Rebroadcasting packet in NetSim MANET\VANETs

Software: NetSim Standard V14.0, Microsoft Visual Studio 2022

Project Download Link:
https://github.com/NetSim-TETCOS/MANET-VANET-Rebroadcast-v14.0/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and setup the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

Introduction:

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: It is the process of broadcasting the received message to all the other nodes in the ad-hoc network.

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.

Real-world Context of MANET:

In the context of military communication, MANET rebroadcasting can be employed to enable efficient and secure communication among Soldiers in a distributed network.

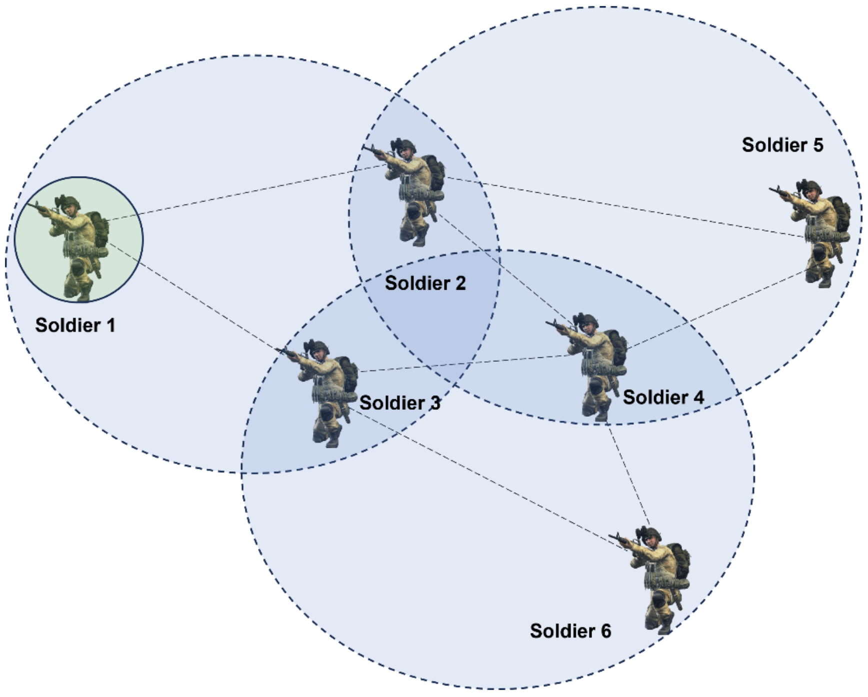


Figure 1: Soldiers using MANET rebroadcasting to communicate in a distributed network.

1. Soldier 1 broadcasts the message to all of his neighbors. This means that all soldiers within the range of Soldier 1 will receive the message.

2. Once a neighbor soldiers receives the message, it will rebroadcast it to all of its neighbor Soldiers which are in range of that soldier. This process continues until all soldiers have received the message in the Network.
3. As you can observe, all the Soldiers in the network are able to have a conversation using MANET rebroadcasting. This allows them to share information and coordinate their actions quickly and effectively.
4. Rebroadcasting allows the soldiers to share information about the enemy's position quickly and efficiently.

Real-world Context of VANETs:

VANET rebroadcasting enables efficient and secure communication among vehicles in a distributed network, allowing for real-time information sharing, improved safety, and enhanced traffic management.

