

Dynamic Clustering in WSN

Software Recommended: NetSim Standard v12.1 (32/64 bit), Visual Studio 2015/2017/2019, MATLAB (32/64 bit)

Clustering in WSN:

Clustering is the process 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 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:

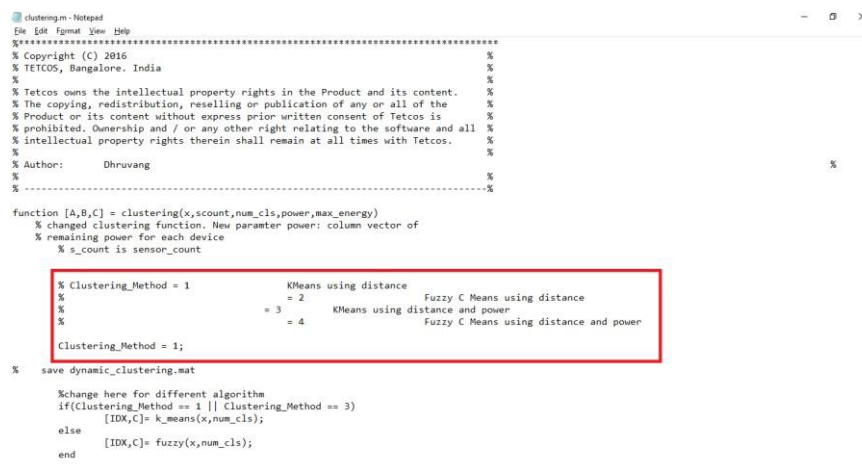
Dynamic Clustering is implemented in NetSim by Interfacing with MATLAB for the purpose of mathematical calculation. The sensor coordinates are fed as input to MATLAB and k-means algorithm that is implemented in MATLAB is used to dynamically perform clustering of the sensors into n number of clusters.

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 as shown below:



```
clustering.m - Notepad
File Edit Format View Help
% 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,max_energy)
% changed clustering function. New parameter power: column vector of
% remaining power for each device
% s_count is sensor_count

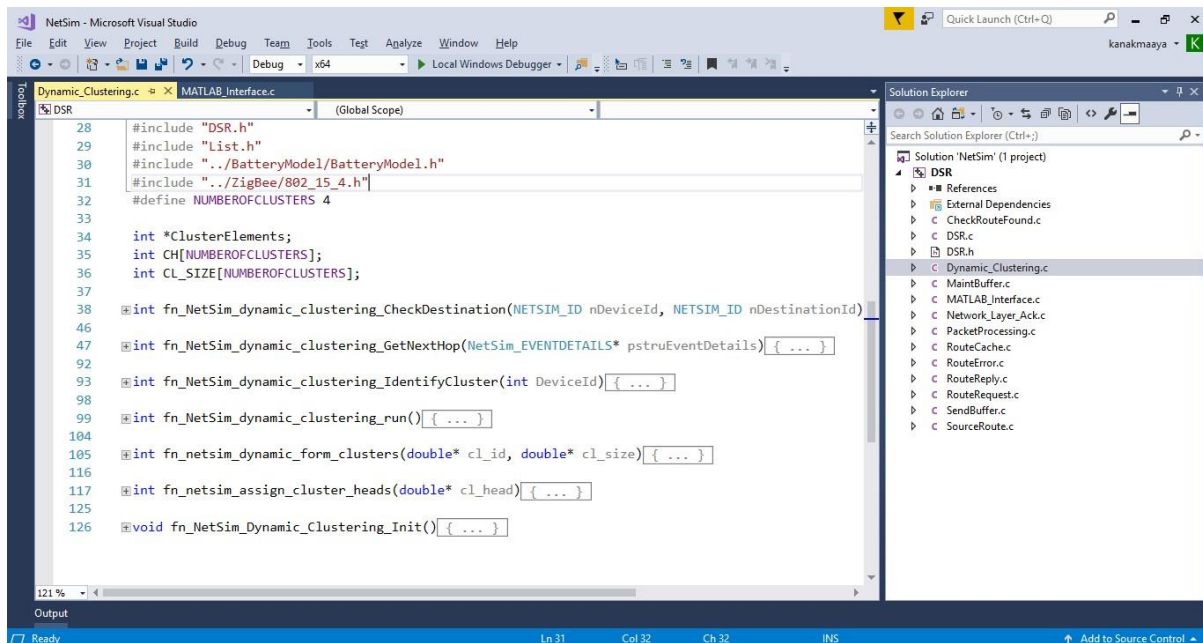
% Clustering_Method = 1
% = 2 KMeans using distance Fuzzy C Means using distance
% = 3 KMeans using distance and power
% = 4 Fuzzy C Means using distance and power
Clustering_Method = 1;

% save dynamic_clustering.mat

% Change here for different algorithm
