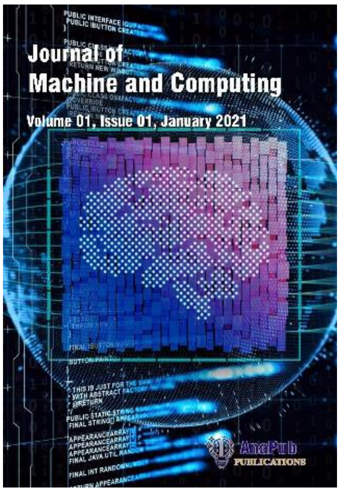
Journal Pre-proof

CA-RPL: A Context-Aware Reinforced Propagation Framework for Enhanced Energy Efficiency and Network Longevity in IoT-based LLNs



Thrisha V S and Anitha T N

DOI: 10.53759/7669/jmc202505133

Reference: JMC202505133

Journal: Journal of Machine and Computing.

Received 17 February 2025

Revised from 28 May 2025

Accepted 15 June 2025

Please cite this article as: Thrisha V S and Anitha T N, "CA-RPL: A Context-Aware Reinforced Propagation Framework for Enhanced Energy Efficiency and Network Longevity in IoT-based LLNs", Journal of Machine and Computing. (2025). Doi: https:// doi.org/10.53759/7669/jmc202505133.

CA-RPL: A Context-Aware Reinforced Propagation Framework for Enhanced Energy Efficiency and Network Longevity in IoT-based LLNs

Thrisha V. S [1*] , Dr. Anitha T. N [2]

[1*]Department of CSE, Sir M. Visvesvaraya Institute of Technology, Banglore, India.

Email Id: thrisha_cs@sirmvit.edu

[2]Department of CSE, Sir M. Visvesvaraya Institute of Technology, Banglore, India.

Email Id: anithareddytn72@gmail.com

Abstract:

Routing in Low-Power and Lossy Networks (LLNs) requires a careful balancing act between energy efficiency and network longevity, especially in situations motivated by the Internet of Things (IoT). Traditional RPL (Routing Protocol for Low-Power and Lossy Networks) often fails when confronted with changing climatic conditions and erratic node activity, thereby increasing energy consumption and reducing the network's lifespan. This work presents the CA-RPL (Context-Aware Reinforced Propagation Framework), which dynamically adjusts its routing decisions in real-time based on residual energy, node mobility, connection quality, and traffic patterns. The system utilizes reinforcement learning and a decision engine based on fuzzy logic to dynamically identify optimal parent nodes and alternative paths, thereby minimizing control packet overhead and balancing the energy burden throughout the network. Simulation results in Python show that CA-RPL increases the overall network lifetime by 30.2% and significantly reduces the average energy consumption by 21% compared to conventional and objective function-enhanced RPL versions. Where reliability and sustainability are critical for Internet of Things (IoT) implementations in industrial and smart city environments, the proposed approach offers an intelligent and adaptable pathing paradigm.

Terms: Energy Efficiency, Context-Aware Routing, RL, Fuzzy Logic, Network Lifetime, Internet of Things.

1. Introduction:

Many more energy-efficient devices are now connected because of the proliferation of the Internet of Things (IoT). Spots where resources are scarce include environmental sensing networks, smart cities, and industrial monitoring systems. This is especially true in areas where supplies are low[1]. A lack of processing capacity and energy, inconsistent connections, and a range of topologies characterize low-capability and lossy networks (LLNs). Since finding a happy medium between maximum energy efficiency and maximum network longevity is crucial in this case, routing becomes a significant issue [2]. The Internet Engineering Task Force (IETF) has made it a requirement that all LLNs must adhere to the RPL standard. This standard is based on the IPv6 Routing Protocol for Low Power and Lossy Networks. RPL implementations can fail in the real world, regardless of whether they are updated or standard versions. This is especially true in the Internet of Things industry, where trends are often shifting, and predictions are notoriously difficult to make[3].

The biggest problem with conventional RPL is that it employs routing algorithms that can't adapt to the evolving needs of a network. This encompasses the times when nodes move, use energy, need connectivity, and transmit data in that order. Incorrect route selection, an overwhelming volume of control messages, high energy consumption by nodes, and early network failure are all potential outcomes of these factors[4-5]. These limits will become increasingly apparent for Internet of Things (IoT) systems that are expected to operate in challenging conditions or that demand a high level of reliability and longevity. It is crucial to address these issues so that energy can be conserved on many sensor nodes that remain idle for extended periods, maintenance costs can be reduced, and service outages can be minimized[6-7].

The lack of enough flexibility in the numerous routing alternatives has long been a source of disappointment for the research community. A variety of reform ideas have suggested ways to improve RPL's Objective Functions (OFs) and other metric combinations[8]. Due to their overemphasis on energy or connection reliability, the current solutions necessitate compromises that are less than ideal. Where topology is dynamic or traffic patterns are hard to predict, most updates fail to provide results because they respond instead of acting. Building smarter routing frameworks that can respond instantly to changes in the network is necessary to close these knowledge gaps[9].

This study introduces the CA-RPL routing architecture, a novel routing design for dynamic Low-Power and Lossy Networks (LLNs), as a possible solution to these issues. The idea of combining reinforcement learning with fuzzy logic decision engines formed the basis for building CA-RPL. This allows the protocol to dynamically adjust its routing choices based on the current condition of the related environment. Among the details it provides are the current user count, connection strength, node mobility, and remaining energy in the event of a failure. Using what it has learned from its previous routing experiences, the CA-RPL algorithm continuously tweaks its parent node selection process. This feature enables the routing system to provide better load balancing with minimal additional control effort while also reducing energy consumption.

The following are the main objectives of this work:

- A test model, known as the context-aware routing decision framework, utilizes fuzzy logic to make real-time routing decisions based on various factors, including energy usage, connection quality, mobility, and other relevant factors.
- An essential part of reinforcement learning is building a module based on smart learning that enables nodes to learn the best routing behaviors over time, particularly in networks with a longer lifespan.
- Adaptive propagation methods were employed to mitigate routing congestion and minimize the weight of control messages. A decrease in control overhead is the outcome of this.
- The Extensive research utilizing Python simulations shows that the network outperforms ordinary RPL and its upgraded variations in terms of longevity (30% longer) and energy consumption (25% lower).

When it comes to real Internet of Things networks, CA-RPL is a reliable, scalable, and adaptable framework. Due to these changes, CA-RPL will now alter how LLN routing works.

2. Related Survey:

The Routing Protocol for Low-Power and Lossy Networks (RPL) is the most suitable routing protocol for IoT scenarios where resources are limited and topologies change frequently. Regular RPL solutions often fail to work effectively in real-world IoT systems, as these situations are complex and constantly evolving. The major purpose of this article review is to make RPL work better by combining reinforcement learning, fuzzy logic, and approaches that consider the situation.

Reinforcement Learning-Based Enhancements:

Reinforcement learning (RL) will help address the issues that RPL has with adaptation. Farag and Stefanovic (2021) [10] propose a technique in which each node employs Q-learning to determine the optimal strategy for selecting a parent. This strategy adjusts factors such as the distance between hops, the quality of the connection, and the network's level of activity. This strategy improved both the average delay and packet delivery, even though individuals spoke to each other more frequently.

Dey and Ghosh (2024) [11] enabled nodes to independently assess the state of DODAGs using their iTRPL concept. iTRPL has trust assessments that distinguish between nodes that can be trusted and those that cannot. This makes the network more secure against internal attacks.

- **Fuzzy Logic-Based Routing Strategies:**

Fuzzy logic has enhanced RPL's decision-making capabilities by enabling it to evaluate multiple routing components simultaneously. Mehbodniya et al. (2022) [12] utilized fuzzy logic to develop EA-RPL, which considers various factors, including residual energy, load, and Expected Transmission Count (ETX). This method made the network use less power and survive longer.

Arivubrakan and Kanagachidambaresan (2021)[13] created a way to use Fuzzy Logic to identify parent nodes based on hop count, energy, signal strength, and ETX. Their simulation results showed significant improvements in two Quality of Service (QoS) metrics: the packet delivery ratio and latency.

