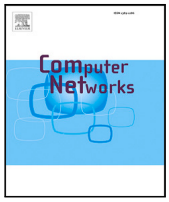
Computer Networks 237 (2023) 110105

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



A novel routing optimization strategy based on reinforcement learning in perception layer networks

Tan Haining [a], Tao Ye [b], Sadaqat ur Rehman [c,*], Obaid ur Rehman [d], Shanshan Tu [b], Jawad Ahmad [e]

[a] Shenzhen High-Tech Industrial Park Information Network Co., Ltd, China
[b] Faculty of Information Technology, Beijing University of Technology, Beijing, 100124, China
[c] School of Sciences, Engineering and Environment, University of Salford, Manchester, UK
[d] Department of Electrical Engineering, Sarhad University of Science and IT, Pakistan
[e] School of Computing, Engineering and Built Environment, Edinburgh Napier University, Edinburgh, UK

ARTICLE INFO

Keywords:
Reinforcement learning
Trickle timer
IoT routing
Optimization
Automation
Wireless sensor networks

ABSTRACT

Wireless sensor networks have become incredibly popular due to the Internet of Things' (IoT) rapid development. IoT routing is the basis for the efficient operation of the perception-layer network. As a popular type of machine learning, reinforcement learning techniques have gained significant attention due to their successful application in the field of network communication. In the traditional Routing Protocol for low-power and Lossy Networks (RPL) protocol, to solve the fairness of control message transmission between IoT terminals, a fair broadcast suppression mechanism, or Drizzle algorithm, is usually used, but the Drizzle algorithm cannot allocate priority. Moreover, the Drizzle algorithm keeps changing its redundant constant k value but never converges to the optimal value of k. To address this problem, this paper uses a combination based on reinforcement learning (RL) and trickle timer. This paper proposes an RL Intelligent Adaptive Trickle-Timer Algorithm (RLATT) for routing optimization of the IoT awareness layer. RLATT has triple-optimized the trickle timer algorithm. To verify the algorithm's effectiveness, the simulation is carried out on Contiki operating system and compared with the standard trickling timer and Drizzle algorithm. Experiments show that the proposed algorithm performs better in terms of packet delivery ratio (PDR), power consumption, network convergence time, and total control cost ratio.

1. Introduction

Emerging network applications like the Industrial Internet [1], the Internet of Things [2,3], and edge computing [4] have raised the efficiency standards for routing protocols in recent years as a result of the Internet's rapid development. However, the current network is no longer as reliable as the conventional wired network, and the network connection status varies often as a result of the access of many mobile IoT terminals [5–7]. The network has substantial difficulties making effective and flexible routing decisions because of the conflict between application needs and network characteristics. Additionally, as service types increase, more optimization goals—including those involving PDR, power usage, network convergence time, and the network's overall control overhead ratio—become necessary [8–10].

In the current dynamic network [11–14], to ensure quality of service (QoS), improving the hardware infrastructure is not only costly but also has limitations on performance improvement. A large number of IoT terminals access the network and need to formulate efficient, adaptive, and intelligent routing strategies to ensure the reliable operation of the IoT. Therefore, many network optimization schemes based on mathematical models have been proposed [15–17]. Numerous limitations and idealized assumptions are used in the majority of these studies to simplify specific scenarios so that mathematical techniques can effectively handle network optimization challenges [18–22]. However, the challenging aspect of actual situations lies in the abundance of unknowns, making it difficult to guarantee the usefulness of this method. Additionally, individually modelling each task not only hampers the network's scalability but also restricts a generic model from simultaneously addressing multiple routing optimization tasks. Therefore, it is anticipated that the RL [22–29] will offer a fresh approach to solving this issue. Deep learning models [30–33] are often able to provide better performance, but it is not always necessary or appropriate to use them. First, the state space and action space defined

* Corresponding author.
E-mail address: s.rehman15@salford.ac.uk (S. ur Rehman).

https://doi.org/10.1016/j.comnet.2023.110105
Received 17 February 2023; Received in revised form 23 July 2023; Accepted 12 November 2023
Available online 14 November 2023

in this paper are small and simple, so basic reinforcement learning methods may be sufficient to achieve good performance. Second, deep reinforcement learning requires greater computational cost as well as more computational data, which may be too complex to use on simple problems.

The development of hardware components like the central processing unit (CPU) and graphics processing unit (GPU) has boosted the learning [34,35] and generalization [36,37] capabilities of the RL model. High precision and significant versatility are the hallmarks of data-driven intelligent routing [38–45]. Different network optimization challenges can be solved using models trained on multiple datasets without the need for elaborate network environment assumptions and modelling [46–48].

RPL (Routing Protocol for low-power and Lossy Networks) is a wireless network routing protocol with low power consumption and easy packet loss. It is an active protocol based on distance vector, running on IEEE 802.15.4, and optimized for multi-hop and many-to-one communication, but it also supports one-to-one messages. The trickle-timer mechanism maintains the transmission interval of the DIO control information of the RPL protocol. In addition, the Trickle-Timer, as an efficient and lightweight routing update algorithm, has gradually become one of the main algorithms for IoT routing. Based on this, we combine the characteristics of RL and trickle timer and propose an intelligent adaptive trickle timer algorithm to improve the efficient operation of IoT routing.

The main contributions of this paper are as follows:

- This paper proposes an adaptive trickle timer algorithm based on RL. In addition, according to the characteristics of the standard trickle timer algorithm, the proposed RLATT algorithm performs triple optimization on the trickle timer algorithm.
- In the experimental section, we assessed the PDR (Packet Delivery Ratio), power consumption, network convergence time, and total network control overhead ratio of the proposed RLATT algorithm. We compared it with other algorithms in the literature and concluded that the proposed algorithm outperformed the existing ones.

The paper's organization is as follows: Section 2 presents the related work, which mainly includes the core concepts related to this paper. Section 3 presents the RL-based IoT routing optimization algorithm. Simulation experiments and result analysis are given in Section 4. Finally, Section 5 provides the conclusion.

2. Related work

2.1. RPL overview

IoT networks usually consist of small battery-powered devices with limited memory, and computing power, so traditional communication protocols are not very applicable. At the network layer, many control overheads are involved for efficient data delivery, consuming valuable resources of these devices. Routing Protocol for Low Power and Lossy Networks (RPL) [49], which organizes the Internet of Things network into a Destination-Oriented Directed A-cyclic Graph (DODAG) with Multi-access edge computing (MEC) as the root, i.e., the DODAG root, was created to handle the operation at the network layer. Network-level traffic ultimately ends up at the DODAG root, which connects the topology to other IPv6 domains (e.g., the Internet). Using DODAG Information Object (DIO) [50] messages, uplink routes are built and maintained. Each node in the PRL protocol broadcasts its communication straight to a designated parent node, which is chosen using the destination function [51]. AK Idrees et al. proposed a review on RPL protocol optimization, covering the fundamental concepts, optimization techniques, application situations, and potential future developments for RPL protocol [52]; A comparison study was carried out by H. Kharrufa *et al.*on a number of RPL protocol optimization strategies, including DAG-based, multicast-based, greedy-based, and optimization based on routing tables [53].

2.2. DODAG build

To implement the construction of DODAG, RPL specifies some control messages: DODAG Information Solicitation (DIS), Destination Advertisement Object (DAO), DIO, etc. The graph construction process starts from MEC, and the root first uses DIO to broadcast information about the graph. Neighbouring nodes listening to the root receive and process DIO messages and decide whether to join the graph based on the destination function, broadcast path spend, etc. Once a neighbour node is added to the graph, it has a route to the DODAG root, and the root becomes the parent of this node. The neighbour node then calculates its rank value in the graph and sends a DAO message containing the routing prefix information to its parent node. IoT terminals can also use DIS messages to proactively request graph information from neighbouring nodes. This process is repeated for all neighbouring nodes until a DODAG with MEC as the root node is constructed in the whole network (see Fig. 1).

DIS control messages are used for neighbour node discovery, DIO is used to broadcast DODAG information, and DAO transmits destination information upwards. Among these control messages, the main overhead is due to the DIO messages, which are broadcasted at each time interval. The RPL protocol aims to lessen node energy usage and speed up network convergence. A finite quantity of energy resources are used up during the transmission of control messages. As a result, the network's DIO message transmission is inversely correlated with energy usage. One of the primary design objectives of the IoT communication protocol is to preserve network performance while minimizing the quantity of control messages.

