

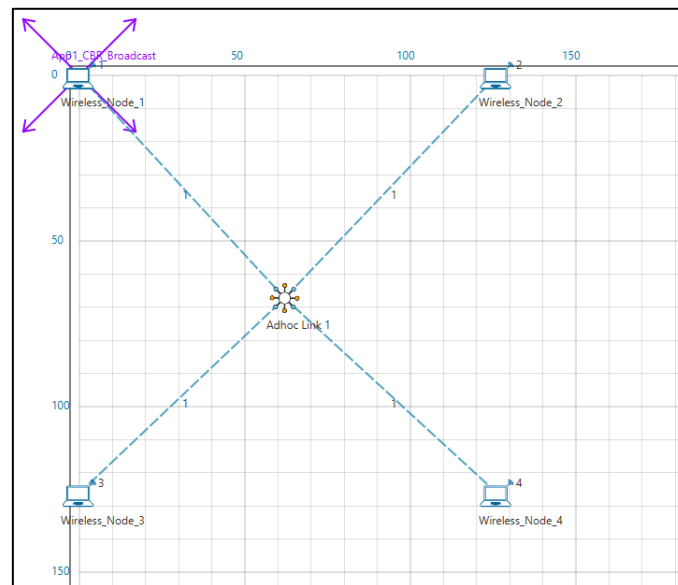
Rebroadcasting packet in NetSim MANET/VANETs

Software Used: NetSim Standard v12.2 (32bit/ 64bit), Microsoft Visual Studio 2019

Broadcasting:

Broadcasting is the process of sending a message from one node to all other nodes in an ad-hoc network. It is a fundamental operation for communication in ad-hoc networks as it allows for the update of network information and route discovery at every node.

Rebroadcasting:



Wireless Node 1 initiates a broadcast message and the message is received by nodes 2, 3 and 4. 2, 3 and 4 rebroadcast the message if they have not broadcasted that before. Furthermore, this implementation involves a Rebroadcast_Probability based on which the nodes resend the packets.

Probability-based rebroadcasting - The decision of rebroadcasting is based upon a random probability. This probability may be as simple as flipping a coin or it may be very complex involving probabilities which include parameters such as node density, duplicate packets received, battery power or a particular nodes participation within the network etc. Users can change the Rebroadcast_Probability macros present in Rebroadcast.c file as shown below:

```
ReBroadcast.c Application.c
Application (Global Scope)
12 *
13 *
14 #include "main.h"
15 #include "Application.h"
16
17 #define REBROADCAST_PROBABILITY 1.0
18 #define MAX_WAIT_FOR_REBROADCAST (100*SECOND)
19
```

Rebroadcasting in NetSim:

To implement this project in NetSim, we have created an additional Rebroadcast.c file inside Application project. The file contains the following functions:

- `void rebroadcast_packet();`

This function is used to rebroadcast the packet.

- `static bool isRebroadcastAllowed();`

This function is used to check whether rebroadcasting is allowed or not.

- `void rebroadcast_add_packet_to_info();`

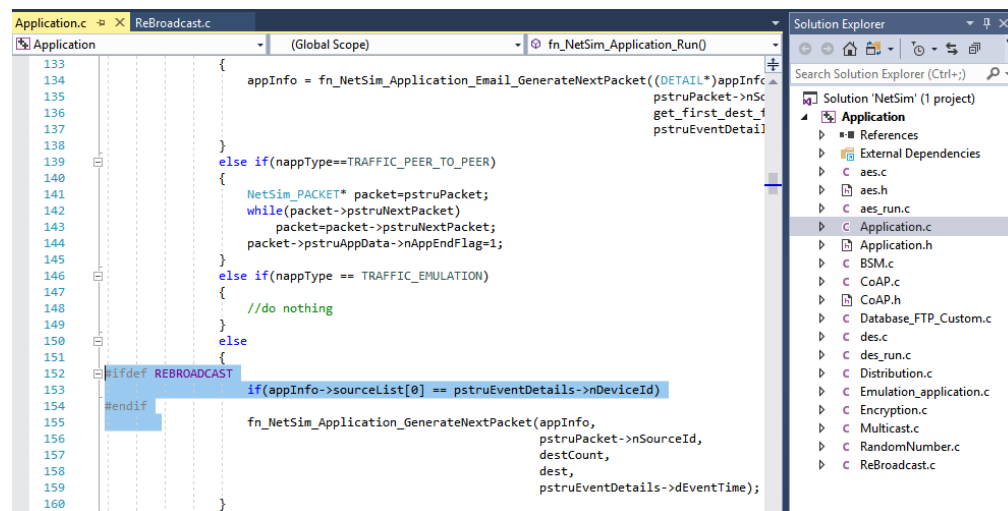
This function is used to add the packet to rebroadcast list.

