SinkHole Attack in LEACH

Software Recommended: NetSim standard v14.0, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Sinkhole-in-LEACH-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-settingupnetsim-file-exchange-projects

Introduction :

Leach(Low – Energy Adaptive Clustering hierarchy) 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 election involves interacting with the MAC layer to obtain the remaining power of the sensors.

Real-world context:

In real-world smart cities, different monitoring sensors are deployed to monitor various aspects of the city, such as traffic and air quality consumption. In this case, we are considering the LEACH protocol, which is used to maintain clusters of sensor nodes. In our example, we consider air, traffic, noise, and energy as clusters.

Air Quality Monitoring Cluster:

Monitors air quality parameters such as PM2.5, PM10, and NO2 to assess air pollution levels and guide air quality management strategies.

Traffic Monitoring Cluster:

Collects traffic data on vehicle volume, speed, and congestion to improve traffic flow, optimize transportation networks, and reduce traffic-related emissions.

Noise Monitoring Cluster:

Measures noise levels in various environments to identify and address noise pollution sources, protect public health, and promote noise-sensitive land use planning.

Energy Monitoring Cluster:

Gathers data on energy consumption patterns and identifies energy-saving opportunities to promote energy efficiency, reduce energy costs, and contribute to sustainable energy management.

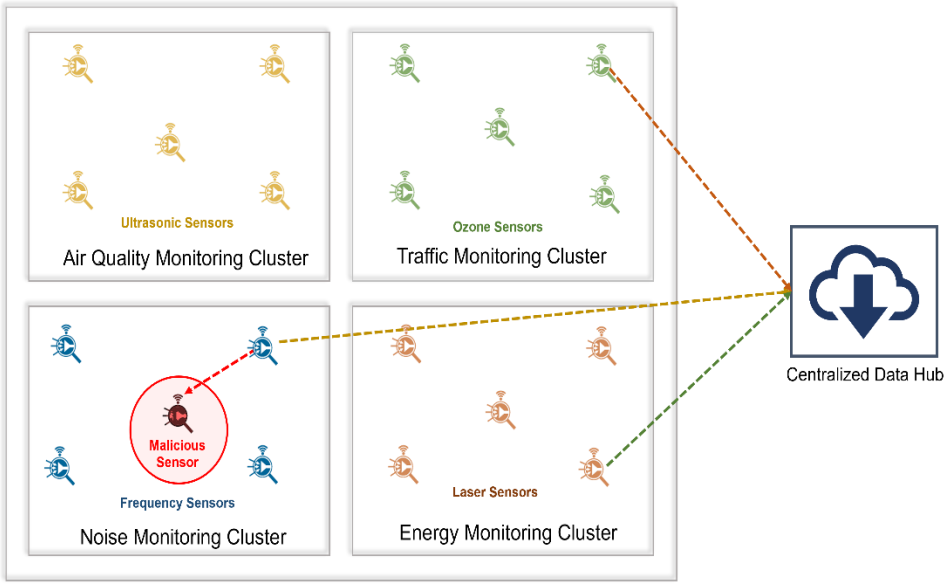


Figure 1:Real world smart city monitoring clusters

Sinkhole Attack in Leach Overview:

1. In a smart city, there are four clusters of sensor nodes such as noise, air, traffic, and energy. Each cluster has a cluster head, which is responsible for collecting data from the other nodes in the cluster and forwarding it to the sink node.

2. A malicious sensor node enters the noise monitoring cluster and advertises false battery information to make itself appear to be a more attractive cluster head than the legitimate cluster head. As a result, the malicious sensor node is elected as the cluster head.

3. Once elected as the cluster head, the malicious sensor node starts intercepting and diverting all data packets from the other nodes in the cluster. It does this by responding to route requests and diverting data packets through itself.

4. As a result, all data packets from the noise monitoring cluster are routed through the malicious sensor node. The malicious sensor node then intentionally discards all incoming data packets, thereby preventing any data from reaching the sink node.

Implementation:

A **LEACH.c** file is added to 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.

