Chapter 3

Energy-Efficient and QoS-Aware Data Routing in Node Fault Prediction-Based IoT Networks

3.1 Introduction

In large-scale IoT networks, multi-hop data routing methods have been used for data transfer over long distances, resulting in improved energy efficiency across the network [34, 35, 36]. However, multi-hop data routing leads to poor Quality of Service (QoS), such as higher data latency, more data interference, low data throughput, inefficient bandwidth utilization, and reduced network lifetime due to non-uniform energy consumption over the network [37, 38]. Nevertheless, cognitive multi-hop data routing [63] can limit the data interference phenomenon over the network, resulting in reliable data recovery at the gateway. Furthermore, the intelligent data routing framework [65] supports uniform energy consumption, leading to reduced data transmission delay and higher network lifetime. Therefore, it is necessary to consider novel machine learning-based techniques for multi-hop data routing, yielding energy-efficient solutions with improved QoS at the end user.

Moreover, in a multi-hop data routing framework, the existence of faulty nodes further degrades the performance of the network by keeping transmitted data packets with them. Hence, another important problem in multi-hop data routing is selecting relay devices for data transmission in a distributed manner. Essentially, for optimal data routing, the selection of relay devices must be done in such a way that they are not faulty nodes. In this context, several methods have been proposed to detect faulty nodes in the network during data transmission [108, 109]. In addition, various fault-tolerant methods [110, 111], have also been developed for data transfer over a wireless sensor network (WSN). The primary reasons for a node to be faulty are hardware failure, low battery, heavy workload, and human error [112]. Additionally, IoT devices (IoDs) are exposed to hostile conditions, which makes them susceptible to a variety of malfunctions and attacks [113]. However, the signal's analysis, generated by the sensor nodes, is used for the identification of the faulty state of a sensor node. Therefore, IoT needs intelligent signal analysis to classify the state of a sensor node in the network to improve its service performance.

In order to address the challenges of optimal data routing in multi-hop IoT networks, several methods have been proposed in the literature. For instance, various sensor node clustering techniques [114, 59], and alternative data routing schemes using weighted Dijkstra's algorithms [43, 58], are examples of popular multi-hop data routing methods. Moreover, recently, the introduction of small-world phenomena [60, 62, 115] towards multi-hop data transmission has also been investigated, resulting in improved network performance in many contexts. It has also been suggested to use mobile sink nodes for energy-balanced data transmission. Here, it is important to note that the network developed using the aforementioned methods yields an optimal multi-hop data transmission over WSNs. On the other hand, to further improve the performance of multi-hop data routing, several node fault detection and fault-tolerant data routing methods have also been proposed in the literature. These methods identify the faulty

nodes once they occur in the network. The primary methods used for fault detection and fault-tolerant data routing are clustering-based approaches, decision trees, and Markov chain algorithms [116, 117]. Additionally, in recent years, a few node fault prediction methods have also been examined [64, 101] to predict faulty nodes prior to the network. Predicting faulty nodes in the network improves the network performance when compared to fault detection and fault-tolerant methods. However, as per our best knowledge, none of the methods addresses the problem of node fault prediction and optimal data routing jointly, resulting in improved network performance. Moreover, machine learning (ML) based methods have hitherto not been investigated to address the aforementioned challenges jointly in an IoT network.

Motivated by the above discussion, in this chapter, a novel energy-efficient and QoS-aware optimal data routing method is proposed in node fault prediction-based IoT networks. The method utilizes unsupervised learning and reinforcement learning frameworks towards predicting the faulty nodes and data transfer in the network, respectively. Firstly, a novel method is proposed using the local outlier factor (LOF) for fault prediction over an IoT network. LOF is an unsupervised learning-based outlier node fault prediction algorithm that checks the density of each data point to verify whether a data point is an outlier or not. Subsequently, a novel Q-learning-based data routing method is proposed to avoid data transmission through faulty nodes. The method considers five parameters such as residual energy ($\mathcal{E}$), total data transmitted (D_T), total data received (D_R), node interference ($\mathcal{I}$), and node hold delay (T_D) as the node states for node fault prediction. Whether a node is faulty or not is predicted depending on node state values. The node states of a faulty sensor node are considered outliers, which signifies that the faulty node's state differs from the normal node states. After the prediction of faulty nodes, these nodes were avoided for data transfer by altering the Q-values of the actions at various states. Hence, the primary contributions of this chapter are as follows:

- A novel method of node fault prediction is proposed, which avoids data loss due to faulty nodes. The proposed method of node fault prediction is an unsupervised learning method, therefore, no labeled dataset is required. The prediction method is a light ML algorithm that does not require much memory for computation.
- A novel energy and QoS-aware data routing method has been proposed using Q-learning. The method uses node and network characteristics for its functioning, and hence, no external data is required.
- Novel measurement models to measure the performance of the proposed method are proposed. These models compute the node fault prediction accuracy, average data transmission time, network data throughput, and residual energy of the network.
- The performance of the proposed method is also evaluated over a real-field IoT testbed. Specifically, the Intel Berkeley Research Lab dataset is employed, which contains real sensor measurements collected from a deployed IoT network. The results obtained from this dataset illustrate the significance of the proposed method for real-time applications in medium- and large-scale IoT networks.
- The results of the proposed method are compared with the existing methods in literature such as low energy adaptive clustering hierarchy (LEACH) [118], balanced and energy efficient multi-hop routing (BEEMH) [119], K-means++ [120], novel modified low energy adaptive clustering hierarchy (NM-LEACH) [121], and Q-learning based energy-efficient and balanced consumption data gathering routing protocol (QLEEBDG) [122] over both, simulated IoT testbed and real field dataset.

3.2 Network Model and Problem Formulation

The network model of the proposed approach is represented as a graph $G = (\mathcal{N}, \mathcal{M}, \mathcal{L})$. Where, $\mathcal{N}$ indicates the set of IoDs, $\mathcal{N} = \{N_1, N_2, \cdots, N_j, \cdots, N_N\}$, $\mathcal{M}$ defines the set

of gateways, $\mathcal{M} = \{M_1, M_2, \cdots, M_m, \cdots, M_M\}$, and $\mathcal{L}$ constitutes the set of possible links formed in the network depending on the transmission power level of the IoDs. Table 3.1 shows the terminologies used in this chapter. Accordingly, when the distance

Table 3.1: Terminologies and definitions

Symbols/Parameters	Meaning
$\mathcal{N}$, $\mathcal{M}$, $\mathcal{L}$, $\mathcal{F}$	Set of N IoDs, gateways, possible links, and faulty nodes
d_{jk}	Distance between jth and kth IoDs
d_{max}	Threshold distance between jth and kth IoDs
η_{jk}	Incidence matrix element for node-pair j and k
$\mathcal{E}$, $\mathcal{H}$, $\mathcal{Q}$, $\mathcal{V}$, $\mathcal{P}$	Residual energies, hop counts, queue sizes, transmission powers, flag bits over IoDs
$P_k(f)$	Flag bit predicting fault status of kth IoD
L, W	Length and width of the network
D^i_{jm}, E^i_{jm}, T^i_{jm}	Delay, throughput, and energy for data transfer from j to m at ith iteration
p	Total number of data packets transmitted
ℓ	Length of a data packet
$E^i_{j,(res)}$	Residual energy of jth node at ith iteration
$\mathcal{I}$	Interference observed over a node
∂	Threshold of residual energy
$D_{\text{tra.}}$, $D_{\text{pro.}}$	Transmission and processing delay
$D_{\text{pkt.}}$, $D_{\text{que.}}$	Packet and queuing delay
$D_{\text{con.}}$	Delay from conventional multihop routing
R^m_i, R^j_i	Data received by gateway m, data generated by node j at ith iteration
β^i_{jk}	Data transferred from j to k at ith iteration
S^i_k	State of sensor node k at ith iteration
D_T, D_R	Total data transmitted and received
T_D	Message hold delay
A^i_k, A^z_k	Data point A for node k at iteration i, z
χ	Reachability distance
$\Delta(A^i_k, K^i_k)$	Distance between data point A and its Kth nearest neighbor
$\psi(A^i_k)$	Local reachability density for A^i_k
$K_{(A^i_k)}$	K-neighbors of A
$\varphi(A^i_k)$	Local outlier factor for A^i_k
α, γ	Learning rate and discount factor
s, s'	Current and next state
a, $\mathcal{A}$	Action taken and action set
$Q(s, a)$	Q-value for action a at state s
ϵ	Probability of exploitation
$\Re$, $\Re_E$, $\Re_T$	Reward, reward for residual energy, and for data delay
$\mathcal{L}_j$	Virtual links for node j
λ, μ	Distribution parameter and Poisson mean
$E_{tx}(k, d)$	Energy to transmit k-bit data over distance d
ε_e, ε_{fs}, ε_{mp}	Energy for transceiver switching, free space, and multi-path

