

## Rebroadcasting packet in NetSim MANET/VANETs

**Software Used:** NetSim Standard v13.0 (32bit/ 64bit), Microsoft Visual Studio2019

**Project Download Link:**

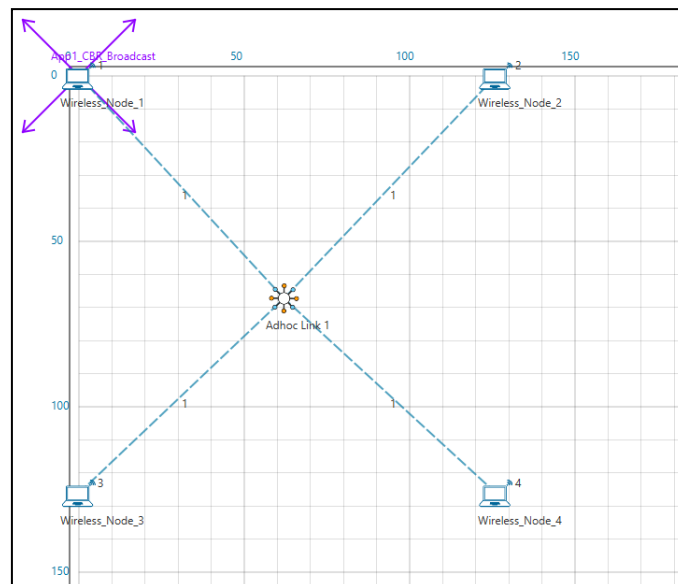
<https://github.com/NetSim->

[TETCOS/NetSim Batch Auto Simulation v13.0/archive/refs/heads/main.zip](https://github.com/NetSim-NetSim/NetSim-Batch-Auto-Simulation-v13.0/archive/refs/heads/main.zip)

### 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();`

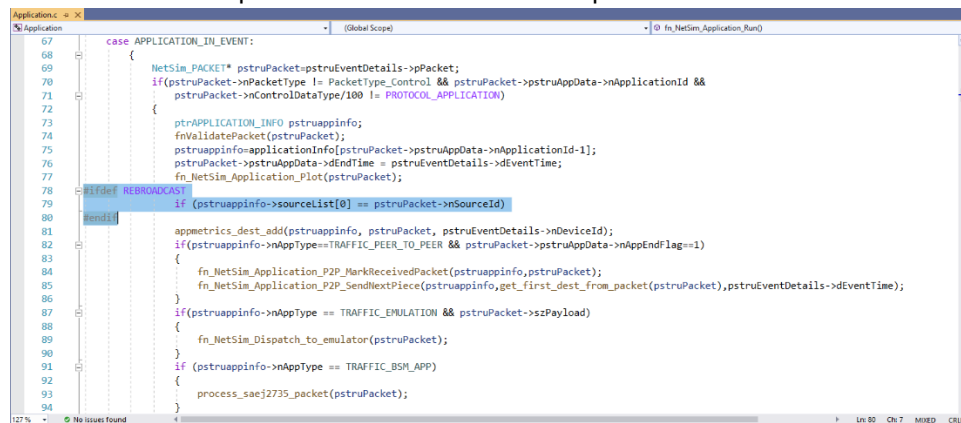
This function is used to add the packet to rebroadcast list.

- `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_IN_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.

