IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 13, 2021, accepted May 18, 2021, date of publication May 27, 2021, date of current version June 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3084217

RIATA: A Reinforcement Learning-Based Intelligent Routing Update Scheme for Future Generation IoT Networks

ZULQAR NAIN[1], ARSLAN MUSADDIQ[2], YAZDAN AHMAD QADRI[1], ALI NAUMAN[1], MUHAMMAD KHALIL AFZAL[3], (Senior Member, IEEE), AND SUNG WON KIM[1]

[1]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea
[2]ICT Convergence Research Center, Kumoh National Institute of Technology, Gumi 39177, South Korea
[3]Department of Computer Science, Comsats University Islamabad, Wah campus Rawalpindi 47040, Pakistan

Corresponding author: Sung Won Kim (swon@yu.ac.kr)

This work was supported in part by the Ministry of Science and Information and Communication Technology (ICT) (MSIT), South Korea, under the Information Technology Research Center (ITRC) support program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2021-2016-0-00313, and in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2018R1D1A1A09082266.

ABSTRACT Future generation Internet of Things (IoT) communication infrastructure is expected to pave the path for innovative applications like smart cities, smart grids, smart industries, and smart healthcare. To support these diverse applications, the communication protocols are required to be adaptive and intelligent. At the network layer, an efficient and lightweight algorithm known as trickle-timer is designed to perform the route updates and it utilizes control messages to share the updated route information between IoT nodes. Trickle-timer tends to generate higher control overhead ratio and achieves lower reliability. Therefore, this article aims to propose an RL-based Intelligent Adaptive Trickle-Timer Algorithm (RIATA). The proposed algorithm performs three-fold optimization of the trickle-timer algorithm. Firstly, the RIATA assigns higher probability to control message transmission to nodes that have received an inconsistent control message in the past intervals. Secondly, the RIATA utilizes RL to learn the optimal policy to transmit or suppress a control message in the current network environment. Lastly, the RIATA selects an adaptive redundancy constant value to avoid unnecessary transmissions of control messages. Simulation results show that RIATA outperforms the other state-of-the-art mechanisms in terms of reducing control overhead ratio by an average of 21%, decreasing the average total power consumption by 10%, and increasing the packet delivery ratio by 4% on an average.

INDEX TERMS Internet of Things (IoT), trickle-timer, reinforcement learning, RPL.

I. INTRODUCTION

The Internet of Things (IoT) is a communication paradigm that enables physical devices to be integrated with the cyber-world. Recently, IoT has grown exponentially [1]. Billions of devices or things are getting connected to the internet for various applications such as health monitoring, smart industries, smart homes, and smart cities. These devices are usually equipped with sensing and communication capabilities to form an IoT communication infrastructure. The IoT network is generally composed of small battery-powered devices that possess limited memory and computational capabilities. Traditional communication protocols are not suitable for resource-constrained tiny sensor devices. At the network layer, for efficient data delivery, there are a number of control overheads involved that consume valuable resources of these devices [2]. To handle the network layer operation, the Internet Engineering Task Force (IETF) is continuously striving to design an efficient routing protocol for Low Power and Lossy Networks (LLNs). The IETF working group proposed the Routing Protocol for Low Power and Lossy Networks (RPL) and adopted it in March 2012 [3].

The RPL is an IPv6-based proactive distance vector routing protocol [4]. It builds the Destination-Oriented Directed Acyclic Graph (DODAG) of the network based on different objective functions such as Objective Function zero (OF0) and Minimum Rank with Hysteresis Objective Function (MRHOF). The OF0 utilizes hop counts as a routing metric while the MRHOF is based on Expected Transmission Count (ETX). The DODAGs are built with the help of three

The associate editor coordinating the review of this manuscript and approving it for publication was Stefano Scanzio.

main control messages: DODAG Information Solicitation (DIS), DODAG Destination Advertisement Object (DAO), and DODAG Information Object (DIO) [5]. The DIS control message is used for neighbor discovery, and DIO is utilized to broadcast DODAG information. Similarly, the DAO control message is used to propagate the destination information upward along the DODAG. In DODAG control messages, the primary overhead is owing to DIO messages because these are broadcasted in each interval. The objective of the RPL protocol is to reduce the nodes' energy consumption and minimize the network convergence time. The transmission of control messages consumes limited energy resources. Thus, the energy consumption is directly proportional to the transmission of DIO messages in the network. Maintaining the network performance and reducing the number of control messages is one of the most important design goals of IoT communication protocols.

The transmission of DIO control messages is maintained by a mechanism called a trickle-timer. Owing to its reliability and scalability, the trickle-timer algorithm has become a major research topic among researchers, and it is the focus of numerous recent IoT-related research projects. The trickle-timer algorithm is designed to increase or decrease the frequency of DIO transmissions based on network conditions. It increases the transmission rate of DIO messages if the network is found to be inconsistent. Similarly, it decreases the DIO transmission rate if the network is consistent [6].

The trickle-timer algorithm uses the following key variables for its operation:

1) Minimum interval size I_{min} and maximum interval size I_{max}.
2) Current interval size I.
3) Random time in current interval t_{Timer}
4) Total number of consistent DIO messages received in the current interval c.
5) Number of consistent DIO messages that must be received in the current interval to suppress the DIO transmission, also known as the redundancy constant k.

Fairness in control messages transmission among IoT nodes is one of the major challenges faced by the trickle-timer algorithm. Many researchers have proposed solutions for this problem. One such solution is implemented by a fair broadcast suppression mechanism. In this mechanism, a variable s keeps track of DIO suppression by a node in an interval. In the next interval, a node with DIO suppression in the previous interval is assigned a higher priority to transmit the DIO message.

One of the recently proposed Drizzle algorithm [7] achieves fairness in DIO transmission among nodes by assigning different DIO transmission probabilities to them depending on their DIO transmission history. The variable s keeps track of the total number of DIO transmissions performed by a node in the past intervals. A node with a higher number of DIO transmissions in the past is assigned a lower probability of DIO transmissions in subsequent intervals and vice versa. The Drizzle algorithm also removes the listen-only period, which results in faster propagation of network information to the neighboring nodes. Thus, it results in a shorter network convergence time. However, Drizzle is unable to assign priority to the node that has received an inconsistent DIO messages in the past intervals. In addition, Drizzle algorithm continuously changes it redundancy constant k value and never converges to an optimal value of k.

The capabilities of the trickle-timer algorithm can be enhanced by utilizing machine learning (ML) based mechanisms. Reinforcement Learning (RL) algorithms play a vital role in imparting intelligence to wireless networks in order to enhance their performance [8]. RL-based protocols have also demonstrated their capabilities in other computing fields such as networking [9], computer vision [10], natural language processing [11], security [12], and computer games [13].

To enhance the capabilities of the trickle-timer algorithm, we propose a RL-based Intelligent Adaptive Trickle-Timer Algorithm (RIATA). The contributions of this study are as follows:

1) *Intelligent DIO Transmission:* RL is employed to perform intelligent decisions of DIO transmission or suppression.
2) *Priority Assignment:* RIATA assigns higher priority in terms of transmission probabilities to nodes that have received an inconsistent DIO message in past intervals.
3) *Adaptive Selection of Redundancy Constant:* Redundancy constants are adaptively selected. This adaptive selection is based on the average number of DIO messages received in past intervals.

The remainder of this paper is organized as follows. Section II describes related research work. Section III outlines the proposed RL-based trickle-timer algorithm. Section IV presents a discussion on the performance evaluation of the proposed algorithm, and Section V outlines the potential application areas of the proposed method. Section VI concludes this study.

II. RELATED RESEARCH WORK

Various approaches have been proposed to enhance the trickle-timer efficiency. Trickle-F is a fair broadcast suppression mechanism that manages the load-balancing problem, which considers the load-balancing issue. It maintains a variable s at every node. If DIO is suppressed in the current interval, then the variable s is incremented by one. In the next interval, a random time t_{Timer} is chosen between $[\frac{I}{2^{(s+1)}}, \frac{I}{2^{(s)}}]$. A node with a higher value of s is given a higher probability of transmitting the DIO message early in the next interval. Thus, in this way, it can achieve fair DIO broadcast suppression among nodes [14].

