Wireless Sensor Networks Cluster Head Selection using Fuzzy Logic

Novelty of the Work: Fuzzy Logic-Based Cluster Head Selection in WSNs

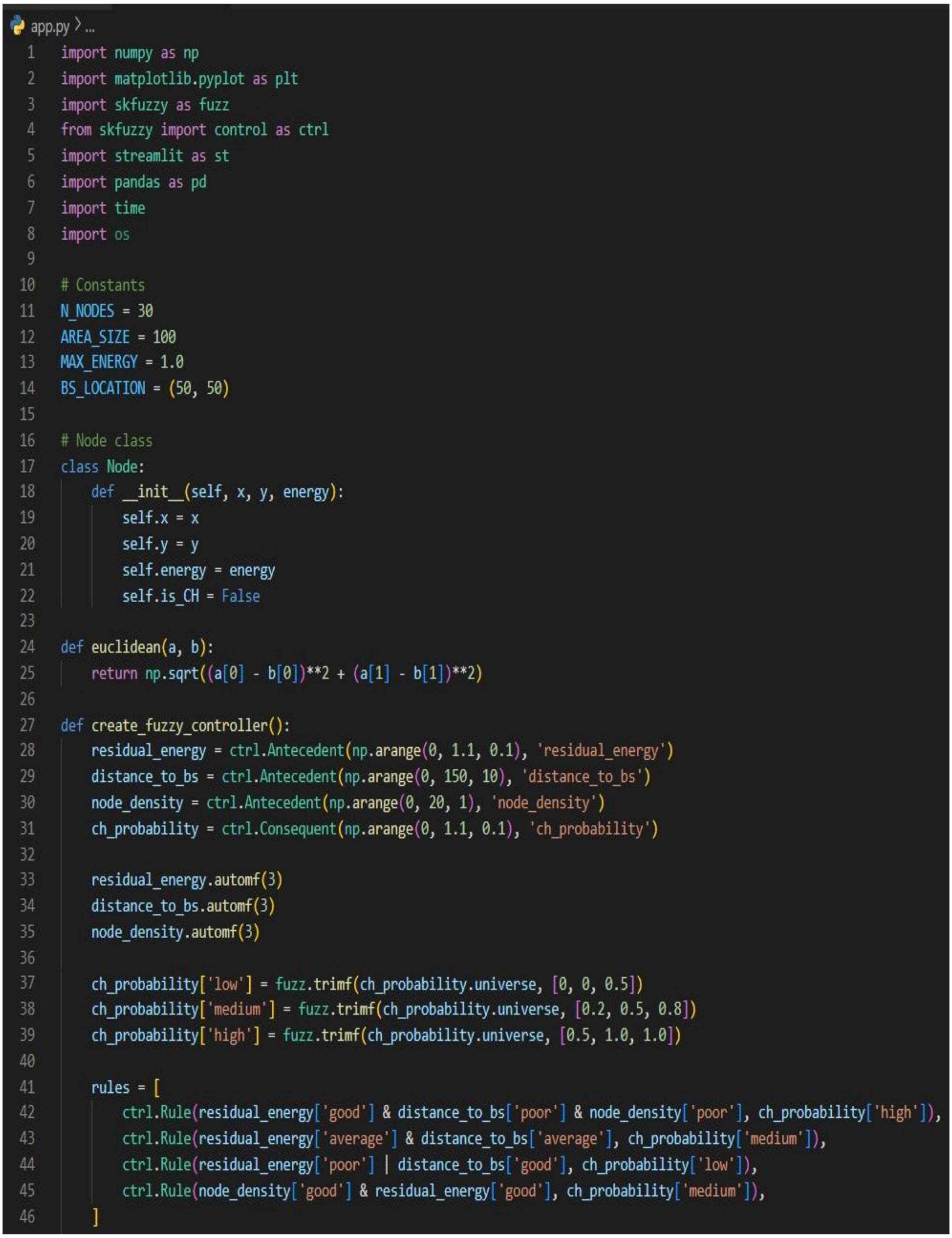
Tushar Gupta [12312021], Nitish Kumar Choubey [12312031], Harshit Singhal [12312016], Nikhil Nagar [12312007], Devansh Bansal [12312006]

