Dynamic Clustering in WSN

Clustering is the process partitioning a group of sensor 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

**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.

**Dynamic Clustering in NetSim with MATLAB Interfacing:**

Dynamic Clustering is implemented in NetSim by Interfacing is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and 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 return the cluster id’s of each sensor, 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 are not fixed.

The codes required for the mathematical calculations done in MATLAB are written to a clustering.m file and added to the DSR project.

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.

**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 the next hop is its cluster head.

If the current sensor is the source device and if it is a cluster head the 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 32- bit version of MATLAB must be installed in your system.

**Steps to run Dynamic Clustering Code in NetSim:**

1. Open the Simulation - Dynamic\_Clustering folder and double click on the NetSim.sln file to open the project in visual studio.
2. Under the DSR project in the solution explorer double click on the MATLAB\_Interface.c file.
3. Inside the fn\_netsim\_matlab\_run() change the path to your systems current path where the clustering.m file is present.

sprintf(buf,"cd 'C:\\Users\\TETCOS\\Desktop\\kmeans\\Simulation - Dynamic\_Clustering\\DSR'");

This file is present in the DSR folder inside the Simulation – Dynamic\_Clustering directory.

1. Right click on the DSR project and select PROPERTIES in the solution explorer to open the project properties. Once this window has opened, make the following changes:

a. Under C/C++ General, add the following directory to the field ADDITIONAL INCLUDE DIRECTORIES:

<Path where MATLAB is installed>\extern\include

b. Under C/C++ Precompiled Headers, select "Not Using Precompiled Headers".

c. Under Linker General, add the directory to the field ADDITIONAL LIBRARY DIRECTORIES:

<Path where MATLAB is installed>\extern\lib\win32\microsoft

d. Under Configuration Properties ->Debugging

Add the following Target path in the *Environment*:

<Path where MATLAB is installed>\bin\win32

e. Under Linker Input, add the following names to the field marked ADDITIONAL DEPENDENCIES:

libmx.lib libmat.lib libeng.lib by separating them with a semicolon.

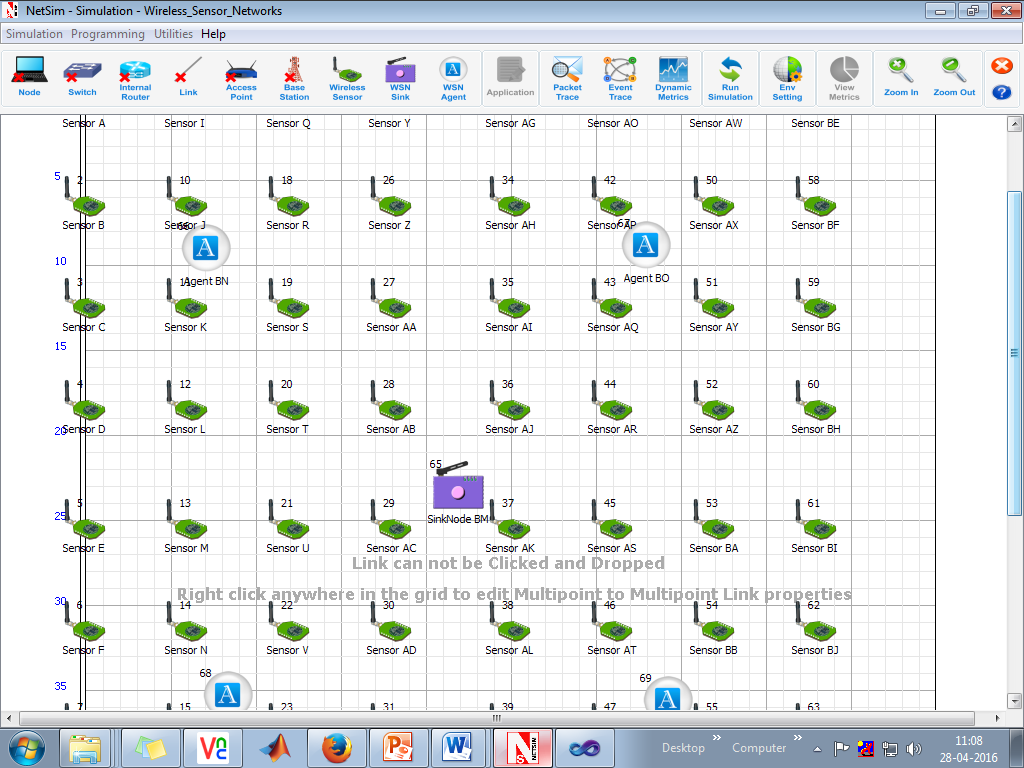
Click on Apply and then on ok.

f. Make sure that the following directory is in the PATH(Environment variable)

<Path where MATLAB is installed>\bin\win32

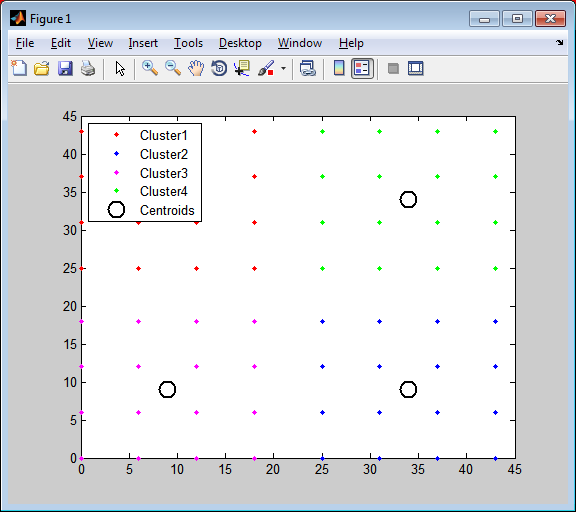
**Note:** If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 32-bit application, the directory in the MATLAB 32-bit installation must be the first one on the PATH).

