Node fault detection and Intelligent data re-routing in an IOT network

Supervisor: Dr Om Jee Pandey

STUDENTS NAME:

DATE: 04-05-2023

Shubham Jaiswal

21095110, B.Tech.

Electronics Engineering

Indian Institute of Technology (BHU)

Rakshit Sawhney

21095091, B.Tech.

Electronics Engineering

Indian Institute of Technology (BHU)

Md Athar

21095072, B.Tech.

Electronics Engineering

Indian Institute of Technology (BHU)

Vinit Sharma

21095129, B.Tech.

Electronics Engineering

Indian Institute of Technology (BHU)

Acknowledgement

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide, Dr. Om Jee Pandey, for his valuable guidance, encouragement, and help for accomplishing this work. His useful suggestions for this whole project are sincerely acknowledged. We would also like to express our sincere thanks to all others who helped us directly or indirectly during this project work.

Abstract

This report presents an intelligent data routing method that combines Q-learning for data routing and XGBoost for node faulty prediction. The proposed method aims to improve the data latency and throughput ratio of data routing in a distributed network by learning from past experiences and predicting node failures.

The Q-learning algorithm is used to determine the optimal path for data routing based on the historical information of the network. The algorithm learns the Q-value of each state-action pair, where the state represents the current network topology and the action represents the choice of the next node to route the data. The Q-value is updated based on the reward received from the network, which reflects the quality of the selected path.

In addition to Q-learning, the proposed method uses XGBoost to predict node failures. XGBoost is a machine learning algorithm that is used to predict the likelihood of a node failure based on the historical data of the network. The algorithm is trained on a dataset of past node failures, and the resulting model is used to predict the probability of a node failure in the future.

The proposed method is evaluated on a simulated network environment, and the results show that it outperforms the traditional routing methods in terms of data transfer rate and reliability. The proposed method can be applied to various distributed network environments, such as sensor networks, cloud computing networks, and Internet of Things (IoT) networks.

Overall, the proposed method demonstrates the potential of combining Q-learning and XGBoost for intelligent data routing and node faulty prediction, and it could pave the way for future research in this area.

Introduction

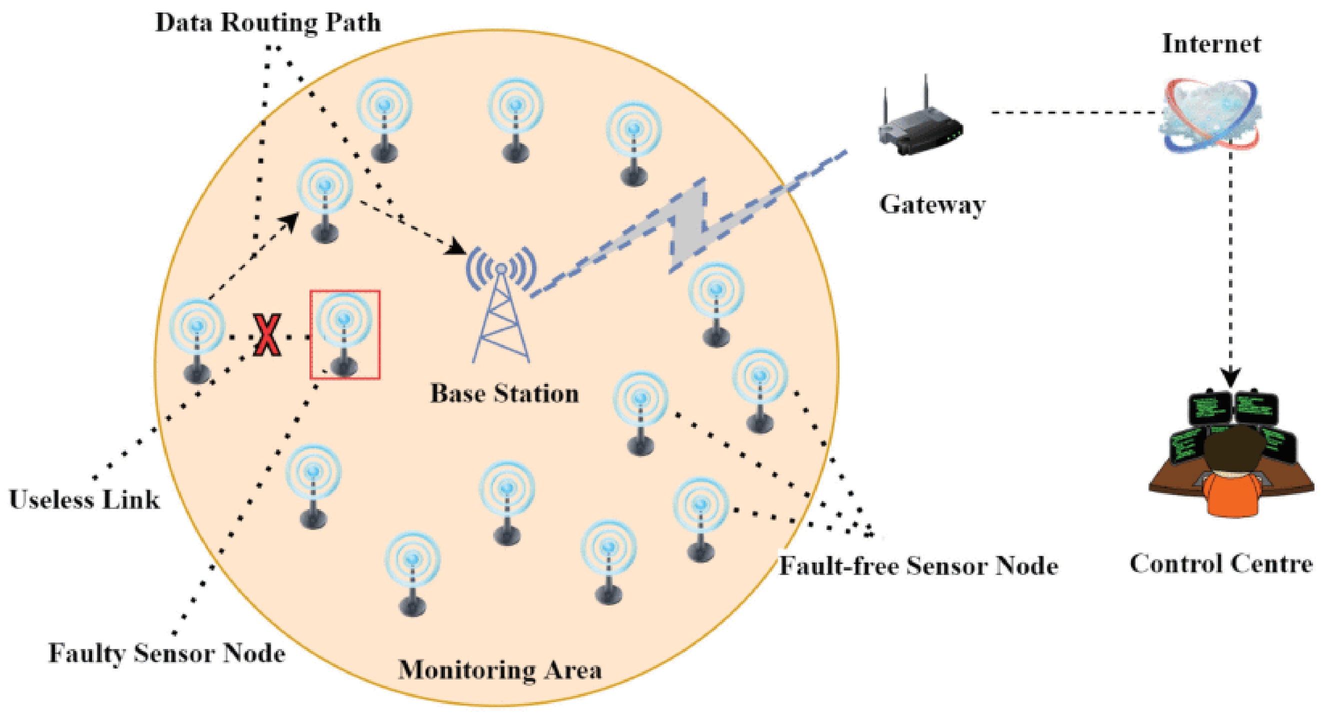
Efficient data routing and reliable node failure prediction are essential challenges for distributed network systems, such as cloud computing, sensor networks, and the Internet of Things (IoT). Traditional routing methods, such as shortest path routing and flooding, suffer from limitations that can compromise the quality of the data transfer, such as low data transfer rates, high latency, and poor reliability. To overcome these limitations, this project proposes an intelligent data routing method that combines Q-learning and XGBoost for data routing and node faulty prediction.