- `static void cleanup_broadcast_info();`

This function is used to clean the broadcast information.

Code modifications done in NetSim:

1. We have added the following lines of code in `fn_NetSim_Application_Run()` function in the `APPLICATION_OUT_EVENT` present in `Application.c` file inside `Application` project. This is used to generate next broadcast packet if the current device is present in the source list.

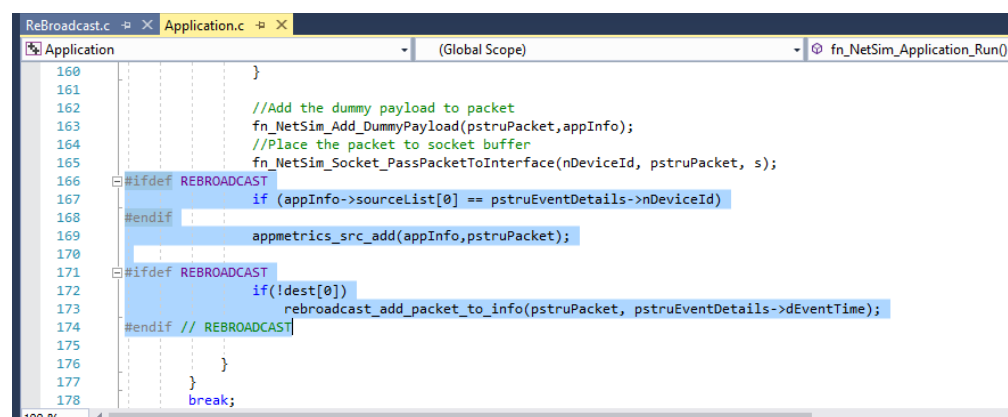


```

133 {
134     appInfo = fn_NetSim_Application_Email_GenerateNextPacket((DETAIL*)appInfo,
135     pstruPacket->nSourceId,
136     get_first_dest_id,
137     pstruEventDetails->nDeviceId);
138 }
139 else if (nappType==TRAFFIC_PEER_TO_PEER)
140 {
141     NetSim_PACKET* packet=pstruPacket;
142     while(packet->pstruNextPacket)
143     packet=packet->pstruNextPacket;
144     packet->pstruAppData->nAppEndFlag=1;
145 }
146 else if (nappType == TRAFFIC_EMULATION)
147 {
148     //do nothing
149 }
150 else
151 {
152     #ifdef REBROADCAST
153     if (appInfo->sourceList[0] == pstruEventDetails->nDeviceId)
154     #endif
155     fn_NetSim_Application_GenerateNextPacket(appInfo,
156     pstruPacket->nSourceId,
157     destCount,
158     dest,
159     pstruEventDetails->dEventTime);
160 }

```

2. The following lines of code are added in the same `fn_NetSim_Application_Run()` function in the `APPLICATION_OUT_EVENT` present in `Application.c` file inside `Application` project. The code checks if the destination is '0' i.e., Broadcast packet, then it adds the packet to rebroadcast list.



```

160 }
161 //Add the dummy payload to packet
162 fn_NetSim_Add_DummyPayload(pstruPacket,appInfo);
163 //Place the packet to socket buffer
164 fn_NetSim_Socket_PassPacketToInterface(nDeviceId, pstruPacket, s);
165 #ifdef REBROADCAST
166 if (appInfo->sourceList[0] == pstruEventDetails->nDeviceId)
167 #endif
168 appmetrics_src_add(appInfo,pstruPacket);
169 #ifdef REBROADCAST
170 if (!dest[0])
171 rebroadcast_add_packet_to_info(pstruPacket, pstruEventDetails->dEventTime);
172 #endif // REBROADCAST
173 }
174 }
175 }
176 }
177 }
178 break;

