Dynamic Clustering in WSN

Software: NetSim Standard v13.2 (64 bit), Visual Studio 2022, MATLAB R2016 or higher

Project Download Link:
https://github.com/NetSim-TETCOS/Dynamic_Clustering_in_WSN_v13.2/archive/refs/heads/main.zip

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

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

Clustering in WSN

Clustering is the process of partitioning a group of sensors into small numbers of clusters. In environments where the sensors are mobile clusters cannot be static. Like cluster heads in each cluster are elected dynamically, the members in each cluster also need to be dynamically identified. Therefore, the size of each cluster is not fixed and can vary depending on the position of the sensors.

Dynamic Clustering helps in efficiently grouping sensors into clusters dynamically. There is no fixed cluster size, and the sensors are divided into the required number of clusters with members of each cluster calculated dynamically.

Clustering using k-means algorithm

kmeans(X,k) partitions the points in the n-by-p data matrix X into k clusters. This iterative partitioning minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. Rows of X correspond to points, columns correspond to variables. kmeans returns an n-by-1 vector IDX containing the cluster indices of each point. By default, kmeans uses squared Euclidean distances. When X is a vector, kmeans treats it as an n-by-1 data matrix, regardless of its orientation.

The sensor positions and number of clusters,

X - a matrix containing the x, y coordinates of the sensors in the scenario.
k- the number of clusters. are passed to k-means algorithm. [IDX,C] = kmeans(X,k).
IDX – Contains the cluster id's of each sensor (i.e) the cluster to which the sensor belongs.
C – Centroids of each cluster.

Clustering using Fuzzy C-Means Algorithm

Fuzzy c-means (FCM) is a data clustering technique in which a dataset is grouped into n clusters with every data point in the dataset belonging to every cluster to a certain degree. For example, a certain data point that lies close to the center of a cluster will have a high degree of belonging or membership to that cluster and another data point that lies far away from the center of a cluster will have a low degree of belonging or membership to that cluster.

Cluster head election based on distance from Centroid

After grouping the sensors into different clusters, the cluster heads are determined based on the distance between the sensor and the centroid of the cluster to which it belongs.

The sensor which is closer to the centroid will be elected as the cluster head. Here the position

values (i.e., value of x-coordinate and y-coordinate) of each sensor are passing from NetSim to MATLAB as a sole parameter.

Cluster head election based on distance and power

After grouping the sensors into different clusters, the cluster heads are determined based on the distance between the sensor and the remaining power of each sensor. After that, the sensors are assigned to the respective cluster.

The sensor which is closer to the centroid and has more power than other sensors will be elected as the cluster head. Here the position values (i.e., the value of the x-coordinate and y-coordinate) of each sensor and power are passed from NetSim to MATLAB as a sole parameter.

Dynamic Clustering in NetSim with MATLAB Interfacing

Dynamic Clustering is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and the k-means algorithm that is implemented in MATLAB is used to dynamically perform clustering of the sensors into n number of clusters.

In addition to clustering, we also determine the cluster head of each cluster mathematically in MATLAB. The distance of each sensor from the centroid of the cluster to which it belongs is calculated. Then the sensor which has the least distance is elected as the cluster head.

From MATLAB we get the cluster id of each sensor, the cluster heads of each cluster, and the size of each cluster.
All the above steps are performed periodically which can be defined as per the implementation. Each time the cluster members and the cluster heads are determined based on the current position and they are not fixed.

The codes required for the mathematical calculations done in MATLAB are written to a clustering.m file and this file is available in the MATLAB folder under bin_x64 of Dynamic_Clustering_Workspace_v13.2

The **clustering.m** file can be run in four different modes cluster head election.

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

- fn_NetSim_dynamic_clustering_CheckDestination()//This function is used to determine whether the current device is the destination.
- fn_NetSim_dynamic_clustering_GetNextHop()//This function statically defines the routes within the cluster and from cluster to sinknode. It returns the next hop based on the static routing that is defined.
- fn_NetSim_dynamic_clustering_IdentifyCluster()//This function returns the cluster id of the cluster to which a sensor belongs.
- fn_NetSim_dynamic_clustering_run()//This function makes a call to MATLAB interfacing function and passes the inputs from NetSim (i.e) the sensor coordinates, number of clusters and the sensor count.
- fn_netsim_dynamic_form_clusters()//This function assigns each sensor to its respective clusters based on the cluster id's obtained from MATLAB.
- fn_netsim_assign_cluster_heads()//This function assigns the cluster heads for each cluster based on the cluster head id's obtained from MATLAB.
- fn_NetSim_Dynamic_Clustering_Init()//This function initializes all parameter values.

