Vehicular Adhoc Networks (VANETs)

Contents

1 Introduction

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).

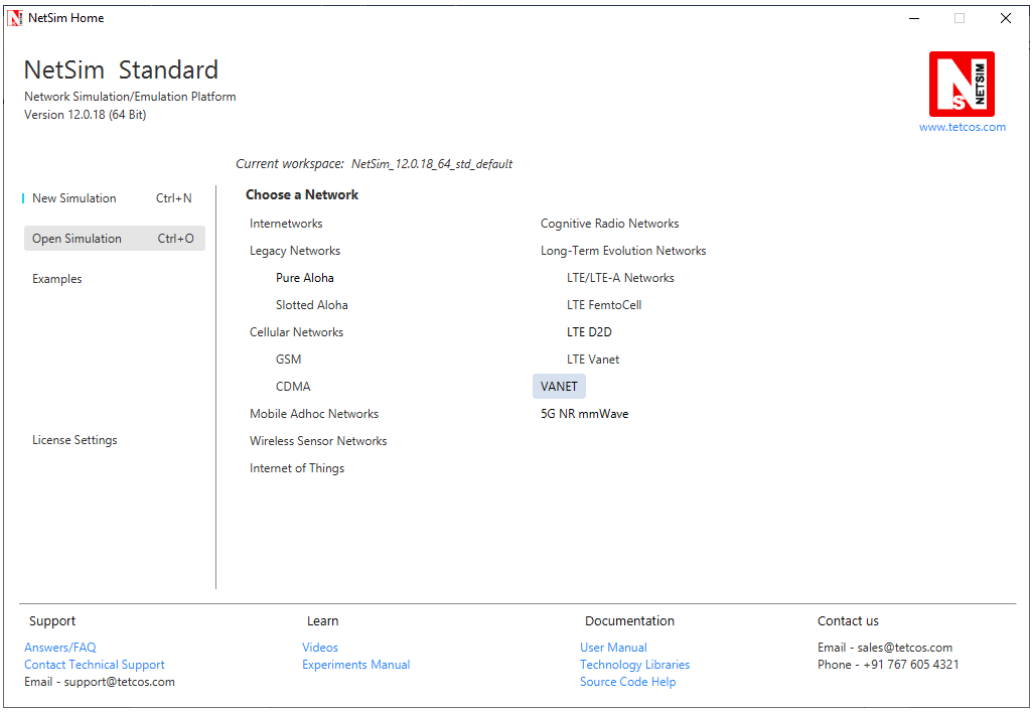
Source C Code:

- 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

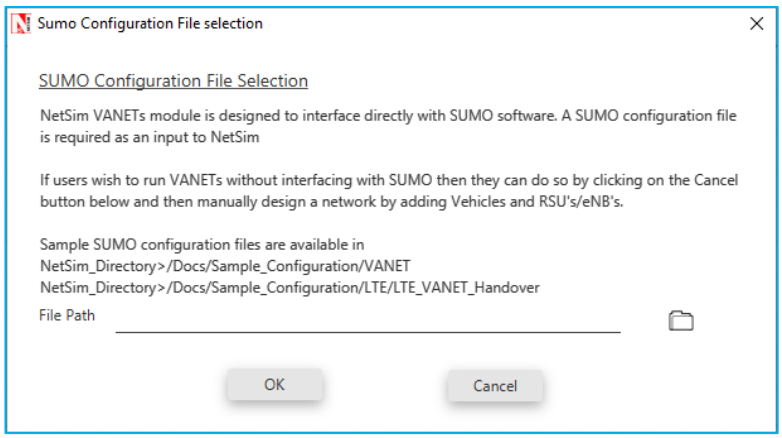
2 Simulation GUI

2.1 Create Scenario

- Open NetSim and click **New Simulation → VANET**



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



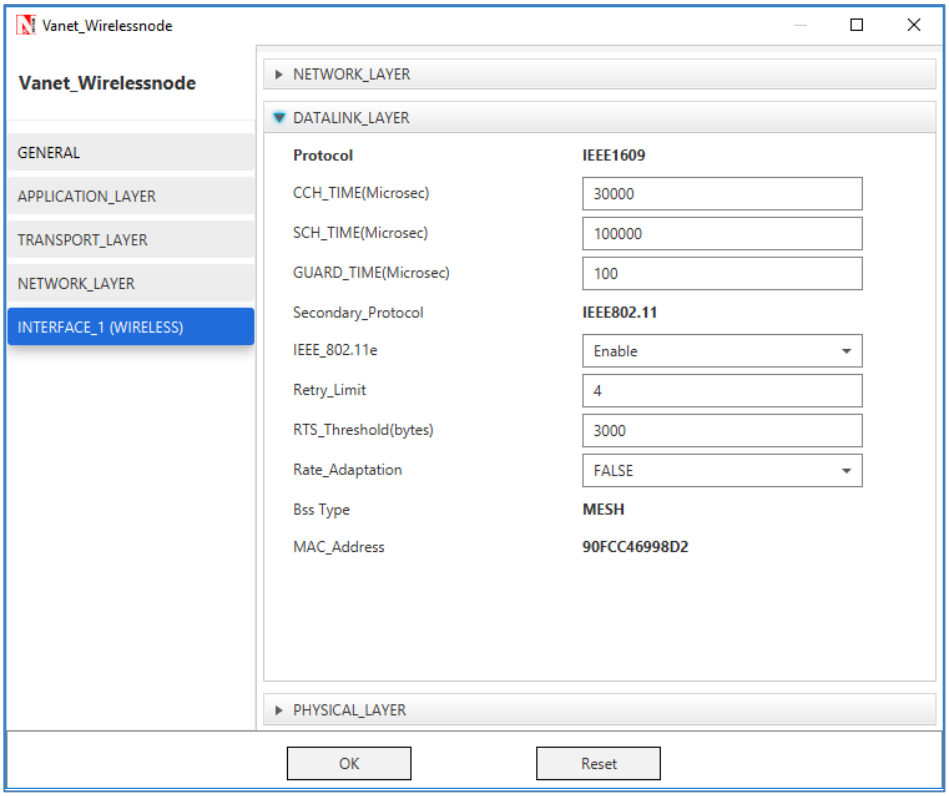
- NetSim VANET module is designed to interface directly with SUMO.
- A SUMO configuration file is required as an input to NetSim.

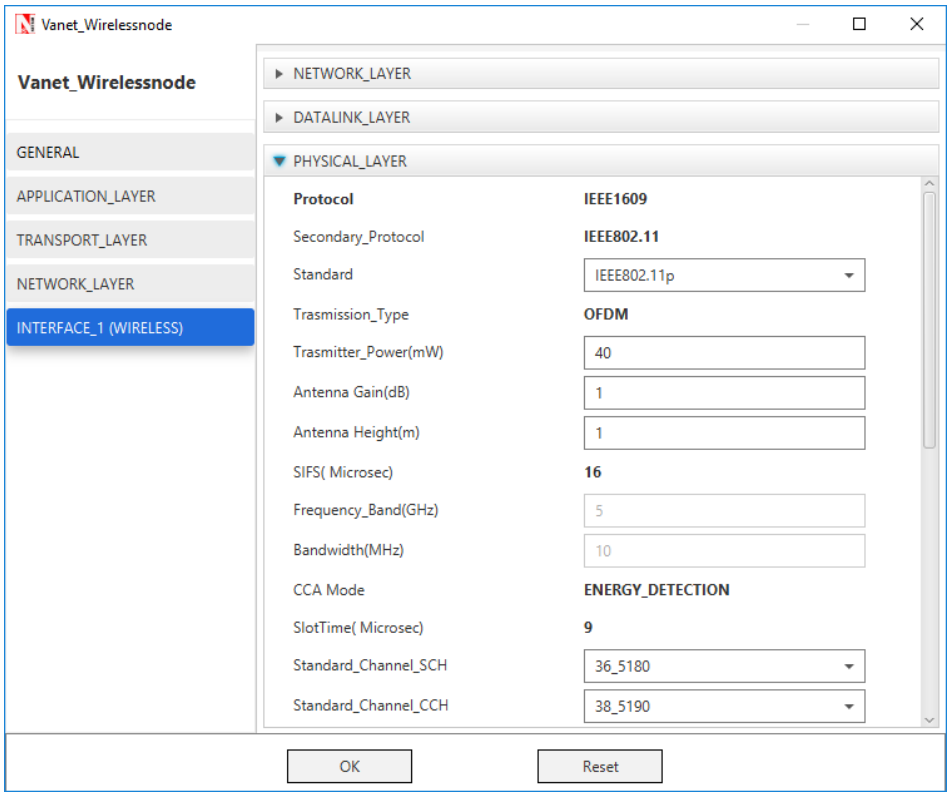
- Sample SUMO configuration files are available inside <NetSim-Installation-Directory>**\Docs\Sample_Configuration\VANET** folder.
- If users wish to run VANET without interfacing with SUMO, then they can do so via **New Simulation → Long-Term Evolution Networks → LTE/LTE-A Networks**.
- Users can either use a Sumo configuration file which is provided inside NetSim's installation directory or use a different user specified SUMO configuration file. This .cfg file contains the path of NETWORK file and VEHICLES file.
- Further help on how to create SUMO configuration files is available at **http://sumo.dlr.de/wiki/Networks/Building_Networks_from_own_XML-descriptions**.

