Collision-aware Adaptive RPL Trickle Timer on 6TiSCH Network using Q-Learning

ARTICLE INFO

Keywords:
6TiSCH, MSF, RPL, Trickle timer, Wireless sensor network, Q-learning

ABSTRACT

Many latency-sensitive applications have emerged because of the rapid growth of wireless sensor networks (WSNs). 6TiSCH (IPv6 over IEEE802.15.4e time-slotted channel hopping (TSCH) mode) is a standard protocol that provides a minimal scheduling function (MSF) and routing protocol for low-power and lossy networks (RPL). To transmit all types of control messages, MSF allocates only one shared cell on every TSCH slotframe, known as a minimal cell. Because of this condition, the RPL trickle timer algorithm encounters collisions in destination-oriented directed acyclic graph information object (DIO) control packet transmission with other control messages. The increasing number of nodes and the high and imbalanced DIO transmission may cause a collision. As a solution, this study introduces Q-trickle, an intelligent and adaptive trickle timer algorithm based on Q-learning. First, Q-trickle uses Q-learning to determine the best policy for transmitting or suppressing DIO based on the minimal cell collision probability. Second, Q-trickle adaptively selects a redundancy constant value to eliminate unnecessary control message transmissions. It is determined based on the local network density and reset events that indicate an unstable network. Third, Q-trickle promotes a greater DIO transmission opportunity to nodes with less successful transmission and a less stable network. Finally, the simulation results indicate that Q-trickle outperformed the benchmarks. Compared to the original trickle timer, Q-trickle improves the packet delivery ratio and network lifetime by 44% and 35%, respectively. Q-trickle also reduced the joining time by 63% and the collision ratio by 62%.

1. Introduction

Wireless sensor networks (WSNs) are critical for the development of Internet of Things (IoT) networks [25]. WSNs allow low-end devices with limited resources to connect to the internet and use various services. 6TiSCH is a WSN standard protocol that combines IEEE 802.15.4 time-slotted channel hopping (TSCH) with adaptation of IPv6 over low-power wireless personal area networks (6LoWPAN), a routing protocol for low-power and lossy networks (RPL), and 6top minimal scheduling function (MSF) scheduling. The join registrar/coordinator (JRC), which is the sink node, starts the 6TiSCH network formation process by broadcasting network information via an enhanced beacon (EB). A new node attempts to join the network by scanning random channels for the control packet EB. After receiving the EB, the node sends a join request (JRQ) packet to the JRC or a preferred parent. It then responds to a join response (JRS) packet. The node awaits the routing layer control packet, known as the destination-oriented directed acyclic graph information object (DIO), which is provided by the RPL. When a node receives a DIO, it can calculate its rank and join the RPL. Finally, the node can send its own EB frame to allow other nodes to join the network.

The RPL builds a route using control message exchange to construct the upward route and join the downward route. Nodes also periodically broadcast DIO messages to maintain a stable routing topology using the trickle timer algorithm. It is used to control transmissions that maintain energy efficiency. However, the standard trickle timer presents several challenges. First, a high DIO transmission frequency setting may collide with other DIO and control packets. A low transmission frequency may result in a longer network formation and inhibit network performance improvement, such as finding optimal routing. Second, in the case of an imbalanced transmission distribution problem, some nodes may transmit more than others most of the time. Recently, reinforcement learning-based IoT networking algorithms [17] have been used in many cases, such as MAC scheduling, congestion control, and routing [1, 4]. Reinforcement learning is a learning strategy in which an agent interacts with the environment to learn what actions to take and how to map situations and feedback to maximize a long-term objective.

Therefore, this study proposes Q-trickle, an adaptive tickle-timer algorithm based on the Reinforcement learning (RL) algorithm called Q-learning (QL), which considers the network condition, including shared cell congestion, network density, and stability. Q-trickle offers intelligent decision-making for DIO transmission or suppression through TSCH slotframe observation. Moreover, the transmission frequency is adjusted via a dynamic redundancy constant and transmission interval parameter, rather than a fixed parameter, leading to an fair transmission distribution. The main contributions of this study are as follows:

- Q-learning-based DIO transmission and suppression in the RPL trickle timer that avoids control message collision at the 6TiSCH minimal cell.
- Adaptive redundancy constant selection based on neighbor density and network stability considering any reset events of the trickle timer.
- Dynamic transmission interval that maintains a balanced allocation of DIO transmission considering transmission probability and network stability.

In the remainder of this paper, we explain the 6TiSCH protocol in Section 2, followed by RPL routing and trickle

ORCID(s):

timer in Section 3. Next, we review related works in Section 5. An elaboration of the proposed methods and algorithms is presented in Sections 6. We demonstrate extensive experiments to measure the effectiveness of the proposed method in Section 7. Finally, conclusions and future research directions are discussed.

2. 6TiSCH Minimal Scheduling Function

6TiSCH [28] is an IPv6 over IEEE 802.15.4e TSCH networks. The time division multiple access (TDMA) based medium access layer and channel hopping are the two fundamental aspects of the TSCH MAC layer mode. The TDMA mechanism provides a collision-free method for accessing the shared medium, resulting in efficient channel access and increased network dependability and predictability. Time is divided into equal-length time slots in the TSCH. Each time slot is long enough for a pair of nodes to exchange a packet and its (optional) acknowledgment. A slotframe is a collection of time slots that repeats over time. 6TiSCH can be used in low-power IoT applications such as smart industries, buildings, infrastructure, and home applications. As depicted in Figure 1, 6TiSCH adapts 6LoWPAN [34] to bridge IPv6 over the network by using header compression, packet fragmentation, and reassembly process to convert IPv6 packets into IEEE 802.15.4 frames. 6TiSCH uses RPL as a network layer routing protocol, which is discussed in depth in Section 3. 6TiSCH additionally allows dynamic scheduling in which link-layer resources (TSCH schedule cells) are dynamically added or removed to match the communication requirements of the applications. The concept of dynamic scheduling is divided into two parts. First, a scheduling function (SF) is the mechanism that chooses when to add/delete cells and triggers 6top protocol (6P) negotiations. Second, the 6P allows neighbor nodes to negotiate which cells to add/remove to/from their schedule.

The default SF for all control packets, including EB, JRQ, JRS, and DIO, is the MSF [2]. MSF is used to extend the minimum scheduling configuration and add child-parent links based on the traffic load. In the shared slot of a slotframe, MSF uses only a single shared cell, also known as the minimal cell. Figure 2 shows the minimal number of cells in a slotframe. Dedicated cells are the remaining cells in a slotframe starting with time slot one, and they are used for data transmission using a communication schedule set by a SF. Dedicated cells consist of autonomous and negotiated cells. Autonomous cells offer connections to neighbors without requiring control signaling. Negotiated cells are then handled by the 6P protocol, which adds or removes them from the schedule depending on a traffic-based reactive policy.

