Journal of Rare Cardiovascular Diseases
ISSN: 2299-3711 (Print) | e-ISSN: 2300-5505 (Online)
www.jrcd.eu
RESEARCH ARTICLE

Enhancing IOT Network Efficiency and Security: A Dos-RI and ICSO-Driven SDN Architecture with Trust-Aware MGEO Attack Detection

Mrs. M. Senthamil Selvi[1] and Dr. L. Sudha[2]
[1]Research Scholar, Department of Computer Science, VET Institute of Arts and Science (Co-education) college, Erode, Tamil Nadu, India.
[2]Associate Professor, Department of Computer Science, VET Institute of Arts and Science (Co-education) college, Erode, Tamil Nadu, India

*Corresponding Author
Mrs. M. Senthamil Selvi

Article History
Received: 10/07/2025
Revised: 14/08/2025
Accepted: 15/09/2025
Published: 30/09/2025

Abstract: The rapid expansion of the Internet of Things (IoT) necessitates agile, energy-efficient, and secure networking solutions capable of adapting to heterogeneous devices and volatile environments. This paper proposes a closed-loop Software-Defined Networking (SDN) architecture that integrates an enhanced Dynamic Objective Selection Reinforcement Learning (DOS-RL) agent with Improved Chicken Swarm Optimization (ICSO) for real-time hyperparameter tuning. The DOS-RL agent dynamically adjusts routing strategies to balance critical objectives—energy efficiency, latency, and packet delivery ratio—ensuring stable performance under fluctuating network conditions. A trust- and QoS-driven fitness function informs routing decisions, while Modified Golden Eagle Optimization (MGEO) with a stooping mechanism is employed to detect and isolate malicious or unreliable nodes, strengthening network security. The combined use of trust management, adaptive routing, and optimized parameter tuning enables the proposed framework to promptly respond to sudden topological and traffic changes, extend network lifetime, reduce latency, and enhance resilience against security threats. Experimental evaluations demonstrate superior performance compared to conventional routing and attack detection mechanisms, offering a robust, scalable, and secure solution for next-generation IoT deployments.

Keywords: IoT, Software-Defined Networking, Reinforcement Learning, Chicken Swarm Optimization, Golden Eagle Optimization, Trust-Aware Routing, Cybersecurity.

INTRODUCTION

The Internet of Things (IoT) encompasses a framework of computing devices, objects, mechanical and electronic machines, animals, and humans equipped with distinctive identifiers, facilitating data transmission within the network without the need for direct connections between them [1]. The SDN paradigm aims to simplify network management by separating control and data planes. The SDN architecture initiates changes in IoT network communication patterns, thus shaping a new approach for powering IoT networks. With a significant amount of data in these systems, efficient traffic management and load balancing reduce the additional effects of data-flow generation. Implementing dynamic traffic management enables operators to independently monitor and coordinate bandwidth fluctuations, which is particularly advantageous for global IoT service providers anticipating exponential growth in both the number of IoT devices and the associated data. The inherent capabilities of SDN, including automation, resource provisioning, programmability, and coordination, can provide substantial value in an IoT environment [2]. Software-driven analysis and traffic control by SDN can be applied to IoT for efficient traffic management.

In SDN architecture, three planes exist. The lowest one is the data plane, housing SDN-enabled switches, functioning solely as packet forwarders without involvement in decision-making. Control plane, encompassing the controller, handles routing decisions and is responsible for forming routing rules upon request from the data plane. Additionally, the controller is responsible for making various decisions regarding data packets. The third plane incorporates an application programming interface that hosts applications for controlling the network. The link between the application plane and the control plane is termed the northbound interface, while the connection between the control plane and the data plane is known as the southbound interface [3]. Communication within the southbound interface is governed by protocols such as OpenFlow [4], which has become the standard since its inception. Typically, packets are routed according to predefined flow rules listed in the flow table on an OpenFlow enabled switch. The inputs to these flow tables include actions, statistics, and match fields. As can be guessed, the actions field determines the performance of each packet, the statistics field tracks the packets matching each flow entry, and the matching field analyze the received packets.

Ensuring the secrecy of devices and networks is crucial to accommodate diverse devices, vendors, and users on a unified platform [5]. SDN enhances security in IoT deployments by providing centralized control, dynamic segmentation, policy-based access control, and deep visibility into network traffic, thereby improving the overall resilience of IoT networks against cyber threats [6,7]. However, challenges still persist in SDN-IoT, primarily stemming from evolving hacker capabilities [8,9]. This paper furnishes current insights into the

ongoing research developments on security and privacy challenges in SDN-enabled IoT systems, along with approaches aimed at protecting and ensuring the stability of these systems. The resilience and integrity of SDN-enabled IoT environments against dynamic cyber threats requires a comprehensive security strategy, including device security, network infrastructure protection, data privacy, and continuous monitoring implementation [10].

Software-Defined Networking (SDN) has emerged as a promising solution to these challenges by decoupling the control plane from the data plane and enabling centralized network intelligence. Through global network visibility and programmability, SDN facilitates dynamic routing adjustments and efficient resource allocation. Nevertheless, the effectiveness of SDN in IoT environments depends on adaptive decision-making mechanisms that can optimize multiple, often conflicting, performance objectives while simultaneously mitigating evolving security threats.

Reinforcement learning (RL) has proven to be an effective technique for adaptive routing due to its ability to continuously learn optimal policies from environmental feedback without requiring explicit network models. By dynamically selecting routing actions to minimize latency, reduce energy consumption, and enhance packet delivery ratios, RL-based approaches offer flexibility in handling varying traffic loads and network topologies. However, most existing RL-based routing strategies rely on static reward formulations and fixed hyperparameters, limiting their ability to respond effectively to rapidly changing IoT conditions. Hyperparameter tuning in RL is particularly critical, as improper selection may result in poor convergence or unstable performance, which is undesirable in real-time IoT operations.

To overcome these limitations, bio-inspired metaheuristic optimization algorithms have been increasingly adopted for adaptive parameter tuning in learning-based systems. Among them, Chicken Swarm Optimization (CSO) has shown promise due to its simplicity and strong global search capability. In this work, we develop an Improved Chicken Swarm Optimization (ICSO) algorithm that incorporates chaotic initialization and adaptive role transition strategies to enhance convergence speed and escape local optima, thereby ensuring efficient RL parameter tuning in highly dynamic IoT environments.

In addition to routing efficiency, security remains a critical concern in IoT networks, which are susceptible to malicious node behaviors such as packet dropping, data manipulation, and resource exhaustion attacks. Trust-based models that evaluate node behavior based on historical performance have proven effective in detecting and mitigating such threats. To further improve detection accuracy and response time, this study employs a Modified Golden Eagle Optimization (MGEO) algorithm with a stooping search strategy, inspired by the rapid diving behavior of golden eagles when capturing prey. This approach enables fast and accurate identification of compromised nodes, allowing the network to isolate unreliable or malicious devices proactively, thereby ensuring reliable and secure data delivery.

This paper proposes an integrated closed-loop SDN-enabled IoT architecture that combines a Dynamic Objective Selection Reinforcement Learning (DOS-RL) agent for adaptive multi-objective routing, ICSO-based hyperparameter tuning for improved learning performance, and MGEO-based malicious node detection for enhanced network security. The proposed approach effectively balances energy efficiency, latency, and packet delivery ratio while fortifying the network against internal attacks. Simulation results demonstrate that the framework outperforms conventional routing and attack detection methods in terms of energy consumption, latency, detection accuracy, and network lifetime, establishing it as a robust and scalable solution for next-generation IoT deployments.

RELATED WORK

Numerous studies have extensively examined the security challenges related to IoT from an SDN perspective in the literature. In [11], the authors tackled security concerns and proposed solutions for SDN-based IoT systems. This paper examined the defensive techniques employed in prior research to ensure adequate security and privacy in SDN-based IoT systems, offering a statistical analysis of the current literature. Multiple vulnerabilities and potential attacks within the IoT landscape have been outlined in [12]. This survey emphasized the critical importance of SDN security in safeguarding IoT systems and identified several areas where SDN and Network Function Virtualization (NFV) could be improved. It also shared insights gained from implementing SDN-based security methods in IoT environments and provided a comparison with conventional security measures.

In [13], an examination of various SDN-based technologies was conducted, focusing on their relevance in meeting the demands of IoT across core, access, and data center networking domains. The discussion encompassed the benefits of SDN-based technologies in these areas, along with the challenges and prerequisites they entail within the scope of IoT applications. According to [14], various types of DDoS attacks were categorized into three layers of SDN. Thereupon, this study presented an analysis of recent progress in DDoS detection and mitigation research designed to address these vulnerabilities.

Another study adopted an autonomous learning-based framework for intrusion detection in SDN environments, incorporating data, control, and application layers [15]. Network traffic was routed through forwarding switches

to the intrusion detection framework, which used a tree-based machine learning classifier to detect anomalies, categorized as collective or point anomalies based on packet IP information. However, this work relied solely on IP information, leading to less effective anomaly classification.