Static Routing

Static Routing is defined in such a way that the sensors in the cluster send the packets to the cluster head. The cluster head then directly sends the packets to the destination (sinknode).
If the current sensor is the source device and if it is not a cluster head, then its next hop is its cluster head.

If the current sensor is the source device and if it is a cluster head, then its next hop is the destination (i.e.) the sinknode.

If the current sensor is not the source, then the packet is sent to the destination (i.e.) the sinknode.

NOTE: To run this code 64- bit version of MATLAB must be installed in your system.

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\R2020a\bin\win64

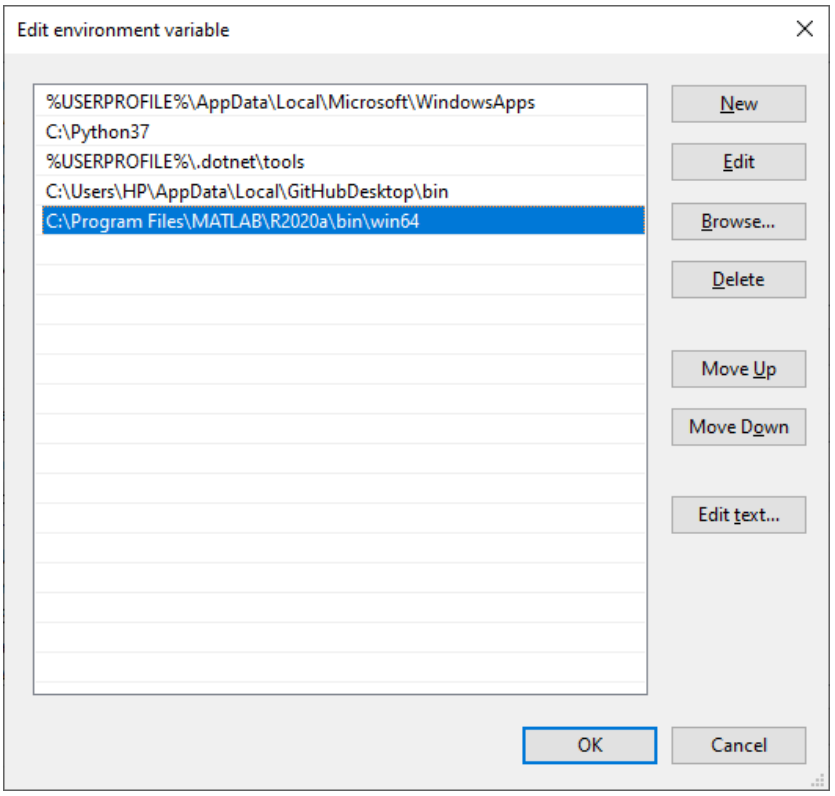

Figure 1: Environment variable PATH

Note: 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 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.

3. Open the Source codes in Visual Studio by going to Your work-> Source code and Clicking on Open code button in NetSim Home Screen window.

4. Under the DSR project in the solution explorer you will be able to see that and **Dynamic_Clustering.c** files which contain source codes related to interactions with MATLAB and handling clustering in NetSim respectively.

5. Right click on the solution in the solution explorer and select Rebuild.

Example

1. Run NetSim as administrative mode.

2. The Dynamic_Clustering_Workspace_v13.2 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 Dynamic_Clustering_Example from the list of experiments.

3. The saved network scenario consists of 64 sensors uniformly distributed in the grid environment along with a sink node forming a Wireless Sensor Network. Traffic is configured from each sensor node to the Sink Node.