```

3. Now add the following code in `fn_NetSim_Application_Run()` function in the `APPLICATION_IN_EVENT` present in `Application.c` file inside `Application` project. It checks

whether the destination is '0' or not. If it is '0', then it rebroadcasts the packet or else deletes the packet.

```

200 {
201     process_saej2735_packet(pstruPacket);
202 }
203
204 #ifdef REBROADCAST
205     UINT destCount;
206     NETSIM_ID* dest = get_dest_from_packet(pstruPacket, &destCount);
207     if (!dest[0])
208     {
209         rebroadcast_packet(pstruPacket,
210             pstruEventDetails->nDeviceId,
211             pstruEventDetails->dEventTime);
212     }
213     else
214     {
215         //Delete the packet
216         fn_NetSim_Packet_FreePacket(pstruPacket);
217     }
218 #endif // REBROADCAST
219 #endif
220
221 #endif
222
223

```

```

186 fnValidatePacket(pstruPacket);
187 pstruappinfo=appInfo[pstruPacket->pstruAppData->nApplicationId-1];
188 pstruPacket->pstruAppData->dEndTime = pstruEventDetails->dEventTime;
189 fn_NetSim_Application_Plot(pstruPacket);
190
191 #ifdef REBROADCAST
192     if (pstruappinfo->sourceList[0] == pstruPacket->nSourceId)
193     {
194         appmetrics_dest_add(pstruappinfo, pstruPacket, pstruEventDetails->nDeviceId);
195         if(pstruappinfo->nAppType==TRAFFIC_PEER_TO_PEER && pstruPacket->pstruAppData->nAppEndFlag==1)
196         {