1. Algorithm

Algorithm 1 Cluster Head Selection using Fuzzy Logic

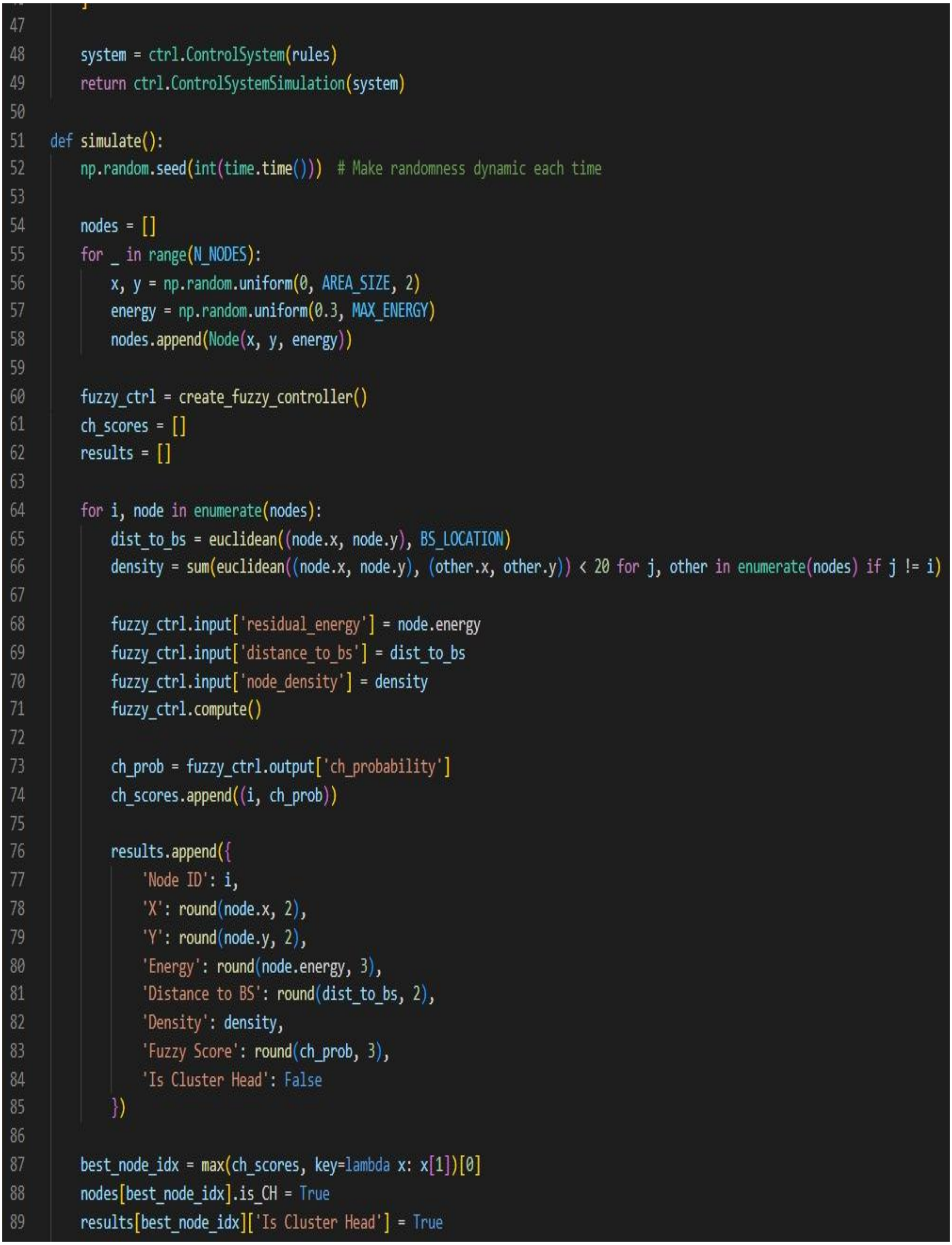
```
1: Input: Number of nodes $N = 30$, Area size, Max Energy = 1.0, Base station location $(x_{bs}, y_{bs})$
2: Output: Cluster Head node ID
3: Initialize list of nodes with random $(x, y)$ positions and energy $\in [0.3, 1.0]$
4: Define fuzzy variables:
     • Inputs: residual_energy, distance_to_bs, node_density (each with 3 fuzzy sets: low, medium, high)
     • Output: ch_probability with sets: low, medium, high
5: Create fuzzy rules such as:
     • IF residual_energy is good AND distance_to_bs is poor AND node_density is poor THEN ch_probability is high
     • IF residual_energy is average AND distance_to_bs is average THEN ch_probability is medium
     • IF residual_energy is poor OR distance_to_bs is good THEN ch_probability is low
6: Initialize fuzzy control system with defined rules
7: for each node $i = 1$ to $N$ do
8:     Compute $d_i \leftarrow \sqrt{(x_i - x_{bs})^2 + (y_i - y_{bs})^2}$
9:     Compute $\delta_i \leftarrow \sum_{j \neq i} \mathbb{I}(d_{ij} < 20)$
10:    Provide inputs to fuzzy system:
     • residual_energy $\leftarrow E_i$
     • distance_to_bs $\leftarrow d_i$
     • node_density $\leftarrow \delta_i$
11:    Run fuzzy inference and obtain output: ch_probability
12:    Store $(i, ch\_probability)$ in score list
13: end for
14: Select node with maximum ch_probability as Cluster Head
15: Mark this node as CH and update results table
16: Save results to results.txt
17: Display node table and plot in Streamlit UI
```

