1. Project Objectives

- **Routing Protocol Comparison:**
  The simulation implements two protocols (DSR and SFA) to study their behavior in dynamic and energy-constrained wireless networks. It compares metrics such as packet delivery ratio, energy consumption, and route establishment efficiency.
- **Energy & Activity Management:**
  Each node has a limited energy budget. The simulation models energy consumption for packet transmissions (with costs based on distance) and reception. Nodes can become inactive if their energy falls below a defined threshold, and they also enter a sleep mode after periods of inactivity.
- **Security & Malicious Behavior:**
  The framework includes a malicious node that simulates a Denial-of-Service (DoS) attack by flooding the network with bogus packets. It also incorporates an alarm mechanism whereby nodes (or a base station) can detect anomalous packet loss or energy discrepancies and broadcast block messages to isolate suspicious nodes.
- **Machine Learning (ML) and Reinforcement Learning (RL):**
  - **Traffic Prediction:** A deep learning model (using TensorFlow/Keras) is trained on simulation data to predict network traffic (e.g., packets sent) based on features like time and protocol type.
  - **Routing Decisions via RL:** A custom OpenAI Gym environment is created for training an RL agent (using PPO from Stable Baselines3) that can make routing decisions based on the current state (such as node energy, sequence number, and routing table state).

---

2. Core Components

a. Global Metrics and Logging

- **Metrics Summary:**
  A global dictionary (`metrics_summary`) tracks the number of various packet types (RREQ, RREP, RERR, forwarding, alarms, ACKs, DoS packets) as well as counts of established routes and deactivated nodes.
- **Logging:**
  The logging system is configured to record detailed debug information to a file (`simulation_output.txt`) and higher-level info to the console. This provides traceability for events such as packet transmissions, energy consumption, route discoveries, and security events.

b. Packet Class

- **Definition:**
  The `Packet` class encapsulates all necessary information about a network packet including:
  - **Type:** (e.g., RREQ, RREP, FORWARDING, ALARM, ACK, DOS, etc.)
  - **Source/Destination:** Identifiers for the sender and intended recipient.
  - **Sequence Number:** For duplicate detection and route management.
  - **Payload and Routing Path:** For carrying data and tracking the route taken.
  - **TTL and Final Destination:** To limit packet lifespan and specify ultimate recipients.

c. Routing Protocol Implementations

- **DSRProtocol:**
  Implements methods specific to the DSR protocol:
  - **Route Discovery:** Broadcasting RREQ packets.
  - **Route Reply:** Sending RREP packets along the reverse path when the destination is reached.
  - **Data Forwarding:** Using discovered routes to forward data packets.
  - **Error Handling:** Sending RERR messages when issues are detected.
- **SFAProtocol:**
  Implements a simplified flooding-based approach where:
  - Route requests (RREQ) are forwarded until the destination is reached.
  - A route reply (RREP) is sent back along the discovered path.
  - Similar mechanisms exist for forwarding and error messages.

d. Node Functionality

- **Node Class:**
  Represents a network node with advanced features:
  - **Energy Management:**
    Nodes consume energy for sending and receiving packets. If the energy falls below a minimum operating level, the node is deactivated.
  - **Sleep Cycle:**
    After a period of inactivity, nodes enter sleep mode to conserve energy and wake up upon activity.
  - **Routing Table:**
    Each node maintains a routing table with multiple (up to five) cached routes per destination. Routes expire after a set time.
  - **Packet Processing:**
    The main process (`run`) handles incoming packets from a SimPy store (queue), processing them according to their type (data, routing, alarm, block, ACK, DOS, etc.).
  - **Sensor Data Transmission:**
    Emulating wireless sensor networks (WSN), nodes periodically sense data and forward it to a base station.