between the node j and k, (d_{jk}), lies below a certain threshold value, (d_{max}), i.e., $(d_{jk} \leq d_{max})$, then the incidence matrix element corresponding to the node-pairs j and k, i.e., $\eta_{jk} = 1$, else, $\eta_{jk} = 0$. Here, d_{max} is the maximum distance over which an IoD is capable of transmitting its data directly. Moreover, the node and network characteristics of the deployed IoT network is represented as $C = (\mathcal{E}, \mathcal{H}, \mathcal{Q}, \mathcal{V})$, where, $\mathcal{E}$ represents the set of residual energies for the IoDs, $\mathcal{H}$ represents the set of number of hop counts between IoD pairs, $\mathcal{Q}$ denotes the set of queue size over the IoDs, and $\mathcal{V}$ defines the set of transmission power level between IoD pairs. In addition, $\mathcal{P}$ denotes the set of flag bits, $\mathcal{P} = \{P_1(f), P_2(f), \cdots, P_k(f), \cdots, P_N(f)\}$, where, $P_k(f)$ is the flag

bit observed at the kth IoD to predict the occurrence of a fault.

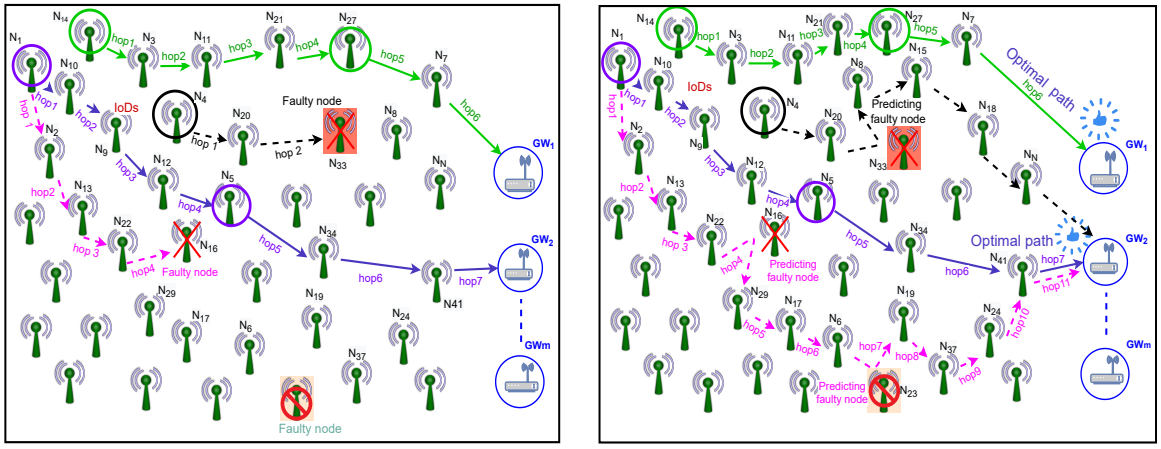

Figure 3.1: An illustration of data routing in multi-hop static IoT networks, with and without node fault prediction

As shown in Figure 3.1, the area of the considered IoT network is $L \times W$ m^2, where L is the length and W is the width of the network. The gateways are fixed within the network, and IoDs are randomly distributed over the area of interest. The data packets generated by the IoDs are transferred to the gateways using a multi-hop data transmission framework. In the considered network, one IoD acts as a source node while the others act as relay devices. Figure 3.1a illustrates data routing without node fault prediction, where data packets are stuck at faulty nodes. Here, nodes in the circle represent IoDs, acting as a source as well as a relay for another source node. Paths shown in green and purple colors indicate that there is no faulty node in the path; hence, data is successfully transferred to the gateway. In contrast, paths shown in pink and black colors indicate failure of data transmission due to the presence of a faulty node in the path. Figure 3.1b illustrates data routing with node fault prediction, where transmitted data packets are re-routed through alternative fault-free paths to ensure successful delivery to the gateway.

Depending on the flag bit status and other resource availability, the source IoD selects the relay IoD for data transfer. The flag bit $P_k(f)$ for the kth IoD is determined

using the proposed LOF algorithm. IoDs with $P_k(f) = 1$ are avoided to relay the data during data transfer from the source IoD to the gateways. A queue is maintained for every IoD while sending and receiving the data packets. The power consumed by the jth node while transferring the data to the kth node depends on the distance between them, i.e., d_{jk}. It also depends on the number of data packets transferred between them. Subsequently, the throughput of the network is affected by the number of hops used during data transmission. It is also a function of the interference experienced by the nodes. Finally, the transmission and reception of data observe more delay due to factors such as low processing, high congestion, low data rate, and large distance of transmission for a given IoD.

3.2.1 Problem Formulation

The main objective during data transmission is to reduce energy consumption, minimize data transmission delay, and improve data throughput. Therefore, the problem of energy-efficient and QoS-aware data routing in node fault prediction-based IoT networks is formulated as follows:

$$\min \underbrace{\sum_{i=1}^{I}\sum_{m=1}^{M}\sum_{j=1}^{N}\left(E^i_{jm}(d,p) + D^i_{jm}(\mathcal{Q},\ell,d) + \frac{1}{T^i_{jm}}(\mathcal{H},\mathcal{I})\right)}_{\text{Sum of the normalized value of energy, delay, and throughput inverse}} \tag{3.1}$$

Subject to:

$$\sum_{j=1}^{N} E^i_{j,(res)} \geq \partial, \quad \forall\{i \in I \text{ and } j \in \mathcal{N}\}, \tag{3.2}$$

$$\underbrace{\sum_{h=1}^{\mathcal{H}}\sum_{j=1}^{N} D^{(j,h)}_{\text{tra.}} + D^{(j,h)}_{\text{pro.}} + D^{(j,h)}_{\text{pkt.}} + D^{(j,h)}_{\text{que.}}}_{\text{Sum of delays occurred in a path} = D_{\text{tot}}} \leq D_{(\text{con.})}, \tag{3.3}$$

$$\underbrace{\sum_{m=1}^{M}\sum_{i=1}^{I} R_i^m \; p(t_i < \tau_i)}_{\text{Total data received at Gateways}} \leq \sum_{j=1}^{N}\sum_{i=1}^{I} R_i^j, \tag{3.4}$$

$$\underbrace{\sum_{\forall j,k}\beta_{jk}^i - \sum_{\forall j,k}\beta_{kj}^i = 0}_{\text{Data flow constraint at a node}} \;,\; \forall i, \quad j \neq k, \text{ and } \{j,k\} \in \mathcal{N}, \tag{3.5}$$

$$P_k(f) = 1, \text{ if } \varphi(A_k^i) > 1, \; \forall k \in \mathcal{N}, \tag{3.6}$$

$$\beta_{jk} = 0, \text{ if } P_k(f) = 1, \quad \forall \{j,k\} \in \mathcal{N}, \; j \neq k. \tag{3.7}$$

In the above problem formulation, (3.1) is the objective function that is represented as the sum of the normalized value of energy consumption (E_{jm}^i), transmission delay (D_{jm}^i), and data throughput (T_{jm}^i). The normalization of the parameters is performed using the formula given by:

$$X_{norm} = \frac{X_{act} - X_{min}}{X_{max} - X_{min}}. \tag{3.8}$$

Here, X_{act}, X_{min}, and X_{max}, are the actual, minimum, and maximum values of the parameter X, which is considered for the normalization. Moreover, E_{jm}^i is the energy consumed while transferring the data from jth node to mth gateway for ith iteration of data transmission. It is the function of the distance (d) between nodes and the total number of data packets (p) to be transmitted. D_{jm}^i is the total time taken to transfer the data from jth node to mth gateway for ith iteration. It is the function of queue size ($\mathcal{Q}$), length of the data packet (ℓ), and distance (d) between nodes. T_{jm}^i is the ratio of total data received at mth gateway to the total data transmitted from jth node at ith iteration. It is the function of a number of hops ($\mathcal{H}$) and interference ($\mathcal{I}$).

Objective function minimization is given in (3.1) and is associated with an important set of constraints, resulting in desired network development for real-time applications. The first constraint (3.2) denotes that the residual energy of the network ($E_{j,(res)}^i$) for all possible IoDs ($\forall j$) at ith iteration must be greater than a threshold value (∂).

