PEARL: Power and Delay-Aware Learning-based Routing Policy for IoT Applications

Conference Paper · May 2022

DOI: 10.1109/RTEST56034.2022.9849862

4 authors:

Sahar Rezagholi Lalani
Sharif University of Technology
2 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Bardia Safaei
Sharif University of Technology
24 PUBLICATIONS **303** CITATIONS
SEE PROFILE

Amir Mahdi Hosseini Monazzah
Iran University of Science and Technology
43 PUBLICATIONS **559** CITATIONS
SEE PROFILE

Alireza Ejlali
Sharif University of Technology
142 PUBLICATIONS **2,210** CITATIONS
SEE PROFILE

PEARL: Power and Delay-Aware Learning-based Routing Policy for IoT Applications

Sahar Rezagholi Lalani*, Bardia Safaei§, Amir Mahdi Hosseini Monazzah††, and Alireza Ejlali‖
*§‖Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
††School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
††School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
Email: *srezagholi@ce.sharif.edu,
§bardiasafaei@sharif.edu, ††monazzah@iust.ac.ir, and ‖ejlali@sharif.edu

***Abstract*—Routing between the IoT nodes has been considered an important challenge, due to its impact on different link/node metrics, including power consumption, reliability, and latency. Due to the low-power and lossy nature of IoT environments, the amount of consumed power, and the ratio of delivered packets plays an important role in the overall performance of the system. Meanwhile, in some IoT applications, e.g., remote health-care monitoring systems, other factors such as End-to-End (E2E) latency is significantly crucial. The standardized routing mechanism for IoT networks (RPL) tries to optimize these parameters via specified routing policies in its Objective Function (OF). The original version of this protocol, and many of its existing extensions are not well-suited for dynamic IoT networks. In the past few years, reinforcement learning methods have significantly involved in dynamic systems, where agents have no acknowledgment about their surrounding environment. These techniques provide a predictive model based on the interaction between an agent and its environment to reach a semi-optimized solution; For instance, the matter of packet transmission, and their delivery in unstable IoT networks. Accordingly, this paper introduces PEARL; a machine-learning based routing policy for IoT networks, which is both, delay-aware, and power-efficient. PEARL employs a novel routing policy based on the q-learning algorithm, which uses the one-hop E2E delay as its main path selection metric to determine the rewards of the algorithm, and to improve the E2E delay, and consumed power simultaneously in terms of Power-Delay-Product (PDP). According to an extensive set of experiments conducted in the Cooja simulator, in addition to improving reliability in the network in terms of Packet Delivery Ratio (PDR), PEARL has improved the amount of E2E delay, and PDP metrics in the network by up to 61% and 72%, against the state-of-the-art, respectively.**

***Index Terms*—IoT, Routing, Handover, RPL, Objective Function, Q-Learning, Delay, Power Consumption, Reliability.**

I. Introduction

The advent of Internet of Things (IoT) has significantly changed human lives. With the integration of various technologies such as sensors, micro-controllers, transmitters, wireless media, and other technologies cooperating with the Internet Protocol, this emerging technology has created a widespread domain of applications. These applications include smart cities, industry, smart agriculture, health monitoring systems, unmanned vehicles, and intelligent transportation systems. Therefore, due to the increasing trend in the number of IoT applications, it has been predicted that there will be more than 9 devices per person at the end of 2025 [1]. On the other hand, according to reports published by the McKinsey Global Institute, IoT could have a financial impact of 3.9 to 11.1 trillion dollars on the global economy by the end of 2025 [2]. Therefore, it is essential to provide a standard architecture for IoT requirements. Many efforts have been made to introduce a comprehensive and flexible IoT architecture, but they have not converged into a unified model. Among them, the three-layer architecture is more common. This architecture consists of perception, network, and application Layers [3].

Based on the definition of IoT, this IP-based communicative infrastructure, connects a large number of resource constraint embedded smart devices without human intervention to operate in relatively unreliable environments. Accordingly, IoT encounters several major challenges such as energy consumption, memory space, processing power, PDR, and E2E delay. Keeping these issues in mind, routing across the IoT network is believed as a vital challenge for this technology. Consequently, the Internet Engineering Task Force (IETF) founded the Routing over Low-Power and Lossy Network (ROLL) working group in 2008 to determine the prerequisites for an appropriate IoT routing protocol. This group provided several documentations to specify the requirements of specific IoT applications such as smart homes, smart cities, smart buildings, and industry [4]. Eventually, the group introduced the IPv6 Routing Protocol for Low-power and lossy networks (RPL) in 2012 [4], [5].

RPL is a distance-vector protocol that determines the routing policies with its key component called Objective Function (OF). These policies are designed according to the requirements of IoT applications. OF determines how each node selects its parent to establish its path towards the sink. OF uses one or more node/link metrics to determine the quality of the links between a nodes and its neighbors [6]. These policies enables RPL to organize the nodes in form of a tree, which is called Destination Oriented Directed Acyclic Graph (DODAG). RPL uses its OF along with four control messages including DODAG Information Object (DIO), Destination Information Solicitation (DIS), Destination Advertisement Object (DAO), and DAO-ACK to create the DODAG [4], [1].