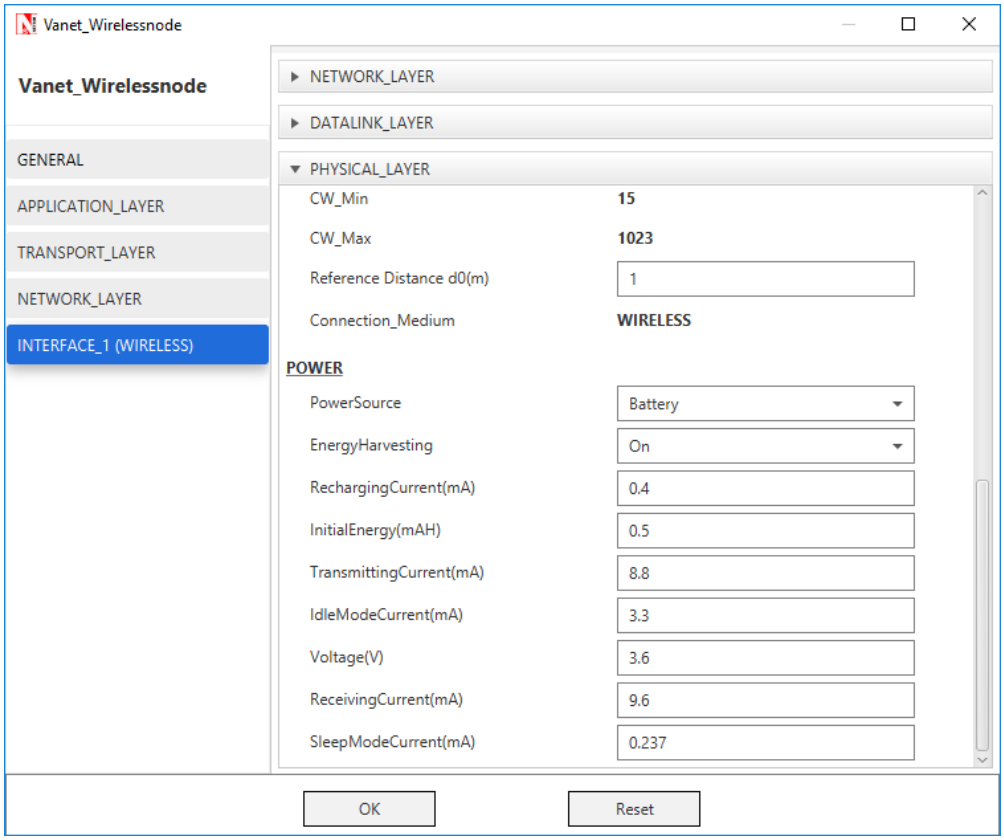
After selecting the Sumo configuration file name, the scenario is opened, with nodes placed at their respective starting positions (tracked from Sumo). Roads and Traffic Lights are also placed exactly as present in SUMO Configuration file.

2.2 Set Node, Link and Application Properties

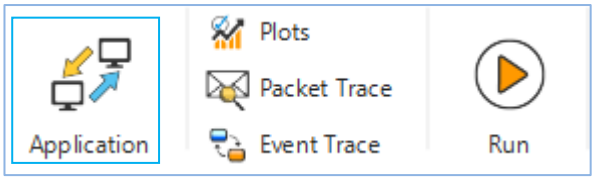
- Right click on the appropriate node or link and select Properties. Then modify the parameters according to the requirements.
- Routing Protocol in Network Layer and all user editable properties in Data Link Layer, Physical Layer and Power are Global.
- In the Global properties, Mobility Model is set to SUMO and it is non-Editable. This signifies that the Node movements will be traced from SUMO.
- File name gives the path to Sumo Configuration file that was given by the user.
- Step Size is taken from the Sumo Configuration file specified which tells the amount of time paused in sumo corresponding to single step of SUMO Simulation.
- In interface_wireless properties, under Physical layer, by default Standard is set to IEEE 802.11p in case of VANET.
- The following are the important properties of VANET Wireless Node (RSU/Vehicle) in Data link and Physical layers.


Vanet_Wirelessnode
Vanet_Wirelessnode
GENERAL
APPLICATION_LAYER
TRANSPORT_LAYER
NETWORK_LAYER
INTERFACE_1 (WIRELESS)
NETWORK_LAYER
DATALINK_LAYER
Protocol
IEEE1609
CCH_TIME(Microsec)
30000
SCH_TIME(Microsec)
100000
GUARD_TIME(Microsec)
100
Secondary_Protocol
IEEE802.11
IEEE_802.11e
Enable
Retry_Limit
4
RTS_Threshold(bytes)
3000
Rate_Adaptation
FALSE
Bss Type
MESH
MAC_Address
90FCC46998D2
PHYSICAL_LAYER
OK
Reset

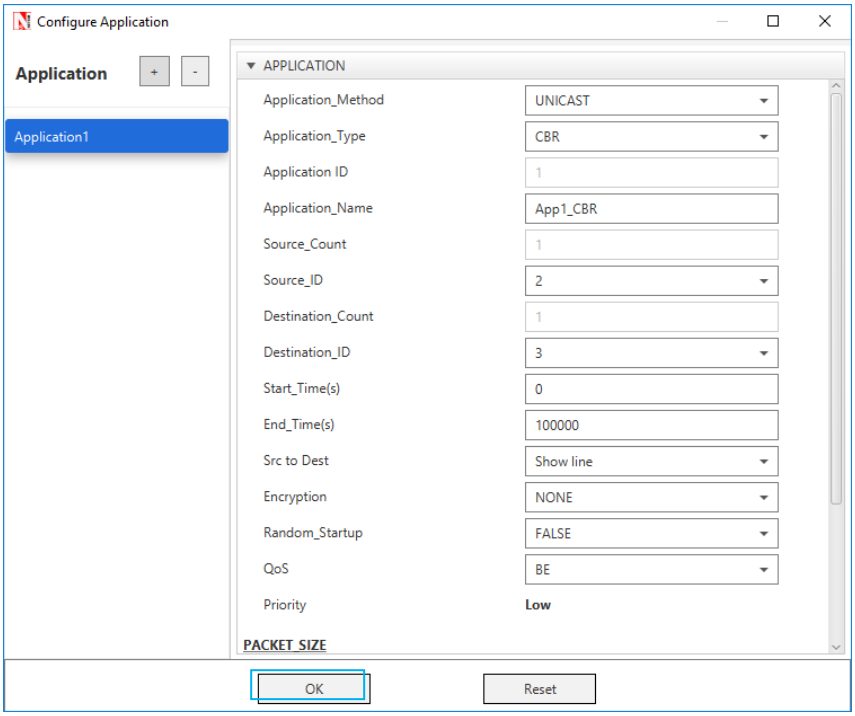

Vanet_Wirelessnode
Vanet_Wirelessnode
GENERAL
APPLICATION_LAYER
TRANSPORT_LAYER
NETWORK_LAYER
INTERFACE_1 (WIRELESS)
NETWORK_LAYER
DATALINK_LAYER
PHYSICAL_LAYER
Protocol
IEEE1609
Secondary_Protocol
IEEE802.11
Standard
IEEE802.11p
Trasmission_Type
OFDM
Trasmitter_Power(mW)
40
Antenna Gain(dB)
1
Antenna Height(m)
1
SIFS(Microsec)
16
Frequency_Band(GHz)
5
Bandwidth(MHz)
10
CCA Mode
ENERGY_DETECTION
SlotTime(Microsec)
9
Standard_Channel_SCH
36_5180
Standard_Channel_CCH
38_5190
OK
Reset



- Click on the Application icon present on the ribbon and set properties. Multiple applications can be generated by using add button in Application properties.