2.3. Trickle timer

The transmission interval of DIO control information is maintained by a trickle timer mechanism. The trickle timer algorithm has been a major research topic for researchers for its reliability and scalability. It has been the focus of many IoT-related research projects in recent years. The trickle timer algorithm aims to increase or decrease the frequency of DIO transmission depending on the network conditions. According to the standard, it increases the transmission rate of DIO messages if a network inconsistency is detected. Among the inconsistency messages mainly include the detection of loops in the network, failure of a node in the network or a node joining the network, etc. Similarly, if the network is consistent, it decreases the DIO transmission rate [54]. T Clausen introduced the basic principles and applications of Trickle Timer in [55]; Philip Levis introduced the Trickle Timer algorithm in [56] in Applications in wireless sensor networks, and some improved methods are proposed. D Nabil described the basic principle and application of the Trickle Timer in the RPL protocol in [57]. The IoT terminal is equipped with a trickle timer for defining the operation interval and a counter for counting the number of consistent messages that are consistent with the messages sent by the node. The parameters in the trickle timer are explained: $I_{\max}$ is the maximum time interval, $I_{\min}$ is the minimum time interval, I is the current time interval, t_{Timer} is the random time in the current interval, c is the total number of consistent DIO received in the current interval to suppress DIO transmission, also known as redundancy constants, received in the current interval to suppress DIO transmission, also known as redundancy constants. The brief flow of the trickle timer algorithm is described as follows:

1. Set I to a value in $\left[I_{\min}, I_{\max}\right]$ to start the first time slot.
2. The first time slot starts by setting $c = 0, t_{\text{Timer}} = [I/2, I]$ is a point taken arbitrarily in the interval, and each time slot stops at I.
3. When the trickle timer receives a consistency message, let $c+ = 1$.
4. At time timer, the trickle timer checks if there is $c < k$ and only allows packets to be sent if $c < k$.

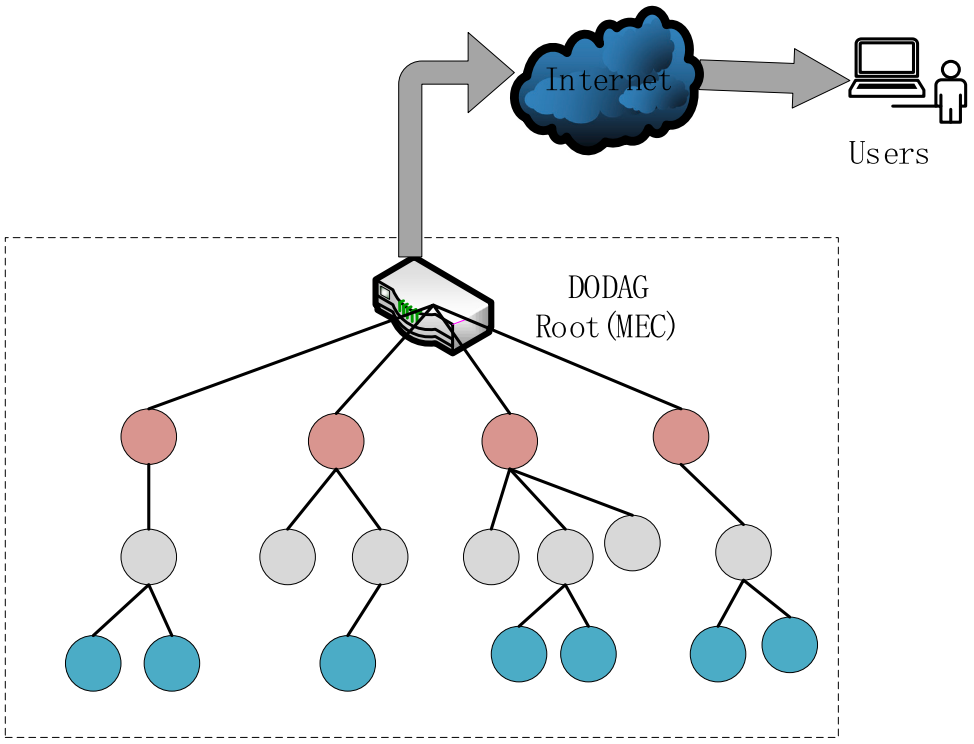


Fig. 1. An overview of RPL Network Model.

5. When I expires, make I^*2, if $I^*2 > I_{\max}$, set $I = I_{\max}$.
6. If the trickle timer receives an inconsistency message, the trickle timer will reset the timer.

One of the key difficulties for trickle timer algorithms is ensuring fairness in the transmission of control messages among IoT terminals. Putting in place a fair broadcast suppression mechanism is one of the fixes. The node's suppression of DIOs during discrete time intervals is monitored in this mechanism by the variable x. Nodes that had DIO suppression during the previous time period are given a higher priority to transmit DIO messages during the following time interval.

To ensure fair DIO transmission among nodes, a recently proposed Drizzle algorithm [58] divides up nodes' DIO transmission rates based on their prior DIO transmissions. The variable s keeps track of all DIO broadcasts made in the previous time period by IoT terminals. IoT terminals are given lower DIO transmission rates in subsequent intervals and vice versa depending on which has more DIO transmissions in the past. Additionally, the Drizzle algorithm does away with listen-only intervals, resulting in quicker network data transmission to neighbouring nodes. It consequently results in quicker network convergence times. However, Drizzle is unable to give priority to nodes that have recently received inconsistent DIO messages. The redundancy constant *mathrmk* value of the Drizzle method is also constantly changing, never converging to the ideal value. We conducted research to address this issue and discovered that the trickling timer algorithm and RL technology work well together as an algorithmic solution.

3. IoT routing optimization algorithm based on RL

This section propose an RL-based adaptive trickle timer method. There are two subsections within this section. The issue and an explanation of RL are presented in the first subsection. The second section goes into great detail on the proposed RLATT algorithm.

As shown in Fig. 2, the overall framework of the RLATT algorithm consists of two parts: the network environment and the agent. The network environment includes a network topology of routers (IoT terminals). The IoT terminals transmit control information among themselves to update routing table information to form a dynamic network environment. The edge server collects information from each IoT terminal in the network in real time. It interacts with the agent information to make decisions in response to state changes in the network.

3.1. Problem analysis

Due to their large effects on a wide range of network performance parameters, including convergence time [59], control overhead ratio [60], power consumption [61,62], and packet delivery rate [63], trickle-down timers have been a popular research issue in the field of network routing. The effort of striking a balance between these measurements is difficult and crucial. The following areas of difficulty are highlighted by the existing literature on trickle timers:

1. The overhead control ratio and power consumption rise when the redundancy constant *mathrmk* is not selected adaptively. Therefore, for the trickling timer method to function properly, adaptive redundancy constant selection is essential.
2. In inter-node DIO transfers, load balancing is harmful for power usage and congestion control. Therefore, in time-varying self-organizing networks, load balancing in DIO transmission is essential to ensuring extended battery life.
3. In order to send the revised DIO information on the network as quickly as feasible, nodes that have received erroneous DIO signals must be allocated a higher DIO transmission rate in subsequent intervals.
4. The trickling timer algorithm's performance can be further optimized by making intelligent and self-learning decisions about DIO transfer or suppression.

Due to its successful implementations in network communications [64], such as channel access [65], routing [66], and parent selection [67], RL approaches have become a well-known machine learning approach. Through the use of Markov Decision Processes (MDPs), RL models are learnt. The trickling timer approach is enhanced by the algorithm given in this chapter using Q-learning. The IoT terminal m is a regarded an agent and possesses a set of states, namely $S = \{s_0, s_1, \ldots s_n\}$ in the

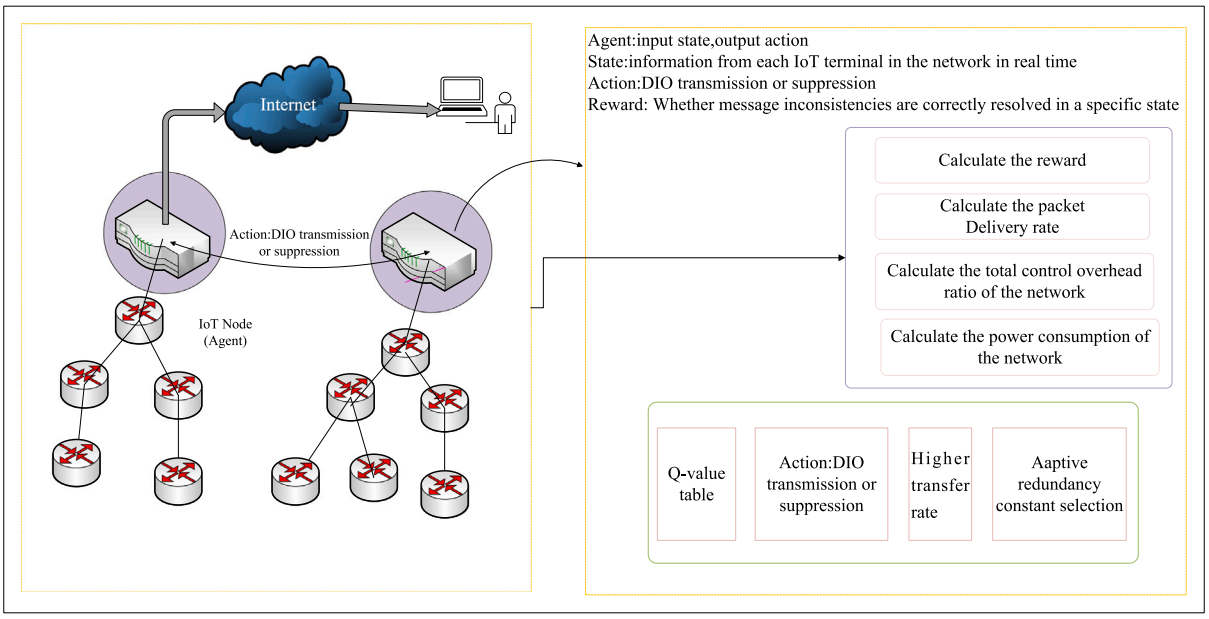


Fig. 2. RL-based IoT routing optimization algorithm RLATT.