if(Clustering_Method == 1 || Clustering_Method == 3)
    [IDX,C]= k_means(x,num_cls);
else
    [IDX,C]= fuzzy(x,num_cls);
end
```

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:



```
28 #include "DSR.h"
29 #include "List.h"
30 #include "../BatteryModel/BatteryModel.h"
31 #include "../ZigBee/802_15_4.h"
32 #define NUMBEROFCLUSTERS 4
33
34 int *ClusterElements;
35 int CH[NUMBEROFCLUSTERS];
36 int CL_SIZE[NUMBEROFCLUSTERS];
37
38 int fn_NetSim_dynamic_clustering_CheckDestination(NETSIM_ID nDeviceId, NETSIM_ID nDestinationId)
39
40
41
42
43
44
45
46
47 int fn_NetSim_dynamic_clustering_GetNextHop(Netsim_EVENTDETAILS* pstruEventDetails){ ... }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93 int fn_NetSim_dynamic_clustering_IdentifyCluster(int DeviceId){ ... }
94
95
96
97
98
99
100
101
102
103
104
105 int fn_netsim_dynamic_form_clusters(double* cl_id, double* cl_size){ ... }
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

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.

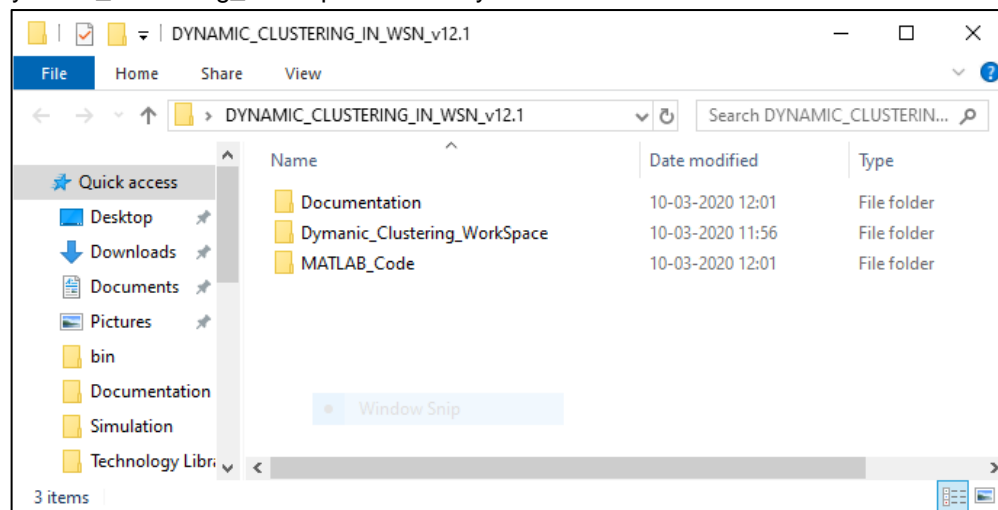
If the current sensor is not the source then the packet is sent to the destination (i.e) the sinknode.

NOTE:

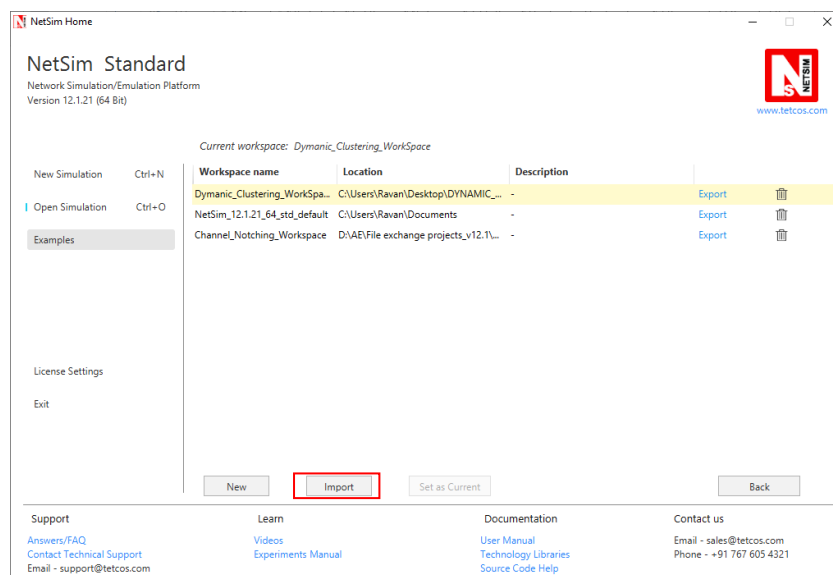
To run this code 64- bit version of MATLAB must be installed in your system.