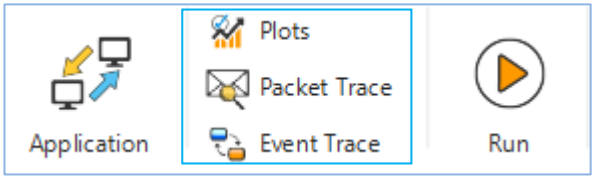
- Set the values according to requirement and click OK.



Detailed information on Application properties is available in section 5 of NetSim User Manual.

2.3 Enable Packet Trace, Event Trace & Plots (Optional)

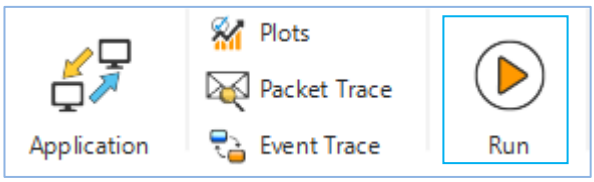
Click Packet Trace / Event Trace icon in the tool bar and click OK. To get detailed help, please refer sections 7.5 and 7.6 in User Manual. Select Plots icon for enabling Plots and click OK.



2.4 Run Simulation

Click on **Run Simulation** icon on the top toolbar. Simulation Time is set from the Configuration File of Sumo. The simulation has three options

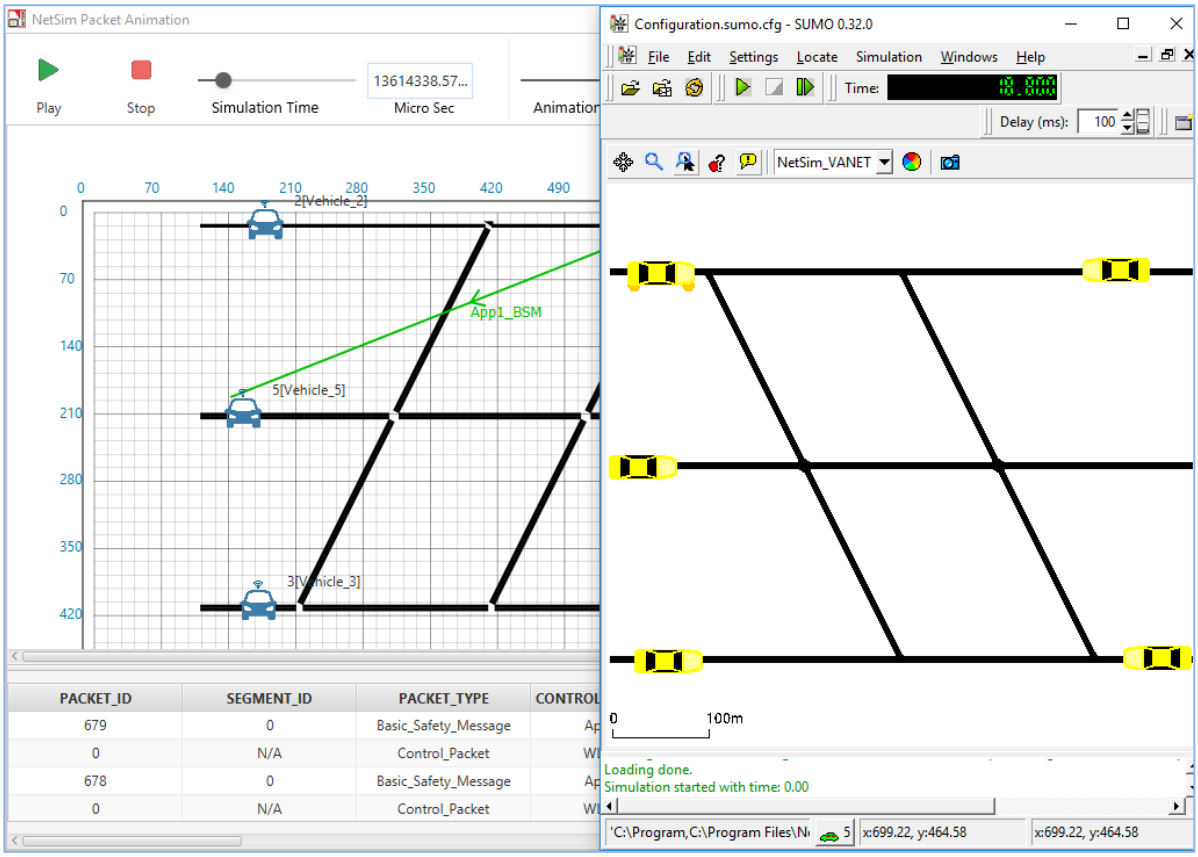
- **Record animation** - which runs Sumo in background. Users can view animation after completion of Simulation.



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

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

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



Users can see the movement of vehicles in SUMO and observe equivalent movement in NetSim. Here users can notice an inversion of view in the GUI, since origin (0, 0) in SUMO is at the left bottom, while origin is at the left top in NetSim.

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

3 Model Features

3.1 IEEE802.11p DSRC/WAVE Protocol Stack

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

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

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

3.2 Implementation of 802.11p protocol in NetSim

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

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

- **Control channel (CCH):** A single radio channel, not a service channel, intended for the exchange of management information, including Wireless Access in Vehicular Environments (WAVE), Service Advertisements, and WAVE Short Messages.
- **Service channel (SCH):** Any channel that is not the control channel, intended for management frames and higher layer information exchanges (Wireless Access in Vehicular Environments [WAVE] Short Message [WSMs].
- **Guard interval:** A time interval at the start of each control channel (CCH) interval and service channel (SCH) interval during which devices that are switching channels do not transmit.

BSM Application:

- DSRC protocol runs with BSM (Basic Safety Message) applications
- BSM is a broadcast packet transmitted at a regular interval, and it can be classified as a beacon style transmission.
- The BSM Application class sends and receives the IEEE 1609 WAVE (Wireless Access in Vehicular Environments) Basic Safety Messages (BSMs). The BSM is a 20-byte packet that is generally broadcast from every vehicle at a nominal rate of 10 Hz. In Netsim this can be configured as a broadcast or as a unicast application.
- This application does not follow the IP stack. It runs WSMP protocol over IEEE 1609. There is no routing, static routes cannot be set and packets are sent directly to the destination.

Working of 802.11p protocol:

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

In VANETs, Vehicles and roadside units (RSUs) are the communicating nodes, providing each other with information, such as safety warnings and traffic information. Both types of nodes are dedicated short-range communications (DSRC) devices. Roadside unit (RSU) The RSU is a wave device usually fixed along the roadside or in dedicated locations such as at junctions or near parking spaces. In NetSim, users can model network traffic between vehicles V2V and between vehicle to infrastructure V2I.

