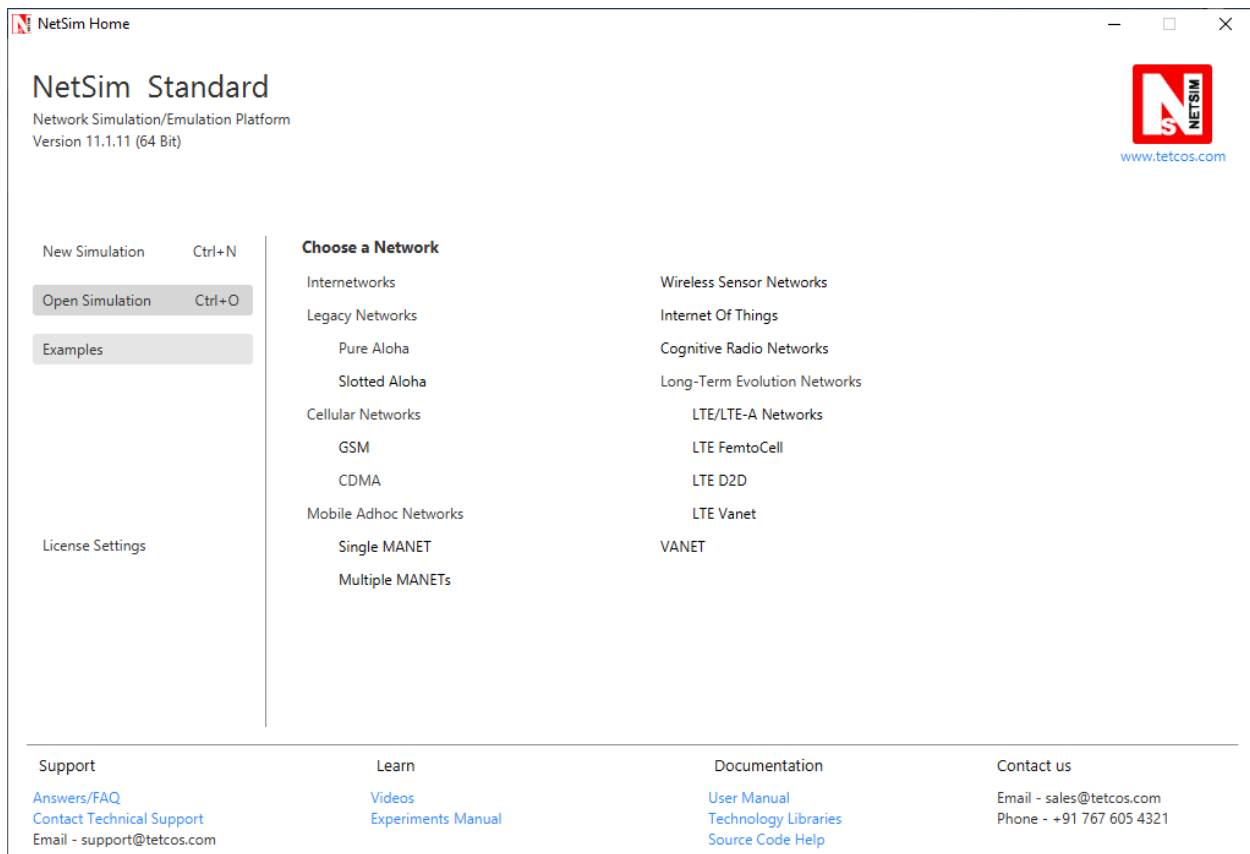


NetSim v11.1 Demo

Step 1

NetSim is an end-to-end, full stack, packet level network simulator and emulator.

<Show in NetSim Tool> It supports a wide range of technologies that include internetworks (which covers LAN, WAN and W LAN), Legacy Networks, cellular networks (which has GSM and CDMA), MANET, VANETs, Wireless Sensor Networks (which include Zigbee), (IOT) Internet of things, Cognitive Radio, and LTE / LTE-Advanced, Femtocell, D2D and LTE based VANETs.



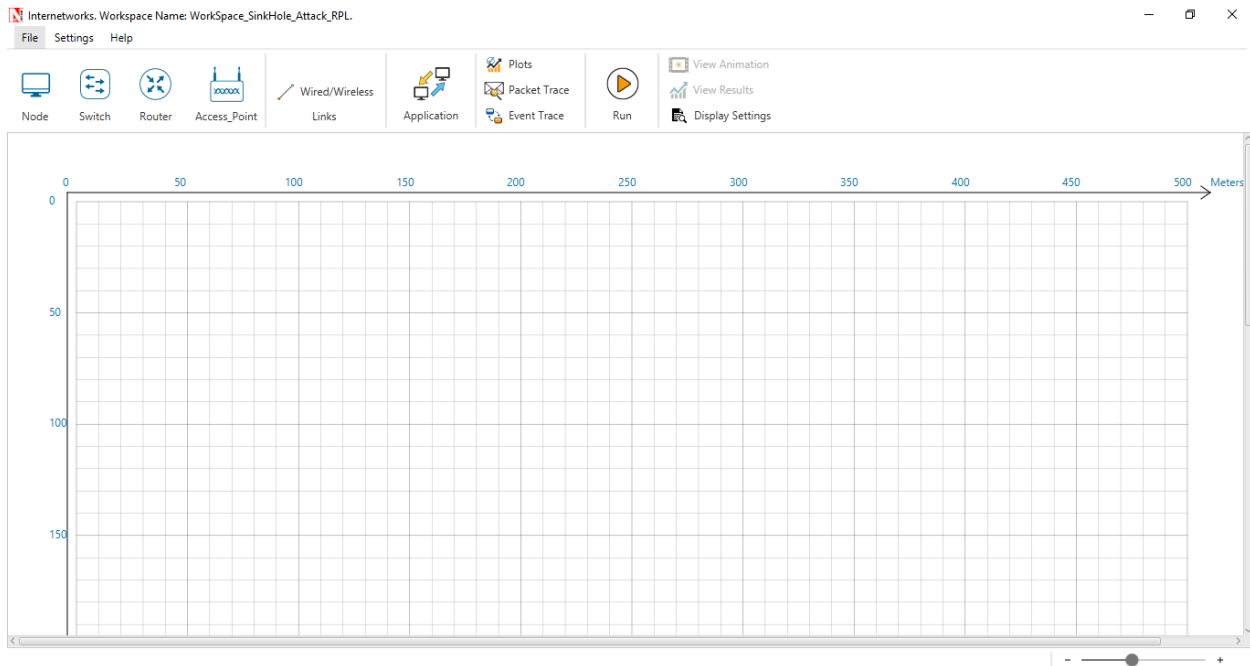
NetSim features an easy to use GUI enables users to create network designs and set device properties

<Go to New Simulation→Internetworks>

Creating a network in NetSim consists of 4 steps.

- Designing the network
- Configuring the network parameters
- Generating network traffic (applications)
- Simulating the network

In the first step, I am going to create a simple network. **<Click on devices and drop>**.

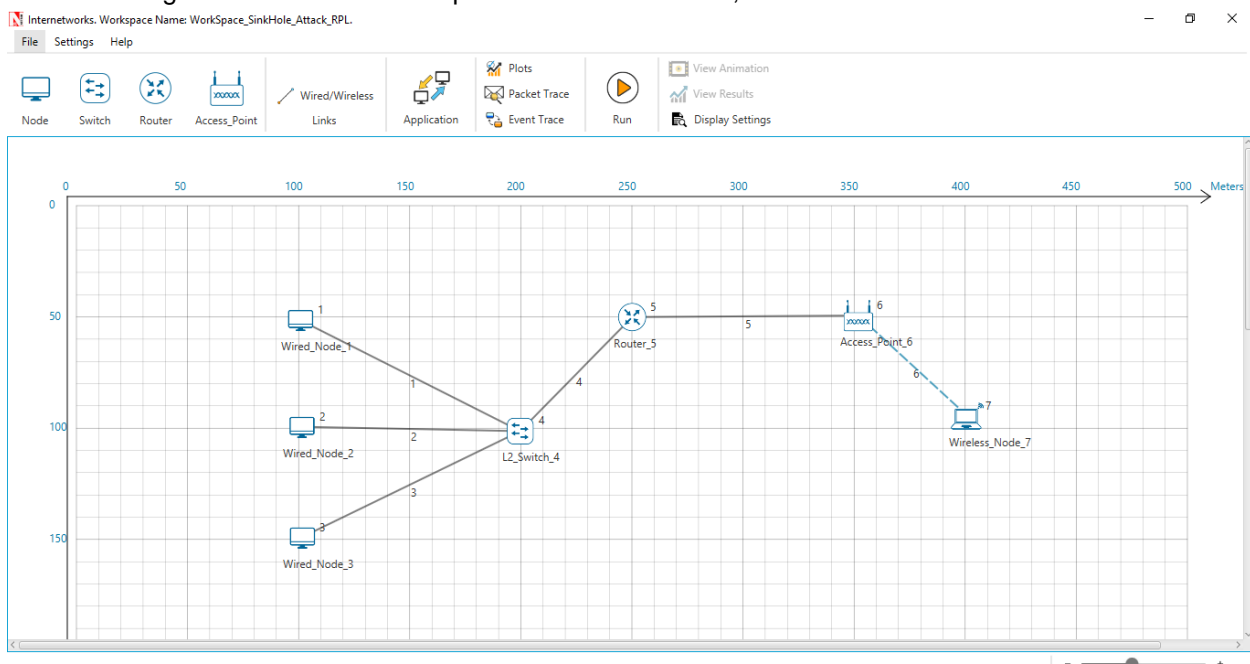


This is the NetSim design window. On the top we can see the ribbon (tool bar) which consists of various categories of devices. Below is the grid environment where the users can design the network.

We drop three wired nodes first, and then a switch followed by router. Then we drop an Access Point and a wireless node.

<Drop 3 wired nodes first. Then a switch followed by a router. Then and AP and a wireless Node.>

Now we have dropped the devices on the grid environment. In order to connect the devices, click on wired/wireless link and then just click on one device followed by the other to connect them. When it comes to establishing a link between access point and wireless node, NetSim establishes a wireless link.



With a few simple steps we have designed a basic network.

Step 2

Next step is to configure devices in NetSim. < Right click on the wireless node select properties>

We can see device properties as per TCP/IP stack. In the general properties, users can change the device name and device X, Y co-ordinates.

In the transport layer, users can specify and configure protocols like TCP and UDP. As you can see the in TCP protocol users can set the maximum segment size (MSS), congestion control algorithms like Old Tahoe, Tahoe, Reno, New Reno, BIC, CUBIC and various other parameters. In UDP protocol the packet size can be set.

In the interface properties, users can modify various network layer parameters like IP address, subnet mask and default gateway.

<Close this property window and right click on router to open its property>

In router, in the application layer, user can specify the routing protocol like OSPF and RIP. In router-interface, we can edit the network layer properties like IP address, subnet mask, and buffer size and users can select the scheduling type for performing Quality of Service like FIFO, Priority, Round Robin, WFQ.

<Close this property window and right click on access point to open its property>

Since Access Point is connected to a wireless node and a router. there are 2 interfaces.

In the wireless interface, in data link layer users can edit rate adaptation, RTS threshold, buffer size and more. In the PHY layer user can set the wireless standard (802.11 a, b, g, n, ac and p), transmitter power, antenna gain, operating channel etc.

<Close AP property window and right click on wired link>

Further, the links can be configured. Users can edit the link speed, Bit error rate and propagation delay.

<Close wired link property window and right click on wireless link>

In wireless link users can also modify the RF Propagation. We have various path loss models such as Friis free space, Log distance, Hata, Cost 231, Indoor office, factory, home etc. Similar fading and shadowing models can also be configured.

This concludes the second step of configuring the network parameters.