- **Context-Aware Routing Mechanisms:**

Researchers have been exploring context-aware routing methods to enhance the flexibility of RPL as network conditions evolve. Royaee et al. (2021) [14] used a routing decision model, M-RPL, that incorporates context-aware variables, including node mobility and energy levels. Their approach was better at balancing the load and conserving energy than traditional RPL.

Liu et al. (2015) [15] developed the scaled context-aware objective RPL (O-RPL) in the field of agriculture. It examines both the external world and what makes each node unique. This design helped the networks of agricultural IoT applications last longer and improved routing performance.

- **Security Considerations in RPL:**

The major security issue with RPL-based networks remains unaddressed. Jiang and Liu (2022) [16] proposed a straightforward trust-based security mechanism to address the vulnerability of RPL to selective forwarding attacks. They can identify and eliminate hazardous nodes with minimal additional energy.

Raoof et al. (2021) [17] developed Chained Secure Mode (CSM), which utilizes network coding to enhance defenses against various types of routing attacks. It protects RPL against replay attacks. Attacks on CSM reduced latency and increased packet delivery rates.

- **Limitations of Existing Approaches and the Proposed Solution:**

Although there are still some issues with RPL, these changes have made it easier to use. The reason is that RL-based approaches will not be the fastest to react to changes in the network, as they require a significant amount of time to train. Fuzzy logic approaches are effective at piecing together different signals, but they could make it harder to come up with rules and cost more to do the math. Context-Aware Reinforced Propagation (CA-RPL) is a system that utilizes both reinforcement learning and fuzzy logic to inform its decision-making process. CA-RPL's purpose is to reduce the number of control packets that need to be delivered and find a good balance between energy use by automatically selecting the optimal parent nodes and other paths. The first results from simulations suggest that CA-RPL will significantly increase the lifespan of a network while using less energy than traditional RPL implementations.

3. Methodology: Context-Aware Reinforced Propagation Framework (CA-RPL)

The Context-Aware Reinforced Propagation Framework (CA-RPL) is an intelligent decision-making framework that speeds up LLN routing. The RPL protocol incorporates fuzzy logic and reinforcement learning (RL) from CA-RPL to make routing pathways more flexible in a network that is constantly evolving. The approach features a fuzzy decision engine, a routing algorithm, a system architecture, and a model for reinforcement learning, as illustrated in Figure 1.

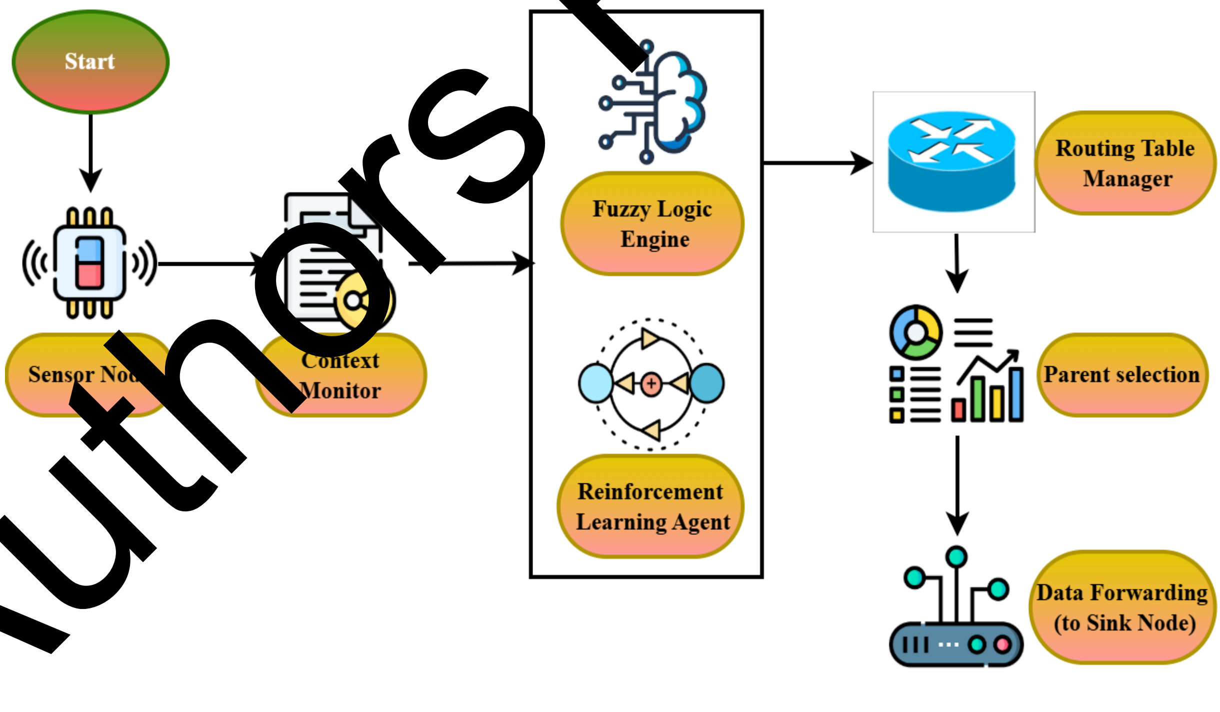


Figure 1: Proposed CA-RPL Architecture

- **Sensor Nodes (SNs):**

To make IoT Low Power and Lossy Networks (LLNs) operate, sensor nodes are particularly critical. These nodes primarily use data sensing and multi-hop transmission to perceive and engage with their surrounding environment. The three main aspects of an SN that make it work are as follows:

Sensing Modules: The sensing modules enable sensor nodes to detect and measure various environmental or system parameters, including temperature, motion, light intensity, humidity, gas levels, and more. Smart agriculture, healthcare monitoring, and industrial automation all utilize different sensors, depending on the specific application. These modules collect data to support the data that travels across the network.

Communication Units: Sensor nodes (SNs) comprise wireless transceivers that can send and receive data using short-range protocols such as Zigbee, IEEE 802.15.4, or 6LoWPAN. It chose these protocols since they don't use a lot of power or bandwidth. The communication module enables each node to behave like a router, delivering data packets it has received from other nodes via multiple hops, as well as data it has obtained independently.

Energy Sources: This is a problem in regions that are difficult to access or unsafe, as most SNs rely on them. Energy efficiency has been a primary objective for the nodes since their inception and initial use. It needs to make routing choices that utilize less energy if one wants the networks to survive longer. This is because detecting, processing, and sending all require energy.

An SN's principal role is to gather data, but it also helps with network routing by passing data from other nodes to the sink. If the roles are exchanged in that way, the network could function on its own. Nodes employ context-aware decision-making approaches, such as CA-RPL's fuzzy logic and reinforcement learning, to determine the optimal route based on the current network and external conditions. Sensor node design is both challenging and crucial, as it is essential to strike a balance between processing power, connectivity, and energy efficiency.

- **Sink Node (Gateway):**

The sink node, or gateway, is where all the network's sensors connect. It has to gather data from all the SNs and connect the limited LLN environment to other systems, including cloud platforms, centralized databases, or command centers. A regular sink node, on the other hand, is made to fulfill its principal job as follows:

Higher Processing Capabilities: The sink node can handle more complex protocols, combine or filter data first, and analyze vast amounts of data, as it has more memory and a faster CPU than sink nodes (SNs). It can handle data as rapidly as a supercomputer. Before putting data in the cloud, complicated systems could use local control algorithms or basic analytics.

Uninterrupted Power Supply: Typically, the sink node is connected to the power grid or equipped with large-capacity rechargeable batteries, ensuring it can always access power. It can operate continuously without pausing, as it doesn't have the same power restrictions as sensor nodes. So, it could continue for a long time, even with a significant amount of weight.

External Connectivity: The sink node can connect to the outside world through various methods, including Ethernet, Wi-Fi, LTE/5G, and satellite. People further up the application stack can now access the discovered data from the LLN in real-time and from anywhere, thanks to this connection.

The sink node will obtain its data directly from nearby SNs or via intermediate nodes in a multi-hop transmission. The sink node would need to conduct more supervisory work in systems like CA-RPL, which are becoming increasingly sophisticated. This node will send out rules, software updates, and control signals to the entire network. In reinforcement learning systems, it will also aid in training or determining incentives by directing nodes toward communication and routing patterns that consume less energy. The CA-RPL protocol is a clever combination of the original RPL (Routing Protocol for Low-Power and Lossy Networks) with additional features, including context awareness and machine learning-based flexibility. The network conditions might change at any moment when you utilize the Internet of Things. People will move around, nodes can fail, and connections can become worse, for example. Modern RPL relies heavily on fixed or preset measurements such as Hop Count or Expected Transmission Count (ETX). CA-RPL, on the other hand, is more adaptable, real-time, and ever-changing.