3.3 NetSim – SUMO interfacing for VANETs (only in Standard/Pro versions)

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

3.4 How to create your own network using SUMO and run through NetSim

A SUMO network can be created either **manually** or using **SUMO NetEdit.**

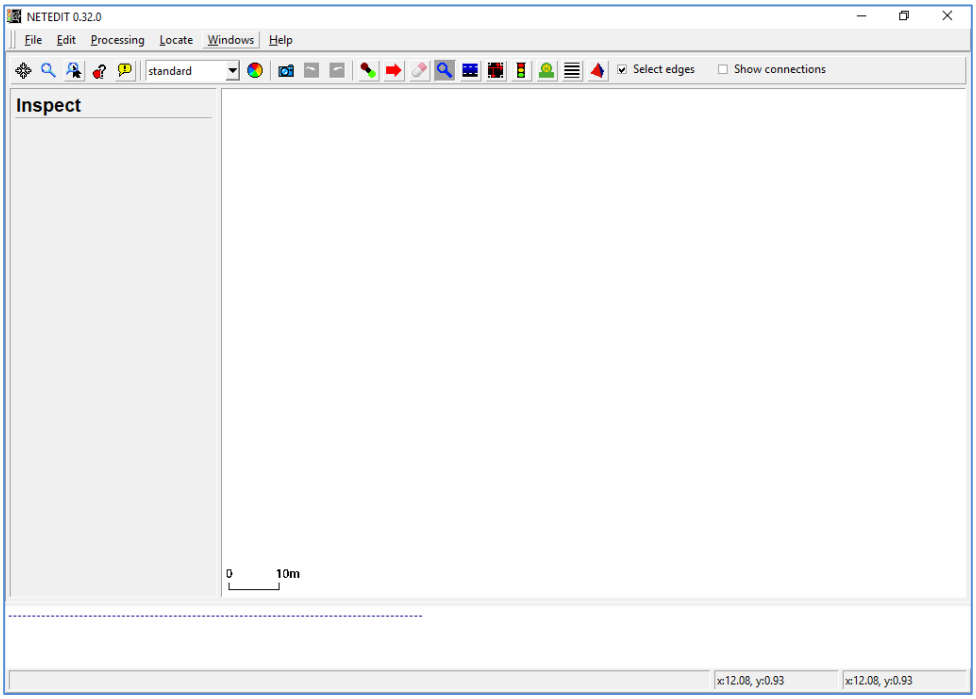
3.4.1 Using SUMO NetEdit utility and randomtrips.py to configure road traffic models:

Netedit is a Road network editor for the road traffic simulation in SUMO. Using this utility, users can quickly design road networks and obtain Network xml file which is part of SUMO configuration.

Steps to create a simple SUMO network using netedit utility:

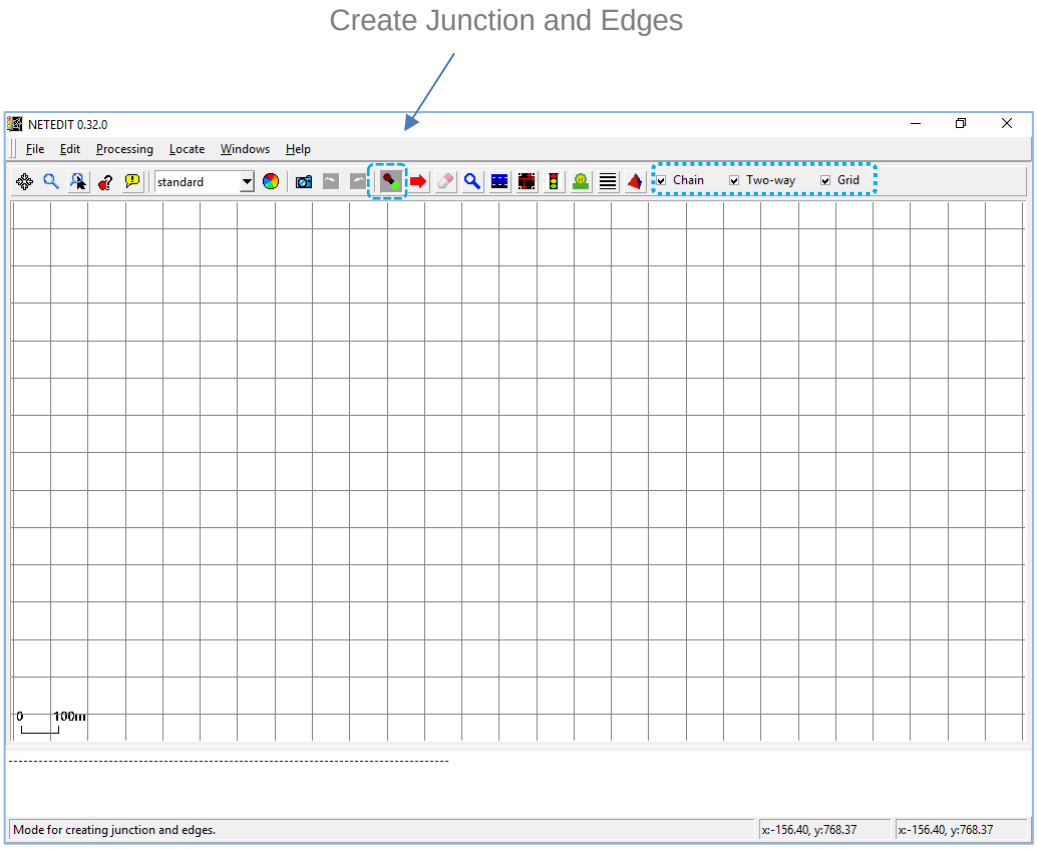
Step 1: Open **netedit** from **<SUMO_INSTALL_DIRECTORY>/bin** (C:\sumo-0.32.0\bin) and select **File-->New Network**

Refer SUMO Documenation: ***"http://sumo.dlr.de/wiki/NETEDIT" for more details on modes of operation***



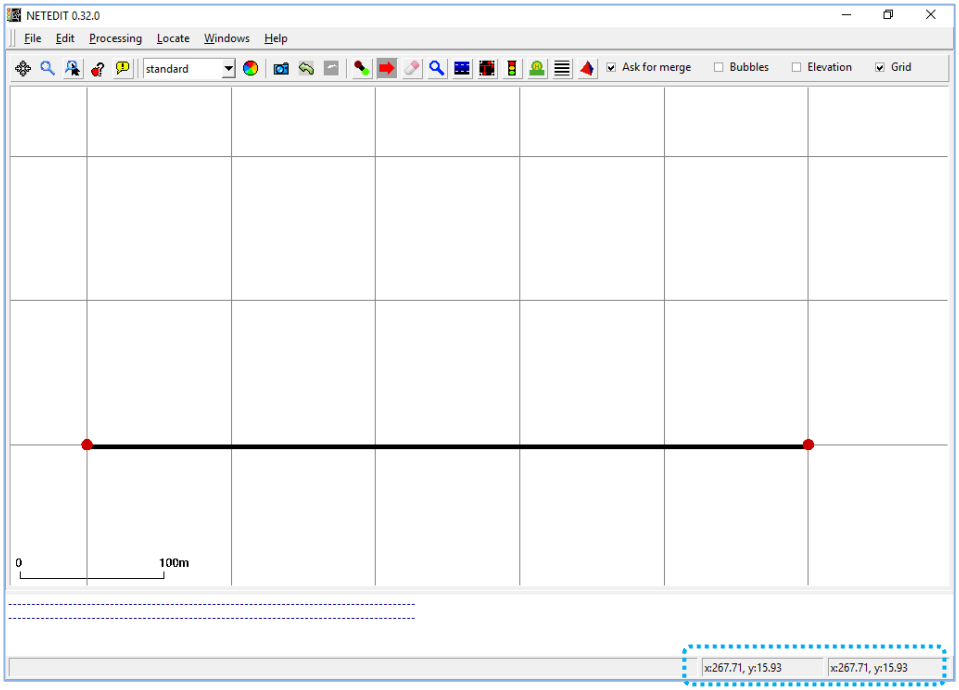
Step 2: Select Creating junction and edges option as shown below or click on character "**e**" in the keyboard.