Due to the lossy and unstable nature of IoT links, latency and reliability are among the most important metrics, especially for applications, which on-time and reliable delivery of data is required, e.g., natural disaster management and health monitoring systems. Furthermore, in a recent study, a new metric called attachability has been introduced, which measures the capability of routing protocols in assisting the mobile or

stationary nodes in joining, and maintaining their connection to the network based on their routing policies [7]. All of these metrics must be considered while designing advanced routing mechanisms, especially in mobile IoT applications. Machine learning methods have shown to be promising for handling these types of challenges in unstable, and dynamic conditions by providing a predictive model [8]. Accordingly, a number of studies have focused on proposing machine-learning based routing mechanisms for IoT, and Wireless Sensor Networks (WSN) [9], [10], [11], [12]. Furthermore, considering the embedded characteristics of IoT devices, which are used in different environmental conditions, it seems that learning methods independent of environmental conditions, and the IoT network scenarios are more appropriate. Hence, reinforcement learning is an appropriate approach for IoT. Meanwhile, due to the limited energy source, and limited memory capacity of IoT devices, a simple learning algorithm such as Q-learning could be well-adjusted with these limitations.

Accordingly, this paper introduces PEARL; a machine-learning based routing mechanism, which consists of an objective function that employs the Q-learning approach to reduce E2E delay and power consumption simultaneously in static IoT networks. Therefore, it mainly targets the Power-Delay-Product (PDP) routing criterion, which was introduced in [13]. Accordingly, network nodes learn the quality of their neighboring nodes and connection links based on the E2E latency with using Q-learning. For this purpose, the one-hop path delay of the selected parent is considered as the reward in the learning algorithm. Finally, once the node learns the quality of its candidate parent nodes, it chooses the best path with the least delay according to the policies defined in its OF. To evaluate PEARL and compare it with state-of-the-art OFs, we have used the Cooja simulation environment. Cooja is a Java-based simulator used for IoT related studies. This simulator is part of the Contiki operating system [14]. Contiki has been designed for devices with limited processing power and power consumption[15]. Based on the results of our experiments, in addition to the improvement of reliability in all of the evaluated scenarios (in terms of PDR), our proposed routing mechanism has improved the amount of E2E delay by up to 61% compared to its counterpart OFs. This improvement is due to using the learning algorithm and tracing node/link metrics in all of the neighboring nodes. Also, it has been observed that our proposed OF has reduced the amount of PDP by up to 72%, enabling this routing mechanism to reduce power consumption and delay simultaneously.

The rest of this paper is organized as follows: Section II explains the standardized RPL, and fundamentals of reinforcement learning, with a focus on Q-learning. A comprehensive description of the PEARL will be provided in section III. The experiments, and evaluation results will be discussed in section IV. In section III, related studies will be reviewed. Finally, the paper will be concluded in Section VI.

II. Background

A. An Overview of the Standard RPL

According to IoT's low power and lossy nature, it is essential to determine an appropriate routing policy to respond to the various challenges such as reliability, latency, power consumption, and stability. In this regard, the IETF introduced the RPL routing protocol for IoT applications. RPL is an IPv6 and distance-vector routing mechanism designed for unstable stationary networks [4]. Given the multi-layer architecture of IoT, this protocol is part of the network layer, and it operates independently on top of different physical layer technologies such as MAC and IEEE802.50.11. RPL creates a tree-shaped network, which is in form of a graph, known as a DAG, with one or more coordinator nodes called root (sink). A DAG consisting of a single sink node is called DODAG. In order to create, update, and maintain the DODAG, RPL utilizes four ICMPv6 control messages: 1) DODAG Information Object (DIO), 2) DODAG Information Solicitation (DIS), 3) Destination Advertisement Object (DAO), and DAO Acknowledgement (DAO-ACK) [1].

The most important control message in RPL is DIO, which is responsible for updating and maintaining the DODAG. At the beginning of DODAG creation, the root sends a broadcast DIO message to all of the neighbor nodes. Each node that receives a DIO from the root selects the root as its preferred parent and updates its routing information and sends them to its neighbor nodes. The DIS message is sent by new nodes to receive DODAG information and explore the neighbor nodes and existing DODAG. Each neighbor node that receives this message sends a unicast DIO message to the DIS sender node. Then the new node selects the best neighbor according to routing policy as its preferred parent [16]. RPL uses DIS and DIO messages to handle the Multi-Point-to-Point (MP2P) traffic pattern. In order to handle Point-to-Point (P2P) and Point-to-Multi-Point (P2MP) traffic patterns, RPL utilizes DAO and DAO-ACK messages. There are two operation modes in these traffic patterns: 1) Storing mode, in which each node maintains its sub-tree information, and 2) Non-storing mode, in which each intermediate node between the end node and the root node does not maintain the path information and sends the DAO message hop by hop in a sequential manner towards the root. Finally, the DAO-ACK will be sent by the receiver node upon an explicit request of the sender node or verification of the received DAO message.

Determining the routing policy in RPL is the responsibility of the most key component of this protocol, known as the objective function. OFs, determine the parent selection policy according to the requirements of IoT applications. For this purpose, OFs use one or more metrics in the DIO message. These metrics can be node metrics that determine the quality of nodes, for instance, the remaining energy, the expected lifetime (ELT), velocity, the geographical location, or link metrics that determine the quality of the link between two nodes like Expected Transmission Count (ETX) and Received Signal Strength Indicator (RSSI). OFs use these metrics to categorize nodes into three sets: 1) Neighbor set, 2) Parent set, and 3) Preferred parent set. Ultimately, each node in the network topology selects its parent from the preferred parent set. In the standard RPL, there are two basic objective functions. First is the Objective Function Zero (OF0), in which the nodes select their preferred parent based on the minimum hop count. The greedy routing policy in this OF selects the parents regardless

of the link situation between the nodes, but the nodes consume the least amount of energy due to the simplicity of the calculations in the stationary networks [17]. The second is the Minimum Rank with Hysteresis Objective function (MRHOF). According to this OF, the nodes choose their preferred parent based on the ETX link metric. Consequently, in this routing policy, nodes select the parent with a more stable link, which increases the number of packets delivered to the root and improves reliability [18].

