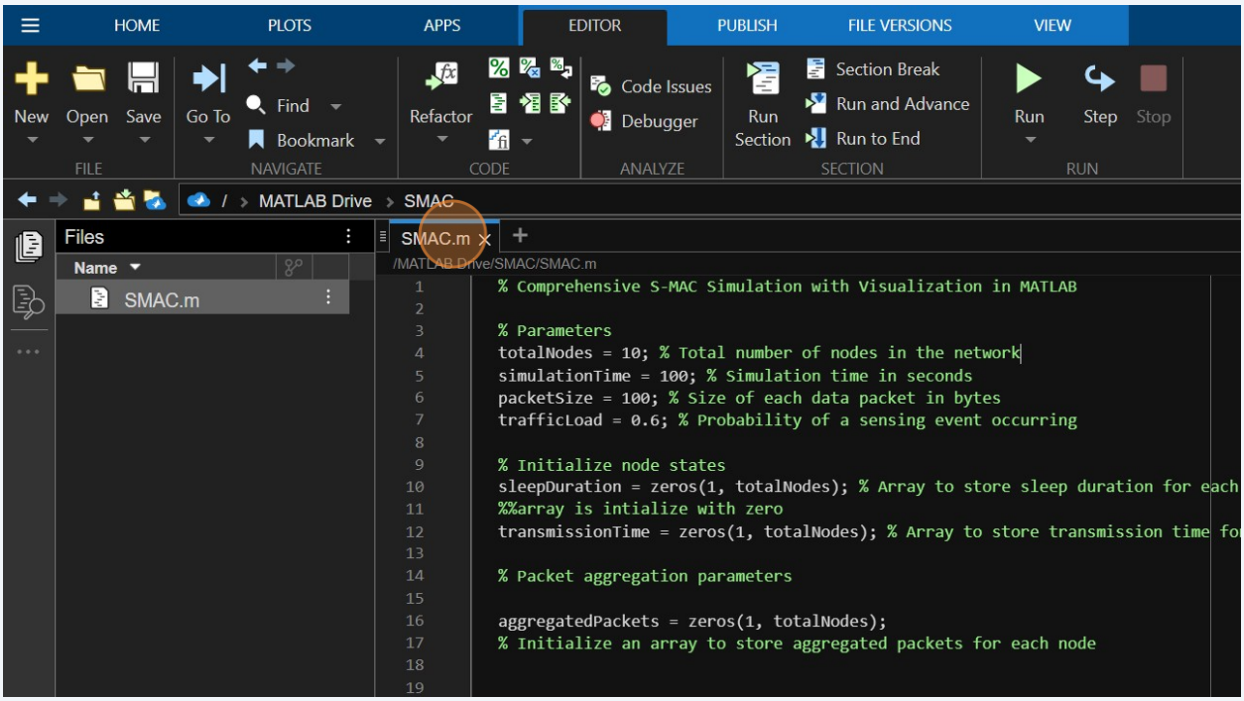
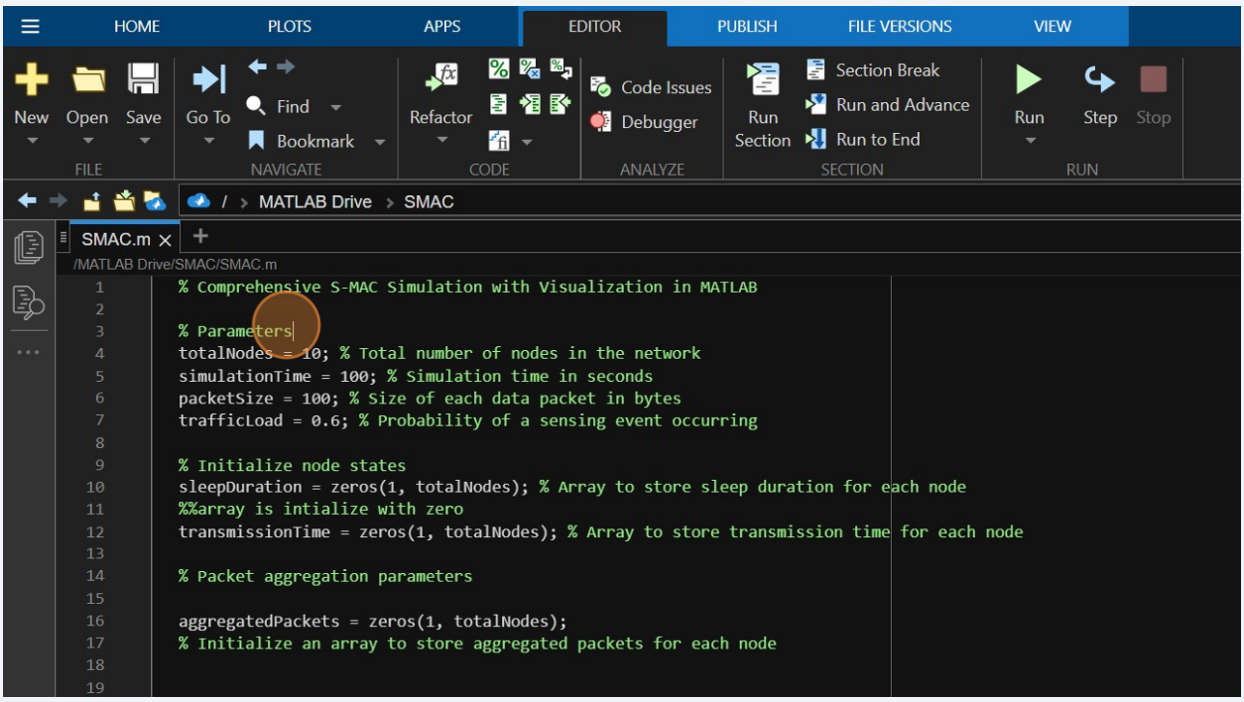
# Step-by-step Implementation of SMAC for network simulation on MATLAB (with code)

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

2 We are going to Implement SMAC So First Lets know about SMAC:
**The Sensor-MAC (S-MAC)** protocol is a specialized medium access control (MAC) protocol tailored for the distinct requirements of wireless sensor networks (WSNs). Prioritizing energy optimization, S-MAC introduces a strategic **duty cycling mechanism**, enabling nodes to alternate between **active and sleep states**, effectively curbing unnecessary energy consumption during periods of inactivity and **alleviating the common issue of idle listening** found in traditional MAC protocols. In the simulated environment, each sensor node adheres to a predefined duty cycle, periodically awakening for synchronization with neighboring nodes. This synchronization involves the exchange of control packets—SYNC for periodic alignment, **RTS** to signal data transmission intent, and **CTS** for confirmation. Implemented with fixed sleep and awake periods, the protocol ensures a consistent sleep-wake cycle, enhancing energy management for prolonged network longevity. The communication during wake periods unfolds in three phases: SYNC for synchronization, RTS signaling data transmission, and CTS confirming readiness. This meticulous design minimizes energy consumption and addresses hidden terminal problems through the **RTS/CTS handshake**, ensuring efficient and collision-free communication in resource-constrained wireless sensor networks.



3 Now lets see the implemented code first , Starting with the initialization of Parameters



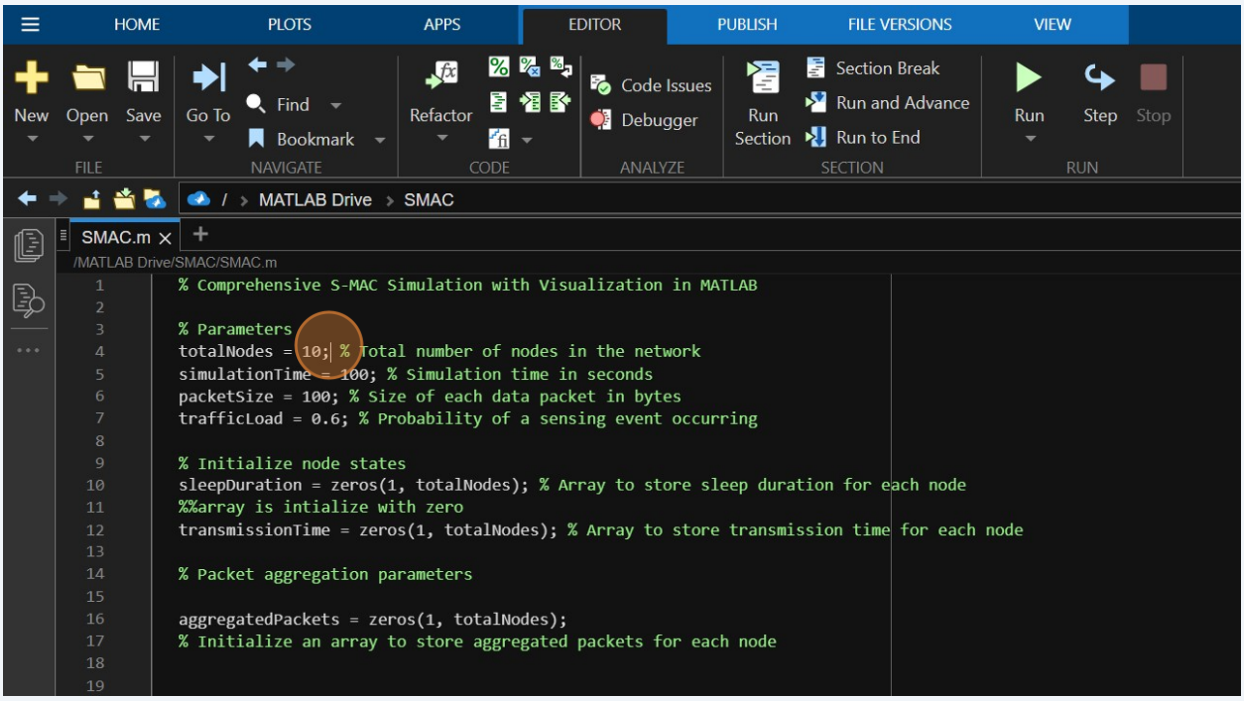
**4** First parameter is the totalNodes that is selfexplanatory totalNodes:

• **Definition:** This parameter represents the total number of nodes within the wireless sensor network (WSN) being simulated.

• **Usage:** It influences the size of arrays and loops in the simulation, as the script iterates over each node during the simulation time.
2nd is
simulationTime:

• **Definition:** This parameter specifies the total duration of the simulation in seconds.

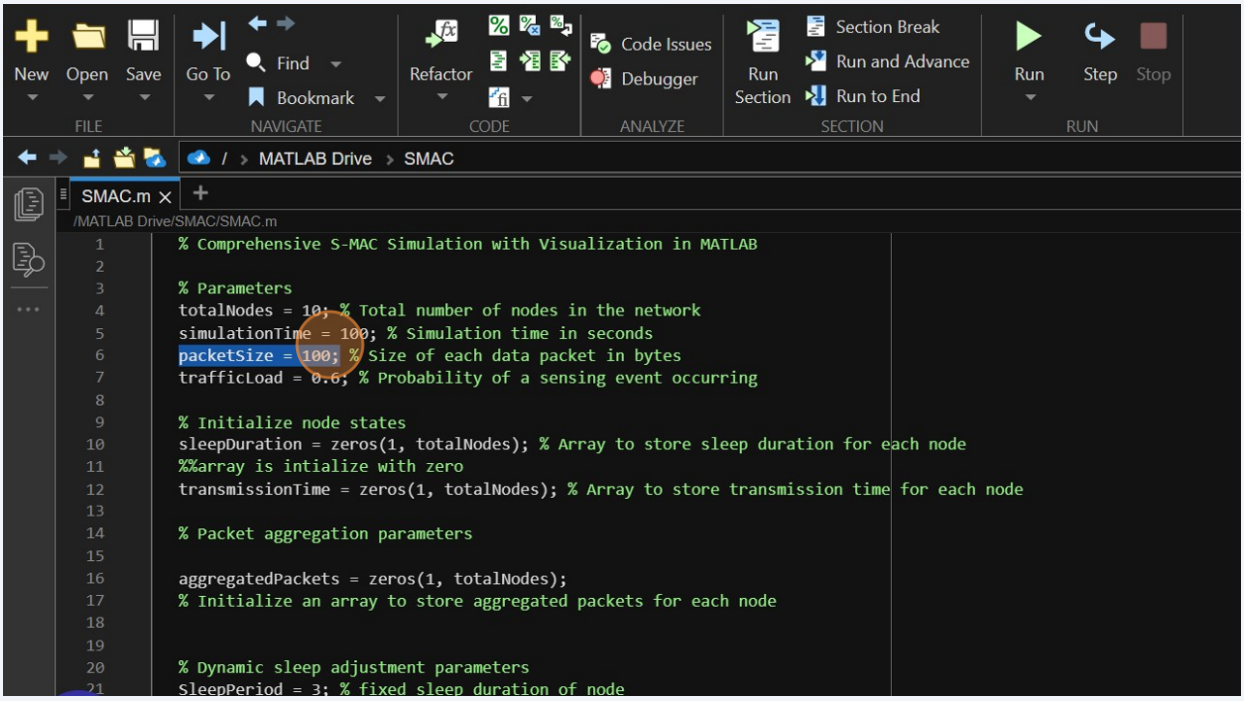
• **Usage:** It determines the overall time span for which the simulation will run, influencing the number of iterations in the simulation loop.



5 Then comes the third one packetSize:

• **Definition:** packetSize denotes the size of each data packet in bytes.

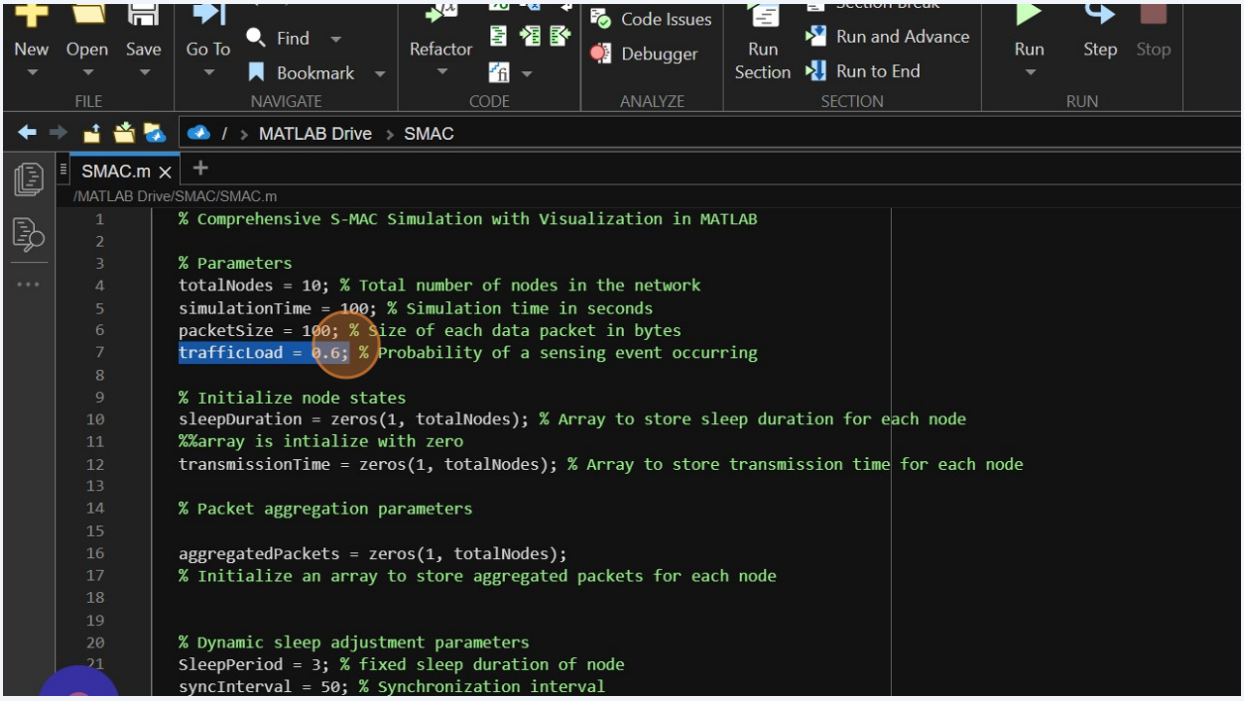
• **Usage:** It is crucial in modeling data transmission as it represents the amount of information exchanged between nodes during communication events.



