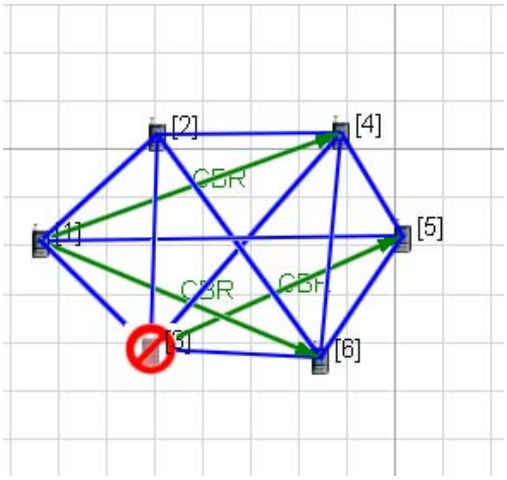


Total unicast message sent

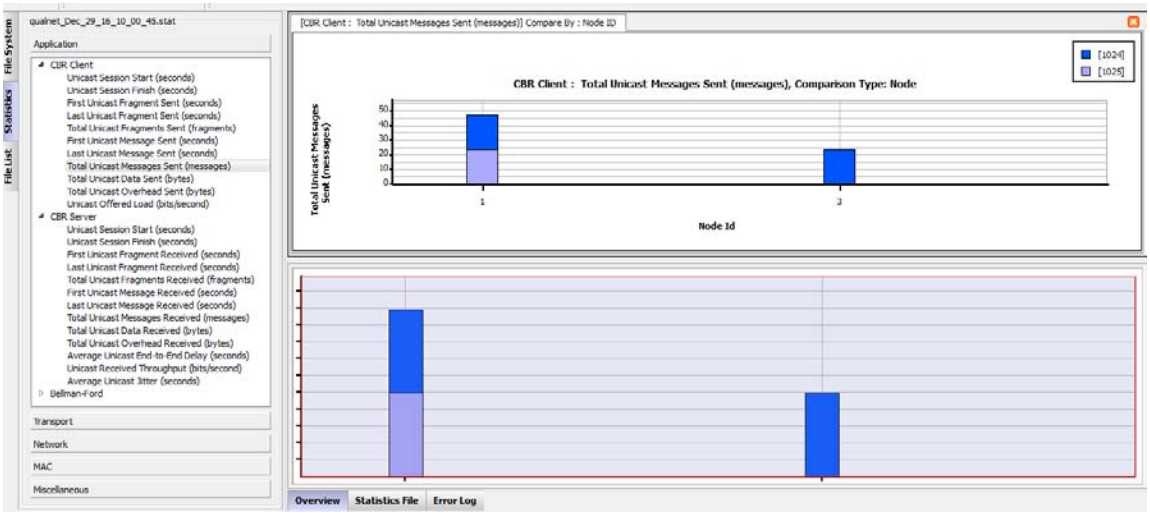


Total unicast message received

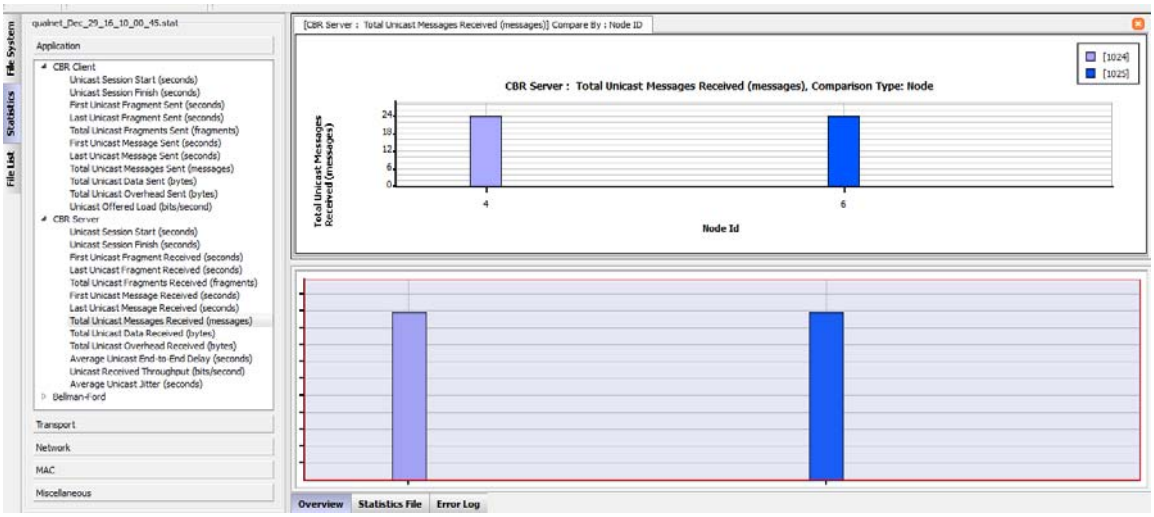
If node is faulty (3rd node is faulty)



Total message sent



Total message received



Exercise:

1. Simulate a point to point network consisting of 10 nodes and analyse the performance by introducing fault nodes.
2. Simulate an Ethernet LAN using n nodes (6-10), change data rate and compare throughput.

Experiment 3

Wired and Wireless LANs

Objective:

A) To Simulate the Ad-hoc network with 30 nodes with multiple traffic. Repeat the simulation of the network by introducing mobility (MANETs) for the nodes and compare the performance of the networks.

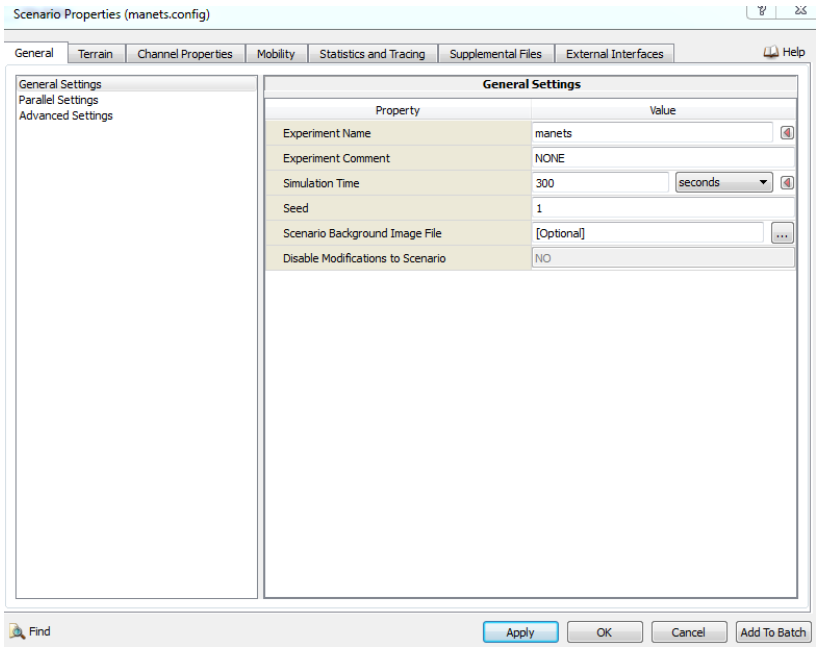
B) To Simulate the Infrastructure Basic Service Set (IBSS) network with 30 nodes with multiple traffic and analyse the performance of the network.

Procedure:

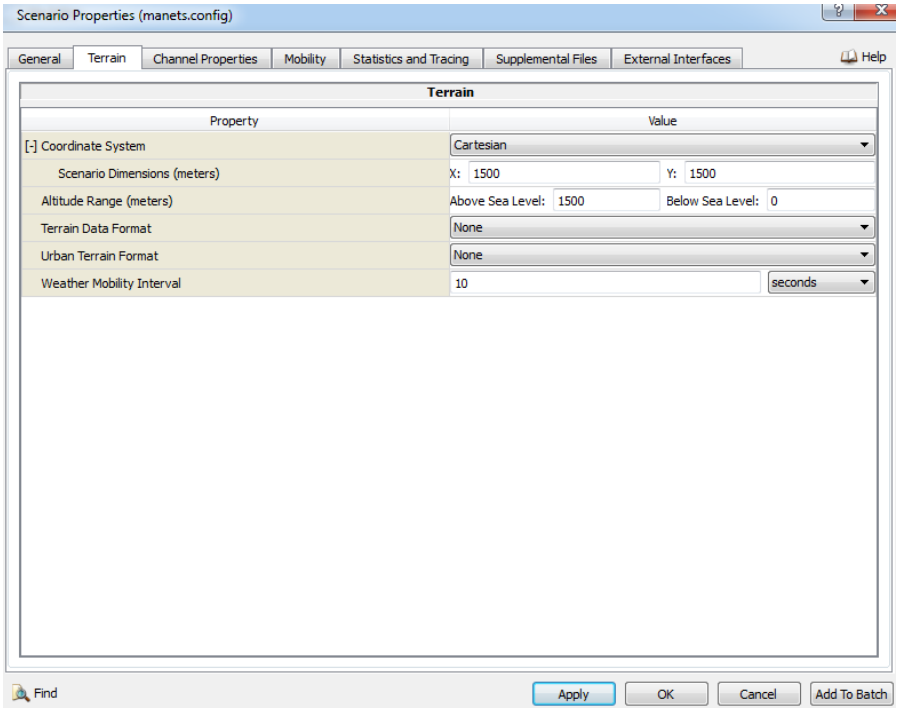
A) Go to File → New → Save as→manets

Select Scenario Properties → General Settings → Give Experiment name, Simulation Time

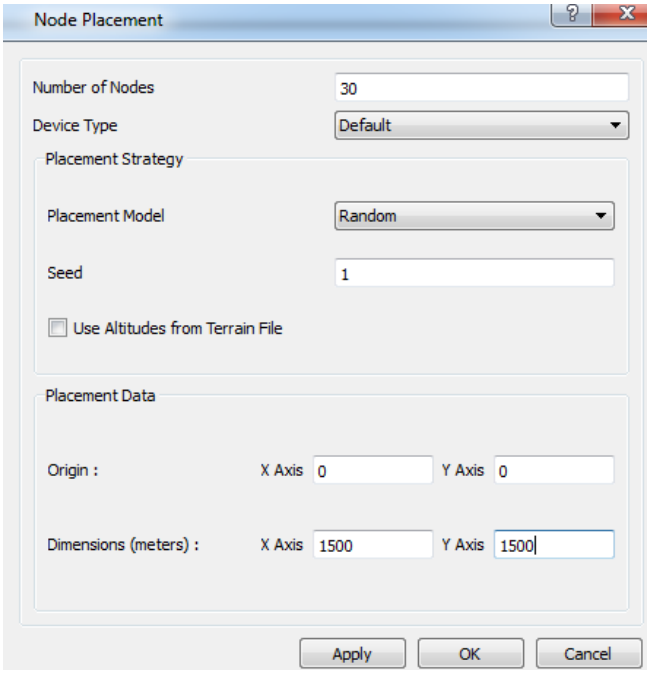
Click Apply, Ok



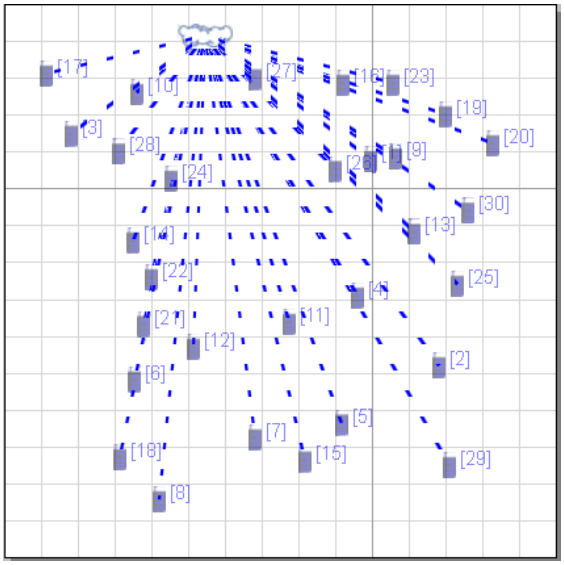
Set the Terrain Properties



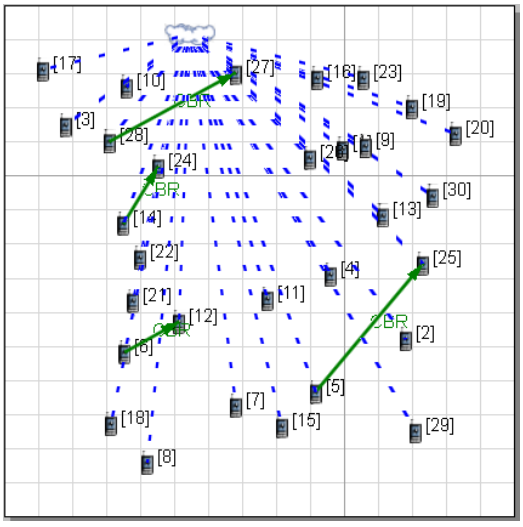
Step 1: Go to Tools → Node Placement



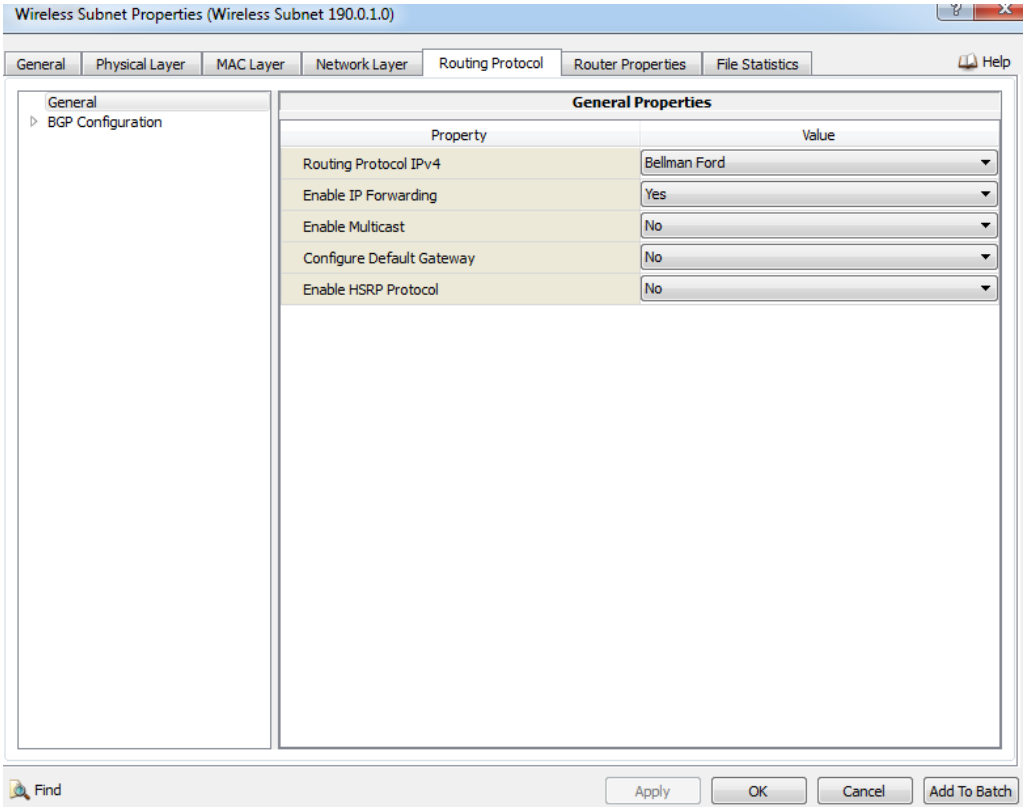
Select all the nodes, under **Network components** --> select “Wireless Network” place in canvas.



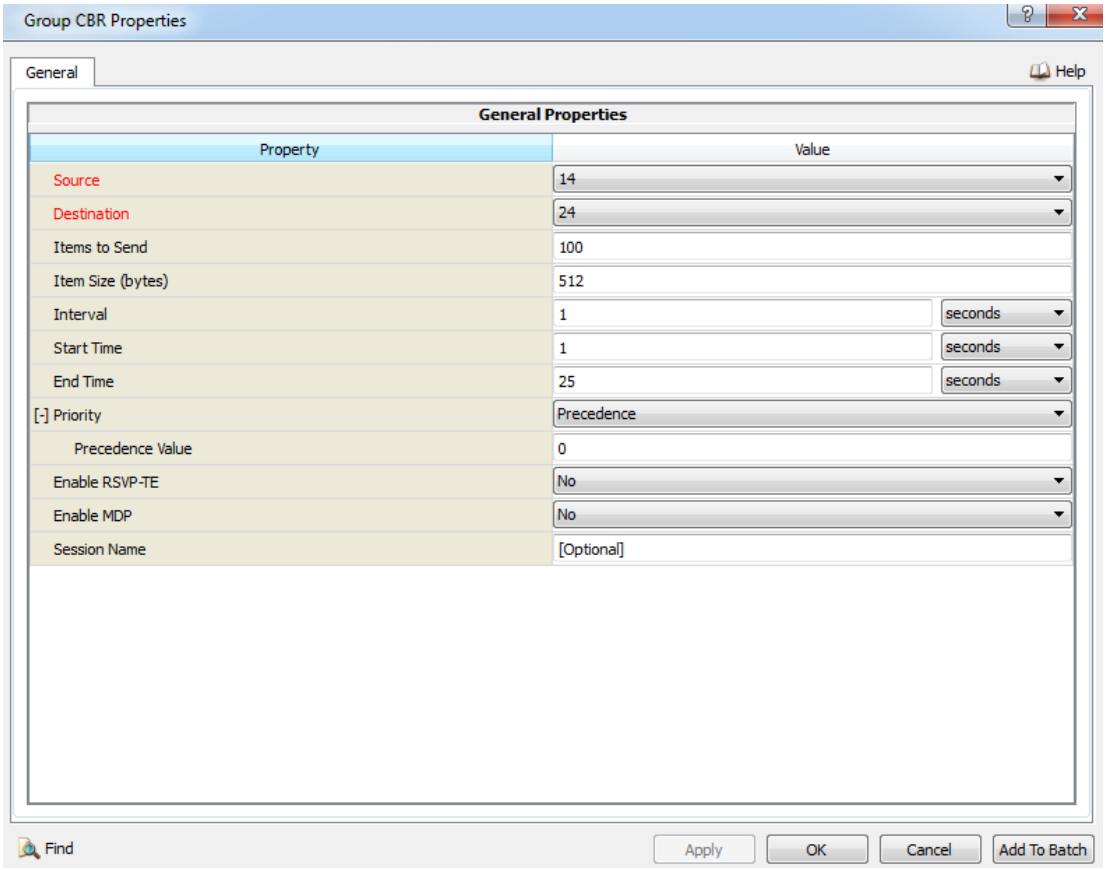
Step 2: Create CBR Traffic between some pair of nodes.(under Applications → select CBR)



Step 3: Go to Table view →Networks
Select Wireless subnet, right click, properties
Select the Routing Protocol as Bellman Ford

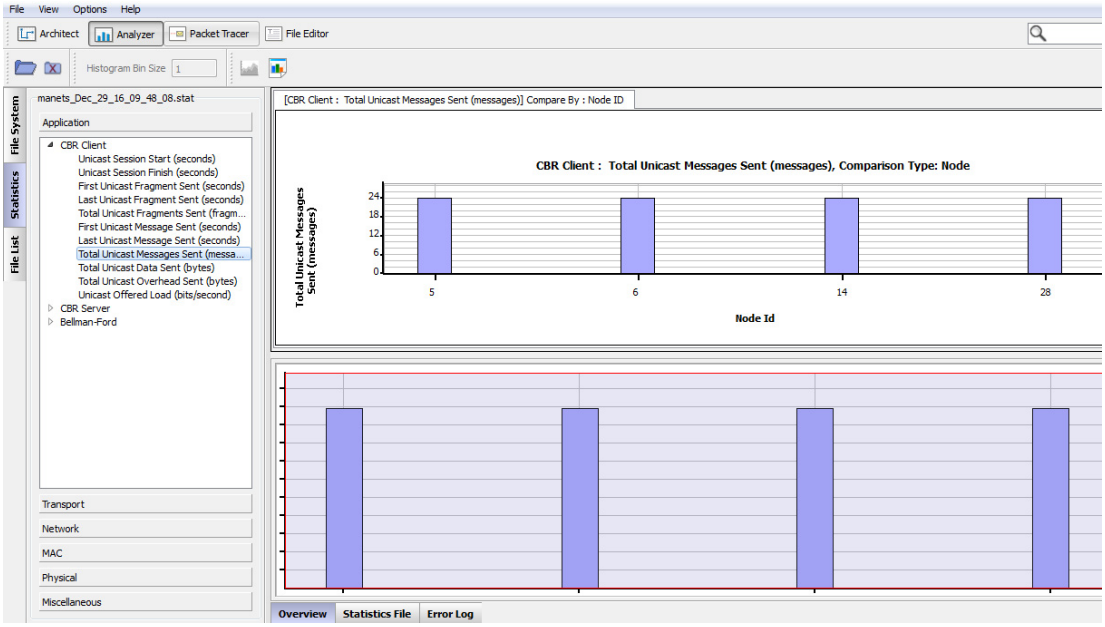


Step 4: Go to Table view → Applications
Select all the CBR sources, Right click, properties