Similarly, the optimized trickle-timer mechanism [15] aims to reduce the latency of the standardized trickle-timer algorithm in resolving inconsistencies. It resets the interval I to I_{min} when an inconsistency is detected and then chooses a random time t_{Timer} between $[0, I_{min}]$ instead of $[\frac{I}{2}, I]$. Thus, resulting in the rapid removal of inconsistencies. However,

it may experience an increase in convergence time due to the listen-only period in the subsequent time intervals.

The adaptive trickle-timer algorithm [16] primarily focuses on having an adaptive redundancy constant k. In this mechanism, each node is allowed to have an adaptive redundancy constant k value depending on the local node density. The value of k must be higher in dense networks and lower in sparse networks.The variable k is given by a function of redundancy counter c, that is, $k = f(c)$.

Conversely, the enhanced trickle-timer (E-trickle) algorithm [17] decreases the network convergence time by removing the listen-only period. It chooses a random time t_{Timer} between $[0, I]$ rather than $[\frac{I}{2}, I]$. It also resets the redundancy counter c at a random time t_{Timer} instead of at the beginning of an interval. To solve the energy consumption problem, an Energy-Aware Adaptive Trickle-Timer (EAAT) algorithm is proposed in [18]. The use of EAAT changes the DIO transmission rate according to the predicted future energy consumption and residual energy of the nodes. Thus, it prolongs the lifetime of the network by decreasing the DIO transmission rate of nodes with less remaining energy.

In the trickle-plus algorithm [19], a network can either have a higher convergence time and lower energy consumption or a lower convergence time and higher energy consumption. Trickle-plus focuses on finding the optimal values for network convergence time and energy consumption. It assigns a new interval size equal to $2 \times I \times SF$, where SF is a shift factor that indicates how many interval doublings can be skipped to achieve a shorter network convergence time with less energy consumption.

A variant of the trickle-timer algorithm, named FL-trickle, was proposed in [20]. The FL-trickle-timer fixes the selection of transmission time at $\frac{I}{2}$ for the first two intervals. Moreover, the FL-trickle-timer also selects a high value of t_{Timer}. These two essential changes lead to quicker update propagation in the case of inconsistency and lower control overheads. However, it does not consider the load-balancing of control message transmission among nodes. Additionally, FL-trickle-timer does not adaptively select a redundancy constant value k.

Another improvement over Trickle-F, named I-Trickle, was proposed in [21]. I-trickle-timer differs from Trickle-F in selecting a random time t_{Timer} among $[0, \frac{I}{2}]$ at the beginning of an interval instead of $[\frac{I}{2^{(s+1)}}, \frac{I}{2^{(s)}}]$ by Trickle-F. Moreover, it also resets the redundancy constant k at the expiry of random time t_{Timer} instead of the start or end of the interval by Trickle-F. Trickle-I achieves lower energy consumption compared to the Trickle-F algorithm. However, the DIO messages transmission load-balancing problem and adaptive selection of the redundancy constant k are not considered.

The Drizzle algorithm presented in [7] achieves fairness in DIO transmission among nodes by assigning different DIO transmission probabilities based on their previous history. This algorithm removes the listen-only period and selects a random time t_{Timer} between $[s \times \frac{I}{n}, (s+1) \times \frac{I}{n}]$, which results in faster propagation of network information to neighbor nodes. Moreover, the Drizzle algorithm adaptively selects its redundancy coefficient k by incrementing or decrementing by one depending on the number of consistent DIO messages received in the current interval. The Drizzle algorithm has proven its dominance over standard trickle, opt-trickle, Trickle-F, and adaptive trickle-timers. However, the Drizzle algorithm is unable to assign priority to a node that has received inconsistent DIO messages in past intervals. In addition, the Drizzle algorithm continuously changes its k value and never converges to an optimal value of k.

The selection of the redundancy coefficient k significantly impacts the control overhead, energy consumption, and time to resolve network inconsistencies. In [22], a mathematical analysis is provided that shows that the single redundancy constant k adopted by standard trickle-timer for all nodes results in higher DIO transmissions and consequently leads to higher energy consumption for nodes with fewer neighboring nodes. The authors in [16] highlighted the vagueness regarding the redundancy constant k configuration in RPL-based networks. For example, the trickle-timer algorithm in Request For Comments (RFC) [6] recommends that the value of k should be between one and five. Alternatively, RPL RFC [3] declares 10 as the default value for k. The most recent IETF draft [23] recommends a k value between three and five. Similarly, the RFC of the multicast protocol for LLNs recommends one as the value of k [24]. These different recommendations indicate that the optimal setting of the redundancy constant k strongly depends on the application scenario.

III. PROPOSED REINFORCEMENT LEARNING-BASED ADAPTIVE TRICKLE-TIMER ALGORITHM

In this section, we present the proposed RL-based adaptive trickle-timer algorithm. This section is divided into three subsections. The first subsection formulates the problem and outlines the system model. The second subsection explains the RL. Finally, the third subsection discusses our proposed RIATA algorithm in detail.

A. PROBLEM FORMULATION AND SYSTEM MODEL

Trickle-timer remains a topic of interest for researchers because of its significant impact on the myriad of performance metrics in a network such as convergence time, control overhead ratio, power consumption, and packet delivery ratio. Striking a balance among these metrics is a critical and complex task. The existing literature on the trickle-timer variants emphasize the challenges in the following areas:

1) A non-adaptive selection of redundancy constant k value leads to increased control overhead ratio and power consumption. therefore, an adaptive redundancy constant k selection is critical for the efficient operation of trickle-timer algorithm.
2) Load balancing in DIO transmission among nodes is detrimental in terms of congestion control as well as power consumption. Thus, achieving load balancing in

DIO transmission is critical to ensure longer battery life in a time-varying ad-hoc network.
3) The nodes which have received an inconsistent DIO message must be given higher DIO transmission probability in coming interval to propagate the updated DIO information as quickly as possible across the network.
4) Intelligent and self-learned decision of DIO transmission or suppression can further fine tune the performance of trickle-timer algorithm.

Table 1 depicts the desired key feature availability in previously proposed major trickle-timer variants. During the writing of this article, there is no known implementation of a trickle-timer that encompasses all these key features.

TABLE 1. Comparison of desired features of trickle-timer algorithms in the state-of-the-art.

Ref.	Adaptive k selection	Load Balancing	Higher DIO transmission probability	Intelligence
[6]	-	-	-	-
[7]	✓	✓	-	-
[14]	-	✓	-	-
[15]	-	-	-	-
[16]	✓	-	-	-
[17]	✓	-	-	-
[18]	-	-	-	-
[19]	-	-	-	-
[20]	-	-	-	-
Proposed	✓	✓	✓	✓

The proposed algorithm utilizes Q-learning to enhance the performance of the trickle-timer algorithm. In our proposed Q-learning based model an IoT node m is considered as an agent and has a set of states, that is, $S = \{s_0, s_1, \ldots, s_n\}$. Moreover, the agent has a set of actions, that is, $A = \{a_0, a_1, \ldots, a_n\}$. The agent performs a particular action a where $a \in A$ in a particular state s where $s \in S$ and obtains a reward R_m^n. Depending upon the received reward, the agent moves to the next state s' where $s' \in S$ with the aim of saving the accumulated reward in the form of a Q-value for a particular state-action pair (s, a). The agent exploits this information for future action selection in a particular state.

B. REINFORCEMENT LEARNING

Embedding ML capabilities into communication networks remains an active research area of this era. A popular ML technique, RL has gained significant attention because of its successful application to network communication domains such as channel access [25], route selection [26], and parent selection [27]. The RL mechanism focuses on learning by interacting with the environment. The learner is referenced as an agent in the RL paradigm. Everything outside the agent is known as the environment. The agent interacts with the environment through the selection of an action among the available action sets. The environment responds back to this action selection. This response is in the form of a reward. The agent changes its state according to the received reward. The agent tries to maximize this reward over time [28]. RL models learn via the Markov Decision Process (MDP), Partially Observed Markov Decision Process (POMDP), Bandit model, and Q-learning [29] techniques. Q-learning, which is an RL technique, has been proven to be effective in a number of wireless network applications. In particular, it has been demonstrated to solve learning problems in computational and energy-constrained networks. Motivated by the promising features of the Q-learning technique, we utilize it to optimize the trickle-timer mechanism.