Constraint (3.3) ensures that the sum of all possible delays that occur in a path must be less than or equal to the delay introduced by conventional multi-hop data routing ($D_{\text{con.}}$). Constraint (3.4) guarantees that the total data received by all M gateways must be less than the total data packets generated by N IoDs over the network. Constraint (3.5) imposes a data flow constraint over a node in the network. To deal with faulty nodes, constraint (3.6) ensures node fault prediction using the states of a sensor node. It indicates that if the LOF (φ) of data point A for the kth node at the ith iteration is greater than 1, then the fault prediction flag associated with the kth node will be 1. Finally, constraint (3.7) denotes that if kth node is faulty, i.e., if the prediction flag associated with the faulty node is 1 $\Rightarrow$ $P_k(f)$=1, then no data will be transferred from any node j to faulty node k, i.e., β^i_{jk}= 0.

It is important to note here that the optimization problem given in (3.1) cannot be solved simultaneously for all the parameters using traditional multi-hop data routing methods. Conventional multi-hop data routing methods result in near-optimal or sub-optimal solutions for the selected parameters. For instance, minimization of energy consumption might lead to higher data latency and vice versa. However, the utilization of the Q-learning framework, which is a type of reinforcement learning, results in a sub-optimal data routing solution for data transfer among IoD pairs. This is because the Q-learning framework is a type of edge learning where an individual device learns the optimal policy model from its actions and achieves maximized reward value, which is a combination of all the parameters considered in the problem. Subsequently, the solution to the optimization problem given in (3.1) is discussed in the following section using the Q-learning framework.

3.3 Proposed Method

This section discusses the proposed method of energy-efficient and QoS-aware data routing in node fault prediction-based IoT networks. First, the method of node fault

prediction using the LOF framework is described. Subsequently, the method of optimal data routing in a node fault prediction-based IoT network is presented, which utilizes Q-learning.

3.3.1 Node Fault Prediction Using LOF Framework

A sensor node can be classified as a normal node or a faulty node, based on its state pattern. In this work, the state of a sensor node k at ith iteration is represented as

$$S_k^i = \{\mathcal{E} \ , \mathcal{I} \ , D_T \ , D_R \ , T_D\}.$$

Here, $\mathcal{E}$, $\mathcal{I}$, D_T, D_R, and T_D represent the residual energy, interference observed, total data transmitted, total data received, and message hold delay, respectively, at node k for ith iteration of data transmission. After each iteration, the state of the sensor node gets

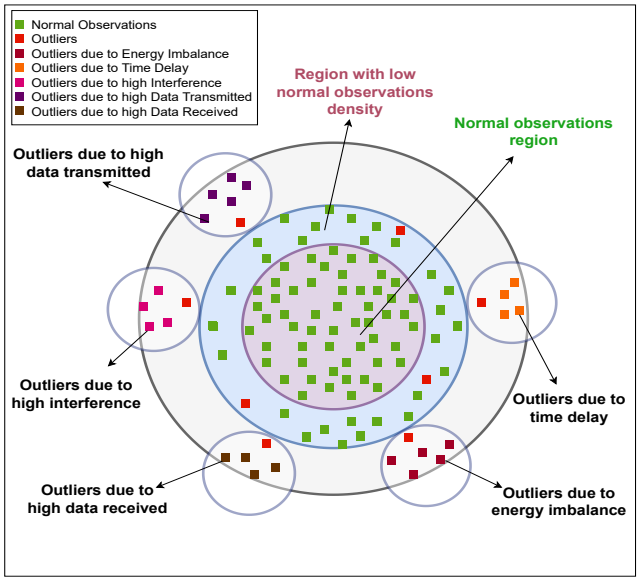

Figure 3.2: An illustration of normal and outlier data points computed using the LOF framework

updated with new values. These updated values are stored in the node to observe the pattern of node state data. After every new update, the oldest data points are removed from memory, and the latest one is added. A sensor node illustrates an abnormal behavior in its state pattern when it is about to be faulty. The abnormal behavior is observed in terms of high energy consumption, more data interference, increased

amount of data transmitted or received, and large message hold delay. Hence, these factors change the state S_k^i of the sensor node abruptly and abnormally. Accordingly, as shown in Figure 3.2, when the states of a node are plotted at various discrete time instants, then by checking the densities of those state points, outliers are predicted. For a normal node, the state points at various instants follow the same pattern (green dots), and hence the density of normal state points is uniform. However, for an abnormal state (outlier), the pattern of a node state is not uniform, and hence, the density around the recent data points changes abruptly. For instance, maroon, pink, and orange dots within small circles denote abnormal nodes due to energy imbalance, high interference, and more time delay, respectively. In addition, red dots within small and big circles indicate outliers due to other factors. Thus, the abnormal state point is considered an outlier that deviates from the normal pattern followed by previous state points. In order to compute the density of data points, a LOF framework is used, herein, which uses specific measures, discussed in the subsequent section.

3.3.1.1 Local Outlier Factor (LOF) Framework

LOF determines how the density of a given data point deviates from its local neighborhood. LOF indicates that an *outlier* is a data point that deviates significantly from the majority of its neighborhood data points. Therefore, the LOF framework takes into account the density of the neighboring area to determine an outlier. Steps involved to compute LOF (φ) are given as follows:

1. *K-distance and K-neighbors:* K-distance is the distance between the data point A_k^i and its Kth nearest neighbor (K_k^i), which is given by

$$\Delta(A_k^i, K_k^i) = \sqrt{\sum_{n=1}^{5} (X_{n,A_k^i} - X_{n,K_k^i})^2}. \tag{3.9}$$

Where, X_{n,A_k^i} and X_{n,K_k^i} denotes the nth coordinates of the data points A_k^i and K_k^i, respectively. Whereas, K-neighbors denoted by $K_{(A_k^i)}$, represent a set that includes the data points that lie in or on the circle of radius K-distance.

2. *Reachability Distance (χ):* It is defined as the maximum of the K-distance of data point A_k^i and distance between data points A_k^i and A_k^z, i.e., $\Delta(A_k^i, A_k^z)$, which is given by

$$\chi(A_k^z, A_k^i) = max(\Delta(A_k^i, K_k^i), \Delta(A_k^i, A_k^z)). \tag{3.10}$$

3. *Local Reachability Density (ψ):* It is the inverse of the average reachability distance of data point A_k^i from its K-neighbors ($K_{(A_k^i)}$), and is given by

$$\psi(A_k^i) = \frac{1}{\sum\limits_{A_k^z \in K_{(A_k^i)}} \frac{\chi(A_k^i, A_k^Z)}{\|K_{(A_k^i)}\|}}. \tag{3.11}$$

Intuitively, a lower value of ψ implies that less density of points is present around the observed data point, and the closest cluster is far from the observed data point.

4. *Local Outlier Factor (φ):* It is determined as the ratio of the average ψ of the $K_{(A_k^i)}$ to ψ of A_k^i.

$$\varphi(A_k^i) = \frac{\sum\limits_{A_k^z \in K_{(A_k^i)}} \psi(A_k^i)}{\|K_{(A_k^i)}\|} \times \frac{1}{\psi(A_k^i)}. \tag{3.12}$$

Generally, if $\varphi(A_k^i) > 1$, then the considered data point is an outlier. However, in some cases, a minimum threshold (ϕ) is set to neglect minute disturbances in data. In this work, $\phi = 1.5$, which alleviates all the possible disturbances, resulting in optimal node fault prediction. *Algorithm 3.1* enumerates the node fault prediction framework proposed in this work.

The method computes a flag variable $P_k(f)$ and updates it regularly after each iteration to keep track of the faulty IoD. Accordingly,

$$P_k(f) = \begin{cases} 1, & \text{if } \varphi(A_k^i) > \phi \\ 0, & \text{else.} \end{cases} \tag{3.13}$$

It indicates that if the LOF value of the latest data point ($\varphi(A_k^i)$) is greater than ϕ, then the sensor node k is prone to be faulty, otherwise, the node k can be considered as a normal node. Therefore, in *Algorithm 3.1*, all the nodes with $P_k(f) = 1$ are added to the set of faulty nodes ($\mathcal{F}$).

Algorithm 3.1 Node Fault Prediction Using LOF Framework

INPUT : Nodes set $\mathcal{N}$= $\{N_1, N_2, \cdots, N_N\}$, state $S_k^i = \{\mathcal{E}, \mathcal{I}, D_T, D_R, T_D\}$, most recent data point A_k^i; $\forall\{i = 1, 2, \cdots, I; k = 1, 2, \cdots, N;\}$ and $A_k^i = S_k^i$.

Set Parameters: Threshold value of $\varphi(A_k^i) = \phi$, number of K-neighbouring points.