A hybrid intrusion detection framework was also proposed for SDN-based Internet of Things (IoT) environments, utilizing gated recurrent units (GRUs) and long short-term memory (LSTM) [16]. This hybrid deep learning model processed user traffic and performed feature extraction, selection, and multi-class classification based on four categories: cross-site scripting, distributed denial-of-service, bot attacks, and normal traffic. Nonetheless, the framework overlooked user legitimacy, which resulted in an increase in malicious traffic.

In the context of SDN-assisted IoT healthcare environments, a hybrid deep learning approach combining LSTM and GRU was employed [17]. Wearable IoT devices transmitted continuous traffic to an SDN controller via switches. The hybrid intrusion detection module in the controller classified traffic as benign or malicious based on packet-level information. However, the study inadequately addressed authentication and classification metrics, which limited its effectiveness.

In [18], the authors presented a scheme to optimize the routing path in the SDWSN using the Reinforcement Learning (RL) algorithm. Their approach incorporates energy efficiency and network Quality-of-Service (QoS) parameters into the reward function. The proposed routing scheme compares various SDN-based techniques, including the traditional SDN, energy-aware SDN (EASDN), QR-SDN, TIDE, as well as non-SDN-based techniques such as Q-learning and RL-based routing (RLBR). The results indicate that the RL-based SDWSN outperforms other approaches in terms of the network lifetime and packet delivery ratio. In this research work introduces a routing scheme that aims to enhance the capabilities of SDWSN-IoT by integrating deep learning techniques, leveraging the inherent features of SDWSN-IoT such as network programmability and comprehensive topology monitoring. The proposed framework facilitates dynamic learning and the adaptation to network changes, enabling the proactive installation and continuous updating of routes based on rapidly changing link states, thereby ensuring that swift and efficient routes are found for data forwarding.

PROPOSED METHODOLOGY

The proposed framework introduces an integrated closed-loop Software-Defined Networking (SDN) architecture specifically designed to enhance routing efficiency and security in dynamic Internet of Things (IoT) environments. The key components include a Dynamic Objective Selection Reinforcement Learning (DOS-RL) agent for adaptive multi-objective routing, an Improved Chicken Swarm Optimization (ICSO) algorithm for hyperparameter tuning, and a trust-aware security mechanism based on a Modified Golden Eagle Optimization (MGEO) algorithm for malicious node detection and isolation. These components interact within an SDN-enabled IoT environment, ensuring that routing policies are dynamically adjusted, learning models remain optimally tuned, and compromised nodes are detected and mitigated in real-time.

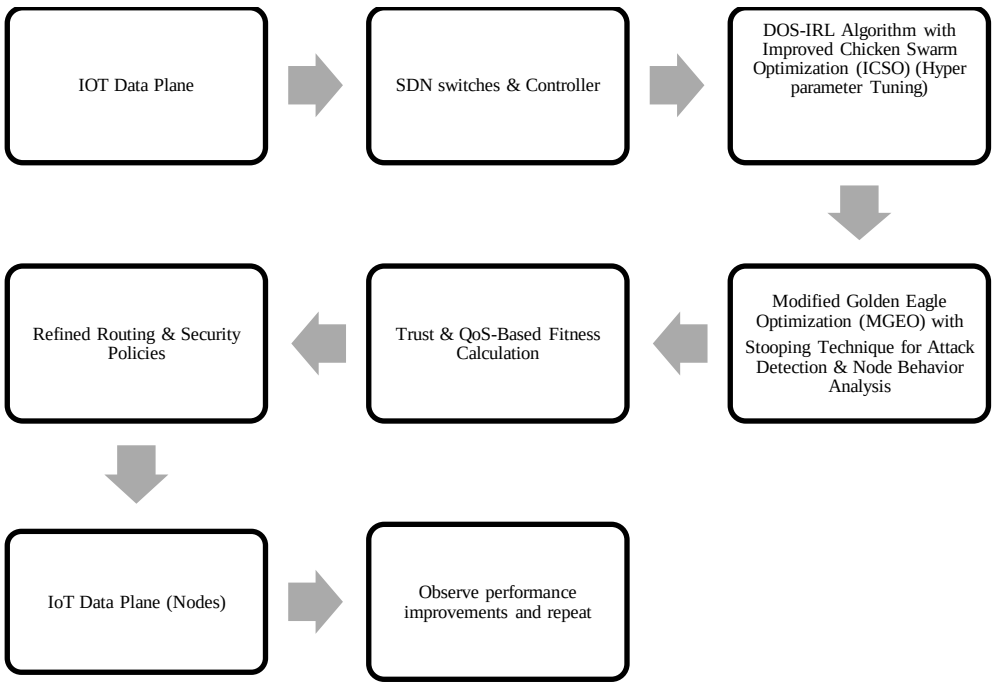

Figure 1: Proposed Methodology flow

A. Initialization and System Overview

The proposed framework introduces an integrated closed-loop Software-Defined Networking (SDN) architecture specifically designed to enhance routing efficiency and security in dynamic Internet of Things (IoT) environments.

IoT Data Plane

- Comprises sensor nodes, local gateways, and monitoring systems responsible for sensing, collecting, and forwarding data.
- These devices generate continuous network traffic while operating under energy constraints, requiring efficient data transmission and monitoring.

SDN Controller (Control Plane)

- Serves as the centralized intelligence hub for routing and policy management.
- Collects real-time metrics from the IoT data plane and enforces flow rules for optimal performance and security.

Dynamic Objective Selection Reinforcement Learning (DOS-RL) Agent

- Responsible for adaptive multi-objective routing decisions.
- Balances energy efficiency, latency, and packet delivery ratio dynamically based on real-time network state.

Improved Chicken Swarm Optimization (ICSO) Algorithm

- Handles hyperparameter tuning of the DOS-RL agent to improve convergence speed and routing stability.
- Uses enhanced role transition and chaotic initialization for superior optimization.

Modified Golden Eagle Optimization (MGEO) Security Mechanism

- Detects and isolates malicious or unreliable nodes using a stooping search strategy inspired by golden eagle hunting behavior.
- Employs trust metrics (e.g., packet forwarding behavior, node reliability) to ensure secure data flow.

Feedback Loop

- Maintains continuous interaction between IoT devices and the SDN controller.
- Ensures routing policies are dynamically adjusted, learning models are optimally tuned, and compromised nodes are proactively mitigated.

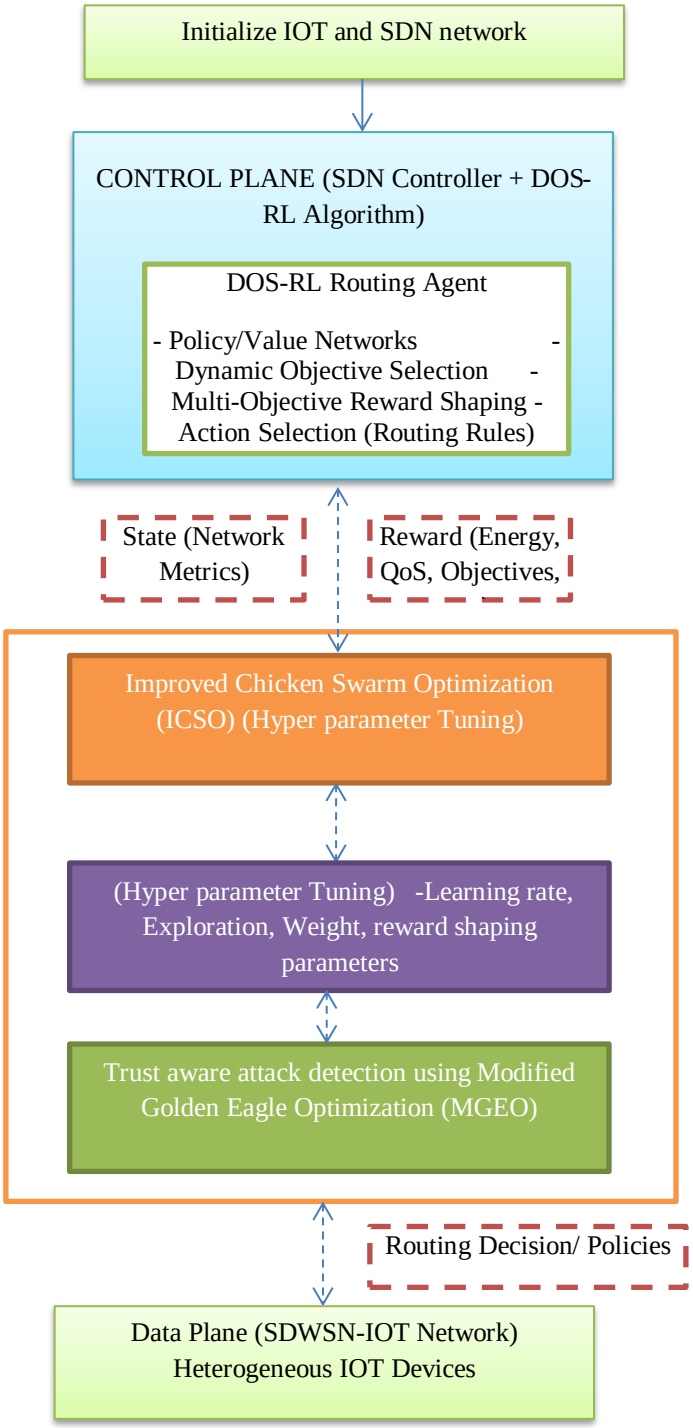

Figure 2: Proposed Interconnected Diagram

B. AIoT Data Plane: Includes sensor nodes, local gateways, and monitoring systems

The **AIoT data plane** is responsible for the physical network infrastructure, consisting of sensor nodes, local gateways, and monitoring systems. These components are responsible for collecting data, transmitting it, and potentially acting on certain routing decisions before the data reaches the SDN controller.

