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

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. Afterthat the sensors are assigned in respective cluster.

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

Dynamic Clustering in NetSim with MATLAB Interfacing:

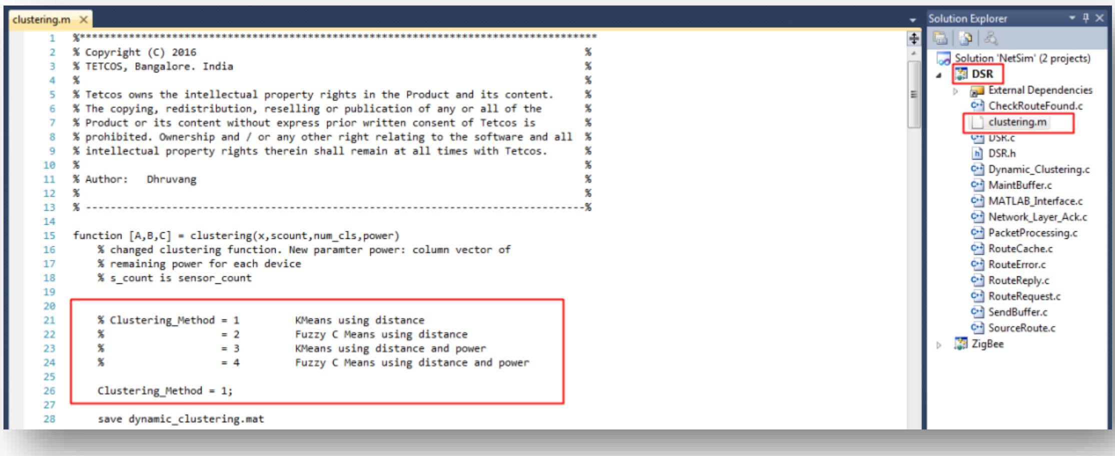
Dynamic Clustering is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and k-means algorithm that is implemented in MATLAB is used to dynamically perform clustering of the sensors into n number of clusters.

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, 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 added to the DSR project.

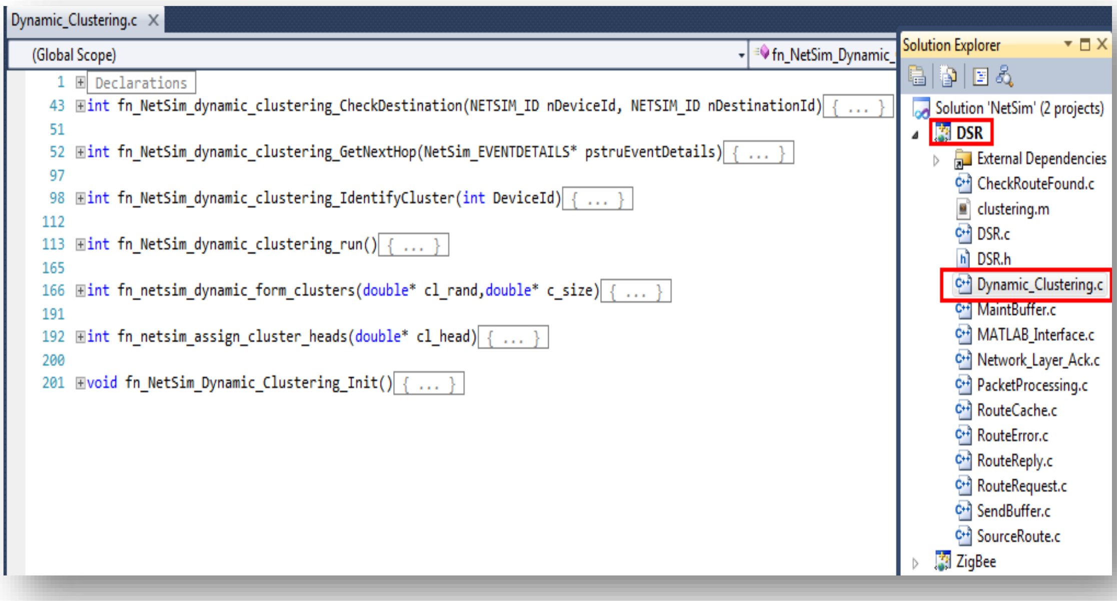


```
%***********************************************************************************
% Copyright (C) 2016                                                               %
% TETCOS, Bangalore. India                                                         %
%                                                                                  %
% Tetcos owns the intellectual property rights in the Product and its content.    %
% The copying, redistribution, reselling or publication of any or all of the      %
% Product or its content without express prior written consent of Tetcos is       %
% prohibited. Ownership and / or any other right relating to the software and all %
% intellectual property rights therein shall remain at all times with Tetcos.     %
%                                                                                  %
% Author:   Dhruvang                                                               %
%                                                                                  %
% --------------------------------------------------------------------------------%

function [A,B,C] = clustering(x,scount,num_cls,power)
    % changed clustering function. New paramter power: column vector of
    % remaining power for each device
    % s_count is sensor_count


    % Clustering_Method = 1         KMeans using distance
    %                   = 2         Fuzzy C Means using distance
    %                   = 3         KMeans using distance and power
    %                   = 4         Fuzzy C Means using distance and power

    Clustering_Method = 1;

    save dynamic_clustering.mat
```