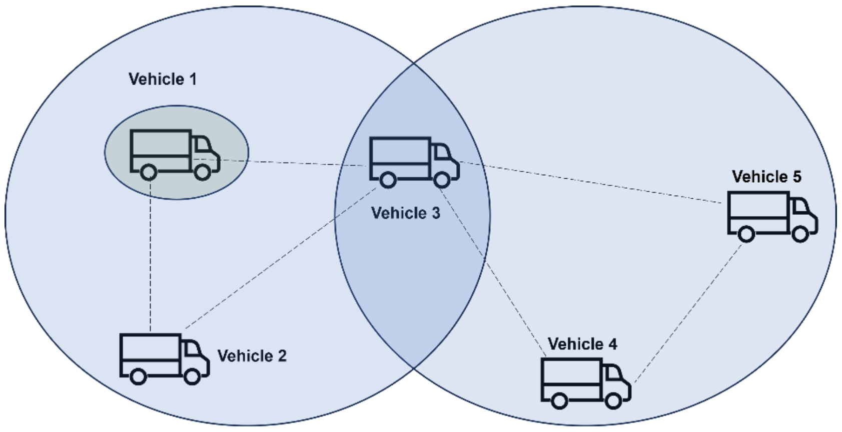
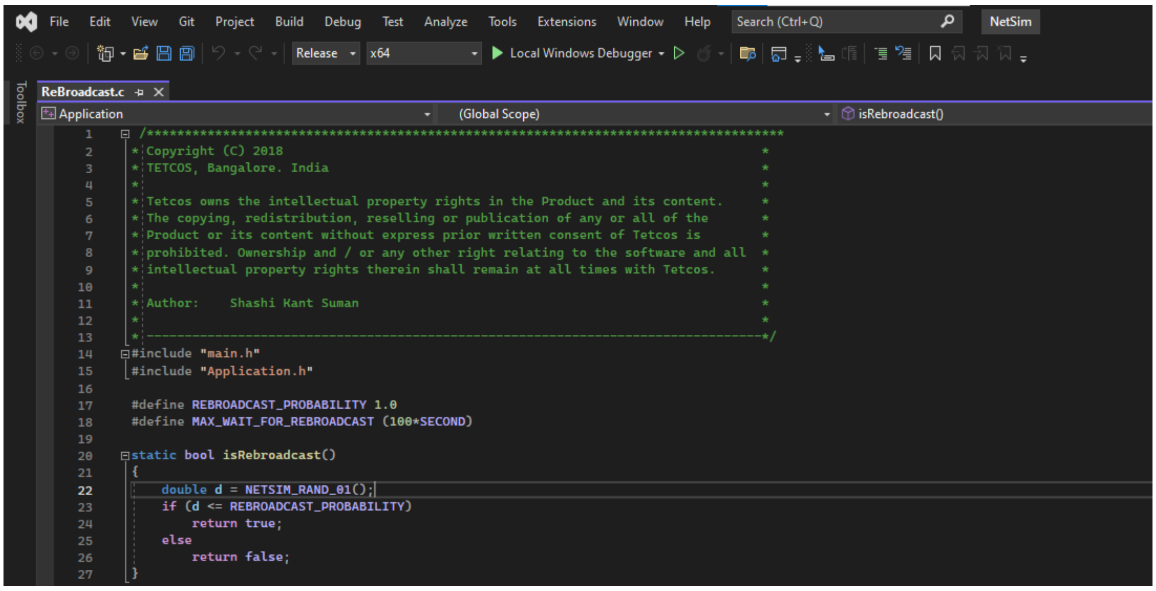


Figure 2: Vehicles using VANET rebroadcasting to communicate in a distributed network.

1. Vehicle 1 broadcasts the message to all of his neighbors. This means that all soldiers within range of Vehicle 1 will receive the message.
2. Once a neighbor soldier receives the message, it will rebroadcast it to all its neighbor Vehicles that are in range of that Vehicle. This process continues until all Vehicles have received the message in the Network.
3. As you can observe, all the Vehicles in the network are able to have a conversation using VANET rebroadcasting. This allows them to share information and coordinate their actions quickly and effectively.
4. Rebroadcasting allows vehicles to share information about accidents, roadblocks, and other hazards, which helps to improve road safety by warning drivers of potential dangers.

Implementation:

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 that 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:



```c
/*******************************************************************************
 * Copyright (C) 2018                                                          *
 * 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:    Shashi Kant Suman                                                *
 *                                                                             *
 * ---------------------------------------------------------------------------*/
#include "main.h"
#include "Application.h"

#define REBROADCAST_PROBABILITY 1.0
#define MAX_WAIT_FOR_REBROADCAST (100*SECOND)

static bool isRebroadcast()
{
    double d = NETSIM_RAND_01();
    if (d <= REBROADCAST_PROBABILITY)
        return true;
    else
        return false;
}
```

Figure 3: Rebroadcast Probability

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.