B. A Brief Overview on the Q-learning

Reinforcement learning is one of the machine learning approaches that is inspired by the nature of human and creature learning. Reinforcement learning is a learning method that deals with how to map actions and situations to achieve maximum reward. In this approach, the Reinforcement learning agent interacts with its environment in a sequence of episodes. This environment consists of several states and possible actions in each state. Eventually, the agent decides the best action in each state based on the maximum reward received from the environment. Reinforcement learning is a Markov Decision Process (MDP) with finite states, actions, and reward sets. Therefore, in the learning process in the current state, there is no need for earlier states and actions, and the values of the immediately preceding state and action are sufficient. there are several algorithms in reinforcement learning. Some are on-policy, and the others are off-policy. The off-policy approach is different from the on-policy approach. The off-policy approach uses two policies. The first is learned about, and that is an optimal policy called the target policy. The Second, which is behavioral policy is to explore all actions to find the optimal actions. Q-learning is an off-policy reinforcement learning algorithm. In this algorithm, there is one table called q-table. Each entry in this table corresponds to a pair of action and state (state, action) [19]. The value of these entries is called the q-value. The q-value of each action and state is calculated and updated according to the bellow equation:

$$Q_{New}(s_t, a_t) = (1\text{-}\alpha)\times\ Q_{old}(s_t, a_t) + \alpha\times\ [r_t + \gamma \times (\max\ Q(s_{t+1}, a))] \quad (1)$$

Where, Q_{new} is the new value of the q-value in the state s_t, and action a_t, α is the learning ratio, Q_{old} is the previous value of the q-value in (s_t,a_t), r_t is the recived reward from the action a_t, γ is the discount factor, and $maxQ(s_{t+1}, a)$ is the maximum q-value in the next state.

III. Description of PEARL Routing Mechanism

In this section, we introduce our proposed PEARL routing mechanism. This routing mechanism consists of an OF, which establishes delay-aware path selection with using the Q-learning approach for evaluating the communication links between a node and its neighbors based on the one-hop E2E delay. Accordingly, PEARL must map the network components to a Q-learning problem. Therefore, in the learning algorithm, it is assumed that the environment of the Q-learning approach is a network topology. According to the contents of section II, in the Q-learning approach, there are limited states in which actions can be taken. In the Q-learning algorithm in PEARL, it is assumed that the states are network nodes and the actions in each state are the selection of neighbors as the parent of that node. Another vital component of the Q-learning approach is the agent that must learn all of the actions in each state and ultimately choose the best action to achieve the maximum reward. As a result, according to the PEARL's Q-learning algorithm, each packet transferred from one end node (leaf node) to the root node is one of the agent components. In fact, each transmitted packet is part of a whole unit known as the agent of the Q-learning algorithm. Therefore, each packet is responsible for the learning process in the Q-algorithm.

In the structure of PEARL, DIO control messages are acting as the agents due to their periodical transmission, and their ability to transmit link metrics and other routing information in their container. Furthermore, it has been assumed that the root node is the final state, which every packet should be delivered to it. Generally speaking, there are two challenges in the employed Q-learning approach as part of the routing in IoT networks. First is the exploration and exploitation phases. One of the important challenges in reinforcement learning algorithms is the trade-off between exploration and exploitation phases. In reinforcement learning algorithms, in order to achieve maximum reward, the agent must select actions that have been trained in the past and have the maximum reward. However, when an agent greedily selects actions with maximum rewards, some actions may not have been trained yet and maybe better qualified than previously trained actions. Therefore, the agent must explore new actions. In this case, the agent is a loser on both sides. Because it also has to learn other actions and not only select the trained actions, in which case it will not receive maximum rewards, and on the other hand, if it greedily selects the trained actions, it will reach the most rewarding actions, but it Will lose the untrained actions with maximum rewards. In PEARL, considering the routing policy and the use of Q-learning approach as an off-policy learning algorithm, it makes this possible that one node selects a candidate parent node with maximum rewards as its preferred parent. At the same time, when a DIO message is received from one of the node's neighbors, the learning process can be implemented online. Therefore, the trade-off between these two phases can be resolved.

Second is the limited memory space of embedded devices. Since q-tables in a Q-learning algorithm has a size equal to $Numberofnodes \times Numberofactions$, maintaining this table in each node consumes a significant amount of memory, especially in dense networks. Therefore, PEARL for each node allocates only one row of this table to the same node. As a result, instead of maintaining the entire q-table, every node holds an array with the size of the number of its neighbors. To achieve the best route with the least delay, PEARL considers the E2E delay of one-hop as a reward for the Q-learning. To calculate the one-hop E2E delay, PEARL inserts the instance of time, which the DIO is sent, in the container of this control message. When the receiving node pulls the DIO message out of its queue and processes it, the receiving node subtracts the current time from the sending time, which is placed in the received DIO message. Therefore, the path delay between a node and its neighboring node (j) can be obtained according

to the following equation:

$$Delay_j = Time_{Current} - Time_{sending} \quad (2)$$

Where $Delay_j$ is the one-hop E2E delay, $Time_{Current}$ is the current time that the node receives the DIO message from one of its neighboring nodes, $Time_{Sending}$ is the time when the DIO message is sent. Given that in the Q-learning algorithms, the purpose is to achieve the maximum reward, so an action with a maximum q-value must be selected in each state. However, according to the PEARL's purpose, to achieve the best route with the minimum delay, to determine the reward with the E2E delay, either the delays must be multiplied by (-1) or the delays must be reduced by a maximum delay value. PEARL calculates the reward for selecting one of the node's neighbors as the preferred parent node according to the following equation, taking into account the maximum E2E delay value.