Steps:

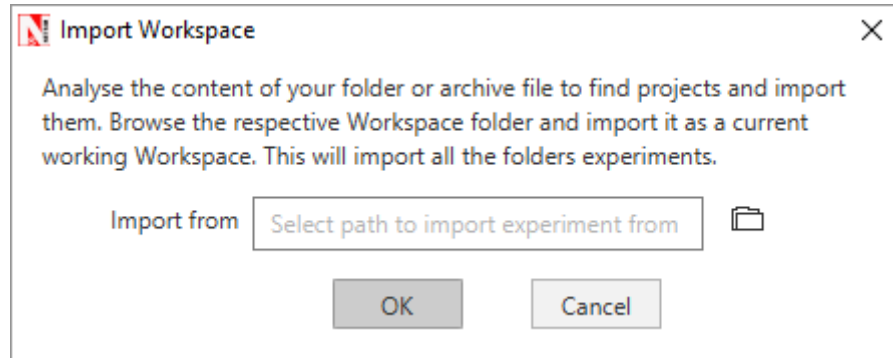
1. The downloaded project folder contains the folders Documentation, MATLAB_Code and Dynamic_Clustering_Workspace directory as shown below:



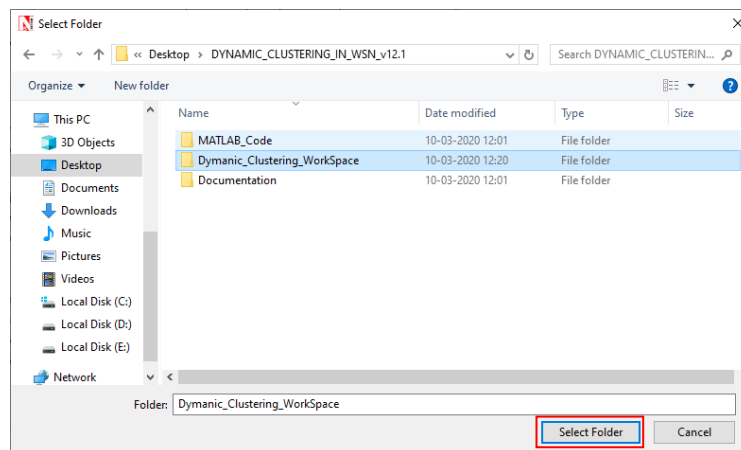
2. Import Dynamic_Clustering_Workspace by going to Open Simulation->Workspace Options->More Options in NetSim Home window. Then select Import as shown below:



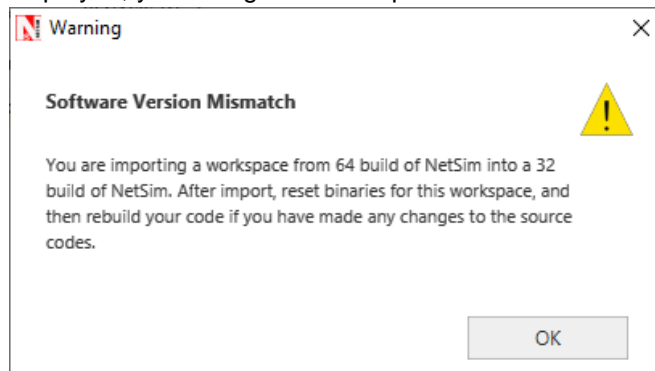
- It displays a window where users need to give the path of the workspace folder and click on OK as shown below:



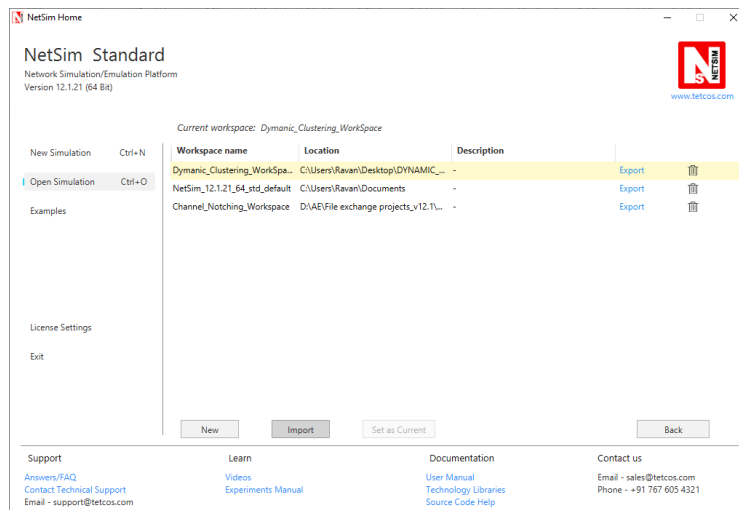
- Browse to the Dynamic_Clustering_Workspace folder and click on select folder as shown below:



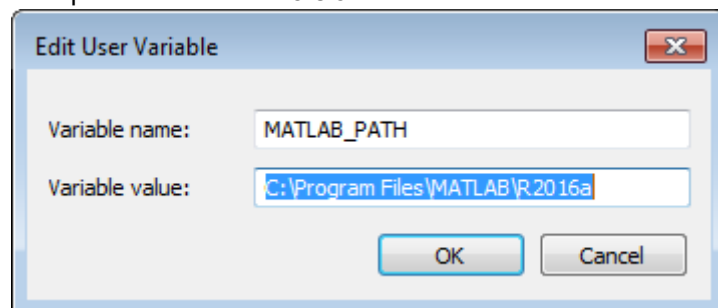