Example:

1. The **MANET_VANET_Rebroadcast_Workspace** comes with a sample network configuration that are already saved. To open this example, go to Your work in the home screen of NetSim and click on the **Rebroadcast_VANET_Example/Rebroadcast_MANET_Example** from the list of experiments.

2. Run the simulation for 100 seconds.

MANET SCENARIO:

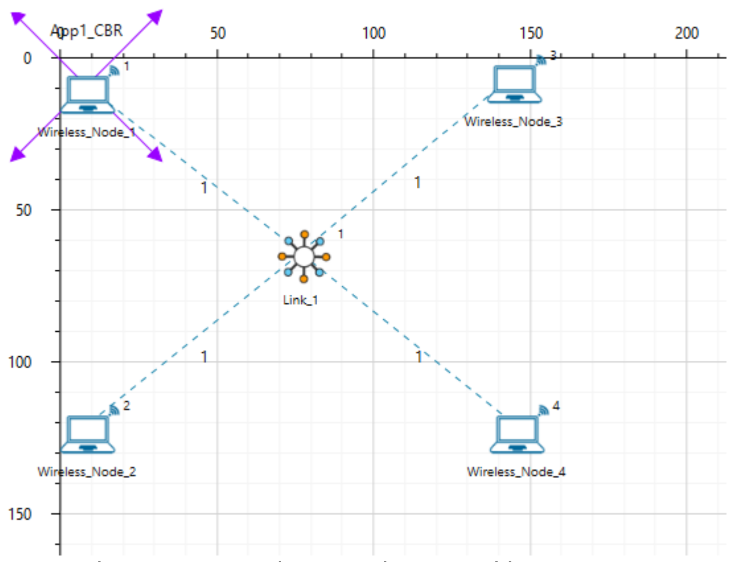
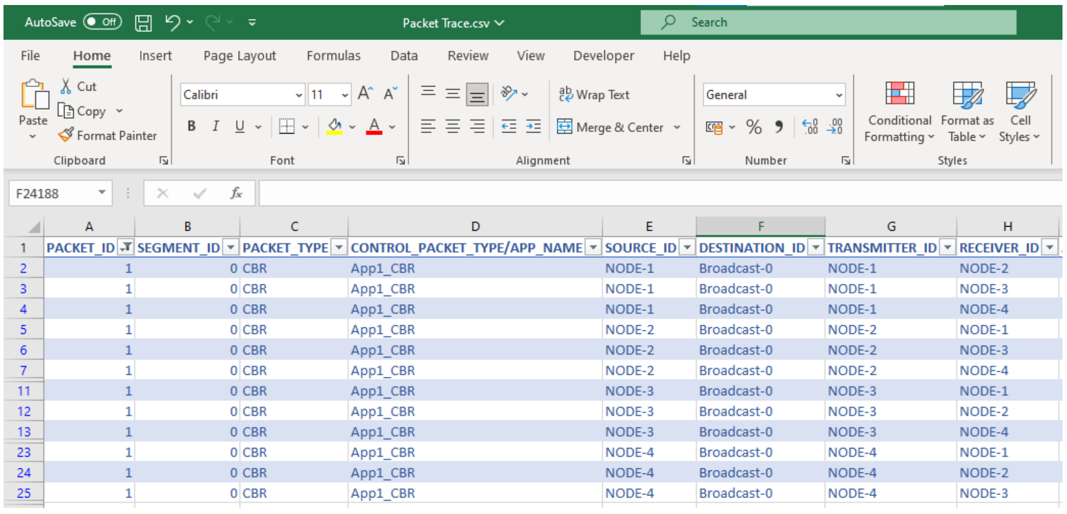


Figure 4: Network Scenario created in MANET

Results and discussion (MANET):

- 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.
- Wireless_Node_1 is broadcasting the packet and it is received by the Wireless_Node 2, 3, and 4 . Then Wireless_Node 2, 3, and 4 will rebroadcast the same packet based on the probability value in Rebroadcast.c file.