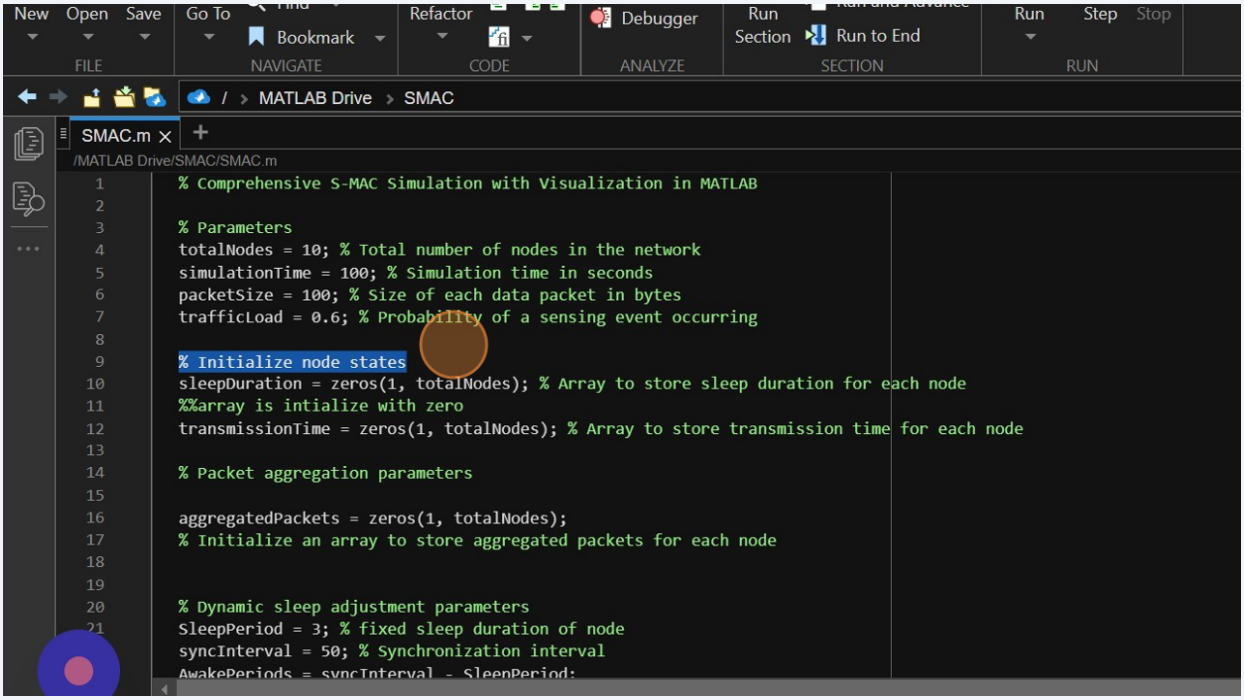
## 6 trafficLoad:

• **Definition:** The trafficLoad parameter represents the probability of a sensing event occurring at a given node during each simulation time step.

• **Usage:** It is employed to simulate the occurrence of sensing events in the network. A higher trafficLoad implies a greater likelihood of nodes sensing and triggering communication events.



**7** Now lets have some arrays to store the states of nodes



**8** The purpose is to initialize the variable sleepDuration. Let me break it down:

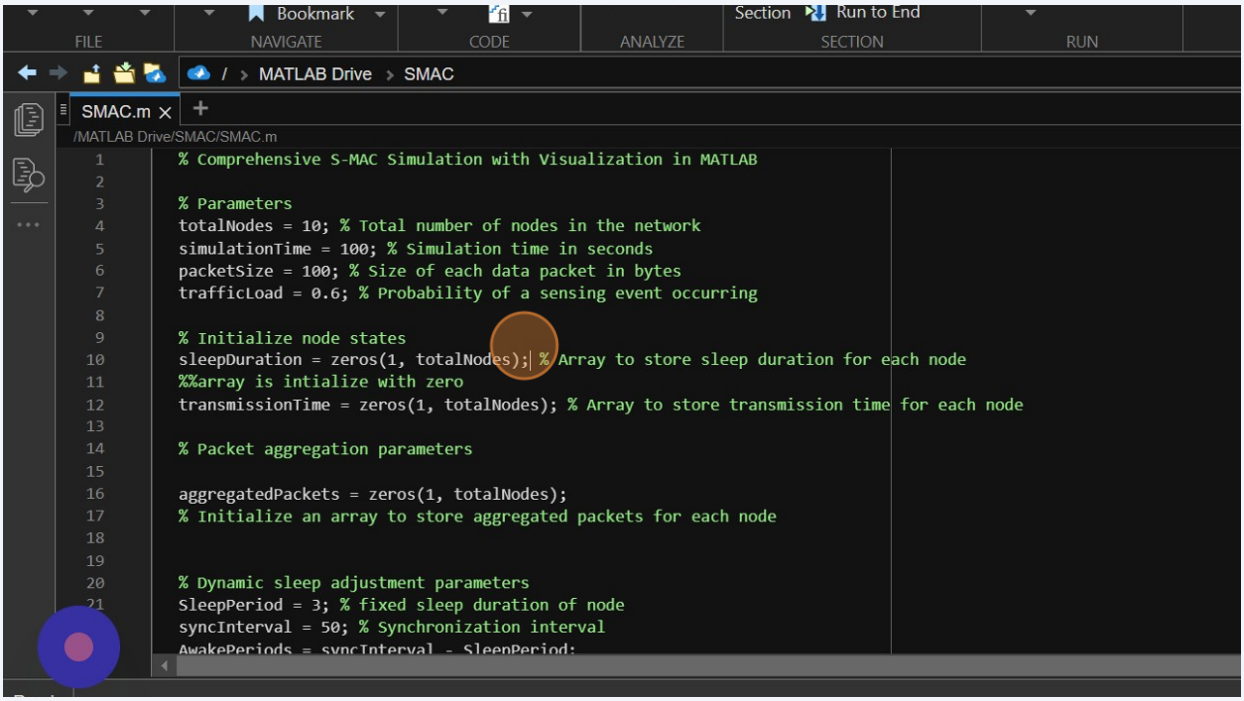
• zeros(1, totalNodes): This part creates a row vector of size totalNodes filled with zeros. The zeros function in MATLAB generates an array of zeros, and in this case, it creates a row vector with totalNodes elements.

• sleepDuration = ...: This part assigns the newly created row vector to the variable sleepDuration. So, sleepDuration is now a row vector containing totalNodes elements, all initialized to zero.

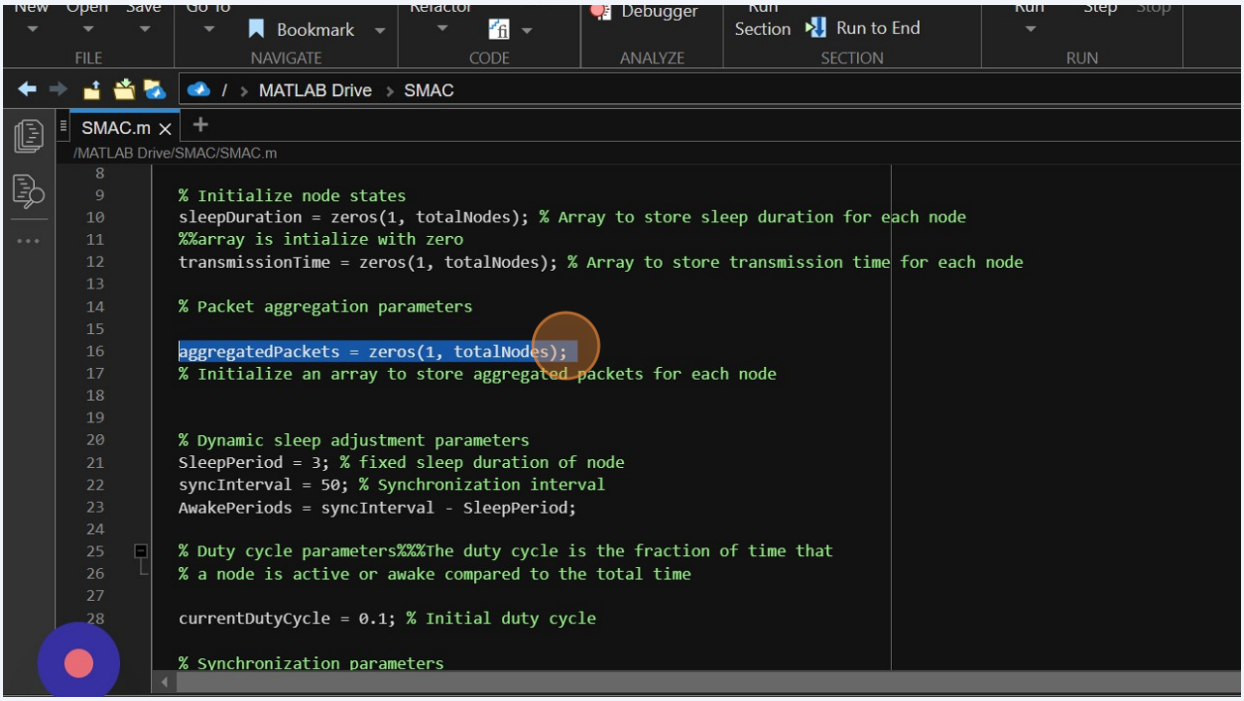
zeros(1, totalNodes): This part creates a row vector of size totalNodes filled with zeros. The zeros function in MATLAB generates an array of zeros, and in this case, it creates a row vector with totalNodes elements.

• transmissionTime = ...: This part assigns the newly created row vector to the variable transmissionTime. So, transmissionTime is now a row vector containing totalNodes elements, all initialized to zero.

This line of code sets up an array to keep track of the transmission time for each node in the network. Each element of the array corresponds to a specific node, and initially, all transmission times are set to zero. As the simulation progresses, this array will be updated to reflect the transmission time for each node based on the logic and conditions within the simulation loop.



9 The line aggregatedPackets = zeros(1, totalNodes); initializes an array named aggregatedPackets with a length of totalNodes, where each element is initially set to zero. This array is intended to store the count of aggregated packets for each node in the network during the simulation.

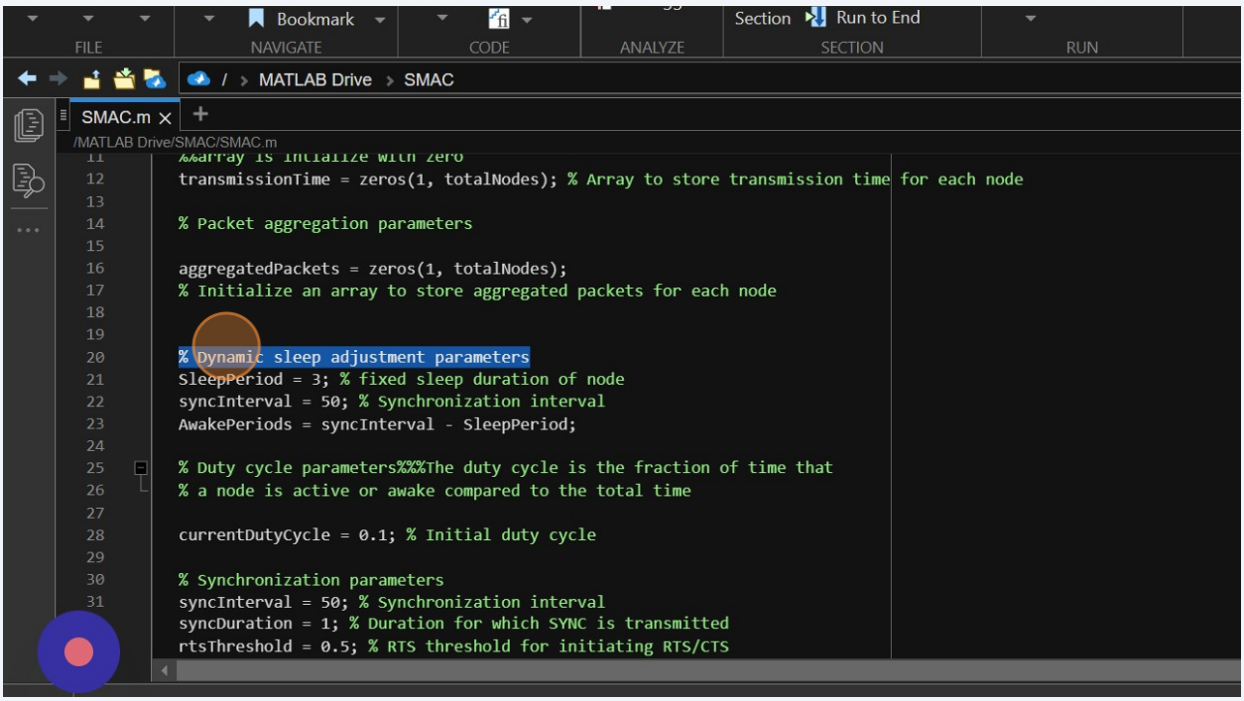


**10** These lines define dynamic sleep adjustment parameters for the S-MAC simulation. Here's a breakdown:

• SleepPeriod = 3;: This parameter represents the fixed sleep duration of a node. Each node will go to sleep for a duration of 3 simulation time units during its sleep period.

• syncInterval = 50;: This parameter sets the synchronization interval, indicating the total duration of the cycle (including both sleep and awake periods) for each node. In this case, it's set to 50 simulation time units.

• AwakePeriods = syncInterval - SleepPeriod;: This line calculates the awake periods within the synchronization interval. It subtracts the sleep duration (SleepPeriod) from the total synchronization interval (syncInterval), resulting in the duration for which the node remains awake. The variable AwakePeriods will be used in the simulation loop to adjust the duty cycle.

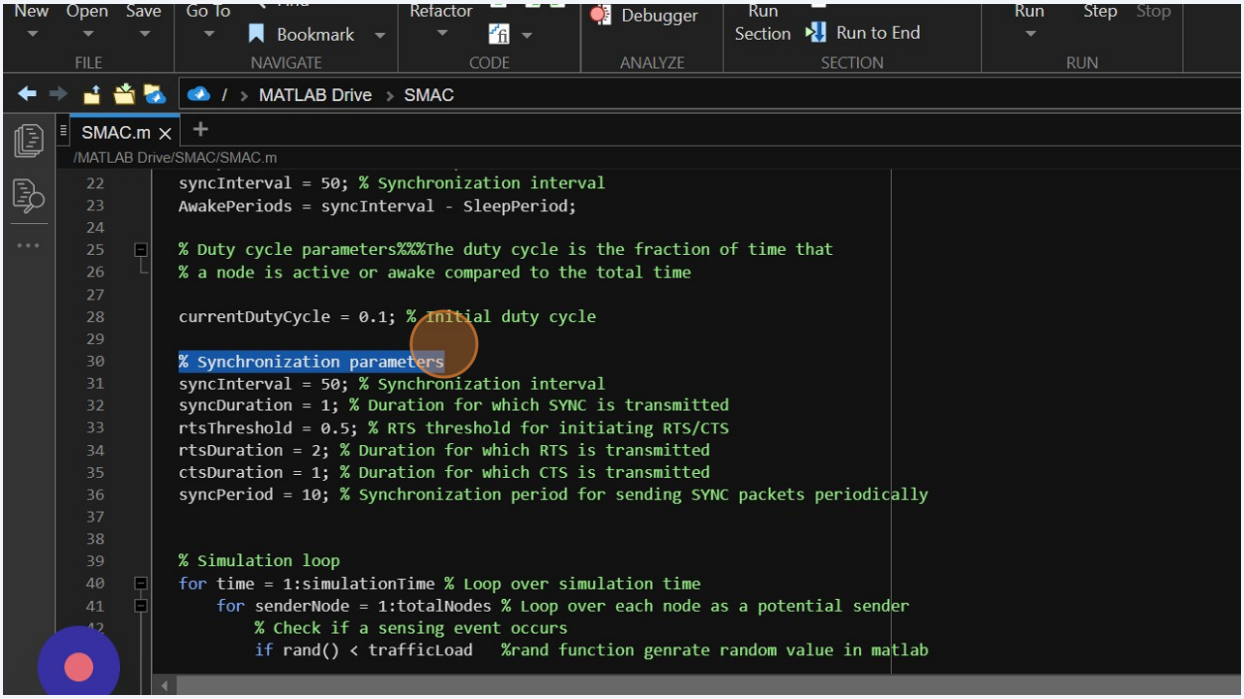


**11** And then we have set current Duty cycle to 0.1 from where the whole simulation starts and going to set some synchronization parameters related to the synchronization process in the S-MAC simulation:

• syncInterval = 50;: This parameter sets the synchronization interval, representing the total duration of the cycle during which nodes synchronize and exchange control packets.

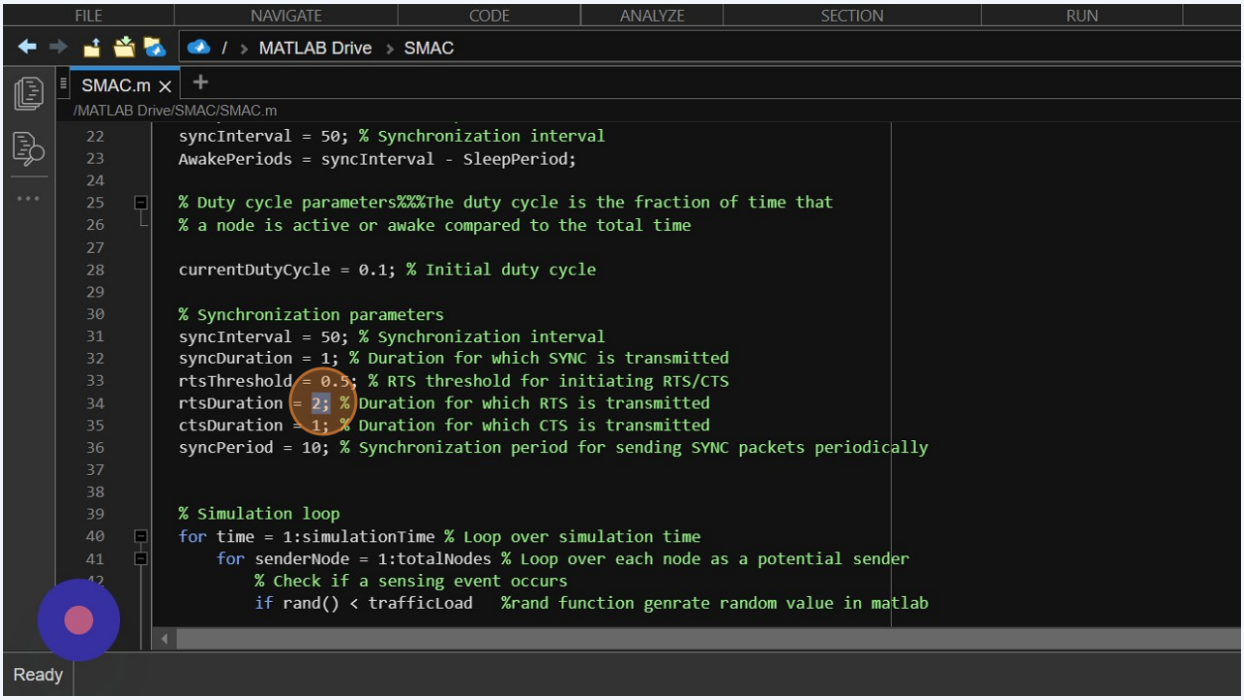
• syncDuration = 1;: It specifies the duration for which SYNC packets are transmitted during the synchronization phase.