As all linked nodes compete for the same physical channel to broadcast their control packets on minimal cells, collisions between control packets transmitted by nodes are inevitable [10]. Hence, a higher network density and control packet transmission frequency can increase collisions in the minimal cell. Frequent transmission of control packets can trigger congestion in shared cells. By contrast, infrequent transmission can impact longer network formation, inhibit network QoS improvement (e.g., finding optimal routing), and other network problem resolutions. Thus, it is necessary to control the node's control packet transmission to avoid network congestion.

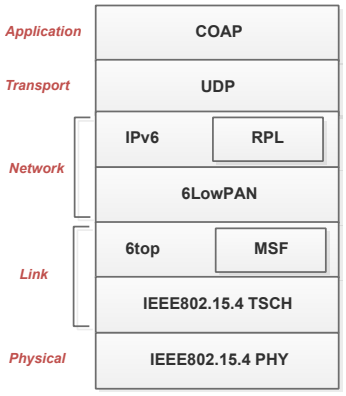

Figure 1: 6TiSCH protocol stack.

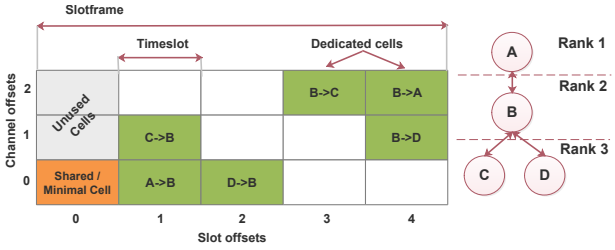

Figure 2: 6TiSCH Minimal Scheduling Function.

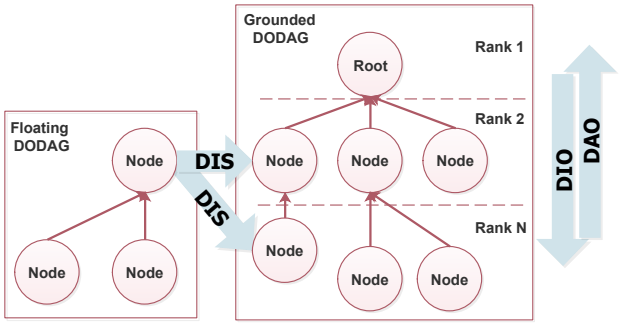

Figure 3: RPL routing.

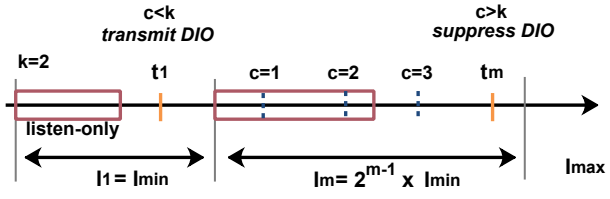

Figure 4: RPL trickle timer.

3. RPL Trickle Timer

The RPL routing protocol is intended to meet the needs of various low-power and lossy networks (LLNs) [32]. RPL builds a route based on routing metrics toward a root node, also known as a sink node or border router, using an objective function (OF). RPL was performed to generate destination-oriented directed acyclic graphs (DODAGs) and determine the optimal path between nodes. Figure 3 illustrates RPL routing. If a DODAG contains a DODAG root, it is grounded. If a DODAG is not grounded, it is floating. The upward route to the leaf node was built using the DODAG information object (DIO). The downward route to the sink node was built using the destination advertisement object (DAO). To join the topology, a new or floating node transmits DODAG information solicitation (DIS). In DODAG, rank denotes a node's level, which is calculated using an OF.

To maintain a stable routing topology, nodes broadcast DIO messages regularly. The transmission of DIO is controlled by a trickle timer to preserve the energy efficiently [20]. It has numerous predefined variables, such as the redundancy constant k, minimum interval size I_{min}, maximum interval size I_{max} determined in Equation 1, current interval I, consistency counter c, and transmission time within interval t. The trickle timer scheme is shown in Figure 4. The steps of the trickle timer are as follows:

1. Initially, select I equal to I_{min}.
2. Beginning at each interval, $c = 0$ is initialized, and t is randomly selected based on Equation 2.
3. If a node receives a consistent transmission, it increments counter c.
4. If a node receives an inconsistent transmission or any other reset events, it resets the trickle timer to Step 1.
5. At time t, if $c < k$ allows its own DIO transmission. Otherwise, DIO is suppressed.
6. On the end of interval, I doubles. If new I is greater than I_{max}, set new I to I_{max} and execute step 2.

The trickle timer has limitations [21], such as a fixed parameter that cannot adapt to network conditions. Then, the distribution is unfair because some nodes may transfer more than others and increase the shared cell collision. The trickle interval can affect network convergence. If it is too long, it can be hardening network convergence. If it is too small, it may not collect sufficient consistent transmission and always decide to send DIO more frequently, which is unnecessary. The trickle timer algorithm defines the frequency of the DIO transmission. When frequent transmission occurs, DIO packets congest in shared cells, resulting in a higher joining time of the nodes. However, infrequent transmission can release more freely shared cells. This results in non-optimal usage and higher joining time of the nodes, as the optimal rate control packet helps network convergence. This indicates that an optimal frequency is needed to maintain network performance. This may be addressed through a dynamic trickle timer setting to control the transmission frequency and reduce control packet collision in the minimal cell based on the network condition.

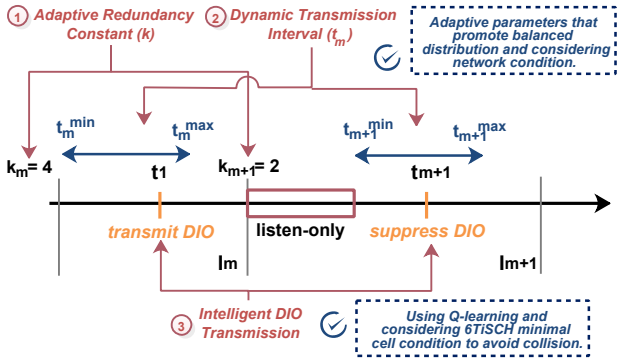

Figure 5: Proposed Q-trickle method.

$$I_m = 2^{m-1} \times I_{\min} \tag{1}$$

$$t_m = \left[\frac{I_m}{2}, I_m\right] \tag{2}$$

4. Reinforcement Learning and Q-Learning

RL [11] is the process of learning to map states and actions to maximize the numerical rewards associated with those interactions. RL comprehends and automates decision-making and goal-driven learning. This entails interactions between agents and their environment. An agent is both a decision-maker and learner. The policy, reward, value function, and environment model are the four components of RL. The policy is a collection of state-action interactions. The policy could be a simple function or lookup table for certain circumstances. A reward reflects the learner's positive and negative experiences. This is the most important cause for any state to change its policy. This demonstrates what is better pragmatically. The value function of a particular state is the total sum of rewards that a learner can accumulate eventually. We are not looking for rewards, but rather the value function. The environmental model is optional in RL. It mimics the behavior of the environment or predicts how the environment will operate, which generally, enables suggestions regarding how the environment will behave.