2. Implementation and working

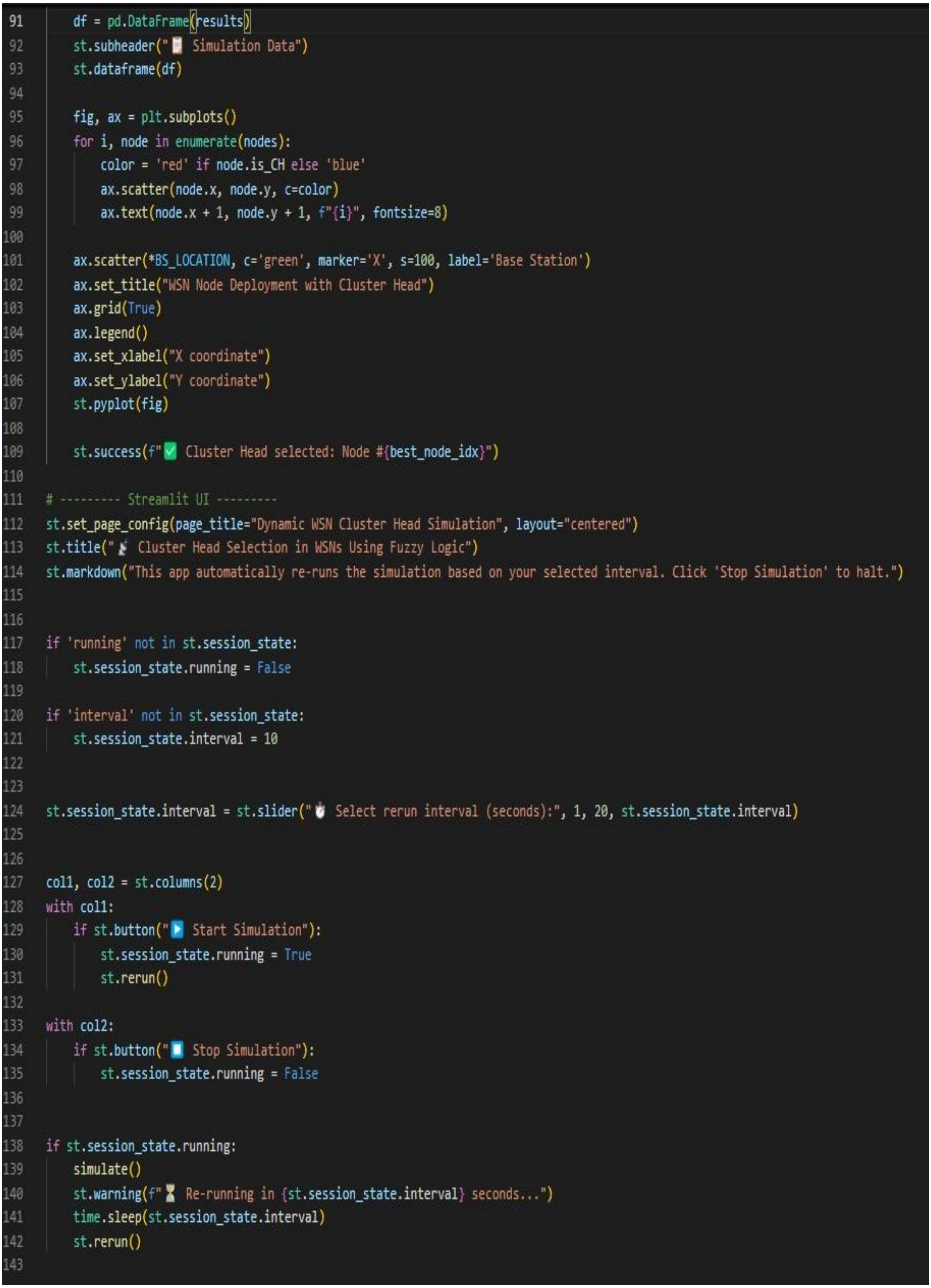


app.py > ...

