Journal of Theoretical and Applied Information Technology

30th September 2023. Vol.101. No 18
© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCING QUALITY OF SERVICE IN INTERNET OF THINGS-BASED CLOUD WIRELESS SENSOR NETWORKS (IC-WSN) USING ROBUST FROG LEAP INSPIRED ROUTING PROTOCOL (RFLIRP)

J.JERLIN ADAIKALA SUNDARI[1], G.PREETHI[2]

[1]Assistant Professor, Department of Computer Technology,
PSG College of Arts and Science, Coimbatore – 641014

[2]Associate Professor, Department of Computer Science,
PRIST University, Thanjavur – 641014
E-mail: jerlinadaikalasundari@psgcas.ac.in, mgpreethi@gmail.com

ABSTRACT

Efficient energy management in routing algorithms for Internet of Things-based Cloud Wireless Sensor Networks (IC-WSN) is crucial for greenhouse farming applications. This paper addresses the problem of energy efficiency in routing algorithms for IC-WSN in greenhouse farming. The reliance on battery-powered wireless sensors in greenhouses necessitates the development of innovative routing protocols that can extend sensor lifespan and minimize maintenance costs. In large-scale greenhouse environments with high sensor density, inefficient routing protocols can lead to excessive energy consumption and premature battery depletion. The Robust Frog Leap Inspired Routing Protocol (RFLIRP) is proposed to address this challenge. RFLIRP intelligently selects energy-efficient paths, considering individual sensor energy levels, distance to the destination, and available alternative routes. By incorporating techniques like data aggregation and compression, RFLIRP significantly reduces energy consumption and enhances the operational life of sensors. This research aims to promote sustainability in agricultural practices by optimizing energy consumption, minimizing maintenance costs, and facilitating uninterrupted data transmission for timely monitoring in greenhouse farming. The results highlight the significance of RFLIRP in improving energy efficiency and operational effectiveness in IC-WSN routing for greenhouse farming, paving the way for sustainable and optimized agricultural practices.

KEYWORDS: *Agricultural, Efficiency, Farming, Greenhouse, Routing, Wireless*

1. INTRODUCTION

Greenhouse farming, also known as protected cultivation, revolutionizes traditional agricultural practices by providing a controlled environment for crop production [1]. Enclosed structures, typically glass or plastic, allow precise temperature, humidity, and light regulation. This creates an ideal plant ecosystem, resulting in higher crop yields, improved quality, and protection against adverse weather cond222itions. One significant benefit of greenhouse farming is its ability to extend the growing season [2]. Farmers can cultivate various plants yearly by shielding crops from external environmental fluctuations. This eliminates the limitations of seasonal changes and enables a consistent supply of fresh produce, providing stability in the market and reducing dependency on imports from distant locations [3].

Greenhouse farming also facilitates resource management. The enclosed environment helps conserve water by minimizing evaporation, and advanced irrigation systems can deliver water directly to the plant roots, reducing waste. Additionally, the controlled setting allows for precise nutrient management, resulting in efficient fertilizer use and reduced environmental impact [4]. This resource efficiency is crucial for sustainable agriculture, particularly in regions facing water scarcity and other resource limitations [5]. Greenhouse farming promotes the protection of crops against pests and diseases. The controlled environment acts as a barrier, preventing the entry of harmful insects and pathogens. This reduces the reliance on chemical pesticides, making greenhouse farming a more environmentally friendly option.

The controlled conditions also allow for early detection and prompt management of any infestations, preventing widespread crop damage and ensuring a healthier harvest [6].

Internet of Things-based Cloud Wireless Sensor Network (IC-WSN) is a groundbreaking technology that merges two transformative fields: the Internet of Things (IoT) and wireless sensor networks (WSN) [7]. This integration combines the capabilities of wireless sensors and cloud computing to create a robust and efficient system for data collection, analysis, and management. IC-WSN utilizes wireless sensors to gather data from the physical environment. These sensors, equipped with various sensing capabilities, capture information such as temperature, humidity, pressure, and motion. The collected data is transmitted through wireless communication protocols to a cloud-based platform [8], [9].The integration of cloud computing in IC-WSN brings significant advantages to the system. Cloud infrastructure offers scalable storage, computational power, and accessibility. It enables the processing and analysis of large volumes of sensor data in real time, facilitating timely decision-making and actionable insights [10]. Additionally, the cloud provides remote management capabilities,allowing users to access and monitor data from anywhere, using various devices. The IC-WSN technology has vast applications across different domains. It can be employed in environmental monitoring, industrial automation, smart cities, healthcare, agriculture, and more [11]. By leveraging the capabilities of IoT and cloud computing, IC-WSN enables efficient data collection, analysis, and communication, leading to improved operational efficiency, resource management, and informed decision-making [12].

IC-WSN technology holds immense importance in the context of greenhouse farming. Greenhouses are controlled environments designed to optimize crop growth, but effective monitoring and management are crucial. IC-WSN addresses this need by enabling real-time data collection and analysis, ensuring precise control and efficient resource utilization [13]. By deploying wireless sensors in greenhouses, IC-WSN allows continuous monitoring of vital parameters like temperature, humidity, soil moisture, and light levels. This data is transmitted to a cloud-based platform, empowering farmers to access and analyze it in real-time remotely [14]. By leveraging IC-WSN, greenhouse farmers can gain valuable insights into environmental conditions, detect anomalies, and make data-driven decisions to optimize crop growth. IC-WSN's ability to provide continuous monitoring and control in greenhouse farming is a game-changer. It enables farmers to make precise adjustments to environmental factors, such as temperature and irrigation, tailored to the specific needs of different plant varieties. This enhances crop quality, reduces resource wastage, and promotes sustainable farming practices [15].

1.1. Problem Statement

The significant problem in routing in IC-WSN for greenhouse farming is the energy efficiency of routing algorithms. Wireless sensors in the greenhouse are typically powered by batteries, which have limited capacity. Efficient energy management is crucial to prolong the lifespan of the sensors and minimize the need for frequent battery replacements. Inefficient routing protocols can result in unnecessary energy consumption, leading to premature battery depletion and increased maintenance costs. This problem becomes even more pronounced in large-scale greenhouse environments with a high density of sensors. Innovative routing algorithms must be developed to address this issue, prioritizing energy efficiency. These algorithms should consider factors such as the energy levels of individual sensors, the distance to the destination, and the available alternative routes. By intelligently selecting energy-efficient paths and employing techniques like data aggregation and compression, routing protocols can significantly reduce energy consumption and prolong the operational life of the sensors.

1.2. Motivation

Efficient energy management in routing algorithms for IC-WSN in greenhouse farming is vital for the sustainability and success of agricultural practices. By prioritizing energy efficiency, we can achieve several compelling motivations for greenhouse farming. Firstly, optimizing energy consumption extends the lifespan of wireless sensors, reducing the need for frequent battery replacements and minimizing maintenance costs. Secondly, efficient routing algorithms contribute to reduced energy consumption, promoting eco-friendly and sustainable farming practices. Additionally, by intelligently selecting energy-efficient paths, we can enhance the overall operational efficiency of the system, ensuring uninterrupted data transmission and timely monitoring of crucial

parameters. Implementing innovative routing protocols that address energy efficiency empowers greenhouse farmers to maximize productivity, reduce resource wastage, and enhance the long-term viability of their operations.

1.3. Objective

The research objective of this study is to develop energy-efficient routing algorithms for IC-WSN in greenhouse farming. The primary focus is to address the significant problem of inefficient energy consumption in routing protocols, which leads to premature battery depletion and increased maintenance costs. The specific research objectives are as follows:

- Investigate the energy dynamics of wireless sensors in greenhouse environments to understand their energy consumption patterns and limitations.
- Design and develop innovative routing algorithms that prioritize energy efficiency by considering individual sensor energy levels, distance to the destination, and available alternative routes.
- Evaluate the performance of the proposed routing algorithms through simulation or practical experimentation, comparing them with existing routing protocols to assess their energy-saving capabilities and effectiveness.
- Explore techniques such as data aggregation and compression to optimize energy consumption in routing further while ensuring reliable and timely data transmission.
- Validate the energy-efficient routing algorithms in real-world greenhouse farming scenarios, assessing their impact on prolonging the operational life of wireless sensors and minimizing maintenance efforts.

