SinkHole Attack in LEACH

Software Recommended: NetSim standard v13.3, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Sinkhole-Attack-in-LEACH-WSN-v13.3/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-upnetsim-file-exchange-projects

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 (WSN). The goal of LEACH is to lower the energy consumption required to create and maintain clusters to improve the lifetime of a wireless sensor network.

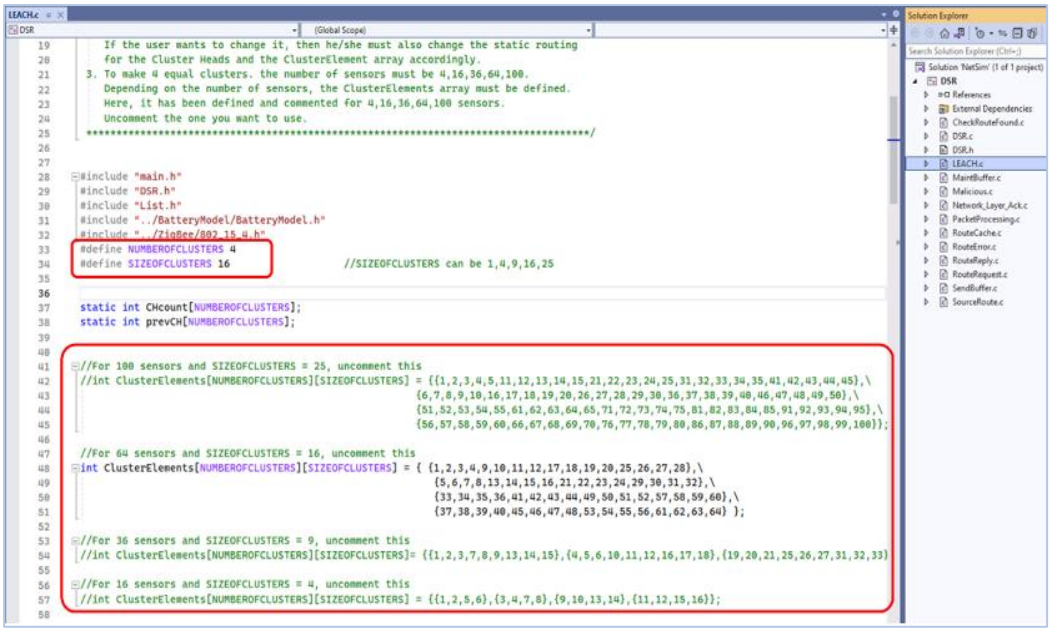
This Cross-Layer Protocol is implemented in NetSim in the MAC layer which involves ZigBee Protocol and the Network layer which involves DSR protocol. The clustering of sensors happens in the Network Layer and the cluster head selection involves interacting with the MAC layer to obtain the remaining power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

Figure 1: Network setup for Sinkhole attack in LEACH



```
   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.
3. 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.
*****************************************************************************************/

#include "main.h"
#include "DSR.h"
#include "List.h"
#include "../BatteryModel/BatteryModel.h"
#include "../ZigBee/802_15_4.h"
#define NUMBEROFCLUSTERS 4
#define SIZEOFCLUSTERS 16              //SIZEOFCLUSTERS can be 1,4,9,16,25

static int CHcount[NUMBEROFCLUSTERS];
static int prevCH[NUMBEROFCLUSTERS];

//For 100 sensors and SIZEOFCLUSTERS = 25, uncomment this
//int ClusterElements[NUMBEROFCLUSTERS][SIZEOFCLUSTERS] = {{1,2,3,4,5,11,12,13,14,15,21,22,23,24,25,31,32,33,34,35,41,42,43,44,45},\
                                                           {6,7,8,9,10,16,17,18,19,20,26,27,28,29,30,36,37,38,39,40,46,47,48,49,50},\
                                                           {51,52,53,54,55,61,62,63,64,65,71,72,73,74,75,81,82,83,84,85,91,92,93,94,95},\
                                                           {56,57,58,59,60,66,67,68,69,70,76,77,78,79,80,86,87,88,89,90,96,97,98,99,100}};

//For 64 sensors and SIZEOFCLUSTERS = 16, uncomment this
int ClusterElements[NUMBEROFCLUSTERS][SIZEOFCLUSTERS] = { {1,2,3,4,9,10,11,12,17,18,19,20,25,26,27,28},\
                                                          {5,6,7,8,13,14,15,16,21,22,23,24,29,30,31,32},\
                                                          {33,34,35,36,41,42,43,44,49,50,51,52,57,58,59,60},\
                                                          {37,38,39,40,45,46,47,48,53,54,55,56,61,62,63,64} };

//For 36 sensors and SIZEOFCLUSTERS = 9, uncomment this
//int ClusterElements[NUMBEROFCLUSTERS][SIZEOFCLUSTERS]= {{1,2,3,7,8,9,13,14,15},{4,5,6,10,11,12,16,17,18},{19,20,21,25,26,27,31,32,33}

//For 16 sensors and SIZEOFCLUSTERS = 4, uncomment this
//int ClusterElements[NUMBEROFCLUSTERS][SIZEOFCLUSTERS] = {{1,2,5,6},{3,4,7,8},{9,10,13,14},{11,12,15,16}};
```