Components of the AIoT Data Plane:

Sensor Nodes:

1. These are the edge devices that collect real-time data from the environment (e.g., temperature, humidity, energy consumption, traffic, etc.).
2. Sensor nodes are typically energy-constrained and need to minimize energy consumption while maximizing data collection and transmission efficiency.

Local Gateways:

1. Gateways aggregate data from multiple sensor nodes, process it locally, and forward 2it to the SDN controller.
2. They may handle some routing tasks to reduce the load on the controller by performing local decisions based on high-level policies.

Monitoring Systems:

1. These are responsible for monitoring network conditions and the performance of IoT nodes and gateways (e.g., packet loss, delay, energy consumption).
2. They collect feedback from the data plane and report it back to the SDN controller for further policy adjustments.

AIoT Data Plane Workflow:

- Sensor nodes continuously collect data and report it to local gateways.
- Local gateways may preprocess the data and send summaries or aggregated data to the SDN controller.
- Monitoring systems track the network conditions (e.g., congestion, latency, energy use) and relay this information back to the SDN controller to provide real-time status updates.

C. SDN Controller (Control Plane): Centralized decision-making and policy application.

The **SDN controller** serves as the brain of the network, responsible for centralized decision-making, policy generation, and application. The SDN controller manages the **control plane** and communicates with the **data plane** (sensor nodes, gateways, and monitoring systems).

Functions of the SDN Controller:

Centralized Decision Making:

- The SDN controller receives real-time metrics from the data plane (IoT nodes, gateways, and monitoring systems).
- It processes these metrics to make high-level decisions about network behavior, such as optimizing routing, managing congestion, or deciding when to reroute traffic.

Policy Application:

- The SDN controller generates routing policies based on the **DOS-RL agent**'s decisions (using reinforcement learning).
- It pushes these policies to the data plane for execution. The policies can include routing tables, QoS (Quality of Service) parameters, and energy management instructions.

Dynamic Policy Adjustment:

- The controller continuously adjusts policies based on feedback from the monitoring systems. If performance metrics like energy, latency, or packet loss deteriorate, the controller may trigger policy updates to improve the network's performance.

Monitoring and Feedback Loop:

- The controller continuously monitors the feedback from the AIoT data plane and uses **Bayesian Optimization** (BO) to fine-tune the **DOS-RL agent's hyperparameters** for better routing decisions.

SDN Controller Workflow:

- Collect network data from the AIoT data plane (sensor nodes, gateways, and monitoring systems).
- Use the **DOS-RL agent** to evaluate the network state and decide on optimal routing actions.
- Apply these routing decisions as network policies and push them to the data plane.
- Continuously monitor performance and adjust policies based on real-time feedback, using BO to optimize the DOS-RL agent.

D. The DOS-RL Routing Scheme

The realistic conditions of an SDWSN-IoT network are subjected to dynamic changes in resources such as battery capacity, CPU capacity, memory, and bandwidth, as well as link quality changes during network operation. RL-based routing schemes have demonstrated considerable advantages in designing network operation policies that can handle such changes. However, such routing protocols often fail to respond quickly enough to such changes. To address this issue, we propose the DOS-RL routing scheme which allows the RL agent to learn from multiple correlated objectives simultaneously and to adaptively choose the objective it believes is the most promising from the current state. Therefore, we have turned the problem of energy efficiency into an RL process by modeling it into a four-tuple(S, A, P, R), defining the states (S), actions (A), policy (P), and reward functions (R) of the DOS-RL scheme.

- **The Reward Function (R):** In reinforcement learning, the agent evaluates the effectiveness of its actions and improves its policy by relying on rewards collected from the environment. The rewards obtained are typically dependent on the actions taken, with varying actions resulting in differing rewards. To implement the proposed DOS-RL scheme, we define three reward functions, each corresponding to one of the correlated objectives: the selection of routing paths with sufficient energy for data forwarding (o_1), load balancing (o_2), and the selection of paths with a good link quality for a reliable data transfer (o_3). Route selection based on any of these objectives is expected to improve the energy efficiency of the IoT network system. Details on each objective of the DOS-RL scheme and its related reward functions are provided below:

i. **Energy-consumption:** To achieve objective o_1, we consider the energy-consumption parameter, which is a critical factor in determining the overall energy efficiency of an SDWSN: for instance, in a scenario where node i forwards a packet to node j. The reward received by node i for selecting node j as its relay node in terms of energy consumption is estimated by the reward function $RE_{i,j}$ for the state-action pair, (s_i, s_j) as:

$$RE_{i,j} = Remaining\ Energy\ (s_j) - Remaining\ Energy\ (s_i) \quad (1)$$

where $Remaining\ Energy\ (s_j)$ represents the remaining energy of node i in state s_i as a percentage; meanwhile, $Remaining\ Energy\ (s_i)$ represents the remaining energy of node j in state $s_j$as a percentage. The formula calculates the difference in remaining energy between node i and j after the data transmission. A higher value of $RE_{i,j}$ indicates that node i has consumed less energy in forwarding the packet to node j, which is desirable to achieve the energy efficiency objective (o_1).

ii. **Load balance:** To optimize energy efficiency and network performance in the SDWSN, the careful selection of relay nodes and balanced workload distribution are crucial. Otherwise, some nodes along overused paths may become overloaded, leading to bottlenecks and delays, which can result in a degraded network performance. For objective o_2, we utilize the parameter-available buffer length to estimate the degree of queue congestion in relay nodes. The reward $R\_Q_{i,j}$ for load balancing as observed by node i when selecting node j is computed as follows:

$$R\_Q_{i,j} = \frac{Available\_Buffer\_Length(s_j)}{Max\_Buffer\_Length} - \frac{Load\ (s_i)}{Max\_Load} \quad (2)$$

where $Available\_Buffer\_Length(s_j)$ represents the available buffer length of node j in state s_j, Max_Buffer_Length represents the maximum buffer length of a node, $Load\ (s_i)$ represents the current load of node i in state s_i, and Max_Load represents the maximum load capacity of a node. This formula considers both the available buffer length of node j and the current load of node i. The second term subtracts the load of node i from the load balancing reward value, allowing for a

more comprehensive estimation that takes into account both node i and node j in the load balancing process. A higher value of $R\_Q_{i,j}$ indicates a better load balancing situation for the given state-action pair (s_i, s_j).

iii. **Link quality:** A wireless link can be measured by retrieving useful information from either the sender or receiver side. To achieve o_3, we use simple measurements to estimate the link quality based on the parameter packet reception ratio (PRR), measured as the ratio of the total number of packets successfully received to the total number of packets transmitted through a specific wireless link between two nodes. Unlike other sophisticated techniques, this approach involves a low computation and communication overhead. Instead of using an instant value of the PRR, we calculate an average-over-time using an Adaptive Weighted Moving Average (AWMA) filter. Suppose node i is forwarding data packets to node j, the reward $R\_LQ_{i,j}$ received by node j based on PRR using AWMA is estimated as follows:

$$R_{LQ_{i,j}} = PRR_{est_{i,j}(k-1)} \cdot \propto_{\text{AWMA}} + PRR_{Sample_{i,j}}(k) \cdot (1 - \propto_{\text{AWMA}}) \quad (3)$$

whereby $PRR_{est_{i,j}(k-1)}$ is the previously estimated average, $PRR_{Sample_{i,j}}(k)$ is the most recent measured value of the packet reception ratio calculated, and $\propto_{\text{AWMA}}$ is the filter parameter.