OUTPUT : Set of faulty nodes $\mathcal{F}$= $\{F_1, F_2, \cdots, F_k\} \Rightarrow P_k(f)$=1,$\forall$ k $\in$ N.

Steps:

1. **for** i= 1,2, $\cdots$, I **do**
2. **for** k= 1,2, $\cdots$, N **do**
3. Determine the K-distance and K-neighbours for the data point A_k^i
4. Compute the Reachability Distance (χ) for every K-neighbour of A_k^i using (3.10)
5. Compute Local Reachability Density (ψ) to obtain the local density of data point A_k^i using (3.11).
6. Compute φ for A_k^i using (3.12).
7. **if** $\varphi_k(A_k^i) > \phi$ **then**
8. $P_k(f)$=1
9. $\mathcal{F} = F_k \cup \mathcal{F}$
10. **else**
11. $P_k(f) = 0$
12. **end**
13. ***end***
14. ***end***

3.3.2 Optimal Data Routing Using Q-learning

In this work, along with the prediction of faulty nodes, a Q-learning-based data routing technique is also proposed for data transfer. Q-learning is a model-free value-based reinforcement learning (RL) algorithm. It is defined using an **agent**, a set of **states**, and a set of **actions**. In RL, the reward is mapped with each action the agent takes, such that the cumulative reward keeps adding up with each simultaneous action. An agent decides its next action based on a policy $\pi(a_t|s_t)$, where s_t is the current state of the agent, a_t is the next action for the agent as returned by the policy π. After the action a_t, the agent receives the reward $\Re_{t+1}$ and changes its state to s_{t+1}. Thus, the main objective is to improve the policy π to maximize the cumulative reward.

In the proposed Q-learning-based data routing method, the message flow across the network is considered as the **agent**, the sensor node N_j at which the message is currently held is considered as the **state** of the agent, and the next node to which the message will be delivered is considered as the **action**. The set of states and actions are defined as, $\mathcal{S} = \{s_1, s_2, s_3, \ldots, s_N\}$ and $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_N\}$, where $\mathcal{A}_j = \{s_k; \forall s_k \in \mathcal{N}_{s_j}\}$. Here, $\mathcal{N}_{s_j}$ represents the set of neighbor nodes of the IoD s_j.

Every sensor node has its own Q-table. The Q-table uses an optimized Q-function Q(s, a) to assist in determining the optimal course of action at a given state. This function is basically a one-to-one mapping in which each action of a state is mapped to an optimal Q-value. This mapping is estimated through iterative updates using the Q-value updation equation. The Q-value is updated using the below equation,

$$Q_{new}(s,a) = (1-\alpha).\underbrace{Q(s,a)}_{\text{old value}} + \alpha\underbrace{\{\Re + \gamma.Q(s',a)\}}_{\text{learned value}}. \tag{3.14}$$

Here, 's' is the current state of the agent, and it takes action 'a' to reach the next state 's''. $Q(s,a)$ is the Q-value associated with the action 'a' at state 's', which is being updated using (3.14). $\Re$ represents the observed reward during that action. $Q(s',a)$

represents the maximum of all Q-values associated with all the actions at state s''. α is the learning rate, and γ is the discount factor for future rewards. The reward function that we used depends on three factors, i.e., residual energy ($\Re_E$) of the next node, expected transmission delay ($\Re_T$) between two nodes, and the number of hops ($\mathcal{H}_i$) the message has incurred. Here, $\Re_E$ is computed using the relation given by:

$$\Re_E = \frac{\text{Residual energy of the next node}}{\text{Initial energy of the present node}}.$$

Additionally, the expected transmission delay is computed using the data latency measurement model given in **Section 3.4.3**.

In particular, reward factors due to residual energy ($\Re_E$) and transmission delay ($\Re_T$) differ in terms of unit and their numerical scale. Hence, this may lead to more weightage of one factor on the total reward. Therefore, both of the factors are normalized, which will return unitless values in a numerical scale between 0 and 1. Here, normalization is performed using (3.8). After normalization, we have the parameters as $\Re^E_{norm}$ and $\Re^T_{norm}$. The nodes with higher residual energy are preferred more for data transmission, hence they should have a positive effect on the total reward, and the nodes with higher transmission delay should have a negative effect on the total reward. Hence, the total reward function is given by:

$$\Re(s,a) = \begin{cases} (\mathcal{H}_i \times \gamma) \times (-1 + \Re^E_{norm} - \Re^T_{norm}), & s' \in N \\ \rho & s' \in M \end{cases}. \tag{3.15}$$

Here, 1 is subtracted from the normalized parameters to make the whole reward term negative. A negative reward reduces the number of steps required by the agent to increase the cumulative reward value, resulting in reduced overall path length and data transmission delay. ρ is the maximum reward associated with the gateway node to give more priority to the gateway nodes. In this work, to carry out the calculations,

the value of ρ is considered as 100.

Algorithm 3.2 enumerates all the steps of Q-learning-based data routing in a node fault prediction-enabled IoT network. Here, the Q-matrix for each IoD is created over the developed network. Subsequently, the Q-values in the matrices are initialized using a distance function given by:

$$Q(s,a) = -2 + \frac{X_{cordinate}(s') - X_{cordinate}(s)}{TR}. \tag{3.16}$$

In (3.16), 'TR' represents the transmission range of the IoD. For best results, the Q-tables must contain well-learned values, hence, the network needs to be simulated for a good number of iterations. This process is known as **offline learning**. In this work, offline learning is done for 10,000 iterations, transmitting one message at a time to one of the gateway nodes.

In offline learning, **epsilon-greedy** technique is used for action selection. The epsilon-greedy method is used to balance **exploration** and **exploitation** by randomly choosing between them. In epsilon-greedy, epsilon (ϵ) refers to the probability of exploiting with a small probability of exploring.

After selecting the next action, Q-value is updated using (3.14). Hence, on completion of offline learning, a well-learned Q-table for every sensor node is obtained, which contains optimal Q-values for every possible action at that state.

$$\mathcal{A}(t) = \begin{cases} \max Q(s,a), & \text{with probability } \epsilon \\ \text{any action } a, & \text{with probability 1-}\epsilon \end{cases}. \tag{3.17}$$

Thereafter, all the sensor nodes reset to their initial states S_k^0. For the first 100 iterations, the optimal path is calculated without predicting faults, assuming the minimum guaranteed lifetime of new sensor nodes. This assumption gave us time to collect enough datasets for the outlier prediction of each node.

Algorithm 3.2 Node Fault Prediction Based Optimal Data Routing Using Q-learning in IoT Network

INPUT : Nodes set $\mathcal{N}$= $\{N_1, N_2,\cdots,N_j,\cdots,N_N\}$, gateways set $\mathcal{M} = \{M_1, M_2,\cdots,M_m,\cdots,M_M\}$, $\forall\{M = 1, 2, \cdots, M; j = 1, 2, \cdots, N;\}$

OUTPUT : *optimal_path_array* [ ]

INITIALIZATION

1. Deploy N sensor nodes and M gateways randomly with initial energy level E and unique identification number over a network area $L \times W$ m^2

2. Create set of virtual links ($\mathcal{L}_j$) using distance values for each sensor node

$\mathcal{L}_j = \{k; \text{if } \eta_{jk} = 1, \forall k \in N\}$ where, $\eta_{jk} = \begin{cases} 1, & \text{if } d_{jk} \leq d_0 \\ 0, & \text{otherwise} \end{cases}$

3. Initialize the Q-matrix for each sensor node N_j

$\mathcal{Q}_j = \{\mathcal{Q}(s, a)\ \forall a \in \mathcal{N}_{s_j}\}$ using (3.16)

OFFLINE LEARNING

4. **for** i= { 1, 2, $\cdots$, I } **do**

5. Generate a message by randomly selecting an IoD from the network.

6. **while** (*current_node* $\notin \mathcal{M}$)

7. Select the next IoD using (3.17) and store in *next_node*.

8. Compute the reward using (3.15)

9. Update the Q-values using (3.14)

10. *current_node* = *next_node*

11. **end**

12. **Reset** all the sensor nodes to their initial energy levels E, and other properties, such as node interference and total data transmitted/received to 0.