Step 3: Enable the check boxes "**chain**", **"two-way" and "Grid"** which are present in the right side corner

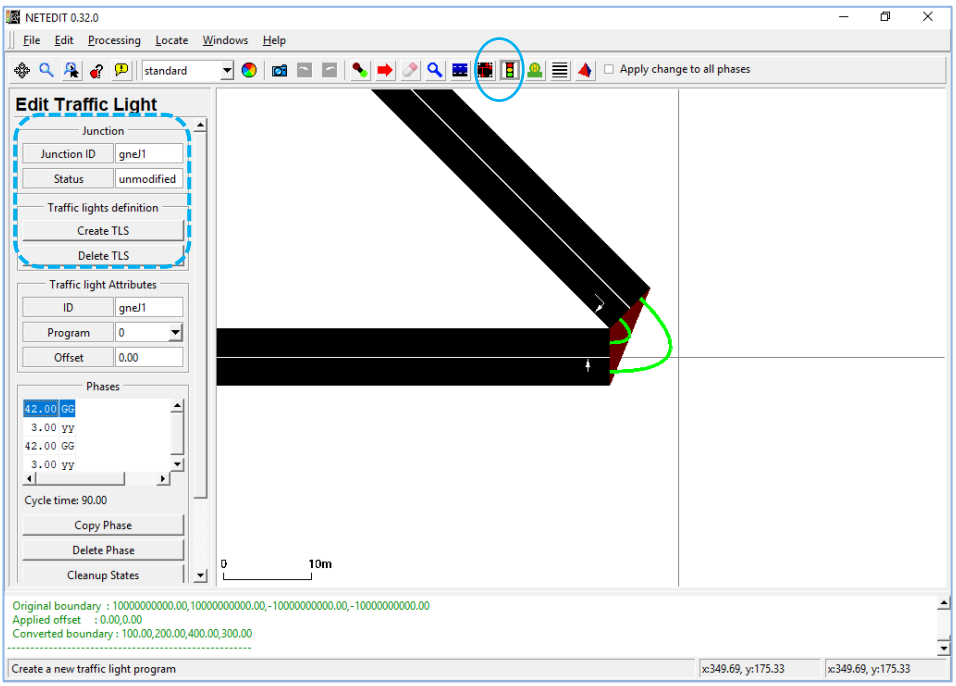


Step 4: Adjust the design area to ensure that the road network lies in the **Positive XY** quadrant. This will help in avoiding complexities when opening the network scenario in NetSim.

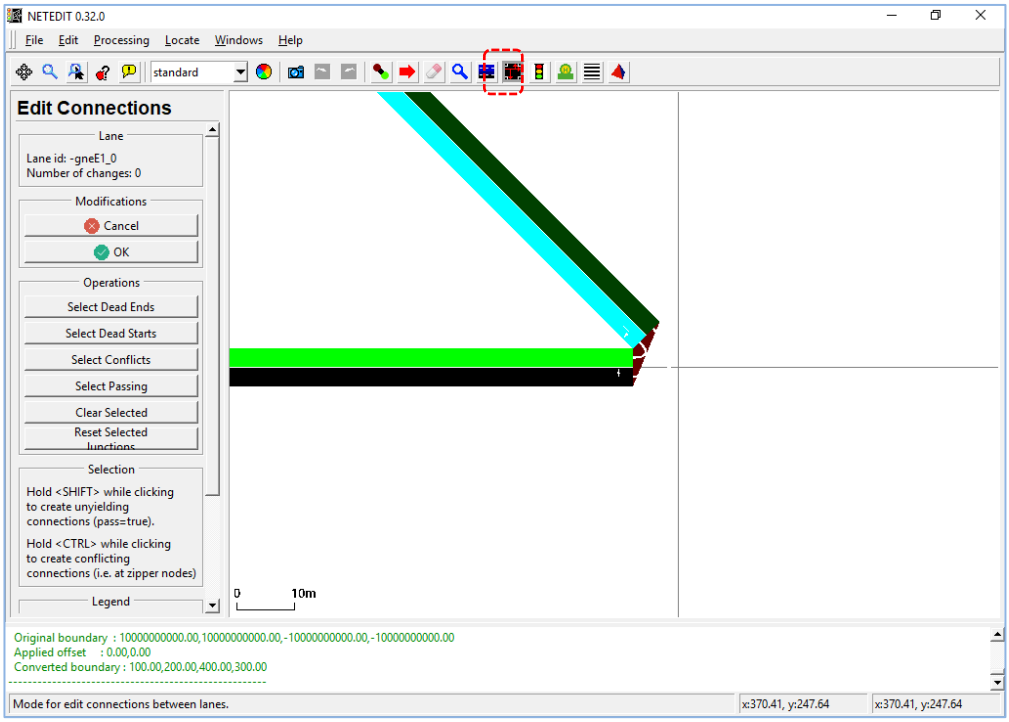
Step 5: Click on grid area to create edges, clicking again will create a new edge which will automatically get connected to the previous edge as shown below



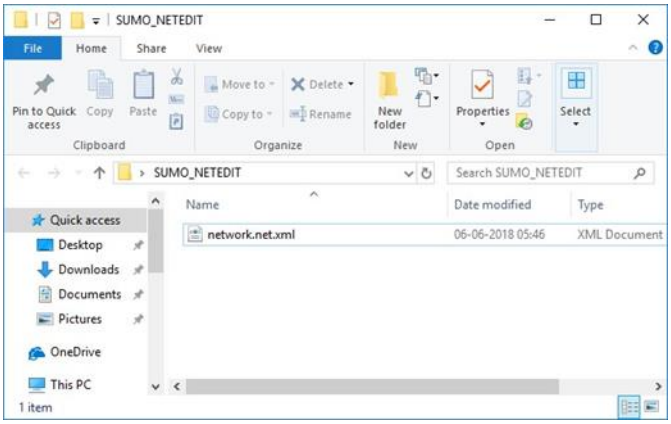
Step 6: Select "**(t) Traffic Lights**". Select the junctions and click on **Create TLS** button on the left to add Traffic Signal to it.



Step 7: Select "**(c) Connect**" icon Select the lanes and ensure connectivity between them.



Step 8: Create a new folder and save the network file (***.net.xml**) over there, say with a name **network.net.xml**

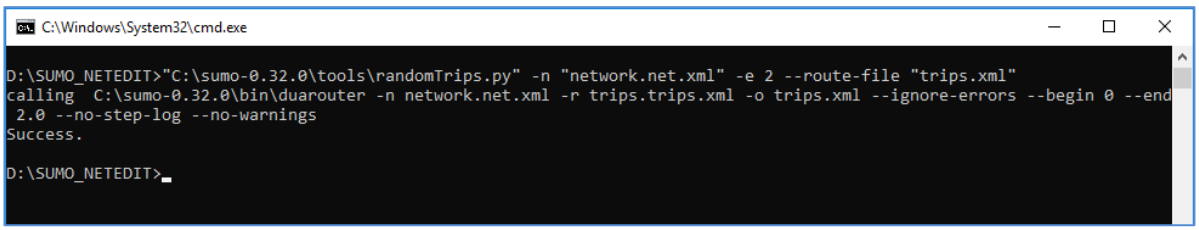


Step 9: Open command prompt with the current working directory as the folder where you have saved the network file in the previous step.

Step 10: Using **randomtrips.py** utility present in **<SUMO_INSTALL_DIRECTORY>/tools** directory create trips file with the command

COMMAND SYNTAX >"C:\sumo-0.32.0\tools\randomTrips.py" -n " ***.net.xml**" -e <**NO_OF_TRIPS**> --route-file "**trips.xml**"

Example Command >"C:\sumo-0.32.0\tools\randomTrips.py" -n "network.net.xml" -e 2 --route-file "trips.xml"



This will create a trips file in your folder along with associated files.

Step 11: Create a SUMO configuration file (***sumo.cfg**) which points to the network and trips file, in your folder which contains the network and route file.

