Volume 3, Number 3, 2026 Pages 1-25

Bioscience Insights

ISSN: 3080-3489 (Print)
ISSN: 3080-3497 (Online)

BiosciInsights.com

Q-LEARNING-BASED TRUST-AWARE CLUSTERING AND ADAPTIVE ROUTING FOR WIRELESS SENSOR NETWORKS IN THE INTERNET OF THINGS: A SIMULATION STUDY

Asad Khan[1*], Adyan Ahmad[2], Muhammad Mustafa[3], Ahmad Saeed[4], Sana Naseer[5]

[*,1,2,4,5]*Department of Computer Science Bacha Khan University Charsadda*
[3]*University of Engineering and Technology, Peshawar*

[*1]asadkhan.cs@bkuc.edu.pk, [2]mradyanahmad@gmail.com, [3]mustafakhan2453@gmail.com, [4]ahmadsaeed@bkuc.edu.pk, [5]mahashmalinkhan@gmail.com

https://doi.org/10.5281/zenodo.23259799

Article Information

Article History

Received:	June 19, 2026
Revised:	August 19, 2026
Accepted:	September 07, 2026
Available Online:	October 02, 2026

Keywords:
Q-learning, Wireless sensor networks, Trust management, Cluster head selection, Internet of Things.

Abstract

Wireless sensor networks (WSNs) supply much of the data used in Internet of Things (IoT) systems, but their battery-powered nodes have limited energy and are exposed to security threats. This study describes a framework in which each sensor node uses Q-learning to support cluster head (CH) selection and routing. A dedicated Checkup Node (CN) estimates trust from direct and indirect observations and moves suspicious nodes first to a gray list and then to a blacklist. Trust is a condition for CH eligibility and one input to next-hop selection. The sink forms the initial clusters with K-means and a cost function based on energy, distance, and link quality. The framework was compared with the TEERP and RLER protocols in a custom Python simulator with 100 static nodes in a 100 m × 100 m area over 1000 rounds in a single run. By round 700, the proposed framework had used 800 J of the 1000 J network energy, compared with 750 J for TEERP and 700 J for RLER. It kept more nodes alive, with 75 nodes remaining at round 1000 against 10 and 47, and its mean trust over the logged rounds was 0.90 against 0.77 and 0.79. Healthcare monitoring is discussed as a motivating application, but no healthcare scenario or medical data were simulated. No attacks, node failures, or larger networks were simulated. The results therefore show that the design is feasible under benign conditions. They do not show improved security or scalability.

INTRODUCTION

The Internet of Things (IoT) links large numbers of devices that sense, process, and exchange data, often through protocols such as MQTT, ZigBee, and LoRa [1]. IoT systems are commonly described in layers that cover devices, networks, data processing, and applications [2]. Wireless sensor networks (WSNs) form the sensing layer of many of these systems. Their nodes collect measurements and relay them toward a collection point, and surveys report their use across a wide range of IoT and Industrial IoT (IIoT) applications [2], [3].

Such deployments face several persistent problems. Attacks such as Sybil, wormhole, and sinkhole attacks threaten the confidentiality, integrity, and availability of sensed data [4], [5]. Most sensor nodes run on batteries, so energy consumption limits how long a network can operate, and power management is a central design requirement [6]. Node lifetime decides how long the network can collect data without maintenance. Traffic also grows as more nodes are added. Context-aware designs, which adapt node behavior to current operating conditions, have been proposed as one response to this variation [7].

Healthcare monitoring shows why these problems matter together. Wearable devices and medical sensors allow patients to be observed in real time, and IoT-based monitoring systems have been designed for both rural and urban areas [8], [9]. Such sensors are battery powered, and energy-efficient protocols have been proposed for body area networks of health sensors for this reason [10]. Data security is one of the main obstacles to wider adoption [8], and the integrity of the readings matters as much as their confidentiality, because care decisions depend on them. Timely delivery is a further concern, since decisions in healthcare and emergency response are time sensitive. In this paper, healthcare serves only as a motivating example, and no healthcare scenario was evaluated.

Machine learning (ML) has been applied to these problems because it can find patterns in network data and support autonomous decisions [5], [11]. Reinforcement learning (RL) suits WSNs because an agent learns from rewards and penalties received while it interacts with its environment, without labeled examples [12]. Q-learning is a model-free RL method that stores the value of each state-action pair in a table and needs no model of the environment dynamics [13]. Deep RL can handle high-dimensional states, but it requires substantial computation, memory, and training time, and Proximal Policy Optimization (PPO) needs extensive hyperparameter tuning. These demands conflict with the limited energy and processing capacity of sensor nodes. For this reason, the present study uses tabular Q-learning, which is computationally lighter.

Two existing protocols define the research problem. The reinforcement-learning-based energy-efficient optimized routing protocol (RLER) combines grid-tree-based clustering with RL for energy-efficient root node selection and extends network lifetime [14]. Its scalability in dense networks is limited. RLER does not handle frequent sink mobility or hardware failures, does not consider Sybil and sinkhole attacks, and does not adjust routing paths and cluster head (CH) roles in real time. The trust-based energy-efficient routing protocol (TEERP) uses three-layer clustering and hardware-based link quality estimators to extend network lifespan and limit malicious activity [15]. Because its cluster configuration is fixed, it adapts poorly and can use resources unevenly. Its energy optimization across network layers is also limited, and the hardware-based estimators add computational overhead and latency.

This study develops a framework that combines Q-learning with awareness of node energy, traffic, congestion, buffer occupancy, and trust. First, it aimed to implement an RL agent for CH selection and routing in WSNs. Second, it aimed to maintain continuous trust assessment and anomaly scoring through a dedicated monitoring node. A third aim was to organize clustering and routing hierarchically so that the design could later be extended to larger networks. Finally, the framework was compared with TEERP and RLER in terms of energy consumption, trust, number of alive nodes, network lifetime, and response time. In the resulting design, each node learns CH selection and routing decisions with Q-learning, while the sink performs the initial clustering. A dedicated Checkup Node (CN) combines direct and indirect trust and isolates suspicious nodes in two stages, gray listing and then blacklisting. One trust estimate serves as a CH eligibility condition and as one input to next-hop selection. Results of the comparison with the two baselines are reported together with their limits.

LITERATURE REVIEW

IoT and WSN Foundations and Challenges

Surveys of IoT architectures describe the components, communication technologies, and protocols needed to manage large numbers of devices [16]. Work on WSN-IoT systems shows how these architectures support environmental monitoring and control [17]. Early studies defined the design space of WSNs and the technical problems of distributed sensing [18], [19], and later surveys of design methods still stress connectivity, interoperability, and energy consumption [20]. Integration with IoT technologies has extended what WSNs can do and how their data are managed [2], [3]. A typical WSN consists of sensor nodes, communication protocols, and a base station. Recent designs add cloud or edge processing and hybrid topologies to improve scalability [3], [17].

Security and privacy remain major concerns, because distributed IoT systems and WSNs face many kinds of attacks [4], [5]. Energy-efficient protocols such as Low-Energy Adaptive Clustering Hierarchy (LEACH) reduce the energy spent on data transmission [21]. Clustering has been studied widely as a way to improve the performance, lifetime, and resource management of WSNs [22]. Processing and storing the large volume of data that sensor networks produce is a further difficulty [11].