QL is a popular RL model [31]. In the QL model, an agent has a set of states $s \in S$ and actions $a \in A$. The node obtains a reward $r(s, a)$ by performing a particular action in a particular state. The agent shifts to the next state based on the obtained reward and stores the collected reward in the form of a Q-value for a specific state-action pair $Q(s, a)$. The agent uses this knowledge to choose future actions in a given state. QL is an off-policy algorithm with ε-greedy policy to estimate the return (total discounted future reward) for state-action pairs, assuming a greedy policy was followed. It

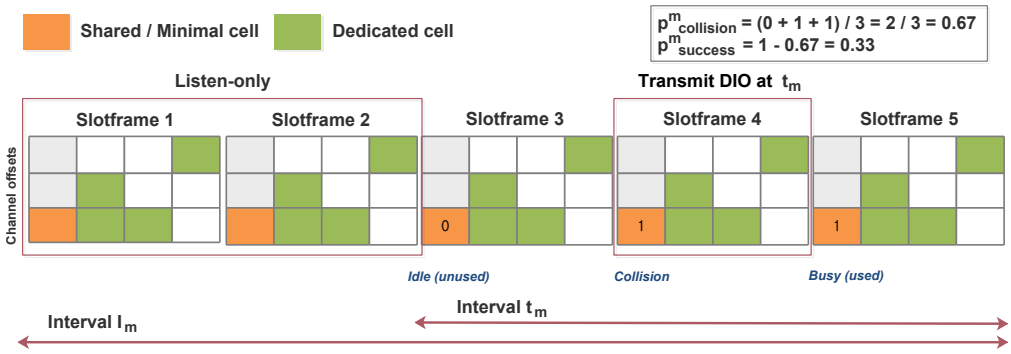

Figure 6: Collision detection on minimal cells over trickle timer.

selects an action based on the probability of ϵ to allow exploration and exploitation action. The greedy optimal action and ϵ-greedy policy control are presented in Equations 3 and 4, respectively.

QL employs a model-free RL approach called the temporal difference (TD). This allows the agent to learn from each action it performs. Instead of updating the agent's knowledge after each episode, the TD updates the agent's experience after each step (action), reaching the goal or end state. The Q-value update is expressed in Equation 6, which is calculated from the old Q-value and learned Q-value with a certain learning rate α between 0 and 1. The learning rate α determines how quickly the new value overrides the previous value. When $\alpha = 0$, the agent ignores existing information and does not learn new values; however, when $\alpha = 1$, the agent is required to analyze only the most recent input and ignore prior knowledge. Equation 5 shows the learning estimate or TD error. It is calculated from the new Q-value given new states, their possible actions (optimal target value), and the current Q-value (predicted value) with a certain discounted reward factor of β between 0 and 1. The value of β influences how important future rewards are to an agent compared to current rewards. The agent evaluates the present reward more vigorously when $\beta = 0$. The agent is then prompted to prioritize long-term benefits over a short-term reward when $\beta = 1$.

$$a^{\text{opt}} = \text{argmax}_a \, Q(s, a) \tag{3}$$

$$a_t = \begin{cases} a^{\text{opt}} & 1 - \epsilon \text{ (exploitation)} \\ a & \epsilon \text{ (exploration)} \end{cases} \tag{4}$$

$$\Delta Q(s, a) = \left\{ r(s, a) + \beta \times \max_a Q\left(s', a'\right) \right\} - Q(s, a) \tag{5}$$

$$Q'(s, a) = (1 - \alpha) \times Q(s, a) + \alpha \times \Delta Q(s, a) \tag{6}$$

5. Related Work

Kalita et al. [13] made full use of the available channels at time slot zero for each slotframe. Instead of employing a single shared cell per slotframe, an allocation policy was used to assign multiple shared cells. In a subsequent study [14], the proposed method dynamically modified the priority of control packets to provide sufficient routing information throughout the network. It also allowed nodes with urgent packets to transmit them immediately. Vallati et al. [27] dynamically raised the number of shared cells in every slotframe, based on the number of control packets generated in the network. However, because the additional shared cells are introduced in separate time slots of a slotframe, these additional shared slots affect the default data transmission schedule. In [9, 15], MSF was modified to suit a particular case problem, namely heavy traffic loads on industry and efficient routing for mobile nodes. Autonomous scheduling like Alice and Orchestra [16, 5] use MAC address-based hash function. Separated slotframes are used for EB in TSCH, control messages in RPL, and independent data frames. As a result, the shared cells were eliminated. Their issue is that when traffic rates are high or network size increases, performance declines drastically because of static allocation. However, MSF does not experience significant changes owing to its flexibility to adapt to traffic situations. The length of the slotframe has a significant impact on autonomous performance. With autonomous scheduling, the number of packets buffered in the local queue is usually close to the maximum value. It exhibits a large end-to-end delay compared with MSF, where the queue is always under control [26].

Several techniques have been proposed for improving trickle timer efficiency. By enhancing the suppression mechanism, I-trickle [8] eliminates the load-balancing issue. At the end of the current interval, it resets the redundancy to zero, which lowers the amount of energy used. By establishing listen-only intervals based on the number of neighbors, E-trickle [33] solves the load-balancing problem. It uses less energy and has a faster convergence time. It has a significant traffic control overhead to keep track of the number of neighbors. FL-trickle [18] offers a modified trickle timer technique in which the minimum interval I_{min} is set to a greater

value to reduce overhead. To reduce the transmit latency of the DIO, it fixes the transmission time t at $I/2$ rather than choosing it at random. A larger double interval reduces traffic control messages and saves more energy. EAAT [23] suggested an energy-aware adaptive trickle timer method. This method was designed to regulate DIO transmission based on residual energy (RE) and future energy (FE). According to the node energy status, it modifies the redundancy constant k. It reduce energy consumption, convergence time, and overhead. Multi-K [3] calculates the redundancy constants at each node locally based on the number of neighbors. Trickle-D [30] modifies the redundancy constant based on Jain's fairness index to increase global fairness, ensuring that all nodes have the same number of transmissions. Trickle-Plus [6] tweaks the trickle by introducing a shift factor that oversteps the calculated number of double intervals. Drizzle [7] fulfills DIO transmission fairness by allocating nodes to various DIO transmission probabilities based on their transmission history. The variable s records the total number of DIO transmissions made by a node in the previous intervals. A node with a larger number of previous DIO broadcasts has a lower probability of DIO transmissions in the following intervals, and vice versa. RIATA [24] uses QL to determine DIO transmission, with inconsistent DIO reception as a reward. It provides nodes that have received an inconsistent control message in the past intervals a higher probability of transmitting control messages, and it chooses an adaptive redundancy constant value to avoid unnecessary control message transmissions. ACPB [12] modifies the trickle timer and proposes a slotframe window (SW)-based adaptive scheme that restricts nodes from transmitting their control packets frequently and reduces congestion further. When it receives the DIS, it resets and continues to the last trickle state. However, it does not consider other reset events and does not explore RL to optimize the network process. However, all of these works, except ACPB, do not study the congestion in a shared cell correlated with the trickle timer in the 6TiSCH RPL part.