Figure 2: Leach.c file in source code

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 on for 4,16,36,64,100 sensors. Uncomment the one you want to use.

The File contains the following functions:

fn_NetSim_LEACH_CheckDestination() // to check whether the current device is the destination or not.

fn_NetSim_LEACH_GetNextHop() // For getting the next hop device id.

fn_NetSim_LEACH_AssignClusterHead() // For electing the Cluster head based on Remaining energy.

fn_NetSim_LEACH_IdentifyCluster() // To determine the cluster to which a sensor belongs.

Sinkhole Attack on LEACH:

In this project, we are implementing a 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 IDs of malicious nodes can be set inside the **fn_NetSim_DSR_MaliciousNode()** function.

Steps:

1. To open-source code open **NetSim > Your Work > Source Code > Open code**
2. In **LEACH.c** present inside DSR Project, the number of clusters and Size of clusters are decided, and in Malicious. c the malicious node is set You can set them to any value, and the node

Note: By default, Malicious Node is set to 4 and NUMBER OF CLUSTERS – 4, SIZE OF CLUSTERS – 16, If changed Rebuild the Solution as shown below:

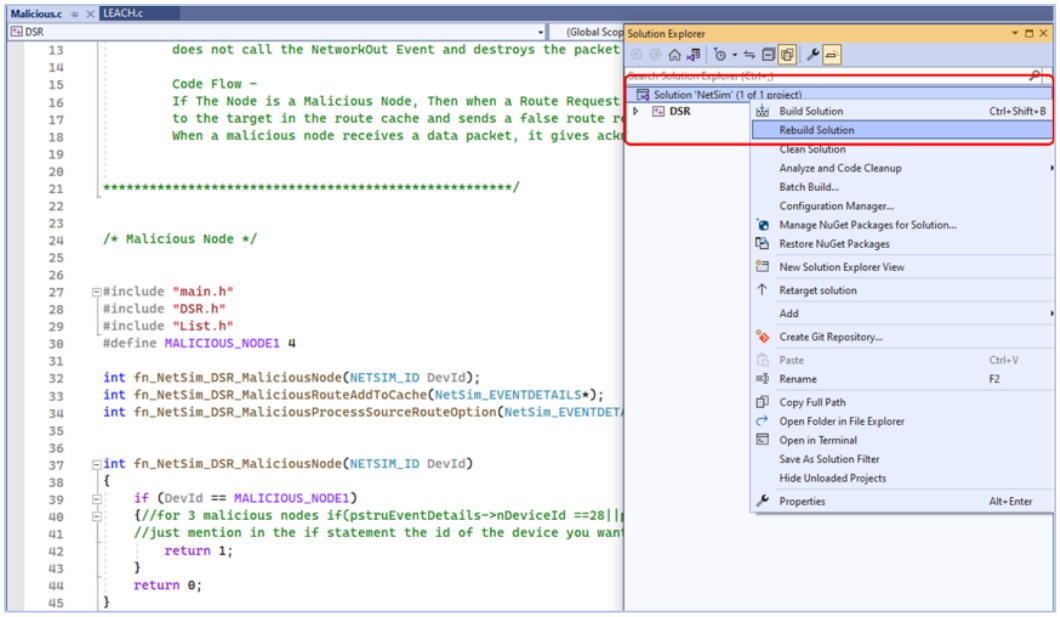
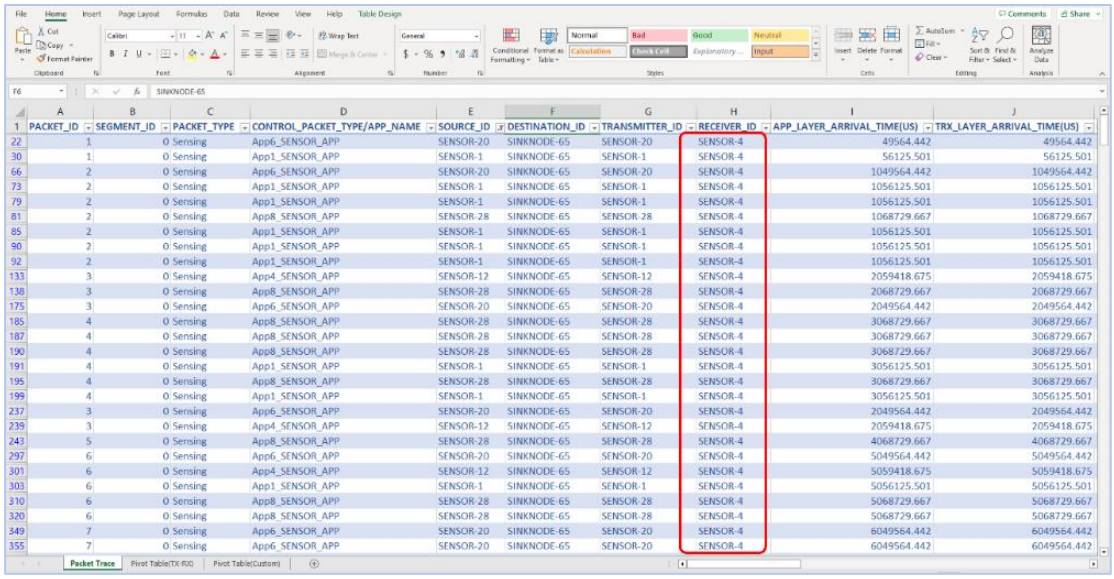


