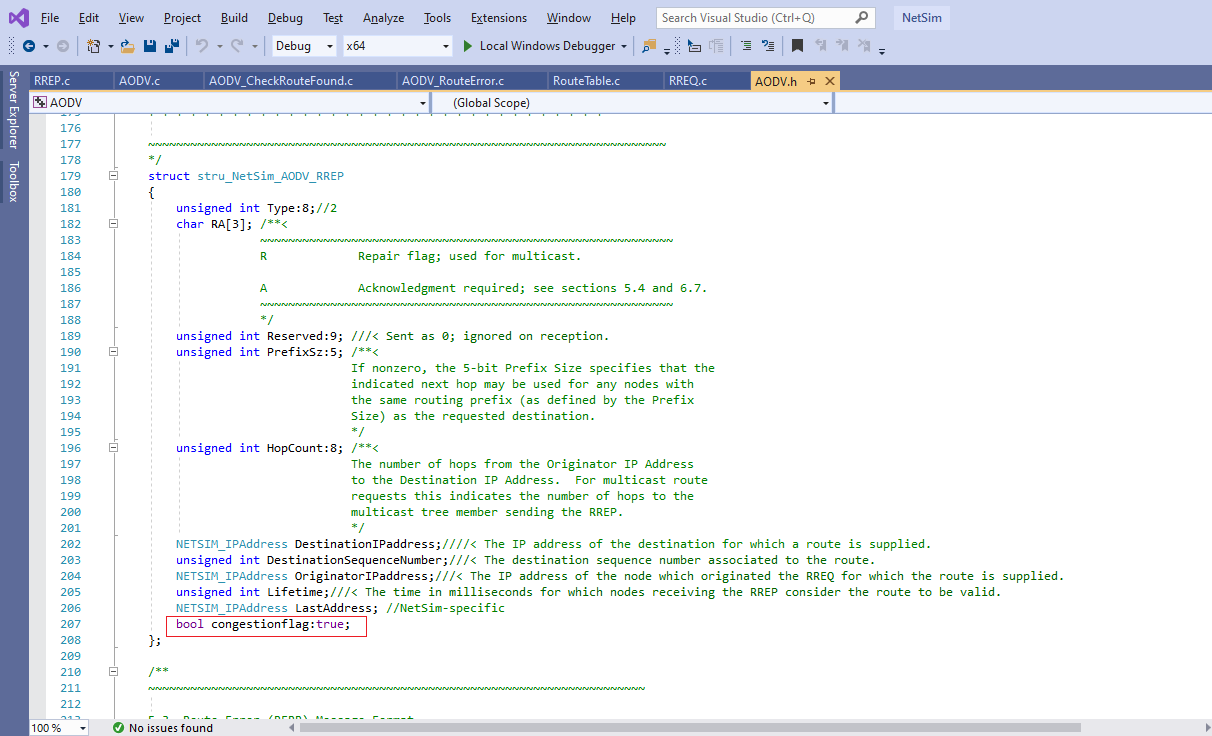
Performance analysis of CC\_AODV

using NetSim™ (v 12.1)

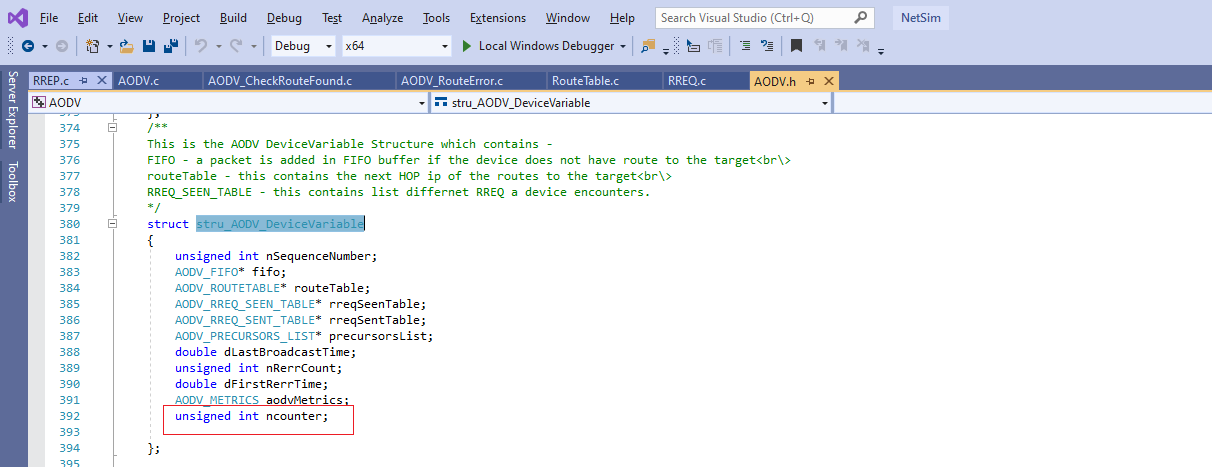
The process of CC-AODV explains how to establish the route. First, the source node performs a flooding broadcast RREQ package in the entire network. When RREQ package arrives to the intermediate node, the router checks the congestion counter whether it is less than a certain predetermined value. If the comparison yields less than the counter, the routing table updates and forwarding to next router; otherwise, the router drops the RREQ package. Once the RREQ arrives to the corresponding destination, the RREP is generated by the router. In CC-AODV, the congestion flag is added to the RREP header. There are two cases of which a RREP is generated corresponding to a RREQ. One is from the source node to establish the route and the other is from the neighbour nodes to maintain the route. When the destination node receives the RREQ from the source node, it generates the RREP with the congestion flag set to true. While the RREP unicast back to the corresponding source node, passing by the intermediate node, the router checks the congestion flag. If it is true, the counter increases; otherwise, the counter keeps the same. Then, the router updates the routing information.