With the aforementioned open issues on optimizing the trickle timer and minimizing cell collision, Q-trickle is proposed by employing QL to support the decision-making of DIO transmission in a trickle timer considering minimal cell congestion. Q-trickle also provides an adaptive parameter that provides a balanced transmission distribution among nodes considering network density and stability. Finally, a comparison between prior works and our proposed method is presented in Table 1.

6. Proposed Q-trickle Algorithm

The proposed Q-trickle algorithm, an adaptive RPL trickle timer using QL, is shown in Figure 5. The method has three improvement features compared to the baseline trickle timer algorithm. First, intelligent DIO transmission and suppression decisions using QL consider minimal cell conditions to avoid transmission collisions. Second, an adaptive redundancy constant that takes into account local network density and stability to reinforce a balanced distribution. Third, the dynamic transmission interval based on the transmission probabilities and network conditions to maintain transmission priority among nodes. The Q-trickle flowchart is illustrated in Figure 7, and the related notations are provided in Table 2.

6.1. Intelligent DIO Transmission

To improve the performance of the trickle timer algorithm, the proposed algorithm employs QL. Each node acts as an agent that maintains trickle state m as QL state s, action $a = 0$ as DIO suppression, and action $a = 1$ as DIO transmission. The states, actions, and rewards are described in Equation 7. The reward is calculated from the probability of idle and successful transmission of minimal cells within the transmission interval expressed in Equation 8. The probability is the inversion of the collision probability, which is expressed in Equation 9. The number of minimal cells is determined using Equation 10. An illustration of the collision or success probability is shown in Figure 6. There are three slotframes, and each slotframe maintains one minimal cell. The transmission timer t is established in the second slotframe. Each slotframe minimal cell is assigned a value of 1 for collided transmission or busy conditions and 0 for successful transmission or idle conditions. These flags will be used to calculate the collided transmission probability, and subsequently the success transmission probability.

The proposed algorithm chooses a random number between zero and one to decide between exploration and exploitation. The DIO will be broadcast, if the number of received consistent DIO messages c_m are less than k_m during the exploration phase. DIO transmission is inhibited, if the number of received consistent DIO messages c_m are larger than k_m. During the exploitation phase, the optimal action is selected based on the largest accumulated reward for a certain state-action pair measured by the Q-value across the previous periods.

$$\begin{aligned} A &= \{0, 1\} \\ S &= \{1, \ldots, m\} \\ r(s, a) &= p^m_{success} \end{aligned} \tag{7}$$

$$p^m_{success} = 1 - p^m_{collision} \tag{8}$$

$$p^m_{collision} = \frac{1}{N^m_{cell}} \times \sum_{n=0}^{N^m_{cell}-1} T_n \quad \begin{matrix} T_n = 0 \text{ if idle/transmitted.} \\ T_n = 1 \text{ if busy/collision.} \end{matrix} \tag{9}$$

$$N^m_{cell} = \frac{t^{max}_m - t^{min}_m}{L \times E} \tag{10}$$

6.2. Adaptive Redundancy Constant k_m

The performance of the trickle timer is significantly affected by the selection of the redundancy constant. For example, consider a network in which the first node has $k = 1$, and the second has $k = 2$. The second node can end up transmitting every interval. Hence, the danger of mismatched k

Table 1
Comparison with related works.

Algorithm	Features	Adaptive redundancy constant	Dynamic interval	Intelligent DIO transmission	Collision detection	Reset event handling
Original [20]	Original Trickle Timer	✗	✗	✗	✗	✗
I-trickle [8]	Reset the counter c at random time t	✗	✗	✗	✗	✗
E-trickle [33]	Setting listen-only interval based on the number of neighbors	✗	✓	✗	✗	✗
FL-trickle [18]	Fixed time t and set high I_{min}	✗	✗	✗	✗	✗
EAAT [23]	Set k based on the node energy condition	✓	✗	✗	✗	✗
Multi-K [3]	Set k based on the number of neighbors	✓	✗	✗	✗	✗
Trickle-D [30]	Set k based on Jain fairness index	✓	✗	✗	✗	✗
Trickle-Plus [6]	Use the shift factor to manage interval doubling	✗	✓	✗	✗	✗
Drizzle [7]	Adaptive t interval and k based on transmitted DIO	✓	✓	✗	✗	✗
RIATA [24]	Adaptive trickle timer parameter using QL	✓	✓	✓	✗	✗
ACPB [12]	Dynamic trickle timer setting for faster network bootstrapping	✓	✓	✗	✓	Partial
Q-trickle	Intelligent trickle timer using QL and shared cell condition	✓	✓	✓	✓	✓

values is an uneven transmission load, which can deplete the energy of some nodes in a low-power network. The proposed Q-trickle algorithm adjusts its redundancy constant k_m dynamically, which is set differently for each trickle state interval m. Q-trickle considers network density to be related to the number of neighbors, and the network stability is related to node stability from receiving any trickle timer's reset event. Reset events make a trickle state, and the parameter returns to its initial value. It allows frequent transmission to handle unstable or inconsistent conditions. Several reset events are established. First, the preferred parent changes when a node finds a better parent and constructs a new rank. Second, global and local repair requires a node to rejoin the DODAG. Third, a DIS message is received from a node that wants to join the DODAG. Fourth, inconsistent DIO, such as infinite rank or different DODAG versions, is obtained. Last, a looping packet is detected when a node receives an upward packet from its parent. Thus, reset events are counted and formulated into network reset and stable probabilities, which are expressed in Equations 11 and 12.

Finally, the redundancy constant is expressed using Equation 13. The minimum k_m is set to 1, and the maximum k_m to 10 as specified by the RPL standard. An illustration of the k_m calculation is presented in Table 6. When the network is unstable, as shown by the p_{reset} tending to 1, the k_m value increases. A higher value of k_m allows more DIO transmission in a network, and vice versa. It allows the quick resolution of unstable or inconsistent conditions around the node.

$$p_{reset} = \frac{N_{reset}}{N_{states}} \tag{11}$$

$$p_{stable} = 1 - p_{reset} \tag{12}$$

$$k = 1 + \lceil \min(N_{nbr}, k_{max}) \times p_{reset} \rceil - 1 \tag{13}$$

6.3. Dynamic Transmission Interval t_m

The proposed algorithm assigns higher DIO transmission probabilities to nodes with more DIO suppression, the possibility of DIO transmission collision, and network stability, as determined by the frequency of the trickle reset. They are calculated using Equations 8, 12 and 14. Q-trickle algorithm dynamically adjusts its transmission interval t_m, which is set differently for each trickle state interval m. The transmission interval t_m is expressed using Equation 15. The interval lower bound t_m^{min} ranges from 0 to $I_m/2$, and the interval upper bound t_m^{max} ranges from $I_m/2$ to I_m. The trickle timer selects a random time within those bound to execute DIO transmission or suppression. Therefore, the maximum value of the bound is based on the original trickle timer interval in Equation 2. We have modified it to vary the transmission probability based on the network conditions. An illustration of the t_m calculation is presented in Table 4. The three proposed methods have been added to the original trickle timer algorithm, which is outlined in Algorithm 1.