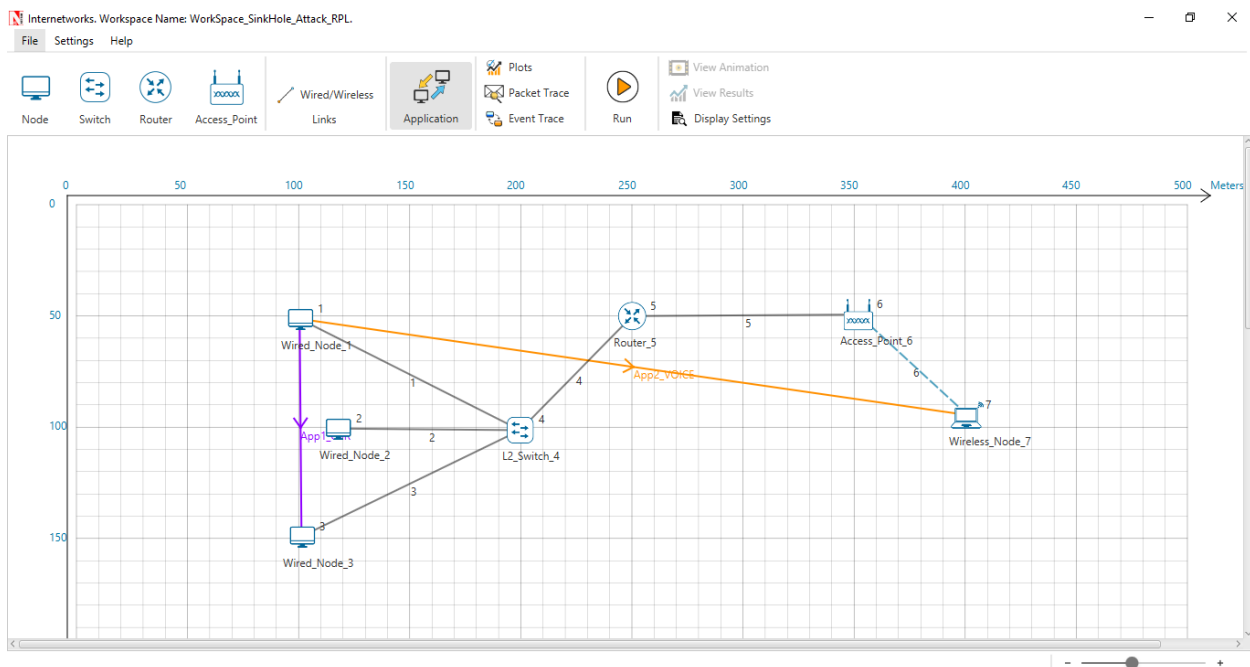
In the third step users need to model the traffic. <Click on the Application icon and set the properties>. Here users can see the different types of application protocols like CBR, custom, email, Video, Voice, Database, FTP, HTTP and Peer-to-peer.

Now I am configuring a CBR (constant bit rate) application, specifying the source as node 1 and destination as node 3 and setting the packet size and inter arrival time <set inter arrival time=2000>

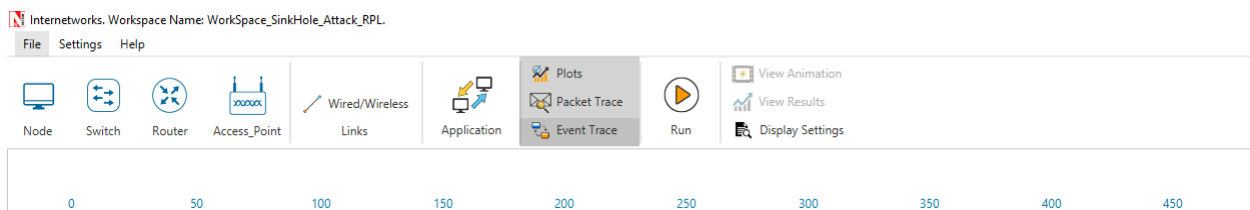
In addition, I am configuring a voice application from source as node1 to destination as node7
<Add one more application>.

Now as you can see CBR application is shown from node1 to node 3 and voice application is shown from node1 to node7.

This completes the third step of modeling applications for generating traffic

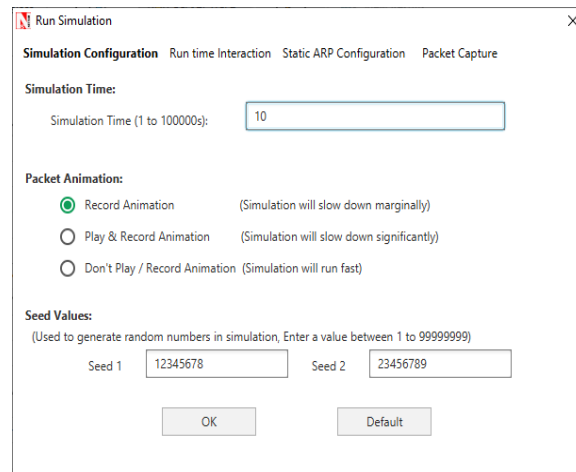


Before we run the simulation, we enable Plots and Packet Trace options. Enabling Plots allows the user to plot the value of a parameter over simulation time. On enabling Packet Trace, NetSim logs a set of chosen parameters for every packet as it flows through the network such as arrival time, queueing time, payload etc.



Step 3

The last step is simulating the network scenario created. Click on Run Simulation present in the Ribbon. Simulation time is the virtual time duration for which the network is simulated. We setting the simulation time to 10 seconds and running the simulation.



Once the simulation starts, NetSim Simulation console appears and displays the progress of the simulation

```

C:\Users\TETC05-PC\NetSim_11.1.11_64_std_default\bin\bin_x64\NetSimCore.exe
License Manager Output. Product>Edition>Maj_ver>Min_ver>Lic_type>Components>
netsim>std>11>0>rlm_hw>11111111>1010>
NetSim license validated
Installing heart-beat...
Heartbeat status = 0 (0 indicates successful)
NetworkStack loaded from path- C:\Users\TETC05-PC\NetSim_11.1.11_64_std_default\bin\bin_x64\NetworkStack.dll

***
NetSim start
Network Stack loaded
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\TETC05~1\AppData\Local\Temp\NetSim\Plot_*
Metrics variables initialized
Protocol variables initialized
Executing command --- DEL "C:\Users\TETC05~1\AppData\Local\Temp\NetSim\*.pcap"
Could Not Find C:\Users\TETC05~1\AppData\Local\Temp\NetSim\*.pcap
Emulation is disabled
Could Not Find C:\Users\TETC05~1\AppData\Local\Temp\NetSim\Animation_*
Applications created

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

22 % is completed... Simulation Time=2180.000 ms Event Id=6256

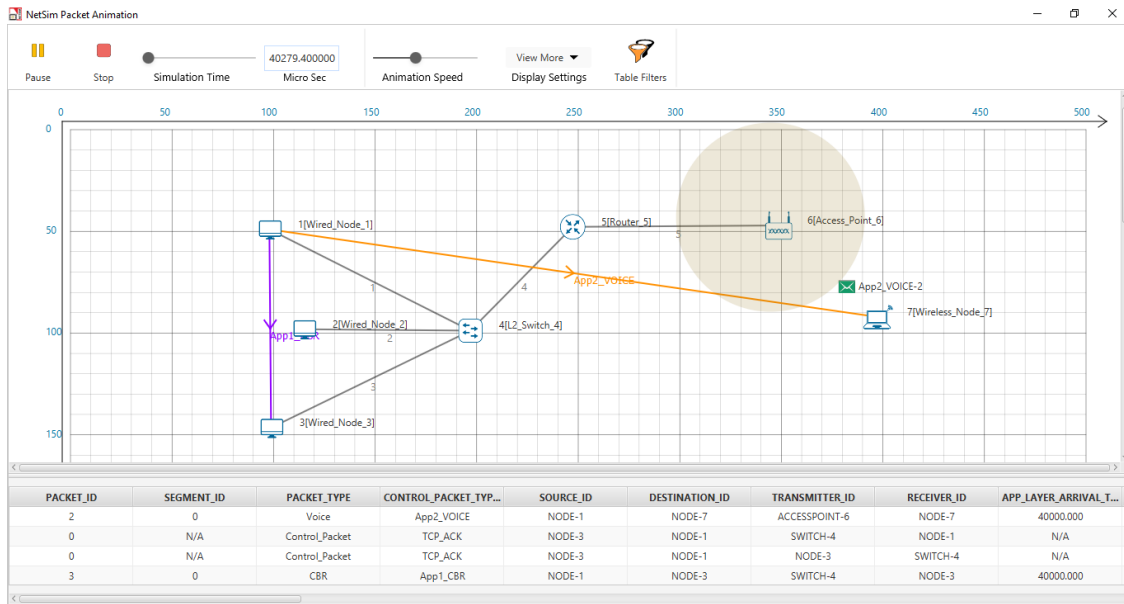
```

At the end of the simulation, NetSim provides outputs in two windows. One is the results dashboard and the other is the Packet Animator.

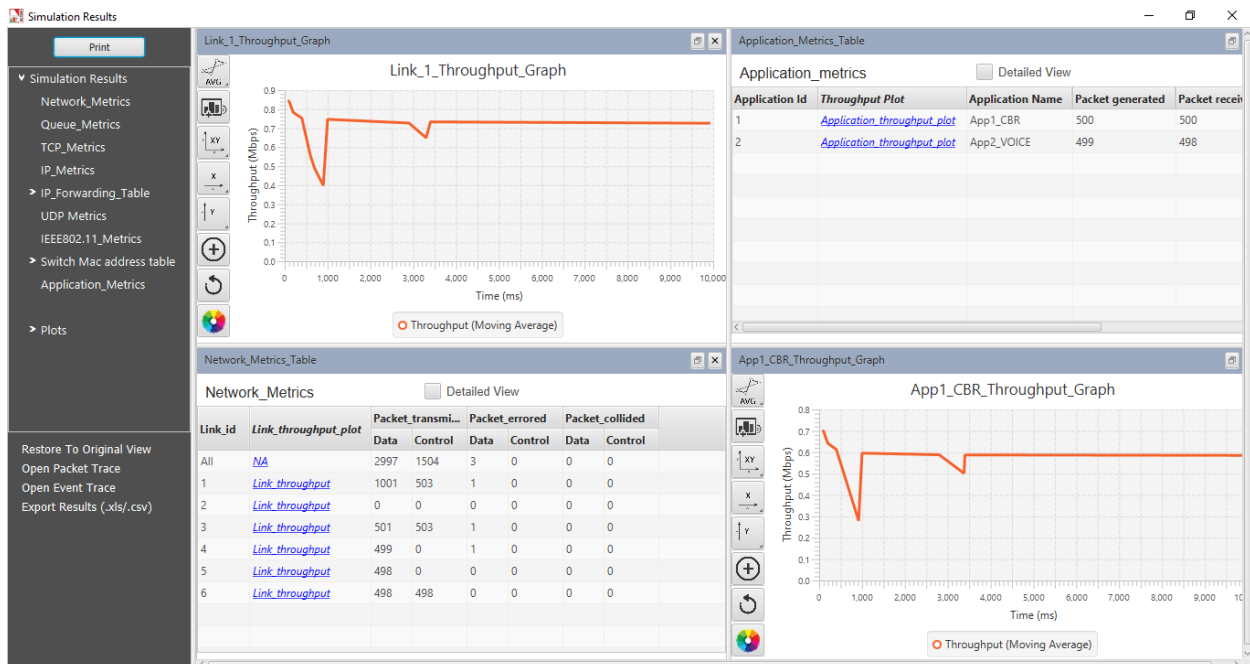
<Show the two window icons on the windows tool bar. One has NetSim icon and a small graph, the other has NetSim icon and small play button>

The packet animator enables users to view the traffic flow through the network. While viewing packet animation, users can see the flow of packets from source to destination. Blue color packet denotes control packet, green color is used for data packet and red color is error/collided packet. Speed controls are available to increase and decrease the animation speed. The table at the bottom has over 30 fields of information for each packet

<Use the scroll bar at the bottom of the table to scroll right to show the various parameter. This table is shown only if packet trace is enabled in the design window>



In addition to the animator, NetSim also displays a results dashboard upon completion of simulation. This contains various performance metrics and plots. With the help of metrics, users can analyze the behavior of the modeled network and can compare the impact of different algorithms on end-to-end behavior.



The **network metrics** allows users to view the values of the metrics obtained based on the overall network. Here users can see various parameters like number of packets successfully transmitted, errored and collided packets.

