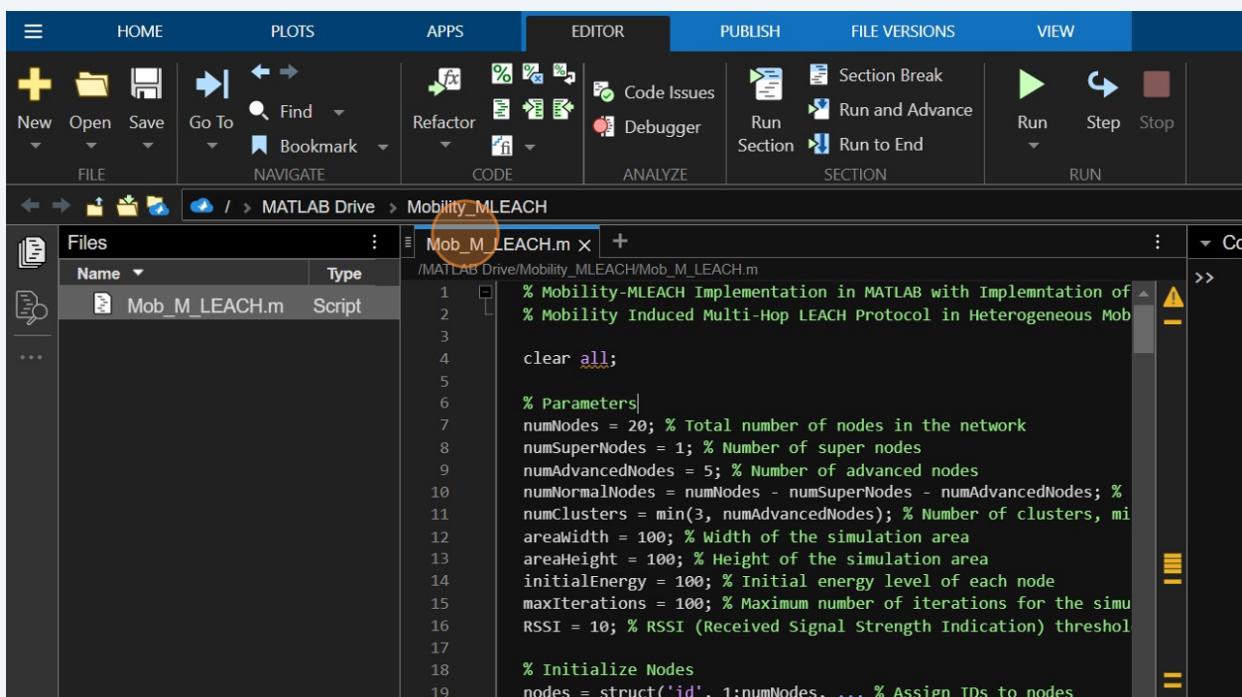


Mobility Induced Multi-Hop LEACH Protocol Scribe in Heterogeneous Mobile Network Implementation in MATLAB (with Code) By Laraib Azmat (21-SE-23) and Zain Ejaz (21-SE-71)

1 Navigate to <https://matlab.mathworks.com/>

2 Mobility-MLEACH (Mobility-Multi-level Energy Adaptive Clustering Hierarchy) is an advanced protocol designed for wireless sensor networks, integrating mobility-awareness with energy-efficient clustering. This protocol combines the principles of MLEACH, a variant of the well-known LEACH (Low Energy Adaptive Clustering Hierarchy), with mobility models to enhance network performance in dynamic environments.

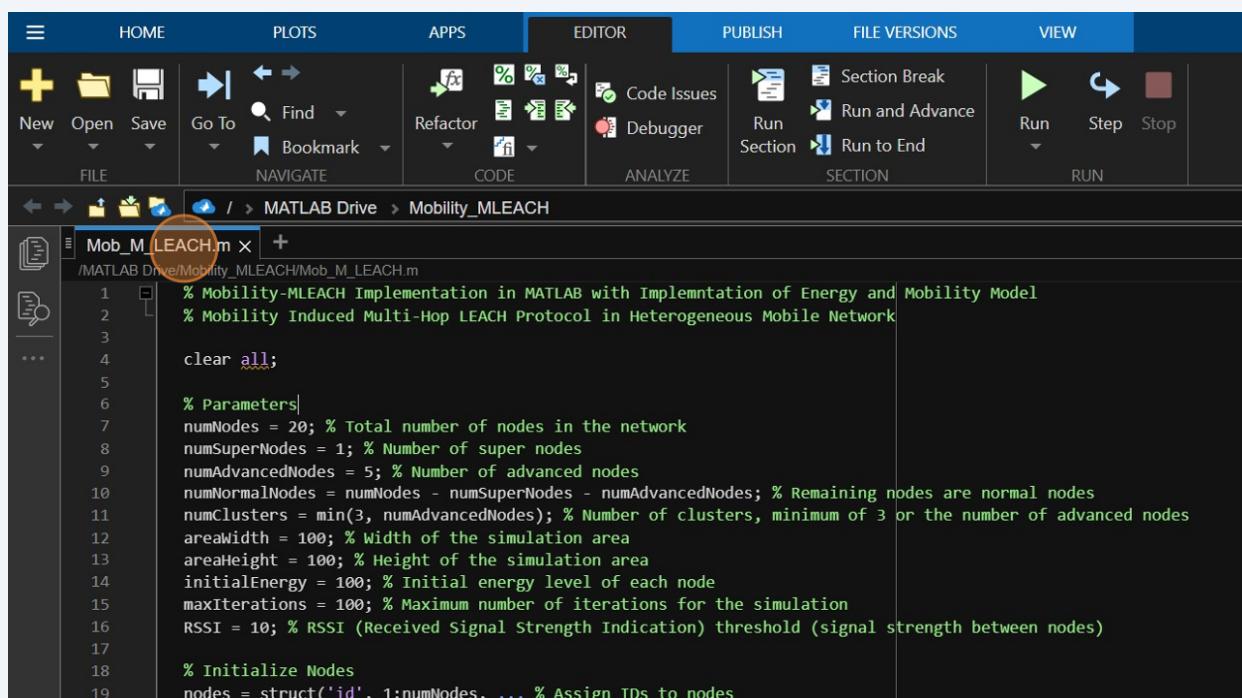


The screenshot shows the MATLAB IDE interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR (selected), PUBLISH, FILE VERSIONS, and VIEW. The left sidebar has sections for FILE, NAVIGATE, CODE, ANALYZE, SECTION, and RUN. The central workspace shows a script named 'Mob_M_LEACH.m' with the following code:

```
% Mobility-MLEACH Implementation in MATLAB with Implementation of  
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mob  
  
clear all;  
  
% Parameters  
numNodes = 20; % Total number of nodes in the network  
numSuperNodes = 1; % Number of super nodes  
numAdvancedNodes = 5; % Number of advanced nodes  
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; %  
numClusters = min(3, numAdvancedNodes); % Number of clusters, mi  
areaWidth = 100; % Width of the simulation area  
areaHeight = 100; % Height of the simulation area  
initialEnergy = 100; % Initial energy level of each node  
maxIterations = 100; % Maximum number of iterations for the simu  
RSSI = 10; % RSSI (Received Signal Strength Indication) threshol  
  
% Initialize Nodes  
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
```

3 The key features and components of Mobility-MLEACH include:

1. **Mobility Awareness:** Unlike traditional clustering protocols that assume static network topologies, Mobility-MLEACH acknowledges the dynamic nature of wireless sensor networks where nodes may move unpredictably due to factors like user mobility, environmental changes, or node failures. It incorporates mobility models to predict and adapt to node movements effectively.
2. **Multi-level Clustering:** Similar to MLEACH, Mobility-MLEACH employs a multi-level clustering approach to organize nodes into clusters based on their energy levels. This hierarchical structure helps in reducing energy consumption and extending network lifetime by evenly distributing the energy load among nodes.
3. **Energy Efficiency:** Mobility-MLEACH focuses on energy efficiency as a primary objective. By dynamically adjusting cluster heads based on both residual energy and node mobility patterns, it aims to prolong network lifetime while maintaining network connectivity and data delivery.
4. **Adaptability:** The protocol is designed to adapt to changing network conditions, including variations in node energy levels, mobility patterns, and environmental factors. It dynamically reconfigures cluster heads and cluster memberships to optimize energy consumption and network performance



```
% Mobility-MLEACH Implementation in MATLAB with Implementation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network

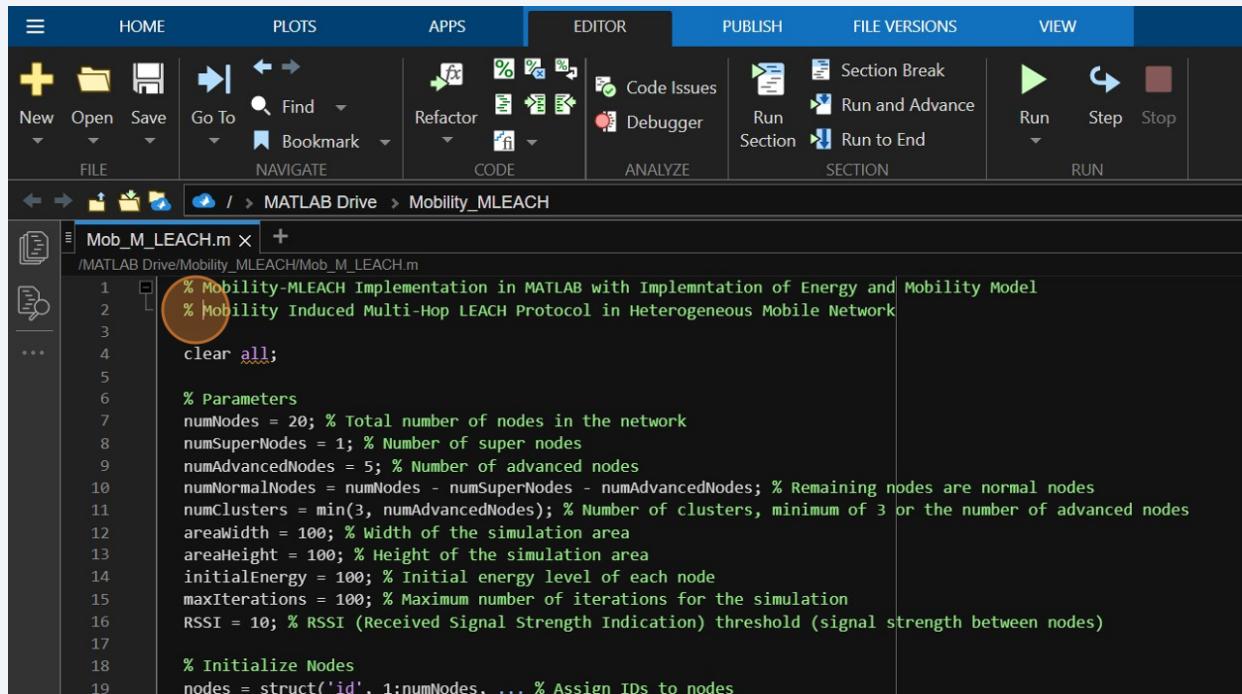
clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
```

4

Mobility induced means there is mobility present(induced) in the nodes and multi hops means data transmission is divided into multiple hops and staes that will lead to efficient use of energy and send data for longer distances
Heterogeneous means that it will contain multiple types of nodes (advanced, super and normal)



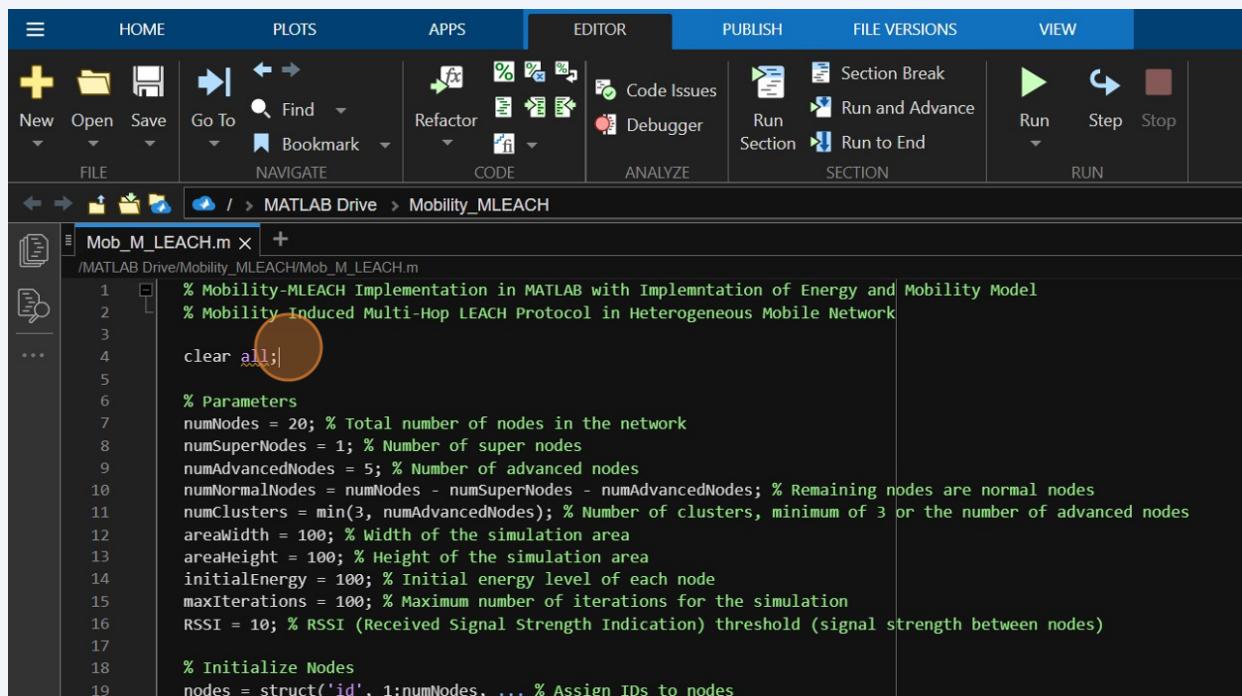
```
% Mobility-MLEACH Implementation in MATLAB with Implementation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network
clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
```

5

The clear all statement at the beginning of the MATLAB script is used to clear all variables from the workspace. This means it removes all variables that are currently stored in memory.

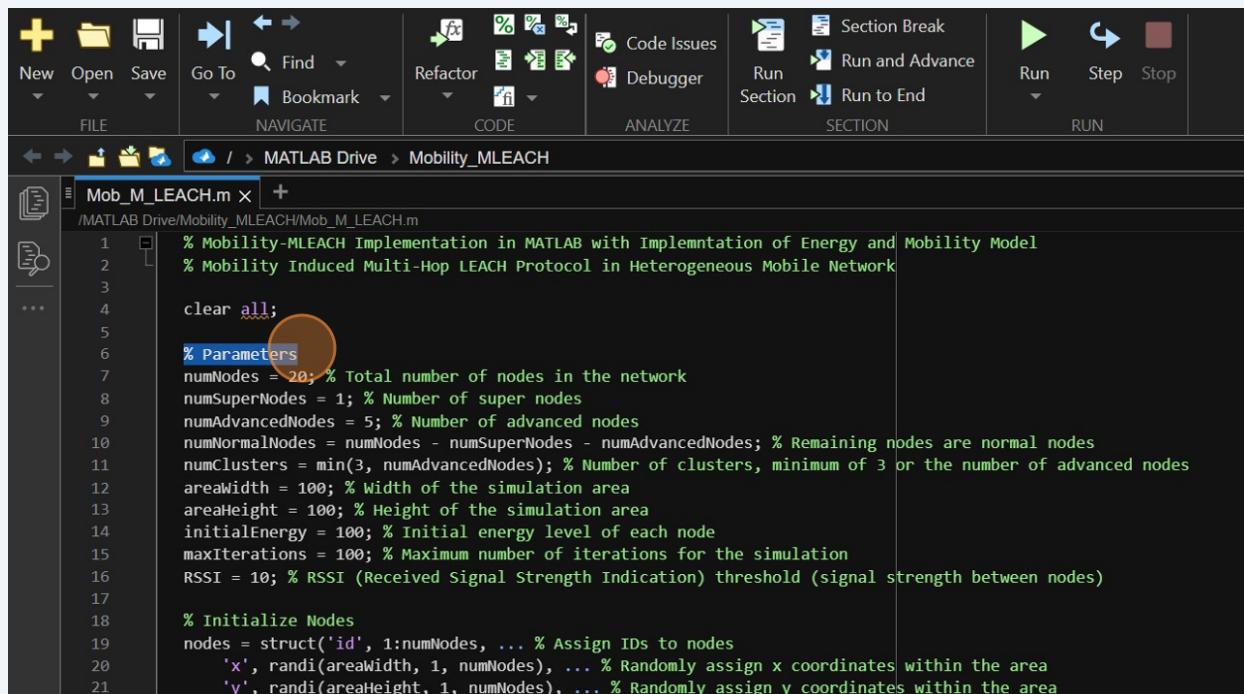


```
% Mobility-MLEACH Implementation in MATLAB with Implementation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network
clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
```

6 Firstly Lets start to initialize some perimeters to use in the code below



```
% Mobility-MLEACH Implementation in MATLAB with Implementation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network

clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
               'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
               'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area)
```

7

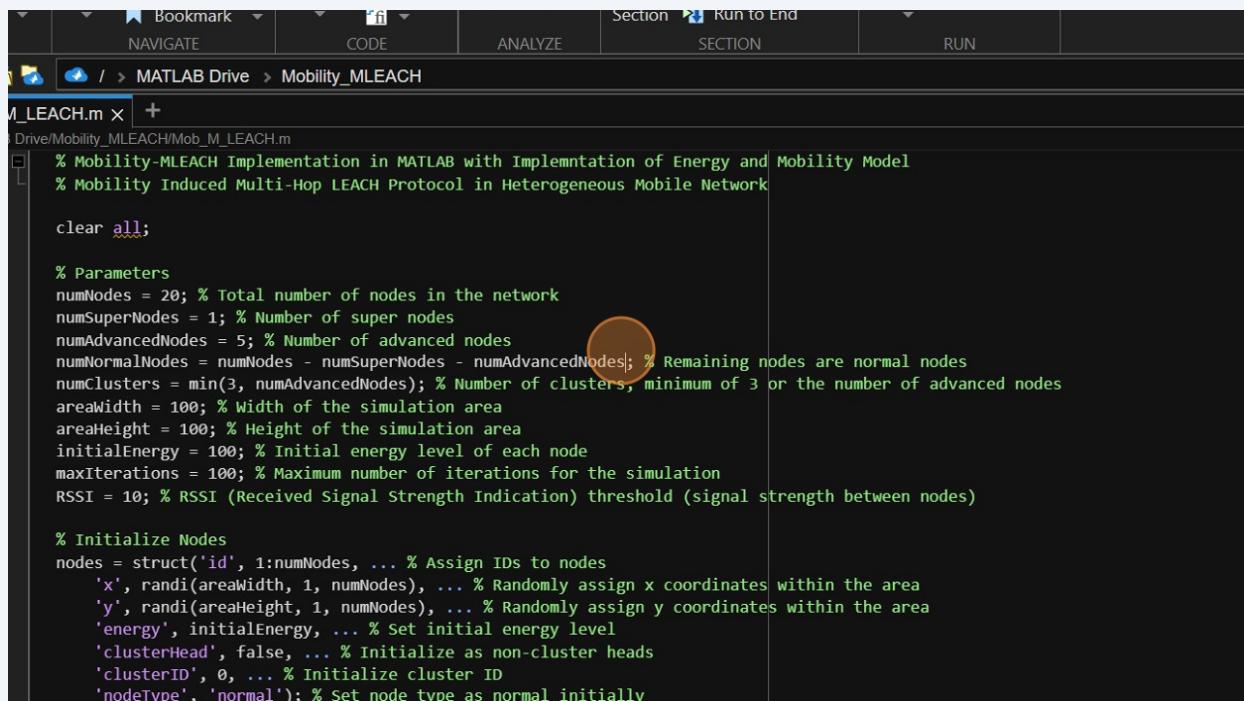
• **Total Number of Nodes** (numNodes): This parameter defines the total number of nodes present in the network. It includes all types of nodes: super nodes, advanced nodes, and normal nodes.

• **Number of Super Nodes** (numSuperNodes): Super nodes are a special type of node that may have higher capabilities or functionalities compared to other nodes. They might have more energy, higher processing power, or longer communication range.

• **Number of Advanced Nodes** (numAdvancedNodes): Advanced nodes are another special type of node in the network. They might have slightly more capabilities than normal nodes but less than super nodes. They are typically used in clustering or routing algorithms for more efficient network management.

• **Number of Normal Nodes** (numNormalNodes): These are the regular nodes in the network that do not possess any special characteristics like super nodes or advanced nodes. They are usually the majority of the nodes in the network.

• **Number of Clusters** (numClusters): This parameter determines how many clusters will be formed in the network. Clustering is a technique used in many wireless sensor network protocols for better organization and management. The number of clusters may depend on various factors such as network size, node density, and energy constraints. In this case, it is set to be the minimum of 3 or the number of advanced nodes, ensuring that there are enough advanced nodes to act as cluster heads in the network.



The screenshot shows a MATLAB code editor window titled 'Mob_M_Leach.m'. The code implements the Mobility-MLEACH protocol. A red circle highlights the line of code: 'numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes'. The code also initializes nodes with random coordinates ('x' and 'y'), initial energy, and sets them as non-cluster heads ('clusterHead').

```

% Mobility-MLEACH Implementation in MATLAB with Implementation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network

clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

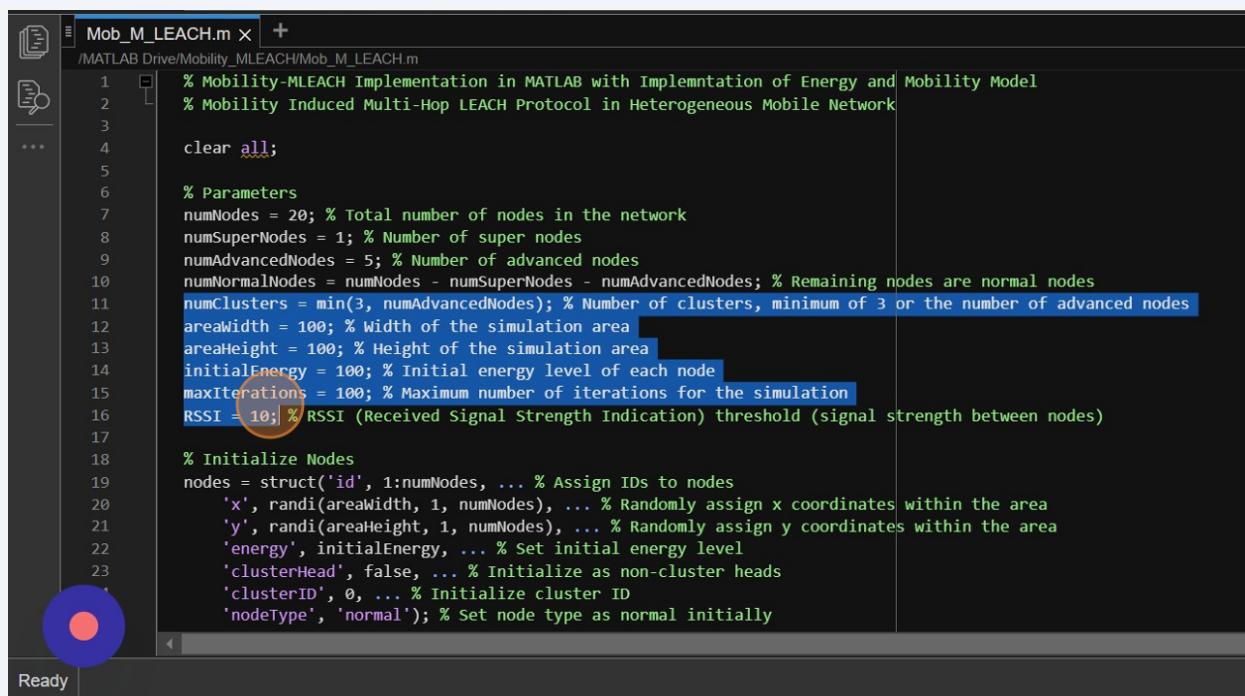
% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
    'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
    'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
    'energy', initialEnergy, ... % Set initial energy level
    'clusterHead', false, ... % Initialize as non-cluster heads
    'clusterID', 0, ... % Initialize cluster ID
    'nodeType', 'normal'); % Set node type as normal initially

```

8

These parameters define various aspects of the simulation environment and the nodes within it:

- areaWidth: This parameter specifies the width of the simulation area, usually measured in meters or some other distance unit. It determines the extent of the area in which nodes can move and communicate.
- areaHeight: Similar to areaWidth, this parameter specifies the height of the simulation area, defining the vertical extent of the simulation space.
- initialEnergy: This parameter sets the initial energy level for each node in the network. The energy level represents the amount of energy available for the node to perform tasks such as data transmission, reception, and processing. It's a crucial parameter in energy-constrained networks, as nodes need to manage their energy efficiently to prolong network lifetime.
- maxIterations: This parameter defines the maximum number of iterations or time steps for which the simulation will run. Each iteration typically represents a discrete time step in the simulation, during which various actions such as node movement, energy consumption, and data transmission take place.
- RSSI: RSSI stands for Received Signal Strength Indication. It's a measure of the power level of the signal received by a node from another node or source. In wireless communication, RSSI is often used as an indicator of the strength of the received signal, which can be used to estimate the distance between nodes or to determine if a signal is strong enough for reliable communication.



