Wireless Routing Protocol

Optimization of IoT Routing Protocols

Harshil (20BCT0125)[a], Syed Ahsan Mehdi (20BCT0180)[a],

Ananya Aditi (20BCT0170)[a], Prachurya Priyadarshini (20BCT0155)[a]

[a]School of Computer Science and Engineering
Vellore Institute of Technology

ABSTRACT

The past few years have seen a spike in the popularity of wireless sensor networks and have also raised many concerns over the protocols deployed in them for communication. The nodes in wireless sensor networks are very low on resources and are at a higher security risk which require the routing algorithms to be both low on overhead and more flexible in the aspect of when the network is being attacked. We propose a deep learning-based routing protocol which can be used when the system is under attack by predicting the reliability of a link but be more energy efficient when the system is safer to prolong the network's lifetime. Since the paper tackles two problems, the framework is also divided into two parts. The first part being about the deep learning model for resilient communication during attacks and the second part being energy efficient routing when under normal conditions.

1. Introduction

Wireless Sensor Network (WSN) consists of a number of sensor nodes placed in the monitoring area. These sensor nodes cooperate with each other to collect monitoring information to the base station. With the development of information technology, wireless sensor networks have been widely used in agricultural irrigation management, military intrusion monitoring, industrial control and other _elds. How to reduce the node energy consumption and maintain energy balance has been a hot topic in wireless sensor network research. With the development of energy acquisition technology, it is possible for the network to run indefinitely by equipping nodes with energy collection devices such as solar cells and thermoelectric cells.

Email Addresses: harshil.2020@vitstudent.ac.in (Harshil)
syedahsan.mehdi2020@vitstudent.ac.in (Syed Ahsan Mehdi)
ananya.aditi2020@vitstudent.ac.in (Ananya Aditi)
priyadarshini.p2020@vitstudent.ac.in (Prachurya Priyadarshini)

The WSN architecture is based on the collaboration of small sensor nodes. Each node has a battery that indicates its energy capacity. In WSN, sensor nodes collect data and route it through intermediate nodes and wireless links to transfer data to the sink node. Therefore, an effective routing protocol is necessary to transfer data to the sink node. WSN has a wide range of applications in industry, environmental monitoring, health monitoring, military and etc. However, these networks suffer in terms of computing power, energy constraints and security. The nodes clustering, selection of cluster heads (CHs) and routing through them can reduce the number of nodes participating in the route and thus lead to a reduction in energy consumption. In clustering-based routing, CHs receive data from member nodes and then transmit the data to the sink node via single-hop or multi-hop routing. Multi-hop routing provides an opportunity for nodes to control malicious activity on the WSN. However, with the increasingly fine network control and the expansion of network scale, more and more nodes are emerged. The network traffic grows exponentially and the demand is becoming more diverse. The traditional routing algorithm based on shortest path algorithm has the problem of slow convergence speed, which is not suitable for wireless sensor network, and the response to network changes may lead to serious congestion. Therefore, optimizing the routing process of WSN is crucial to ensure the service quality and promote the development and evolution of WSN.

In recent years, machine learning (ML) has attracted much attention from many researchers due to its capability in intelligent decision-making and large-scale data processing. These advantages make it possible to be an effective weapon to solve the current deadlock between network maintenance and management. In order to avoid the disadvantages of traditional routing methods, many researchers try to introduce intelligent algorithms such as machine learning into WSN routing mechanism. Unfortunately, it is a huge challenge to design a routing mechanism that employs ML to realize real-time and customizable optimization. Therefore, WSN development based on energy and resource is essential.

In view of the above situation and challenges, in this work, we focus on the use of a deep reinforcement learning for routing optimization in wireless sensor network. We integrate the deep reinforcement learning mechanism Deep Deterministic Policy Gradient (Deep Learning) into the routing process of WSN, which can effectively avoid the storage resources and time overhead of lookup tables brought by the maintenance of large-scale Q tables. The proposed mechanism has the advantages of low latency, high throughput, and it can realize black-box optimization in continuous time. In addition, the experimental results demonstrate that Deep Learning optimized WSN routing method not only converges well, but also realizes better stability and performance than previous routing methods such as LEACH (Low-energy adaptive clustering hierarchy).

2. Comparative Literature Analysis

2.1 Literature Survey

2.1.1 *Deep reinforcement learning (DRL) based mobility load balancing (MLB) algorithm*

Yue Xu, et al. ,2019 [1] uses the algorithm along with a two-layer architecture to solve the large-scale load balancing problem for ultra-dense networks (UDNs). Min Xiang et al. 2022 [22] focused on Modeling analysis performed for WSN to obtain system state, migration action set, and system reward. Q-values of switch migration using Double Deep Q-Network (DDQN). Pengjun Wang et al.,2022 [24] used MATLAB in which the nodes were distributed using random

function. The researchers used deep learning to divide the network state by finding the minimum number of hopes from each node to the sink node. Now using this network state data, the routing can be optimized by using the minimum number of nodes.

2.1.2 *AODV, DSDV*

Weidong Fang et al.,2019 [3] proposes a trust management system for of computer, an industrial wireless sensor network (F-IWSN). This is done so because the F-IWSN has weak security policies specially for internal attacks. To tackle this the proposed system makes the source node to calculate a "Trust value" of the net hop node based on historical interaction information with the other node based on which a gaussian distribution model is made. The second part of the paper deals with routing scheme to estimate if a trade-off between security and energy is helpful. This is done with the help of grey-based AODV. Muhammed Adil et al. 2020 [6] uses Hybrid routing scheme using a dynamic cluster-based static routing protocol (DCBSRP), leveraging the ad hoc on-demand distance vector (AODV) routing protocol and low-energy adaptive clustering hierarchy (LEACH) protocol. Bal Krishna Saraswat et al.,2015 [12] proposes the simulation results in order to choose the best routing protocol to give the highest performance when implement the routing protocols in the target mobile grid application. This simulation gives the results for three ad hoc routing protocols named DSDV, DSR and AODV.

2.1.3 *SDN Architecture*

Elham Hajian et al. 2022 [21] aims at a novel SDN architecture with BS and controller discovery, link, and virtual routing, is proposed to decrease load distribution and extending lifetime. A new method of load-balancing routing using SDN and virtualization is proposed. Chunlei Xu et al.,2020 [10] uses DDPG algorithm to for routing optimization in software defined networks. The deep learning algorithm uses three signals such as action, state, reward. The agent performs an action on the environment with the intention of altering the weight of the network links.

2.1.4 *Protocols in WSN*

De-gan Zhang et al., 2019 [2] proposed a system for marginal wireless sensor networks where the traffic is very high when compared to normal wireless sensor networks. This high traffic leads to transmission delay and data loss caused by collision. The authors propose a system where the data being transmitted, will be divided into multiple packets and then sent through multiple channels and then aggregated again at the destination node. This ensures an even distribution of traffic across the network. Arafat Habib et al.,2019 [5] discusses Feedback routing Optimization which uses Q learning and sends information about environment, current status to other nodes as feedback. Some other routing protocols that are compared in this paper include Multi agent reinforcement learning on QoS, Routing protocols to optimise network lifetime (RLLO), RL-based Routing Protocol for Multi-hop WSNs, etc. M.V.N.R. Pavan Kumar et al., 2022 [8] focuses on estimating implementation costs and learning about design, implementation, and statistical practice of GPRS algorithm in WSNs. Ibrahim A. Abd El-Moghith and Saad M. Darwish, 2021 [14] offers a trusted routing method that combines deep blockchain and Markov Decision Processes (MDPs) in order to enhance the routing security and efficiency of WSNs. V. Sridhar et al. 2021 [16] proposed a machine learning technique for effective routing in WSNs. It can route packets around busy locations by selecting nodes with higher energy and lower load. The proposed SRTRBC technique is composed of two steps: route path construction and congestion-aware MIMO routing. Bing Han et al. 2021 [17] considered how to add the security mechanism to our energy efficient hierarchical

clustering routing protocol for energy harvesting aware WSNs If there are malicious nodes eavesdropping on or attacking the WSN. This paper presents an efficient multi-hop routing protocol (EMRP) for efficient data dissemination in IoT-enabled WSNs where hierarchy-based energy-efficient routing is involved. It considers a rank-based next-hop selection mechanism.

2.1.5 *LEACH*

Muhammed Adil et al. 2020 [6] uses Hybrid routine scheme using a dynamic cluster-based static routing protocol (DCBSRP), leveraging the ad hoc on-demand distance vector (AODV) routing protocol and low-energy adaptive clustering hierarchy (LEACH) protocol. Prachi Maheshwari el at. 2020 [7] uses the Butterfly Optimization Algorithm (BOA) to select the best cluster head from a collection of nodes. Ant Colony Optimization (ACO) is used to determine the best path between the cluster head and the base station. Jiale Liang et al., 2021 [18] proposed a cooperative adaptive routing algorithm on mode selection in energy heterogeneous WSNs and two lemmas which are sufficient conditions for the use of the multihop cooperation mode as LEACH protocol is a classical routing protocol, but it is not applicable to heterogeneous WSNs. Based on these lemmas, an adaptive routing algorithm combined with DEEC is proposed. Zongshan Wang et al, 2020 [9] analysed the existing "clustering" concept, and have identified that the main problem in this method is effective selection of cluster heads. This problem has been tackled in this paper using Artificial Bee Colony (ABC) algorithm which takes account into the factors of network cluster head energy, cluster head density, cluster head location and other similar factors. For intra cluster communication the authors proposed polling control mechanism.