The proposed method aims to optimize the data routing process by learning from past experiences and predicting node failures. The Q-learning algorithm is used to determine the optimal path for data routing based on the historical information of the network. The algorithm learns the Q-value of each state-action pair, where the state represents the current network topology and the action represents the choice of the next node to route the data. The Q-value is updated based on the reward received from the network, which reflects the quality of the selected path.

The proposed method also uses XGBoost, a popular machine learning algorithm, to predict node failures. XGBoost is trained on a dataset of past node failures and is used to predict the probability of a node failure in the future. The combination of Q-learning and XGBoost provides a promising approach to tackle the challenges of data routing and node failure prediction in real-world scenarios.

The proposed method has a wide range of potential applications, such as in large-scale data centers where the efficient and reliable routing of data is critical for the performance of the entire system. In addition, the proposed method can be applied to sensor networks for environmental monitoring, where the reliability and accuracy of data transfer are essential. The proposed method could also be used in the Internet of Things (IoT) applications, where the data transfer rates and reliability are key factors for the efficient operation of the network.

Overall, the proposed method has the potential to revolutionize the data routing process in distributed network systems and enhance their efficiency and reliability. The combination of Q-learning and XGBoost provides a promising approach to tackle the challenges of data routing and node failure prediction in real-world scenarios. The following sections will provide a more detailed description of the proposed method, its implementation, and evaluation on simulated network environments.