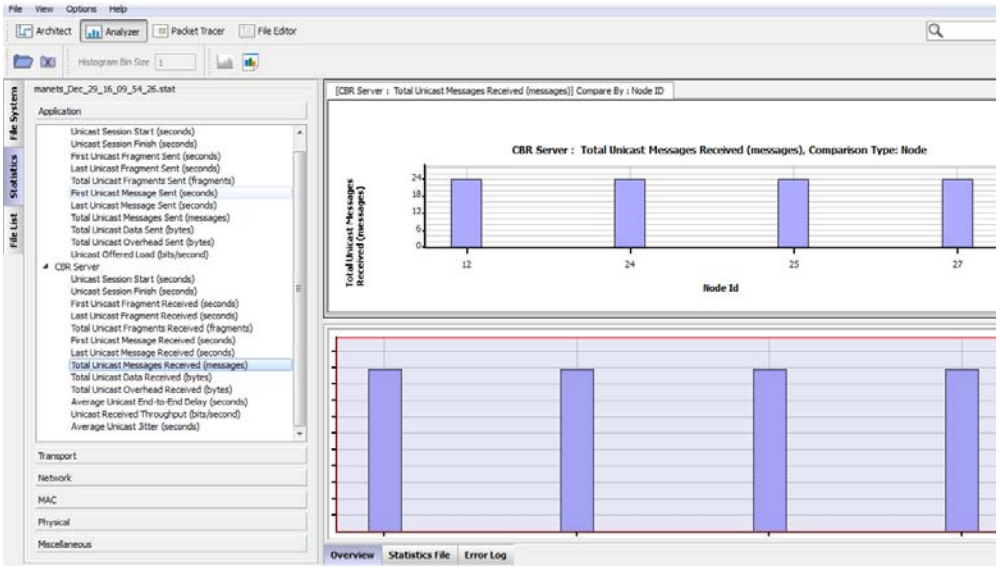
Step 5: Save the Scenario, Run Simulation, Play, Analyze Statistics of Current Scenario.

Observe “Total Unicast Messages sent”.



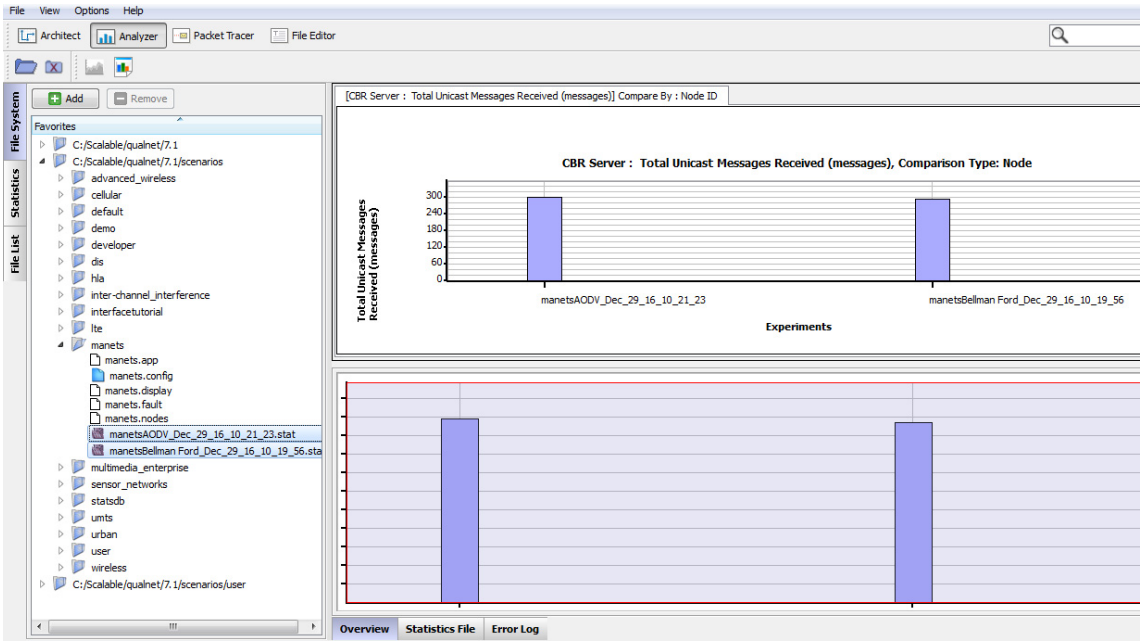
Similarly, observe “Total Unicast Messages Received”, Throughput, End-to-End Delay, and Jitter.

Step 6: Change the Routing Protocol to AODV and repeat the procedure.
Compare the statistics (Total Unicast Messages received, End-to-End Delay, Throughput, Jitter etc.)

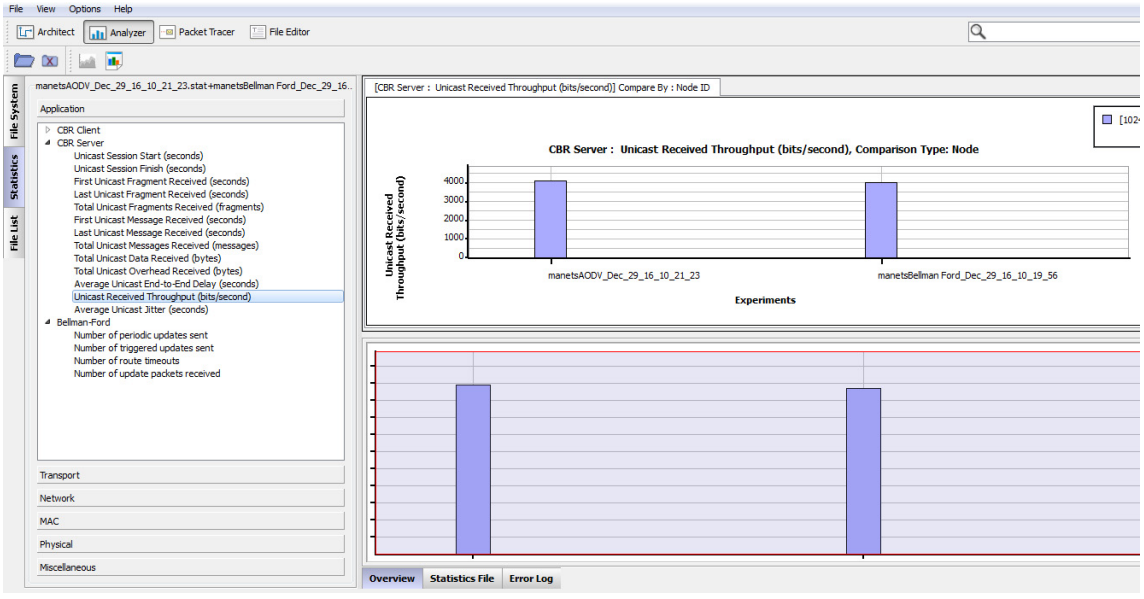


Step 7: We can also compare the performance of the AODV and Bellman Ford Protocols as follows:
Give the different experiment name (scenario Properties → General → Experiment name) for each Routing Protocol (Table view → Networks → Routing Protocol)
After Run Simulations

Go to File system, select the statistics files (.stat) in the Experiment name folder (path: qualnet/7.1/scenarios/user/manets) Right click, Analyze.

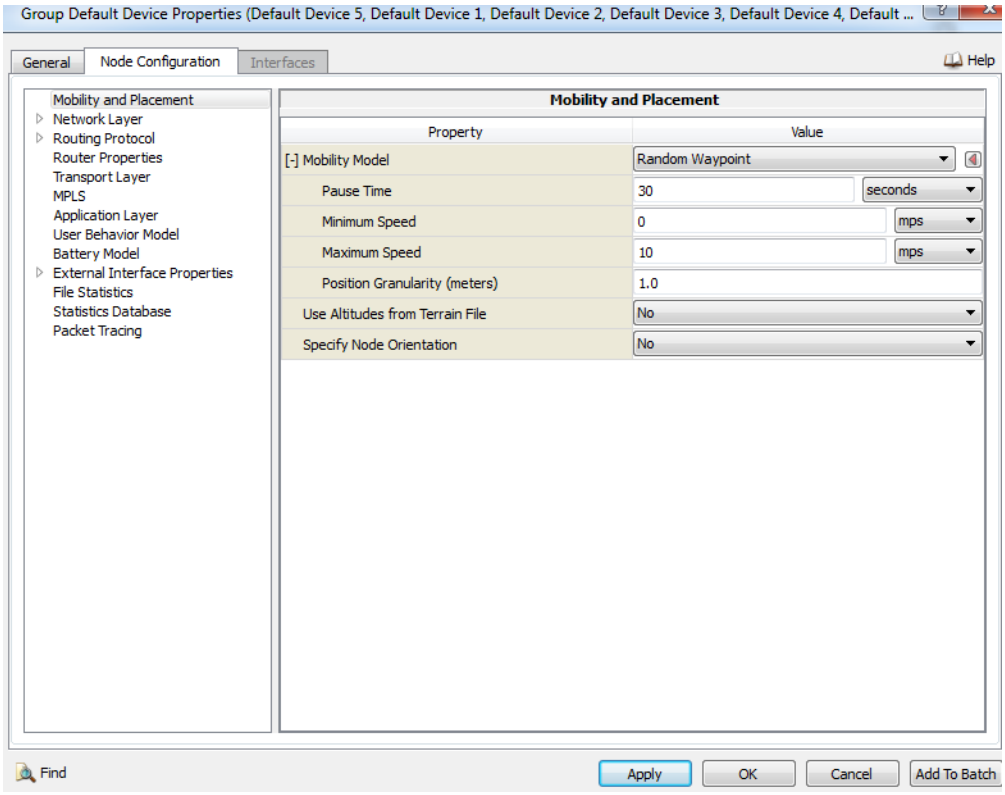


Go to Statistics; compare the End-to-End Delay, Throughput and Jitter.



With Mobility (MANETs):

Step 8: In the same scenario, Go to Table view → Nodes →select all the nodes, right click, properties → Node Configuration → Mobility and Placement → Mobility Model → Random Waypoint

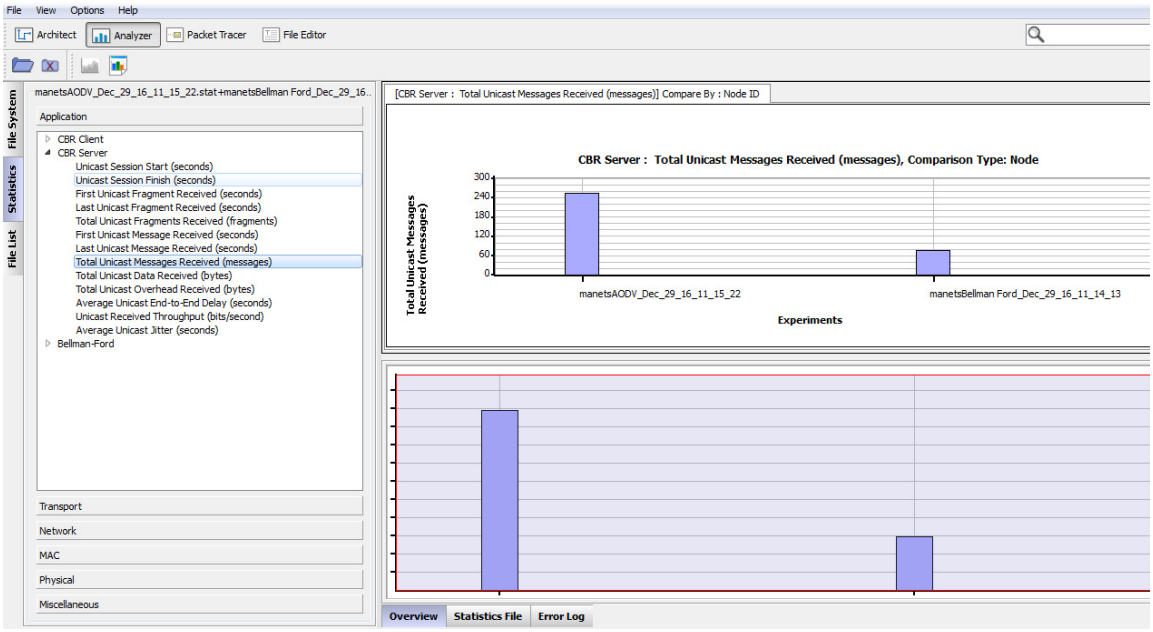


Step 9: Save the Scenario, Run Simulation, Play, Analyze Statistics of Current Scenario.

Repeat the procedure for AODV Routing Protocol. (Change the scenario name to "manetsAODV")

Step 10: Go to File system → select the both the statistics files (.stat), right click Analyze.

Go to Statistics, compare the Total Unicast messages received, End-to-End Delay, Throughput, Jitter etc.



B) **Procedure:**

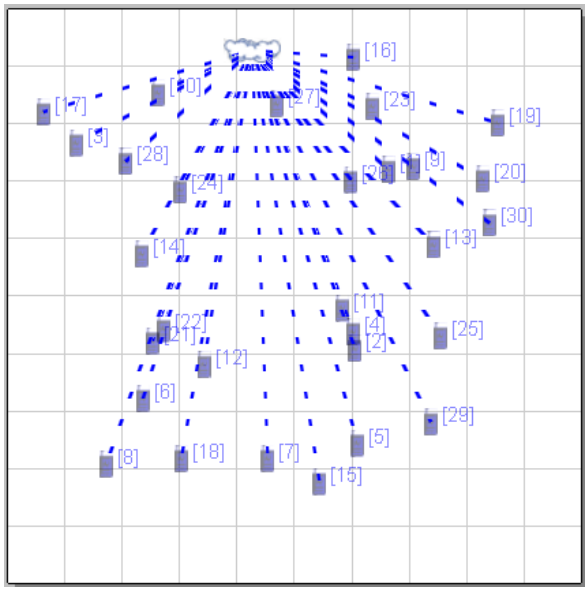
Create a folder with your registration number under "C://qualnet/7.4/scenarios/user/"

Go to File→new→save as→wifiaccesspoint

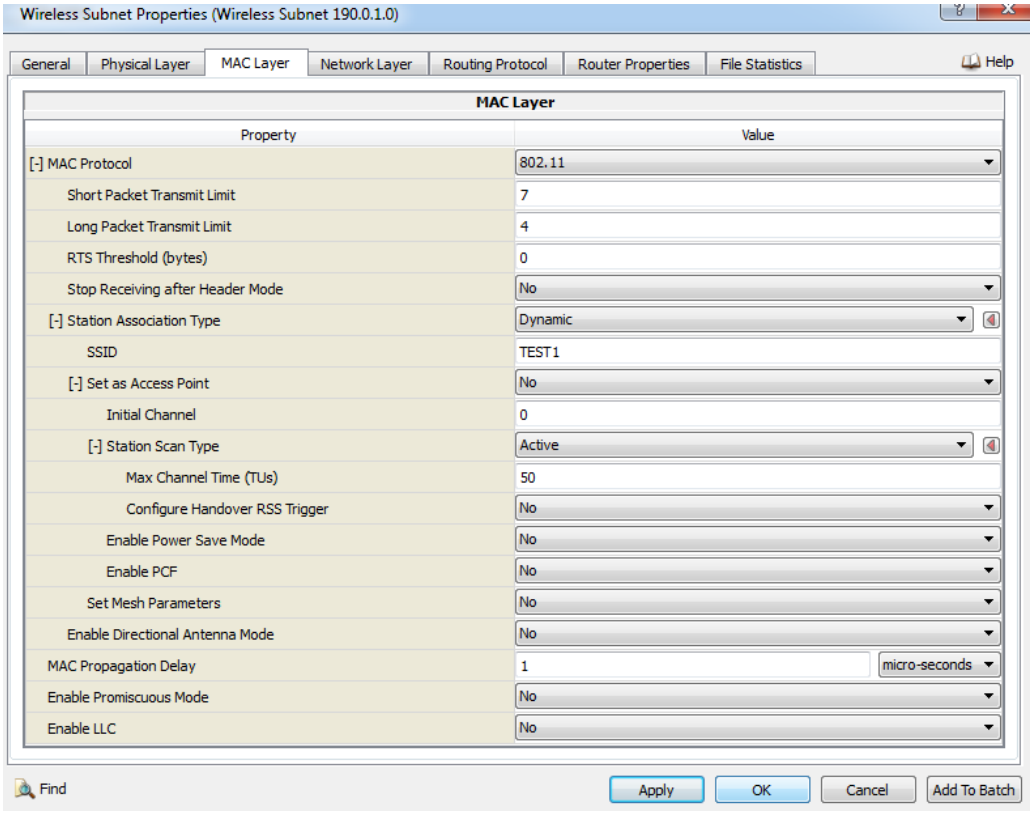
Go to Scenario Properties→General Settings → Give Experiment name, Simulation Time: 300 Seconds

Terrain → scenario Dimensions →1000 x 1000 meters

Step 1: Go to Tools → Node placement → Number of nodes → 30 Click Apply, Ok Select all the nodes, under Network Components→ select Wireless Network, and drag on to canvas.