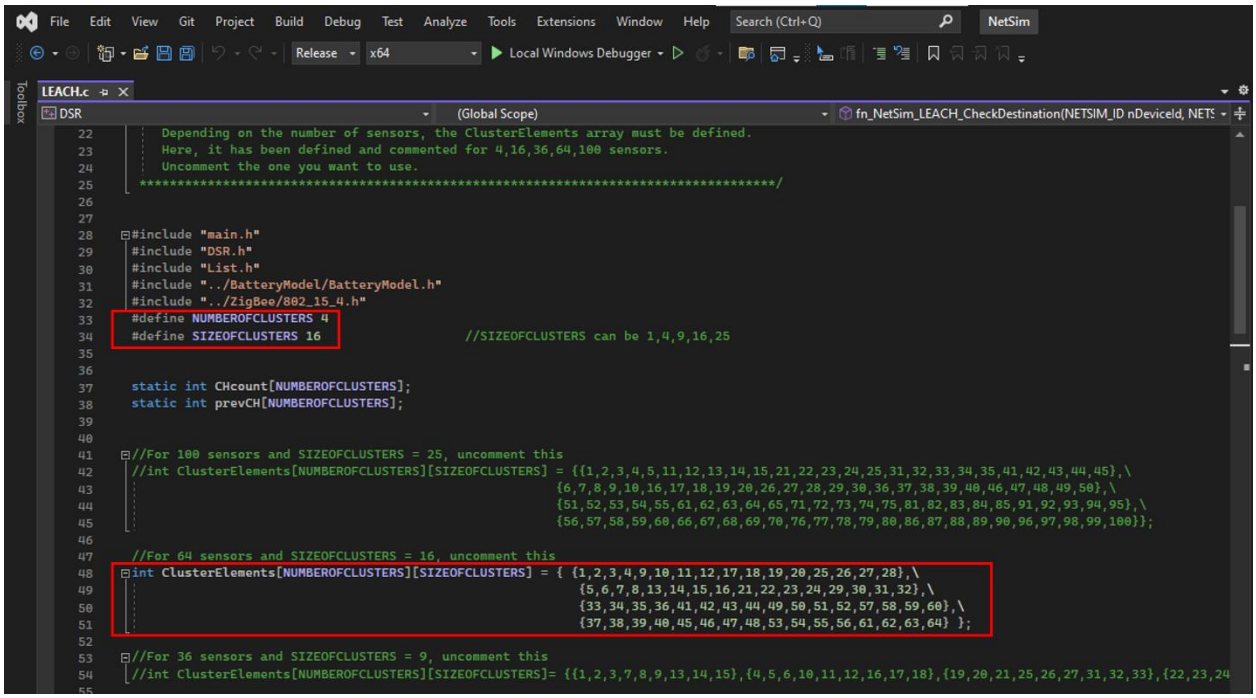


Figure 2:Leach.c file in source code

2. To make 4 equal clusters the number of sensors must be 4,16,36,64,100. Depending on the number of sensors, the Cluster Elements 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.

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.

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.

Note: By default, Malicious Node is set to 22 and NUMBER OF CLUSTERS – 4, SIZE OF CLUSTERS – 16, If changed Rebuild the Solution as shown below:

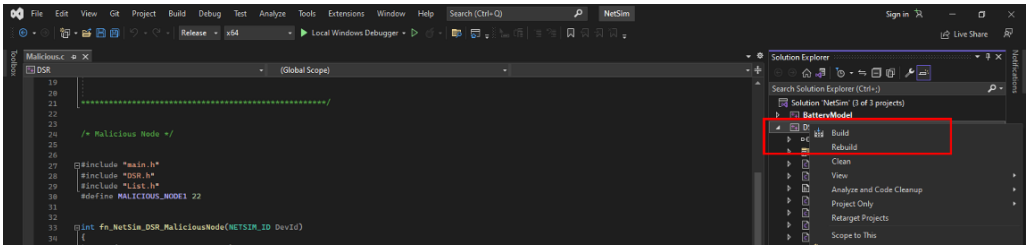


Figure 3:Solution Rebuild in source code

Right Click on the DSR project and Rebuild.

Example:

1. The **Sinkhole-in-LEACH-Workspace** comes with a sample network configuration that is already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **Sinkhole-in-LEACH-Example**. from the list of experiments.
2. The network scenario consists of 64 sensors uniformly placed along with the SINKNODE as shown below.