$$p_{transmit} = \frac{DIO_{transmit}}{N_{states}} \tag{14}$$

$$t_m = \left[(p_{transmit} \times p_{success}^{m-1}) \times \frac{I_m}{2}, \frac{I_m}{2} + (p_{stable} \times \frac{I_m}{2}) \right] \tag{15}$$

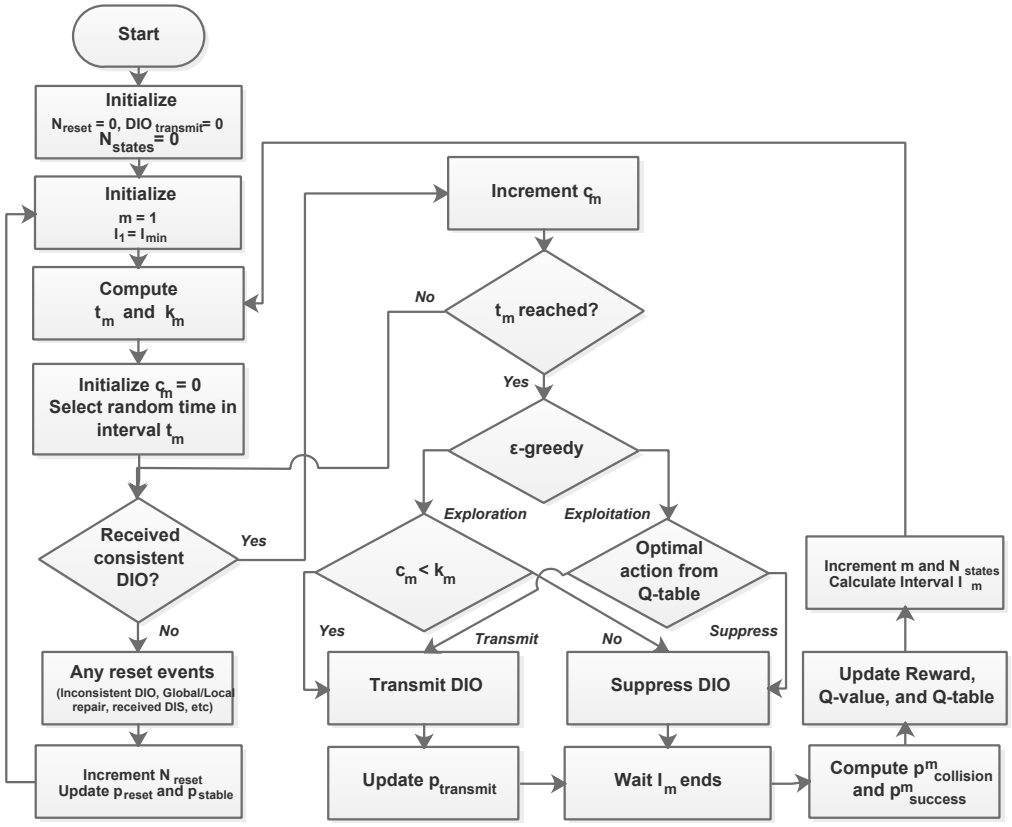

Figure 7: Q-trickle flowchart.

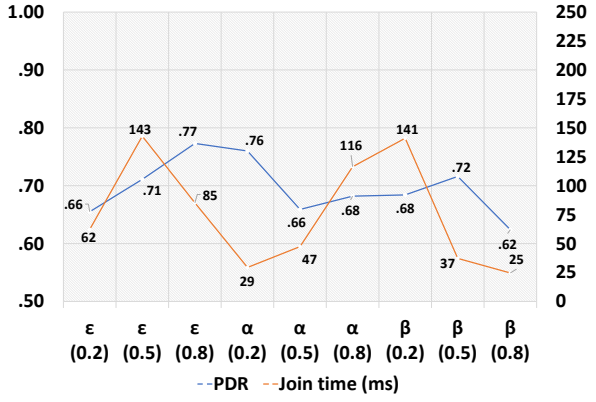

Figure 8: PDR and average join time over various QL parameters.

7. Performance Evaluation

We conducted experiments using the 6TiSCH simulator [22] using parameters presented in Table 3. The 6TiSCH simulator uses the piston–hack loss model to construct node link quality [19]. Q-trickle was compared to related benchmark algorithms, namely RIATA [24] and ACPB [12]. RIATA was selected because of a similar approach in proposing RL for trickle timers, while ACPB considers 6TiSCH minimal cell conditions for improving the trickle timer. Several evaluation metrics were used in this study. First, the average joining time, in milliseconds, of the nodes to the RPL DODAG. Second, the packet delivery ratio (PDR) of the end-to-end packet reception ratio is expressed in Equation 16. Third, the average network lifetime in years is described in Equation 17 based on the 6TiSCH energy model [29]. Finally, the transmission collision ratio is formed in Equation 18. For the experiments, suitable QL ε, α, and β rates were determined for Q-trickle, and then the benchmarks and the proposed method were evaluated over several nodes and minimum intervals.

$$PDR = \frac{\text{Number of received packets}}{\text{Number of sent packets}} \tag{16}$$

$$\text{Lifetime} = \frac{\text{Battery}_{Capacity} \times (L \times E)}{Q_{\text{slotframe}} * 3600 * 24 * 365} \tag{17}$$

$$p^{collision}_{transmit} = \frac{DIO^{collision}_{transmit}}{DIO_{transmit}} \tag{18}$$

7.1. Q-Learning Parameter Selection

In this experiment, the optimal QL variables of exploration rate ε, learning rate α, and discount factor β were selected based on the PDR and network-joining time. While

Table 2
List of notations.

Notation	Description
I_{min}	Minimum trickle timer interval
I_{max}	Maximum trickle timer interval
I_m	Trickle timer interval at state m
c_m	Counter of consistent DIO at state m
k_m	Redundancy constant at state m
m	Trickle timer state
N^m_{cell}	Number of cells at state m
N_{nbr}	Number of neighbors
N_{reset}	Number of resets
N_{states}	Total states that has been passed
L	Slotframe length in timeslots
E	Timeslot duration in milliseconds
$s \in S$	States in QL
$a \in A$	Action in QL
$r(s,a)$	Reward of state s and action a
$Q(s,a)$	Q-values of state s and action a
t_m	Interval of DIO transmission
$DIO_{suppress}$	Number of DIO suppression
$DIO_{transmit}$	Number of DIO transmission
$DIO^{collision}_{transmit}$	Number of collided DIO transmission
$DIO^{success}_{transmit}$	Number of success DIO transmission

Table 3
List of parameters.