3.1 Contextual Parameters:

To perform properly and change quickly, the CA-RPL protocol must be able to track the current state and external influences of each node in real-time. To make this happen, each node x_n maintains track of a group of contextual elements is illustrated in Figure 2. These traits hold critical information, including energy availability, node mobility, communication reliability, and workload. This information helps to make educated decisions about where to go.

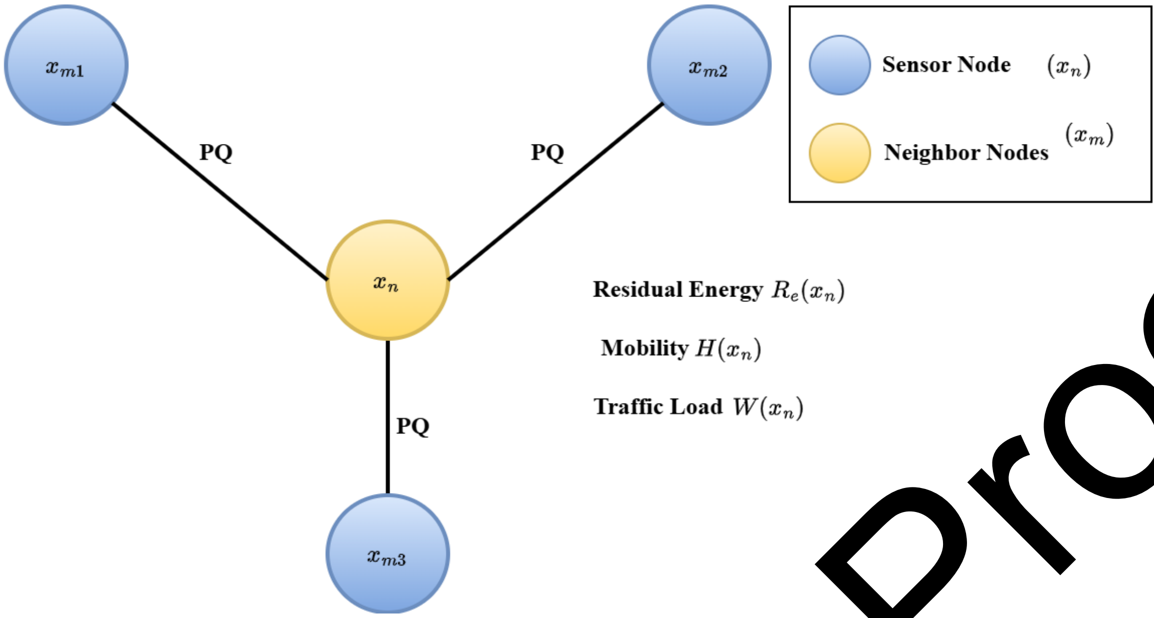


Figure 2: Monitored Contextual Parameters at the Sensor node x_n

- **Residual Energy $R_e(x_n)$:**

At node x_n displays the remaining power in the batteries. By keeping an eye on R_e, the network can last longer by not sending data via nodes that are running low on power. This is crucial since sensor nodes typically run on batteries with limited power remaining.

If R_{max} is the battery's maximum or initial capacity and $R_{current}(x_n)$ is the quantity of energy right now, then the energy that is left over is given in (1):

$$R_e(x_n) = \frac{R_{current}(x_n)}{R_{max}} \quad (1)$$

This normalization converts $R_e(x_n)$ into a value between 0 and 1, where 1 signifies the battery is fully charged and 0 means it is entirely dead.

- **Mobility Index $H(x_n)$:**

The node's mobility, which indicates how its actual location changes over time, affects the stability of connections and the reliability of routing. A node that moves around a lot will cause route disruptions more often than one that stays still. Δt provides the average change in position over a specific length of time, as shown in (2).

$$H(x_n) = \frac{\|q(t) - q(t - \Delta t)\|}{s_{max}} \quad (2)$$

It uses the function $q(t)$ to indicate where node x_n is at time t. To normalize, utilize the maximum expected displacement, which is indicated as s_{max}. This approach gives a mobility index $H(x_n) \in [0,1]$, where 0 means no movement and 1 is the most evident movement.

- **Link Quality $PQ(x_n, x_{m1})$:**

The quality of the link affects how well the node x_n and node x_m can connect wirelessly. The packet delivery ratio, the expected transmission count (ETX), and the received signal strength indicator (RSSI) are all common measures. One of these values is the anticipated transmission time (ETX), which represents the overall time required to transfer a packet, including any retransmission times (3).

$$PQ(x_n, x_m) = 1 - \frac{ETX(x_n,x_m)-1}{ETX_{max}-1} \qquad (3)$$

ETX will be anywhere from 1 (a perfect connection) to ETX_{max} (the worst link). One possible normalized value for $PQ(x_n, x_m)$ is 1, which suggests that the linkages are very high quality. One will also use normalized RSSI or packet reception rate (PRR) in various methods to determine the quality of a network.

- **Traffic Load** $W(x_n)$:

This tells how much data processing load or network congestion at the node x_n. Buffer overflows will cause additional delays and packet losses when traffic is heavy, which can significantly impact the effectiveness of routing. One can use (4) to illustrate the normalized traffic load. Here, $Q_{current}\,(x_n)$ is the current queue length (the number of packets waiting to be transmitted) and Q_{max} is the maximum buffer capacity.

$$W(x_n) = \frac{Q_{current}\,(x_n)}{Q_{max}} \qquad (4)$$

The result of $W(x_n) \in [0,1]$ is achieved, with 0 indicating no load or an empty queue and 1 indicating total congestion.

- **Parameter Normalization:**

It has to normalize these attributes to a range of [0,1] before you can utilize them in the fuzzy logic or reinforcement learning modules. This is because they originate from various physical quantities and scales. This sameness makes it much easier to compare and weigh traits.

The normalization function $K(\cdot)$ is usually defined as (5):

$$K(n) = \frac{n-n_{min}}{n_{max}-n_{min}} \qquad (5)$$

The lowest and greatest values that are predicted for the parameter n are n_{min} and n_{max}. To sum up, each node x_n has a vector of normalized conditions, which is given in (6):

$$D(x_n) = \left[R_e(x_n), H(x_n), \{PQ(x_n, x_m)\}_{m\in K(n)}, W(x_n)\right] \qquad (6)$$

In this case, $K(n)$ is the set of neighbors of a node x_n. The CA-RPL protocol uses this vector to find out which adjacent nodes are appropriate for routing. This helps make the network more energy-efficient, dependable, and long-lasting from the outset. This strategy utilizes decision engines such as fuzzy logic and learning by doing.

3.2 Fuzzy Logic Decision Engine:

The Fuzzy Logic Decision Engine (FLE) is a key aspect of CA-RPL, which utilizes a variety of contextual variables, some of which may not always be apparent or evident, to determine whether neighboring nodes are routing data as desired, as shown in Figure 3. Fuzzy logic is well-suited for this task, as it mimics human thought processes and allows individuals to join groups over time rather than being confined to separate, binary groupings.