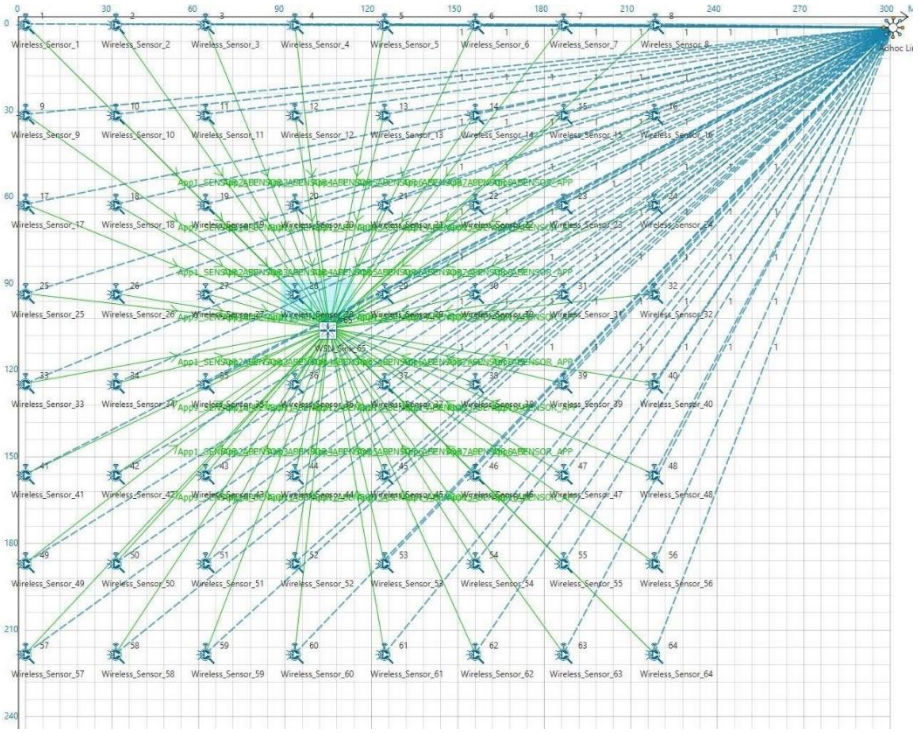

Figure 2: Network Scenario in this project

4. Run simulation and press any key to continue. NetSim simulation console will show the following message in the console “Waiting for NetSim MATLAB Interface to connect...”. NetSim will automatically open MatlabInterface.exe console window

5. It will open the MatlabInterface.exe console window. You will observe that as the simulation starts in NetSim, MATLAB gets initialized and the graph associated with energy consumption in the sensor network is plotted during runtime.

Results and discussion

A total of 64 sensors are placed evenly on the grid environment and each sensor is set to have equal initial energy.

At the end of the simulation, NetSim provides Battery Model Metrics which provide detailed information related to energy consumption in each sensor node with respect to transmission, reception, idle mode, sleep mode, etc. as shown below:

Battery model_Table

Battery model

Device Name	Initial energy(mJ)	Consumed energy(mJ)	Remaining Energy(mJ)	Transmitting energy(mJ)	Receiving energy(mJ)	Idle energy(mJ)	Sleep energy(mJ)
WIRELESS_SENSOR_1	6480.000000	557.662753	5922.337247	19.364115	0.000000	538.298638	0.000000
WIRELESS_SENSOR_2	6480.000000	556.639152	5923.360848	18.304387	0.000000	538.334765	0.000000
WIRELESS_SENSOR_3	6480.000000	557.662753	5922.337247	19.364115	0.000000	538.298638	0.000000
WIRELESS_SENSOR_4	6480.000000	560.268282	5919.731718	22.061604	0.000000	538.206678	0.000000
WIRELESS_SENSOR_5	6480.000000	559.430790	5920.569210	21.194554	0.000000	538.236237	0.000000
WIRELESS_SENSOR_6	6480.000000	557.848862	5922.151138	19.556793	0.000000	538.292070	0.000000
WIRELESS_SENSOR_7	6480.000000	557.011371	5922.988629	18.689743	0.000000	538.321628	0.000000
WIRELESS_SENSOR_8	6480.000000	557.104425	5922.895575	18.786082	0.000000	538.318344	0.000000
WIRELESS_SENSOR_9	6480.000000	556.546098	5923.453902	18.208048	0.000000	538.338050	0.000000
WIRELESS_SENSOR_10	6480.000000	561.914510	5918.085490	21.001876	2.755953	538.156682	0.000000
WIRELESS_SENSOR_11	6480.000000	557.848862	5922.151138	19.556793	0.000000	538.292070	0.000000
WIRELESS_SENSOR_12	6480.000000	560.809337	5919.190663	19.364115	3.248087	538.197135	0.000000
WIRELESS_SENSOR_13	6480.000000	559.430790	5920.569210	21.194554	0.000000	538.236237	0.000000
WIRELESS_SENSOR_14	6480.000000	557.011371	5922.988629	18.689743	0.000000	538.321628	0.000000
WIRELESS_SENSOR_15	6480.000000	783.464984	5696.535016	70.905416	181.696020	530.863548	0.000000

Figure 3: NetSim provides Battery Model Metrics

This information can also be obtained at different points of simulation time either to log or to send to other external tools. The battery information and the position coordinates are passed to MATLAB periodically for clustering (number of clusters is set to 4), cluster head election and to obtain energy consumption plots.