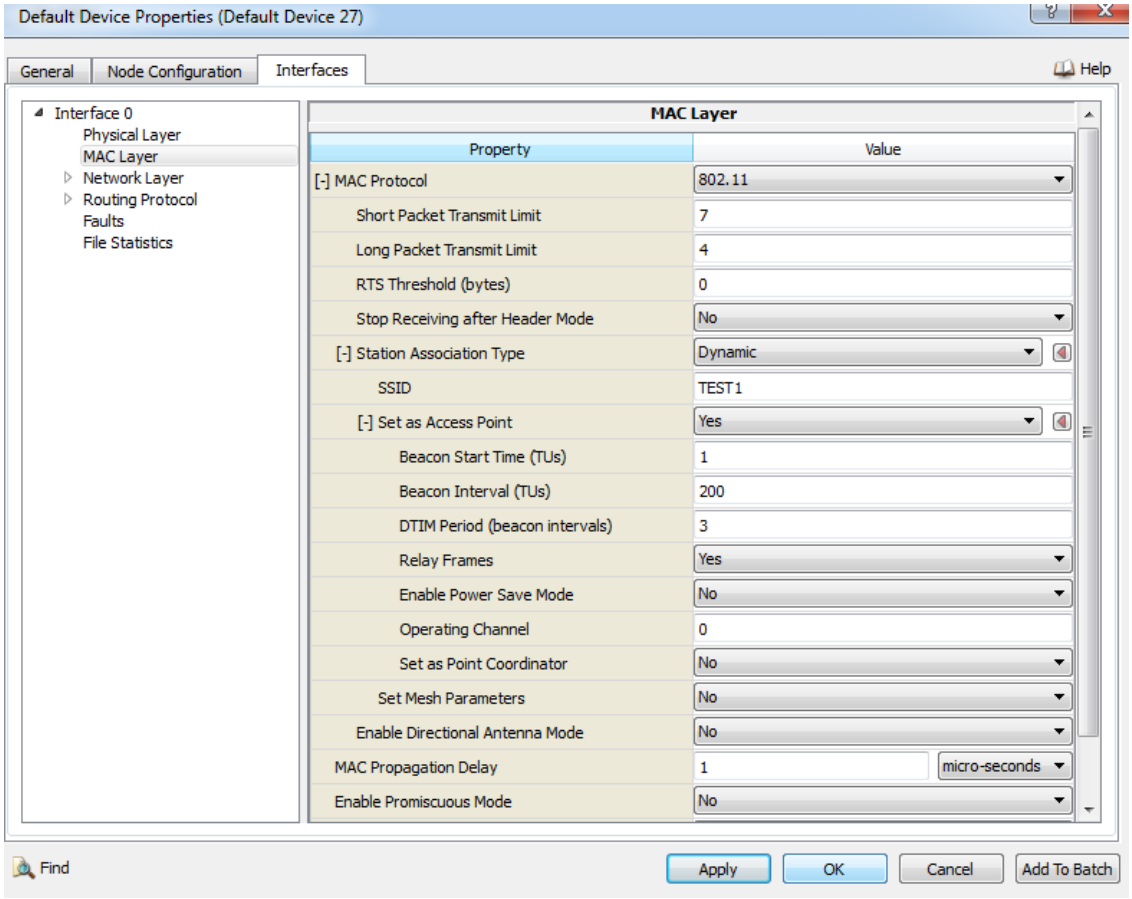


Step 2: Go to Table view → Networks, right click, properties → MAC Layer → Station Association Type→ Dynamic Station Scan Type → Active

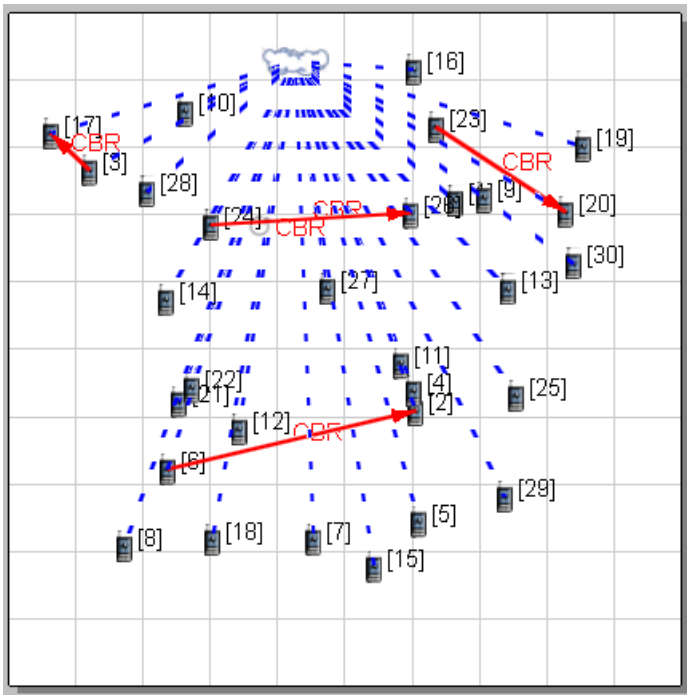


Step 3: To set Access point:

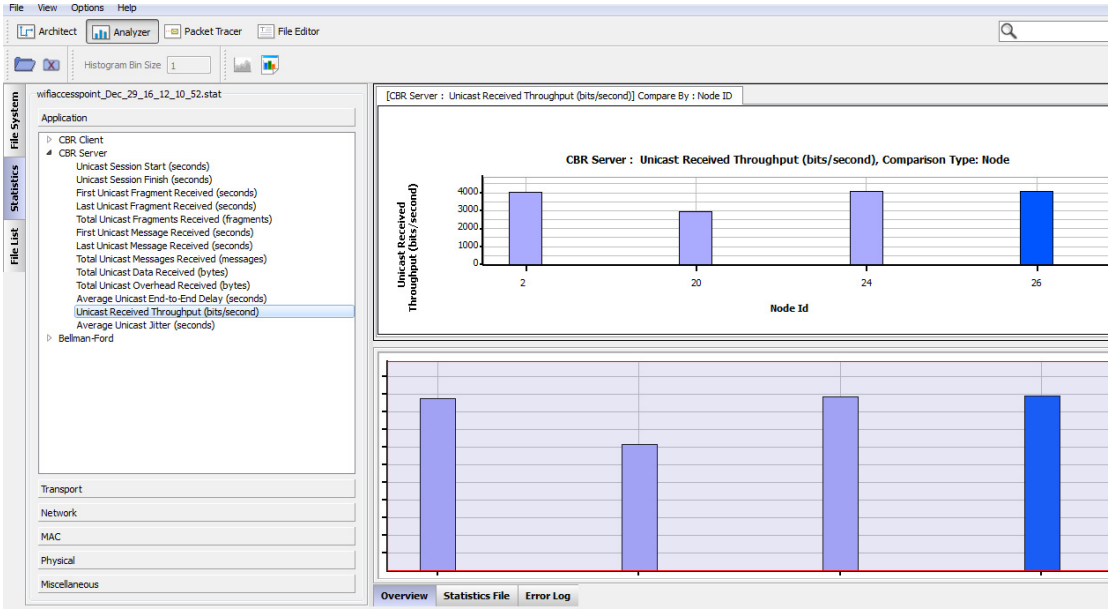
Select one of the nodes (which you want to set as access point) in the canvas, right click, properties, interfaces, MAC Layer → Set as Access point → YES



Step 4: Create CBR Traffic between few pairs of nodes.

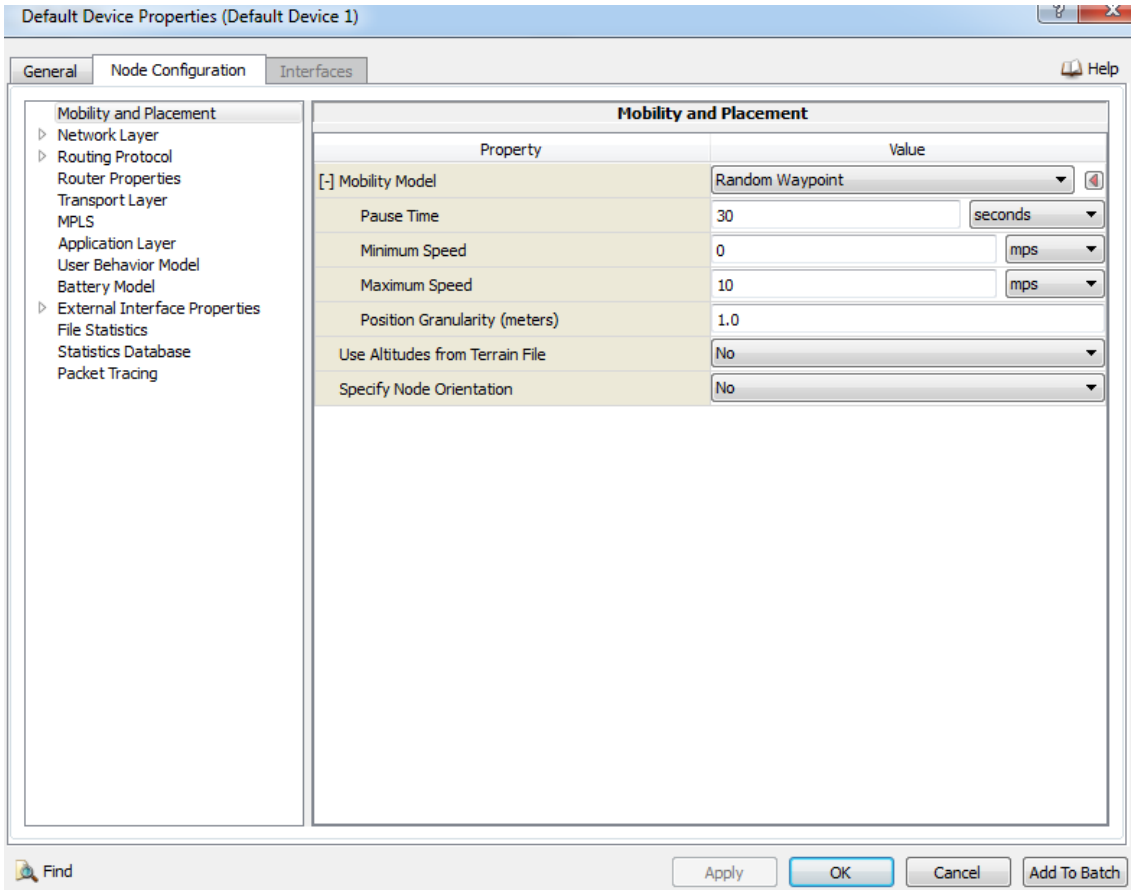


Step 5: Save the scenario, Run Simulation, Play and Analyze Statistics of Current Scenario.

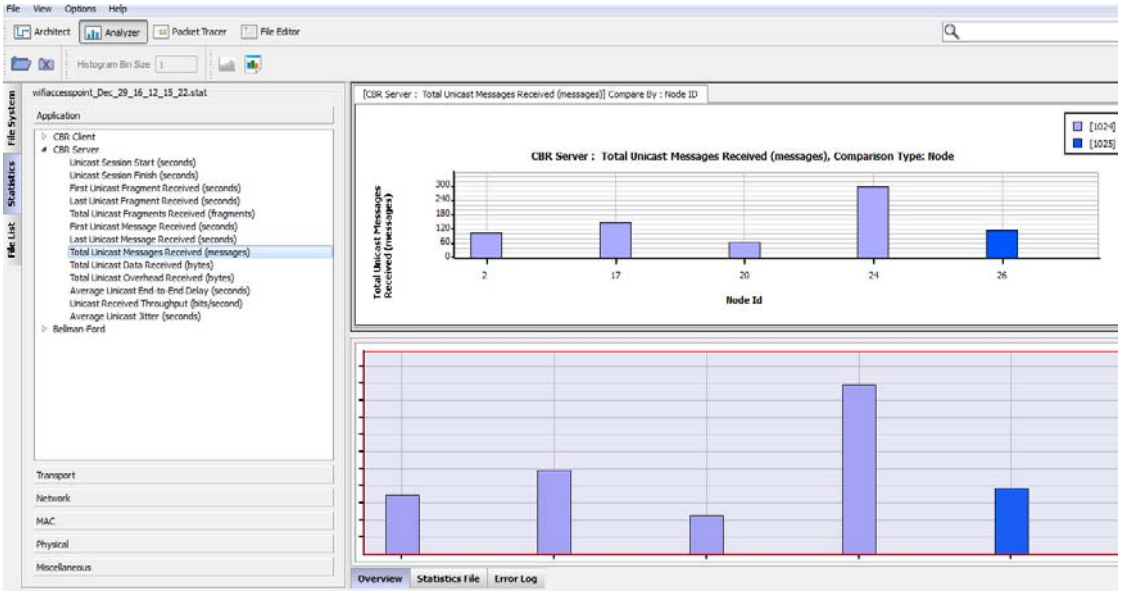


Step 6: For creating Mobility for the Nodes:

Go to Table View → Nodes → Select all the nodes, right click, properties → Node Configuration → Mobility and Placement → Mobility Model → Random Way Point



Step 7: Save the Scenario, Run Simulation, Play, Analyze Statistics of Current Scenario Observe the "Total Unicast Messages Sent" at Client.



Similarly, observe "Total Unicast Messages Received", Throughput, End-to-End Delay, and Jitter.

Exercise 1: Simulate CSMA/CA network and analyse the performance over CSMA network.

Experiment 4

Wireless Sensor Networks & Wi-Max Networks

Objective:

A) To simulate a cluster based Wireless Sensor Network (WSN) with 18 nodes, two cluster heads and a base station (i.e., PAN coordinator) and analyse the performance.

B) To Simulate a Wi-Max network with two base stations and 10 nodes in each cell. Analyse the performance with multiple traffics.

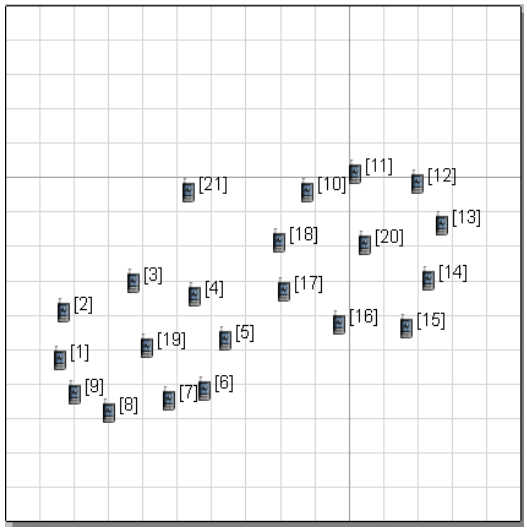
Procedure:

A) Go to File →new→save as→wsnnetworks

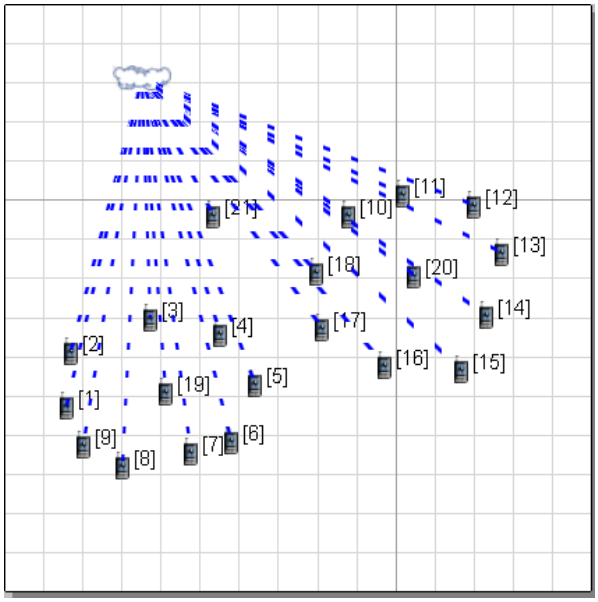
Select Scenario Properties → General Settings → Give Experiment Name, Simulation Time:300 Seconds

Terrain → Scenario Dimensions →1000 x 1000 meters

Step 1: Select "default" device (present under Standard Tool Set) and place the nodes on canvas as follows:

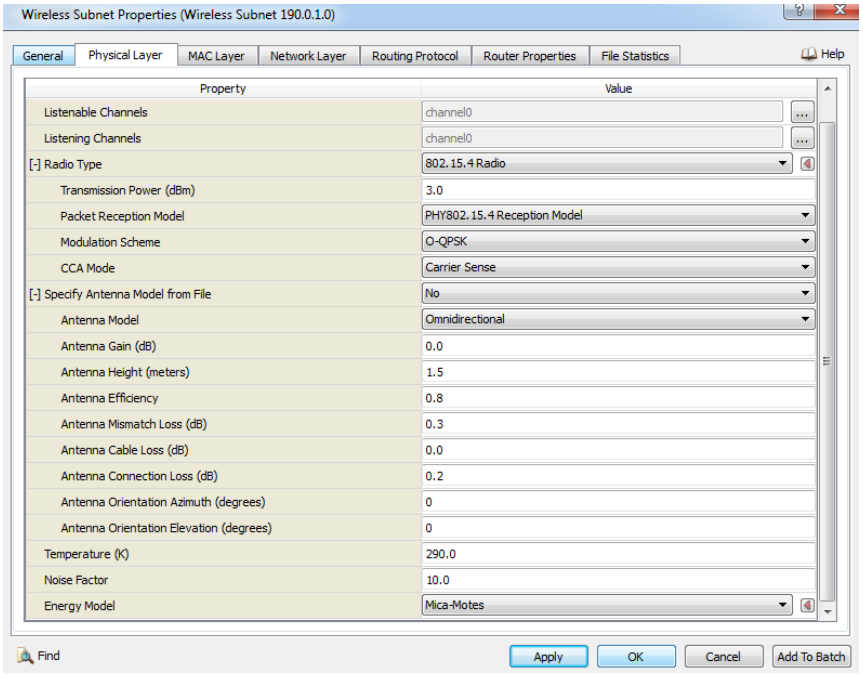


Select all the nodes in the Canvas, select Wireless Network (under Network Components) and place on the Canvas.

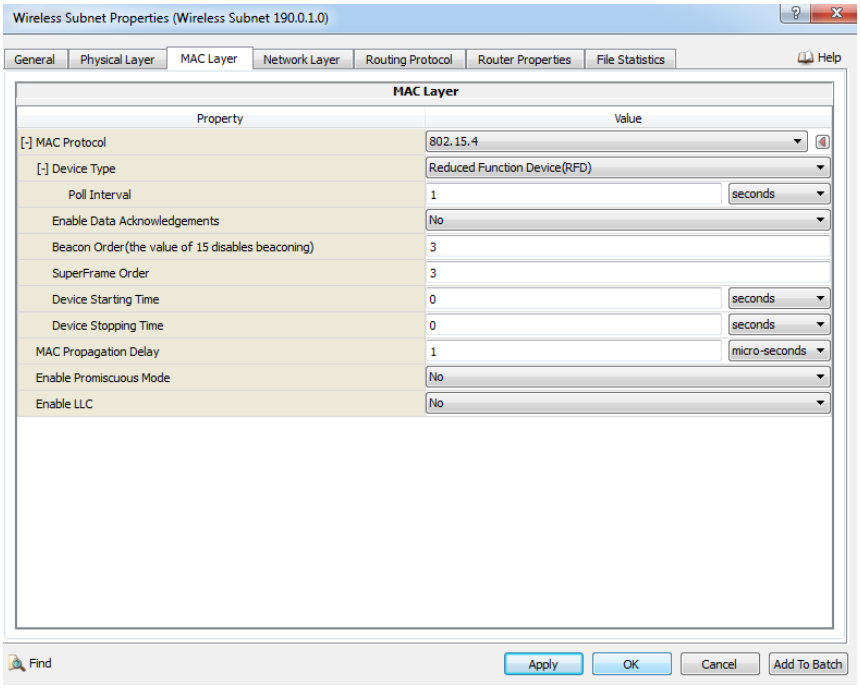


Step 2: Go to Table View → Networks → select subnet properties right click, properties

Under Physical Layer → Radio Type: 802.15.4 Radio, Energy Model: Mica-Motes



MAC Layer: change MAC Protocol: 802.15.4

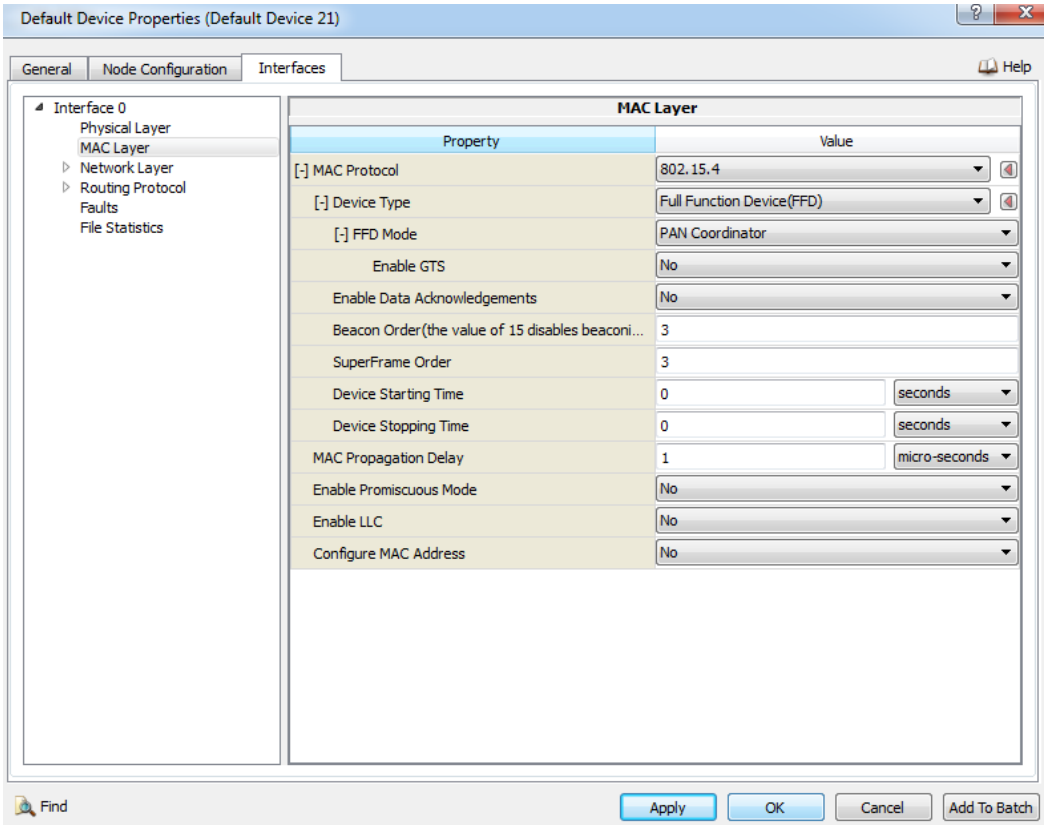


Step 3: To make node as Coordinator:

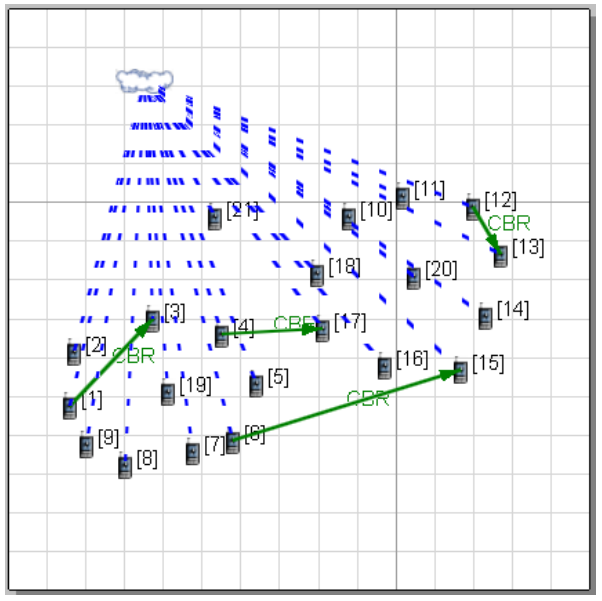
Select the two nodes on Canvas (which you want to make the coordinators), right click, properties, Node Configuration → Interfaces → MAC Layer → Device Type: Full Function Device (FFD), FFD Mode: coordinator.