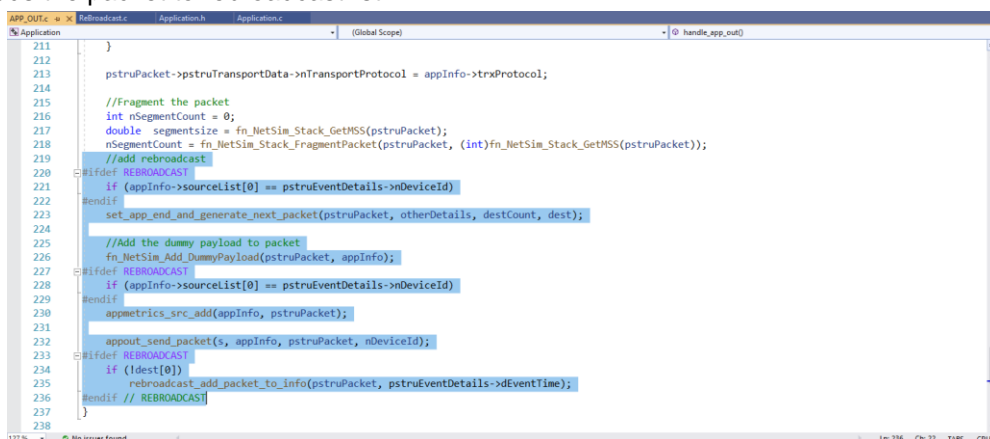


```

67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
    case APPLICATION_IN_EVENT:
    {
        NetSim_PACKET* pstruPacket=pstruEventDetails->pPacket;
        if(pstruPacket->nPacketType != PacketType_Control && pstruPacket->pstruAppData->nApplicationId &&
           pstruPacket->nControlData/100 != PROTOCOL_APPLICATION)
        {
            ptrAPPLICATION_INFO pstruappinfo;
            fnValidatePacket(pstruPacket);
            pstruappinfo=applicationInfo[pstruPacket->pstruAppData->nApplicationId-1];
            pstruPacket->pstruAppData->dEndTime = pstruEventDetails->dEventTime;
            fn_NetSim_Application_Plot(pstruPacket);
        }
        #ifdef REBROADCAST
        if (pstruappinfo->sSourceList[0] == pstruPacket->nSourceId)
        {
            appmetrics_dest_add(pstruappinfo, pstruPacket, pstruEventDetails->nDeviceId);
            if(pstruappinfo->nAppType==TRAFFIC_PEER_TO_PEER && pstruPacket->pstruAppData->nAppEndFlag==1)
            {
                fn_NetSim_Application_P2P_MarkReceivedPacket(pstruappinfo,pstruPacket);
                fn_NetSim_Application_P2P_SendNextPiece(pstruappinfo,get_first_dest_from_packet(pstruPacket),pstruEventDetails->dEventTime);
            }
            if(pstruappinfo->nAppType == TRAFFIC_EMULATION && pstruPacket->szPayload)
            {
                fn_NetSim_Dispatch_to_emulator(pstruPacket);
            }
            if (pstruappinfo->nAppType == TRAFFIC_BSM_APP)
            {
                process_saej2735_packet(pstruPacket);
            }
        }
    }
    #endif
}

```

2. The following lines of code are added in the `handle_app_out()` function present in `APP_OUT.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.



```

211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
    }
    pstruPacket->pstruTransportData->nTransportProtocol = appInfo->trxProtocol;

    //Fragment the packet
    int nSegmentCount = 0;
    double segmentSize = fn_NetSim_Stack_GetMSS(pstruPacket);
    nSegmentCount = fn_NetSim_Stack_FragmentPacket(pstruPacket, (int)fn_NetSim_Stack_GetMSS(pstruPacket));

    //add rebroadcast
    #ifdef REBROADCAST
    if (appInfo->sSourceList[0] == pstruEventDetails->nDeviceId)
    {
        set_app_end_and_generate_next_packet(pstruPacket, otherDetails, destCount, dest);

        //Add the dummy payload to packet
        fn_NetSim_Add_DummyPayload(pstruPacket, appInfo);
    }
    #endif
    #ifdef REBROADCAST
    if (appInfo->sSourceList[0] == pstruEventDetails->nDeviceId)
    {
        appmetrics_src_add(appInfo, pstruPacket);
        appout_send_packet(s, appInfo, pstruPacket, nDeviceId);
    }
    #endif
    if (ldest[0])
    {
        rebroadcast_add_packet_to_info(pstruPacket, pstruEventDetails->dEventTime);
    }
    #endif // REBROADCAST
}

```

3. Now add the following code in `fn_NetSim_Application_Run()` function in `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.

```

91  if (pstruappinfo->nAppType == TRAFFIC_BSM_APP)
92  {
93      process_saej2735_packet(pstruPacket);
94  }
95  #ifdef REBROADCAST
96      uint destCount;
97      NETSIM_ID* dest = get_dest_from_packet(pstruPacket, &destCount);
98      if (ldest[0])
99      {
100         rebroadcast_packet(pstruPacket,
101                             pstruEventDetails->nDeviceId,
102                             pstruEventDetails->dEventTime);
103     }
104     else
105     {
106     }
107     //Delete the packet
108     fn_NetSim_Packet_FreePacket(pstruPacket);
109     //add
110 }
111 // REBROADCAST
112 #endif
113 }
114 }
115 // Here which type is placed is only getting processed next one is not getting processed
116 else if (pstruPacket->nControlDataType == packet_COAP_REQUEST)
117 {
118 }