Network_Metrics_Table										
Network_Metrics <input checked="" type="checkbox"/> Detailed View										
Link_id	Link_throughput_plot	Packet_transmi...		Packet_errored		Packet_collided		Bytes_transmitted(bytes)	Payload_transmitted(bytes)	Overhead_transmitted(bytes)
		Data	Control	Data	Control	Data	Control			
All	NA	2997	1504	3	0	0	0	2037650	1780340	257310
1	Link throughput	1001	503	1	0	0	0	906044	811300	94744
2	Link throughput	0	0	0	0	0	0	0	0	0
3	Link throughput	501	503	1	0	0	0	797732	730000	67732
4	Link throughput	499	0	1	0	0	0	106786	79680	27106
5	Link throughput	498	0	0	0	0	0	106572	79680	26892
6	Link throughput	498	498	0	0	0	0	120516	79680	40836

The **TCP** metrics shows the values of the segment sent, segment received, segment retransmitted and many more.

TCP_Metrics_Table												
TCP_Metrics <input checked="" type="checkbox"/> Detailed View												
Source	Destination	Local Address	Remote Address	Syn Sent	Syn-Ack Sent	Segment Sent	Segment Received	Segment Retransmitted	Ack Sent	Ack Received	Duplicate segment received	
WIRED_NODE_1	ANY_DEVICE	11.1.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0
WIRED_NODE_2	ANY_DEVICE	11.1.1.3:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0
WIRED_NODE_3	ANY_DEVICE	11.1.1.4:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0
ROUTER_5	ANY_DEVICE	11.1.1.1:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0
WIRELESS_NODE_7	ANY_DEVICE	11.2.1.2:0	0.0.0.0:0	0	0	0	0	0	0	0	0	0
WIRED_NODE_1	WIRED_NODE_3	11.1.1.2:82	11.1.1.4:36934	1	0	500	0	2	1	500	0	0
WIRED_NODE_3	WIRED_NODE_1	11.1.1.4:36934	11.1.1.2:82	0	1	0	500	0	500	1	0	0

Application metrics displays the various metrics based on the application running in the network scenario like throughput, delay and more.

Application_Metrics_Table						
Application_metrics <input type="checkbox"/> Detailed View						
Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)
1	Application throughput plot	App1_CBR	500	500	0.584000	26460.607464
2	Application throughput plot	App2_VOICE	499	498	0.063744	649.485290

<Click on packet trace file to open it> As we had enabled **packet trace**, NetSim logs the trace file with a set of chosen parameters for every packet as it flows through the network such as arrival times, queuing times, payload, overhead, errors, collisions etc. As you can see this is a very large file running into several thousand lines with detailed information.

<Show>Here NetSim also provides options for users to save experiments and print the result.

<Close the results and the Design window and go back to the home screen of NetSim>

Step 4

Save / open simulations & workspaces in NetSim:

We will now quickly look at how to save/open experiments and briefly look into workspaces in NetSim.

<Go to NetSim Home page. Click on Open simulation>

When you design & simulate a network in NetSim it is saved as an experiment.

Open simulation option is used to open the saved experiment present in current workspace. Here users can view all the different experiments saved with Experiment name, Date modified & Network type. Users can also Export/Import the experiment so that they can run it any other system or save it elsewhere as a backup.

All experiments are saved within a Workspace. In a logical sense, a workspace contains all the experiments, source code files, executable files, icons, etc.

The default workspace of NetSim will have the Master Source code and the Master Binaries (Compiled files)

To create a new workspace, click on Wokspace options present in Open Simulation Men, then select More Options. A New Workspace pop-up window appears where you can input the Workspace Name, Description and Workspace Path (where you want to create new workspace).

Once workspace created by default set as current workspace, users create experiments and save them within the workspace.

Users can also Export/import entire workspaces

Step 5

Examples in Netsim:

Click on “Examples” present in Netsim homepage

It allows the users with Access to Network-wise simulation examples on the left and Experiments on the right. 30+ experiments are available covering the entire range of technologies in NetSim. This is extremely useful for educational institutions planning to use NetSim for their Under-grad network lab courses.

<Click on the book icon on the right>. The documentation is extensive, and each experiment comes with sections such as objective, theory, network setup, results, analysis and inference.

Step 6

IOT simulation

<Click on IOT>

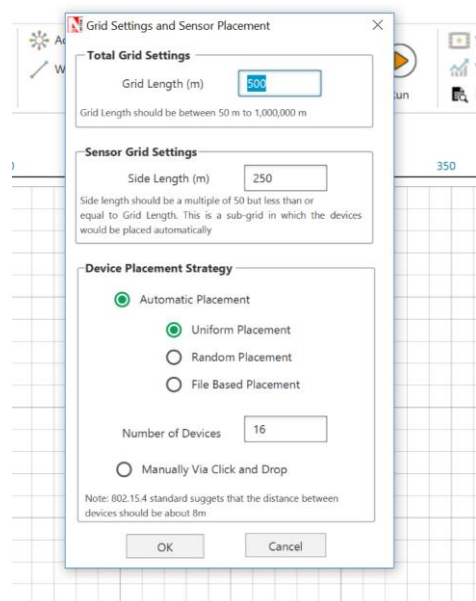
We will now show a quick simulation of an IOT Network.

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.

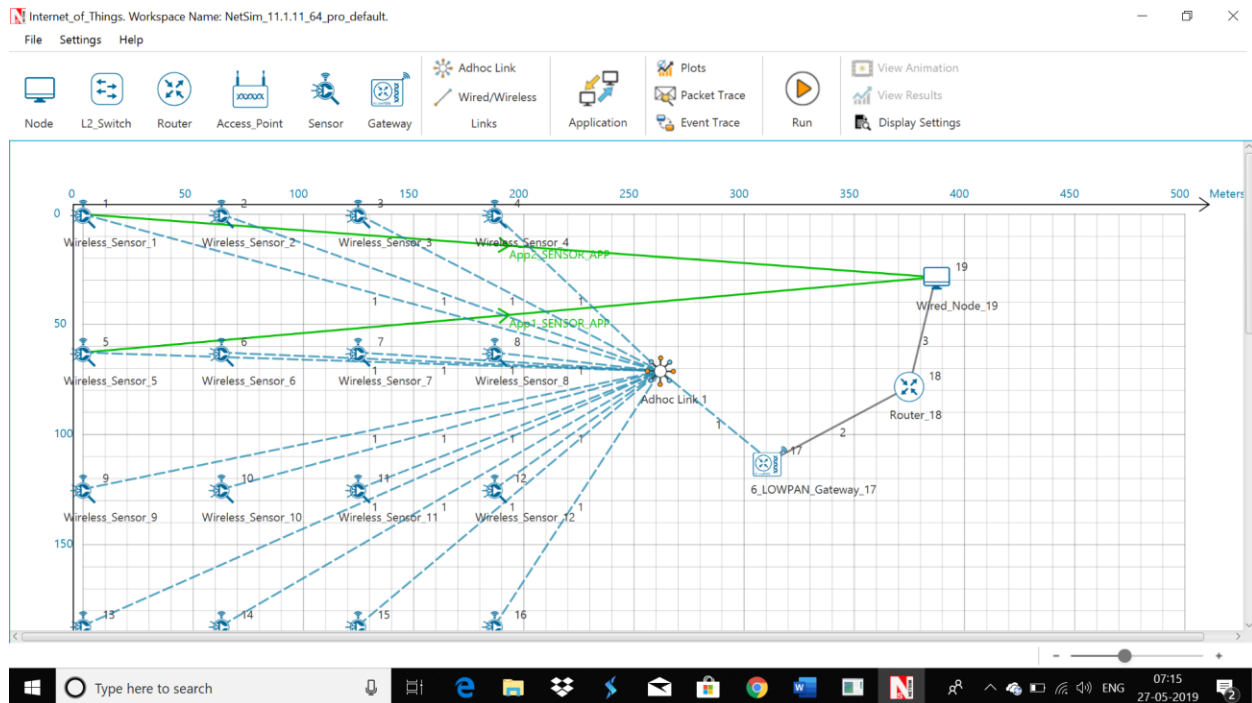
The first window which you see is a 'fast config' window



This allows for configuration of the "sensor network" part of the IOT. The number of sensors and their placement can be done.

We then create a small IOT network by adding a gateway, a router and a node (server). We drop one gateway from the tool bar onto the grid. Connect this to the adhoclink1 using an adhoc link. Subsequently we drop one router and one node. Then we connect the router to the gateway using a link and the router to the node using a link

We configure sensor application from two sensors to the node (server). We click on application and add one application thus having a total of 2 applications. We configure app from sensor 1 to server (node 19 in the screenshot) and another from sensor 5 to server.



We now run the simulation and see the results
 <Run the simulation and show results as explained previously>

Writing your own code in NetSim <Ensure Visual studio 2019, community edition or higher, is installed>

In the second part of this demonstration, we will show how easy it is to modify the underlying source code of NetSim

Go to Open Simulation -> Workspace Options and click on the Open Code button. This will automatically open the protocol source codes in visual studio. Inside Solution Explorer pane in Visual Studio double click on the protocol which you want to modify.

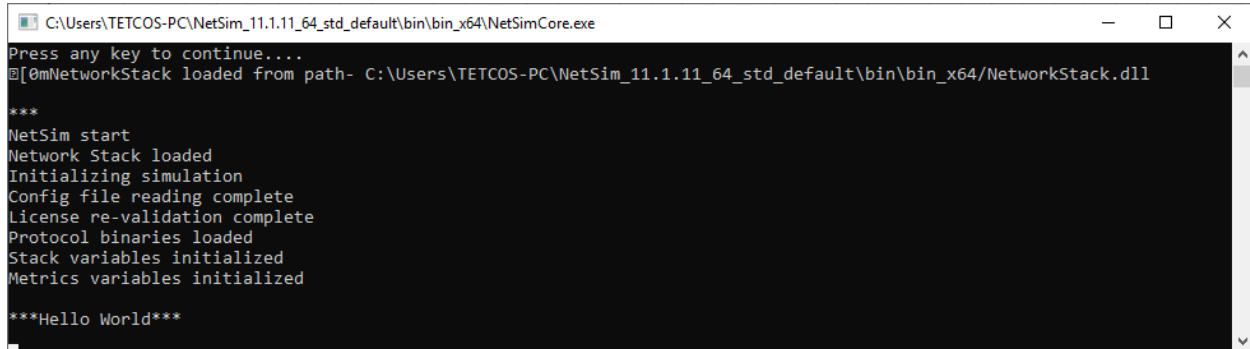
For this demo I will show editing TCP protocol. These source codes are implemented as per relevant International Standards. <Open TCP.h> This code is as per TCP RFCs listed here. <scroll down> Here you can see the TCP parameters and the variables declared as per the RFC.

- Now we open the TCP.C file. Inside the init() initialization function, we will add,

```
fprintf(stderr, "\n***Hello World***\n");
_getch ();
```
- This is to print "Hello World" in the Console.
- Now we had completed editing the code which is step 1. <set x64 or win32 configuration depending on NetSim build installed> Then we rebuild the TCP project by right clicking on the TCP project and select rebuild. Once the build is successful, we move on to the next step.

We will now reopen the previously saved Internetwork scenario which involves TCP protocol <Go to Open Simulation and click on the previously saved Internetwork example>. We will now run the simulation.

NetSim displays a warning indicating code modifications and further as we proceed, we can see the “Hello World” message. As you can see editing source code is extremely easy for users.



```
C:\Users\TETCOS-PC\NetSim_11.1.11_64_std_default\bin\bin_x64\NetSimCore.exe
Press any key to continue....
[0mNetworkStack loaded from path- C:\Users\TETCOS-PC\NetSim_11.1.11_64_std_default\bin\bin_x64\NetworkStack.dll
***
NetSim start
Network Stack loaded
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Metrics variables initialized
***Hello World***
```

NetSim also provides options for interfacing with external software's. Users can also interface NetSim with MATLAB, with SUMO for VANETS, Wireshark for packet capture, etc.

MATLAB Interfacing with NetSim – Dynamic Clustering File Exchange Project:

Clustering in WSN:

Clustering is the process partitioning a group of sensor into small numbers of clusters. In environments where the sensors are mobile clusters cannot be static. Like cluster heads in each cluster are elected dynamically, the members in each cluster also need to be dynamically identified. Therefore the size of each cluster is not fixed and can vary depending on the position of the sensors.

Dynamic Clustering helps in efficiently grouping sensors into clusters dynamically. There is no fixed cluster size and the sensors are divided into the required number of clusters with members of each cluster calculated dynamically.

Clustering using k-means algorithm:

kmeans(X,k) partitions the points in the n-by-p data matrix X into k clusters. This iterative partitioning minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. Rows of X correspond to points, columns correspond to variables. kmeans returns an n-by-1 vector IDX containing the cluster indices of each point. By default, kmeans uses squared Euclidean distances. When X is a vector, kmeans treats it as an n-by-1 data matrix, regardless of its orientation.