Step 4: To make a Node as PAN Coordinator:

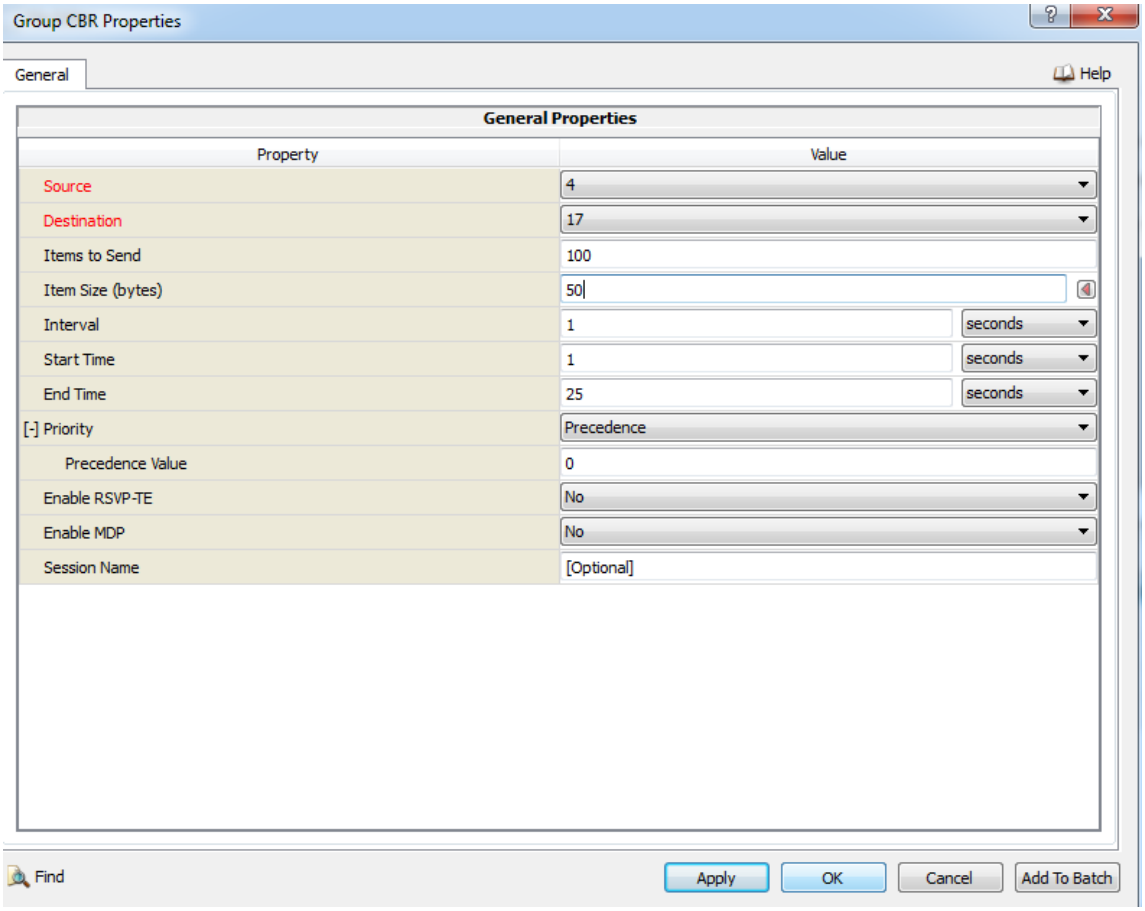
Select the node on Canvas (which you want to make the PAN coordinator), right click, properties, Node Configuration → Interfaces → MAC Layer → Device Type: Full Function Device (FFD), FFD Mode: PAN coordinator.



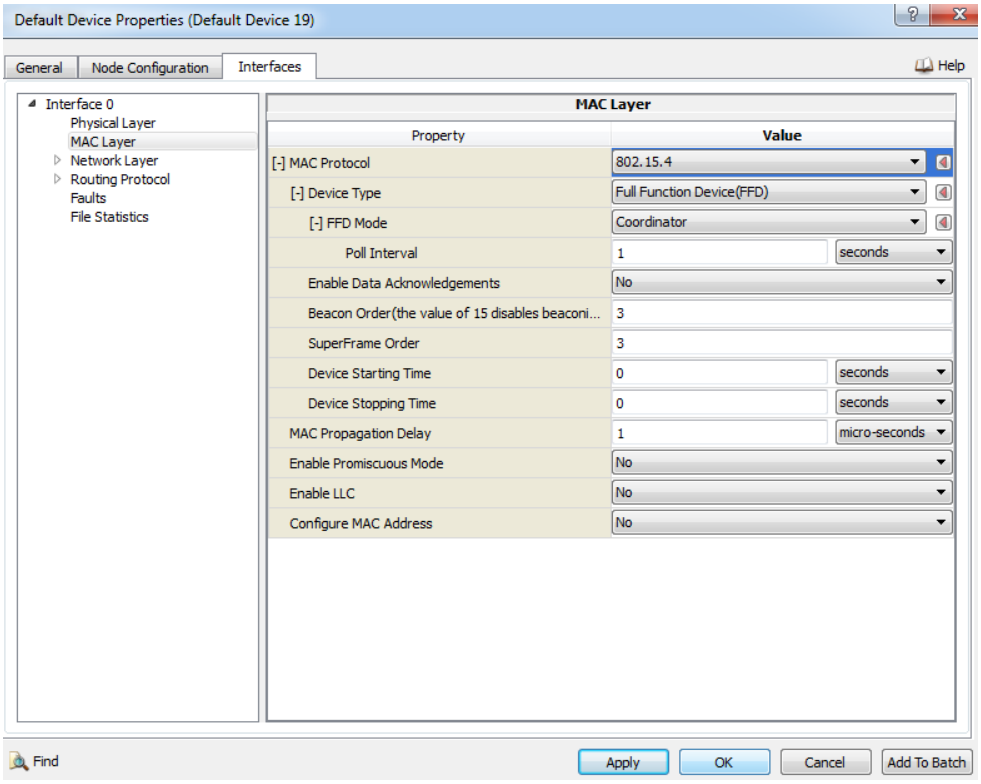
Step 5: Create CBR Traffic: Select few pairs of nodes and give CBR Traffic.



Step 6: Go to Table view → Applications → select all the CBR connections→ right click, properties → Item Size (bytes): 50 (it must be less than 70 bytes)



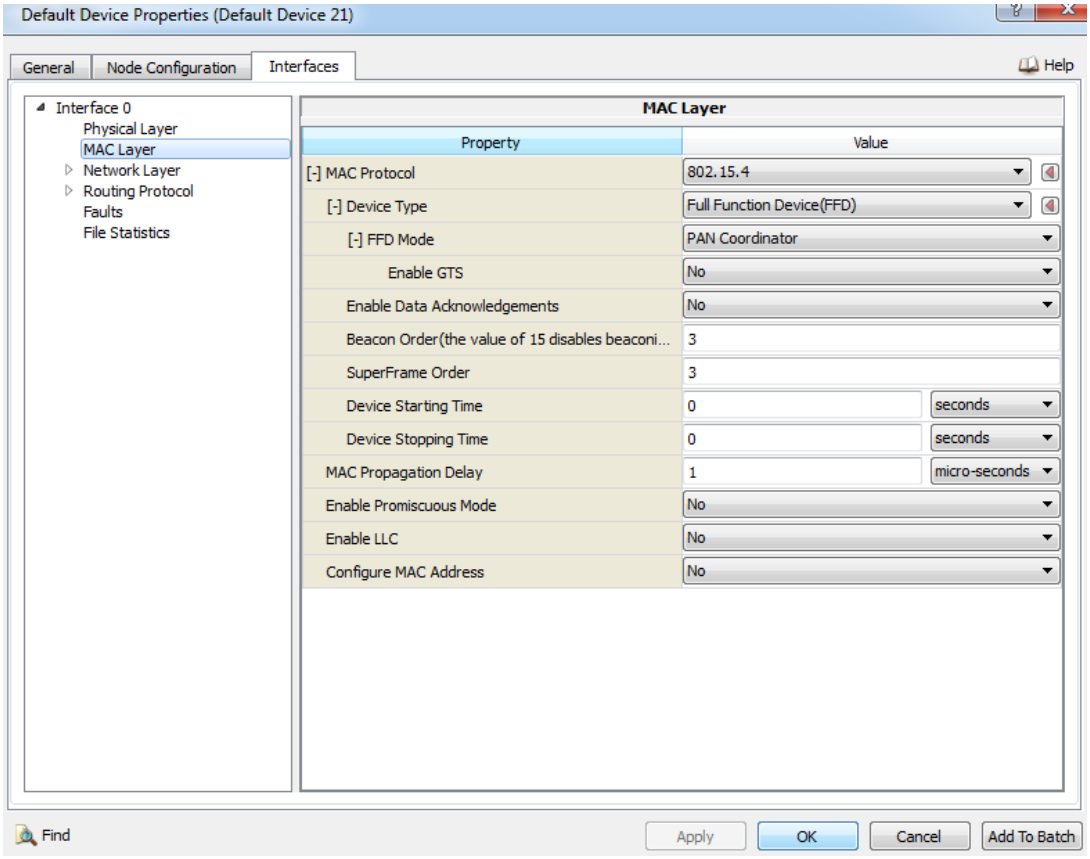
Step 7: Save the Scenario, Run Simulation, Play and Analyze Statistics of Current Scenario.



Step 8: To make a node as PAN Coordinator:

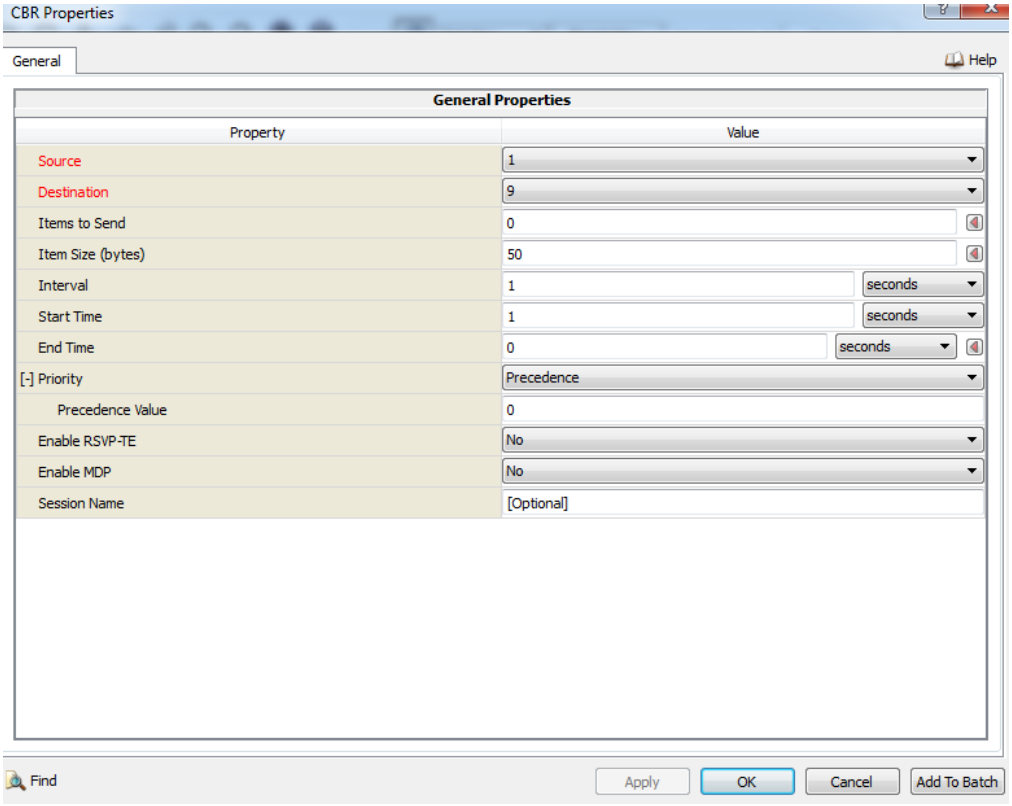
Select the node, right click, properties

Interfaces → MAC Layer → Device Type: Full Function Device (FFD), FFD Mode: PAN Coordinator

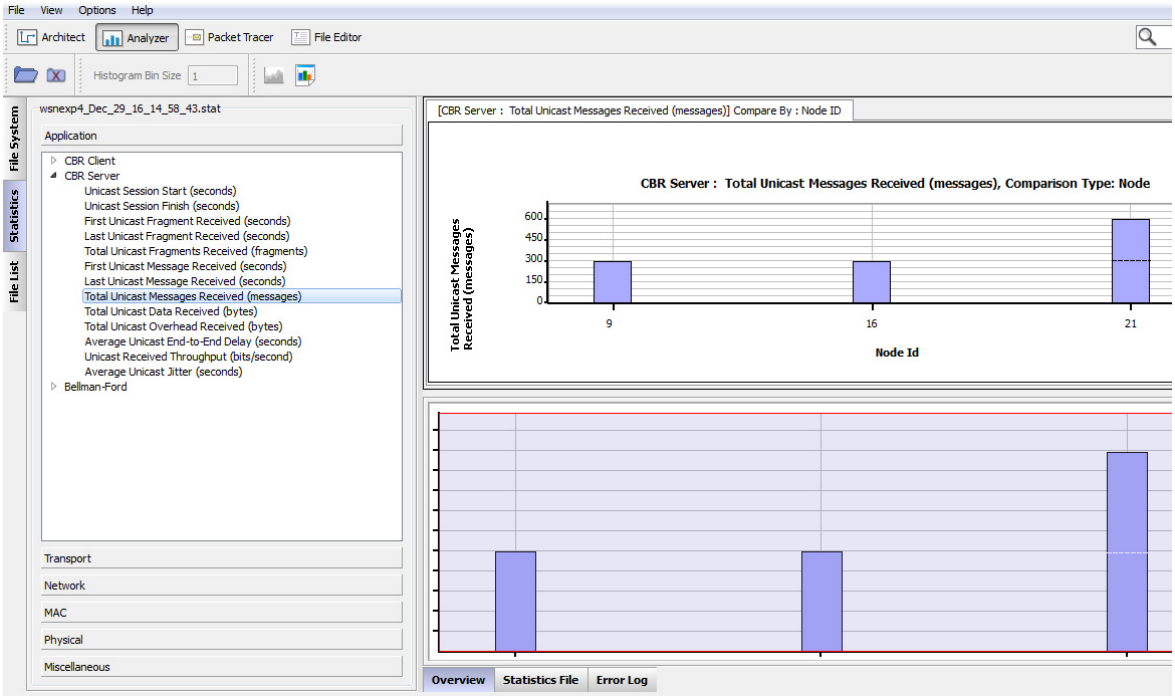


Step 9: Create CBR Traffic between some pairs of nodes.

Go to Table view →Applications → select all the CBR applications, right click, properties → Items to send: 0, Item size (bytes):50, End Time: 0



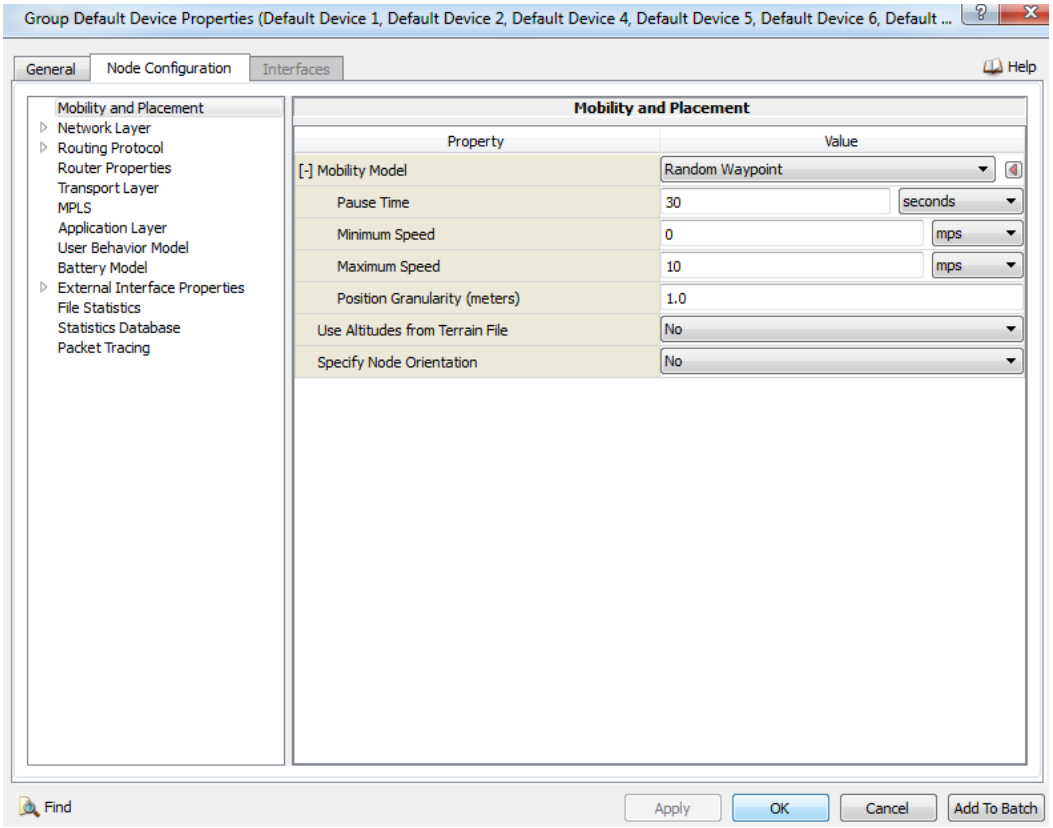
Step 10: Save the Scenario, Run Simulation, Play, and Analyze Statistics of Current Scenario. Verify the Total Unicast Messages received at server, End-to-End Delay, Throughput, and Jitter.