2.1.6 *Q-Learning*

Wan-Kyu Yun and Sang-Jo Yoo.,2021 [13] proposed Q-learning-based data-aggregation-aware energy-efficient routing algorithm. The reward-based system is implemented using reinforcement learning. The reward is made up of three different parameters which are efficiency of data aggregation energy of the node which is being used and the energy used during communication. Hussein Jawad Abdullah et al.,2022 [25] used many key concepts such as hierarchical routing, clustering and routing based on capsules to reduce the resource consumption and to optimize routing algorithm. Saleh M. Altowaijri et al. 2022 [23] includes division of network region into phases and evenly distributed cluster heads in it. They used Q learning which represented parameters like residual energy and hop length for calculating the Q-value.

2.1.7 *Machine Learning Algorithms*

Jong Hun Woo, et al. 2021 [11] focuses on Machine learning technology based on deep neural networks. A deep neural network-based reinforcement learning algorithm. Maryam Hajiee1 et al., 2021 [15] proposed an energy-aware trust and opportunity-based routing (ETOR) algorithm with respect to a novel hybrid fitness function. This algorithm has two main steps: one is to select secure nodes based on tolerance constant and the other is to select opportunistic nodes from secure nodes to perform routing. Ibraheem Barbahan et al.,2021 [19] method is based on wisdom of crowds and helps in allowing distributed routing algorithm, DQN-Routing, to generalise better to new network structures that were not present in the trained model. In this methodology the training process is divided into two parts i.e. pretraining on a topology and the second one is to continue training the system on another topology. Padmalaya Nayak et al.,2021 [20] explores the use of machine learning algorithms for optimizing routing in wireless sensor networks. The authors compared different machine learning techniques to compare the routing capabilities of nodes such as Artificial Neural Network (ANN), Bayesian-based routing, data routing using K-means and computational evolution algorithm, etc.

2.1.8 Bellman-Ford Algorithm

Kevin Schmid et al. 2021 [20] proposes a new algorithm, called Streaming Bellman-Ford, for computing single-source shortest paths in graphs with non-negative edge weights. The algorithm is designed to handle graphs that are too large to fit in memory and to be fault-tolerant in the face of node or link failures during computation. The Streaming Bellman-Ford algorithm operates by processing the graph in a series of rounds, where each round computes shortest paths for a subset of the vertices. In each round, the algorithm applies a novel optimization called early termination, which stops the computation for a subset of vertices when their shortest path is already known. The authors prove that the algorithm has the same theoretical complexity as the classic Bellman-Ford algorithm, but with significantly lower practical running time.

2.1.9 *Neural Networks & Load Balancing Routing Protocol*

Xinlu Li et. al. 2019 [4] EBAR (Effective load Balancing ant routing protocol). For routing, the authors used a pseudo random algorithm along with heuristic values to accelerate routing and have taken the energy levels into consideration while calculating the most efficient path to prolong the lifetime of the network by conserving the energy

2.2 Comparative Analysis of Existing Protocols

2.2.1 AODV

Ad-hoc On-demand Distance Vector (AODV) is a routing protocol used in mobile ad-hoc networks (MANETs). AODV is designed to be scalable, efficient, and adaptable to dynamic network topologies. AODV establishes a route between a source and a destination node only when required. This reduces the overhead of routing information and conserves network resources. When a source node needs to send a packet to a destination node, and it does not have a valid route to the destination node, it initiates a route discovery process. The route discovery process involves broadcasting a Route Request (RREQ) packet that propagates through the network until it reaches the destination or an intermediate node with a fresh enough route to the destination. Once a route is established, AODV monitors the route and detects any route failures or link failures. If a failure is detected, AODV initiates a route maintenance process to repair the route or find a new route to the destination. AODV uses sequence numbers to avoid routing loops and ensure the freshness of routing information. Each node maintains a sequence number for each destination, and the sequence number is incremented when a node learns a new route or receives an updated route. AODV uses hop-by-hop routing, where a packet is forwarded from one node to another until it reaches the destination. Each intermediate node maintains a route table that contains the next-hop information for each destination.

2.2.1.1 Pseudocode

Algorithm 1: AODV

```
ReceiveReply (Packet P) {

If(P has an entry in Route Table){
  select Dest_Seq_No from routing table

    if(P.Dest_Seq_No > Dest_Seq_No) {
      update entry of P in routing table
      unicast data packets to the route
      specified in RREP
```

```
    }

   else {
         discard RREP

  }
}

else {
if(P.Dest_Seq_No >=Srs_Seq_No) {
     Make entry of P in routing table
}

else {
discard this RREP

      }
  }
}
```

Table-1: Complexity of AODV

Space Complexity	O(n)
Best Case Time Complexity	O(1)
Average Case Time Complexity	O(d*log(n))
Worst Case Time Complexity	O(d*n*log(n))

where d is the maximum number of hops and n is the number of nodes in the network.

Table-1 shows the time and space complexities of AODV Routing Protocol

2.2.1.2 Limitations

Scalability: AODV can have scalability issues in large networks due to the need to maintain routing tables and control message overhead.

Latency: AODV relies on a route discovery process, which can introduce significant delays in the delivery of messages. This can be a problem in time-sensitive applications such as video or voice communication.

Route Maintenance Overhead: AODV requires frequent route maintenance messages to be sent, which can cause additional overhead in the network.

Security: AODV does not provide any inherent security mechanisms, making it vulnerable to attacks such as spoofing or impersonation.

Resource utilization: AODV i not be suitable for networks with limited resources such as battery-powered devices, as it requires additional processing power and energy consumption to maintain routing tables and exchange control messages.

Congestion control: AODV does not have any built-in congestion control mechanisms, which can lead to network congestion and packet loss in high-traffic scenarios.

Fig-1: shows AODV Routing Protocol depicting the area, number of nodes, grid size, base stations and the transmission range. Source nodes are highlighted in green and the Sink nodes are highlighted in yellow.

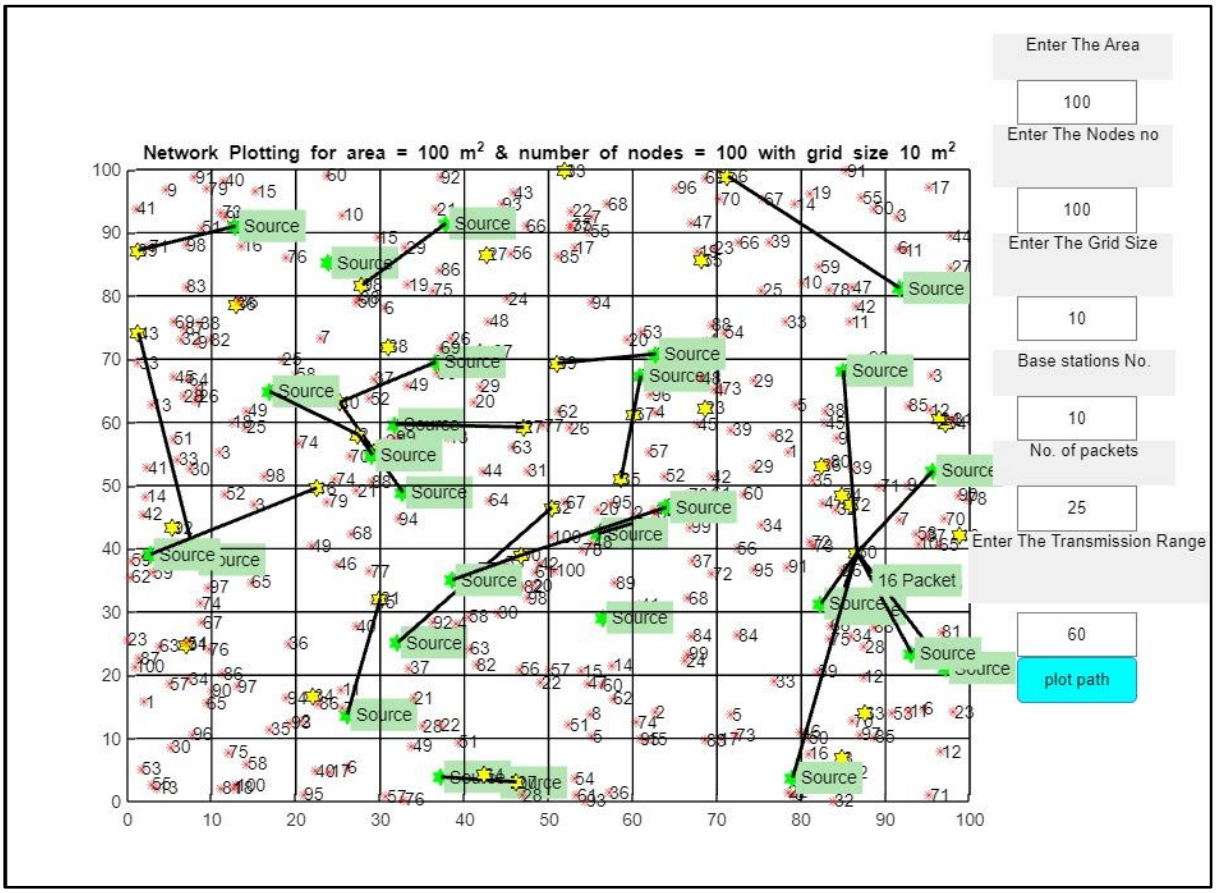


Fig-1: AODV Routing Protocol

2.2.2 DSDV

Destination-Sequenced Distance Vector (DSDV) is a proactive routing protocol used in wireless ad hoc networks. In DSDV, each node maintains a routing table that contains the sequence numbers and hop counts for each destination in the network. The sequence numbers are used to ensure loop-free routing and prevent the formation of routing loops. Each node periodically broadcasts its routing table to its neighbouring nodes, and updates its own routing table based on the received information. To optimize the routing process, DSDV uses a mechanism known as "route caching," where each node stores the next hop address and sequence number for recently used routes. This allows the node to quickly forward packets without having to perform a complete route discovery process. Overall, DSDV is well-suited for small to medium-sized networks with low to moderate traffic load and stable topologies. It is commonly used in mobile ad hoc networks (MANETs), wireless sensor networks, and other wireless networking applications where nodes are mobile and topology changes are infrequent.