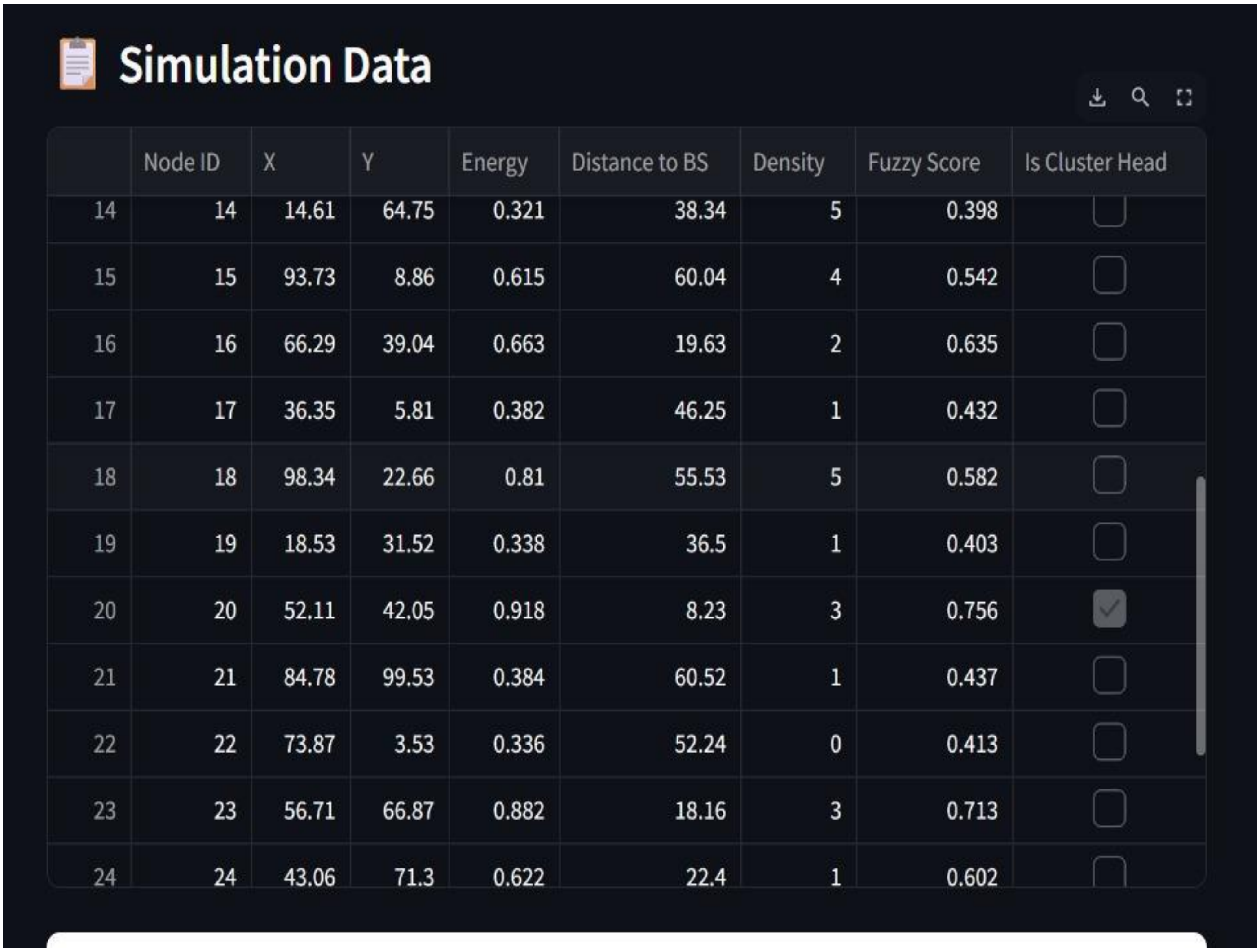
```python
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import streamlit as st
import pandas as pd
import time
import os

# Constants
N_NODES = 30
AREA_SIZE = 100
MAX_ENERGY = 1.0
BS_LOCATION = (50, 50)

# Node class
class Node:
    def __init__(self, x, y, energy):
        self.x = x
        self.y = y
        self.energy = energy
        self.is_CH = False

def euclidean(a, b):
    return np.sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2)

def create_fuzzy_controller():
    residual_energy = ctrl.Antecedent(np.arange(0, 1.1, 0.1), 'residual_energy')
    distance_to_bs = ctrl.Antecedent(np.arange(0, 150, 10), 'distance_to_bs')
    node_density = ctrl.Antecedent(np.arange(0, 20, 1), 'node_density')
    ch_probability = ctrl.Consequent(np.arange(0, 1.1, 0.1), 'ch_probability')

    residual_energy.automf(3)
    distance_to_bs.automf(3)
    node_density.automf(3)

    ch_probability['low'] = fuzz.trimf(ch_probability.universe, [0, 0, 0.5])
    ch_probability['medium'] = fuzz.trimf(ch_probability.universe, [0.2, 0.5, 0.8])
    ch_probability['high'] = fuzz.trimf(ch_probability.universe, [0.5, 1.0, 1.0])

    rules = [
        ctrl.Rule(residual_energy['good'] & distance_to_bs['poor'] & node_density['poor'], ch_probability['high']),
        ctrl.Rule(residual_energy['average'] & distance_to_bs['average'], ch_probability['medium']),
        ctrl.Rule(residual_energy['poor'] | distance_to_bs['good'], ch_probability['low']),
        ctrl.Rule(node_density['good'] & residual_energy['good'], ch_probability['medium']),
    ]
```



```python
    ]

    system = ctrl.ControlSystem(rules)
    return ctrl.ControlSystemSimulation(system)

def simulate():
    np.random.seed(int(time.time()))  # Make randomness dynamic each time

    nodes = []
    for _ in range(N_NODES):
        x, y = np.random.uniform(0, AREA_SIZE, 2)
        energy = np.random.uniform(0.3, MAX_ENERGY)
        nodes.append(Node(x, y, energy))

    fuzzy_ctrl = create_fuzzy_controller()
    ch_scores = []
    results = []

    for i, node in enumerate(nodes):
        dist_to_bs = euclidean((node.x, node.y), BS_LOCATION)
        density = sum(euclidean((node.x, node.y), (other.x, other.y)) < 20 for j, other in enumerate(nodes) if j != i)

        fuzzy_ctrl.input['residual_energy'] = node.energy
        fuzzy_ctrl.input['distance_to_bs'] = dist_to_bs
        fuzzy_ctrl.input['node_density'] = density
        fuzzy_ctrl.compute()

        ch_prob = fuzzy_ctrl.output['ch_probability']
        ch_scores.append((i, ch_prob))

        results.append({
            'Node ID': i,
            'X': round(node.x, 2),
            'Y': round(node.y, 2),
            'Energy': round(node.energy, 3),
            'Distance to BS': round(dist_to_bs, 2),
            'Density': density,
            'Fuzzy Score': round(ch_prob, 3),
            'Is Cluster Head': False
        })

    best_node_idx = max(ch_scores, key=lambda x: x[1])[0]
    nodes[best_node_idx].is_CH = True
    results[best_node_idx]['Is Cluster Head'] = True
```