```

4. We have added the following function declarations in Application.h file.

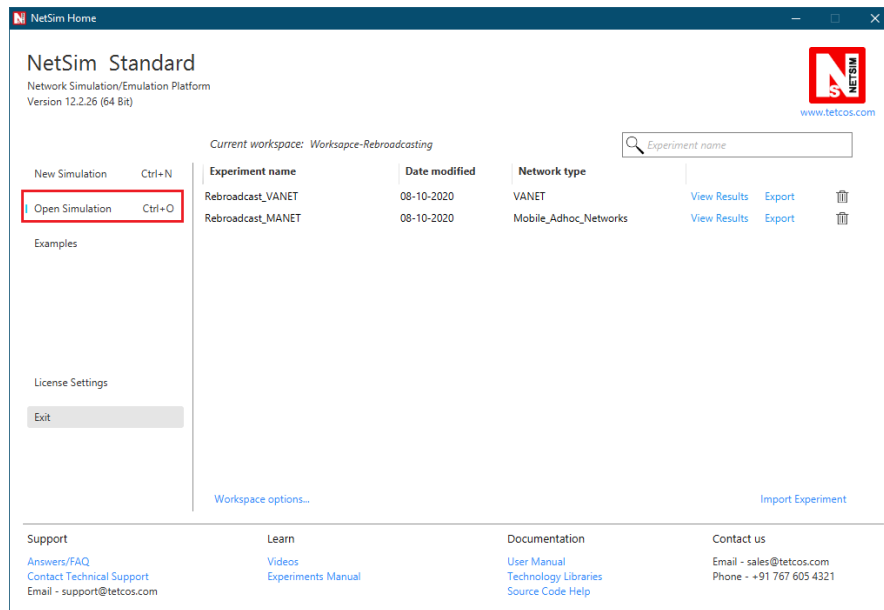
```

436 void appmetrics_dest_add(app_info* appInfo, netsim_packet* packet, netsim_id dest);
437 int fn_netsim_application_metrics_f(metricswriter metricswriter);
438
439 //Application Interface function
440 void fnCreatePort(app_info* info);
441 int fnCreateSocketBuffer(app_info* appInfo);
442
443 int fn_netsim_add_dummyPayload(netsim_packet* packet, app_info*);
444
445 //Encryption
446 char xor_encrypt(char ch, long key);
447 int aes256(char* str, int* len);
448 int des(char* buf, int* len);
449
450
451 #ifdef REBROADCAST
452 void rebroadcast_add_packet_to_info(netsim_packet* packet,
453     double time);
454 void rebroadcast_packet(netsim_packet* packet,
455     netsim_id device,
456     double time);
457 #endif

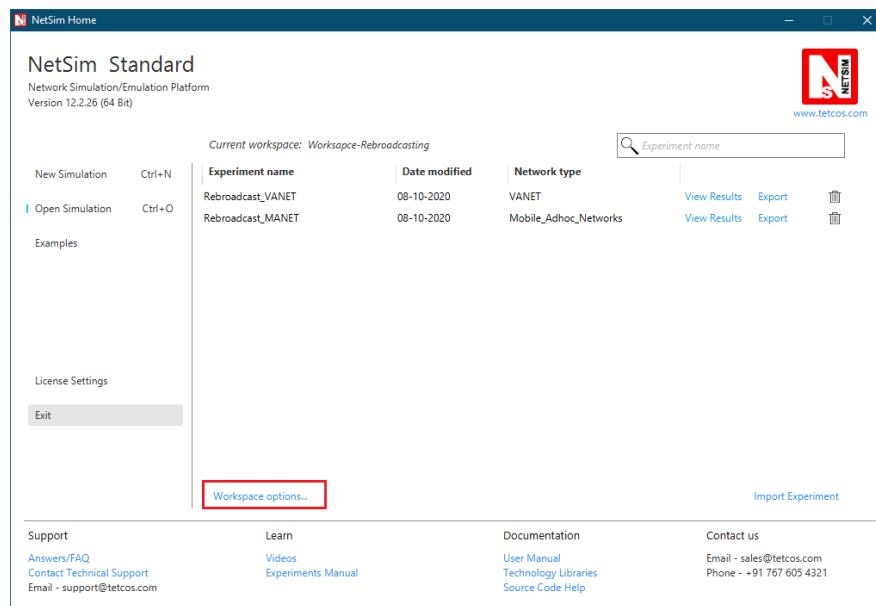
```

Steps:

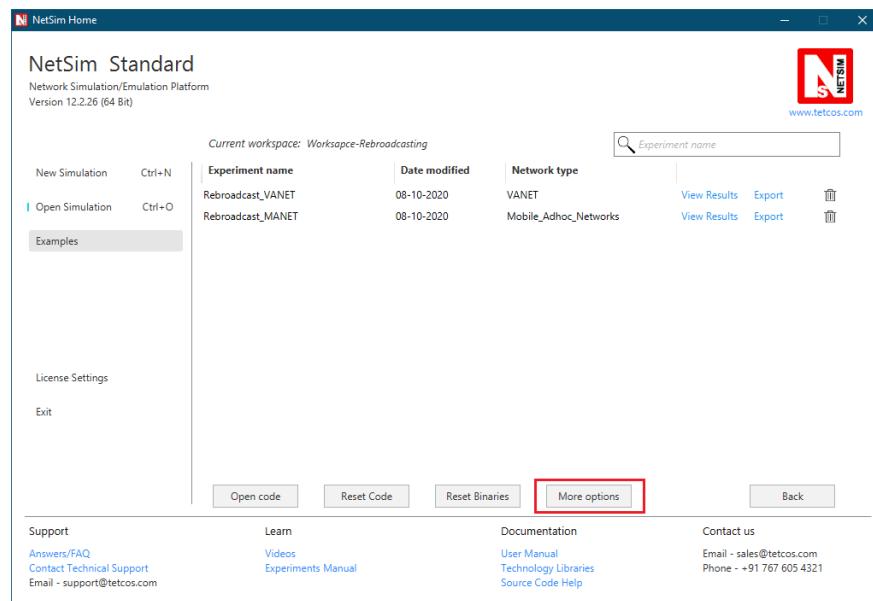
- After you unzip the downloaded project folder, Open NetSim Home Page click on **Open Simulation** option,