ONLINE LEARNING

Finding Optimal Path

13. **for** j = { 1, 2, $\cdots$, N } **do**

14. **for** i= { 1, 2, $\cdots$, I } **do**

15. **if** $i \leq 100$

16. **Jump** to **24**
17. **else**
18. Compute $\mathcal{F}$ using **Algorithm 3.1**
19. **for** $\forall F_k \in \mathcal{F}$ **do**
20. F_k sends a message to all its neighboring nodes that it is prone to fault
21. Neighboring nodes update the corresponding Q-values using (3.18)
22. **end**
23. **for** $\forall m \in \mathcal{M}$ **do**
24. Initialize *optimal_path* array, *total_reward* to 0 and *selected_device* to IoD_1
25. Add *selected_device* to *optimal_path* array
26. **while** (*selected_device* != gateway m)
27. Select the next IoD using (3.17) with $\epsilon = 1$ and store it in *next_device*
28. Compute action reward using (3.15) and add it to *total_reward*
29. *selected_device* = *next_device*
30. Add *selected_device* to *optimal_path*
31. **end**
32. Select the most optimal path corresponding to the gateway with the highest *total_reward*
33. **end**

Data Routing

34. Set *curr_device* = IoD_1 and *optimal_path* index $r = 0$
35. **while** (*curr_device* != m)
36. *next_device* = *optimal_path* [r+1]
37. Update $S_j^i = \{\mathcal{E}\ ,\ \mathcal{I}\ ,\ D_T\ ,\ D_R\ ,\ T_D\}$
38. Update Q-values using (3.14)
39. *curr_device* = *next_device*
40. $r = r + 1$
41. **end**
42. **end**

Though ignorance of fault occurrence in the first 100 iterations does not affect the network quality much, because sensor nodes have a minimum lifetime for which they can function without faults. After 100 iterations, each time before message generation, $\varphi(A_k^i)$ of all the nodes is checked to know whether a node is showing normal or abnormal behavior. The abnormal nodes are stored in the set $\mathcal{F}$. All the nodes in the set $\mathcal{F}$ send a message to all their neighboring nodes about their abnormal behavior. Neighboring nodes, after receiving the message, update the Q-values of actions associated with the faulty node as the next state, using

$$Q(s, a_f) = min(Q(s, a)) - 1. \tag{3.18}$$

This makes the action associated with the faulty node as the least priority action. Thus, while calculating the optimal path, faulty nodes are neglected or ignored. As there are four gateways, the message has four different options for its destination. However, only one gateway is needed for transmission. Hence, to select the best gateway for

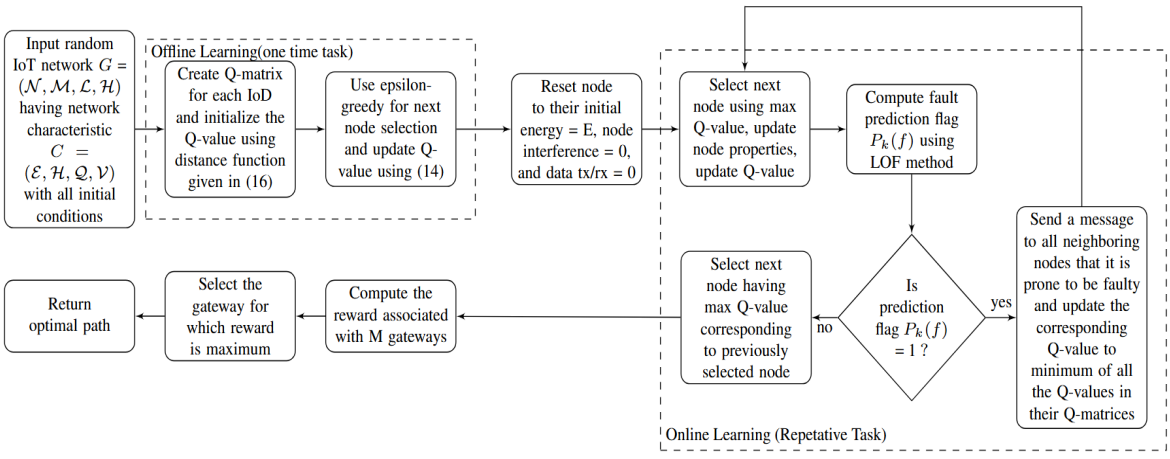

Figure 3.3: Block diagram illustrating the method of energy-efficient and QoS-aware data routing in node fault prediction-based IoT networks

transmission, expected rewards are calculated for the optimal paths to all the gateways. The gateway for which the expected reward is highest is selected, and the optimal path is returned. While calculating the optimal path, ϵ is kept as 1 to select the action

having the maximum Q-value. Now, the agent follows the returned path to reach the gateway node. Meanwhile, since the network is dynamic, i.e., it keeps changing in terms of the residual energy of nodes, the availability of each node for data transmission also keeps on changing. Accordingly, the Q-values are also regularly updated using (3.14). Hence, the proposed method successfully avoids faulty sensor nodes while obtaining the optimal path during data routing.

The block diagram shown in Figure 3.3 summarizes the complete process of computing an optimal data routing path along with the prediction of faulty nodes in the IoT network. The offline learning module illustrates a one-time operation. However, the repetitive steps used towards achieving the faulty node aware-optimal path for each network device are depicted within the online learning module.

3.4 Measurement Models

In this section, various measurement models used for the performance evaluation of the proposed method are discussed. The performance of the node fault prediction method is measured using accuracy, precision, and recall values. However, the QoS of the proposed method is measured in terms of throughput ratio and data latency. Finally, an energy consumption measurement model is given, which considers the amount of data transferred and distance as parameters.

3.4.1 Node Fault Prediction Measurement Model

In this model, two matrices are constructed: one for the actual class and the other for the prediction class. Further, we set 1 (or) + for the faulty node and 0 (or) - for the non-faulty node. In the next step, we compare these two matrices and set:

- True Positive (TP): If both classes are positive. It indicates that the node predicted as faulty is actually faulty.
- True Negative (TN): If both classes are negative. It shows that the node predicted

as non-faulty is actually non-faulty.

- False Positive (FP): If the prediction class is positive and the actual class is negative. It represents that the node is actually non-faulty but predicted as faulty.
- False negative (FN): If the prediction class is negative and the actual class is positive. It denotes that the node is actually faulty but predicted as non-faulty.

Table 3.2: Confusion matrix

	Predicted class(+)	Predicted class(-)
Actual class (+)	True Positive (TP)	False Negative (FN)
Actual class (-)	False Positive (FP)	True Negative(TN)

These measures are referred to as the confusion matrix. Accordingly, Table 3.2 illustrates the confusion matrix. The above evaluation indices are used to measure the accuracy, precision, and recall values given by;

3.4.1.1 Accuracy

It indicates, out of all the predictions, how many predictions are correct.

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{3.19}$$

3.4.1.2 Precision

It indicates that out of all the nodes that we predicted as faulty, one is actually faulty.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{3.20}$$

3.4.1.3 Recall

It denotes, out of all the faulty nodes, how many are correctly predicted.

$$Recall = \frac{\text{TP}}{\text{TP + FN}}. \tag{3.21}$$

These values indicate the effectiveness of the proposed node fault prediction method. These parameters are computed at each iteration with the help of TP, FN, FP, and TN values.

3.4.2 Throughput Ratio Measurement Model

The ratio of total data received to total data generated over the network is known as the throughput ratio. If the total data received and total data generated are given by:

$$Total\ Data\ Received = \sum_{m=1}^{M}\sum_{i=1}^{I} R_i^m \tag{3.22}$$

$$Total\ Data\ Generated = \sum_{j=1}^{N}\sum_{i=1}^{I} R_i^j. \tag{3.23}$$

Then the throughput ratio is computed as:

$$Throughput\ Ratio = \frac{\sum_{m=1}^{M}\sum_{i=1}^{I} R_i^m}{\sum_{j=1}^{N}\sum_{i=1}^{I} R_i^j}. \tag{3.24}$$

Where R_i^m is the total number of data packets received by mth gateway at ith iteration and R_i^j is the total number of data transferred by jth node at ith iteration.

3.4.3 Data Latency Measurement Model

In a multi-hop IoT network, there are various types of delays, such as transmission delay, processing delay, packetization delay, and queuing delay, which are defined as:

3.4.3.1 Transmission Delay

It is the time required to transfer data from one IoD to another via a direct link. It varies as the distance between the IoD pair changes. It is given by:

$$D_{\text{tra.}} = \frac{\text{Separation between nodes (m)}}{\text{Transmission speed (m/s)}} = \frac{\text{SNs}}{\text{TS}}. \tag{3.25}$$

3.4.3.2 Processing Delay

The amount of time taken in processing the data packets by an IoD is called processing delay. It is caused due to several tasks performed by the IoD on a data packet, including checking for errors while transmission, and planning where to go next, among others. To compute the processing delay, the random exponential variable is used, i.e., given by:

$$D_{\text{pro.}}(x, \lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.26}$$

Where, $\lambda > 0$ is called the distribution parameter.

