QL-TSCH-plus: A Q-learning distributed scheduling algorithm for TSCH networks

Mehdi Kherbache, Moufida Maimour, Eric Rondeau

QL-TSCH-plus : A Q-learning Distributed Scheduling Algorithm for TSCH Networks

Mehdi Kherbache
CRAN Laboratory, Université de Lorraine
CNRS UMR 7039, Campus Sciences, BP 70239
F-54000 Nancy, France
mehdi.kherbache@univ-lorraine.fr

Moufida Maimour
CRAN Laboratory, Université de Lorraine
CNRS UMR 7039, Campus Sciences, BP 70239
F-54000 Nancy, France
moufida.maimour@univ-lorraine.fr

Eric Rondeau
CRAN Laboratory, Université de Lorraine
CNRS UMR 7039, Campus Sciences, BP 70239
F-54000 Nancy, France
eric.rondeau@univ-lorraine.fr

***Abstract*—Addressing the requirements of Industrial Internet of Things (IIoT) in Industry 4.0, the Time Slotted Channel Hopping (TSCH) protocol of the IEEE 802.15.4e amendment has been proposed. However, the lack of a defined scheduling procedure in the standard remains an open research area. Existing reinforcement learning-based scheduling proposals demonstrate great potential for this technique due to the ongoing observations within the network environment. Beneficial for real-world scenarios where network conditions are volatile and unpredictable. This work presents QL-TSCH-plus, an enhancement of the existing QL-TSCH scheduler that reduces energy consumption by adapting the Action Peeking mechanism to a distributed scheme. Instead of continuously listening to neighboring nodes communication, QL-TSCH-plus allows nodes to broadcast the learned transmission slots for updating the Action Peeking Tables and allocating reception slots, reducing energy use by up to 47% compared to QL-TSCH. This novel approach also maintains reliability and timeliness, demonstrating significant potential for efficient scheduling in TSCH networks, making it suitable for the IIoT.**

***Index Terms*—TSCH scheduling, Reinforcement Learning, IIoT, Industry 4.0.**

I. Introduction

The growing developments in industrial applications for Industry 4.0 bring new challenges and requirements to be satisfied. The Industrial Internet of Things (IIoT) is the one of the baseline technologies leading to the fourth industrial revolution [1]. These networks facilitate the establishment of intelligent factories, in which machinery and production systems are interconnected. They communicate amongst themselves to refine production processes, leading to cost reduction, enhanced efficiency, and the enablement of data-driven decision making. The IIoT is characterized by strict requirements in terms of reliability, real-time performance, energy efficiency, among others [2]. Several organizations and working groups are working on satisfying such requirements and providing ground efforts for a standardized IIoT [3]. IEEE 802.15.4e amendment [4] proposed TSCH (Timed Slotted Channel Hopping) [5] as a MAC (Media Access Control) protocol tailored to fulfill the industrial needs for both reliability and timeliness. It mainly finds its application domains in industrial automation and process control, which specifically comprise energy production applications, water management systems, food production plants, pharmaceutical manufacturing facilities, and oil and gas process plants. In TSCH, nodes follow a two dimensional slotframe providing time slotted access and channel hopping for maximum network efficiency. However, only the slotframe structure is defined in the standard without specifying the schedule definition procedure, leaving it an open research area. This gained a huge interest in the networking research community where several scheduling algorithms have been proposed in various fashions [3]; centralized, distributed, autonomous and even Reinforcement Learning (RL) based ones.