Table 1
List of symbols.

Notation	Definition
DIO_m^{sent}	Tracks the number of DIO transmissions performed by IoT endpoint m; resets to zero when inconsistencies are detected
n	Keeps a record of the number of intervals between two resets to I_{min}
c_k	Keeps the current value of the redundancy factor k, which is achieved by increasing or decreasing the default redundancy factor value k
$incon_m^n$	Tracking the number of inconsistent DIO messages received by IoT endpoint m during the current interval n
I	Save the length of the current interval
i_m^n	Random time selected by the IoT terminal m for DIO transmission in the current time interval n
c_m^n	Tracks the number of consistent DIOs received by the IoT terminal m during the current interval n; this value is reset to 0 at the endof the current interval or when an inconsistency is detected
$Qvalue_m$	Save Q values for each state–action pair for IoT terminal m
R_m^n	Save the rewards received by node m in interval n
$rand$	A random number between 0 and 1 to select the exploration or exploitation phase
$explore$	Percentage of time the tracking algorithm explores the environment
I_{min}	This is the minimum interval size, and choosing this interval size will result in a higher DIO transfer rate
I_{max}	This is the maximum interval size, selecting this interval size will result in the slowest DIO send rate
$DIOCount_m^n$	Total number of consistent DIO messages received in the past time interval; this variable is used to calculate the average number of DIOs received in the past time interval

proposed Q-learning-based model. Additionally, the agent has a set of actions, which are $A = \{a_0, a_1, \ldots, a_n\}$. The agent takes a certain action A while being in a specific state S and receiving a reward R_m^n. The agent proceeds to the next state s, where $s \in S$, based on the reward earned. Saving the collected reward as the Q-value of the specific state–action pair is the goal (S, A). In order to decide what to do in the future in a certain state, the agent uses this information.

3.2. Overview of the RLATT algorithm

The RLATT algorithm proposed in this paper triple optimizes the trickle timer algorithm.

1. Q-learning is used by RLATT to assess whether DIO is being transmitted or suppressed. This results in wise choices about the transfer or suppression of DIO. The best control overhead ratio and lower power consumption result from intelligent transfer or suppression.
2. A speedier resolution of network inconsistencies is ensured by RLATT, which offers high transmission rates for nodes that received inconsistent DIO messages during the previous period.
3. Based on the value of the average number of consistent DIO messages received over the previous period, RLATT adaptively sets its redundancy constant k value. The dynamic redundancy constant k is set for various nodes based on their local network density by this adaptive selection. The main notations of the proposed RLATT algorithm are compiled in Table 1.

The proposed RLATT method divides time into intervals, with the agent having two alternative states throughout each interval: s_0 = DIO suppression and s_1 = DIO transmission. The intelligent IoT device carries out two actions: it can either keep the present state s or change

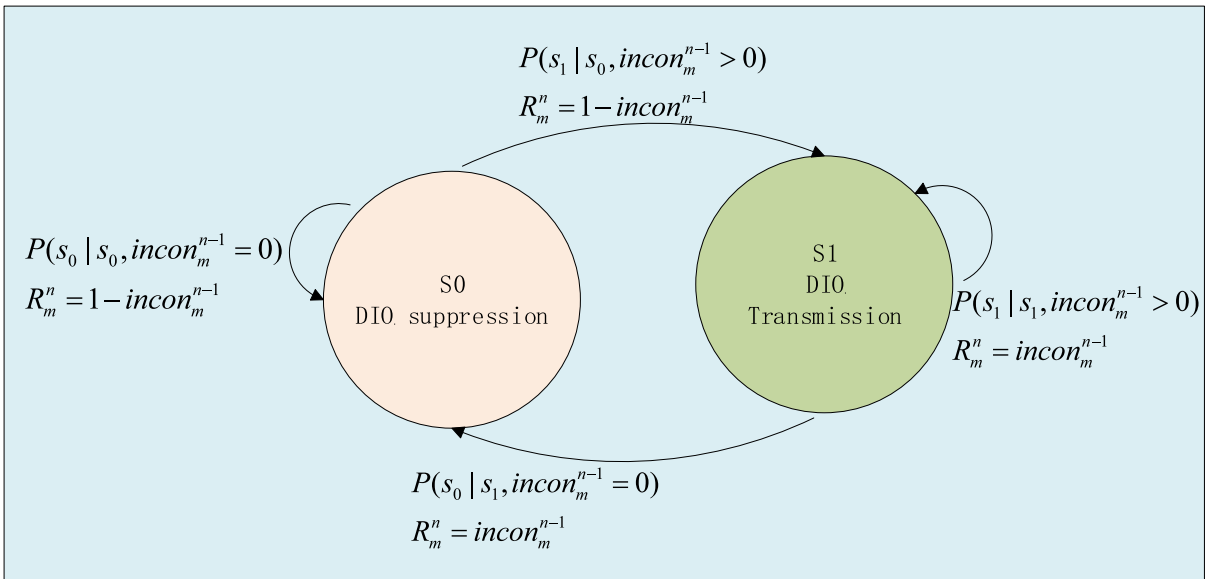


Fig. 3. State transfer diagram of RLATT algorithm.

it to s', where $s \in$ S. The quantity of consistent and inconsistent DIO messages that were received at each time interval is recorded by RLATT. The number of consistent DIO messages that node m has received during the current period n is represented by the variable c_m^n. The number of inconsistent DIO messages that node m has received in the current interval n is shown by the variable incon $_m^n$.

3.2.1. Intelligent decision of DIO transmission or suppression

The proposed RLATT algorithm's state transfer diagram is shown in Fig. 3. If the agent in the proposed model is in state s_1, the DIO transfer state, and it receives inconsistent DIO messages in the previous interval $n-1$ with the number of incon_m^{n-1}, then the agent will remain in the same state to resolve the inconsistency quickly and will receive a positive reward R_m^n in the current interval n because the transfer DIO message decision is correct. However, when an agent is in state s_1 and received zero inconsistent DIO messages in the previous interval $n-1$, it will move to state s_0. Since the network is stable and transmitting DIO messages is redundant, the agent receives a zero reward R_m^n in the current interval n because its decision to transmit DIO messages in the past interval was redundant and in the previous interval the agent did not receive any inconsistent messages. Now let us assume that the agent is in state s_0 and received zero inconsistent DIO messages in the previous time interval n−1. In this case, since the network is stable, the agents collect positive rewards for their correct DIO suppression decisions. However, if in-state s_0 the agent receives inconsistent messages incon n_m^{n-1} in the previous interval $n-1$, the agent receives a negative reward for its incorrect DIO suppression decision, and it moves to state s_1 for DIO transmission in the next interval to resolve the inconsistency.

In each time interval n, the proposed RLATT algorithm calculates the reward R_m^n for a specific IoT terminal m using Eq. (1), and R_m^n is defined as follows:

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

RLATT uses the computed reward R_m^n to measure the learning assessment, i.e., ΔQ. The value of $\Delta Q(s,a)$ 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:

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

Where $\gamma \in [0,1]$ is the discount factor, the value of γ determines how much importance the agent places on future rewards. That is, setting $\gamma = 0$ means that the agent considers current rewards more positively, while setting $\gamma = 1$ causes the agent to seek long-term rewards based on current rewards, and the agent uses the computed $\Delta Q(s,a)$ to calculate Q^{new} as shown in the following Equation:

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

Where $a \in [0,1]$ is the learning rate, the learning rate a determines how quickly the new values cover the previous values. If $a = 0$, the agent does not learn new values and only uses prior knowledge, while $a = 1$, the agent considers only the latest information and ignores prior knowledge. The learning rate is crucial in determining how quickly the agent converges to the optimal Q value. $\max_a Q(s,a)$ represents the estimate of the next state–action pair. This value converges to the optimal value of the state–action pair when the agent runs for a longer time, as follows:

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

The agent uses the Q value learned in Eq. (3) during the exploration phase to select an action $a_{(\text{current})}$ that leads to the maximum cumulative positive reward in a particular state in the past, as follows:

$$a_{\text{current}}^{\text{optimal}} = \arg\max_a Q(s,a) \tag{5}$$

To make a trade-off between exploration and exploitation, the proposed RLATT algorithm uses an ε-greedy mechanism. In the ε-greedy mechanism, the agent explores at a rate of ε and exploits at a rate of $1-\varepsilon$.