2.0 LITERATURE REVIEW

"Sector-based Random Routing Scheme" [16] aims to prevent adversaries from accurately determining the location of the data source within the network. In this, the network area is divided into sectors, and each sector contains multiple sensor nodes. When a data packet needs to be transmitted, the source node randomly selects a sector within its proximity and forwards the packet to a sensor node within that sector. This random selection process prevents adversaries from pinpointing the exact location of the data source. This scheme introduces uncertainty in the routing process, making it challenging for adversaries to trace the origin of the data packets. The sector-based random routing scheme offers advantages regarding source location privacy preservation, ensuring that the actual location of the source node remains hidden from potential attackers. "Energy and Collision Aware WSN Routing Protocol" [17] is a specialized protocol designed for Sustainable and Intelligent IoT applications. It addresses the need for efficient routing in WSNs while considering energy conservation and collision avoidance. This protocol optimizes the network's energy consumption by dynamically selecting routes that minimize energy usage. It also takes into account collision avoidance mechanisms to enhance data transmission reliability. By intelligently managing energy resources and reducing collisions, the protocol promotes sustainability and prolongs the network's operational lifetime.

"Deep Learning-Based Routing Protocol" [18] investigates the effectiveness of a routing protocol for efficient data transmission in 5G WSN communication. The protocol utilizes deep learning techniques to optimize routing decisions and enhance network performance. Through this analysis, researchers evaluate metrics such as throughput, latency, and packet delivery ratio to assess the protocol's efficiency. By leveraging the power of deep learning, the routing protocol aims to intelligently adapt to changing network conditions and improve the overall data transmission in 5G WSNs. "Two-Level Clustering and Routing Algorithms" [19] designed to prolong the lifetime of Wind Farm-Based WSNs. These algorithms utilize a two-level clustering approach to manage energy resources in the network efficiently. At the first level, clusters are formed based on the proximity of sensor nodes to a cluster head, reducing the communication overhead. At the second level, within each cluster, a routing algorithm is employed to establish energy-efficient routes for data transmission. This two-level approach optimizes energy consumption by minimizing long-distance transmissions and maximizing the utilization of available energy. By prolonging the network lifetime, these algorithms enhance the monitoring and control capabilities of Wind Farm-Based WSNs, leading to improved efficiency and reliability in wind farm operations.

"Energy-Saving Routing Protocol" [20] is designed for WSNs focusing on energy conservation. This protocol leverages Voronoi adaptive clustering techniques to achieve efficient energy utilization and prolong the network lifetime. It employs a two-step process: clustering and routing. In the clustering phase, Voronoi adaptive

clustering is utilized to form clusters, where cluster heads are selected based on their proximity to the centroid of their respective clusters. This clustering approach helps to evenly distribute the energy load transmission. "Dynamic Cluster-Based Routing Protocol" [21] is a novel approach developed for WSNs to enhance routing efficiency. This protocol combines the power of metaheuristic algorithms and dynamic cluster-based routing techniques. The metaheuristic algorithm, such as genetic algorithm or particle swarm optimization, is applied to dynamically optimize cluster formation and selection of cluster heads. By intelligently adapting to network changes, such as node failures or mobility, the protocol ensures the formation of efficient clusters. Dynamic cluster-based routing facilitates the establishment of energy-efficient routes for data transmission within the network.

"Multi-Objective ACO-based Routing" [22] is a sophisticated routing protocol designed for Wireless Sensor Networks (WSNs) that takes into account multiple objectives, including Quality of Service (QoS) and cross-layer optimization. The protocol incorporates a Multi-Objective Ant Colony Optimization (ACO) algorithm to route data packets in the network efficiently. By simultaneously considering multiple objectives, this protocol aims to balance different performance metrics, such as energy consumption, latency, and reliability. Additionally, it leverages cross-layer optimization techniques to exploit the interactions between different layers of the network protocol stack, enabling more efficient routing decisions. "Q-Learning-based Routing Protocol" [23] is a novel routing protocol designed for Wireless Sensor Networks (WSNs) that focuses on data aggregation and energy efficiency. The protocol utilizes Q-learning, a reinforcement learning algorithm, to optimize routing decisions based on the network's energy constraints and the need for data aggregation. By learning from past experiences, the protocol dynamically adapts its routing strategy to maximize energy efficiency while ensuring effective data aggregation. This approach reduces energy consumption by minimizing unnecessary data transmissions and promoting data aggregation at intermediate nodes.

"Energy-Efficient Routing Protocol" [24] is a specialized protocol designed to optimize energy consumption in three-dimensional sensor networks. This protocol addresses the unique challenges of 3D WSNs, where sensors are deployed in a three-dimensional space. The among nodes and minimize energy consumption. In the routing phase, energy-efficient paths are dynamically established from source nodes to their respective cluster heads, enabling effective data protocol employs energy-efficient routing strategies to minimize energy consumption during data transmission. It considers node distance, residual energy levels, and communication costs to select the most energy-efficient paths. "Dynamic Clustering Green Communication Routing" [25] incorporates dynamic clustering techniques and green communication strategies to optimize routing decisions in ITS environments. By dynamically forming clusters of vehicles based on proximity and communication requirements, it minimizes energy consumption and communication overhead. Additionally, it employs green communication techniques, such as power control and adaptive modulation, to further reduce energy usage and enhance network efficiency. This protocol promotes energy efficiency and enables effective communication and data transmission in intelligent transportation systems.

"Destination-Oriented Routing Algorithm (DORA)" [26] is a routing algorithm developed explicitly for Energy-Balanced WSNs. It addresses the challenge of routing data in networks comprising energy-constrained sensor nodes. DORA aims to balance energy consumption among nodes, thereby preventing premature depletion of resources and prolonging the network lifetime. By employing a destination-oriented approach, nodes make forwarding decisions based on proximity to the destination and residual energy. This algorithm dynamically selects energy-efficient paths to minimize energy consumption. It also adapts to network changes, such as node failures or mobility, to ensure reliable data delivery. "Particle Swarm Optimization Routing Scheme (PSORS)" [27] utilizes the principles of the core PSO algorithm to optimize the routing process. In this, sensor nodes are represented as particles in a multidimensional search space, where each particle's position represents a potential routing path. The PSORS mimics the social behavior of a swarm of particles, where each particle adjusts its position based on its own experience and the best position discovered by the swarm. The particles explore the search space to find optimal routing paths that minimize energy consumption, maximize network coverage, or satisfy other performance objectives. Through iterations, the particles converge towards better routing solutions. The scheme dynamically adapts to changing network conditions and node energy

levels, optimizing the routing paths accordingly. It also considers factors such as transmission distance, traffic load, and network topology.

3.0 ROBUST FROG LEAP INSPIRED ROUTING PROTOCOL (RFLIRP)

3.1. Ad-hoc On-Demand Distance Vector (AODV)

The Ad hoc On-Demand Distance Vector (AODV) routing protocol is a widely used reactive routing protocol designed for wireless ad-hoc networks. AODV dynamically establishes routes between nodes as needed, allowing efficient network routing with constantly changing topologies. AODV is a reactive protocol, and routes are established only when needed. This reduces the control overhead and conserves network resources. It also supports both unicast and multicast communication in MANETs. AODV offers an efficient and scalable solution for routing in dynamic ad hoc networks, where nodes are mobile, and network topology changes frequently. Its on-demand nature ensures that routes are established as required, leading to efficient utilization of network resources and adaptability to changing network conditions. The steps involved in AODV are:

- **Route Discovery:** When a source node wants to send data to a destination node and doesn't have a valid route, it initiates a route discovery process. The source broadcasts a Route Request (RREQ) packet to neighboring nodes. The RREQ contains the source and destination addresses and a unique sequence number.
- **Route Reply:** When a node receives an RREQ, it checks its routing table to see if it has a route to the destination. If not, it rebroadcasts the RREQ to its neighbors unless it has already processed the RREQ. The RREQ also includes a hop count metric incremented at each hop to keep track of the distance.
- **Route Establishment:** When the RREQ reaches the destination node or a node with a valid route to the destination, the receiving node generates a Route Reply (RREP) packet. The RREP contains the destination and source addresses, the sequence number, and the route metric. The RREP is then unicast back to the source node following the reverse path of the RREQ.
- **Route Maintenance:** Once the source node receives the RREP, it creates or updates a route entry in its routing table. Intermediate nodes also update their routing tables based on the RREP information. Routes are established by maintaining a next-hop entry for each destination. If a link in the route breaks, AODV uses a local repair mechanism to re-establish the route.
- **Route Expiry and Removal:** Each route entry in AODV has a lifetime associated with it. If a node does not receive any packets for a route within its lifetime, it assumes that the route is no longer valid and removes it from its routing table. AODV supports periodic route advertisement and route error mechanisms to keep the routes fresh.