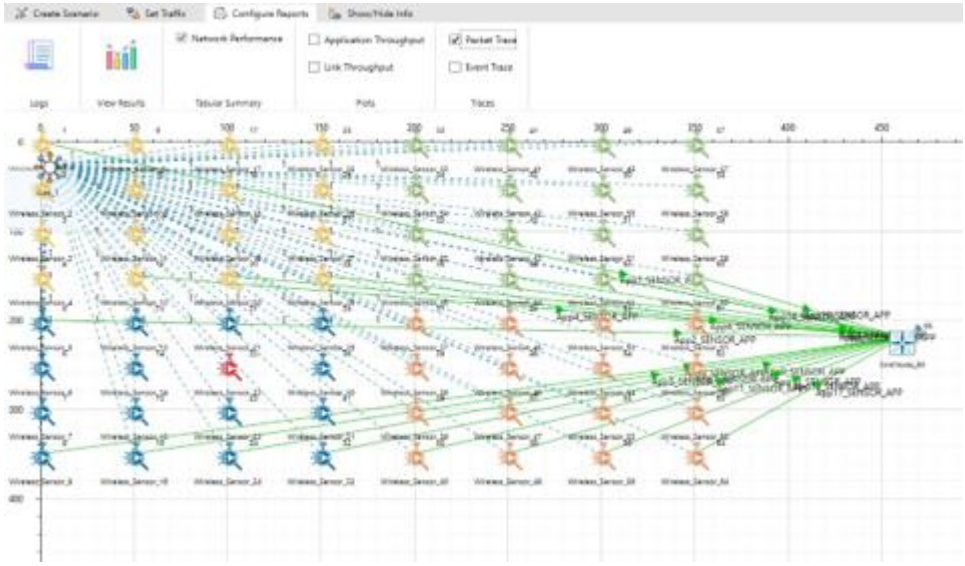


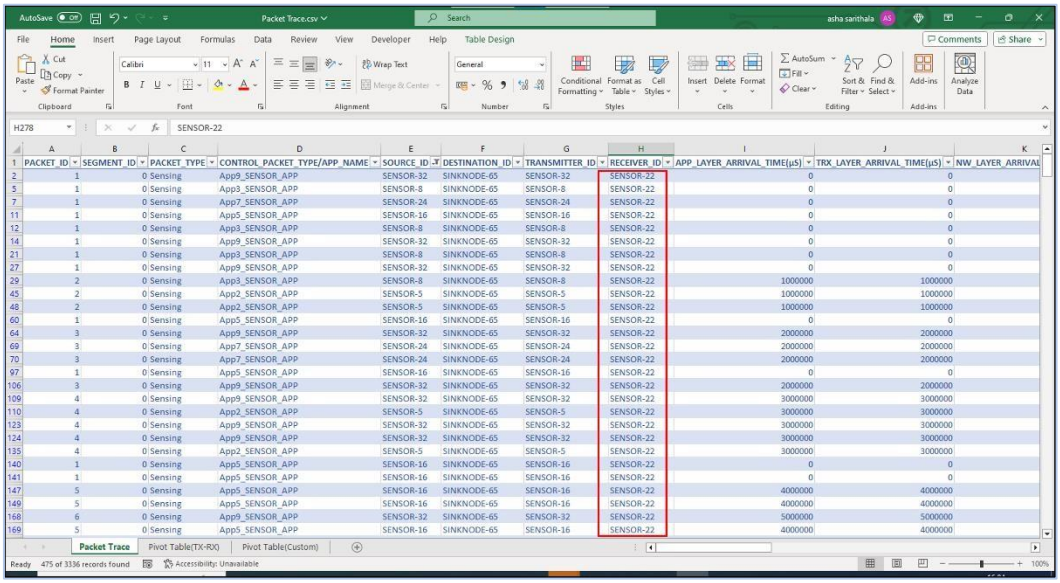
Figure 4:Network setup for Sinkhole attack in LEACH

3. Run the simulation for 100 seconds.

Results and discussion:

- View the packet trace .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 1, 3, and 4. In cluster 2, the cluster members transmit packets to the malicious node (device id 22) since it advertises false battery information to become a cluster head.