• rtsThreshold = 0.5;: This parameter sets the RTS (Request to Send) threshold, representing the probability threshold for a neighboring node to initiate the RTS/CTS process. In the simulation, if a random number is less than this threshold, a neighbor initiates the process.



12

• rtsDuration = 2;: It specifies the duration for which the RTS packet is transmitted during the RTS phase.

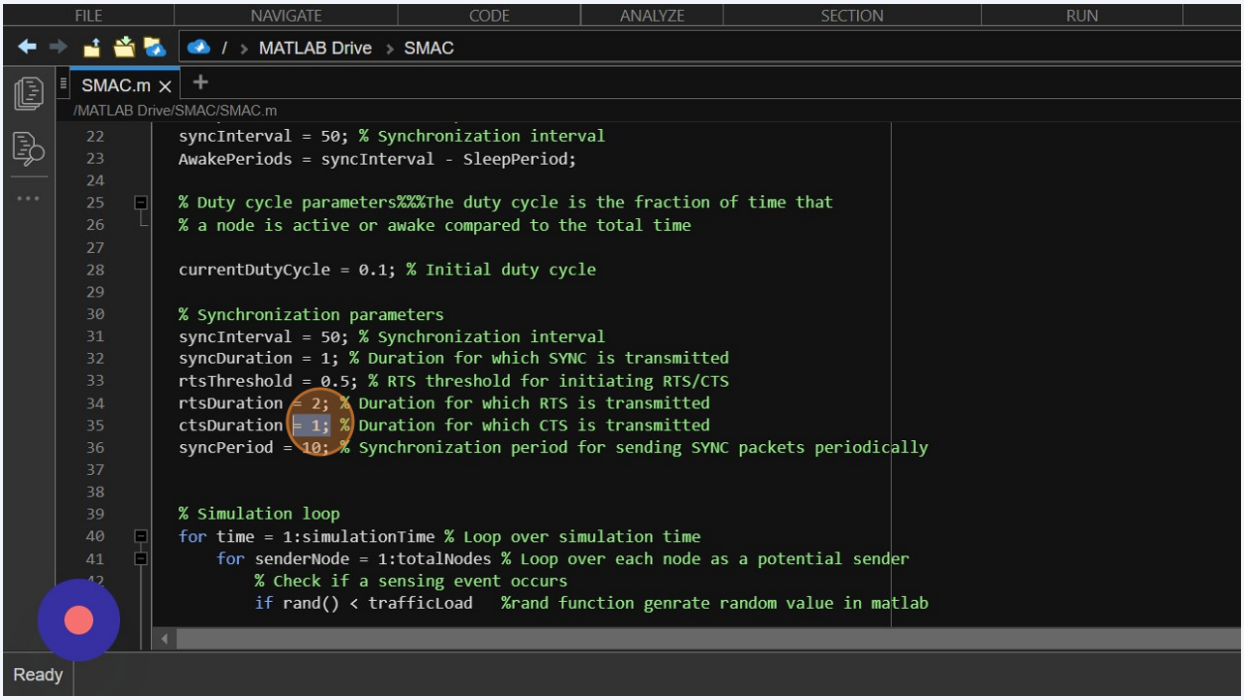


13

• ctsDuration = 1;: It specifies the duration for which the CTS (Clear to Send) packet is transmitted in response to an RTS packet.

• syncPeriod = 10;: This parameter sets the synchronization period, indicating how often SYNC packets are transmitted periodically during the simulation.

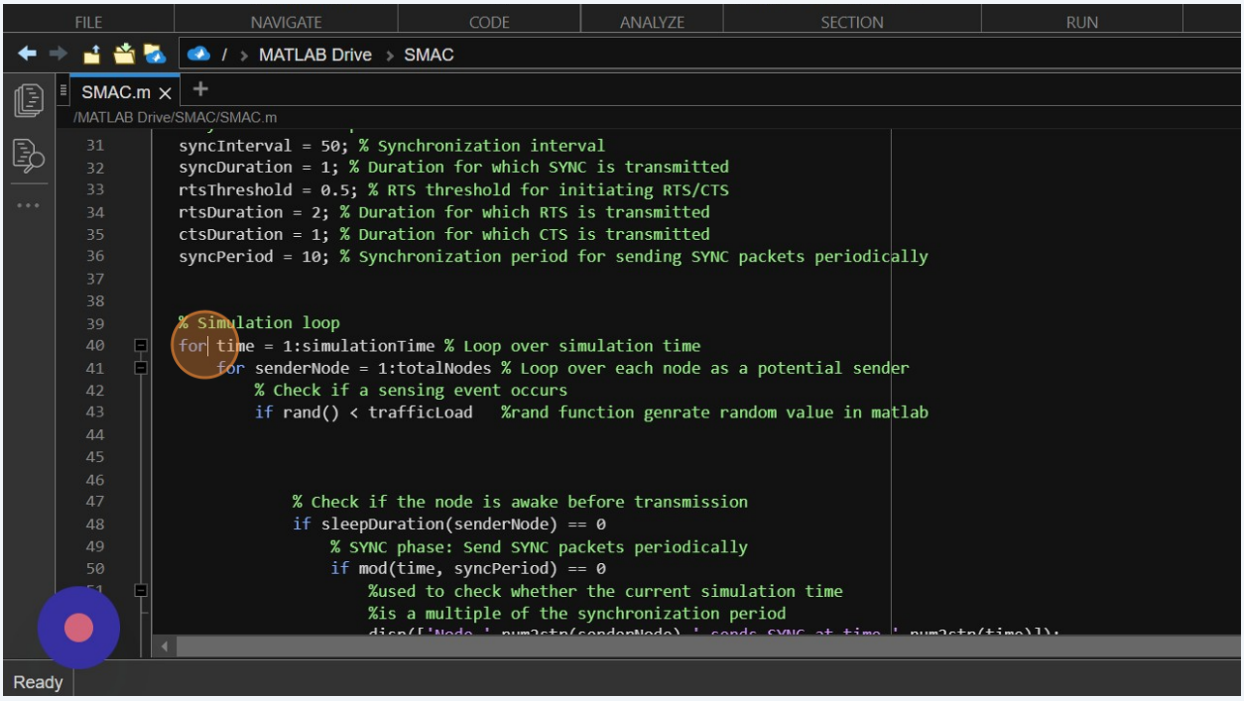
These parameters play a crucial role in governing the timing and duration of various synchronization-related activities, contributing to the overall coordination and communication efficiency in the S-MAC protocol simulation



**14** Now here comes the simulation Loop
Here's an overview of the key components within the loop:

- **Outer Loop (**for time = 1:simulationTime**):**

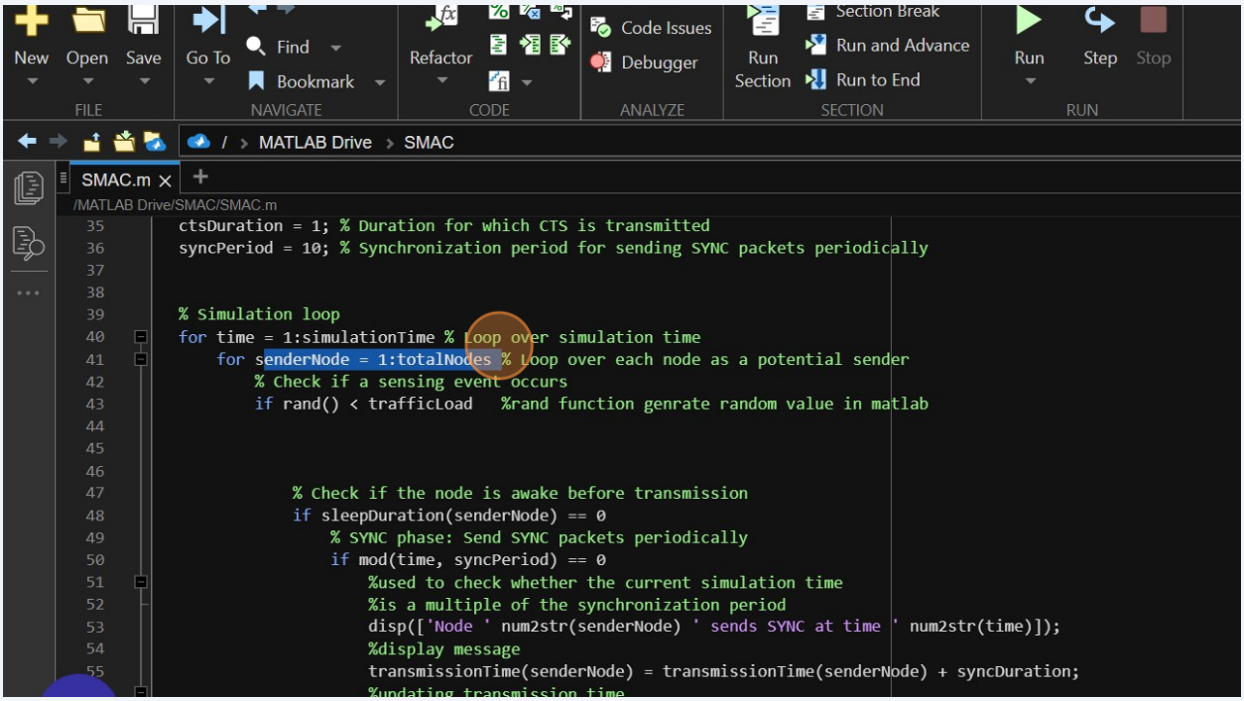
- The outer loop iterates over the simulation time, progressing the simulation forward in discrete time steps.



## 15 **Inner Loop (**for senderNode = 1:totalNodes**):**

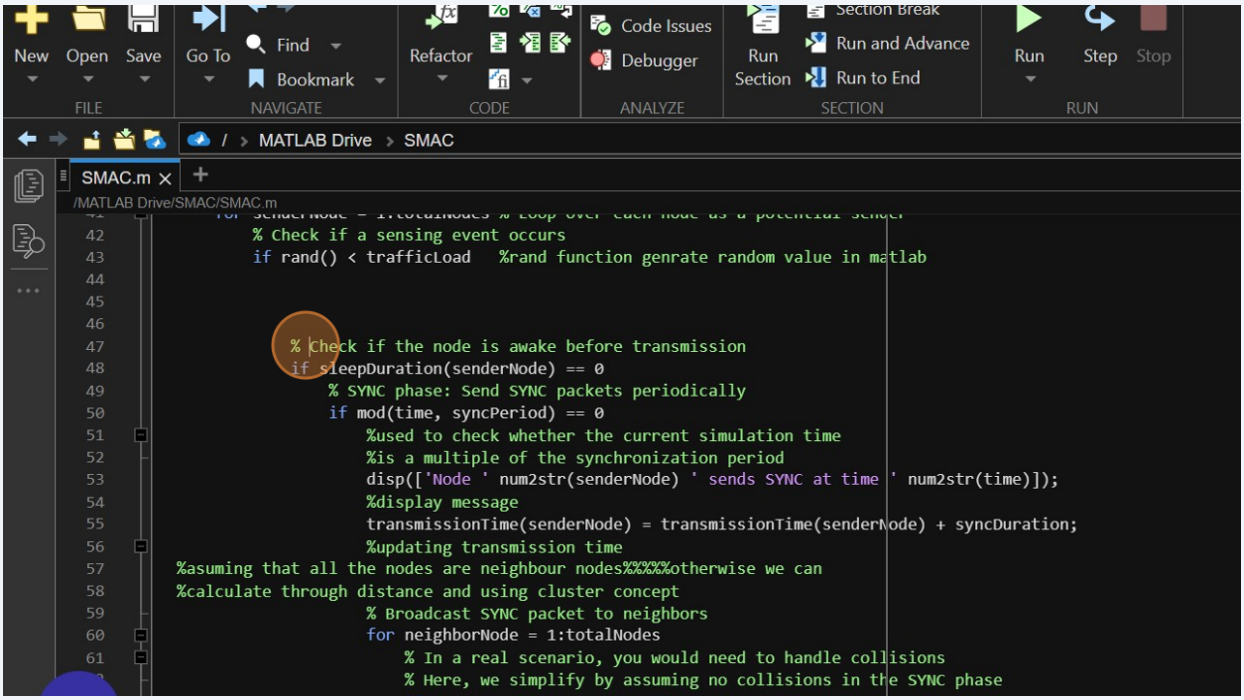
• The inner loop iterates over each node in the network, considering each node as a potential sender. **Sensing Event Check (**if rand() < trafficLoad**):**

• A random number is generated, and if it is less than the specified traffic load probability (trafficLoad), it is considered a sensing event. This models the probability of a node sensing an event.



16 • **Node Wakefulness Check (**if sleepDuration(senderNode) == 0**):**

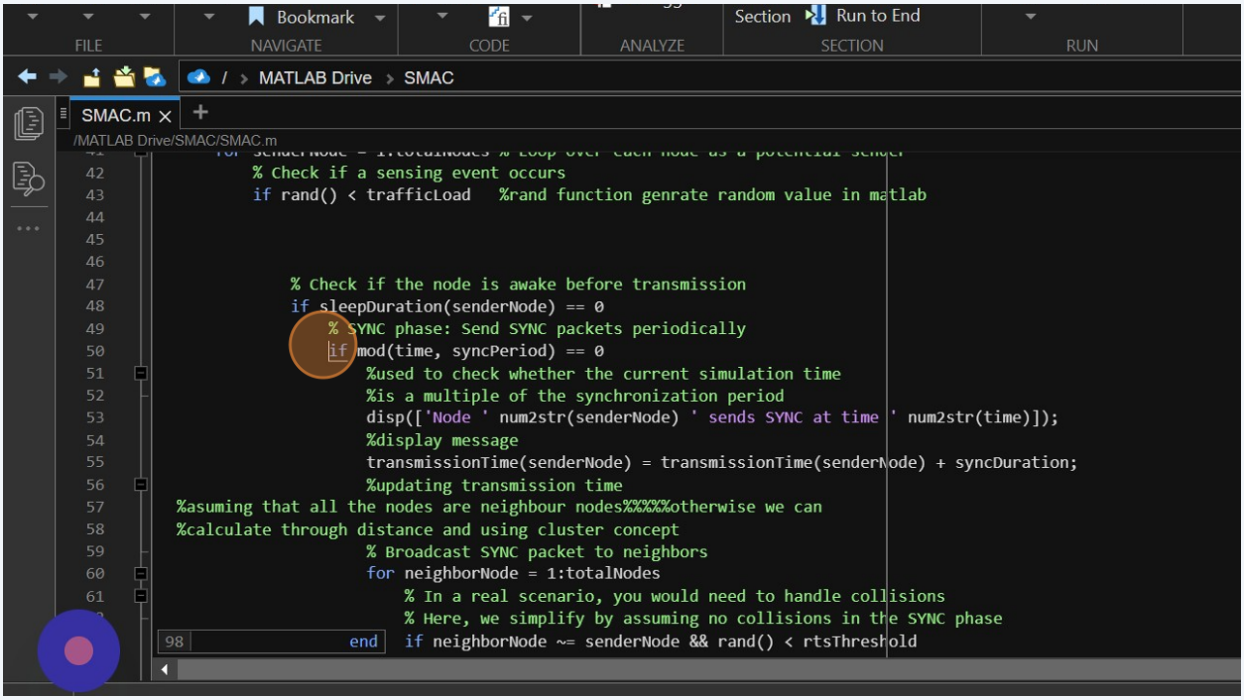
• Checks if the node is awake (sleep duration is zero) before allowing any transmission.



17 • **SYNC Phase (**if mod(time, syncPeriod) == 0**):**

• During the SYNC phase, nodes periodically send SYNC packets. The condition checks if the current simulation time is a multiple of the synchronization period (syncPeriod).

• If true, the node broadcasts SYNC packets to neighbors, updating transmission times, and simulating synchronization.