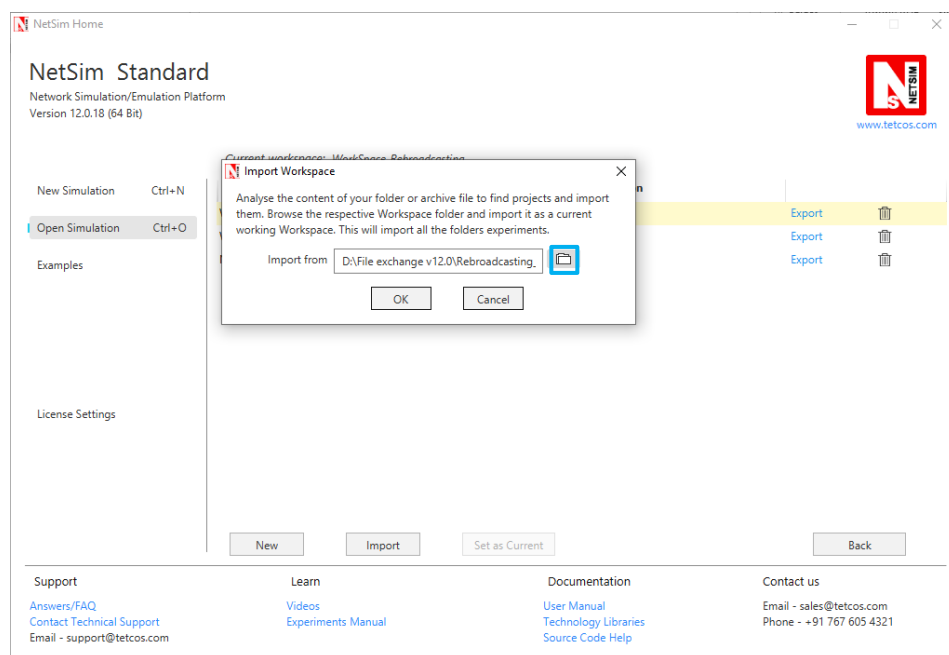
- Click on **Workspace options**



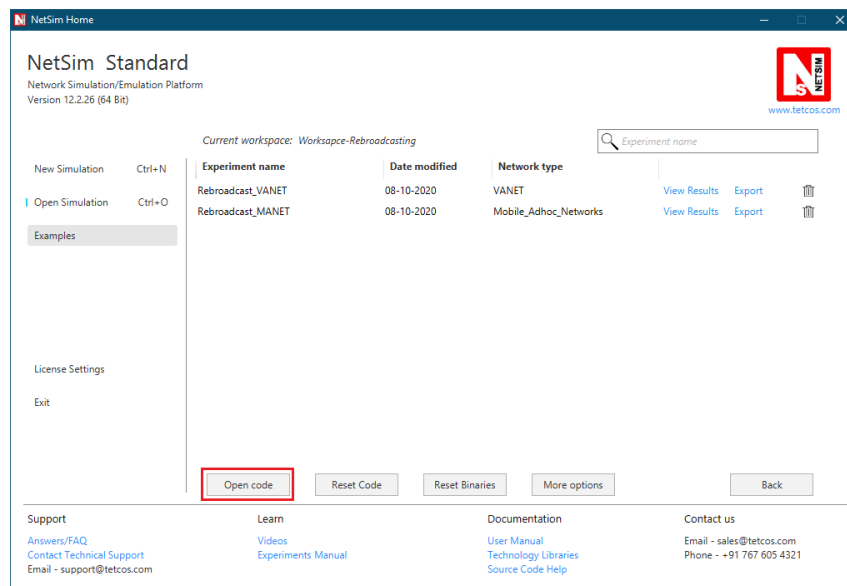
- Click on **More Options**,



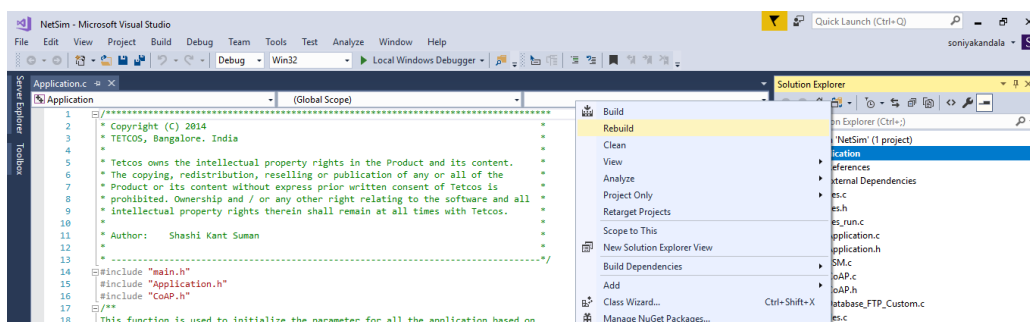
- Click on **Import**, browse the extracted folder path and go into the Workspace_Rebroadcasting directory. Click on the select folder button and then on **OK**.