In order to implement CC-AODV following code modification done in AODV Protocol

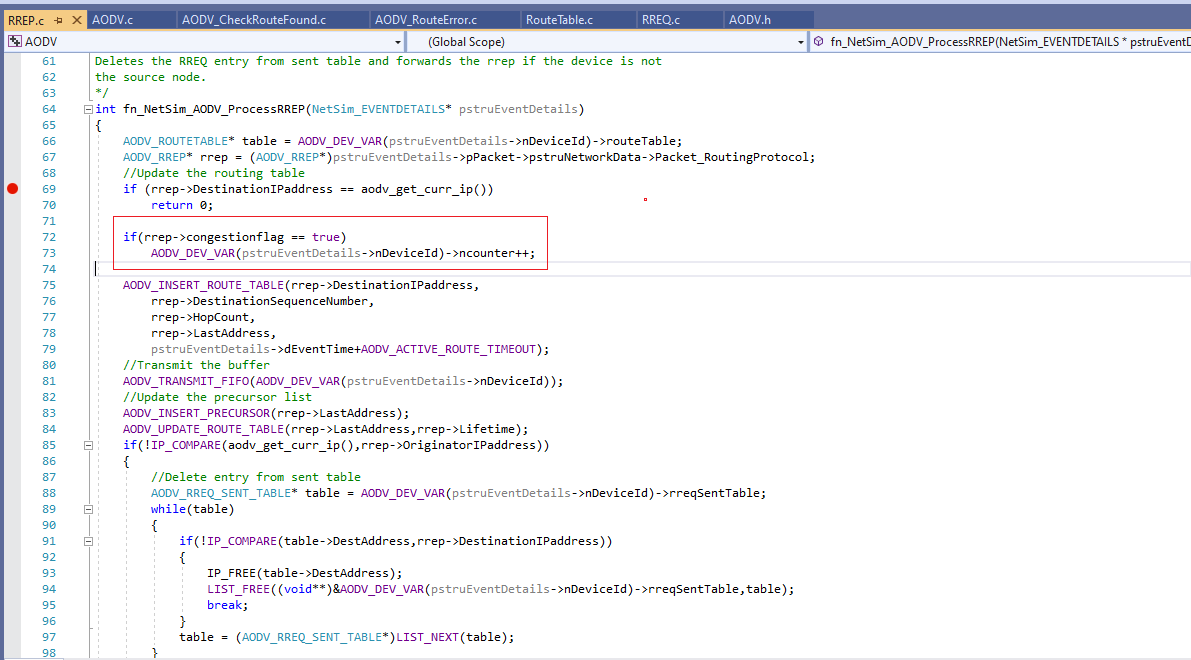
1. The RREP structure stru\_NetSim\_AODV\_RREP is defined in AODV.h has been modified to include a Congestion flag for implementing CC-AODV