RL has been employed for Time Division Multiple Access (TDMA) scheduling, including TSCH, due to its exceptional adaptability in uncertain environments, such as those found in wireless networks [6]. The exploration of reinforcement learning methods to enhance the performance of TDMA-based networks has resulted in several innovative proposals. [7] introduces a Q-learning MAC protocol (QL-MAC), which uses a Q-learning scheme to adjust the node's duty-cycle. This protocol takes traffic predictions and the state of neighboring nodes into account to minimize energy use, proving effective in reducing energy consumption with a minimal impact on Packet Delivery Ratio (PDR) compared to CSMA/CA. Nevertheless, it does not provide any assurances for low latency. Building on this concept of energy conservation, the RL-TSCH approach [8] is introduced. This technique proposes a new method for scheduling TSCH nodes, in which the operation of a node's radio is determined based on its current and previous state. While it significantly reduces energy consumption, it does not offer low latency guarantees, similar to the QL-MAC protocol. Moreover, RL-TSCH's performance was benchmarked against the TSCH Minimal Scheduling Algorithm (MSA), indicating a more comprehensive evaluation may be necessary. Further

expanding the use of reinforcement learning in the context of TSCH networks, Phung et al. [9] proposed a scheduler to support multiple Quality of Service (QoS) objectives. Two instances of the Routing Protocol for Low-Power and Lossy Networks (RPL) are utilized, serving delay-sensitive and regular data respectively. It provides lower data delivery latency than Orchestra [10], the first autonomous scheduler for TSCH networks, while maintaining comparable energy consumption. However, the evaluation was only performed through Matlab simulations, suggesting that additional experimental validation on a real embedded system may be necessary.

The QL-TSCH scheduler [11] utilizes Q-Learning to reduce collisions while allowing contention in TSCH networks, where each node acts as a QL agent. Rewards are given based on transmission success, influencing the Q-value updates in the Q-table, which mirrors the unicast slotframe size. To tackle the non-stationarity issue in the network, represented as a MAS (Multi-Agent System), the protocol introduces an Action Peeking (AP) mechanism that monitors the usage rate of each slot in the unicast slotframe via an Action Peeking Table (APT). An epsilon-greedy strategy drives exploration and exploitation phases, encouraging initial exploration and gradual shift towards exploitation. The results highlight its high performance compared to Orchestra in terms of reliability and latency, without considering its energy consumption. This latter deemed to be a significant limitation of QL-TSCH, as stated in [12], due to the constant need for nodes to monitor neighboring node communications, an integral part of the action peeking mechanism. Consequently, nodes never attain a sleep state, leading to substantial energy consumption in the network.

In this work, we introduce QL-TSCH-plus as an enhancement to QL-TSCH scheduler to reduce its energy consumption. Our approach introduces a distributed scheme for the AP mechanism while maintaining QL-TSCH original reward function. Instead of constantly monitoring its neighbor nodes' communications, a node now broadcasts the learned transmission slot to its neighboring nodes, allowing the receivers to update their APT for the transmitted slot. Moreover, this additional broadcast traffic is also exploited for Rx slots allocation while ensuring their alignment with the RPL [13] (Routing Protocol for Low-Power and Lossy Networks) routing tree. QL-TSCH-plus reduces energy consumption by as much as 47% compared to QL-TSCH without compromising the network reliability and timeliness.

The rest of the paper is organized as follows. Section II covers TSCH protocol in detail. Then, QL-TSCH scheduling algorithm is presented in Section III before detailing our contribution in Section IV. Performance evaluation of both algorithms is covered in Section V before concluding the paper and discussing insights into potential future perspectives in Section VI.

II. Overview of TSCH

The IEEE 802.15.4e amendment [4] introduced several contention-free channel access modes, including TSCH, which has attracted significant attention from academia and industry due to its superior performance in real-time constrained applications. TSCH is targeted towards domains like industrial automation and process control, providing support for multi-hop and multi-channel communications. This is achieved through a combination of time slotted access and channel hopping mechanisms [5].

The former mechanism significantly reduces collisions between competing nodes, boosting throughput and delivering deterministic latency to applications. Meanwhile, the channel hopping mechanism allows multiple nodes to simultaneously communicate over different channels, enhancing the network's capacity and reliability by mitigating interference's adverse effects.