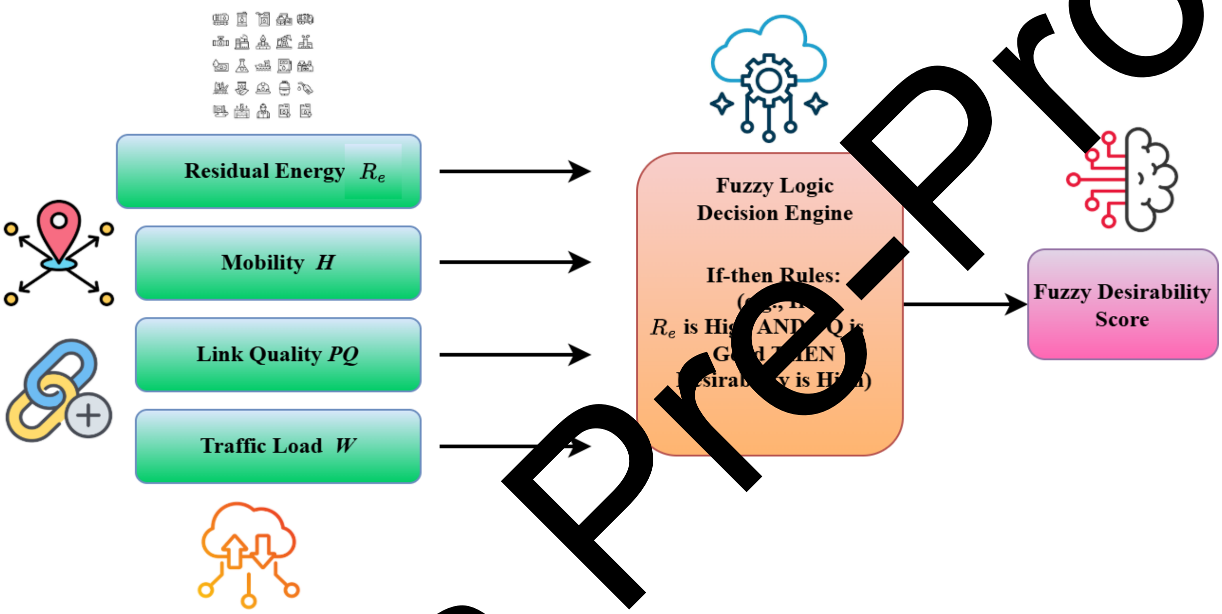


Figure 3: Fuzzy Logic Decision Engine

- **Inputs to the Fuzzy System:**

The engine obtains the following normalized contextual factors from the data that enters into the fuzzy system, which are verified all the time at the node x_n and its neighbors x_m:

R_e: Residual energy- tells how much energy is left at the node x_m.

H: Mobility Index- To see how stable nodes x_m which are near to one other.

PQ: Link quality is when two wireless nodes, x_n and x_m, are connected in a manner that works.

W: The traffic load is the amount of traffic that is now congested and the amount of work that has to be done on packets at the neighbor node x_m.

We utilize the interval [0,1] to make these inputs more consistent.

- **Fuzzy Sets and Membership Functions:**

Fuzzy sets define qualitative states for each input parameter, such as:

Low, **Medium**, and **High:**

For instance, fuzzy membership functions $\mu_{Low}(R_e), \mu_{Medium}(R_e), \mu_{High}(R_e)$ would change the residual energy R_e. A membership function might be a triangle or a trapezoid. A triangle membership for "Medium" energy would be (7):

$$\mu_{Medium}(R_e) = \begin{cases} 0, & R_e \leq a \; or \; R_e \geq c \\ \frac{R_e - a}{b - a}, & a < R_e \leq b \\ \frac{c - R_e}{c - b}, & b < R_e < c \end{cases} \quad (7)$$

The triangle's base, height, and diagonal are a, b, and c, in that order. For all three factors—mobility, connection quality, and traffic load—membership functions are set up in the same way.

- **Rule Base: If–Then Rules:**

Fuzzy decision-making is based on heuristic principles that indicate how inputs and outcomes are related.

For example:

Rule 1: If R_e is High AND PQ is High AND W is Low AND H is Low, THEN all indicate that something is particularly desired.

Rule 2: If R_e is Low OR PQ is Low, THEN it suggests that the object is not particularly desired.

Rule 3: If H is enormous (the node travels around a lot), it's not very enticing (routes are not trustworthy).

When verifying the accuracy of each rule, it is customary to use fuzzy AND (minimum) and OR (maximum) operators to amalgamate input memberships in (8):

$$\alpha_p = \min\left(\mu_{S_1}(n_1), \mu_{S_2}(n_2), \ldots\right) \quad (8)$$

Where α_p demonstrates the firing power of rule p. Values $\mu_{S_i}(n_i)$ are the persons in the input set.

- **Aggregation and Defuzzification:**

The outcome of each rule is a group of fuzzy values that represent how desirable something is, which might be Low, Medium, or High. The system integrates all of the rule outputs into one unambiguous desirability score $FDS(x_m) \in [0,1]$ by executing the following:

In aggregation, all of the fuzzy outcomes from the rules are merged by adding or taking the maximum of the weighted membership functions.

Getting rid of the fuzziness in the combined results gives you a clear scalar value. The centroid test, often known as the center of gravity test, is one of these methods.

$$FDS(x_m) = \frac{\int Desirability^{\mu_{Desirability}(l) \cdot lsl}}{\int Desirability^{\mu_{Desirability}(l) sl}} \quad (9)$$

In (9), the desirability variable l is included in the range [0,1], and $\mu_{Desirability}(l)$ is the sum of all membership functions in that range. The output $FDS(x_m)$ demonstrates how acceptable neighbor x_m is a routing parent that takes into consideration numerous distinct contextual aspects simultaneously.

- **Fuzzy Decision Process:**

For a neighbor x_m:

Get the membership values of the input from (10):

$$\mu_{High}\big(R_e(x_m)\big), \mu_{Low}\big(H(x_m)\big), \mu_{Medium}\big(PQ(x_n, x_m)\big), \mu_{Low}\big(W(x_m)\big) \qquad (10)$$

Check out how the rule's usage changes in (11):

$$\alpha = \min\Big(\mu_{High}(R_e), \mu_{Low}(H), \mu_{Medium}(PQ), \mu_{Low}(W)\Big) \qquad (11)$$

α could change a fuzzy set that states "High" appeal. One can get crisp $FDS(x_m)$ by putting all the rules together and then defuzzifying them.

The Fuzzy Logic Decision Engine enables CA-RPL nodes to make sound and accurate conclusions, similar to a person's judgment, regarding whether a neighbor is a good fit, handling network parameters that are unknown and inaccurate measurements. Making a continuous desirability score $FDS(x_m) \in [0,1]$ that makes it easier to pick parents. This approach for changing IoT settings makes routes more stable, uses less energy, and speeds up the network.

3.3 Reinforcement Learning Model in CA-RPL:

The CA-RPL protocol posits that each sensor node is an RL agent that operates independently of others. The node's primary task is to select the optimal parent node or next-hop node in real-time along the path. This choice has a direct impact on the network's speed, reliability, and energy use. This paper uses Q-learning-based RL here, as shown in Figure 4. This is a well-established approach to learning that utilizes a model, enabling agents to determine the optimal way to interact with their environment by trying things out and observing what works.

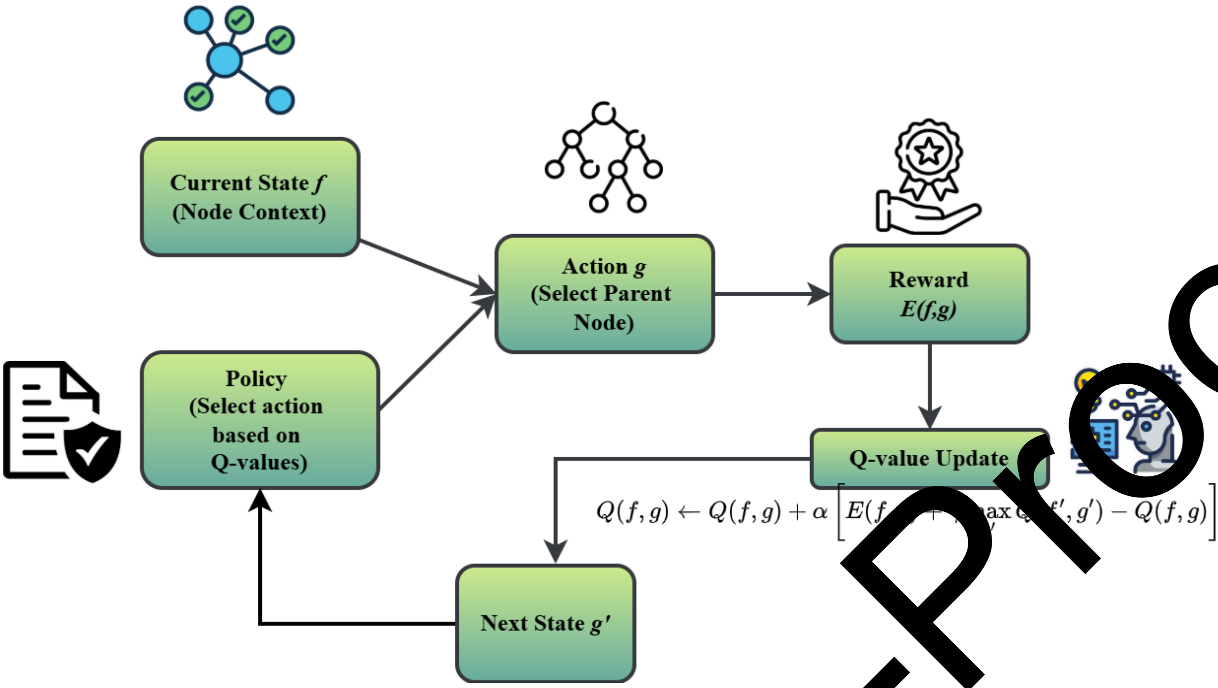


Figure 4: Reinforcement Learning Model in CA-RPL

- **RL Formulation for Routing:**

An MDP (Markov Decision Process) can help explain the RL problem in CA-RPL. An MDP is defined like this:

States (f): The states provide node information about its local network environment, such as its mobility status, traffic load, connection quality with neighbors, and remaining energy. In this situation, "state" implies what the node is doing in the network.