Cluster head election using distance alone as a parameter

Running simulations with Clustering Method set to 1 and 2 in the clustering.m file will provide energy consumption plots for k-means and fuzzy c-means algorithms respectively as shown below:

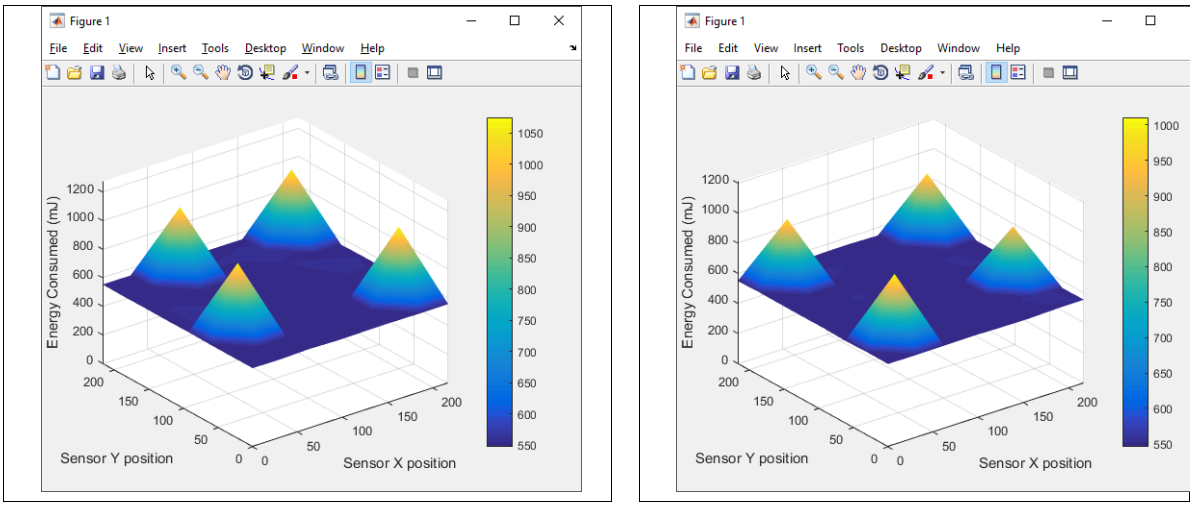

Figure 4: Energy consumption plots for k means and fuzzy c-means algorithms using method 1 and 2

As it is seen from the plot, there are 4 peaks in the plot corresponding to higher energy consumption in the nodes in the center of the cluster, as they always become the cluster heads. This is because the distance is used as a parameter for electing the cluster heads.

Cluster head election using distance and remaining energy as parameters

Running simulations with the Clustering Method set to 3 and 4 in the **clustering.m** file will provide energy consumption plots for k-means and fuzzy c-means algorithms respectively as shown below:

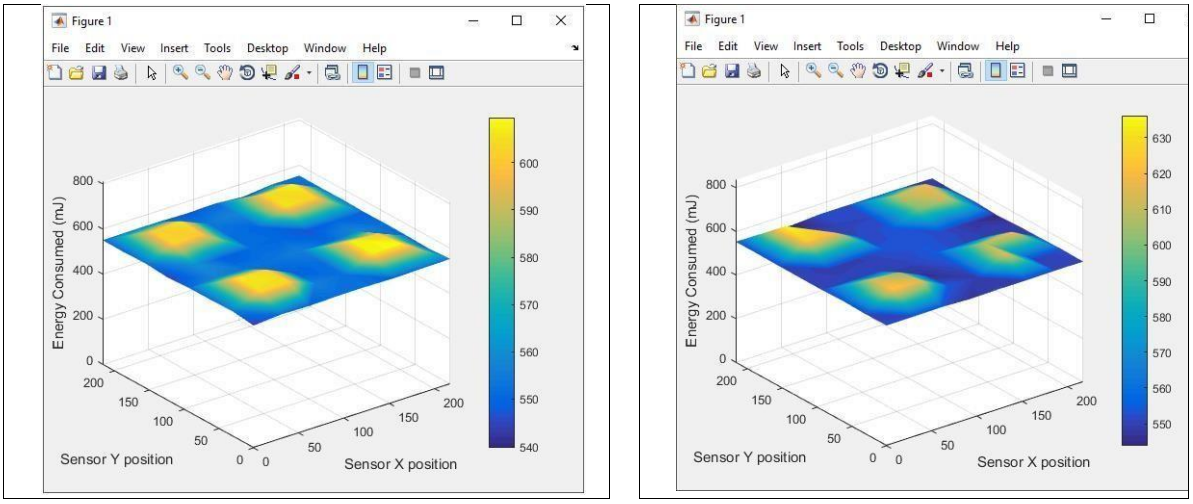

Figure 5: Energy consumption plots for k means and fuzzy c-means algorithms using method 3 and 4

In the initial phase the plot resembles the previous one. However, as time passes, it can be observed that the power is consumed by all the sensors at approximately the same rate.

There are no sharp peaks in this plot unlike the previous one because modified K-means consider the power level of each sensor and thus sensors other than those in the center of the cluster will also get a chance to be elected as the cluster head in their respective cluster.