Nodes synchronize to a periodic slotframe composed of several timeslots in TSCH, repeated throughout the network's lifecycle. Node communication follows a defined schedule, optimized for maximum efficiency. This schedule can be modeled as a matrix, with available channels as rows and timeslots in a slotframe as columns. Each matrix cell represents a specific link, defined by its coordinates (ChannelOffset, SlotOffset), and can be reserved for a single link or shared by multiple links. If a collision occurs, the CSMA-CA algorithm is executed.

The frequency at which two nodes communicate in a timeslot is calculated by the formula

$$f = F[(ASN + channelOffset)\%N_{channels}] \qquad (1)$$

where ASN represents the total elapsed timeslots since the network service began, incremented in each timeslot. Function F can be implemented as a lookup table, typically defined as the hopping sequence specified in TSCH. For instance, if four channels are used, the hopping sequence could be 15, 20, 25, 26. Notably, Equation (1) can yield different frequencies for the same link across different timeslots, thus enabling the channel hopping mechanism. Several scheduling algorithms have been proposed in the literature for TSCH networks [3]. Centralized, distributed, autonomous and based on reinforcement learning [6].

III. QL-TSCH

Reinforcement Learning (RL) is a subfield of machine learning that focuses on sequential decision-making [14]. In an RL paradigm, an agent learns to interact with an environment to achieve a specific goal. The agent learns through trial and error, choosing actions that lead to states offering high rewards. The aim is to develop a policy, a mapping of states to actions, that maximizes the sum of rewards, also known as the return, over the course of an episode.

Q-Learning [15], a model-free algorithm in the RL framework, enables an agent to learn the value of an action taken in a particular state. The key idea behind Q-Learning is the Q-function, or action-value function, denoted as $Q(s, a)$, which estimates the future return for taking action a in state s. In

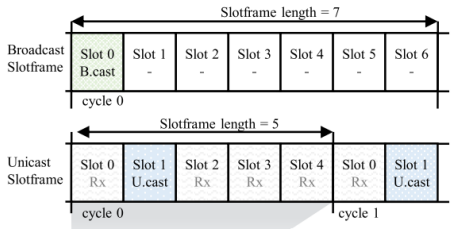


Fig. 1: Slotframes design of QL-TSCH, from [11]

the process of learning, the Q-function is updated using the following rule:

$$Q(s,a) \leftarrow \alpha\ Q(s,a) + (1-\alpha)(r + \gamma\ max_a\ Q(s',a)) \quad (2)$$

where α is the learning rate that controls how much of the new information we incorporate, r is the immediate reward, γ is the discount factor that quantifies the importance of future rewards, and $max_a\ Q(s',a))$ represents the largest Q-value that can be obtained from the actions that can be taken in the next state.

Applying Q-Learning in the context of Time Slotted Channel Hopping (TSCH) networks, specifically for scheduling communication, brings several advantages. Firstly, Q-Learning allows for dynamic, adaptive scheduling based on the ongoing interactions and observations within the network. This adaptability is beneficial in real-world scenarios where network conditions are volatile and unpredictable. Furthermore, Q-Learning's model-free nature makes it suitable for complex environments where an accurate model of the environment's dynamics is not available or is too computationally expensive to use. With its ability to learn optimal policies from interactions with the environment, Q-Learning presents a promising approach to develop efficient, reliable, and adaptable scheduling mechanisms for TSCH networks.

The QL-TSCH scheduler, proposed in [11], is based on Q-Learning, allowing contention but striving to minimize collisions. In this protocol, there are two types of slotframes: broadcast and unicast, as shown in Figure 1. For the broadcast slotframe, it has 7 timeslots in length, with only the first slot being allocated for shared transmission (Tx) and reception (Rx), while the node turns off its radio during the remaining slots. In contrast, the unicast slotframe, which is used for data communication, can have different sizes (9, 15, 25). Here, only one slot is considered a Tx cell, and all other slots act as Rx cells.