To create Mobility for Sensor Nodes:

Go to Table view → Nodes → select all the nodes, right click, properties → Node Configuration → Mobility and Placement

Mobility Model: Random Waypoint



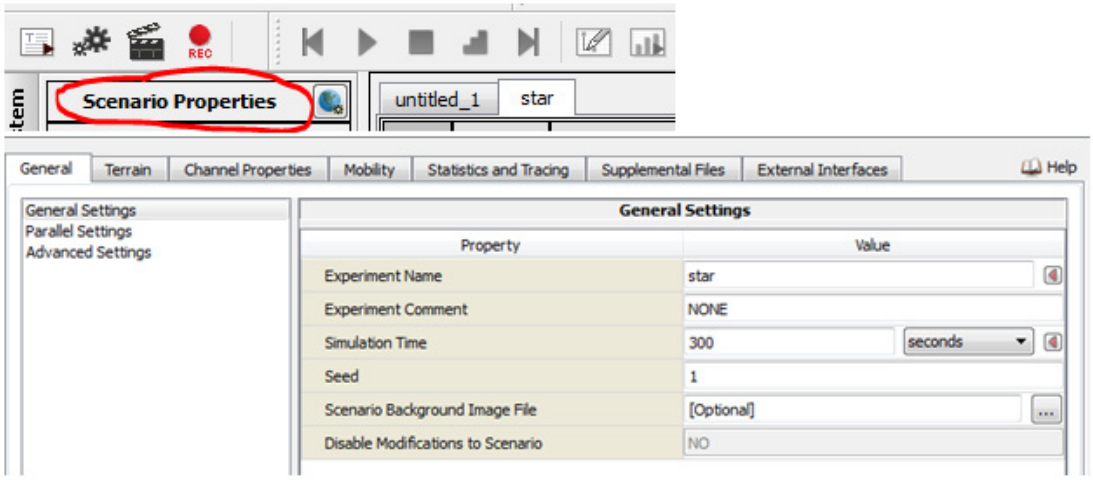
Save the Scenario, Run Simulation, Play and Analyze Statistics of Current Scenario.

Observe the Total Unicast Messages Received, End-to-End Delay, Throughput, and Jitter.

B) **Procedure:**Go to file → New → Save as →WiMax

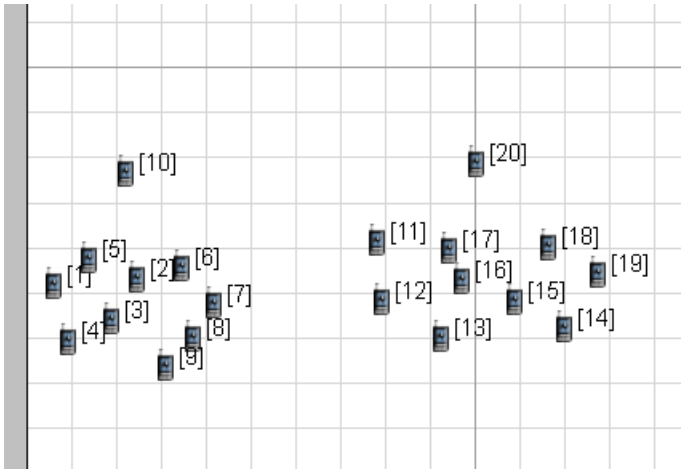
Go to scenario properties → General setting → Give experiment name and simulation time

Click Apply, Ok

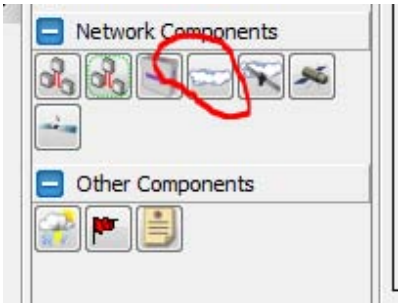


Keep the default value in other fields
Click Apply and Ok.

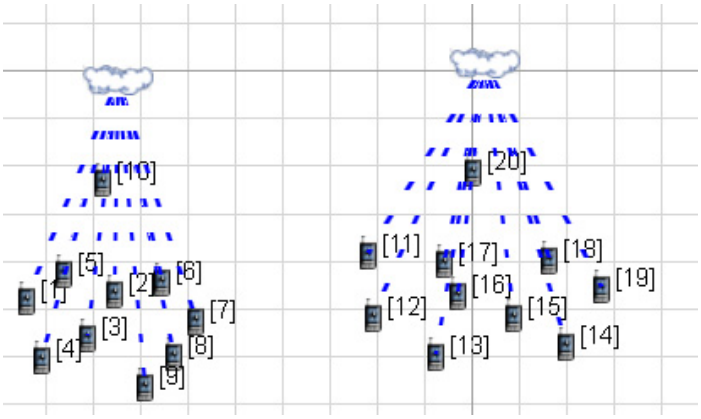
Step 1: Place nodes: in such way that one node in each as a base station (cluster head: in this 10th and 20th are the base)



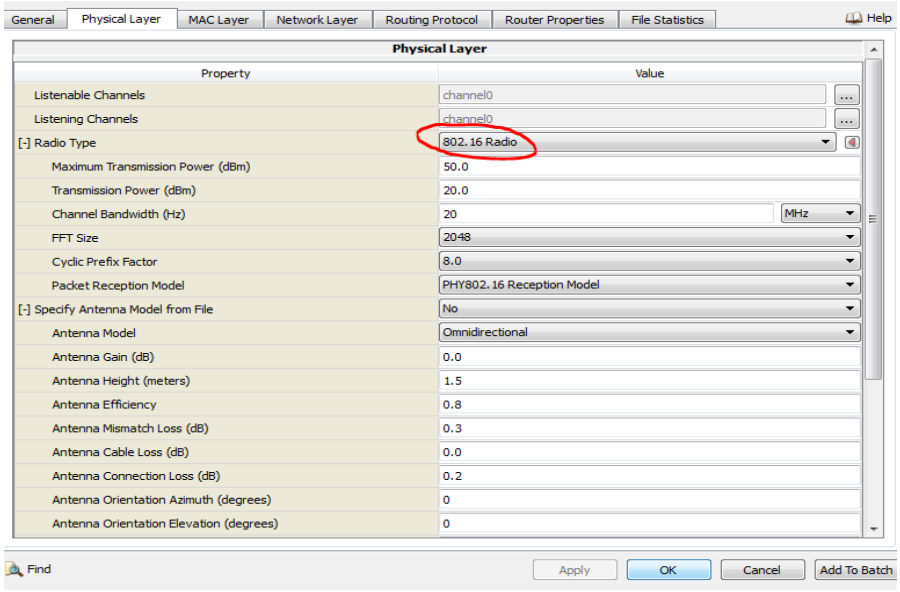
Select set of 10 nodes and place subnets for each set

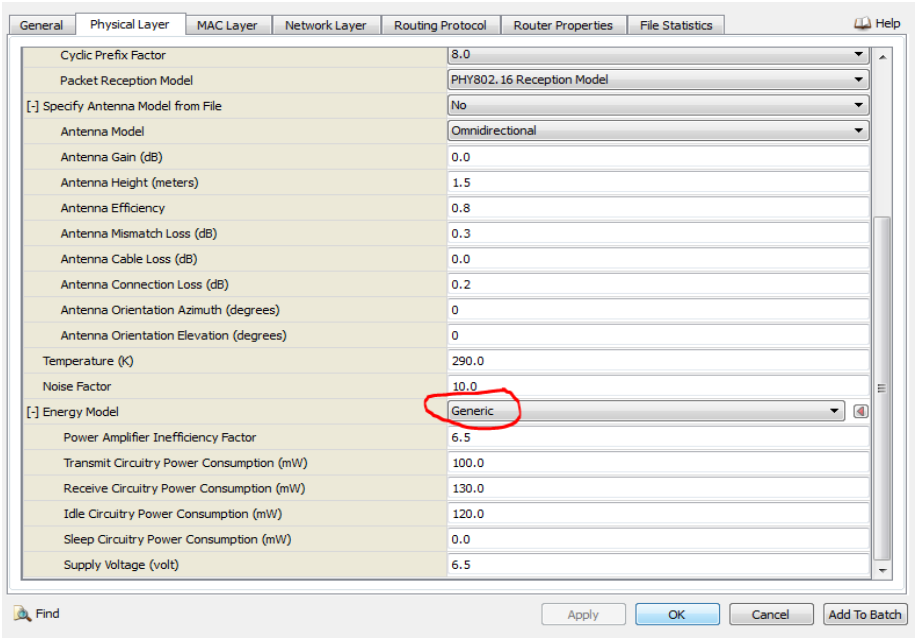


Right Click on subnet go to properties

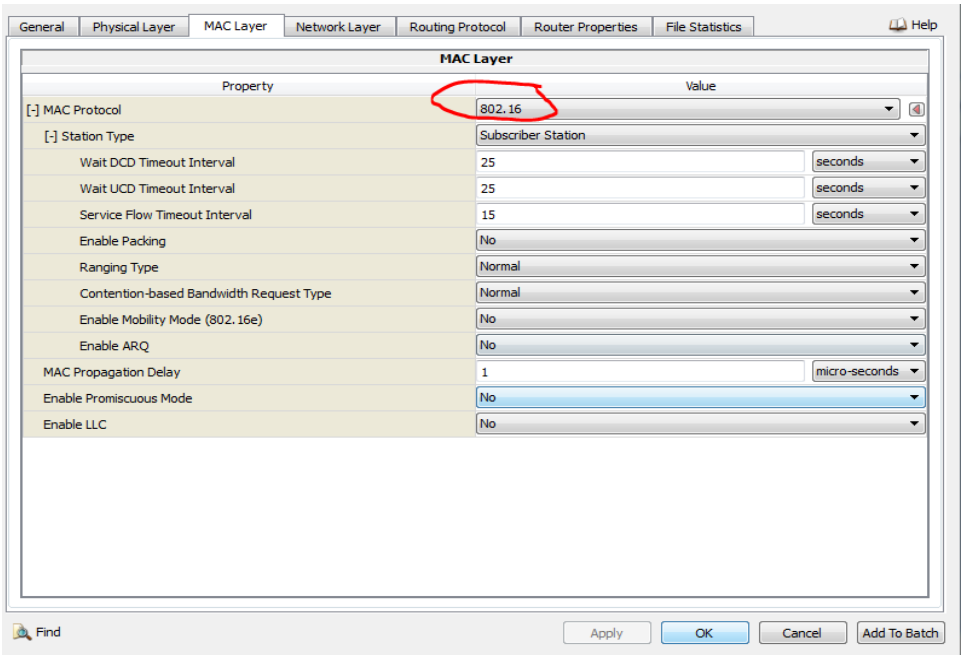


Step 2: Go to physical layer and set radio type: 802.16 radio (it's for Wimax) and energy model (optional)



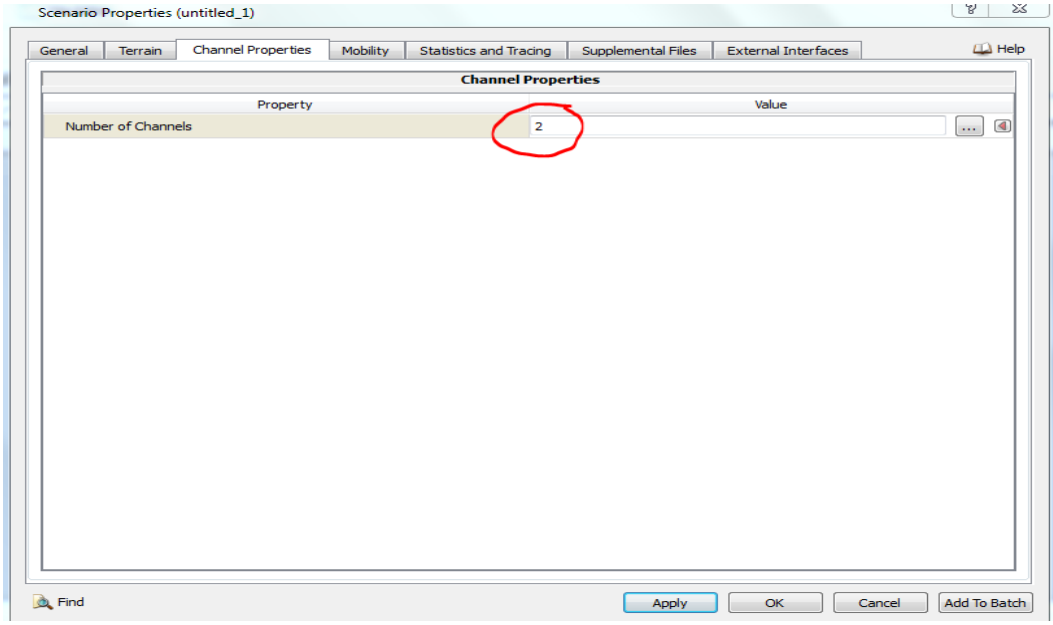


Go to MAC layer and set MAC protocol as (802.16)

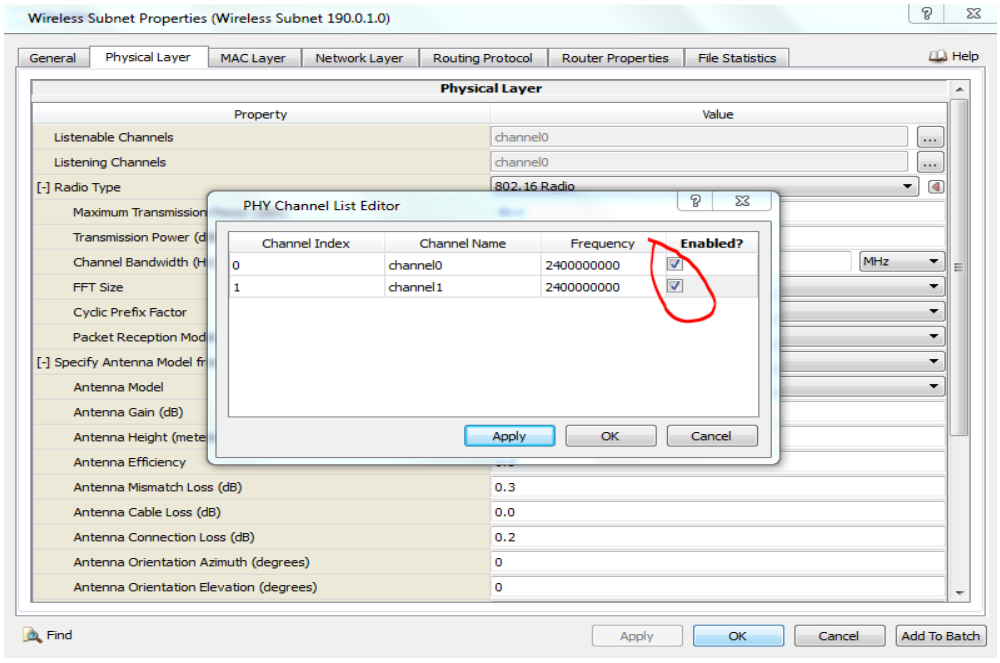


Similarly set all the parameters for other subnet

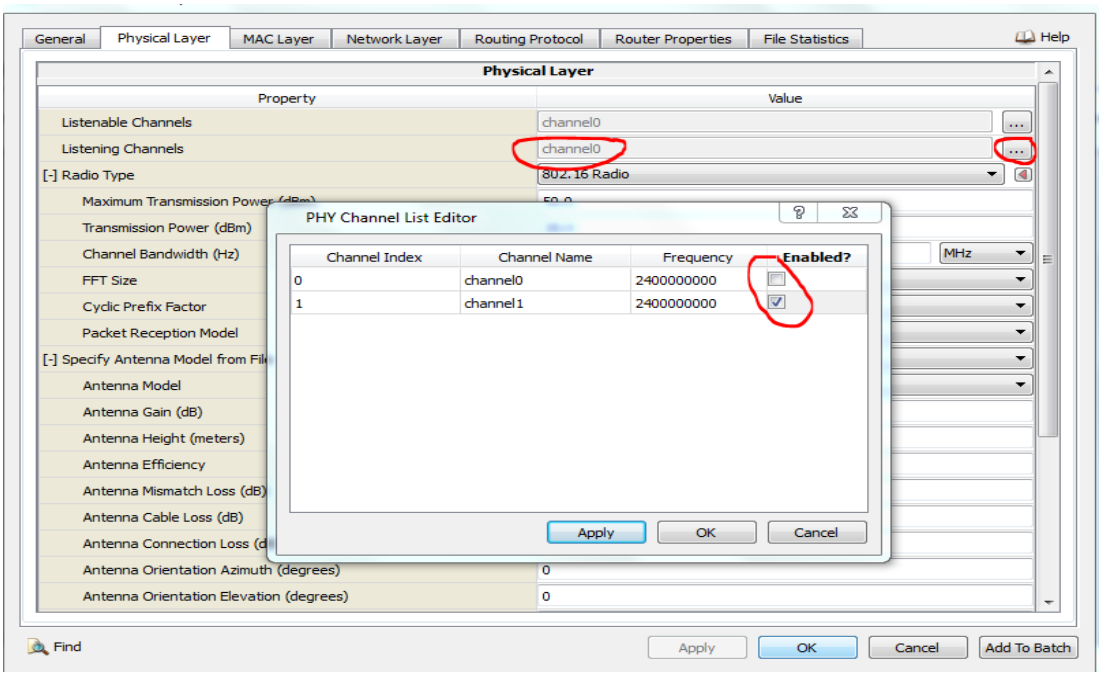
Step 3: Go to scenario properties to configure channels (Number of channel = 2, 1 channel is for one base hence for two base 2 channels are required). Apply and OK



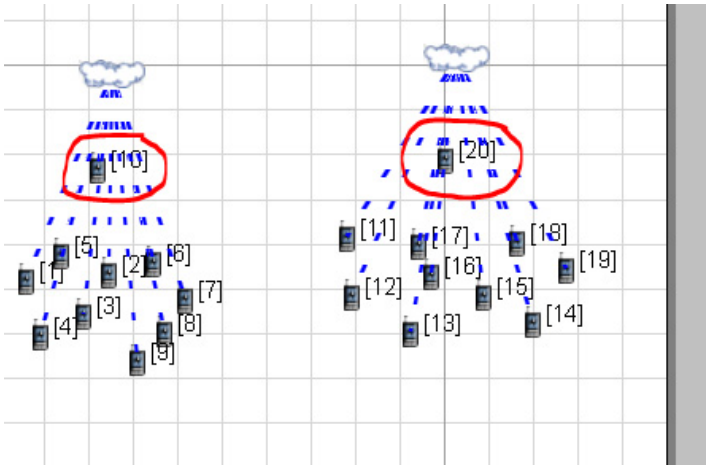
Step 4: Go to subnet properties: Click physical layer and go to Listenable channel activate both channels. Click Apply and Ok (This is for first subnet)