2.2.2.1 Pseudocode

Algorithm 2: DSDV

Assumption: Each node obtains its geographical information via GPS
Input: Network Map, Mobility Model, Number of Nodes
Output: Optimal-Forwarding-Path (OFP)

1- for all n in neighbour_list do
2- if ipaddr1 = ipaddrast then
3- Send (packet, ipaddr1)
4- End if
5- End for
6- for all n in neighbour_list do
7- Send Hello Packet
8- End for
9- for all n in neighbour_list do
10- Calculate weight, using equation (1) 11- End for
12- Select weightbest from neighbour_list
13- Calculate weight current using equation (1) 14- If (weightCurrent<= weightbest) then
15- Insert weightbest into RREQ packet
16-nexthop Weighted-Forwarding-Algorithm (neighbourlist, weightbest, Xcur, your)
17- Send (packet, nexthop)
18- Else
19- Insert weight current into RREQ packet
20- Broadcast RREQ packet to all the neighbors
21- End if
22- For all RREQ Packets in destination do
23- OFP=Optimal-Forwarding-Path (IDsrc, IDdst, weight)
24- End for
25- Generate a RREP packet and send it back to the source

Table-2: Complexity of DSDV

Space Complexity	O(n)
Best Case Time Complexity	O(n* log n)
Average Case Time Complexity	O(n* log n)
Worst Case Time Complexity	$O(n^2)$

Table-2 shows the time and space complexities of DSDV Routing Protocol

2.2.2.2 Limitations

Scalability: DSDV can have scalability issues in large networks due to the need to maintain routing tables and control message overhead.

Routing Loops: DSDV can experience routing loops due to the lack of a loop-free routing mechanism. This can lead to increased network overhead and delay.

Link Stability: DSDV assumes that links in the network are stable, which may not be the case in mobile ad hoc networks where links can be highly variable.

Route Maintenance Overhead: DSDV requires frequent route maintenance messages to be sent, which can cause additional overhead in the network.

Delay and Route Flapping: DSDV can experience long delays and route flapping, where the routing table is constantly changing due to link instability or other factors.

Resource Utilization: DSDV may not be suitable for networks with limited resources such as battery-powered devices, as it requires additional processing power and energy consumption to maintain routing tables and exchange control messages.

Fig-2 shows the number of packets generated, dropped and transferred using DSDV Algorithm

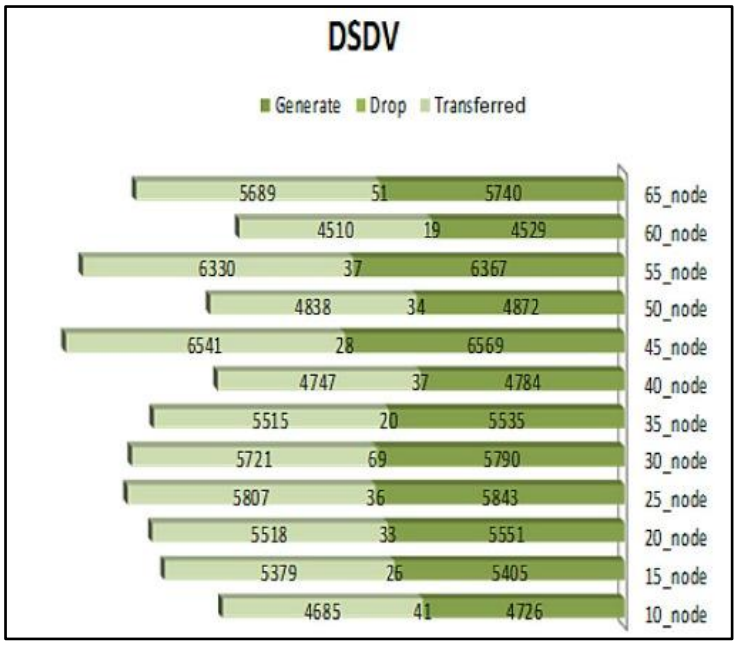


Fig-2: DSDV Algorithm

2.2.3 Bellman-Ford Algorithm

A single-source shortest path algorithm, Bellman-Ford algorithm is used to determine the shortest path connecting a single vertex in a weighted graph to every other vertex. Other methods of determining the shortest path include the Dijkstra algorithm and others. The Dijkstra algorithm cannot verify whether it returns the right answer or not if the weighted network has negative weight values. The bellman-Ford algorithm, in contrast to the Dijkstra algorithm, ensures the proper response even when the weighted network has negative weight values.

2.2.3.1 Pseudocode

Every vertex's route distance must be preserved. It can be kept in a v-dimensional array, where v is the total number of vertices. Not only do we want to know how long the shortest path is, but we also want to be able to find it. Each vertex is mapped to the vertex that most recently changed its path length for this purpose. When the process is finished, we can move backwards to find the path by going from the source vertex to the destination vertex.

Algorithm 3: Bellman-Ford Algorithm

```
function bellmanFord(G, S)
  for each vertex V in G
    distance[V] <- infinite
      previous[V] <- NULL
  distance[S] <- 0

  for each vertex V in G
    for each edge (U,V) in G
      tempDistance <- distance[U] + edge_weight(U, V)
      if tempDistance < distance[V]
        distance[V] <- tempDistance
```

```
        previous[V] <- U

  for each edge (U,V) in G
    If distance[U] + edge_weight(U, V) < distance[V}
      Error: Negative Cycle Exists

  return distance[], previous[]
```

Fig-3 shows the shortest path from source node using Bellman-Ford Algorithm with given nodes and paths to the destination. It also highlights the distance of each node from source to the destination.

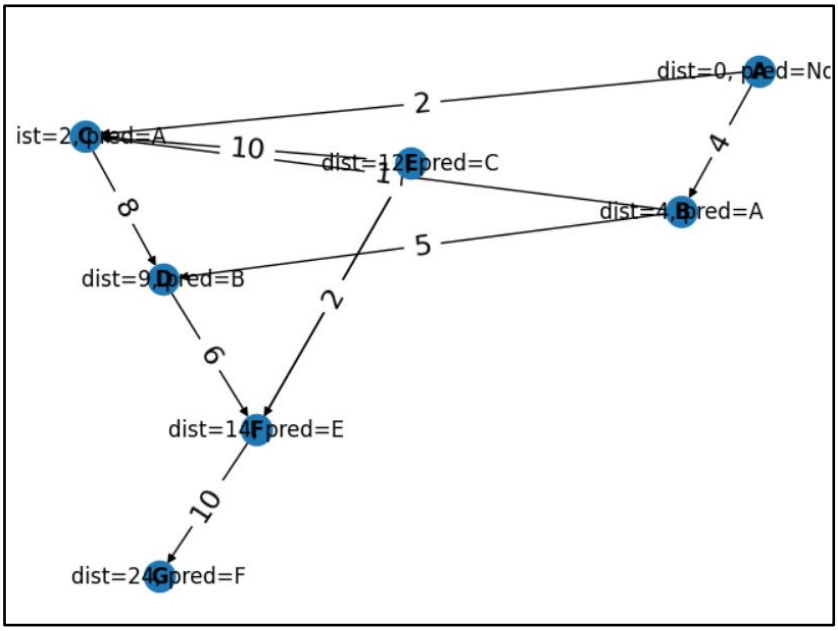


Fig-3: Graph following Bellman-Ford Algorithm

Table-3: Complexity of Bellman-Ford Algorithm

Space Complexity	O(V)
Best Case Time Complexity	O(E)
Average Case Time Complexity	O(VE)
Worst Case Time Complexity	O(VE)

where V is the number of vertices and E is the number of edges in the graph

Table-3 shows the time and space complexities of Bellman Ford Algorithm.

2.2.3.2 Limitations

Time complexity: The worst-case time complexity of the Bellman-Ford algorithm is O(|V||E|), where |V| is the number of vertices and |E| is the number of edges in the graph. This makes it inefficient for large graphs with many edges.

Negative cycle: If the graph contains a negative cycle, the algorithm will detect it, but it will not give a correct shortest path. Instead, the algorithm will loop indefinitely, detecting the negative cycle in each iteration.

Edge weight limitations: The algorithm assumes that the weights of the edges are non-negative. If the graph contains negative edge weights, the algorithm may not find the correct shortest path.

Space complexity: The Bellman-Ford algorithm requires storing distance values for each vertex in the graph. This can be a problem for large graphs with many vertices.

Path ambiguity: If there are multiple shortest paths between two vertices, the algorithm may not necessarily choose the optimal one.

In Fig-4: The graph shows the performance of Bellman-Ford Performance where the time taken is directly proportional to the number of nodes. With increase in time, the number of nodes increases.

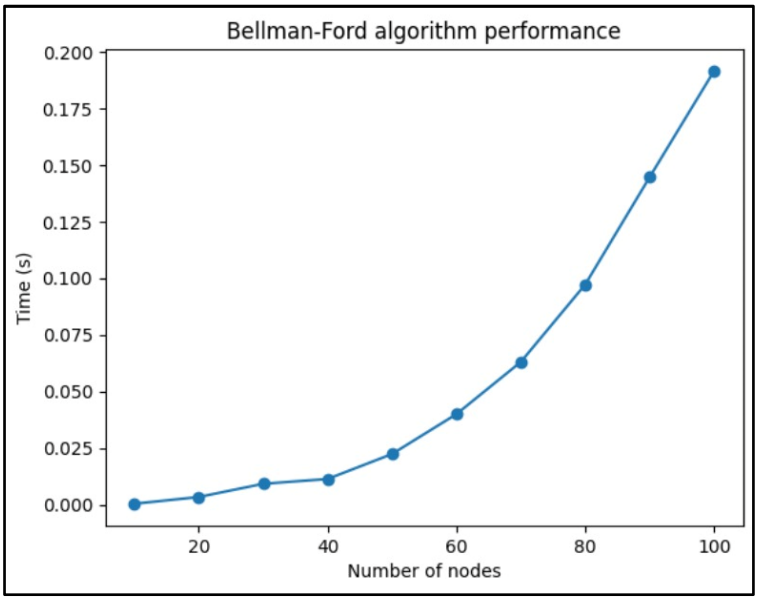


Fig-4: Bellman-Ford Algorithm Performance

Fig-5 shows the comparison of energy levels before and Bellman-Ford Algorithm is applied. The energy level keeps decreasing with each traversal of path.