1. Right click on the ZigBee project in the solution explorer and select Rebuild.
2. Right click on the DSR project in the solution explorer and select Rebuild.
3. Copy the newly built libDSR.dll and libZigBee.dll from the DLL folder inside the Simulation – Dynamic\_clustering Directory.
4. Replace the DLL’s in the bin folder inside NetSim Installation Directory, after renaming the original libDSR.dll and libZigBee.dll.
5. Create a Network Scenario in WSN (for eg. 64 sensors) and set mobility to the sensors by setting the velocity to let say 10 m/s.

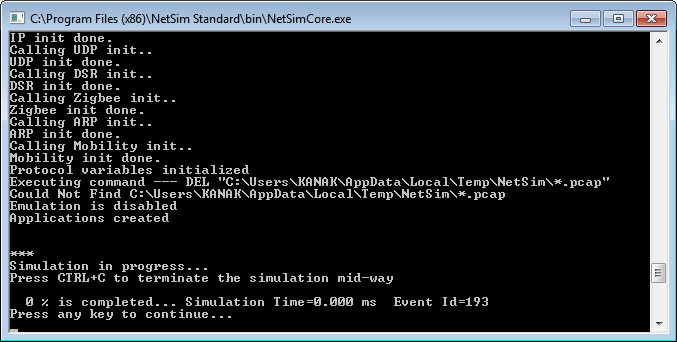


1. Run the Scenario. You will observe that as the scenario starts and MATLAB plots the graph for the cluster that is formed currently.

At time 0 seconds

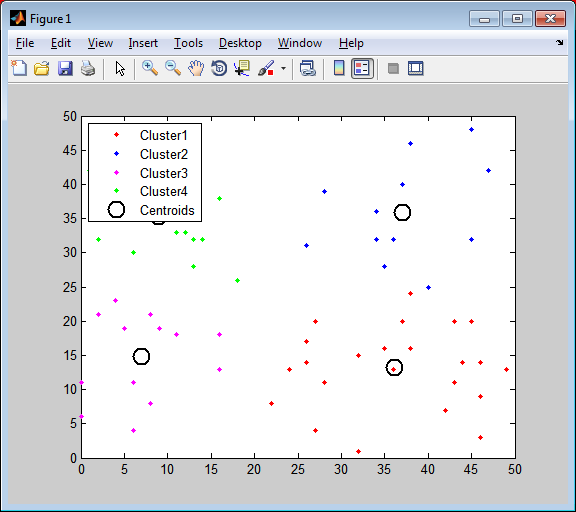


1. NetSim Simulation window waits for user interrupt. Press any key to continue.

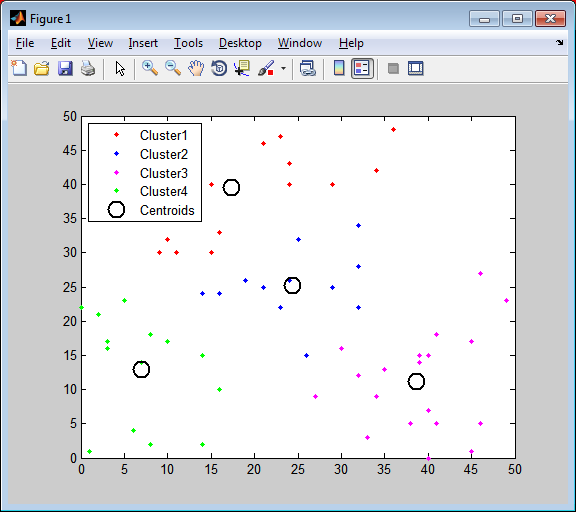


1. After every ten seconds this process continues and graphs are plotted. The device positions vary as we have set mobility to the sensors.

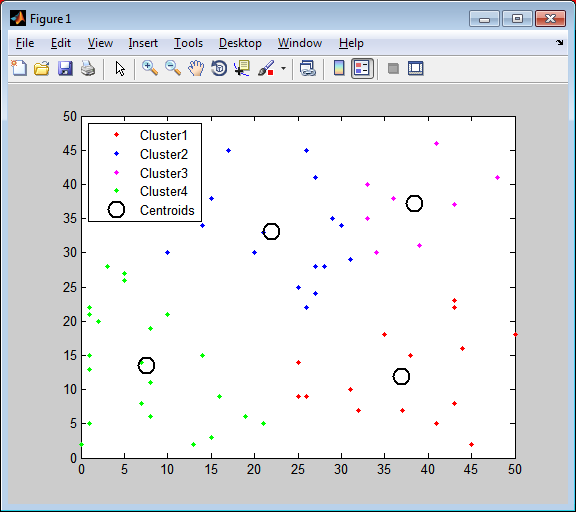
At time 10 seconds:



At time 20 seconds:



At time 50 seconds:



1. View packet animation. You will notice that sensors in a cluster are sending the packets to the cluster head. The cluster heads directly send the packets to the sinknode.
2. Modifications can be done to the code for further analysis:
3. In the Dynamic\_Clustering.c file you can set the total number of clusters to be formed.

#define NUMBEROFCLUSTERS 4

This is set to 4 by default.

1. In the DSR.h file the time interval for clustering can be set.
2. #define CLUSTER\_INTERVAL 10\*SECOND

This is set to 10 seconds by default. This means for every 10 seconds new clusters are formed and accordingly cluster heads are elected.