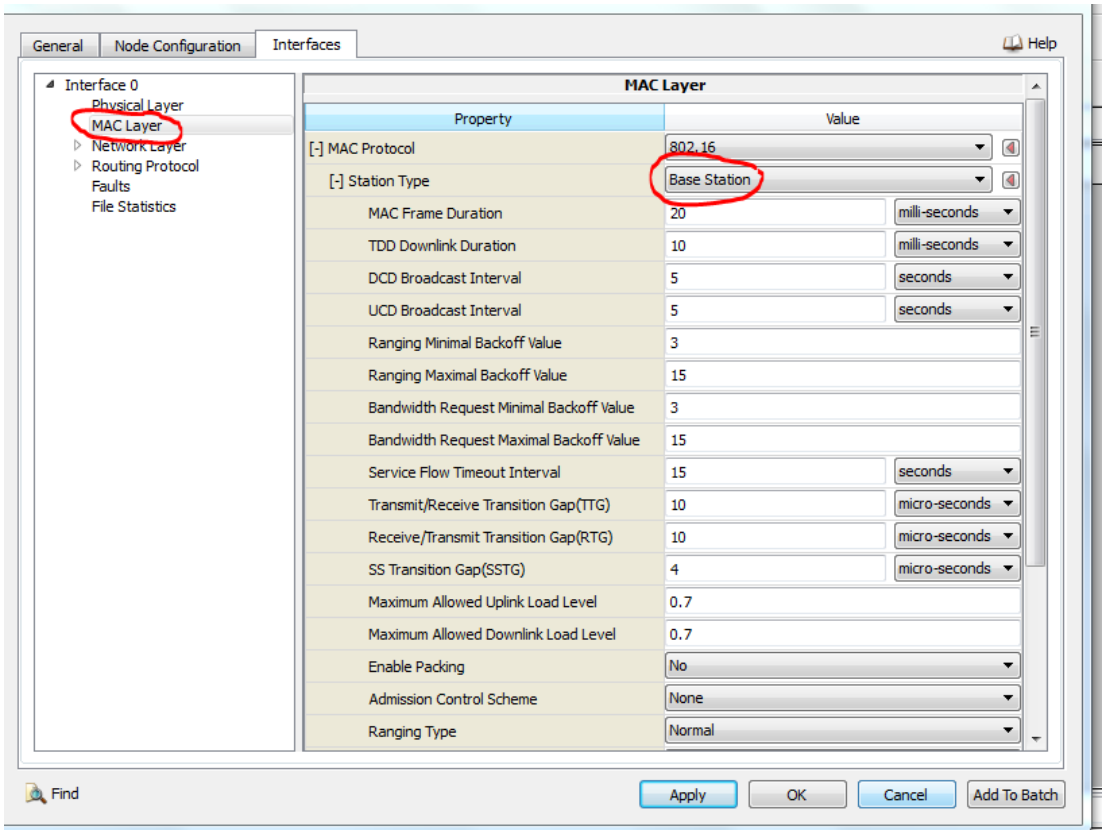


Go to subnet properties click physical layer and go to Listening channel activate channel 1. Click Apply and OK (This is for second subnet)

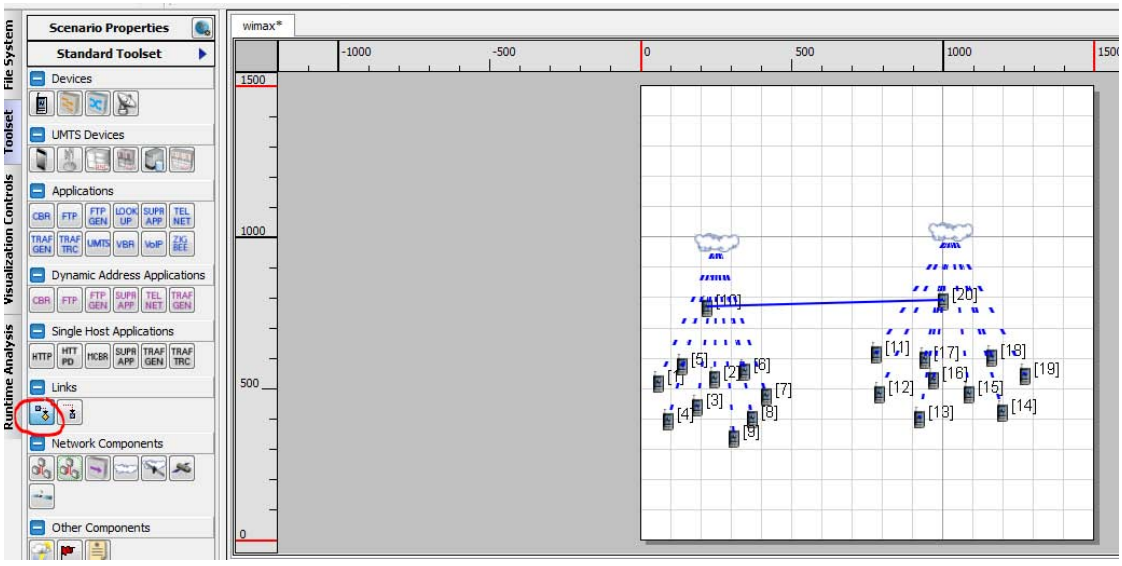


Step 5: Select 10th node properties to make 10th node as Base Station. Similarly for 20th node as Base station

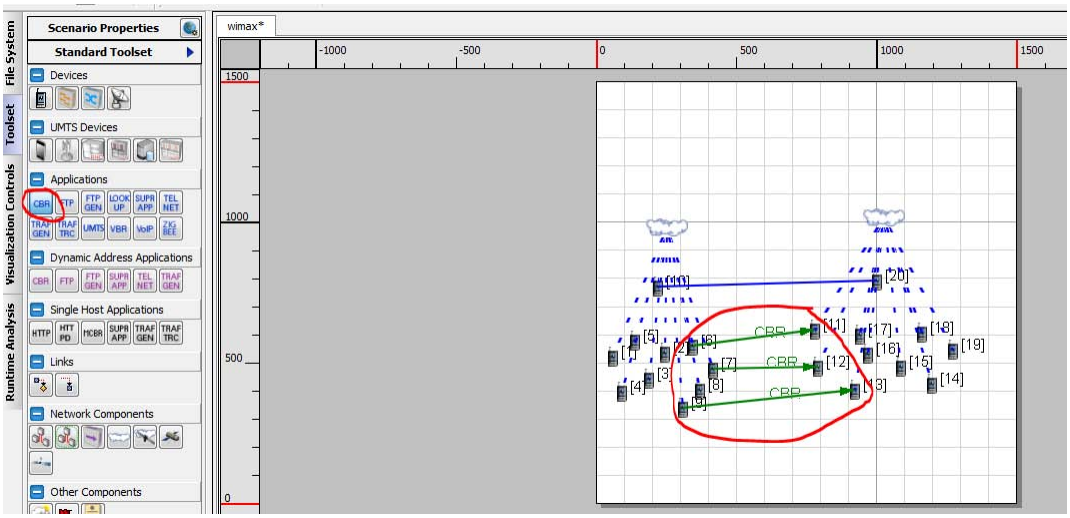




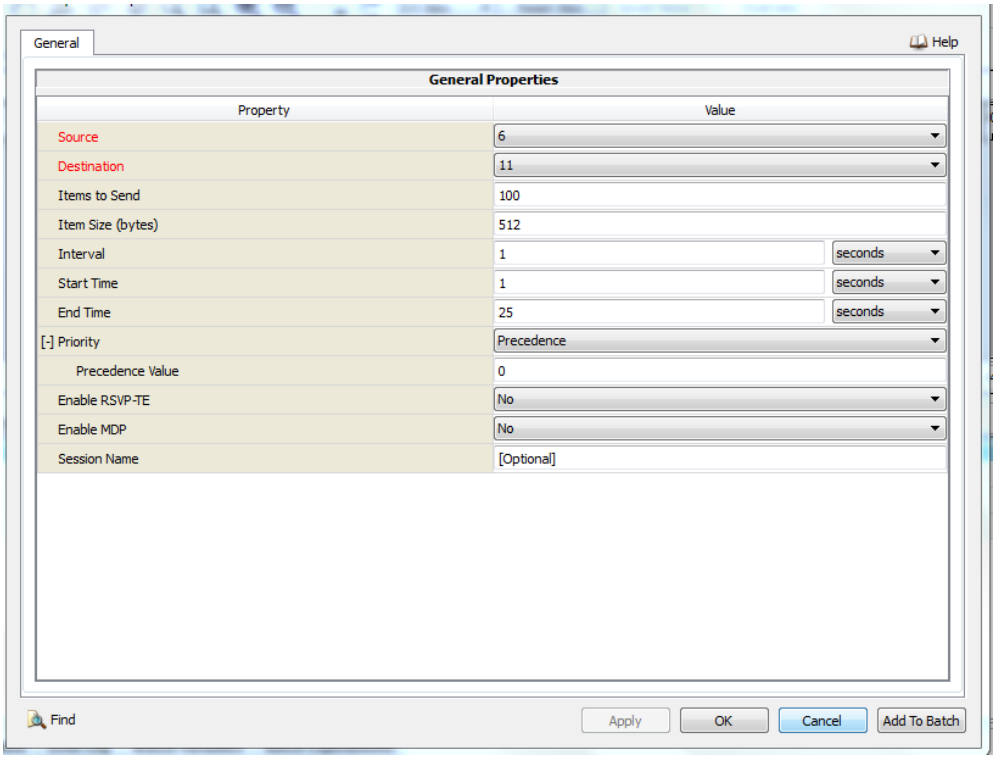
Link between two base stations: click on the link and connect two base stations



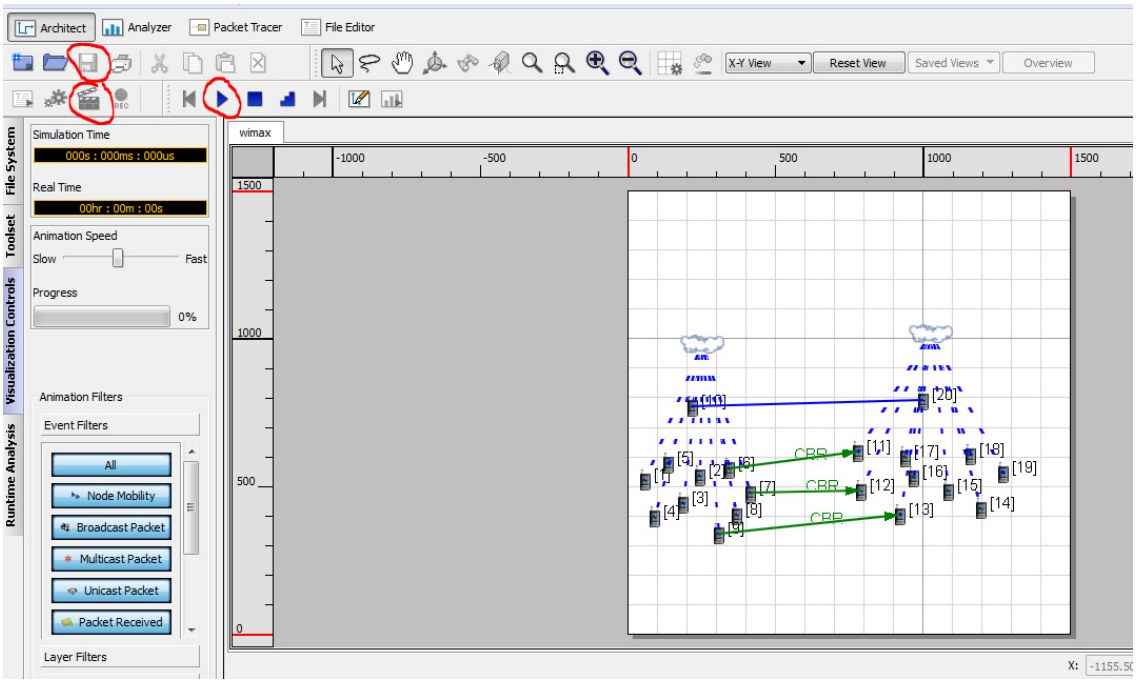
Step 6: Configure the traffic between the nodes: Click on the CBR and connect through the CBR to insert traffic using CBR.

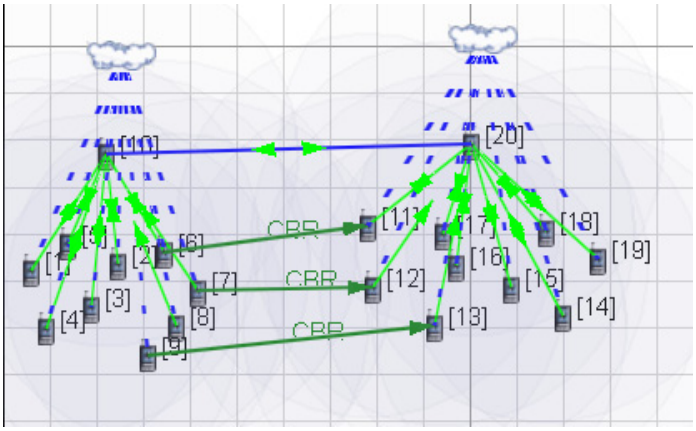


To configure CBR (constant bit rate): go to table view under applications select all by right clicking and again right click on properties

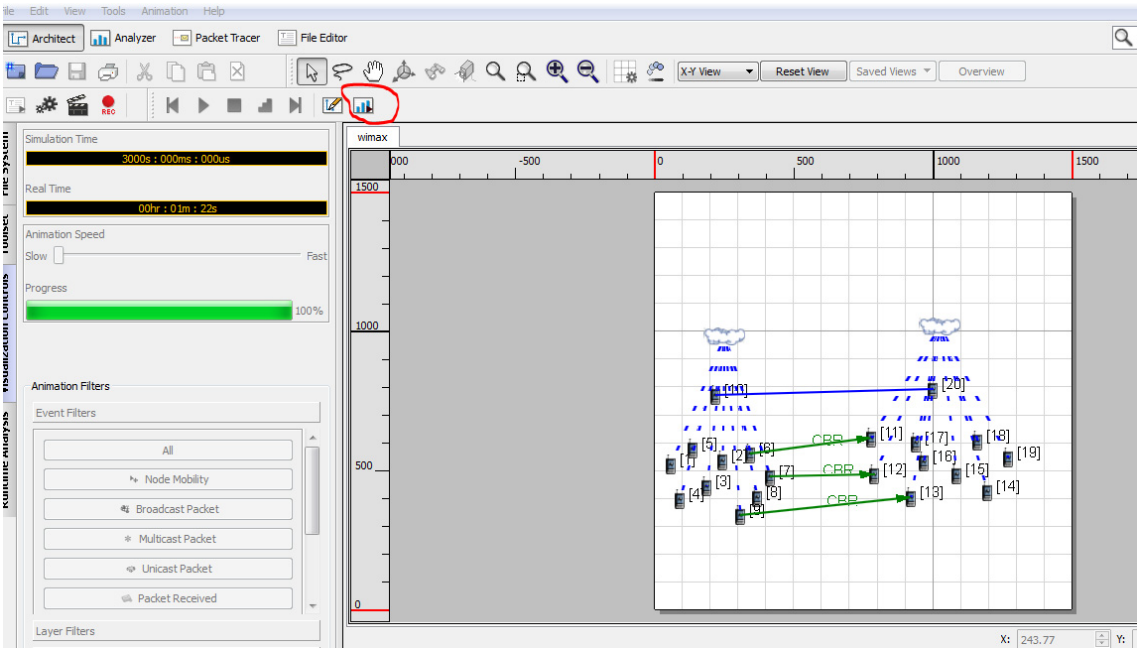


Step 7: Save the scenario, run and play

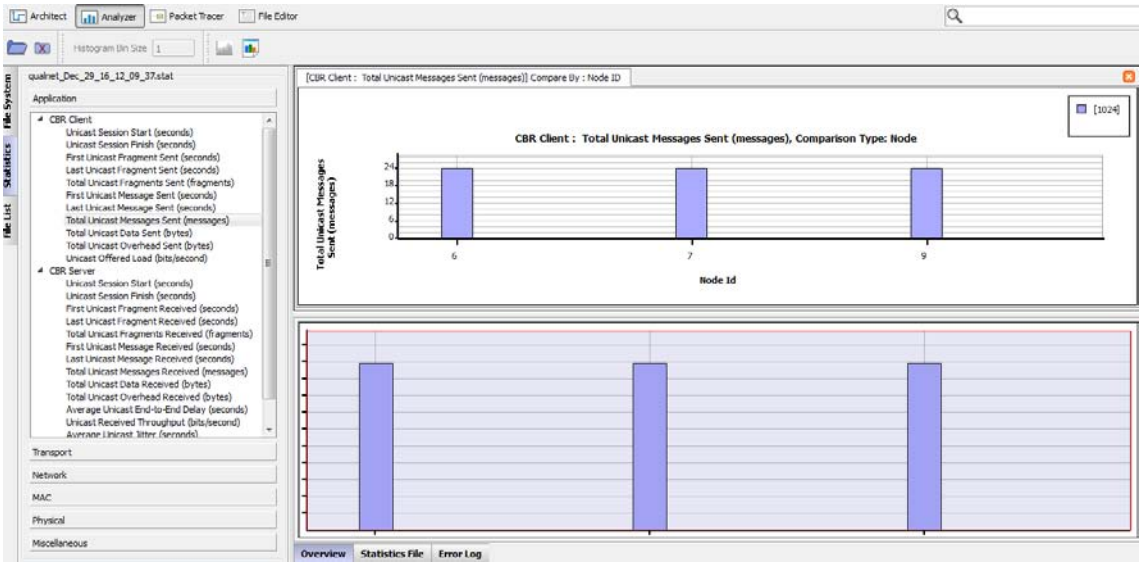




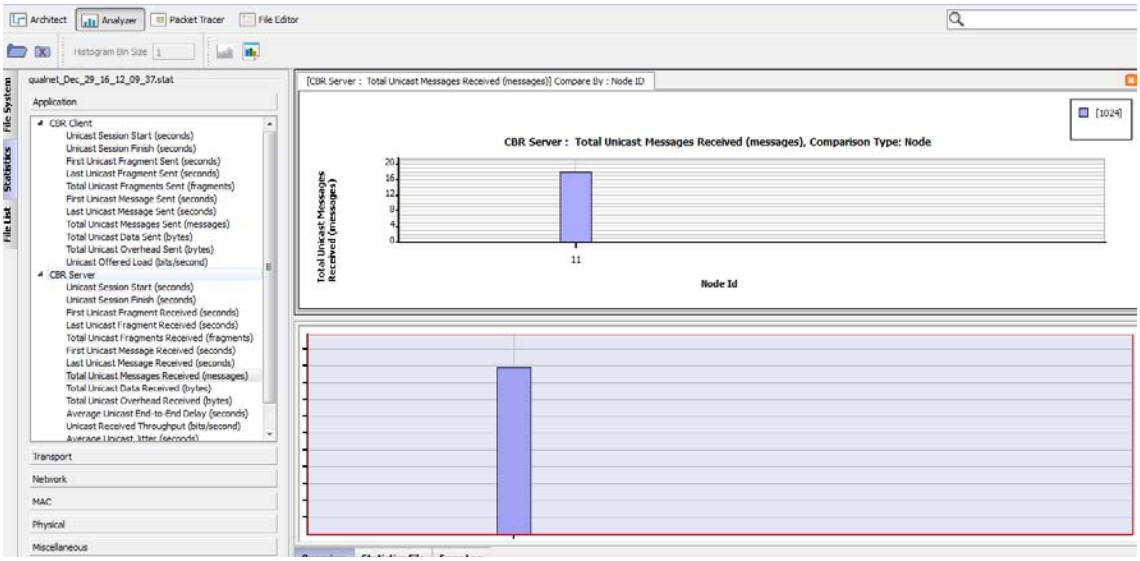
After simulation click analyse



Total unicast message sent



Total unicast message received



PART B

Experiment 5

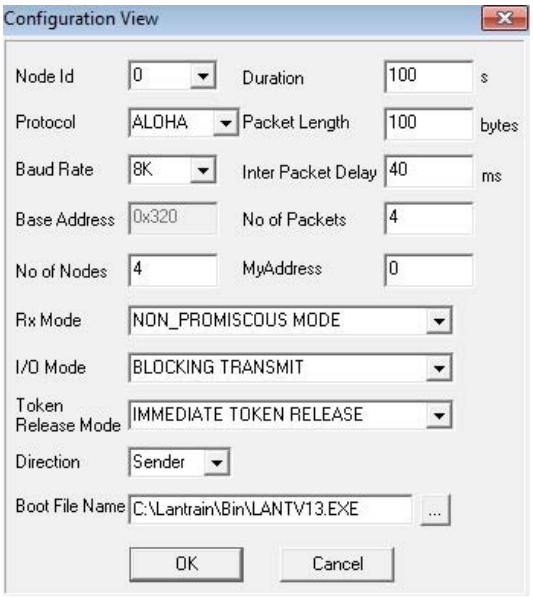
ALOHA Protocol

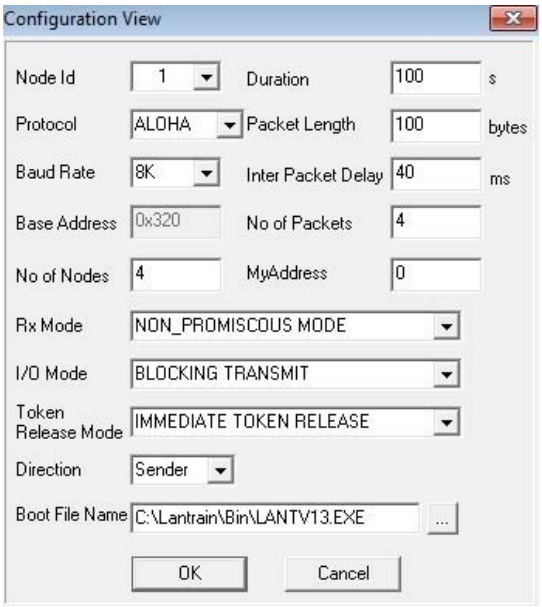
Aim: To implement the ALOHA protocol for packet communication between a number of nodes connected to a common bus.