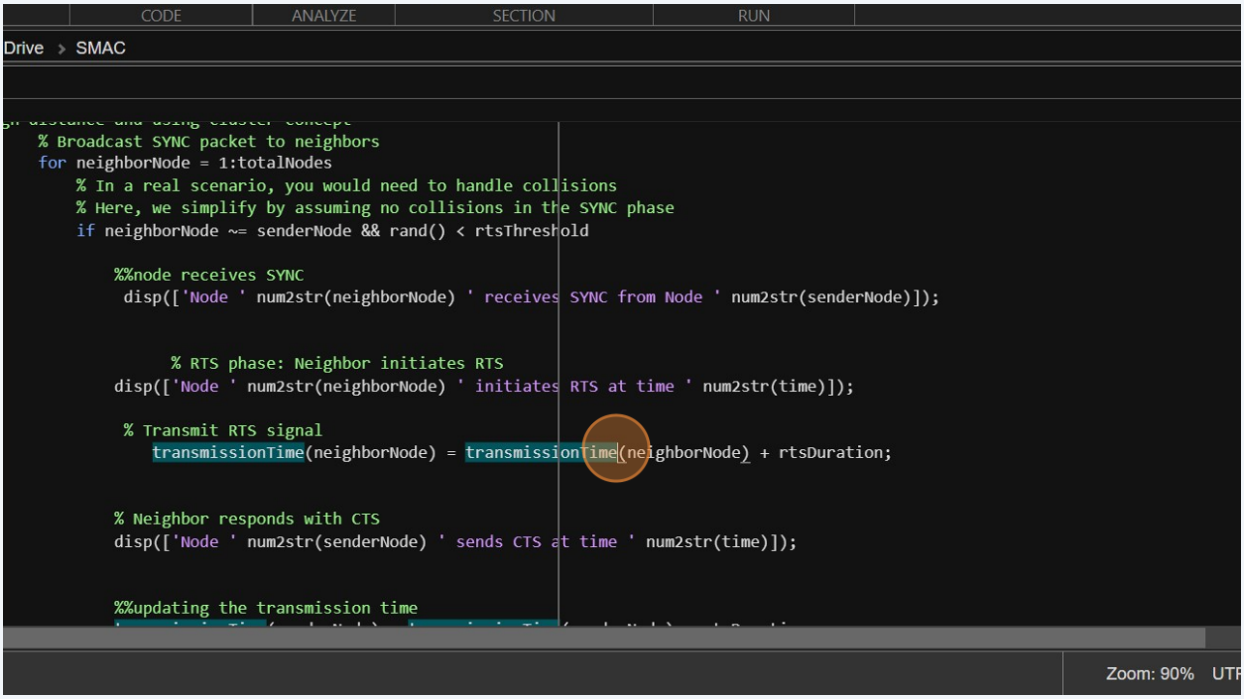
18 • **RTS/CTS Handshake:**

• If a neighbor receives SYNC and satisfies a random threshold (rand() < rtsThreshold), it initiates the RTS/CTS handshake:

• The neighbor initiates RTS, updating its transmission time.

• The sender node responds with CTS, updating its transmission time.

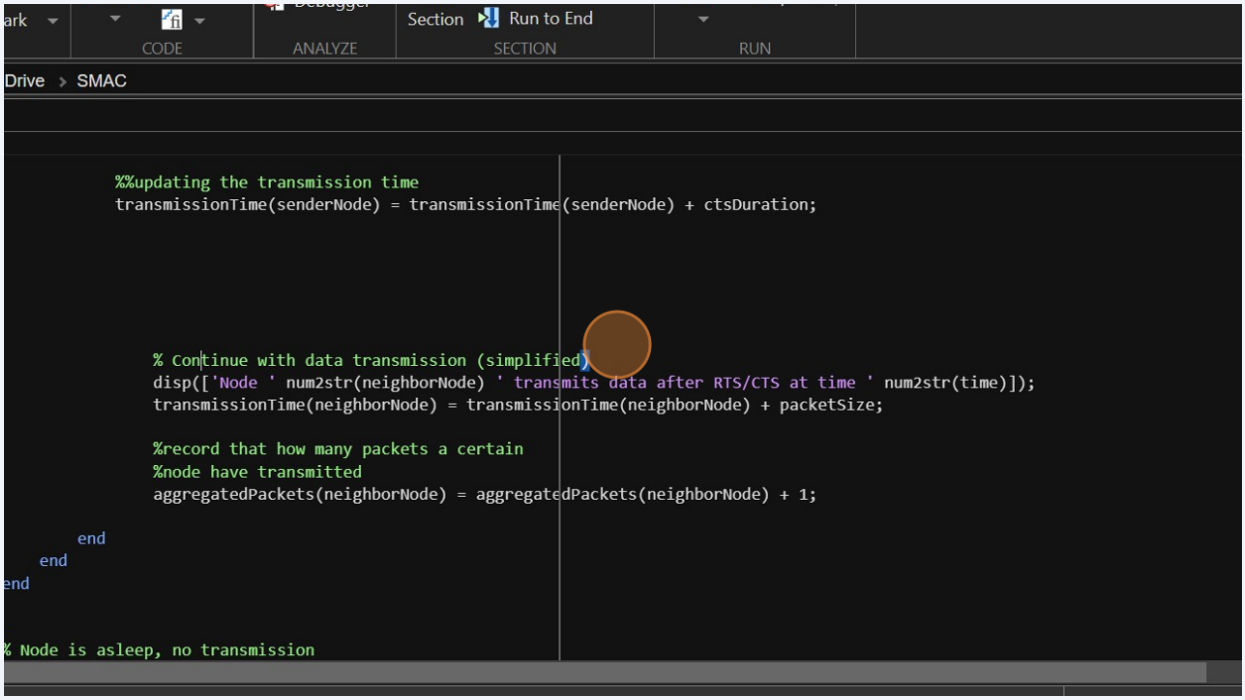
• Following this, data transmission is simulated, updating transmission times and recording the number of aggregated packets.



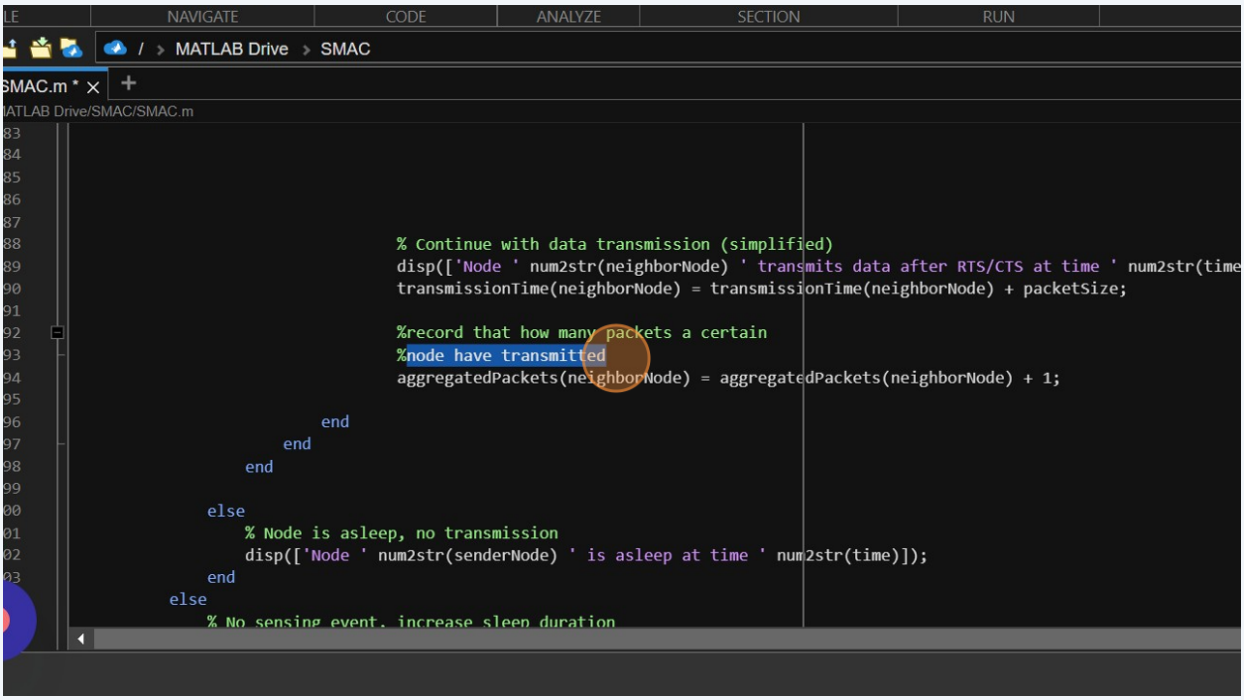
19 • **Sleep Duration Adjustment (**else**):**

• If no sensing event occurs, the sleep duration of the node is increased, and the sleep duration is reset based on the duty cycle after reaching the sleep period.

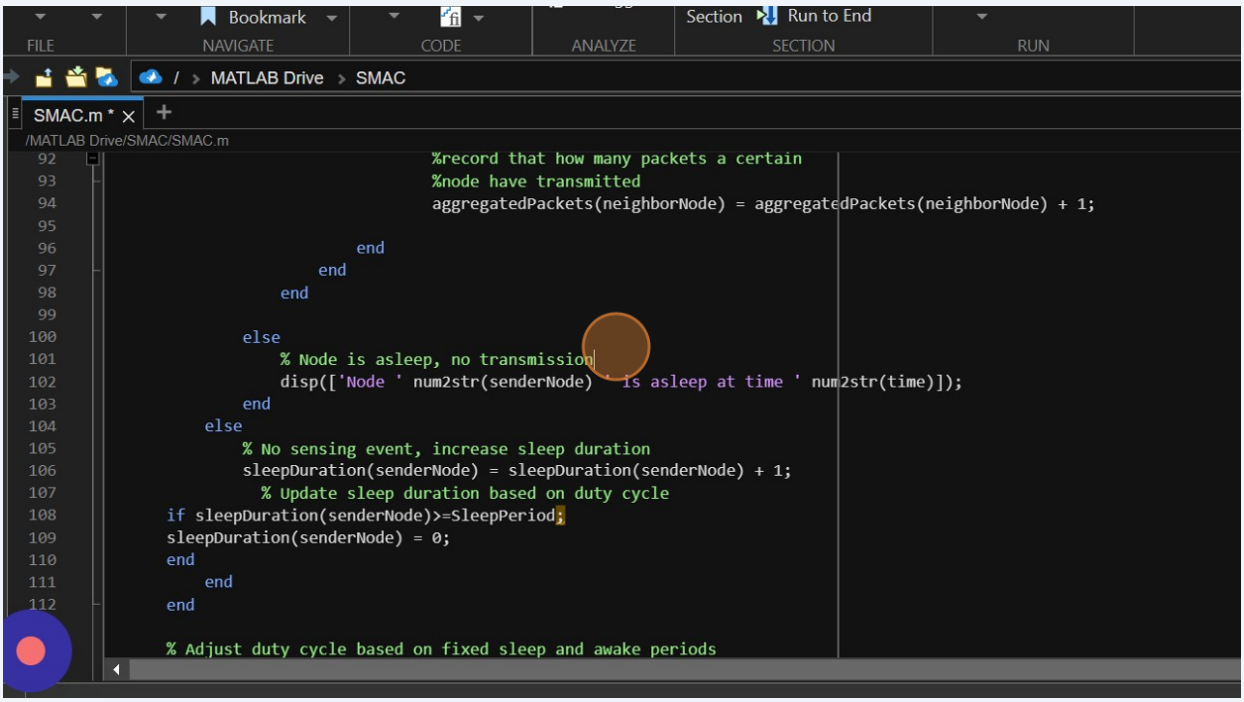
This loop captures the dynamics of the S-MAC protocol, including synchronization, RTS/CTS handshakes, and sleep-wake cycles, providing insights into the behavior of nodes in a wireless sensor network over time.



## 20 This part increments the count of aggregated packets for the neighbor node. It's a record-keeping mechanism to keep track of the number of packets transmitted by each node during the simulation.



**21** **If the node is determined to be asleep (sleep duration is not zero), the simulation prints a message indicating that the node is asleep at the current simulation time. In this case, no transmission occurs.**
**If no sensing event occurs, the sleep duration of the node is increased by 1 unit. Additionally, there is a check to determine if the sleep duration has exceeded the fixed sleep period (SleepPeriod). If it has, the sleep duration is reset to 0,**



22

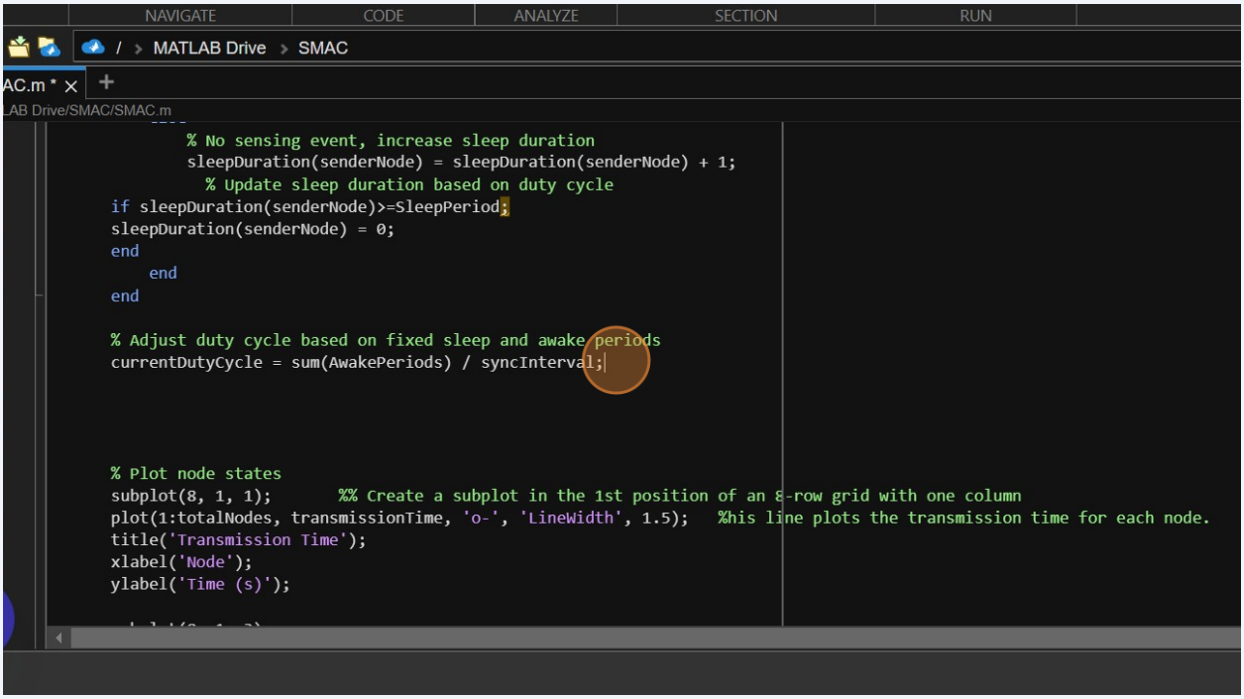
This section calculates the current duty cycle based on the fixed sleep and awake periods. Here's a breakdown:

• sum(AwakePeriods): This part sums up the awake periods across all nodes. AwakePeriods is calculated as the difference between the synchronization interval (syncInterval) and the fixed sleep duration (SleepPeriod). It represents the total duration during which nodes are awake within a synchronization interval.

• / syncInterval: The sum of awake periods is then divided by the synchronization interval to calculate the duty cycle. The duty cycle is defined as the fraction of time a node is active or awake compared to the total synchronization interval.

• currentDutyCycle: The result is assigned to the variable currentDutyCycle, representing the current duty cycle of the network.

This calculation provides an indication of how much time, on average, nodes are active compared to their total synchronization interval. It reflects the duty cycling behavior of the nodes in the network.



**23** This code segment is responsible for creating a subplot to visualize the transmission time for each node. Here's a breakdown:

• subplot(8, 1, 1);: This line creates a subplot in an 8-row grid with one column and places the subsequent plot in the first position. This means the subsequent plot will be positioned at the top of the grid.

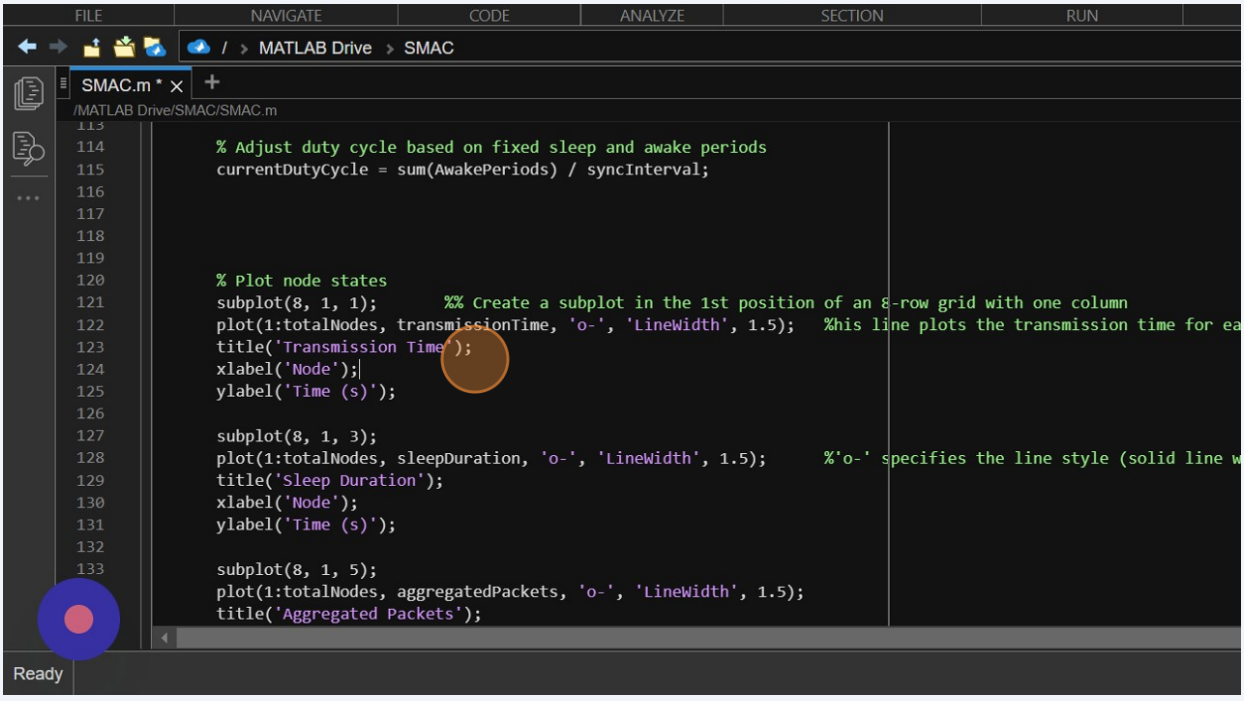
• plot(1:totalNodes, transmissionTime, 'o-', 'LineWidth', 1.5);: This line generates the actual plot. It plots the transmission time for each node, where the x-axis represents the node indices (1 to totalNodes), and the y-axis represents the transmission time. The data points are connected with lines ('o-' specifies the line style - solid line with circles at data points), and the line width is set to 1.5.