The sensor positions and number of clusters,

X - a matrix containing the x, y coordinates of the sensors in the scenario

k- the number of clusters

are passed to k-means algorithm.

[IDX,C] = kmeans(X,k)

IDX – Contains the cluster id's of each sensor (i.e) the cluster to which the sensor belongs.

C – Centroids of each cluster

Clustering using Fuzzy C-Means Algorithm:

Fuzzy c-means (FCM) is a data clustering technique in which a dataset is grouped into n clusters with every data point in the dataset belonging to every cluster to a certain degree. For example, a certain data point that lies close to the center of a cluster will have a high degree of belonging or membership to that cluster and another data point that lies far away from the center of a cluster will have a low degree of belonging or membership to that cluster.

Cluster head election based on distance from Centroid:

After grouping the sensors into different clusters, the cluster heads are determined based on the distance between the sensor and the centroid of the cluster to which it belongs.

The sensor which is closer to the centroid will be elected as the cluster head. Here the position values (i.e. value of x-coordinate and y-coordinate) of each sensor are passing from NetSim to MATLAB as a sole parameter.

Cluster head election based on distance and power:

After grouping the sensors into different clusters, the cluster heads are determined based on the distance between the sensor and the remaining power of each sensor. After that the sensors are assigned in respective cluster.

The sensor which is closer to the centroid and has the more power than other sensor will be elected as the cluster head. Here the position values (i.e. value of x-coordinate and y-coordinate) of each sensor and power are passing from NetSim to MATLAB as a sole parameter.

Dynamic Clustering in NetSim with MATLAB Interfacing:

Dynamic Clustering is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and k-means algorithm that is implemented in MATLAB is used to dynamically perform clustering of the sensors into n number of clusters.

In addition to clustering we also determine the cluster head of each cluster mathematically in MATLAB. The distance of each sensor from the centroid of the cluster to which it belongs is calculated. Then the sensor which has the least distance is elected as the cluster head.

From MATLAB we get the cluster id of each sensor, cluster heads of each cluster and the size of each cluster.

All the above steps are performed periodically which can be defined as per the implementation. Each time the cluster members and the cluster heads are determined based on the current position and they are not fixed.

The codes required for the mathematical calculations done in MATLAB are written to a **clustering.m** file as shown below:

```

clustering.m - Notepad
File Edit Format View Help
%*****
% Copyright (C) 2016
% TETCOS, Bangalore, India
%
% Tetcos owns the intellectual property rights in the Product and its content.
% The copying, redistribution, reselling or publication of any or all of the
% Product or its content without express prior written consent of Tetcos is
% prohibited. Ownership and / or any other right relating to the software and all
% intellectual property rights therein shall remain at all times with Tetcos.
%
% Author: Dhruvang
%
% -----
function [A,B,C] = clustering(x,scount,num_cls,power,max_energy)
% changed clustering function. New parameter power: column vector of
% remaining power for each device
% s_count is sensor_count

% Clustering_Method = 1          KMeans using distance      Fuzzy C Means using distance
%                               = 2                          = 3
%                               = 3          KMeans using distance and power
%                               = 4          Fuzzy C Means using distance and power
Clustering_Method = 1;

% save dynamic_clustering.mat

% Change here for different algorithm
if(Clustering_Method == 1 || Clustering_Method == 3)
    [IDX,C]= k_means(x,num_cls);
else
    [IDX,C]= fuzzy(x,num_cls);
end

```

The **clustering.m** file can be run in four different modes cluster head election.

A **Dynamic_Clustering.c** file is added to the DSR project which contains the following functions:

```

28 #include "DSR.h"
29 #include "List.h"
30 #include "../BatteryModel/BatteryModel.h"
31 #include "../ZigBee/802_15_4.h"
32 #define NUMBEROFCLUSTERS 4
33
34 int *ClusterElements;
35 int CH[NUMBEROFCLUSTERS];
36 int CL_SIZE[NUMBEROFCLUSTERS];
37
38 int fn_NetSim_dynamic_clustering_CheckDestination(NETSIM_ID nDeviceId, NETSIM_ID nDestinationId)
39
40
41 int fn_NetSim_dynamic_clustering_GetNextHop(Netsim_EVENTDETAILS* pstruEventDetails){ ... }
42
43 int fn_NetSim_dynamic_clustering_IdentifyCluster(int DeviceId){ ... }
44
45 int fn_NetSim_dynamic_clustering_run(){ ... }
46
47 int fn_netsim_dynamic_form_clusters(double* cl_id, double* cl_size){ ... }
48
49 int fn_netsim_assign_cluster_heads(double* cl_head){ ... }
50
51 void fn_NetSim_Dynamic_Clustering_Init(){ ... }

```

fn_NetSim_dynamic_clustering_CheckDestination()

This function is used to determine whether the current device is the destination.

fn_NetSim_dynamic_clustering_GetNextHop()

This function statically defines the routes within the cluster and from cluster to sinknode. It returns the next hop based on the static routing that is defined.

fn_NetSim_dynamic_clustering_IdentifyCluster()

This function returns the cluster id of the cluster to which a sensor belongs.

fn_NetSim_dynamic_clustering_run()

This function makes a call to MATLAB interfacing function and passes the inputs from NetSim (i.e) the sensor coordinates, number of clusters and the sensor count.

fn_netsim_dynamic_form_clusters()

This function assigns each sensor to its respective clusters based on the cluster id's obtained from MATLAB.

fn_netsim_assign_cluster_heads()

This function assigns the cluster heads for each cluster based on the cluster head id's obtained from MATLAB.

fn_NetSim_Dynamic_Clustering_Init()

This function initializes all parameter values.

Static Routing:

Static Routing is defined in such a way that the sensors in the cluster send the packets to the cluster head. The cluster head then directly sends the packets to the destination (sinknode).

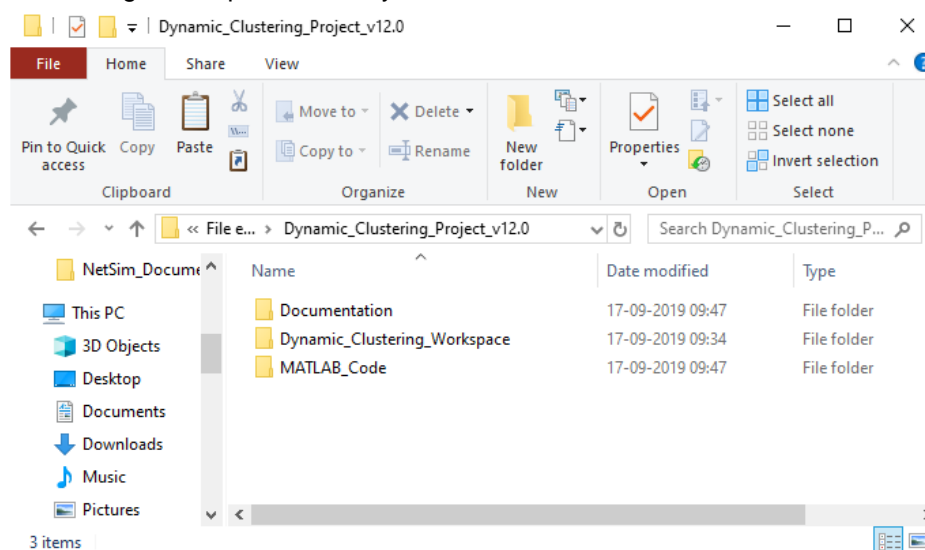
If the current sensor is the source device and if it is not a cluster head then its next hop is its cluster head.

If the current sensor is the source device and if it is a cluster head then its next hop is the destination (i.e) the sinknode.

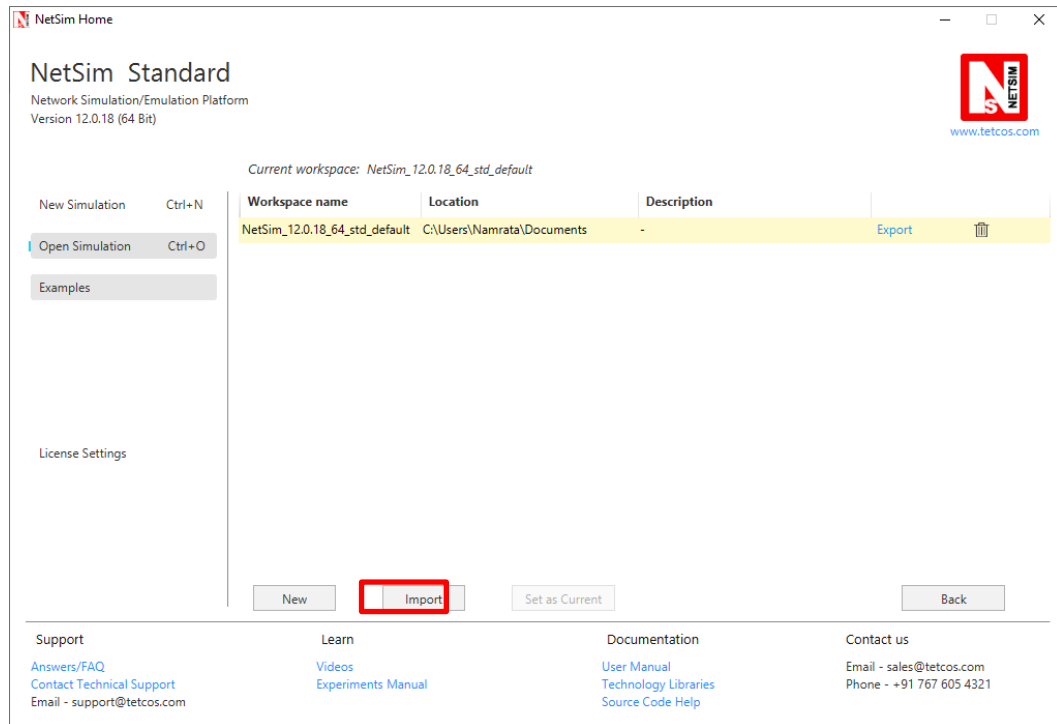
If the current sensor is not the source then the packet is sent to the destination (i.e) the sinknode.

Steps:

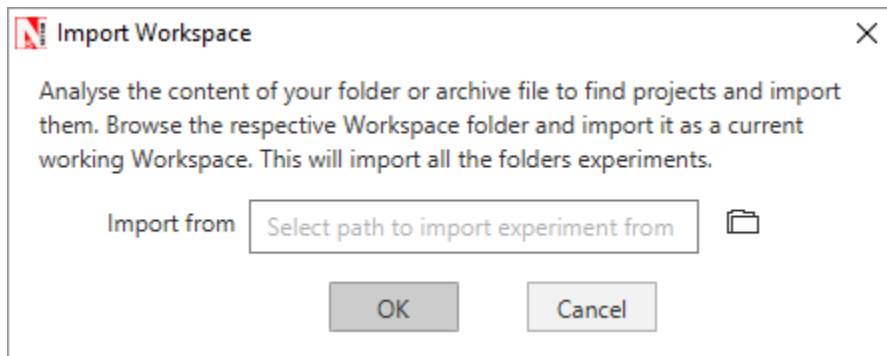
1. The downloaded project folder contains the folders Documentation, MATLAB_Code and Dynamic_Clustering_Workspace directory as shown below:



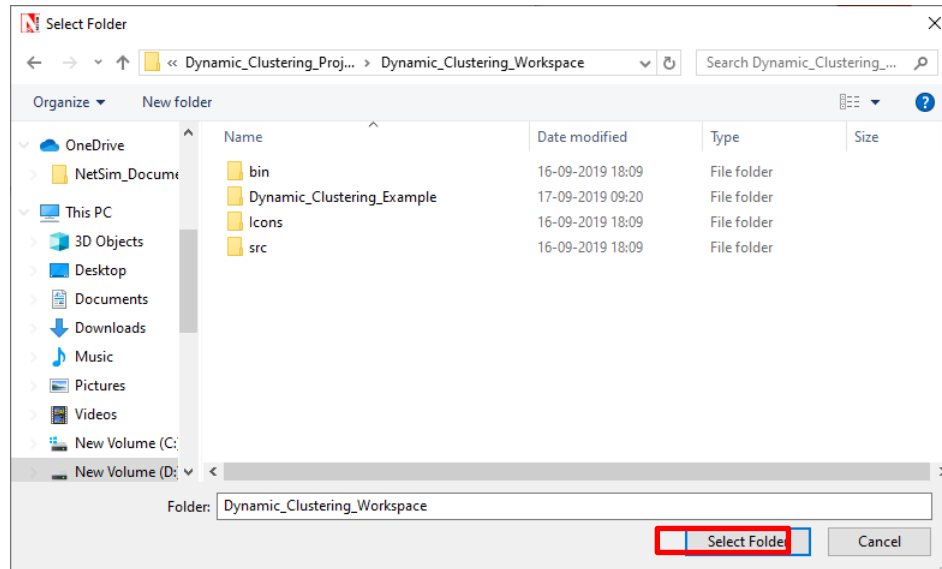
2. Import Dynamic_Clustering_Workspace by going to Open Simulation->Workspace Options->More Options in NetSim Home window. Then select Import as shown below:



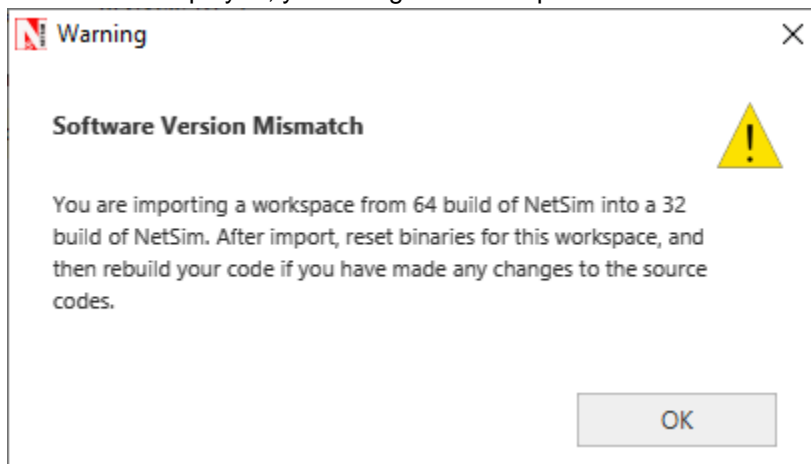
- It displays a window where users need to give the path of the workspace folder and click on OK as shown below:



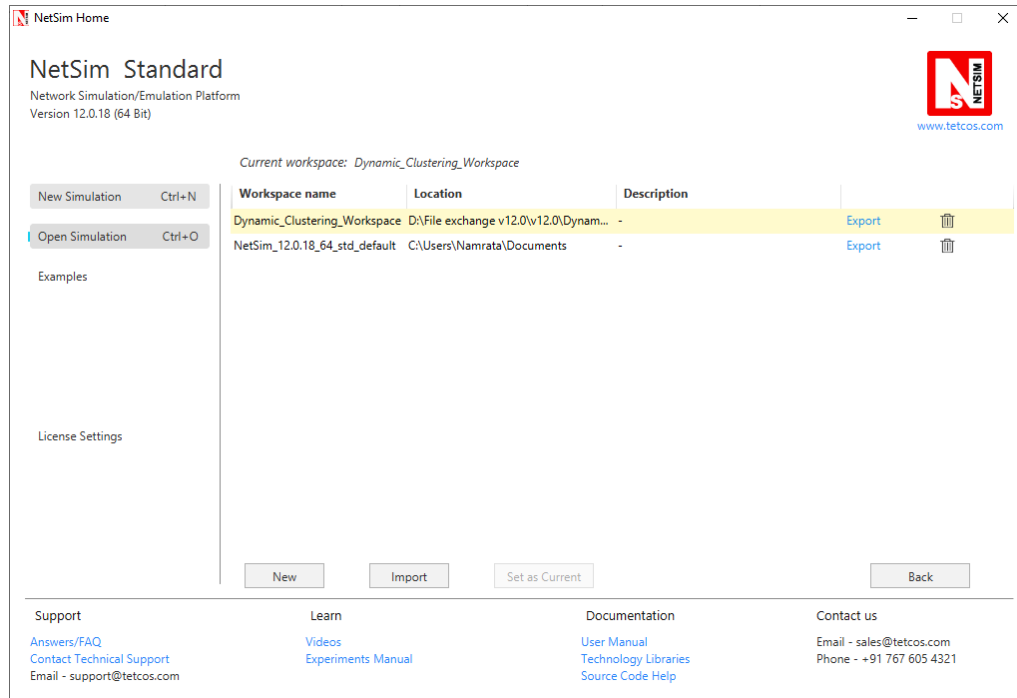
- Browse to the Dynamic_Clustering_Workspace folder and click on select folder as shown below:



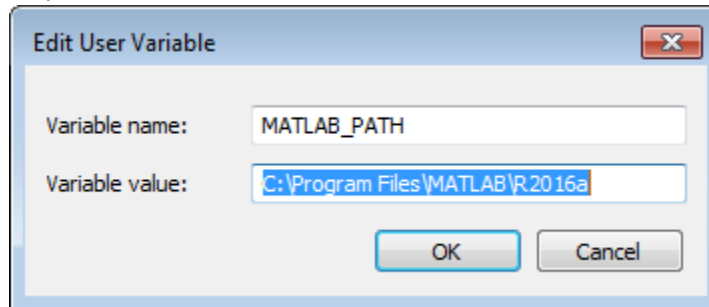
5. After this click on OK button in the Import Workspace window.
6. While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



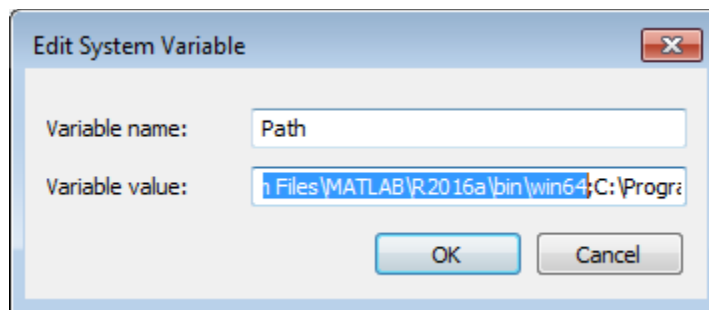
7. The Imported workspace will be set as the current workspace automatically. To see the imported workspace, click on Open Simulation->Workspace Options->More Options as shown below:



8. Create a user variable with the name of MATLAB_PATH and provide the path of the installation directory of user's respective MATLAB version.



9. Make sure that the following directory is in the PATH(Environment variable)
<Path where MATLAB is installed>\bin\win64



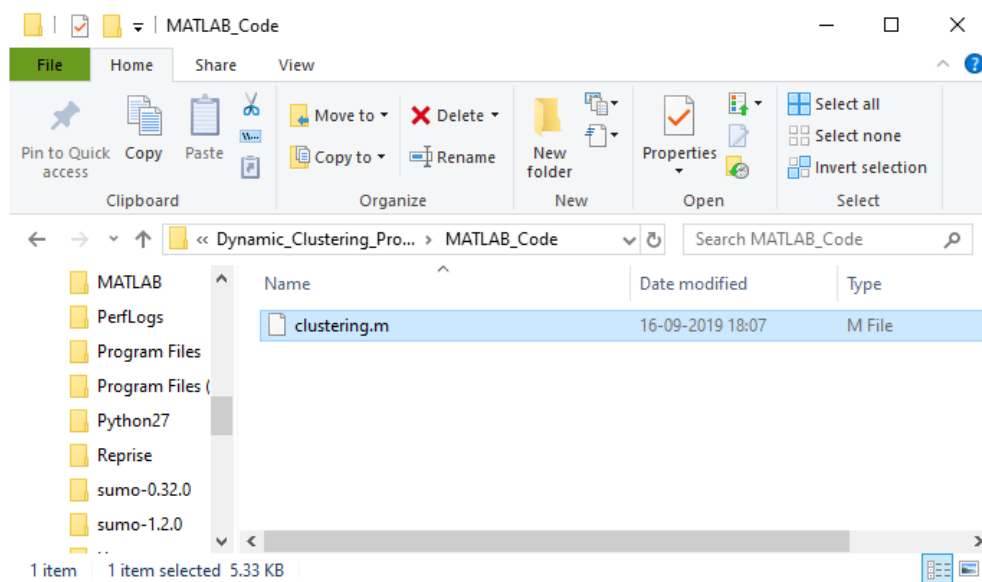
Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

10. Open Command prompt as admin and execute the command "matlab -regserver". This will register MATLAB as a COM automation server and is required for NetSim to start MATLAB automation server during runtime.

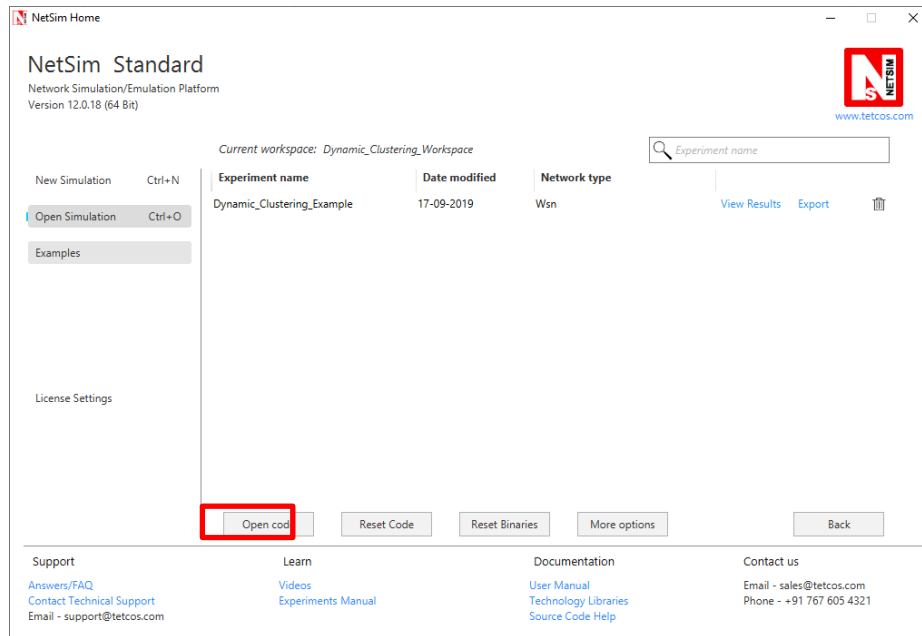
```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>matlab -regserver_
```

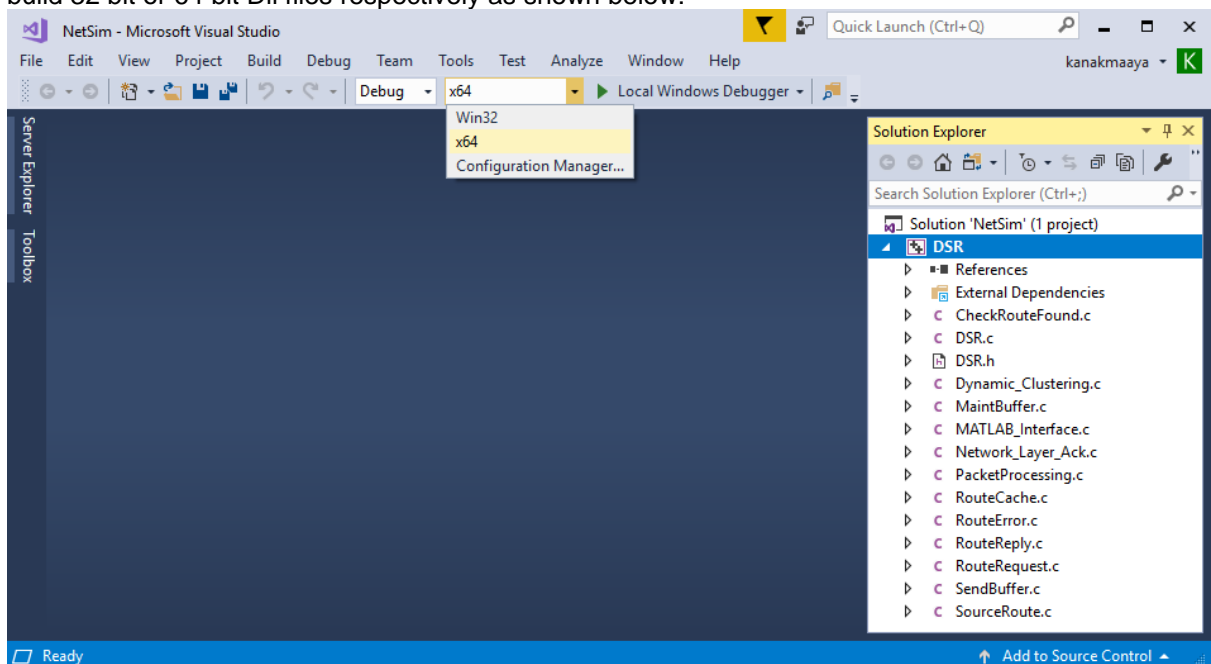
11. Place **clustering.m** present in the MATLAB_Code folder inside the root directory of MATLAB. For Example: “C:\Program Files\MATLAB\R2016a”.