Procedure:

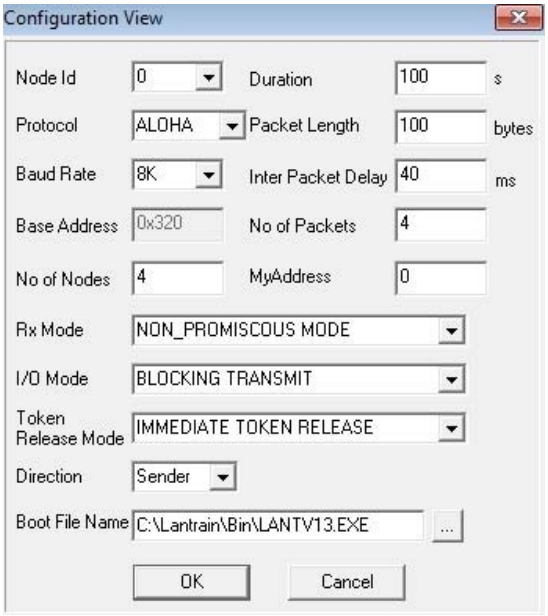
1. Click on the **MAC** icon on the desktop of both PC's.
2. Click **Configuration** button in the tool window in both the PC's.

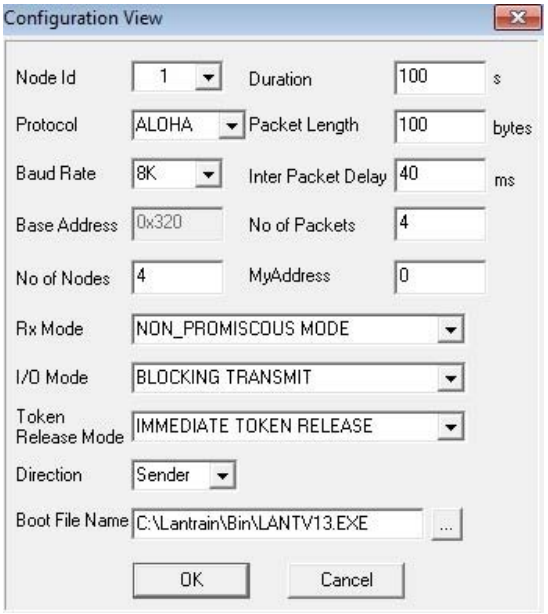
PC 1





PC 2





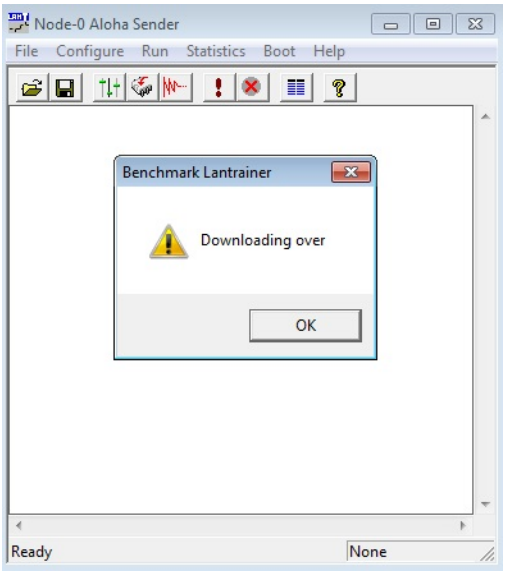
Configuration Setting:

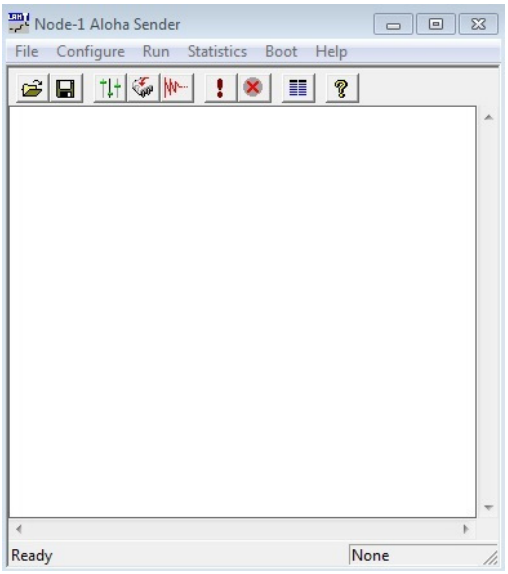
PC 1		**PC 2**	
Node Id	**0** on config menu 1 **1** on config menu 2	**Node Id**	**0** on config menu 1 **1** on config menu 2
Protocol	ALOHA	**Protocol**	ALOHA
Baud Rate	8 Kbps (at both config menu & NEU)	**Baud Rate**	8 Kbps (at both config menu & NEU)
Duration	100 Seconds	**Duration**	100 Seconds
Packet Length	100 bytes	**Packet Length**	100 bytes
Direction	Sender	**Direction**	Sender

Note: All the nodes have to be configured as '**Senders**'. Set the topology as '**Bus**'.

3. Click **OK** button.
4. Download the driver into the NIU. Click **BOOT** tab on the tool window of PC1 and click **OK** button. Booting from any one of the applications is sufficient.

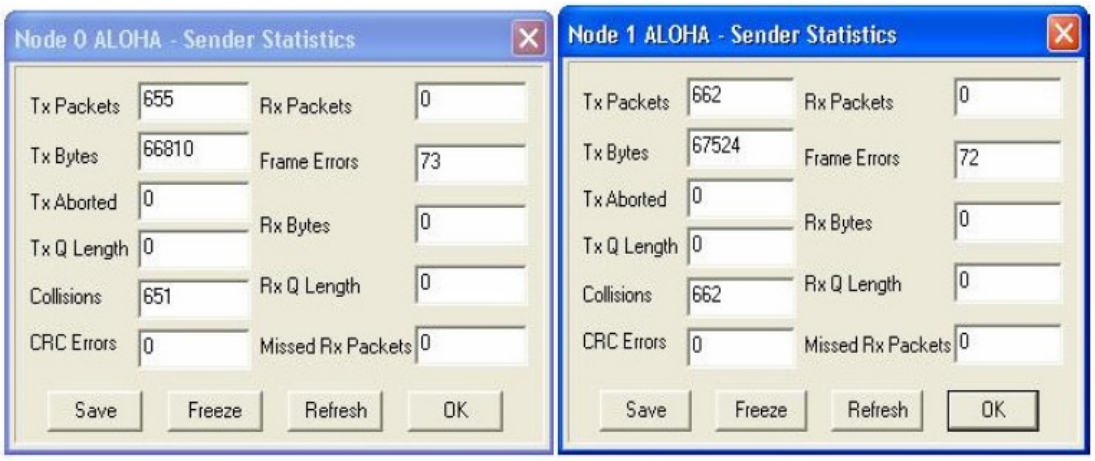
PC 1



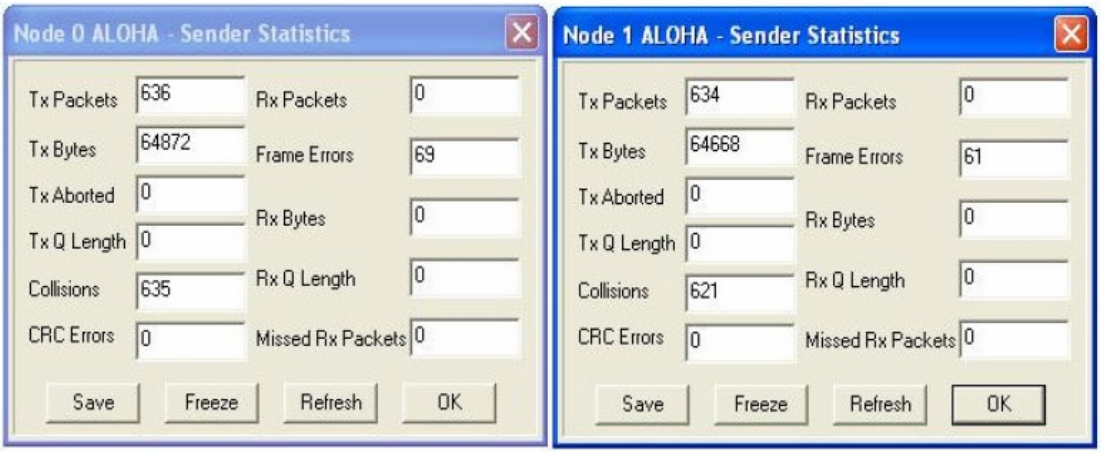


5. Repeat **step 4** for PC2.
6. Run the experiment: Click **RUN,** then S**tart** from each application.
7. View the statistics window for results. Click **Statistics** button on the window. Only Txd packets and collision count are taken into account for MAC calculation.
8. Note down the readings once the experiment is completed.
9. Repeat the above steps for various values of $\mathbf{t_a}$.
10. Calculate the Practical offered load from the below given formula and plot the graph for the practical Offered load v/s Throughput.

PC 1



PC 2



Calculations:

Theoretical Offered Load ($G_{Theoretical}$): $G_{Theoretical} = \frac{N*P}{C*t_a}$

G – Generated load in the network.

N – Number of nodes participating in the network. For example: 4 nodes (Using 2 computers)

P – Packet length expressed in bits; say 100 bytes (800 bits).

C – Data rate normally set as 8Kbps, which is selected in the NEU.

t_a – Inter Packet Delay (IPD) expressed in milliseconds; the time interval between two consecutive packets generated.

Let us assume $\mathbf{t_a}$= 40 milliseconds and substitute the above mentioned parameters in the above equation which leads to G =10. Likewise assume various values for $\mathbf{t_a}$ to generate offer loads in the range of 0.1 to 10. Substitute the value of $\mathbf{t_a}$ in the configuration menu.

Practical Throughput ($X_{Practical}$) from the obtained readings:

Successfully transmitted packet by a node = Txd Packets - Collision Count

$$X_{Practical} = \frac{Sum\ of\ successfully\ Txd\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ msec * Data\ rate}$$

Theoretical Throughput ($X_{Theoretical}$): $X_{Theoretical} = G_{Theoritical} e^{-2G_{Theoritical}}$

Practical Offered load ($G_{Practical}$):

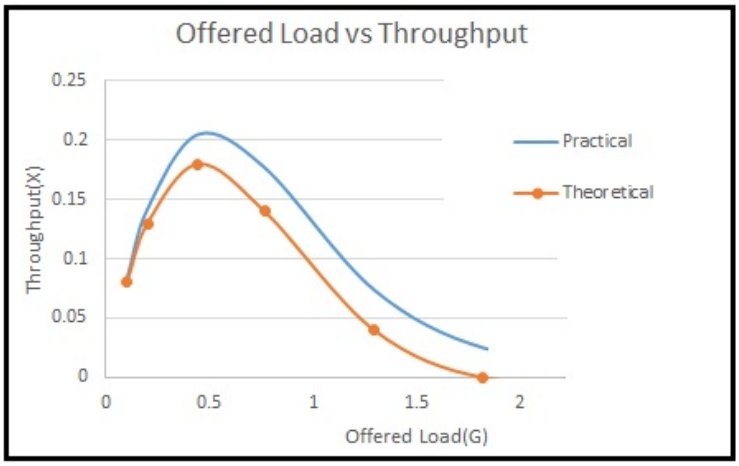
$$G_{Practical} = \frac{Sum\ of\ transmitted\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ msec * Data\ rate}$$

Result Tabulation:

IPD (m sec)	Packets Transmitted by each node				Packets Transmitted Successfully			
	Txd1	Txd2	Txd3	Txd4	Txd1	Txd2	Txd3	Txd4
4000								
2000								
800								
400								
200								
100								
40								

IPD (mSec)	Sum of Transmitted packets in all nodes	Sum of Successfully Transmitted packets in all nodes	$G_{Theoretical}$	$G_{Practical}$	$X_{Theoretical}$	$X_{Practical}$
4000						
2000						
800						
400						
200						
100						
40						

Expected Graph:



Exercise: Repeat the experiment for various values of Packet length, Node, Data rate.

Experiment 6

A. CSMA Protocol

Aim: To implement the CSMA protocol for packet communication between a number of nodes connected to a common bus.

Procedure:

1. Follow the procedure given in the Experiment 1 (ALOHA)

Setting the Configuration menu:

PC 1		PC 2	
Node Id	**0** on config menu 1 **1** on config menu 2	**Node Id**	**0** on config menu 1 **1** on config menu 2
Protocol	CSMA	**Protocol**	CSMA
Baud Rate	8 Kbps (at both config menu & NEU)	**Baud Rate**	8 Kbps (at both config menu & NEU)
Duration	100 Seconds	**Duration**	100 Seconds
Packet Length	100 bytes	**Packet Length**	100 bytes
Bit Delay	10 (at NEU)	**Bit Delay**	10 (at NEU)
Direction	Sender	**Direction**	Sender

Calculations:

Practical Throughput ($X_{Practical}$) from the obtained readings:

Successfully transmitted packet by a node = Txd Packets - Collision Count

$$X_{Practical} = \frac{Sum\ of\ successfully\ Txd\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ mSec * Data\ rate}$$

Theoretical Throughput ($X_{Theoretical}$):

$$X_{Theoretical} = \frac{G(1+G+aG(1+G+\frac{aG}{2}))e^{-G(1+2a)}}{G(1+2a)-(1-e^{-aG})+(1+aG)e^{-G(1+a)}}$$

Practical Offered Load ($G_{Practical}$):

$$G_{Practical} = \frac{Sum\ of\ transmitted\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ mSec * Data\ rate}$$

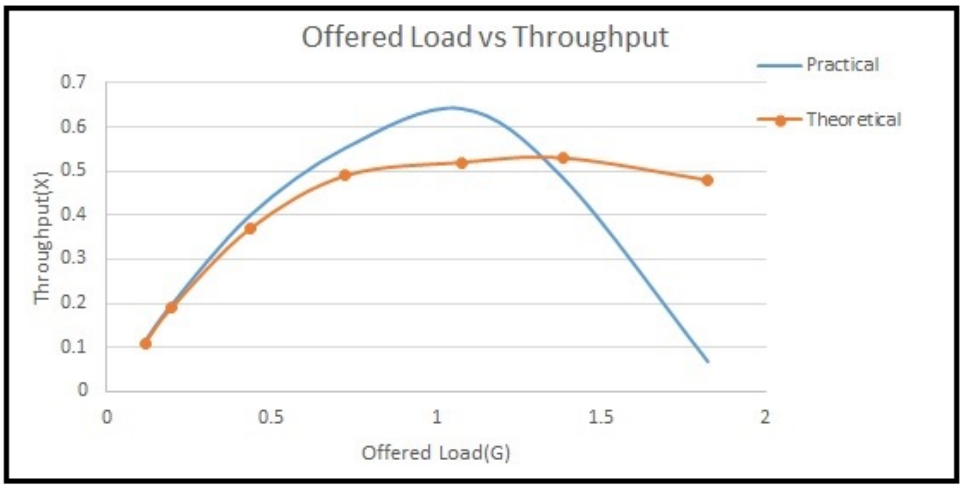
Result Tabulation:

For bit delay = 10

IPD (m sec)	Packets Transmitted by each node				Packets Transmitted Successfully			
	Txd1	Txd2	Txd3	Txd4	Txd1	Txd2	Txd3	Txd4
4000								
2000								
800								
400								
200								
100								
40								

IPD (mSec)	Sum of Transmitted packets in all nodes	Sum of Successfully Transmitted packets in all nodes	$G_{Theoretical}$	$G_{Practical}$	$X_{Theoretical}$	$X_{Practical}$
4000						
2000						
800						
400						
200						
100						
40						

Expected Graph:



Exercise: Repeat the experiment for various values of Packet length, Node, Data rate.

B. CSMA/CD Protocol

Aim: To implement the CSMA/CD protocol for packet communication between a number of nodes connected to a common bus.

Procedure:

1. Follow the procedure given in the Experiment 1 (ALOHA)

Setting the Configuration menu:

PC 1		PC 2	
Node Id	**0** on config menu 1 **1** on config menu 2	**Node Id**	**0** on config menu 1 **1** on config menu 2
Protocol	CSMA	**Protocol**	CSMA
Baud Rate	8 Kbps (at both config menu & NEU)	**Baud Rate**	8 Kbps (at both config menu & NEU)
Duration	100 Seconds	**Duration**	100 Seconds
Packet Length	100 bytes	**Packet Length**	100 bytes
Bit Delay	0 (at NEU)	**Bit Delay**	0 (at NEU)
Direction	Sender	**Direction**	Sender

Calculations:

Practical Throughput ($X_{Practical}$) from the obtained readings:

Successfully transmitted packet by a node = Txd Packets - Collision Count

$$X_{Practical} = \frac{Sum\ of\ successfully\ Txd\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ mSec * Data\ rate}$$

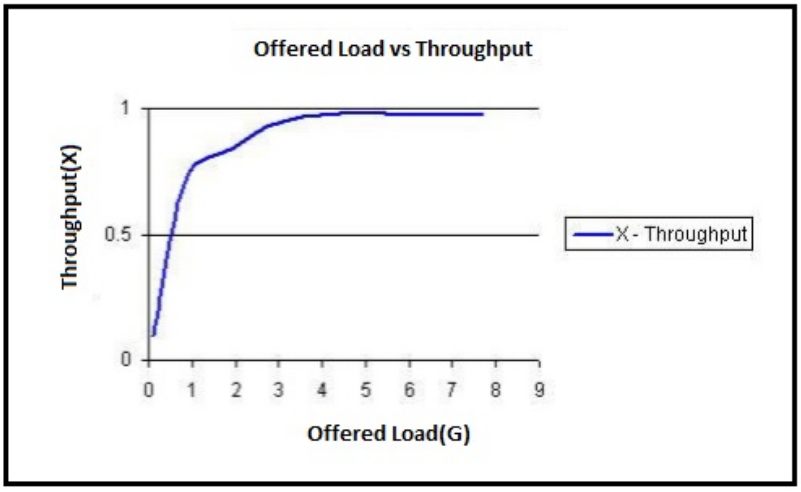
Practical Offered Load ($G_{Practical}$):

$$G_{Practical} = \frac{Sum\ of\ transmitted\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ mSec * Data\ rate}$$

Result Tabulation:

IPD (mSec)	Tx1	Tx2	Tx3	Tx4	$G_{Practical}$	$X_{Practical}$
4000						
2000						
800						
400						
200						
100						
40						

Expected Graph:



Exercise: Repeat the experiment for various values of Packet length, Node, Data rate.

C. Token Bus

Aim: To implement the token passing access in BUS-LAN.

Procedure:

1. Click on the **TOKEN BUS** icon on the desktop of both PC's.
2. Click **Configuration** button in the tool window in both the PC's.

Configuration Setting:

PC 1		PC 2	
Node Id	**0** on config menu 1 **1** on config menu 2	**Node Id**	**0** on config menu 1 **1** on config menu 2
Protocol	ALOHA	**Protocol**	ALOHA
Baud Rate	8 Kbps (at both config menu & NEU)	**Baud Rate**	8 Kbps (at both config menu & NEU)
Duration	100 Seconds	**Duration**	100 Seconds
Packet Length	1000 bytes	**Packet Length**	1000 bytes
My Address	0 on config menu 1 and 1 on menu 2	**My Address**	2 on config menu 1 and 3 on menu 2
Direction	Sender	**Direction**	Sender

Note: All the nodes have to be configured as '**Senders**'. Set the topology as '**Bus**'.