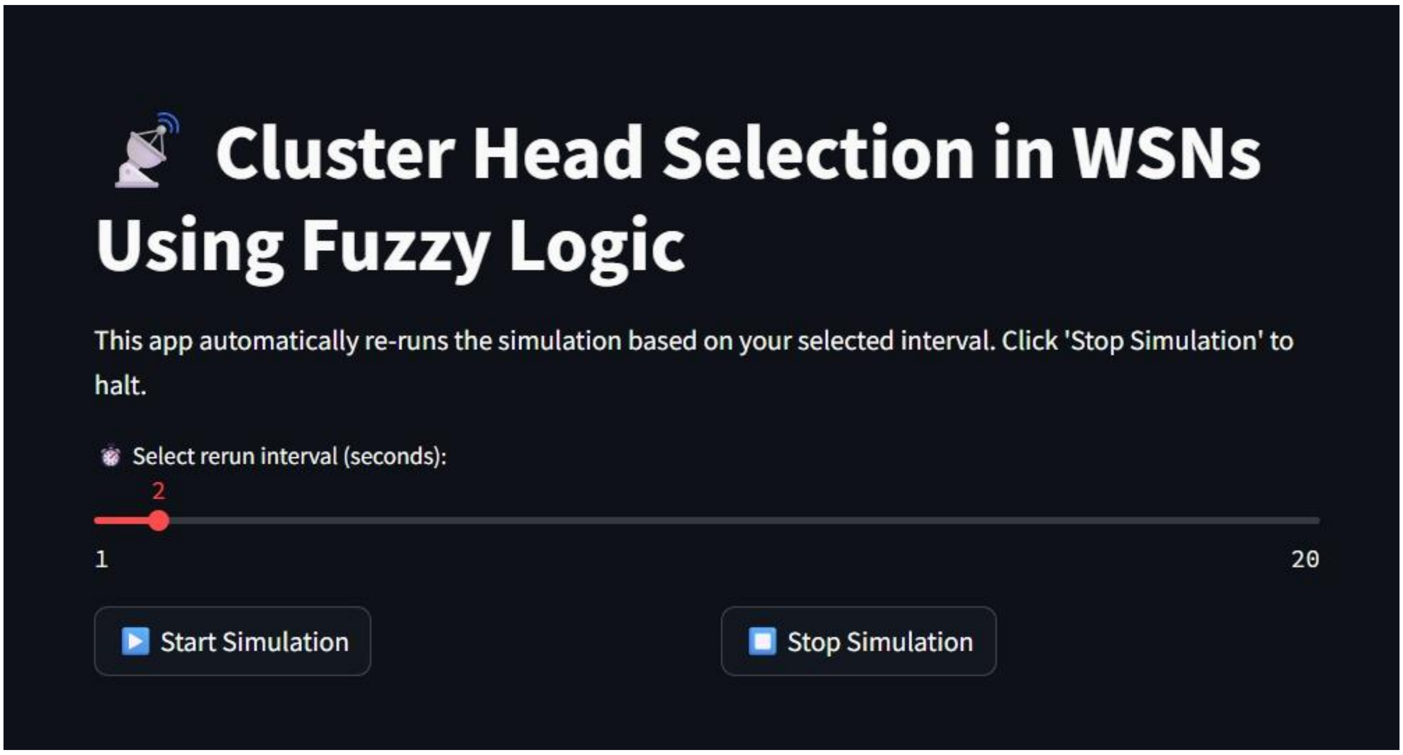


```python
    df = pd.DataFrame(results)
    st.subheader("📋 Simulation Data")
    st.dataframe(df)

    fig, ax = plt.subplots()
    for i, node in enumerate(nodes):
        color = 'red' if node.is_CH else 'blue'
        ax.scatter(node.x, node.y, c=color)
        ax.text(node.x + 1, node.y + 1, f"{i}", fontsize=8)

    ax.scatter(*BS_LOCATION, c='green', marker='X', s=100, label='Base Station')
    ax.set_title("WSN Node Deployment with Cluster Head")
    ax.grid(True)
    ax.legend()
    ax.set_xlabel("X coordinate")
    ax.set_ylabel("Y coordinate")
    st.pyplot(fig)

    st.success(f"✅ Cluster Head selected: Node #{best_node_idx}")

# --------- Streamlit UI ---------
st.set_page_config(page_title="Dynamic WSN Cluster Head Simulation", layout="centered")
st.title("📡 Cluster Head Selection in WSNs Using Fuzzy Logic")
st.markdown("This app automatically re-runs the simulation based on your selected interval. Click 'Stop Simulation' to halt.")


if 'running' not in st.session_state:
    st.session_state.running = False

if 'interval' not in st.session_state:
    st.session_state.interval = 10


st.session_state.interval = st.slider("⏱️ Select rerun interval (seconds):", 1, 20, st.session_state.interval)


col1, col2 = st.columns(2)
with col1:
    if st.button("▶️ Start Simulation"):
        st.session_state.running = True
        st.rerun()

with col2:
    if st.button("⏹️ Stop Simulation"):
        st.session_state.running = False


if st.session_state.running:
    simulate()
    st.warning(f"⏳ Re-running in {st.session_state.interval} seconds...")
    time.sleep(st.session_state.interval)
    st.rerun()
```