C. PROPOSED RIATA ALGORITHM

The proposed RIATA algorithm performs three-fold optimization of the trickle-timer algorithm. First, RIATA applies Q-learning to make decisions for DIO transmission or suppression. This results in an intelligent decision regarding DIO transmission or suppression. Intelligent transmission or suppression leads to an optimal control overhead ratio and reduced power consumption. Second, RIATA provides high transmission probabilities to nodes that have received inconsistent DIO messages in the previous interval. It ensures a quicker resolution of inconsistencies across the network. Finally, RIATA sets its redundancy constant k value adaptively based on the value of the average number of consistent DIO messages received in past intervals. This adaptive selection sets the dynamic redundancy constant k for different nodes according to their local network density. The key notations of the proposed RIATA algorithm are summarized in Table-2.

TABLE 2. List of notations used in the paper.

Variables	Description
DIO_m^{sent}	Keeps track of number of DIO transmissions performed by a node m. Will be reset to zero when an inconsistency is detected.
n	Keeps the record of number of intervals between two reset to I_{min}.
c_k	Holds the current value of redundancy coefficient k and this value is achieved by incrementing or decrementing default redundancy coefficient value k.
$incon_m^n$	Keeps track of number of inconsistencies received in an interval n by a node m.
I	Holds the length of current interval.
t_m^n	Random time selected by a node m in current interval n for DIO transmission.
c_m^n	Keeps track of consistent DIO messages received in current interval n by a node m. This is reset to 0 at the end of current interval or whenever an inconsistency is detected.
$Q\ value_m[2][2]$	Holds Q value for every state action pair by for a node m.
R_m^n	Holds the reward received in an interval n by a node m.
$rand$	A random number between zero and one to select exploration or exploitation phase.
$explore$	It keeps track of how much percentage of time algorithm explores the environment.
I_{min}	This is the smallest interval size, electing of this interval size results in higher DIO transmission rate.
I_{max}	This is the largest interval size, electing of this interval size results in slowest DIO sending rate .
$DIOCount_m^n$	Total consistent DIO messages received in the past intervals. This variable is used to calculate average number of DIO received in the past intervals.

In the proposed RIATA algorithm time is divided into intervals; in every interval, the agent has two possible states, that is, $s_0 = DIO\ suppression$ and $s_1 = DIO\ transmission$. The intelligent IoT device performs two actions that are either to remain in the current state s or change state from s to s', where $s' \in S$. In every interval, RIATA maintains a record of the number of consistent and inconsistent DIO messages received. A variable c_m^n holds the number of consistent DIO messages received in the current interval n by a node m, while a variable $incon_m^n$ holds the number of inconsistent DIO messages received in the current interval n by a node m.

1) INTELLIGENT DECISION FOR DIO TRANSMISSION OR SUPPRESSION

Figure 1 shows the state-transition diagram of proposed RIATA algorithm. If the agent in the proposed model is in state s_1, that is, the DIO transmission state, and it receives an inconsistent DIO message $incon_m^{n-1}$ in the previous interval, $n-1$, then the agent will remain in the same state to quickly resolve inconsistencies and will receive a positive reward R_m^n in the current interval n because the decision to transmit the DIO message was correct. However, when an agent is in state s_1 and it receives zero inconsistent DIO messages in the previous interval, $n-1$, it then moves to state s_0. Because it is redundant to transmit DIO message because the network is stable, the agent obtains a zero reward R_m^n in the current interval n because its decision to transmit the DIO message in the past interval was redundant. The agent did not receive any inconsistencies in the previous interval. Now let us suppose that the agent is in state s_0 and receives zero inconsistent DIO message, $incon_m^{n-1}$, in the previous interval, $n-1$. In this case, the agent collects a positive reward for its correct decision of DIO suppression, as the network is stable. However, if in state s_0, the agent receives an inconsistent message $incon_m^{n-1}$ in the previous interval, $n-1$, then the agent receives a negative reward for its incorrect decision of DIO suppression and it moves to state s_1 for DIO transmission in the next interval to resolve inconsistencies.

In every interval n, the proposed RIATA algorithm calculates the reward R_m^n for a particular node m using Eq. (1).

$$R_m^n = \begin{cases} 1 - incon_m^n & \text{if } s = s_0 \\ incon_m^n & \text{if } s = s_1. \end{cases} \tag{1}$$

The RIATA utilizes the calculated reward R_m^n to measure the learning estimate, that is, ΔQ. The $\Delta Q(s,a)$ value is an improved learning estimate for a particular state-action pair, where $a \in A$ and $s \in S$, $\Delta Q(s,a)$ is defined as follows:

$$\Delta Q(s,a) = \{R_m^n(s,a) + \beta \times max_a Q(s',a)\} - Q(s,a) \tag{2}$$

where $0 \leq \beta \leq 1$ is the discount factor. The value of β determines how much importance an agent assigns to future rewards over the current reward, that is, setting $\beta = 0$ means that the agent considers the current reward more vigorously. While setting $\beta = 1$ makes the agent aim for a long-term reward over the current reward. The agent utilizes this calculated $\Delta Q(s,a)$ to calculate $Q^{new}(s,a)$, as given in Eq. (3):

$$Q^{new}(s,a) = Q(s,a) + \alpha \times \Delta Q(s,a) \tag{3}$$

where $0 \leq \alpha \leq 1$ is the learning rate. The learning rate α determines how quickly the new value overrides the previous value. If $\alpha = 0$, the agent does not learn new values and only exploits prior knowledge, whereas $\alpha = 1$ forces an agent to consider the most recent information and ignore prior knowledge. The learning rate plays a vital role in determining how quickly an agent converges to an optimal Q value. The expression $max_a Q(s',a)$ in 2: represents the estimated value for the next state-action pair. When an agent operates for a longer time, this value converges to an optimal value for a state-action pair as follows:

$$\lim_{t \to \infty} Q(s,a) = Q_{optimal}(s,a) \tag{4}$$

The agent utilizes the learned Q–value in Eq. (3) during the exploration phase and chooses an action $a_{(current)}$ that has resulted in the maximum accumulated positive reward in the past at a particular state, i.e,

$$a_{current}^{optimal} = arg_a max Q(s,a) \tag{5}$$

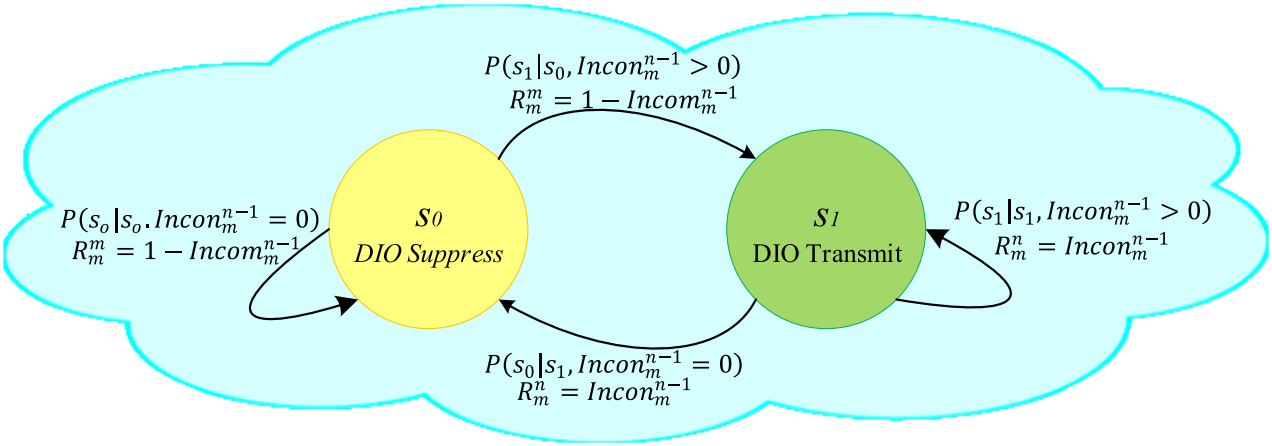

FIGURE 1. State-transition diagram of the proposed algorithm.