Refer: http://sumo.dlr.de/wiki/Tutorials/Hello_Sumo

Include parameter (To Run in NetSim)
"<step-length value="0.4"/>"

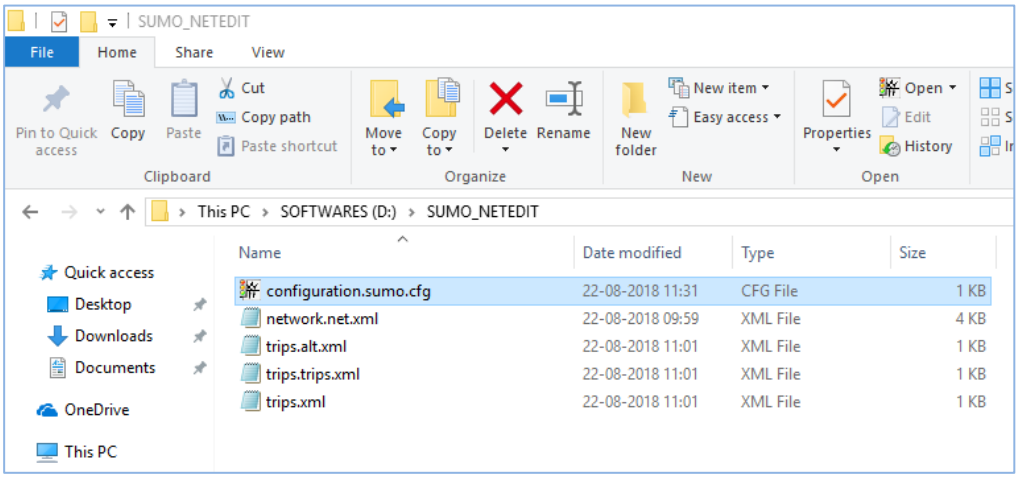
Following is a sample SUMO Configuration:

```
<configuration>
   <input>
      <net-file value="network.net.xml"/>
      <route-files value="trips.trips.xml"/>
```

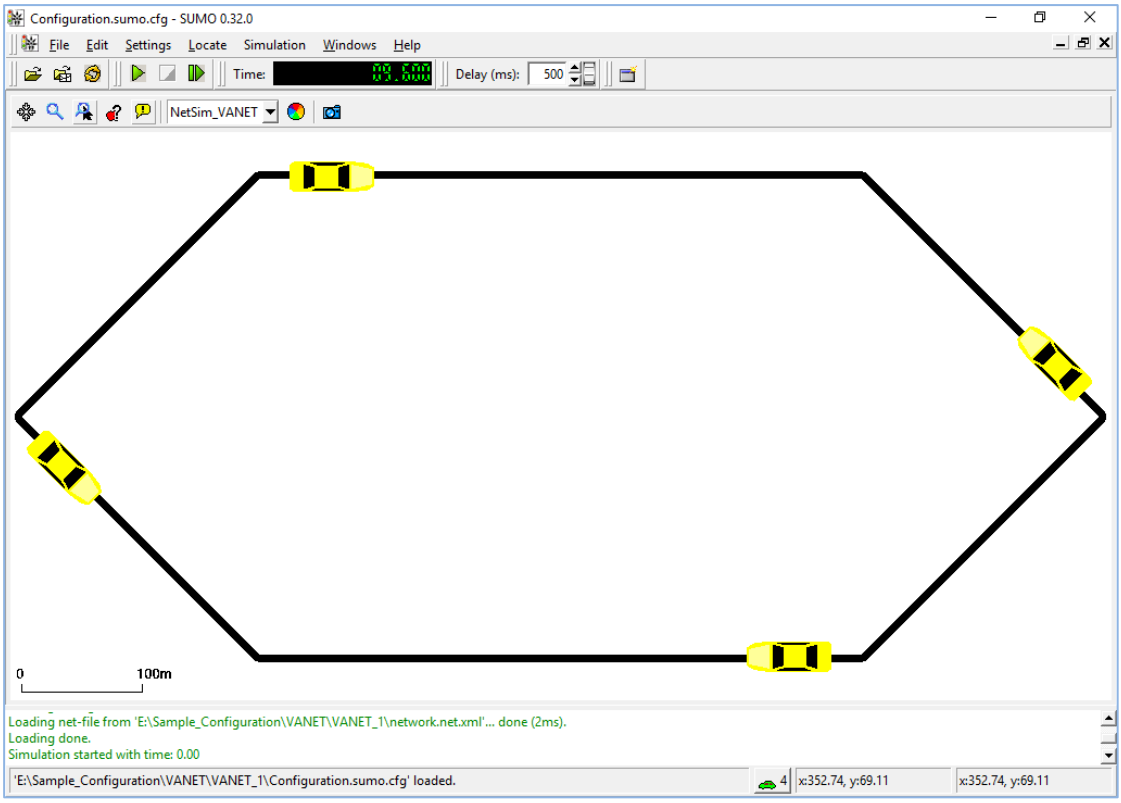
```
    </input>
    <time>
        <begin value="0"/>
        <end value="100"/>
        <step-length value="0.4"/>
    </time>
</configuration>
```

Note: *Save above content as* ***Configuration.sumo.cfg***

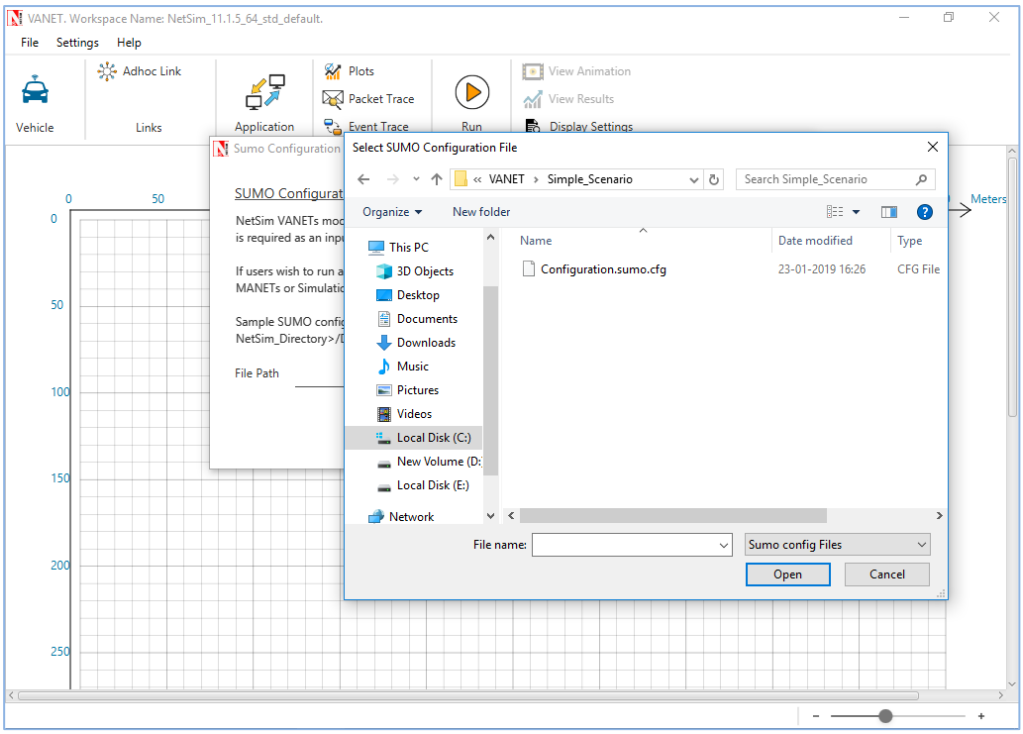
You can copy the above contents to create a SUMO configuration file in your folder.

