PEGASIS in WSN

Software: NetSim Standard v14.3, Visual Studio 2022, MATLAB 2019 or above

Project Download Link:

https://github.com/NetSim-TETCOS/PEGASIS-WSN-v14.3/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-up-netsim-file-exchange-projects

1 Introduction

In Wireless Sensor Networks energy efficiency plays a crucial role as the sensors are generally battery powered. Hierarchical routing protocols can be used to overcome this constraint.

PEGASIS – Power Efficient Gathering in Sensor Information Systems is one such hierarchical routing protocol that follows a chain-based approach and a greedy algorithm. The sensor nodes organize themselves to form a chain. If any node dies in between then the chain is reconstructed to bypass the dead node. A leader or a cluster head node is assigned, and it takes care of transmitting data to the base station/ sink node. The main goal of PEGASIS is to receive and transmit data to and from the neighbor and take turns being the cluster head for transmission to the Sink Node.
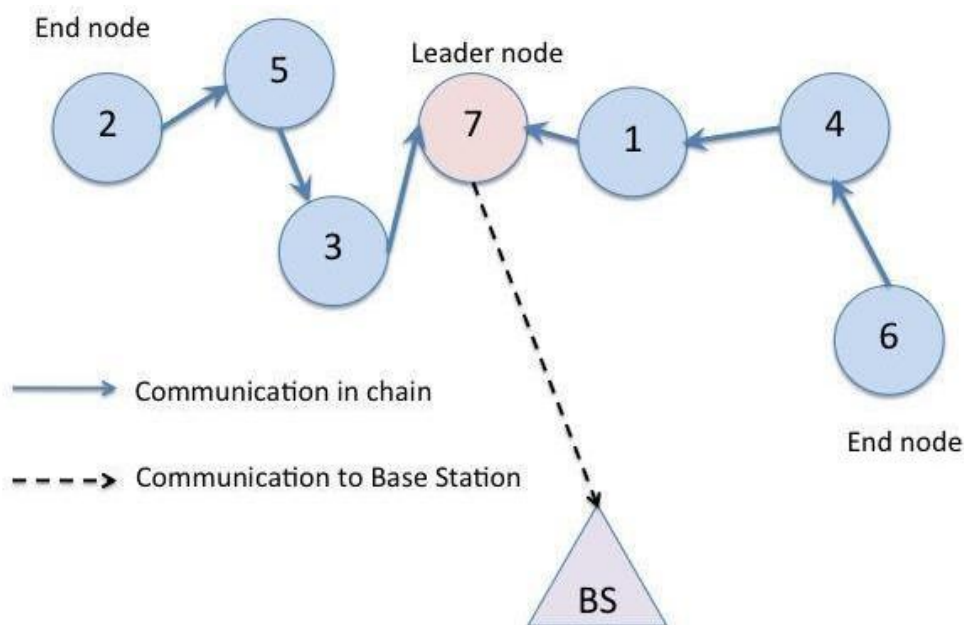

Figure 1:Network organization and data forwarding mechanisms within PEGASIS

2 PEGASIS in NetSim with MATLAB Interfacing

PEGASIS algorithm is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and PEGASIS

algorithm that is implemented in MATLAB is used to dynamically form a chain between the nodes and to elect one of them as a head node.

From MATLAB the order of devices in the chain and the head node id is retrieved to perform routing in NetSim.

All the above steps are performed during each transmission and can be defined as per the implementation. Each time a node dies, the chain will be reconstructed. Also, nodes take turn to become the head node in each iteration.

