SinkHole Attack in LEACH

Software Recommended: NetSim Standard v11.0 64 bit, Visual Studio 2015/2017

Project Download Link:

https://github.com/NetSim-TETCOS/SinkHole_Attack_in_LEACH_v11.0/archive/master.zip

Low-energy adaptive clustering hierarchy ("LEACH"):

LEACH is a MAC protocol which is integrated with clustering and a simple routing protocol in wireless sensor networks (WSNs). The goal of LEACH is to lower the energy consumption required to create and maintain clusters in order to improve the life time of a wireless sensor network.

This Cross Layer Protocol is implemented in NetSim in MAC layer which involves ZigBee Protocol and Network layer which involves DSR protocol. The clustering of sensors happens in the Network layer and the Cluster head election involves interacting with the MAC layer to obtain the remaining power of the sensors.

A **LEACH.c** file is added to the DSR project.

1. For this implementation of LEACH, the number of Clusters is fixed as 4 and all the 4 clusters are equal. If the user wants to change it, then he/she must also change the static routing for the Cluster Heads and the ClusterElement array accordingly.
2. To make 4 equal clusters the number of sensors must be 4,16,36,64,100. Depending on the number of sensors, the ClusterElements array must be defined. Here, it has been defined and commented for 4,16,36,64,100 sensors. Uncomment the one you want to use.

The file contains the following funcitons:

fn_NetSim_LEACH_CheckDestination()

This function is used to check whether the current device is the destination (i.e) the sinknode or not. Else the packet will be forwarded to the next hop.

fn_NetSim_LEACH_GetNextHop()

This function is used to identify the next hop in cases where the current device is either a sensor within the cluster or the cluster head. Static routes are defined in this function. It returns the Device id of the next hop.

fn_NetSim_LEACH_AssignClusterHead ()

This function is used to dynamically assign cluster heads within a cluster based on the residual energy. The sensor with higher remaining power in comparison to other sensors within the same cluster will be elected as the cluster head.

fn_NetSim_LEACH_IdentifyCluster()

This function is used to determine the cluster to which a sensor belongs. It returns the cluster id of the cluster.

Sinkhole attack on LEACH:

In this project we are implementing sinkhole attack on top of the LEACH project where a malicious node advertises false battery information to become a cluster head. Upon being elected as a cluster

head, it attracts network traffic from all its cluster members and destroys the packets without forwarding them to the sink/base station.

Implementation:

A file **malicious.c** is added to the DSR project which contains the following functions:

- **fn_NetSim_DSR_MaliciousNode( )**

  This function is used to identify whether a current device is malicious or not in-order to establish malicious behavior.

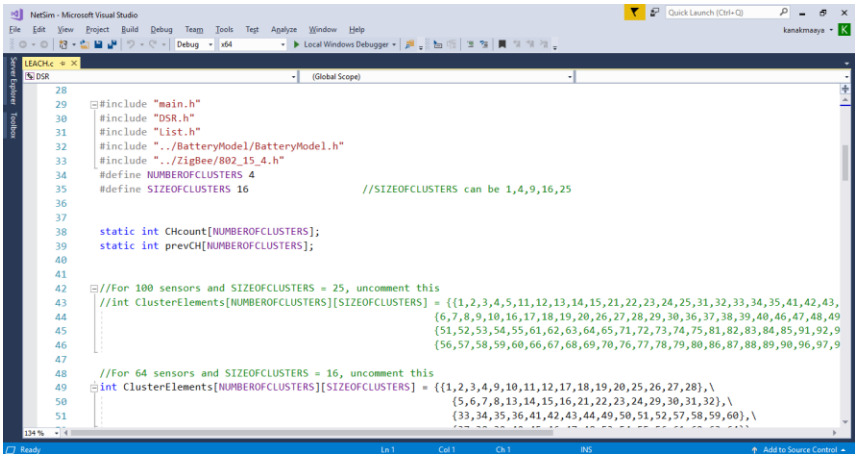
- **fn_NetSim_DSR_MaliciousProcessSourceRouteOption()**

  This function is used to drop the received packets if the device is malicious, instead of forwarding the packet to the next hop.