• title('Transmission Time');: Sets the title of the subplot to 'Transmission Time'.

• xlabel('Node');: Adds a label to the x-axis, indicating that it represents node indices.

• ylabel('Time (s)');: Adds a label to the y-axis, indicating that it represents time in seconds.

This subplot provides a visual representation of how the transmission time varies across different nodes in the network over the course of the simulation. The plot allows for an analysis of the temporal behavior of node transmissions.



**24** This code segment creates a subplot to visualize the sleep duration for each node. Here's an explanation:

• subplot(8, 1, 3);: This line creates a subplot in an 8-row grid with one column and places the subsequent plot in the third position. This means the subsequent plot will be positioned in the third row of the grid.

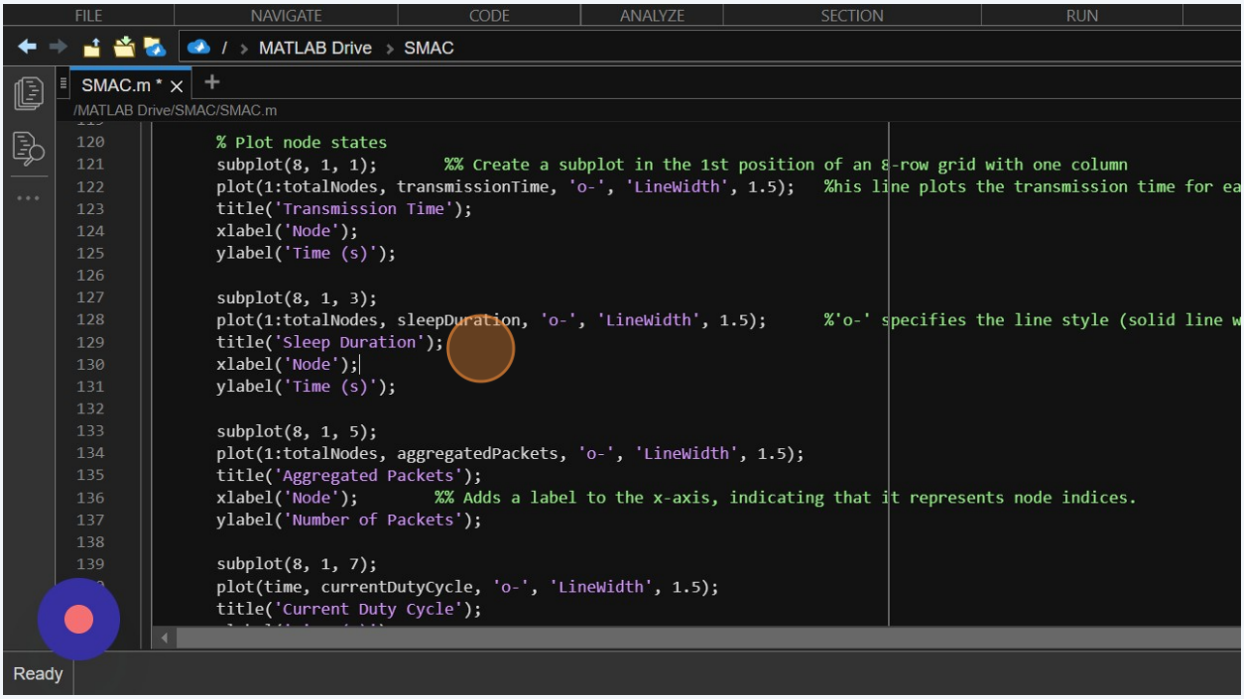
• plot(1:totalNodes, sleepDuration, 'o-', 'LineWidth', 1.5);: This line generates the actual plot. It plots the sleep duration for each node over time, where the x-axis represents the node indices (1 to totalNodes), and the y-axis represents the sleep duration. The data points are connected with lines ('o-' specifies the line style - solid line with circles at data points), and the line width is set to 1.5.

• title('Sleep Duration');: Sets the title of the subplot to 'Sleep Duration'.

• xlabel('Node');: Adds a label to the x-axis, indicating that it represents node indices.

• ylabel('Time (s)');: Adds a label to the y-axis, indicating that it represents time in seconds.

This subplot provides a visual representation of how the sleep duration varies across different nodes in the network. It allows for the analysis of the temporal behavior of sleep patterns for each node during the simulation.



**25** This code segment creates a subplot to visualize the aggregated number of packets for each node. Here's an explanation:

• subplot(8, 1, 5);: This line creates a subplot in an 8-row grid with one column and places the subsequent plot in the fifth position. This means the subsequent plot will be positioned in the fifth row of the grid.

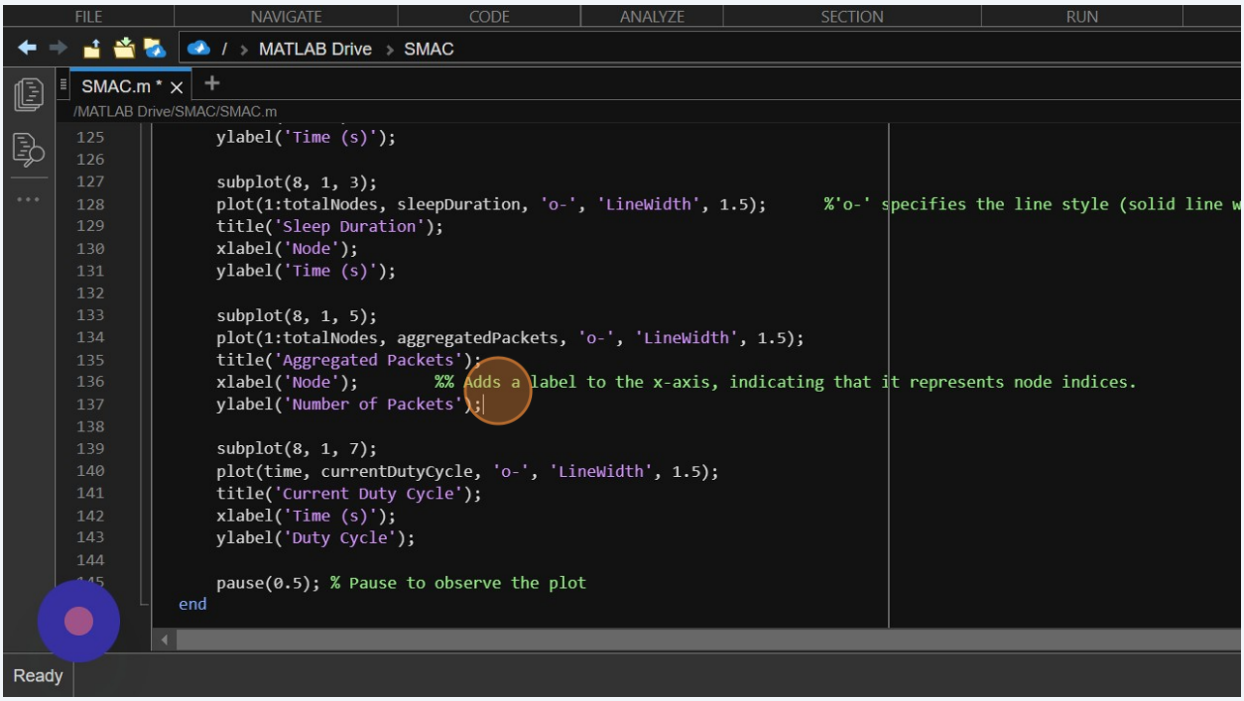
• plot(1:totalNodes, aggregatedPackets, 'o-', 'LineWidth', 1.5);: This line generates the actual plot. It plots the aggregated number of packets for each node, where the x-axis represents the node indices (1 to totalNodes), and the y-axis represents the aggregated number of packets. The data points are connected with lines ('o-' specifies the line style - solid line with circles at data points), and the line width is set to 1.5.

• title('Aggregated Packets');: Sets the title of the subplot to 'Aggregated Packets'.

• xlabel('Node');: Adds a label to the x-axis, indicating that it represents node indices.

• ylabel('Number of Packets');: Adds a label to the y-axis, indicating that it represents the number of aggregated packets.

This subplot provides a visual representation of how the number of aggregated packets varies across different nodes in the network over the course of the simulation. It allows for the analysis of the data aggregation behavior for each node.



**26** This code segment creates a subplot to visualize the current duty cycle of the network over time. Here's an explanation:

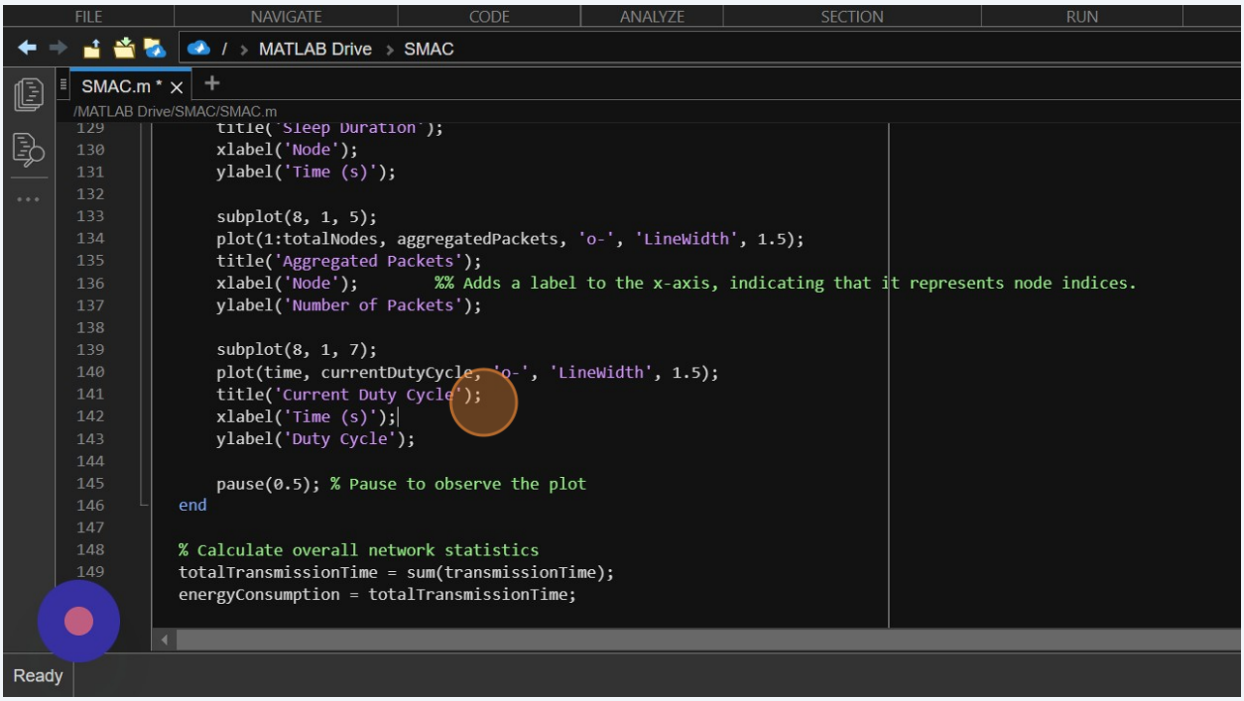
• subplot(8, 1, 7);: This line creates a subplot in an 8-row grid with one column and places the subsequent plot in the seventh position. This means the subsequent plot will be positioned in the seventh row of the grid.

• plot(time, currentDutyCycle, 'o-', 'LineWidth', 1.5);: This line generates the actual plot. It plots the duty cycle of the network over time, where the x-axis represents time (time), and the y-axis represents the duty cycle (currentDutyCycle). The data points are connected with lines ('o-' specifies the line style - solid line with circles at data points), and the line width is set to 1.5.

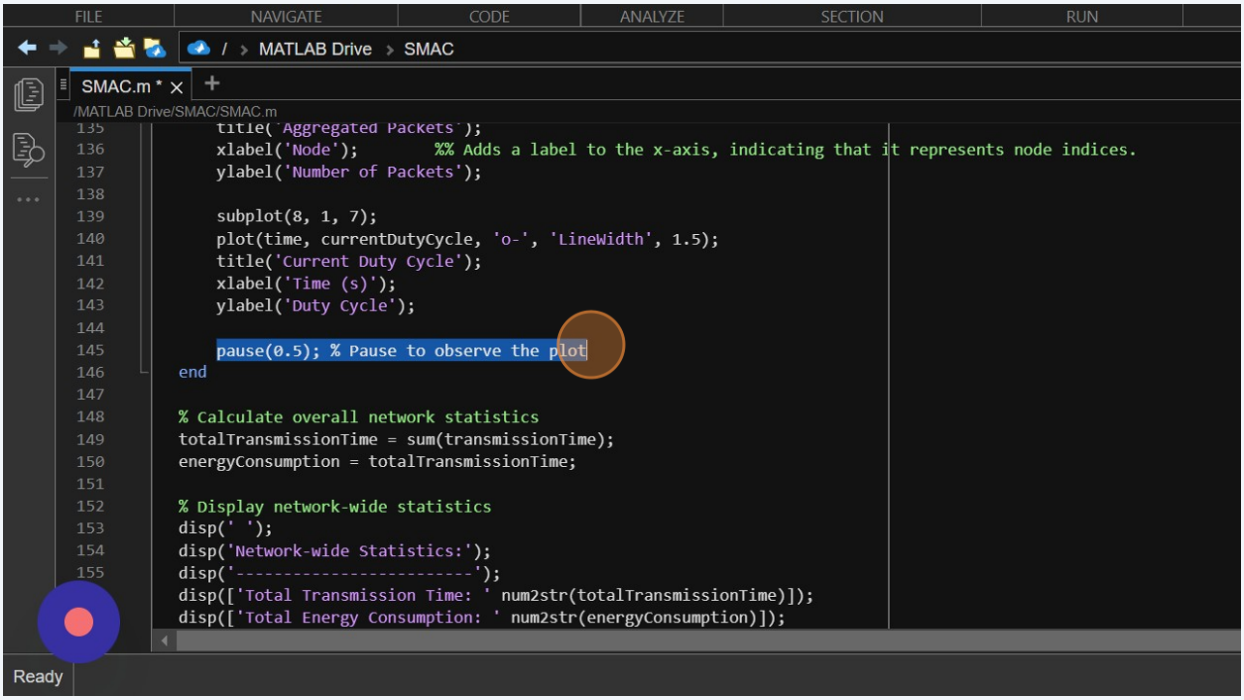
• title('Current Duty Cycle');: Sets the title of the subplot to 'Current Duty Cycle'.

• xlabel('Time (s)');: Adds a label to the x-axis, indicating that it represents time in seconds.

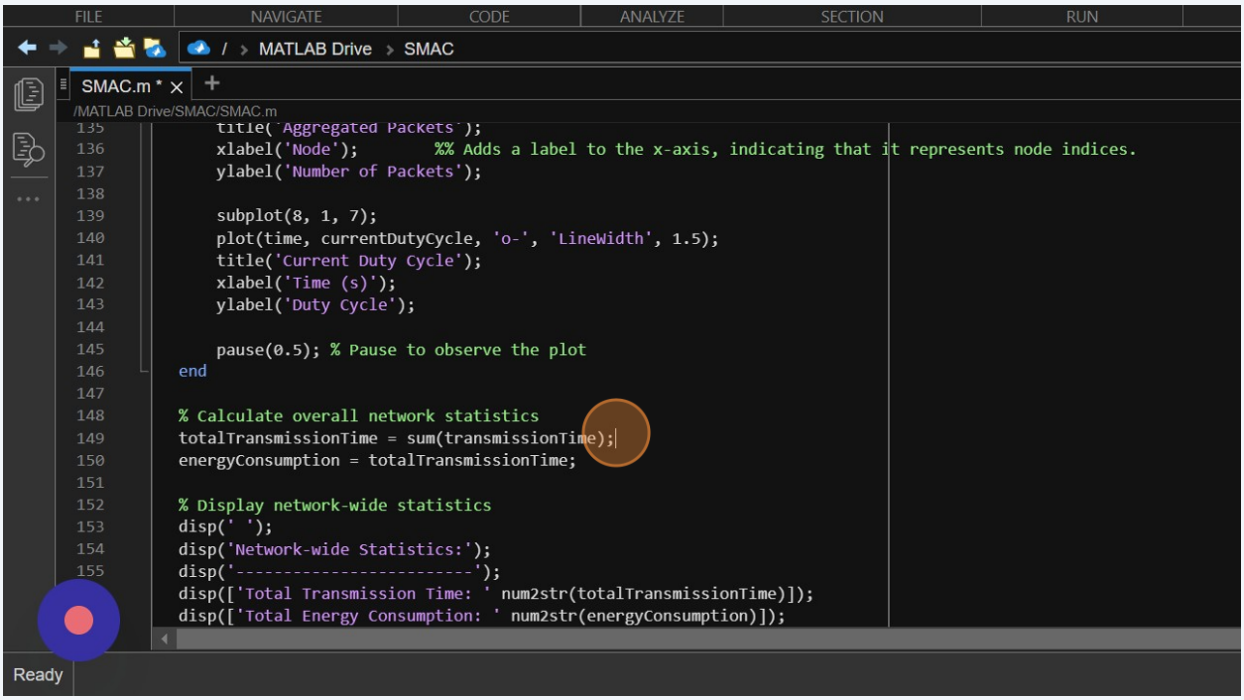
• ylabel('Duty Cycle');: Adds a label to the y-axis, indicating that it represents the duty cycle.