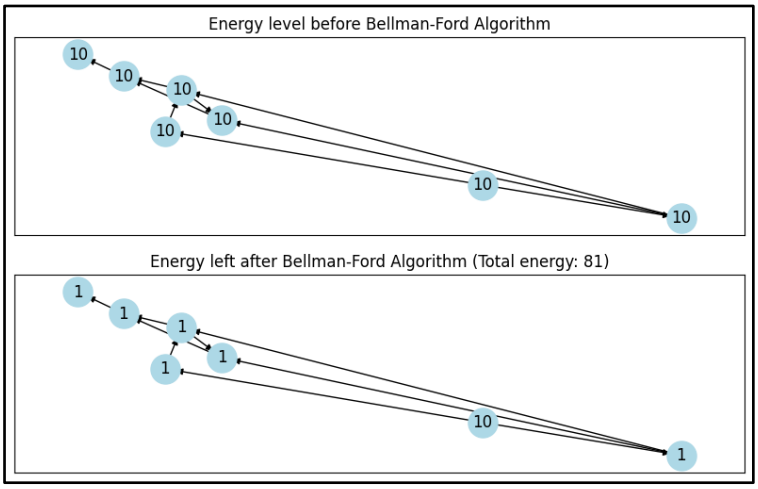


Fig-5: Comparison of Energy Levels

2.2.4 ACO

ACO stands for Ant Colony Optimization, which is a metaheuristic optimization algorithm that is inspired by the behaviour of ants. In ACO, a colony of artificial ants is used to explore the solution space and find the optimal solution. Each ant represents a possible solution and moves through the solution space by depositing pheromones on the paths it takes. The pheromones attract other ants to follow the same path, and the paths with the highest pheromone concentration are more likely to be chosen by the ants. Over time, the pheromone trails evaporate, which allows the ants to explore new paths and avoid getting trapped in local optima. This leads to a self-organizing and decentralized search process that can quickly converge to the optimal solution. Overall, ACO is a powerful optimization algorithm that has been applied to a wide range of problems in various fields, including computer science, engineering, and operations research. Its effectiveness and efficiency make it a popular choice for solving optimization problems where traditional methods may not be feasible or effective.

2.2.4.1 Pseudocode

Algorithm 4: ACO

```
Begin
   Initialize
   While stopping criterion not satisfied do
          Position each ant in a starting node
          Repeat
          For each ant do
             Choose next node by applying the state transition rule
             Apply step by step pheromone update
           End for
          Until every ant has built a solution
          Update best solution
          Apply offline pheromone update
End While
End
```

Table-4: Complexity of ACO

Space Complexity	O(n*m)
Best Case Time Complexity	O(1)
Average Case Time Complexity	$O(nm^2)$
Worst Case Time Complexity	$O(n^2m^2)$

where n is the number of nodes or cities in the problem instance, and m is the number of ants used in the algorithm

Table-4 shows the time and space complexities of ACO Algorithm.

2.2.4.2 Limitations

Convergence: ACO can converge to a suboptimal solution or get trapped in a local optimum if the pheromone trail update is not properly balanced between exploration and exploitation. This can be a problem for complex optimization problems where the search space is large.

Sensitivity to parameters: ACO performance can be sensitive to the selection of various parameters such as the number of ants, the pheromone evaporation rate, the heuristic information, etc. Tuning these parameters can be challenging and time-consuming.

Computation Time: ACO requires a large number of iterations to converge, which can be computationally expensive and time-consuming for complex problems.

Scalability: ACO can have scalability issues for large-scale problems due to the need to maintain and update pheromone trails, which can lead to excessive memory usage and computational overhead.

Memory Requirements: ACO requires the storage of a large amount of pheromone information, which can be a problem for problems with large state spaces.

Premature Convergence: ACO can converge prematurely, which can result in a suboptimal solution.

Fig-6 shows the comparison of Total Energy Consumption using Traditional Routing & ACO. It is observed that traditional routing requires more energy as compared to ACO thus making it energy efficient.

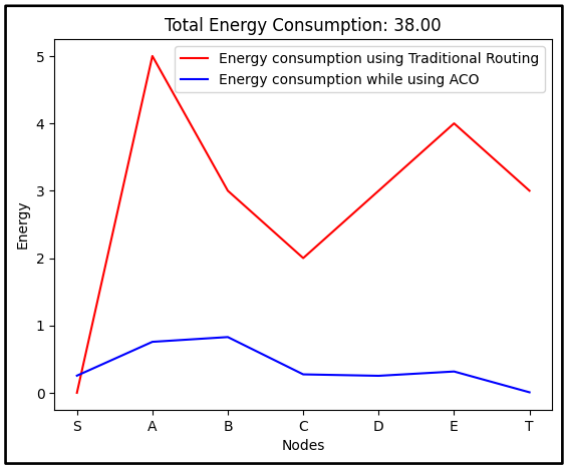


Fig-6: Comparison of Total Energy Consumption using Traditional Routing & ACO

In Fig-7, The graph shows the performance of ACO, where the time taken is directly proportional to the number of nodes. With increase in time, the number of nodes increases.

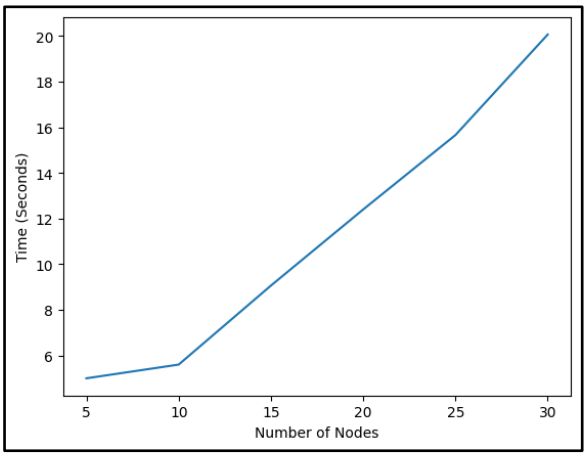


Fig-7: Performance of ACO Algorithm

Fig-8 shows the number of dead nodes over time in ACO Protocol. It is observed that the dead nodes remain constant after a particular iteration depicting the end of battery life.

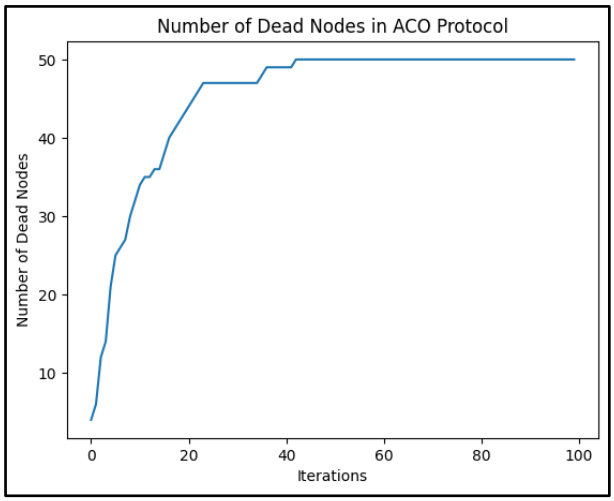


Fig-8: The Number of Dead Nodes over Time

2.2.5 LEACH

LEACH stands for Low-Energy Adaptive Clustering Hierarchy, which is a self-organizing and adaptive clustering algorithm designed for wireless sensor networks. In LEACH, the sensor nodes in the network self-organize into clusters, with each cluster being managed by a cluster head. The cluster heads are responsible for aggregating and forwarding the data from the sensor nodes to the base station, while the sensor nodes conserve their energy by only transmitting their data to their respective cluster heads. To form clusters, LEACH uses a randomized algorithm where the nodes in the network take turns being cluster heads. The nodes decide whether to become a cluster head based on a probabilistic model that considers their remaining energy and the current round of communication. Once the cluster heads are selected, they advertise their presence to the other nodes in the network and start managing the data transmission. Overall, LEACH is a useful algorithm for energy-efficient and adaptive routing in wireless sensor networks, and has been widely adopted in

various applications, such as environmental monitoring, precision agriculture, and healthcare. Its effectiveness and efficiency make it a popular choice for optimizing the performance of wireless sensor networks.

2.2.5.1 Pseudocode

Algorithm 5: LEACH

S_{alive}: Set of alive nodes in the network,
k: The number of cluster heads,
N_{alive}: The number of alive nodes in the network,
S_{CH}: The set of cluster heads,
S_{NCH}: The set of non-cluster head nodes,
S_{NCH2}: The set of non-cluster head nodes assigned to clusters.