3.4.3.3 Packetization Delay

The time required to transfer data packets into the channel is known as the packetization delay. It occurs because of the link data rate and is given by:

$$D_{\text{pkt.}} = \frac{\text{Length of Packet(bits)}}{\text{Rate of Transmission (kbps)}} = \frac{\text{LP}}{\text{RT}}. \tag{3.27}$$

3.4.3.4 Queuing Delay

The time duration for which the data packets are queued up after being processed by an IoD is called queuing delay. Queueing delay is considered a Poisson-distributed random variable and is computed as:

$$D_{\text{que.}} = P_{distribution}\,(k\ \ events) = \frac{\mu^k e^{-\mu}}{k!}. \tag{3.28}$$

Where μ is the mean of the Poisson distribution. As a result, the total delay observed by an IoD during the transmission of data is given by:

$$D_{\text{total}} = D_{\text{tra.}} + D_{\text{pro.}} + D_{\text{pkt.}} + D_{\text{que.}} \tag{3.29}$$

3.4.4 Energy Consumption Measurement Model

As deployed sensor nodes perform sensing, processing, transmitting, and receiving operations, their energy is dissipated. The primary reason for energy dissipation is data transmission, which depends on the distance between transmitter and receiver (d), and the number of packets transmitted. If the distance between transmitter and receiver is below a threshold d_o, then the free space model is used in which power loss is proportional to d^2. Otherwise, the multi-path fading model in which power loss is proportional to d^4 is considered. The transmitter consumes energy to execute radio electronics and amplification, whereas the receiver dissipates its energy in radio electronics only. Hence, the energy consumed by the source IoD to transmit a k-bit data packet to a receiver located at a distance d is given by [123]

$$E_{tx}(k,d) = \begin{cases} k\varepsilon_e + k\varepsilon_{fs}d^2, & d < d_0 \\ k\varepsilon_e + k\varepsilon_{mp}d^4, & d \geq d_0 \end{cases} \tag{3.30}$$

Whereas, the energy consumed by the receiver IoD to receive k-bit data packet is given by

$$E_{rx} = k \times \varepsilon_e, \tag{3.31}$$

where ε_e represents the energy consumed by switching on and off the receiver and transmitter circuitry, ε_{fs} denotes the energy consumed by the free space path loss model,

and ε_{mp} represents the energy consumed by the multi-path power loss model. The values of the energy model used for experiments are $\varepsilon_e = 50 \times 10^{-9}$ J/bit, $\varepsilon_{fs} = 10 \times 10^{-12}$ J/bit/m^2, and $\varepsilon_{mp} = 0.013 \times 10^{-12}$ J/bit/m^4.

3.5 Performance Analysis

The performance evaluation of the proposed method is carried out using both a simulated IoT testbed and a real-field dataset. First, the experimental conditions used for the performance evaluation are discussed. Thereafter, the results analysis is provided in terms of node prediction, QoS, and energy efficiency. In particular, node fault prediction is examined in terms of accuracy, precision, and recall values, while QoS is observed in terms of throughput ratio and average data latency. On the other hand, energy efficiency is measured using network lifetime and the energy balancing phenomenon.

3.5.1 Experimental Conditions

In this section, the experimental settings used to evaluate the performance of the proposed method are described.

3.5.1.1 Simulated IoT Testbed

Network size considered for simulation is 80 m × 80 m. In this area, a total of 104 sensor nodes (IoDs) are deployed, out of which 4 are gateways. IoDs are randomly distributed using uniform distribution, and gateways are located at coordinates (80,70), (80,50), (80,30), and (80,10). The transmission range of the IoDs is 20 m. IoDs transfer data to the gateway or neighbor IoD, directly or in a multi-hop manner, depending on their radio range connectivity. At each iteration of data transmission, all IoDs transfer the generated data to the gateway either directly or via a multi-hop fashion. The data packet sizes are 1100 bytes, 1200 bytes, and 1300 bytes for the nodes having 1 sensor, 2 sensors, and 3 sensors, respectively.

3.5.1.2 Real-field IoT Testbed

Experiments are also carried out on a real-field IoT testbed. The positions of sensor and gateway nodes and the dimensions of the network from the testbed are used. Faults have been generated randomly in the network by varying the data generation and latency of the nodes. The datasets are collected using the information provided by the 54 sensors deployed in the Intel Berkeley research lab [124]. The considered network size for the real-field dataset is 45 m $\times$ 35 m, and the total number of sensor nodes deployed is 56, including 2 gateways. Mica2Dot sensors with weatherboards collected timestamped topology information, along with humidity, temperature, light, and voltage values once every 31 seconds. Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform. Throughput ratio, average data latency, network lifetime, and residual energy of the network have been measured on this real-field IoT testbed.

3.5.2 Node Fault Prediction Analysis

The performance of the proposed method of node fault prediction is evaluated in terms of accuracy, precision, and recall values. These values are measured using node fault prediction measurement models given in **Section 3.4.1**.

3.5.2.1 Accuracy Performance

Accuracy performance analysis of the proposed method is performed through a comparison with different node fault prediction models, including the decision tree, SVM, and K-means. The results for the simulation and real-field dataset are illustrated in Figure 3.4a and Figure 3.4b, respectively. Figure 3.4a shows that the accuracy of the proposed method over a simulated IoT testbed achieves a value equal to 1. However, other methods achieve it closer to 1. Additionally, Figure 3.4b illustrates that the accuracy of the proposed method over the real-field dataset also attains a value equal to

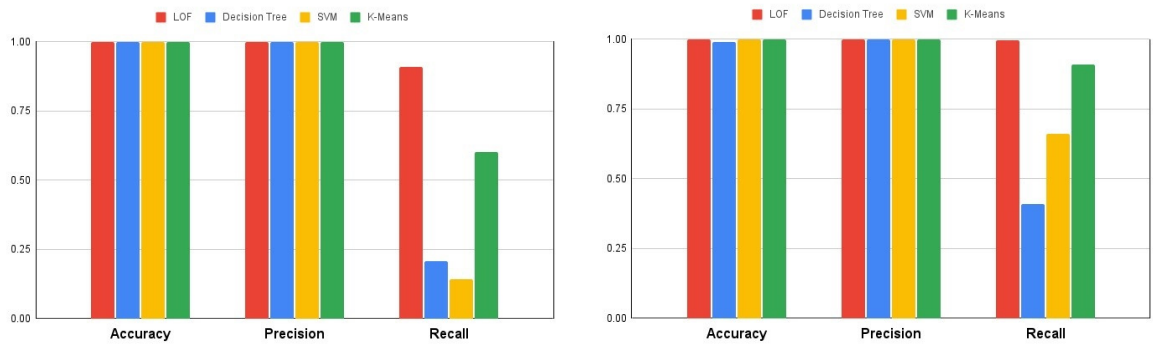

Figure 3.4: An illustration of node fault prediction analysis in terms of accuracy, precision, and recall values lying between [0, 1]. (a) performance analysis of all the methods over a simulated IoT testbed, and (b) performance analysis of all the methods over a real-world dataset

1, while other methods attain a value equal to or less than 1.

3.5.2.2 Precision Performance

Precision performance analysis of the proposed method is also carried out through a comparison with different node fault prediction models, such as decision tree, SVM, and K-means. Figure 3.4a and Figure 3.4b show that the precision value of the proposed method is equal to 1. However, other methods also achieve the precision value equal to 1 over both the simulated IoT testbed and the real-field dataset, respectively.

3.5.2.3 Recall Performance

The recall analysis for the simulation and real-field dataset is shown in Figure 3.4a and Figure 3.4b, respectively. From Figure 3.4a, it is observed that the proposed method has a higher value of recall (0.90) when compared to other node fault prediction models such as decision tree (0.22), SVM (0.18), and K-means (0.62). On the other hand, from Figure 3.4b, it is noted that the recall value is 0.99 using the proposed method. Whereas, this value is 0.40, 0.65, 0.90, using decision tree, SVM, and K-means, respectively. Henceforth, the proposed method shows better results in terms of recall over all other existing methods.