The screenshot shows a MATLAB script named 'Mob_M_LEACH.m' in a code editor. The code implements the Mobility-MLEACH protocol. It starts with a header comment and then initializes variables. A specific line of code, 'RSSI = 10;', is highlighted with a red oval. The code also includes comments explaining the parameters and their meanings.

```
% Mobility-MLEACH Implementation in MATLAB with Implementation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network

clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
    'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
    'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
    'energy', initialEnergy, ... % Set initial energy level
    'clusterHead', false, ... % Initialize as non-cluster heads
    'clusterID', 0, ... % Initialize cluster ID
    'nodeType', 'normal'); % Set node type as normal initially
```

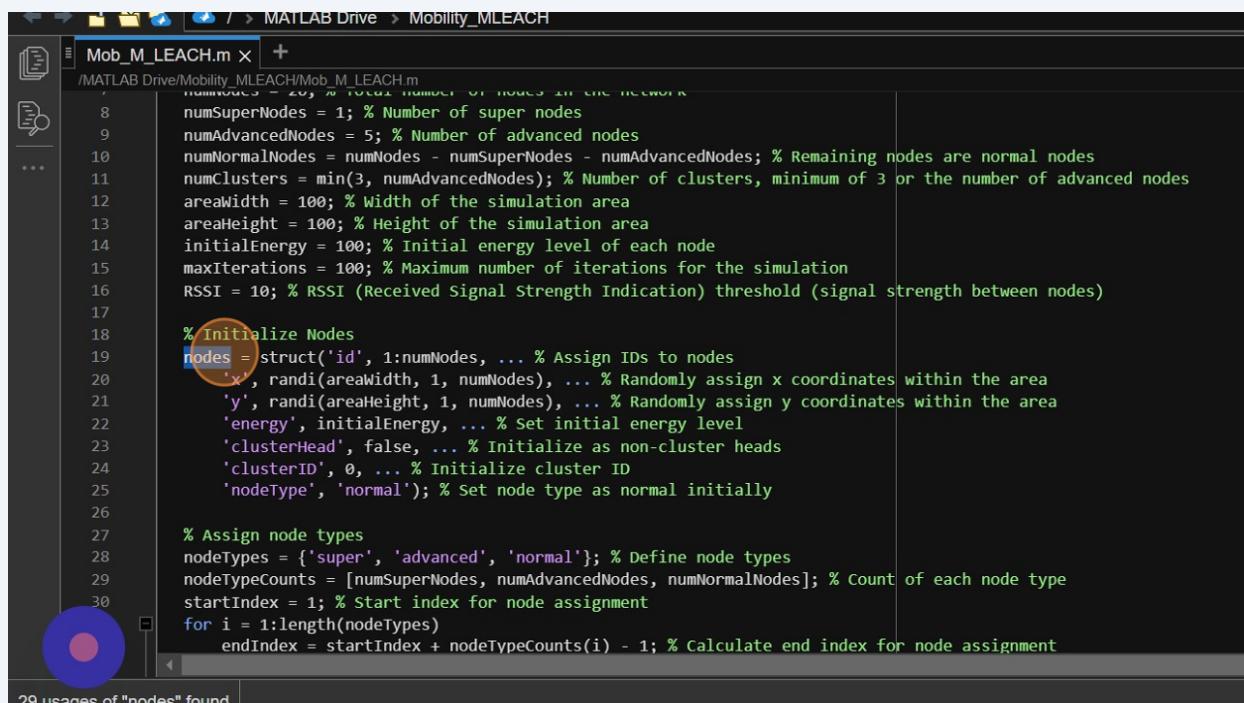
9 Now lets initialize the nodes as these are having a complete structure

```
% MATLAB Drive / Mobility_MLEACH / Mob_M_LEACH.m
% MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
%
% numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)
%
% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
    'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
    'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
    'energy', initialEnergy, ... % Set initial energy level
    'clusterHead', false, ... % Initialize as non-cluster heads
    'clusterID', 0, ... % Initialize cluster ID
    'nodeType', 'normal'); % Set node type as normal initially
%
% Assign node types
nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
nodeTypeCounts = [numSuperNodes, numAdvancedNodes, numNormalNodes]; % Count of each node type
startIndex = 1; % Start index for node assignment
for i = 1:length(nodeTypes)
    endIndex = startIndex + nodeTypeCounts(i) - 1; % Calculate end index for node assignment
```

10

Let's break down what each field of the nodes struct represents:

- id: This field represents the unique identifier for each node in the network. Nodes are assigned IDs from 1 to numNodes.
- x: This field represents the x-coordinate of each node's position within the simulation area. The x-coordinates are randomly assigned within the range [1, areaWidth].
- y: Similar to x, this field represents the y-coordinate of each node's position within the simulation area. The y-coordinates are randomly assigned within the range [1, areaHeight].
- energy: This field represents the current energy level of each node. All nodes start with the same initial energy level (initialEnergy).
- clusterHead: This field indicates whether a node is a cluster head (true) or not (false). Initially, all nodes are initialized as non-cluster heads.
- clusterID: This field represents the cluster ID to which each node belongs. Initially, the cluster ID is set to 0 for all nodes.
- .nodeType: This field specifies the type of each node. In this case, all nodes are initialized as "normal" nodes. However, based on the earlier definitions of numSuperNodes, numAdvancedNodes, and numNormalNodes, the node types can be differentiated as super, advanced, or normal.



```
Matlab Drive > Mobility_MLEACH > Mob_M_MEACH.m
Mob_M_MEACH.m X + / 
/MATLAB Drive/Mobility_MLEACH/Mob_M_MEACH.m
numNodes = 20; % Total number of nodes in the network
8 numSuperNodes = 1; % Number of super nodes
9 numAdvancedNodes = 5; % Number of advanced nodes
10 numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
11 numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
12 areaWidth = 100; % Width of the simulation area
13 areaHeight = 100; % Height of the simulation area
14 initialEnergy = 100; % Initial energy level of each node
15 maxIterations = 100; % Maximum number of iterations for the simulation
16 RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)
17
18 % Initialize Nodes
19 nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
20 'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
21 'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
22 'energy', initialEnergy, ... % Set initial energy level
23 'clusterHead', false, ... % Initialize as non-cluster heads
24 'clusterID', 0, ... % Initialize cluster ID
25 'nodeType', 'normal'); % Set node type as normal initially
26
27 % Assign node types
28 nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
29.nodeTypeCounts = [numSuperNodes, numAdvancedNodes, numNormalNodes]; % Count of each node type
30 startIndex = 1; % Start index for node assignment
for i = 1:length(nodeTypes)
    endIndex = startIndex + nodeTypeCounts(i) - 1; % Calculate end index for node assignment
```

11

• **nodeTypes**: This variable is a cell array containing the names of the different node types. In this case, it includes 'super', 'advanced', and 'normal'.

• **nodeTypeCounts**: This variable is an array containing the counts of each node type. It corresponds to the numbers of super nodes, advanced nodes, and normal nodes specified earlier.

• **startIndex**: This variable is used to keep track of the start index for assigning node types.

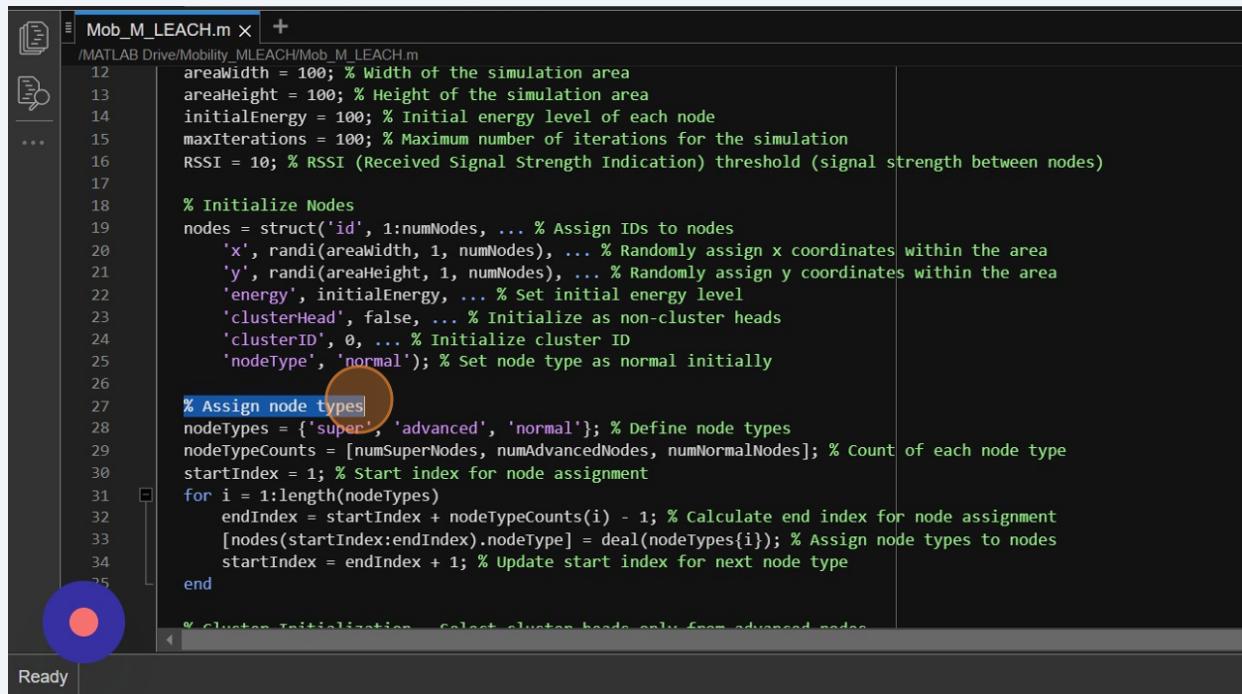
• **for loop**: This loop iterates over each node type.

• **endIndex calculation**: It calculates the end index for node assignment based on the start index and the count of nodes for the current type.

• **Node type assignment**: It assigns the current node type (`nodeTypes{i}`) to the nodes within the range from `startIndex` to `endIndex`.

• **Update startIndex**: It updates the `startIndex` for the next node type assignment by adding the count of nodes for the current type and 1 (since indices are 1-based in MATLAB).

This loop ensures that nodes are appropriately assigned the specified node types based on the counts provided.



```
MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
12 areaWidth = 100; % Width of the simulation area
13 areaHeight = 100; % Height of the simulation area
14 initialEnergy = 100; % Initial energy level of each node
15 maxIterations = 100; % Maximum number of iterations for the simulation
16 RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)
17
18 % Initialize Nodes
19 nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
20 'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
21 'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
22 'energy', initialEnergy, ... % Set initial energy level
23 'clusterHead', false, ... % Initialize as non-cluster heads
24 'clusterID', 0, ... % Initialize cluster ID
25 'nodeType', 'normal'); % Set node type as normal initially
26
27 % Assign node types
28 nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
29.nodeTypeCounts = [numSuperNodes, numAdvancedNodes, numNormalNodes]; % Count of each node type
30 startIndex = 1; % Start index for node assignment
31 for i = 1:length(nodeTypes)
32     endIndex = startIndex + nodeTypeCounts(i) - 1; % Calculate end index for node assignment
33     [nodes(startIndex:endIndex).nodeType] = deal(nodeTypes{i}); % Assign node types to nodes
34     startIndex = endIndex + 1; % Update start index for next node type
35 end
36
% Cluster Initialization - Select cluster heads only from advanced nodes
```

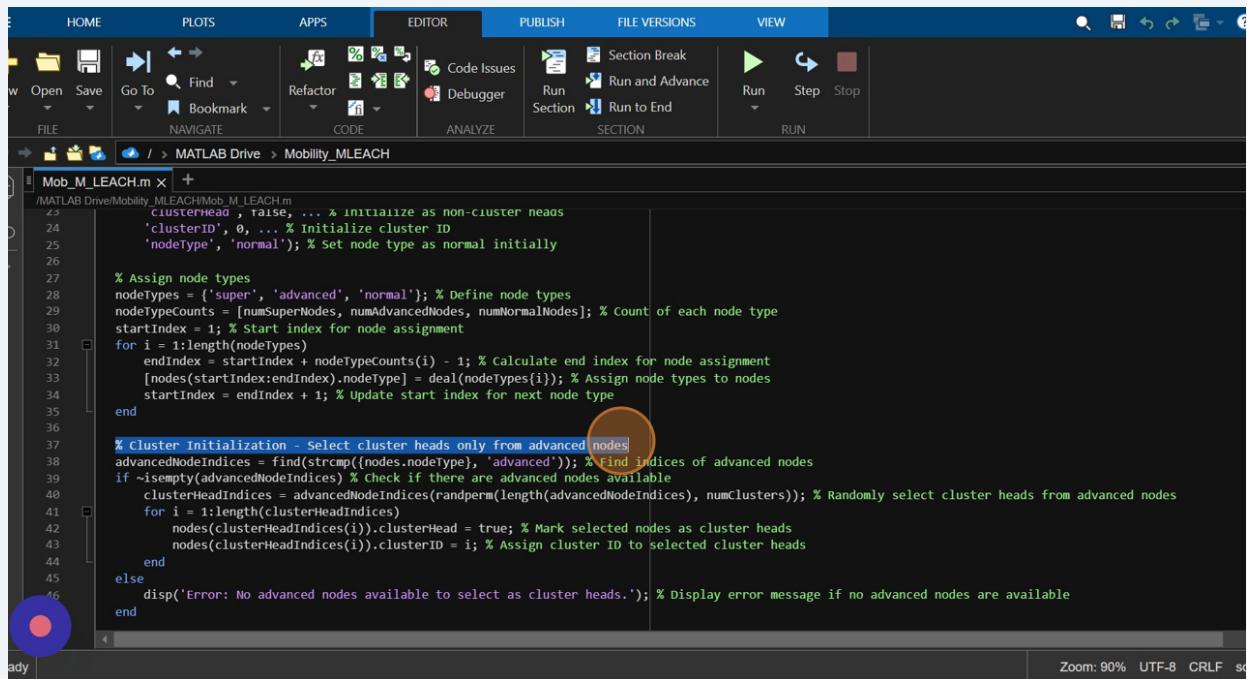
- advancedNodeIndices: This line finds the indices of nodes that have the node type 'advanced'.

• **Check for availability of advanced nodes:** The if ~isempty(advancedNodeIndices) condition checks if there are any advanced nodes available in the network. If there are no advanced nodes (isempty returns true), it displays an error message.

• **Select cluster heads from advanced nodes:** If there are advanced nodes available, the code proceeds to select cluster heads from them. It does so by randomly permuting the indices of advanced nodes (randperm(length(advancedNodeIndices), numClusters)) and selecting numClusters number of them.

• **Mark selected nodes as cluster heads:** For each selected cluster head, it sets the clusterHead field to true and assigns a unique clusterID to it. The clusterID is assigned sequentially starting from 1.

This section ensures that cluster heads are selected only from advanced nodes and assigns them the appropriate properties. If no advanced nodes are available, it notifies the user about the error.



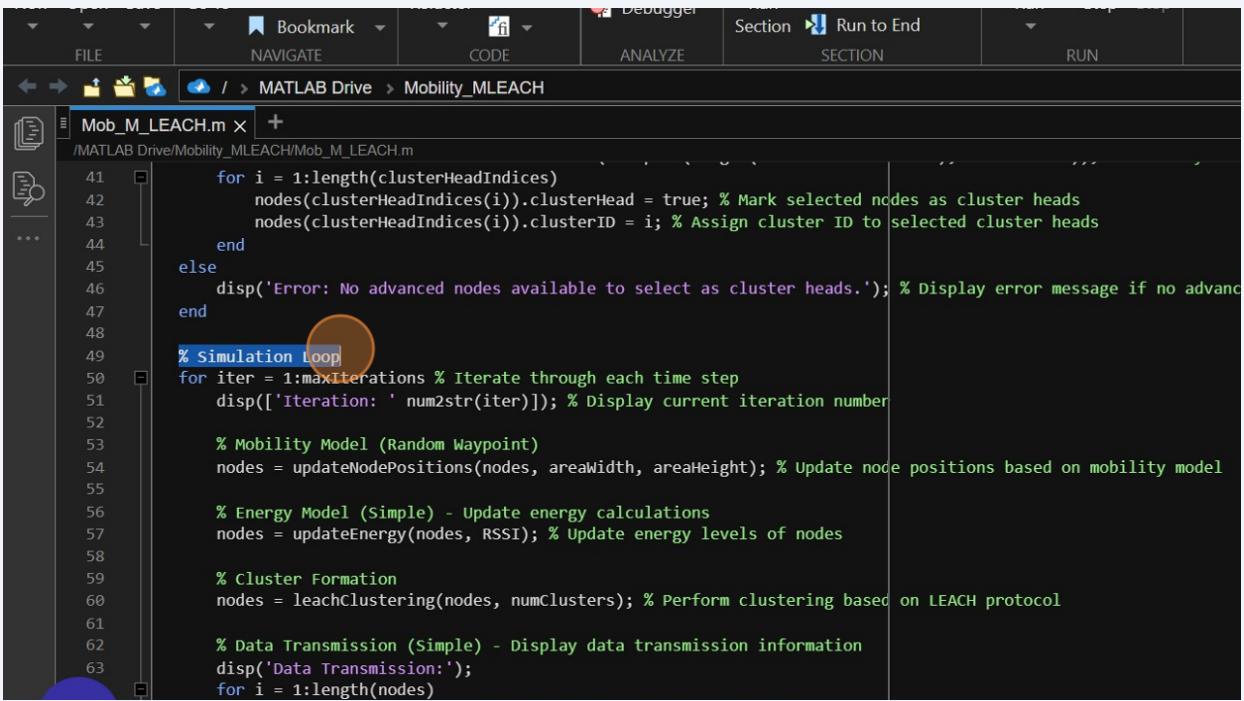
```

Mob_M_LEACH.m
MATLAB Drive\Mobility\MLEACH\Mob_M_LEACH.m
1 % Initialize as non-cluster heads
2 clusterhead = false; ...
3 % Initialize cluster ID
4 clusterID = 0;
5 % Set node type as normal initially
6 nodetype = 'normal';
7
8 % Assign node types
9 nodetypes = {'super', 'advanced', 'normal'}; % Define node types
10.nodeTypeCounts = [numSuperNodes, numAdvancedNodes, numNormalNodes]; % Count of each node type
11 startIndex = 1; % Start index for node assignment
12 for i = 1:length(nodetypes)
13     endIndex = startIndex + nodeTypeCounts(i) - 1; % Calculate end index for node assignment
14     [nodes(startIndex:endIndex).nodeType] = deal(nodetypes{i}); % Assign node types to nodes
15     startIndex = endIndex + 1; % Update start index for next node type
16 end
17
18 % Cluster Initialization - Select cluster heads only from advanced nodes
19 advancedNodeIndices = find(strcmp(nodes.nodeType, 'advanced')); % Find indices of advanced nodes
20 if ~isempty(advancedNodeIndices) % Check if there are advanced nodes available
21     clusterHeadIndices = advancedNodeIndices(randperm(length(advancedNodeIndices), numClusters)); % Randomly select cluster heads from advanced nodes
22     for i = 1:length(clusterHeadIndices)
23         nodes(clusterHeadIndices(i)).clusterHead = true; % Mark selected nodes as cluster heads
24         nodes(clusterHeadIndices(i)).clusterID = i; % Assign cluster ID to selected cluster heads
25     end
26 else
27     disp('Error: No advanced nodes available to select as cluster heads.'); % Display error message if no advanced nodes are available
28 end

```

13

Now lets see the simulation loop
This is the loop where all of the simulation starts

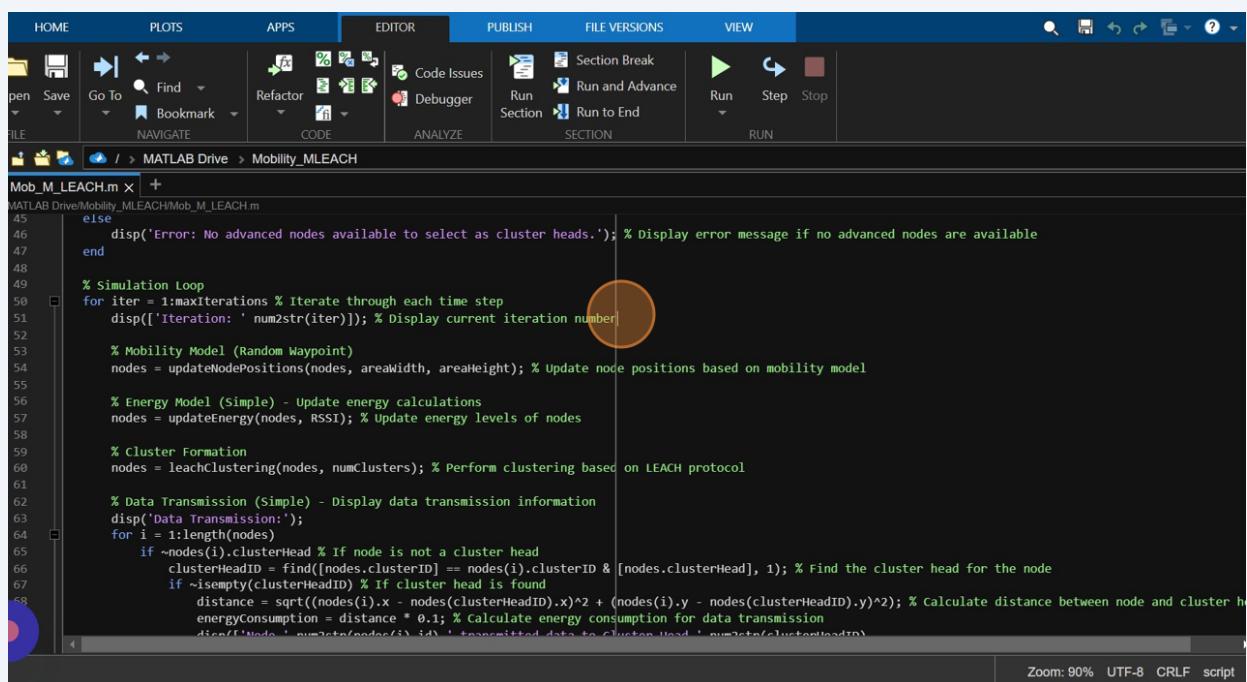


```
FILE Bookmark fi Debugger
NAVIGATE CODE ANALYZE Section Run to End SECTION RUN
< > / > MATLAB Drive > Mobility_MLEACH
Mob_M_LEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
41 for i = 1:length(clusterHeadIndices)
42     nodes(clusterHeadIndices(i)).clusterHead = true; % Mark selected nodes as cluster heads
43     nodes(clusterHeadIndices(i)).clusterID = i; % Assign cluster ID to selected cluster heads
44 end
45 else
46     disp('Error: No advanced nodes available to select as cluster heads.'); % Display error message if no advanced
47 end
48
49 % Simulation Loop
50 for iter = 1:maxIterations % Iterate through each time step
51     disp(['Iteration: ' num2str(iter)]); % Display current iteration number
52
53     % Mobility Model (Random Waypoint)
54     nodes = updateNodePositions(nodes, areaWidth, areaHeight); % Update node positions based on mobility model
55
56     % Energy Model (Simple) - Update energy calculations
57     nodes = updateEnergy(nodes, RSSI); % Update energy levels of nodes
58
59     % Cluster Formation
60     nodes = leachClustering(nodes, numClusters); % Perform clustering based on LEACH protocol
61
62     % Data Transmission (Simple) - Display data transmission information
63     disp('Data Transmission:');
64     for i = 1:length(nodes)
```

14

Here's an explanation of what happens in this loop:

- **for loop:** Iterates from 1 to maxIterations, which represents the maximum number of iterations for the simulation.
- **Display iteration number:** Each iteration, it displays the current iteration number using `disp(['Iteration: ' num2str(iter)])`. This helps in monitoring the progress of the simulation.
- **Mobility Model (Random Waypoint):** It calls the `updateNodePositions` function to update the positions of nodes based on a mobility model. In this case, the Random Waypoint model is used. This function is responsible for simulating the movement of nodes within the simulation area. The updated node positions are stored back in the `nodes` structure



```
HOME PLOTS APPS EDITOR PUBLISH FILE VERSIONS VIEW
File Save Go To Find Refactor Code Issues Debugger Run Section Break Run and Advance Run Step Stop
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > Mobility_MLEACH > Mob_M_LEACH.m
Mob_M_LEACH.m + MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
45
46     disp('Error: No advanced nodes available to select as cluster heads.');// Display error message if no advanced nodes are available
47 end
48
49 % Simulation Loop
50 for iter = 1:maxIterations % Iterate through each time step
51     disp(['Iteration: ' num2str(iter)]); // Display current iteration number
52
53     % Mobility Model (Random Waypoint)
54     nodes = updateNodePositions(nodes, areawidth, areaheight); // Update node positions based on mobility model
55
56     % Energy Model (Simple) - Update energy calculations
57     nodes = updateEnergy(nodes, RSSI); // Update energy levels of nodes
58
59     % Cluster Formation
60     nodes = leachClustering(nodes, numClusters); // Perform clustering based on LEACH protocol
61
62     % Data Transmission (Simple) - Display data transmission information
63     disp('Data Transmission:');
64     for i = 1:length(nodes)
65         if ~nodes(i).clusterHead % If node is not a cluster head
66             clusterHeadID = find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1); // Find the cluster head for the node
67             if ~isempty(clusterHeadID) % If cluster head is found
68                 distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2); // Calculate distance between node and cluster head
69                 energyConsumption = distance * 0.1; // Calculate energy consumption for data transmission
70                 distance = distance * numNodes(nodes(i).id) * transmittedData * numNodes(clusterhead);
71             end
72         end
73     end
74 end
```

Zoom: 90% UTF-8 CRLF script

15

• Input arguments:

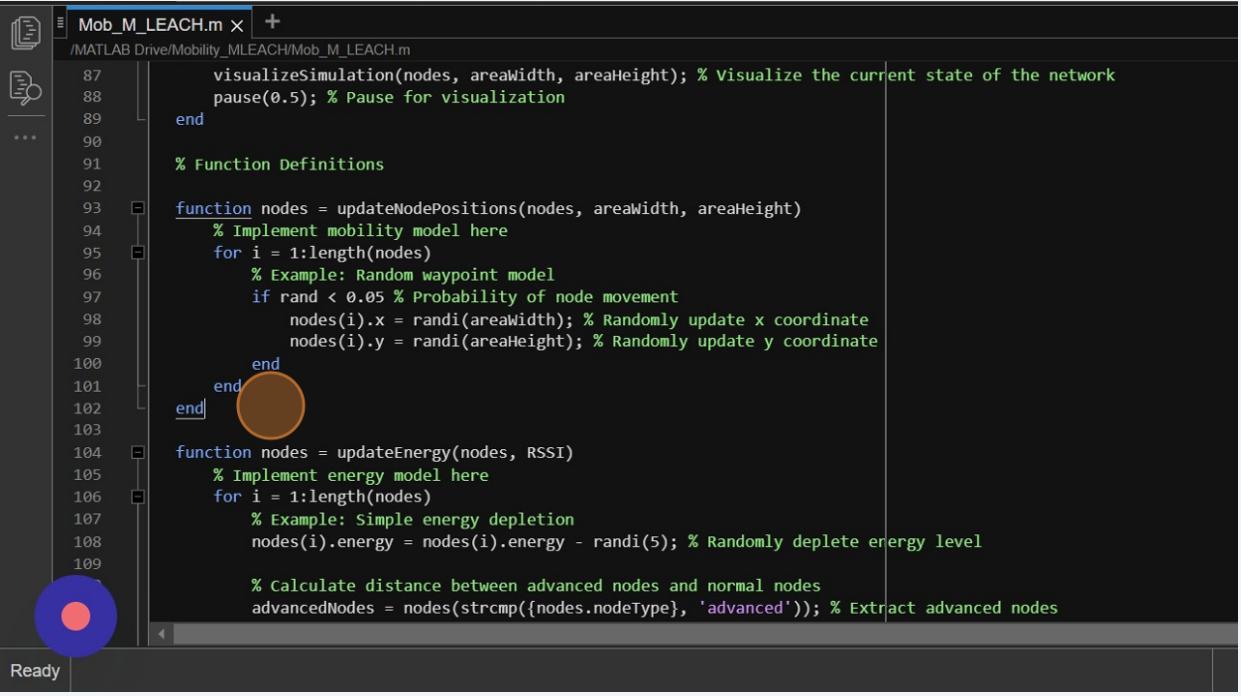
- nodes: A structure array containing information about the nodes in the network.
- areaWidth: The width of the simulation area.
- areaHeight: The height of the simulation area.