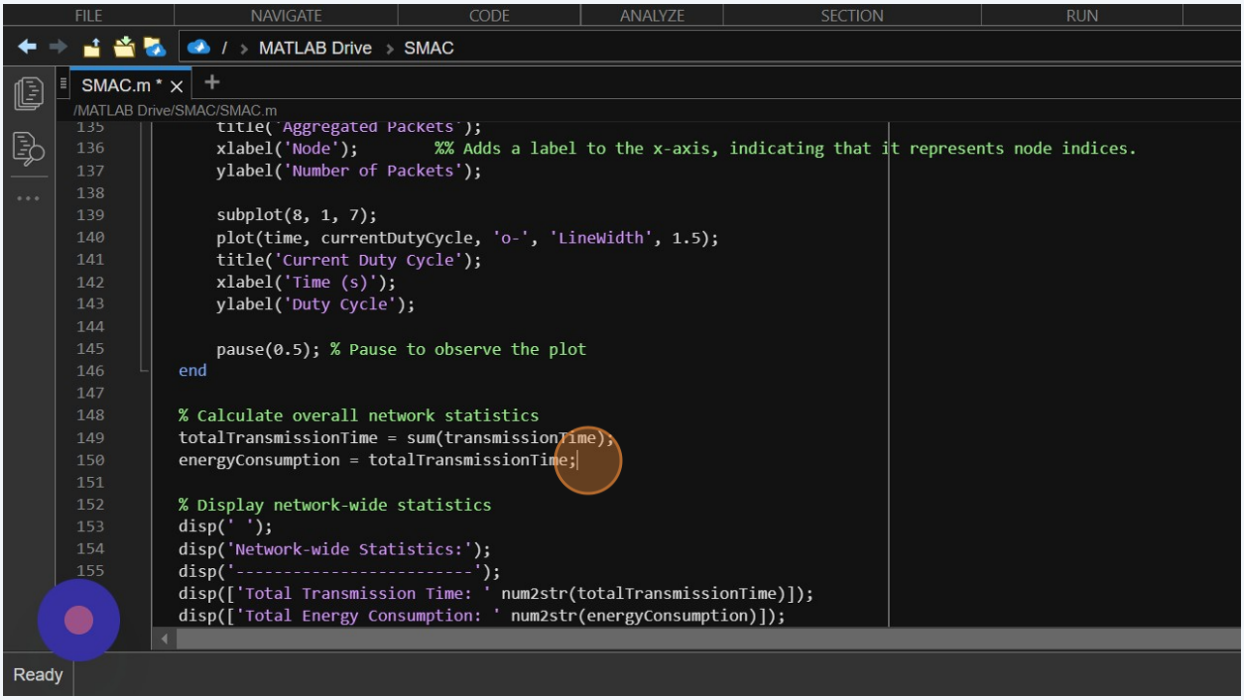
## 27 Pauses the execution for 0.5 seconds, allowing time to observe the plot.



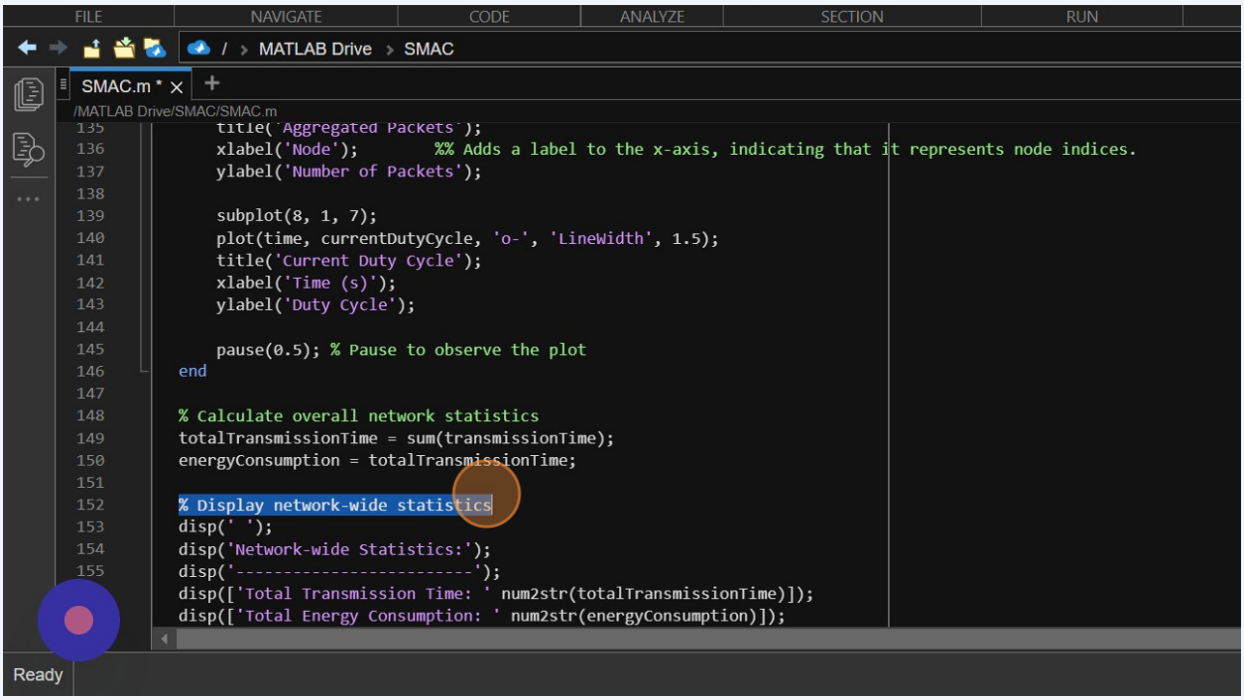

## 28 The total transmission time here is calculated by the sum of all the transmission Time involved in the simulation



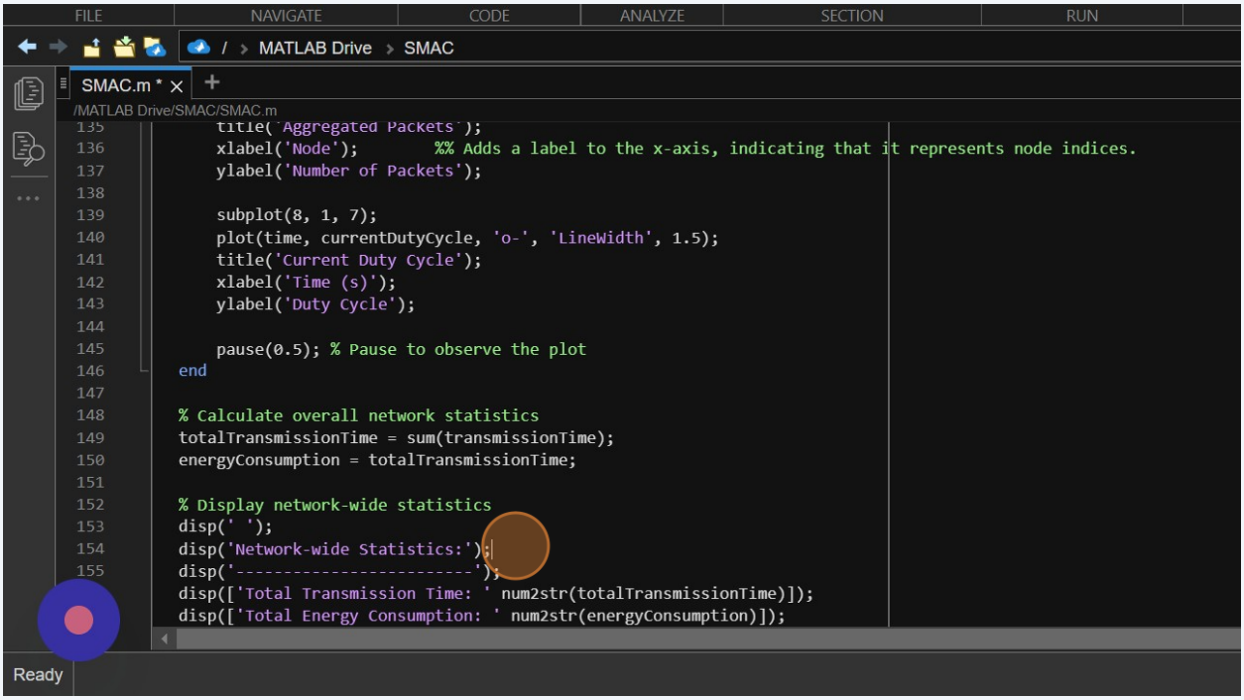
**29** Then energy consumption will be obtained by the totalTransmissionTime



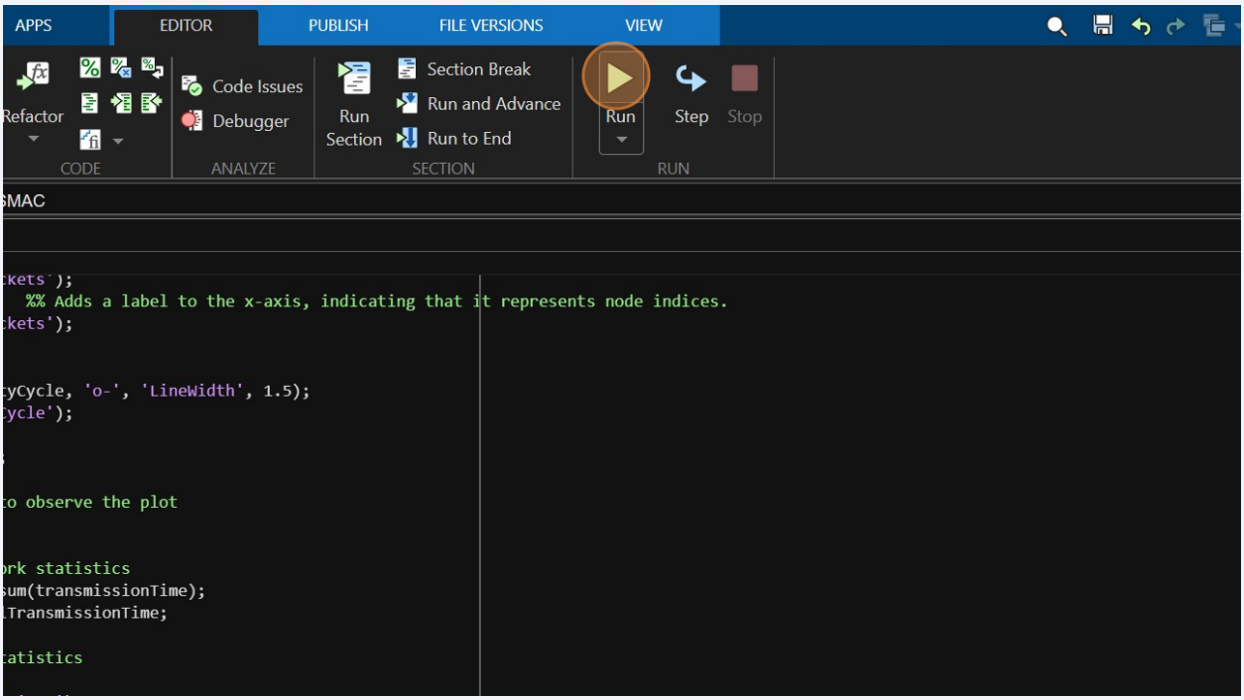

**30** Now here is the code to display the overall statistics at the end of simulation



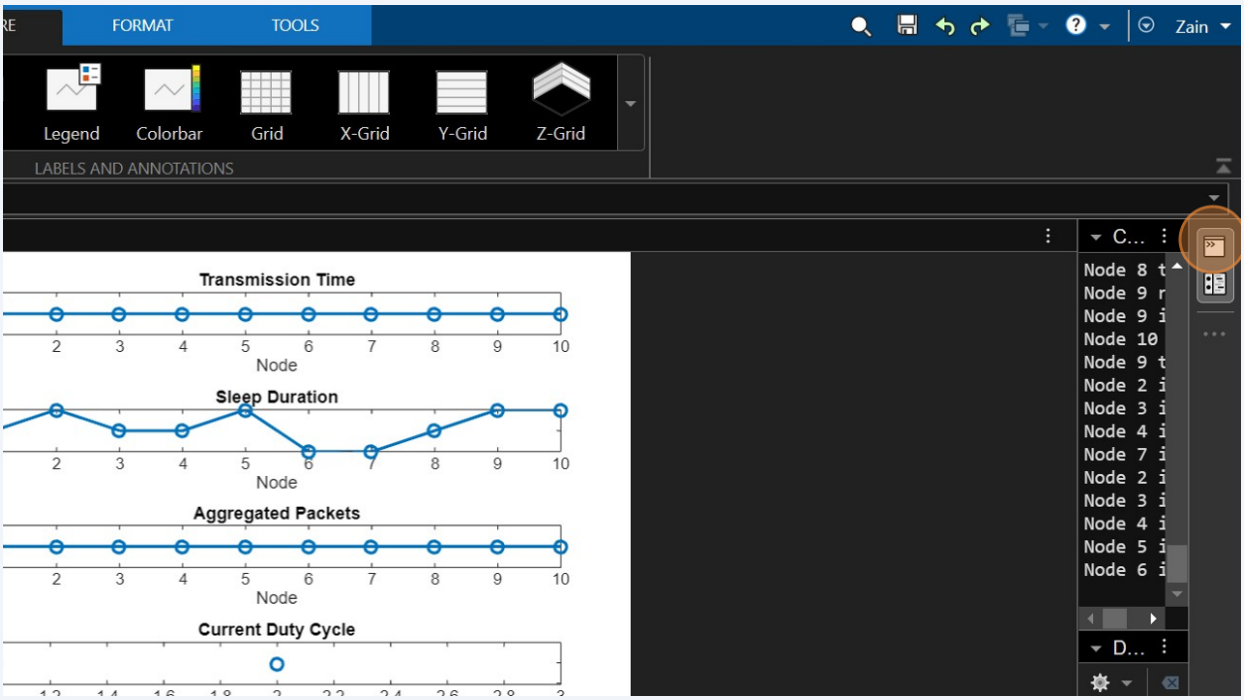
**31** This is the header before printing the stats after that the two values previously calculated in 149 and 150 number line is displayed



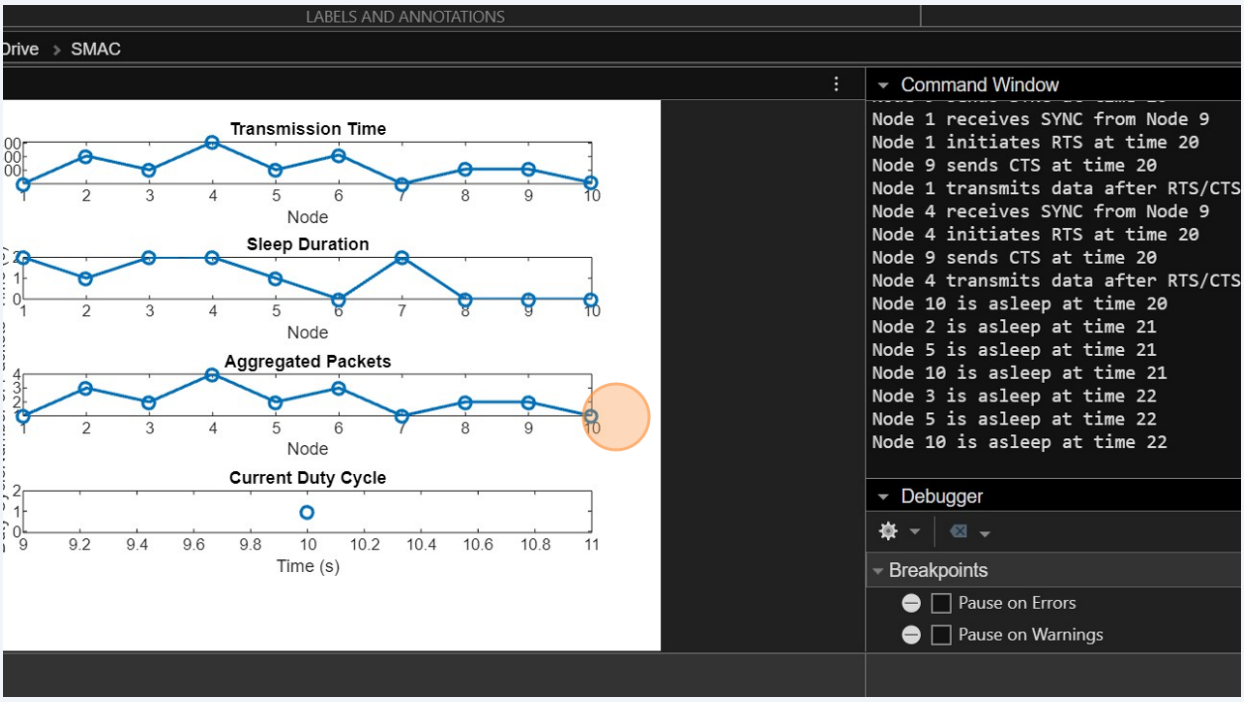

**32** Now lets run it to see the simulation and output



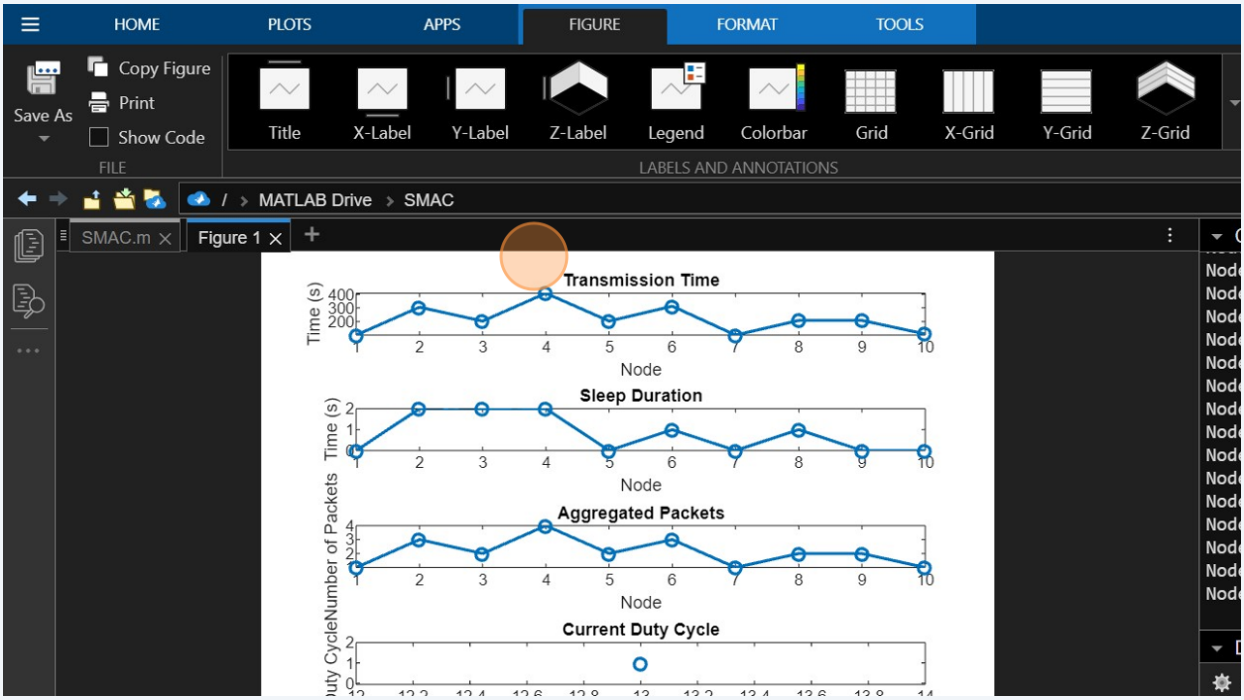
**33** A figure will be shown after running the program
Lets see the commands displayed in the terminal that was previously discussed in the code by using disp commands