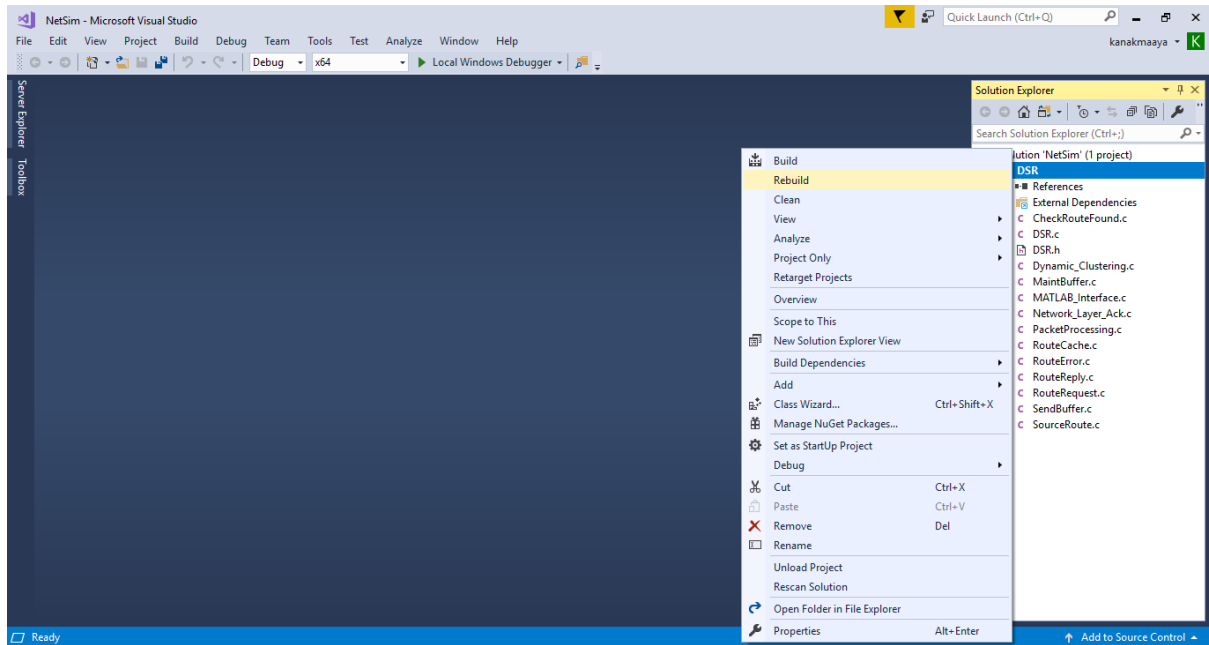
12. Open the Source codes in Visual Studio by going to Open Simulation-> Workspace Options and Clicking on Open code button as shown below:



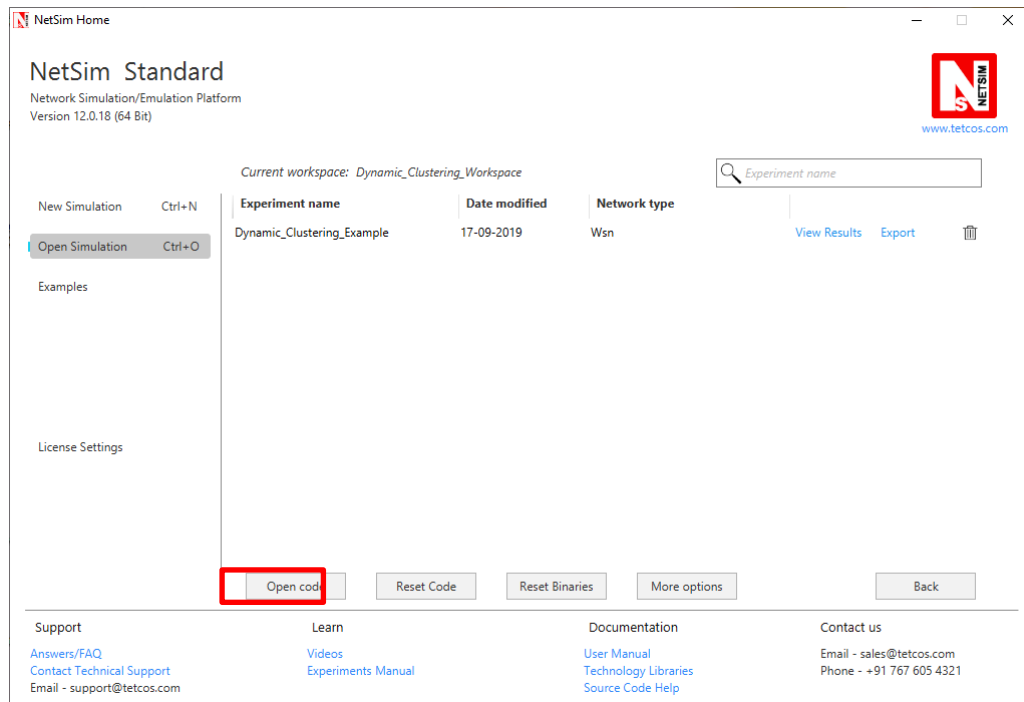
13. Under the DSR project in the solution explorer you will be able to see that **MATLAB_Interface.c** and **Dynamic_Clustering.c** files which contain source codes related to interactions with MATLAB and handling clustering in NetSim respectively.
14. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit DLL files respectively as shown below:



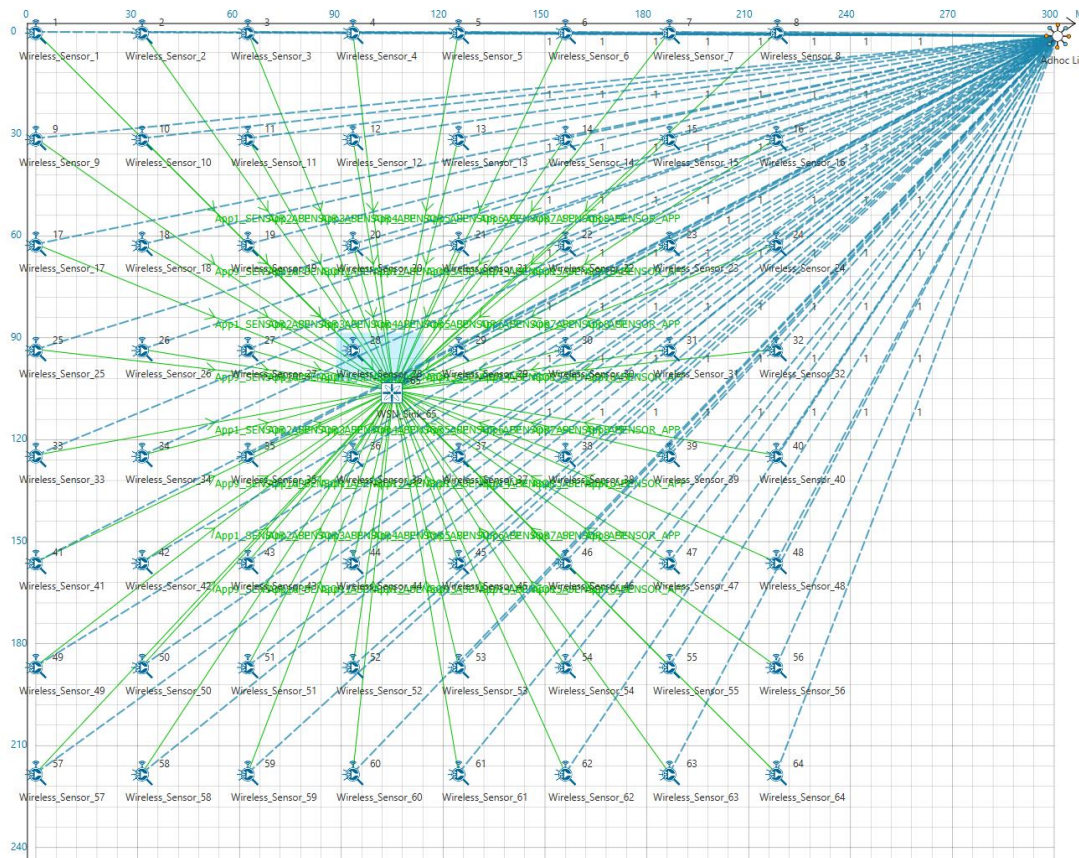
15. Right click on the DSR project in the solution explorer and select Rebuild.



16. Upon successful build modified libDSR.dll file gets automatically updated in the directory containing NetSim binaries.
17. Run NetSim as Administrative mode.
18. Then Dynamic_Clustering_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the Dynamic_Clustering_Example that is present under the list of experiments as shown below:



19. The saved network scenario consisting of 64 sensors uniformly distributed in the grid environment along with a sink node forming a Wireless Sensor Network. Traffic is configured from each sensor node to the Sink Node.



20. Run the Scenario. You will observe that as the simulation starts in NetSim, MATLAB gets initialized and graph associated with energy consumption in the sensor network is plotted during runtime.

Analysis:

A total of 64 sensors are placed evenly on the grid environment and each sensor is set to have equal initial energy.

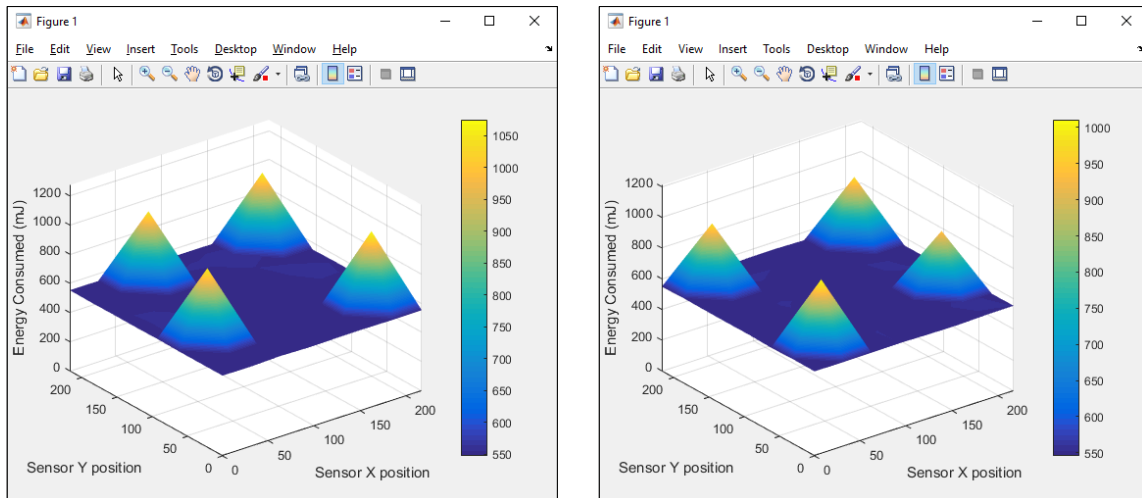
At the end of the simulation, NetSim provides Battery Model Metrics which provides detailed information related to energy consumption in each sensor node with respect to transmission, reception, idle mode, sleep mode etc as shown below:

Battery model_Table						
Batter model <input checked="" type="checkbox"/> Detailed View						
Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)
WIRELESS_SENSOR_1	6480.000000	558.318835	5921.681165	20.038487	0.000000	538.280348
WIRELESS_SENSOR_2	6480.000000	556.923016	5923.076984	18.593404	0.000000	538.329613
WIRELESS_SENSOR_3	6480.000000	558.504945	5921.495055	20.231165	0.000000	538.273780
WIRELESS_SENSOR_4	6480.000000	557.202180	5922.797820	18.882420	0.000000	538.319760
WIRELESS_SENSOR_5	6480.000000	556.457743	5923.542257	18.111709	0.000000	538.346034
WIRELESS_SENSOR_6	6480.000000	557.853562	5922.146438	19.556793	0.000000	538.296770
WIRELESS_SENSOR_7	6480.000000	557.481344	5922.518656	19.171437	0.000000	538.309907
WIRELESS_SENSOR_8	6480.000000	555.806361	5924.193639	17.437337	0.000000	538.369024
WIRELESS_SENSOR_9	6480.000000	557.574399	5922.425601	19.267776	0.000000	538.306623
WIRELESS_SENSOR_10	6480.000000	629.662276	5850.337724	35.163691	58.563994	535.934592

This information can also be obtained at different points of simulation time either to log or to send to other external tools. The battery information and the position coordinates are passed to MATLAB periodically for clustering (number of cluster is set to 4), cluster head election and to obtain energy consumption plots.