• Output:

- Updated nodes structure array with new node positions.

• Function body:

- **Mobility model implementation:** This function iterates through each node in the nodes array.
 - **Random waypoint model:** It simulates node movement using a random waypoint model. In each iteration, for each node, it checks if a randomly generated value between 0 and 1 (rand) is less than 0.05, indicating a probability of 5% for node movement (this probability value can be adjusted according to the specific mobility model being simulated).
 - If the condition is met, it updates the node's position (x and y coordinates) by randomly selecting new coordinates within the simulation area (areaWidth and areaHeight) using randi function.



```
Mob_M_LEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
87     visualizeSimulation(nodes, areaWidth, areaHeight); % Visualize the current state of the network
88     pause(0.5); % Pause for visualization
89 end
90
91 % Function Definitions
92
93 function nodes = updateNodePositions(nodes, areaWidth, areaHeight)
94     % Implement mobility model here
95     for i = 1:length(nodes)
96         % Example: Random waypoint model
97         if rand < 0.05 % Probability of node movement
98             nodes(i).x = randi(areaWidth); % Randomly update x coordinate
99             nodes(i).y = randi(areaHeight); % Randomly update y coordinate
100        end
101    end
102 end
103
104 function nodes = updateEnergy(nodes, RSSI)
105     % Implement energy model here
106     for i = 1:length(nodes)
107         % Example: Simple energy depletion
108         nodes(i).energy = nodes(i).energy - randi(5); % Randomly deplete energy level
109
110         % Calculate distance between advanced nodes and normal nodes
111         advancedNodes = nodes(strcmp({nodes.nodeType}, 'advanced')); % Extract advanced nodes
```

16

• Input arguments:

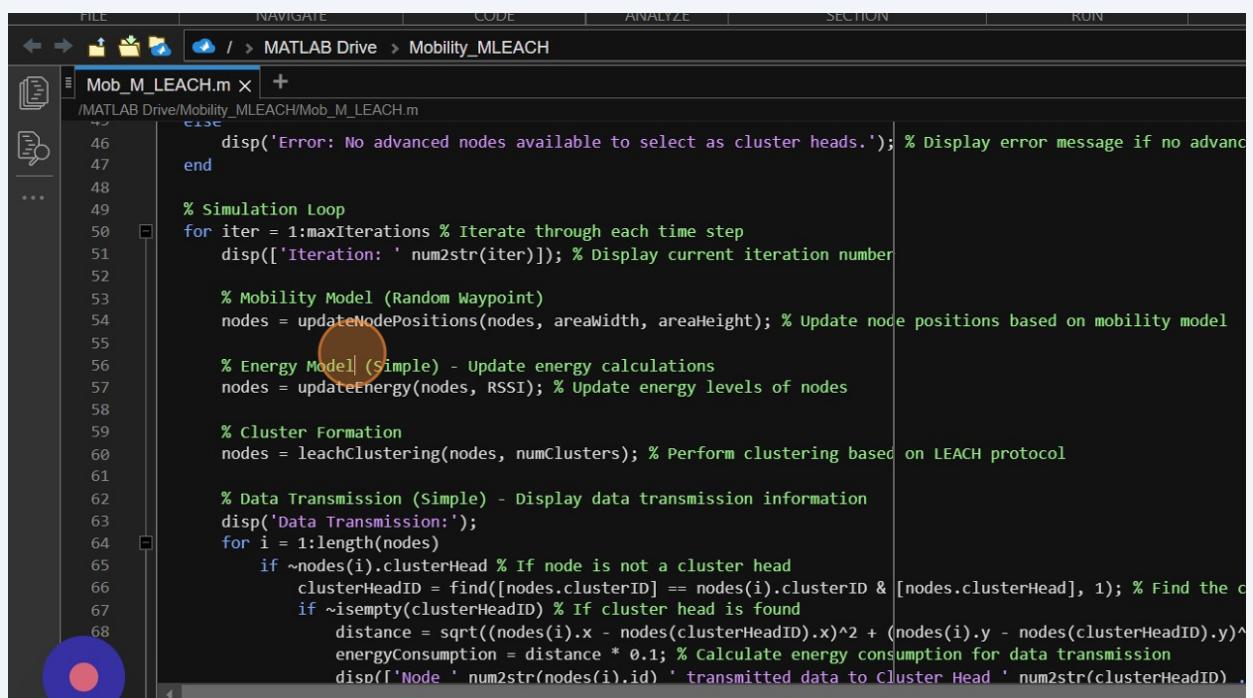
- nodes: The structure array containing information about the nodes in the network.
- RSSI: The Received Signal Strength Indication threshold, which is a measure of the signal strength between nodes.

• Function body:

- **Energy model implementation:** This function iterates through each node in the nodes array.

- **Simple energy depletion:** It simulates energy depletion by randomly reducing each node's energy level (nodes(i).energy) by a random integer value between 1 and 5 (inclusive) using randi(5). This represents a basic model of energy consumption.

- **Distance calculation and energy model update:** For each normal node, it calculates the distance between it and all advanced nodes. If the distance is less than the RSSI threshold, it calculates several energy-related parameters such as Emob, Ec, Egrp, and Enet. However, these calculations are currently placeholders (0) and need to be implemented based on the specific energy model and routing protocol being simulated.



```
FILE NAVIGATE CODE ANALYZE SECTION RUN
MATLAB Drive / > MATLAB Drive > Mobility_MLEACH
Mob_M_MEACH.m X +
/MATLAB Drive/Mobility_MLEACH/Mob_M_MEACH.m
46 disp('Error: No advanced nodes available to select as cluster heads.');?>
47 end
48
49 % Simulation Loop
50 for iter = 1:maxIterations % Iterate through each time step
51     disp(['Iteration: ' num2str(iter)]); % Display current iteration number
52
53     % Mobility Model (Random Waypoint)
54     nodes = updateNodePositions(nodes, areaWidth, areaHeight); % Update node positions based on mobility model
55
56     % Energy Model (Simple) - Update energy calculations
57     nodes = updateEnergy(nodes, RSSI); % Update energy levels of nodes
58
59     % Cluster Formation
60     nodes = leachClustering(nodes, numClusters); % Perform clustering based on LEACH protocol
61
62     % Data Transmission (Simple) - Display data transmission information
63     disp('Data Transmission:');
64     for i = 1:length(nodes)
65         if ~nodes(i).clusterHead % If node is not a cluster head
66             clusterHeadID = find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1); % Find the cluster head
67             if ~isempty(clusterHeadID) % If cluster head is found
68                 distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2);
69                 energyConsumption = distance * 0.1; % Calculate energy consumption for data transmission
70                 disp(['Node ' num2str(nodes(i).id) ' transmitted data to Cluster Head ' num2str(clusterHeadID) '.']);
71             end
72         end
73     end
74 end
```

- **Input arguments:**

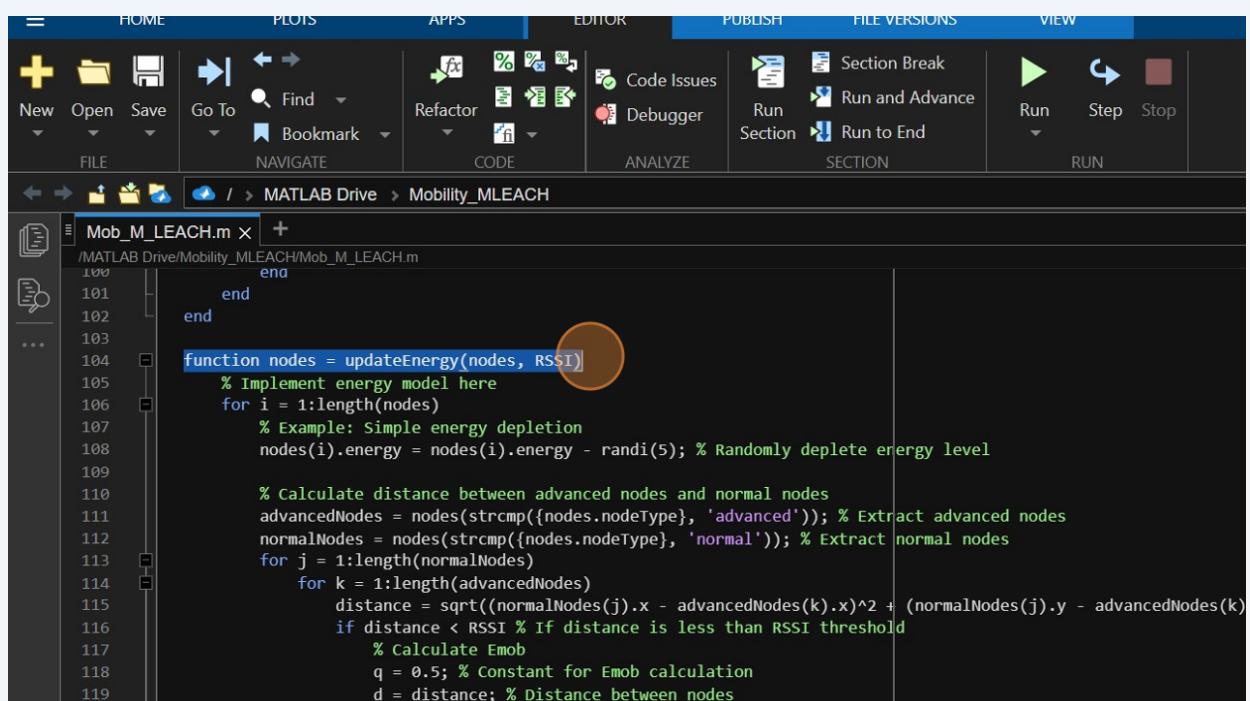
- nodes: The structure array containing information about the nodes in the network.
- RSSI: The Received Signal Strength Indication threshold, which is a measure of the signal strength between nodes.

- **Function body:**

- **Energy model implementation:** This function iterates through each node in the nodes array.

- **Simple energy depletion:** It simulates energy depletion by randomly reducing each node's energy level (nodes(i).energy) by a random integer value between 1 and 5 (inclusive) using randi(5). This represents a basic model of energy consumption.

- **Distance calculation and energy model update:** For each normal node, it calculates the distance between it and all advanced nodes. If the distance is less than the RSSI threshold, it calculates several energy-related parameters such as Emob, Ec, Egrp, and Enet. However, these calculations are currently placeholders (0) and need to be implemented based on the specific energy model and routing protocol being simulated.

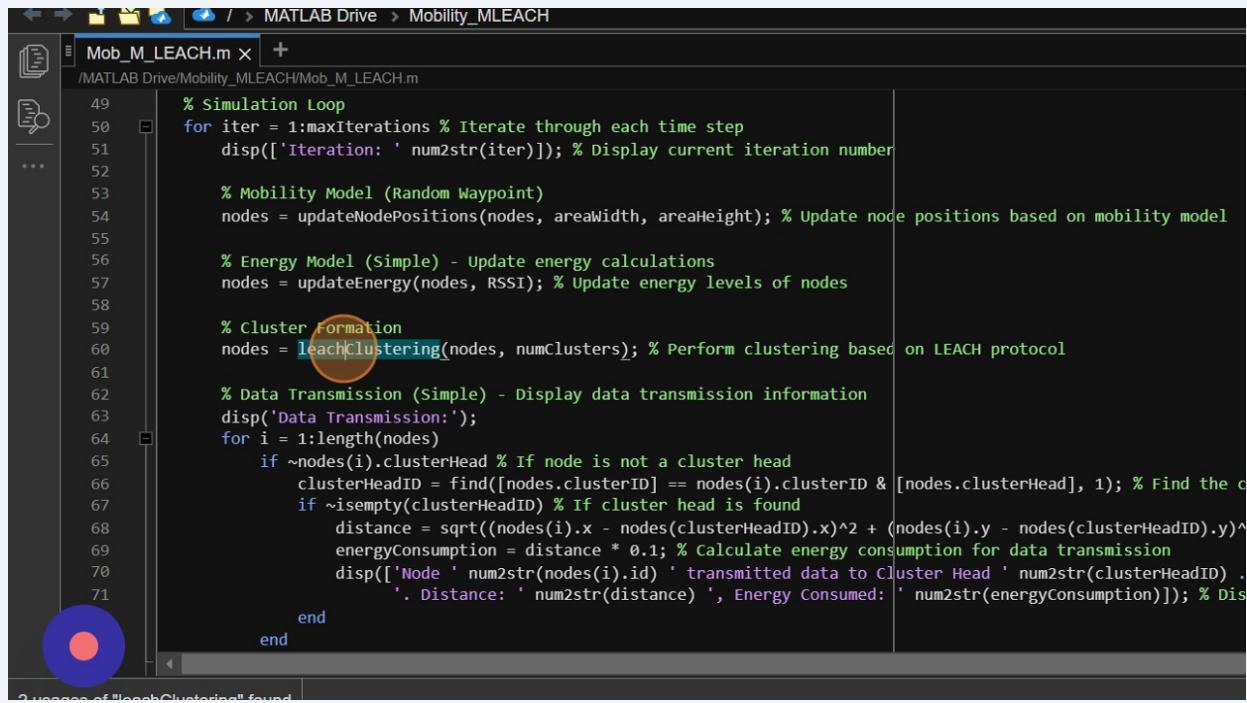


```

Mob_M_LEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
100
101 end
102 end
103
104 function nodes = updateEnergy(nodes, RSSI)
105 % Implement energy model here
106 for i = 1:length(nodes)
107 % Example: Simple energy depletion
108 nodes(i).energy = nodes(i).energy - randi(5); % Randomly deplete energy level
109
110 % Calculate distance between advanced nodes and normal nodes
111 advancedNodes = nodes(strcmp({nodes.nodeType}, 'advanced')); % Extract advanced nodes
112 normalNodes = nodes(strcmp({nodes.nodeType}, 'normal')); % Extract normal nodes
113 for j = 1:length(normalNodes)
114 for k = 1:length(advancedNodes)
115 distance = sqrt((normalNodes(j).x - advancedNodes(k).x)^2 + (normalNodes(j).y - advancedNodes(k).y)^2);
116 if distance < RSSI % If distance is less than RSSI threshold
117 % Calculate Emob
118 q = 0.5; % Constant for Emob calculation
119 d = distance; % Distance between nodes

```

18 This function is to create clusters



```
MATLAB Drive > Mobility_MLEACH
Mob_M_LEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m

49 % Simulation Loop
50 for iter = 1:maxIterations % Iterate through each time step
51     disp(['Iteration: ' num2str(iter)]); % Display current iteration number
52
53 % Mobility Model (Random Waypoint)
54 nodes = updateNodePositions(nodes, areaWidth, areaHeight); % Update node positions based on mobility model
55
56 % Energy Model (Simple) - Update energy calculations
57 nodes = updateEnergy(nodes, RSSI); % Update energy levels of nodes
58
59 % Cluster Formation
60 nodes = leachClustering(nodes, numClusters); % Perform clustering based on LEACH protocol
61
62 % Data Transmission (Simple) - Display data transmission information
63 disp('Data Transmission:');
64 for i = 1:length(nodes)
65     if ~nodes(i).clusterHead % If node is not a cluster head
66         clusterHeadID = find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1); % Find the c
67         if ~isempty(clusterHeadID) % If cluster head is found
68             distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2);
69             energyConsumption = distance * 0.1; % Calculate energy consumption for data transmission
70             disp(['Node ' num2str(nodes(i).id) ' transmitted data to Cluster Head ' num2str(clusterHeadID) '.']);
71             disp(['Distance: ' num2str(distance) ', Energy Consumed: ' num2str(energyConsumption)]);
72         end
73     end
74 end

2 usages of "leachClustering" found
```

19

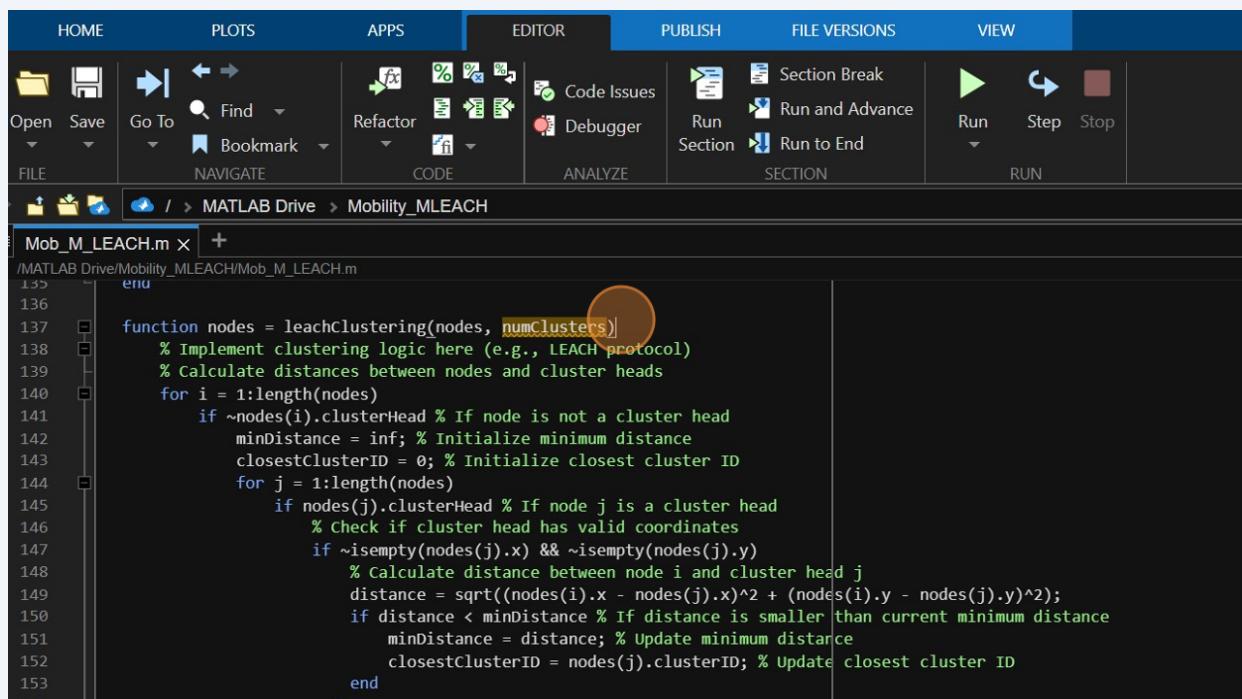
- **Input arguments:**

- nodes: The structure array containing information about the nodes in the network.
- numClusters: The number of clusters to be formed.

- **Function body:**

- **Cluster formation logic:** This function implements the clustering logic, specifically based on the LEACH protocol or a similar clustering algorithm.

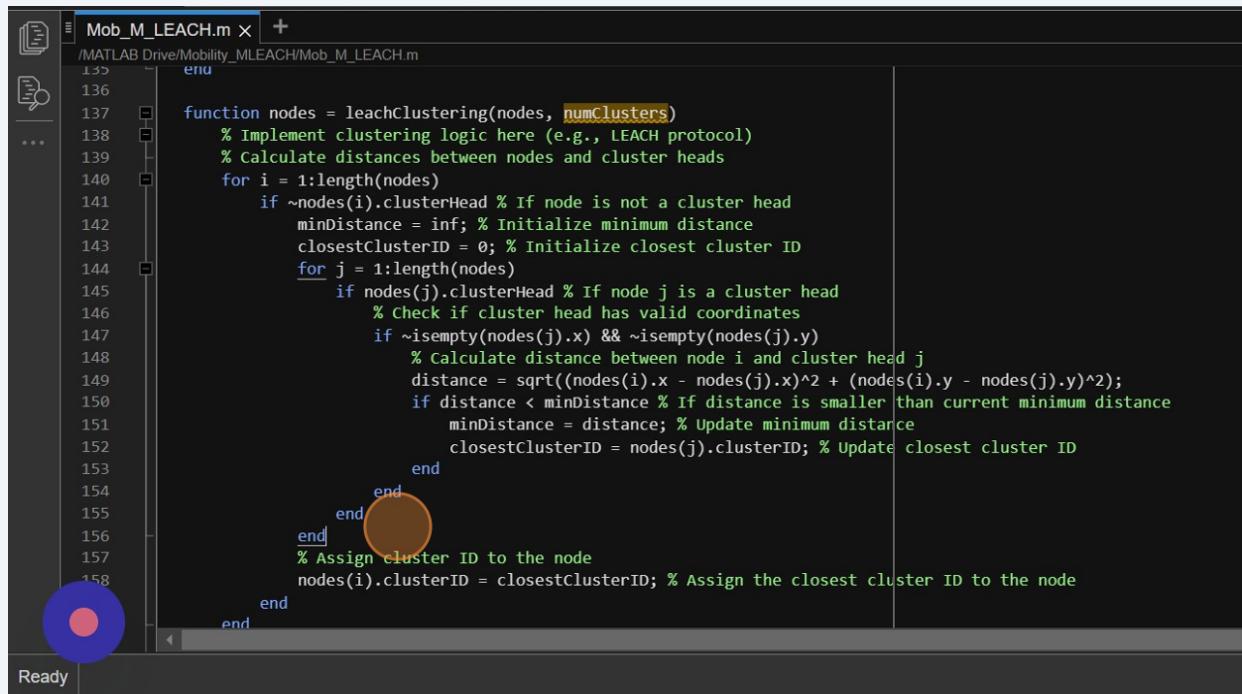
- **Calculate distances between nodes and cluster heads:** It iterates through each node in the nodes array. For each node that is not already a cluster head (~nodes(i).clusterHead), it calculates the distance to each cluster head and assigns the node to the cluster whose head is closest.



```
function nodes = leachClustering(nodes, numClusters)
    % Implement clustering logic here (e.g., LEACH protocol)
    % Calculate distances between nodes and cluster heads
    for i = 1:length(nodes)
        if ~nodes(i).clusterHead % If node is not a cluster head
            minDistance = inf; % Initialize minimum distance
            closestClusterID = 0; % Initialize closest cluster ID
            for j = 1:length(nodes)
                if nodes(j).clusterHead % If node j is a cluster head
                    % Check if cluster head has valid coordinates
                    if ~isempty(nodes(j).x) && ~isempty(nodes(j).y)
                        % calculate distance between node i and cluster head j
                        distance = sqrt((nodes(i).x - nodes(j).x)^2 + (nodes(i).y - nodes(j).y)^2);
                        if distance < minDistance % If distance is smaller than current minimum distance
                            minDistance = distance; % Update minimum distance
                            closestClusterID = nodes(j).clusterID; % Update closest cluster ID
                        end
                    end
                end
            end
        end
    end
end
```

20

- **Assign cluster ID:** It assigns the ID of the closest cluster to the node (`nodes(i).clusterID`). This ID will be used later for data transmission and other cluster-related operations.



```
Mob_M_LEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
135     end
136
137 function nodes = leachClustering(nodes, numClusters)
138     % Implement clustering logic here (e.g., LEACH protocol)
139     % Calculate distances between nodes and cluster heads
140     for i = 1:length(nodes)
141         if ~nodes(i).clusterHead % If node is not a cluster head
142             minDistance = inf; % Initialize minimum distance
143             closestClusterID = 0; % Initialize closest cluster ID
144             for j = 1:length(nodes)
145                 if nodes(j).clusterHead % If node j is a cluster head
146                     % Check if cluster head has valid coordinates
147                     if ~isempty(nodes(j).x) && ~isempty(nodes(j).y)
148                         % Calculate distance between node i and cluster head j
149                         distance = sqrt((nodes(i).x - nodes(j).x)^2 + (nodes(i).y - nodes(j).y)^2);
150                         if distance < minDistance % If distance is smaller than current minimum distance
151                             minDistance = distance; % Update minimum distance
152                             closestClusterID = nodes(j).clusterID; % Update closest cluster ID
153                         end
154                     end
155                 end
156             end
157             % Assign cluster ID to the node
158             nodes(i).clusterID = closestClusterID; % Assign the closest cluster ID to the node
159         end
160     end
161 end
```

21

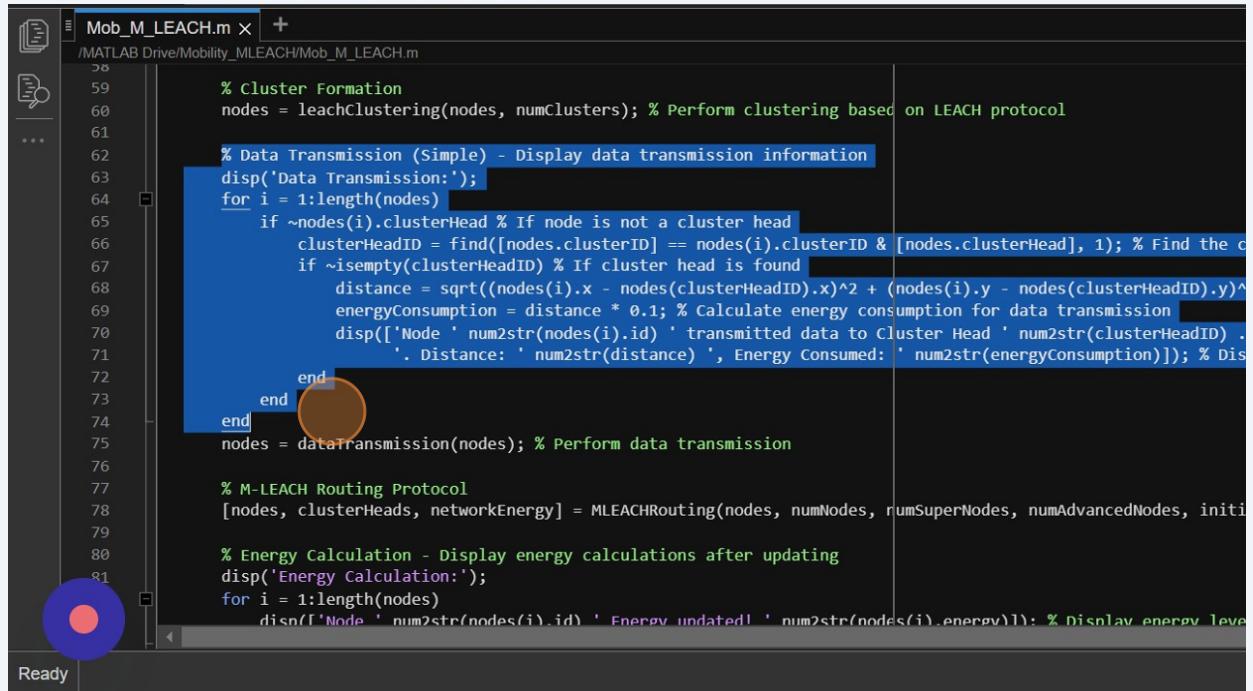
- **Data transmission information display:**

- It first displays a message indicating data transmission is taking place within the network (`disp('Data Transmission:');`).
- It iterates through each node in the nodes array.
- For each non-cluster head node (~`nodes(i).clusterHead`), it finds the cluster head corresponding to the node's cluster (`clusterHeadID`) using `find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1)`.
- If a cluster head is found (~`isempty(clusterHeadID)`), it calculates the distance between the node and the cluster head using the Euclidean distance formula.
- It then calculates the energy consumption for data transmission based on the distance (`energyConsumption = distance * 0.1`), assuming a simple energy model.
- Finally, it displays the data transmission information including the node ID, cluster head ID, distance, and energy consumed.