Machine Learning for IoT Networks

A survey of ML in real-time IoT systems covers deep learning for task scheduling, anomaly detection, and resource management, including deep RL models that adapt to changing conditions [23]. Deep learning for IoT resource management has also been reviewed, including deep Q-learning, which approximates Q-values with neural networks to handle complex, high-dimensional states [24]. For WSN security, an intrusion detection scheme based on C4.5 and CART classifiers has been proposed to detect and classify attacks [25]. In a comparison of classifiers for malicious agricultural IoT traffic, random forests and support vector machines gave the highest accuracy, although they took longer to run [11].

Reinforcement Learning and Trust-Based Routing in WSNs

RL has been studied as a basis for routing and resource management in IoT environments [26]. RLER combines RL with grid-tree-based clustering and a hybrid BAT-Crow search algorithm to reduce energy use and extend network lifetime [14]. In this setting, Q-learning estimates the long-term value of each action available to a sensor node without a model of the environment. TEERP takes a different approach and adds trust metrics to routing, so that energy-efficient communication is restricted to reliable nodes [15].

The two protocols have complementary strengths and weaknesses. By reducing the participation of compromised nodes, TEERP improves security. Limiting routing to trusted nodes may, however, prevent it from finding the best path, and its centralized trust handling becomes complex as the network grows. RLER adapts its routes to energy levels and network conditions and may reach higher long-term energy efficiency, but it is computationally demanding and may need additional security mechanisms. More generally, complex ML algorithms are hard to run on constrained WSN nodes. Their accuracy depends on training data, which are often scarce in WSN applications, and many of them need more processing time than real-time operation allows.

Several studies have applied Q-learning directly to WSN routing and clustering. Yun and Yoo used Q-learning to select next hops while taking data aggregation into account [27]. In the scheme of Su et al., each sensor updates the Q-values of its neighbors from residual energy, transmission distance, and direction to balance energy consumption [28]. Kim et al. applied Q-learning to parent selection in tree-based routing [29]. For clustering, Bourebia and Li proposed a greedy RL-based clustering scheme [30], and Sharma et al. combined clustering with RL in ReLeC to extend network lifetime [31]. A broader survey of RL and deep RL for the wireless IoT is given in [32]. Trust is usually handled by separate mechanisms. Keum and Ko learned trust elements with Q-learning to choose trusted paths in mission-critical WSNs [33], and SMTrust used trust and mobility metrics to counter rank and blackhole attacks in IoT networks that run the Routing Protocol for Low-Power and Lossy Networks (RPL) [34]. In the works reviewed here, learning-based clustering and trust-based routing are addressed in separate

schemes, whereas the present design applies one trust estimate to both CH eligibility and next-hop selection. Table 1 compares these schemes with the proposed framework as they are described in the cited works.

Table 1. Learning method, decisions addressed, and trust or security element of related WSN schemes and the proposed framework, as described in the cited works. A blank cell means that the attribute is not stated in that description.

Scheme	Learning method	Decisions addressed	Trust or security element
RLER [14]	RL with hybrid BAT-Crow search	Root node selection and routing with grid-tree clustering	Sybil and sinkhole attacks not considered
TEERP [15]		Routing with fixed three-layer clustering	Trust metrics and hardware-based link quality estimators
Yun and Yoo [27]	Q-learning	Next-hop selection with data aggregation	
Su et al. [28]	Q-learning	Next-hop selection from residual energy, distance, and direction	
Kim et al. [29]	Q-learning	Parent selection in tree-based routing	
Bourebia and Li [30]	RL	Greedy energy-efficient clustering	
ReLeC [31]	RL	Clustering and routing to extend lifetime	
Keum and Ko [33]	Q-learning	Trusted path selection	Learned trust elements
SMTrust [34]		Secure routing in RPL-based IoT	Trust and mobility metrics against rank and blackhole attacks
Proposed framework	Q-learning at each node	CH selection and next-hop selection after K-means clustering by the sink	CN trust from direct and indirect observation, gray list, and blacklist

Research Gap

The reviewed work points to several open needs. Learning methods must run in real time within IoT systems, and energy-saving mechanisms must adapt to changing network conditions. Security protocols must respond to new threats, and solutions must stay efficient as networks grow. Algorithms also need to be light enough for constrained nodes, and hybrid approaches, including edge computing, can move processing away from sensor nodes. Existing protocols address only some of these needs. RLER offers adaptive, learning-based routing without specific defenses against Sybil and sinkhole attacks, while TEERP provides trust-based security within a fixed cluster structure. As described in the cited works, the Q-learning schemes reviewed above target either routing or clustering without a trust model shared between the two. The framework presented here combines node-level Q-learning for CH selection and routing with continuous trust monitoring by a dedicated CN.

RESEARCH METHODS

Research Framework

The study consisted of a literature review, the design of an RL-based framework for WSN operation, its implementation in a simulator, and a comparison with TEERP and RLER. Figure 1 shows the operational flow. After deployment, each node computes its initialization parameters and reports them to the sink node. The sink calculates cost function values, performs K-means clustering, and selects the CHs and the CN. The CN then initializes the trust values, and the nodes begin sensing and transmitting data. A CH aggregates and forwards data, and a cluster member (CM) forwards its data to its CH. Meanwhile, the CN runs the monitoring and blacklisting procedure shown in Figure 2. Policies and Q-values are updated at each stage.

Network Architecture and System Model

Sensor nodes with sensing, computation, and communication capabilities were placed at random locations. A single sink node at a fixed location collects data and exchanges information with external networks. Nodes are grouped into clusters, and each cluster is coordinated by a CH that collects data from its CMs and forwards it to the sink. CH roles are assigned dynamically through the RL framework to balance energy use and load, and a CN monitors network performance and security through trust assessment. Communication is multi-hop, and time division multiple access (TDMA) is used to reduce collisions. The energy model covers sensing, processing, and transmission.