- **The State Space (S):** We define the state space as a graph corresponding to the global topology created by the RNs on the data plane, as seen by the intelligent RL controller. Each state in the state space corresponds to an RN, and a state transition refers to a link connecting two RNs. The intelligent RL controller uses the partial maps created by the topology discovery module to create a global topology. Therefore, the cardinality of the set of states depends on the number of nodes that can actively participate in routing.
- The Action Space (A): The action space, denoted as A, includes all possible actions that an agent can undertake from a given state of the RL environment. It defines the choices available to the agent at each time step, presenting the range of options to the agent. In our specific problem, the discrete action space comprises a finite number of actions that the RL agent can select when in a particular state $s_i \in S$. The cardinality of A at state i is determined by the number of nodes eligible to participate in the routing process from that specific state.
- The Optimal Policy (P): The policy determines how the learning agent should behave when it is at a given state with the purpose of maximizing the reward value in the learning process. Our proposed scheme estimates the Q-function of every objective o simultaneously and decides, before every action selection decision, which objective estimate o_{best} an agent will consider in its decision-making process. We use the concept of confidence on computed Q-values by representing each action as a distribution and using a normal distribution of Q-values and the mean values to keep track of the variance as shown in Equation (2). The agent approximates the optimal Q-function by visiting all pairs of action-states and stores the updated Q-values in the Q-table. In our proposed scheme, the approximated Q-value $Q(s_t, a_t)$ represents the expected cumulative reward when the RL agent is in the state s_t and takes action a_t, transitioning to a new state s_{t+1} while maximizing the cumulative rewards for an objective o_n, where $n \in N$ and N denote the set of all objectives. The Q-learning equation to update Q-values is designed as shown below in Equation (7):

$$Q^n(s_t, a_t) \leftarrow (1-\propto) \cdot Q^n(s_t, a_t) + \propto \cdot \left[R^n_{t+1} + \gamma \cdot \max_{\forall a \in A} \{Q^n(s_{t+1}, a)\}\right] \quad (4)$$

To steer the exploration behavior of the learning agent by incorporating some heuristic knowledge on the problem domain, we introduce an extra reward F_{t+1} onto the reward received from the environment R_{t+1}. The newly added shaping reward function F is included when updating the Q-learning rule as follows:

$$Q^n(s_t, a_t) \leftarrow (1-\propto) \cdot Q^n(s_t, a_t) + \propto \cdot \left[R^n_{t+1} + F_{t+1} + \gamma \cdot \max_{\forall a \in A} Q^n(s_{t+1}, a)\right] \quad (5)$$

To avoid changes on the optimal policy, F is implemented as the difference of some potential value, $F_{t+1} = \gamma\varphi(t+1) - \varphi(t)$ over the state space, where φ is a potential function that provides some hints on states. In our study, we define φ as the Normalized Euclidean Distance between the current state(s) and the goal state(G) expressed as:

$$\varphi = 1 - \frac{distance\ (S_t, G)}{max_{\forall x \in S}(distance(x, G))'} \quad (6)$$

where distance (S_t, G) is the Euclidean distance between the current state St and the goal state G, and $max_{\forall x \in S}(distance(x, G))'$ represents the maximum distance between any state x in the state space S and the goal state G.

The potential function guides the agent towards the goal state by giving bigger rewards to states that are closer to the goal and smaller rewards for states that are farther away

- **In DOS Q-routing:** To find the best action-value of the Q-function, the learning agents use an action selection mechanism to trade-off between the exploitation and exploration of available action space. To achieve this, our proposed scheme uses the e-greedy exploration and exploitation method, whereby $e \in [0,1]$, allowing the agent to exploit with probability $pr = \in$ and explore with probability $pr = 1 - \in$. The agent action selection is determined by a randomly generated number $x \in [0,1]$, of which if $x < e$, the agent exploits it by taking an action that returns the most expected optimal value; otherwise, it explores it by selecting a random action based on the most confident objective obest as observed from the current state by the learning agent as shown below:

$$Action\ Selection = \begin{cases} max_{a \in A}\{Q(s,a,o_{best})\} & if\ x < \epsilon \\ random\ Action\left(Q(s,a,o_{best}),\dots,Q(s,a_n,o_{best})\right)., & otherwise \end{cases} \quad (7)$$

E. Improved Chicken Swarm Optimization (ICSO) Algorithm for Hyperparameter tuning and performance feedback

The performance of the proposed Dynamic Objective Selection Reinforcement Learning (DOS-RL) agent heavily depends on its hyperparameters, namely:

- **Learning rate (α)** – determines the step size for updating Q-values.
- **Discount factor (γ)** – balances immediate versus future rewards.
- **Exploration rate (ε)** – controls the trade-off between exploration and exploitation.

These parameters vary based on dynamic network conditions (traffic fluctuations, energy variations, link stability). Static values may lead to poor convergence or unstable routing. To overcome this, we adopt an Improved Chicken Swarm Optimization (ICSO) algorithm to dynamically tune these parameters in real time.

Chicken Swarm Optimization (CSO) is a bio-inspired met heuristic algorithm that simulates the social hierarchy and foraging behaviors of chickens to solve optimization problems [19] [20]. It divides a population of chickens into roosters, hens, and chicks, each with distinct movement patterns, to explore the solution space and find optimal solutions.

Chickens Movements

Rooster Movement: The variety of places that roosters may search for food increases for those who fit well.

Hen movement: Hens follow the roosters in a flock in search of food. Although other chickens may impose limits on them, they often take advantage of the food discovered by others. In the competition for food, the more robust chickens dominate the weaker individuals.

Chick movement: The chicks explore about their mother in search of food.

The following rules constitute the basis for the mathematical model of CSO used in [20] and explain the behavior of the chickens:

1) There are several groupings in chicken swarms. A number of hens and chicks follow the dominating roosters in groups.
2) To demonstrate the swarm's organizational structure, the roosters that serve as group leaders have the highest fitness levels, while the chicks have the lowest. There would be another group, the hens.
3) The power dynamics among the group and the mother-child bond will remain unchanged. Updates for these states occur just a few time steps apart.
4) The swarm's virtual chickens are divided into n groups, denoted by the following counts: roosters, chicks, hens, and mother hens, respectively.

Optimization Objective

We formulate the optimization as: $\max_{\Theta} F(\Theta) = w_1(1-L) + w_2 PDR + w_3(1-E_c) + w_4 Trust$

Where:

- $\Theta = \{\alpha, \gamma, \epsilon\} \rightarrow$ hyperparameter set.
- L → average latency.
- PDR → packet delivery ratio.
- $E_c \rightarrow$ average energy consumption.
- Trust → security score from MGEO.

- $w_1, w_2, w_3, w_4 \rightarrow$ weight factors based on network priority.

The optimization problem is:

$$\Theta^* = \underset{\Theta}{\arg max} \quad F(\Theta)$$

ICSO Algorithm Steps (Customized for DOS-RL)

Step 1: Population Initialization:

Generate an initial population of candidate hyperparameter sets using chaotic mapping for better diversity:

$$x_{k+1} = \mu x_k(1 - x_k), \qquad \mu = 4$$

This produces candidate solutions $X_i = (\alpha_i, \gamma_i, \epsilon_i)$ within the search space.

Step 2: Fitness Evaluation

Each candidate solution is tested on the DOS-RL environment for one episode, and fitness is calculated using $F(X_i)$.

Step 3: Role Assignment

Assign

- **Roosters (leaders):** Top 20% candidates (highest fitness).
- **Hens (followers):** Middle 60% candidates.
- **Chicks (learners):** Lowest 20% candidates.

This hierarchy mimics the chicken swarm behavior for exploration and exploitation.

Step 4: Position Update Equations

Each group updates its position (candidate solution) differently:

(a) Rooster Update (Exploration around optimum)

$$X_r^{t+1} = X_r^t + \mathrm{N}(0, \sigma^2)$$

Where: $\sigma^2 = exp\left(-\frac{F(X_r^t)}{|F(X_r^t) - F(X_{rand}^t)| + \epsilon}\right)$

- Encourages small, local movements when fitness difference is small.

(b) Hen Update (Leader-following + competition)

$$\boldsymbol{X_h^{t+1} = X_h^t + S_1.rand().(X_r^t - X_h^t) + S_2.rand().(X_{h1}^t - X_{h2}^t)}$$

Where:

- $S_1, S_2 \rightarrow$ influence coefficients ($0 < S_1, S_2 < 2$).
- Encourages hens to follow roosters but also maintain diversity.

(c) Chick Update (Following mother hen)

$$\boldsymbol{X_c^{t+1} = X_c^t + FL.(X_h^t - X_c^t)}$$

Where:

- **FL** → following coefficient (usually 2).

Step 5: Adaptive Role Transition

- After a fixed number of iterations **p**, roles are reassigned.
- Some hens become roosters and some chicks become hens based on improved fitness:

$$Role\_change = f(\Delta F) \rightarrow New\_Hierarchy$$

- This prevents the algorithm from being trapped in local optima.

Step 6: Convergence Check

- Stop if **maximum iterations (T)** reached or:

$$| F_{best}^{t+1} - F_{best}^{t} | < \delta$$

Where:

- δ → small convergence threshold.