- After this click on OK button in the Import Workspace window.
- While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



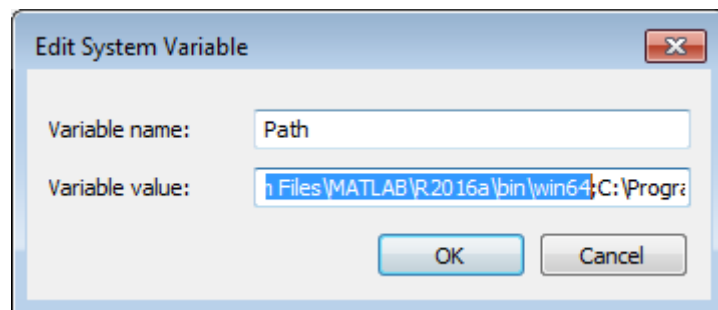
- The Imported workspace will be set as the current workspace automatically. To see the imported workspace, click on Open Simulation->Workspace Options->More Options as shown below:



8. Create a user variable with the name of MATLAB_PATH and provide the path of the installation directory of user's respective MATLAB version.



9. Make sure that the following directory is in the PATH(Environment variable)
<Path where MATLAB is installed>\bin\win64



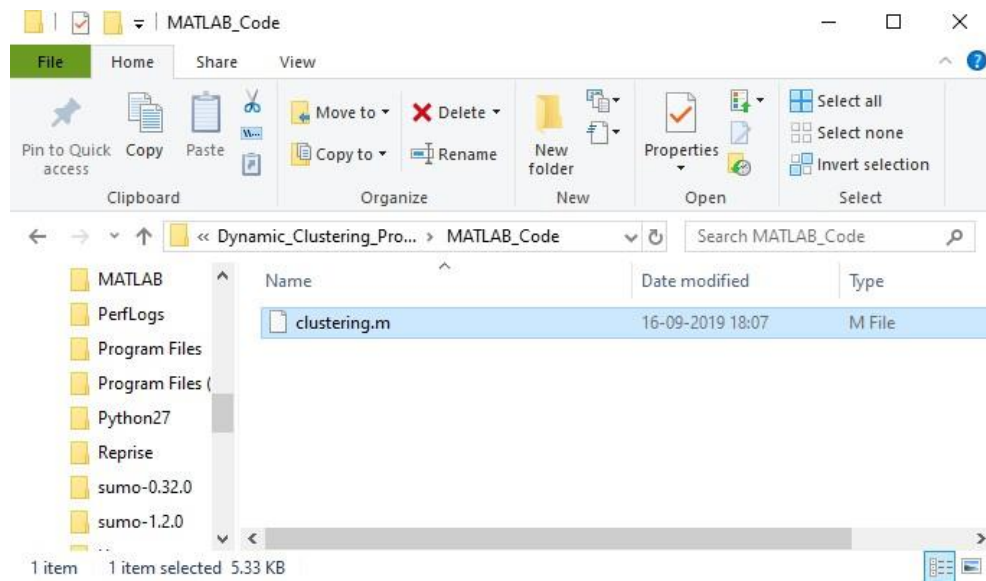
Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

10. 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.