Algorithm 1: AODV

Input:
Source node (S)
Destination node (D)
Data to be transmitted

Output:
Established route from S to D

Procedure:
Step 1: If S has a valid route to D in its routing table, proceed to Step 6.
Step 2: If S does not have a route to D:
  a. Create a Route Request (RREQ) packet with S as the source and D as the destination.
  b. Broadcast the RREQ to neighboring nodes.
Step 3: When a node N receives an RREQ:
  a. If N has a route to D, update the route sequence number if necessary.
  b. If N is D, create a Route Reply (RREP) packet and unicast it back to S using the reverse path of the RREQ.
  c. If N is an intermediate node, forward the RREQ to neighboring nodes.
Step 4: When the RREP reaches S:
  a. Create or update a route entry in S's routing table for D.
  b. Set the next-hop entry for D to the address from which the RREP was received.
Step 5: If S wants to send data to D, check for a valid route entry in S's routing table.
Step 6: If there is a valid route entry:
  a. Encapsulate the data into a packet and send it to the next-hop node according to the route entry.

Step 7: If there is no valid route entry, repeat steps 2-4 to establish a route from S to D.
Step 8: Data transmission is complete.

3.2. Frog Leap Inspired Algorithm

The Frog Leap Inspired Algorithm (FLIA) is a nature-inspired optimization algorithm that mimics the collaborative behavior of frogs to solve complex optimization problems. Introduced in 2003 by Eusuff and Lansey, FLIA is based on frogs exchanging information to improve their solutions collectively. With its efficient and robust approach, FLIA has gained popularity in various domains as a powerful optimization technique.

3.2.1. Adjusting settings and starting a population

During the first stage of the FLIA, various settings are adjusted, and a population of frogs is initialized. To begin, the algorithm requires setting parameters such as the number of subgroups (c) and sub-iterations (NE). These parameters are crucial in determining the behavior and efficiency of the algorithm. Once the settings are configured, the algorithm creates a population of T frogs. The frogs are generated randomly, and the problem's decision variables define their positions in the search space. These decision variables represent the parameters or variables that the algorithm seeks to optimize. After the frogs' positions are established, their fitness values are computed. The fitness value represents each frog's objective or fitness function evaluation, quantifying how well a solution satisfies the optimization criteria. It measures the quality of the frog's position within the search space.

The fitness evaluation process involves applying the objective or fitness function to each frog's position, which results in a numerical value representing its fitness. This evaluation measures how close each frog's position is to the optimal solution. A preliminary assessment of their quality is obtained by calculating the fitness values for all the frogs in the population. This assessment serves as a basis for subsequent stages of the algorithm, where the frogs will be grouped, ordered, and further refined to converge towards an optimal solution. It is important to note that the specific approach for generating the initial population of frogs and computing their fitness values may vary depending on the problem being addressed and the nature of the optimization task.

3.2.2. Clustering

In the second stage of the FLIA, the population of frogs is divided into subgroups based on their fitness values. This process involves clustering the frogs and organizing them in a specific order within each subgroup. The initial population of frogs, which was generated and evaluated in Stage 1, serves as the starting point for the clustering process. The frogs are ordered based on their fitness values, from the fittest to the most petite fit. This ordering clearly distinguishes between superior and inferior solutions within the population. To perform the clustering, the algorithm utilizes the concept of subgroups. The number of subgroups, denoted as c, is determined during the initial setup of the algorithm. Each subgroup comprises a population subset, with the frogs assigned to different subgroups based on their order.

The assignment of frogs to subgroups follows a specific pattern. The first frog, the fittest in the population, is assigned to the first subgroup. The second frog is assigned to the second subgroup, and so on, until the c-th frog is assigned to the c-th subgroup. Once the c-th subgroup is reached, the assignment process wraps around, and the next frog $(c + 1 - th)$ is assigned to the first subgroup. This pattern continues until all frogs are assigned to a subgroup. By clustering the frogs this way, the algorithm ensures that each subgroup has a diverse range of fitness levels. This diversity allows for exploring the solution space within each subgroup, including promising solutions (frogs with high fitness) and less promising ones (frogs with lower fitness). The ordering of frogs within each subgroup also provides an inherent ranking of solutions. The first frog in each subgroup represents the best solution within that subgroup, while the last frog represents the worst solution. This ranking information is essential for subsequent stages of the algorithm, as it guides the search process towards improving the quality of solutions within each subgroup. It's important to note that the number of subgroups (c) can vary depending on the problem and algorithm settings. The choice of c influences the balance between exploration and exploitation in the search process. A more significant number of subgroups allows for a more diversified exploration, while a smaller number promotes a more focused exploitation of promising solutions.

3.2.3. Intra-group searching

In the third stage of the FLIA, an intra-group search is conducted within each subgroup to refine the solutions and improve their fitness

values. This process involves iteratively updating the positions of the frogs, mainly focusing on the poorest solution and its neighboring solutions within the subgroup. Within a subgroup, the poorest solution is P_n and the best is P_v. The goal is to improve the position of P_n by iteratively updating its coordinates based on the positions of P_v and its neighboring solutions. The intra-group search involves the following equations:

(a). Step size update

$$E = rand * (P_v - P_n) \quad (1)$$

$rand$ is a random number between 0 and 1. The update step size (E) is calculated by multiplying the difference between the positions of P_v and P_n by the random number $rand$. This step size determines the magnitude and direction of the update for P_n.

(b). Updating the Position of P_n:

$$P_{n'} = P_n + E, E_{min} \leq E \leq E_{max} \quad (2)$$

The position of P_n is updated by adding the updated step size E to its current position. E_{min} and E_{max} represent the minimum and maximum allowed update step sizes, respectively. This update ensures that P_n moves towards a potentially better position.

(c). Checking Fitness and Replacing P_n:

If the fitness value of $P_{n'}$is greater than that of P_n, then $P_{n'}$ replaces P_n. Otherwise, if $P_{n'}$is not an improvement, the position of $P_{n'}$ is further modified using the following equations:

$$E = rand * (P_j - P_n) \quad (3)$$

$$P_{n'} = P_n + E, E_{min} \leq E \leq E_{max} \quad (4)$$

where P_j represents the position of the currently best solution within the subgroup. The update step size E is calculated similarly to Equation 1, using the difference between P_j and P_n multiplied by a random number rand. Equation 4 then updates the position of $P_{n'}$ using the calculated E within the allowed step size limits.

(d). Replacement with Random Frog:

If the fitness value of $P_{n'}$is still inferior to the fitness value of P_n, P_n is replaced with a new frog chosen randomly using the following equation:

$$P_{n'} = d + rand(1, Y) \otimes (v, d) \quad (5)$$

where d and v represent vectors defining the upper and lower bounds of the decision variables, respectively. Y represents the dimensionality of the optimization problem. The term rand$(1, Y)$ generates a random vector with Y components, each between 0 and 1. The $\otimes$ symbol denotes entry-wise multiplication.

The intra-group search continues by iteratively applying these steps to improve the position of P_n within the subgroup. This process is repeated until a specific termination condition or a predefined limit is reached. The ultimate objective is to enhance each subgroup's overall quality of solutions.

3.2.4. Global communications

Global data synchronization facilitates information exchange and collaboration among subgroups. This stage aims to harness the entire population's collective intelligence and enhance the algorithm's overall search capabilities. The process of Stage 4 can be further expanded as follows:

(a). Reconfiguring Subgroups:

After completing the intra-group searching in Stage 3, the algorithm combines all the frogs from the subgroups to create a unified community of T frogs. This reconfiguration step ensures that the knowledge and diversity accumulated within each subgroup are brought together to benefit the entire population.

(b). Subset Classification:

The frogs are classified into subsets once the population is restructured into a single community. This classification is typically based on their fitness values, where frogs with similar fitness levels are grouped within each subset. Subset classification promotes efficient information exchange and exploration among frogs with comparable performance levels.

(c).Intra-group searching and global information exchange:

In this, the algorithm performs a cyclic process involving intra-group searching and global information exchange. The process begins with intra-group searching, where each subset independently undergoes its local search to refine the solutions within the subset. This intra-group search is similar to the process described in Stage 3. The algorithm facilitates global information exchange after the intra-group search within each subset. This exchange allows frogs from different

subsets to share valuable information, such as the best solutions or fitness values obtained within their respective subsets. Information sharing promotes knowledge transfer and guides the search towards more promising regions of the solution space. The specific mechanisms for global information exchange can vary depending on the implementation of FLIA. It may involve inter-subgroup communication, where selected representatives or elite frogs from each subset exchange information. Alternatively, it could involve disseminating information through modifying specific algorithm parameters or adaptive strategies based on the aggregated knowledge from different subsets.