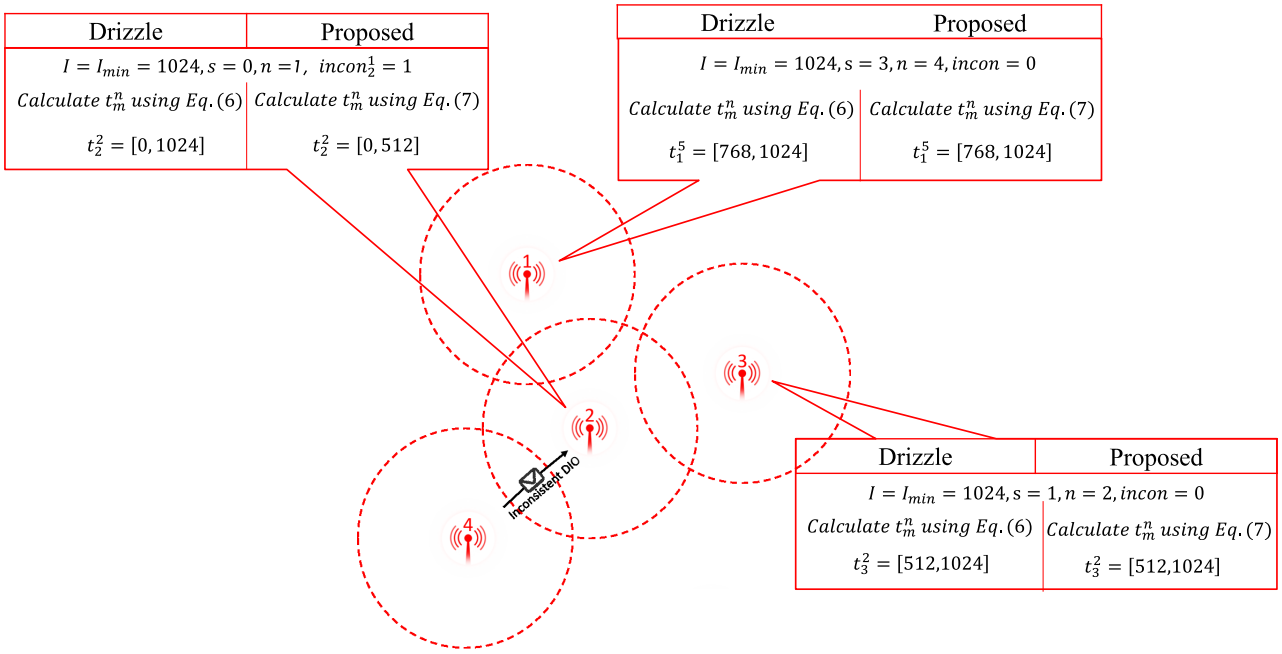

FIGURE 2. Example to show the DIO transmission timer calculation of both Drizzle and proposed algorithm.

To make a trade-off between exploration and exploitation, the proposed RIATA algorithm employs an epsilon-greedy mechanism. In the epsilon-greedy mechanism, the agent explicitly explores the environment with a probability epsilon and performs exploitation for the remaining time.

The proposed algorithm picks up a random number between 0 and 1 to choose among exploration and exploitation. During exploration phase if the number of received consistent DIO messages c_m^n are less than k then DIO is transmitted and s_1 is selected as the next state. On the other hand, if number of received consistent DIO messages c_m^n are less than k then DIO transmission is suppressed and s_0 is selected as the next state. While, in exploitation phase an optimal action is chosen using Eq. (5). In Eq. (5) the proposed algorithm selects an action which has resulted in highest accumulated reward for a particular state action pair in the past intervals.

2) HIGHER TRANSMISSION PROBABILITY

The proposed algorithm assigns higher DIO transmission probabilities to the nodes that has received an inconsistent DIO message in past intervals. It helps to quickly resolve inconsistencies and keep the network status updated. Figure. 2 explains the random time t_m^n selection procedure for the transmission of DIO messages for the next interval using the proposed RIATA and Drizzle algorithms. Let us suppose that the Node 4 DODAG is not up to date and it broadcasts an inconsistent DIO message onto the network. Node 2 receives this inconsistent DIO message, as Nodes 2 and 4 are within transmission range of each other. To select a DIO transmission time t_m^n in the next interval, the Drizzle algorithm selects the random time t_m^n using Eq. (6), While RIATA selects the random time t_m^n using Eq. (7):

$$t_m^n = [s \times \frac{I}{n}, (s+1) \times \frac{I}{n}] \tag{6}$$

$$t_m^n = [DIO_m^{sent} \times \frac{I}{n + incon_m^n}, (DIO_m^{sent} + 1) \times \frac{I}{n + incon_m^n}] \tag{7}$$

$$c_k = \frac{\sum_{I=1}^{n} DIOCount_m^n}{n} \tag{8}$$

The Drizzle algorithm selects a random time t_m^n for the coming interval without considering the total number of received inconsistent DIO messages in the past interval. However, RIATA utilizes this information and selects a random time t_m^n while taking into account the total number of inconsistent DIO messages received in the past interval. While running the Drizzle algorithm, Node 2 selects a random time t_m^n between 0 and 1024. On the other hand, RIATA selects a random time t_m^n between 0 and 512. It is evident that RIATA assigns a higher DIO transmission probability to Node 2 in the coming interval by selecting a random number between 0 and 512 instead of 0 and 1024 by the Drizzle algorithm.

3) ADAPTIVE REDUNDANCY CONSTANT SELECTION

As discussed in Section II, the dynamic selection of the redundancy constant significantly affects the trickle-timer performance. The proposed RIATA algorithm dynamically adjusts its redundancy constant according to the local network density. The RIATA algorithm accumulates the total number of DIO messages received in the past intervals and selects its redundancy constant c_k by taking the average of the total number of DIO messages received in the past intervals using Eq. (8).

The RIATA sums up the total number of DIO messages received in the past intervals and then takes the average value. Upon reception of an inconsistent DIO message, the variable n, which holds the current interval number, and variable $DIOCount_m^n$, which accumulates the total number of DIO messages received in the past intervals, are reset to zero. Moreover, c_k is initialized with the highest value possible, that is, $c_k = k$, to ensure DIO transmission in the coming interval and avoid DIO suppression. Algorithm 1 outlines the Pseudo code of the proposed algorithm.

Algorithm 1 Reinforcement-Learning Based Adaptive Trickle-Timer Algorithm

procedure Initialization
  $I \leftarrow I_{min}$, $c_k \leftarrow k$, $s_m \leftarrow 0$, $c_m^n \leftarrow 0$, $n \leftarrow 1$, $incon_m^n \leftarrow 0$, $reward \leftarrow 0$, $DeltaQ \leftarrow 0$, Q-table $\leftarrow 0$
procedure New Interval
  calculate t_m^n by using Eq. (7)
procedure Consistent DIO Received
  $c_m^n += 1$
procedure Inconsisten DIO Recieved
  I $\leftarrow I_{min}$, $DIOSent_m \leftarrow 0$, $c_m^n \leftarrow 0$, $n \leftarrow 1$, $DIOCount_m^n \leftarrow 0$, $incon_m^n += 1$
procedure t_m^n Expired
  pick a random number between [0,1] (rand)
  to explore or exploit
  if $rand <= explore$ **then**
    if $c_m^n < c_k$ **then** DIO transmit(s_1),$DIO_m^{sent}++$
    else DIO suppress(s_0)
  else select optimal action which has resulted in highest accumulated reward in the past using Eq. (5).
procedure Interval Expired
  calculate R_m^n using Eq. (1)
  calculate ΔQ using Eq. (2)
  update Q-table using Eq. (3)
  $I \leftarrow I \times 2$
  if $I > I_{max}$ **then** $I \leftarrow I_{max}$
  if $DIOCount_m^n = 0$ **then** $c_K \leftarrow k$
  else calculate c_k using Eq. (8)
  $n += 1$
  $DIOCount_m^n += c_m^n$
  $inconc_m^n \leftarrow 0$

IV. PERFORMANCE EVALUATION

To evaluate the performance of our proposed algorithm, a set of parameters are configured using the Cooja 3.0 emulator running on the Contiki operating system. It is an open-source emulator specifically designed for IoT devices [30].