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.



```
Declarations
int fn_NetSim_dynamic_clustering_CheckDestination(NETSIM_ID nDeviceId, NETSIM_ID nDestinationId){ ... }
int fn_NetSim_dynamic_clustering_GetNextHop(NetSim_EVENTDETAILS* pstruEventDetails){ ... }
int fn_NetSim_dynamic_clustering_IdentifyCluster(int DeviceId){ ... }
int fn_NetSim_dynamic_clustering_run(){ ... }
int fn_netsim_dynamic_form_clusters(double* cl_rand,double* c_size){ ... }
int fn_netsim_assign_cluster_heads(double* cl_head){ ... }
void fn_NetSim_Dynamic_Clustering_Init(){ ... }
```

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:

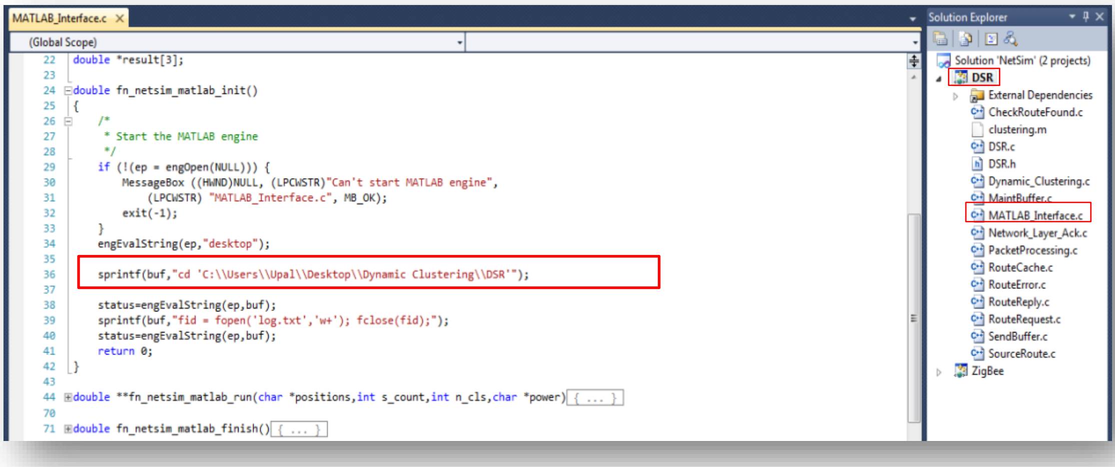
To run this code 32- bit version of MATLAB must be installed in your system.

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.