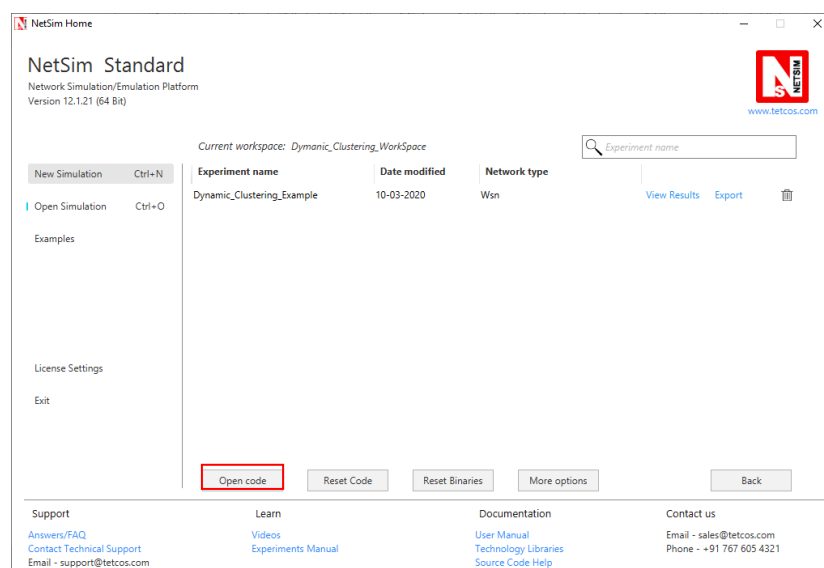
```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>matlab -regserver_
```

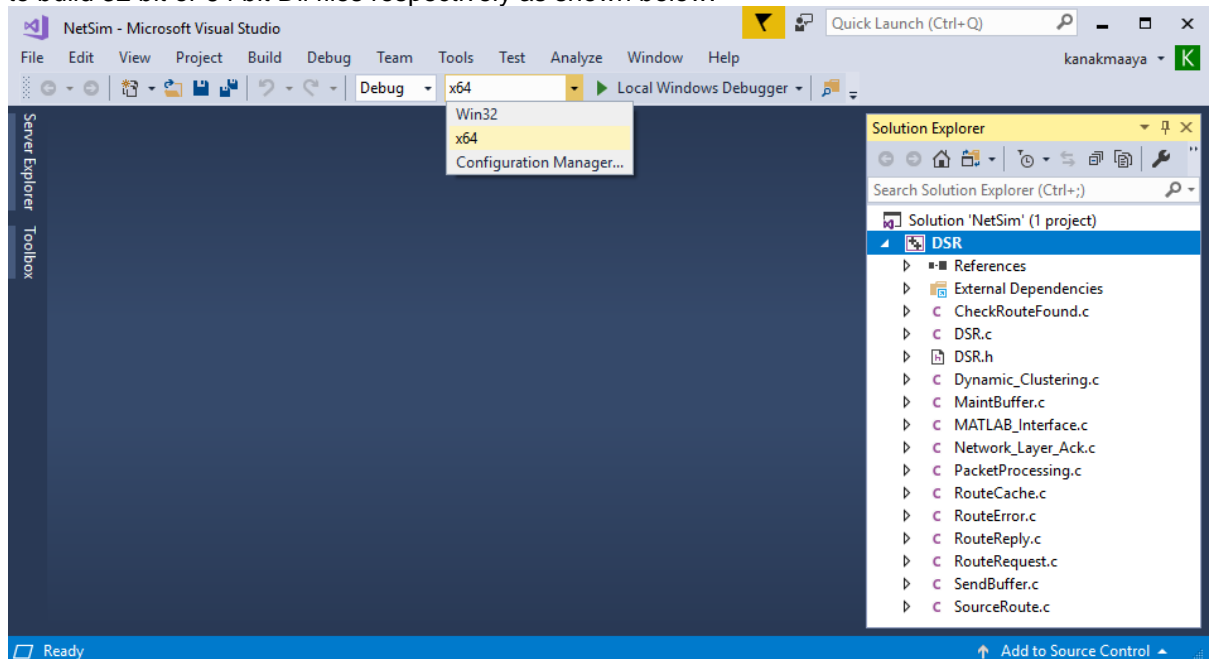
11. Place **clustering.m** present in the MATLAB_Code folder inside the root directory of MATLAB. For Example: “C:\Program Files\MATLAB\R2016a”.