Figure 3: solution Rebuild in source code

3. Rebuild the solution by right clicking on solution > Rebuild solution.
4. The **Workspace_Sinkhole_in_LEACH** comes with an inbuilt network scenario example, Open the scenario.
5. Run the simulation.

Results and discussion:

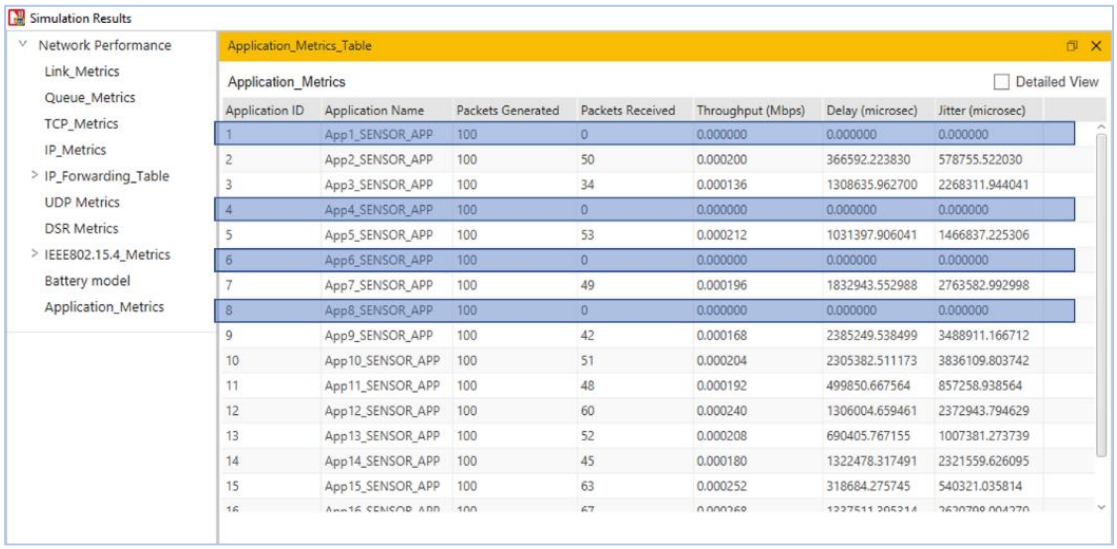
- View the packet animation. You will note that the sensors directly start transmitting packets without route establishment since the routes are statically defined in LEACH. You will also note that the cluster heads keep changing dynamically in Clusters 2, 3, and 4. In cluster1, the cluster members transmit packets to the malicious node (device id 4) since it advertises false battery information to become a cluster head.
- This can be observed in the Packet trace by applying filters to the 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 Cluster1 are sending data packets to the malicious node (Sensor-4) since it is the Cluster Head



PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME(US)	TRX_LAYER_ARRIVAL_TIME(US)
22	1	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	49564.442	49564.442
30	1	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	56125.501	56125.501
66	2	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	1049564.442	1049564.442
73	2	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	1056125.501	1056125.501
79	2	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	1056125.501	1056125.501
81	2	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	1068729.667	1068729.667
85	2	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	1056125.501	1056125.501
90	2	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	1056125.501	1056125.501
92	2	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	1056125.501	1056125.501
133	3	0 Sensing	App4_SENSOR_APP	SENSOR-12	SINKNODE-65	SENSOR-12	SENSOR-4	2059418.675	2059418.675
138	3	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	2068729.667	2068729.667
175	3	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	2049564.442	2049564.442
185	4	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	3068729.667	3068729.667
187	4	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	3068729.667	3068729.667
190	4	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	3068729.667	3068729.667
191	4	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	3056125.501	3056125.501
195	4	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	3068729.667	3068729.667
199	4	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	3056125.501	3056125.501
237	3	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	2049564.442	2049564.442
239	3	0 Sensing	App4_SENSOR_APP	SENSOR-12	SINKNODE-65	SENSOR-12	SENSOR-4	2059418.675	2059418.675
243	5	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	4068729.667	4068729.667
297	6	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	5049564.442	5049564.442
301	6	0 Sensing	App4_SENSOR_APP	SENSOR-12	SINKNODE-65	SENSOR-12	SENSOR-4	5059418.675	5059418.675
303	6	0 Sensing	App1_SENSOR_APP	SENSOR-1	SINKNODE-65	SENSOR-1	SENSOR-4	5056125.501	5056125.501
310	6	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	5068729.667	5068729.667
320	6	0 Sensing	App8_SENSOR_APP	SENSOR-28	SINKNODE-65	SENSOR-28	SENSOR-4	5068729.667	5068729.667
349	7	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	6049564.442	6049564.442
355	7	0 Sensing	App6_SENSOR_APP	SENSOR-20	SINKNODE-65	SENSOR-20	SENSOR-4	6049564.442	6049564.442

Figure 4: Packet trace file referring to malicious node reception of transmitted packets

- This will have a direct impact on the Application Throughput which can be observed in the Application Metrics table present in the NetSim Simulation Results window. The throughput for applications 1, 4, 6, and 8 is zero throughputs since the source ids belong to cluster1 having a malicious node (device id 4)



Application ID	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (microsec)	Jitter (microsec)
1	App1_SENSOR_APP	100	0	0.000000	0.000000	0.000000
2	App2_SENSOR_APP	100	50	0.000200	366592.223830	578755.522030
3	App3_SENSOR_APP	100	34	0.000136	1308635.962700	2268311.944041
4	App4_SENSOR_APP	100	0	0.000000	0.000000	0.000000
5	App5_SENSOR_APP	100	53	0.000212	1031397.906041	1466837.225306
6	App6_SENSOR_APP	100	0	0.000000	0.000000	0.000000
7	App7_SENSOR_APP	100	49	0.000196	1832943.552988	2763582.992998
8	App8_SENSOR_APP	100	0	0.000000	0.000000	0.000000
9	App9_SENSOR_APP	100	42	0.000168	2385249.538499	3488911.166712
10	App10_SENSOR_APP	100	51	0.000204	2305382.511173	3836109.803742
11	App11_SENSOR_APP	100	48	0.000192	499850.667564	857258.938564
12	App12_SENSOR_APP	100	60	0.000240	1306004.659461	2372943.794629
13	App13_SENSOR_APP	100	52	0.000208	690405.767155	1007381.273739
14	App14_SENSOR_APP	100	45	0.000180	1322478.317491	2321559.626095
15	App15_SENSOR_APP	100	63	0.000252	318684.275745	540321.035814

Figure 5: Simulation results window