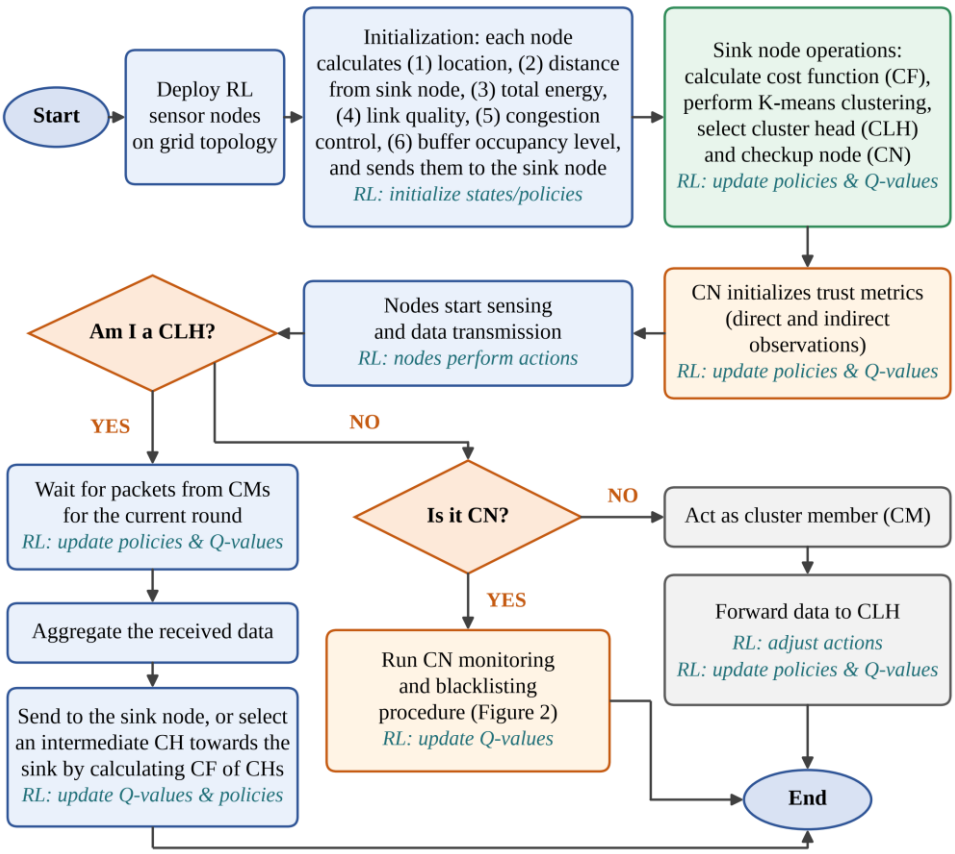


Figure 1. Operational flow of the proposed framework, from node initialization, clustering, and CH and CN selection by the sink to data transmission, CN monitoring, and Q-value updates.

Application Context and Scope

Healthcare monitoring is used in this paper as a motivating application, and its requirements go beyond the conditions that were simulated. A network that monitors patients would need to give urgent readings priority over routine ones and to deliver them within fixed delay limits. It would also have to cope with nodes that move with the patient, follow health-sector privacy standards for patient data such as the Health Insurance Portability and Accountability Act (HIPAA), and resist nodes that inject false

data. None of these requirements was simulated. Nodes were static, all traffic was handled alike, no delay limit was imposed, and no false data were injected. As an illustration only, some parts of the design relate to these needs. Trust monitoring by the CN is intended to isolate nodes whose behavior departs from the expected pattern, CH re-election by residual energy is intended to spread the energy load, and the dynamic resource allocation mechanism gives real-time data priority over periodic data. Whether any of these would meet clinical requirements was not tested and is not claimed.

Q-Learning Formulation

Each sensor node, whether a CM, CH, or CN, is modeled as an agent. Its state is described by its residual energy, cost function (CF) value, role, position in the grid, buffer occupancy, and congestion status, together with the energy, CF, and trust values of its neighbors and any malicious activity it has detected or suspects among them. The CH selection and routing procedures use small discrete state sets. For CH selection these are high energy, low energy, idle, and active, and for routing they are high traffic, low traffic, high energy, and low energy. Available actions are data transmission to the CH, participation in CH selection, recalculation of the CF value, monitoring (CN only), and routing decisions. Rewards are given for efficient transmission with low energy use and no packet loss, for energy conservation, for detection of malicious nodes by the CN, and for CF calculations that lead to a suitable CH. Penalties apply to unnecessary energy use, packet loss, misrouting, and failure to detect malicious activity. Table 2 lists the reward values, which were set by hand.

Each node keeps its own Q-table and updates it with the Q-learning rule in Equation (1) [13]. In this rule, $Q(s,a)$ is the value of action a in state s, α is the learning rate, r is the immediate reward, γ is the discount factor, and $\max_{a'} Q(s',a')$ is the highest Q-value available in the next state s'. Actions are chosen with an ϵ-greedy strategy, where ϵ is the exploration rate. The policy then selects, in each state, the action with the highest Q-value. Throughout the paper, α, γ, and ϵ refer only to these learning parameters.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

The objective of every agent is to use energy efficiently through suitable CH selection, to route data over trusted paths, to detect and limit malicious behavior, and to keep nodes active for as long as possible.

Table 2. Reward and penalty values assigned to events and decisions in the Q-learning scheme of the proposed framework. The values were set by hand.

Event or decision	Reward/penalty	Rationale
Selecting a CH with the highest trust and residual energy	+10	Encourages energy-aware and secure leadership
Successful packet routing through trusted paths	+5	Reinforces reliable communication
Identifying and isolating a malicious node	+8	Promotes active security enforcement
Selecting a malicious or energy-drained node as CH	−10	Prevents insecure leadership
Failed routing or repeated retransmissions	−5	Discourages unstable paths
CN confirms indirect misbehavior via trust degradation	+4	Rewards collaborative monitoring
CN fails to detect malicious activity	−4	Penalizes oversight or delayed response

Topology Initialization and Cluster Formation

During initialization, each node determines its coordinates by GPS or by trilateration from three reference nodes. It then computes its Euclidean distance dis_i to the sink and reports its initial energy E_i. The sink records the total initial energy of the network as the sum of these values. Link quality

between two nodes is assessed from the received signal strength indicator (RSSI), where higher values indicate better links, and from the packet delivery ratio (PDR), the share of packets sent by one node that the other receives. The link quality LQ_i of node i summarizes the RSSI and PDR values measured with its neighbors. Congestion is expressed as the current queue length divided by the maximum queue length, and buffer occupancy as the number of stored bytes divided by the buffer capacity. Values close to 1 indicate heavy congestion or a nearly full buffer.

All parameters are sent to the sink, which arranges them in one vector per node containing the coordinates, distance to the sink, link quality, congestion, and buffer occupancy. The sink then applies K-means clustering. It initializes K centroids at random, assigns each node to the cluster with the nearest centroid, and recomputes each centroid as the mean of the vectors assigned to it. These two steps are repeated until the assignments stop changing or change very little. In what follows, G_k denotes the set of nodes assigned to cluster k.

Within each cluster, the node with the highest CF value becomes the initial CH. The CF of node i combines residual energy, distance to the sink, and link quality, as shown in Equation (2), where λ_1, λ_2, and λ_3 are weights. Distance enters with a negative sign so that nodes far from the sink receive a lower score.

$$CF_i = \lambda_1 E_i - \lambda_2\, dis_i + \lambda_3\, LQ_i \tag{2}$$

The CN is selected by the sink from the nodes of a cluster. Its selection score combines residual energy with the strategic importance of the node's position, such as its centrality or connectivity, through two further weights.

Dynamic Cluster Head Selection

The sink performs K-means clustering and assigns the initial CHs at deployment. Later CH changes are made within each cluster. In the RL-based CH selection procedure, each node initializes its Q-table to zero, sets α, γ, and ϵ, and chooses between two actions, becoming a CH or not, in the states high energy, low energy, idle, and active. A node that becomes a CH is rewarded according to factors such as its energy and its proximity to the base station. Remaining a regular node is rewarded for saving energy. Each node observes its state, selects an action with the ϵ-greedy policy, executes it, observes the new state, and updates its Q-value with Equation (1). These steps repeat for a fixed number of rounds or until convergence, and CH selection is reviewed periodically. Algorithm 1 lists the procedure.

The Q-value used in CH selection takes residual energy, trust, node degree, congestion level, and buffer occupancy into account. Trust acts as an eligibility condition. Only a node whose trust T_i is at or above the threshold T_{th}, set to 0.75 in this study, can become a CH. Each node periodically recomputes its CF with the weights and signs of Equation (2), which gives the updated value $CF_i(t)$ at round t. A re-election starts when the energy of the current CH falls below a predefined threshold. Each node then compares its updated CF with those of its eligible neighbors and votes for the best candidate, so that the new CH of cluster k is given by Equation (3).

$$CH_{\text{new}} = \text{argmax}_{i \in G_k,\, T_i \geq T_{\text{th}}} CF_i(t)$$

Algorithm 1. RL-based CH selection performed by each sensor node.