The cyclic process of intra-group searching and global information exchange continues iteratively. Each iteration allows further refining of solutions within each subset and integration of new knowledge from the global exchange. This iterative process enhances the exploration and exploitation capabilities of the algorithm, leading to the convergence towards better solutions.

Algorithm 1: FLIA

Input:
- T: Population size
- c: Number of subgroups
- NE: Number of sub-iterations for each population
- E_{min}: Minimum update step size
- E_{max}: Maximum update step size
- Other problem-specific parameters

Output:
- The best solution found during the algorithm execution.

Procedure:

Step 1: **Initialize:**
- Set algorithm parameters and initialize the population.

Step 2: Adjusting settings and starting a population:
- Generate an initial population of T frogs randomly.
- Evaluate the fitness of each frog in the population.

Step 3: Clustering:
- Order the frogs in the population based on their fitness values.
- Divide the population into c subgroups, ensuring a diverse range of fitness levels within each subgroup.

Step 4: Intra-group Searching:
- Repeat the following steps for each subgroup:
- Identify the poorest solution P_n and the best solution P_v within the subgroup.
- Perform intra-group searching for the subgroup for NE sub-iterations:
- Repeat the following steps for each sub-iteration:
- Update the step size E using Eq.(1)
- Update the position of P_n using Eq.(2)
- Check the fitness of $P_{n'}$ and replace P_n if $P_{n'}$ is better.
- If $P_{n'}$is not an improvement, modify its position using Eq.(3) and Eq.(4).
- If $P_{n'}$ is still inferior, replace P_n with a new random frog using Eq(5).
- Repeat the intra-group searching process until NE sub-iterations are completed.

Step 5: Global Communications:
- Reconfigure the subgroups into a unified population of T frogs.
- Classify the frogs into subsets based on their fitness levels.
- Repeat the following steps until a termination criterion is met:
- Perform intra-group searching within each subset.
- Facilitate global information exchange between the subsets to share knowledge and solutions.
- Update the frogs' positions and fitness values based on the collective information.
- Output the best solution obtained during the algorithm execution.

3.3. Robust Frog Leap-Inspired Routing Protocol

The Robust Frog Leap-Inspired Routing Protocol ($RFLIRP$) is an optimization-based routing protocol based on FLIA that incorporates Levy flight, a random walk process to improve the algorithm's optimization capabilities. By integrating Levy flight into the routing process, $RFLIRP$ enhances the exploration and adaptability of FLIA, enabling it to find robust routes in dynamic network environments. $RFLIRP$ offers an innovative approach to routing optimization, leveraging the natural behavior of frogs and the principles of Levy

flight to enhance the performance and resilience of routing protocols in complex and evolving network scenarios. Bio-inspired Optimization [28], [29], [38]–[41], [30]–[37] has several potentials to solve various research issues.

3.3.1. Levy flight

Levy Flight is a random walk process characterized by long, infrequent steps, also known as jumps, that follow a heavy-tailed probability distribution. It is named after the French mathematician Paul Lévy, who first introduced the concept. Levy Flight has been widely used in various fields, including optimization algorithms, for its ability to explore large search spaces more efficiently than traditional random walks. In Levy Flight, the step lengths of the random walk are generated according to a probability distribution with a heavy tail, typically following a power-law distribution. This means there is a higher probability of taking longer steps than a normal distribution. The heavy-tailed property of Levy Flight allows occasional long jumps, enabling the exploration of distant areas in the search space. The steps involved in Levy Flight are provided in Algorithm 2.

Algorithm 2: Levy Flight

Input:

- Maximum number of iterations (max_{iter})
- Step length scaling factor ($scale_{factor}$)

Output:

- Final position after the Levy Flight process ($final_{position}$)

Procedure:

Step 1: Initialize the starting position ($current_{position}$) in the search space.
Step 2: Set the iteration count ($iter_{count}$) to 0.
Step 3: Repeat the following steps until the max_{iter} is reached:
Step 4: Generate a step length ($step_{length}$) from a Levy distribution.
Step 5: Generate a random direction ($step_{direction}$) uniformly from the available directions in the search space.
Step 6: Calculate the step vector by multiplying the step length by the step direction.
Step 7: Update the current position by adding the step vector to the current position.
Step 8: Increment the $iter_{count}$ by 1.
Step 9: Store $final_{position}$ as the current position.
Step 10: Return the $final_{position}$ as the output.

3.3.2. Enhanced local and global search:

RFLIRP incorporated a novel Lévy flight update approach, which improved the algorithm's local and global search capabilities. Using Lévy flight, *RFLIRP* introduced a mix of short- and long-distance searches, enhancing the algorithm's ability to explore the solution space. Frequent short-distance searches enhanced the local search capability, enabling the algorithm to refine solutions quickly. Meanwhile, the occasional long-distance searches prevented the algorithm from getting trapped in suboptimal solutions, improving its global search capability.

(a). Local Search Enhancement:

To enhance the local search capability, *RFLIRP* utilizes short-distance searches that help refine solutions quickly. *RFLIRP* can mathematically represent this enhancement by introducing a parameter controlling the short-distance searches' step size. Let's call this parameter α. During the local search phase, when updating the position of a solution and the same is expressed as Eq.(6).

$$NewPosition = OldPosition + a * ShortDistance\ Vector \quad (6)$$

Here, ShortDistanceVector represents a vector randomly sampled from a small step size distribution, such as a Gaussian distribution with a slight standard deviation. The α parameter scales the step size, allowing for more or less exploration in the local neighborhood.

By adjusting the value of α, you can control the extent of the local search. An enormous α value would result in larger steps, allowing for more exploration of the local neighborhood. Conversely, a smaller α value would restrict the search to smaller steps, focusing on fine-tuning the solutions.

(b). Global Search Enhancement:

To enhance the global search capability, *RFLIRP* incorporates occasional long-distance searches to prevent the algorithm from getting trapped in suboptimal solutions. The Lévy flight update approach introduces a mix of short- and long-distance searches. The Lévy flight is a random process that follows a probability distribution called the Lévy distribution. This distribution has heavy tails, which means it allows for occasional long-distance jumps in the search space. To incorporate Lévy flight into the algorithm, this research can modify the position update as Eq.(7).

$$NewPosition = OldPosition + \beta * LevyFlightVector \quad (7)$$

Here, LévyFlightVector represents a vector randomly sampled from the Lévy distribution, and β controls the step size of the long-distance searches. The value of β determines the magnitude of the jumps, with larger values leading to more extensive exploration in the global search space.

Adjusting the β parameter allows you to control the balance between local and global exploration. Higher β values encourage larger jumps, increasing the algorithm's ability to escape local optima. Lower β values restrict the exploration to smaller jumps, focusing more on the local refinement. By incorporating the above modifications into the $RFLIRP$ algorithm, this research work mathematically enhance both the local and global search capabilities. The α parameter controls the step size for short-distance searches, enhancing local search, while the β parameter determines the step size for long-distance searches, enhancing global search.

Algorithm 3: Enhanced Local and Global Search

Input:
- Population size: The number of solutions in each generation.
- Maximum number of iterations: The maximum number of iterations the algorithm will run.
- α (local search step size): The step size parameter for the local search phase.
- β (global search step size): The step size parameter for the global search phase.

Output:
- The best solution found: The solution with the highest fitness value is obtained by the algorithm.

Procedure:
Step 1: Initialize the algorithm parameters: Population size, maximum number of iterations, α, and β.
Step 2: Generate an initial population of solutions randomly.
Step 3: Evaluate the fitness of each solution in the population.
Step 4: Repeat the following steps until reaching the maximum number of iterations:
Step 5: Perform the local search phase
Step 6: Perform the global search phase
Step 7: Output the best solution found.

3.3.3. Improved global search with differential mutation

The differential mutation operator is responsible for perturbing the existing solutions to generate new mutated solutions. It introduces exploration and diversification in the search process. The specific implementation of the mutation operator depends on the problem domain and the characteristics of the search space.

Let's denote the set of solutions as $S = \{S1, S2, \ldots, SN\}$, where N is the population size. Each solution Si consists of a vector of variables representing a potential solution. The mutation operator M takes S as input and returns a mutated solution, denoted as Mi. The differential mutation operator can be defined as Eq.(8).

$$Mi = Si + F * (Sa - Sb) + F * (Sc - Sd) \quad (8)$$

Here, Sa, Sb, Sc, and Sd have randomly selected solutions from S, excluding the current solution Si. F is the differential weight, a user-defined parameter that controls the amplification of the difference vectors $(Sa - Sb)$ and $(Sc - Sd)$.