The algorithm picks a random number between 0 and 1 to choose between exploration and exploitation. In the exploration phase, if the number of received consistent DIO messages c_m^n is less than k, DIO is transmitted and s_1 is selected as the next state. On the other hand, if the number of received consistent DIO messages c_m^n is greater than k, DIO transmission is suppressed and s_0 is selected as the next state. And in the utilization phase, the best action is selected using Eq. (5). The algorithm presented in Eq. (5) selects the action that has yielded the highest cumulative reward for a particular state–action pair in the past time interval.

3.2.2. Higher transmission rate

The proposed technique gives nodes that received inconsistent DIO messages in the previous time interval a greater DIO transmission

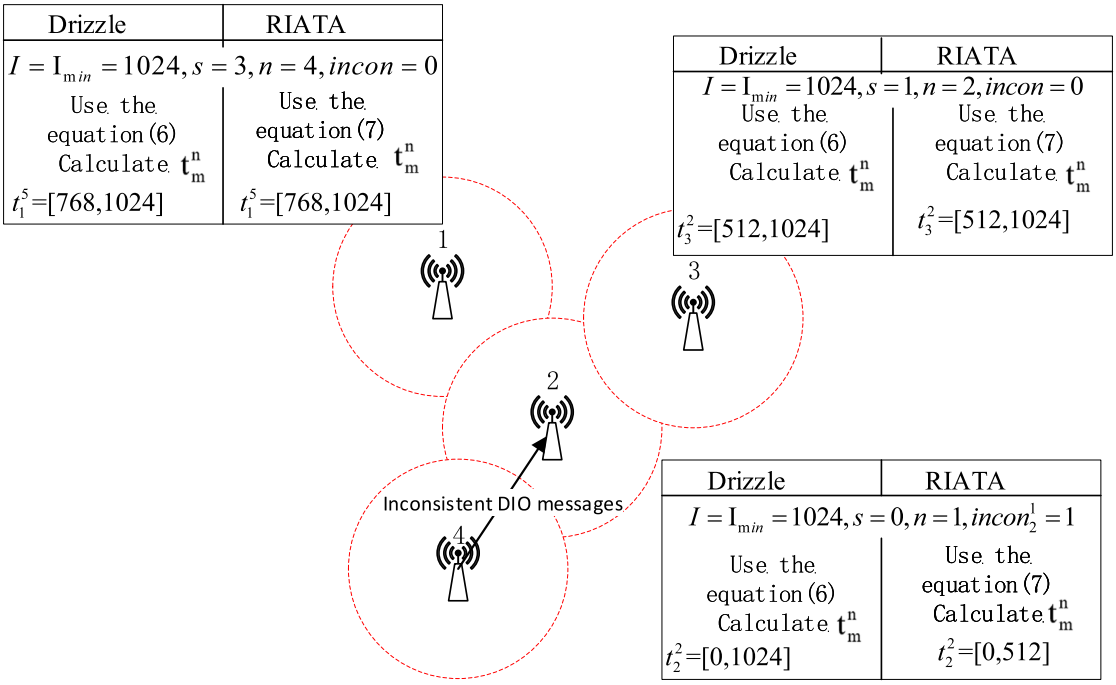


Fig. 4. Drizzle and RLATT algorithms for DIO transfer timer calculation.

rate, which aids in promptly resolving the inconsistent problem and maintaining the network state.

The random time selection procedure for transmitting DIO messages at the following time interval using the RLATT and Drizzle algorithms is described in Fig. 4. We assume that DODAG IoT terminal 4 is out-of-date and sending erratic DIO messages to the network. Due to the proximity of IoT terminals 2 and 4, IoT terminal 2 has received this inconsistent DIO message. To select the DIO transmission time t_m^n in the next interval, the Drizzle algorithm makes Eq. (6) to select the random time t_m^n, while the RLATT algorithm uses Eq. (7) to select the random time t_m^n :

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

$$t_m^n = \left[DIO_m^{\text{sent}} \times \frac{I}{n + \text{incon}_m^n}, (DIO_m^{\text{sent}} + 1) \times \frac{I}{n + \text{incon}_m^n} \right] \tag{7}$$

$$c_k = \frac{\sum_{l=1}^{n} DIO \text{ Count}_m^n}{n} \tag{8}$$

The Drizzle algorithm chooses a random time t_m^n for the upcoming time interval without considering the total number of inconsistent DIO messages received in the past time interval. However, RLATT makes use of this data and selects a random time while taking into account the total amount of inconsistent DIO messages that were received throughout the previous period of time. When running the Drizzle algorithm, IoT terminal 2 selects a random time t_m^n between 0 and 1024. on the other hand, RLATT selects a random time t_m^n between 0 and 512. it is apparent that in the next time interval, RLATT, by selecting a random number between 0 and 512 instead of 0 and 1024 as selected by the Drizzle algorithm Assigning a higher DIO transfer rate to IoT terminal 2.

3.2.3. Adaptive redundancy constant selection

The performance of the trickling timer is strongly impacted by the dynamic selection of redundancy constants. Depending on the local network density, the method dynamically modifies the redundancy constant. Eq. (8) is used to calculate the average of the total number of DIO messages received over the past time interval, which is then added up by the RLATT algorithm to get its redundancy constant c_k.

RLATT sums up the total number of DIO messages received in the past time interval and then takes the average value. In case of receiving inconsistent DIO messages, the variable n that holds the current time interval and the variable DIOCount $_m^n$ that accumulates the total number of DIO messages received in the past time interval are reset to zero. c_k is initialized to the highest possible value, i.e., c_k = k, to ensure DIO transmission in the upcoming interval and avoid DIO suppression. The proposed algorithm's pseudo-code is described in algorithm 1.

4. Experimental analysis

The Cooja 3.0 emulator running on the Contiki operating system is configured with a set of parameters in this study. This is an open-source IoT device emulator [68].

This section provides simulation results and an analysis of the algorithm. The normal trickle timer algorithm and the Drizzle algorithm, two variations of the trickling timer technique, are contrasted with the proposed algorithm. In Table 2, all simulation parameters are listed in detail. Among them, the number of IoT terminals: refers to the number of IoT devices or sensor nodes used in the simulation. Redundancy constant k is a redundancy coefficient for specifying the number of redundant packets used when sending data. I_{min} and I_{max} are used to specify the range of time intervals during the simulation. Simulation time refers to the length of time the emulator performs the simulation. The Media Access Control (MAC) layer is the protocol layer responsible for managing communication between nodes in the Internet of Things. The adaptation layer is an intermediate layer located between the MAC layer and the network layer, and is used to deal with adaptation issues between different wireless technologies and protocols. Wireless medium means a wireless channel or transmission medium between nodes. In wireless communications, signal transmission can be affected by loss and interference. The Cooja emulator provides different loss models and loss rate configuration options for simulating signal transmission reliability and packet loss.

The standard algorithm and the Zolertia *Z1* specification are both followed by all simulation settings utilized in this investigation. Simulations were run with varying network sizes of 25, 50, 75, and 100

Algorithm 1 Adaptive trickle timer algorithm based on RL

Input: $I \longleftarrow I_{min}$, $c_k \longleftarrow k$, $s_m \longleftarrow 0$, $c_m^n \longleftarrow 0$, $n \longleftarrow 1$, $incon_m^{n-1} \longleftarrow 0$, $reward \longleftarrow 0$, $\Delta Q \longleftarrow 0$, $Q-table \longleftarrow 0$

Random time : use equation(6) *to calculate* t_m^n
Receive consistent DIO then : $c_m^n += 1$
Receiving an inconsistent DIO then : $I \longleftarrow I_{min}$, $DIOSent_m \longleftarrow 0$, $c_m^n \longleftarrow 0$, $n \longleftarrow 1$, $DIOCount_m^n \longleftarrow 0$, $incon_m^n += 1$
while *the random time* t_m^n *expires :* **do**
    Choose a random number between [0, 1](*rand*) *to explore and use*
    if $rand \leq explore$ **then**
        if $c_m^n < c_k$ **then**
            $DIO\ transfer(s_1)$, $DIO_m^{sent}++$
        else
            $DIO\ suppression(s_0)$
        end if
    else
        Use equation(5) *to select the best action that has led to the highest cumulative reward in the past*
    end if
end while
while *the time interval expires :* **do**
    Calculate R_m^n *according to Equation*(1)
    Calculate ΔQ *according to Equation*(2)
    Update the $Q-$*tableaccording to Equation*(3)
    UpdateI $: I \longleftarrow I * 2$
    if $I > I_max$ **then**
        $I \longleftarrow I_max$
    end if
    if $DIOCount_m^n = 0$ **then**
        $c_k \longmapsto k$
    else
        Calculate c_k *using Equation*(8)
    end if
    $n += 1$
    $DIOCount_m^n += c_m^n$
    $incon_m^{n-1} \longleftarrow 0$
end while

Table 2
Parameter settings.