1. Initialize the Q-table with $Q(s, a) = 0$ for all states and actions.
2. Set the learning rate α, the discount factor γ, and the exploration rate ϵ.
3. Define the actions {become CH, not become CH}.
4. Define the states {high energy, low energy, idle, active}.
5. Define the rewards: R_{CH} for becoming a CH, based on factors such as energy and proximity to the base station, and R_{nonCH} for not becoming a CH, based on energy conservation.
6. For each node in the network:
7. Observe the current state from the energy level and status of the node.
8. Select an action with the ϵ-greedy policy: with probability ϵ choose a random action, otherwise choose the action with the highest $Q(s, a)$.
9. Execute the action: become a CH and begin gathering data, or stay a regular node.
10. Observe the new state after energy consumption and data transmission.
11. Update $Q(s, a)$ with Equation (1).
12. Repeat steps 6 to 11 for a fixed number of rounds or until convergence.
13. Re-evaluate CH selection periodically from network conditions and energy consumption.

Trust Management by the Checkup Node

The CN is selected as described above and is placed at a hierarchical level of the network, for example between the sink and the clusters or in a high-traffic region. Depending on network scale and risk exposure, the design allows one or two CNs per network. Every node, including the CN, starts with 10 J, so the CN differs from other nodes in its role rather than in its battery. Five functions are assigned to it. First, it monitors packets from CHs and members directly and records delay, drop rates, and retransmissions. Second, it assesses trust indirectly from long-term behavior logs, such as acknowledgment consistency, route hops, and patterns of data deviation. Third, it detects and isolates malicious nodes and passes warnings to the rest of the network. When CH performance degrades, the CN also supports re-election and may take the CH role for a time. Its fifth function is to advise the routing layer to prefer paths with higher trust and stable energy. Figure 2 shows the full monitoring procedure.

Trust is built from direct and indirect observations. The direct observation DO_{ij} of node j by node i is a weighted sum of data transmission reliability DTR_{ij}, energy consumption EC_{ij}, and response to routing requests RR_{ij}, with weights ω_1, ω_2, and ω_3, as in Equation (4).

$$DO_{ij} = \omega_1 DTR_{ij} + \omega_2 EC_{ij} + \omega_3 RR_{ij} \tag{4}$$

The indirect observation IO_{ij} is the average of the direct observations of node j reported by the other nodes. Equation (5) combines the two into the trust value T_j of node j, with weight ρ on direct and $1 - \rho$ on indirect evidence. This combination is used when trust is initialized and again in each round with the latest observations. The framework defines a scaling of trust values to the range 0 to 1 using the minimum and maximum trust values observed in the network, but whether this scaling was applied in the reported run was not recorded. Once computed, the trust values are broadcast by the CN to all nodes, and each node updates its trust table.

$$T_j = \rho\, DO_{ij} + (1 - \rho)\, IO_{ij} \tag{5}$$

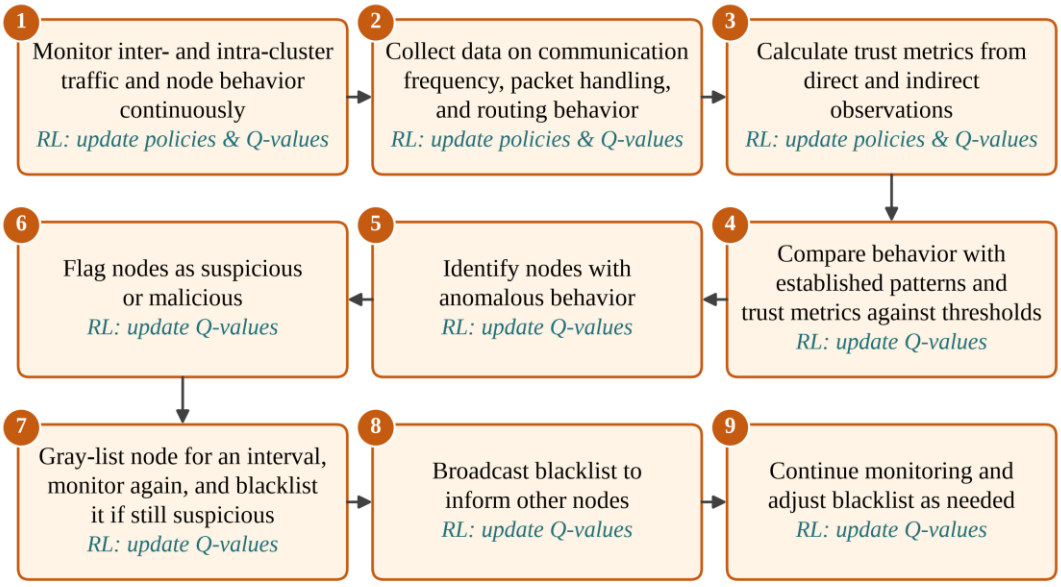


Figure 2. Monitoring, trust evaluation, and blacklisting procedure executed by the Checkup Node (CN).

Anomaly Detection, Gray Listing, and Blacklisting

The CN monitors traffic between pairs of nodes and collects behavioral data D_i for each node, consisting of its communication frequency, packet handling rate, and routing behavior pattern. An anomaly score A_i compares the observed behavior with the expected behavior D_{expected}, relative to the standard deviation σ_D of the behavior data. In its adaptive form, given in Equation (6), a small factor η derived from RL feedback is added to the denominator.

$$A_i = \frac{|D_i - D_{\text{expected}}|}{\sigma_D + \eta}$$

When A_i exceeds a threshold θ, the node is flagged as suspicious and placed on a gray list, where it is watched more closely for a set interval. A node whose score has not fallen below θ by the end of this interval is moved to the blacklist. Nodes whose trust falls below the trust threshold are also treated as malicious. The CN broadcasts the blacklist to all nodes. Blacklisted nodes may no longer transmit or route data, and every node removes them from its routing table. The threshold θ is adjusted by an amount derived from recent threat observations, and nodes also apply the Q-learning update to adapt their trust evaluation policies. Algorithm 2 summarizes the monitoring and blacklisting procedure.

Algorithm 2. Monitoring, trust update, and blacklisting performed by the CN.

1. Monitor the traffic between each pair of nodes, including communication frequency, packet handling, and routing behavior.
2. Update the trust value of each node from direct and indirect observations with Equation (5).
3. Compute the anomaly score A_i of each node by comparing its current behavior with its expected behavior, as in Equation (6).
4. If $A_i > \theta$, flag node i as suspicious and add it to the gray list.
5. If A_i has not fallen below θ within the evaluation interval, move node i from the gray list to the blacklist.
6. If $T_i < T_{th}$, add node i to the blacklist.
7. Broadcast the blacklist to all nodes.
8. Each node removes the blacklisted nodes from its routing table.