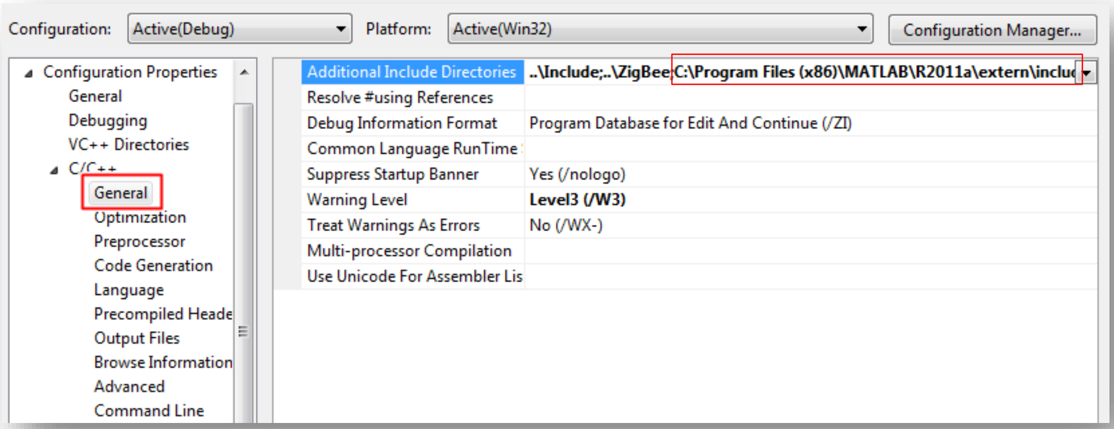
3. Inside the fn_netsim_matlab_init() change the path to your systems current path where the clustering.m and MATLAB_Interface.c files are present.
   sprintf(buf,"cd 'C:\\Users\\Upal\\Desktop\\Dynamic_Clustering_9.1\\DSR'");
   This file is present in the DSR folder inside the Simulation – Dynamic_Clustering directory.



```c
double *result[3];

double fn_netsim_matlab_init()
{
    /*
     * Start the MATLAB engine
     */
    if (!(ep = engOpen(NULL))) {
        MessageBox ((HWND)NULL, (LPCWSTR)"Can't start MATLAB engine",
            (LPCWSTR) "MATLAB_Interface.c", MB_OK);
        exit(-1);
    }
    engEvalString(ep,"desktop");

    sprintf(buf,"cd 'C:\\Users\\Upal\\Desktop\\Dynamic Clustering\\DSR'");

    status=engEvalString(ep,buf);
    sprintf(buf,"fid = fopen('log.txt','w+'); fclose(fid);");
    status=engEvalString(ep,buf);
    return 0;
}

double **fn_netsim_matlab_run(char *positions,int s_count,int n_cls,char *power){ ... }

double fn_netsim_matlab_finish(){ ... }
```

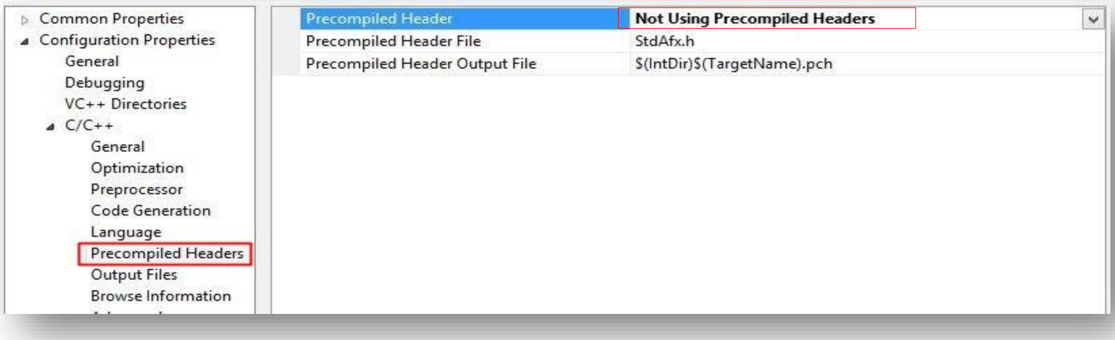
4. Right click on the DSR project and select PROPERTIES in the solution explorer to open the project properties. Once this window has opened, make the following changes:

a. Under C/C++ General, add the following directory to the field ADDITIONAL INCLUDE DIRECTORIES:
<Path where MATLAB is installed>\extern\include