The codes required for the mathematical calculations done in MATLAB are written to a **PEGASIS.m** file and this file available in MATLAB folder under bin_x64 of **PEGASIS_in_WSN_Workspace** Workspace.
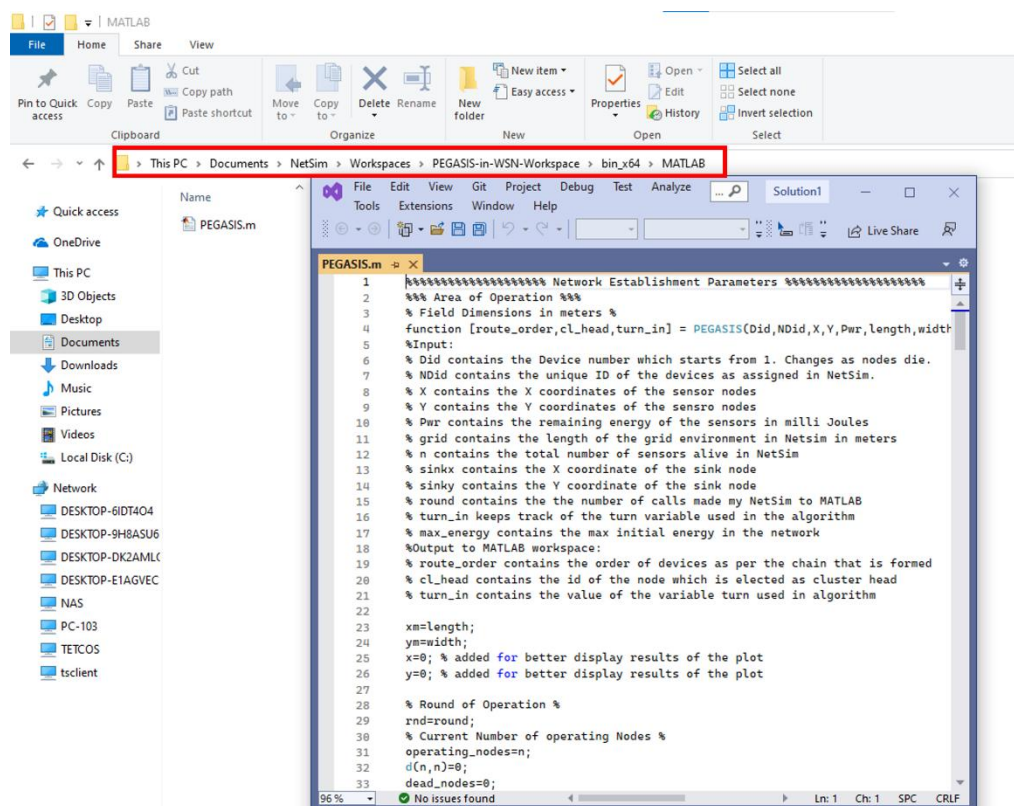

Figure 2: MATLAB file path

3 Implementation

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

- **fn_NetSim_PEGASIS_CheckDestination();** //This function is used to determine whether the current device is the destination of a packet or an intermediate node.

- **fn_NetSim_PEGASIS_GetNextHop();** //This function handles routing in the sensor network by determining the next hop device based on the chain that is formed as part of PEGASIS protocol.

- **fn_NetSim_PEGASIS_run();** //This function makes a call to MATLAB interfacing function and passes the inputs from NetSim to MATLAB and also retrieves the computed parameters from MATLAB workspace for further calculations in NetSim.
- **fn_NetSim_PEGASIS_form_clusters();** //This function updates the information obtained from MATLAB to identify the head node and the neighbouring nodes in the PEGASIS chain.
- **fn_NetSim_PEGASIS_next_closest_node();** //This function is used to identify the neighbouring node and next hop for routing packets.
- **fn_NetSim_PEGASIS_Init();** //This function initializes the parameters specific to PEGASIS algorithm and identifies the number of sensors that are alive throughout the simulation.

Static Routing: Static Routing is defined in such a way that the sensors send packets to neighbouring node in the PEGASIS chain which is closest to the head node. Once packet arrives at the head node it is forwarded to the destination or the sinknode.

4 Steps to simulate

1. Add the following MATLAB install directory path in the Environment PATH variable
   <MATLAB_INSTALL_DIRECTORY>\bin\win64
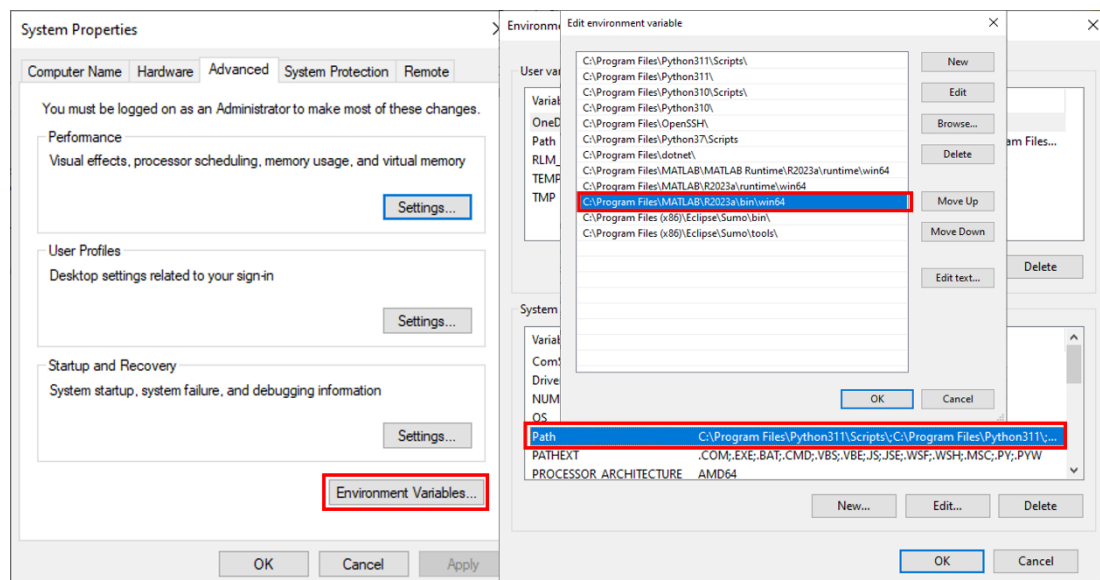   For eg: C:\Program Files\MATLAB\R2020b\bin\win64