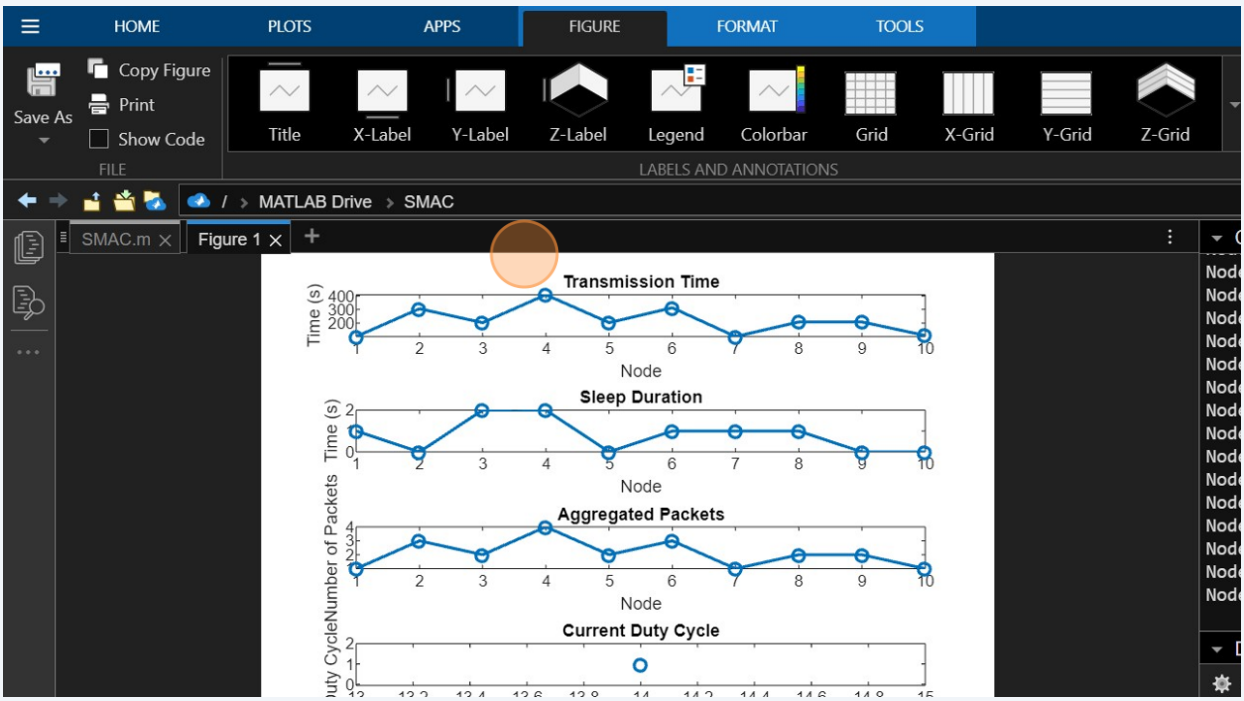

**34** Here figure and command window both are shown



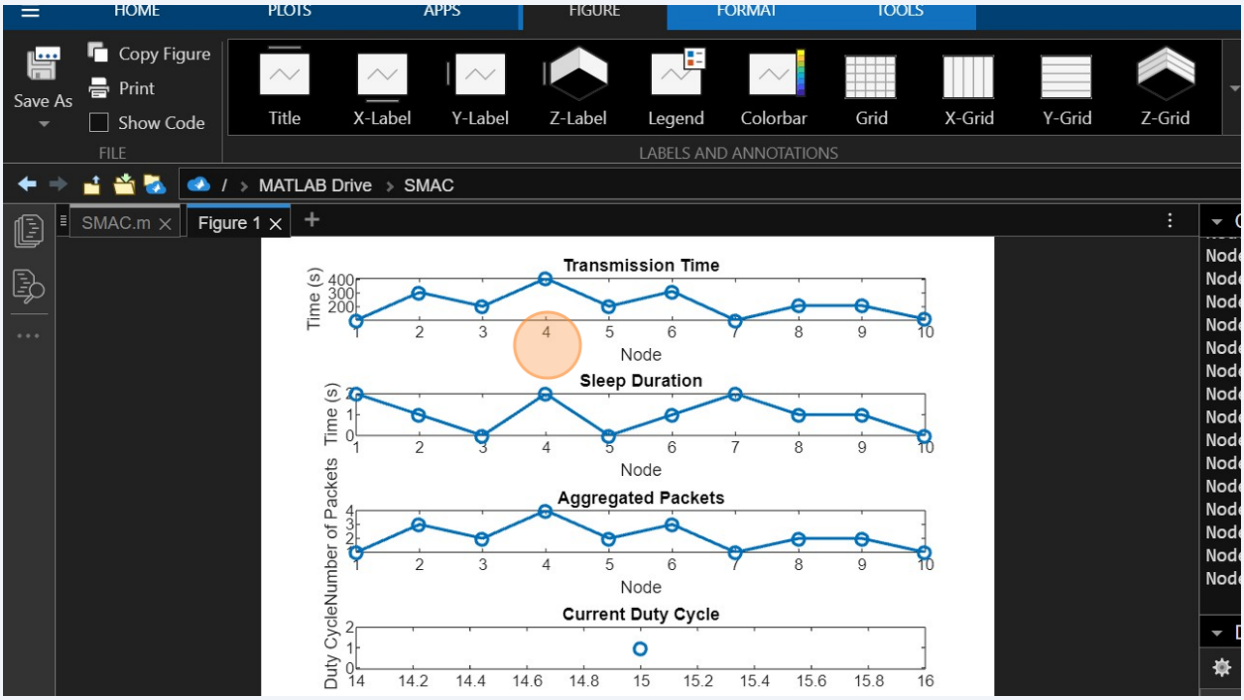
**35** On the top of figure we plotted the first subplot as Transmission time and title was set to "Transmission Time"



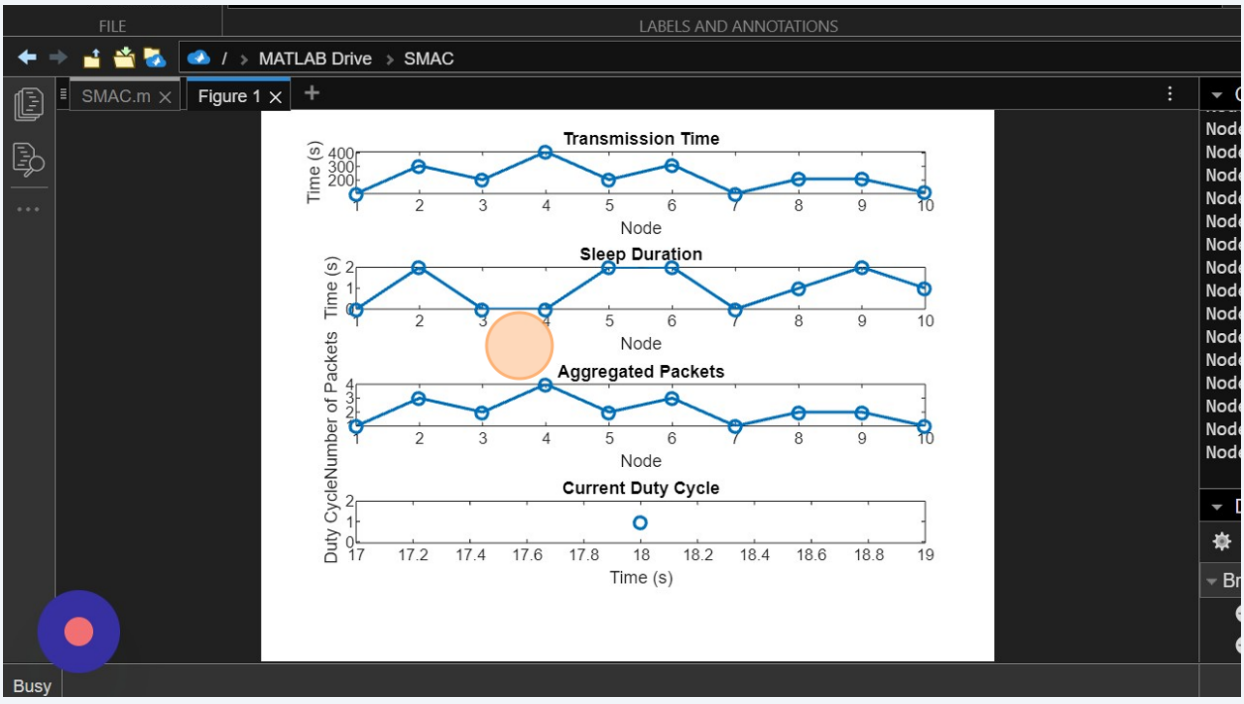

**36** In the trasmission time graph (subplot) on y axis there is time displayed in seconds and in this figure circles are showing the nodes In this simulation there are 10 nodes
We can take it on random positions as well



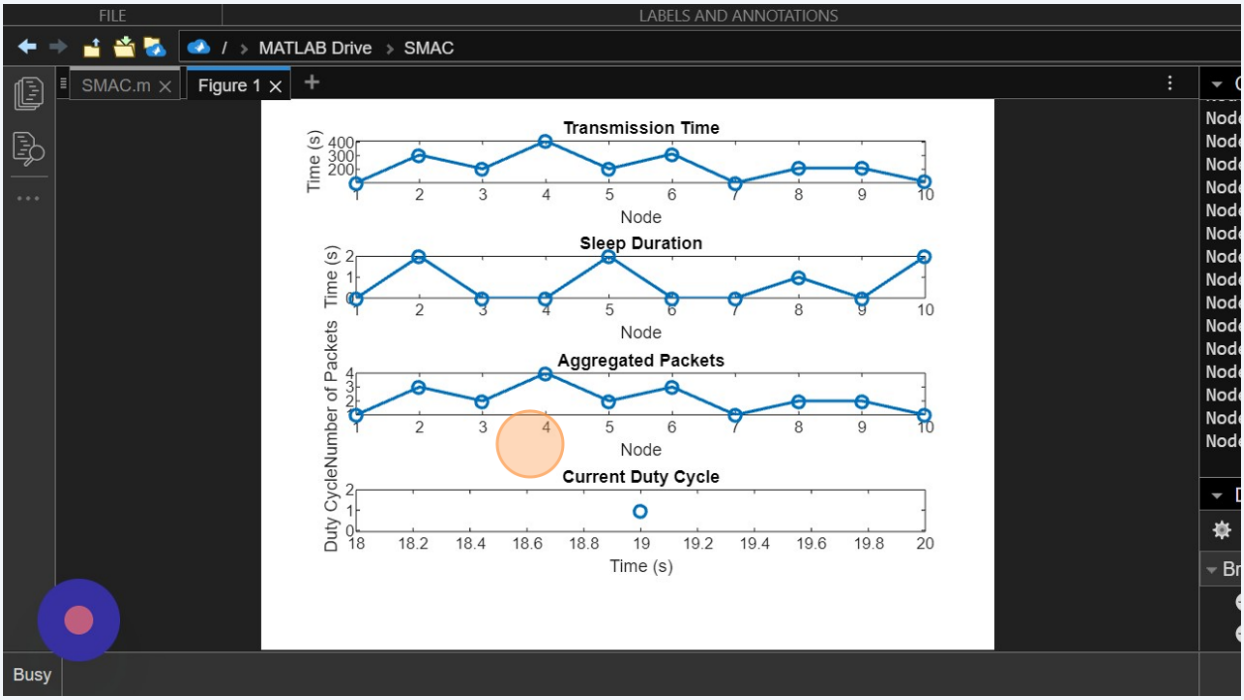
**37** Next thing is sleep duration that is displayed here in the Subplot titled Sleep Duration and showing the sleep duration of each node in seconds that how much time the node is sleeping



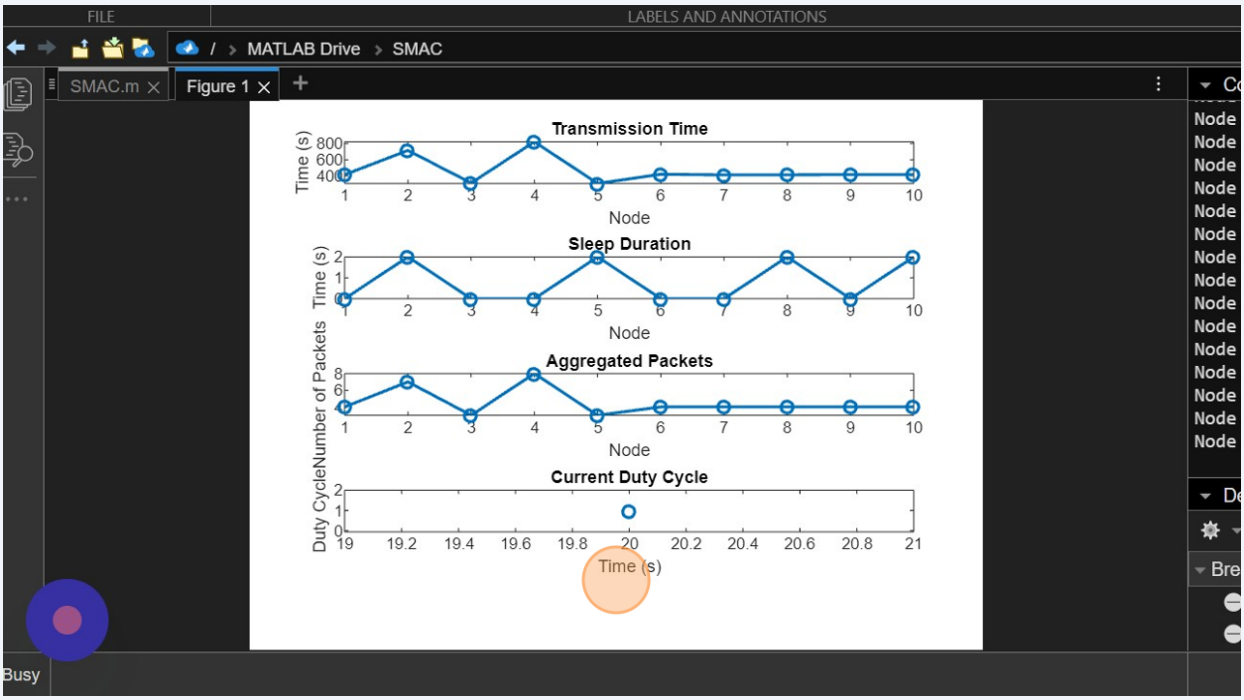

**38** Then here comes the aggregated packets and in y axis here the number of packets are shown that each of 10 nodes is having how many aggregated packets