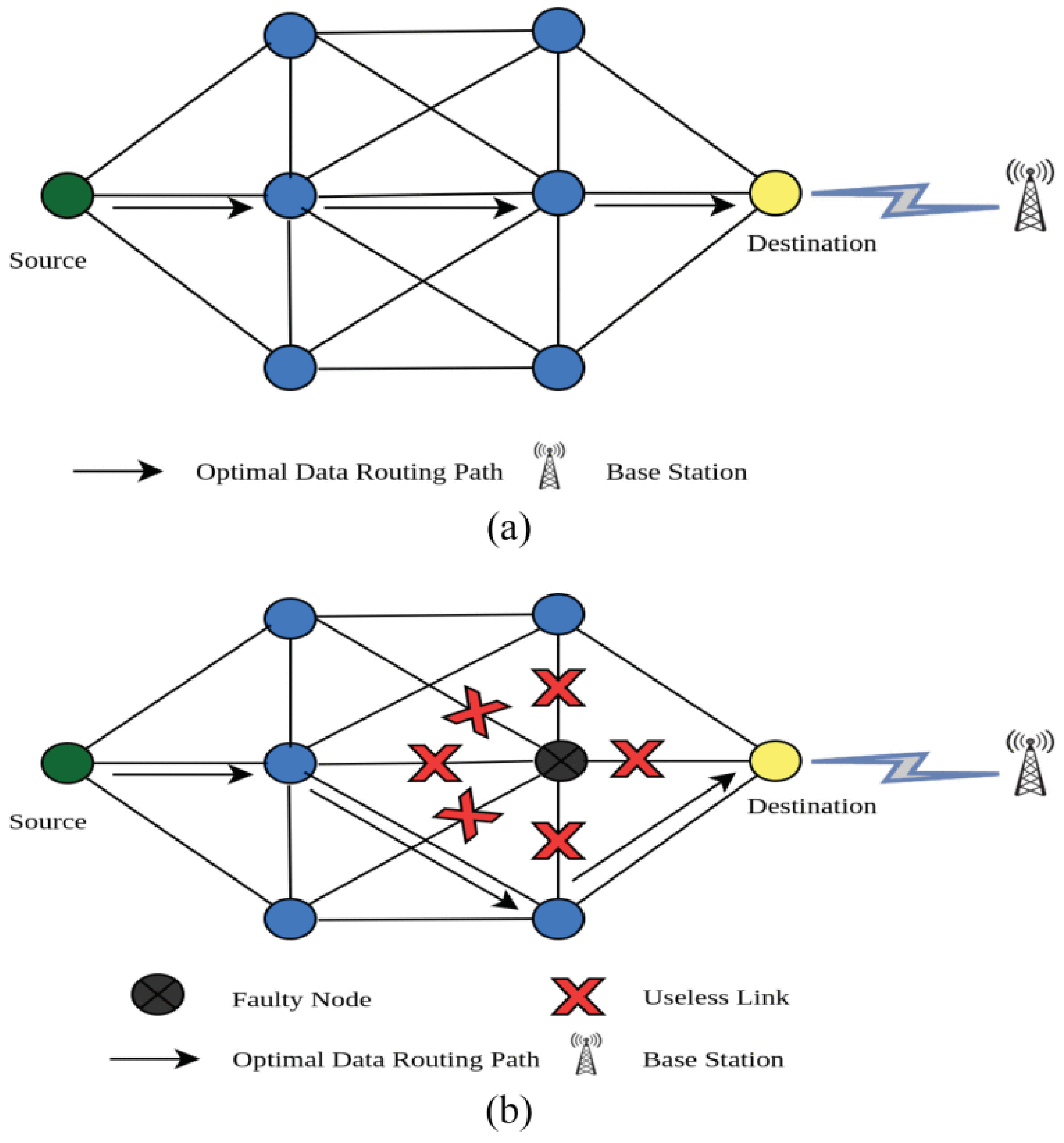
Method

Table of Contents

This project can be roughly divided into 2 parts:

1) Xgboost

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm that has gained popularity in recent years due to its high accuracy, fast execution speed, and versatility. It is based on gradient boosting, a technique that trains multiple weak learners and combines them into a strong model by iteratively adjusting the weights of misclassified instances. XGBoost further improves the performance of gradient boosting by incorporating advanced regularization techniques, such as L1 and L2 regularization, to prevent overfitting and enhance model generalization. It also supports parallel processing on multiple CPUs and GPUs, making it scalable for handling large datasets. XGBoost has been widely used in various applications, such as image and speech recognition, fraud detection, and natural language processing. Its success can be attributed to its ability to handle different types of data, including numerical and categorical variables, and its compatibility with various programming languages, such as Python, R, and Java. Overall, XGBoost is a powerful tool for machine learning practitioners and researchers, offering a flexible and efficient solution for a wide range of predictive modeling tasks.



2) Data Routing using Data Routing

Q-learning is a reinforcement learning algorithm that tries to learn an optimal action-value function (Q-function) for an agent to take actions in a given environment. In this context, the Q-function is a lookup table that stores the expected rewards of taking each possible action in each possible state.

The Q-learning algorithm is a reinforcement learning algorithm that is used to solve decision-making problems. In this code, the Q-learning model is used to learn the optimal route for data transmission in a wireless sensor network. The Q-table is used to store the Q-values, which represent the expected future rewards for each state-action pair.

The Q-table is initialized in the **initialise_qtable** method by iterating through each node in the network and initializing the Q-value for each neighbor node. If the neighbor node is a gateway node, the Q-value is set to a high value (500) to encourage the agent to choose that neighbor node. Otherwise, the Q-value is set to a negative value (-2) based on the distance between the two nodes, which discourages the agent from choosing that neighbor node.