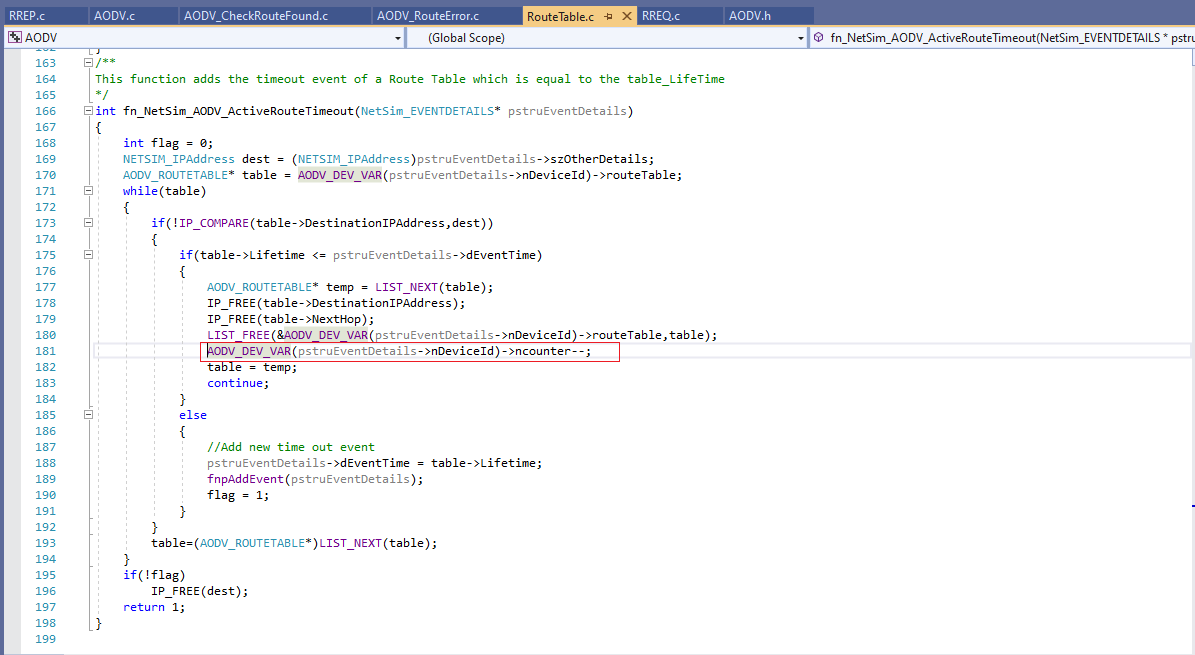


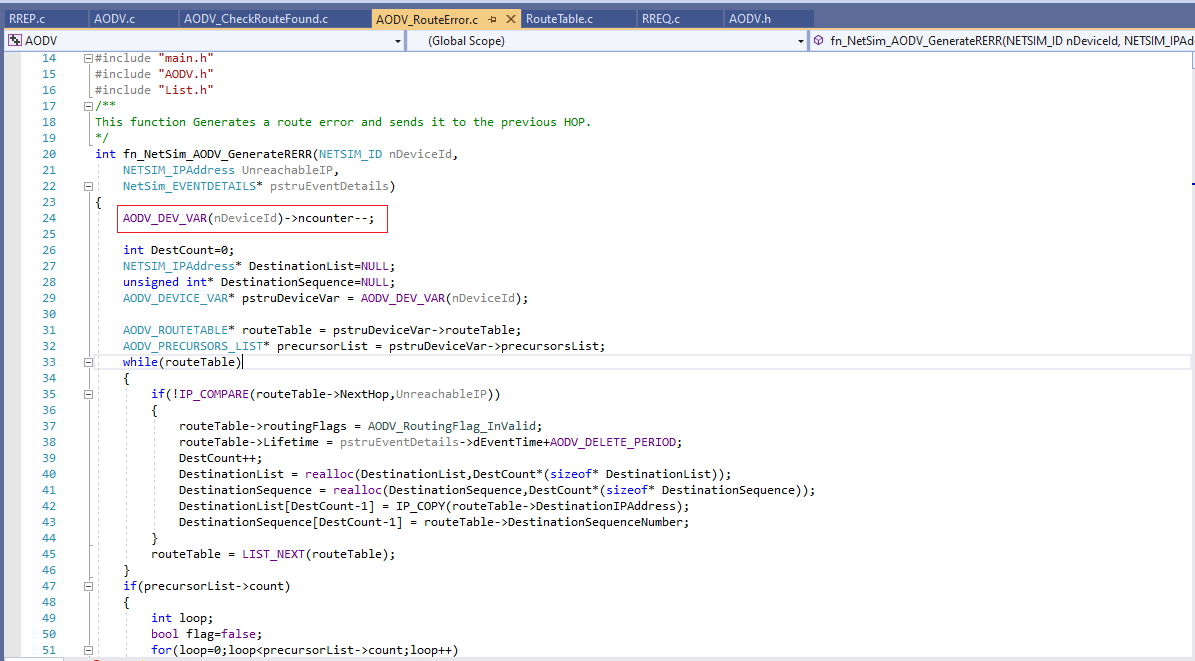
1. The DeviceVariable Structure stru\_AODV\_DeviceVariable is defined in AODV.h file has been modified to include a congestion counter for implementing CC-AODV