Cluster head election using distance alone as a parameter:

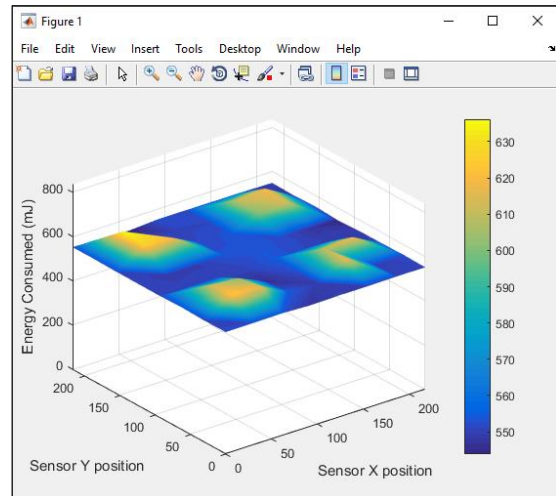
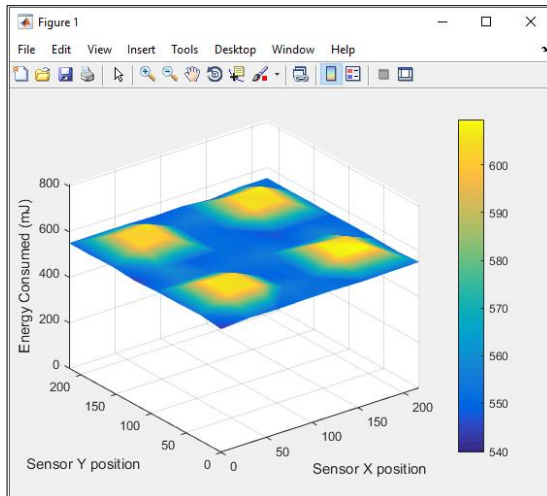
Running simulations with Clustering Method set to 1 and 2 in the **clustering.m** file will provide energy consumption plots for kmeans and fuzzy c-means algorithms respectively as shown below:



As it is seen from the plot, there are 4 peaks in the plot corresponding to higher energy consumption in the nodes in the center of the cluster, as they always become the cluster heads. This is because distance is used as a parameter for electing the cluster heads.

Cluster head election using distance and remaining energy as parameters:

Running simulations with Clustering Method set to 3 and 4 in the **clustering.m** file will provide energy consumption plots for kmeans and fuzzy c-means algorithms respectively as shown below:



In the initial phase the plot resembles the previous one. However as the time passes, it can be observed that the power is consumed by all the sensors at approximately the same rate.

There are no sharp peaks in this plot unlike the previous one because modified K-means takes into account the power level of each sensor and thus sensors other than those in the center of the cluster will also get a chance to be elected as the cluster head in its respective cluster.

Step 7:

VANET – SUMO Interfacing with NetSim:

Note: NetSim VANET component is available only in standard and pro version

NetSim VANET library supports the following protocols

- IEEE 802.11p, IEEE 1609 WAVE
- Layer 3 Routing – AODV, DSR, OLSR, ZRP
- PHY Layer RF Propagation
 - Pathloss
 - Shadowing
 - Fading
- Source C Code
- Automatic import of road network and vehicles from SUMO
- Wide range of output metrics including Delay, Throughput, Error, Retransmission, etc.
- Interfacing between SUMO & NetSim via Traffic control interface (TraCI).

Source C Code:

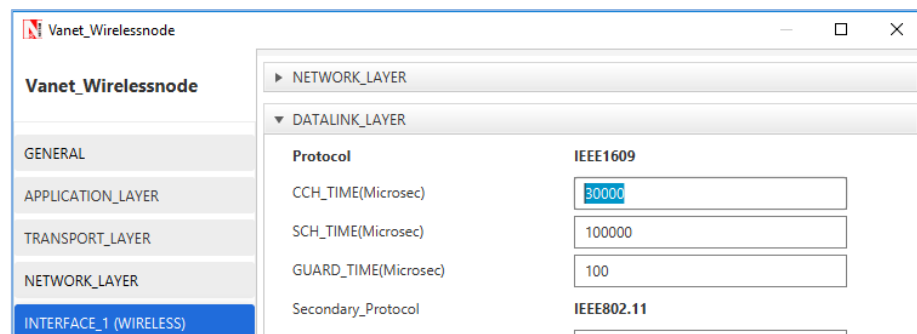
- Source code related to interfacing of SUMO and NetSim is available in Sumo_interface.c file inside mobility folder/project.

- NetSim VANET library runs MANET routing protocols in Layer 3. These are AODV, DSR, OLSR, and ZRP.
- The following dlls are executed when VANET simulations run:
 - libMobility.dll
 - libAODV.dll (or) libDSR.dll (or) libZRP.dll
 - libIEEE802.11.dll
- IEEE 1609
- DSRC – J2735

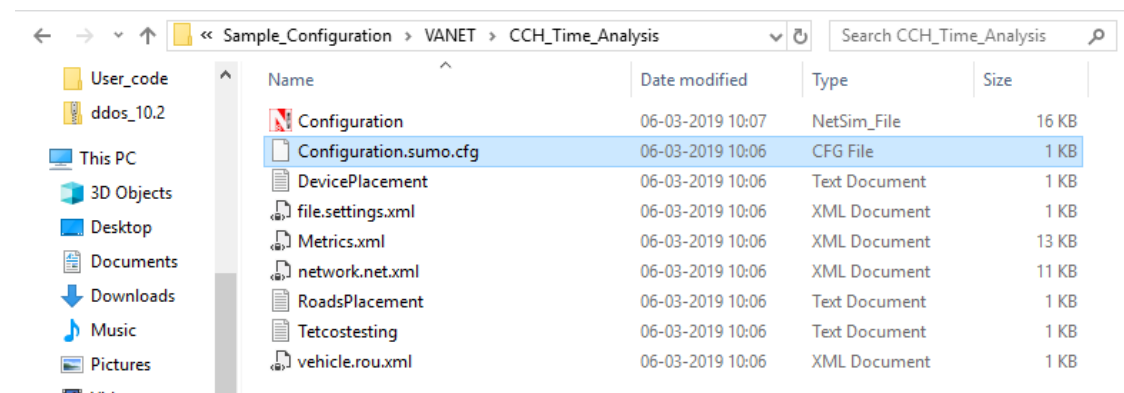
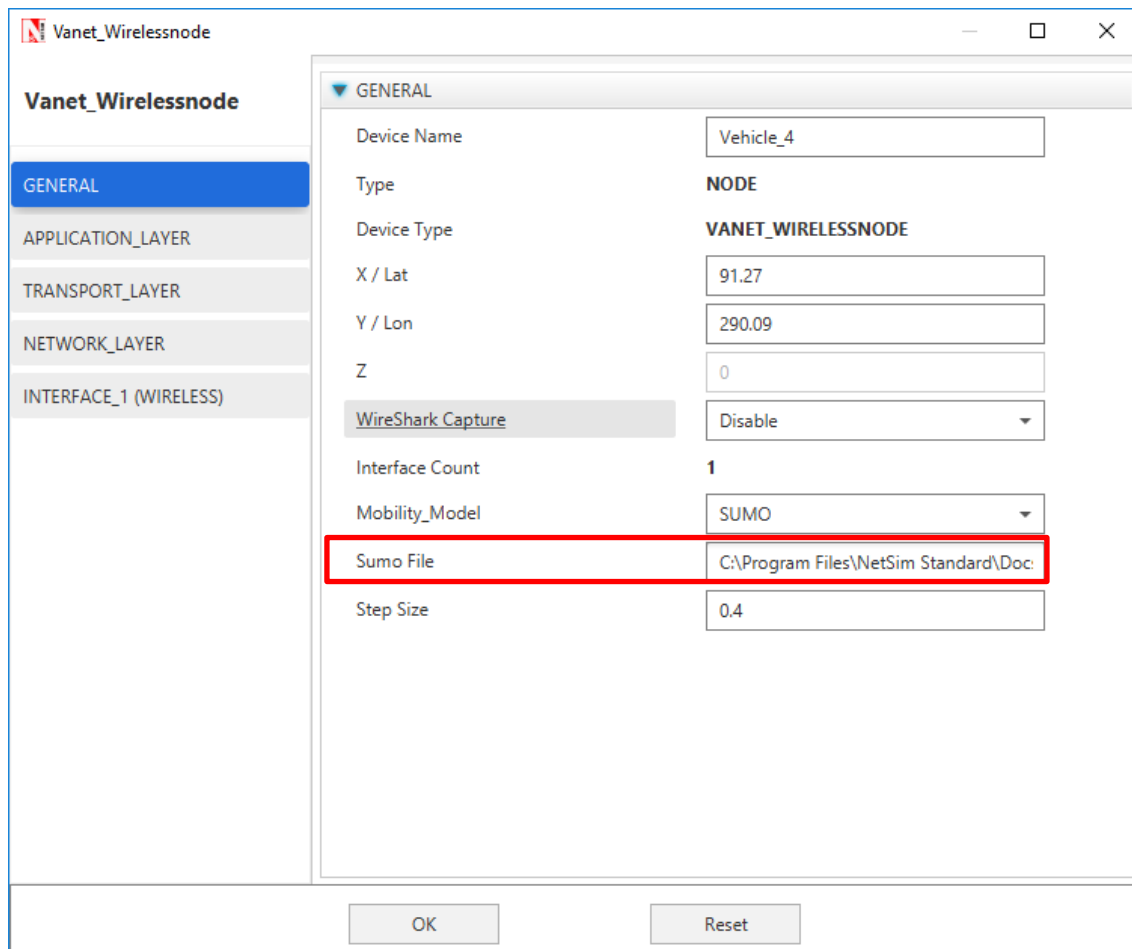
Open NetSim, Select Examples->VANETs->CCH-Time-Analysis

Settings done for this sample experiment:

- In INTERFACE_1(WIRELESS) → Data Link Layer, CCH_TIME_INTERVAL – 30000(Microsec)



- Set Application BSM (Basic_Safety_Message) with 3Mbps generation rate
 - Packet size – 100 Bytes
 - IAT – 266 (Microsec)
- Open Wireless Node/Vehicle General Properties and select the Configuration.sumo.cfg file from the Docs folder of NetSim Install Directory < C:\Program Files\NetSim Standard\Docs\Sample_Configuration\VANET\CCH_Time_Analysis> as shown below



- Enable Packet trace option
- Run simulation and note down the number of data packets transmitted (BSM Packets) from Packet Trace by filter PACKET_TYPE to Basic_Safety_Message and throughput for sample 1
- Modify the CCH_TIME_INTERVEL to 50000, 70000 microseconds for sample 2 and 3 respectively

CCH_Time (micro seconds)	Throughput (Mbps)
30000	0.65
50000	0.94
70000	1.15

Inference: The BSM application is transmitted on CCH. Hence as we increase the CCH time, the throughput of the BSM application increases.