Adaptive Routing

Routing decisions are based on Q-values. The routing procedure uses the states high traffic, low traffic, high energy, and low energy, and the actions forward data, wait, and change route. Forwarding on an optimal path earns a reward. Waiting when no optimal route is available and forwarding on a congested path are penalized. To choose the next hop, a node considers the Q-values of candidate neighbors, which reflect trust and energy, together with hop count, signal strength, the path success ratio recorded by the CN, and load balancing to avoid congested nodes. It then selects the candidate with the highest Q-value. Apart from the exclusion of blacklisted nodes, trust therefore enters next-hop selection as one input among several. Each routing path is updated from its previous state, the current energy of the node, and any nodes added to or removed from the network. Algorithm 3 lists the routing procedure.

Algorithm 3. RL-based adaptive routing performed by each sensor node.

1. Initialize the Q-table with $Q(s, a) = 0$ for all states and actions.
2. Set the learning rate α, the discount factor γ, and the exploration rate ϵ.
3. Define the actions {forward data, wait, change route}.
4. Define the states {high traffic, low traffic, high energy, low energy}.
5. Define the rewards: $+R_{\text{forward}}$ for forwarding data on the optimal path, $-R_{\text{wait}}$ for waiting when no optimal route is available, and $-R_{\text{congestion}}$ for forwarding data on a congested path.
6. For each node:
7. Observe the current state from the traffic load and energy level.
8. Select an action with the ϵ-greedy policy: with probability ϵ choose a random action, otherwise choose the action with the highest $Q(s, a)$.
9. Execute the action: transmit data to the next hop or base station, delay transmission until an optimal path is found, or reroute data to a less congested path.
10. Observe the new state from the updated traffic conditions and energy use.
11. Update $Q(s, a)$ with Equation (1).
12. Repeat steps 6 to 11 for a fixed number of rounds or until convergence.
13. Re-evaluate routing decisions periodically from updated network conditions and energy levels, so that nodes adjust their routes to current feedback.

Supporting Mechanisms

The design also includes several supporting mechanisms. For fault tolerance, each node sends periodic heartbeat signals to its neighbors. A node that does not respond within a timeout is marked as failed, and the CH or base station is informed. Routing tables are then updated, and the tasks of the failed node pass to neighbors with enough energy. A redundancy mechanism keeps backup nodes and alternative routes for critical nodes. Another mechanism monitors node energy, link quality, and traffic load and reroutes data before a disruption occurs. Dynamic resource allocation moves tasks away from low-energy nodes, gives real-time data priority over periodic data, and adjusts bandwidth to traffic. Edge devices placed near CHs for local filtering, aggregation, and preliminary analysis are also part of the design. None of these mechanisms was evaluated, because edge and fog integration was not simulated and no node failures were injected.

Notation and Recorded Parameter Values

Table 3 lists the symbols used in Equations (1) to (6) and Algorithms 1 to 3, with the value recorded for the reported run where one exists. The recorded values are the number of nodes, the initial node energy, and the trust threshold, while the packet size and energy coefficients appear in Table 4. Table 3 marks every other value as not recorded.

Table 3. Symbols used in Equations (1) to (6) and Algorithms 1 to 3, with their meaning and the value recorded for the reported run. Computed means that the quantity is calculated during operation.

Symbol	Meaning	Value in the run
$Q(s,a)$	Value of action a in state s; s' is the next state	Learned, initialized to 0
r	Immediate reward	Table 2
α, γ, ϵ	Learning rate, discount factor, exploration rate	Not recorded
E_i	Residual energy of node i	10 J at deployment
dis_i	Euclidean distance from node i to the sink	Computed
LQ_i	Link quality of node i from RSSI and PDR	Computed
$\lambda_1, \lambda_2, \lambda_3$	Weights of energy, distance, and link quality in the CF	Not recorded
$CF_i, CF_i(t)$	Cost function of node i, and its value at round t	Computed
K, G_k	Number of clusters; set of nodes in cluster k	K not recorded
T_i, T_j	Trust value of a node	Computed
T_{th}	Trust threshold for CH eligibility and blacklisting	0.75
DO_{ij}, IO_{ij}	Direct and indirect observation of node j by node i	Computed
$DTR_{ij}, EC_{ij}, RR_{ij}$	Transmission reliability, energy consumption, response to routing requests	Computed
$\omega_1, \omega_2, \omega_3$	Weights of the direct observation	Not recorded
ρ	Weight of direct against indirect evidence	Not recorded
$D_i, D_{expected}, \sigma_D$	Observed and expected behavior data, and their standard deviation	Computed
η	Small adaptive factor derived from RL feedback	Not recorded
A_i, θ	Anomaly score of node i; anomaly threshold	θ not recorded
N	Number of sensor nodes	100

Experimental Setup

The framework was evaluated in a custom discrete-event simulator written in Python in Jupyter Notebook. Standard simulators such as NS-3, OMNeT++, CupCarbon, and TOSSIM were not used, because they do not natively support distributed Q-learning agents, dynamic CH and CN roles, decentralized trust evaluation with indirect monitoring, or reward and penalty adaptation during a run. Table 4 summarizes the configuration. TEERP and RLER were run as routing schemes in the same simulator with the parameters in Table 4. Five metrics were recorded: remaining network energy, average trust, number of alive nodes, network lifetime, and response time.

Remaining energy and average trust were logged every 100 rounds up to round 700, and the number of alive nodes every 100 rounds up to round 1000, with no value for round 600. Network lifetime is taken here as the half-nodes-dead (HND) point, the round by which 50 of the 100 nodes have run out of energy, and the first node death (FND) is also noted. The simulation output also contains a rounds-survived value for each scheme, 900 rounds for TEERP, 920 for RLER, and 950 for the proposed framework. Its definition was not recorded, and it does not agree with the alive-node counts, which show 75 nodes of the proposed framework still alive at round 1000, so it is not used. Response time was recorded as one value per scheme, but the procedure used to measure it was not recorded. Data management efficiency and scalability were recorded only as qualitative ratings of high or medium, not as measured quantities, so neither is reported as a result. All results come from one run with a single random deployment.

Table 4. Simulation environment, input parameters, and run conditions used to compare TEERP, RLER, and the proposed framework.

Aspect	Setting
Environment and language	Jupyter Notebook; Python 3.x
Libraries	NetworkX, Random/Math, NumPy, Matplotlib, Pandas
Simulator type	Discrete event-based custom simulator
Number of nodes	100 (uniform/random placement)
Deployment area	100 m × 100 m (2D static grid)
Network layers	Node, CH, CM, CN, sink
Simulation rounds	1000
Routing schemes	TEERP, RLER, proposed RL-based framework
Initial node energy	10 J
Packet size	4000 bits
Transmission energy (E_{tx})	50 nJ/bit
Aggregation energy	5 nJ/bit/signal
Trust threshold	0.75
Rewards / penalties	+10, +8, +5, +4 / −10, −5, −4 (see Table 2)
CH selection criteria	Residual energy, trust (eligibility), node degree, congestion, buffer occupancy
CN selection criteria	Residual energy and position
Routing decisions	Q-value optimization
Independent runs	1 (single random deployment)
Attack scenario	Not simulated
Node failure injection	Not simulated
Lifetime definition	Half of nodes dead (HND)

RESULTS AND DISCUSSION

Energy Consumption

Table 5 and Figure 3(a) give the remaining network energy from round 0 to round 700, starting from 1000 J (100 nodes with 10 J each). The proposed framework had the lowest remaining energy at round 100 and at every logged round from round 400 onward. At round 700 it retained 200 J, compared with 250 J for TEERP and 300 J for RLER. It had therefore consumed 800 J by round 700, against 750 J for TEERP and 700 J for RLER. In the first 100 rounds alone, it consumed 180 J, against 150 J for TEERP and 130 J for RLER. This higher consumption may reflect the processing and monitoring added by the learning scheme, but energy per component was not logged, so this could not be tested, and the consumption cannot be divided among sensing, transmission, aggregation, monitoring, and Q-table updates.