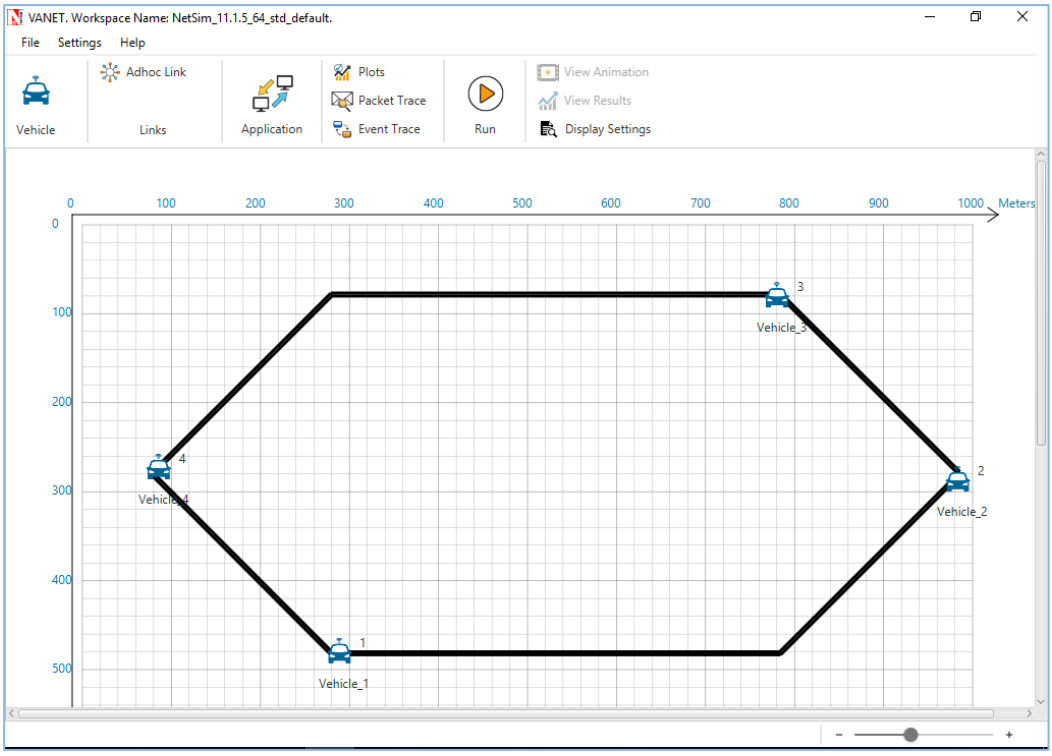


Step 12: Open Configuration.sumo.cfg by double clicking or open SUMO using **sumo-gui.exe** present in **<SUMO_INSTALL_DIRECTORY>/bin**. Open scenario in SUMO using **Open->Simulation** and verify whether the network loads and simulation happens as per the configuration done.



Step 13: Open the SUMO scenario via NetSim VANET by selecting VANET under the New option in the NetSim Home Screen. Browse and locate the SUMO Configuration file present in your directory to load the road traffic network in NetSim GUI. The road network created in SUMO will be automatically replicated in NetSim GUI environment.





Step 14: Configure traffic between vehicles using the Application icon, enable trace files And plots

Step 15: Click on Run Simulation button. It is preferable to specify the **simulation time** less than or equal to the end time specified in sumo configuration (**sumo.cfg**) file.

3.4.2 Creating your own network in SUMO manually:

Step 1: Create a node file using any code editor (like notepad, notepad++ etc) and the file extension will be .nod.xml. It represents the junctions in the road. Each of these attributes has a certain meaning and value range: node_id means unique name of each junction, x-y is the positions of node and type can be "priority", "traffic_light", "rail_crossing", “rail_ signal”etc.(Refer: https://sumo.dlr.de/wiki/Networks/PlainXML#Node_Descriptions).

NODES.nod.xml

```
<nodes>
<node id="n0" x="100" y="300" type="priority"/>
<node id="n1" x="300" y="300" type="traffic_light"/>
<node id="n2" x="500" y="300" type="rail_crossing"/>
<node id="n3" x="700" y="300" type="priority"/>
<node id="n4" x="300" y="100" type="traffic_light"/>
<node id="n5" x="500" y="100" type="traffic_light"/>
</nodes>
```

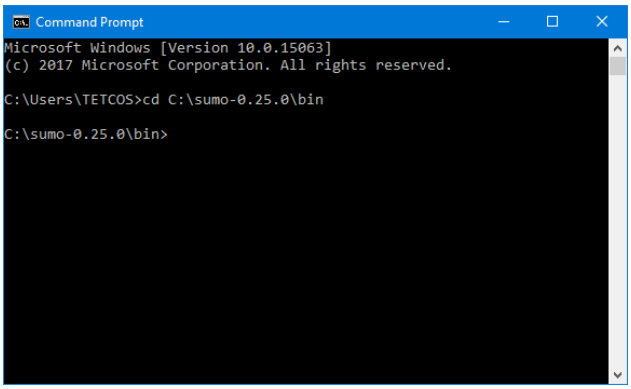
Step 2: Create an edge file that describes how the junctions or nodes are connected to each other. The extension of this file is .edg.xml. Each edge is unidirectional and starts at the "from"-node and ends at the "to"-node. For each edge, some further attributes should be

supplied, being the number of lanes the edge has (numLanes), the maximum speed allowed on the edge speed. Furthermore, the priority may be defined optionally. (Refer: https://sumo.dlr.de/wiki/Networks/PlainXML#Edge_Descriptions).

EDGE.edg.xml

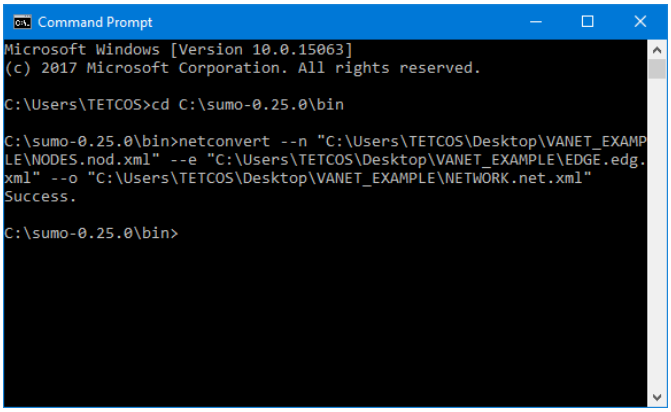
```
<edges>
<edge id="L01" from="n1" to="n0" priority="75" nolanes="2" speed="40"/>
<edge id="R01" from="n0" to="n1" priority="75" nolanes="2" speed="40"/>
<edge id="L12" from="n2" to="n1" priority="75" nolanes="2" speed="30"/>
<edge id="R12" from="n1" to="n2" priority="75" nolanes="2" speed="30"/>
<edge id="L23" from="n3" to="n2" priority="75" nolanes="2" speed="30"/>
<edge id="R23" from="n2" to="n3" priority="75" nolanes="2" speed="30"/>
<edge id="D04" from="n0" to="n4" priority="75" nolanes="2" speed="60"/>
<edge id="U04" from="n4" to="n0" priority="75" nolanes="2" speed="60"/>
<edge id="D14" from="n1" to="n4" priority="75" nolanes="2" speed="40"/>
<edge id="U14" from="n4" to="n1" priority="75" nolanes="2" speed="40"/>
<edge id="D35" from="n3" to="n5" priority="75" nolanes="2" speed="60"/>
<edge id="U35" from="n5" to="n3" priority="75" nolanes="2" speed="60"/>
<edge id="D25" from="n2" to="n5" priority="75" nolanes="2" speed="40"/>
<edge id="U25" from="n5" to="n2" priority="75" nolanes="2" speed="40"/>
<edge id="L45" from="n5" to="n4" priority="75" nolanes="2" speed="40"/>
<edge id="R45" from="n4" to="n5" priority="75" nolanes="2" speed="40"/>
</edges>
```

Step 3: Open Command Prompt and change the directory to the binary folder of sumo using cd command. **“cd C:\ sumo-0.32.0\bin”**

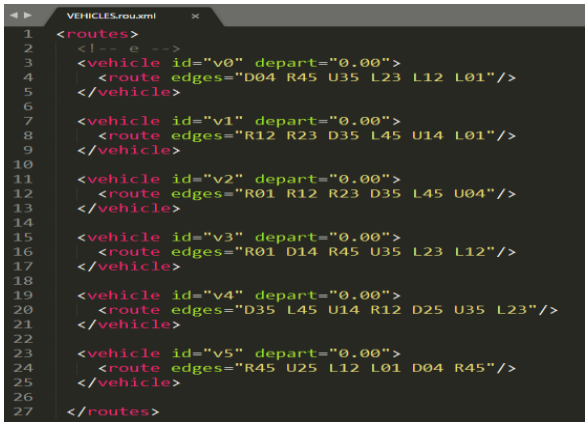


Step 4: Generate Network file by using NETCONVERT command. Make a folder named like VANET_Example and place the .nod.xml and .edg.xml files i.e. NODES.nod.xml and EDGE.edg.xml respectively.