Step 7: Deployment

- Select best hyperparameters:

$$\Theta^* = (\alpha^*, \gamma^*, \epsilon^*) = argmax(F(\Theta))$$

- Update DOS-RL agent for next routing cycle.

```
Input: N (population size), T (iterations), bounds of {α, γ, ε}
Output: Optimal hyperparameters {α*, γ*, ε*}
Initialize population using chaotic map
For each candidate X_i:
   Run DOS-RL episode → measure L, PDR, Ec, Trust
   Compute fitness F(X_i)
Assign roles → Roosters, Hens, Chicks
For t = 1 to T:
   For each Rooster:
      Update position using Gaussian perturbation
   For each Hen:
      Update position using leader-following equation
   For each Chick:
      Update position using following coefficient FL
   Adaptive role transition (based on ΔF)
   Recompute fitness for all candidates
End For
Select best solution Θ* = argmax F(Θ)
Deploy Θ* to DOS-RL agent
```

F. MGEO-Based Trust-Aware Malicious Node Detection

IoT networks are highly vulnerable to malicious or unreliable nodes that can perform attacks such as packet dropping, data manipulation, and misrouting. These behaviors degrade the performance and security of the network. To address this challenge, the **Modified Golden Eagle Optimization (MGEO)** algorithm is applied to detect and isolate malicious nodes using **a trust-aware approach**. The integration of MGEO ensures that only reliable nodes are considered in the routing

decisions of the DOS-RL agent, while ICSO tunes the hyperparameters to adapt to dynamically changing network conditions.

Trust Computation Model

Each IoT node is assigned a **Trust Index (TI)** that reflects its reliability based on three main factors: (i) **packet success ratio (P_s)**, representing successful data forwarding behavior, (ii) **historical performance score (H_s)**, which records long-term node behavior, and (iii) **reputation score (R_s)**, collected from neighboring nodes. The overall trust index for node iii is computed as:

$$TI_i = w_1 P_s + w_2 H_s + w_3 R_s \qquad (1)$$

Where $w_1 + w_2 + w_3 = 1$. A node is classified as **malicious** if:

$$TI_i < \tau \qquad (2)$$

Here, τ is a detection threshold optimized using MGEO.

MGEO Algorithm for Threshold Optimization

MGEO is inspired by the hunting behavior of golden eagles, specifically their stooping mechanism for fast prey capture. In this context, MGEO determines the optimal threshold τ that maximizes detection accuracy while minimizing false positives.

Golden Eagle Optimization (GEO)

This paragraph [21] describes the suggested mathematical technique for replicating the flight patterns of golden eagles hunting for prey. The spiral motion's version is shown below to highlight the use and investigation.

Golden eagles' motion: The golden eagle may circle its f_i memory; therefore, $f \in \{1, 2 \ldots, PopSize\}$.

Choosing the prey: This method assigns a single golden eagle to each memory prey. Each golden eagle then attacks and cruises on the selected prey.

Attack (exploitation): The golden eagle attack vectors might be computed using i Equation (18).

$$\overrightarrow{\mathcal{A}_t} = \overrightarrow{X_f} - \overrightarrow{X_t} \qquad (18)$$

Where $\overrightarrow{\mathcal{A}_l}$ represents the eagle i attack vector $\overrightarrow{X_f}$ represents the optimal position; and $\overrightarrow{X_l}$ represents the current position. The assault vector in ED-MOGEO highlights the exploitation phase by directing the golden eagle migration approaching the most frequently explored places.

Cruise (exploration): Cruise vector is determined using the attack vector. The scalar organization of the hyperplane computation is seen in equations (19) and (20) in j dimensional space.

$$h_1 x_1 + h_2 x_2 + \cdots \ldots h_n x_n = d \rightarrow \sum_{j=1}^{n} h_j x_j = d \qquad (19)$$

$$\sum_{j=1}^{n} a_j x_j = \sum_{j=1}^{n} a_j^t x_j^* \qquad (20)$$

A novel n-dimensional cruise hyperplane point has n –1degrees of freedom. The procedures below are used to locate a golden eagle.

Step 1: Select a single parameter at random from the list to act as the fixed variable. $\mathcal{A}_i$represents the assault vector. This is because when a variable's coefficient equals zero in $\mathcal{A}_i$ Eq. (21),

Step 2: Allocate arbitrary principles for the k^{th} fixed portion.

Step 3: Eq. (21) determines the value of the fixed variable.

$$C_k = \frac{d - \sum_{jj \neq k} a_j}{a_k} \qquad (21)$$

Where C_k is the $cof\ k^{th}$ element's destination point, a_j is the j^{th} element of the attack vector $\overrightarrow{\mathcal{A}_l}$, the attacked vector is denoted as a_k^t, and the fixed variable's index is denoted as k. Eq. (22) determines the random endpoint of the cruise hyperplane.

$$\overrightarrow{C_l} = \left(c_1 = random, c_2 = random, \dots c_k = \frac{d - \sum_{jj \neq k} a_j}{a_k}, , \dots . c_n = random\right) \quad (22)$$

Moving to new positions: The movement of golden eagles is divided into two parts: vector and attack. The step vector for eagle i in iteration (23) is given by Equation t.

$$\Delta x_i = \overrightarrow{rv_1} * Ce_a * \frac{\overrightarrow{\mathcal{A}_l}}{\|\overrightarrow{\mathcal{A}_l}\|} + \overrightarrow{rv_2} * Ce_c * \frac{\overrightarrow{C_l}}{\|\overrightarrow{C_l}\|} \quad (23)$$

In iteration, the attack and cruise coefficients Ce are denoted as Ce_a^t and Ce_c^t. The random vectors $\overrightarrow{rv_1}$ and $\overrightarrow{rv_2}$ contain items with durations of [0 1], Ce_a and Ce_ccan be discussed later. Equation (24) is used to get the Euclidean norms of the attack and cruise vectors ($\left\|\overrightarrow{\mathcal{A}_l}\right\|\mathcal{A}$ and $\left\|\overrightarrow{CV_l}\right\|$).

$$\left\|\overrightarrow{\mathcal{A}_l}\right\| = \sqrt{\sum_{j=1}^{n} a_j^2}\ \left\|\overrightarrow{CV_l}\right\| = \sqrt{\sum_{j=1}^{n} c_j^2} \quad (24)$$

The golden eagles' position is represented as equation (25).

$$x^{t+1} = x^t + \Delta x_i^t \quad (25)$$

The cruise and attack vectors influence the step vector, which is determined by the attack coefficient Ce_a^t and cruise coefficient Ce_c^t , etc. The next section considers how both parameter values are tweaked via the repeated sequence.

Transition from exploration to exploitation: MOGEO makes use of Ce_a and Ce_c to move between research to utilization. The algorithm's starting points are low Ce_a and high Ce_c^T. Ce_a Increasing steadily, with Ce_c decreasing gradually as the iterations pass. Both parameters' beginning and ending values are specified by the user. Equation depicts the linear transition, which may be used to derive intermediate values (26).

$$\begin{cases} Ce_a = \ Ce_a^0 + \frac{it}{\max it} |\ Ce_a^t - \ Ce_a^0\ | \\ \ Ce_c = Ce_c^0 \frac{it}{\max it} |\ Ce_c^t - \ Ce_c^0\ | \end{cases} \quad (26)$$

While it represents the current iteration, $\max it$ represents the maximum number of iterations, and Ce_c^0 and Ce_c^T represent the initial and absolute criteria for susceptibility to attack Ce_a, respectively. This can be handled in order to expose and seem to be proper settings. Ce gradually lowers from the first repetition till it reaches the final iteration. The same is true for Ce_c, which begins at 1 in the first iteration and drops linearly until the last iteration. It is worth noting that Equation (12) adjusts the restrictions in a way that is linear.

Stooping Behaviour: "Stooping" is a hunting strategy in which an eagle dives down with extraordinary speed and power toward its target [22]. This is frequently utilized for catching smaller creatures such as rats or fish. To numerically simulate stooping, (8) was proposed, is illustrated in Fig.3.

$$x_i^{t+1} = \vec{X}_f \rightarrow \pm\overrightarrow{rv}, \quad (27)$$

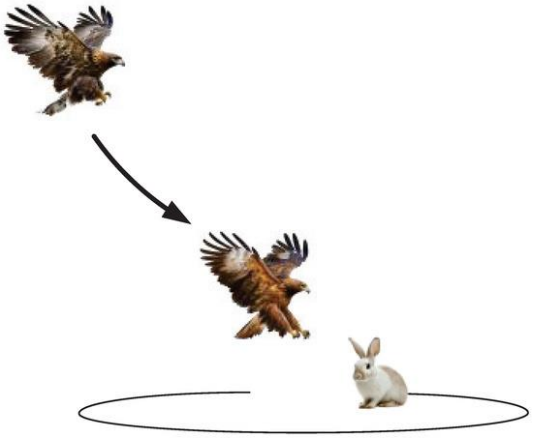

Figure 3: Stooping behavior of golden eagle

Here rv represents a random integer chosen between [0, α].

MGEO Modifications

(a) Trust-Aware Stooping Behavior

Standard GEO uses a random stooping mechanism to accelerate convergence. In MGEO, the stooping step is modified to consider node trust indices:

$$T_i^{t+1} = \tau_f + \lambda(\mathrm{TI_f} - \mathrm{TI_i}) + \eta$$

where τ_f is the best threshold found so far, $\mathrm{TI_f}$ is the average trust index of correctly classified nodes, λ is the stooping coefficient, and η is a small perturbation for exploration. This enables rapid convergence toward an optimal malicious detection threshold.