Parameter	Value
Simulation platform	6TiSCH Simulator
Protocol and scheduling	6TiSCH, MSF
RPL OF	MRHOF
Topology and loss model	Random (Pister-Hack)
Data rate	1 pkt/s
Total area	2000 m^2
Maximum redundancy constant	10
Channel offsets	16
Slotframe length	101 slots
Time slot duration	10 ms
Maximum trickle states	8
Each experiment duration	30 min
QL ε, α, and β rate	0.2, 0.5, 0.8
Number of nodes	10, 50, 100
Minimum interval	5 s, 10 s, 20 s

evaluating each parameter, the values of the other parameters were set as 0.5. For example, when evaluating $\varepsilon = 0.2$, we set $\alpha = 0.5$ and $\beta = 0.5$. The number of nodes was set as 50 and the minimum interval as 5 s. The results are shown in Figure 8. Higher exploration rates resulted in a higher PDR and a shorter joining time because the agent spends more time examining the available possibilities in the environment to obtain the optimal action. The agent does not have enough time to examine all accessible network choices when the exploration rate is low. The best results were obtained when the exploration rate was set to 0.8. The figure depicts how ΔQ converged faster when $\alpha = 0.2$ and $\beta = 0.5$. The convergence of ΔQ indicates that the agent has learned its environment and can exploit optimal actions in the future. Thus, the PDR and joining time were better than those of the others. Finally, we selected $\varepsilon = 0.8$, $\alpha = 0.2$, and $\beta = 0.5$, for use in the next evaluation. They performed the best, with an average joining time of 50.3 s as the and a PDR of 0.749.

7.2. Node Join Time

In this experiment the average node joining time of the benchmarks over a different number of nodes and minimum interval was evaluated. The results are shown in Figure 9. An increasing number of nodes resulted in an increasing joining time. In the beginning, the difference between joining times was not significant between RIATA, ACPB, and Q-trickle. The minimum interval is a trivial parameter. When we set a low I_{min} of 5 s, the network may fall into bursty traffic. If I_{min} is as high as 20 s, the network suffers from a longer network formation, including joining the RPL route. Hence, a moderate value, such as I_{min} equal to 10 s is suitable for the network. It may provide an appropriate time that is not too short or long to exchange control messages and construct the network. We observed this pattern when the number of nodes was 50 or 100. The original had the highest joining time compared to the others. The original suffers from using static settings and cannot adapt to traffic changes. RIATA cannot handle congestion on the shared cell; thus, it was higher than ACPB and Q-trickle. Q-trickle performed better than ACPB because it intelligently avoids congestion using QL optimization, while ACPB is restricted to broadcasting more than two control packets (one EB frame and one DIO packet that we cannot ensure) within the slotframe window amount of time. It must take care and determine the optimal value for another new parameter, the slotframe window. Finally, Q-trickle provided the best performance overall, with a 63% average improvement in decreasing joining time compared to the original.

7.3. Packet Delivery Ratio

In this experiment, the PDR of the benchmarks over a different number of nodes and minimum intervals was evaluated. The results are shown in Figure 10. The increasing number of nodes decreased the PDR. Initially, the difference between PDRs is not significant between benchmarks. This is because few nodes (ten) have low traffic loads. The increasing value of I_{min} increased the PDR. A higher I_{min} al-

Table 4
Illustration of transmission interval calculation.

I_m	$p_{transmit}$	$p_{success}^{m-1}$	p_{stable}	t_m^{min}	t_m^{max}	listen-only	Description
10 s	0	1	1	0 s	10 s	0 s	Initial state: no listen-only to let sooner transmission and faster RPL formation
10 s	1	1	1	5 s	10 s	5 s	Frequent transmit: listen-only is increased to allow other transmissions
10 s	1	1	0.2	5 s	6 s	5 s	Trickle reset frequently: t_m^{max} is decreased to allow sooner transmission
10 s	0.2	1	1	1s	10 s	1 s	Infrequent transmit: t_m^{min} is decreased to allow more transmission
10 s	1	0.2	1	1 s	10 s	1 s	Transmission often collided: t_m^{min} is reduced to avoid collision

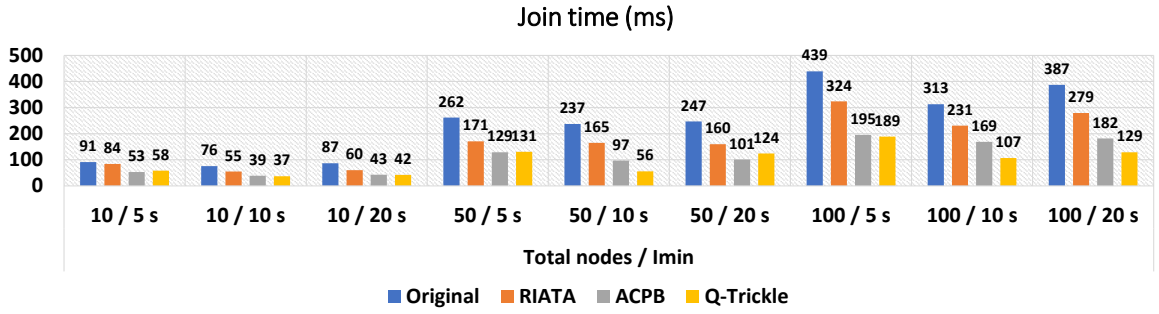

Figure 9: Average join time.

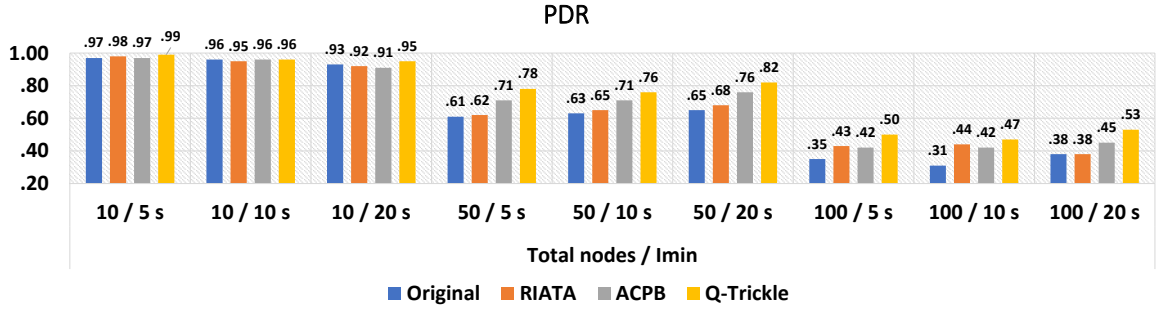

Figure 10: Packet delivery ratio.