To generate a mutated solution, the mutation operator combines the differences between selected solutions with the current solution, scaled by the differential weight F. This process adds diversity to the population, allowing the algorithm to explore new regions of the search space.

(a) Global Search Update Equation:

To incorporate differential mutation into the $RFLIRP$ algorithm, this research modifies the equation for updating the solution's position during the global search phase. Let's denote the old position of a solution Si as $OldPosition$ and the mutated position as $MutatedPosition$.

$$MutatedPosition = OldPosition + MutatedVector \quad (9)$$

where $MutatedVector$ represents the vector difference between the mutated solution Mi and the current solution Si. The $MutatedVector$ is calculated using Eq.(10).

$$MutatedVector = Mi - Si \quad (10)$$

The mutation operator M generates the mutated solution Mi, which is then used to

calculate the vector difference MutatedVector. Adding this MutatedVector to the old position of the solution updates the position to the new MutatedPosition.

By incorporating the differential mutation operator in the global search phase, *RFLIRP* introduces additional exploration and diversification. This helps the algorithm overcome local optima and effectively search for more globally optimal solutions. The perturbation introduced by the mutation operator allows *RFLIRP* to explore new regions of the search space and potentially discover better solutions that were not reachable through the previous search process. The mathematical expansion of the improved global search capability involves defining and applying the differential mutation operator, which perturbs existing solutions to generate new mutated solutions. This mutation process, combined with the solution update equation, enhances the exploration and diversification in the search process, improving the algorithm's ability to find globally optimal solutions.

3.3.4. Simplified search procedure

The *RFLIRP* simplifies the search procedure by eliminating time-consuming condition update phases. These phases in the original FLIA involve updating and adjusting various parameters or conditions during the search process. *RFLIRP* focuses on the core principles and mechanisms of FLIA while streamlining the algorithm and reducing unnecessary computational overhead. The specific changes made to eliminate time-consuming condition update phases and how they impact the computational complexity and efficiency of the algorithm are listed below.

- **Time-Consuming Condition Update Phases:** The FLIA includes time-consuming condition update phases, where the algorithm updates and adjusts various parameters or conditions during the search process. These update phases aim to fine-tune and adapt the algorithm's behaviour to the problem. However, they can impose a significant computational burden, especially for complex optimization problems.
- **Simplification in RFLIRP:** *RFLIRP* simplifies the search procedure by omitting these time-consuming condition update phases. Instead, it focuses on the core principles and mechanisms of FLIA while reducing unnecessary computational overhead. By eliminating these phases, *RFLIRP* streamlines the algorithm and improves its computational efficiency without compromising its search performance.
- **Impact on Computational Complexity and Efficiency:** The omission of time-consuming condition update phases in *RFLIRP* has several implications for the computational complexity and efficiency of the algorithm:
- **Reduced Computational Complexity:** By removing the time-consuming condition update phases, *RFLIRP* reduces the number of calculations and operations required during each iteration. This reduction in complexity leads to faster execution times and lower computational demands, making *RFLIRP* more efficient in terms of computational resources.
- **Improved Efficiency:** The simplification in *RFLIRP* improves the overall efficiency of the algorithm. The algorithm can allocate more computational resources towards the actual search process with fewer computational operations. This improved efficiency allows *RFLIRP* to handle larger problem instances or execute more iterations within a given time frame.
- **Practical Feasibility for Real-World Applications:** The simplified search procedure in *RFLIRP* makes the algorithm more practical and feasible for real-world applications. By reducing the computational complexity and improving efficiency, *RFLIRP* becomes more accessible to a wider range of problems and computational resources. It enables the algorithm to tackle real-world optimization challenges efficiently and effectively.

Algorithm 4: *RFLIRP*

Input:
- Population size: The number of frogs in each generation.
- Maximum number of iterations: The maximum number of iterations the algorithm will run.
- Other problem-specific parameters and constraints.

Output:
- Best solution found: The solution with the highest fitness value obtained by the algorithm.

Procedure:

Step 1: Initialize the algorithm parameters, including population size and maximum

number of iterations.
Step 2: Generate an initial population of frogs randomly.
Step 3: Evaluate the fitness of each frog in the population.
Step 4: Repeat the following steps until reaching the maximum number of iterations:
Step 5: Shuffle the population:
Step 6: Randomly permute the order of the frogs in the population.
Step 7: Perform the leaping phase:
Step 8: For each frog in the population:
Step 9: Select a random frog (possibly itself) as a leader.
Step 10: Generate a Lévy flight vector.
Step 11: Update the frog's position using the Lévy flight vector.
Step 12: Evaluate the fitness of the new position.
Step 13: Update the population:
Step 14: Sort the population based on fitness in descending order.
Step 15: Replace the worst-performing frogs with the best-performing frogs from the previous iteration, preserving the order.
Step 16: Output the best solution found.

4.0 SIMULATION SETTINGS

GNS3 (Graphical Network Simulator-3) is a powerful and popular network simulation software widely used by network engineers, researchers, and students. It provides a user-friendly graphical interface for designing and simulating complex network topologies, making it an invaluable tool for network planning, testing, and troubleshooting. With GNS3, users can create virtual network environments that accurately replicate real-world network scenarios. It supports the simulation of a wide range of network devices, including routers, switches, and firewalls, allowing users to configure and interconnect them as needed. GNS3 leverages real operating systems and virtual machine images to emulate network devices, providing a realistic environment for testing network configurations and protocols. The software offers many features, such as network visualization, packet capture and analysis, and the ability to integrate with external virtualization platforms. GNS3 also supports network automation and orchestration, allowing users to leverage APIs and scripting languages for network programmability. GNS3 empowers network professionals to gain hands-on experience, experiment with different network setups, and enhance their networking skills in a virtualized and risk-free environment. Its versatility and extensive feature set make it an indispensable tool in network simulation and education.

Table 1. Simulation Settings

Simulation Setting	Value(s)
Node Count	1500
Network Area Size	150m x 225m
Topology	Random Graph
Traffic Pattern	Poisson
Simulation Duration	900 seconds (i.e., 15 minutes)
Deployment Model	Event-Driven
Obstacle Placement	Random
Transmit Energy	0.1 Joules/bit
Receive Energy	0.05 Joules/bit
Idle Energy	1.0 mW
Sleep Energy	0.1 mW
Battery Capacity	2000 mAh
Simulation Environment	GNS-3
Experimental Repetitions	10

5.0 RESULTS AND DISCUSSION

5.1. Packet Delivery Ratio

Figure 1 depicts the results of a packet delivery ratio analysis, which evaluates the performance of three routing algorithms: DORA, PSORS, and RFLIRP. Table 2 provides the values of the result.

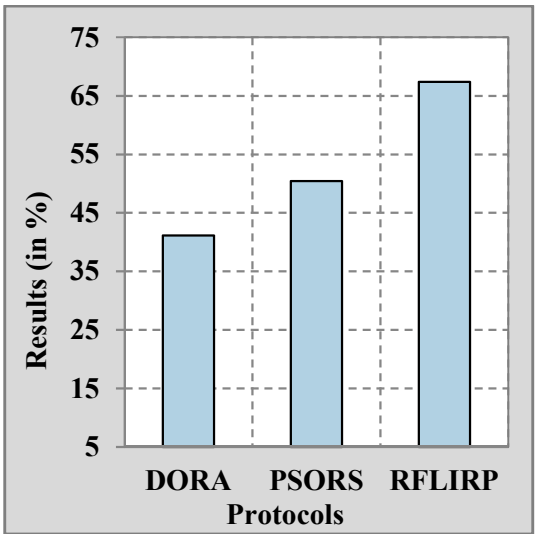

Figure 1. Packet Delivery Ratio Analysis

DORA prioritizes forwarding packets to their intended destination nodes as a destination-oriented algorithm. However, this approach may not always yield optimal routing decisions in complex or congested network scenarios. PSORS leverages swarm intelligence to guide routing decisions, enabling exploration of the network topology and discovering optimal routes for packet transmission. However, as the network complexity increases with more nodes, maintaining efficient swarm behavior and finding optimal routes becomes more challenging. RFLIRP is likely designed with adaptive routing strategies that can dynamically adjust to changing network conditions. This adaptability allows RFLIRP to optimize the packet delivery process, ensuring efficient and reliable routing paths even in dynamic or congested network environments.