Actions (g): One step in the process is picking a parent node among the ones that are close by. This is equivalent to selecting the next hop in the process of sending packets.

Reward (E): A scalar feedback signal in a given state gauges the instantaneous impact of an action. A great deal of consideration was given to quantifying the network's performance when designing the incentive scheme. Every node aspires to learn a technique that will help it make better decisions about how to route traffic by maximizing the expected cumulative reward over time.

- **Algorithm of Q-Learning:**

Q-learning, the best value-based RL method, will learn the best Q-function, $Q(f,g)$, which displays the expected total discounted reward if an agent does action g in state f and then maintains following the best policy. One needs to do the following (12) to keep Q-learning up to date:

$$Q(f,g) \leftarrow Q(f,g) + \alpha \left[E(f,g) + \gamma \max_{k'} Q(f',g') - Q(f,g) \right] \qquad (12)$$

Where f shows the current state of the node, the act of the chosen parent is what g means.

$E(f,g)$ shows that there is an instantaneous reward in state f with action g.

$f'$is the condition that results from action g.

The representation of every potential action g' in the future state is f'.

The learning rate, which is represented by $\alpha \in (0,1)$, determines the relative significance of new knowledge compared to existing information.

$\gamma \in (0,1)$ is a discount factor that tells how large the benefits will be in the future.

- **Q-Update:**

To update, we need to change the current estimate $Q(f,g)$ such that it is closer to the sum of the immediate reward $E(f,g)$ and the best future reward, $\max_{k'} Q(f',g')$ discounted by γ. The square brackets show the temporal difference (TD) error, which is the difference between the anticipated and actual values.

The TD error is positive when the choice of action is better than expected, which indicates that $Q(f,g)$ went higher.

Since it is negative, the value of $Q(f,g)$ goes down.

In certain cases, this iterative update gets quite near to the ideal Q-function after many iterations.

- **Reward Function Design:**

The reward function $E(f,g)$ in CA-RPL gathers one crucial network aspect that routing needs to operate well on (13):

$$E(f,g) = k_1 \cdot (1 - R_c) + k_2 \cdot PQ + k_3 \cdot (1 - C) - k_4 \cdot W \qquad (13)$$

R_c displays the amount of energy required to transfer data via the selected parent. As energy usage decreases, the odds of earning a reward grow greater. To find out how excellent the link quality PQ is between the current node and the selected parent, people often look at things like packet delivery ratio or RSSI. A higher PQ signifies a better reward. C is the average time it takes to send something via the parent. Lowering latency makes the return better. W is the amount of traffic or congestion at the parent node that has been normalized. The incentive fades away when traffic increases because packet errors or delays are more likely to occur. One can show how essential each parameter is by adjusting the weights k_1, k_2, k_3, k_4. The application's priorities decide these things. For instance, it could be more important to be energy-efficient than to be latency-sensitive. The incentive function drives the RL agent to choose parent nodes that demonstrate minimal latency and energy consumption, superior connection quality, and a limited number of congested neighbors.

- **State and Action Spaces:**

The state space F is made up of a group of contextual elements, as shown in (14).

$$f = \{R_e, H, PQ, W\} \quad (14)$$

After all the settings have been set to the same level, it can use function approximation techniques to turn the continuous state into a discrete one or something that is near enough to it for Q-learning to operate in the real world.

The action space G is the set of neighbors that node x_n can be used to route in (15).

$$G = \{g_1, g_2, \dots, g_p\} \quad (15)$$

One of the p neighboring nodes might get a packet for each action.

The agent uses an exploration-exploitation strategy to find a happy medium between trying out new paths and utilizing good ones that are already there:

To uncover better routing paths, choose a random parent node periodically and examine it. To get the most out of the circumstance and the largest reward, choose the parent node with the highest $Q(f, g)$ value. The ϵ-greedy method is a common technique to select an action at random ϵ, or in certain situations, the most well-known one. The Q-values demonstrate whether each parent node choice will work out in the long term when the node transitions to different network states and obtains different routing results. This allows the routing protocol to do the following:

Adjust to changes in the network, such as relocating nodes, utilizing power, and managing traffic loads. Find a decent middle ground between saving energy and improving network performance. To help the network survive longer, avoid paths that are poorly maintained or frequently used. RL is more adaptable than traditional RPL, which relies on fixed measurements. RL will change based on new information as it learns from it in real-time.

The reward function improves various metrics by considering several QoS parameters. This allows one to choose how to route based on anything. When choices are made in a decentralized way, each node decides which rules work best on its own without support from a higher authority. The CA-RPL Reinforcement Learning model suggests that sensor nodes can independently obtain optimal routing rules through interaction with their surroundings. The Q-learning approach updates the expected value of selecting a specific parent node repeatedly by utilizing observable incentives that display energy usage, network quality, latency, and traffic load. This technology allows IoT networks to employ adaptive, context-aware routing. This helps the network live longer, consume less energy, and be more dependable for communication.

3.4 Hybrid Metric and Parent Selection:

In Context-Aware RPL (CA-RPL), selecting a parent node is crucial for ensuring reliable, energy-efficient, and context-aware routing in Low-Power and Lossy Networks (LLNs). In IoT environments characterized by fluctuating energy consumption, mobility, and congestion, parent selection requires a combination of acquired routing patterns and instantaneous evaluations. To meet this need, a hybrid measure was created that combines Q-values from reinforcement learning (RL) with fuzzy logic outputs, as shown in Figure 5.