Step 8:

5G NR mmWave:

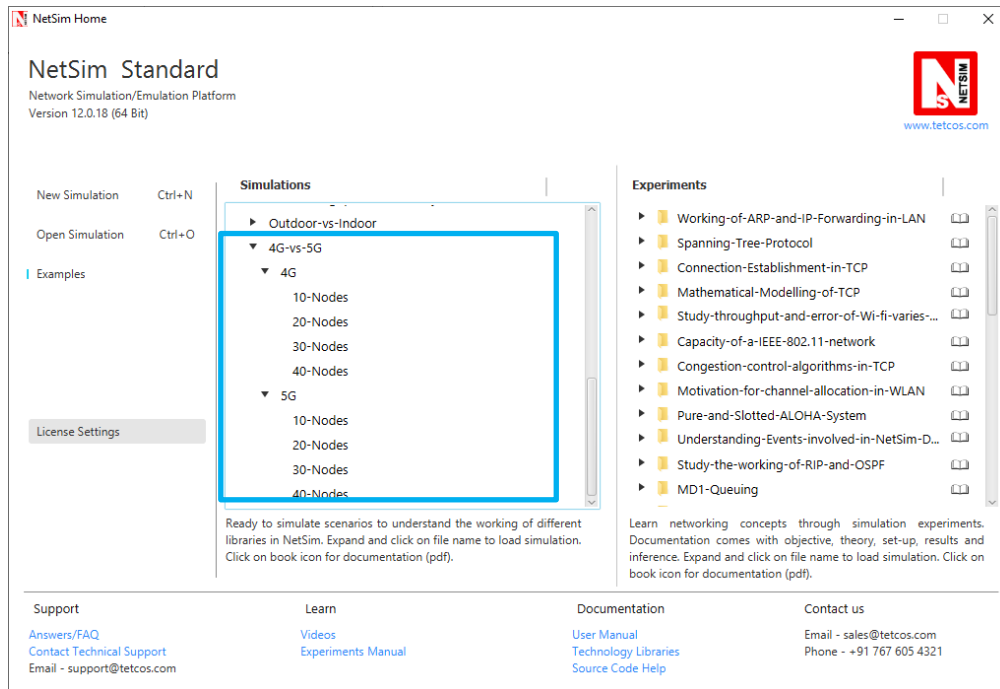
NetSim 5G library features full stack, end-to-end, packet level simulation of 5G mmWave networks. The 5G library is based on Rel 15 / 3GPP 38.xxx series.

NetSim 5G library models all layers of the protocol stack as well as applications running over the network. This 5G library is architected to connect to the base component of NetSim (and in turn to other components) which provides functionalities such as TCP/IP stack protocols, Wireless protocols, Routing algorithms, Mobility, Output Metrics, Animation, Traces etc.

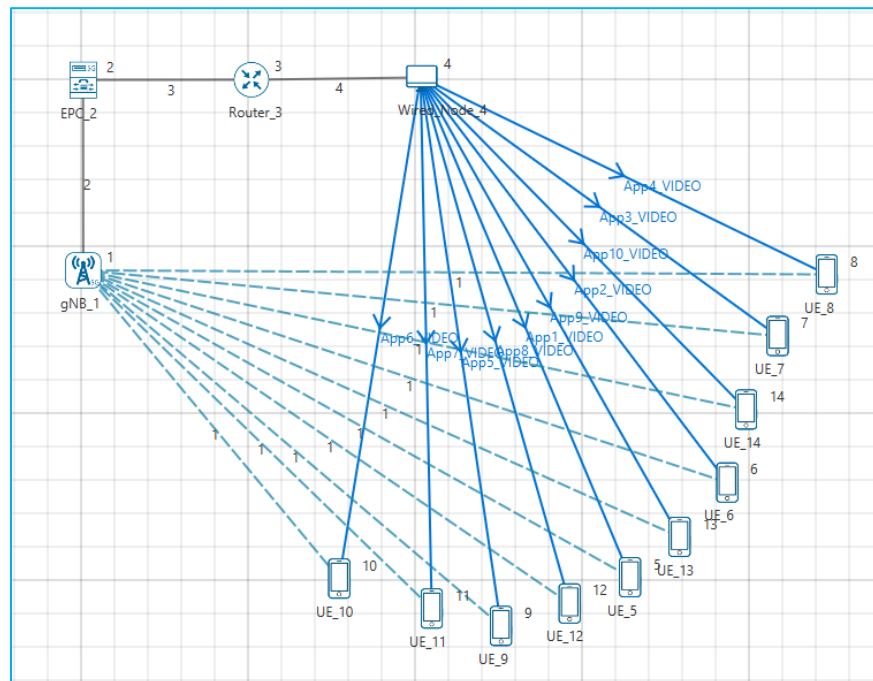
5G NR mmWave comes with a palette of various devices like Wired & Wireless Nodes, L2 Switch & Access Point, EPC (Enhanced Packet Core) & Router, gNB (Equivalent of eNB in 5G), UE (User Equipment), and Building.

4G vs. 5G

Open NetSim, Select Examples ->5G NR-mmWAVE -> 4G-vs-5G



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



4G:

Settings done in example config file:

1. Set grid length as 1000m from Environment setting

- Set the following property as shown in below given table:

gNB Properties -> Interface (LTE_NR)	
Outdoor scenario	URBAN_MACRO
Channel Characteristics	PATHLOSS_ONLY
LOS Mode	USER_DEFINED
LOS Probability	0
Frequency Range	FR1
MU	0
Channel Bandwidth (MHz)	15
MCS Table	QAM64

- Frequency range FR1, $\mu = 0$, Bandwidth = 15 Mhz with QAM 64 MCS table represent a 4G configuration
- Disabled TCP (Transport Layer) in all the devices.
- Set Uplink speed and Downlink speed as 10000 Mbps.
- Set 10 applications Source id as 4 and Destination id as 5, 6, 7, 8, 9, 10, 11, 12, 13, and 14 and set the properties as shown below. This would generate 10 Mbps of traffic per user

Application Properties	
Frame Per Sec	100
Pixel Per Frame	100000
Mu	1

- Run simulation for 1 sec. After simulation completes go to metrics window and note down throughput and delay value from application metrics.

Increase number of UE's and number of applications as 20, 30, and 40 and note down throughput and delay value from application metrics.

5G:

Settings done in example config file:

- For the above 4G scenario set the following given properties:

gNB Properties -> Interface (LTE_NR)	
Outdoor scenario	URBAN_MACRO
Channel Characteristics	PATHLOSS_ONLY

LOS Mode	USER_DEFINED
LOS Probability	0
Frequency Range	FR2
MU	3
Channel Bandwidth (MHz)	400
MCS Table	QAM256

- Frequency range FR2, $\mu = 3$, Bandwidth = 400 Mhz with QAM 256 MCS table represent a 5G configuration
- Run simulation for 1 sec. After simulation completes go to metrics window and note down throughput and delay value from application metrics.

Increase number of UE's and number of applications as 20, 30, and 40 and note down throughput and delay value from application metrics.

$$\text{Throughput Per User (Mbps)} = \frac{\text{Sum of throughputs (Mbps)}}{\text{Number of User}}$$

$$\text{Delay Per User } (\mu s) = \frac{\text{Sum of Delays } (\mu s)}{\text{Number of User}}$$

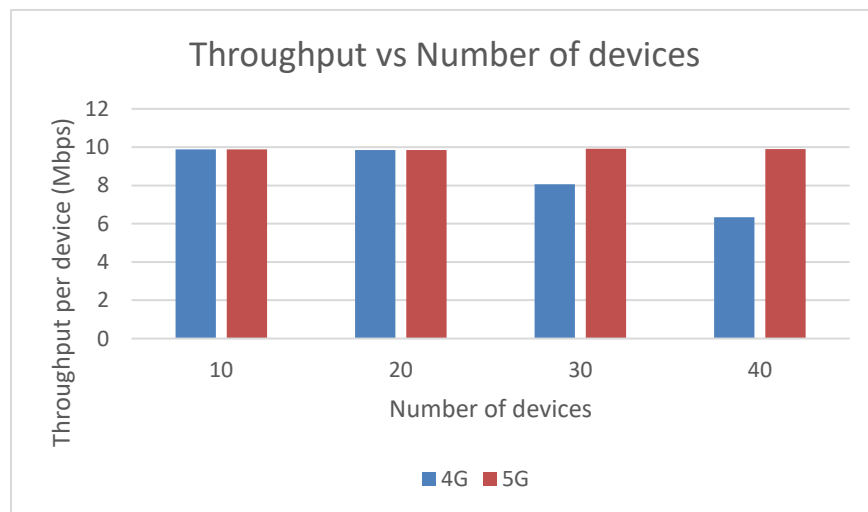
Result: The results are tabulated below

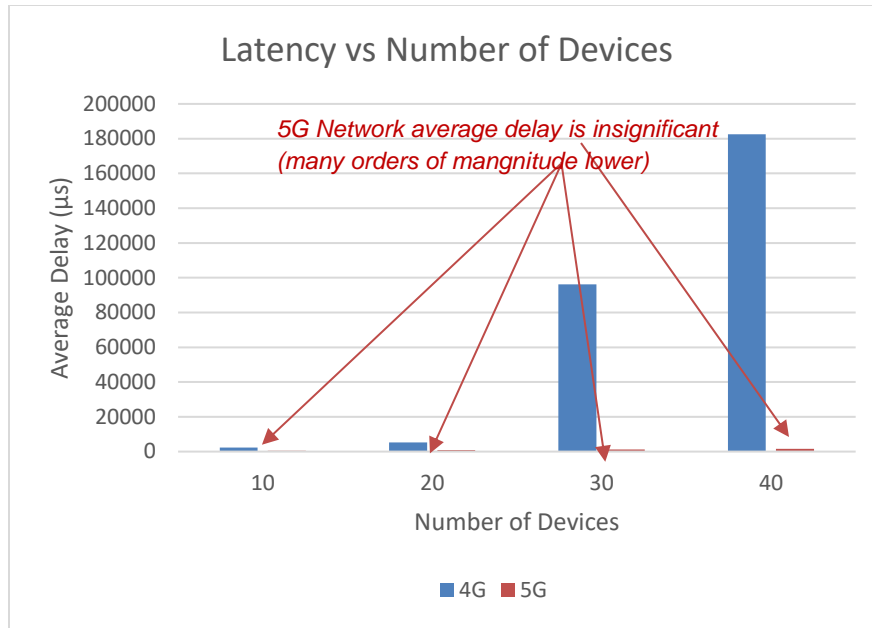
Number of users	4G (10 devices downloading video)		5G (10 devices downloading video)	
	Throughput (Mbps)	Delay (μs)	Throughput (Mbps)	Delay (μs)
1	10.05	1999.00	10.05	408.67
2	9.54	2516.48	9.54	361.76
3	9.67	2437.94	9.67	331.23
4	9.89	2992.24	9.89	511.54
5	9.83	2004.63	9.83	458.04
6	9.93	2011.32	9.93	650.04
7	9.74	2009.29	9.74	436.86
8	10.09	2013.35	10.09	753.14
9	9.90	2978.78	9.90	596.33
10	10.17	2999.00	10.17	788.79
Average per user	9.88	2396.20	9.88	529.64
Aggregate Throughput	98.80	-	98.80	-

Aggregated and Average throughput and delay per user with different values of users is shown below:

Number of Users	4G (Devices downloading video)			5G (Devices downloading video)		
	Throughput (Mbps)		Delay (μ s)	Throughput (Mbps)		Delay (μ s)
	Throughput per user	Aggregate throughgput	Average Delay	Throughput per user	Aggregate	Average Delay
10	9.88	98.80	2396.20	9.88	98.80	529.64
20	9.85	196.93	5290.09	9.85	196.93	857.28
30	8.06	241.72	96200.32	9.91	297.16	1186.93
40	6.34	253.58	182605.65	9.90	395.94	1517.37

As the number of users increases the the throughput per user starts dropping in 4G. However, 5G is able to provide necessary bandwidth for each user to receive at the full rate of 10 Mbps. The latency value increases linearly in 5G whereas the increase is exponential in 4G.





This ends on technical demonstration. And we will now go through some of the documentation available

<show from homepage window>

- **NetSim User Manual**
- **Technology Library Documentation** – Explains features and examples specific to each network technology.
- **NetSim Experiment Manual** - experiment manual with experiments across various technologies
- **Source code help** <show for R & D education customer>

And finally, we will take just a minute to explain all the support which we provide.

- Knowledge Base – Contains hundreds of technical articles, FAQs divided into specialized categories.
- Raise support tickets – Contact our support by writing to us.
- Monthly Webinars - Free webinars which you can attend for learning about new technologies and the latest features in NetSim
- YouTube Videos
- File Exchange Projects - R & D project documentation and codes covering IOT, WSN, MANET, Cognitive Radio and more
- GitHub Repositories – Existing projects can be cloned and forked to use as a base for your project.

Thank you for watching this video