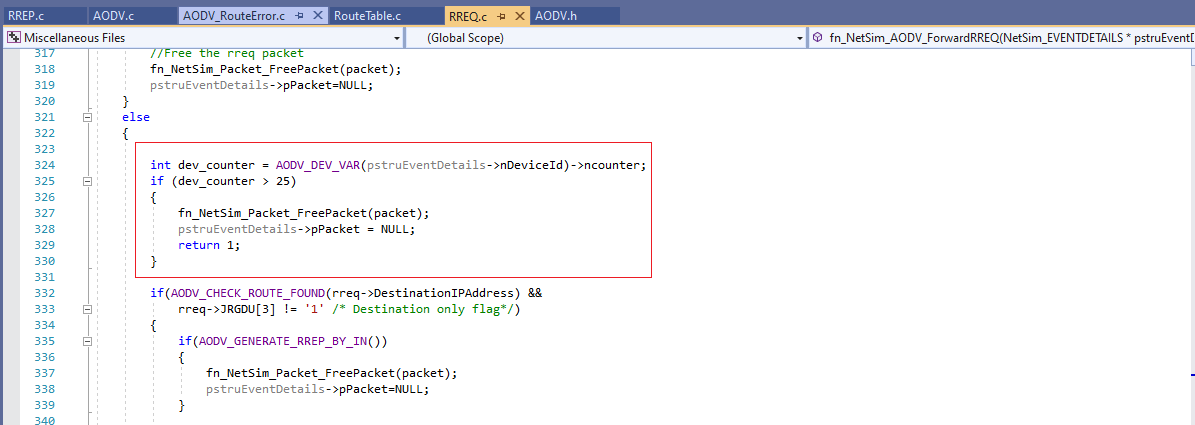
1. The source codes of functions in **RREP.c**, **RouteTable.c** and **AODV\_RouteError.c** has been modified suitably to Increment, Decrement the congestion counter accordingly





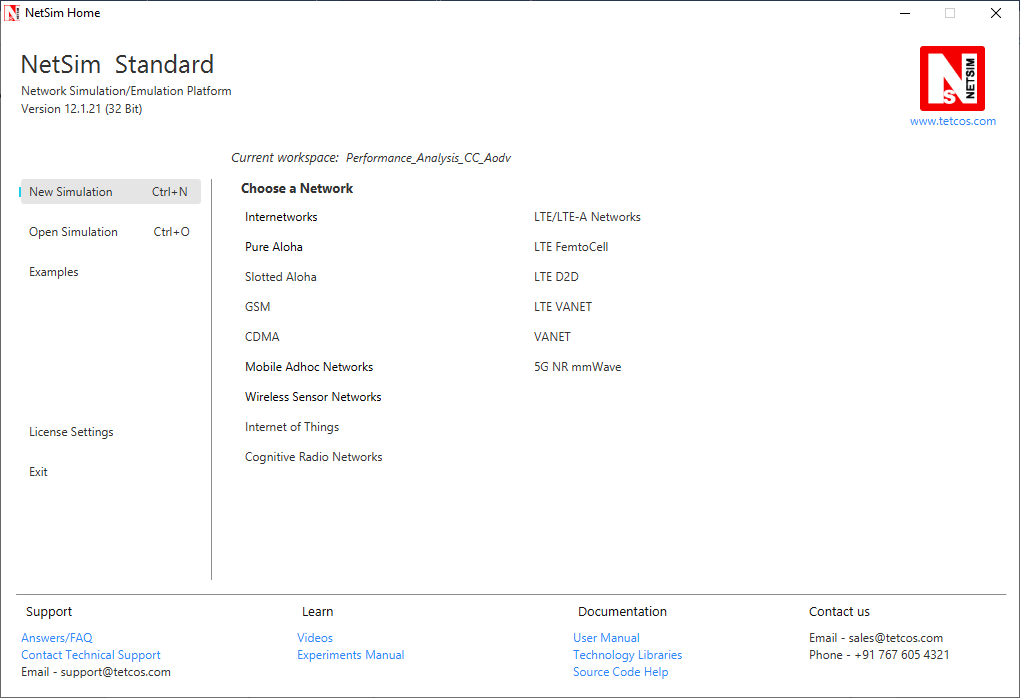


1. The source codes and functions related to Route request are defined in the file **RREQ.c.** The **fn\_NetSim\_AODV\_ProcessRREQ()** function that is part of this file has been modified suitably to check the value of the congestion counter in the received RREQ packet and accordingly forward or drop the packet