$$Reward_j = \alpha \times \frac{Delay_{max} - Delay_j}{Delay_{max}} \quad (3)$$

Where $Reward_j$ is the reward obtained when the neighbor node j is selected as the node's preferred parent, $Delay_{max}$ is the maximum time when a packet is transmitted from one node and received by another node. This time includes the transmission time and the time when the receiving node has pulled the packet (DIO message) out of its queue and has started processing the packet. In order to combine this reward with other rewards and penalties, PEARL normalizes the calculated reward and multiplies it by a weight called α.

Given that the root node is the final state and the ultimate objective of routing is to deliver data packets to the this node, PEARL considers a maximum reward value when a node chooses the root node as its preferred parent. Furthermore, PEARL considers another reward (r) in addition to the reward value for nodes one-hop away from the root node. This is beneficial in terms of power consumption, and E2E delay due to the short distances between the one-hop nodes and the root, compared with other nodes. Therefore, the amount of consumed power for transceiving data would be reduced. So, PEARL places the parent ID in the DIO container as well. In order to avoid loops in the network, in the body of Q-learning algorithm, if the node selects the candidate parent with a greater rank than the node, a penalty (-R) is assigned to the candidate parent. Finally, the calculated delay reward is allocated to other network nodes that do not meet the previous conditions. According to the description, the reward is determined by the following equation:

$$Reward_{CP} = \begin{cases} R_{max} & If\ the\ node\ is\ root \\ -R & If\ CP_{rank} > Node_{rank} \\ R_{calculated} + r & If\ CP\ is\ a\ one-hop\ node \\ R_{Calculated} & Otherwise \end{cases} \quad (4)$$

Here, $Reward_{CP}$ is the node's reward when it chooses CP as its preferred parent.

After determining the reward and other configurations, the q-value is calculated and updated according to (1). In PEARL, the maximum q-value of the node is placed in the DIO container to obtain the maximum cumulative reward or path with the minimum delay from one end node to the root node. Therefore, in calculating and updating the q-value of each node, according to (1), the maximum q-value of the candidate parent is applied. Therefore, the PEARL routing policy has considered both one-hop delay and path delay. Finally, in the parent selection function, when a node wants to select the best parent, it selects a parent with the maximum q-value from among its candidate parents. Since network nodes select their parents based on the maximum q-value, so at the beginning of the topology and when the nodes are new in the q-table, an initial value based on the conditions explained for calculating the reward, is assigned to each neighboring node observed. Then, after some episodes and meeting the node with its candidate parents, the q-value converges to its actual value, and then network nodes learn the optimal path with the least E2E delay in the network. According to the contents expressed, the PEARL routing policy is divided into two algorithms: 1) Q-learning algorithm, and 2) Parental selection algorithm, which we will discuss in the following.

A. Q-Learning Algorithm

In PEARL, the Q-learning algorithm is embedded in the DIO process function. Therefore, this algorithm is implemented when a DIO message is received from the neighboring nodes of the node N_C. According to Algorithm 1, in order to update the q-values based on (1), PEARL first defines learning factors (lines 1 and 2). PEARL then calculates the reward obtained from the neighboring node according to the one-hop delay calculated by (2) (lines 5 and 6). The q-table for each node is an array of structures with two components, q-value and candidate parent identification. In order to update the q-value based on (1), if a DIO message is received from a neighbor that N_C has not yet met, the new neighbor is inserted into the parent table and q-table, and then an initial value according to the node's conditions is assigned to its q-value (lines 7 to 11). On the other hand, if the DIO message is received from the met node, its q-value will be updated according to the node's conditions described (lines 12 to 25). Finally, after updating the q-values, the maximum q-value is obtained among all neighboring nodes in the node's q-table. This value and other metrics, such as the parent ID and time of sending the DIO, are then updated and placed in the DIO message to notify neighboring nodes of the node's updated routing information by sending it (line 26).

B. Parent Selection Algorithm

The parent selection algorithm is the second algorithm that PEARL uses to determine the routing policy. According to this algorithm, which has been represented in Algorithm 2, nodes select their best parent when the parent selection function is called due to receiving a DIO message or according to the trickle timer. In this algorithm, the current node is called N_C. If the node's parent table is empty, the node will broadcast a DIS message to explore its neighbors to prevent disconnection with the DODAG (lines 3 to 5). Otherwise, when the q-table is empty, and the node has no options to select its best parent, the node maintains the current parent to prevent DODAG disconnection (lines 6 to 8). On the other hand, when the node