Simulation results of the proposed RIATA algorithm are presented along with an analysis. The proposed algorithm is compared with two variants of the trickle-timer algorithm, that is, the standard trickle-timer and the Drizzle algorithm. The complete list of simulation parameters is provided in Table 3. All simulation parameters utilized in this study are in accordance with the standardized algorithm and Zolertia Z1 mote specifications [31]. Simulations are performed for variable network sizes of 25, 50, 75, and 100 nodes. Packet loss rate of 0%, 10%, 20% and 30% is applied to UDGM model for the network size of 25, 50, 75 and 100 nodes respectively to observe the behaviour of trickle-timer at variable packet loss ratio. There are two types of data transmission rates utilized in the simulations. One is with a fixed data rate of one packet every 40s, while in other scenarios, a variable data rate is applied, where every node selects a random number on start-up in the range of 0 to 60 and selects it as their data rate.To analyze the performance under these scenarios, we evaluate the Packet Delivery Ratio (PDR), total network control overhead ratio, power consumption, and convergence time of the network for the proposed RIATA algorithm and compare them with those of other state-of-the-art mechanisms. The PDR is calculated as expressed in Eq. (9).

TABLE 3. Simulation parameters.

Parameter name	values
Number of sensor nodes	25, 50, 75, 100
Redundancy constant k	5, 7, 10
I_{min} / I_{max}	2^{10} / 2^{20}
Simulation time	60 minutes
MAC/adaptation layer	Contikimac/6LowPAN
Radio medium	Unit disk graph medium (UDGM)
Loss model	Distance loss
Loss ratio	0%, 10%, 20%, 30%
Data packets transmission rate	Variable, 01 packet/40s
Trickle-timer algorithms	Standard, Drizzle, RIATA

$$PDR = \frac{\sum_{i=1}^{m} Total\ packets\ received_i}{\sum_{i=1}^{m} Total\ packets\ sent_i} \tag{9}$$

The total control overhead ratio of the network is calculated by first taking the sum of the total number of DIO, DAO, and DIS messages sent by each node and then taking the sum of the control overhead generated by each node to obtain the total number of control overheads of the entire network as per Eq. (10):

$$\sum_{i=1}^{m} DIO_i + \sum_{i=1}^{m} DAO_i + \sum_{i=1}^{m} DIS_i \tag{10}$$

Subsequently, the ratio of total control overheads generated by the entire network to the total number of control and data packets generated by the network is considered. We also used the Energest module of Cooja to calculate the total power consumed by each sensor node in the network. The total power consumption is calculated using Eq. (11).

$$\frac{Energest\ value \times current \times voltage}{R_{timer} \times Runtime} \tag{11}$$

where the Energest value is the time in ticks that a node has spent in a specific communication mode, that is, the CPU Idle (CPUI), Low-Power (LPM), Transmission (Tx), or Reception (Rx) modes. Moreover, the current and voltage values are in

accordance with the Z1 mote specification. R_{Timer} holds the number of ticks per second, which is 32768 for the Contiki timer, while Runtime is the time interval between the two Energest tracking points.

Network convergence time is the time taken by all the nodes to get connected to a DODAG, forming a network for communicating with another. Network convergence time can be calculated by subtracting the time when the last node joins the network from the time when the first node joins the network.

A. ANALYSIS UNDER FIXED DATA RATE SCENARIO

Figure. 3a displays the PDR of standard trickle-timer, Drizzle, and RIATA at a fixed data rate of 1 packet every 40 s. The simulation results show that, as the number of network nodes increases, the PDR decreases for all trickle-timer variants because of the increased network contention and congestion. Additionally a higher loss ratio is applied at higher network sizes. This results in frequent packet drops as network gets dense; consequently, a lower PDR is attained. Standard trickle-timer has the lowest PDR for all network sizes because of its inability to adjust its redundancy constant k according to the underlying local network density. Drizzle constantly increases or decreases its redundancy constant k value; this hopping helps Drizzle to attain a higher PDR as compared to standard trickle-timer at lower network densities. However, Drizzle never converges to an optimal redundancy constant k value. Thus, as the network size increases, the PDR starts degrading. Conversely, the RIATA is able to attain the highest PDR as compared to standard trickle-timer and Drizzle because of its ability to adjust its redundancy constant k value according to the local network density. Second, RIATA utilizes Q-learning to learn the optimal decision of DIO suppression and transmission over time.

Figure. 3b shows the control overhead ratio of all the trickle-timer variants considered in this study. The control overhead ratio increases as the number of network nodes increases. Standard trickle-timer has the highest control overhead ratio because it is unable to dynamically adjust its redundancy constant value. However, Drizzle achieved a lower control overhead ratio as compared to standard trickle-timer owing to constantly incrementing or decrementing its redundancy constant k value by one. Alternatively, RIATA learns the optimal decision to transmit or suppress a DIO message over time because of its learning capability. Therefore, it suppresses unnecessary DIO transmissions, leading to a lower control overhead ratio. Additionally, RIATA also adjusts its redundancy constant k value as per the local network density. This helps to reduce unnecessary DIO transmissions during the exploration phase.

Figure. 3c shows the normalized total power consumed by the standard trickle, Drizzle, and RIATA algorithms. The results indicate that the power consumption increases with an increase in the number of network nodes. The RIATA algorithm consumes the lowest power compared to other state-of-the-art mechanisms, because RIATA is able to achieve a lower control overhead ratio, as shown in Figure. (b). Reducing unnecessary DIO transmissions helps the RIATA to save energy and eventually makes it capable of operating for a longer duration, as compared to the other two trickle-timer variants.

Figure. 3d shows the network convergence time of all three trickle-timer variants. At lower network densities, Drizzle and RIATA have lower network convergence times as compared to standard trickle, because Drizzle and RIATA eliminate the listen-only period at the beginning of every interval, which results in quicker network propagation at lower network densities. Eventually, this leads to a lower network convergence time for Drizzle and RIATA. The RIATA achieves a lower network convergence time as compared to Drizzle, as RIATA assigns higher DIO transmission probabilities to nodes that have received an inconsistent DIO message in the past interval. This assists in quickly resolving inconsistencies and enables a quicker network convergence time, as explained in Figure. 2.

However, as the network size increases, Drizzle and RIATA start to experience higher network convergence times because they have removed the listen-only period, and this elimination has increased the workload on the radio duty-cycling algorithm to avoid congestion and contention. The slight increase in network convergence time at higher network density is insignificant compared to other performance enhancements as shown in Figure. 3.

B. ANALYSIS UNDER VARIABLE DATA RATE SCENARIO

In IoT networks, sensors generally tend to have variable data rate requirements to support hybrid applications. Therefore, to simulate the behavior of the three trickle-timer variants under such a variable traffic environment, we performed simulations under variable data rates, where every sensor node selects a random number from 1 to 60 on start-up and selects this as its data rate. Figure. 4 show the PDR, control overhead ratio, normalized total network power consumption, and network convergence time of standard trickle, Drizzle, and RIATA under a variable data rate. The simulation results show that the proposed algorithm outperforms other trickle-timer variants by achieving a higher PDR and attaining a lower control overhead ratio. This lower control overhead ratio leads to lower power consumption, which eventually increases the battery life of the sensor nodes. The RIATA is able to achieve a higher PDR and lower control overhead ratio in variable traffic environment as well because of its ability to learn the optimal decision of DIO transmission and suppression over time by utilizing its learning capability. Additionally, RIATA adaptively selects its redundancy constant k value, which helps in adjusting the DIO transmission and suppression rate during the algorithm exploration phase. Although RIATA experiences a slight increase in network convergence time at larger network sizes, as discussed earlier. However, in variable traffic environment as well the other performance enhancements are significant compared to slight increase in network convergence time.