```

```

67  case APPLICATION_IN_EVENT:
68  {
69      NetSim_PACKET* pstruPacket=pstruEventDetails->pPacket;
70      if(pstruPacket->nPacketType != PacketType_Control && pstruPacket->pstruAppData->nApplicationId &&
71          pstruPacket->nControlDataType/100 != PROTOCOL_APPLICATION)
72      {
73          ptrAPPLICATION_INFO pstruappinfo;
74          fnValidatePacket(pstruPacket);
75          pstruappinfo=applicationInfo(pstruPacket->pstruAppData->nApplicationId-1);
76          pstruPacket->pstruAppData->dEndTime = pstruEventDetails->dEventTime;
77          fn_NetSim_Application_Plot(pstruPacket);
78      }
79      #ifdef REBROADCAST
80      if (pstruappinfo->sSourceList[0] == pstruPacket->nSourceId)
81      {
82          appmetrics_dest_add(pstruappinfo, pstruPacket, pstruEventDetails->nDeviceId);
83          if(pstruappinfo->nAppType==TRAFFIC_PEER_TO_PEER && pstruPacket->pstruAppData->nAppEndFlag==1)
84          {
85              fn_NetSim_Application_P2P_MarkReceivedPacket(pstruappinfo,pstruPacket);
86              fn_NetSim_Application_P2P_SendNextPiece(pstruappinfo,get_first_dest_from_packet(pstruPacket),pstruEventDetails->dEventTime);
87          }
88          if(pstruappinfo->nAppType == TRAFFIC_EMULATION && pstruPacket->szPayload)
89          {
90              fn_NetSim_Dispatch_to_emulator(pstruPacket);
91          }
92          if (pstruappinfo->nAppType == TRAFFIC_BSM_APP)
93          {
94              process_saej2735_packet(pstruPacket);
95          }
96      }
97      #endif
98  }