    - **ML & RL Integration:**
      Hooks allow the node to predict network traffic and make routing decisions based on ML predictions or RL agent actions.
- **MaliciousNode Subclass:**
  Inherits from `Node` and continuously sends out DOS packets to simulate a network attack.
- **BaseStation Subclass:**
  A special node that primarily handles ALARM messages and coordinates network-wide responses such as broadcasting block messages when a malicious node is detected.

e. Network Topology and Simulation Environment

- **Network Class:**
  Manages the network topology using NetworkX to represent nodes and edges. Key functionalities include:
    - **Node and Edge Creation:**
      Nodes are created (including normal, base station, and malicious nodes), and edges are either provided via lists or generated randomly.
    - **Packet Delivery:**
      Uses SimPy's event scheduling to simulate delays and deliver packets to the appropriate node's queue.
    - **Neighbor Discovery:**
      Provides methods to retrieve the neighbors of a given node based on the network graph.
- **SimPy Environment:**
  SimPy is used as the simulation engine to schedule and run all processes concurrently (packet sending, route discovery, sensor transmissions, sleep management, etc.).

f. Data Collection and Performance Evaluation

- **Data Collection Process:**
  Periodically collects metrics such as node energy, packet counts, and protocol type, storing them in a Pandas DataFrame. This data is later exported to CSV files for further analysis.
- **Performance Metrics:**
  After the simulation runs, metrics are computed to compare the performance of DSR and SFA. These include:
    - Total packets sent and received.
    - Packet Delivery Ratio (PDR).
    - Total energy consumed.
    - Detection rate of malicious behavior.

g. Machine Learning and Reinforcement Learning

- **Traffic Prediction Model:**
  After the initial simulation run, collected data is used to train a neural network (using

Keras) that predicts traffic load based on time and protocol. A scaler (from scikit-learn) is used to normalize the features.

- **RL Agent Training:**
  A custom Gym environment (`RoutingEnv`) is defined to represent the node's state (energy, sequence number, and routing table count). An RL agent is then trained using PPO (from Stable Baselines3) to learn how to make effective routing decisions.
- **Integration in Simulation:**
  In a subsequent simulation run, nodes are equipped with the trained ML and RL models. This enhanced simulation tests how adaptive routing decisions (informed by predictions and RL policies) can impact network performance and energy efficiency.

---

3. Simulation Flow

1. **Initialization:**
   - Two separate network topologies are set up—one for DSR and one for SFA—using either provided edge lists or randomly generated connections.
   - Nodes are instantiated with initial energy, positions, routing tables, and the appropriate protocol handler.
2. **First Simulation Run:**
   - Various processes are launched:
     - **Route Discovery Processes:** Nodes periodically initiate route discovery.
     - **Packet Forwarding Processes:** Nodes forward data packets based on their current routing tables.
     - **Alarm Processes:** Nodes with high packet loss trigger ALARM messages to the base station.
   - Data is collected periodically for performance analysis.
   - At the end of the run, simulation data is exported, and the traffic prediction ML model is trained using the collected data.
3. **Second Simulation Run (Enhanced with ML/RL):**
   - A new simulation environment is created with nodes now equipped with the ML (for traffic prediction) and RL (for routing decisions) models.
   - Additional processes such as RL-based routing initiation are started.
   - Data collection continues, and after the simulation ends, performance metrics are computed and compared between the two protocols.
4. **Results and Summary:**
   - Detailed performance metrics (e.g., packet delivery ratio, energy consumption, malicious detection rate) are computed.
   - The simulation summary and metrics are exported to CSV and text files for analysis.
   - Logging output provides a step-by-step trace of simulation events for debugging and validation.

---

4. Conclusion

This project demonstrates a multi-faceted simulation environment that not only tests traditional ad hoc routing protocols (DSR and SFA) under various network conditions and attacks but also integrates modern ML/RL techniques to enhance routing decisions. The simulation framework allows for:

- **In-depth Performance Analysis:** By collecting and exporting rich datasets and metrics.
- **Security Testing:** Through simulated malicious attacks and an alarm/blocking mechanism.
- **Adaptive Routing:** Using both predictive and learning-based methods to improve overall network efficiency.