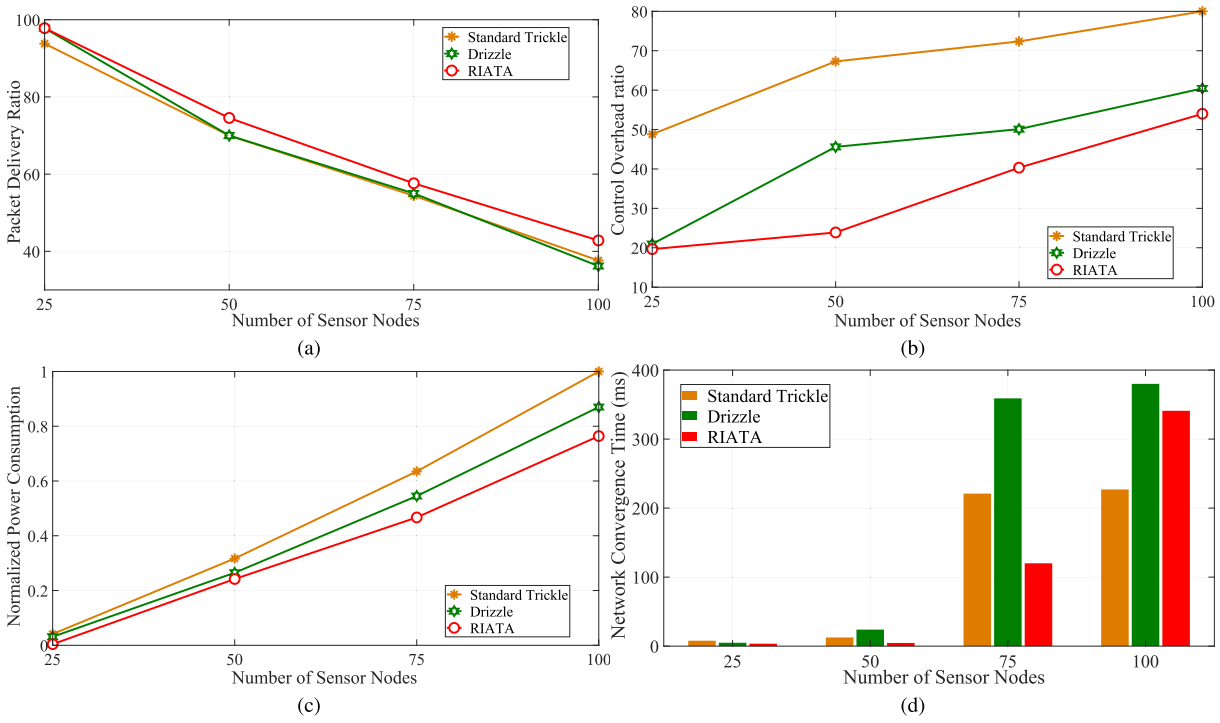

FIGURE 3. Fixed data rate scenario; (a) Packet delivery ratio (PDR); (b) Control overhead ratio; (c) Normalized total network power consumption; (d) Total network convergence time (ms).

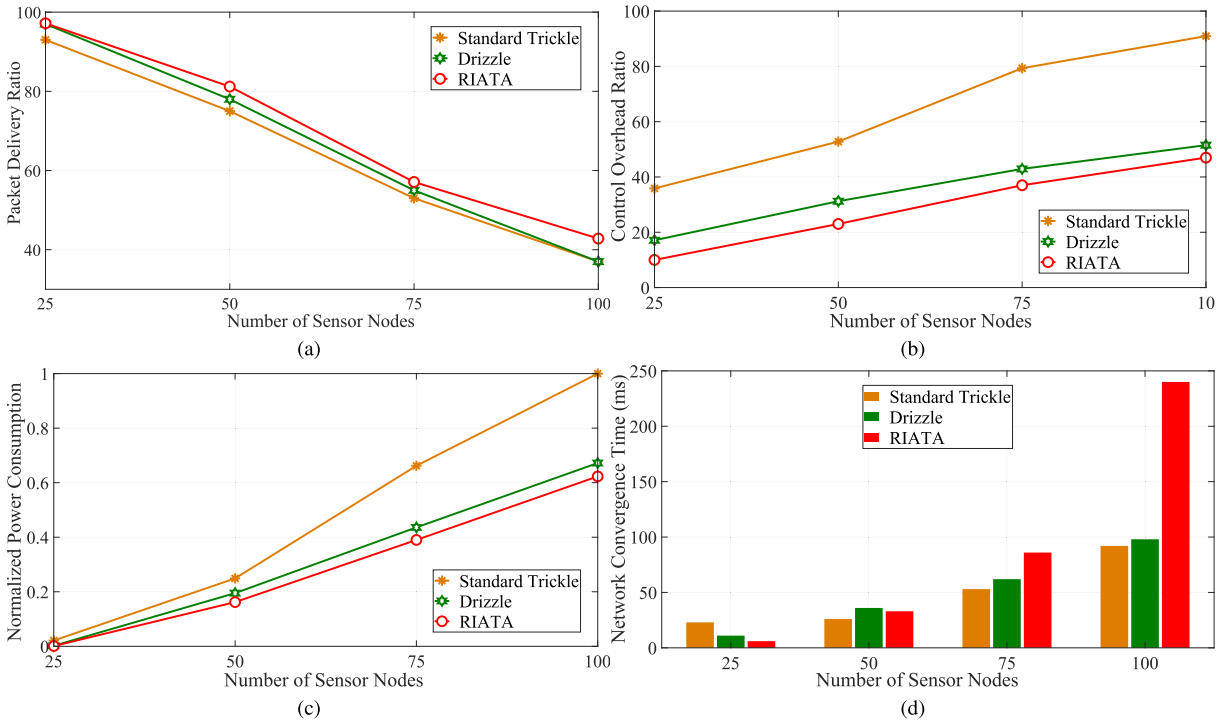

FIGURE 4. Variable data rate scenario; (a) Packet delivery ratio (PDR); (b) Control overhead ratio; (c) Normalized total network power consumption; (d) Total network convergence time (ms).

C. ANALYSIS WITH VARIABLE REDUNDANCY CONSTANT

As briefly discussed in Section II, the dynamic selection of the redundancy constant significantly affects the trickle-timer performance. To illustrate RIATA's ability in dynamically selecting the redundancy constant value as per the local network density, we evaluated these algorithms under variable redundancy constant values. Figure. 5 show the PDR, control overhead ratio, and power consumption of all the algorithms considered in this study. Standard trickle-timer was most sensitive to the variation in redundancy constant