```

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

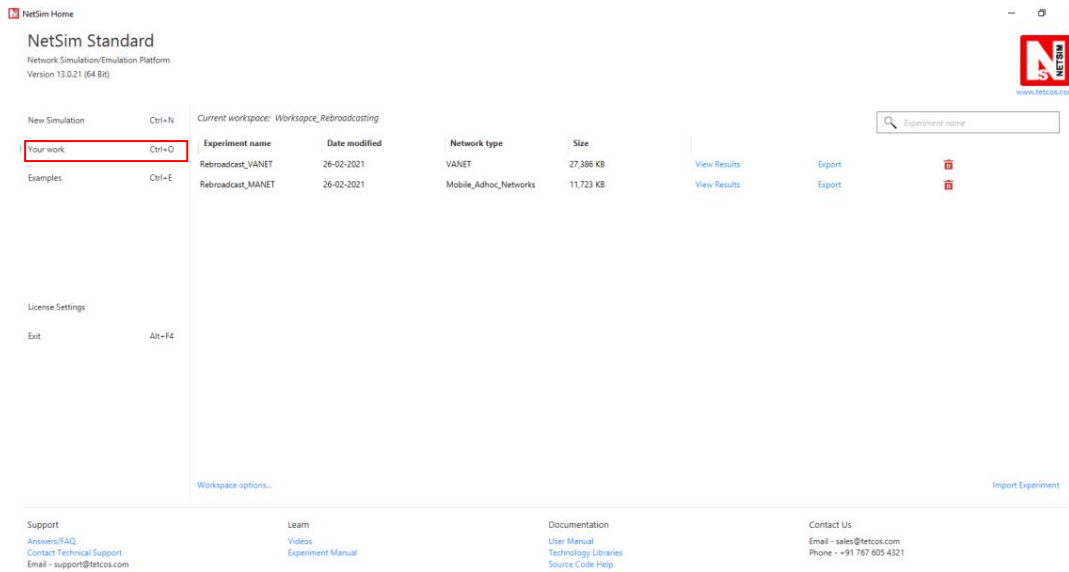
```

452  int fn_NetSim_Application_Metrics_F(PMETRICSWRITER metricsWriter);
453
454
455  //Application Interface function
456  void fnCreatePort(ptrAPPLICATION_INFO info);
457  int fnCreateSocketBuffer(ptrAPPLICATION_INFO appInfo);
458  void P2P_create_socket(ptrAPPLICATION_INFO appInfo, NETSIM_ID src, NETSIM_ID dest);
459
460  int fn_NetSim_Add_DummyPayload(NetSim_PACKET* packet, ptrAPPLICATION_INFO);
461
462  //Encryption
463  char xor_encrypt(char ch, long key);
464  int aes256(char* str, int* len);
465  int des(char* buf, int* len);
466
467  //Application event handler
468  void handle_app_out();
469  #define REBROADCAST
470  void rebroadcast_add_packet_to_info(NetSim_PACKET* packet, double time);
471  void rebroadcast_packet(NetSim_PACKET* packet, NETSIM_ID devId, double time);
472  #endif
473

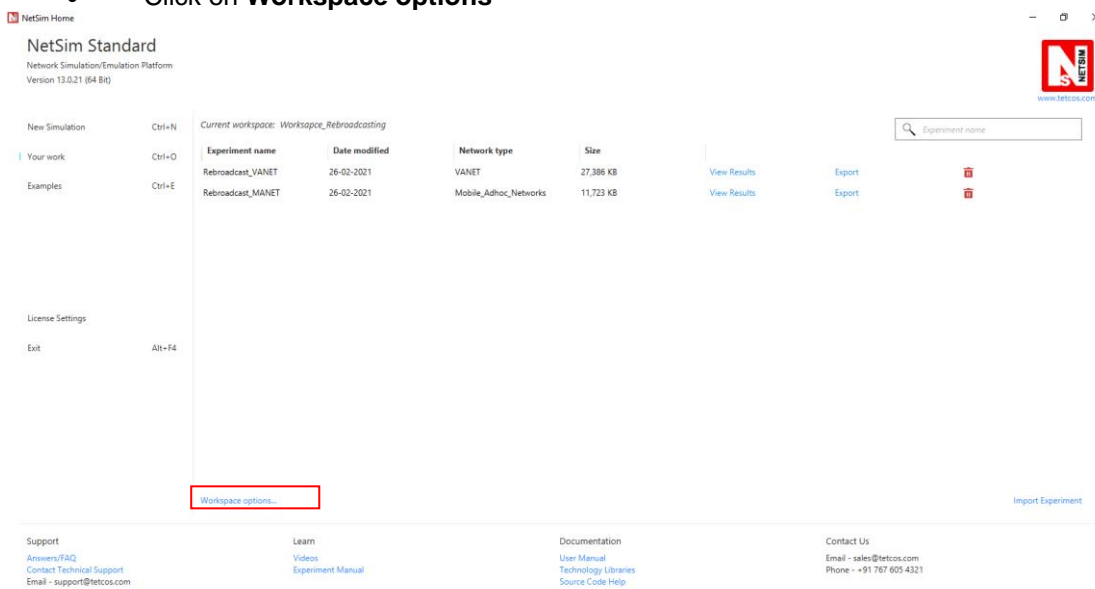
```