- **Data transmission function call:**

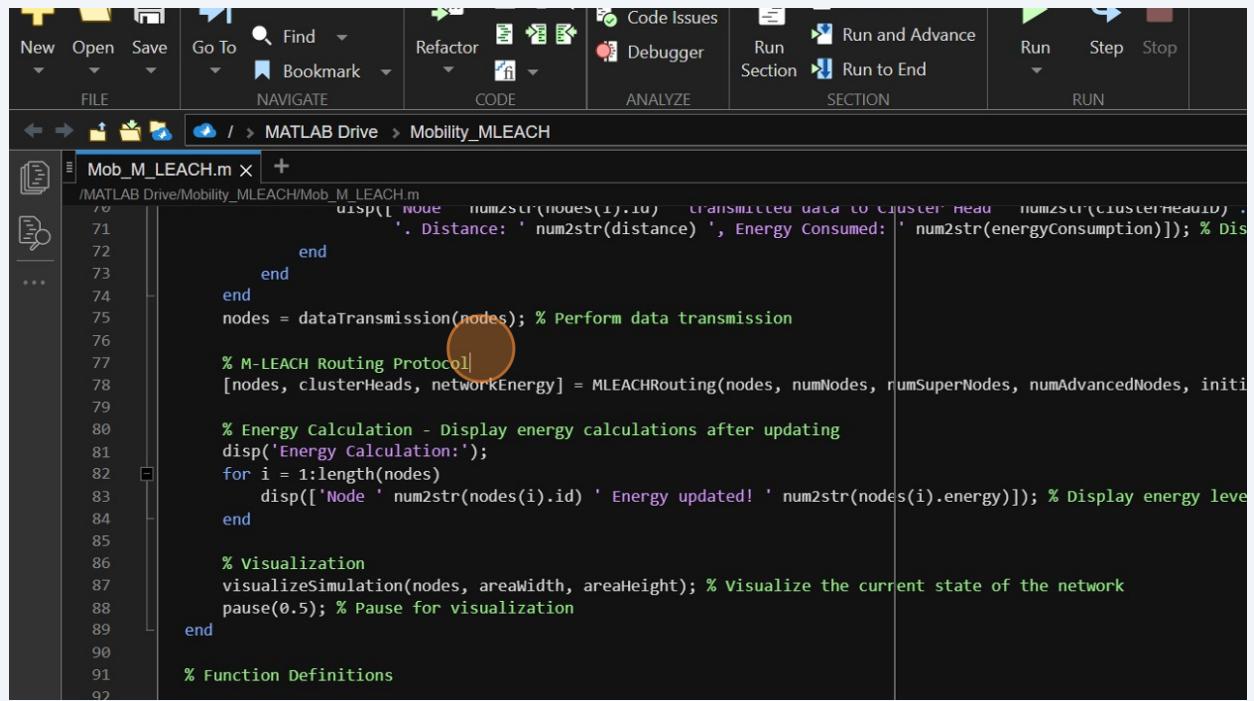
- After displaying the data transmission information, it calls the `dataTransmission` function to perform data transmission operations, which may involve updating energy levels or other relevant parameters for nodes in the network.

Overall, this code segment provides a simple display of data transmission information and may perform further data transmission operations through the `dataTransmission` function call.



```
Mob_M_LEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
58
59     % Cluster Formation
60     nodes = leachClustering(nodes, numClusters); % Perform clustering based on LEACH protocol
61
62     % Data Transmission (Simple) - Display data transmission information
63     disp('Data Transmission:');
64     for i = 1:length(nodes)
65         if ~nodes(i).clusterHead % If node is not a cluster head
66             clusterHeadID = find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1); % Find the c
67             if ~isempty(clusterHeadID) % If cluster head is found
68                 distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2);
69                 energyConsumption = distance * 0.1; % Calculate energy consumption for data transmission
70                 disp(['Node ' num2str(nodes(i).id) ' transmitted data to Cluster Head ' num2str(clusterHeadID) '.
71                         ' Distance: ' num2str(distance) ', Energy Consumed: ' num2str(energyConsumption)]); % Dis
72             end
73         end
74     end
75
76     nodes = dataTransmission(nodes); % Perform data transmission
77
78     % M-LEACH Routing Protocol
79     [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes, initi
80
81     % Energy Calculation - Display energy calculations after updating
82     disp('Energy Calculation:');
83     for i = 1:length(nodes)
84         disp(['Node ' num2str(nodes(i).id) ' Energy updated! ' num2str(nodes(i).energy)]); % Display energy level
```

22 Here MLEACH Routing is called and passed some perimeters



```
1 % M-LEACH Routing Protocol
2 [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes, initilizationType);
3
4 % Energy Calculation - Display energy calculations after updating
5 disp('Energy Calculation:');
6 for i = 1:length(nodes)
7     disp(['Node ' num2str(nodes(i).id) ' Energy updated! ' num2str(nodes(i).energy)]);
8 end
9
10 % Visualization
11 visualizeSimulation(nodes, areaWidth, areaHeight);
12 pause(0.5); % Pause for visualization
13
14 % Function Definitions
```

23

Explanation:

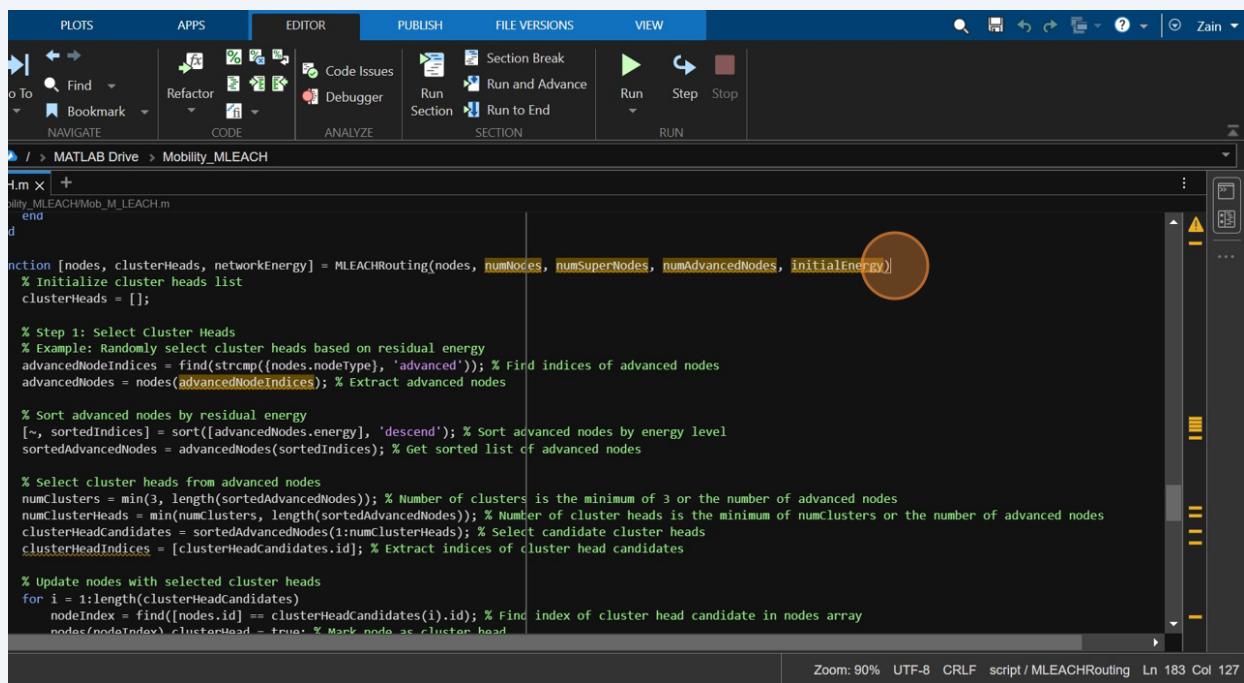
• Input arguments:

- nodes: The structure array containing information about the nodes in the network.
- numNodes: Total number of nodes in the network.
- numSuperNodes: Number of super nodes in the network.
- numAdvancedNodes: Number of advanced nodes in the network.
- initialEnergy: Initial energy level of each node.

• Output arguments:

- nodes: Updated structure array containing information about the nodes in the network.
- clusterHeads: List of cluster heads selected by the M-LEACH routing protocol.
- networkEnergy: Total network energy after energy consumption due to mobility and data transmission.

•



The screenshot shows the MATLAB Editor interface with the following details:

- Toolbar:** PLOTS, APPS, EDITOR (highlighted), PUBLISH, FILE VERSIONS, VIEW.
- Code Area:** The script file `MLEACHRouting.m` is open. A red circle highlights the line of code where the cluster heads list is initialized: `clusterHeads = [];`
- Bottom Status Bar:** Zoom: 90%, UTF-8, CRLF, script / MLEACHRouting, Ln 183 Col 127.

```
function [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes, initialEnergy)
% Initialize cluster heads list
clusterHeads = [];

% Step 1: Select Cluster Heads
% Example: Randomly select cluster heads based on residual energy
advancedNodeIndices = find(strcmp({nodes.nodeType}, 'advanced')) % Find indices of advanced nodes
advancedNodes = nodes(advancedNodeIndices); % Extract advanced nodes

% Sort advanced nodes by residual energy
[~, sortedIndices] = sort([-advancedNodes.energy], 'descend'); % Sort advanced nodes by energy level
sortedAdvancedNodes = advancedNodes(sortedIndices); % Get sorted list of advanced nodes

% Select cluster heads from advanced nodes
numClusters = min(3, length(sortedAdvancedNodes)); % Number of clusters is the minimum of 3 or the number of advanced nodes
numClusterHeads = min(numClusters, length(sortedAdvancedNodes)); % Number of cluster heads is the minimum of numClusters or the number of advanced nodes
clusterHeadCandidates = sortedAdvancedNodes(1:numClusterHeads); % Select candidate cluster heads
clusterHeadIndices = [clusterHeadCandidates.id]; % Extract indices of cluster head candidates

% Update nodes with selected cluster heads
for i = 1:length(clusterHeadCandidates)
    nodeIndex = find((nodes.id) == clusterHeadCandidates(i).id); % Find index of cluster head candidate in nodes array
    nodes(nodeIndex).clusterHead = true; % Mark node as cluster head
end
```

24

• Function body:

• Step 1: Select Cluster Heads:

- It selects cluster heads based on residual energy. In this example, it randomly selects cluster heads from the advanced nodes based on their energy levels.

The screenshot shows the MATLAB IDE interface with the following tabs at the top: AVIGATE, CODE, ANALYZE, SECTION, and RUN. The current file is ACH/Mob_M_LEACH.m. The code in the editor is as follows:

```
[nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes, initialEnergy)
% Initialize cluster heads list
clusterHeads = [];

% Step 1: Select Cluster Heads
% Example: Randomly select cluster heads based on residual energy
advancedNodeIndices = find(strcmp({nodes.nodeType}, 'advanced')) % Find indices of advanced nodes
advancedNodes = nodes(advancedNodeIndices); % Extract advanced nodes

% Sort advanced nodes by residual energy
sortedIndices = sort([advancedNodes.energy], 'descend'); % Sort advanced nodes by energy level
sortedAdvancedNodes = advancedNodes(sortedIndices); % Get sorted list of advanced nodes

% Select cluster heads from advanced nodes
numClusters = min(3, length(sortedAdvancedNodes)); % Number of clusters is the minimum of 3 or the number of advanced nodes
numClusterHeads = min(numClusters, length(sortedAdvancedNodes)); % Number of cluster heads is the minimum of numClusters or the number of advanced nodes
clusterHeadCandidates = sortedAdvancedNodes(1:numClusterHeads); % Select candidate cluster heads
clusterHeadIndices = [clusterHeadCandidates.id]; % Extract indices of cluster head candidates

% Update nodes with selected cluster heads
for i = 1:length(clusterHeadCandidates)
    nodeIndex = find([nodes.id] == clusterHeadCandidates(i).id); % Find index of cluster head candidate in nodes array
    nodes(nodeIndex).clusterHead = true; % Mark node as cluster head
end
```

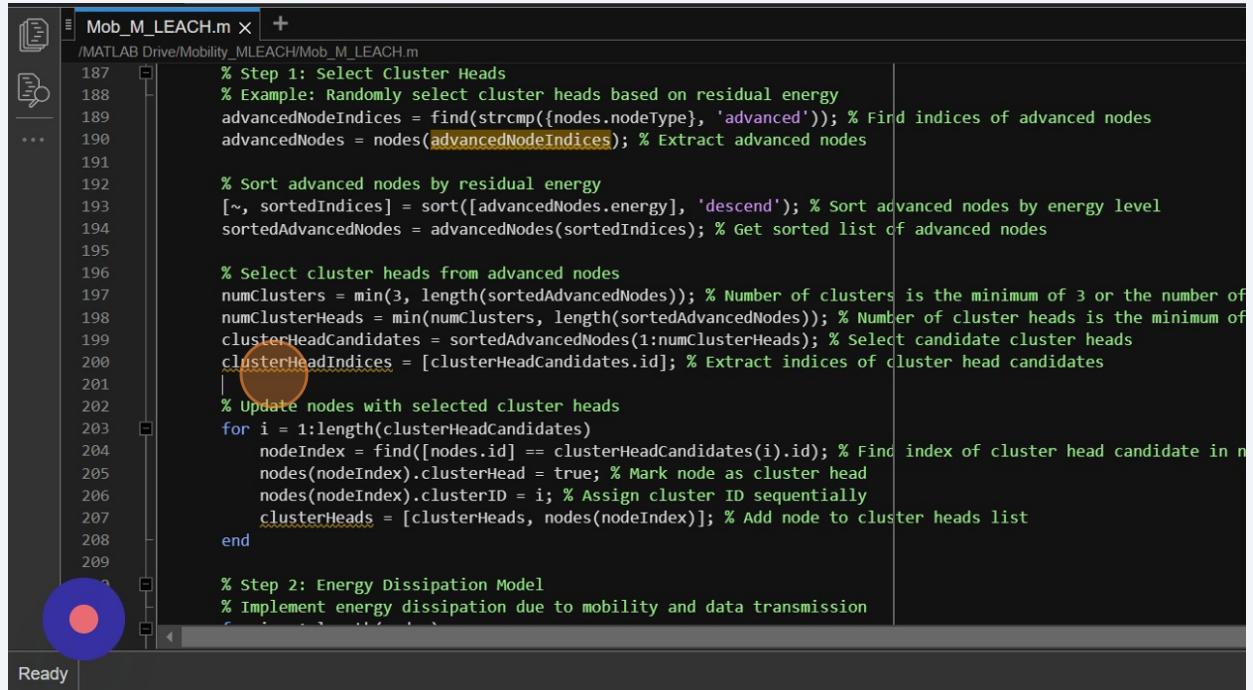
25 Select cluster heads from advanced nodes

Number of clusters is the minimum of 3 or the number of advanced nodes

Number of cluster heads is the minimum of numClusters or the number of advanced nodes

Select candidate cluster heads

Extract indices of cluster head candidates



The screenshot shows a MATLAB code editor window titled "Mob_M_LEACH.m". The code is written in MATLAB and performs the following steps:

- Step 1: Select Cluster Heads**
 - Example: Randomly select cluster heads based on residual energy
 - advancedNodeIndices = find(strcmp({nodes.nodeType}, 'advanced'));
 - advancedNodes = nodes(advancedNodeIndices); % Extract advanced nodes
- Sort advanced nodes by residual energy**
 - [~, sortedIndices] = sort([advancedNodes.energy], 'descend');
 - sortedAdvancedNodes = advancedNodes(sortedIndices); % Get sorted list of advanced nodes
- Select cluster heads from advanced nodes**
 - numClusters = min(3, length(sortedAdvancedNodes)); % Number of clusters is the minimum of 3 or the number of advanced nodes
 - numClusterHeads = min(numClusters, length(sortedAdvancedNodes)); % Number of cluster heads is the minimum of numClusters and the number of advanced nodes
 - clusterHeadCandidates = sortedAdvancedNodes(1:numClusterHeads); % Select candidate cluster heads
 - clusterHeadIndices = [clusterHeadCandidates.id]; % Extract indices of cluster head candidates
- Update nodes with selected cluster heads**
 - for i = 1:length(clusterHeadCandidates)
 - nodeIndex = find([nodes.id] == clusterHeadCandidates(i).id); % Find index of cluster head candidate in nodes
 - nodes(nodeIndex).clusterHead = true; % Mark node as cluster head
 - nodes(nodeIndex).clusterID = i; % Assign cluster ID sequentially
 - clusterHeads = [clusterHeads, nodes(nodeIndex)]; % Add node to cluster heads list
- Step 2: Energy Dissipation Model**
 - Implement energy dissipation due to mobility and data transmission

A red circle highlights the line of code "clusterHeadIndices = [clusterHeadCandidates.id];".

26

• Step 2: Energy Dissipation Model:

- It models energy dissipation due to node movements and data transmission.
- Energy consumption due to node movements is simulated by randomly reducing the energy level of each node.
- Energy consumption due to data transmission is calculated for each non-cluster head node by considering the distance to its cluster head and a constant energy consumption rate.

The screenshot shows a MATLAB code editor window titled 'Mob_M_LEACH.m'. The code implements the LEACH protocol with mobility. It starts by marking nodes as cluster heads and assigning them cluster IDs sequentially. Then, it enters a loop for each node. Inside the loop, it handles energy consumption due to movement (random reduction) and data transmission (calculated based on distance to the cluster head). The code uses various MATLAB functions like 'randi' for movement energy, 'sqrt' for distance calculation, and logical operators for cluster head identification. A large orange circle highlights the word 'end' at the end of the inner loop, indicating the completion of the node processing. The status bar at the bottom left says 'Ready'.

```
205 nodes(nodeIndex).clusterHead = true; % Mark node as cluster head
206 nodes(nodeIndex).clusterID = i; % Assign cluster ID sequentially
207 clusterHeads = [clusterHeads, nodes(nodeIndex)]; % Add node to cluster heads list
208 end
209
210 % Step 2: Energy Dissipation Model
211 % Implement energy dissipation due to mobility and data transmission
212 for i = 1:length(nodes)
213     % Example: Energy consumption due to node movements
214     energyConsumptionMovement = randi(5); % Random energy consumption due to node movement
215     nodes(i).energy = nodes(i).energy - energyConsumptionMovement; % Update energy level
216
217     % Energy consumption due to data transmission
218     if ~nodes(i).clusterHead % If node is not a cluster head
219         % Find the cluster head for the node
220         clusterHeadID = find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1);
221         if ~isempty(clusterHeadID) % If cluster head is found
222             % Calculate distance between node and cluster head
223             distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2);
224             energyConsumptionTransmission = distance * 0.1; % Calculate energy consumption for data transmission
225             nodes(i).energy = nodes(i).energy - energyConsumptionTransmission; % Update energy level
226         end
227     end
228
229 % Step 3: Network Energy Calculation
```

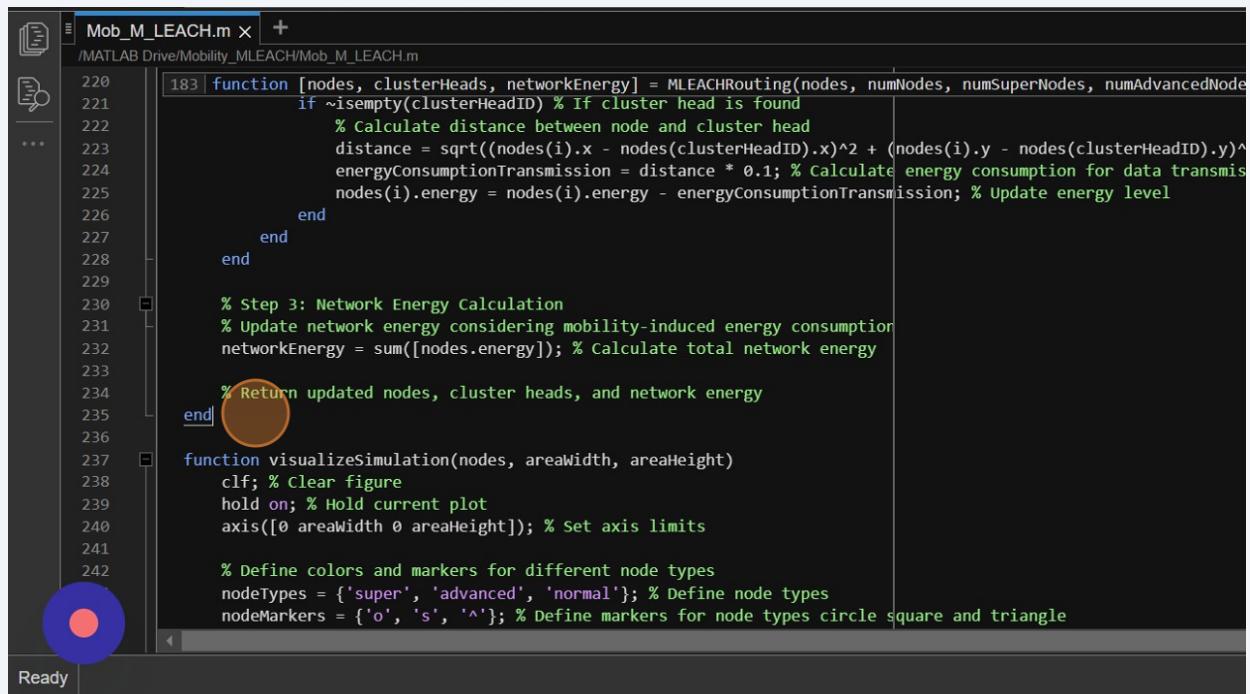
27

- **Step 3: Network Energy Calculation:**

- It calculates the total network energy after considering energy consumption due to mobility and data transmission.

- **Return values:**

- The function returns the updated nodes array, the list of selected cluster heads, and the total network energy.



```
Mob_M_LEACH.m X +
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
220 | 183 | function [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes
221 | if ~isempty(clusterHeadID) % If cluster head is found
222 |     % calculate distance between node and cluster head
223 |     distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2);
224 |     energyConsumptionTransmission = distance * 0.1; % Calculate energy consumption for data transmission
225 |     nodes(i).energy = nodes(i).energy - energyConsumptionTransmission; % Update energy level
226 | end
227 | end
228 |
229 |
230 | % Step 3: Network Energy Calculation
231 | % Update network energy considering mobility-induced energy consumption
232 | networkEnergy = sum([nodes.energy]); % Calculate total network energy
233 |
234 | % Return updated nodes, cluster heads, and network energy
235 | end
236 |
237 | function visualizeSimulation(nodes, areaWidth, areaHeight)
238 | clf; % Clear figure
239 | hold on; % Hold current plot
240 | axis([0 areaWidth 0 areaHeight]); % Set axis limits
241 |
242 | % Define colors and markers for different node types
243 | nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
244 | nodeMarkers = {'o', 's', '^'}; % Define markers for node types circle square and triangle
```

28 Energy calculation display:

- It displays a message indicating that energy calculations are being shown (disp('Energy Calculation:')).
- It iterates through each node in the nodes array.
 - For each node, it displays a message showing the node ID and its updated energy level (disp(['Node ' num2str(nodes(i).id) ' Energy updated! ' num2str(nodes(i).energy)])).

```
Mob_M_LEACH.m X +  
/MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m  
70 % M-LEACH Routing Protocol  
71 [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes, initialEnergy);  
72 % Energy Calculation - Display energy calculations after updating  
73 disp('Energy Calculation:');  
74 for i = 1:length(nodes)  
75     disp(['Node ' num2str(nodes(i).id) ' Energy updated! ' num2str(nodes(i).energy)]); % Display energy level  
76 end  
77 % Visualization  
78 visualizeSimulation(nodes, areaWidth, areaHeight); % Visualize the current state of the network  
79 pause(0.5); % Pause for visualization  
80 end  
81 % Function Definitions  
82 function nodes = updateNodePositions(nodes, areaWidth, areaHeight)  
83 % Implement mobility model here  
84 for i = 1:length(nodes)  
85     % Move node i to a random position within the area  
86     % Update node i's position in the nodes matrix  
87 end  
88 % End of updateNodePositions function  
89 % End of Mob_M_LEACH.m script
```

29

Explanation:

• Visualization:

- It calls the visualizeSimulation function to generate a visualization of the current state of the network based on the information stored in the nodes array, as well as the dimensions of the simulation area (areaWidth and areaHeight).

• Pause for visualization:

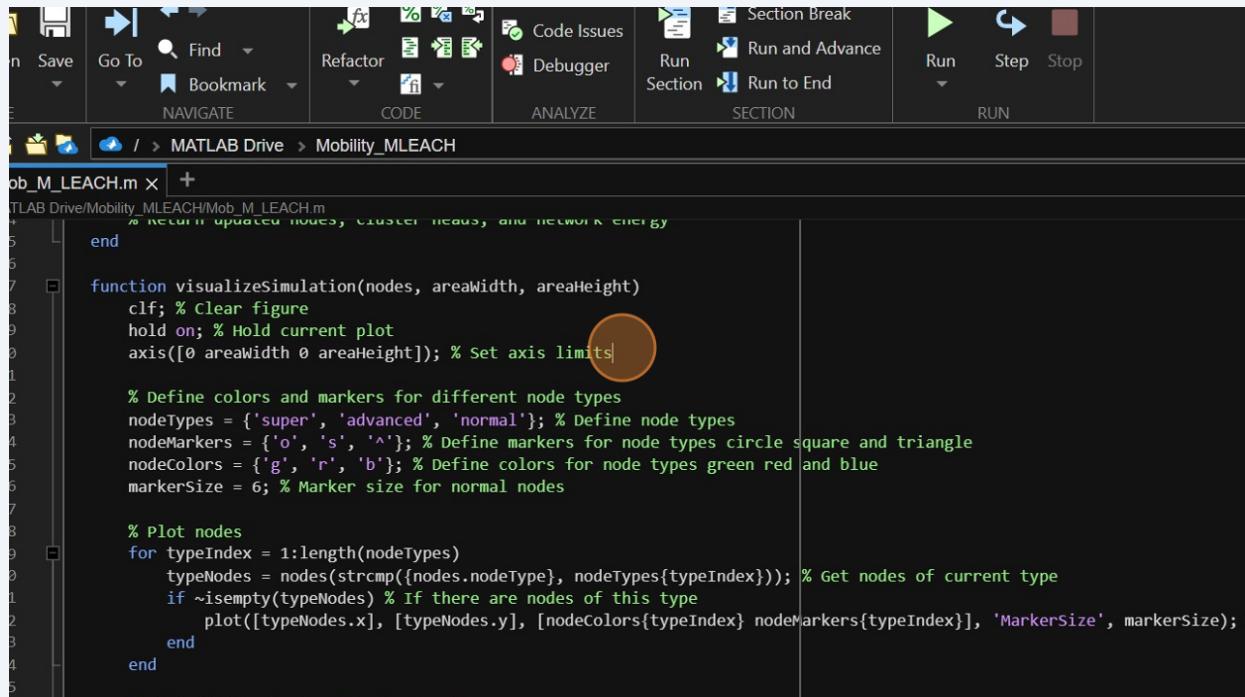
- After displaying the visualization, it pauses the execution for 0.5 seconds using the pause function. This gives time for the visualization to be observed before proceeding with the next iteration of the simulation.