	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
11	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-1
12	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-2
13	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-4
23	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-1
24	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-2
25	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-3

Figure 6: NetSim Packet Trace

VANETs SCENARIO:

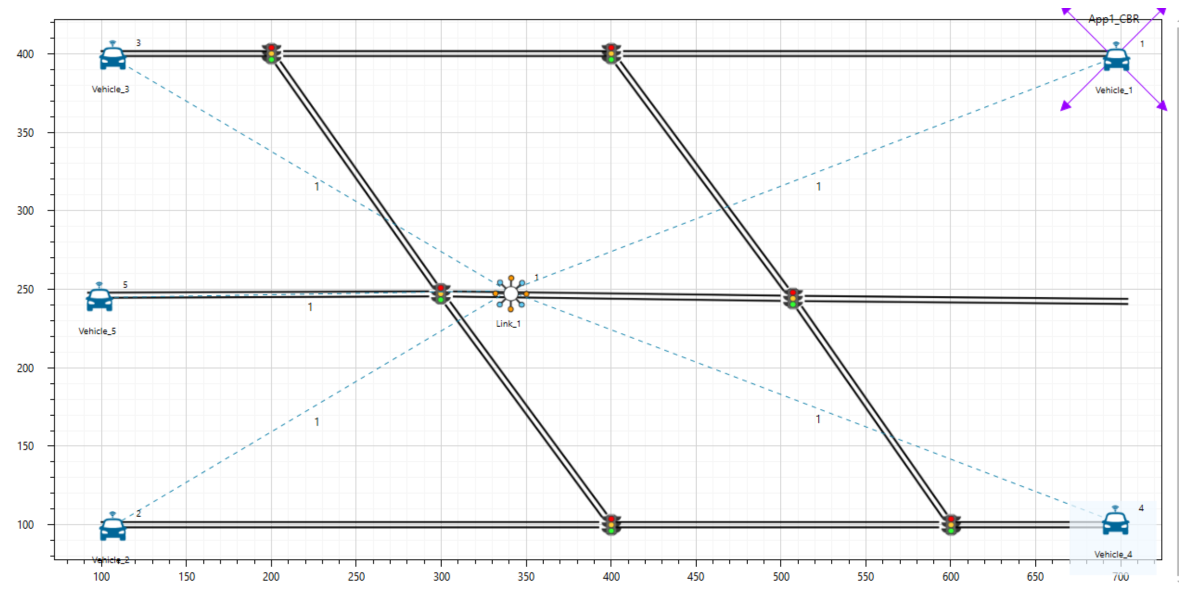
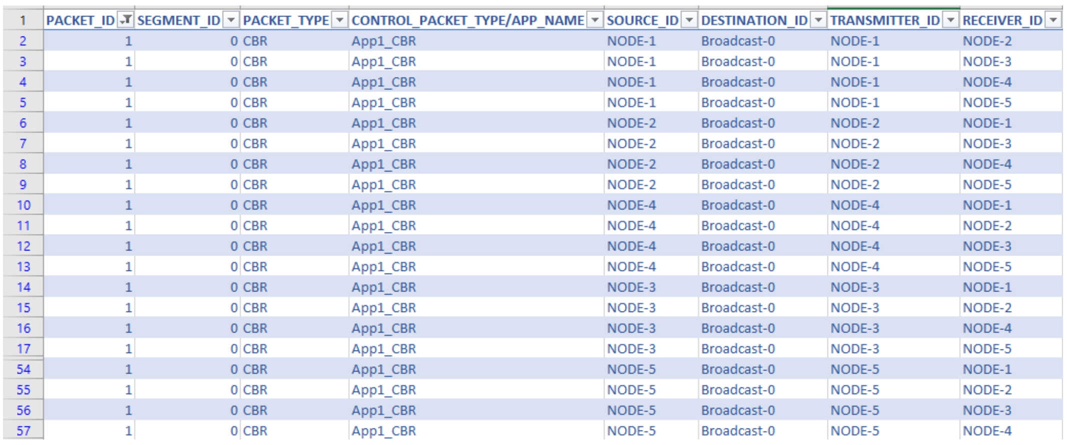


Figure 5: Network Scenario created in VANET

Results and discussion (VANETs):

- 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.
- Vehicle-1 is broadcasting the packet and it is received by the Vehicles 2, 3, 4 and 5. Then Vehicles 2, 3, 4 and 5 will rebroadcast the same packet based on the probability value in Rebroadcast.c file.