Table 5. Remaining network energy (J) at each logged round for TEERP, RLER, and the proposed framework in the single simulation run (100 nodes with 10 J each, 1000 J in total).

Round	0	100	200	300	400	500	600	700
TEERP	1000	850	700	600	590	480	350	250
RLER	1000	870	750	680	620	500	400	300
Proposed	1000	820	720	670	500	400	300	200

Average Trust

Table 6 and Figure 3(b) give average trust from round 0 to round 700. Trust in the proposed framework rose from 0.80 to 0.97, whereas TEERP rose from 0.72 to 0.84 and RLER from 0.72 to 0.86. Averaged over the eight logged rounds, trust was 0.90 for the proposed framework, 0.77 for TEERP, and 0.79 for RLER. The higher values agree with the continual trust assessment performed by the CN. These values are produced by the framework's own trust model inside the simulation. Because no attack scenario with known malicious nodes was run, they show how node interactions were scored, not how well malicious nodes were detected. Detection and false-positive rates could therefore not be computed.

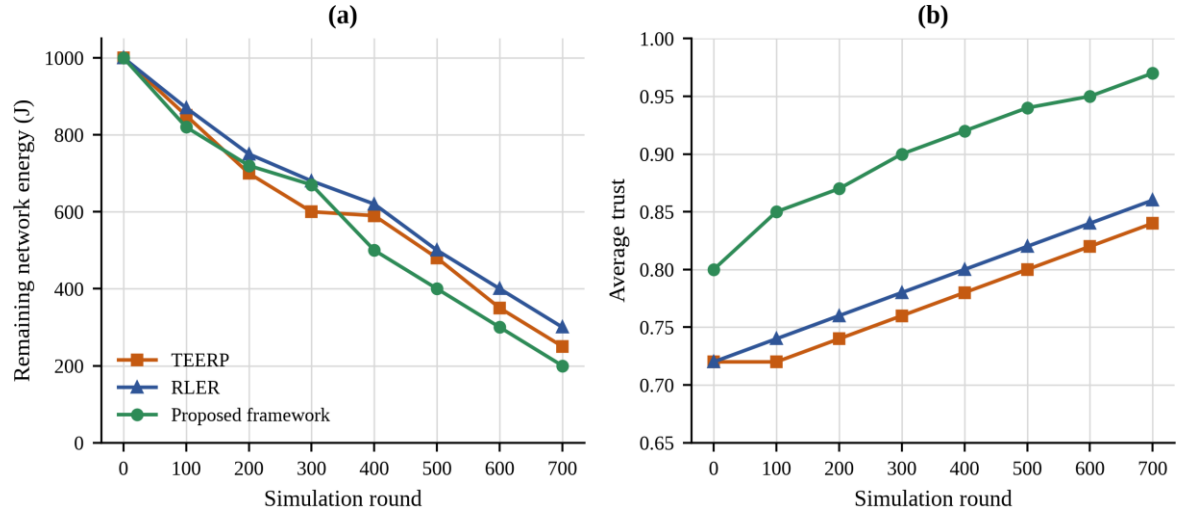


Figure 3. Single-run results for TEERP, RLER, and the proposed framework from round 0 to round 700: (a) remaining network energy (J), plotted from Table 5, and (b) average trust on a scale of 0 to 1, plotted from Table 6.

Table 6. Average trust (scale 0 to 1) at each logged round for TEERP, RLER, and the proposed framework in the single simulation run.

Round	0	100	200	300	400	500	600	700
TEERP	0.72	0.72	0.74	0.76	0.78	0.80	0.82	0.84
RLER	0.72	0.74	0.76	0.78	0.80	0.82	0.84	0.86
Proposed	0.80	0.85	0.87	0.90	0.92	0.94	0.95	0.97

Number of Alive Nodes and Network Lifetime

The proposed framework had more alive nodes than both baselines at every logged round after round 0 (Table 7 and Figure 4(a)). This gap grew in later rounds. At round 500 the proposed framework had 90 alive nodes, against 75 for TEERP and 85 for RLER, and at round 1000 it had 75, against 10 and 47. Averaged over the ten logged rounds, the number of alive nodes was 89.2, 68.5, and 81.1, respectively. Under the HND definition, TEERP reached the end of its lifetime at round 800 and RLER between rounds 900 and 1000. The proposed framework did not reach the HND point within the 1000 simulated rounds. In all three schemes the first node died before round 100, so FND does not distinguish them. The design is meant to spread the load through CH re-election based on residual energy. Energy per node was not logged, so it could not be checked whether this explains the larger number of surviving nodes.

Table 7. Number of alive nodes (of 100) at each logged round for TEERP, RLER, and the proposed framework in the single simulation run. No value was logged for round 600.

Round	0	100	200	300	400	500	700	800	900	1000
TEERP	100	95	90	85	80	75	65	50	35	10
RLER	100	97	94	91	88	85	79	70	60	47
Proposed	100	98	96	94	92	90	85	82	80	75

Response Time

Response time was 0.05 s for the proposed framework and RLER and 0.06 s for TEERP (Figure 4(b)). The measurement procedure was not recorded, so these values are reported as given and are not used to draw conclusions about delay.

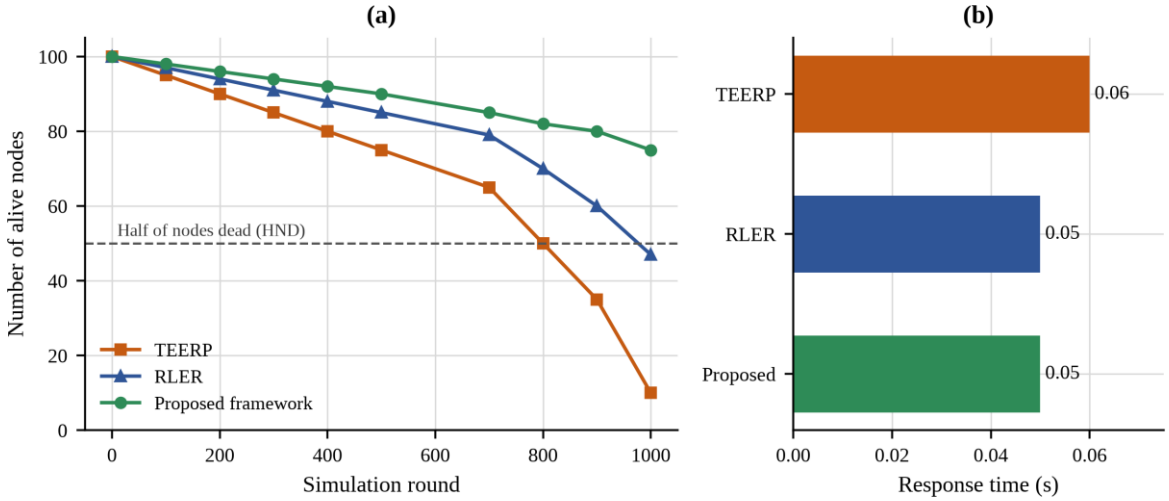


Figure 4. Single-run results for TEERP, RLER, and the proposed framework: (a) number of alive nodes (of 100) at each logged round, plotted from Table 7, with the dashed line marking the half-nodes-dead level of 50 nodes, and (b) response time (s) as recorded for each scheme.