The screenshot shows a MATLAB code editor window titled 'Mob_M_LEACH.m'. The code is a script for a M-LEACH routing protocol. It includes sections for data transmission, energy calculation, visualization, and a function definition. A yellow circle highlights the line 'pause(0.5); % Pause for visualization' at line 88. The code is as follows:

```
1 disp(['Node ' num2str(nodes(i).id) ' transmitted data to Cluster Head ' num2str(clusterHeadID) ...
2     '. Distance: ' num2str(distance) ', Energy Consumed: ' num2str(energyConsumption)])'; % Dis
3 end
4 end
5 nodes = dataTransmission(nodes); % Perform data transmission
6
7 % M-LEACH Routing Protocol
8 [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNodes, initia
9
10 % Energy Calculation - Display energy calculations after updating
11 disp('Energy Calculation:');
12 for i = 1:length(nodes)
13     disp(['Node ' num2str(nodes(i).id) ' Energy updated! ' num2str(nodes(i).energy)]); % Display energy leve
14 end
15
16 % Visualization
17 visualizeSimulation(nodes, areaWidth, areaHeight); % Visualize the current state of the network
18 pause(0.5); % Pause for visualization
19 end
20
21 % Function Definitions
22
23 function nodes = updateNodePositions(nodes, areaWidth, areaHeight)
24 % Implement mobility model here
25 for i = 1:length(nodes)
```

30

- **Clear figure:** clf clears the current figure, ensuring that each visualization iteration starts with a clean slate.
- **Hold on:** hold on allows subsequent plot commands to be overlaid on the current plot, preserving the existing content.
- **Set axis limits:** axis([0 areaWidth 0 areaHeight]) sets the limits for the x and y axes based on the dimensions of the simulation area.



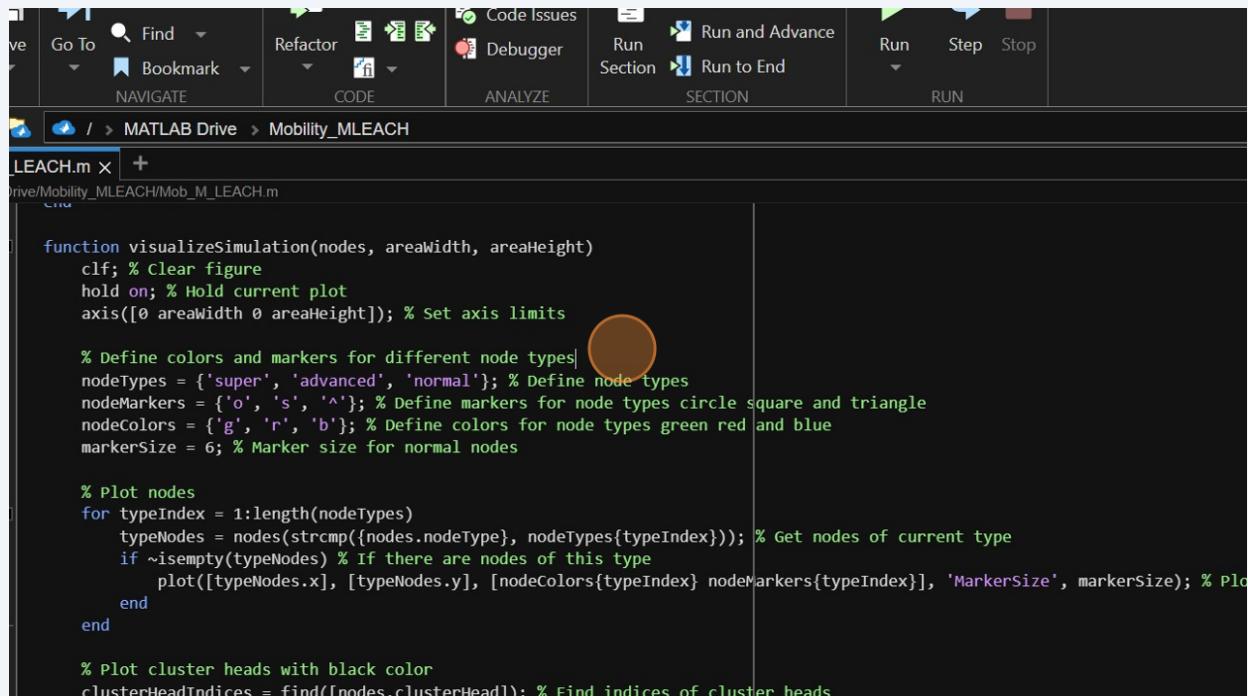
The screenshot shows the MATLAB IDE interface. The top menu bar includes Save, Go To, Find, Refactor, Code Issues, Debugger, ANALYZE, Run Section, Run and Advance, Run to End, SECTION, Run, Step, Stop, and RUN. Below the menu is a toolbar with icons for Save, Open, New, MATLAB Drive, and Mobility_MLEACH.m. The code editor window displays the file 'Mob_M_LEACH.m' with the following content:

```
1 %>>> MATLAB Drive/Mobility_MLEACH/Mob_M_LEACH.m
2 %>>> % return updated nodes, cluster heads, and network energy
3
4 function visualizesimulation(nodes, areaWidth, areaHeight)
5     clf; % Clear figure
6     hold on; % Hold current plot
7     axis([0 areaWidth 0 areaHeight]); % Set axis limits
8
9     % Define colors and markers for different node types
10    nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
11    nodeMarkers = {'o', 's', '^'}; % Define markers for node types circle square and triangle
12    nodeColors = {'g', 'r', 'b'}; % Define colors for node types green red and blue
13    markerSize = 6; % Marker size for normal nodes
14
15    % Plot nodes
16    for typeIndex = 1:length(nodeTypes)
17        typeNodes = nodes(strcmp({nodes.nodeType}, nodeTypes{typeIndex})); % Get nodes of current type
18        if ~isempty(typeNodes) % If there are nodes of this type
19            plot([typeNodes.x], [typeNodes.y], [nodeColors{typeIndex} nodeMarkers{typeIndex}], 'MarkerSize', markerSize);
20        end
21    end
22
```

A brown circle highlights the line of code 'axis([0 areaWidth 0 areaHeight]); % Set axis limits'.

31

Node types, markers, and colors: Defines different types of nodes and their corresponding markers and colors.



```

function visualizeSimulation(nodes, areaWidth, areaHeight)
    clf; % Clear figure
    hold on; % Hold current plot
    axis([0 areaWidth 0 areaHeight]); % Set axis limits

    % Define colors and markers for different node types
    nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
    nodeMarkers = {'o', 's', '^'}; % Define markers for node types circle square and triangle
    nodeColors = {'g', 'r', 'b'}; % Define colors for node types green red and blue
    markerSize = 6; % Marker size for normal nodes

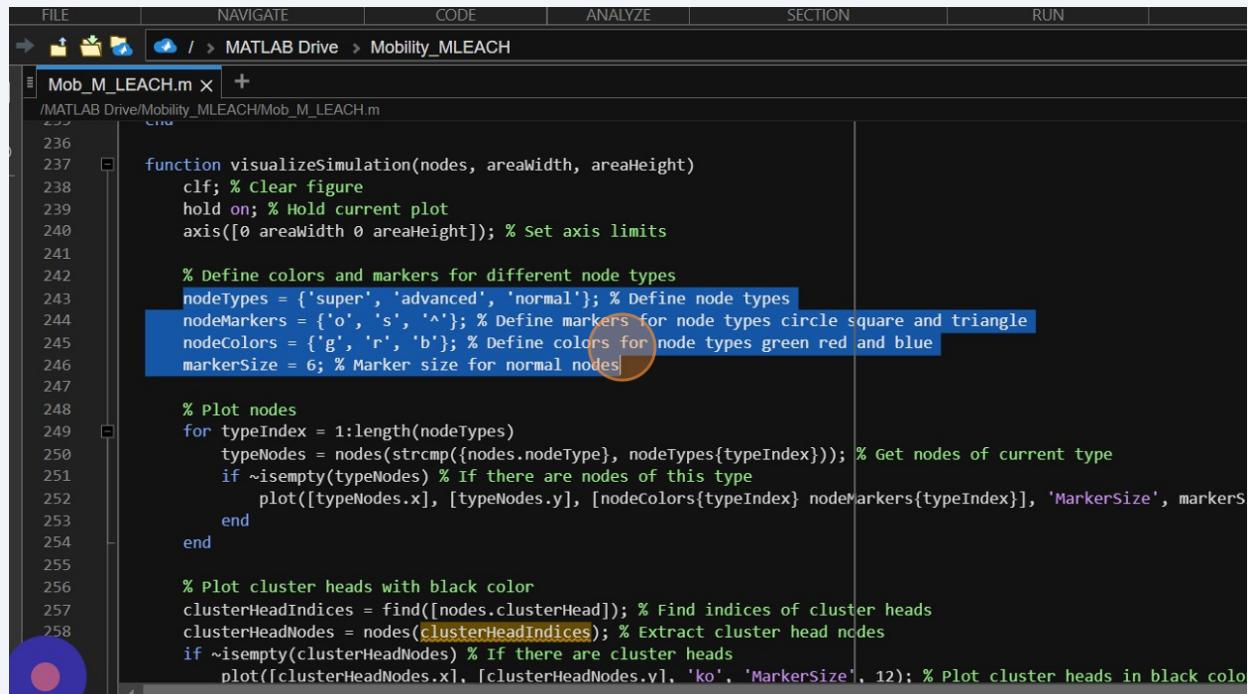
    % Plot nodes
    for typeIndex = 1:length(nodeTypes)
        typeNodes = nodes(strcmp({nodes.nodeType}, nodeTypes{typeIndex})); % Get nodes of current type
        if ~isempty(typeNodes) % If there are nodes of this type
            plot([typeNodes.x], [typeNodes.y], [nodeColors{typeIndex} nodeMarkers{typeIndex}], 'MarkerSize', markerSize); % Plot
        end
    end

    % Plot cluster heads with black color
    clusterHeadIndices = find([nodes.clusterHead]); % Find indices of cluster heads

```

32

Here markers circle square and triangle are used and colors red green and blue are used and markersize for nodes is 6



```

function visualizeSimulation(nodes, areaWidth, areaHeight)
    clf; % Clear figure
    hold on; % Hold current plot
    axis([0 areaWidth 0 areaHeight]); % set axis limits

    % Define colors and markers for different node types
    nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
    nodeMarkers = {'o', 's', '^'}; % Define markers for node types circle square and triangle
    nodeColors = {'g', 'r', 'b'}; % Define colors for node types green red and blue
    markerSize = 6; % Marker size for normal nodes

    % Plot nodes
    for typeIndex = 1:length(nodeTypes)
        typeNodes = nodes(strcmp({nodes.nodeType}, nodeTypes{typeIndex})); % Get nodes of current type
        if ~isempty(typeNodes) % If there are nodes of this type
            plot([typeNodes.x], [typeNodes.y], [nodeColors{typeIndex} nodeMarkers{typeIndex}], 'MarkerSize', markerSize);
        end
    end

    % Plot cluster heads with black color
    clusterHeadIndices = find([nodes.clusterHead]); % Find indices of cluster heads
    clusterHeadNodes = nodes(clusterHeadIndices); % Extract cluster head nodes
    if ~isempty(clusterHeadNodes) % If there are cluster heads
        plot([clusterHeadNodes.x], [clusterHeadNodes.y], 'ko', 'MarkerSize', 12); % Plot cluster heads in black color
    end

```

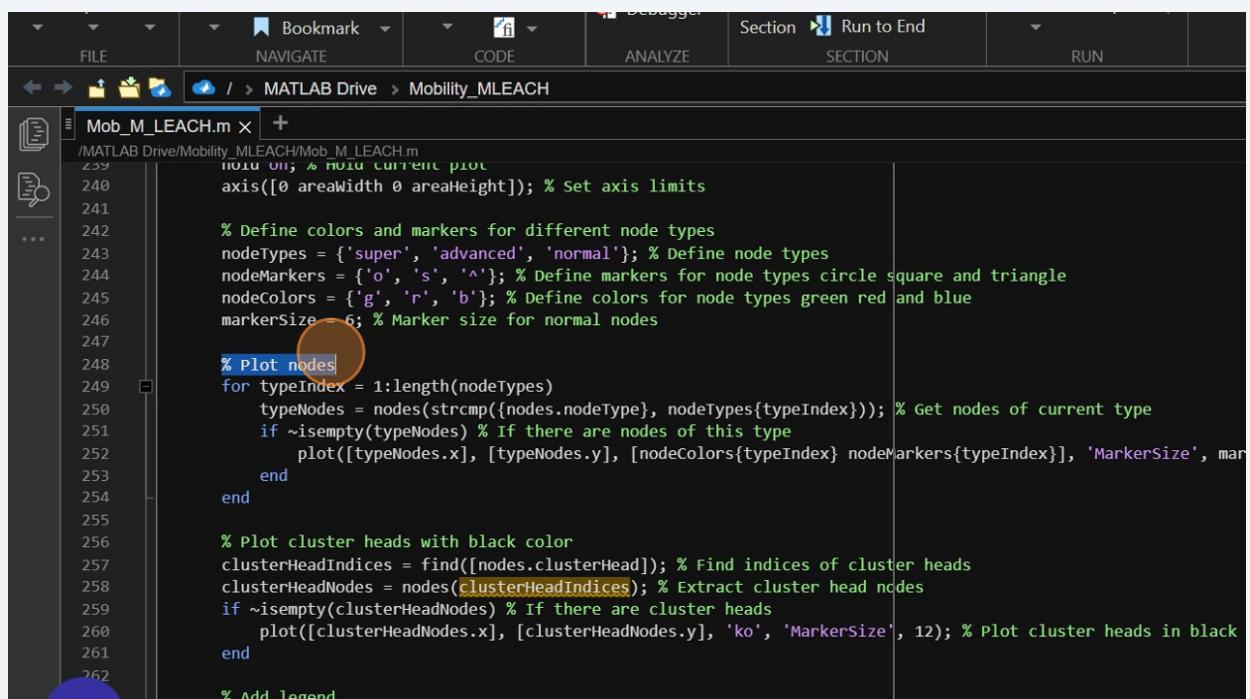
- **Iterating over node types:** The loop iterates over each type of node specified in `nodeTypes`.

• **Filtering nodes:** For each type of node, it filters out nodes of that type using logical indexing based on the node type stored in the `nodes` structure.

• **Plotting nodes:** It plots the nodes of the current type using the `plot` function. The x-coordinates of the nodes are given by `[typeNodes.x]`, and the y-coordinates by `[typeNodes.y]`. The color and marker style are determined by `nodeColors{typeIndex}` and `nodeMarkers{typeIndex}`, respectively.

• **Marker size:** The marker size for normal nodes is specified by the variable `markerSize`.

This section ensures that nodes of different types are plotted with distinct markers and colors, allowing for easy visualization and differentiation in the network simulation.



```

FILE      BOOKMARK fi DEBUGGER
NAVIGATE ANALYZE
CODE      SECTION RUN
/ > MATLAB Drive > Mobility_MLEACH
Mob_M_MEACH.m x +
/MATLAB Drive/Mobility_MLEACH/Mob_M_MEACH.m
239 hold on; % Hold current plot
240 axis([0 areaWidth 0 areaHeight]); % Set axis limits
241
242 % Define colors and markers for different node types
243 nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
244 nodeMarkers = {'o', 's', '^'}; % Define markers for node types circle square and triangle
245 nodeColors = {'g', 'r', 'b'}; % Define colors for node types green red and blue
246 markerSize = 6; % Marker size for normal nodes
247
248 % Plot nodes
249 for typeIndex = 1:length(nodeTypes)
250     typeNodes = nodes(strcmp({nodes.nodeType}, nodeTypes{typeIndex})); % Get nodes of current type
251     if ~isempty(typeNodes) % If there are nodes of this type
252         plot([typeNodes.x], [typeNodes.y], [nodeColors{typeIndex} nodeMarkers{typeIndex}], 'MarkerSize', markerSize);
253     end
254 end
255
256 % Plot cluster heads with black color
257 clusterHeadIndices = find([nodes.clusterHead]); % Find indices of cluster heads
258 clusterHeadNodes = nodes(clusterHeadIndices); % Extract cluster head nodes
259 if ~isempty(clusterHeadNodes) % If there are cluster heads
260     plot([clusterHeadNodes.x], [clusterHeadNodes.y], 'ko', 'MarkerSize', 12); % Plot cluster heads in black
261 end
262
263 % Add legend

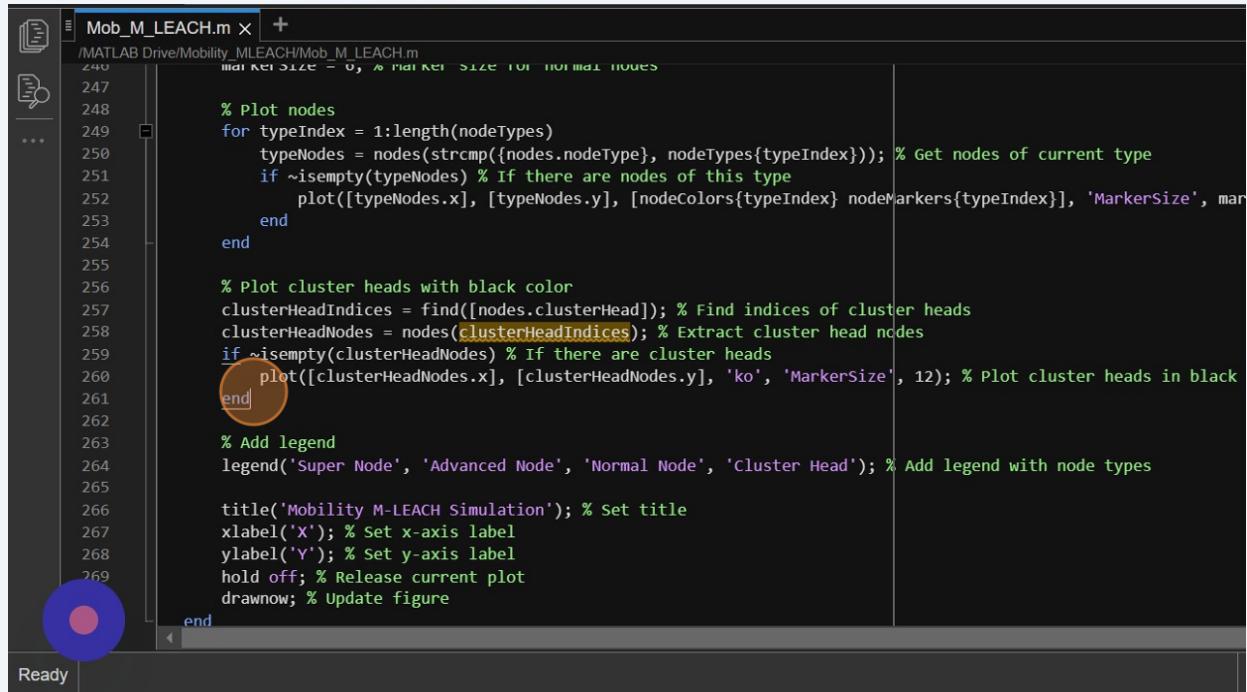
```

34

• **Find cluster heads:** It finds the indices of cluster heads in the nodes structure by using the logical array [nodes.clusterHead], which contains true for cluster heads and false otherwise. The find function returns the indices of true elements.

• **Extract cluster head nodes:** It extracts the cluster head nodes from the nodes structure based on the indices found in the previous step.

• **Plot cluster heads:** It checks if there are any cluster head nodes. If there are, it plots them using the plot function. The x-coordinates of the cluster head nodes are given by [clusterHeadNodes.x], and the y-coordinates by [clusterHeadNodes.y]. They are plotted as black circles (`'ko'`) with a marker size of 12.



The screenshot shows a MATLAB code editor window with the file `Mob_M_LEACH.m` open. The code is a script for a Mobility M-LEACH simulation. A specific section of the code is highlighted with a yellow oval, which corresponds to the question in the text above. The highlighted code is as follows:

```
% Plot cluster heads with black color
clusterHeadIndices = find([nodes.clusterHead]); % Find indices of cluster heads
clusterHeadNodes = nodes(clusterHeadIndices); % Extract cluster head nodes
if ~isempty(clusterHeadNodes) % If there are cluster heads
    plot([clusterHeadNodes.x], [clusterHeadNodes.y], 'ko', 'MarkerSize', 12); % Plot cluster heads in black
end
```

The MATLAB interface includes a toolbar at the top, a code navigation sidebar on the left, and a status bar at the bottom indicating "Ready".

35

- The legend function is used to add a legend to the plot. Each string argument corresponds to a label for the corresponding plot element. In this case, the legend will have labels for 'Super Node', 'Advanced Node', 'Normal Node', and 'Cluster Head'.

```
title('Mobility M-LEACH Simulation'); % Set title
```

- The title function sets the title of the plot to 'Mobility M-LEACH Simulation'.

```
xlabel('X'); % Set x-axis label
```

- The xlabel function sets the label for the x-axis to 'X'.

```
ylabel('Y'); % Set y-axis label
```

- The ylabel function sets the label for the y-axis to 'Y'.

```
hold off; % Release current plot
```

- The hold off command releases the current plot, meaning that subsequent plotting commands will clear the current plot before creating a new one.

```
drawnow; % Update figure
```

- The drawnow function updates the figure, ensuring that all changes made to the plot are reflected immediately

```
247
248
249 % Plot nodes
250 for typeIndex = 1:length(nodeTypes)
251     typeNodes = nodes(strcmp({nodes.nodeType}, nodeTypes{typeIndex})); % Get nodes of current type
252     if ~isempty(typeNodes) % If there are nodes of this type
253         plot([typeNodes.x], [typeNodes.y], [nodeColors{typeIndex} nodeMarkers{typeIndex}], 'MarkerSize', 10);
254     end
255
256 % Plot cluster heads with black color
257 clusterHeadIndices = find([nodes.clusterHead]); % Find indices of cluster heads
258 clusterHeadNodes = nodes(clusterHeadIndices); % Extract cluster head nodes
259 if ~isempty(clusterHeadNodes) % If there are cluster heads
260     plot([clusterHeadNodes.x], [clusterHeadNodes.y], 'ko', 'MarkerSize', 12); % Plot cluster heads in black
261 end
262
263 % Add legend
264 legend('Super Node', 'Advanced Node', 'Normal Node', 'Cluster Head'); % Add legend with node types
265
266 title('Mobility M-LEACH Simulation'); % set title
267 xlabel('X'); % Set x-axis label
268 ylabel('Y'); % Set y-axis label
269 hold off; % Release current plot
270 drawnow; % Update figure
271
```

36 Now run it to observe the simulation

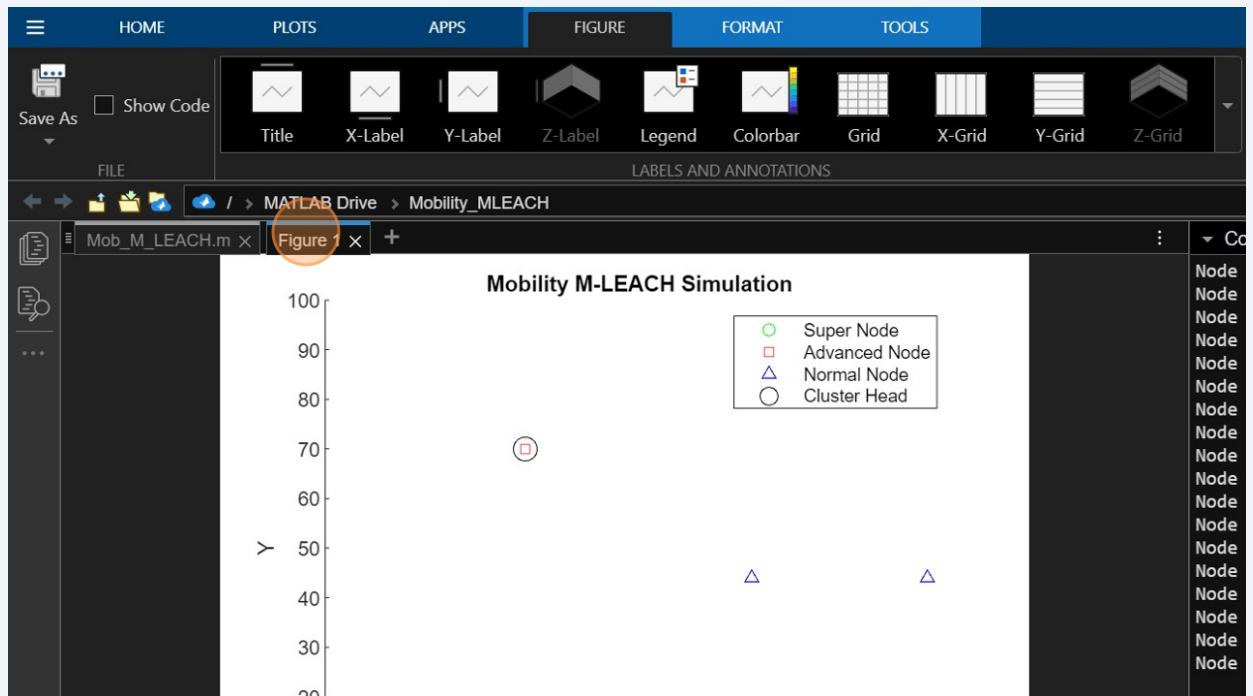
The screenshot shows the MATLAB interface with the 'RUN' tab highlighted in blue at the top menu bar. The 'RUN' tab contains three buttons: 'Run' (highlighted with a red circle), 'Step', and 'Stop'. Below the tabs, there are three main sections: 'CODE', 'ANALYZE', and 'SECTION'. The 'CODE' section displays the following MATLAB script:

```
length(nodeTypes)
s(strcmp({nodes.nodeType}, nodeTypes{typeIndex})); % Get nodes of current type
Nodes % If there are nodes of this type
odes.x, [typeNodes.y], [nodeColors{typeIndex} nodeMarkers{typeIndex}], 'MarkerSize', markerSize); % Plot nodes with appropriate color

with black color
find([nodes.clusterHead]); % Find indices of cluster heads
odes(clusterHeadIndices); % Extract cluster head nodes
eadNodes) % If there are cluster heads
dNodes.x, [clusterHeadNodes.y], 'ko', 'MarkerSize', 12); % Plot cluster heads in black color

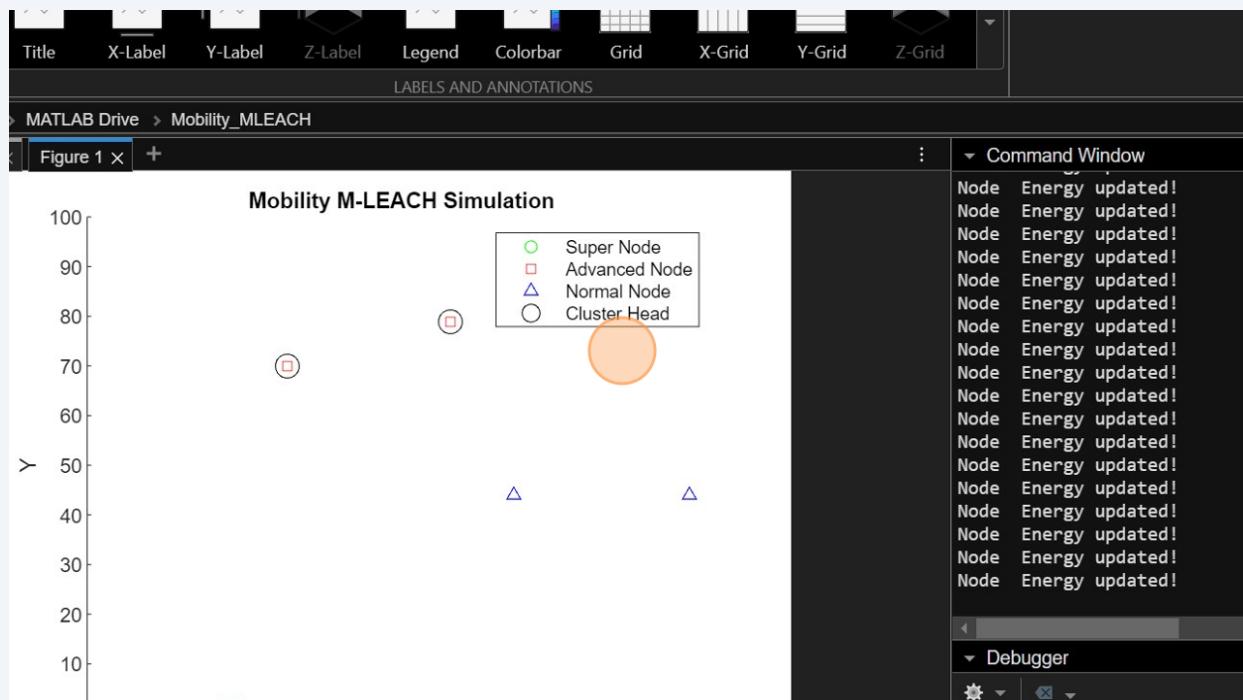
'Advanced Node', 'Normal Node', 'Cluster Head'); % Add legend with node types
```