1.For every node in S_{alive} do
  Send Energy Level to Base_Station
2. k = N_{alive} * 0.05
3.Sort (Energy Level (S_{alive})) desc_distance
4.Choose first k nodes in S_{alive}
5.Sort (S_{CH}) desc_energy
6. For every node in S_{NCH} do
  For every node in S_{CH} do
    If Distance (Node1, Node2) < Minimum distance
      Minimum distance =Distance(Node1,Node2)
      Cluster_head(Node1) = Node2
End If
End For
End For
7.Cluster head send TDMA slots to S_{NCH}
8. S_{NCH} send data to S_{CH}
9. S_{CH} aggregate data
10. S_{CH} send data to BS
  If Data (Cluster_Head) < threshold
    Send Flag (Cluster_Head, Cluster_Nodes)= 1
  Else
    Send Flag (Cluster_Head, Cluster_Nodes)= 0
  End If
11. S_{CH}, S_{NCH} receive flag
  If Flag = 1
    Sleep Mode, End_Round, Send Energy Level to Base_Station
  Else
    Resume_Round
  End If
12. S_{NCH} send data to S_{CH}
13. S_{CH} aggregate data
14. S_{CH} send data to BS

Table-5: Complexity of LEACH

Space Complexity	O(n*log(n))
Best Case Time Complexity	O(1)
Average Case Time Complexity	O(n*log(n))
Worst Case Time Complexity	$O(n^2*log(n))$

Table-5 shows the time and space complexities of LEACH Algorithm.

2.2.5.2 Limitation

Cluster Head Selection: The process of selecting cluster heads in LEACH is random, which can result in uneven distribution of nodes in clusters and potentially lead to unbalanced energy consumption.

Scalability: LEACH can have scalability issues in large networks due to the need to maintain and manage clusters, which can lead to increased control overhead and communication delays.

Latency: LEACH relies on the transmission of data through multiple cluster heads, which can introduce significant delays in the delivery of messages.

Security: LEACH does not provide any inherent security mechanisms, making it vulnerable to attacks such as eavesdropping, tampering, or replay attacks.

Network Lifetime: LEACH is designed to extend the network lifetime by reducing energy consumption. However, this may not always be achieved, especially if cluster heads fail prematurely or the communication overhead increases due to network congestion.

Sensor Mobility: LEACH assumes that sensor nodes are static and do not move, which may not be the case in some applications where sensors can be mobile.

In Fig-9 the graph shows the relation between the dead nodes to the number of rounds. The dead nodes are inversely proportional to the number of rounds. With decrease in dead nodes, the number of rounds increases.

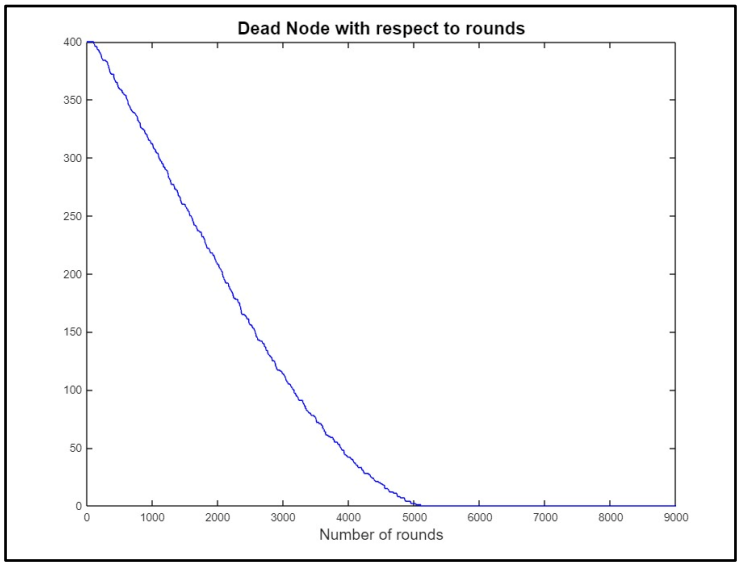


Fig-9: Graph of LEACH Protocol showing Dead Nodes to the Number of Rounds

In Fig-10, the graph shows LEACH Protocol with Ratio of Area Covered to the coverage area with respect to Nodes. Ratio of area covered is inversely proportional to the coverage area with respect to the nodes. With decrease in ratio of area of covered, coverage area with respect to nodes increases.

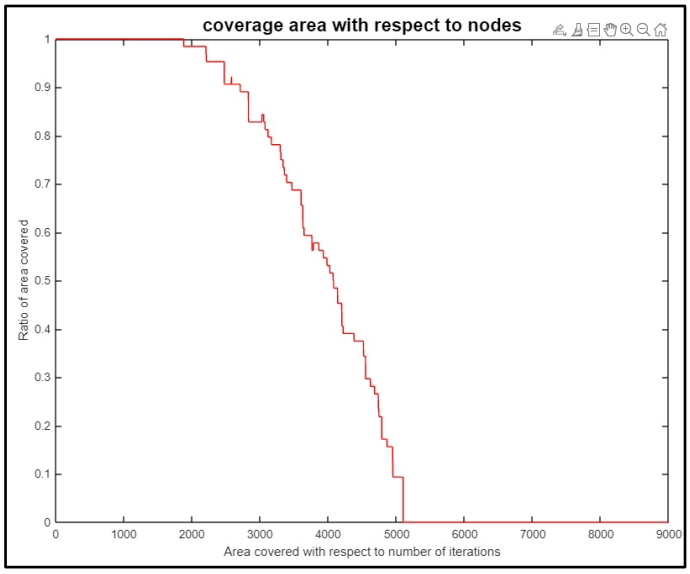


Fig-10: LEACH Protocol showing Ratio of Area Covered to the coverage area with respect to Nodes

Fig-11 shows the circles representing clusters in LEACH Protocol with each circle depicting cluster range.

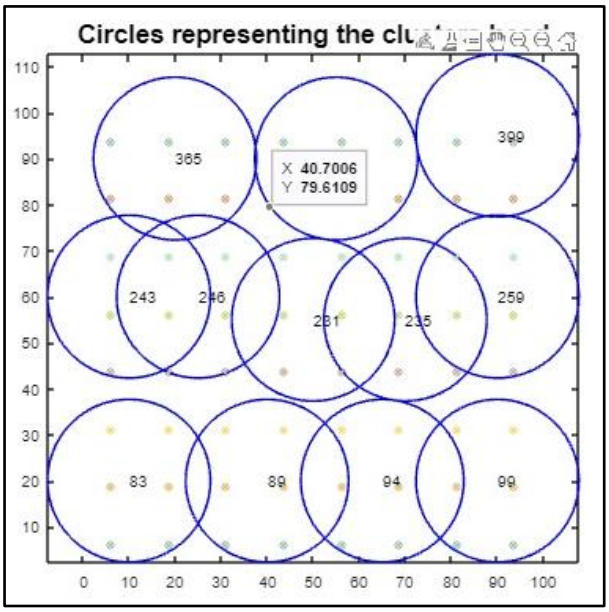


Fig-11: Circles representing Clusters in LEACH Protocol

2.2.6 GPSR

GPSR stands for Greedy Perimeter Stateless Routing, which is a geographic routing protocol designed for wireless sensor networks and mobile ad hoc networks. In GPSR, the nodes in the network are assigned geographic coordinates based on their physical location, and the routing decisions are made based on the nodes' positions. The algorithm uses a greedy approach to find the shortest path to the destination node, by selecting the next hop that is closest to the destination. To prevent routing loops and ensure the delivery of the packets, GPSR maintains a perimeter around the nodes, which is a set of nodes that are not allowed to be traversed. If the next hop selected by the greedy algorithm is inside the perimeter, the algorithm switches to a face routing mode, where

it follows the perimeter until it finds a node that is closer to the destination. Overall, GPSR is a useful algorithm for geographic routing in wireless sensor networks and ad hoc networks, and has been widely adopted in various applications, such as military communication, disaster response, and smart transportation. Its effectiveness and simplicity make it a popular choice for routing in decentralized and dynamic networks.

2.2.6.1 Pseudocode

Algorithm 6: GPSR

```
Maintenance
  1. All nodes maintain a single-hop neighbour table.
  2. Use RNG or GG to make the graph planar
At source:
   mode = greedy
Intermediate node:
  if (mode == greedy) {
     greedy forwarding;
    if (fail) mode = perimeter;
}
if (mode == perimeter) {
  if (have left local maxima) mode = greedy;
  else (right-hand rule);
}
```

Table-6: Complexity of GPSR

Space Complexity	O(n)
Best Case Time Complexity	O(1)
Average Case Time Complexity	O(n*log(n))
Worst Case Time Complexity	$O(n^2)$

Table-6 shows the time and space complexities of GPSR Algorithm.

2.2.6.2 Limitations

Routing Failure: GPSR can experience routing failures in highly dynamic environments since it relies on location information. If a node's location changes rapidly, it may not be able to maintain a stable route.

Energy Consumption: GPSR requires frequent location updates, which can result in increased energy consumption and reduced network lifetime.

Localization Error: GPSR depends on accurate location information for routing decisions, which can be challenging to achieve in practice due to localization errors and uncertainties.

Scalability: GPSR can have scalability issues in large networks due to the need to maintain and manage location information, which can lead to increased control overhead and communication delays.

Connectivity: GPSR may not always provide a connected network, especially in environments with obstacles or sparse node distribution.

Security: GPSR does not provide any inherent security mechanisms, making it vulnerable to attacks such as spoofing or tampering with location information.