3.5.3 Quality-of-Service Analysis

To evaluate the performance of the proposed method in terms of QoS, parameters such as throughput ratio and average data latency are examined. In this, the proposed method is evaluated against a diverse set of well-established energy-efficient routing protocols to ensure a comprehensive performance comparison. BEEMH [119] represents balanced multi-hop routing strategies aimed at minimizing energy consumption and distributing the communication load evenly among nodes. LEACH [118] serves as a foundational baseline in hierarchical routing due to its historical significance and extensive adoption in wireless sensor networks. NM-LEACH [121] is a refined version of the widely adopted LEACH protocol, introducing an improved cluster head selection process to enhance network lifetime, thus representing cluster-based routing approaches. QL-EEBDG [122] incorporates Q-learning for energy-balanced routing, originally designed for underwater sensor networks but adapted here for IoT, enabling a direct comparison with reinforcement learning-based routing strategies. K-means++ [120] offers a clustering approach that optimizes energy efficiency by forming balanced clusters. The inclusion of these protocols, covering multi-hop, cluster-based routing, and reinforcement learning-based categories, allows for a robust and fair evaluation of the proposed method's efficiency in diverse IoT network conditions.

3.5.3.1 Throughput Ratio Performance

Figures 3.5a and 3.5b show the throughput ratio analysis for the simulated IoT testbed and real-field dataset, respectively. As shown in Figure 3.5a, on the simulated IoT testbed, the proposed method transfers a large number of data packets ($> 95\%$) to the gateway at the end of 2000 iterations. Whereas, under the same circumstances, fewer data packets are transmitted to the gateway using Q-learning (70%), LEACH (82%), BEEMH (58%), K-means++ (57%), NM-LEACH (80%), and QL-EEBDG (42%). Moreover, the throughput ratio of the proposed method is approx-

imately the same when compared to the direct transmission technique. Additionally, Figure 3.5b illustrates that the proposed method transfers a large number of data packets ($> 90\%$) to the gateway at the end of 2000 iterations on the real-field dataset. Whereas, under the similar conditions, less number of data packets are transmitted to the gateway using Q-learning ($< 75\%$), LEACH ($< 80\%$), BEEMH ($< 70\%$), K-means++ ($< 60\%$), NM-LEACH ($< 70\%$), and QL-EEBDG ($< 70\%$).

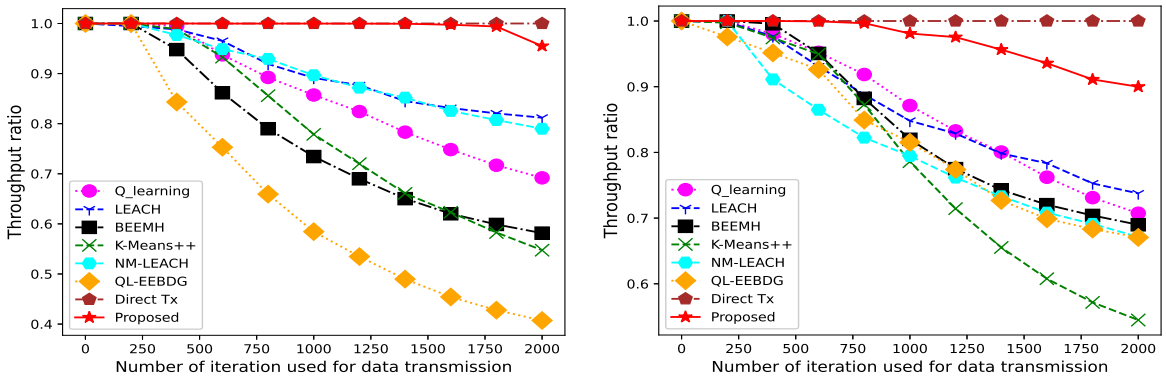

Figure 3.5: An illustration of the throughput ratio with varying number of iterations over the network. The throughput ratio is evaluated using the proposed method over (a) a simulated IoT testbed, (b) a real-field dataset and compared with the Q-learning, LEACH, BEEMH, K-means++, NM-LEACH, QL-EEBDG, and direct transmission methods

3.5.3.2 Average Data Latency Performance

Figure 3.6a and Figure 3.6b depict the average data latency for all transmissions during each cycle of data transfer over the simulated IoT testbed and real-field dataset, respectively.

Figure 3.6a illustrates that the proposed method has an average data latency of less than 8 seconds over the simulated IoT testbed at 2000 iterations. However, the average data latency for the QL-EEBDG, BEEMH, and Q-learning approaches is more than 12 s. The average data latency while using K-means++, NM-LEACH, LEACH, and direct transmission is minimal because there are fewer hops in the data transmission. Figure

3.6b illustrates that the proposed method has a data latency of 5.8 s, which is less than QL-EEBDG (12.2 s), Q-learning (9.8 s), and BEEMH (9.7 s) over the real field IoT testbed. However, LEACH, NM-LEACH, and K-means++ methods have slightly less average data latency in comparison to the proposed method because these methods do not use prior fault node prediction, which may cause data loss during data transmission.

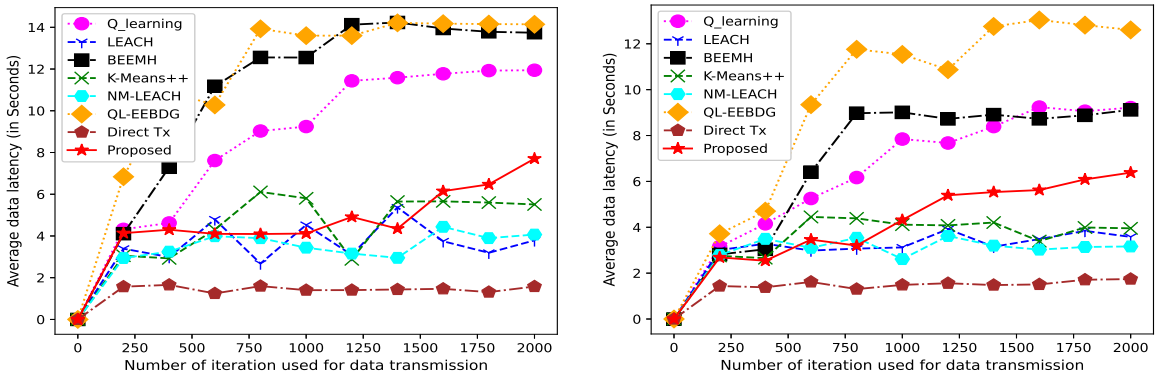

Figure 3.6: An illustration of the average data latency (in seconds) with varying number of iterations over the network. The average data latency is evaluated using the proposed method over (a) simulated IoT testbed, (b) real-field dataset, and compared with the Q-learning, LEACH, BEEMH, K-means++, NM-LEACH, QL-EEBDG, and direct transmission methods

3.5.4 Energy-Efficiency Analysis

The energy-efficiency of the proposed method is evaluated using three measures, i.e., network lifetime, residual energy of the network, and energy balancing across the network.

3.5.4.1 Network Lifetime Performance

The reliability of a network is measured by how many times data is sent to the gateways before the network fails. The energy consumption measurement model given in Section V-D is used to evaluate the network lifetime performance. The results for the simulated

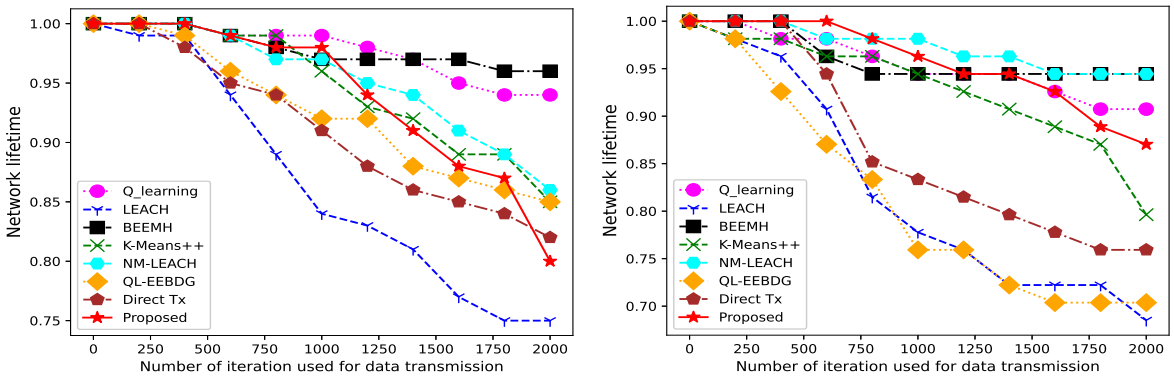

Figure 3.7: An illustration of the network lifetime with varying number of iterations. The total network lifetime is evaluated using the proposed method over (a) a simulated IoT testbed, (b) a real-field dataset and compared with the Q-learning, LEACH, BEEMH, K-means++, NM-LEACH, QL-EEBDG, and direct transmission methods

IoT testbed and the real-field dataset are shown in Figure 3.7a and Figure 3.7b, respectively. As shown in Figure 3.7a, the network lifetime of the proposed method is equal to or better than that of Q-learning, BEEMH, QL-EEBDG, K-means++, NM-LEACH, and direct transmission methods up to 1000 iterations. For 1000 to 1750 iterations, the network lifetime of the proposed method is still better than QL-EEBDG, direct transmission, and LEACH. However, Q-learning, BEEMH, and NM-LEACH have better results than the proposed method, as these methods do not use prior faulty nodes prediction, which may cause data loss during data transmission.