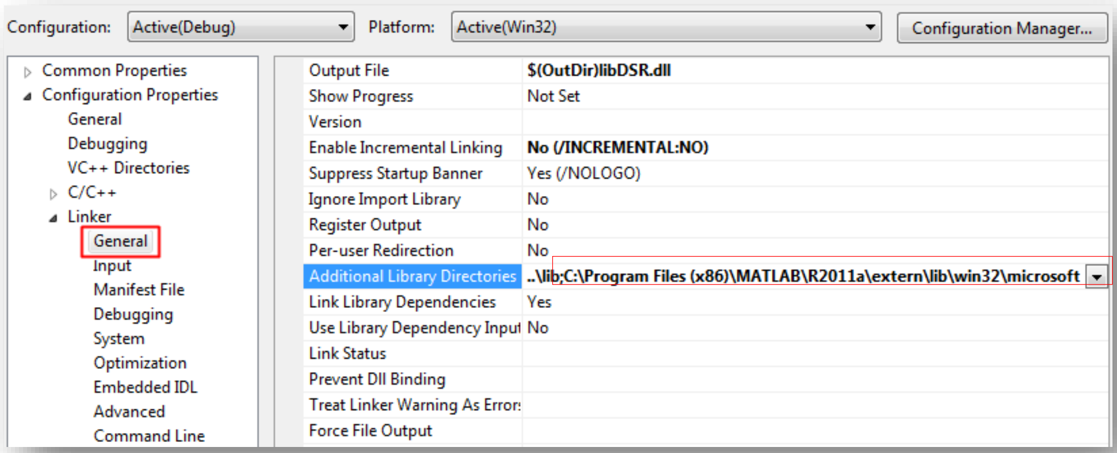
Additional Include Directories	..\Include;..\ZigBee;C:\Program Files (x86)\MATLAB\R2011a\extern\incluc
Resolve #using References	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Multi-processor Compilation	
Use Unicode For Assembler Lis	

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



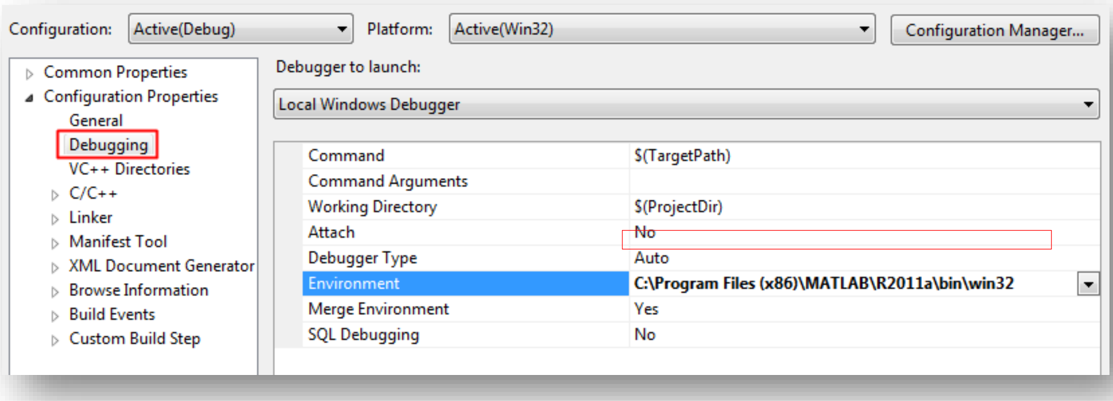
Precompiled Header	**Not Using Precompiled Headers**
Precompiled Header File	StdAfx.h
Precompiled Header Output File	$(IntDir)$(TargetName).pch

c. Under Linker General, add the directory to the field ADDITIONAL LIBRARY DIRECTORIES:
<Path where MATLAB is installed>\extern\lib\win32\microsoft



Output File	$(OutDir)libDSR.dll
Show Progress	Not Set
Version	
Enable Incremental Linking	No (/INCREMENTAL:NO)
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	..\lib;C:\Program Files (x86)\MATLAB\R2011a\extern\lib\win32\microsoft
Link Library Dependencies	Yes
Use Library Dependency Input	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	

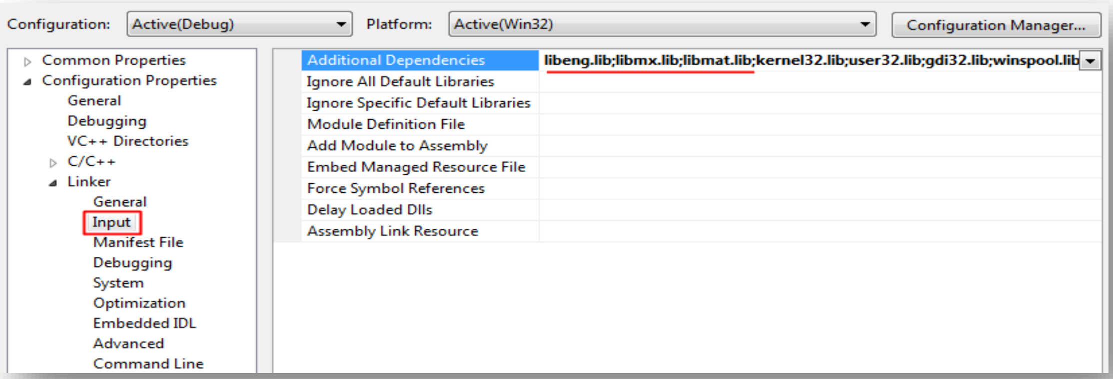
d. Under Configuration Properties ->Debugging
Add the following Target path in the *Environment*:
<Path where MATLAB is installed>\bin\win32