Parameter	Value
Number of IoT terminals	25, 50, 75, 100
Redundancy constants k	5, 7, 10
I_{min}, I_{max}	2^{10}, 2^{20}
Simulation time	60 min
MAC layer, Adaptation layer	Contikimac, 6 LoWPAN
Wireless Media	Disk Map Media (UDGM)
Loss Model	Distance loss
Loss rate	10%, 20%, 30%
Packet transmission rate	Standard, Dizzle, RLATT
Trickle timer algorithm	Dackage

nodes. To observe the trickling timer's behaviour under varying packet loss rates, the UDGM model was subjected to packet loss rates of 0%, 10%, 20%, *and* 30% between 25%, 50%, 75%, *and* 100%. The simulations used two different kinds of data transfer rates. One is a fixed data rate of one packet every 40 s, however in other situations a variable data rate is used, where each node selects a random value between 0 and 60 as their data rate at launch. We assessed the suggested RLATT algorithm's PDR, power consumption, network convergence time, and total control overhead ratio of the network and compared it to other cutting-edge methods to analyse the performance in various settings. Eq. (9) illustrates the procedure for determining the PDR.

$$PDR = \frac{\sum_{i=1}^{m} \text{Total packets received}_{\,i}}{\sum_{i=1}^{m} \text{Total packets sent}_{\,i}} \tag{9}$$

As indicated in Eq. (10), the overall control overhead ratio of the network is computed by adding up the total number of DIO, DAO, and DIS messages that each node has sent, followed by adding up the total amount of control (AOC) overhead that each node has generated.

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

The ratio of the overall amount of control overhead to the total amount of data and control packets generated by the network is then taken into account. To determine the total amount of power utilized by each IoT terminal in the network, we also used Cooja's Energest module. Eq. (11) is used to compute the overall amount of power (AOP) consumed.

$$AOP = \frac{\text{Energest value} \times \text{current} \times \text{voltage}}{\text{Rtimer} \times \text{Runtime}} \tag{11}$$

Among these, the Energy consumption values (Energest values) represent the number of seconds that the node spent using a certain mode of communication. CPU Idle (CPUI), Low-Power (LPM), Transmit (Tx), or Receive (Rx) modes are the specialized modes. Additionally, the current (current) and voltage (voltage) values are consistent with the Z1 node specification, the contiki timer's R_ - Timer stores the number of beats per second, which is 32768, and runtime is the amount of time between Energest tracking points.

The length of time it takes for all nodes to join a DODAG and create a network that can talk to one another is known as the network convergence time. By deducting the first node joining the network time from the last node joining the network time, the network convergence time is determined.

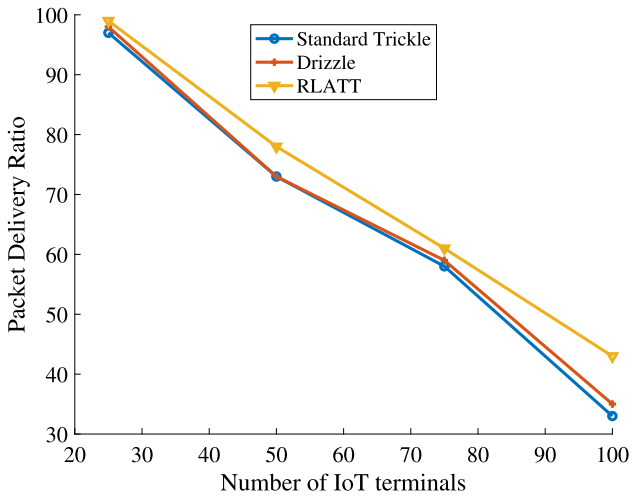


Fig. 5. PDR with different number of IoT terminals.

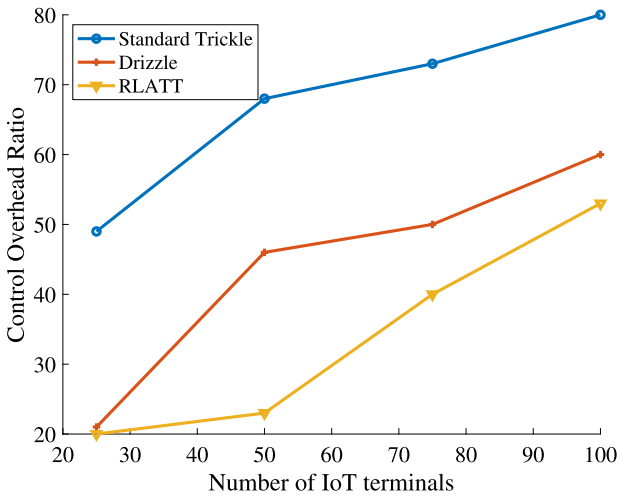


Fig. 6. Control overhead ratio with different numbers of IoT terminals.

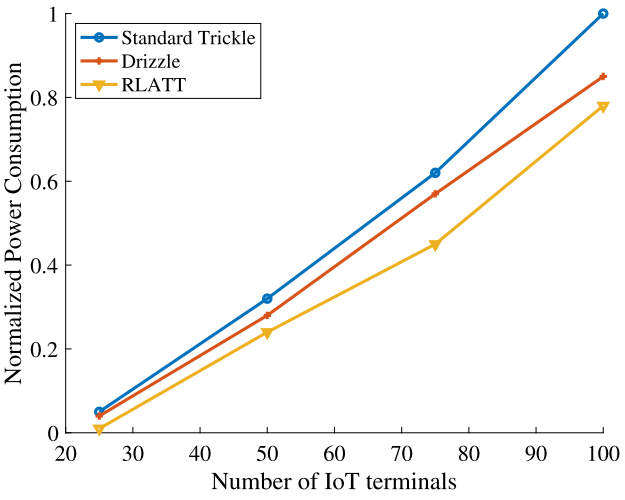


Fig. 7. Power consumption at different numbers of IoT terminals.

4.1. Analysis in a fixed data rate scenario

The PDR for the common trickle timer, Drizzle, and RLATT are shown in Figs. 5 at a fixed data rate of 1 packet per 40 s. One primary reason for the diminishing PDR is the limited network resources available to accommodate the increasing number of terminals. With more devices in the network, there is a higher level of contention and congestion, which leads to reduced efficiency in packet delivery. The network infrastructure faces challenges in handling the surge in data traffic, resulting in longer queuing times and higher packet loss rates. This ratio is very high in the prior art (because the prior art has been widely used), so the improved upper limit is not big, but at least there is a certain improvement. In addition, at larger network sizes, higher loss rates are applied. In addition to network resource limitations, the PDR also decreased significantly at larger network sizes. As the network became more densely populated with IoT devices, the loss rates experienced by the transmitted packets increased. This densification creates an environment with higher levels of interference and contention, leading to a higher likelihood of packet collisions and, consequently, more frequent packet losses. As the network becomes dense, this leads to frequent packet loss, and, therefore, a lower PDR is obtained. Due to its inability to change its redundancy constant k in accordance with the underlying local network density, the typical trickle timer achieves the lowest PDR. Drizzle continuously changes the value of its redundancy constant k, and this jump enables drizzle to achieve a lower PDR at lower network densities than the conventional trickle. Drizzle, on the other hand, never achieves the ideal value for the redundancy constant k. As a result, the PDR starts to decline as the size of the network grows. As opposed to the typical trickle timer and Drizzle, RLATT is able to achieve the highest PDR because it may modify the value of its redundancy constant k in accordance with the density of the local network. The best method to broadcast or suppress control signals is discovered by RLATT using Q-learning.

The control overhead ratios for each trickle timer variable taken into consideration in this study are shown in Figs. 6. With an increase of IoT terminals, the control overhead ratio rises. The inability to dynamically modify its constant redundancy settings causes the typical trickling timer to have the greatest control overhead ratio. However, because of the varying value of its redundancy constant k, Drizzle achieves a lower control overhead ratio than the typical trickle timer. Additionally, because to its capacity for learning, RLATT gradually decides whether it is best to send or suppress DIO messages. As a result, it reduces superfluous DIO transfers, which lowers the control overhead ratio. Additionally, RLATT modifies the value of its redundancy constant k in accordance with the density of the local network. As a result, fewer pointless DIO transfers are made during the discovery stage.

The total normalized power used by the common trickle timer, Drizzle, and RLATT algorithms is displayed in Fig. 7. We store and use the trained RL technique parameters in the simulated experimental environment for presenting the experimental findings. As a result, the experimental process takes longer than the conventional way to reach the suboptimal or ideal answer. It is expected that the power consumption for hardware scheduling will be lower once RLATT is integrated with the conventional RPL protocol. The findings indicate that as the number of IoT terminals rises, so does power consumption. The RLATT method uses the least amount of power when compared to other cutting-edge techniques because it is able to achieve a lower control overhead ratio, as shown in Fig. 7, and the elimination of pointless DIO transfers allows it to run longer than the other two trickling timers.