37 Here on "Figure 1" you can see the simulation



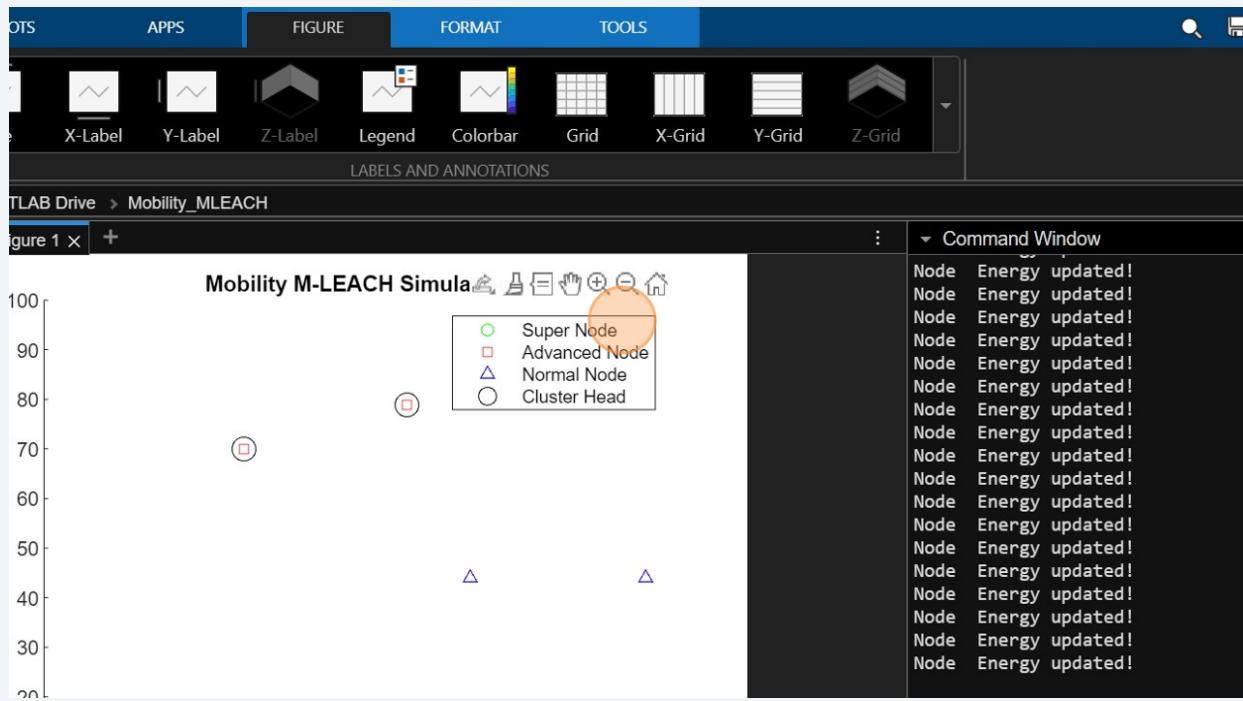
38

Here you can see the legend that has 4 nodes representations and names



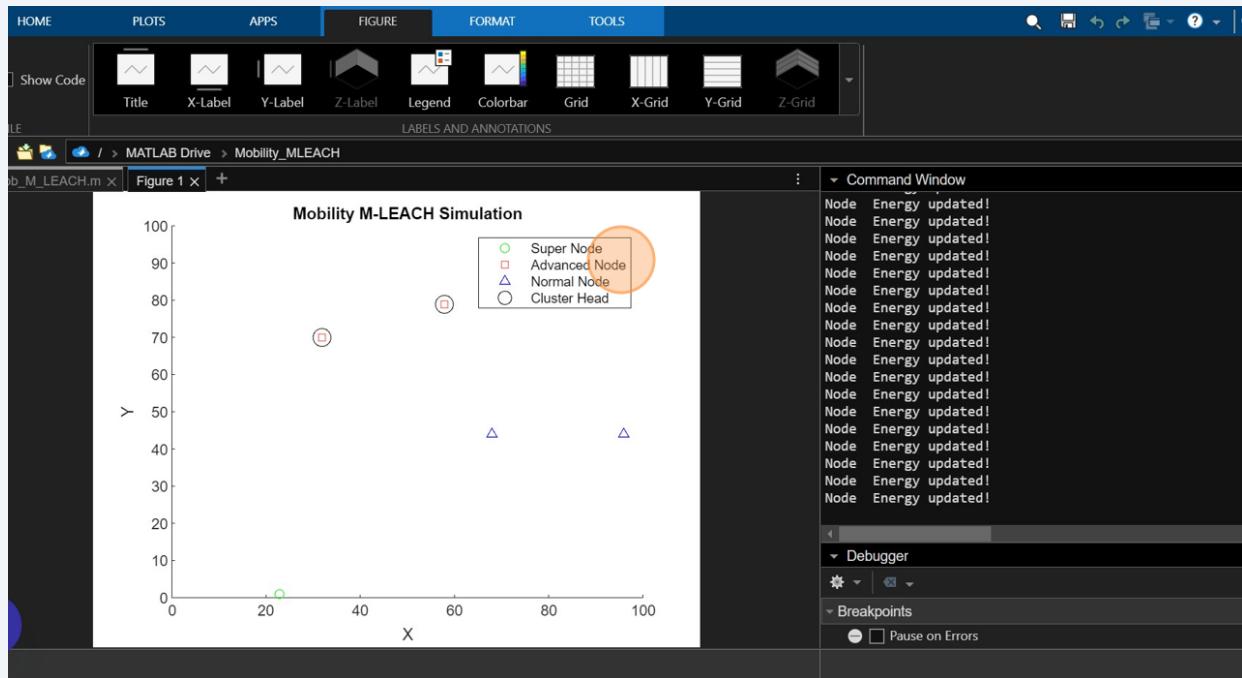
39

As the cluster heads nodes are selected from advance nodes so here the cluster head is drawn over the advance node red square



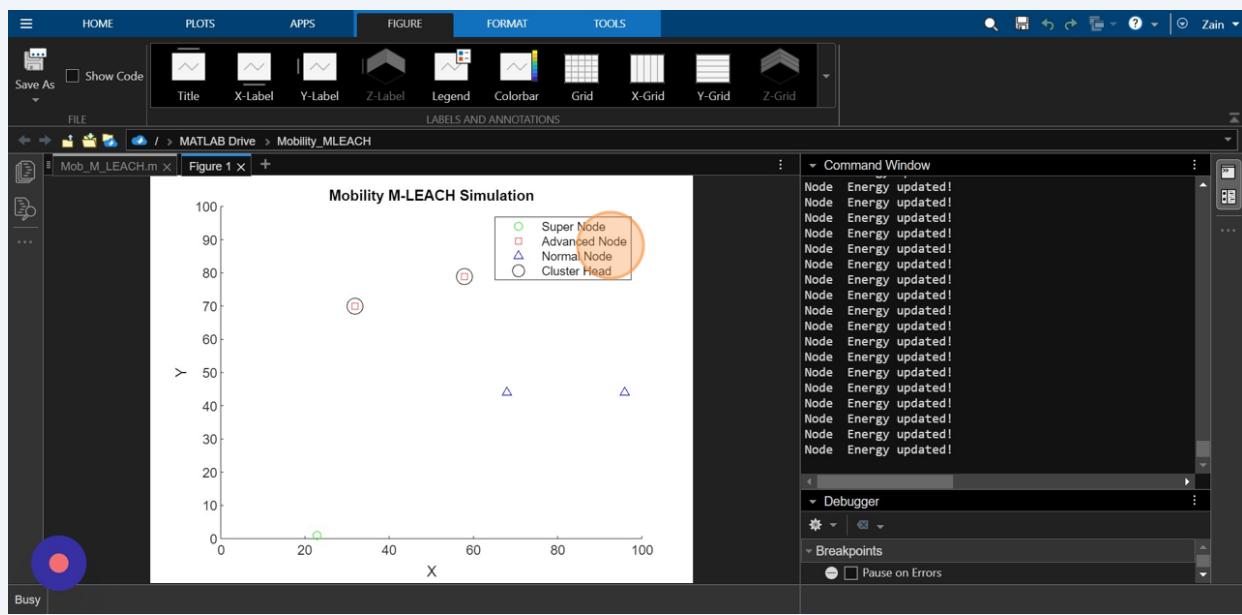
40

On the x axis there is label X and on y axis it is labeled Y
And title of screen is Mobility M-Leach Simulation



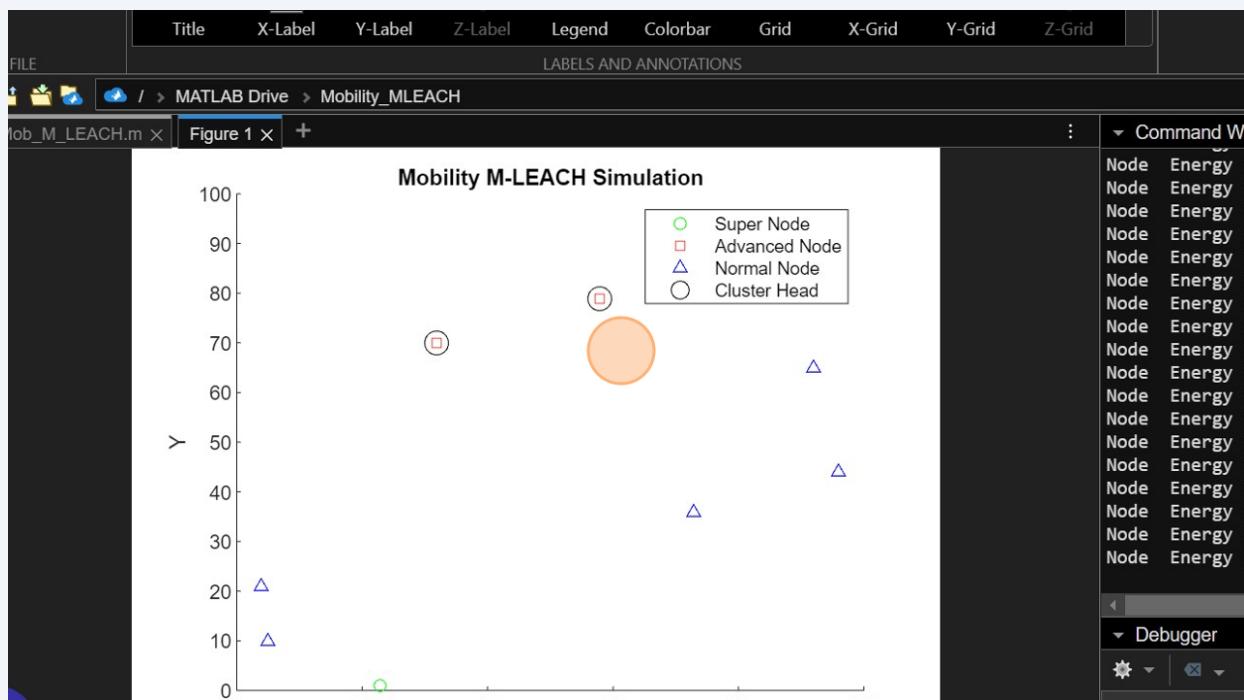
41

The total screen size you can see the width and height both are 100 that we have set in the code perimeters



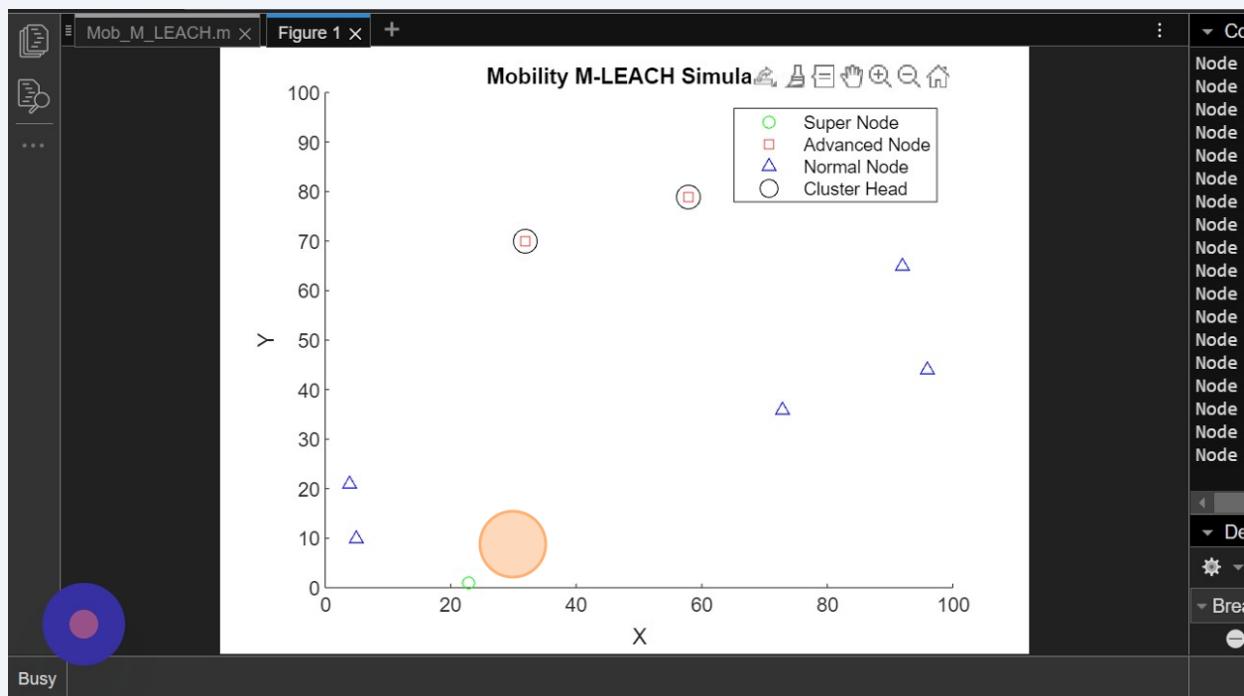
42

Here you can see mobile nodes are moving with each iteration and the number of nodes are according to the perimeters

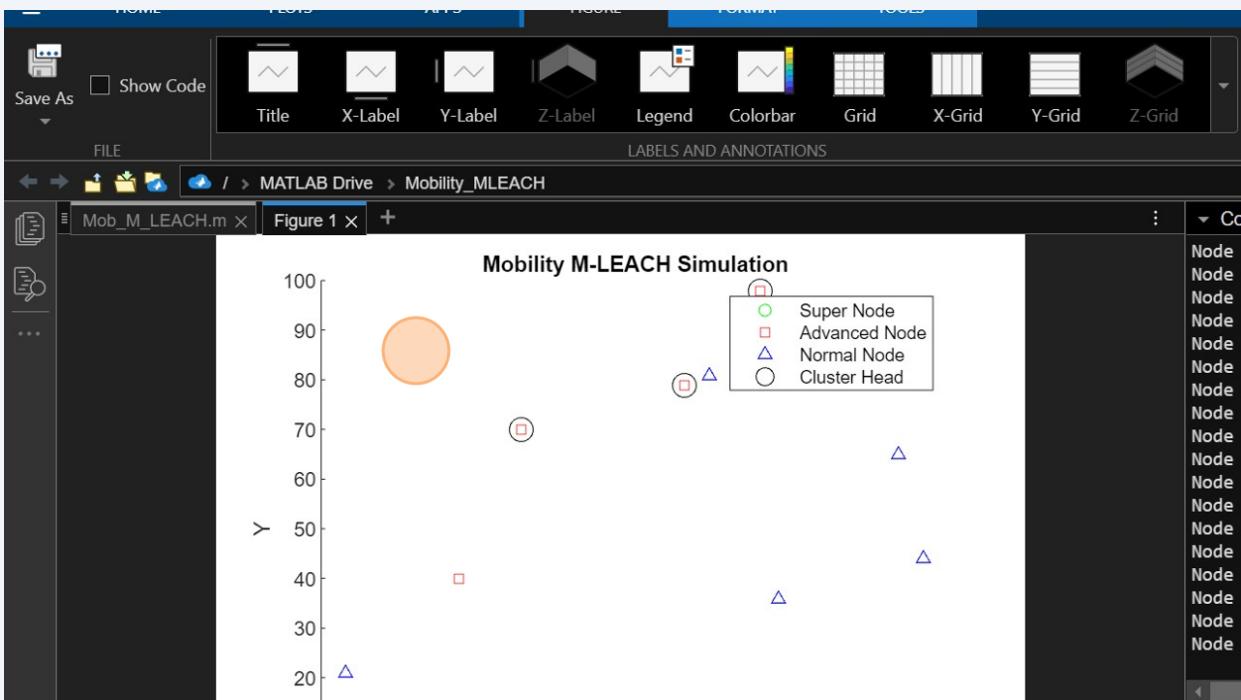


43

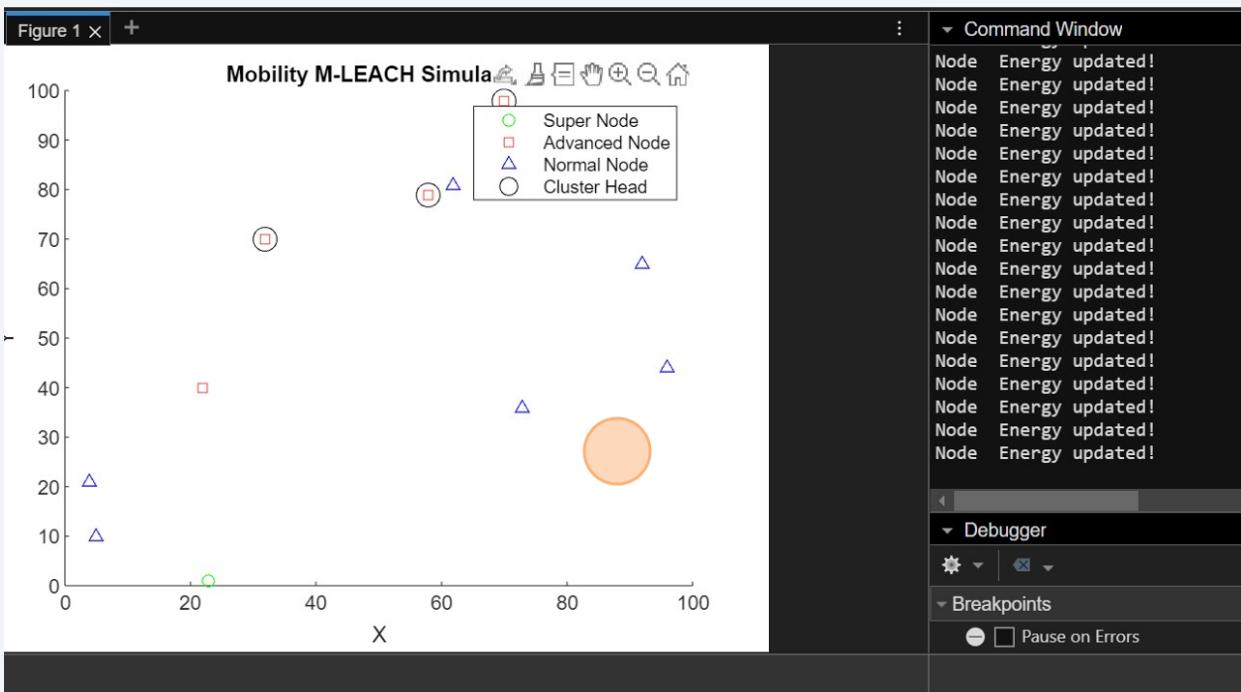
Here you can see 1 super node as we have 1 in perimeters



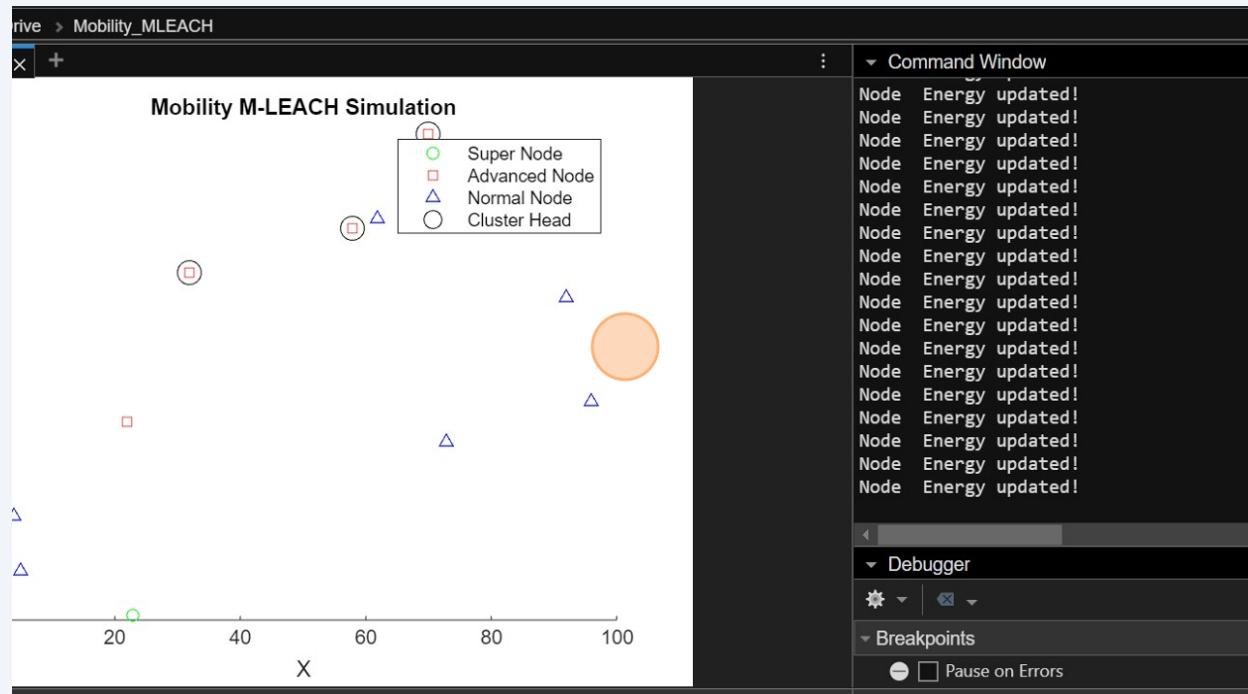
44 Click here.



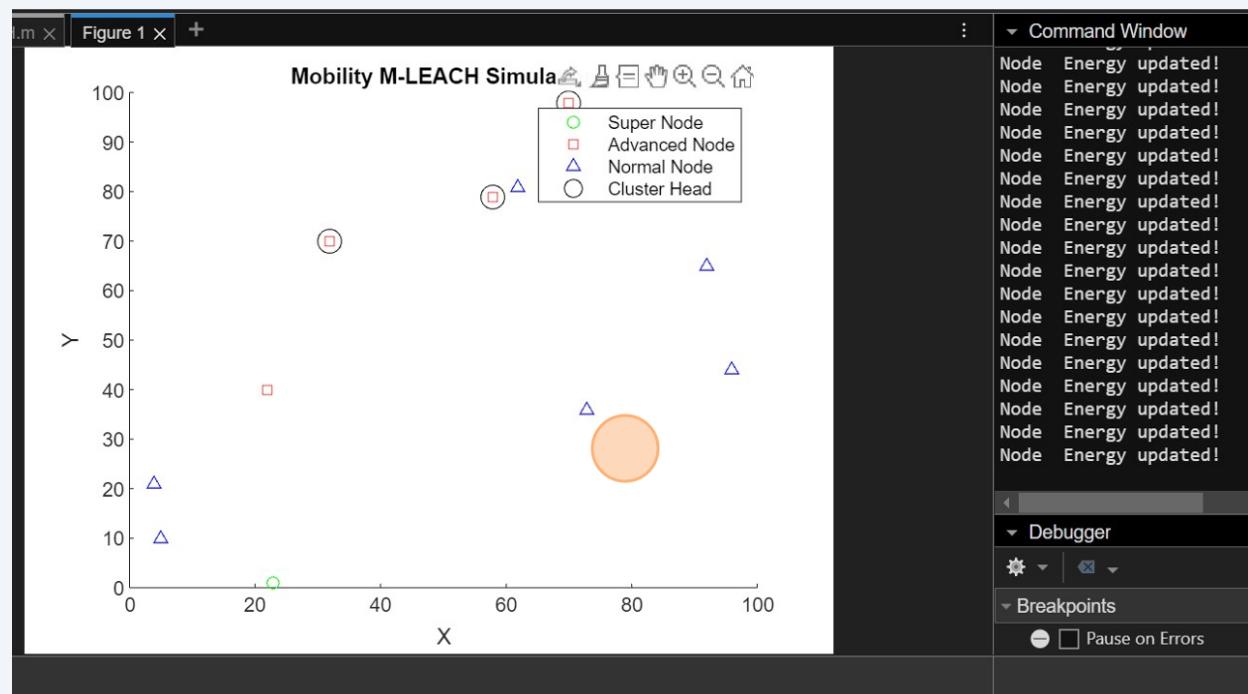
45 Click here.