- Go to home page, Click on **Open Simulation** → **Workspace options** → **Open code**



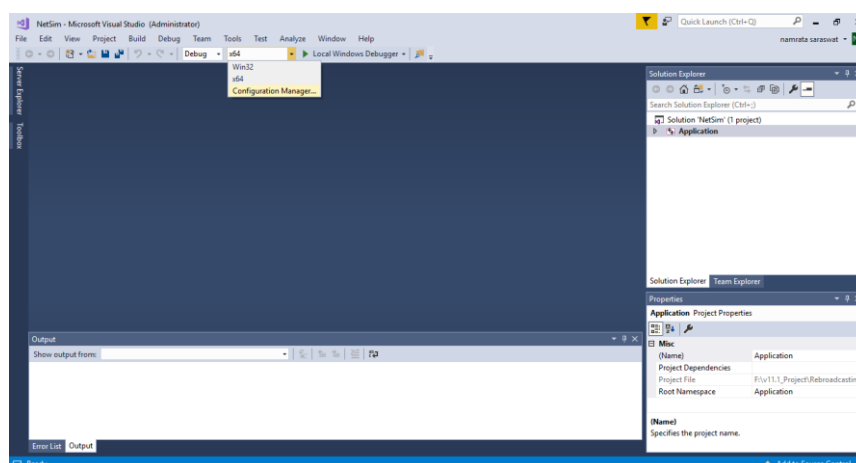
- Right click on Solution in Solution Explorer and select 'Rebuild solution'.



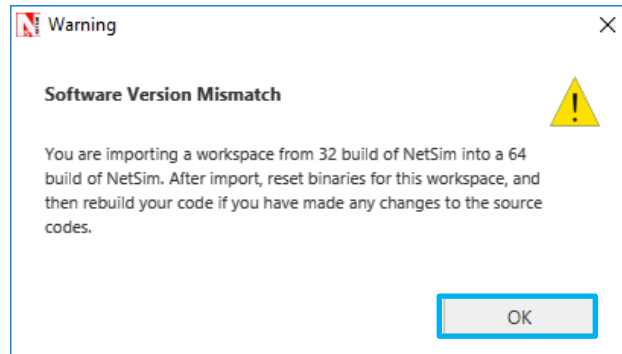
- Upon rebuilding, **libApplication.dll** will automatically get updated in the respective bin folder of the current workspace.

Note:

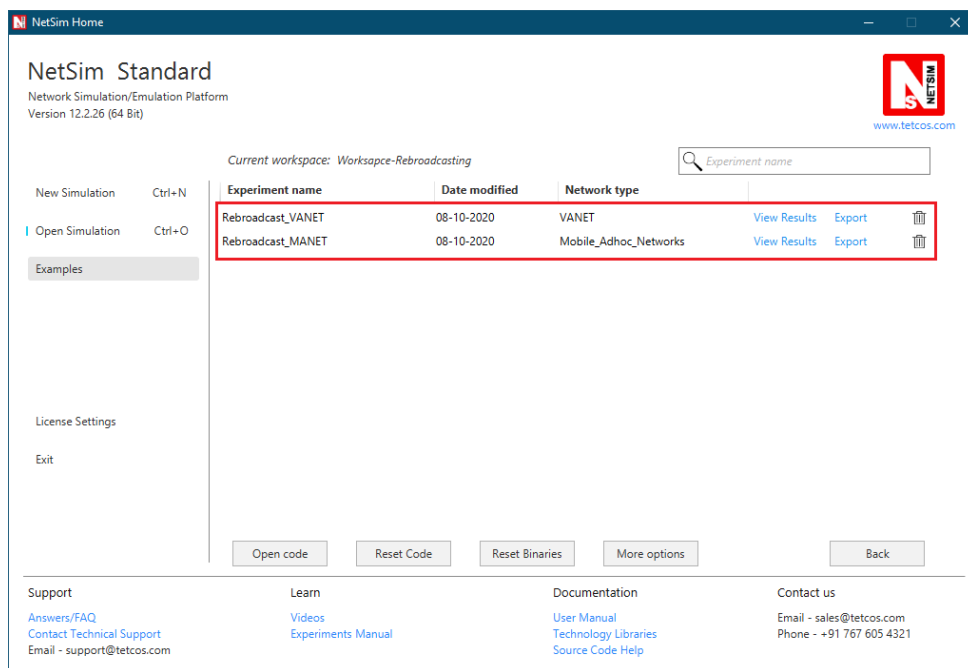
1. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



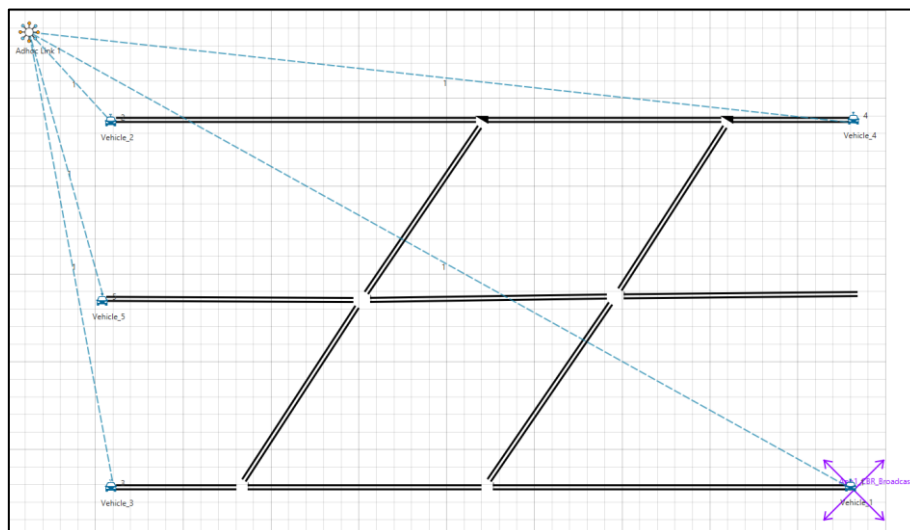
2. While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



- Go to NetSim home page, click on **Open Simulation**, Click on **Rebroadcasting_VANET_Example/ Rebroadcasting_MANET_Example** and run the simulation for 100 seconds.



VANET SCENARIO:



- In the above scenario, Vehicle-1 is broadcasting the packet and it is received by the Vehicles 2, 3 and 4. Then Vehicles 2, 3, and 4 will rebroadcast the same packet based on the probability value in Rebroadcast.c file.
- After simulation, open Packet Trace and filter Packet_Id to '1' or any other id and observe that the nodes other than source are rebroadcasting the same packet.

	A	B	C	D	E	F	G	H
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
2	1	0	CBR	App1_CBR	NODE-1	Broadcast-0	NODE-1	NODE-2
3	1	0	CBR	App1_CBR	NODE-1	Broadcast-0	NODE-1	NODE-3
4	1	0	CBR	App1_CBR	NODE-1	Broadcast-0	NODE-1	NODE-4
5	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-1
6	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-3
7	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-4
8	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-1
9	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-2
10	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-4
20	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-1
21	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-2
22	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-3

- Note that Users SHOULD NOT use the performance metrics provided at the end of simulation but should rather calculate the network performance metrics from the packet trace.
- Users can also create their own network scenarios in **Single MANET/VANET** and run the simulation.