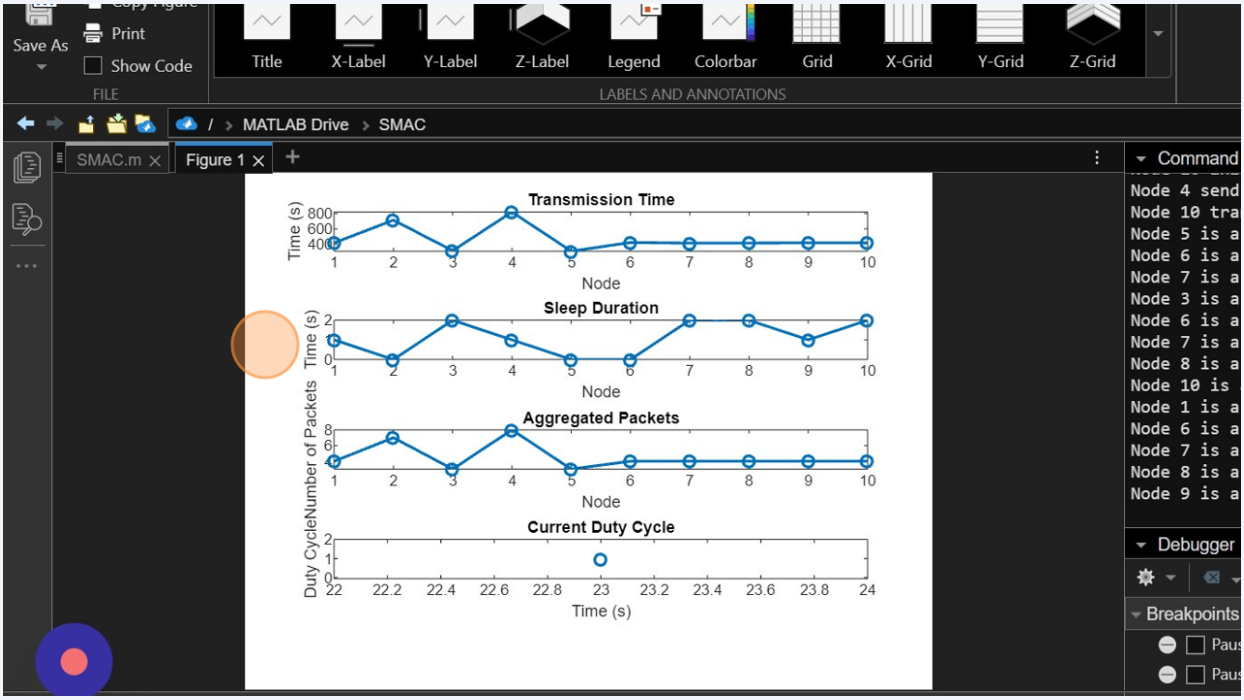
**39** Then comes the Current Duty Cycle that is what we have set in the perimeters and initial was 0.5 and total dutycycles are 100




**40** On the x axis of duty cycles time is shown bcz with time transmission occurs and duty cycles are changed by adding new to previous one

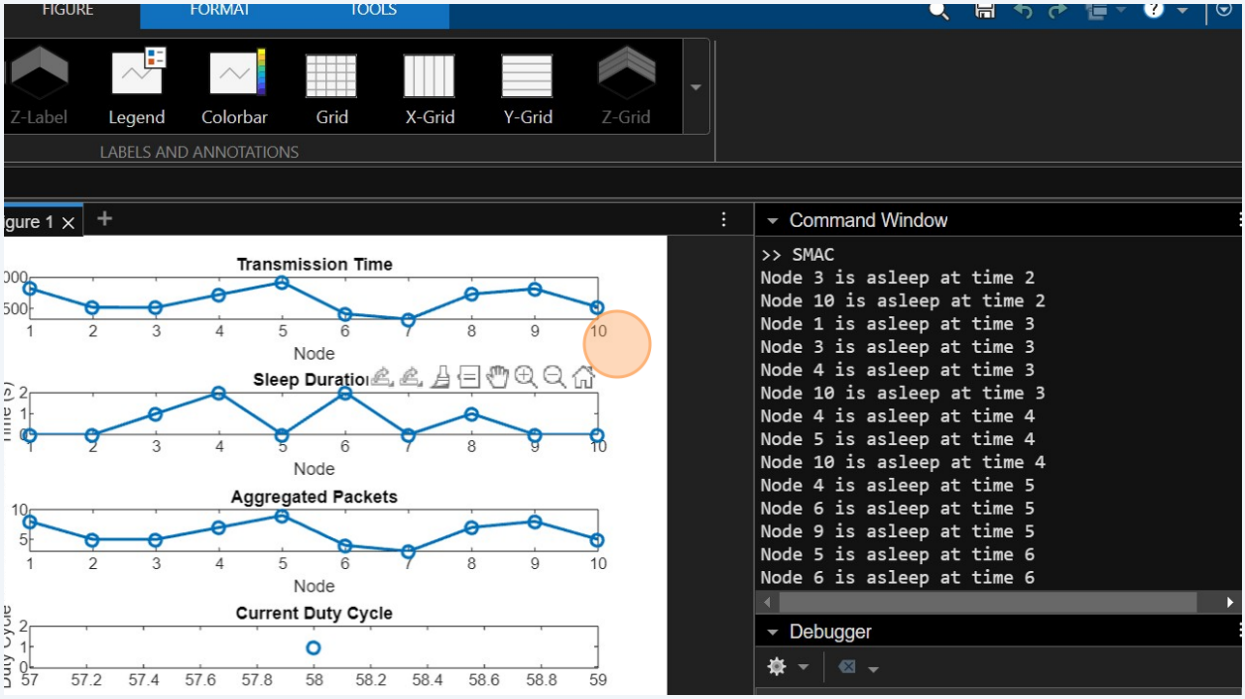


## 41 Sample figure at 23 Duty Cycles

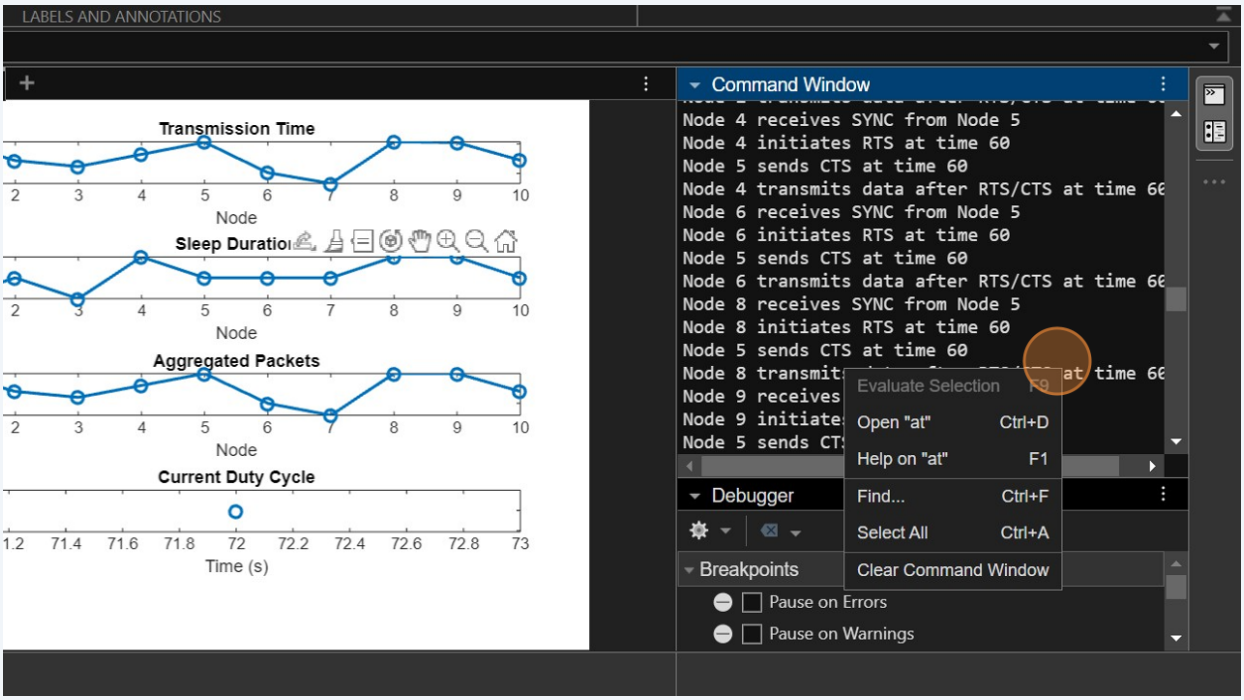


# Step-by-Step Guide to Running and Analyzing SMAC Implementation in MATLAB

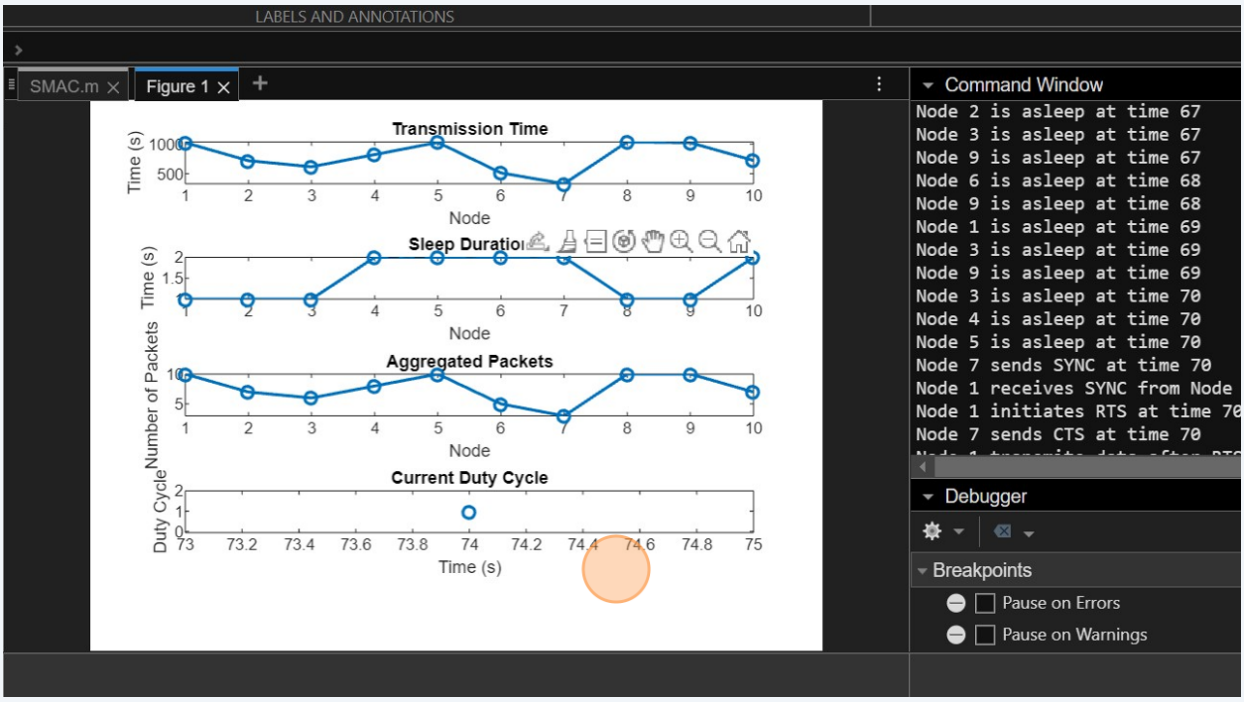
1 Again lets observe it at 58 duty cycles and in the command window from start we can see that the nodes at sleep mode are displayed and their time to sleep is also shown



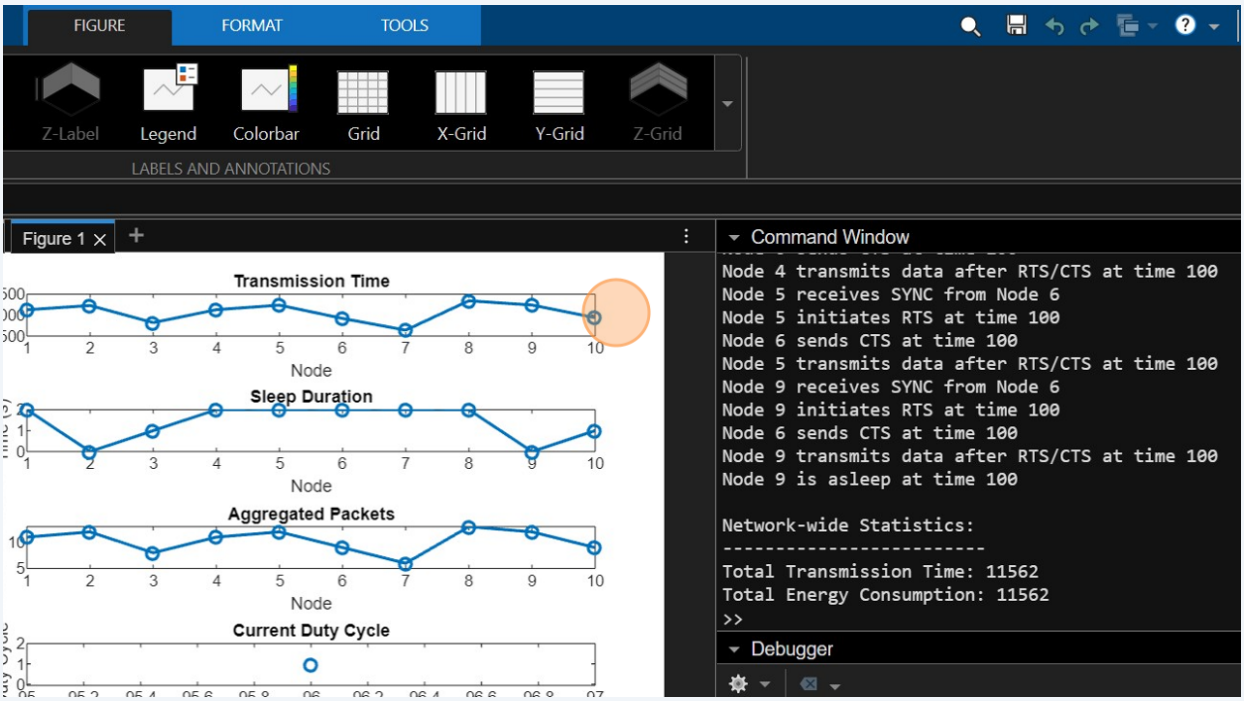
2 Now observing some other commands that were previously discussed in code and theory that node transmitting data is also displayed after the RTS CTS handshake and also the Sync packet recieved by nodes numbers




3 Similarly a node initiates RTS at some time and sends CTS at some time and display the msg



**4** And after 100 duty cycles the total Network Wide Statistics are displayed as we have seen at the end of our code
Both Total Transmission time as well as total energy consumption is calculated and displayed



**5**

So this was all about simulation of SMAC and output
The code is Given Below:

```
% Comprehensive S-MAC Simulation with Visualization in MATLAB

% Parameters
totalNodes = 10; % Total number of nodes in the network
simulationTime = 100; % Simulation time in seconds
packetSize = 100; % Size of each data packet in bytes
trafficLoad = 0.6; % Probability of a sensing event occurring

% Initialize node states
sleepDuration = zeros(1, totalNodes); % Array to store sleep duration for each node%%array is intialize with zero
transmissionTime = zeros(1, totalNodes); % Array to store transmission time for each node

% Packet aggregation parameters

aggregatedPackets = zeros(1, totalNodes); % Initialize an array to store aggregated packets for each node

% Dynamic sleep adjustment parameters
SleepPeriod = 3; % fixed sleep duration of node
syncInterval = 50; % Synchronization interval
AwakePeriods = syncInterval - SleepPeriod;

% Duty cycle parameters%%%The duty cycle is the fraction of time that a node is active or awake compared to the total time

currentDutyCycle = 0.1; % Initial duty cycle

% Synchronization parameters
syncInterval = 50; % Synchronization interval
syncDuration = 1; % Duration for which SYNC is transmitted
rtsThreshold = 0.5; % RTS threshold for initiating RTS/CTS
rtsDuration = 2; % Duration for which RTS is transmitted
ctsDuration = 1; % Duration for which CTS is transmitted
syncPeriod = 10; % Synchronization period for sending SYNC packets periodically

% Simulation loop
for time = 1:simulationTime % Loop over simulation time
for senderNode = 1:totalNodes % Loop over each node as a potential sender
% Check if a sensing event occurs
if rand() < trafficLoad %rand function genrate random value in matlab

% Check if the node is awake before transmission if sleepDuration(senderNode) == 0 % SYNC phase: Send SYNC packets periodically if mod(time, syncPeriod) == 0 %used to check whether the current simulation time is a multiple of the synchronization period disp([&#x27;Node &#x27; num2str(senderNode) &#x27; sends SYNC at time &#x27; num2str(time)]); %display message
transmissionTime(senderNode) = transmissionTime(senderNode) + syncDuration; %updating transmission time
```

```
%asuming that all the nodes are neighbour nodes%%%%%otherwise we can
%calculate through distance and using cluster concept
% Broadcast SYNC packet to neighbors
for neighborNode = 1:totalNodes
% In a real scenario, you would need to handle collisions
% Here, we simplify by assuming no collisions in the SYNC phase
if neighborNode ~= senderNode && rand() < rtsThreshold

` %%node receives SYNC disp(['Node ' num2str(neighborNode) ' receives SYNC
from Node ' num2str(senderNode)]); % RTS phase: Neighbor initiates RTS
disp(['Node ' num2str(neighborNode) ' initiates RTS at time ' num2str(time)]); %
Transmit RTS signal transmissionTime(neighborNode) =
transmissionTime(neighborNode) + rtsDuration; % Neighbor responds with CTS
disp(['Node ' num2str(senderNode) ' sends CTS at time ' num2str(time)]);
%%updating the transmission time transmissionTime(senderNode) =
transmissionTime(senderNode) + ctsDuration; % Continue with data transmission
(simplified) disp(['Node ' num2str(neighborNode) ' transmits data after RTS/CTS at
time ' num2str(time)]); transmissionTime(neighborNode) =
transmissionTime(neighborNode) + packetSize; %record
```