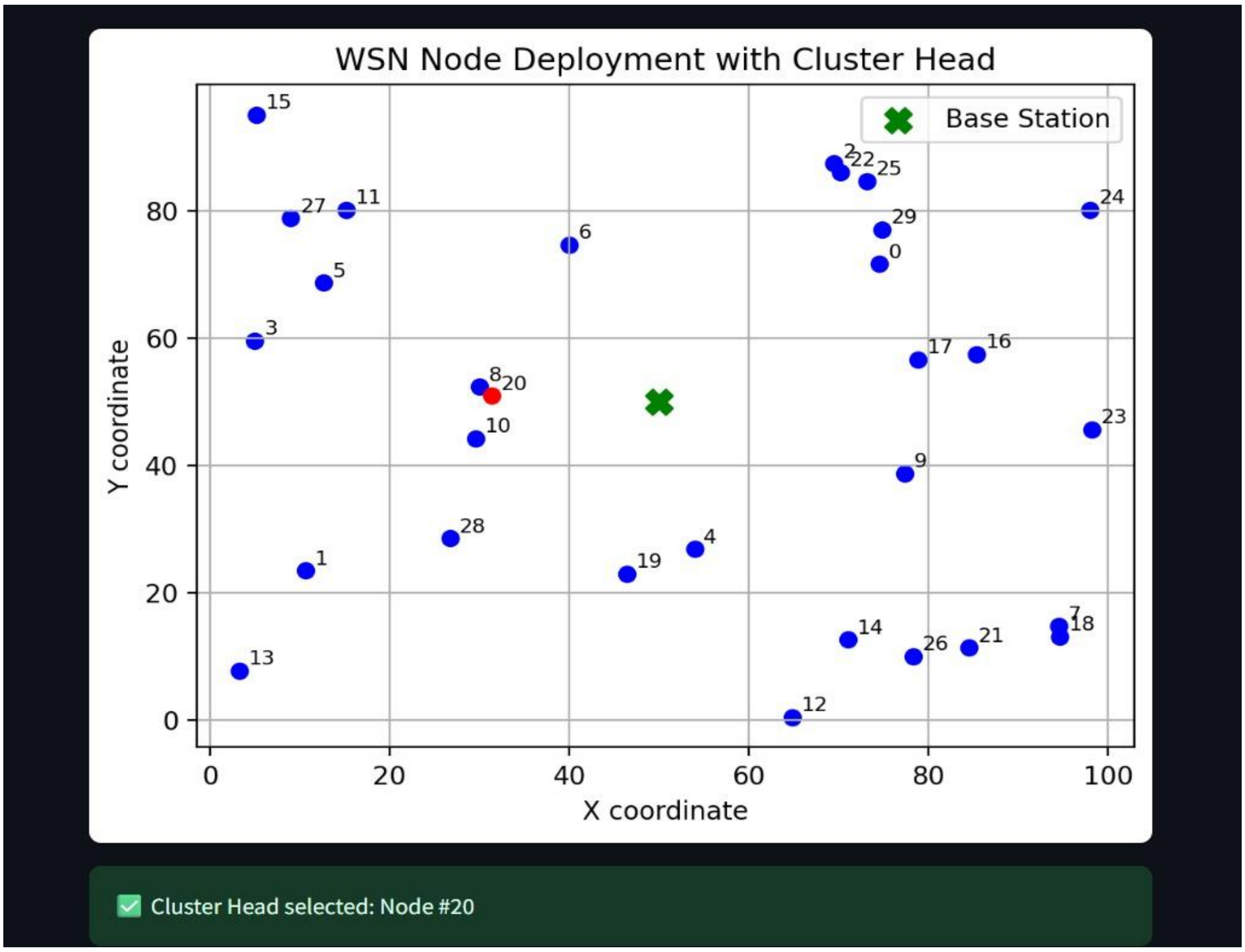
Simulation Data

	Node ID	X	Y	Energy	Distance to BS	Density	Fuzzy Score	Is Cluster Head
14	14	14.61	64.75	0.321	38.34	5	0.398	☐
15	15	93.73	8.86	0.615	60.04	4	0.542	☐
16	16	66.29	39.04	0.663	19.63	2	0.635	☐
17	17	36.35	5.81	0.382	46.25	1	0.432	☐
18	18	98.34	22.66	0.81	55.53	5	0.582	☐
19	19	18.53	31.52	0.338	36.5	1	0.403	☐
20	20	52.11	42.05	0.918	8.23	3	0.756	✓
21	21	84.78	99.53	0.384	60.52	1	0.437	☐
22	22	73.87	3.53	0.336	52.24	0	0.413	☐
23	23	56.71	66.87	0.882	18.16	3	0.713	☐
24	24	43.06	71.3	0.622	22.4	1	0.602	☐



Cluster Head Selection in WSNs Using Fuzzy Logic

This app automatically re-runs the simulation based on your selected interval. Click 'Stop Simulation' to halt.



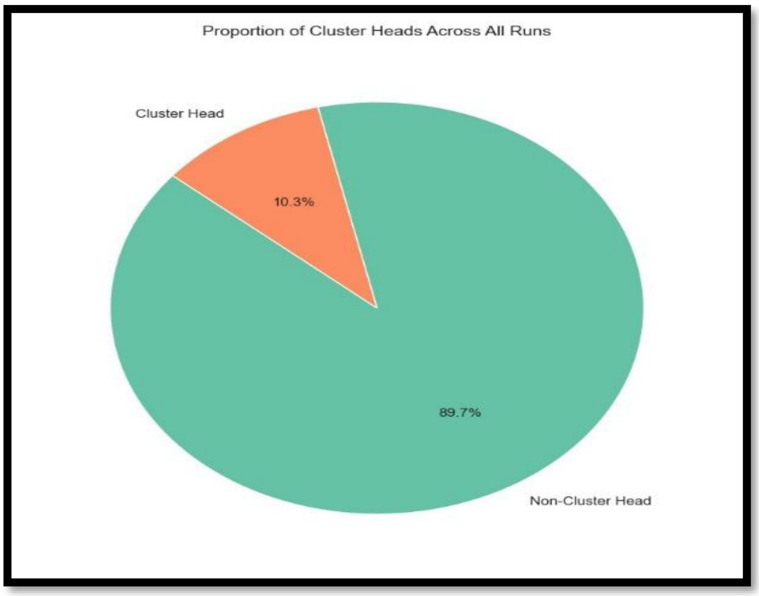
✅ Cluster Head selected: Node #20

3. Graphs

I. Proportion of Cluster Heads Across All Runs (Pie Chart)

This pie chart represents the proportion of sensor nodes selected as Cluster Heads (CHs) versus non-CH nodes across all simulation runs.