Debugger to launch:
Local Windows Debugger

Command	$(TargetPath)
Command Arguments	
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	C:\Program Files (x86)\MATLAB\R2011a\bin\win32
Merge Environment	Yes
SQL Debugging	No

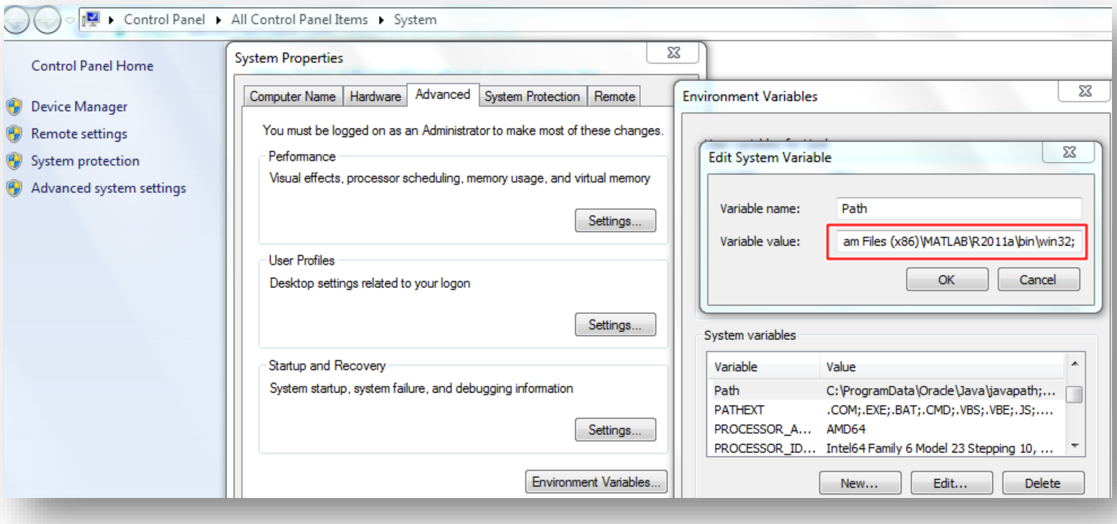
e. Under Linker Input, add the following names to the field marked ADDITIONAL DEPENDENCIES:
libmx.lib; libmat.lib; libeng.lib by separating them with a semicolon.



Additional Dependencies	libeng.lib;libmx.lib;libmat.lib;kernel32.lib;user32.lib;gdi32.lib;winspool.lib
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

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



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your logon

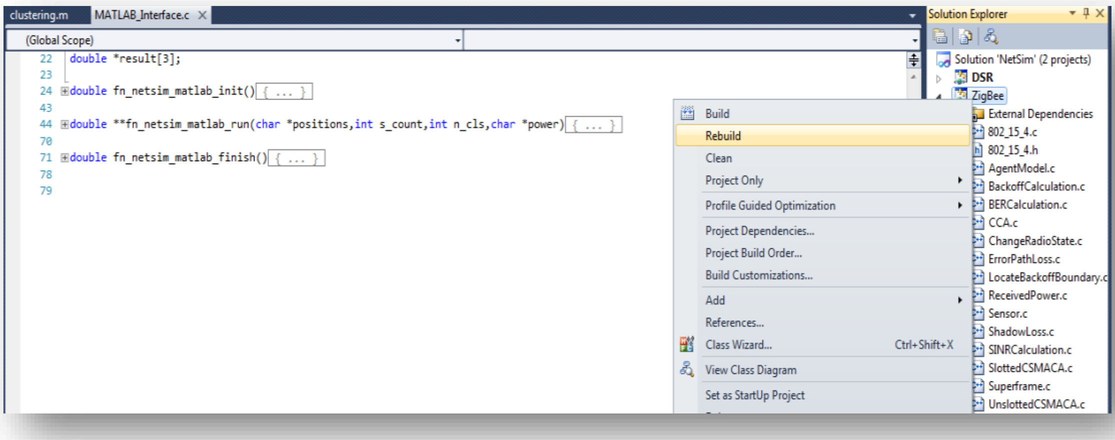
Startup and Recovery
System startup, system failure, and debugging information

Variable name: Path
Variable value: am Files (x86)\MATLAB\R2011a\bin\win32;

Variable	Value
Path	C:\ProgramData\Oracle\Java\javapath;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64
PROCESSOR_ID...	Intel64 Family 6 Model 23 Stepping 10, ...