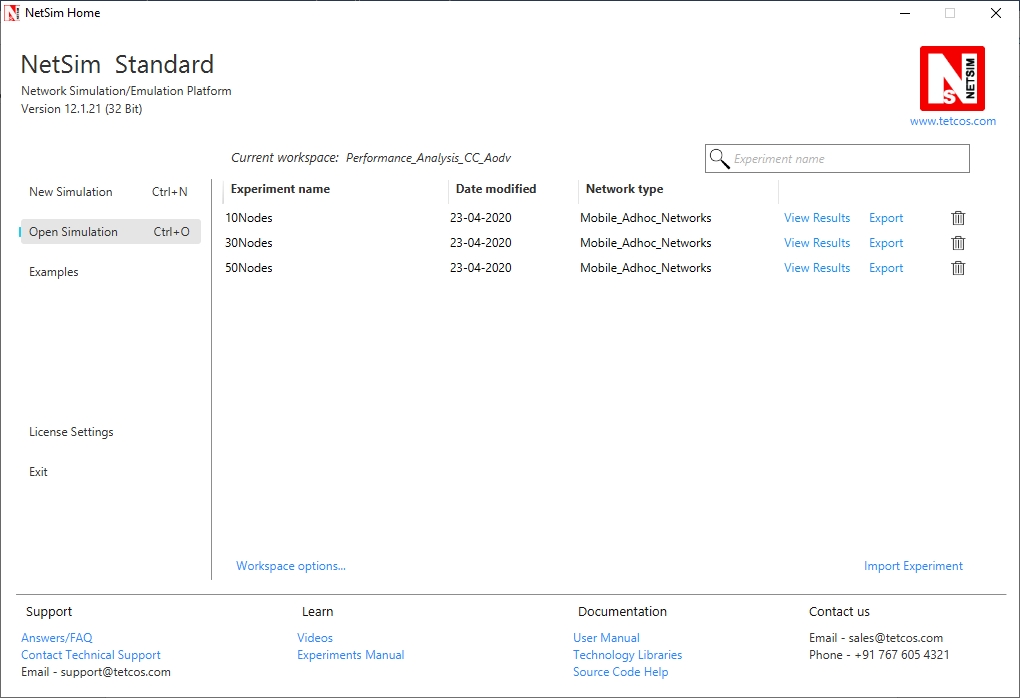


# Steps:

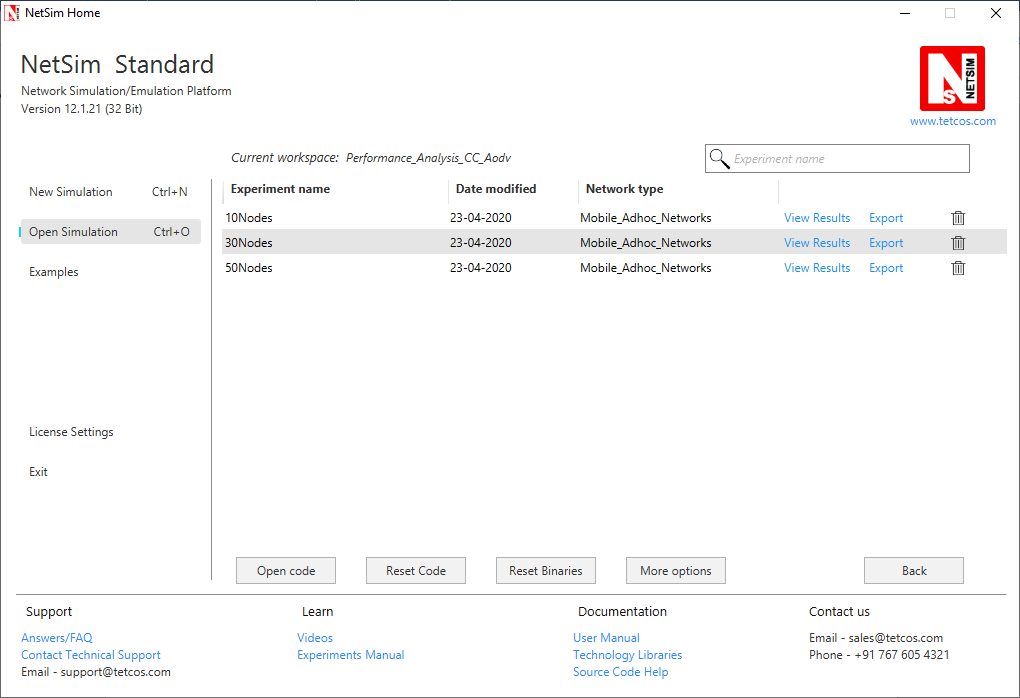
1. After you unzip the downloaded project folder, Open NetSim Home Page click on **Open Simulation**