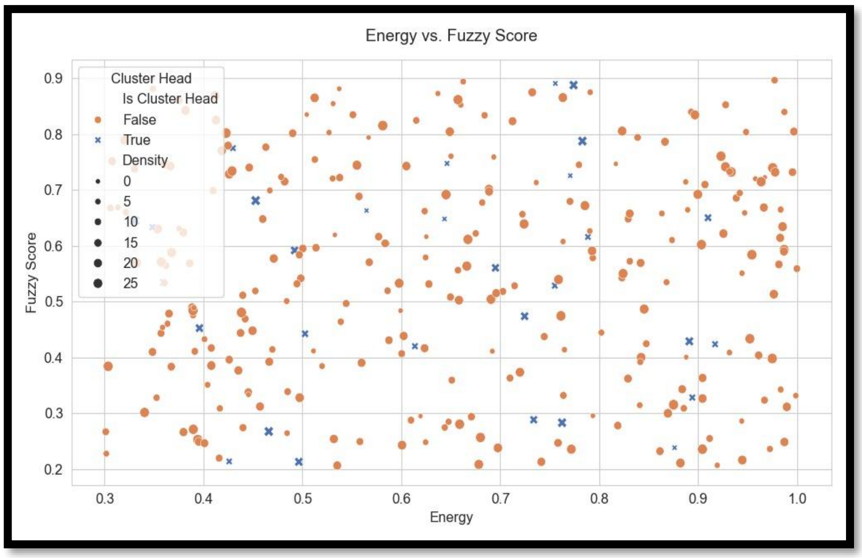
- **Observation**: Only 10.3% of the nodes were selected as CHs, while 89.7% remained regular nodes.
- **Inference**: The fuzzy logic-based system selects a small, optimal subset of nodes as CHs, ensuring energy-efficient communication by avoiding excessive cluster formation and maintaining balance in network overhead.



II. Energy vs. Fuzzy Score (Scatter Plot)

This scatter plot shows the relationship between the residual energy of nodes and their computed fuzzy scores, with additional information on node density and CH status.

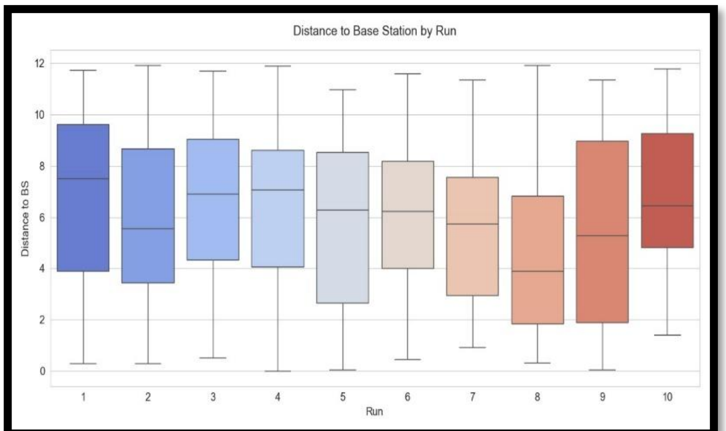
- **Orange Dots**: Represent nodes not selected as CHs.
- **Blue Crosses**: Indicate nodes selected as CHs.
- **Size of markers** corresponds to local node density.
- **Observation**:
    - CHs tend to lie in the higher fuzzy score range.
    - Nodes with higher energy and moderate to high density receive better fuzzy scores.
- **Inference**: The fuzzy inference system effectively integrates multiple parameters (energy, density, distance) to determine node suitability.



III. Distance to Base Station by Run (Box Plot)

This box plot displays the variation in distances from nodes to the base station across multiple simulation runs.

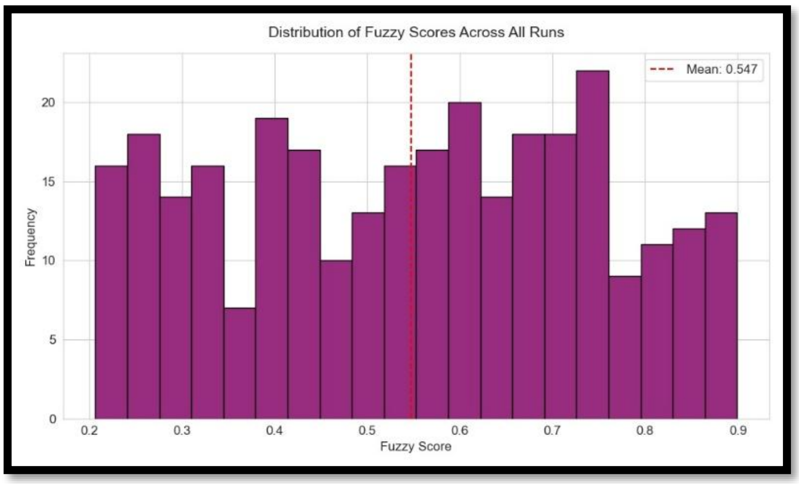
- **Observation**:
    - The median and spread of distances shift slightly across runs, indicating dynamic node positioning or random initializations.
    - Runs with a lower median distance likely yielded CHs closer to the base station, which is favorable for energy conservation.
- **Inference**: This analysis helps understand the distribution trends and ensures that the fuzzy system remains adaptive regardless of node spread.