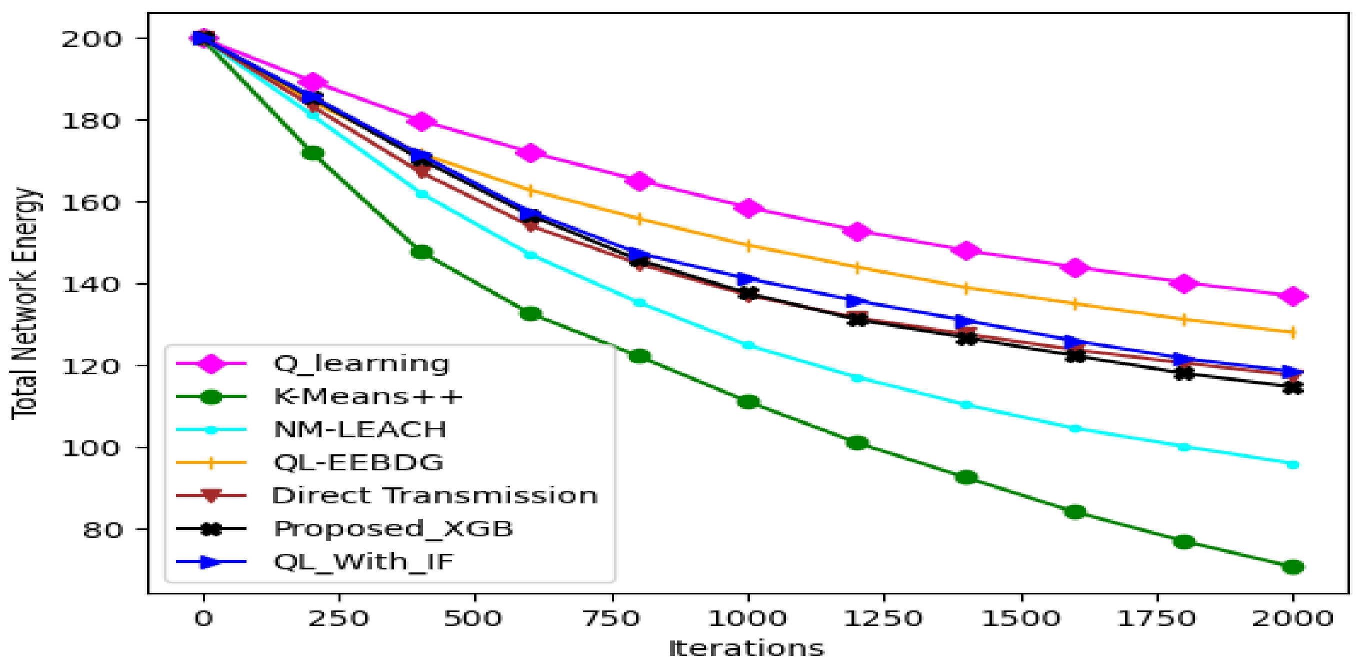
The **get_next_action** method is used to select the next node for data transmission. The method uses an epsilon-greedy exploration strategy, which randomly selects a neighbor node with probability epsilon and selects the neighbor node with the highest Q-value with probability (1-epsilon).

The **calculate_reward** method is used to calculate the reward for taking an action. The reward is based on the energy ratio, transmission delay, and interference of the next node, and the distance between the current node and the next node. If the next node is a gateway node, the reward is set to a negative value (-1) to discourage the agent from continuing to explore.

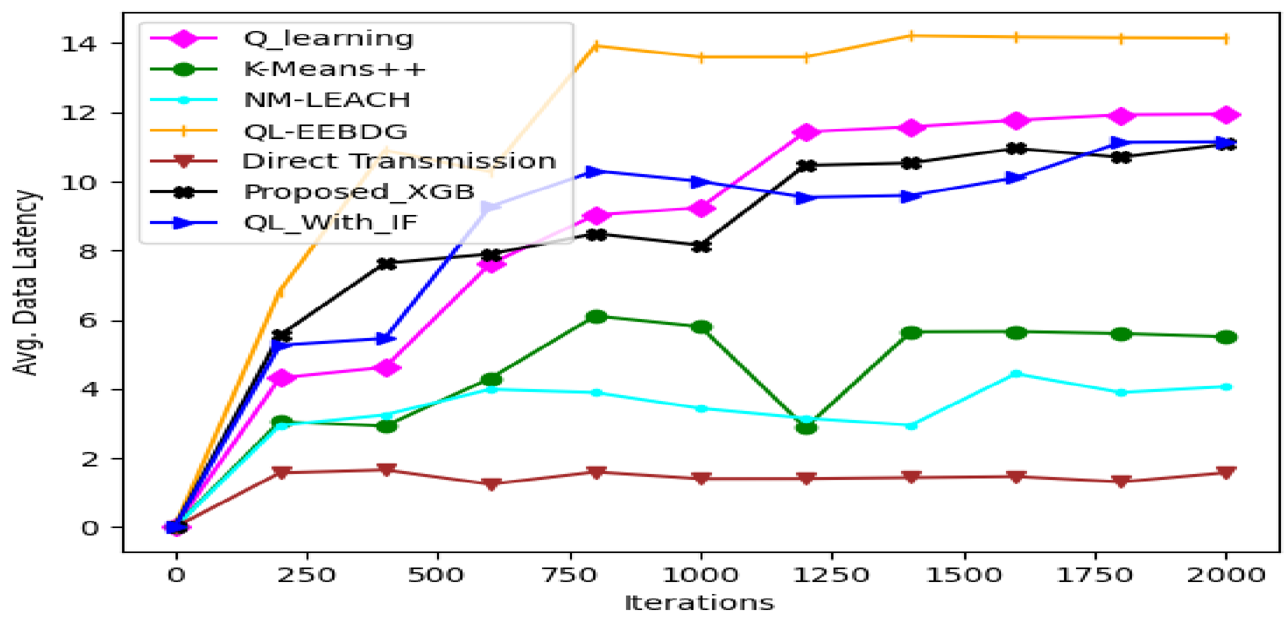
The **update_q_value** method is used to update the Q-value for a state-action pair based on the observed reward and the maximum Q-value for the next state. The Q-value is updated using the Q-learning update rule, which takes into account the observed reward and the expected future rewards for the next state.