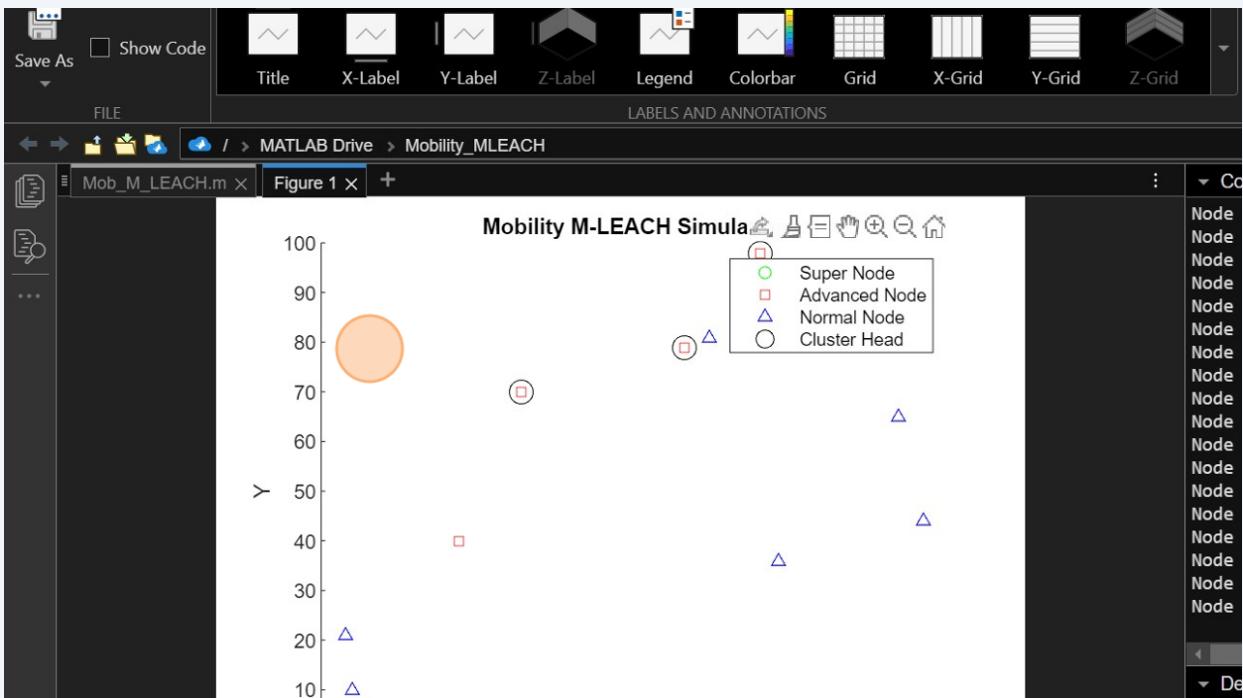
46 Click here.



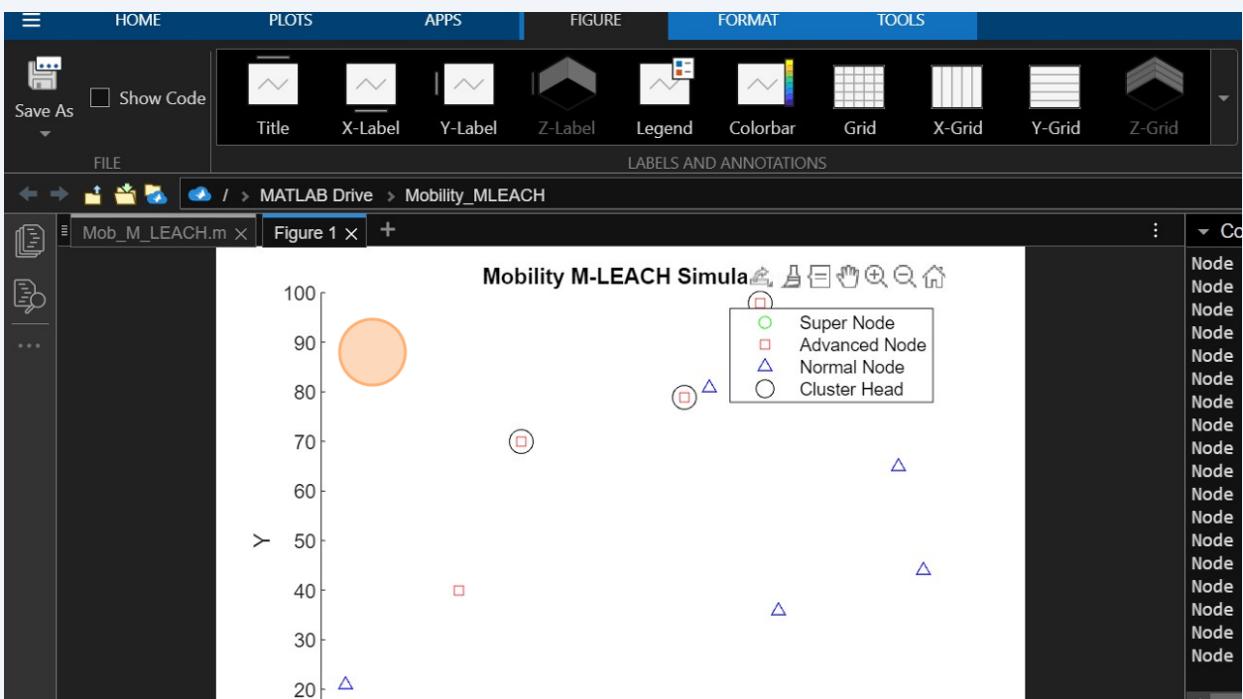
47 Click here.



48 Click here.

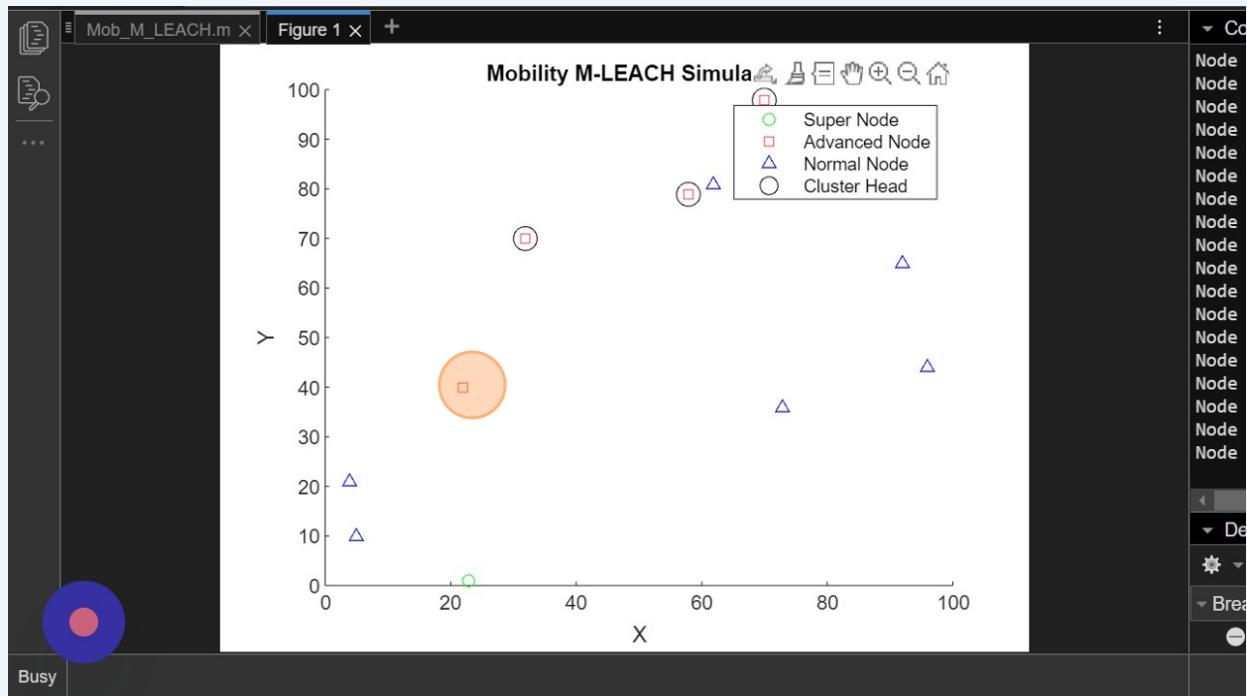


49 Click here.



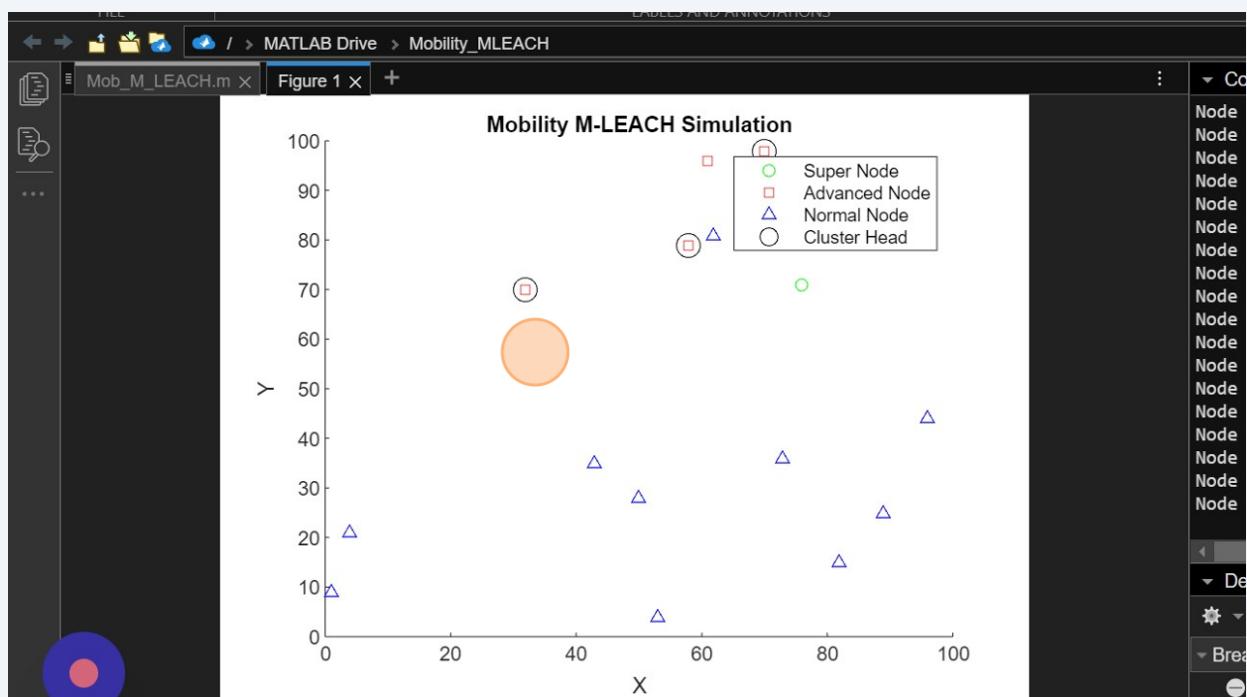
50

Here you can see some advance nodes are not cluster head
Only the selected CH are Encircled Black and denoted as cluster head



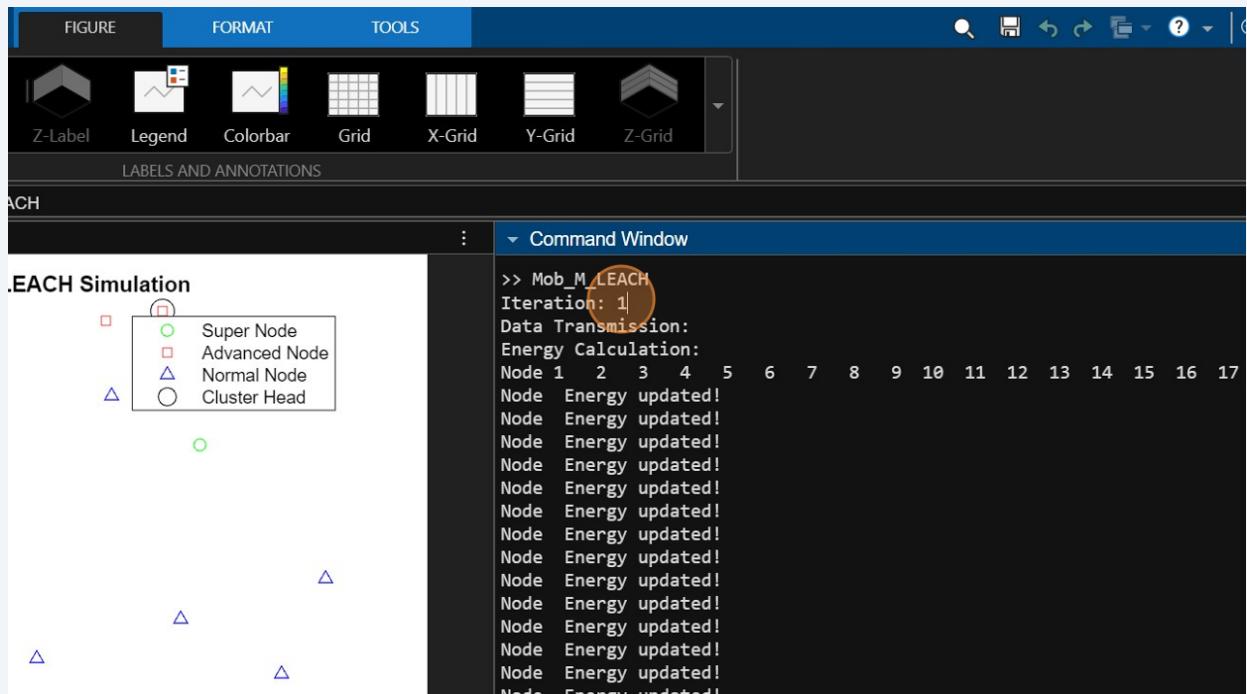
51

Sample output at 100 iterations



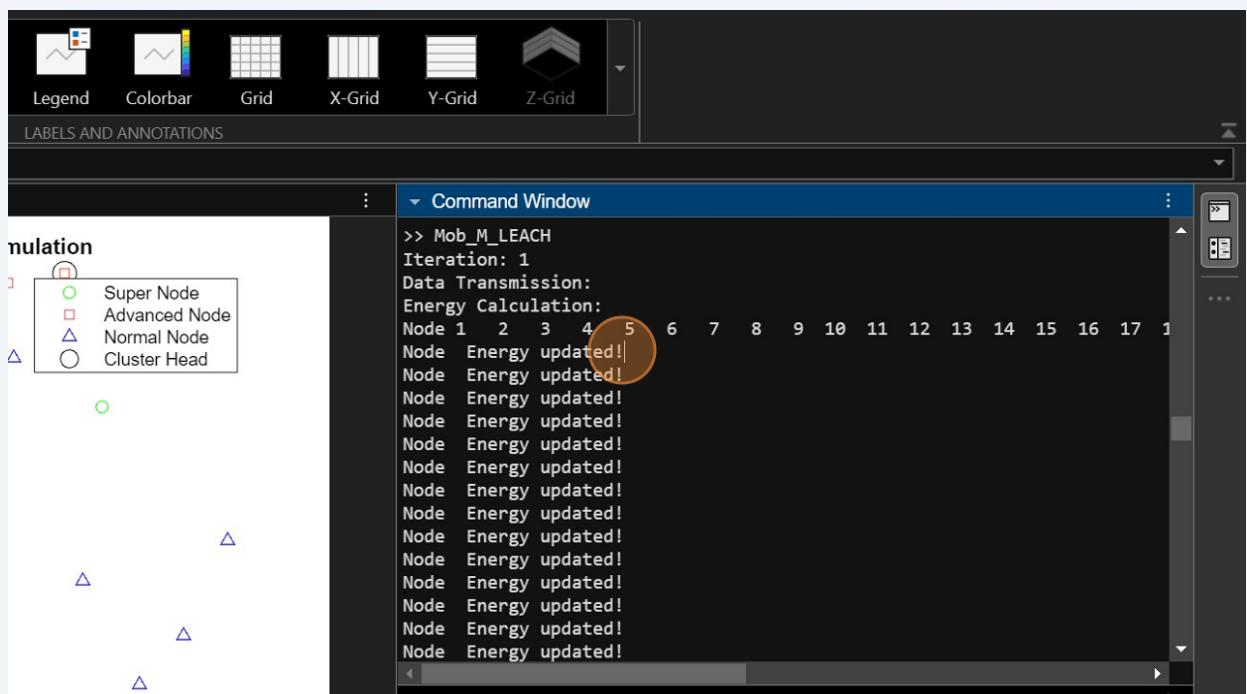
52

Here on Iteration 01:
Data Transmission:
Energy Calculation:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 Energy up..."



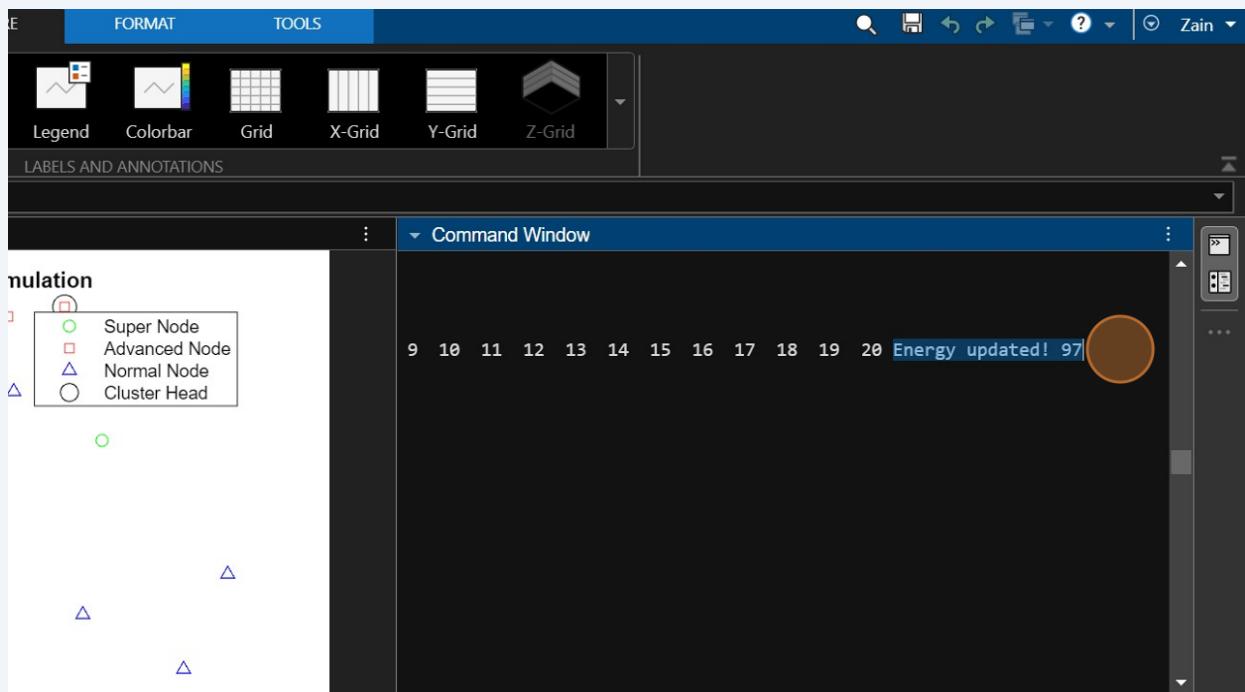
53

Here you can observe the energy updation of each node



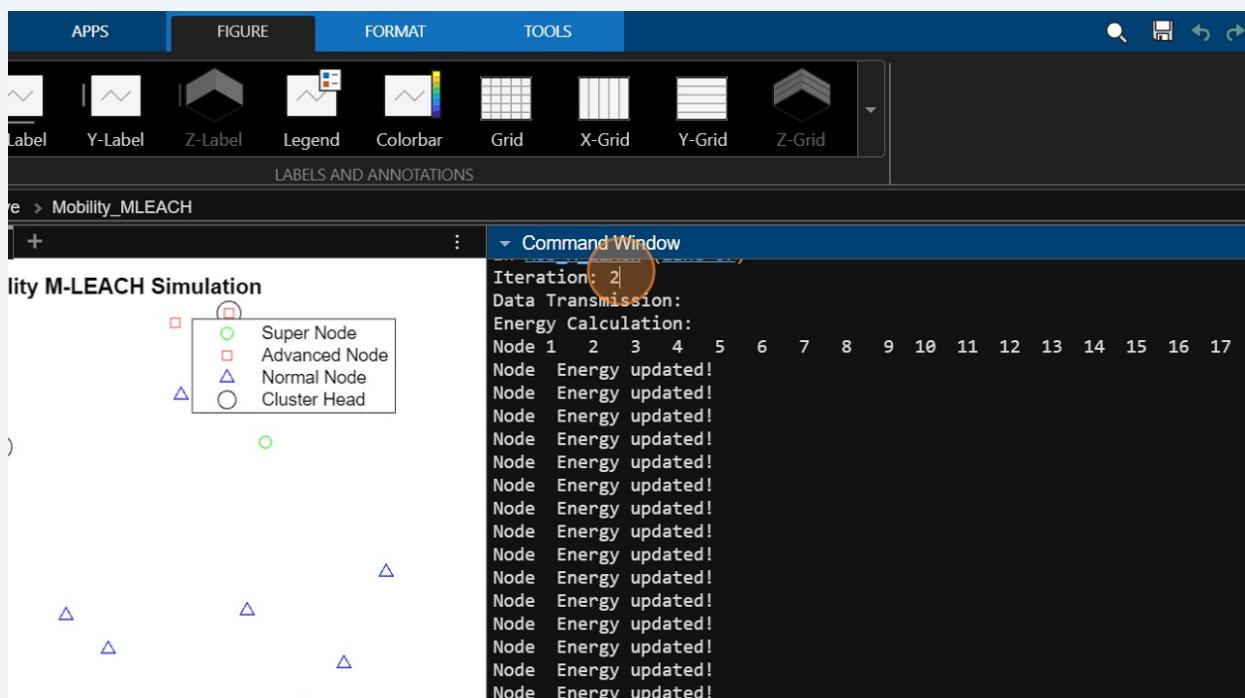
54

Due to mobility you can see the energy dissipation here as i started with 100 as initial Energy but now it started decreasing with number of iterations

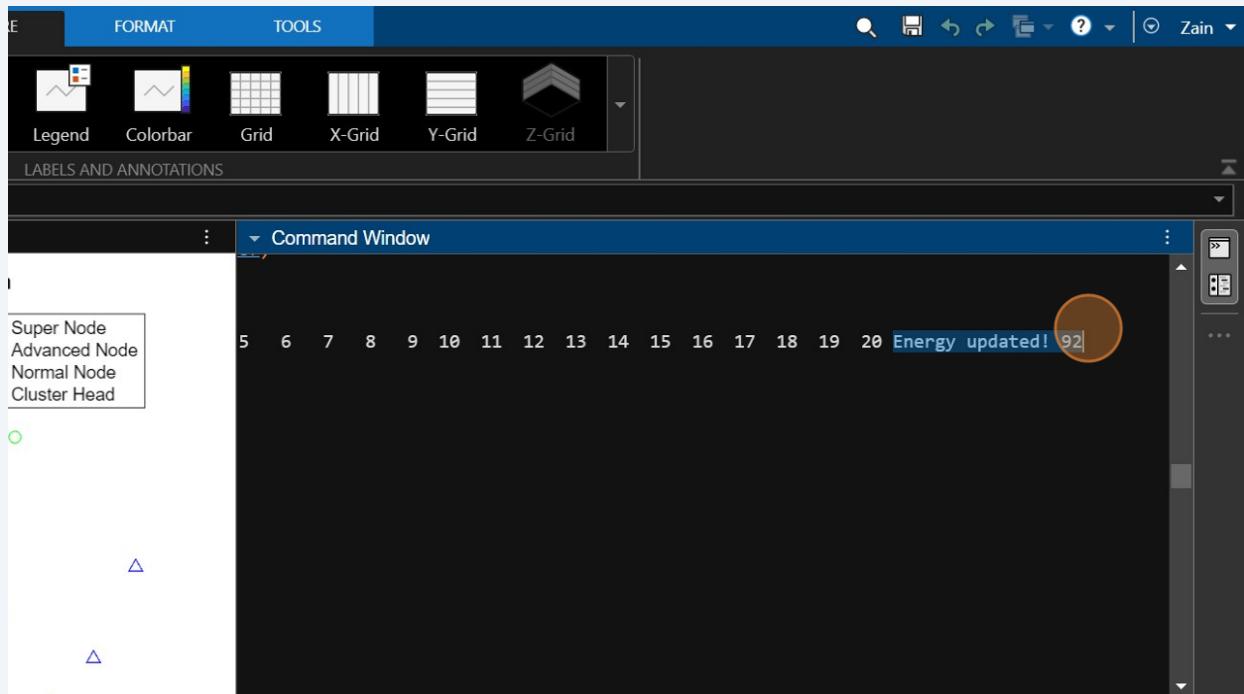


55

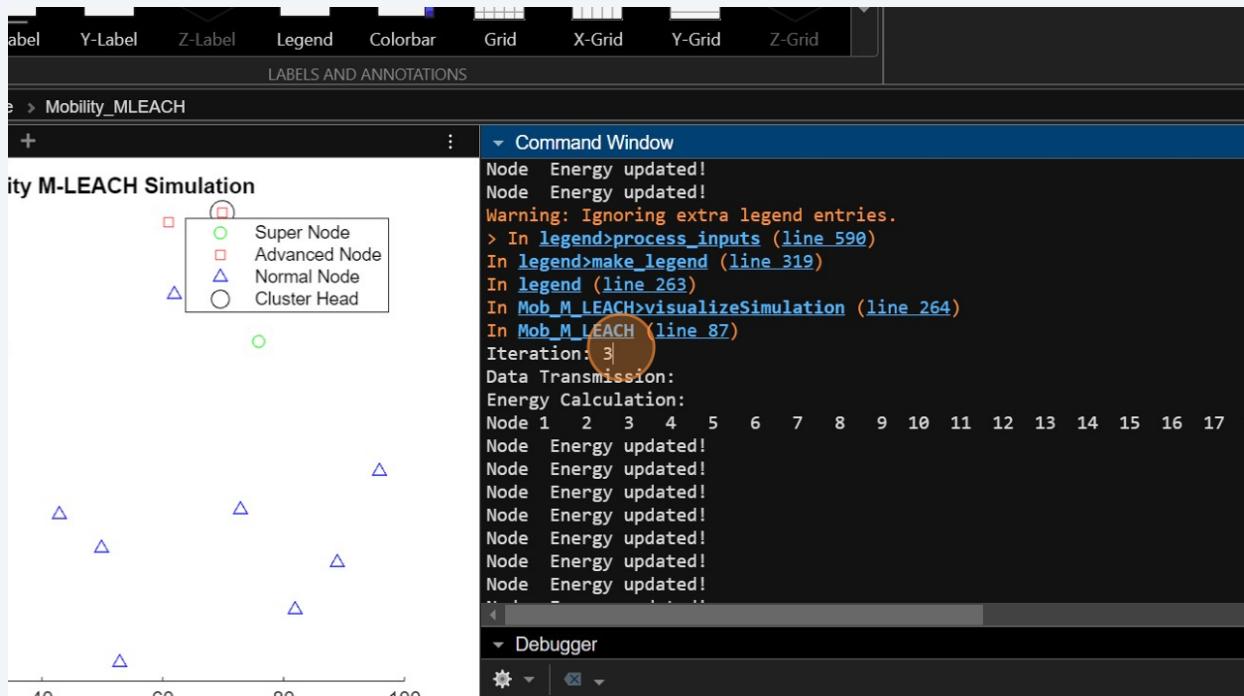
Iteration 2



56 Updated energy



57 Iteration 3



58 Energy updated

The screenshot shows the MATLAB interface with the Command Window active. The window displays a series of numbers from 7 to 20 followed by the message "Energy updated! 86". A blue circle highlights the number 86. To the left of the Command Window, there is a legend for node types: Super Node (green circle), Advanced Node (blue triangle), Normal Node (orange circle), and Cluster Head (yellow triangle). Below the legend is a horizontal axis with tick marks at 80 and 100.