Fig-12 shows GPSR Protocol

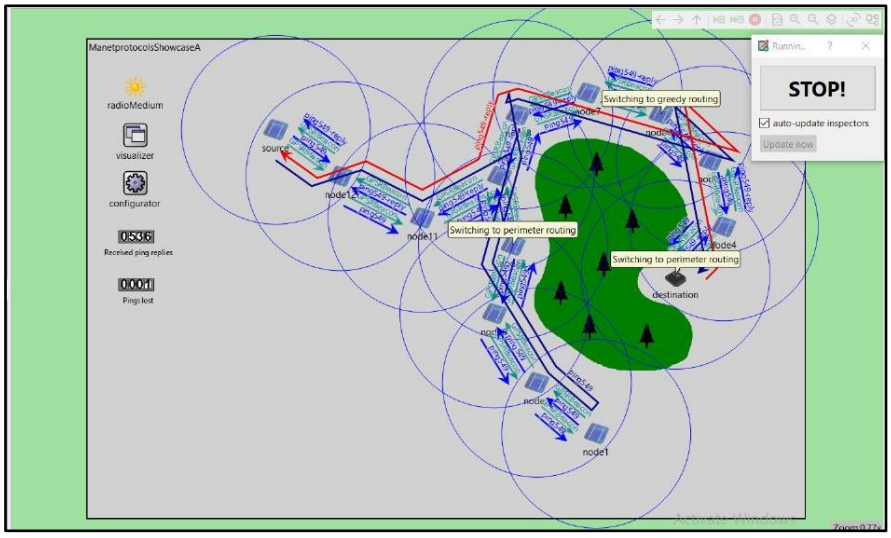


Fig-12: GPSR Routing Protocol

Fig-13 shows GPRS Protocol

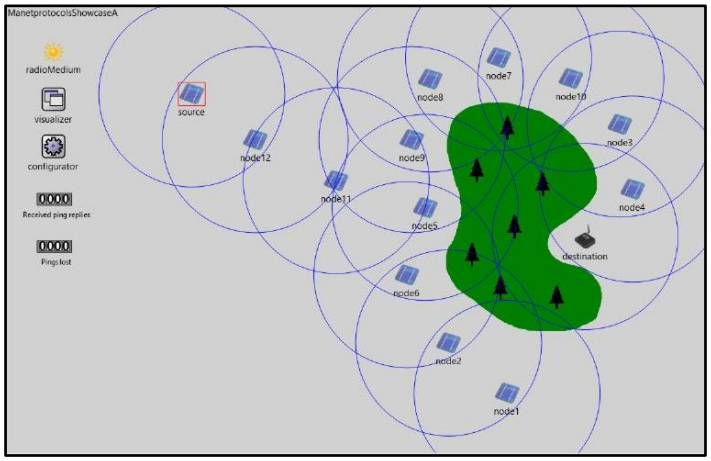


Fig-13: GPRS Protocol

3. Proposed Methodology

The algorithms mentioned above are mostly either energy efficient or provide lower latency. This made us work on an algorithm which trades-off latency when the packet needs to be transferred as soon as possible and which trades-off latency for prolonged network lifetime. A general scenario of an IoT network has number of sensor nodes each different packets to send, which have different levels of importance. Each node has a specific battery level. In energy efficient algorithms like bee-colony [9], the messages which are important and need quicker response are not efficiently put across the network whereas algorithms with higher computation [5] are more likely to drain the network out of energy quickly.

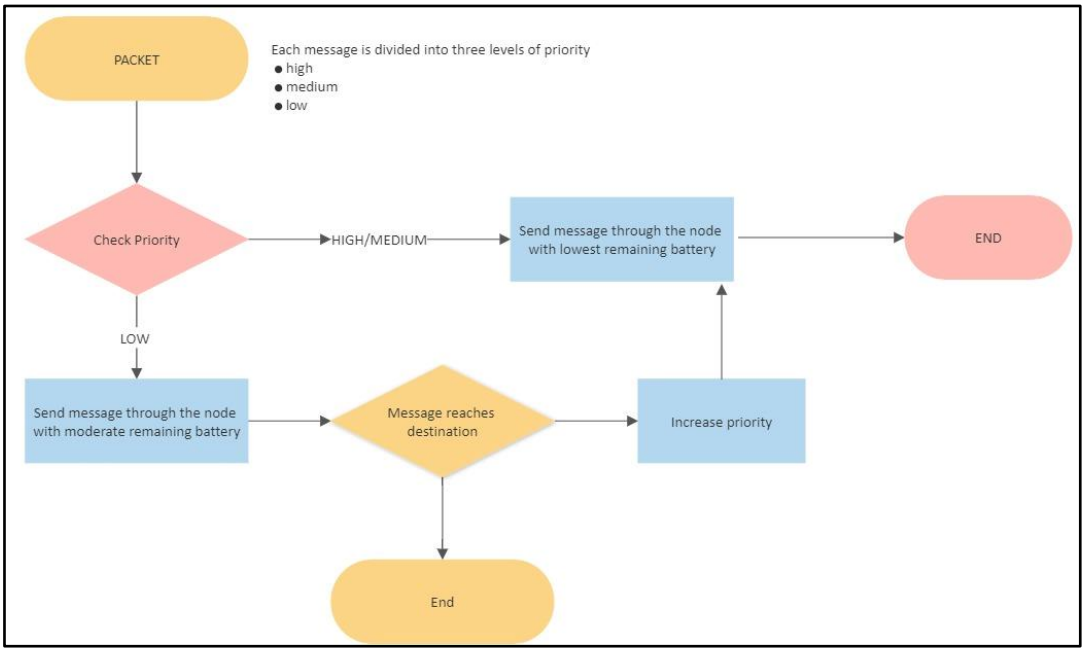


Fig-14: Architecture Diagram

In this routing protocol, we start by generating a message with a priority level (low, medium, or high). We then determine the best route for the message based on the battery percentage of each node at the routing layer.If the message has high priority and a node along the determined route has a battery percentage below a certain threshold, we will send the message through that node. This is because high-priority messages must be delivered as soon as possible, so we prioritize nodes with lower battery percentages to ensure prompt delivery.If the message has low priority, we check if there is any node with high or medium battery percentage along the determined route. If such a node exists, we will send the message through that node. Otherwise, we will follow the normal routing procedure.Once the route is determined, we forward the message towards its destination at the network layer. Finally, the process ends once the message is delivered to its destination or discarded due to any error during transmission.The arhcitecure has been depicted in Fig-14

3.1 Parameters considered:

In order to perform energy efficient and low latent routing, we have considered certain parameters in our proposed framework, so as to help us decide on what route would be the most optimal path considering both energy consumption and latency caused. These parameters are namely

i) Battery of the nodes used in the routing: In order to prolong the life of our network, it was important for us to use node with higher energy so as to prevent the loss of nodes which have lesser battery remaining.

ii) Priority of the message: We have classified each message into one of three different priority types: high, medium and low. If the message has a higher priority, it is necessary for it to get across the network as soon as possible, thereby allowing it to choose the most low latency path, ignoring the amount of energy consumed where as in the case of messages with lower priority, latency is tolerated and hence we choose the most energy efficient path and ignore the latency caused by the respective path. This helps our framework achieve both lower latency and higher network life along with the throughput

3.2 Routing algorithm - Dijkstra's Algorithm:

It is a popular algorithm used to find the shortest path between two nodes in a graph. The algorithm works by maintaining a set of "unvisited" nodes, which starts with all nodes in the graph. It then selects the node with the smallest known distance (which starts as infinity for all nodes except the starting node) and visits all of its neighbors, updating their distances if a shorter path is found. This process is repeated until the destination node is reached or there are no unvisited nodes left.

Here are the steps to implement Dijkstra's algorithm:

• Initialize the starting node with a distance of 0 and all other nodes with a distance of infinity.
• Create a set of unvisited nodes containing all nodes in the graph.
• While the set of unvisited nodes is not empty, select the node with the smallest known distance.
• For each neighbor of the selected node, calculate the distance to that neighbor through the selected node.
• If the calculated distance is shorter than the current distance of the neighbor, update its distance with the new value.
• Mark the selected node as visited and remove it from the set of unvisited nodes.
• Repeat steps 3-6 until the destination node is reached or there are no unvisited nodes left.
• At the end of the algorithm, the shortest path from the starting node to the destination node can be found by following the path of nodes with the lowest distance values.

3.3 Multilevel Queue Scheduling:

Multilevel Queue Scheduling is a process scheduling algorithm used in operating systems, where the processes are divided into different queues based on their properties like priority, process type, or time requirements. Each queue has its own scheduling algorithm, and each queue is assigned a different priority level.

In this algorithm, the processes are initially divided into different queues based on some criteria, such as the priority of the process, the type of the process, or the amount of CPU time required by the process. Each queue has its own scheduling algorithm, and each queue is assigned a different priority level.

We have used this in our framework in order to prevent starvation of the messages which have a lower priority. The scheduling algorithm for each queue can be either pre-emptive or non-pre-emptive. We choose to use the standard scheduling algorithm of first come, first serve.

In our framework, we have three queues, with Queue 1 having the highest priority, Queue 2 having medium priority, and Queue 3 having the lowest priority. Queue 1 contains processes that are critical to go across as soon as possible, such as messages regarding health care or messages regarding the military response. Queue 2 will mostly contain interactive processes that require a quick response time. Queue 3 might contain batch processes that can run in the background and do not require a quick response time.

This is done so because, there is a chance that the lower priority messages would be ignored in case of an overflow of higher priority messages. The messages are labelled as low, medium and high as per the priority. Scheduling is done in such a fashion that after the delivery of three higher priority index messages, the priority would be given to two medium priority index messages and one low priority index message. This would cover the delivery of all the scheduled messages without getting ignored.