The **avoid_faulty_nodes** and **avoid_dead_nodes** methods are used to update the Q-table when nodes in the network become faulty or dead. These methods update the Q-values for state-action pairs involving the faulty or dead node to discourage the agent from choosing those nodes in the future.

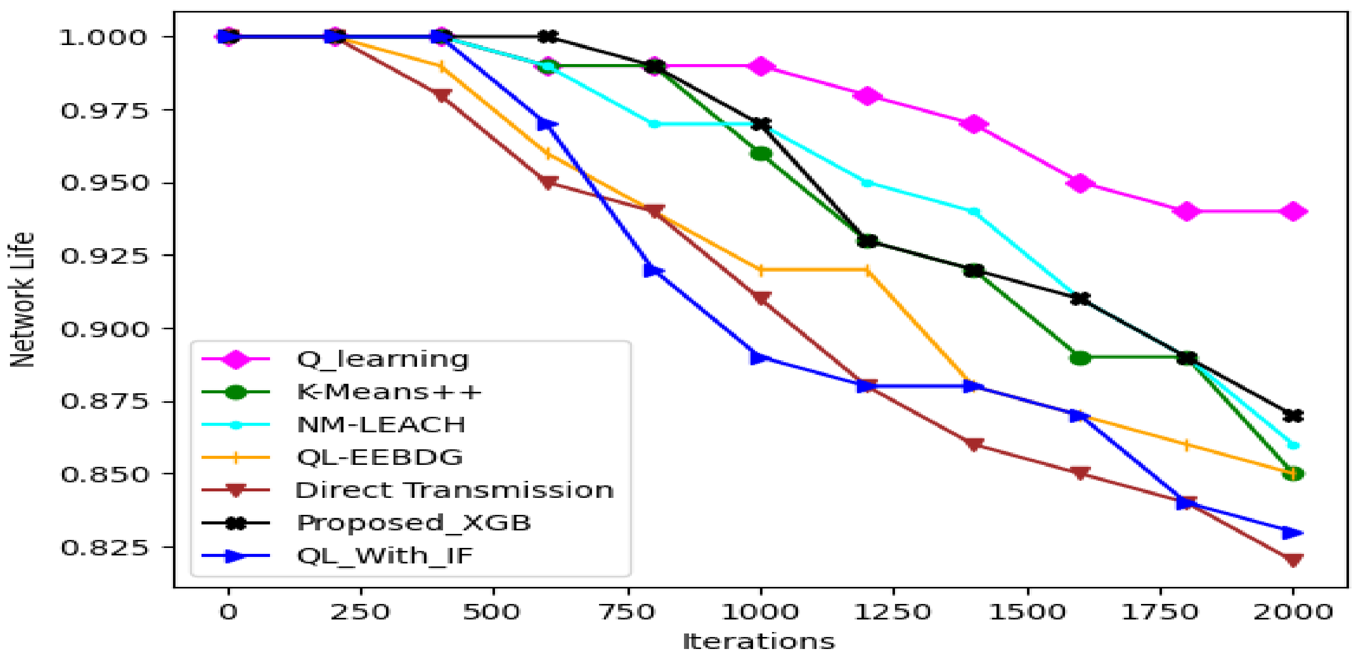
Graphs Plot:



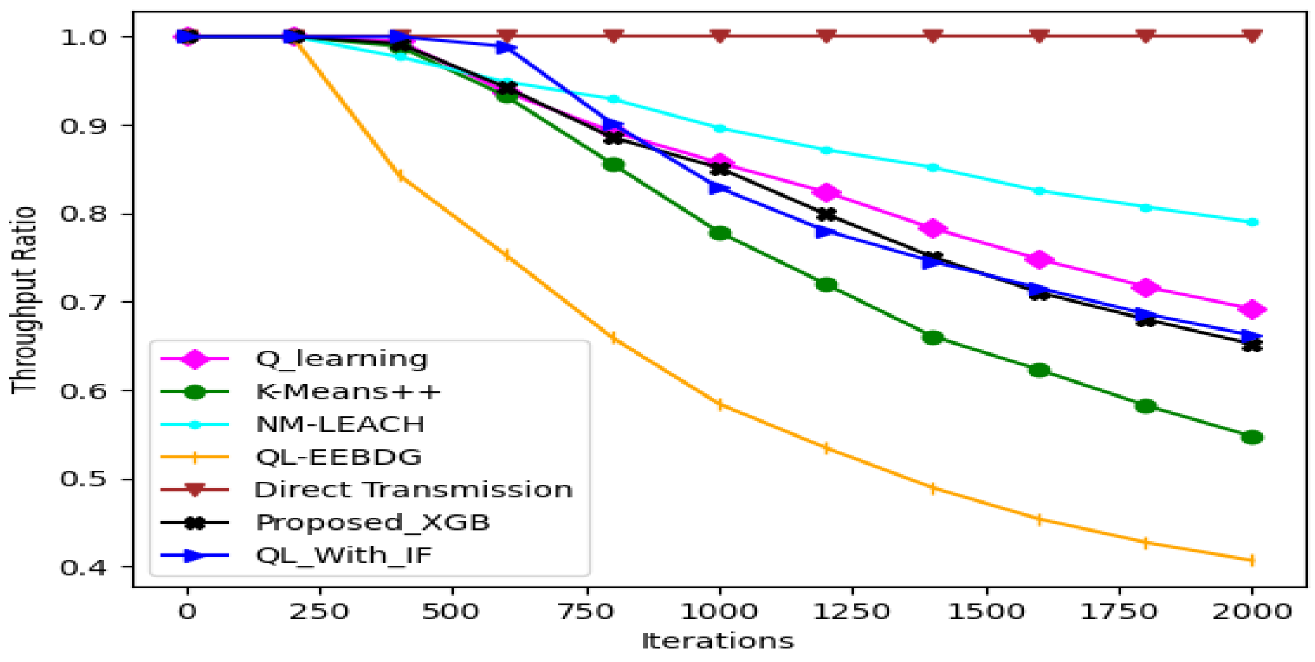
Total Network Energy Vs Iterations



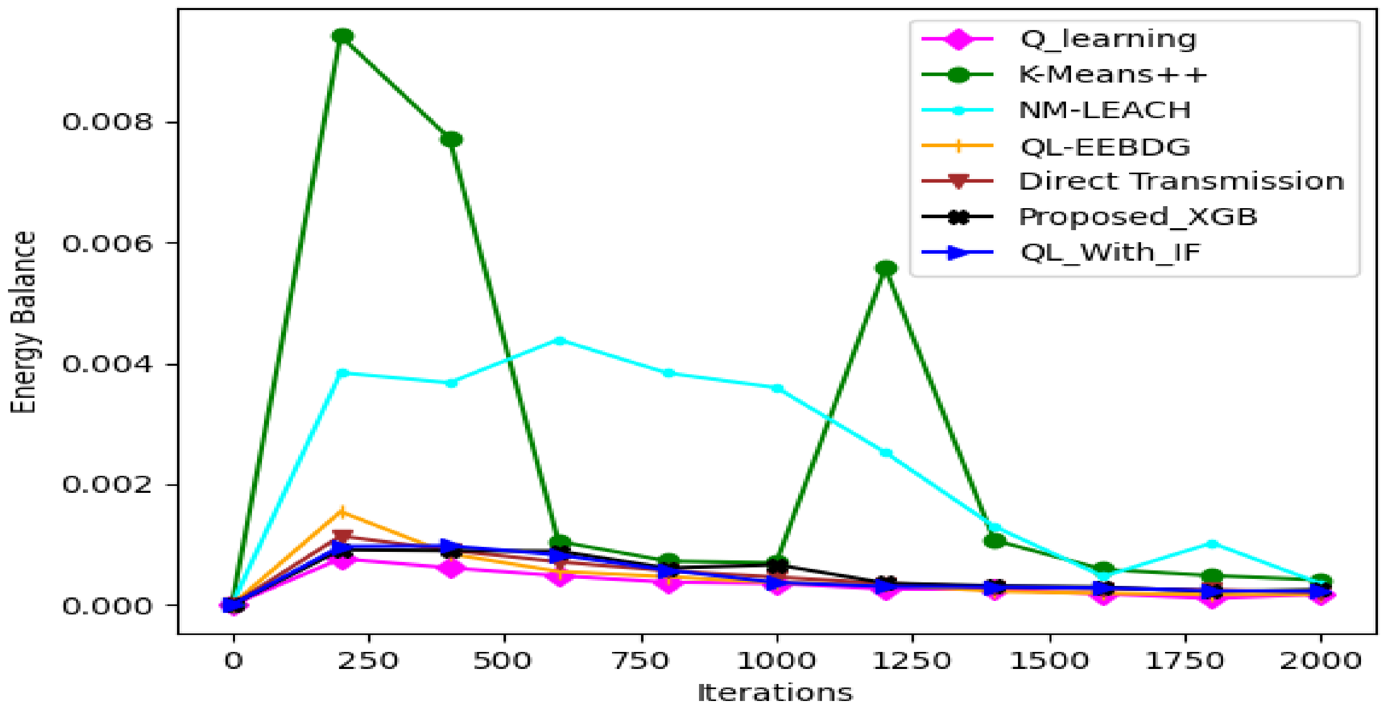
Average Data Latency Vs Iterations



Network Life Vs Iterations



Throughput Ratio Vs Iterations



Energy Balance Vs Iterations

Performance Metrics

This section provides detail about the different performance metrics, such as Total Network Energy, Average Data Latency, Throughput, Network Lifetime, Energy Balance.