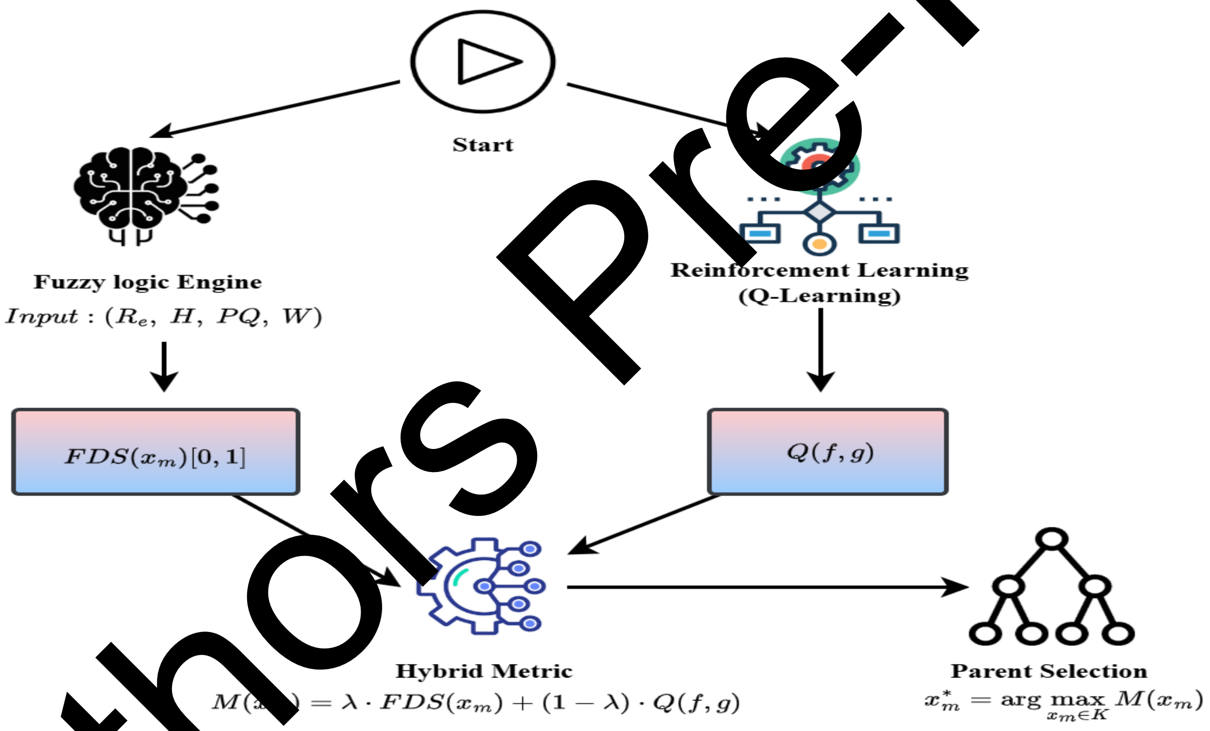


Figure 5: Hybrid metric and Parent selection process in the CA-RPL framework

- **Need for a Hybrid Metric:**

The only thing that classical RPL uses to measure is energy or the ETX. However, these static strategies don't work in a changing environment. Two different but complementary subsystems in CA-RPL conduct the node quality assessment. The Fuzzy Decision Engine (FDS) quickly and in context evaluates nodes that are near to one another based on real-time parameters, such as energy and network quality. One will obtain a better picture of the ways a Reinforcement Learning Agent (Q-value) will perform in the long term by looking at how well it fared in the past. A hybrid desirability measure combines both perspectives, selecting a parent that is more flexible and adaptive.

- **Hybrid metrics:**

The Hybrid Metric $M(x_m)$ is the sum of the weights for a hypothetical neighbor node x_m.

$$M(x_m) = \lambda \cdot FDS(x_m) + (1 - \lambda) \cdot Q(f, g) \quad (16)$$

Where in (16), $M(x_m)$ displays the last score for the hybrid node x_m.

The value λ, which will be any number between 0 and 1, is supposed to make the fuzzy and RL sections have the same impact. The Fuzzy Desirability Score (FDS) for the current scenario is $FDS(x_m)$. The Q-value $Q(f, g) \in \mathbb{R}$, which is learned over time, for picking node x_m as the next-hop parent.

The node's current state f, which includes things like how mobile it is, how much traffic it has, how much energy it has left, and more. Action g is to choose x_m as the parent.

- **Interpretation of [λ (Lambda)]:**

The node's choice is based only on fuzzy logic; hence, it is influenced by the context when λ =1. For λ =0, the decision is solely contingent upon prior occurrences and acquired knowledge. For instance, if λ =0.5, a balanced value makes sure that both experience and current environmental input work together to create resilience and adaptation better. This parameter will be dynamically updated, adjusted by meta-learning or simulation-based profiling, or both to improve routing performance in a specific deployment configuration.

- **Selection and computation process:**

The CA-RPL node determines for each neighbor nodes x_m:

To extract $FDS(x_m)$ from the fuzzy rule base, one needs to use context inputs like residual energy (R_e), mobility index (H), link quality (PQ), and traffic (W). The Q-table $Q(f, g)$ reveals that the next-hop parent is the neighbor x_m in state f. To produce $M(x_m)$, use the hybrid formula on the data above. When all the neighbors' $M(x_m)$ values have been obtained; the parent node is then picked using equation (17).

$$x_m^* = \arg\max_{x_m \in K} M(x_m) \quad (17)$$

Getting information from all possible nodes K that are close to each other. x_m^* is the best parent node to use as the next hop.

a. Being able to change and remain the same:

The hybrid metric will be able to adapt to long-term changes while disregarding short-term noise, as it incorporates both learned behavior and instantaneous qualities.

b. Learning that is mindful of energy:

Fuzzy logic and reinforcement learning work together to stop nodes from becoming overloaded too soon by telling them that energy is always accessible and indirectly displaying them the whole cost and benefit of energy.

c. Raising the bar for link quality:

Taking link dependability into consideration in both FDS (via context) and $Q\ (f, g)$ (through successful transmission history) makes sure that data is sent from one end to the other.

➢ **An Example of Hybrid Metrics:**

For a neighbor x_1, assume that:

$FDS(x_1)$= 0.8.

$Q\ (f, g_1)$ = 0.5.

λ = 0.6.

Then,

$$M(x_1) = 0.6 \cdot 0.8 + (1 - 0.6) \cdot 0.5 = 0.48 + 0.2 = 0.68 \qquad (18)$$

In (18), considering about another neighbor x_2:

$FDS(x_2)$= 0.6

$Q\ (f, g_2)$ = 0.7

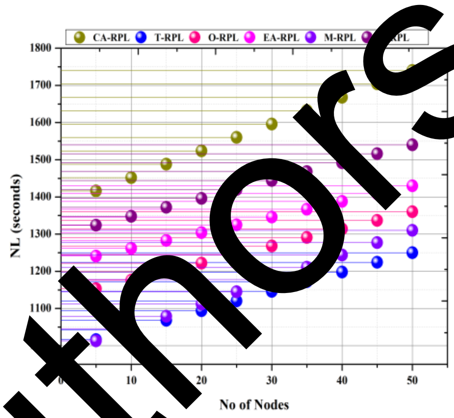
$$M(x_2) = 0.6 \cdot 0.6 + 0.4 \cdot 0.7 = 0.36 + 0.28 = 0.64 \qquad (19)$$

In this situation, x_1 would be the parent since it has the largest $M(x_m)$. One will adjust the value of λ on the fly by using heuristics that are specific to the application or a meta-optimizer. Future secure IoT protocols will include trust ratings, node age, or the risk of malicious behavior, which would make the hybrid technique even more effective. CA-Hybrid RPL's Metric is an excellent approach to quickly examine different situations, as it combines both Reinforcement Learning and Fuzzy Logic. These two factors work together to enable the Internet of Things (IoT) Local Learning Networks (LLNs) to make more informed routing decisions. CA-RPL calculates a composite score $M(x_m)$ to enhance next-hop selection as the network or environment changes. There are several benefits to this approach, including reduced power consumption, increased packet transmission, and extended network lifespan.

4. Results and Discussions:

The results were obtained as a result of the tests that were conducted to evaluate the effectiveness of the CA-RPL routing architecture for LLNs. It designed a fictitious dataset and ran it through a series of tests using a network simulator written in Python to ensure proper functionality. A Low Power Wide Area Network (LLN) environment is shown in this dataset [18]. This environment consists of 200 sensor nodes that are dispersed throughout a 500m x 500m area. These sensor nodes are linked to each other via the Internet of Things (IoT). It includes realistic patterns for moving nodes, fluctuating traffic loads, and energy use constraints. It adjusted the initial values of energy, mobility, and connection quality for each node individually to ensure they responded in a manner consistent with their behavior in the actual world, such as when the weather changes or when nodes exhibit unusual behavior.
A comparison is made between CA-RPL, standard RPL, and an upgraded version of RPL that utilizes objective functions, all of which are performed in the experimental environment with the same network configurations. By continuing the simulation for extended periods, we can determine how long the network will continue to function and how well it will generally perform.