IV. Distribution of Fuzzy Scores Across All Runs (Histogram)

This histogram illustrates the frequency distribution of fuzzy scores computed across all simulation runs.

- **Vertical dashed line**: Indicates the mean fuzzy score (~0.547).
- **Observation**:
  - Fuzzy scores range from 0.2 to 0.9, with a relatively uniform distribution.
  - A slight skewness toward higher scores suggests more nodes qualify as potential CH candidates under favorable conditions.
- **Inference**: The fuzzy system provides a balanced range of outputs, allowing flexibility in CH selection without clustering bias.

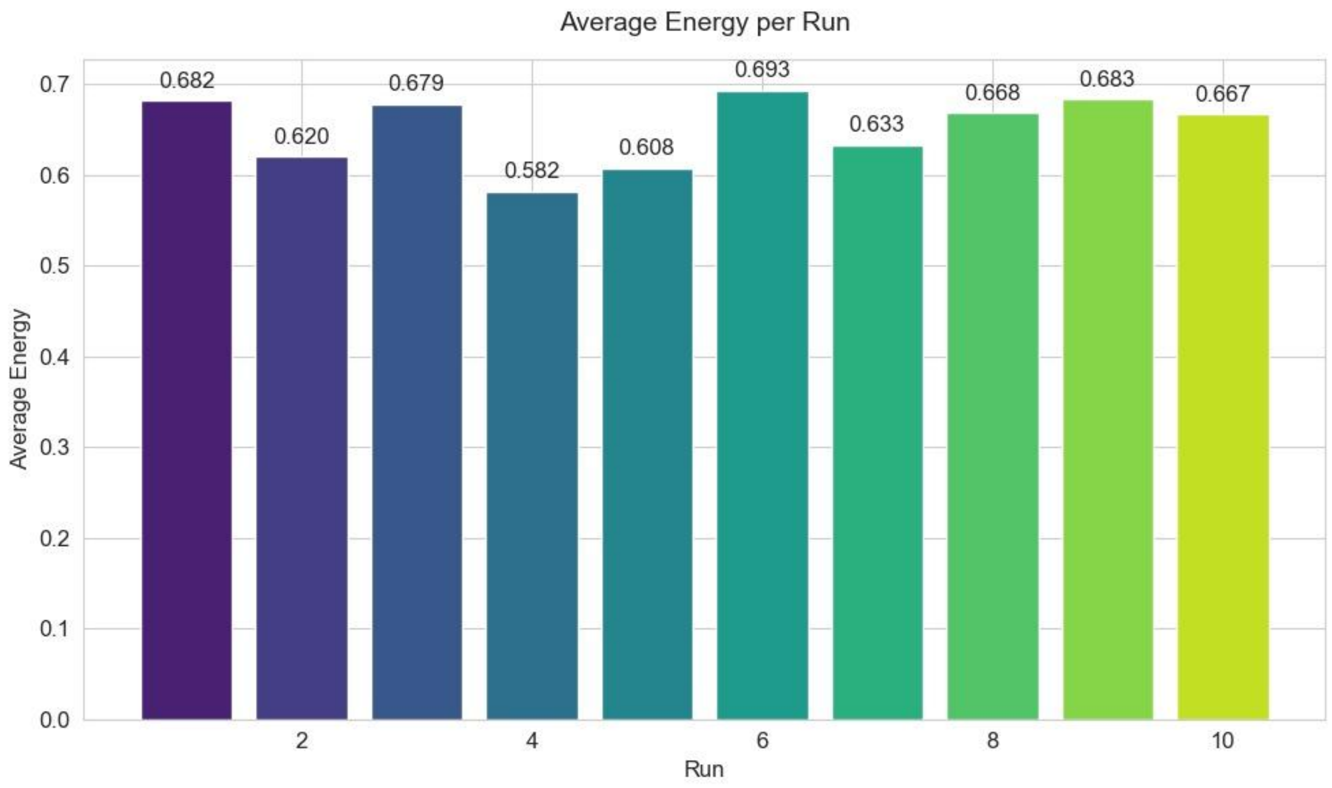


V. Graph: Average Energy per Run

This bar chart illustrates the average residual energy of sensor nodes at the end of each simulation run in the wireless sensor network.

- **X-axis**: Represents the simulation run number (from 1 to 10).
- **Y-axis**: Indicates the average energy of nodes after the cluster head (CH) selection process.
- **Observation**:
  - The average energy per run fluctuates slightly between 0.582 and 0.693.
  - Run 6 has the highest average residual energy (0.693), indicating more energy-efficient CH selection and communication during that run.
  - Run 4 shows the lowest average energy (0.582), possibly due to suboptimal node distribution or higher communication cost during that round.
- **Inference**:
  - The energy levels across runs show that the fuzzy logic-based CH selection strategy helps maintain a consistently moderate-to-high average energy level, promoting network longevity.
  - Slight variations between runs reflect the effect of randomized node placement and dynamic network conditions, but overall energy management remains effective.

This graph supports the conclusion that the fuzzy-based approach achieves balanced energy usage and adapts well to different run conditions.



Average Energy per Run
Average Energy
Run
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2
4
6
8
10
0.682
0.620
0.679
0.582
0.608
0.693
0.633
0.668
0.683
0.667