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-1	Broadcast-0	NODE-1	NODE-5
6	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-1
7	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-3
8	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-4
9	1	0	CBR	App1_CBR	NODE-2	Broadcast-0	NODE-2	NODE-5
10	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-1
11	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-2
12	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-3
13	1	0	CBR	App1_CBR	NODE-4	Broadcast-0	NODE-4	NODE-5
14	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-1
15	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-2
16	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-4
17	1	0	CBR	App1_CBR	NODE-3	Broadcast-0	NODE-3	NODE-5
54	1	0	CBR	App1_CBR	NODE-5	Broadcast-0	NODE-5	NODE-1
55	1	0	CBR	App1_CBR	NODE-5	Broadcast-0	NODE-5	NODE-2
56	1	0	CBR	App1_CBR	NODE-5	Broadcast-0	NODE-5	NODE-3
57	1	0	CBR	App1_CBR	NODE-5	Broadcast-0	NODE-5	NODE-4

Figure 7: NetSim Packet Trace

Note: 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.

Appendix: NetSim source code modifications

Changes to handle_app_out() function, in APP_OUT.c file, within Application project

/*The code checks if the destination is ‘0’ i.e., Broadcast packet, then it adds the packet to rebroadcast list*/

```
        //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->sourceList[0] == pstruEventDetails->nDeviceId)
#endif // REBROADCAST
                set_app_end_and_generate_next_packet(pstruPacket, otherDetails, destCount, dest);

        //Add the dummy payload to packet
        fn_NetSim_Add_DummyPayload(pstruPacket, appInfo);
#ifdef REBROADCAST
        if (appInfo->sourceList[0] == pstruEventDetails->nDeviceId)
#endif // REBROADCAST
                appmetrics_src_add(appInfo, pstruPacket);

        appout_send_packet(s, appInfo, pstruPacket, nDeviceId);
#ifdef REBROADCAST
        if (!dest[0])
                rebroadcast_add_packet_to_info(pstruPacket, pstruEventDetails->dEventTime);
#endif // REBROADCAST

}
```

Changes to int fn_NetSim_Application_Run()function in the APPLICATION_IN_EVENT, in Application.c file, within 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.*/

```
#ifdef REBROADCAST
if (pstruappinfo->sourceList[0] == pstruPacket->nSourceId)
#endif // REBROADCAST
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),pstruEve
ntDetails->dEventTime);
}
if(pstruappinfo->nAppType == TRAFFIC_EMULATION && pstruPacket->szPayload)
{
fn_NetSim_Dispatch_to_emulator(pstruPacket);
}
```

```
if (pstruappinfo->nAppType == TRAFFIC_BSM_APP)
{
process_saej2735_packet(pstruPacket);
}
#ifdef REBROADCAST
UINT destCount;
NETSIM_ID* dest = get_dest_from_packet(pstruPacket, &destCount);
if (!dest[0])
{
rebroadcast_packet(pstruPacket,
pstruEventDetails->nDeviceId,
pstruEventDetails->dEventTime);
}
else
{
#elif
//Delete the packet
fn_NetSim_Packet_FreePacket(pstruPacket);
#endif // REBROADCAST
#ifdef REBROADCAST
}
```

Added the following function declarations in Application.h file, within Application project

```
int fn_NetSim_Add_DummyPayload(NetSim_PACKET* packet, ptrAPPLICATION_INFO);

//Encryption
char xor_encrypt(char ch, long key);
int aes256(char* str, int* len);
int des(char* buf, int* len);

//Application event handler
void handle_app_out();
#define REBROADCAST
void rebroadcast_add_packet_to_info(NetSim_PACKET* packet, double time);
void rebroadcast_packet(NetSim_PACKET* packet, NETSIM_ID devId, double time);
 #endif
```