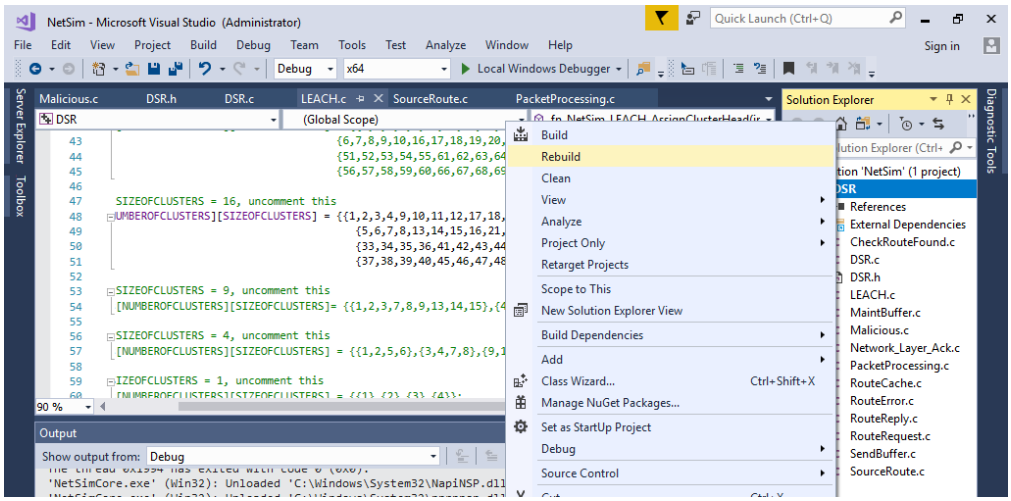
You can set any device as malicious and you can have more than one malicious node in a scenario. Device id's of malicious nodes can be set inside the **fn_NetSim_DSR_MaliciousNode()** function.

Steps:

1. Open the Code folder and double click on the NetSim.sln file to open the project in visual studio.

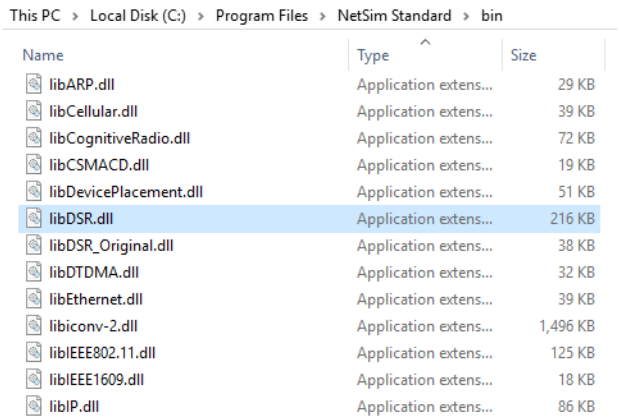


2. Right click on the **DSR** Project and select rebuild.

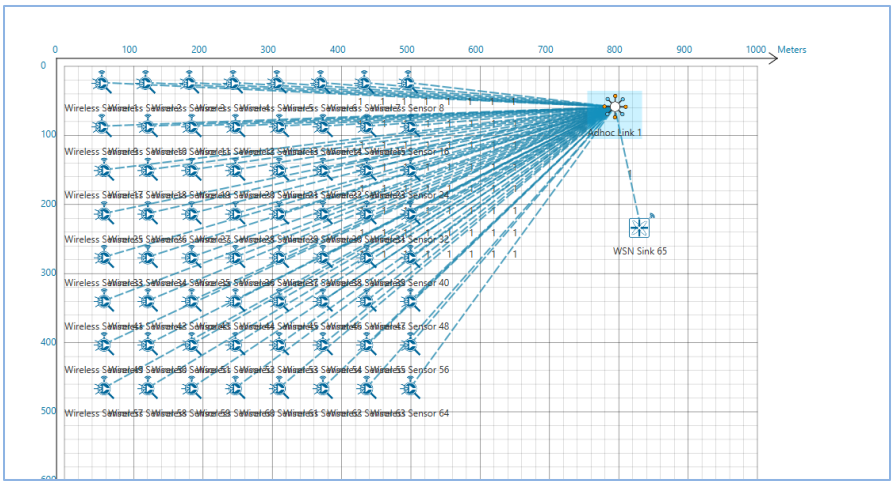


3. Now copy the **libDSR.dll** from the **DLL** folder.

4. Replace the DLL in NetSim bin folder in the NetSim installation directory (Eg: **C:\Program Files\NetSim Standard\bin**) after renaming the original **libDSR.dll** file.