Algorithm 1: Q-Learning Function

Input: DIO Message from Neighbor j
Output: Updated Q_Table
1 **Define:** $Learning\_Rate$, $Discount\_Factor$;
2 **Define:** $\alpha, Currenttime(T_C)$;
3 **Define:** Current node (N_c);
4 **Begin:**
5 DelayMetric(j)← T_C $\Delta DIO_j.SentTime$;
6 Reward$_j$ ← $Calculate\_Reward(DelayMetric(j), Delay_{max}, \alpha)$;
7 **if** *Neighbor j is new* **then**
8 Parent_Set[j].Parent$_{ID}$ ← $DIO_j.ID$
9 Q_Table[j].Parent$_{ID}$ ← $Parent\_Set[j].Parent_{ID}$
10 Q_Table[j].Parent$_{qvalue}$ ← $QValue_{Initial}$;
11 **end**
12 **if** *Neighbor j exists in the Q table* **then**
13 **if** *j is the root node* **then**
14 Update Q_Table[j].Parent$_{qvalue}$ with ($Reward_{max}$+$Reward_j$)
15 **end**
16 **else if** *j is the one-hop neighboring node of the root node* **then**
17 Update Q_Table[j].Parent$_{qvalue}$ with $Reward_j$+ r;
18 **end**
19 **else if** $DIO_j.Rank > N_c.Rank$ **then**
20 Update Q_Table[j].Parent$_{qvalue}$ with $Penalty$;
21 **end**
22 **else**
23 Update Q_Table[j].Parent$_{qvalue}$ with $Reward_j$;
24 **end**
25 **end**
26 */*Update the routing metrics and the highest Q-value in the DIO message*/;*
27 **exit: The Q_Table is updated;**
28

Algorithm 2: Parent Selection

Input: DIO message from neighboring nodes
Output: The best parent
1 **Define:** Current node (N_c);
2 **Begin:**
3 **if** *(Parent_set is empty)* **then**
4 Send(DIS.Broadcast);
// The node should Broadcast DIS message to its neighbor nodes in one hop level
5 **end**
6 **else if** *Q_Table is empty* **then**
7 return dag.PreferredParent;
8 **end**
9 **else if** *Q_Table is not empty* **then**
10 $P_{max} \leftarrow Find\_Q_{Max}(Parent\_Set, Q\_Table)$;
11 **if** $P_{max}.DelayMetric < 0$ **then**
12 N_C.Rank ← -1
13 **end**
14 **else**
15 N_C.Rank ← $P_{max}.Rank + P_{max}.DelayMetric$;
16 **end**
17 return P_{max};
18 **end**
19 **exit: The best parent is chosen;**
20

has at least one candidate parent in the q-table, the parent with the maximum q-value will be selected as the best parent. The node then updates its rank relative to its preferred parent, and the one-hop delay with it. In this case, if the delay value is negative, this value is unacceptable, and the node rank is set to -1 (lines 9 to 18). Hence, the rank of a node is determined based on the E2E delay, and if two nodes have the same parent, their rank can differ from each other based on the path delay of their common parent.

IV. System Setup and Results

We have used the Cooja simulator to evaluate, and compare PEARL with its counterparts in static IoT networks. Cooja is a Java-based simulator used in many IoT studies and it can simulate network nodes with different software and hardware characteristics. Cooja is a flexible simulator; Therefore, many parts of it can be expanded and modified according to the requirements of networks with additional functions. For instance, radio media, energy harvester modules, and mobile nodes (with a plugin) could be simulated in dynamic scenarios. Cooja is part of the Contiki Operating System (OS). OS developers attempted to develop a flexible, lightweight OS for embedded devices that are coping with serious limitations in terms of processing power and memory. Their efforts led into introduction of Contiki OS, which is a C-based operating system ported to many COTS microcontrollers. Examples of these micro-controllers include the Texas Instruments MSP430, and the Atmel AVR. To evaluate

TABLE I: Zolertia One Specifications

Parameter	Value
Micro-Controller Unit (MCU)	2ndMSP430 generation
Architecture	16 bit RISC (Upgraded to 20 bits)
Radio Module	CC2420
Operating MCU Voltage Range	1.8V$<V<$3.6V
CC2420 Voltage Range	2.1V$<V<$3.6V
Operating Temperature	-40$^{\circ}C<\theta<$+85$^{\circ}C$
Operating System Clock Frequency	$f<$16MHz
MCU Active Mode Current @ V_{cc} = 3V	2mA
MCU Low Power (Standby) Mode Current	0.5 µA
Off Mode Current	0.1 µA
Radio Transmitting Mode @ 0dBm	17.4mA
Radio Receiving Mode Current	18.8mA
Radio IDLE Mode Current	426 µA

PEARL, we have employed Zolertia one (Z1) platforms, a class of IoT platforms developed by Zolertia. These platforms utilize the CC2420 radio frequency module for their wireless communications. Furthermore, the CPU employed in the architecture of these platforms is the Texas instruments MSP430. These micro-controllers have a 16-bit RAM and operate at a frequency of 1 MHz. Other specifications of this platform have been illustrated in Table I.

In order to evaluate PEARL, we have considered two scenarios with different node numbers by 25 and 50. In each scenario, nodes were randomly distributed over an area with a size of $10000m^2$. According to Table II, in our simulations, to generate various traffic rates in different scenarios, each node of the network sends User Datagram Protocol (UDP) packets with one of the following packet rate levels: 1) 0.5 Packet/Minute, 2) 1.5 Packet/Minute, and 3) 2.5 packs/minute. As mentioned earlier, the Z1 nodes have been set to send their data packets with a transmission power of 1 MHz (with a power of 0dBm). The transmission range of all nodes in the scenarios has been set to 20 meters and the interference range to 30 meters. In order to accurately evaluate and have sufficient time for network nodes to learn all their neighboring nodes and send their data from the optimal learned path, a simulation

TABLE II: Simulation Configuration

Parameter	Value
Sensing Area	10000m^2
Number of Transmitter Nodes	25, 50
Number of Sink Nodes	1
Communication Range	20m
Interference Range	30m
Transmission Power	0dBm
Traffic rate	0.5, 1.5, 2.5Pkt/Minute
Payload Size	16Bytes
Simulation Duration	3600s

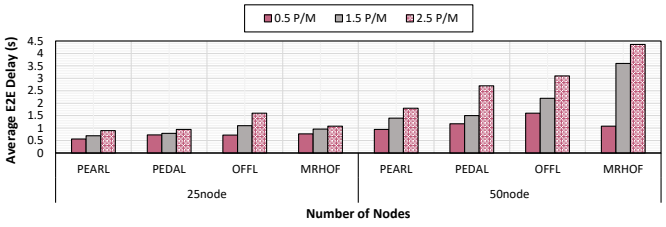

Fig. 1: Evaluated E2E Delay of PEARL in various network scenarios.

time of 3600 seconds is provided for all of the scenarios. Finally, the PEARL will be compared to three state-of-the-art RPL-based routing policies, including MRHOF [18], OFFL [20], and PEDAL [13], which are all explained in Section V.