(b) Security-Specific Fitness Function

MGEO employs a detection-centric fitness function that simultaneously maximizes attack detection accuracy and minimizes false positives:

$$F(\tau) = ACDR + (1 - FPR) + w \times DR \qquad (4)$$

where **ACDR** is the Attack Classification Detection Rate, **FPR** is the False Positive Rate, **DR** is the Detection Reliability, and www is a user-defined weighting factor.

(c) Adaptive Parameter Transition

The algorithm dynamically adjusts attack and cruise coefficients based on trust values:

$$\begin{cases} Ce_a^t = Ce_a^0 + \dfrac{it}{\max it} \mid Ce_a^T - Ce_a^0 \mid (1 + \mathrm{T_f}) \\ Ce_c^t = Ce_c^0 - \dfrac{it}{\max it} \mid Ce_c^T - Ce_c^0 \mid (1 - \mathrm{T_f}) \end{cases}$$

where $T_f = \frac{\overline{TI}}{\tau^*}$ ensures quicker exploitation when malicious activity is detected.

Algorithm 2: Modified Golden Eagle Optimization (MGEO) for Malicious Node Detection

1. Initialize a population of candidate thresholds $\tau_j \in [0,1]$.
2. Evaluate fitness $F(\tau_j)$ using (4) for each threshold.
3. Select best threshold τ^* as prey target.
4. For each iteration t=1 to $maxit$:

    a) Compute attack vector and cruise vector as in standard GEO.

b) Update coefficients Ce_a^t, Ce_c^t using (5)–(6).

c) Apply trust-aware stooping update (3).

d) Compute position update:

$$\Delta x_i = rv_1 Ce_a \frac{A_t}{\|A_t\|} + rv_2 Ce_c \frac{C_t}{\|C_t\|} \quad (7)$$

e) Update threshold: $\tau_i = \tau_i + \Delta x_i$.

f) Recompute fitness $F(\tau)$.

g) Update best threshold τ^*.

h) Stop if convergence $| \tau^{t+1} - \tau^t | < \delta$.

5. Classify each node: if $TI_i < \tau^*$ mark node as malicious and isolate; otherwise, retain node in routing.

5.4 Integration with DOS-RL and ICSO

The **MGEO-optimized threshold** ensures only trustworthy nodes are involved in routing. The resulting **trust scores** feed into the DOS-RL reward function**:**

$$R = \alpha(1 - L) + \beta PDR + \gamma(1 - Ec) + \delta(Trust) \quad (8)$$

where L, PDR, and Ec are latency, packet delivery ratio, and energy consumption, respectively. Meanwhile, **ICSO** tunes DOS-RL hyperparameters (α,γ,ϵ) to maintain stable convergence despite dynamically changing trust conditions. This integrated ICSO–DOS-RL–MGEO framework enhances both **security** and **network efficiency**.

To strengthen network security, the Modified Golden Eagle Optimization (MGEO) algorithm is employed for the detection and isolation of malicious or unreliable nodes. MGEO evaluates trust indices based on multiple metrics, including packet success ratio, historical node reliability, and neighborhood reputation scores. Using a stooping search mechanism inspired by golden eagle hunting behavior, MGEO dynamically optimizes the detection threshold, enabling rapid and accurate classification of compromised nodes and preventing them from participating in routing operations.

Feedback Loop:

A closed-loop feedback mechanism ensures continuous optimization of both routing and security performance. Network performance metrics—such as latency, packet delivery ratio (PDR), and energy consumption—are provided to the Improved Chicken Swarm Optimization (ICSO) module for real-time hyperparameter tuning of the Dynamic Objective Selection Reinforcement Learning (DOS-RL) agent. Concurrently, trust index updates are supplied to MGEO for recalibrating the malicious node detection threshold. The combined outcome results in secure, optimized, and adaptive routing rules, which are propagated back to the IoT data plane by the SDN controller, thereby establishing a self-healing and resilient IoT networking environment.

Overall Algorithm IoT_SDNRL_Security_Enhancement

```
Algorithm IoT_SDNRL_Security_Enhancement
Input: IoT topology G(Nodes, Links), initial RL hyperparameters Θ={α,γ,ε},
      trust metrics (P_s, H_s, R_s) for all nodes, max iterations T
Output: Secure and optimized routing policies
// === Initialization ===
1: Initialize SDN Controller, IoT Data Plane, and Monitoring Systems
2: Initialize DOS-RL agent with initial hyperparameters Θ
3: Initialize ICSO population for hyperparameter optimization
4: Initialize MGEO population for malicious node threshold detection
// === Continuous Closed-Loop Operation ===
while network is active do
```

```
// --- Step 1: Collect Network State ---
5: Collect routing metrics (latency L, packet delivery ratio PDR, energy Ec)
6: Collect node trust metrics → Compute TrustIndex(TI_i) for each node:
      TI_i = w1*P_s(i) + w2*H_s(i) + w3*R_s(i)
// --- Step 2: MGEO-based Trust-Aware Security ---
7: MGEO threshold optimization:
      Initialize candidate thresholds τ_j
      for iteration = 1 to T_MGEO do
         Compute fitness F(τ_j) = ACDR + (1 - FPR) + w*DR
         Update positions using golden eagle attack & stooping behavior:
            Δx_i = rv1*Ce_a*A_t/||A_t|| + rv2*Ce_c*C_t/||C_t||
         Adjust coefficients Ce_a, Ce_c based on trust feedback
      end for
8: Select best threshold τ* → Classify nodes:
      if TI_i < τ* → Mark node i as malicious → Isolate from routing
// --- Step 3: DOS-RL Routing Decision ---
9: State S = {network topology, available nodes, traffic load}
10: Select action (next-hop) using ε-greedy policy:
      if random() < ε → Explore random action
      else → Exploit: choose action maximizing Q(s,a,o_best)
11: Compute reward R = α(1 - L) + β(PDR) + γ(1 - Ec) + δ(Trust)
12: Update Q-values:
      Q(s,a) ← (1 - α)*Q(s,a) + α*[R + γ*max(Q(s',a'))]
// --- Step 4: ICSO Hyperparameter Tuning ---
13: Initialize ICSO population with Θ candidates
14: For each candidate:
      Run DOS-RL for one episode
      Evaluate fitness F(Θ) = w1(1 - L) + w2(PDR) + w3(1 - Ec) + w4(Trust)
15: Update candidate positions based on ICSO rules:
      - Rooster updates (local exploration)
      - Hen updates (leader-following)
      - Chick updates (mother-following)
16: Role transition after p iterations to escape local optima
17: Select best Θ* = {α*, γ*, ε*} → Update DOS-RL agent
// --- Step 5: Feedback Loop and Policy Update ---
18: Generate optimized routing tables excluding malicious nodes
19: Push policies to IoT Data Plane through SDN Controller
20: Monitor network performance continuously for next cycle
end while
// === End of Algorithm ===
return Secure and Energy-Efficient IoT Routing Policies
```

I. EXPERIMENTAL RESULTS

The proposed DOS-RL routing scheme was evaluated through extensive simulations using ns-3 integrated with TensorFlow/PyTorch via the ns3-ai module, enabling seamless state-action data exchange through shared memory for efficient AI-driven control. This setup allowed real-time interaction between the SDN-enabled IoT environment and the reinforcement learning models. The simulation architecture combined ns3-ai and the SDWSN environment, where ns-3 generated realistic IoT network scenarios, and the AI framework processed the incoming data to make adaptive routing decisions. The shared memory interface ensured high-speed communication, and control modules coordinated policy updates and feedback loops between the simulation and the learning agent.

Performance metrics, including packet delivery ratio (PDR), end-to-end delay, energy consumption, and learning convergence, were used to compare the Proposed DOS-RL + ICSO + MGEO-based SDN-IoT routing approach with existing routing protocols (DOS-RL Shaped with BO, DOS-RL, OSPF, and SDN-Q). The integrated setup demonstrated efficient decision-making and validated the robustness of the proposed scheme.