The network lifetime of the proposed method is more than 90% up to 1500 iterations. The results obtained over the real-field dataset are shown in Figure 3.7b. As shown in Figure 3.7b, the proposed method has a higher network lifetime in comparison with other existing methods such as LEACH, QL-EEBDG, direct transmission, and K-means++. The proposed method has 92% network lifetime up to 1750 iterations, which is more than the corresponding values for the LEACH (73%), QL-EEBDG (70%), direct transmission (77%), and K-means++ (87%).

3.5.4.2 Residual Energy Performance

The residual energy of a network is the total network energy, which is computed by adding the energies of all the nodes after each iteration of data transmission. Here,

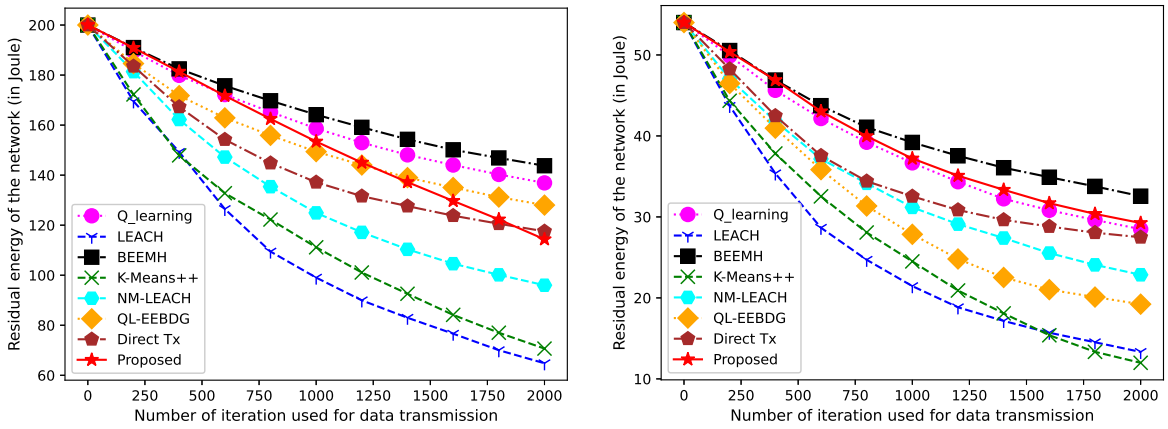

Figure 3.8: An illustration of total residual energy of the network with varying number of iterations. The total residual energy of the network evaluated using the proposed method over (a) simulated IoT testbed, (b) real-field dataset and compared with the Q-learning, LEACH, BEEMH, K-means++, NM-LEACH, QL-EEBDG, and direct transmission methods

the initial energy of the sensor node is 2 Joule (J). Figure 3.8a shows the variation of residual energy with varying number of iterations, over the simulated IoT testbed, while Figure 3.8b depicts the variation of residual energy over the real-field dataset.

From Figure 3.8a, it can be seen that the proposed method has 125 J of residual energy after 2000 iterations, which is 30% more than the LEACH protocol and 27.5% more than the K-means++ method, whereas the NM-LEACH has 8% less residual energy when compared to the proposed method after 2000 iterations. However, the corresponding values of residual energy for QL-EEBDG, direct transmission, Q-learning, and BEEMH methods are 130 J, 140 J, 150 J, and 160 J, respectively. Few methods have higher residual energy as these methods do not use prior faulty nodes prediction, which may cause data loss during data transmission. The results obtained from the real-field dataset are shown in Figure 3.8b. It can be seen that the proposed method

has higher residual energy (35 J) as compared to Q-learning (34 J), direct transmission (33 J), NM-LEACH (26 J), QL-EEBDG (22 J), LEACH (13 J), and K-means++ (12 J) methods. The BEEMH method has slightly higher residual energy (38 J), as the BEEMH method does not avoid faulty nodes.

3.5.4.3 Energy Balancing Performance

Energy balancing is evaluated in terms of the standard deviation (S.D.) of energy consumed by all the nodes at each iteration. Figure 3.9a depicts energy balance analysis for

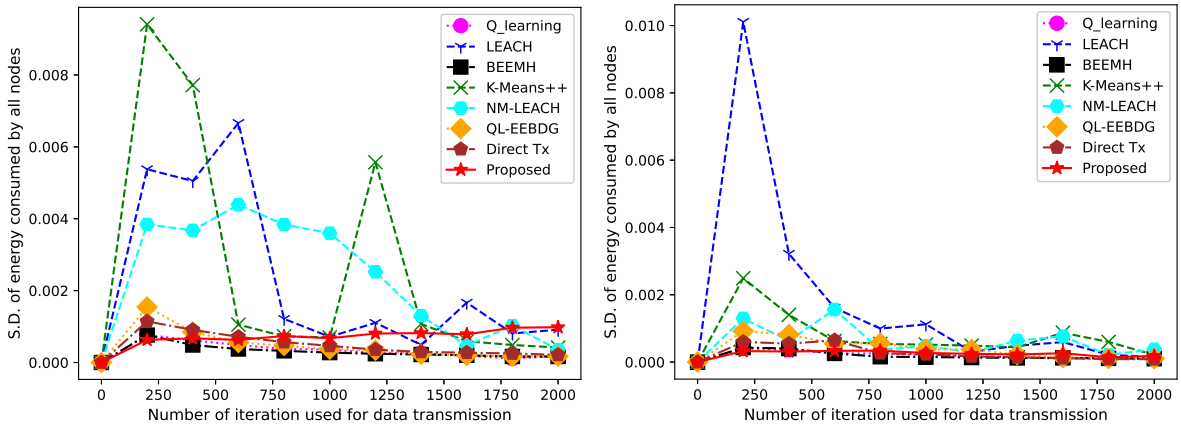

Figure 3.9: An illustration of variation of standard deviation (S.D.) of energy consumed over the network with varying number of iterations used for data transmission. The variation of S.D. of the proposed method is evaluated over (a) simulated IoT testbed, (b) real-field dataset, and compared with the Q-learning, LEACH, BEEMH, K-means++, NM-LEACH, QL-EEBDG, and direct transmission methods

the simulated IoT testbed. On the other hand, Figure 3.9b shows the energy balance analysis over the real-field dataset. Figure 3.9a illustrates that the proposed method has less variation in the S.D. of energy consumed by all the nodes at each iteration when compared to K-means++, LEACH, NM-LEACH, and QL-EEBDG methods, which shows balanced utilization of energy using the proposed method. The K-means++, LEACH, and NM-LEACH have higher S.D. of energy consumption due to long-range

data transmission. The results of the Q-learning, direct transmission, and BEEMH methods are close to the proposed method. Figure 3.9b shows that the LEACH method has a very high standard deviation of energy consumption for some nodes during the data transmission in the network. Hence, the performance of the proposed method is better than the LEACH, K-means++, NM-LEACH, QL-EEBDG, and direct transmission methods.

3.6 Conclusions

In this chapter, a novel method of joint node fault prediction and optimal data routing over IoT networks is proposed. The approach integrates an unsupervised local outlier factor (LOF) based framework for early identification of faulty nodes with a Q-learning based routing mechanism to ensure reliable and efficient data delivery. The method avoids data losses due to faulty nodes in the IoT network. The performance of the proposed method is evaluated over both a simulated testbed and a real-field dataset. The obtained results show that the proposed method achieves a throughput ratio of over 95% on the simulated IoT testbed and over 90% on the real-field dataset, significantly outperforming existing methods such as Q-learning (70% and <75%), LEACH (82% and <80%), and BEEMH (58% and <70%). From an energy efficiency perspective, the proposed method maintains a network lifetime above 90% up to 1500 iterations in the simulated testbed and 92% up to 1750 iterations in the real-field dataset. It also provides residual energy improvements of up to 30% over LEACH and 27.5% over K-means++, while ensuring balanced energy usage across nodes through a lower standard deviation of energy consumption than LEACH, K-means++, and NM-LEACH. Therefore, for next-generation IoT networks, the proposed method provides a robust, fault-tolerant, and energy-efficient routing solution capable of maintaining high performance even in fault-prone, resource-constrained environments.