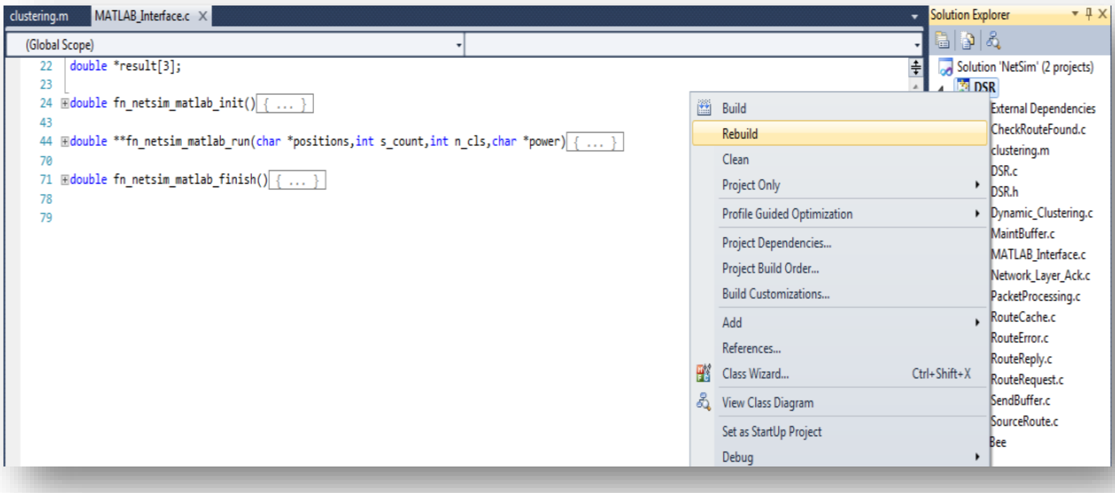
Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 32-bit application, the directory in the MATLAB 32-bit installation must be the first one on the PATH).

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



```
double *result[3];

double fn_netsim_matlab_init(){ ... }

double **fn_netsim_matlab_run(char *positions,int s_count,int n_cls,char *power){ ... }

double fn_netsim_matlab_finish(){ ... }
```

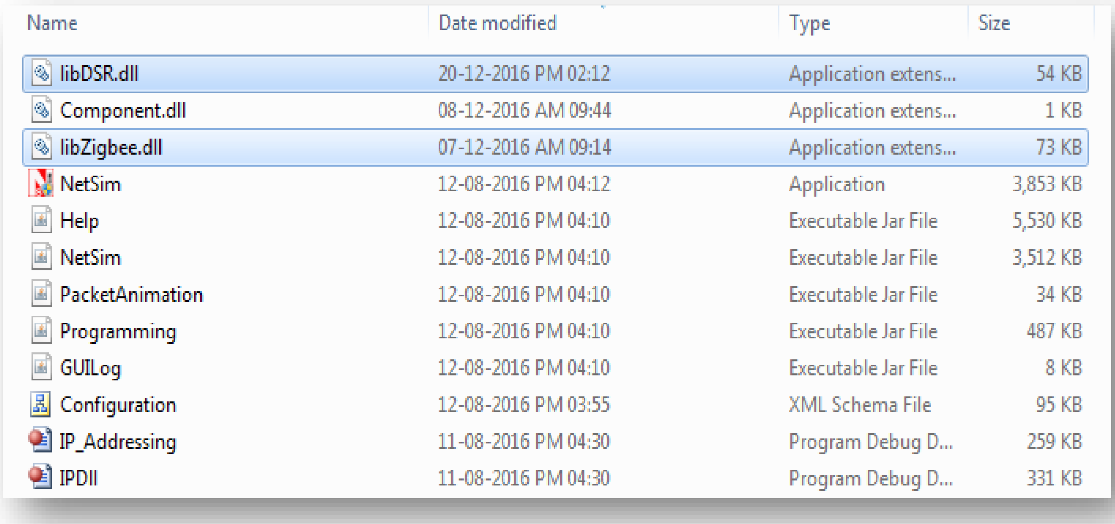
6. Right click on the DSR project in the solution explorer and select Rebuild.



```
double *result[3];

double fn_netsim_matlab_init(){ ... }

double **fn_netsim_matlab_run(char *positions,int s_count,int n_cls,char *power){ ... }

double fn_netsim_matlab_finish(){ ... }
```