Figure 3: Environment variable PATH

Note:
- To run this code 64- bit version of MATLAB must be installed in your system.
- If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

2. Open the Command prompt as admin and execute the command "matlab -regserver". This will register MATLAB as a COM automation server and is required for NetSim to start MATLAB automation server during runtime.

5 Example

1. Run NetSim in administrative mode.

2. The **PEGASIS-in-WSN-Workspace** comes with a sample network configuration that is already saved. To open this example, go to Your work on the home screen of NetSim and click on the **PEGASIS_in-WSN-Example** from the list of experiments.

3. The network scenario consists of 100 sensors deployed randomly in a 50x50m grid environment with a sink node placed in the center.
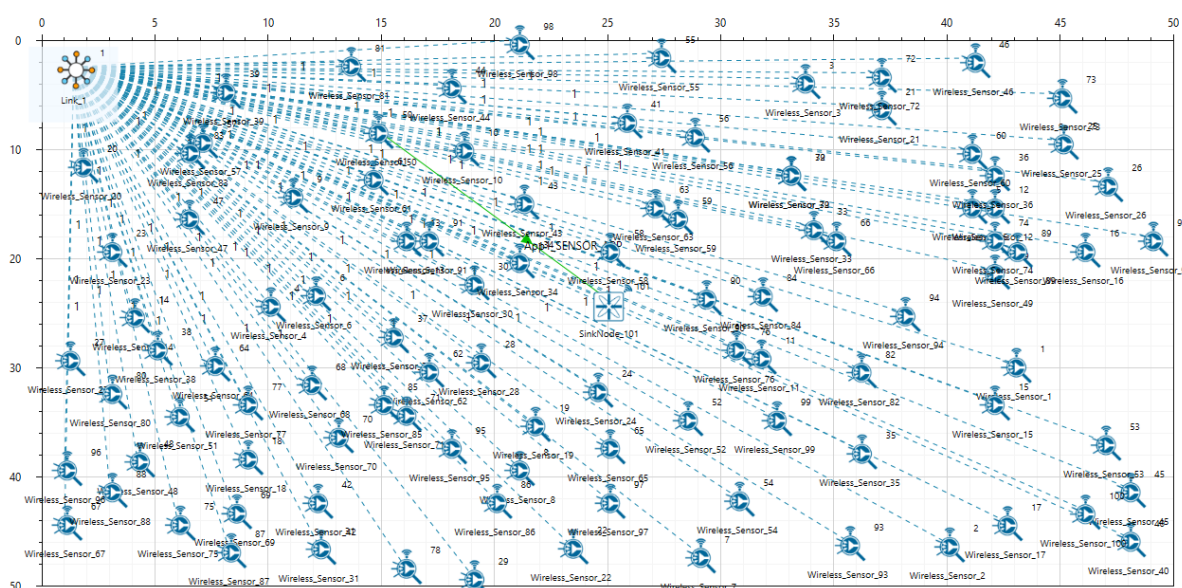