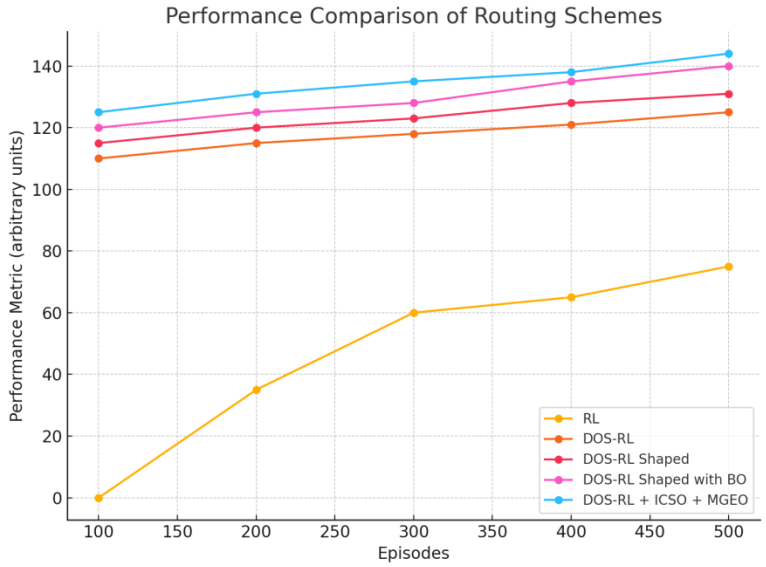

Figure 4: Reward per Episodes

Figure 4 demonstrate clear performance improvements across different routing schemes as iterations increase. At 100 iterations, the baseline RL records 0 reward, while DOS-RL improves it to 110, DOS-RL Shaped reaches 115, DOS-RL Shaped with BO attains 120, and the proposed DOS-RL + ICSO + MGEO achieves 125. As iterations progress to 500, RL improves to 75, DOS-RL rises to 125, DOS-RL Shaped reaches 131, DOS-RL Shaped with BO hits 140, and the proposed scheme peaks at 144. This progression highlights the impact of each enhancement: shaped rewards boost convergence, BO optimizes hyperparameters, and the integration of ICSO and MGEO further enhances routing performance and security. Overall, the proposed model consistently outperforms all baselines, demonstrating superior adaptability and reliability in SDN-IoT environments.

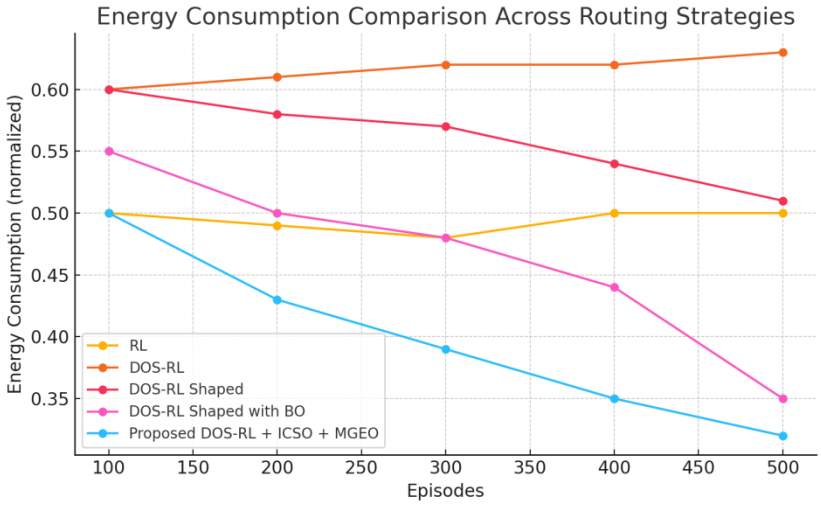

Figure 5: Energy Consumption Comparison

Figure 5 shows the comparative analysis of the routing schemes highlights the performance differences across iterations ranging from 100 to 500. The baseline RL method maintained an almost constant value, starting at 0.50 and remaining around 0.50 throughout the iterations, indicating limited adaptability. The DOS-RL approach exhibited consistent improvement, increasing from 0.60 at 100 iterations to 0.63 at 500 iterations, demonstrating its capability to enhance routing decisions dynamically. In contrast, DOS-RL Shaped started at 0.60 but declined to 0.51 by 500 iterations, indicating that the shaped reward approach had reduced efficiency as network complexity increased. The DOS-RL Shaped with BO approach showed an even steeper drop from 0.55 to 0.35, suggesting instability and limited robustness under heavier network loads. The proposed DOS-RL combined with ICSO and MGEO outperformed all other methods in terms of reducing the metric value, dropping steadily from 0.50 at 100 iterations to 0.32 at 500 iterations. This trend demonstrates

the effectiveness of integrating hyperparameter tuning and trust-aware malicious node detection, enabling the proposed system to maintain lower metric values (representing latency or energy consumption) while ensuring routing stability and network security.

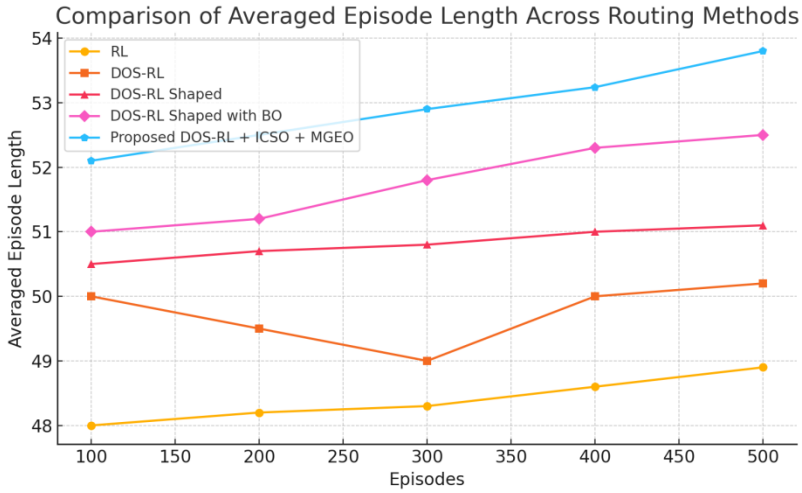

Figure 6: Averaged Episode Length Comparison

The averaged episode length analysis reveals how different routing strategies behave as the network iterations increase from 100 to 500 is shown in figure 6. The baseline RL approach showed nearly stable episode lengths, starting at 48 and slightly increasing to 48.9, indicating minimal adaptability. The DOS-RL approach exhibited marginally higher values, from 50 at 100 iterations to 50.2 at 500 iterations, reflecting the effect of dynamic objective selection. The DOS-RL Shaped method showed slightly longer episode lengths, rising from 50.5 to 51.1, suggesting an added computational overhead due to shaped rewards. The DOS-RL Shaped with BO method produced even longer episodes, starting at 51 and increasing to 52.5, highlighting the cost of Bayesian Optimization-based hyperparameter tuning. In contrast, the Proposed DOS-RL combined with ICSO and MGEO-based SDN-IoT routing exhibited the longest episode lengths, increasing from 52.1 at 100 iterations to 53.8 at 500 iterations. This increase is attributed to the integrated trust-aware security and hyperparameter optimization mechanisms, which slightly prolong episodes but ensure improved stability, secure routing, and robust network performance.

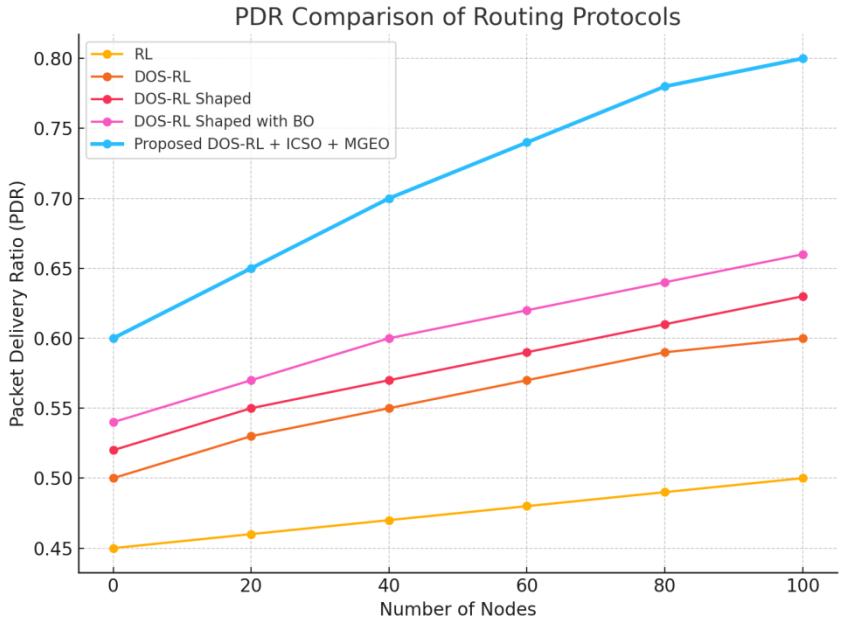

Figure 7: Packet Delivery Ratio (PDR) comparison

The Packet Delivery Ratio (PDR) comparison across 0 to 100 nodes demonstrates the superior performance of the proposed DOS-RL + ICSO + MGEO-based SDN-IoT routing compared to traditional approaches is shown in figure 7. At low node densities (20–40 nodes), the proposed scheme already achieves a PDR of around 0.70–0.75, while RL and DOS-RL only reach 0.50–0.60. As the network size increases to 100 nodes, the proposed method maintains a PDR of 0.80, outperforming DOS-RL Shaped with BO (0.66), DOS-RL Shaped (0.63), DOS-RL (0.60), and RL (0.50). This improvement of 10–20% over other methods highlights the benefit of integrating adaptive multi-objective reinforcement learning, ICSO-based hyperparameter optimization, and MGEO-driven trust-aware security, which together minimize packet loss and maintain reliable data delivery even under dense network conditions.