## Steps:

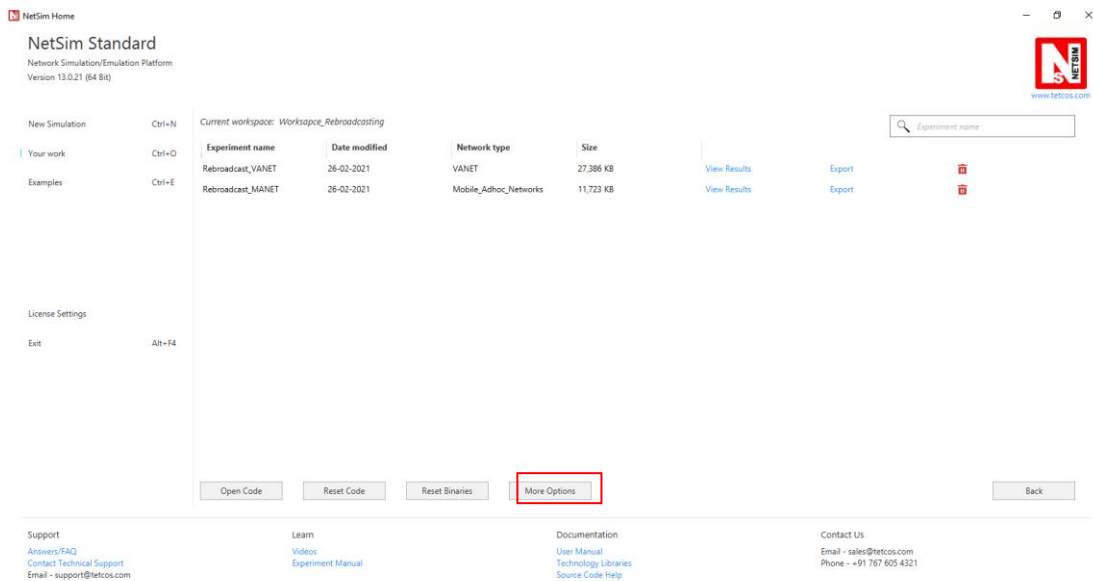
- After you unzip the downloaded project folder, Open NetSim Home Page click on **Your Work** option,