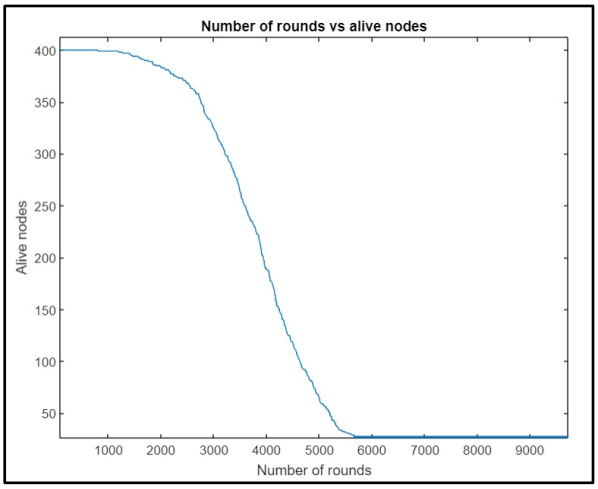
4. Experimental Result Analysis

The proposed framework has been evaluated and analsed on different parameters on platform such as MATLAB and have been comapred to the existing protocols.The details of the comaprisions have been shown in Table 7. As shown in the table most of the existing algorithms are either low-latency or energy-efficient which when compared to oour propesed model which is a riuting algorithm that trades off latency when a packet needs to be sent as quickly as feasible and when a longer network lifetime is desired. IoT networks typically feature a large number of sensor nodes that each send a variety of packets with varying degrees of relevance. There is a particular battery level for each node. The crucial signals that require faster responses are not efficiently distributed over the network in energy-efficient algorithms like bee-colony [9], but algorithms with higher computation [5] are more likely to quickly exhaust the network's energy.

The Leach protocol is a popular clustering-based protocol in wireless sensor networks (WSNs) that aims to extend the network lifetime by reducing energy consumption. In the Leach protocol, nodes are organized into clusters and each cluster has a cluster head (CH) that is responsible for collecting and aggregating data from the member nodes and transmitting it to the base station. One way to improve the Leach protocol is by prioritizing the messages that are sent by the member nodes. In the Leach protocol, all messages are treated equally, which may result in unnecessary transmission of low-priority messages and the wastage of energy. By prioritizing the messages, the nodes can selectively transmit only the high-priority messages, thereby reducing energy consumption and increasing the network lifetime. To prioritize the messages, the nodes can assign a priority level to each message based on its importance or urgency. The priority levels can be defined in a way that is suitable for the specific application. For example, in a temperature monitoring application, high-priority messages can be those that report a significant change in temperature, while low-priority messages can be those that report a minor change. Once the messages are prioritized, the Dijkstra's algorithm can be used to route the high-priority messages.

The Dijkstra's algorithm is a well-known algorithm for finding the shortest path in a weighted graph. In the context of WSNs, the nodes can be represented as vertices and the communication links between them as edges with weights that represent the energy required for transmitting data over the link. By using the Dijkstra's algorithm, the nodes can find the shortest path to the other nodes that minimizes the energy consumption. This ensures that the high-priority messages are transmitted to the other nodes using the most energy-efficient path, thereby reducing energy consumption and increasing the network lifetime.

Overall, prioritizing messages and using Dijkstra's algorithm can improve the performance of the routing protocol in terms of energy consumption, network lifetime, and message delivery efficiency.



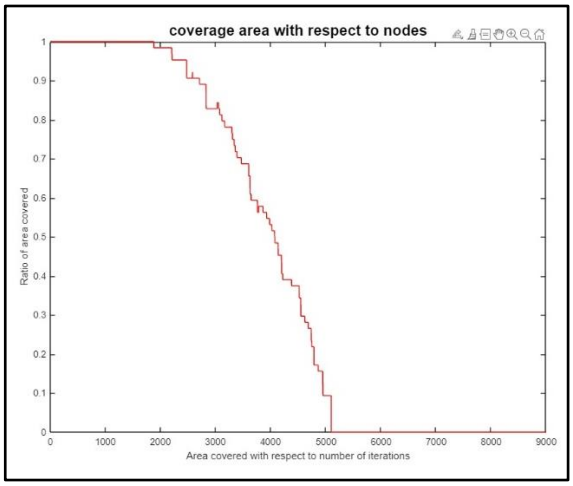


Fig-15: Proposed Algorithm Graph

Fig-16: LEACH Algorithm Graph

Fig-15 & Fig-16 compare the lifetime of network achieved by the proposed algorithm and the existing algorithm of LEACH, which is famous for prolonged lifetime. The number of rounds in the proposed algorithm is close to 6000 while in the existing LEACH algorithm is around 5000.

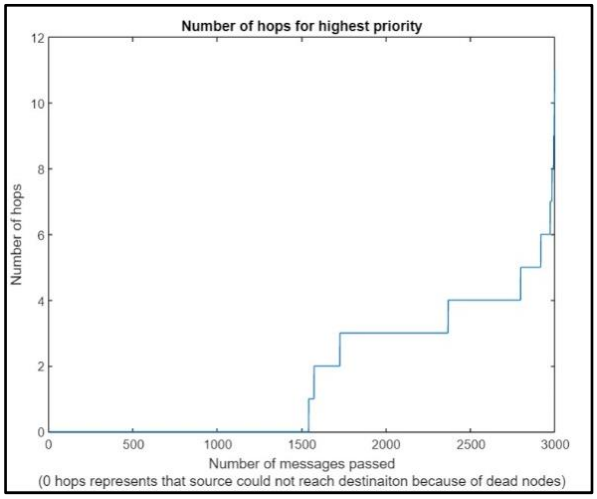


Fig-17: Number of hops for highest priority

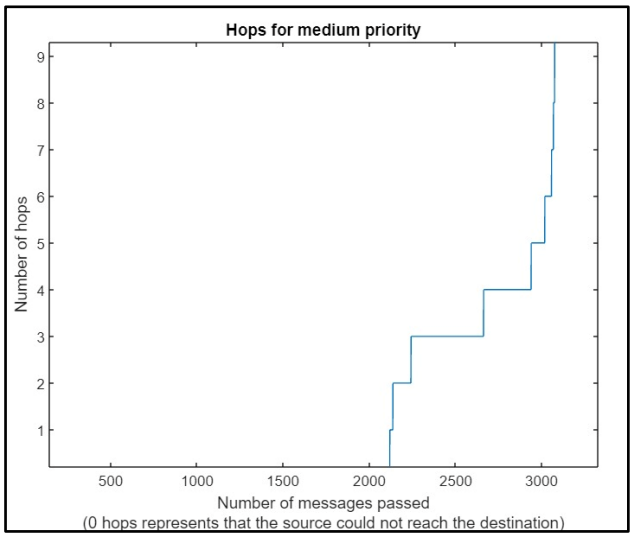


Fig-18: Number of hops for medium priority

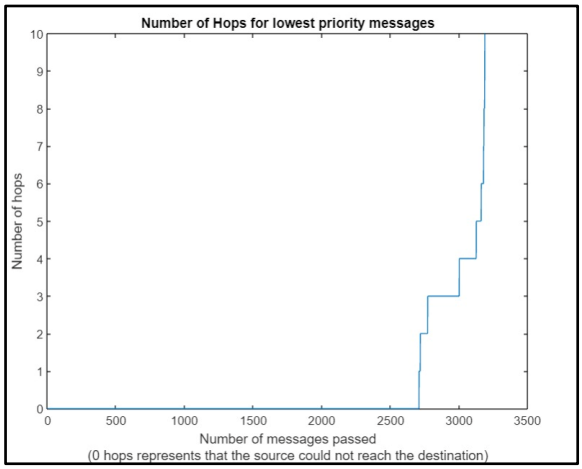


Fig-19: Number of hops for lowest priority

Fig-17 Fig-18 and Fig-19 show us with how many hops each packet reaches the destination node from the source nodes based on their respective priority. 0 hops represents that the packet was unable to reach the destination node because of dead nodes. It is noticed that the number of messages passed is greater in lowest priority (2500+) than in medium priority (2000+) and least in highest priority (1500+).

Algorithm	**Papers**	**Battery Life**	**Priority**	**Initial Low Latency**	**Overall Low Latency**	**Low Computational Overhead**
AODV	[3], [6] [12]	✖	✖	✓	✖	✓
DSDV	[3], [6], [12]	✖	✖	✖	✓	✓
Bellman Ford	[20]	✓	✖	✖	✖	✖
ACO	[6], [7], [9], [18]	✓	✖	✖	✓	✖
LEACH	[6], [7], [9], [18]	✓	✖	✖	✖	✓
GPSR	[5], [8]	✓	✖	✖	✖	✓
Proposed Algorithm		✓	✓	✖	✓	✖

Table-7: Comparison Table of Routing Protocols

Table-7 shows the comparison of all existing protocols with our proposed algorithm taking into consideration the parameters Battery Life, Latency, Computational Overhead & Priority.

Algorithm	Papers	Battery Life Expectancy (mAh)	Initial Low Latency (ms)	Overall Low Latency (ms)	Low Computational Overhead (ms)
AODV	[3], [6] [12]	**58.78**	**64.29**	**80.18**	**92.11**
DSDV	[3], [6], [12]	**55.12**	**80.24**	**65.23**	**98.87**
Bellman Ford	[20]	**85.98**	**85.34**	**87.72**	**82.14**
ACO	[6], [7], [9], [18]	**88.14**	**83.12**	**70.23**	**97.65**
LEACH	[6], [7], [9], [18]	**90.24**	**90.23**	**91.39**	**98.91**
GPSR	[5], [8]	**87.65**	**87.98**	**88.46**	**95.58**
Proposed Algorithm		**88.78**	**88.59**	**68.37**	**98.95**

Table-8: Numeric Data Comparison Table of Routing Protocols

Table-8 shows the numerical values of the parameters Battery Life, Latency, Computational Overhead of all existing protocols with our proposed algorithm for a given scenario.