The network convergence times for each of the three trickle timer versions are shown in Fig. 8. Because Drizzle and RLATT do away with the listen-only periods at the start of each interval, which speeds up network transmission at lower network densities, the network convergence times for Drizzle and RLATT are shorter than those for the traditional trickle timer at lower network densities. For Drizzle and RLATT, this ultimately leads to faster network convergence times. Because RLATT

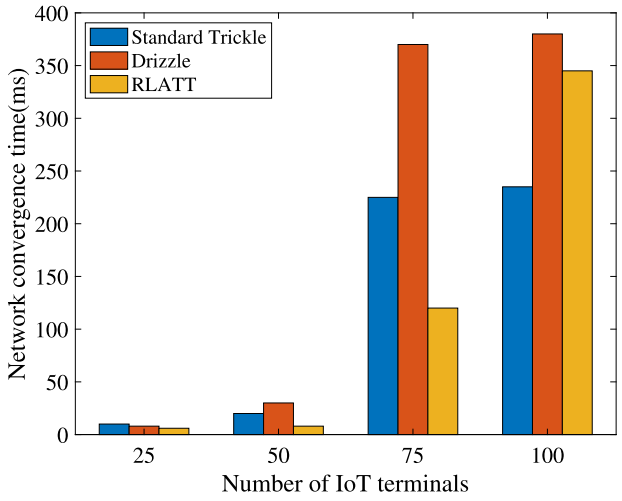


Fig. 8. Convergence times of three trickle timer variants with different numbers of IoT terminals.

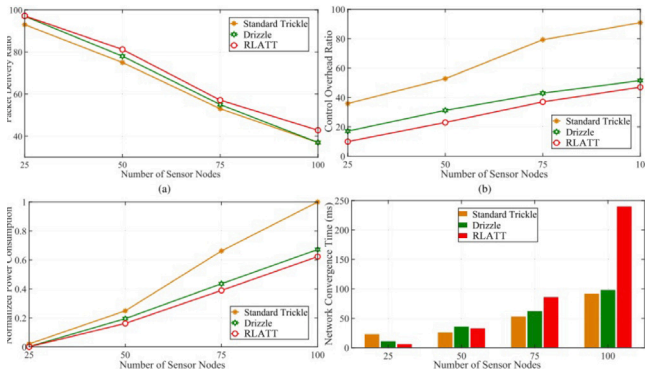


Fig. 9. Variable data rate scenario: (a) packet delivery rate PDR; (b) control overhead ratio; (c) power consumption; (d) total network convergence time (ms).

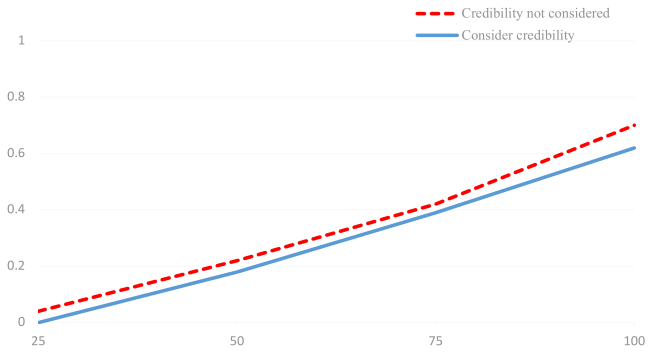


Fig. 10. The influence of credibility on power consumption under different numbers of IoT terminals.

provides greater DIO transmission rates to nodes that received inconsistent DIO signals in the previous interval, RLATT achieves faster network convergence times than Drizzle.

However, when the network size grows, Drizzle and RLATT start to encounter longer network convergence times since they do away with listen-only cycles, which increases the radio duty cycle algorithm's job in order to prevent congestion and contention. The minor delay in network convergence time at larger network densities, however, is negligible in comparison to the other performance improvements examined above.

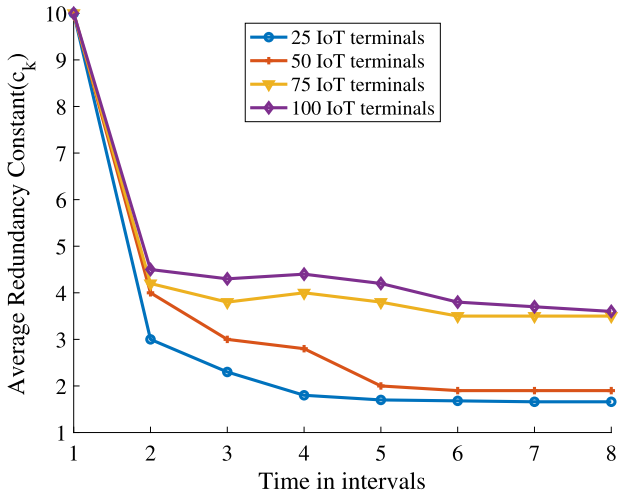


Fig. 11. Variation in the choice of adaptive redundancy constant c_k values for different network densities.

4.2. Analysis in variable data rate scenarios

IoT endpoints frequently demand varying data rates in IoT networks in order to accommodate hybrid applications. Therefore, we did simulations at various data rates, where each IoT terminal selects from 1 to 60 and chooses it as its data rate, in order to model the behaviour of these three trickle timer variants in this variable data rate environment. For regular trickling timer, Drizzle, and RLATT at varying data rates, Fig. 9 displays the PDR, control overhead ratio, normalized total network power consumption, and network convergence time. In comparison to existing trickle timers, simulation findings demonstrate that this approach has higher PDR and lower control overhead. The battery life of IoT terminals is ultimately extended by this lower control overhead ratio's impact on power usage. Due to its capacity to learn the best course of action for DIO transmission and suppression over time by exploiting its learning capabilities, it can achieve greater PDR and a lower control overhead ratio in an environment with changing data rates. Additionally, during the algorithm exploration phase, RLATT adaptively selects the value of its redundancy constant k, which aids in adjusting the DIO transmission and suppression rate. Although, as was already indicated, for bigger network sizes, the RLATT's network convergence time marginally increases. In contrast to a minor increase in network convergence time, other performance improvements are significant in contexts with variable data rates.

As shown in Fig. 10, the influence of credibility on power consumption under different IoT terminals under the circumstances of considering credibility and not considering credibility, it can be seen that power consumption increases with the increase of the number of IoT terminals, But considering reliability can help save energy by reducing unnecessary DIO transfers compared to the case without considering reliability, and ultimately allow it to run longer.

4.3. Analysis of adaptive redundancy constant c_k selection under different network densities

Fig. 11 illustrates how the adaptive redundancy constant c_k value varies for various network densities in the suggested algorithm. The c_k value is set to the same value as k during network initialization. k is currently set to 10 in this instance. However, the suggested approach adaptively chooses the ck value as the network changes over time. The local network density serves as the basis for this adaptive selection. In order to account for this, the suggested method chooses a higher c_k value for dense networks and a lower c_k value for networks with lower densities.

4.4. Computational and memory complexity analysis

Determining the best state and action pairs in the RL paradigm requires studying the entire state and action space by exploring every available option. The number of actions and states is finite and unobservable. Each IoT terminal maintains a Q value for each possible state and action pair. The number of states and actions significantly affects the computational complexity of the proposed algorithm. The computational complexity of this algorithm is $O(s)(a)$, where s represents the number of states and a represents the number of actions available in each state. The number of state and action pairs on a specific IoT terminal is independent of the network size as the Q-learning strategy is applied to the local IoT terminal.

ROM and RAM are two key resources of IoT terminals. The proposed RLATT algorithm requires additional memory to satisfy its RL-based implementation. Therefore, it requires an additional 3021 and 2431 bytes of ROM, respectively, compared to the standard trickle timer and the Drizzle algorithm. RLATT maintains more variables to satisfy Q-value tables, exploration and learning rates, and discounting factors, and maintains the average number of DIO transmissions received in past intervals for efficient operation. Therefore, its operation requires only 48 and 36 bytes of additional memory compared to the standard trickle timer and Drizzle algorithms.

5. Conclusion

A large number of IoT terminals access the network and need to formulate efficient, adaptive, and intelligent routing strategies to ensure the reliable operation of the IoT. After analysing the fair broadcast suppression mechanism and Drizzle algorithm used in the traditional RPL protocol, based on this, according to the characteristics of RL and trickle timer, this paper proposes an intelligent adaptive trickle timer algorithm, RLATT based on reinforcement learning. The algorithm is then used with the typical trickle timer and Drizzle algorithms on the Cooja3.0 emulator running on the Contiki operating system and compared in a number of areas, including PDR, power usage, network convergence time, and the network's overall control overhead ratio. Experiments show that RLATT can be applied to heterogeneous perception networks and can effectively improve the routing efficiency of perception layer networks. But the current experiment of this algorithm is only carried out on the emulator. In future work, we will apply this algorithm to the real network environment and further optimize it.

Declaration of competing interest

The authors declare no conflict of interest.

Data availability

Data will be made available on request.

Acknowledgement

The author would like to thank University of Salford for their support.