Comparative Summary

Table 8 collects the quantitative results. Every value in it is taken or computed directly from Tables 5 to 7 or from the recorded response times. By round 700 the proposed framework had used more energy than both baselines. It also kept more nodes alive, reached a higher average trust, and outlasted the HND point of both baselines, while its response time matched that of RLER.

Table 8. Summary of the single-run comparison of TEERP, RLER, and the proposed framework. All values are taken or computed from Tables 5 to 7 and the recorded response times.

Metric	Proposed	TEERP	RLER
Energy consumed by round 700	800 J	750 J	700 J
Mean trust (rounds 0 to 700)	0.90	0.77	0.79
Trust at round 700	0.97	0.84	0.86
Alive nodes at round 1000	75	10	47
Mean alive nodes (logged rounds)	89.2	68.5	81.1
Network lifetime (HND)	Not reached by round 1000	800 rounds	900 to 1000 rounds
Response time	0.05 s	0.06 s	0.05 s

Discussion

The results fit the design rationale drawn from the literature. RLER extends lifetime through learning-based routing but has no specific defenses against Sybil and sinkhole attacks [14]. TEERP adds trust-based security within a fixed cluster structure and incurs overhead from hardware-based link estimation [15]. Q-learning routing schemes such as [27] and [28] focus on energy efficiency, and the trust-learning scheme in [33] does not manage cluster roles. In the proposed framework, Q-learning adjusts CH roles and routing to energy and congestion, and the CN supplies trust values that set CH eligibility and inform next-hop choice. The higher trust values and the larger number of surviving nodes in later rounds agree with this design. Higher total energy use is the main cost observed in the run.

These findings have several limitations. All results come from a single run with one random deployment of 100 static nodes, so no variance or confidence interval can be given. The steady rise of the baseline trust values in particular should be checked over repeated runs with different seeds. No attack scenario with a defined fraction of malicious nodes or a defined attack type was simulated. The trust and anomaly mechanisms were therefore not tested against adversaries, and security and resilience remain design aims rather than demonstrated results. Because no node failures were injected, heartbeat-based fault handling and redundancy were not exercised, and scalability was not tested beyond 100 nodes. The rewards in Table 2 were set by hand. No ablation without Q-learning or without the CN was run, and no non-learning baseline such as LEACH [21] was included, so the separate effect of learning is not isolated. The values of α, γ, ϵ, the weights in Equations (2), (4), and (5), the anomaly threshold, the number of clusters, and the number of CNs used in the run were not recorded. Sensitivity to these parameters and to the state discretization was not examined, and neither was the convergence of the Q-tables.

Further limits concern the data and the simulator. The baselines were implemented in the same custom simulator, so their results reflect that implementation. Energy and trust were logged only up to round 700, and the alive-node count is missing for round 600. The energy series ends at round 700 while the alive-node series continues to round 1000, so the two cover different ranges, and their consistency could not be checked from the available data. Synthetic data were used, and the simulator did not model mobility, hardware failures, noise, signal fading, environmental interference, or edge and fog integration. On real hardware, continuous monitoring and Q-table updates may strain nodes with limited energy, processing power, and memory. Periodic CH re-election may create bottlenecks in very large networks, and malicious nodes could try to manipulate the reward system. Possible mitigations include sleep-wake scheduling, edge offloading or a smaller state-action space, multi-tier clustering, and secure node verification with adaptive trust-based rewards. No medical data, patient scenario, or clinical validation was involved, and the framework is not a medical device.

CONCLUSIONS

This study presented a Q-learning-based framework for WSNs in IoT settings. Each sensor node acts as an RL agent for CH selection and routing, and a dedicated CN maintains trust through direct and indirect observation, anomaly scoring, gray listing, and blacklisting. Trust sets CH eligibility and is one input to next-hop selection. In a single-run Python simulation with 100 static nodes over 1000 rounds, the framework used more energy than TEERP and RLER by round 700 (800 J against 750 J and 700 J) but kept more nodes alive, with 75 nodes remaining at round 1000 against 10 and 47. It did not reach the half-nodes-dead point within the run, while TEERP reached it at round 800 and RLER between rounds 900 and 1000. Its mean trust over the logged rounds was 0.90, against 0.77 and 0.79, and its recorded response time of 0.05 s equaled that of RLER. These results suggest that lightweight Q-learning combined with continuous trust monitoring can extend the working life of a clustered WSN at the price of higher total energy use.

The objectives were met only in part. An RL agent for CH selection and routing was implemented and compared with both baselines, which answers the first and fourth objectives within the limits of one run. Trust assessment ran throughout the simulation, but anomaly scoring was not tested against attacks, so the second objective was met only for the benign case. The third objective, extension to larger networks, was not tested. Since the evidence comes from one run of a static simulated network

without attacks, the findings are preliminary. Future work should repeat the experiments over multiple seeds and network sizes, simulate attacks with known malicious-node fractions to measure detection and false-positive rates, add ablation and LEACH baselines, and log energy per component. The reward function and learning rates also need refinement. A healthcare-oriented evaluation would add urgent traffic classes, node mobility, real physiological signals, and attack injection. Deep and multi-agent Q-learning could be explored for larger deployments, and the framework should be validated on physical testbeds.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," IEEE Commun. Surveys Tuts., vol. 17, no. 4, pp. 2347-2376, 2015. https://doi.org/10.1109/COMST.2015.2444095

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," IEEE Internet Things J., vol. 4, no. 5, pp. 1125-1142, Oct. 2017. https://doi.org/10.1109/JIOT.2017.2683200

[3] D. Kandris and E. Anastasiadis, "Advanced wireless sensor networks: Applications, challenges and research trends," Electronics, vol. 13, no. 12, Art. no. 2268, Jun. 2024. https://doi.org/10.3390/electronics13122268

[4] A. Jahangeer, S. U. Bazai, S. Aslam, S. Marjan, M. Anas, and S. H. Hashemi, "A review on the security of IoT networks: From network layer's perspective," IEEE Access, vol. 11, pp. 71073-71087, 2023. https://doi.org/10.1109/ACCESS.2023.3246180

[5] M. Faris, M. N. Mahmud, M. F. M. Salleh, and A. Alnoor, "Wireless sensor network security: A recent review based on state-of-the-art works," Int. J. Eng. Bus. Manag., vol. 15, 2023. https://doi.org/10.1177/18479790231157220

[6] M. A. Jamshed, K. Ali, Q. H. Abbasi, M. A. Imran, and M. Ur-Rehman, "Challenges, applications, and future of wireless sensors in Internet of Things: A review," IEEE Sensors J., vol. 22, no. 6, pp. 5482-5494, Mar. 2022. https://doi.org/10.1109/JSEN.2022.3148128

[7] N. Najem, D. Benhaddou, M. R. Abid, H. Darhmaoui, N. Krami, and O. Zytoune, "Context-aware wireless sensors for IoT-centeric energy-efficient campuses," in Proc. IEEE Int. Conf. Smart Computing (SMARTCOMP), Hong Kong, China, 2017, pp. 1-6. https://doi.org/10.1109/SMARTCOMP.2017.7946995