For the DORA algorithm, the packet delivery ratio starts at 51.81% for 150 nodes and gradually decreases to 29.16% for 1500 nodes. On average, the DORA algorithm achieves a packet delivery ratio of 41.16%. In the case of the PSORS algorithm, the packet delivery ratio starts at a higher value of 58.67% for 150 nodes but also experiences a downward trend, reaching 42.02% for 1500 nodes. On average, the PSORS algorithm achieves a packet delivery ratio of 50.43%. Lastly, the RFLIRP algorithm begins with a packet delivery ratio of 73.79% for 150 nodes, significantly higher than the other two algorithms. However, it also experiences a gradual decline, with a packet delivery ratio of 59.67% for 1500 nodes. On average, the RFLIRP algorithm achieves a packet delivery ratio of 67.38%.

Based on the average packet delivery ratio values, it can infer that the RFLIRP algorithm performs the best among the three, with the PSORS algorithm in the middle and the DORA algorithm exhibiting the lowest performance.

Table 2. Packet Delivery Ratio Result Values

Nodes	DORA	PSORS	RFLIRP
150	51.81	58.67	73.79
300	49.79	56.48	73.11
450	47.48	53.69	72.16
600	46.80	53.06	69.53
750	45.12	52.17	67.72
900	39.91	50.31	66.81
1050	35.53	48.33	65.19
1200	34.01	45.68	64.00
1350	31.99	43.82	61.83
1500	29.16	42.02	59.67
Average	**41.16**	**50.43**	**67.38**

5.2. Throughput

Figure 2 illustrates the results of a throughput analysis, which assesses the data transmission capacity of three routing algorithms: DORA, PSORS, and RFLIRP. Table 3. presents the throughput values achieved by each algorithm at varying node densities. By examining Figure 2, it can be observed that throughput decreases as the number of nodes increases.

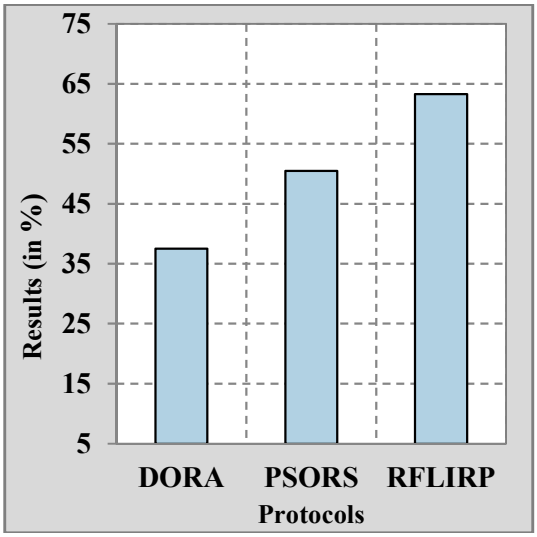

Figure 2. Throughput Analysis

DORA achieves moderate throughput values because it primarily focuses on forwarding packets towards their destination nodes. While this approach ensures packets reach their intended destinations, it may not fully optimize the network resources or consider alternative paths that could enhance throughput. PSORS demonstrates improved throughput compared to DORA due to its utilization of swarm intelligence. By simulating the behavior of a swarm, PSORS explores the network topology and discovers more optimal routes for packet transmission. This exploration helps find better paths and utilize network resources more

efficiently, improving the overall throughput. RFLIRP achieves the highest throughput values among the three algorithms. This can be attributed to its robust routing strategies inspired by frog leaping behavior. RFLIRP adapts to changing network conditions and selects relay nodes based on proximity to the destination, available energy, and link quality. This adaptability and intelligent selection of relay nodes enable RFLIRP to utilize network resources, effectively improving throughput performance.

DORA demonstrates throughput values ranging from 33.05% at 150 nodes to 42.51% at 1500 nodes. On average, DORA achieves a throughput of 37.52%. These results suggest that DORA can handle moderate data transmission but may face limitations in scenarios with higher node densities. PSORS exhibits throughput values ranging from 46.27 units at 150 nodes to 54.51% at 1500 nodes. On average, PSORS achieves a throughput of 50.48%. These findings indicate that PSORS performs better than DORA regarding data transmission efficiency, implying its potential to handle data more effectively across the network. RFLIRP stands out with the highest throughput values among the three algorithms. Its throughput ranges from 55.36% at 150 nodes to 71.23% at 1500 nodes. On average, RFLIRP achieves a throughput of 63.31%. These results highlight RFLIRP's superior capability in efficiently transmitting significant data across the network.

Table 3. Throughput Result Values

Nodes	DORA	PSORS	RFLIRP
150	33.05	46.27	55.36
300	33.49	46.81	57.50
450	34.23	48.36	59.86
600	35.35	48.64	60.15
750	36.02	50.10	64.53
900	36.72	50.91	64.86
1050	40.69	52.68	65.00
1200	41.29	52.78	66.04
1350	41.90	53.81	68.58
1500	42.51	54.51	71.23
Average	**37.52**	**50.48**	**63.31**

The throughput analysis reveals that RFLIRP outperforms both PSORS and DORA in achieving higher throughput values. PSORS demonstrates moderate throughput performance, while DORA exhibits relatively lower throughput values. The routing algorithm's choice should consider the network's specific requirements and the desired balance between throughput and other performance metrics.

5.3. Packet Delay

Packet Delay Analysis is a quantitative assessment of the time it takes for packets to traverse a network and reach their destination. In Figure 3, the analysis is based on the values provided in Table 4, which presents the packet delay results for three different routing algorithms: DORA, PSORS, and RFLIRP.

The packet delay analysis of the three routing protocols, DORA, PSORS, and RFLIRP, reveals distinct performance characteristics. DORA, with an average packet delay of 12,884.4 milliseconds, exhibits the highest delay among the three algorithms. This can be attributed to DORA's destination-oriented approach, which analyses destination addresses and selects paths based on network topology. While DORA provides a functional routing solution, its delay is relatively higher, indicating potential inefficiencies in path selection or congestion management.

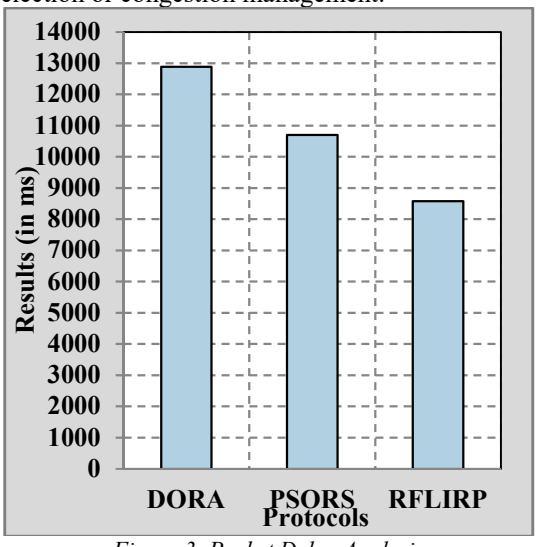

Figure 3. Packet Delay Analysis

PSORS demonstrates improved performance with an average packet delay of 10,703.2 milliseconds. The PSORS leverages the concepts of fitness evaluation, particle movement, and convergence to find optimal paths. By iteratively searching for better routes, PSORS can

achieve a lower average packet delay compared to DORA. This suggests that PSORS' optimization process effectively minimizes delays by selecting paths that optimize metrics such as delay, bandwidth utilization, and link quality.

RFLIRP exhibits the lowest average packet delay of 8,580.9 milliseconds. RFLIRP's working mechanism, inspired by the leaping behavior of frogs, incorporates robust path selection based on factors such as link stability, congestion, and reliability. By leveraging these criteria, RFLIRP can identify more efficient paths with reduced delays. The superior performance of RFLIRP indicates that its frog leaping-inspired approach enhances packet routing efficiency, resulting in lower delays compared to DORA and PSORS.

The packet delay analysis based on Table 4 reveals that RFLIRP outperforms DORA and PSORS regarding average packet delay. This can be attributed to RFLIRP's robust path selection mechanism, which prioritizes link stability and congestion avoidance. While DORA and PSORS offer viable routing solutions, their relatively higher delay values indicate areas for improvement. The performance of RFLIRP, with its frog-leaping-inspired approach, highlights the effectiveness of its routing strategy in minimizing delays and optimizing packet routing in the network.

Table 4. Packet Delay Result Values

Nodes	DORA	PSORS	RFLIRP
150	12455	9903	7262
300	12490	9966	7288
450	12512	10284	7341
600	12549	10341	8567
750	12767	10492	8784
900	12982	10552	9130
1050	13090	10675	9167
1200	13206	11079	9208
1350	13253	11348	9211
1500	13540	12392	9851
Average	**12884.4**	**10703.2**	**8580.9**

5.4. Energy Consumption

Figure 4 compares the average energy consumption for three routing algorithms: DORA, PSORS, and RFLIRP. Table 5 presents each algorithm's energy consumption, allowing us to evaluate its efficiency. According to Figure 4, DORA has an average energy consumption of 82.98%, while PSORS has an average energy consumption of 65.74%. Lastly, RFLIRP demonstrates the lowest average energy consumption among the three algorithms, with a value of 49.10%.