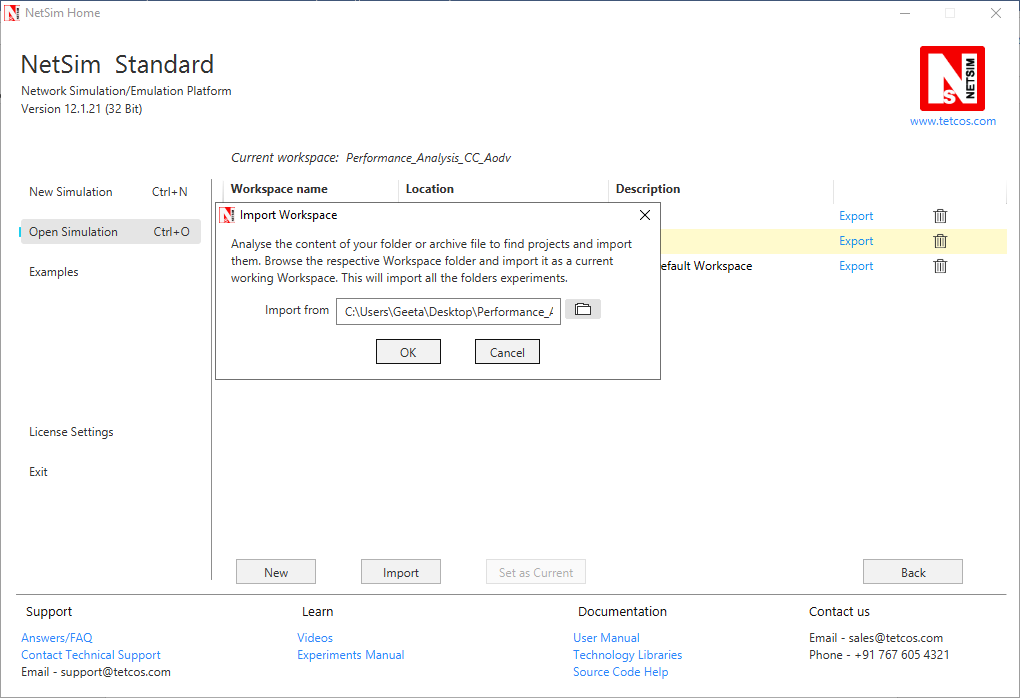
1. Click on **Workspace options**



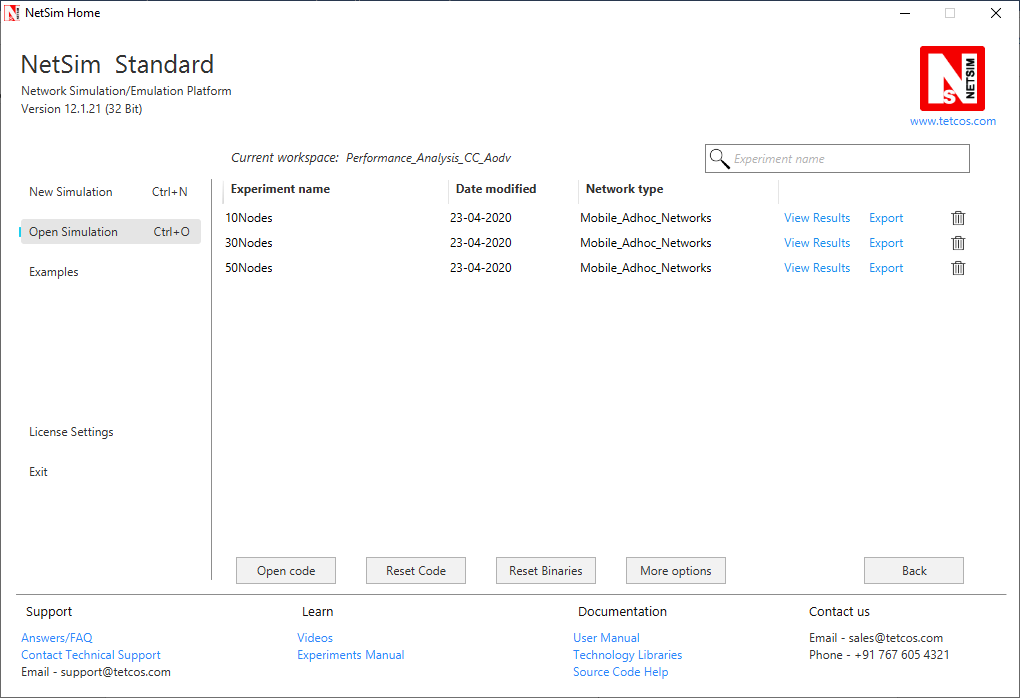
1. Click on **More Options,**



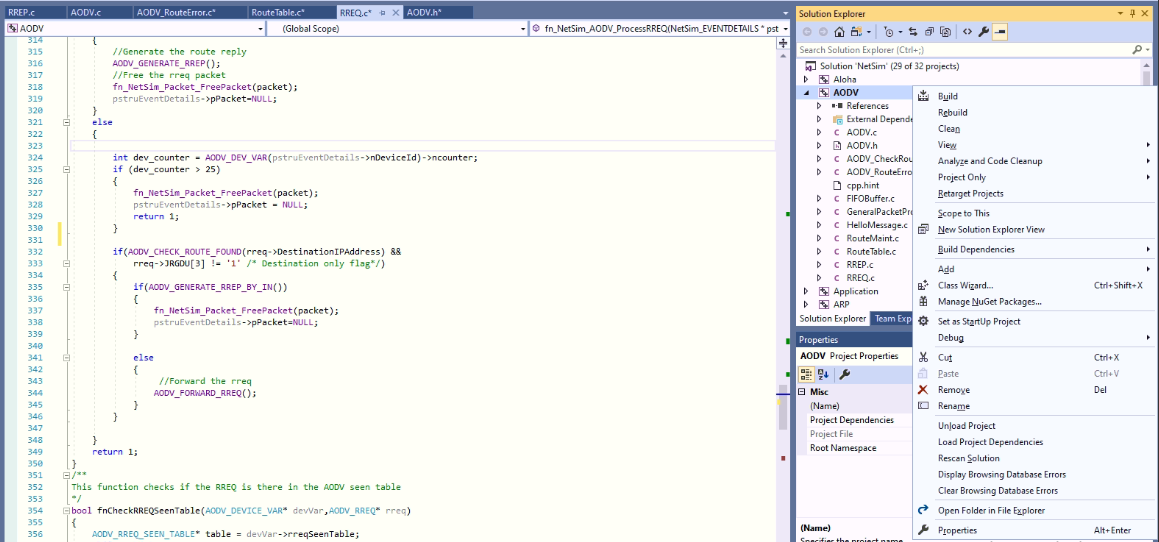
1. Click on **Import**, browse the extracted folder and go into the WorkSpace\_Performance\_Analysis\_CC\_AODV directory, click on select folder and then click on OK.



1. Go to home page, Click on **Open Simulation** 🡪 **Workspace options** 🡪 **Open code**