7. Copy the newly built libDSR.dll and libZigBee.dll from the DLL folder inside the Simulation – Dynamic_clustering Directory.

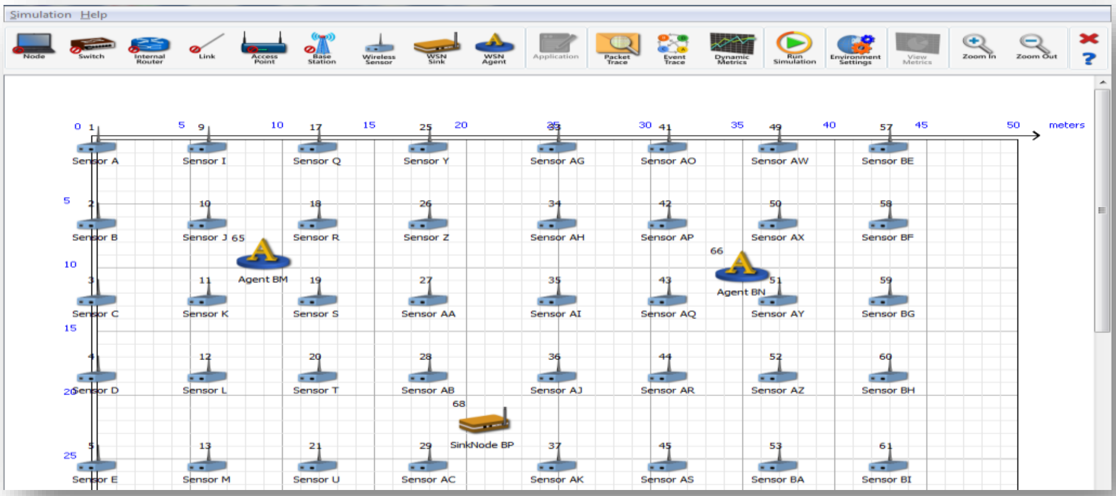
8. Replace the DLL's in the bin folder inside NetSim Installation Directory, after renaming the original libDSR.dll and libZigBee.dll.



Name	Date modified	Type	Size
libDSR.dll	20-12-2016 PM 02:12	Application extens...	54 KB
Component.dll	08-12-2016 AM 09:44	Application extens...	1 KB
libZigbee.dll	07-12-2016 AM 09:14	Application extens...	73 KB
NetSim	12-08-2016 PM 04:12	Application	3,853 KB
Help	12-08-2016 PM 04:10	Executable Jar File	5,530 KB
NetSim	12-08-2016 PM 04:10	Executable Jar File	3,512 KB
PacketAnimation	12-08-2016 PM 04:10	Executable Jar File	34 KB
Programming	12-08-2016 PM 04:10	Executable Jar File	487 KB
GUILog	12-08-2016 PM 04:10	Executable Jar File	8 KB
Configuration	12-08-2016 PM 03:55	XML Schema File	95 KB
IP_Addressing	11-08-2016 PM 04:30	Program Debug D...	259 KB
IPDll	11-08-2016 PM 04:30	Program Debug D...	331 KB