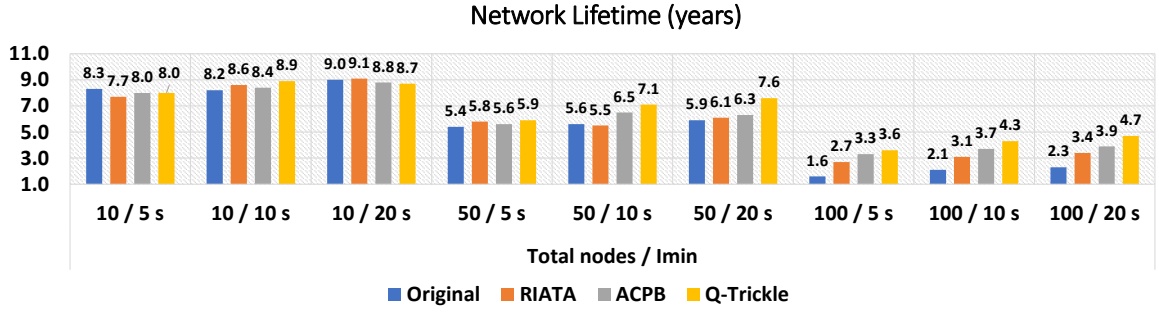

Figure 11: Network lifetime.

Table 5
DIO transmission collision ratio.

Total nodes	I_{min}	$p^{collision}_{transmit}$			
		Original	RIATA	ACPB	Q-trickle
10	5 s	0.08	0.08	0.02	0.05
	10 s	0.05	0.04	0.00	0.01
	20 s	0.01	0.01	0.00	0.00
50	5 s	0.27	0.16	0.14	0.07
	10 s	0.20	0.12	0.08	0.02
	20 s	0.14	0.07	0.00	0.01
100	5 s	0.38	0.32	0.21	0.15
	10 s	0.31	0.27	0.19	0.12
	20 s	0.29	0.24	0.16	0.09

Table 6
Illustration of redundancy constant calculation.

N_{nbr}	6	6	6	6	6	6	6	6	6	6	6
p_{reset}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
k_m	1	2	2	3	3	4	4	5	5	6	6

lows the network to maintain lower traffic loads and less congestion inside the minimal cells for transmitting control messages. The original achieved worse performance because it could not adapt to the network condition. Q-trickle performed better than RIATA and ACPB owing to optimized decision-making on transmitting control packets with collision avoidance and maintaining a balanced transmission distribution. Finally, Q-trickle outperformed the others and maintained an average improvement of 44% in increasing PDR compared to the original.

7.4. Network Lifetime

In this experiment, the network lifetime of the benchmarks over a different number of nodes and minimum intervals was evaluated. The results are shown in Figure 11. The increasing number of nodes decreased the network lifetime. Initially, the difference between the network lifetime was not significant between the benchmarks. This is because few nodes (ten) have low traffic loads. Increasing the value of I_{min} increased network lifetime. A higher I_{min} allows the network to maintain lower traffic loads and less congestion inside the minimal cells for transmitting control messages. The original achieved worse performance because it could not adapt to the network condition. RIATA did not perform well because it does not consider congestion within the TSCH schedule of the control packet. Q-trickle obtained a higher network lifetime than the others owing to optimized intelligent transmission and suppression, which maintains lower energy and efficient transmission because it ensures that the transmission avoids collision. Finally, Q-trickle outperformed the benchmark and achieved an average improvement of 35% in increasing the network lifetime compared to the original.

Algorithm 1 Q-trickle algorithm

function INITILIZATION
  $N_{states} = 0$
  $N_{reset} = 0$
  $DIO_{transmit} = 0$
  TrickleStart()
function TRICKLESTART
  $m = 1$
  $I_1 = I_{min}$
  IntervalBegins()
function INTERVALBEGINS
  $c_m = 0$
  Calculate k_m using Eq. (13)
  Calculate t_m using Eq. (15)
  Select random time within t_m
function TIMERREACHED
  if $rand < \varepsilon$ **then** ▷ Exploration
    if $c_m < k_m$ **then**
      Transmit DIO
      Update $p_{transmit}$ using Eq. (14)
    else
      Suppress DIO
  else ▷ Exploitation
    if $a_{opt} = 1$ **then**
      Transmit DIO
      Update $p_{transmit}$ using Eq. (14)
    else
      Suppress DIO
function INTERVALENDS
  Compute $p^m_{collision}$ and $p^m_{success}$ using Eq. (9, 8)
  Calculate $r(s,a)$, $Q(s,a)$, and $\Delta Q(s,a)$ using Eq. (7, 6, 5)
  Update Q-table
  Increment trickle state m and N_{states}
  if $m > m_{max}$ **then**
    $m = m_{max}$
  Calculate I_m using Eq. (1)
  IntervalBegins()
function RESETTIMER
  Update p_{reset} and p_{stable} using Eq. (8, 11)
  TrickleStart()
function CONSISTENTTRANSMISSION
  $c = c + 1$
function INCONSISTENTTRANSMISSION
  ResetEventCallback()
function RESETEVENTCALLBACK ▷ When received inconsistent DIO, global/local repair, received DIS, etc.
  ResetTimer()

7.5. Collided Transmission Ratio

In this experiment, the collision transmission ratio of the benchmarks over a different number of nodes and minimum intervals was evaluated. The results are presented in Table 5. The increasing number of nodes decreased the PDR. Initially, the difference between the PDRs was not significant

between the benchmarks. This is because few nodes (ten) have low traffic loads. When we set I_{min} equal to 5 s, the network fell into bursty traffic and resulted in a higher collision ratio. However, if I_{min} is set to 10 s or 20 s, the network will have a longer interval and obtain a lower collision probability over the transmission. When the total number of nodes was 10, Q-trickle had a higher collision ratio than ACPB because it requires some time to learn over the network, capturing the optimal action, which reduces the collision ratio. However, when the total number of nodes was increased to 50 and 100, the collision ratio of Q-trickle was lower. Indeed, it requires time to learn the optimal decision to transmit or suppress the DIO. After a while, it will converge and provide better results. Original and RIATA suffer from a high collision ratio because they do not consider control packet collisions over shared cells. Overall, Q-trickle had the best performance in lowering the collision ratio and maintained a 62% average improvement in reducing collision compared to the original.

8. Conclusion

The challenges accompanied by a trickle timer over a 6TiSCH network are collision for control packet transmission at the minimal cell. It led to more prolonged network joining time and lower network performance, such as finding the optimal route. The trickle timer also suffers from an imbalanced transmission distribution owing to the static setting without considering the network condition. Therefore, the Q-trickle algorithm is proposed to avoid the congestion of control messages and reinforce a fair transmission distribution. The proposed method intelligently decides to transmit or suppress DIO via QL, which lowers the probability of shared cell collision. Then, Q-trickle maintains an adaptive redundancy constant based on neighbor density and observed network consistency measured with trickle reset events. Further, Q-trickle develops transmission intervals that balance transmission priority with respect to the probability of successful transmission and network stability. We evaluated Q-trickle using the 6TiSCH simulator and compared it to existing trickle timer algorithms. The results showed that Q-trickle reduces network joining time, lowers collision ratio, increases PDR, and extends network lifetime. In future work, we will conduct more experiments on different environments and scenarios on a real testbed. Subsequently, we will consider using unused cells on a shared cell's channel offset to improve control packet transmission performance.