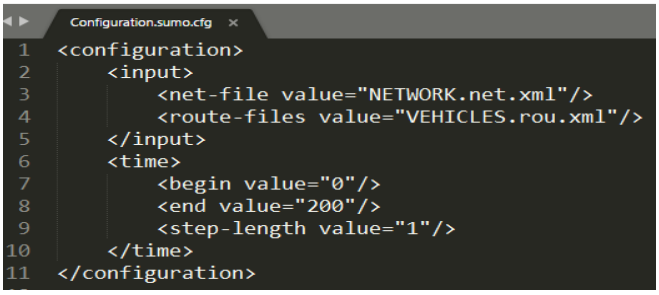
netconvert --n “<path where the .nod.xml file is present>\<filename>.nod.xml” --e “<path where the .edg.xml file is present>\<filename>.edg.xml” --o “<path where both input files are present>\<filename>.net.xml”



Step 5: Create a .rou.xml file that describes the direction of the vehicle's movement.



Step 6: Create a sumo configuration file file using any code editor (like notepad, notepad++ etc) and the extension is .sumo.cfg. Place the file inside the same folder where the network file (i.e. NETWORK.net.xml) and route file (i.e. VEHICLES.rou.xml) are present.



Step 7: Now open "New Simulation → VANET". Choose the Configuration.cfg.xml from the specified folder and run simulation using NetSim

4 Featured Examples

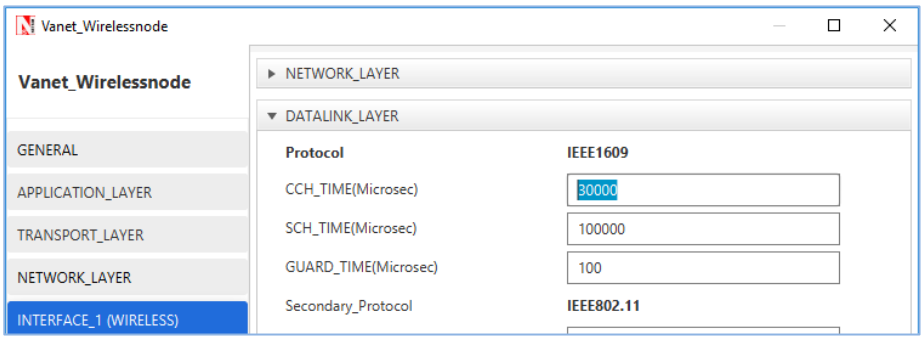
Sample configuration files for all networks are available in Examples Menu in NetSim Home Screen. These files provide examples on how NetSim can be used – the parameters that can be changed and the typical effect it has on performance.

4.1 CCH Time Interval

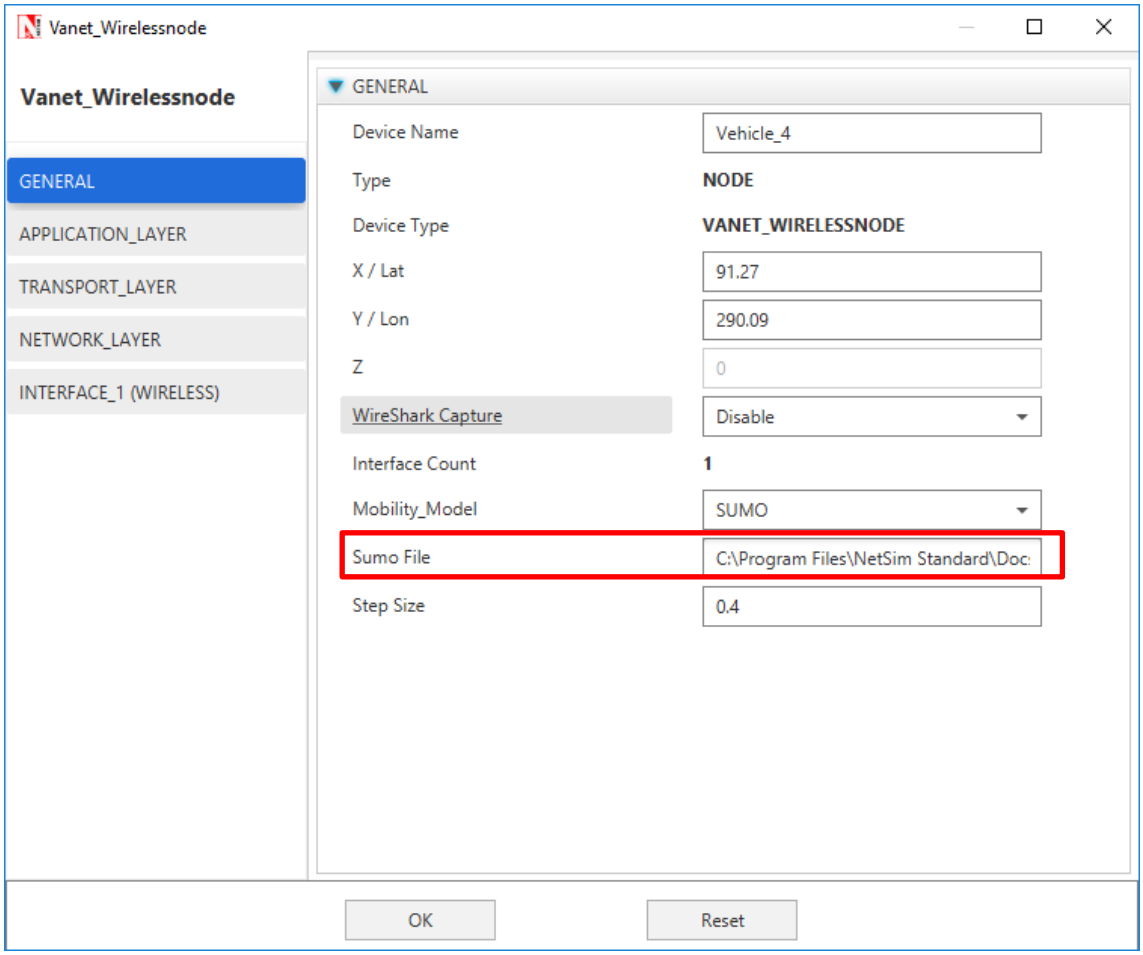
Open NetSim, Select Examples->VANETs->CCH-Time-Analysis

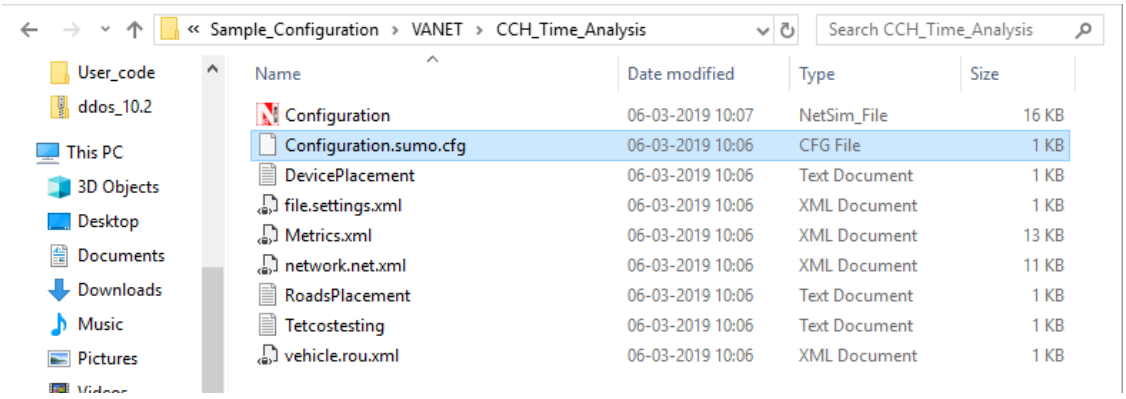
Settings done for this sample experiment:

- In INTERFACE_1(WIRELESS) → Data Link Layer, CCH_TIME_INTERVEL – 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.

5 Reference Documents

IEEE 802.11p, IEEE1609: Standards for Wireless Access in Vehicular Environment (WAVE)

6 Latest FAQs

Up to date FAQs on NetSim's VANETs library is available at https://tetcos.freshdesk.com/support/solutions/folders/14000118424