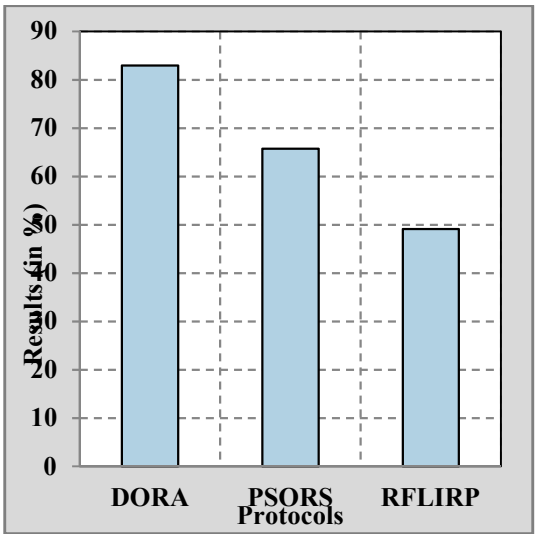

Figure 4. Energy Consumption Analysis

The higher energy consumption of DORA could be attributed to its destination-oriented routing approach, which may involve longer routes or suboptimal resource utilization, leading to increased energy usage. PSORS optimizes energy consumption by iteratively searching for better routes, considering link quality and bandwidth utilization factors. This optimization process contributes to the lower energy consumption observed in PSORS compared to DORA. RFLIRP consumes the least energy compared to DORA and PSORS. This superior energy efficiency can be attributed to RFLIRP's robust path selection mechanism inspired by the leaping behavior of frogs. RFLIRP selects paths that prioritize energy efficiency, link stability, and congestion avoidance, resulting in reduced energy consumption in the network.

The Energy Consumption Analysis based on Table 5 indicates that RFLIRP demonstrates the highest energy efficiency, followed by PSORS and DORA. The lower energy consumption of RFLIRP and PSORS compared to DORA suggests that their routing mechanisms are more effective in optimizing energy usage. RFLIRP's energy efficiency can be attributed to its robust path selection inspired by frog leaping behavior, while PSORS achieves lower energy consumption through particle swarm optimization. These findings highlight the importance of energy-efficient routing protocols in reducing energy consumption and improving network sustainability.

Table 5. Energy Consumption Result Values

Nodes	DORA	PSORS	RFLIRP
150	74.51	56.39	43.15
300	75.61	57.94	43.62
450	77.90	60.36	44.21
600	81.21	60.98	45.09
750	82.27	61.51	47.02
900	84.47	68.90	51.70
1050	86.63	69.51	52.88
1200	87.80	71.74	53.16
1350	88.75	73.96	54.76
1500	90.63	76.16	55.45
Average	**82.98**	**65.74**	**49.10**

5.5. Network Lifetime

The Network Lifetime Result Values graph provides a detailed comparison of the average network lifetimes achieved by three routing algorithms: DORA, PSORS, and RFLIRP, as presented in Table 6. Table 6 presents the average network lifetime values for each algorithm, measured in percentage units, allowing for a comprehensive analysis of their performance in terms of network longevity.

DORA achieves an average network lifetime of 17.19%. This indicates that, on average, the network employing the DORA routing algorithm can sustain its operations for approximately 17.19% of the total expected network lifespan. This relatively lower network lifetime could be attributed to potential inefficiencies in energy consumption or routing decisions within the DORA algorithm, leading to shorter network usability. PSORS demonstrates a significantly higher average network lifetime of 37.41%. This implies that, on average, the network utilizing the PSORS algorithm can maintain its functionality for around 37.41% of the total expected network lifespan. The improved network lifetime of PSORS can be attributed to its particle swarm optimization-based approach, which optimizes energy consumption, load balancing, and network connectivity, thereby enhancing the overall network lifespan.

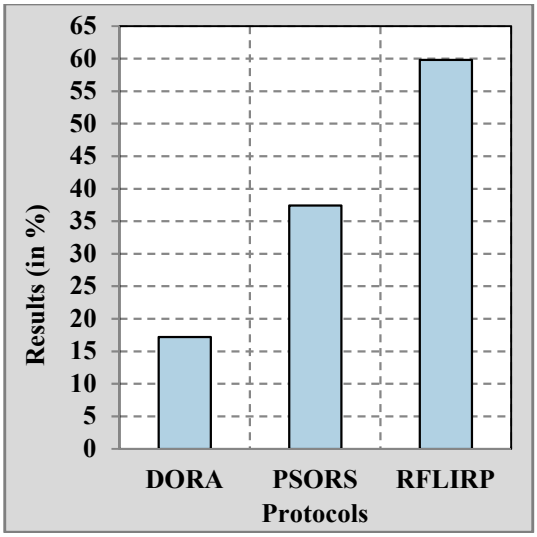

Figure 5. Network Lifetime Analysis

RFLIRP routing protocol showcases the highest average network lifetime among the three algorithms, with a value of 59.83%. This suggests that, on average, the network employing the RFLIRP protocol can sustain its operations for approximately 59.83% of the total expected network lifespan. The significantly longer network lifetime achieved by RFLIRP highlights the effectiveness of its robust path selection mechanism, which draws inspiration from frog leaping behavior. RFLIRP's path selection strategy prioritizes energy efficiency, link stability, and congestion avoidance, improving network longevity and enhancing overall network performance.

The Network Lifetime Result Values graph and Table 6 demonstrate that RFLIRP achieves the highest average network lifetime,

followed by PSORS and DORA. This indicates that RFLIRP's routing mechanism, which considers energy efficiency and stability factors, contributes to an extended network lifespan. On the other hand, DORA exhibits a relatively shorter network lifetime, indicating potential areas for improvement in energy optimization and network longevity. PSORS strikes a balance between the other two algorithms, offering an average network lifetime.

Table 6. Network Lifetime Result Values

Nodes	DORA	PSORS	RFLIRP
150	24.81	50.53	66.10
300	23.96	49.68	65.74
450	21.13	46.24	64.10
600	18.71	44.56	63.39
750	17.63	37.49	62.36
900	16.73	31.14	62.09
1050	13.43	29.72	55.71
1200	12.77	29.14	54.68
1350	12.22	27.86	53.48
1500	10.51	27.79	50.65
Average	**17.19**	**37.41**	**59.83**

6. CONCLUSION

The energy efficiency of routing algorithms in Internet of Things-based Cloud Wireless Sensor Networks (IC-WSN) for greenhouse farming is crucial for optimizing resource utilization and ensuring sustainable agricultural practices. The Robust Frog Leap Inspired Routing Protocol (RFLIRP) presented in this paper effectively addresses the energy consumption challenge in IC-WSN routing. The experimental results validate the efficacy of RFLIRP in significantly reducing energy consumption compared to existing routing protocols. By intelligently selecting energy-efficient paths and implementing data aggregation and compression techniques, RFLIRP extends the operational lifespan of sensors, minimizes maintenance costs, and promotes eco-friendly farming practices. The adoption of RFLIRP empowers greenhouse farmers to enhance productivity, reduce resource wastage, and ensure the long-term viability of their operations. Furthermore, RFLIRP enables uninterrupted data transmission and timely monitoring of critical parameters in large-scale greenhouse environments. Further research can focus on refining RFLIRP and exploring additional optimization techniques to improve energy efficiency and overall performance in IC-WSN routing for greenhouse farming applications.

REFERENCES

[1] G. Bittante and C. Cipolat-Gotet, "Direct and indirect predictions of enteric methane daily production, yield, and intensity per unit of milk and cheese, from fatty acids and milk Fourier-transform infrared spectra," *J. Dairy Sci.*, vol. 101, no. 8, pp. 7219–7235, 2018, doi: 10.3168/jds.2017-14289.

[2] J. Wang, A. Elbery, and H. A. Rakha, "A real-time vehicle-specific eco-routing model for on-board navigation applications capturing transient vehicle behavior," *Transp. Res. Part C Emerg. Technol.*, vol. 104, pp. 1–21, 2019, doi: 10.1016/j.trc.2019.04.017.

[3] H. A. Mupambwa, M. K. Hausiku, A. D. Nciizah, and E. Dube, "The unique Namib desert-coastal region and its opportunities for climate smart agriculture: A review," *Cogent Food Agric.*, vol. 5, no. 1, 2019, doi: 10.1080/23311932.2019.1645258.