References

[1] M. Poornima, H. Vimala, J. Shreyas, Holistic survey on energy aware routing techniques for IoT applications, J. Netw. Comput. Appl. 213 (2023) 103584.
[2] A. Churcher, R. Ullah, J. Ahmad, S. Ur Rehman, F. Masood, M. Gogate, F. Alqahtani, B. Nour, W.J. Buchanan, An experimental analysis of attack classification using machine learning in IoT networks, Sensors 21 (2) (2021) 446.
[3] C. Liu, T. Wu, Z. Li, T. Ma, J. Huang, Robust online tensor completion for IoT streaming data recovery, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[4] X. Zhou, X. Yang, J. Ma, I. Kevin, K. Wang, Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT, Mob. Inf. Syst. 2021 (8) (2021) 1–9.
[5] N. Sun, T. Li, G. Song, H. Xia, Network security technology of intelligent information terminal based on mobile internet of things, Mob. Inf. Syst. 2021 (8) (2021) 1–9.
[6] F. Wang, D. Jiang, H. Wen, H. Song, Adaboost-based security level classification of mobile intelligent terminals, J. Supercomput. 75 (11) (2019) 7460–7478.
[7] Y. Chen, W. Liu, MAC layer energy consumption and routing protocol optimization algorithm for mobile ad hoc networks, Complexity 2021 (24) (2021) 1–12.
[8] O.U. Rehman, S. Yang, S. Khan, S.U. Rehman, A quantum particle swarm optimizer with enhanced strategy for global optimization of electromagnetic devices, IEEE Trans. Magn. 55 (8) (2019) 1–4.
[9] S.U. Rehman, S. Tu, O.U. Rehman, Y. Huang, C.M.S. Magurawalage, C.-C. Chang, Optimization of CNN through novel training strategy for visual classification problems, Entropy 20 (4) (2018) 290.
[10] O.U. Rehman, S. Tu, S.U. Rehman, S. Khan, S. Yang, Design optimization of electromagnetic devices using an improved quantum inspired particle swarm optimizer, Appl. Comput. Electromagn. Soc. J. (2018) 951–956.
[11] Q. Ni, J. Guo, W. Wu, H. Wang, Influence-based community partition with sandwich method for social networks, IEEE Trans. Comput. Soc. Syst. 10 (2) (2022) 819–830.
[12] Y. Yao, F. Shu, Z. Li, X. Cheng, L. Wu, Secure transmission scheme based on joint radar and communication in mobile vehicular networks, IEEE Trans. Intell. Transp. Syst. (2023).
[13] H. Wang, W. Sun, D. Jiang, R. Qu, A MTPA and flux-weakening curve identification method based on physics-informed network without calibration, IEEE Trans. Power Electron. (2023).
[14] X. Zhou, L. Zhang, SA-FPN: An effective feature pyramid network for crowded human detection, Appl. Intell. 52 (11) (2022) 12556–12568.
[15] S. Ghorbani, Z. Yang, P.B. Godfrey, Y. Ganjali, A. Firoozshahian, Drill: Micro load balancing for low-latency data center networks, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017, pp. 225–238.
[16] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C.L. Lim, R. Soulé, {Semi-oblivious} traffic engineering: The road not taken, in: 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 18, 2018, pp. 157–170.
[17] G. Kaur, P. Chanak, M. Bhattacharya, Energy-efficient intelligent routing scheme for IoT-enabled WSNs, IEEE Internet Things J. 8 (14) (2021) 11440–11449.
[18] L. Zhang, C. Sun, G. Cai, L.H. Koh, Charging and discharging optimization strategy for electric vehicles considering elasticity demand response, eTransportation 18 (2023) 100262.
[19] J. Zhang, C. Zhu, L. Zheng, K. Xu, ROSEFusion: random optimization for online dense reconstruction under fast camera motion, ACM Trans. Graph. (TOG) 40 (4) (2021) 1–17.
[20] J. Xu, Z. Yang, Z. Wang, J. Li, X. Zhang, Flexible sensing enabled packaging performance optimization system (FS-PPOS) for lamb loss reduction control in E-commerce supply chain, Food Control 145 (2023) 109394.
[21] W. Wang, J. Xu, W. Zhang, B. Glamuzina, X. Zhang, Optimization and validation of the knowledge-based traceability system for quality control in fish waterless live transportation, Food Control 122 (2021) 107809.
[22] X. Zhang, S. Fang, Y. Shen, X. Yuan, Z. Lu, Hierarchical velocity optimization for connected automated vehicles with cellular vehicle-to-everything communication at continuous signalized intersections, IEEE Trans. Intell. Transp. Syst. (2023).
[23] V.K. Mutombo, S. Lee, J. Lee, J. Hong, EER-RL: energy-efficient routing based on reinforcement learning, Mob. Inf. Syst. 2021 (2021).
[24] D.K. Dake, J.D. Gadze, G.S. Klogo, H. Nunoo-Mensah, Multi-agent reinforcement learning framework in SDN-IoT for transient load detection and prevention, Technologies 9 (3) (2021) 44.
[25] D.K. Sharma, J.J. Rodrigues, V. Vashishth, A. Khanna, A. Chhabra, RLProph: a dynamic programming based reinforcement learning approach for optimal routing in opportunistic IoT networks, Wirel. Netw. 26 (6) (2020) 4319–4338.
[26] S. ur Rehman, S. Tu, M. Waqas, Y. Huang, O. ur Rehman, B. Ahmad, S. Ahmad, Unsupervised pre-trained filter learning approach for efficient convolution neural network, Neurocomputing 365 (2019) 171–190.
[27] Y. Yao, J. Zhao, Z. Li, X. Cheng, L. Wu, Jamming and eavesdropping defense scheme based on deep reinforcement learning in autonomous vehicle networks, IEEE Trans. Inf. Forensics Secur. 18 (2023) 1211–1224.
[28] Q. Fu, Z. Li, Z. Ding, J. Chen, J. Luo, Y. Wang, Y. Lu, ED-DQN: an event-driven deep reinforcement learning control method for multi-zone residential buildings, Build. Environ. (2023) 110546.
[29] Z. Peng, J. Hu, K. Shi, R. Luo, R. Huang, B.K. Ghosh, J. Huang, A novel optimal bipartite consensus control scheme for unknown multi-agent systems via model-free reinforcement learning, Appl. Math. Comput. 369 (2020) 124821.
[30] D. Li, K.D. Ortegas, M. White, Exploring the computational effects of advanced deep neural networks on logical and activity learning for enhanced thinking skills, Systems 11 (7) (2023) 319.
[31] Y. Zheng, X. Lv, L. Qian, X. Liu, An optimal bp neural network track prediction method based on a ga–aco hybrid algorithm, J. Mar. Sci. Eng. 10 (10) (2022) 1399.