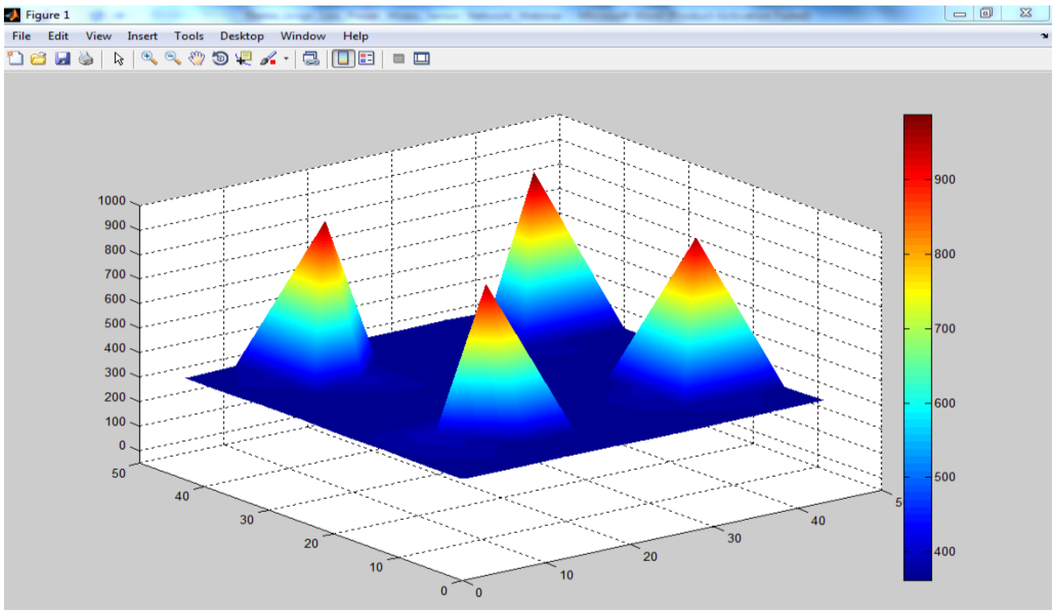
9. Run NetSim as Administrative mode.

10. Create a Network Scenario in WSN (for eg. 64 sensors) and make sure that the velocity of the sensors is set to 0. This property will be available in the Global Properties of the sensor nodes.



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

Case 1: Clustering_Method = 1 (K-means plot for power consumption)

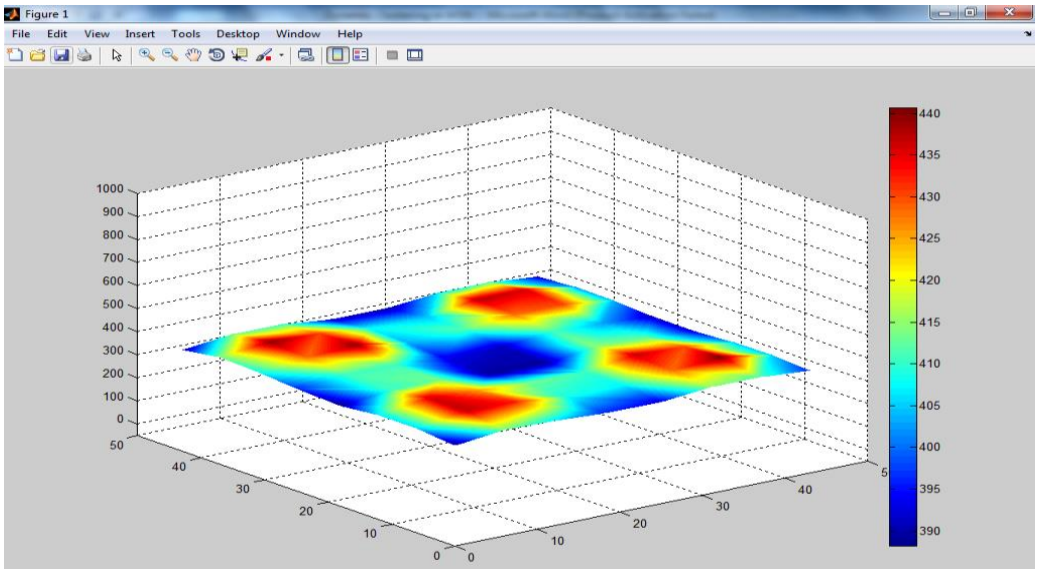


64 sensors are placed evenly on x-y plane and each sensor is given a fixed amount of initial power. The number of clusters has been fixed to 4.

The z axis represents the power consumed while the sensors are placed on the x, y plane.

Case 1 represents k-means with distance. As it is seen from the plot, there are 4 peaks in the plot corresponding to 4 sensors that will be selected as the cluster heads.

Case 2: Clustering_Method = 3(Modified K-means plot for power consumption)



Setting 3 represents modified k-means with distance and power.

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

There are no peaks in this plot unlike the previous one because modified K-means takes into account the power level of each sensor and thus each sensor will be appointed as the cluster head in its respective cluster.