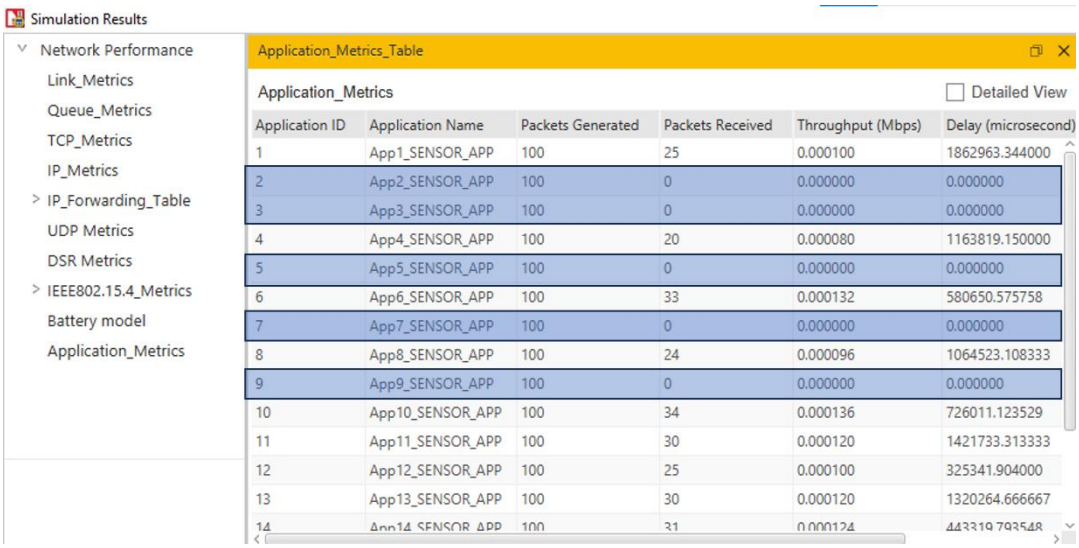
- This can be observed in the Packet trace by applying filters to the Source_ID column by selecting only Sensor-5, 8,16,24,32. You will be able to see that the receiver id is sensor-22 throughout the simulation. All the nodes in Cluster2 are sending data packets to the malicious node (Sensor-22) since 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(µS)	TRX_LAYER_ARRIVAL_TIME(µS)	NW_LAYER_ARRIVAL
2	1	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	0	0	
5	1	0 Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	0	0	
7	1	0 Sensing	App7_SENSOR_APP	SENSOR-24	SINKNODE-65	SENSOR-24	SENSOR-22	0	0	
11	1	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	0	
12	1	0 Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	0	0	
14	1	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	0	0	
21	1	0 Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	0	0	
27	1	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	0	0	
29	2	0 Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	1000000	1000000	
45	2	0 Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	1000000	1000000	
48	2	0 Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	1000000	1000000	
60	1	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	0	
64	3	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	2000000	2000000	
69	3	0 Sensing	App7_SENSOR_APP	SENSOR-24	SINKNODE-65	SENSOR-24	SENSOR-22	2000000	2000000	
70	3	0 Sensing	App7_SENSOR_APP	SENSOR-24	SINKNODE-65	SENSOR-24	SENSOR-22	2000000	2000000	
97	1	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	0	
106	3	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	2000000	2000000	
109	4	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	3000000	3000000	
110	4	0 Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	3000000	3000000	
123	4	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	3000000	3000000	
124	4	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	3000000	3000000	
135	4	0 Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	3000000	3000000	
140	1	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	0	
141	1	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	0	
147	5	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	4000000	4000000	
149	5	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	4000000	4000000	
168	6	0 Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	5000000	5000000	
169	5	0 Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	4000000	4000000	

Figure 5: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 2, 3, 5,7 and 9 is zero since the source ids belong to cluster2 having a malicious node (device id 22).



Application ID	Application Name	Packets Generated	Packets Received	Throughput (Mbps)	Delay (microsecond)
1	App1_SENSOR_APP	100	25	0.000100	1862963.344000
2	App2_SENSOR_APP	100	0	0.000000	0.000000
3	App3_SENSOR_APP	100	0	0.000000	0.000000
4	App4_SENSOR_APP	100	20	0.000080	1163819.150000
5	App5_SENSOR_APP	100	0	0.000000	0.000000
6	App6_SENSOR_APP	100	33	0.000132	580650.575758
7	App7_SENSOR_APP	100	0	0.000000	0.000000
8	App8_SENSOR_APP	100	24	0.000096	1064523.108333
9	App9_SENSOR_APP	100	0	0.000000	0.000000
10	App10_SENSOR_APP	100	34	0.000136	726011.123529
11	App11_SENSOR_APP	100	30	0.000120	1421733.313333
12	App12_SENSOR_APP	100	25	0.000100	325341.904000
13	App13_SENSOR_APP	100	30	0.000120	1320264.666667
14	App14_SENSOR_APP	100	31	0.000124	443319.793548

Figure 6: Simulation results window