[32] H. Liu, H. Yuan, J. Hou, R. Hamzaoui, W. Gao, Pufa-gan: a frequency-aware generative adversarial network for 3d point cloud upsampling, IEEE Trans. Image Process. 31 (2022) 7389–7402.
[33] F. Qiao, Z. Li, Y. Kong, A privacy-aware and incremental defense method against GAN-based poisoning attack, IEEE Trans. Comput. Social Syst. (2023).
[34] Y. Liu, G. Li, L. Lin, Cross-modal causal relational reasoning for event-level visual question answering, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[35] S. Lu, M. Liu, L. Yin, Z. Yin, X. Liu, W. Zheng, The multi-modal fusion in visual question answering: a review of attention mechanisms, PeerJ Comput. Sci. 9 (2023) e1400.
[36] S. Jiang, C. Zhao, Y. Zhu, C. Wang, Y. Du, et al., A practical and economical ultra-wideband base station placement approach for indoor autonomous driving systems, J. Adv. Transp. 2022 (2022).
[37] X. Liang, Z. Huang, S. Yang, L. Qiu, Device-free motion & trajectory detection via RFID, ACM Trans. Embed. Comput. Syst. (TECS) 17 (4) (2018) 1–27.
[38] B. Dai, Y. Cao, Z. Wu, Z. Dai, R. Yao, Y. Xu, Routing optimization meets machine intelligence: A perspective for the future network, Neurocomputing 459 (2021) 44–58.
[39] H.B. Salameh, S. Otoum, M. Aloqaily, R. Derbas, I. Al Ridhawi, Y. Jararweh, Intelligent jamming-aware routing in multi-hop IoT-based opportunistic cognitive radio networks, Ad Hoc Netw. 98 (2020) 102035.
[40] K. Cao, B. Wang, H. Ding, L. Lv, R. Dong, T. Cheng, F. Gong, Improving physical layer security of uplink noma via energy harvesting jammers, IEEE Trans. Inf. Forensics Secur. 16 (2020) 786–799.
[41] G. Liu, A Q-Learning-based distributed routing protocol for frequency-switchable magnetic induction-based wireless underground sensor networks, Future Gener. Comput. Syst. 139 (2023) 253–266.
[42] P. Cong, Y. Zhang, Z. Liu, T. Baker, H. Tawfik, W. Wang, K. Xu, R. Li, F. Li, A deep reinforcement learning-based multi-optimality routing scheme for dynamic IoT networks, Comput. Netw. 192 (2021) 108057.
[43] M.A. Taherkhani, R. Kawaguchi, N. Shirmohammad, M. Sato, BlueParking: An IoT based parking reservation service for smart cities, in: Proceedings of the Second International Conference on IoT in Urban Space, 2016, pp. 86–88.
[44] S.U. Rehman, M. Waqas, S. Tu, A. Koubaa, O. ur Rehman, J. Ahmad, M. Hanif, Z. Han, Deep Learning Techniques for Future Intelligent Cross-Media Retrieval, Tech. Rep., CISTER-Research Centre in Realtime and Embedded Computing Systems, 2020.
[45] S. Rehman, S. Tu, Y. Huang, G. Liu, et al., CSFL: A novel unsupervised convolution neural network approach for visual pattern classification, AI Commun. 30 (5) (2017) 311–324.
[46] S. Lu, Y. Ding, M. Liu, Z. Yin, L. Yin, W. Zheng, Multiscale feature extraction and fusion of image and text in VQA, Int. J. Comput. Intell. Syst. 16 (1) (2023) 54.
[47] X. Liu, S. Wang, S. Lu, Z. Yin, X. Li, L. Yin, J. Tian, W. Zheng, Adapting feature selection algorithms for the classification of chinese texts, Syst. 11 (9) (2023) 483.
[48] Y. Guo, C. Zhang, C. Wang, X. Jia, Towards public verifiable and forward-privacy encrypted search by using blockchain, IEEE Trans. Dependable Secur. Comput. (2022).
[49] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, Tech. Rep., 2012.
[50] N. Accettura, L.A. Grieco, G. Boggia, P. Camarda, Performance analysis of the RPL routing protocol, in: 2011 IEEE International Conference on Mechatronics, IEEE, 2011, pp. 767–772.
[51] B. Ghaleb, A.Y. Al-Dubai, E. Ekonomou, A. Alsarhan, Y. Nasser, L.M. Mackenzie, A. Boukerche, A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1607–1635.
[52] A.K. Idrees, A. Witwit, Energy-efficient load-balanced RPL routing protocol for internet of things networks, Int. J. Internet Technol. Secur. Trans. IJITST (3) (2021) 11.
[53] H. Kharrufa, H. Al-Kashoash, A.H. Kemp, A game theoretic optimization of RPL for mobile internet of things applications, IEEE Sens. J. (2018) 1.
[54] P. Levis, T. Clausen, J. Hui, O. Gnawali, J. Ko, The trickle algorithm, in: Internet Engineering Task Force, RFC6206, 2011, pp. 1–13.
[55] T. Clausen, O. Gnawali, J.G. Ko, J. Hui, The trickle algorithm, 2011, Rfc.
[56] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks, in: Conference on Symposium on Networked Systems Design & Implementation-Volume, 2004.
[57] D. Nabil, T. Djamel, M. Faiza, Trust-based RPL for the internet of things, in: Computers & Communication, 2016.
[58] B. Ghaleb, A.Y. Al-Dubai, E. Ekonomou, I. Romdhani, Y. Nasser, A. Boukerche, A novel adaptive and efficient routing update scheme for low-power lossy networks in IoT, IEEE Internet Things J. 5 (6) (2018) 5177–5189.
[59] Y. Shi, J. Xi, D. Hu, Z. Cai, K. Xu, RayMVSNet++: learning ray-based 1D implicit fields for accurate multi-view stereo, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[60] C. Zhang, L. Zhou, Y. Li, Pareto optimal reconfiguration planning and distributed parallel motion control of mobile modular robots, IEEE Trans. Ind. Electron. (2023).
[61] Y. Zheng, P. Liu, L. Qian, S. Qin, X. Liu, Y. Ma, G. Cheng, Recognition and depth estimation of ships based on binocular stereo vision, J. Mar. Sci. Eng. 10 (8) (2022) 1153.
[62] Z. Xiao, J. Shu, H. Jiang, G. Min, H. Chen, Z. Han, Perception task offloading with collaborative computation for autonomous driving, IEEE J. Sel. Areas Commun. 41 (2) (2022) 457–473.
[63] Q. Liu, H. Yuan, R. Hamzaoui, H. Su, J. Hou, H. Yang, Reduced reference perceptual quality model with application to rate control for video-based point cloud compression, IEEE Trans. Image Process. 30 (2021) 6623–6636.
[64] H. Jiang, M. Wang, P. Zhao, Z. Xiao, S. Dustdar, A utility-aware general framework with quantifiable privacy preservation for destination prediction in LBSs, IEEE/ACM Trans. Netw. 29 (5) (2021) 2228–2241.
[65] R. Ali, N. Shahin, Y.B. Zikria, B.-S. Kim, S.W. Kim, Deep reinforcement learning paradigm for performance optimization of channel observation–based MAC protocols in dense WLANs, IEEE Access 7 (2018) 3500–3511.
[66] Z. Mammeri, Reinforcement learning based routing in networks: Review and classification of approaches, IEEE Access 7 (2019) 55916–55950.
[67] A. Musaddiq, Z. Nain, Y. Ahmad Qadri, R. Ali, S.W. Kim, Reinforcement learning-enabled cross-layer optimization for low-power and lossy networks under heterogeneous traffic patterns, Sensors 20 (15) (2020) 4158.
[68] T. Contiki, The open source operating system for the internet of things, 2016.

Haining Tan is currently a senior consultant of Chinatelecom Shenzhen company, received the Ph.D. degree in enterprise management. He has many years of experience in communication enterprise management. His research direction focuses on Internet of things, edge computing and data security.

Tao Ye received the Ph.D. degree in computer science and technology from Beijing University of Technology. He has long been engaged in the research of information security and trusted computing technology. His research interests include privacy computing, data security and computational intelligence.

Sadaqat ur Rehman (M'18) received his B.Sc. degree from the Department of Computer Systems Engineering, University of Engineering and Technology Peshawar and M.Sc. degrees from the Department of Electrical Engineering, Sarhad University of Science and IT in 2011 and 2014, respectively. Dr. Sadaqat ur Rehman pursued his Ph.D. degree (Sept. 2015–Jun. 2019) with the Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China. He worked as Artificial Intelligence Engineer in Schlumberger (Beijing Geoscience Centre) from November 2019 to June 2020, where he developed different Deep Learning/Machine Learning models for Drilling Dynamics Computation Engine.

He is currently working as Assistant Professor with the Department of Computer Science, University of Salford, UK. He has produced a world leading research activity in the fields of data science, machine learning, intelligent systems (with emphasis on artificial neural networks), semantic multimedia analysis, optimization and affective computing. He has published over 65 papers in international journals and proceedings of international conferences. His research has been highly referenced (about 1200+ citations with an h-index of 20 in Google Scholar). Also, he is co-chair, TPC member and reviewer of prestigious international conferences and journals including, BMLI 2020, Smarttech 2020, CSAE 2018, 2019, 2020, IEEE Access, Neurocomputing, IEEE TCSVT, Scientific Reports – Nature, Soft Computing, Signal Processing.

Email: s.rehman15@salford.ac.uk
Homepage and Publication: https://www.researchgate.net/profile/Sadaqat_Rehman

Obaid Ur Rehman is Professor of Electrical Engineering in Sarhad University of Sciences and IT. He received his M.Sc. degree in Computer Engineering from The University of Liverpool UK, and PhD degree in Electrical Engineering from Zhejiang University. His current research interests are optimization techniques, computer networks, deep learning, genetic algorithms and computational electromagnetics.

Email: Obaid.ee@suit.edu.pk

Homepage and Publication: https://www.researchgate.net/profile/Obaid_Rehman9

Shanshan Tu received the Ph.D. degree from the Computer Science Department, Beijing University of Post and Telecommunication, in 2014. He is currently an Assistant Professor with the Faculty of Information Technology, Beijing University of Technology, China. He was with the Department of Electronic Engineering, Tsinghua University as a Post-Doctoral Researcher from 2014 to 2016. He visited the University of Essex for joint doctoral training from 2013 to 2014. His research interests are in the areas of cloud computing security and Deep learning.

Email: sstu@bjut.edu.cn

Homepage and Publication: https://www.researchgate.net/profile/Shanshan_Tu2

Jawad Ahmad (Member, IEEE) is currently an experienced researcher with more than ten years of cutting-edge research and teaching experience in prestigious institutes, including Edinburgh Napier University, U.K., Glasgow Caledonian University, U.K., Hongik University, South Korea., and HITEC University, Taxila, Pakistan. He has coauthored more than 50 research articles, in international journals and peer-reviewed international conference proceedings. He has taught various courses both at Undergraduate (UG) and Postgraduate (PG) levels during his career. He regularly organizes timely special sessions and workshops for several flagship IEEE conferences. He is an invited reviewer for numerous world-leading high-impact journals (reviewed more than 50 journal articles to date). His research interests include cyber security, multimedia encryption, machine learning, and application of chaos theory in cyber security.

Email:jawadkhattak@ieee.org

Homepage and Publication: https://www.researchgate.net/profile/Jawad_Ahmad_Khattak