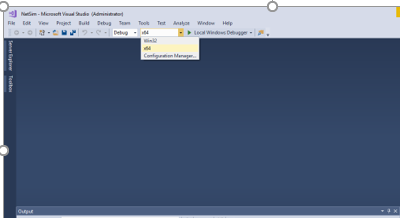
1. Right click on the AODV Project and select rebuild.



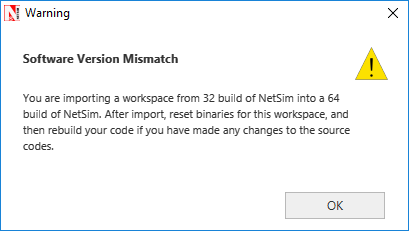
1. Upon rebuilding, **libAodv.dll** will automatically get updated in the respective bin folder of the current workspace.

**Note:**

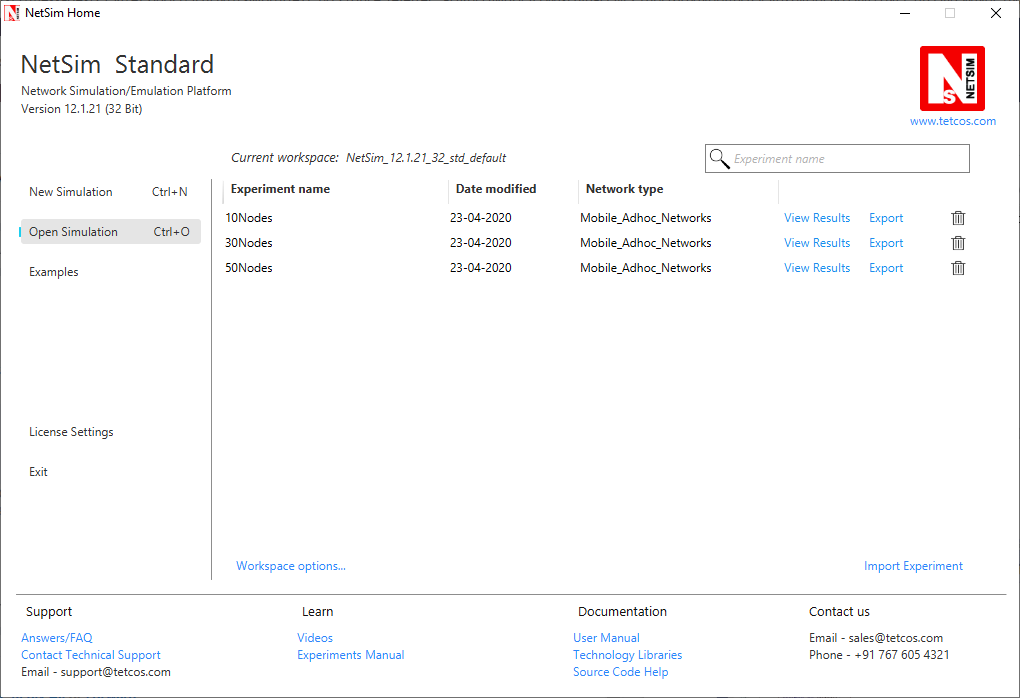
* Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