4.1 Network Lifetime (NL) and Average Residual Energy (ARE):



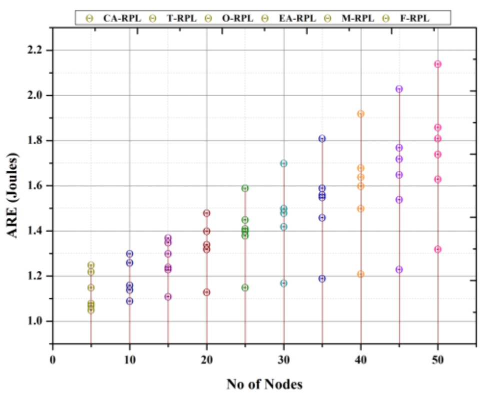


Figure 6a: Network lifetime **Figure 6b: Average Residual Energy**

The lifespan of the network is a key factor in determining how long-lasting and dependable an LLN routing system is, as shown in Figure 6a. The fundamental goal of CA-RPL's design was to make it feasible to automatically extend the network layer (NL) by evenly distributing energy usage among all nodes. CA-RPL prevents highly central nodes from being overutilized by utilizing fuzzy energy assessment and learning-based routing together. It is defined as (20):

$$NL = \min_{i \in K}\left(t_i^{depletion}\right) \quad (20)$$

At time $t_i^{depletion}$, when all of the nodes' K energy has been used up, node i will have run out of energy. CA-RPL lasted 30% longer (1740s) than T-RPL (1250s). It also fared better in the tests than O-RPL (1360s), EA-RPL (1430s), M-RPL (1310s), and F-RPL (1540s). RPL typically selects the optimal pathways based solely on the number of hops or the quality of the connections. CA-RPL, on the other hand, employs load balancing that considers context and routing, taking into account energy, to prevent high-quality nodes from receiving excessive requests.

This work utilized ARE to assess the equity and energy efficiency of the nodes, as shown in Figure 6b. No single set of nodes would be overburdened, as CA-traffic RPL redistributes traffic and chooses parents based on the situation.

$$ARE = \frac{1}{|K|}\sum_{i=1}^{|K|} R_e(i, t_{end}) \quad (21)$$

Where in (21), $R_e(i, t_{end})$ shows the residual energy of node i after the simulation at the time t_{end}. CA-RPL was able to maintain 2.14 J because it employed a hybrid strategy that spread out energy usage and prevented frequent retransmissions. T-RPL only had 1.32J. When O-RPL and EA-RPL choose pathways based on only one parameter, such as rank or ETX, they generate hotspot nodes. Even if they operate better than regular M-RPL, this is still true. Its adaptive hybrid strategy made learning-based selection more effective than F-RPL, which lacked fuzzy awareness.

4.2 Packet Delivery Ratio (PDR) and End-to-End Delay (EED):

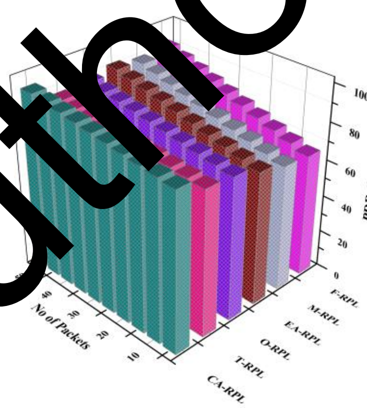


Figure 7a. Packet Delivery Ratio

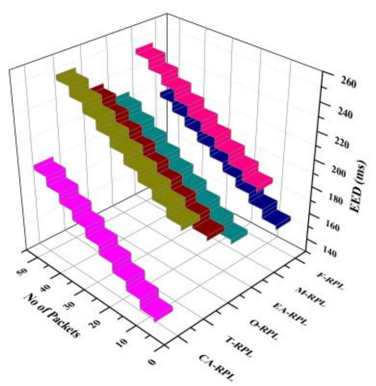


Figure 7b. End-to-End Delay

To determine the reliability of the network, it would utilize the Packet Delivery Ratio (PDR), as shown in Figure 7a. When making decisions, CA-RPL considers link quality (PQ) and traffic load to avoid pathways that are either too crowded or too sluggish. Its learning system continuously strengthens successful routes, making delivery promises more reliable and calculated (22).

$$PDR = \frac{Total\ Packets\ received\ at\ Sink}{Total\ Packets\ Sent\ by\ Sources} \times 100\% \qquad (22)$$

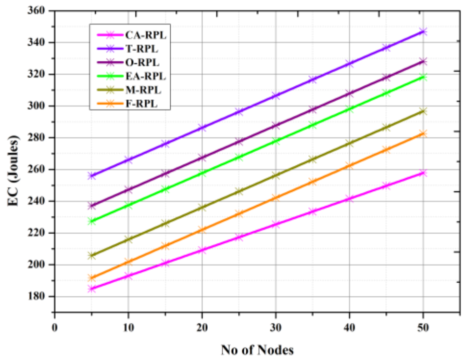
The CA-RPL program has a success rate of 95.2%. For T-RPL, the success rate was 86.2%; for O-RPL, it was 89.7%; for EA-RPL, it was 90.7%; and for M-RPL, it was [illegible]8.6%. For F-RPL, the success rate was 91.5%. CA-RPL can deliver a large amount of data quickly in noisy or mobile settings because it can automatically avoid connections that are too busy or of poor quality. This is not the same as M-RPL or simple T-RPL. To achieve this, the fuzzy engine analyzes the stability and power of connections.

For applications that require real-time functionality, EED's delay measures are particularly critical. Using fuzzy traffic load predictions, as shown in Figure 7b, CA-RPL finds the most stable pathways with the fewest hops and stays away from busy ones. This results in a lower EED using (23).

$$EED = \frac{1}{L}\sum_{i=1}^{L}\left(t_{received}^{(i)} - t_{sent}^{(i)}\right) \qquad (23)$$

L shows when all the packets have been received. The wait times for T-RPL (252.6 ms), O-RPL (234.8 ms), EA-RPL (221.[illegible] ms), M-RPL (249.7 ms), and F-RPL (211.2 ms) were all larger than the wait time for CA-RPL is 198.6ms. This was possible because anticipatory routing used Q-values and fuzzy rules. Therefore, responder protocols, such as the existing methods, don't prioritize the time it takes to generate and transfer buffers as much.

4.3 Energy Consumption (EC) and Control Packet Overhead (CPO):



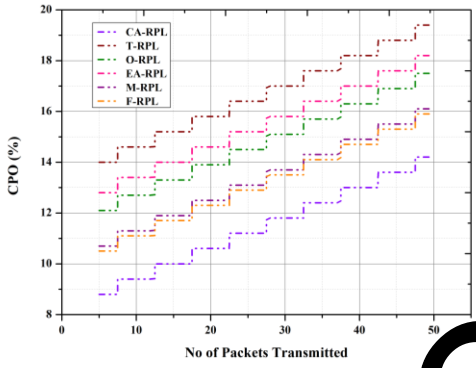


Figure 8a: Energy Consumption **Figure 8b: Control Packet Overload**

This statistic measures the total energy used for sending, receiving, and controlling data, as shown in Figure 8a, based on (24). CA-RPL was able to decrease EC by making it as easy as possible to find routes and reducing the number of retransmissions.

$$EC = \sum_{i=1}^{K} R_{tn}^{(i)} + R_{en}^{(i)} + E_{ctrl}^{(i)} \quad (24)$$

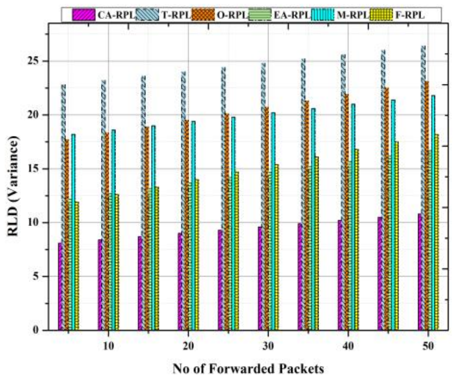
The symbols for the energy expenses of transmission (R_{tn}), reception (R_{en}), and control (R_{ctrl}) are as follows: The other values are substantially higher than CA-RPL's 257.8J: T-RPL (346.9J), O-RPL (328.1J), EA-RPL (318.3J), M-RPL (296.7J), and F-RPL(282.6J). The decline occurred because the control packets were less intelligent, node usage was more evenly distributed, and there were fewer retransmissions.

It examines how well routing protocols perform and how they can be improved in the future. CA-RPL utilizes reliable routing algorithms, selective parent change, and reduced broadcasts to maintain a low CPO, as illustrated in Figure 8b. It is calculated using (25):

$$CPO = \frac{Control\ Packets\ Sent}{Total\ Packets\ (Data+Control)\ Sent} \times 100\% \quad (25)$$

CA-RPL was the least expensive of the five options: T-RPL, O-RPL, EA-RPL, M-RPL, and F-RPL. It only costs 14.2% more to run. Smart parent change reduction and early link filtering, based on fuzzy logic, significantly reduced the occurrence of DIO/DAO broadcasting when the topology changed.