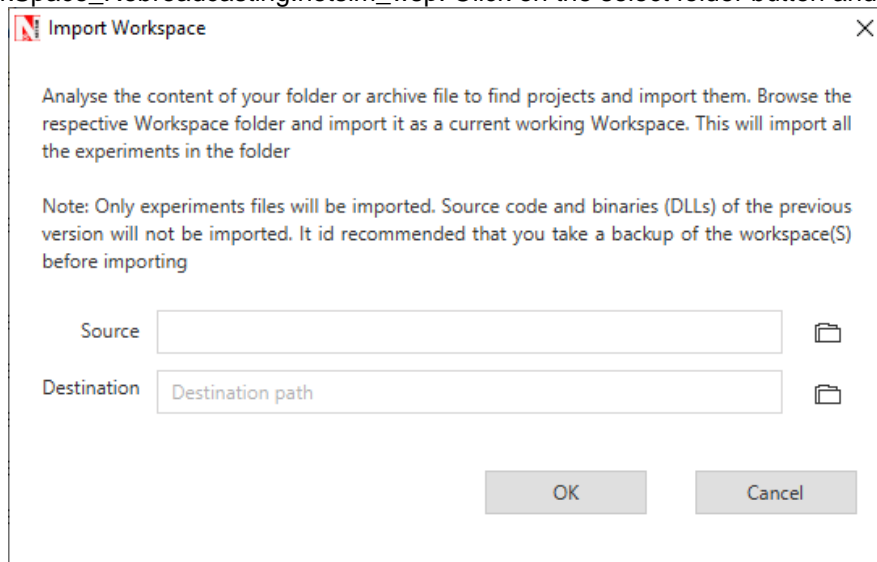
- Click on **Workspace options**



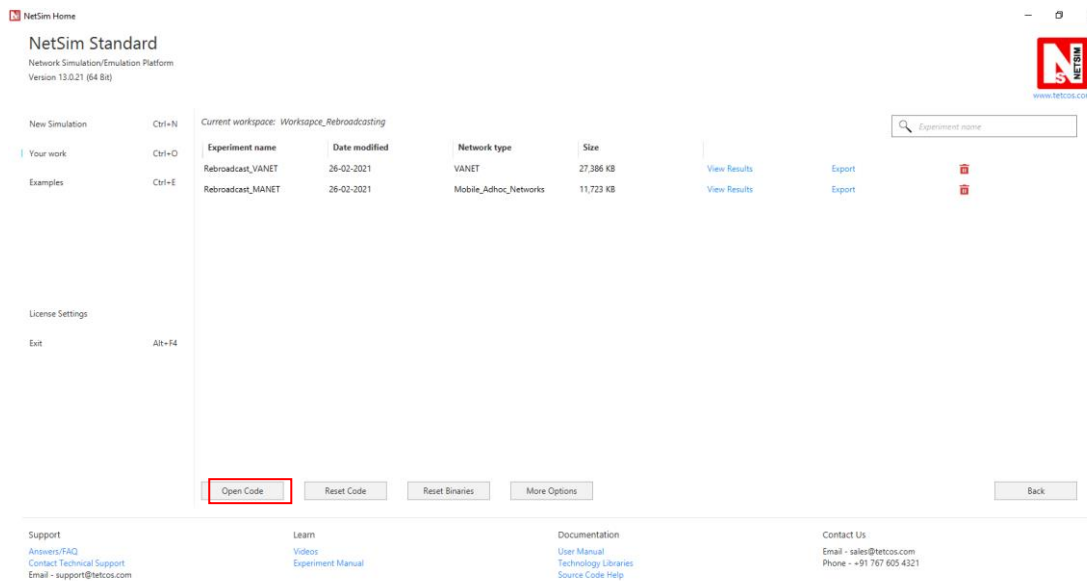
- Click on **More Options**,



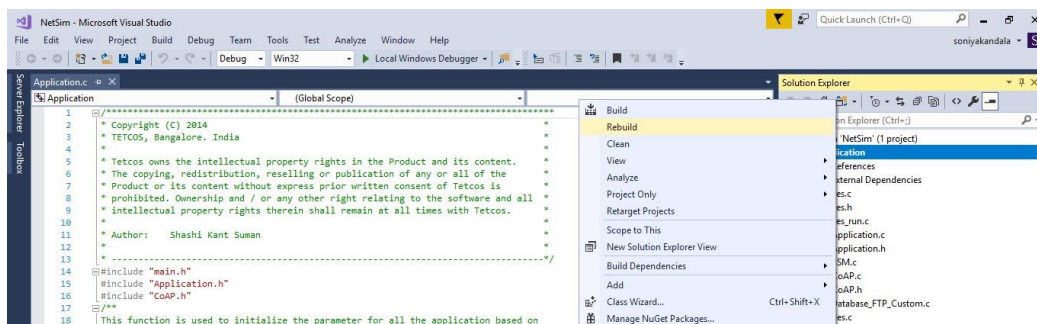
- Click on **Import**, browse the extracted folder path and go into the `WorkSpace_Rebroadcasting.netsim_wsp`. Click on the select folder button and then on **OK**.



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



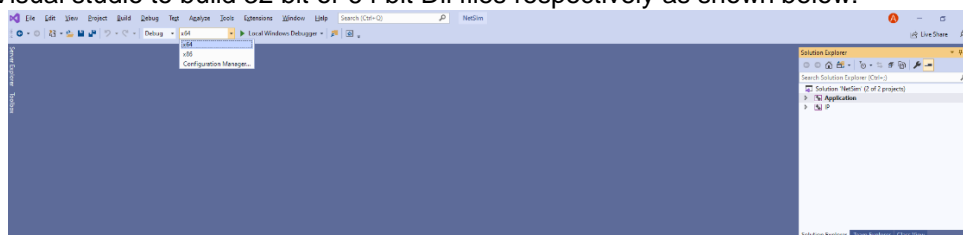
- Right click on Solution in Solution Explorer and select 'Rebuild solution'.



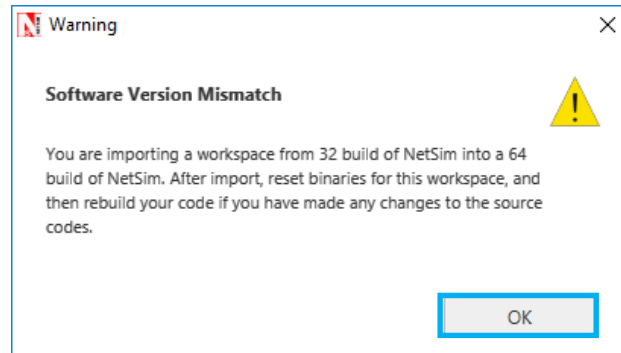
- Upon rebuilding, **libApplication.dll** will automatically get updated in the respective bin folder of the current workspace.