The network is represented as a multi-agent system where each node serves as a QL agent learning from its environment and the neighboring nodes to determine the appropriate slot for Tx. To handle the non-stationarity problem that might disrupt the algorithm's convergence in the multi-agent system, QL-TSCH incorporates an Action Peeking (AP) mechanism. In this scheme, a node listens to the medium during Rx slots and logs this data into the Action Peeking Table (APT), which has the same size as the unicast slotframe. Basically, the APT records the usage rate of each slot. As a result, the protocol can

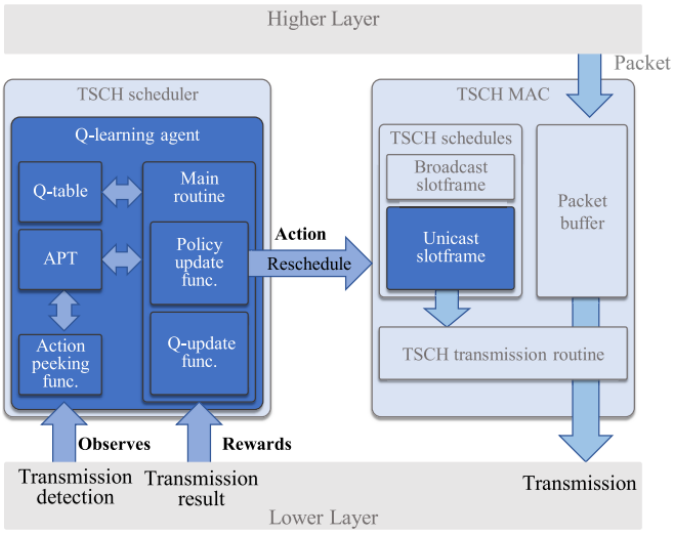


Fig. 2: Structure diagram of QL-TSCH, from [11]

select the least utilized slot (i.e., the one with the minimum value) from the APT as a Tx slot.

The protocol awards a node based on the success of its transmission on the Tx slot, providing a reward of 0 for a successful transmission and -1 for a failed one. This $0/-1$ reward design was found to be the most efficient after testing different setups by the QL-TSCH authors. Following the receipt of the reward, the node updates the respective Q-value in the Q-table, which matches the size of the unicast slotframe. Figure 2 summarizes the global operation of QL-TSCH algorithm. After each unicast slotframe cycle, both the Q-table and the APT are updated, and a new Tx slot is scheduled. QL-TSCH adopts an epsilon-greedy exploration/exploitation strategy. During exploration, a node updates its Q-table and APT after transmitting on the Tx slot and selects the next Tx slot with the minimum APT value, indicating that it is exploring the environment by trying different timeslots. Conversely, during exploitation, the node selects the Tx slot based on the Q-table values, specifically the timeslot with the highest Q-value, suggesting that the node has earned the most rewards from this timeslot. QL-TSCH incorporates a policy update function that encourages nodes to start with exploration and then gradually shift to exploitation as time passes.

IV. QL-TSCH-PLUS

A major drawback of QL-TSCH is that it requires a node to constantly listen to communications between its neighboring nodes. This is an inherent part of the action peeking mechanism used to address the non-stationarity issue in the network, which is viewed as a MAS. As a result, the nodes never enter a sleep state, leading to a high energy consumption within the network. To mitigate the high energy consumption issue, we introduce an enhancement to the action peeking mechanism. Recognizing that the primary contributor to increased energy consumption is the active listening feature, our proposed