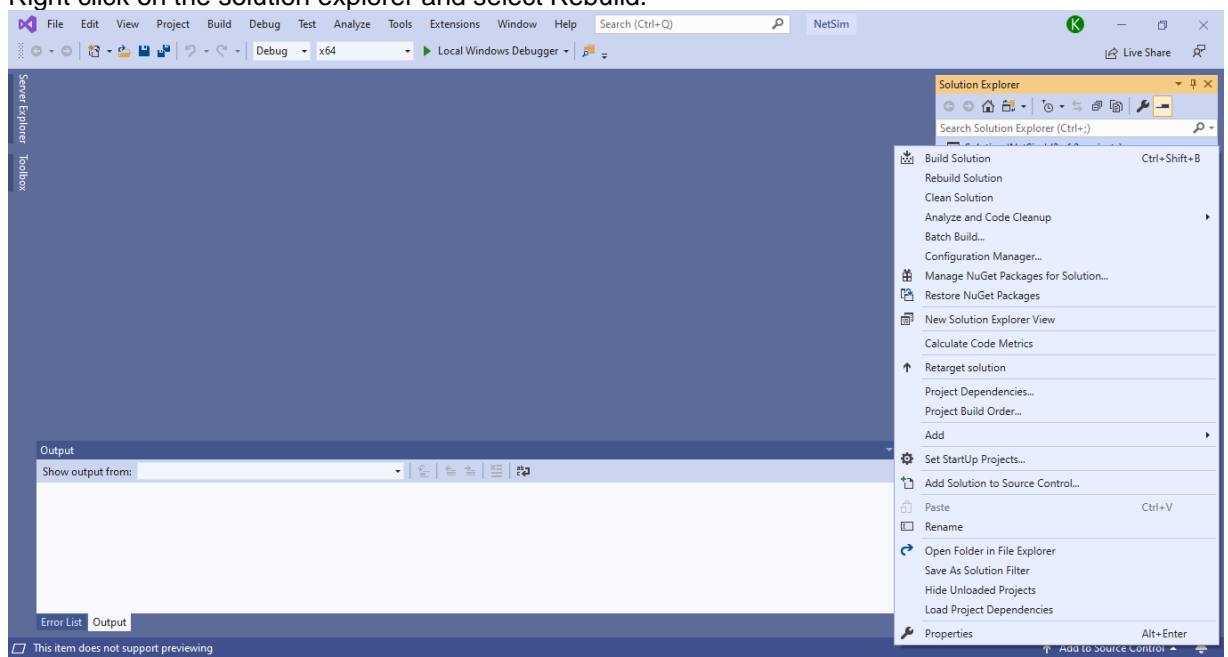
12. Open the Source codes in Visual Studio by going to Open Simulation-> Workspace Options and Clicking on Open code button as shown below:



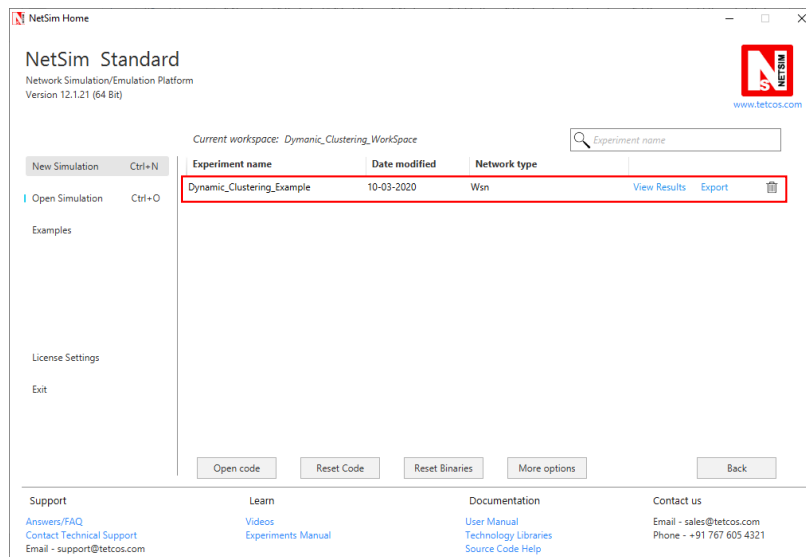
13. Under the DSR project in the solution explorer you will be able to see that **MATLAB_Interface.c** and **Dynamic_Clustering.c** files which contain source codes related to interactions with MATLAB and handling clustering in NetSim respectively.
14. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit DLL files respectively as shown below:



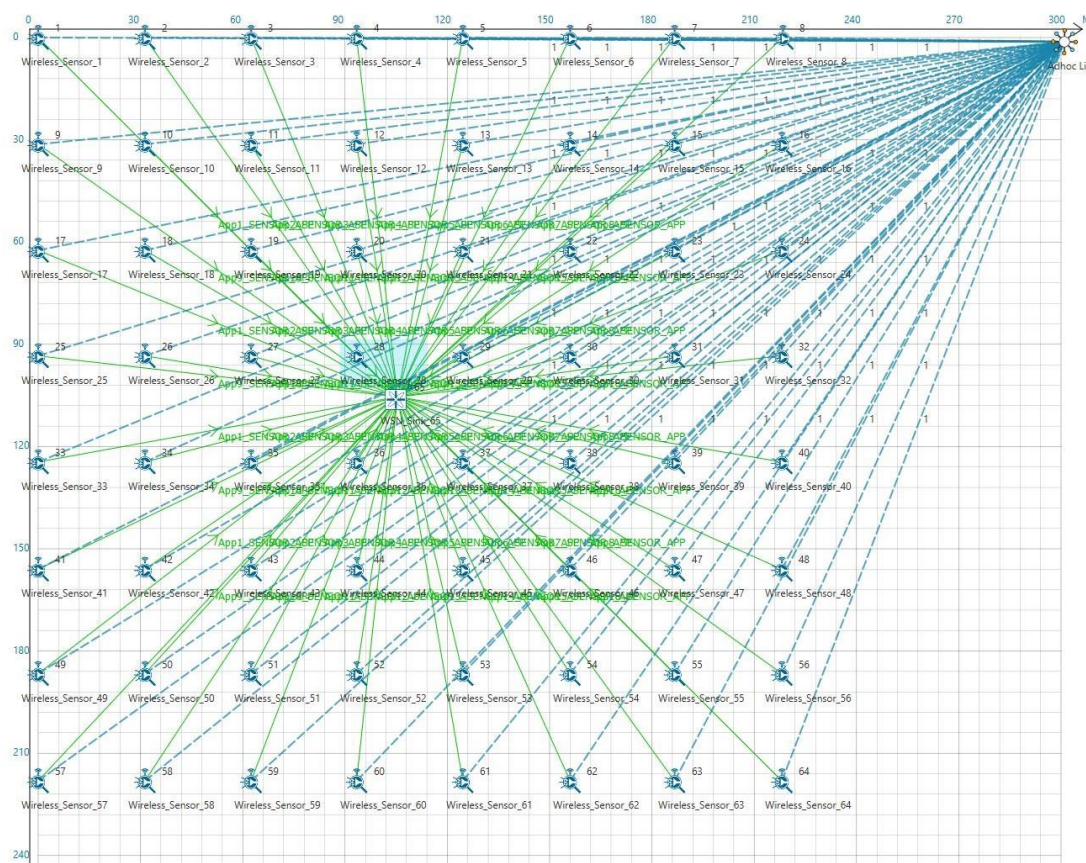
15. Right click on the solution explorer and select Rebuild.



16. Upon successful build modified libDSR.dll and libzigbee.dll files gets automatically updated in the directory containing NetSim binaries.
17. Run NetSim as Administrative mode.
18. Then Dynamic_Clustering_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the Dynamic_Clustering_Example that is present under the list of experiments as shown below:



19. The saved network scenario consisting 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.



20. Run the Scenario. You will observe that as the simulation starts in NetSim, MATLAB gets initialized and graph associated with energy consumption in the sensor network is plotted during runtime.