#### Note:

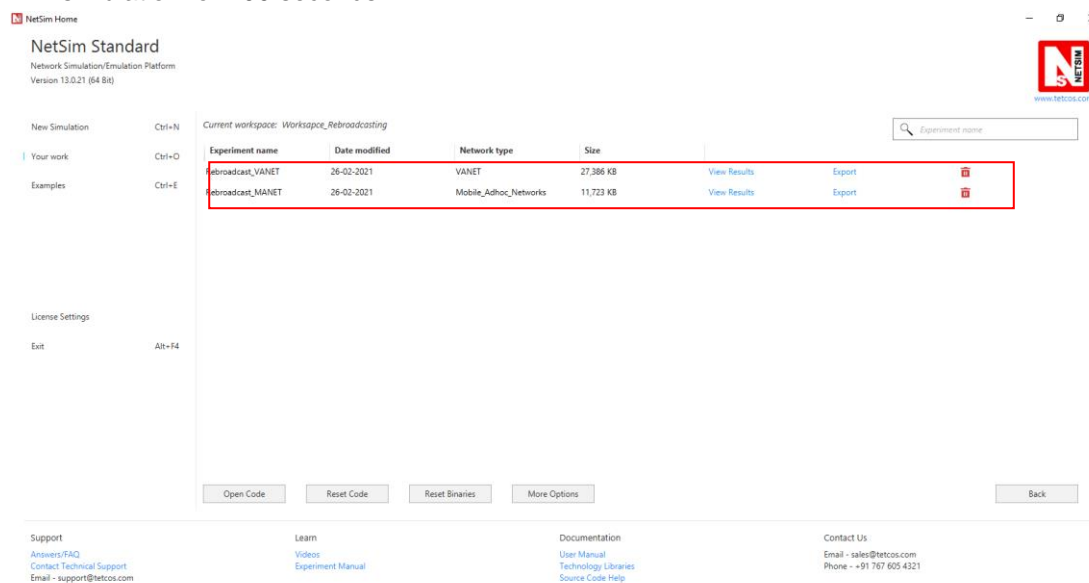
1. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit DLL files respectively as shown below:



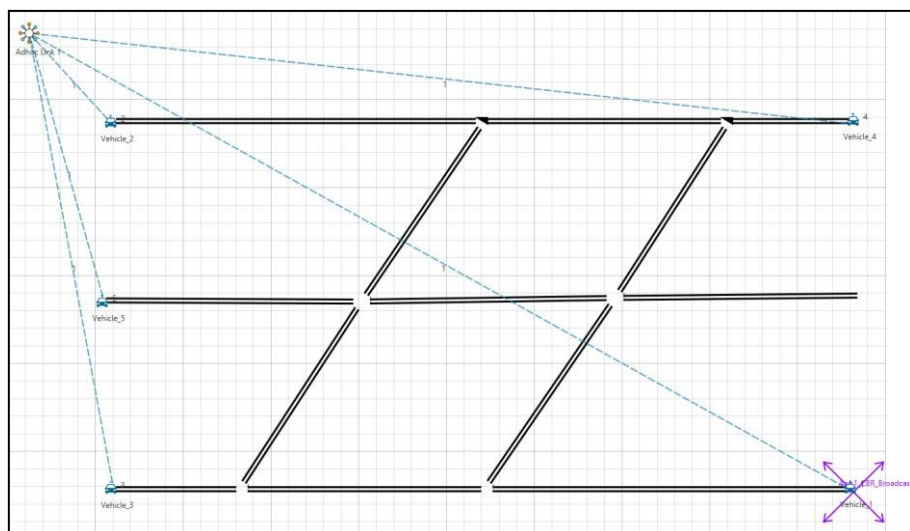
2. While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



- Go to NetSim home page, click on **Your work**, 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
	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
1								
2	1		O CBR	App1_CBR	NODE-1	Broadcast-0	NODE-1	NODE-2
3	1		O CBR	App1_CBR	NODE-1	Broadcast-0	NODE-1	NODE-3
4	1		O CBR	App1_CBR	NODE-1	Broadcast-0	NODE-1	NODE-4
5	1		O CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-1
6	1		O CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-3
7	1		O CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-4
8	1		O CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-1
9	1		O CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-2
10	1		O CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-4
20	1		O CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-1
21	1		O CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-2
22	1		O 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.