[8] Z. N. Aghdam, A. M. Rahmani, and M. Hosseinzadeh, "The role of the Internet of Things in healthcare: Future trends and challenges," Comput. Methods Programs Biomed., vol. 199, Feb. 2021. https://doi.org/10.1016/j.cmpb.2020.105903

[9] M. N. Bhuiyan et al., "Design and implementation of a feasible model for the IoT based ubiquitous healthcare monitoring system for rural and urban areas," IEEE Access, vol. 10, pp. 91984-91997, 2022. https://doi.org/10.1109/ACCESS.2022.3202551

[10] N. Geddes, G. Sen Gupta, and F. Hasan, "An energy efficient protocol for wireless body area network of health sensors," in Proc. 2019 IEEE Int. Instrumentation and Measurement Technology Conf. (I2MTC), May 2019, pp. 1-6. https://doi.org/10.1109/I2MTC.2019.8826862

[11] O. Bin Samin, N. A. A. Algeelani, A. Bathich, G. M. Adil, A. Qadus, and A. Amin, "Malicious agricultural IoT traffic detection and classification: A comparative study of ML classifiers," J. Adv. Inf. Technol., vol. 14, no. 4, pp. 811-820, 2023. https://doi.org/10.12720/jait.14.4.811-820

[12] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," IEEE Access, vol. 8, pp. 209320-209344, 2020. https://doi.org/10.1109/ACCESS.2020.3038605

[13] C. J. C. H. Watkins and P. Dayan, "Q-learning," Mach. Learn., vol. 8, no. 3-4, pp. 279-292, May 1992. https://doi.org/10.1007/BF00992698

[14] A. Keerthika and V. Berlin Hency, "Reinforcement-learning based energy efficient optimized routing protocol for WSN," Peer-to-Peer Netw. Appl., vol. 15, no. 3, pp. 1685-1704, May 2022. https://doi.org/10.1007/s12083-022-01315-6

[15] M. Ilyas et al., "Trust-based energy-efficient routing protocol for Internet of things-based sensor networks," Int. J. Distrib. Sens. Netw., vol. 16, no. 10, Oct. 2020. https://doi.org/10.1177/1550147720964358

[16] P. P. Ray, "A survey on Internet of Things architectures," J. King Saud Univ. Comput. Inf. Sci., vol. 30, no. 3, pp. 291-319, Jul. 2018. https://doi.org/10.1016/j.jksuci.2016.10.003

[17] R. Sharma, S. Prakash, and P. Roy, "Methodology, applications, and challenges of WSN-IoT," in Proc. Int. Conf. Electrical and Electronics Engineering (ICE3), Gorakhpur, India, Feb. 2020, pp. 502-507. https://doi.org/10.1109/ICE348803.2020.9122891

[18] K. Römer and F. Mattern, "The design space of wireless sensor networks," IEEE Wireless Commun., vol. 11, no. 6, pp. 54-61, Dec. 2004. https://doi.org/10.1109/MWC.2004.1368897

[19] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," Comput. Netw., vol. 38, no. 4, pp. 393-422, Mar. 2002. https://doi.org/10.1016/S1389-1286(01)00302-4

[20] M. S. Bensaleh, R. Saida, Y. H. Kacem, and M. Abid, "Wireless sensor network design methodologies: A survey," J. Sensors, vol. 2020, Art. no. 9592836, 2020. https://doi.org/10.1155/2020/9592836

[21] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in Proc. 33rd Annu. Hawaii Int. Conf. System Sciences, Maui, HI, USA, Jan. 2000. https://doi.org/10.1109/HICSS.2000.926982

[22] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," Comput. Commun., vol. 30, no. 14-15, pp. 2826-2841, Oct. 2007. https://doi.org/10.1016/j.comcom.2007.05.024

[23] J. Bian et al., "Machine learning in real-time Internet of Things (IoT) systems: A survey," IEEE Internet Things J., vol. 9, no. 11, pp. 8364-8386, Jun. 2022. https://doi.org/10.1109/JIOT.2022.3161050

[24] S. O. Olatinwo and T.-H. Joubert, "Deep learning for resource management in Internet of Things networks: A bibliometric analysis and comprehensive review," IEEE Access, vol. 10, pp. 94691-94717, 2022. https://doi.org/10.1109/ACCESS.2022.3195898

[25] P. Gite, K. Chouhan, K. Murali Krishna, C. Kumar Nayak, M. Soni, and A. Shrivastava, "ML based intrusion detection scheme for various types of attacks in a WSN using C4.5 and CART classifiers," Mater. Today Proc., vol. 80, pp. 3769-3776, 2023. https://doi.org/10.1016/j.matpr.2021.07.378

[26] A. Musaddiq, T. Olsson, and F. Ahlgren, "Reinforcement-learning-based routing and resource management for Internet of Things environments: Theoretical perspective and challenges," Sensors, vol. 23, no. 19, Art. no. 8263, Oct. 2023. https://doi.org/10.3390/s23198263

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

[28] X. Su, Y. Ren, Z. Cai, Y. Liang, and L. Guo, "A Q-learning-based routing approach for energy efficient information transmission in wireless sensor network," IEEE Trans. Netw. Serv. Manag., vol. 20, no. 2, pp. 1949-1961, Jun. 2023. https://doi.org/10.1109/TNSM.2022.3218017

[29] B.-S. Kim, B. Suh, I. J. Seo, H. B. Lee, J. S. Gong, and K.-I. Kim, "An enhanced tree routing based on reinforcement learning in wireless sensor networks," Sensors, vol. 23, no. 1, Art. no. 223, Jan. 2023. https://doi.org/10.3390/s23010223

[30] N. E. H. Bourebia and C. Li, "A greedy energy efficient clustering scheme based reinforcement learning for WSNs," Peer-to-Peer Netw. Appl., vol. 15, no. 6, pp. 2572-2588, 2022. https://doi.org/10.1007/s12083-022-01368-7

[31] T. Sharma, A. Balyan, R. Nair, P. Jain, S. Arora, and F. Ahmadi, "ReLeC: A reinforcement learning-based clustering-enhanced protocol for efficient energy optimization in wireless sensor networks," Wireless Commun. Mobile Comput., vol. 2022, Art. no. 3337831, 2022. https://doi.org/10.1155/2022/3337831

[32] M. S. Frikha, S. M. Gammar, A. Lahmadi, and L. Andrey, "Reinforcement and deep reinforcement learning for wireless Internet of Things: A survey," Comput. Commun., vol. 178, pp. 98-113, Oct. 2021. https://doi.org/10.1016/j.comcom.2021.07.014

[33] D. Keum and Y.-B. Ko, "Trust-based intelligent routing protocol with Q-learning for mission-critical wireless sensor networks," Sensors, vol. 22, no. 11, Art. no. 3975, May 2022. https://doi.org/10.3390/s22113975

[34] S. M. Muzammal, R. K. Murugesan, N. Z. Jhanjhi, M. Humayun, A. O. Ibrahim, and A. Abdelmaboud, "A trust-based model for secure routing against RPL attacks in Internet of Things," Sensors, vol. 22, no. 18, Art. no. 7052, Sep. 2022. https://doi.org/10.3390/s22187052.