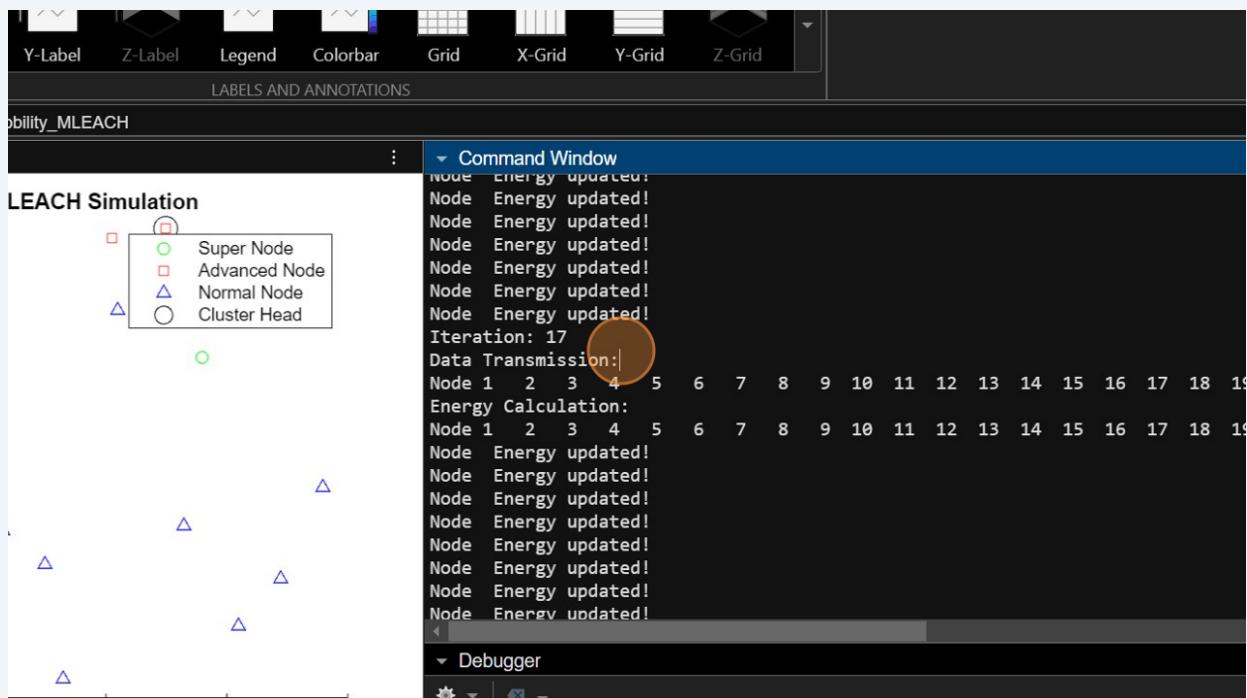
```
end entries.  
(line 590)  
319)  
ulation (line 264)  
7 8 9 10 11 12 13 14 15 16 17 18 19 20 Energy updated! 86|  
8  
△  
△  
△  
△  
80 100
```

59 Energy keeps on decreasing with the mobility and transmission

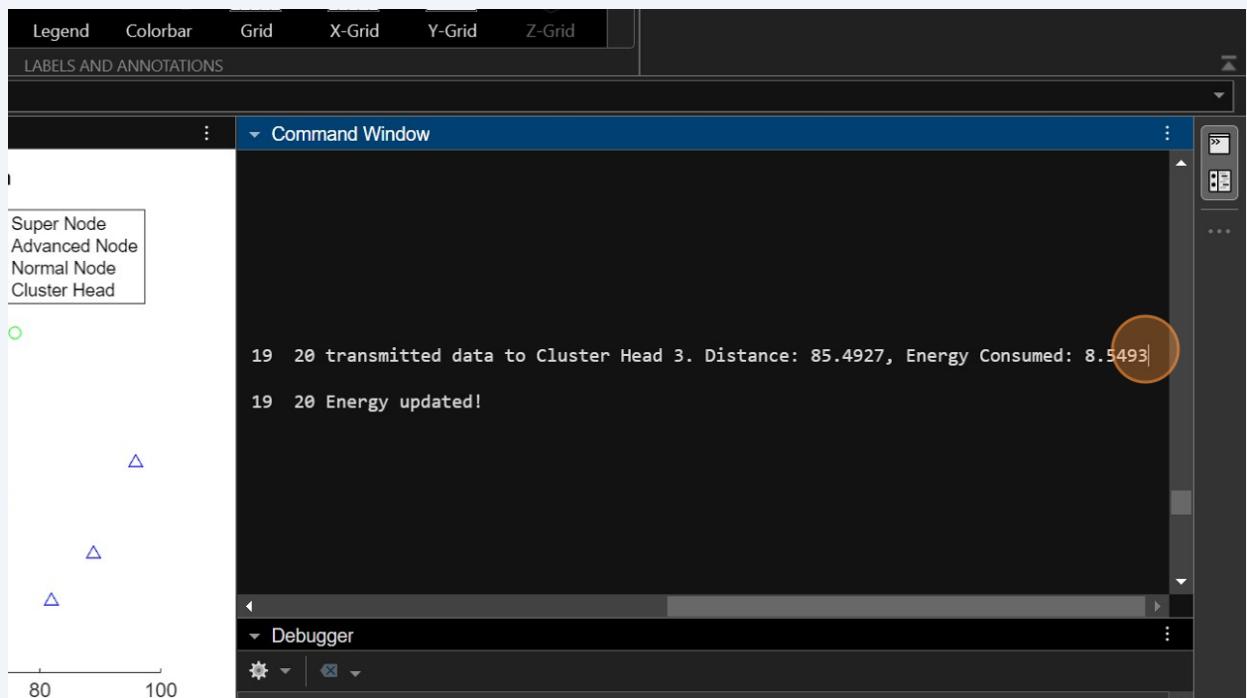
The screenshot shows the MATLAB interface with the Command Window active. The window displays a series of numbers from 5 to 20 followed by the message "Energy updated! 80". A blue circle highlights the number 80. To the left of the Command Window, there is a legend for node types: Super Node (green circle), Advanced Node (blue triangle), Normal Node (orange circle), and Cluster Head (yellow triangle). Below the legend is a horizontal axis with tick marks at 80 and 100.

```
inputs (line 590)  
(line 319)  
izeSimulation (line 264)  
87)  
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 Energy updated! 80|  
8  
△  
△  
△  
△  
80 100
```

60 Iteration 17



61 Sample output



62 The cluster head to whom data is transmitted is also mentioned

The screenshot shows a software interface with a "Command Window" tab open. The window displays the following text:

```
19 20 transmitted data to Cluster Head 3. Distance: 85.4927, Energy Consumed: 8.5493
19 20 Energy updated!
```

A brown circle highlights the text "Cluster Head 3.". On the left side of the interface, there is a legend with four entries: "Super Node" (green circle), "Advanced Node" (blue triangle), "Normal Node" (blue triangle), and "Cluster Head" (green circle). Below the legend is a coordinate system with a horizontal axis from 80 to 100 and a vertical axis from 100 to 80.

63 On"Iteration: 17
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster
..."

The screenshot shows a software interface with a "Command Window" tab open. The window displays the following text:

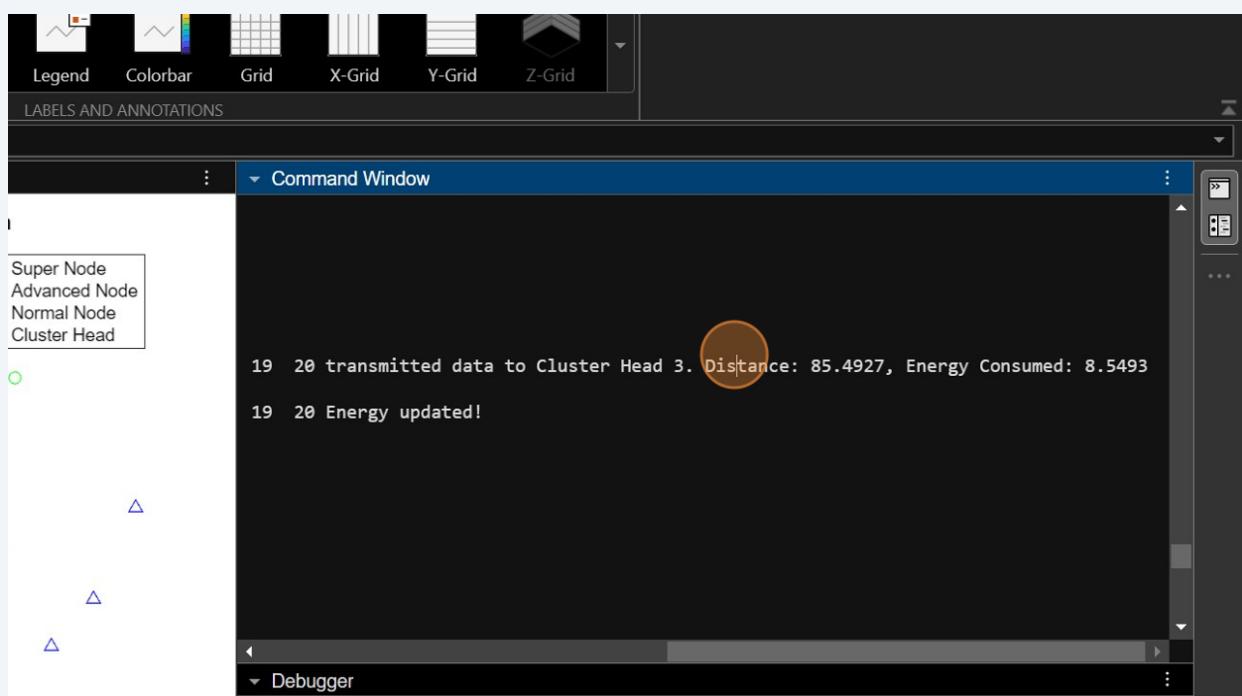
```
19 20 transmitted data to Cluster Head 3. Distance: 85.4927, Energy Consumed: 8.5493
19 20 Energy updated!
```

A brown circle highlights the text "Cluster Head 3.". On the left side of the interface, there is a legend with four entries: "Super Node" (green circle), "Advanced Node" (blue triangle), "Normal Node" (blue triangle), and "Cluster Head" (green circle). Below the legend is a coordinate system with a horizontal axis from 80 to 100 and a vertical axis from 100 to 80.

64

On "Iteration: 18
Data Transmission:

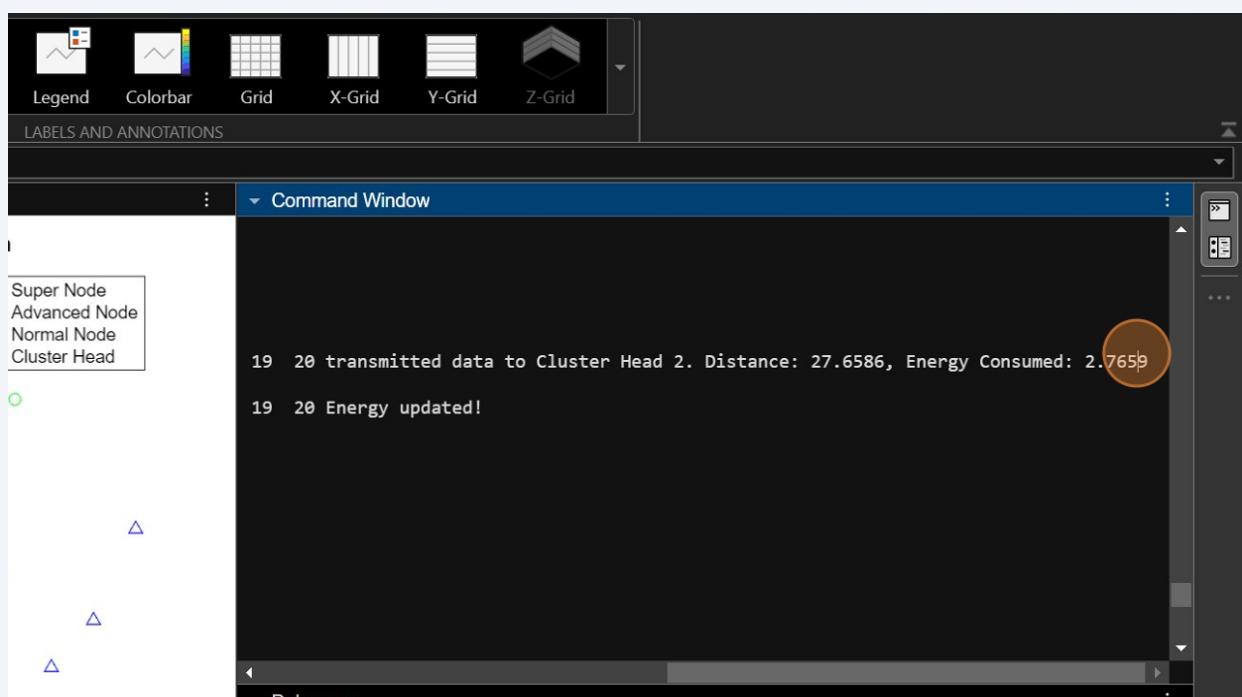
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster
..."



65

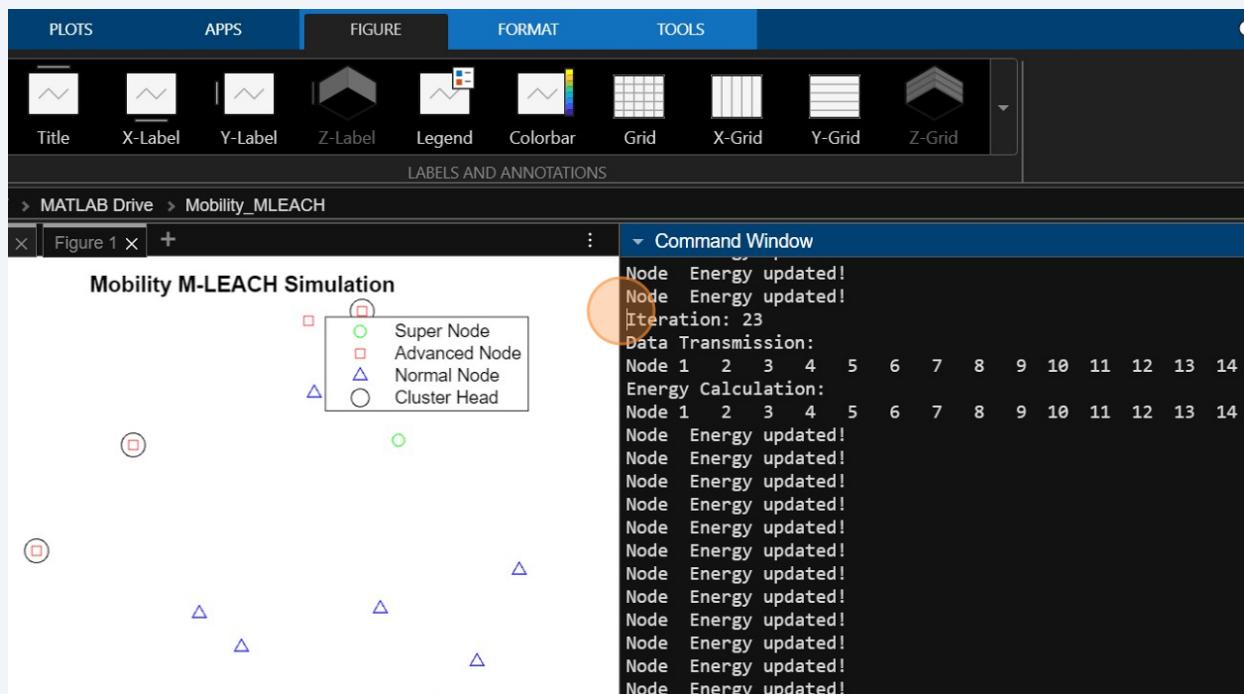
On "Iteration: 20

Data Transmission: (Energy COnsumed Distance values calculated



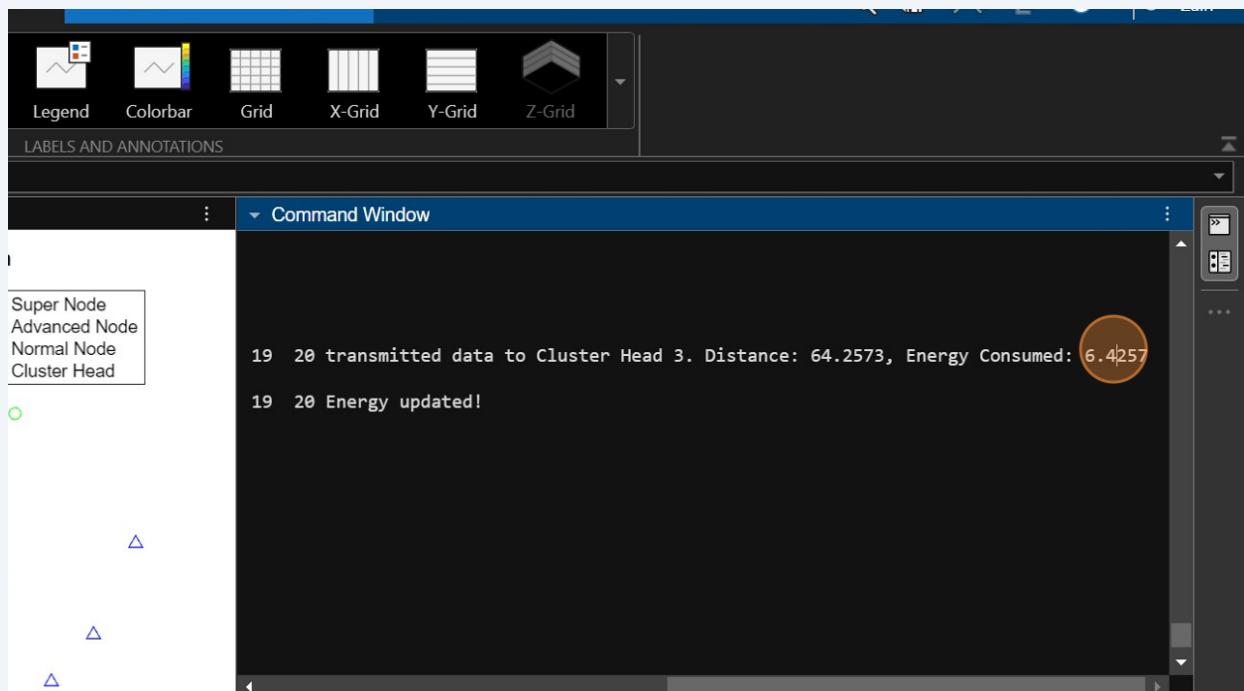
66

Click "Iteration: 23
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster
..."

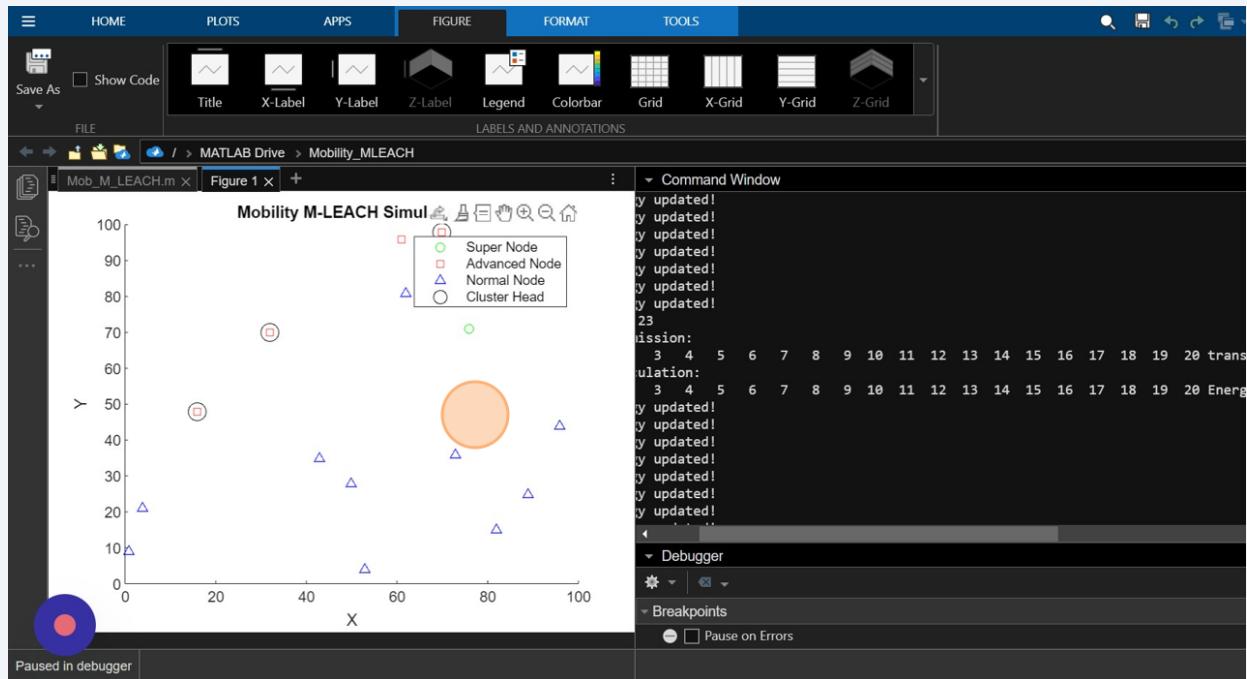


67

On "Iteration: 23
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster
..."



68 sample output at iteration 23



```
% Mobility-MLEACH Implementation in MATLAB with Implemntation of Energy
and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network

clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
'energy', initialEnergy, ... % Set initial energy level
'clusterHead', false, ... % Initialize as non-cluster heads
'clusterID', 0, ... % Initialize cluster ID
'nodeType', 'normal'); % Set node type as normal initially

% Assign node types
nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
nodeTypeCounts = [numSuperNodes, numAdvancedNodes, numNormalNodes]; % Count of each node type
startIndex = 1; % Start index for node assignment
for i = 1:length(nodeTypes)
    endIndex = startIndex + nodeTypeCounts(i) - 1; % Calculate end index for node assignment
    [nodes(startIndex:endIndex).nodeType] = deal(nodeTypes{i}); % Assign node types to nodes
    startIndex = endIndex + 1; % Update start index for next node type
end
```

```

% Cluster Initialization - Select cluster heads only from advanced nodes
advancedNodeIndices = find(strcmp({nodes.nodeType}, 'advanced')); % Find
indices of advanced nodes
if ~isempty(advancedNodeIndices) % Check if there are advanced nodes available
clusterHeadIndices =
advancedNodeIndices(randperm(length(advancedNodeIndices), numClusters)); %
Randomly select cluster heads from advanced nodes
for i = 1:length(clusterHeadIndices)
nodes(clusterHeadIndices(i)).clusterHead = true; % Mark selected nodes as cluster
heads
nodes(clusterHeadIndices(i)).clusterID = i; % Assign cluster ID to selected cluster
heads
end
else
disp('Error: No advanced nodes available to select as cluster heads.');?>
Display
error message if no advanced nodes are available
end

% Simulation Loop
for iter = 1:maxIterations % Iterate through each time step
disp(['Iteration: ' num2str(iter)]); % Display current iteration number

`% Mobility Model (Random Waypoint) nodes = updateNodePositions(nodes,
areaWidth, areaHeight); % Update node positions based on mobility model %
Energy Model (Simple) - Update energy calculations nodes = updateEnergy(nodes,
RSSI); % Update energy levels of nodes % Cluster Formation nodes =
leachClustering(nodes, numClusters); % Perform clustering based on LEACH
protocol % Data Transmission (Simple) - Display data transmission information
disp('Data Transmission:'); for i = 1:length(nodes) if ~nodes(i).clusterHead % If
node is not a cluster head clusterHeadID = find([nodes.clusterID] ==
nodes(i).clusterID & [nodes.clusterHead], 1); % Find the cluster head for the node
if ~isempty(clusterHeadID) % If cluster head is found distance = sqrt((nodes(i).x -
nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2); % Calculate
distance between node and cluster head energyConsumption = distance * 0.1; % %
Calculate energy consumption for data transmission disp(['Node '
num2str(nodes(i).id) ' transmitted data to Cluster Head ' num2str(clusterHeadID)
... '. Distance: ' num2str(distance) ', Energy Consumed: '
num2str(energyConsumption)]); % Display data transmission information end end
en

```