Figure 4: Network scenario consists of 100 sensors

4. Run the Simulation for 100 Seconds.

6 Results and discussion

You will observe that as the simulation starts, NetSim automatically initializes MATLAB and the plots will be displayed as shown below:
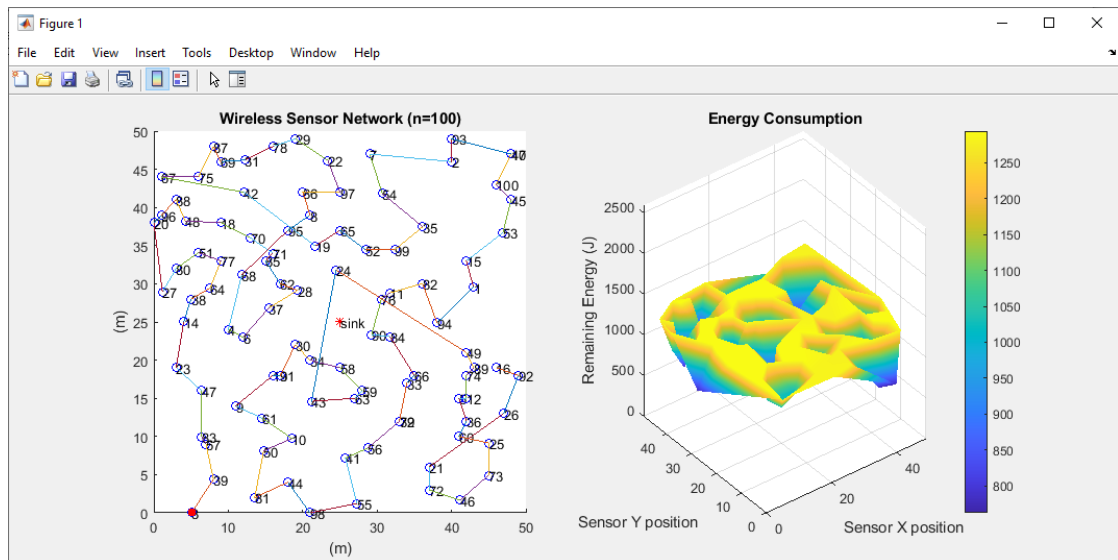
Figure 5: NetSim automatically initializes MATLAB and the plots
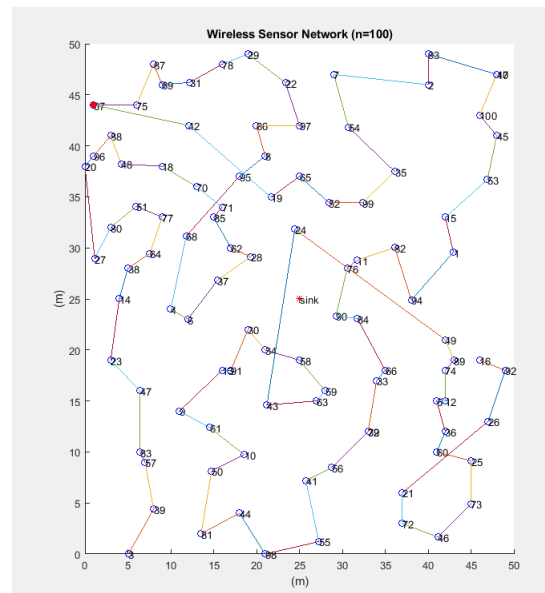
Plot 1:

Figure 6: Nodes that are part of the network in NetSim Shown in MATLAB

The above plot contains the nodes that are part of the network in NetSim. All sensor nodes are initially alive. N in the plot indicates the total sensor nodes that are currently alive. Sink node is highlighted in the center and the node marked in red acts as the head node for the current transmission.

All nodes are connected to form a chain which is used to forward packets to the destination or the sink node. This chain may get reconstructed whenever one or more node runs out of energy and dies. The value of 'n' keeps getting updated during the simulation.

Note:

The head node election is based on selecting the farthest operational node from the Sink node, as the initial cluster head.
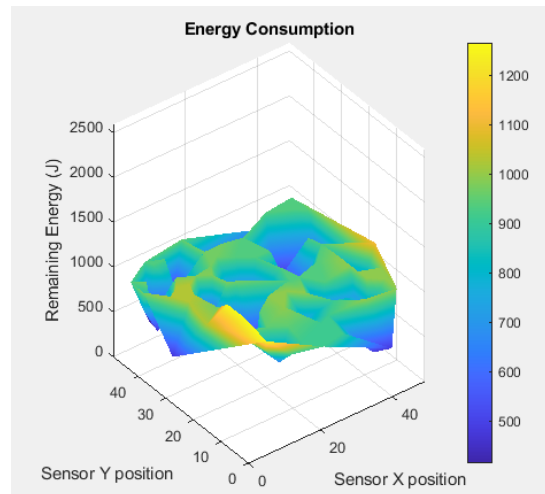
Plot 2:

Figure 7: Energy level in the sensor network

The above plot denotes the energy level in the sensor network. Sensors have different levels of energy remaining in the battery right from the simulation start. The z axis represents the remaining energy while the sensors are placed on the x, y plane. As simulation progresses this plot gets updated dynamically based on energy consumed by the sensors in the network.