A. E2E Delay Evaluation

In many IoT applications, such as Remote Health-care Monitoring Systems (RHMS), and Natural Disaster Management (NDM), it is essential to receive data correctly and on time. In these applications, delays can have detrimental effects, and in some cases, can even lead to catastrophes. Therefore, it is necessary to propose a routing policy that takes into account the route delay in these cases. In addition, many IoT applications operate on a point-by-point basis based on multi-point traffic patterns, meaning that the data is collected from end nodes and sent to the root node for processing. Hence, the E2E path delay, which is the delay in sending a packet from the end node to the root node, can be a suitable parameter to improve the performance of the routing policy in terms of latency. For this aim, PEARL has been proposed to improve E2E latency, taking into account the E2E delay along with other performance parameters.

To assess E2E latency, we have measured the average E2E delay for transmitting a data packet and receiving it by the root node. Fig. 1 illustrates our observations in different scenarios. Accordingly, in all scenarios, PEARL improves the average E2E delay compared to its counterparts. According to Fig. 1, in scenarios with a limited number of nodes with 25, reduces the E2E delay by up to 23%, 27%, and 43%, respectively, compared to PEDAL, MRHOF, OFFL. Furthermore, with increasing the rate of packet transmissions in the network, E2E delay has increased for all the evaluated OFs. The main reason is the increase in network congestion and the long waiting time of the packet in the receiver's node buffer. On the other hand, in scenarios where the number of nodes is limited to 50, PEARL has significantly improved this parameter by up to 33%, 61%, and 42%, compared to PEDAL, MRHOF, OFFL, respectively. Fig. 1 also illustrates that as the number of nodes in the network topology increases, the delay increases due to the increase in the transmission of data packets and control packets. Since MRHOF and OFFL consider the ETX metric for their routing policy, they prefer a longer and more reliable path to a faster, shorter one. Hence, both of these OFs have the longest E2E delay. On the other hand, PEDAL exploits the path delay metric and has less E2E latency than the previous two OFs. PEARL has the lowest E2E latency by selecting the optimal path obtained from the Q-learning algorithm. PEARL considers the delay of an E2E hop and the path delay in its algorithm and then uses learning to track the delay of all paths and selects the optimal path. Therefore, unlike PEDAL, it chooses a path with a heuristic method.

B. PDP Evaluation

The power-delay product was first introduced by [21] in 1992. This parameter can be interpreted as the amount of energy consumed in each switching, and it can be calculated by multiplying the power consumption and I/O delay in the internal circuits of embedded devices. Inspired by this parameter, [13] used PDP as a routing metric by multiplying power consumption, and E2E delay. Since the devices used in IoT networks, are embedded devices, they have serious limitations in their energy resources. Therefore, PDP can be useful for comparing power consumption along with E2E delay, especially in IoT applications with critical level of required latency. PDP is calculated based on the following equation:

$$PDP = Power_{Consumed} \times Delay_{E2E} \tag{5}$$

Here, $Power_{Consumed}$ is the power consumption of each node in the network topology, and $Delay_{E2E}$ is the path delay of the nodes to the root node. In this evaluation, the average PDP of all of the network nodes has been evaluated. Fig. 2 depicts the results of our evaluations in different scenarios. Accordingly, PDP for all evaluated OFs increases when the number of nodes or the rate of packet transmission increases. The main reason for this is the increase in the transmission of data packets and control packets. According to [22], the amount of consumed power relates to transceiver and processing activities. In the meantime, the transmission and receiving modules consume a significant portion of the power in each node. Therefore, power consumption increases with increasing the transmission and reception activities in the nodes. Hence, according to Fig. 2 in dense and congested networks, PDP will be increased. According to our observations, PEARL reduces PDP by up to 54% and 43% compared with OFFL and MRHOF, respectively, and is equivalent to PEDAL in scenarios with a maximum of 25 nodes. Furthermore, PEARL improves the PDP by up to 72%, 58%, and 24% compared with OFFL, MRHOF, and PEDAL, respectively, in scenarios limited to 50 nodes.

The main reason for this improvement is the use of the learning method and the preference of nodes located near the sink and the short path with the least delay in the routing

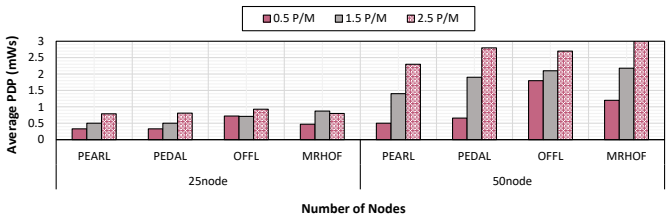

Fig. 2: PDP evaluation of PEARL in various network scenarios.