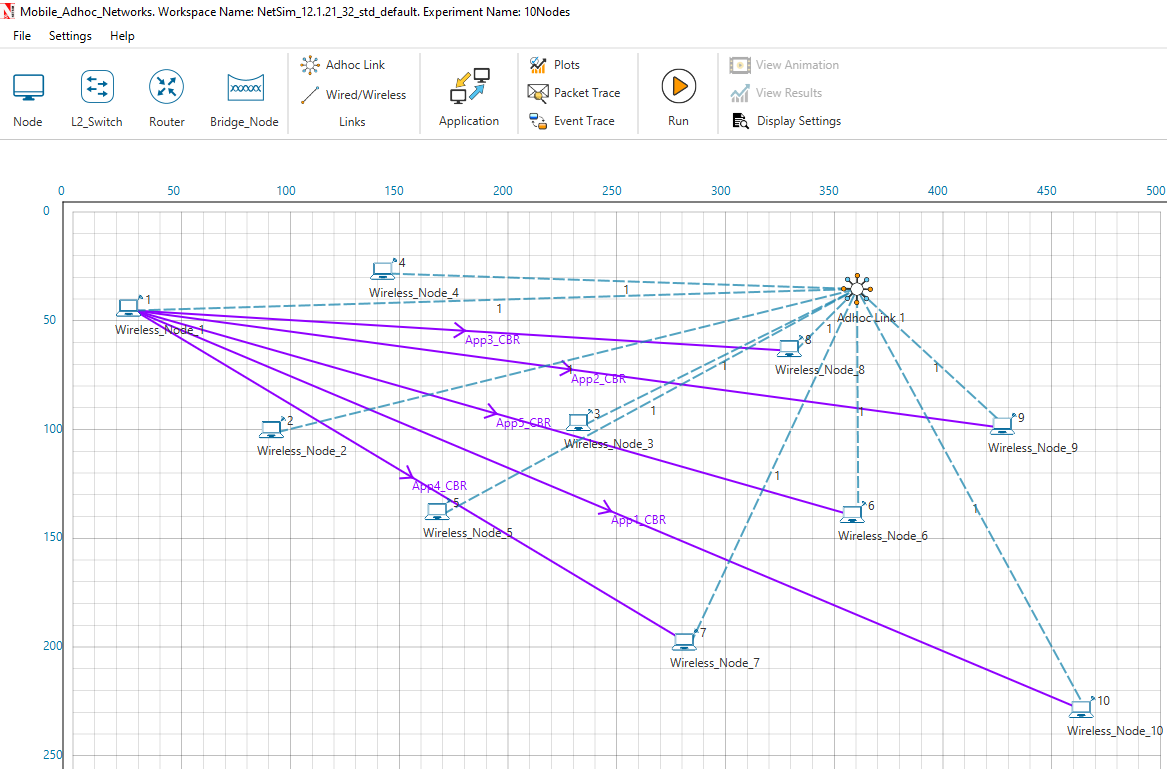
* While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



1. Go to NetSim home page, click on **Open Simulation**, Click on **10Nodes\_Example**.



1. Now create a Network Scenario in NetSim Mobile Adhoc Network as per the screenshot below.



A sample Configuration.netsim file is also provided in the Config\_File folder along with this project which can be opened in NetSim directly.

1. Run the simulation for 30 sec

Simulations have been carried out using a different number of nodes in a network to symbolize different practical applications of wireless network. For example, 10 nodes symbolize a small network that can be used in an agricultural setup. 30 nodes symbolize a medium size network that can be used in an industrial setup and a large 50 nodes network that can be used in an army base.

**Result:**

Performance of CC-AODV has been compared with other reactive protocol AODV based on different performance metrics such as Throughput, End to End delay etc.

|  |  |  |
| --- | --- | --- |
|  | AODV | CC\_AODV |
| 10Nodes | 0.068813 | 0.937328 |
| 30Nodes | 0.073237 | 0.861634 |
| 50Nodes | 0.138772 | 0.847382 |

Table 1 : Throughput comparison between Aodv and CC\_Aodv

As per the Table 1 the proposed CC-AODV has higher throughput than the AODV. In CC-AODV, the internal nodes can be utilized much efficiently than AODV because the counter helps to reroute the path if the internal node is busy. This can increase the network channel utilization.

|  |  |  |
| --- | --- | --- |
|  | AODV | CC\_AODV |
| 10Nodes | 1101969.555 | 1399783.998 |
| 30Nodes | 182358.7791 | 1988943.975 |
| 50Nodes | 732298.3575 | 2305555.48 |

Table 2: End to End delay comparison between Aodv and CC\_Aodv

Table 2 demonstrate that CC-AODV has slightly higher End-to-End performance than the AODV, the result is achieved by rerouting the path of the data if the router is on a busy state