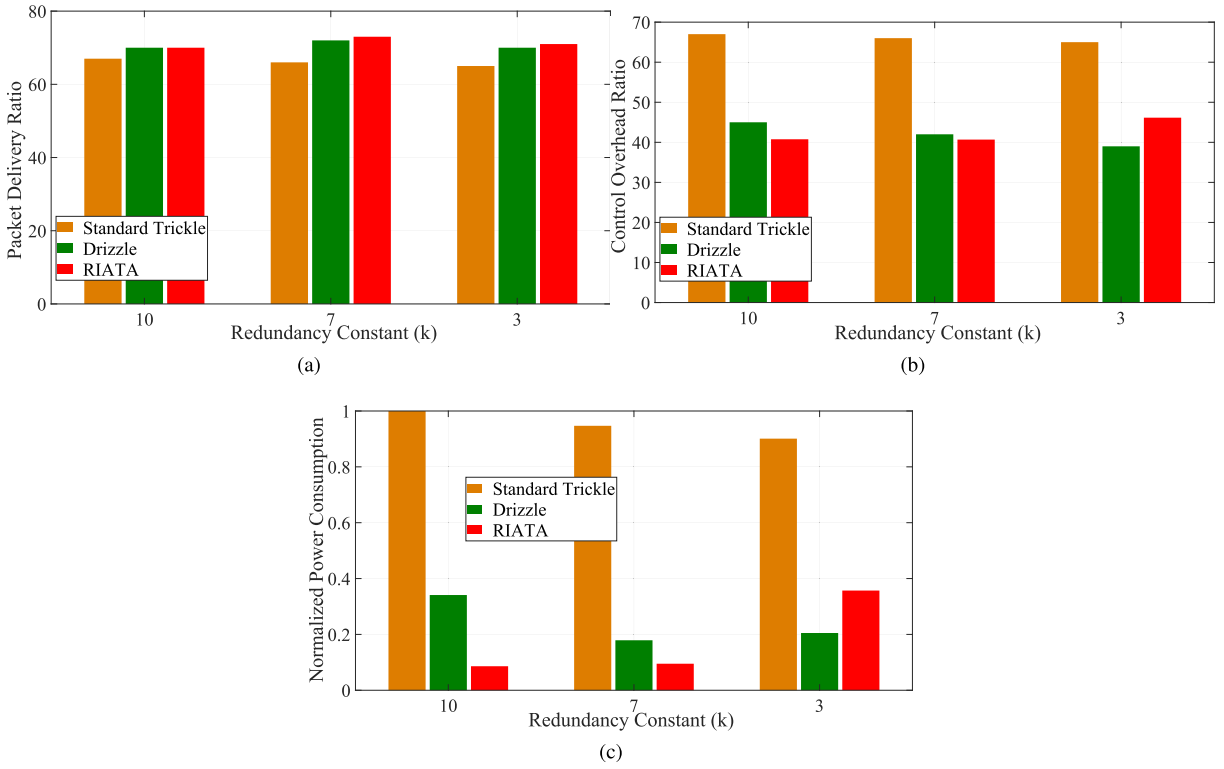

FIGURE 5. Variable redundancy constant k; (a) Packet delivery ratio (PDR); (b) Control overhead ratio; (c) Normalized total network power consumption.

k values, and at higher values of k, it yielded a higher control overhead ratio. Standard trickle-timer transmits a DIO message if the number of received DIO messages in that interval is less than the redundancy constant k. Therefore, for higher values of k, standard trickle-timer requires additional DIO messages to be received in an interval before suppressing a DIO transmission, leading to increased DIO transmissions. In contrast, Drizzle is less sensitive to the variation in k value as it constantly alters its redundancy constant k value between 1 and maximum value of k. Therefore, at lower values of k, it never readjusts its k value according to the local network density, resulting in a lower PDR. Drizzle starts to perform additional DIO suppressions, followed by DIO transmissions, which delays network information propagation and leads to a lower PDR. On the other hand, the RIATA adjusts the k value according to the local network density, as it takes the average of DIO transmissions received in past intervals and sets it as its redundancy constant value. This enables RIATA to achieve a higher PDR, lower control overhead ratio, and lower power consumption at higher redundancy constant values. However, at a lower redundancy constant value, RIATA experiences higher control overhead ratio with increased power consumption. This is because it adjusts the redundancy constant k as per the local network density.

D. ANALYSIS WITH VARIABLE EXPLORATION RATES

The exploration rate also has a significant impact on the performance of an agent in an RL-based environment. We may want an agent to explore more in the beginning so it can explore all the possible options quickly. However, once the agent explores all possible actions, it is desirable to exploit the best learned action. On-the-contrary, it is advisable to continue exploration for optimal action selection. Therefore, appropriate balance between exploration and exploitation rates is crucial for agent performance in the long run.

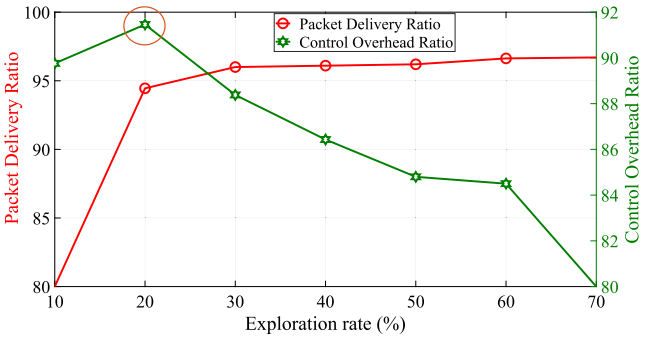

FIGURE 6. Proposed Algorithm behaviour under varying exploration rates.

Figure. 6 shows the performance of the proposed RIATA algorithm under different exploration rates. At higher exploration rates, as the agent spends more time exploring the available options in the environment, the proposed algorithm achieves a higher PDR and a lower control overhead ratio. As we decrease the exploration rate, the control overhead ratio starts to increase. At lower exploration rates, that is, below 20%, the control overhead ratio starts to decrease and the PDR also starts to decline drastically because the agent does not have adequate time to explore all available network options. An exploration rate exceeding 20% yields reasonable

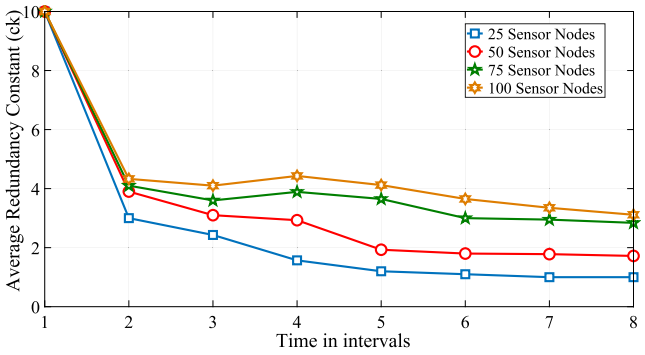

FIGURE 7. Adaptive Selection of redundancy constant c_k.

performance in sensor networks. For more dynamic networks where the environment changes more frequently, we can select a higher exploration rate of 70%. Thus, RIATA utilizes 70% as its exploration rate. The simulation results show that, at this exploration rate, the RIATA achieves a lower control overhead ratio and higher PDR as compared to standard trickle-timer and Drizzle, as shown in Figure. 3 and 4.

E. ANALYSIS OF ADAPTIVE REDUNDANCY CONSTANT c_k SELECTION UNDER VARIOUS NETWORK SIZES

Figure. 7 shows variation of the adaptive redundancy constant c_k value selection under various network densities in the proposed algorithm. When the network initializes, the c_k value set equal to the value of k, which is set to 10 in this case. However, as soon as the network progresses over time the proposed algorithm adaptively selects the c_k value. This adaptive selection is based on the local network density. Therefore, the proposed algorithm selects a higher value of c_k for dense network sizes and a lower value of c_k for lower network densities.

F. ANALYSIS OF COMPUTATION AND MEMORY COMPLEXITY

Determining the optimal state and action pair in the RL paradigm requires the investigation of the entire state and the action space by exploring every available option. The number of actions and states are finite and therefore observable. Every sensor node maintains a Q-value for every possible state and action pair. The number of states and actions significantly affects the computational complexity of the proposed algorithm. The proposed algorithm has a computational complexity of $O(s)(a)$, where s represents the number of states and a represents the number of actions available at every state. The number of states and action pair on a particular sensor node are independent of the network size. As the Q-learning policy is applied locally on the sensor node.

Read-Only Memory (ROM) and Random-Access Memory (RAM) are the two critical resources of IoT nodes. The proposed RIATA algorithm requires additional memory to cater to its RL-based implementation. Thus, it requires an additional 3021 and 2431 bytes of ROM as compared to the standard trickle-timer and Drizzle algorithms, respectively. The RIATA maintains a greater number of variables to cater to the Q-value table, exploration and learning rate, and discount factor and also to maintain the average number of DIO transmissions received in the past intervals for its efficient operation. Therefore, it requires only 48 and 36 bytes of additional RAM for its operation as compared to the standard trickle-timer and Drizzle algorithms, respectively.