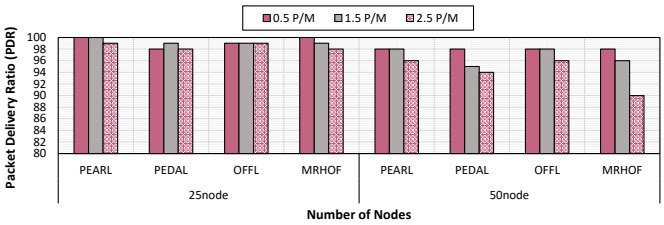

Fig. 3: PDR measurement of PEARL in different network scenarios.

policy. Consequently, unlike OFFL and MRHOF, PEARL selects the path with the least delay and is closest to the root node. In this algorithm, the nodes also learn the optimal paths that have suitable conditions during the simulation time.

C. Packet Delivery Ratio

PDR can be considered as an indication of reliability in IoT, and wireless applications. In order to evaluate the reliability in the network, we have calculated PDR according to (6). Hence, PDR is the ratio of the number of packets delivered to the total number of packets sent to the root node [23].

$$PDR = \frac{Packet_{delivered}}{Packet_{total}} \tag{6}$$

Due to resource constraints and lossy connections in IoT networks, packet loss is inevitable, especially in dynamic or congested networks. According to our experiments on static networks, congestion is one of the main reasons for data packet loss. As shown in Fig. 3, independent of the type of OFs, in scenarios where the number of nodes is constant, the network's congestion is increased as the data packet rate increases. Considering embedded devices deployed in IoT networks, these small devices have significant limitations in their storage. Accordingly, their data buffers are small. These buffers could get filled quickly by increasing the data rate, leading into a congested network. Consequently, when newly received packets are encountered with a full buffer, they have no choice and will be dropped. On the other hand, increasing the number of network nodes increases the number of data packets and control packets. on the other hand, by increasing the number of network nodes, the number of data packets and control packets will be increased; consequently, network congestion will occur. Indeed, PEARL has a higher PDR than its counterparts. The main reason is utilizing the Q-learning algorithm and considering the status of the paths through the operation of the network. Therefore, in this algorithm, when a node observes a declining trend in the quality of its path to the root node by receiving the penalty or reward reduction, the q-value associated with that path decreases, and the node prevents data packet loss by predicting the optimal path. Therefore in all scenarios, PEARL provides a higher PDR than its counterparts.

V. Related Studies

Due to the wide range of IoT applications with different requirements, the two basic OFs in the standard RPL, namely the OF0 and MRHOF, could not meet all of these requirements. Therefore, since the introduction of RPL, many efforts have been made to improve this protocol. Therefore, some studies have attempted to improve RPL by introducing improved RPL protocols, while others have tried to propose new objective functions. The purpose of these studies is to reduce E2E latency, enhance reliability [24], reduce energy consumption, etc. Some of these studies are reviewed in this section.

The authors in [22] have proposed an elaborated cross-layer OF known as ELITE. ELITE is an energy-efficient objective function that uses a novel routing metric called Strobe per Packet Ratio (SPR). A strobe is a packet that is sent from the sending node to the receiver node to inform the receiver node (after they wake up) about the incoming data packets. This OF operates based on Radio Duty Cycling (RDC) protocol in the mac layer, and in conjunction with the ContikiMAC protocol [22]. Another study is OFFL [20]. This OF increases the packet delivery ratio by using a fuzzy logic mechanism. OFFL uses linguistic variables and logical rules to determine the quality of the candidate parents. For this purpose, this OF combines four metric nodes/links, including E2E delay, number of hop, ETX, and residual energy of nodes. Power-Delay-Product (PDP) is another novel metric used in the PEDAL objective function [13]. This metric is calculated by multiplying the total power consumption and the total delay in the path from one node to the root node. This OF aims to reduce power consumption and E2E delay simultaneously in stationary networks [13]. In [25], the authors attempt to learn the quality of the link between nodes and the candidate parent using a reinforcement learning algorithm known as learning automata. Based on the value learned, the algorithm assigns a probability to the link quality metric (ETX), then the ETX with maximum probability is considered as the link metric for the link between a node and its candidate parent. Eventually, each node of the network topology selects the best parent with the minimum ETX.

In addition to studies that improve RPL in stationary networks, other studies have also been performed for dynamic and mobile networks. The author of [26], and [27] tried to upgrade the standard RPL to an improved version of this protocol called ME-RPL and MRPL. These two protocols increase the network reliability by changing the structure of the control message and trickle timer. MAOF is an objective function that has uses five node/link metrics, including rank, ETX, Expected lifetime (ELT), RSSI, and Euclidean distance, to select the best parent. In this OF, a dynamic trickle timer is

employed to control the sending of DIO messages. The goal of MAOF is to increase the reliability of mobile networks [28]. REFER is an improved version of MAOF, in which the PDR on the mobile networks has been improved by proposing an insertion and replacement policy for the neighbor table management. The authors in [29] have proposed an objective function named ARMOR. ARMOR has introduced Time-to-Reside (TTR) metric to determine the quality of candidate parent nodes and manage the neighbor table. TTR is calculated based on the relative velocity of the node to its candidate parent and the Euclidean distance. Ultimately, the candidate parent with the maximum TTR will be selected as the most stable parent. Due to this routing policy, network reliability has increased.