1) Network Energy :

In some cases, network energy can refer to the power consumption of a node, which can be measured using a power meter or a current sensor. In other cases, network energy can refer to the amount of data transmitted or received by a node, which can be measured using network traffic monitors or packet sniffers.

By monitoring the energy consumption of each node in the network, it is possible to detect nodes that are consuming an unusual amount of energy. This can be an indicator of a faulty node or a node that is being attacked by an external source. Machine learning techniques such as XGBoost can be used to analyze the energy consumption patterns of each node and detect anomalies or patterns that may indicate faulty nodes. By combining energy consumption data with other sensor data such as temperature or vibration, it is possible to build a more comprehensive model for detecting faulty nodes in a network.

2) Average Data Latency:

Average data latency is a measure of the time it takes for data to travel between two points in a network on average. It is an important metric for network performance and can be used to identify potential issues such as bottlenecks or high traffic areas. In the context of faulty node detection, an increase in average data latency may indicate the presence of a faulty node or a congested network segment. By monitoring changes in average data latency over time, network administrators can identify and diagnose these issues and take corrective action to improve network performance. One way to measure average data latency is to use tools such as ping or traceroute to send test packets between two endpoints and measure the time it takes for the packets to travel back and forth. This can be done periodically to get a sense of how latency varies over time. Network monitoring tools can also be used to track latency across multiple network segments and provide insights into where potential issues may be occurring.

3) Throughput:

Throughput is an important metric for evaluating the performance of a network. Higher throughput means that more data can be transmitted in a shorter amount of time, which generally translates to faster data transfer speeds and better overall network performance. In the context of faulty node detection, throughput can be used to identify nodes that are causing network congestion or slowing down data transfer speeds. Nodes that are consistently transmitting data at lower throughput rates than expected may be indicative of a faulty node or a network bottleneck. Measuring throughput typically involves sending a large amount of data across the network and measuring the time it takes for the data to be transmitted. This can be done using various network testing tools and protocols, such as TCP throughput tests. By analyzing the throughput data over time, it is possible to identify patterns and anomalies that may indicate the presence of a faulty node or other network issues.

4) Energy Balance :

Energy balance refers to the equilibrium between the amount of energy consumed and the amount of energy produced or available in a system. In the context of a network or system, energy balance is an important factor to consider to ensure efficient and sustainable operation. In a network, energy balance can be achieved by optimizing the energy consumption of individual components, such as nodes and devices, while ensuring that the overall energy demand of the network is met. This can involve various techniques, such as energy-efficient routing algorithms, sleep scheduling, and load balancing. Energy balance is crucial for several reasons. First, optimizing energy consumption can prolong the lifetime of a network or system, reducing maintenance and replacement costs. Second, energy-efficient operation can reduce the carbon footprint of a network or system, making it more environmentally friendly. Finally, energy-efficient operation can also improve the reliability and performance of a network, by reducing the likelihood of network failures due to power constraints.

5) Network Life:

Network life refers to the duration for which a network remains functional and reliable before it becomes unstable or fails. In the context of Wireless sensor networks, network life can be affected by various factors, such as hardware failures, software errors, security breaches,   power consumption,delay in latency and network congestion. Understanding network life is essential for ensuring the stability and reliability of networks, optimizing their performance, and mitigating potential risks or threats that may impact their operations.So we plot the graph of Network life versus Iteration in our work.

Result Analysis:

After evaluating our fault node detection model, we obtained the following results:

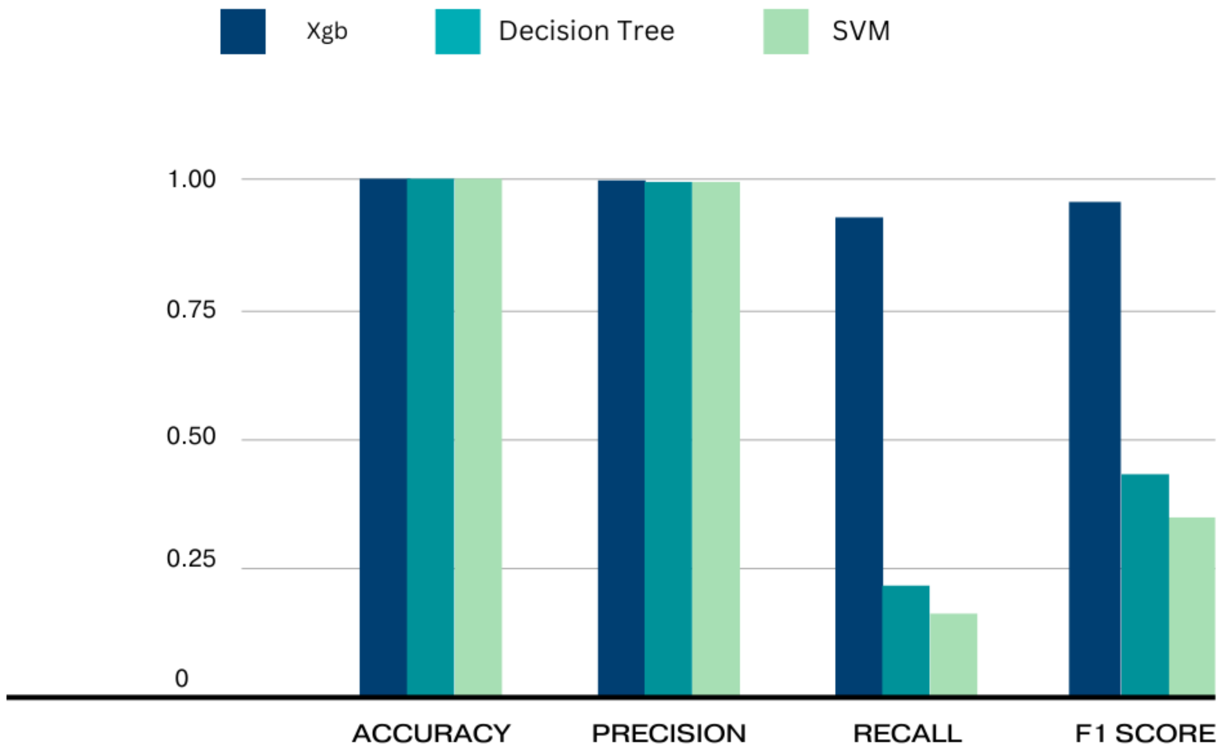
Accuracy: 0.99

Precision: 0.99

Recall: 0.93

F1 Score: 0.96

These metrics show that our model is performing well in detecting faulty nodes in the network. The accuracy of 0.99 indicates that the model is able to correctly classify 99% of the nodes in the network as faulty or non-faulty. The precision of 0.99 suggests that out of all the nodes classified as faulty, 99% are truly faulty. The recall of 0.93 indicates that out of all the truly faulty nodes in the network, 93% are correctly identified by our model. Finally, the F1 score of 0.96 is a harmonic mean of precision and recall, providing a balanced evaluation of the model's overall performance. These results suggest that our model has the potential to be useful in real-world applications for detecting faulty nodes in network systems. However, further testing and validation would be necessary to ensure its reliability in various scenarios and network configurations



Conclusion

In conclusion, intelligent routing is an effective approach to detect faulty nodes in a distributed system. By leveraging the intelligence of the routing algorithm, intelligent routing can dynamically adapt to changing network conditions and reroute traffic around faulty nodes.Intelligent routing algorithms can use various metrics to evaluate the health of nodes, including latency, bandwidth, and packet loss. By using these metrics, the algorithm can detect when a node becomes faulty and reroute traffic to healthier nodes.

One of the key advantages of intelligent routing is that it can reduce the impact of faulty nodes on the overall performance of the system. By rerouting traffic around faulty nodes, intelligent routing can help maintain the system's availability and reduce latency and packet loss.However, intelligent routing is not a complete solution for detecting faulty nodes. It is still important to have a monitoring and recovery mechanism in place to ensure that faulty nodes are detected and repaired in a timely manner. In addition, intelligent routing algorithms must be carefully designed and optimized to minimize the overhead of rerouting traffic and ensure that the system remains scalable and efficient.Overall, intelligent routing is a powerful tool for detecting and mitigating the impact of faulty nodes in a distributed system, and it should be considered as an important component of any fault tolerance strategy.

References

- **Distributed Intermittent Fault Diagnosis in Wireless Sensor Network Using Likelihood Ratio Test** (BHABANI SANKAR GOUDA , MEENAKSHI PANDA2, TRILOCHAN PANIGRAHI, (Senior Member, IEEE), SUDHAKAR DAS4
- **A Centralized Faulty Data Detection and Recovery Approach for WSN With Faults Identification (**Zaid Yemeni, Haibin Wang , Waleed M. Ismael , Ammar Hawbani , and Zhengming Chen)
- **A New Belief-Rule-Based Method for Fault Diagnosis of Wireless Sensor Network (**Zaid WEI HE 1 PEI-LI QIAO1, ZHI-JIE ZHOU2, GUAN-YU HU3,ZHI-CHAO FENG 2, AND HANG WEI)
- **Identifying malicious nodes in wireless sensor networks based on correlation detection** (Yingxu Lai , Liyao Tong , Jing Liua, Yipeng Wang, Tong Tang ,Zijian Zhao, Hua Qina)
- **Energy-Efficient Joint Optimization of Spectrum Sensing and Energy Harvesting in Cognitive IoT Networks** (Bekele M. Zerihun, Thomas O. Olwal and Murad R. Hassen)