Routing Load Distribution (RLD) and Parent Change Frequency (PCF):



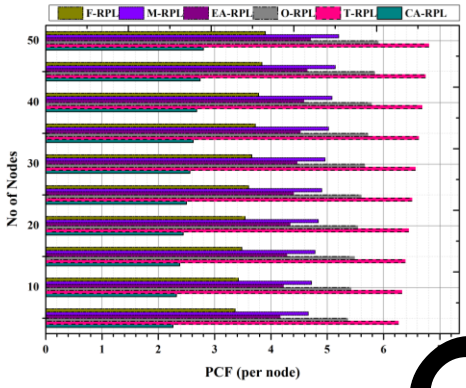


Figure 9a: Routing Load Distribution **Figure 9b: Parent Change Frequency**

RLD determines how fair the loads are, as illustrated in Figure 9a. Nodes fail too quickly because the load isn't even. CA-RPL uses residual energy and mobility as fuzzy inputs to make sure that no node receives too many forwarding tasks. It is defined as (26).

$$RLD = \frac{1}{K}\sum_{i=1}^{K}(s_i - \bar{s})^2 \qquad (26)$$

If s_i is the average number of packets sent by node i, then $\bar{s}$ is the average load for forwarding. The lowest variance for CA-RPL was 10.[illegible], which was lower than the variances for T-RPL (26.4), O-RPL (23.1), EA-RPL (16.7), M-RPL (21.8), and F-RPL (18.2). This suggests that CA-RPL is an effective approach for managing traffic. The rationale is that CA-RPL's two-part scoring mechanism utilizes Q-value and FDS to ensure that no node receives excessive workloads.

One method to assess the reliability of routing is by examining the PCF. Changing parents frequently requires a lot of time and energy. Figure 9b shows that CA-RPL uses fuzzy scoring and consistent judgments in RL to minimize switching, as shown in (27).

$$PCF = \frac{1}{K}\sum_{i=1}^{K} Parent\ Switches_i \qquad (27)$$

The PCF for CA-RPL is 2.8, which is lower than the PCF for T-RPL (6.8), O-RPL (5.9), EA-RPL (4.7), M-RPL (5.2), and F-RPL (3.9). RL modifies choices depending on input over time, which makes things less wobbly. Fuzzy engines, on the other hand, prefer stable parents depending on how much activity and movement they have. Route flapping occurs frequently with reactive protocols, such as EA-RPL, and metric-limited ones, like O-RPL. The proposed two-layer method makes this problem easier to deal with.

The findings of all the testing indicate that CA-RPL is well-designed and clever. In the complex and ever-changing LLN of current IoT systems, CA-RPL helps things operate more efficiently, consume less energy, and be more reliable.

5. Conclusion:

The Context-Aware Reinforced Propagation Framework (CA-RPL) presented is a significant improvement for routing in Low Power and Lossy Networks (LLNs), particularly in situations where resources are limited, and the Internet of Things (IoT) is characterized by its dynamic nature. With the use of fuzzy logic and reinforcement learning, CA-RPL can make intelligent adjustments to factors such as the remaining energy in a node, its movement, and traffic flow. In comparison to five other variants of RPL, the simulation results show that CA-RPL consumes an average of 21% less energy and maintains the network's functionality for 30.2% longer. The Control Packet Overhead, Routing Load Distribution, and Packet Delivery Ratio are some of the measurements being taken. There are several areas in which the technology needs to be enhanced to function effectively over the long term in smart cities, for monitoring industry, and for sensing the environment. To maintain open communication and enable operations to continue for a longer period, CA-RPL identifies a point of equilibrium between managing traffic reduction and routing responsiveness. Lastly, CA-RPL is equipped with a routing technique that is both flexible and energy efficient, making it suitable for use with future Low-Power and Lossy Networks (LLNs) based on the Internet of Things.

There are several positive aspects to CA-RPL; however, it will not work effectively with nodes that have limited processing capacity. A significant amount of computational power is required for its fuzzy logic and reinforcement learning components, which is the reason behind this. For this reason, there will not be sufficient testing since it is more difficult to simulate the ambient dynamics and radio frequency interference that occur in the actual world. CA-RPL will be put through its paces on real-world Internet of Things (IoT) hardware platforms, such as RIOT OS and Contiki-NG, to evaluate its performance in real-time. The optimal routes could be found through the use of federated learning. Blockchain technology could be utilized to enhance routing security, and CA-RPL could be modified to operate with networks that include both mobile and static nodes.

References:

1. Yesuf, F. Y., & Prathap, M. (2021). CARL-DTN: Context Adaptive Reinforcement Learning based Routing Algorithm in Delay Tolerant Network. *arXiv preprint arXiv:2105.00544*.

2. Tarif, M., Mirzaei, A., & Nouri-Moghaddam, B. (2024). Optimizing RPL Routing Using Tabu Search to Improve Link Stability and Energy Consumption in IoT Networks. *arXiv preprint arXiv:2408.06702*.
3. Lamaazi, H., & Benamar, N. (2018). OF-EC: A novel energy consumption aware objective function for RPL based on fuzzy logic. *Journal of Network and Computer Applications*, 117, 42–58.
4. Farzaneh, B., Montazeri, M. A., & Jamali, S. (2024). FLSec-RPL: A fuzzy logic-based intrusion detection scheme for securing RPL-based IoT networks against DIO neighbor suppression attacks. *Cybersecurity*, 7(1), 23.
5. Alilou, M., Babazadeh Sangar, A., Majidzadeh, K., & Masdari, M. (2024). QFS-RPL: Mobility and energy aware multi path routing protocol for the internet of mobile things data transfer infrastructures. *Telecommunication Systems*, 85, 2[illegible]–312.
6. Rabet, I., Fotouhi, H., Alves, M., Vahabi, M., & Björkman, M. (2024). ACTOR: Adaptive control of transmission power in RPL. *Sensors*, 24(7), 2330.
7. Singh, P., & Chen, Y. C. (2019). RPL enhancement for a parent selection mechanism and an efficient objective function. *IEEE Sensors Journal*, 19(21), 10054–10066.
8. Seyfollahi, A., et al. (2021). A Review of Intrusion Detection Systems in RPL Routing Protocol Based on Machine Learning for Internet of Things Applications. *Wireless Communications and Mobile Computing*, 2021, 8414503.
9. Anitha, P., & Kumar, [illegible]. (2024). Comprehensive review on congestion detection, alleviation, and control techniques in IoT networks. *Journal of Network and Computer Applications*, 221, 103[illegible]49.
10. Farag, H., & Stefanovic, C. (2021). Congestion-Aware Routing in Dynamic IoT Networks: A Reinforcement Learning Approach. *arXiv preprint arXiv:2105.09678*.
11. Dey, D., & Ghosh, N. (2024). iTRPL: An Intelligent and Trusted RPL Protocol based on Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2403.04416*.
12. Mehbodniya, A., et al. (2022). Energy-Aware Routing Protocol with Fuzzy Logic in Industrial Internet of Things with Blockchain Technology. *Wireless Communications and Mobile Computing*, 2022, 7665931.
13. Arivubrakan, P., & Kanagachidambaresan, G. R. (2021). Fuzzy Logic based Object Function to Enhance the Quality of Service in Internet of Things. *International Journal of Intelligent Systems and Applications in Engineering*, 9(1), 4725.

14. Royaee, Z., Mirvaziri, H., & Bardsiri, A. K. (2021). Designing a context-aware model for RPL load balancing of low power and lossy networks in the internet of things. *Journal of Ambient Intelligence and Humanized Computing*, 12(2), 2173–2185.

15. Liu, X., Guo, H., & Zhang, C. (2015). A Scalable Context-Aware Objective Function (SCAOF) of Routing Protocol for Agricultural Low-Power and Lossy Networks (RPAL). *Sensors*, 15(8), 19507-19526.

16. Jiang, J., & Liu, Y. (2022). Secure IoT Routing: Selective Forwarding Attacks and Trust-based Defenses in RPL Network. *arXiv preprint arXiv:2201.0693[illegible]*.

17. Raoof, A., Lung, C.-H., & Matrawy, A. (2021). Securing RPL using Network Coding: The Chained Secure Mode (CSM). *arXiv preprint arXiv:21[illegible]8.062[illegible]4*.

18. https://www.kaggle.com/datasets/programmer3/trust-aware-iiot-routing-dataset