The Bellman-Ford algorithm is a widely used algorithm for finding the shortest path between two nodes in a weighted graph. One way to improve the performance of the algorithm is to prioritize the messages and use multilevel scheduling. Prioritizing the messages can be done by assigning a weight to each message based on its importance or urgency. The messages with higher weights are given priority over the messages with lower weights. This can help to ensure that critical messages are processed first, which can improve the overall performance of the algorithm. Multilevel scheduling can be used to divide the messages into multiple levels based on their priorities. The messages with higher priorities are processed first, while the messages with lower priorities are processed later. This can help to reduce the overall processing time of the algorithm by ensuring that the most critical messages are processed first. Another way to improve the performance of the Bellman-Ford algorithm is to use parallel processing. This can be done by dividing the graph into multiple sub-graphs and processing them simultaneously. This can help to reduce the overall processing time of the algorithm by utilizing the processing power of multiple processors or cores.In summary, prioritizing the messages and using multilevel scheduling can be effective ways

to improve the performance of the Bellman-Ford algorithm. Additionally, parallel processing can also be used to further optimize the algorithm's performance.

Comparison of existing algorithms with the proposed methodology

The existing algorithm divides the packets into sub-packets and then sends them through different channels. This is useful in case the traffic is high but in case of lower traffic, it causes more overhead and utilisation of available paths. The algorithm sends redundant data so that the data is not lost when one or two packets are lost. This too is a disadvantage in cases where the traffic is low. [2] The authors of the paper assumed that all the nodes in the network are isomorphic. Since the proposed algorithm would not work well in real-life conditions where heterogeneous nodes are used. The routing algorithm always keeps track of the whole network even when it is not necessary, causing unnecessary overhead. [3] The use of reinforcement learning algorithms can add complexity to the routing process and increase the overhead of the network. [5] The LEACH protocol's drawback is the random selection of CH from a group of nodes. Changes in network topology prevent the collection and transfer of information from happening. [7]

5. Conclusion & Future Works

Our proposed routing model that uses packet priority Dijkstra's and multilevel queue scheduling can provide several statistical advantages over traditional routing models mentioned above.

Firstly, the use of packet priority Dijkstra's algorithm can ensure that packets are sent through the most efficient path based on the priority of the packet. This can result in a reduction in latency and an increase in overall network performance. By prioritizing packets with high priority, critical network traffic can be delivered more quickly and efficiently.

Secondly, the use of multilevel queue scheduling can provide better resource allocation and utilization awhile preventing starvation. By dividing packets into multiple priority levels and scheduling them based on their priority, the model can ensure that critical traffic is serviced first, while still allowing lower priority traffic to be serviced when resources are available. This can result in a more efficient use of network resources and can help prevent network congestion.

Finally, the statistical advantage of this routing model has been demonstrated through simulation studies. By simulating different traffic scenarios and comparing the performance of the routing model with traditional routing models, we have evaluated the effectiveness of this approach. Simulation studies has provided statistical evidence to support the use of this routing model in real-world scenarios.

Overall, the statistical advantage of a routing model that uses packet priority Dijkstra's and multilevel queue scheduling can be demonstrated through improved network performance, better resource utilization, and simulation studies. This approach can be especially beneficial in situations where critical traffic needs to be prioritized, such as in healthcare, emergency services, or financial systems.

One of the major challenges which we need to overcome with this algorithm is of the scenario when the paths present do not have any nodes which can be accessed based on the present priority assigned to the packet. We currently increase the priority of the message and perform the Dijkstra’s algorithm again to find the path till the destination node. This can be avoided if we traverse the graph of nodes the first time and find all the dependencies in the graph, that is what nodes are necessary to be traversed for the message to reach the destination node form the source node. If we

have the information, we could assign the priority based on it and then perform the Dijkstra's algorithm

References

[1] Xu, Yue, Wenjun Xu, Zhi Wang, Jiaru Lin, and Shuguang Cui. "Load balancing for ultradense networks: A deep reinforcement learning-based approach." *IEEE Internet of Things Journal* 6, no. 6 (2019): 9399-9412.

[2] Zhang, De-gan, Hao Wu, Peng-zhen Zhao, Xiao-huan Liu, Yu-ya Cui, Lu Chen, and Ting Zhang. "New approach of multi-path reliable transmission for marginal wireless sensor network." *Wireless Networks* 26 (2020): 1503-1517.

[3] Fang, Weidong, Wuxiong Zhang, Wei Chen, Yang Liu, and Chaogang Tang. "TMSRS: trust management-based secure routing scheme in industrial wireless sensor network with fog computing." *Wireless Networks* 26 (2020): 3169-3182.

[4] Li, Xinlu, Brian Keegan, Fredrick Mtenzi, Thomas Weise, and Ming Tan. "Energy-efficient load balancing ant based routing algorithm for wireless sensor networks." *IEEE Access* 7 (2019): 113182-113196.

[5] Habib, M. A., Muhammad Yeasir Arafat, and Sangman Moh. "Routing protocols based on reinforcement learning for wireless sensor networks: A comparative study." *Journal of Advanced Research in Dynamical and Control Systems* 14 (2018): 427-435.

[6] Adil, Muhammad, Rahim Khan, Jehad Ali, Byeong-Hee Roh, Qui Thanh Hoai Ta, and Mohammed Amin Almaiah. "An energy proficient load balancing routing scheme for wireless sensor networks to maximize their lifespan in an operational environment." *IEEE Access* 8 (2020): 163209-163224.

[7] Maheshwari, Prachi, Ajay K. Sharma, and Karan Verma. "Energy efficient cluster based routing protocol for WSN using butterfly optimization algorithm and ant colony optimization." *Ad Hoc Networks* 110 (2021): 102317.

[8] M.V.N.R. Pavan Kumar, R. Hariharan. "Improved trustworthy, speed, and energy-efficient GPSR routing algorithm in large-scale WSN" *Elsevier Volume 24 (2022)*

[9] Wang, Zongshan, Hongwei Ding, Bo Li, Liyong Bao, and Zhijun Yang. "An energy efficient routing protocol based on improved artificial bee colony algorithm for wireless sensor networks." *IEEE Access* 8 (2020): 133577-133596.

[10] Xu, Chunlei, Weijin Zhuang, and Hong Zhang. "A deep-reinforcement learning approach for WSN routing optimization." In *Proceedings of the 4th International Conference on Computer Science and Application Engineering*, pp. 1-5. 2020.

[11]Woo, Jong Hun, Byeongseop Kim, SuHeon Ju, and Young In Cho. "Automation of load balancing for Gantt planning using reinforcement learning." *Engineering Applications of Artificial Intelligence* 101 (2021): 104226.

[12] Bal Krishna Saraswat, Manish Bhardwaj and Analp Pathak. "Optimum Experimental Results of AODV, DSDV & DSR Routing Protocol in Grid Environment." *Procedia Computer Science 57* (2015) 1359 – 1366

[13] Yun, Wan-Kyu, and Sang-Jo Yoo. "Q-learning-based data-aggregation-aware energy-efficient routing protocol for wireless sensor networks." *IEEE Access* 9 (2021): 10737-10750.

[14] Abd El-Moghith, Ibrahim A., and Saad M. Darwish. "Towards designing a trusted routing scheme in wireless sensor networks: A new deep blockchain approach." *IEEE Access* 9 (2021): 103822-103834.

[15] Hajiee, Maryam, Mehdi Fartash, and Naiseh Osati Eraghi. "An energy-aware trust and opportunity based routing algorithm in wireless sensor networks using multipath routes technique." *Neural Processing Letters* 53, no. 4 (2021): 2829-2852.

[16] Sridhar, V., K. V. Ranga Rao, V. Vinay Kumar, Muaadh Mukred, Syed Sajid Ullah, and Hussain AlSalman. "A machine learning-based intelligence approach for multiple-input/multiple-output routing in wireless sensor networks." *Mathematical Problems in Engineering* 2022 (2022): 1-13.

[17] Han, Bing, Feng Ran, Jiao Li, Limin Yan, Huaming Shen, and Ang Li. "A novel adaptive cluster based routing protocol for energy-harvesting wireless sensor networks." *Sensors* 22, no. 4 (2022): 1564.

[18] Jiale Liang et al., 2021" cooperative adaptive routing algorithm on mode selection in energy heterogeneous WSNs." *IEEE Access* 22, no. 4 (2021): 1564.

[19] Barbahan, Ibraheem, Vladimir Baikalov, Valeriy Vyatkin, and Andrey Filchenkov. "Multi-agent deep reinforcement learning-based algorithm for fast generalization on routing problems." *Procedia Computer Science* 193 (2021): 228-238.

[20] Kevin Schmid, Giorgos Vernikos, Georgios Chatzopoulos, Dan Alistarh, Ioannis Panageas Alexandros Tzannes, Milan Vojnovic "Bellman-Ford Goes Streaming: Faster and Fault-Tolerant Single-Source Shortest Paths" *ACM SIGMOD International Conference on Management of Data (2021)*

[21] Hajian, Elham, Mohammad Reza Khayyambashi, and Naser Movahhedinia. "A mechanism for load balancing routing and virtualization based on SDWSN for IoT applications." *IEEE Access* 10 (2022): 37457-37476.

[22] Xiang, Min, Mengxin Chen, Duanqiong Wang, and Zhang Luo. "Deep Reinforcement Learning-based load balancing strategy for multiple controllers in WSN." *e-Prime-Advances in Electrical Engineering, Electronics and Energy* 2 (2022): 100038.

[23] Altowaijri, Saleh M. "Efficient next-hop selection in multi-hop routing for IoT enabled wireless sensor networks." *Future Internet* 14, no. 2 (2022): 35.

[24] Wang, Pengjun, Jiahao Qin, Jiucheng Li, Meng Wu, Shan Zhou, and Le Feng. "Dynamic Optimization Method of Wireless Network Routing Based on Deep Learning Strategy." *Mobile Information Systems* 2022 (2022).

[25] Abdullah, Hussein Jawad, and Mohammed Najm Abdullah. "Routing enhancement in wireless sensor networks based on capsule networks: A survey." *International Journal of Nonlinear Analysis and Applications* 13, no. 2 (2022): 1229-1238.