[4] K. Paul *et al.*, "Viable smart sensors and their application in data driven agriculture," *Comput. Electron. Agric.*, vol. 198, 2022, doi: 10.1016/j.compag.2022.107096.

[5] G. Georgiadis, A. Komninos, A. Koskeris, and J. Garofalakis, "Implementing an Integrated Internet of Things System (IoT) for Hydroponic Agriculture," *Internet of Things*. pp. 83–102, 2021. doi: 10.1007/978-3-030-67197-6_5.

[6] M. P. Islam and K. Hatou, "TheLR531v1 – A deep learning multi-branch CNN architecture for day-night automatic segmentation of horticultural crops," *Comput. Electron. Agric.*, vol. 204, p. 107557, 2023, doi: 10.1016/j.compag.2022.107557.

[7] R. Dogra, S. Rani, H. Babbar, and D. Krah, "Energy-Efficient Routing Protocol for Next-Generation Application in the Internet of Things and Wireless Sensor Networks," *Wirel. Commun. Mob. Comput.*, vol. 2022, 2022, doi: 10.1155/2022/8006751.

[8] H. Khalid, S. J. Hashim, S. M. S. Ahmad, F. Hashim, and M. A. Chaudhary, "Robust multi-

gateway authentication scheme for agriculture wireless sensor network in society 5.0 smart communities," *Agric.*, vol. 11, no. 10, 2021, doi: 10.3390/agriculture11101020.

[9] M. Mohinur Rahaman and M. Azharuddin, "Wireless sensor networks in agriculture through machine learning: A survey," *Comput. Electron. Agric.*, vol. 197, p. 106928, 2022, doi: 10.1016/j.compag.2022.106928.

[10] C. Lyu, X. Zhang, Z. Liu, and C. H. Chi, "Selective Authentication Based Geographic Opportunistic Routing in Wireless Sensor Networks for Internet of Things Against DoS Attacks," *IEEE Access*, vol. 7, pp. 31068–31082, 2019, doi: 10.1109/ACCESS.2019.2902843.

[11] L. Cheng *et al.*, "Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks," *Comput. Networks*, vol. 134, pp. 66–77, 2018, doi: 10.1016/j.comnet.2018.01.012.

[12] M. Z. Sharif, N. Di, and F. Liu, "Monitoring honeybees (Apis spp.) (Hymenoptera: Apidae) in climate-smart agriculture: A review," *Appl. Entomol. Zool.*, vol. 57, no. 4, pp. 289–303, 2022, doi: 10.1007/s13355-021-00765-3.

[13] N. Achyutha Prasad *et al.*, "Delay optimization and energy balancing algorithm for improving network lifetime in fixed wireless sensor networks," *Phys. Commun.*, vol. 58, p. 102038, 2023, doi: 10.1016/j.phycom.2023.102038.

[14] N. T. Dinh, T. Gu, and Y. Kim, "Rendezvous Cost-Aware Opportunistic Routing in Heterogeneous Duty-Cycled Wireless Sensor Networks," *IEEE Access*, vol. 7, pp. 121825–121840, 2019, doi: 10.1109/ACCESS.2019.2937252.

[15] D. N. Patel, S. L. G. Joshi, and V. Ravikumar, "Agriculture Monitoring System Using IoT—A Survey," *Advances in Intelligent Systems and Computing*, vol. 1053. pp. 631–648, 2020. doi: 10.1007/978-981-15-0751-9_59.

[16] Y. He, G. Han, H. Wang, J. Adu Ansere, and W. Zhang, "A sector-based random routing scheme for protecting the source location privacy in WSNs for the Internet of Things," *Futur. Gener. Comput. Syst.*, vol. 96, pp. 438–448, 2019, doi: https://doi.org/10.1016/j.future.2019.02.049.

[17] N. R. Patel, S. Kumar, and S. K. Singh, "Energy and Collision Aware WSN Routing Protocol for Sustainable and Intelligent IoT Applications," *IEEE Sens. J.*, vol. 21, no. 22, pp. 25282–25292, 2021, doi: 10.1109/JSEN.2021.3076192.

[18] G. Arya, A. Bagwari, and D. S. Chauhan, "Performance Analysis of Deep Learning-Based Routing Protocol for an Efficient Data Transmission in 5G WSN Communication," *IEEE Access*, vol. 10, pp. 9340–9356, 2022, doi: 10.1109/ACCESS.2022.3142082.

[19] U. M. Durairaj and S. Selvaraj, "Two-Level Clustering and Routing Algorithms to Prolong the Lifetime of Wind Farm-Based WSN," *IEEE Sens. J.*, vol. 21, no. 1, pp. 857–867, 2021, doi: 10.1109/JSEN.2020.3015734.

[20] N. Ma, H. Zhang, H. Hu, and Y. Qin, "ESCVAD: An Energy-Saving Routing Protocol Based on Voronoi Adaptive Clustering for Wireless Sensor Networks," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 9071–9085, 2022, doi: 10.1109/JIOT.2021.3120744.

[21] S. Al-Otaibi, A. Al-Rasheed, R. F. Mansour, E. Yang, G. P. Joshi, and W. Cho, "Hybridization of Metaheuristic Algorithm for Dynamic Cluster-Based Routing Protocol in Wireless Sensor Networksx," *IEEE Access*, vol. 9, pp. 83751–83761, 2021, doi: 10.1109/ACCESS.2021.3087602.

[22] T. Kaur and D. Kumar, "MACO-QCR: Multi-Objective ACO-Based QoS-Aware Cross-Layer Routing Protocols in WSN," *IEEE Sens. J.*, vol. 21, no. 5, pp. 6775–6783, 2021, doi: 10.1109/JSEN.2020.3038241.

[23] W. K. Yun and S. J. Yoo, "Q-Learning-based data-aggregation-aware energy-efficient routing protocol for wireless sensor networks," *IEEE Access*, vol. 9, pp. 10737–10750, 2021, doi: 10.1109/ACCESS.2021.3051360.

[24] Y. Xu, W. Jiao, and M. Tian, "An Energy-Efficient Routing Protocol for 3D Wireless Sensor Networks," *IEEE Sens. J.*, vol. 21, no. 17, pp. 19550–19559, 2021, doi: 10.1109/JSEN.2021.3086806.

[25] R. Dogra, S. Rani, H. Babbar, S. Verma, K. Verma, and J. J. P. C. Rodrigues, "DCGCR: Dynamic Clustering Green Communication Routing for Intelligent Transportation Systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16197–16205, 2022, doi: 10.1109/TITS.2022.3148471.

[26] K. Wang, C. M. Yu, and L. C. Wang, "DORA: A Destination-Oriented Routing Algorithm for Energy-Balanced Wireless Sensor Networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 2080–2081, 2021, doi: 10.1109/JIOT.2020.3025039.

[27] G. Tong, S. Zhang, W. Wang, and G. Yang, "A particle swarm optimization routing scheme for

© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

wireless sensor networks," *CCF Trans. Pervasive Comput. Interact.*, 2022, doi: 10.1007/s42486-022-00118-1.

[28] J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," *World J. Eng.*, vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.

[29] J. Ramkumar and R. Vadivel, "Performance Modeling of Bio-Inspired Routing Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/ijies2019.0228.22.

[30] M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.

[31] J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.

[32] J. Ramkumar, S. Samson Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[33] R. J, "Meticulous Elephant Herding Optimization based Protocol for Detecting Intrusions in Cognitive Radio Ad Hoc Networks," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 8, pp. 4548–4554, 2020, doi: 10.30534/ijeter/2020/82882020.

[34] J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," *Wirel. Pers. Commun.*, vol. 120, no. 2, pp. 887–909, Apr. 2021, doi: 10.1007/s11277-021-08495-z.

[35] J. Ramkumar, "Bee inspired secured protocol for routing in cognitive radio ad hoc networks," *Indian J. Sci. Technol.*, vol. 13, no. 30, pp. 2159–2169, 2020, doi: 10.17485/ijst/v13i30.1152.

[36] J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," *2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022*, pp. 1–6, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9752899.

[37] P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," *2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022*, pp. 1–5, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9753203.

[38] R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.

[39] R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.

[40] R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," *Inc. Internet Things Healthc. Appl. Wearable Devices*, pp. 109–121, 2019, doi: 10.4018/978-1-7998-1090-2.ch006.

[41] J. Ramkumar and R. Vadivel, "CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks," in *Advances in Intelligent Systems and Computing*, 2017, vol. 556, pp. 145–153. doi: 10.1007/978-981-10-3874-7_14.