Analysis:

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

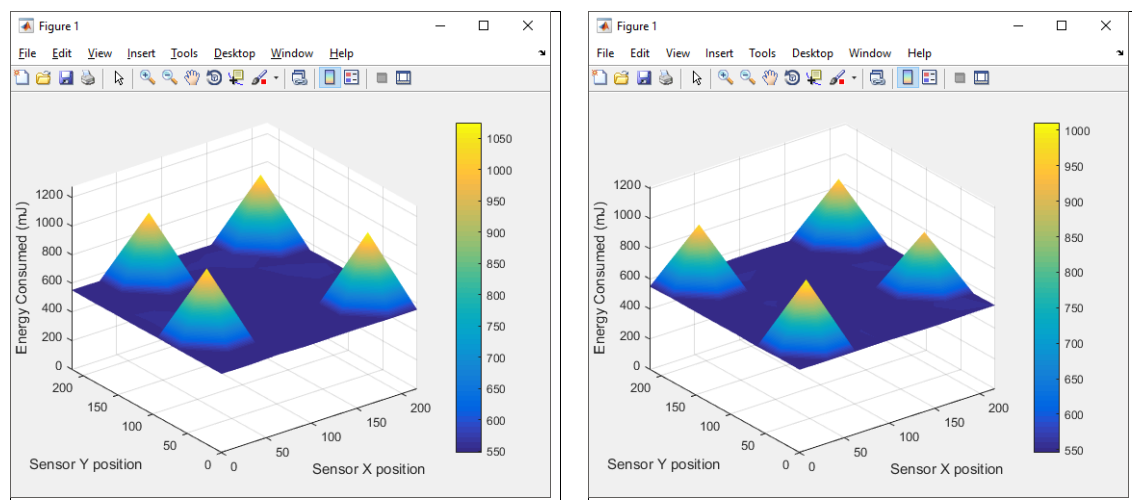
At the end of the simulation, NetSim provides Battery Model Metrics which provides detailed information related to energy consumption in each sensor node with respect to transmission, reception, idle mode, sleep mode etc as shown below:

Battery model							
<input checked="" type="checkbox"/> Detailed View							
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

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 cluster is set to 4), cluster head election and to obtain energy consumption plots.

Cluster head election using distance alone as a parameter:

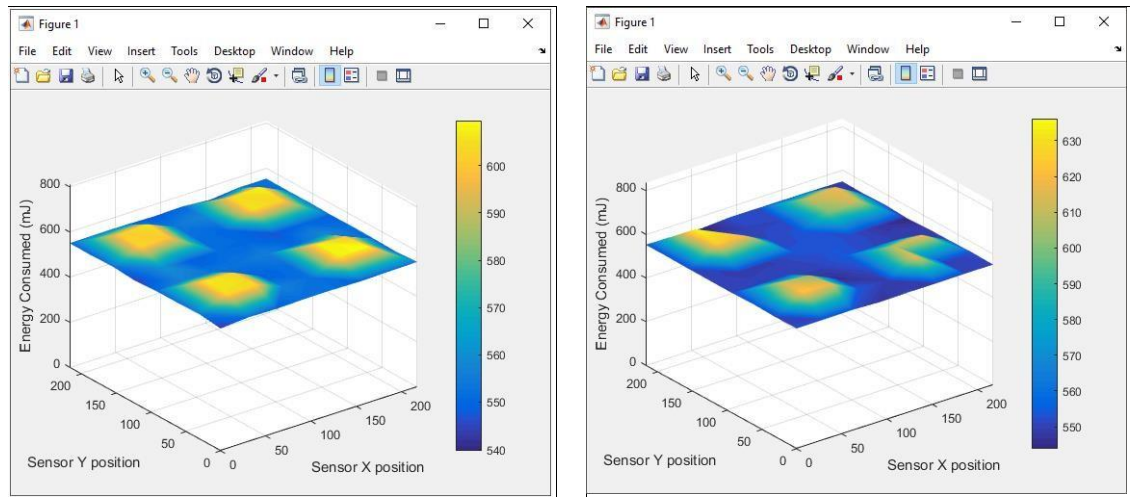
Running simulations with Clustering Method set to 1 and 2 in the **clustering.m** file will provide energy consumption plots for kmeans and fuzzy c-means algorithms respectively as shown below:



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 distance is used as a parameter for electing the cluster heads.

Cluster head election using distance and remaining energy as parameters:

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



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 sharp peaks in this plot unlike the previous one because modified K-means takes into account 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 its respective cluster.