References

[1] Ali, R., Shahin, N., Zikria, Y.B., Kim, B.S., Kim, S.W., 2018. Deep reinforcement learning paradigm for performance optimization of channel observation–based mac protocols in dense wlans. IEEE Access 7, 3500–3511.

[2] Chang, T., Vucinic, M., Vilajosana, X., Duquennoy, S., Dujovne, D., 2019. 6tisch minimal scheduling function (msf). Internet Engineering Task Force, Internet-Draft draft-ietf-6tischmsf-02 .

[3] Coladon, T., Vučinić, M., Tourancheau, B., 2015. Multiple redundancy constants with trickle, in: 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE. pp. 1951–1956.

[4] Di Valerio, V., Presti, F.L., Petrioli, C., Picari, L., Spaccini, D., Basagni, S., 2019. Carma: Channel-aware reinforcement learning-based multi-path adaptive routing for underwater wireless sensor networks. IEEE Journal on Selected Areas in Communications 37, 2634–2647.

[5] Duquennoy, S., Al Nahas, B., Landsiedel, O., Watteyne, T., 2015. Orchestra: Robust mesh networks through autonomously scheduled tsch, in: Proceedings of the 13th ACM conference on embedded networked sensor systems, pp. 337–350.

[6] Ghaleb, B., Al-Dubai, A., Ekonomou, E., Paechter, B., Qasem, M., 2016. Trickle-plus: Elastic trickle algorithm for low-power networks and internet of things, in: 2016 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), IEEE. pp. 103–108.

[7] Ghaleb, B., Al-Dubai, A.Y., Ekonomou, E., Romdhani, I., Nasser, Y., Boukerche, A., 2018. A novel adaptive and efficient routing update scheme for low-power lossy networks in iot. IEEE Internet of Things Journal 5, 5177–5189.

[8] Goyal, S., Chand, T., 2017. Improved trickle algorithm for routing protocol for low power and lossy networks. IEEE Sensors Journal 18, 2178–2183.

[9] Ha, Y., Chung, S.H., 2020. Enhanced 6p transaction methods for industrial 6tisch wireless networks. IEEE Access 8, 174115–174131.

[10] Hermeto, R.T., Gallais, A., Theoleyre, F., 2017. Scheduling for ieee802. 15.4-tsch and slow channel hopping mac in low power industrial wireless networks: A survey. Computer Communications 114, 84–105.

[11] Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: A survey. Journal of artificial intelligence research 4, 237–285.

[12] Kalita, A., Khatua, M., 2021a. Adaptive control packet broadcasting scheme for faster 6tisch network bootstrapping. IEEE Internet of Things Journal .

[13] Kalita, A., Khatua, M., 2021b. Autonomous allocation and scheduling of minimal cell in 6tisch network. IEEE Internet of Things Journal .

[14] Kalita, A., Khatua, M., 2021c. Opportunistic transmission of control packets for faster formation of 6tisch network. ACM Transactions on Internet of Things 2, 1–29.

[15] Kim, M.J., Chung, S.H., 2021. Efficient route management method for mobile nodes in 6tisch network. Sensors 21, 3074.

[16] Kim, S., Kim, H.S., Kim, C., 2019. Alice: Autonomous link-based cell scheduling for tsch, in: Proceedings of the 18th International Conference on Information Processing in Sensor Networks, pp. 121–132.

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

[18] Lamaazi, H., Benamar, N., 2020. Rpl enhancement based fl-trickle: A novel flexible trickle algorithm for low power and lossy networks. Wireless Personal Communications 110, 1403–1428.

[19] Le, H.P., John, M., Pister, K., 2009. Energy-aware routing in wireless sensor networks with adaptive energy-slope control. EE290Q-2 Spring .

[20] Levis, P., Clausen, T., Hui, J., Gnawali, O., Ko, J., 2011. The trickle algorithm. Internet Engineering Task Force, RFC6206 .

[21] Meyfroyt, T.M., 2015. An analytic evaluation of the trickle algorithm: Towards efficient, fair, fast and reliable data dissemination, in: 2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE. pp. 1–2.

[22] Municio, E., Daneels, G., Vučinić, M., Latré, S., Famaey, J., Tanaka, Y., Brun, K., Muraoka, K., Vilajosana, X., Watteyne, T., 2019. Simulating 6tisch networks. Transactions on Emerging Telecommunications Technologies 30, e3494.

[23] Musaddiq, A., Zikria, Y.B., Kim, S.W., 2018. Energy-aware adaptive trickle timer algorithm for rpl-based routing in the internet of things,

in: 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), IEEE. pp. 1–6.

[24] Nain, Z., Musaddiq, A., Qadri, Y.A., Nauman, A., Afzal, M.K., Kim, S.W., 2021. Riata: A reinforcement learning-based intelligent routing update scheme for future generation iot networks. IEEE Access .

[25] Rawat, P., Singh, K.D., Chaouchi, H., Bonnin, J.M., 2014. Wireless sensor networks: a survey on recent developments and potential synergies. The Journal of supercomputing 68, 1–48.

[26] Righetti, F., Vallati, C., Das, S.K., Anastasi, G., 2020. Analysis of distributed and autonomous scheduling functions for 6tisch networks. IEEE Access 8, 158243–158262.

[27] Vallati, C., Brienza, S., Anastasi, G., Das, S.K., 2018. Improving network formation in 6tisch networks. IEEE Transactions on Mobile Computing 18, 98–110.

[28] Vilajosana, X., Pister, K., Watteyne, T., 2017. Minimal ipv6 over the tsch mode of ieee 802.15. 4e (6tisch) configuration. Internet Engineering Task Force RFC series .

[29] Vilajosana, X., Wang, Q., Chraim, F., Watteyne, T., Chang, T., Pister, K.S., 2013. A realistic energy consumption model for tsch networks. IEEE Sensors Journal 14, 482–489.

[30] Vučinić, M., Król, M., Jonglez, B., Coladon, T., Tourancheau, B., 2017. Trickle-d: High fairness and low transmission load with dynamic redundancy. IEEE internet of things journal 4, 1477–1488.

[31] Watkins, C.J., Dayan, P., 1992. Q-learning. Machine learning 8, 279–292.

[32] Winter, T., Thubert, P., Brandt, A., Hui, J.W., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J.P., Alexander, R.K., et al., 2012. Rpl: Ipv6 routing protocol for low-power and lossy networks. rfc 6550, 1–157.

[33] Yassein, M.B., Aljawarneh, S., et al., 2017. A new elastic trickle timer algorithm for internet of things. Journal of Network and Computer Applications 89, 38–47.

[34] Yibo, C., Hou, K.M., Zhou, H., Shi, H.L., Liu, X., Diao, X., Ding, H., Li, J.J., de Vaulx, C., 2011. 6lowpan stacks: A survey, in: 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE. pp. 1–4.