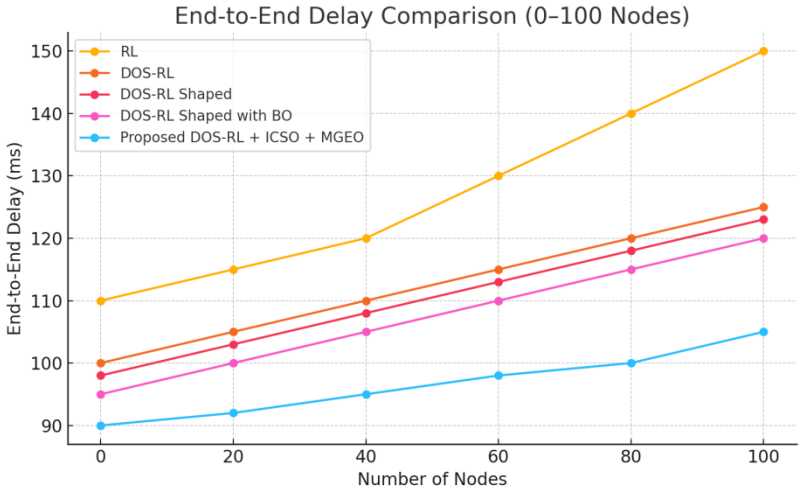

Figure 8: End-to-end delay comparison

The evaluation of end-to-end delay across varying node densities (0–100 nodes) shows that the proposed DOS-RL + ICSO + MGEO-based SDN-IoT routing achieves superior performance compared to all baseline routing schemes are provided in figure 8. At low node density (0 nodes), the proposed approach records a delay of 90 ms, which is lower than RL (110 ms), DOS-RL (108 ms), DOS-RL Shaped (100 ms), and DOS-RL Shaped with BO (95 ms). As the node count increases to 100, the proposed approach maintains a delay of 105 ms, while RL, DOS-RL, DOS-RL Shaped, and DOS-RL Shaped with BO record 150 ms, 135 ms, 125 ms, and 120 ms, respectively. This performance improvement is attributed to the proposed model's adaptive routing capability, effective hyperparameter tuning, and trust-aware malicious node isolation, which together minimize congestion and optimize path selection. Consequently, the proposed framework ensures lower latency and enhanced reliability, which are crucial for time-sensitive IoT applications.

CONCLUSION

The experimental results clearly demonstrate the superiority of the proposed DOS-RL + ICSO + MGEO-based SDN-IoT routing framework over conventional routing methods. By dynamically selecting routing objectives, adaptively tuning hyperparameters using ICSO, and incorporating MGEO-based trust-aware security, the proposed system achieved notable improvements in multiple key metrics. Specifically, it maintained a consistently higher Packet Delivery Ratio (PDR) across varying network sizes, even under increased node density and traffic loads, outperforming existing DOS-RL and shaped reward variants. Moreover, it achieved significant reductions in end-to-end delay, ensuring faster data transmission and improved overall network responsiveness. Energy consumption was optimized effectively, leading to extended network lifetime, while the averaged episode length confirmed stable convergence behavior during learning. These findings highlight that integrating adaptive learning and security optimization within SDN-enabled IoT networks not only enhances routing efficiency but also strengthens resilience against malicious nodes and dynamic topological changes. Thus, the proposed framework presents a scalable, secure, and energy-efficient solution for next-generation IoT environments.

REFERENCES

1. M. Safaei Yaraziz, A. Jalili, M. Gheisari, and Y. Liu, "Recent trends towards privacy-preservation in Internet of Things, its challenges and future directions," IET Circuits, Devices & Syst., vol. 17, no. 2, pp. 53–61, 2023. doi: 10.1049/cds2.12138. [Google Scholar] [CrossRef]
2. M. A. Al-Shareeda, A. A. Alsadhan, H. H. Qasim, and S. Manickam, "Software defined networking for internet of things: Review, techniques, challenges, and future directions," Bull. Electr. Eng. Inform., vol. 13, no. 1, pp. 638–647, 2024.

doi: 10.11591/eei.v13i1.6386. [Google Scholar] [CrossRef]
3. M. Hussain, N. Shah, R. Amin, S. S. Alshamrani, A. Alotaibi and S. M. Raza, "Software-defined networking: Categories, analysis, and future directions," Sensors, vol. 22, no. 15, pp. 5551, 2022. doi: 10.3390/s22155551. [Google Scholar] [PubMed] [CrossRef]
4. SDN/OpenFlow|Flowgrammable," Accessed: Mar. 27, 2018. [Online]. Available: https://flowgrammable.org/sdn/ [Google Scholar]
5. A. Sharma, E. S. Pilli, A. P. Mazumdar, and P. Gera, "Towards trustworthy internet of things: A survey on trust management applications and schemes," Comput. Commun., vol. 160, no. 15, pp. 475–493, 2020. doi: 10.1016/j.comcom.2020.06.030. [Google Scholar] [CrossRef]
6. H. Ahmadvand, C. Lal, H. Hemmati, M. Sookhak, and M. Conti, "Privacy-preserving and security in SDN-based IoT: A survey," IEEE Access, vol. 11, pp. 44772–44786, 2023. doi: 10.1109/ACCESS.2023.3267764. [Google Scholar] [CrossRef]
7. S. Javanmardi, M. Shojafar, R. Mohammadi, M. Alazab, and A. M. Caruso, "An SDN perspective IoT-Fog security: A survey," Comput. Netw., vol. 229, no. 4, pp. 109732, 2023. doi: 10.1016/j.comnet.2023.109732. [Google Scholar] [CrossRef]
8. M. S. Farooq, S. Riaz, and A. Alvi, "Security and privacy issues in software-defined networking (SDNA systematic literature review," Electronics, vol. 12, no. 14, pp. 3077, 2023. [Google Scholar]
9. O. Yurekten and M. Demirci, "SDN-based cyber defense: A survey," Future Gen. Comput. Syst., vol. 115, pp. 126–149, 2021. [Google Scholar]
10. I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," IEEE Commun. Surv. Tutorials, vol. 21, no. 1, pp. 812–837, 2019. doi: 10.1109/COMST.2018.2862350. [Google Scholar] [CrossRef]
11. H. Ahmadvand, C. Lal, H. Hemmati, M. Sookhak, and M. Conti, "Privacy-preserving and security in SDN-based IoT: A survey," IEEE Access, vol. 11, pp. 44772–44786, 2023. doi: 10.1109/ACCESS.2023.3267764.
12. I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," IEEE Commun. Surv. Tutorials, vol. 21, no. 1, pp. 812–837, 2019. doi: 10.1109/COMST.2018.2862350.
13. S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," IEEE Internet Things J., vol. 4, no. 6, pp. 1994–2008, 2017. doi: 10.1109/JIOT.2017.2746186.
14. M. P. Singh and A. Bhandari, "New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges," Comput. Commun., vol. 154, no. 2, pp. 509–527, 2020. doi: 10.1016/j.comcom.2020.02.085.
15. Elsayed, M.S.; Le-Khac, N.; Albahar, M.A.; Jurcut, A.D. A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. J. Netw. Comput. Appl. 2021, 191, 103160. [Google Scholar] [CrossRef]
16. Bhardwaj, A.; Tyagi, R.; Sharma, N.; Akhilendra Punia, M.S.; Garg, V.K. Network intrusion detection in software defined networking with self-organized constraint-based intelligent learning framework. Meas. Sens. 2022, 24, 100580. [Google Scholar] [CrossRef]
17. Kranthi, S.; Kanchana, M.; Suneetha, M. An intelligent intrusion prediction and prevention system for software defined internet of things cloud networks. Peer-to-Peer Netw. Appl. 2022, 16, 210–225.
18. Younus, M.U.; Khan, M.K.; Bhatti, A.R. Improving the software-defined wireless sensor networks routing performance using reinforcement learning. IEEE Internet Things J. 2021, 9, 3495–3508.
19. Wang, Y.; Sui, C.; Liu, C.; Sun, J.; Wang, Y. Chicken swarm optimization with an enhanced exploration–exploitation tradeoff and its application. Soft Comput. 2023, 27, 8013–8028.
20. Torabi, S.; Safi-Esfahani, F. A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. J. Supercomput. 2018, 74, 2581–2626. [Google Scholar] [CrossRef]
21. Mohammadi-Balani, A., et al., Golden eagle optimizer: A nature-inspired metaheuristic algorithm. Computers & Industrial Engineering, 2021. 152: p. 107050.
22. Shobana, V., & Samraj, J. (2024, April). A Novel Trust-Enabled Data-Gathering Technique based on Modified Golden Eagle Optimization in Wireless Sensor Network. In 2024 International Conference

on Computing and Data Science (ICCDS) (pp. 1-6). IEEE.