VI. Conclusion

End-to-end delay, is an important requirement for some IoT applications, which the data is required to be delivered successfully, and on-time, such as remote health-care monitoring systems and disaster management. On the other hand, due to limited resources in IoT devices, power-efficient routing is an important challenge for IoT networks. OFs have a significant role in determining routing policies according to IoT requirements. Since using Q-learning algorithms and tracking IoT network behavior can provide an appropriate predictive model for handling the dynamicity in such IoT networks, in this study, we proposed a delay-aware and power-efficient objective function that utilizes the Q-learning algorithm along with the E2E delay metric to select the optimal path with minimum latency and power consumption. According to our evaluations which have been conducted in a stationary network with different scenarios in terms of packet rate and the number of network nodes, in addition to improvement in reliability, PEARL has improved the average amount of E2E delay by up to 61% and reduced the PDP by up to 72%.

References

[1] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali, "Reliability side-effects in internet of things application layer protocols," in *proceedings of the 2nd International Conference on System Reliability and Safety (ICSRS)*. IEEE, 2017, pp. 207–212.

[2] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "The internet of things: Mapping the value beyond the hype," 2015.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Rpl: Ipv6 routing protocol for low power and lossy networks," *RFC 6550*, 2012.

[5] B. Safaei, A. Mohammadsalehi, K. T. Khoosani, S. Zarbaf, A. M. H. Monazzah, F. Samie, L. Bauer, J. Henkel, and A. Ejlali, "Impacts of mobility models on rpl-based mobile iot infrastructures: An evaluative comparison and survey," *IEEE access*, vol. 8, pp. 167 779–167 829, 2020.

[6] J.-P. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing metrics used for path calculation in low-power and lossy networks," Internet Engineering Task Force (IETF), Tech. Rep., 2012.

[7] B. Safaei, H. Taghizade, A. M. H. Monazzah, K. Talaei, P. Sadeghi, A. Mohammadsalehi, J. Henkel, and A. Ejlali, "Introduction and evaluation of attachability for mobile iot routing protocols with markov chain analysis," *IEEE Transactions on Network and Service Management*, 2022.

[8] D. P. Kumar, T. Amgoth, and C. S. R. Annavarapu, "Machine learning algorithms for wireless sensor networks: A survey," *Information Fusion*, vol. 49, pp. 1–25, 2019.

[9] R. Sun, S. Tatsumi, and G. Zhao, "Q-map: A novel multicast routing method in wireless ad hoc networks with multiagent reinforcement learning," in *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02. Proceedings.*, vol. 1. IEEE, 2002, pp. 667–670.

[10] T. Hu and Y. Fei, "Qelar: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 6, pp. 796–809, 2010.

[11] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in neural information processing systems*, 1994, pp. 671–678.

[12] L. R. Campos, R. D. Oliveira, J. D. Melo, and A. D. D. Neto, "Overhead-controlled routing in wsns with reinforcement learning," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2012, pp. 622–629.

[13] B. Safaei, A. A. M. Salehi, M. Shirbeigi, A. M. H. Monazzah, and A. Ejlali, "Pedal: power-delay product objective function for internet of things applications," in *SAC '19: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. Acm, 2019, pp. 892—895.

[14] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local computer networks, proceedings 2006 31st IEEE conference on*. IEEE, 2006, pp. 641–648.

[15] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[16] B. Safaei, A. A. M. Salehi, A. M. H. Monazzah, and A. Ejlali, "Effects of rpl objective functions on the primitive characteristics of mobile and static iot infrastructures," *Microprocessors and Microsystems*, vol. 69, pp. 79–91, 2019.

[17] P. Thubert, "Objective function zero for the routing protocol for low-power and lossy networks (rpl)," Tech. Rep., 2012.

[18] O. Gnawali, "The minimum rank with hysteresis objective function," Tech. Rep., 2012.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[20] O. Gaddour, A. Koubâa, and M. Abid, "Quality-of-service aware routing for static and mobile ipv6-based low-power and lossy sensor networks using rpl," *Ad Hoc Networks*, vol. 33, pp. 233–256, 2015.

[21] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992.

[22] B. Safaei, A. M. H. Monazzah, and A. Ejlali, "Elite: An elaborated cross-layer rpl objective function to achieve energy efficiency in internet-of-things devices," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1169–1182, 2020.

[23] B. Safaei, A. M. H. Monazzah, T. Shahroodi, and A. Ejlali, "Objective function: A key contributor in internet of things primitive properties," in *Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE, 2018.

[24] S. R. Lalani, A. A. M. Salehi, H. Taghizadeh, B. Safaei, A. M. H. Monazzah, and A. Ejlali, "Refer: A reliable and energy-efficient rpl for mobile iot applications," in *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE, 2020, pp. 1–8.

[25] A. Saleem, M. K. Afzal, M. Ateeq, S. W. Kim, and Y. B. Zikria, "Intelligent learning automata-based objective function in rpl for iot," *Sustainable Cities and Society*, vol. 59, p. 102234, 2020.

[26] I. E. Korbi, M. B. Brahim, C. Adjih, and L. A. Saidane, "Mobility enhanced rpl for wireless sensor networks," in *Third International Conference on The Network of the Future (NOF)*. IEEE, 2012.

[27] H. Fotouhi, D. Moreira, and M. Alves, "Boosting mobility in the internet of things," *Ad Hoc Networks*, vol. 26, pp. 17–35, 2015.

[28] S. Murali and A. Jamalipour, "Mobility-aware energy-efficient parent selection algorithm for low power and lossy networks," *IEEE Internet of Things Journal*, vol. 6, pp. 2593–2601, 2018.

[29] A. Mohammadsalehi, B. Safaei, A. M. H. Monazzah, L. Bauer, J. Henkel, and A. Ejlali, "Armor: A reliable and mobility-aware rpl for mobile internet of things infrastructures," *IEEE Internet of Things Journal*, 2021.