V. POTENTIAL APPLICATION AREAS OF THE PROPOSED METHOD

IoT based networks has gained significant attention in the past decade due to their numerous potential applications such as smart cities, smart grid, smart industries and smart health care. These application scenarios have vast potential to impact our daily lives. For example, next-generation industries demands an efficient IoT communication infrastructure with high reliability and lower power consumption to operate for a longer duration of time. The proposed mechanism for next-generation smart industries is shown in Figure. 8. In this figure numerous IoT devices are involved in critical decision

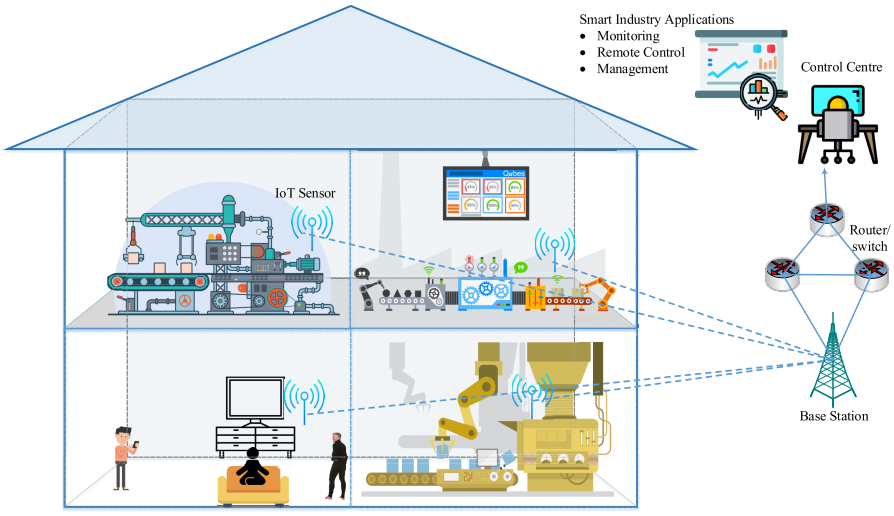

FIGURE 8. Practical application scenario for the proposed mechanism.

making of industrial processes effecting their cost effectiveness. The proposed mechanism enhances the reliability of IoT devices and reduces the power consumption. This reduction in power consumption ensures the longer life span of these battery operated IoT devices. The proposed mechanism can also be applied in other scenarios where reliability and efficient energy consumption are of prime importance.

VI. CONCLUSION

Future IoT communication paradigms require IoT devices to be intelligent and adaptive. The standard IoT communication involves numerous control over heads that greatly impacts its performance. In the standardized RPL protocol the control overheads are managed by a route update mechanism known as trickle-timer. However, trickle-timer does not make intelligent decisions for its control messages exchange. Recently, ML techniques have enabled promising approaches to efficiently optimize IoT devices. Motivated by these ML implementations, in this article, we proposed the RIATA for IoT sensors. The proposed algorithm involves three-fold changes in the trickle-timer algorithm. First, RIATA applies Q-learning for an intelligent decision of DIO transmission or suppression based on past experience. Second, RIATA provides a high DIO transmission probability to nodes in the next interval that have received an inconsistent DIO message in the past interval. This leads to a faster resolution of inconsistencies in the network. Finally, it also adaptively selects its redundancy constant. We implemented the RIATA in the Contiki Cooja emulator and performed an extensive comparative analysis of RIATA with other state-of-the-art trickle-timer algorithms for RPL. The simulation results demonstrate that the RIATA achieves a lower control overhead ratio and consumes less power while maintaining a higher packet delivery ratio, as compared to other techniques.

In future research, we plan to improve the RIATA algorithm to adjust its listen-only period as per local node density. We also plan to focus on federated RL-based RIATA implementation.

REFERENCES

[1] A. Musaddiq, Y. B. Zikria, O. Hahm, H. Yu, A. K. Bashir, and S. W. Kim, "A survey on resource management in IoT operating systems," *IEEE Access*, vol. 6, pp. 8459–8482, 2018.
[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.
[3] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. K. Alexander, *RPL: IPv6 Routing Protocol for Low Power and Lossy Networks*, document RFC 6550, 2012.
[4] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL)," in *Proc. IEEE 7th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2011, pp. 365–372.
[5] N. Accettura, L. A. Grieco, G. Boggia, and P. Camarda, "Performance analysis of the RPL routing protocol," in *Proc. IEEE Int. Conf. Mechatronics*, Apr. 2011, pp. 767–772.
[6] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, *The Trickle Algorithm*, document RFC 6206, Mar. 2011.
[7] B. Ghaleb, A. Y. Al-Dubai, E. Ekonomou, I. Romdhani, Y. Nasser, and A. Boukerche, "A novel adaptive and efficient routing update scheme for low-power lossy networks in IoT," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5177–5189, Dec. 2018.
[8] Y. Yu, S. C. Liew, and T. Wang, "Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels," *IEEE Trans. Mobile Comput.*, early access, Feb. 9, 2021, doi: 10.1109/TMC.2021.3057826.
[9] F. Yao and L. Jia, "A collaborative multi-agent reinforcement learning anti-jamming algorithm in wireless networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 1024–1027, Aug. 2019.
[10] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4423–4430, Oct. 2018.
[11] M. Korpusik and J. Glass, "Deep learning for database mapping and asking clarification questions in dialogue systems," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 27, no. 8, pp. 1321–1334, Aug. 2019.
[12] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2684–2695, Sep. 2019.
[13] Q. Zhang and D. Zhao, "Data-based reinforcement learning for nonzero-sum games with unknown drift dynamics," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2874–2885, Aug. 2019.
[14] C. Vallati and E. Mingozzi, "Trickle-F: Fair broadcast suppression to improve energy-efficient route formation with the RPL routing protocol," in *Proc. Sustain. Internet ICT Sustainability (SustainIT)*, 2013, pp. 1–9.
[15] B. Djamaa and M. Richardson, "Optimizing the trickle algorithm," *IEEE Commun. Lett.*, vol. 19, no. 5, pp. 819–822, May 2015.
[16] T. M. M. Meyfroyt, M. Stolikj, and J. J. Lukkien, "Adaptive broadcast suppression for trickle-based protocols," in *Proc. IEEE 16th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2015, pp. 1–9.
[17] B. Ghaleb, A. Al-Dubai, and E. Ekonomou, "E-trickle: Enhanced trickle algorithm for low-power and lossy networks," in *Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Autonomic Secure Comput., Pervas. Intell. Comput.*, Oct. 2015, pp. 1123–1129.
[18] A. Musaddiq, Y. B. Zikria, and S. W. Kim, "Energy-aware adaptive trickle timer algorithm for RPL-based routing in the Internet of Things," in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2018, pp. 1–6.
[19] B. Ghaleb, A. Al-Dubai, E. Ekonomou, B. Paechter, and M. Qasem, "Trickle-plus: Elastic trickle algorithm for low-power networks and Internet of Things," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2016, pp. 1–6.
[20] H. Lamaazi and N. Benamar, "RPL enhancement based FL-trickle: A novel flexible trickle algorithm for low power and lossy networks," *Wireless Pers. Commun.*, vol. 110, no. 3, pp. 1403–1428, Feb. 2020.
[21] S. Goyal and T. Chand, "Improved trickle algorithm for routing protocol for low power and lossy networks," *IEEE Sensors J.*, vol. 18, no. 5, pp. 2178–2183, Mar. 2018.
[22] T. Coladon, M. Vučinić, and B. Tourancheau, "Multiple redundancy constants with trickle," in *Proc. IEEE 26th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Aug. 2015, pp. 1951–1956.
[23] O. Gnawali and P. Levis, "Recommendations for efficient implementation of RPL, Internet draft," IETF, Chicago, IL, USA, Tech. Rep. draft-gnawali-roll-rpl-recommendations-05, Feb. 2011.
[24] W. J. Hui and R. Kelsey, *Multicast Protocol for Low-Power and Lossy Networks (MPL)*, document RFC 7731, Feb. 2016.
[25] R. Ali, N. Shahin, Y. B. Zikria, B.-S. Kim, and S. W. Kim, "Deep reinforcement learning paradigm for performance optimization of channel observation–based MAC protocols in dense WLANs," *IEEE Access*, vol. 7, pp. 3500–3511, 2019.
[26] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access*, vol. 7, pp. 55916–55950, 2019.
[27] A. Musaddiq, Z. Nain, Y. A. Qadri, R. Ali, and S. W. Kim, "Reinforcement learning-enabled cross-layer optimization for low-power and lossy networks under heterogeneous traffic patterns," *Sensors*, vol. 20, no. 15, p. 4158, Jul. 2020.
[28] S. R. Sutton and G. A. Barto, *Reinforcement Learning: An Introduction*. London, U.K.: MIT Press, 2014.
[29] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.
[30] *Contiki: The Open Source Operating System for the Internet of Things*. Accessed: Feb. 15, 2021. [Online]. Available: http://www.contiki-os.org/
[31] *Z1 Datasheet*. Accessed: Feb. 15, 2021. [Online]. Available: http://wiki.zolertia.com/