3. Click **OK** button.
4. Download the driver to the NIU. Click **BOOT** tab on the tool window of PC 1 & PC 2 and click **OK** button.
5. Start running the experiment from the lowest priority node. While you do this, **THT** window pops up, enter the *Token Holding Time* (THT) (say 10000 ms) in all nodes and press the OK button first in the node, which has highest value of My Address (i.e., from My Address 3).

Calculations:

Practical Throughput ($X_{Practical}$) from the obtained readings:

$$X_{Practical} = \frac{Sum\ of\ successfully\ Txd\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ sec * Data\ rate}$$

Where, Sum of successfully transmitted packets (displayed on the window) is obtained from the statistics at the end of the simulation.

Practical Offered Load ($G_{Practical}$):

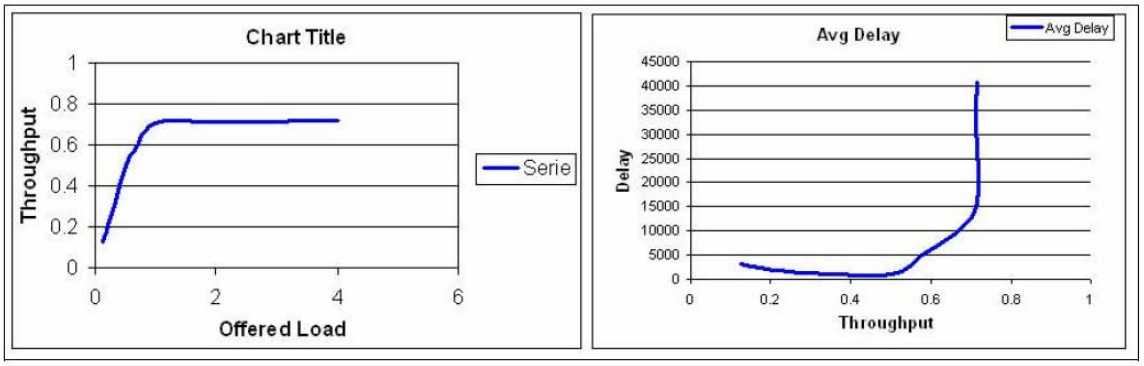
$$G_{Practical} = \frac{N * P * 8}{C * t_a}$$

G – Offered load, N – Number of nodes, P – Packet length in bytes
C – Data rate in bits/sec, t_a – Inter packet delay in milliseconds.

Result Tabulation:

IPD (msec)	Txd1	Txd2	Txd3	Txd4	$G_{Practical}$	$X_{Practical}$	Average Delay
16000							
8000							
4000							
2000							
1000							
100							
40							

Expected Graph:



Exercise:

1. Repeat the experiment for various values of Packet length, Node, Data rate.
2. Repeat the experiment by setting the BER to 10^{-2} in the NEU and also try to stop one of the nodes and observe the behaviour.

Experiment 7

A. Token Ring

Aim: To implement the token passing access in RING-LAN.

Procedure:

1. Click on the **TOKEN RING** icon on the desktop of both PC's.
2. Click **Configuration** button in the tool window in both the PC's.

Configuration Setting:

PC 1		PC 2	
Node Id	**0** on config menu 1 **1** on config menu 2	**Node Id**	**0** on config menu 1 **1** on config menu 2
Protocol	RING	**Protocol**	RING
Baud Rate	8 Kbps (at both config menu & NEU)	**Baud Rate**	8 Kbps (at both config menu & NEU)
Duration	100 Seconds	**Duration**	100 Seconds
Packet Length	1000 bytes	**Packet Length**	1000 bytes
My Address	0 on config menu 1 and 1 on menu 2	**My Address**	2 on config menu 1 and 3 on menu 2
Direction	Sender	**Direction**	Sender

Note: All the nodes have to be configured as '**Senders**'. Set the topology as '**Ring**'.

3. Click **OK** button.
4. Download the driver to the NIU. Click **BOOT** tab on the tool window of PC 1 & PC 2 and click **OK** button.
5. Start running the experiment from the lowest priority node. While you do this, THT window pops up, enter the Token Holding Time (THT) (say 10000 ms) in all nodes and press the OK button first in the node, which has highest value of My Address (i.e., from My Address 3).

Calculations:

Practical Throughput ($X_{Practical}$) from the obtained readings:

$$X_{Practical} = \frac{Sum\ of\ successfully\ Txd\ packets\ in\ all\ nodes * Packet\ Length * 8}{Duration\ of\ experiment\ in\ mSec * Data\ rate}$$

Where, Sum of successfully transmitted packets is obtained from the statistics at the end of the simulation.

Practical Offered Load ($G_{Practical}$):

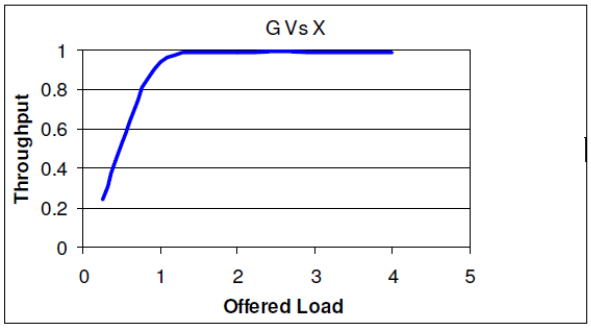
$$G_{Practical} = \frac{N * P * 8}{C * t_a}$$

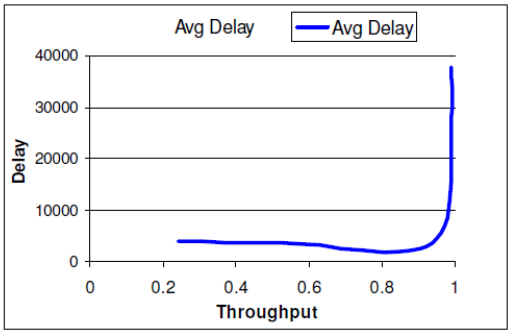
G – Offered load, N – Number of nodes, P – Packet length in bytes
C – Data rate in bits/sec, t_a – Inter packet delay in milliseconds.

Result Tabulation:

IPD(mSec)	Tx1	Tx2	Tx3	Tx4	$G_{Practical}$	$X_{Practical}$	Average Delay
16000							
8000							
4000							
2000							
1000							
100							
40							

Expected Graph:





Exercise:

1. Repeat the experiment for various values of Packet length, Node, Data rate.
2. Repeat the experiment by setting the BER to 10^{-2} in the NEU and also try to stop one of the nodes and observe the behaviour.

B. STOP and WAIT Protocol

Aim: To provide reliable data transfer between two nodes over an unreliable network using the **stop-and-wait** protocol.

Procedure:

1. Click on the **Stop & Wait** icon from the desktop on both PCs.
2. Click the **Configuration** button in the window in both the PC's.

Configuration Setting:

PC 1		PC 2	
Node ID	0	**Node ID**	0
Protocol	CSMA/CD	**Protocol**	CSMA/CD
Inter Packet Delay	400	**Inter Packet Delay**	400
Baud Rate	8 Kbps (At both the configuration menu and NEU)	**Baud Rate**	8 Kbps (At both the configuration menu and NEU)
Duration	100s	**Duration**	100s
Packet Length	1000 bytes	**Packet Length**	1000 bytes
Bit Delay	0 (at NEU)	**Bit Delay**	0 (at NEU)
Direction	Sender	**Direction**	Receiver

3. Click **OK** button and download the driver to the NIU using the **BOOT** button command. Booting from any one of the applications in enough.
4. Run the experiments: Click **RUN**, then **Start** from each application.
5. Set the timeout value to 1000 ms.
6. Note down the no. of successfully Transmitted Packets.
7. Repeat the above steps for various time out values and plot the graph between timeout value & throughput. Find the optimum timeout value from the plot.
8. Explain why the throughput is less compared to CSMA/CD protocol.

Practical Throughput ($X_{Practical}$) from the obtained readings:

$$X_{Practical} = \frac{Sum\ of\ successfully\ Txd\ packets * Packet\ Length * 8}{Duration\ of\ experiment\ in\ msec * Data\ rate}$$

Result Tabulation:

Timeout value in ms	Successfully Tx packets	Practical Throughput
1000		
1500		
2000		
3000		
4000		

C. STOP and WAIT with BER

Aim: To provide reliable data transfer between two nodes over an error network using the stop-and-wait protocol.

Procedure:

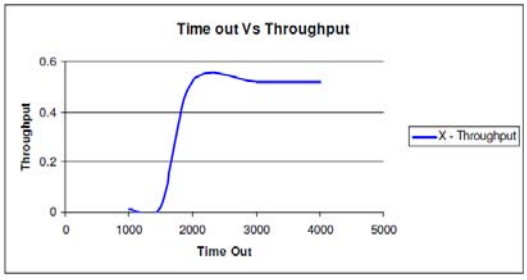
1. Set the error rate to $\mathbf{10^{-2}}$ in NEU.
2. Follow the **Stop-and-Wait** experiment procedure for running the experiment.
3. Set the timeout value as 3000 ms in the sender window.
4. From the Statistics window note down the number of successfully transmitted packets and calculate the throughput. (Calculation of throughput is same as explained in the previous exp.)
5. Repeated the experiment by setting different BER in the NEU.
6. Use the values to plot the graph between BER Vs Throughput.

Result Tabulation:

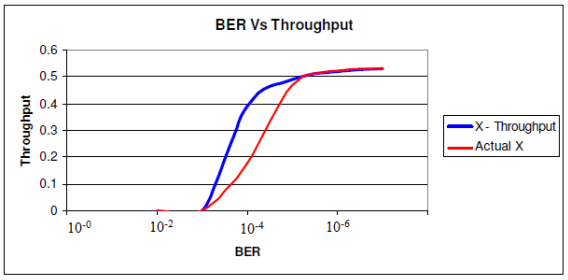
BER	Successfully Txd Packets	Theoretical Throughput	Practical Throughput
$\mathbf{10^{-2}}$			
$\mathbf{10^{-3}}$			
$\mathbf{10^{-4}}$			
$\mathbf{10^{-5}}$			
$\mathbf{10^{-6}}$			

Expected Graph:

Stop and Wait



Stop and Wait with BER



PART C

Experiment 8

Write a C/C++ program to do the following:

i. **Bit stuffing**

ii. **Character count**

iii. **Checksum**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<string.h>
inti,j;
void sender(int b[10],int k)
{
        intchecksum,sum=0;
        printf("\n****SENDER****\n");
        for(i=0;i<k;i++)
                sum+=b[i];
        printf("SUM IS: %d\n",sum);
        checksum=~sum;
        printf("\nSENDER's CHECKSUM IS:%d",checksum);
}

int main()
{
```

```
charstr[100], bstr[100];
        int a[100],m,scheck;
        char choice;
        printf("\n.....YOUR OPTIONS....\n");
        printf("\na.Checksum\nb. Bit stuffing\nc. Character count\n");
        printf("\nEnter your choice:");
        scanf("%c",&choice);
        switch(choice)
        {
           //Checksum Calculation
        case 'a':
        {
                printf("\nENTER SIZE OF THE STRING:");
                scanf("%d",&m);
                printf("\nENTER THE ELEMENTS OF THE ARRAY:");
                for(i=0;i<m;i++)
                scanf("%d",&a[i]);
                sender(a,m);
}
                break;
           //Bit stuffing
        case 'b':
        {
        int count=0;
        printf("Enter the bit string: ");
        scanf("%s",str);
```

```
                for(i=j=0; str[i]; i++)
        {

                        if(str[I ]=='1') count++;
                else            count=0;
                bstr[i+j]=str[i];
                if(count==5)
                {
                j++;
                bstr[i+j]='0';
                count=0;
}
}
        bstr[i+j]='\0';
        printf("\nAfter Bit stuffing : %s\n", bstr);
        }
                break;
 //Inserting  character count code
        case 'c':
        {
        chararr[100]; int x,y;
        printf("\nENTER THE ELEMENTS OF THE ARRAY:");
        scanf("%s",arr);
        printf("\n Resultant Frame using character count =  %d%s\n",strlen(arr)+1,arr);
        }
        break;
```

```
            default:
            printf("\nYou entered an invalid choice run program again");
        }
        getch();
}
```

Exercise: Write a C/C++ program to extract the data from the frames at the receiver using Bit stuffing, Checksum and Character count.

Experiment 9

CRC and Hamming Code

A. Write a C/C++ program to generate codeword at the sender using CRC method

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)
char t[28],cs[28],g[30];
int a,e,c;
voidexor()
{
        for(c = 1;c < N; c++)
        cs[c] = (( cs[c] == g[c])?'0':'1');
}

voidcrc()
{
        for(e=0;e<N;e++)    cs[e]=t[e];
        do{
                if(cs[0]=='1')  exor();
                for(c=0;c<N-1;c++)
        cs[c]=cs[c+1];
                cs[c]=t[e++];
```

```
        }while(e<=a+N-1);
}

int main()
{
        printf("\nEnter data : ");
        scanf("%s",t);
        printf("\n---------------------------------------");
        printf("\nEnterGenerating polynomial in binary : ");
        scanf("%s",g);
        a=strlen(t);
        for(e=a;e<a+N-1;e++)    t[e]='0';
        printf("\n---------------------------------------");
        printf("\nAugmenteddataword : %s",t);
        printf("\n---------------------------------------");
        crc();
        printf("\nChecksum is : %s",cs);
        for(e=a;e<a+N-1;e++)    t[e]=cs[e-a];
        printf("\n---------------------------------------");
        printf("\nFinalcodeword is : %s",t);
        printf("\n---------------------------------------\n\n");
        return 0;
}
```

B. Write a C/C++ program to generate Hamming code at the sender

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include <string.h>
#include <math.h>
#include<iostream>

int main(void)
{
        unsignedint m=0,r=0,l=0,x1=1,x2=1,x0=0;
        char d[1024]= {0};
        int d1[1024],d2[1024];
        printf("Enter the message to be encoded, in binary format: ");
        scanf ("%s",&d);
        //Message Length
         m=strlen(d);

        //Check bits (r)
        for (inti=0; i<20; i++)
        {
                r=i;
                if(m+1+i <= pow(2,i))
```

```
        break;
}

//Codeword length (l)
l =m+r;

//Testing the input in binary
for (inti=0; i<m; i++)
{
        while (!( d[i]=='0' || d[i]=='1'))
        {
        printf("\nPlease enter the input message in binary only.\n");
        exit(0);
        }
}

printf ("\nMessage length (m) = %d\n",m);
printf ("Redundancy bits (r) = %d\n",r);
printf("Codewordlength (l) = %d\n",l);

//Initialization to zero
for (inti=m; i<1024;i++)
        d[i]='0';
for (inti=0; i<1024; i++)
```

```
{   d1[i]=0;    d2[i]=0; }

//Copying string array to intarray, also shifting start index from 0 to 1
for (inti=0; i<m+1; i++)
{
if (d[i]=='1')  d1[i+1]=1;
else            d1[i+1]=0;
}

//Shifting message bits into non parity positions
for (int x2=1; x2<m+r+1; x2++)
{
        float x = (log(x2)/log(2))-(int(log(x2)/log(2)));
        if(x==0 || x==1)
        {   d2[x2]=0; x0=x0+1;  }
        else    d2[x2]=d1[x2-x0];
}

//Finding parity bits
for(int x2=1; x2<m+r+1; x2++)
{
        int x2t=x2;
        for(inti=0; i<r; i++)
        {
```

```
            intipow=pow(2,i);
            if (x2t%2==1)
            d2[ipow]=d2[ipow] ^ d2[x2];
            x2t=x2t/2;
            }
        }

        printf("Code word: ");
        for (intI =1; i<m+r+1; i++)
            printf ("%d",d2[i]);
        printf("\n\n");
}
```

Exercise: Write a C/C++ program to decode the code words at the receiver using CRC and Hamming code.

Experiment 10

Routing Algorithms

A. Write a C/C++ program for Dijkstra's algorithm to find the shortest path

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 99
#define MAX 10
voiddijkstra(int G[MAX][MAX], int n, intstartnode);
void main()
{
        int G[MAX][MAX], i, j, n, u;
        printf("\nEnter the no. of vertices:: ");
        scanf("%d", &n);
        printf("\nEnter the adjacency matrix::");
        for(i=0; i<n; i++)
        for(j=0; j<n; j++)
                scanf("%d", &G[i][j]);
        printf("\nEnter the starting node:: ");
        scanf("%d", &u);
        dijkstra(G,n,u);
        getch();
}

voiddijkstra(int G[MAX][MAX], int n, intstartnode)
{
        int cost[MAX][MAX], distance[MAX], pred[MAX];
        int visited[MAX], count, mindistance, nextnode, i,j;
        for(i=0;i <n;i++)
                for(j=0;j <n;j++)
                        if(G[i][j]==0)
                                cost[i][j]=INFINITY;
                        else
                                cost[i][j]=G[i][j];
```

```
for(i=0;i<n;i++)
{
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count < n-1)
{
        mindistance=INFINITY;
        for(i=0;i <n;i++)
                if(distance[i] <mindistance&&!visited[i])
                {
                        mindistance=distance[i];
                        nextnode=i;
                }
        visited[nextnode]=1;
        for(i=0;i <n;i++)
                if(!visited[i])
                        if(mindistance+cost[nextnode][i] < distance[i])
                        {
                                distance[i]=mindistance+cost[nextnode][i];
                                pred[i]=nextnode;
                        }
                count++;
}
for(i=0;i< n;i++)
        if(I !=startnode)
        {
                printf("\nDistance of %d = %d", i, distance[i]);
                printf("\nPath = %d", i);
                j=i;
                do
```

```
                {
                        J=pred[j];
                        printf(" <-%d", j);
                }
                while(j!=startnode);
        }
}
```

B. Write a C/C++ program for Bellman-Ford algorithm to find the shortest path

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <iostream>

struct Edge
{intsrc, dest, weight;};

struct Graph
{
        int V, E;        // V-> Number of vertices, E-> Number of edges
        struct Edge* edge;      // graph is represented as an array of edges.
};

// Creates a graph with V vertices and E edges
struct Graph* createGraph(int V, int E)
{
        struct Graph* graph = (struct Graph*) malloc( size of (struct Graph) );
        graph->V = V;
        graph->E = E;
        graph->edge =(struct Edge*) malloc( graph->E * size of (struct Edge ) );
        return graph;
}
// A utility function used to print the solution
```

```
voidprintArr(intdist[], int n)
{
        printf("Vertex   Distance from Source\n");
        for (inti = 0; i< n; ++i)
                printf("%d \t\t %d\n", i, dist[i]);
}
voidBellmanFord(struct Graph* graph, intsrc)
{
        int V = graph->V;
        int E = graph->E;
        intdist[V];

   // Step 1: Initialize distances from src to all other vertices as INFINITE
        for (inti = 0; i< V; i++)
                dist[i] = INT_MAX;
        dist[src] = 0;
        for (inti = 1; i<= V-1; i++)
        {
                for (int j = 0; j < E; j++)
                {
                int u = graph->edge[j].src;
                int v = graph->edge[j].dest;
                int weight = graph->edge[j].weight;
                if (dist[u] != INT_MAX &&dist[u] + weight <dist[v])
                dist[v] = dist[u] + weight;
                }
        }
        for (inti = 0; i< E; i++)
        {
                int u = graph->edge[i].src;
                int v = graph->edge[i].dest;
                int weight = graph->edge[i].weight;
                if (dist[u] != INT_MAX &&dist[u] + weight <dist[v])
        printf("Graph contains negative weight cycle");
        }
        printArr(dist, V);
```

```
        return;
}
int main()
{
        int V,E;
        printf("\nEnter the no. of vertics: ");
        scanf("%d", &V);
        printf("\nEnter the no. of Edges: ");
        scanf("%d", &E);
        struct Graph* graph = createGraph(V, E);
        int p,q,r;
        char a='y';
        inti=0;
        while(i<E)
        {
                printf("for  %d edge Enter the  source:", i);
                scanf("%d",&p);
                graph->edge[i].src = p;
                printf("Enter the destination:");
                scanf("%d", &q);
                graph->edge[i].dest =q;
                printf("Enter the weight:");
                scanf("%d",&r);
                graph->edge[i].weight = r;
                i++;
        }
        BellmanFord(graph, 0);
        return 0;
}
```

Exercise: Write a C/C++ program to implement the Distance Vector routing algorithm.