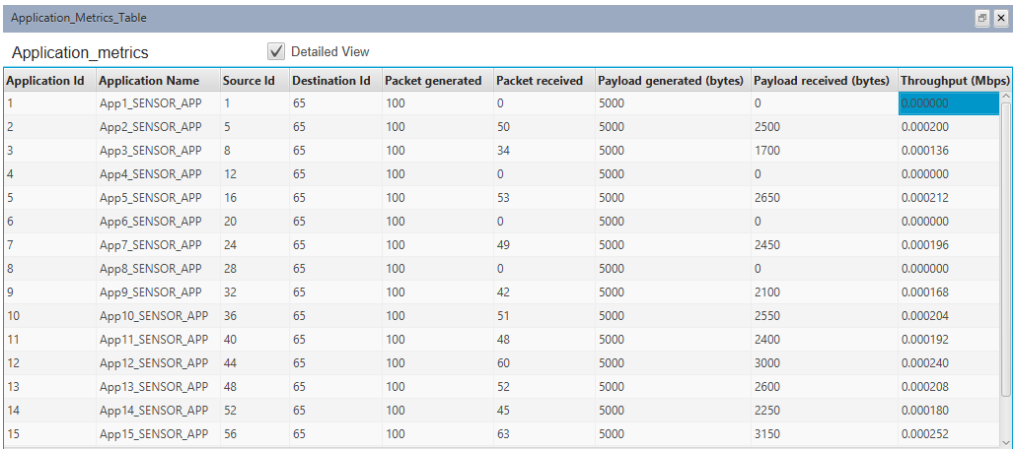
5. Now create a Network Scenario in NetSim WSN Network as per the Number of clusters and size of clusters that are set in the LEACH code. By default the code runs for a scenario with 64 sensors uniformly placed, with the SINKNODE placed as per the screenshot below.



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

6. Run the simulation
7. View the packet animation. You will note that the sensors directly start transmitting packets without route establishment since the routes are statically defined in LEACH. You will also note that the cluster heads keep changing dynamically in Clusters 2, 3 and 4. In cluster1, the cluster members transmits packets to malicious node (device id 4) since it advertises false battery information to become a cluster head.
8. This can be observed in Packet trace by applying filters to Source_ID column by selecting only Sensor-1, 12, 20, 28. You will be able to see that the receiver id is sensor-4 throughout the simulation. All the nodes in the Cluster1 are sending data packets to malicious node (Sensor-4) since it is the Cluster Head.

9. Now undo filter in Source_Id column and apply filter to transmitter_Id column by selecting only Sensor-4. You will be able to see that no data packets are forwarded by the malicious node. The malicious node acting as a sinkhole, drops the packets without forwarding it.
10. This will have a direct impact on the Application Throughput which can be observed in the Application Metrics table present in NetSim Simulation Results window. The throughput for applications 1, 4, 6 and 8 are zero since the source ids belongs to cluster1 having malicious node (device id 4).


Application_Metrics_Table

Application_metrics

Detailed View

Application Id	Application Name	Source Id	Destination Id	Packet generated	Packet received	Payload generated (bytes)	Payload received (bytes)	Throughput (Mbps)
1	App1_SENSOR_APP	1	65	100	0	5000	0	0.000000
2	App2_SENSOR_APP	5	65	100	50	5000	2500	0.000200
3	App3_SENSOR_APP	8	65	100	34	5000	1700	0.000136
4	App4_SENSOR_APP	12	65	100	0	5000	0	0.000000
5	App5_SENSOR_APP	16	65	100	53	5000	2650	0.000212
6	App6_SENSOR_APP	20	65	100	0	5000	0	0.000000
7	App7_SENSOR_APP	24	65	100	49	5000	2450	0.000196
8	App8_SENSOR_APP	28	65	100	0	5000	0	0.000000
9	App9_SENSOR_APP	32	65	100	42	5000	2100	0.000168
10	App10_SENSOR_APP	36	65	100	51	5000	2550	0.000204
11	App11_SENSOR_APP	40	65	100	48	5000	2400	0.000192
12	App12_SENSOR_APP	44	65	100	60	5000	3000	0.000240
13	App13_SENSOR_APP	48	65	100	52	5000	2600	0.000208
14	App14_SENSOR_APP	52	65	100	45	5000	2250	0.000180
15	App15_SENSOR_APP	56	65	100	63	5000	3150	0.000252