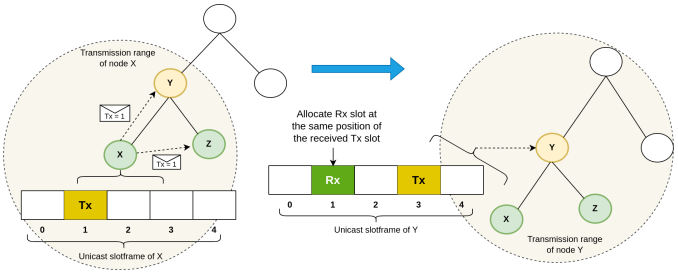


Fig. 3: Rx slots allocation

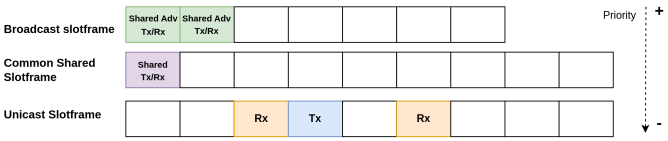


Fig. 4: QL-TSCH-plus slotframes design

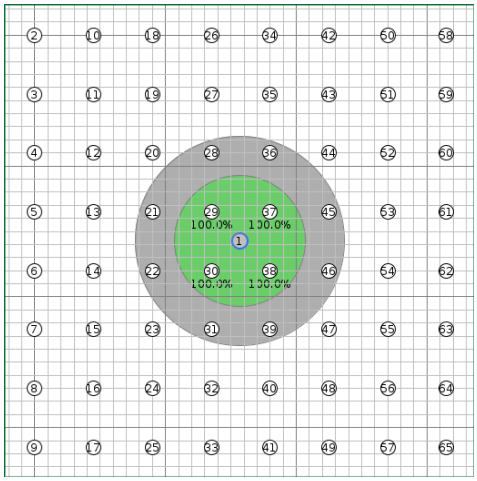


Fig. 5: Network topology

modification involves disabling this feature. This is achieved by eliminating the Rx slots that are allocated across all the slotframe except for the Tx slot. Instead, these slots are allocated according to the routing tree.

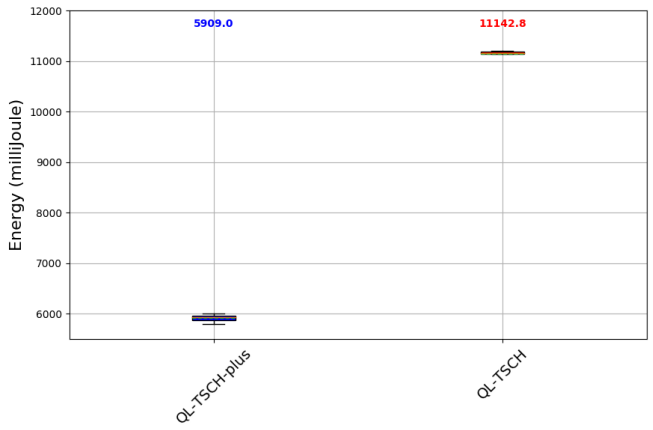
In order to keep the action peeking mechanism correct functioning, a node will not actively sense communication between its neighbors. Instead, it will broadcast the learned unicast Tx slot (e.g., denoted t_i). A neighboring node upon receiving such a packet increments the APT value of t_i. Furthermore, the receiving node verifies if the sender node is a child node in relation to the routing tree, specifically, the RPL DODAG (Destination Oriented Directed Acyclic Graph). If that is the case, it allocates an Rx timeslot that corresponds to the position of the received Tx slot (t_i), as depicted in Figure 3. This guarantees that the Rx slots are in sync with the learned Tx slots in the neighboring nodes to ensure that each node can send data to its RPL parent or receive data from its children nodes. With the need for nodes to exchange a message containing the learned Tx slot for coordination, the algorithm evolves into a distributed one and loses its autonomous nature. This additional data flow, which is of a broadcast type, will be dispatched within the broadcast slotframe, thus, we allocate a second broadcast timeslot for it. On another note, RPL signaling messages can take the form of either unicast or broadcast. If they are unicast, they will be sent in the unicast slotframe, causing them to run simultaneously with the unicast data packets. To manage this, we integrate an additional slotframe equipped with a common shared Tx/Rx slot specifically to accommodate the RPL control packets. This results in three slotframes in total as shown in Figure 4, one for broadcast traffic (TSCH EBs, RPL DIOs, Packets containing the learned Tx slot, etc), one for RPL unicast signaling traffic and one for unicast data traffic, each running on a different channel offset.

V. Performance Evalution

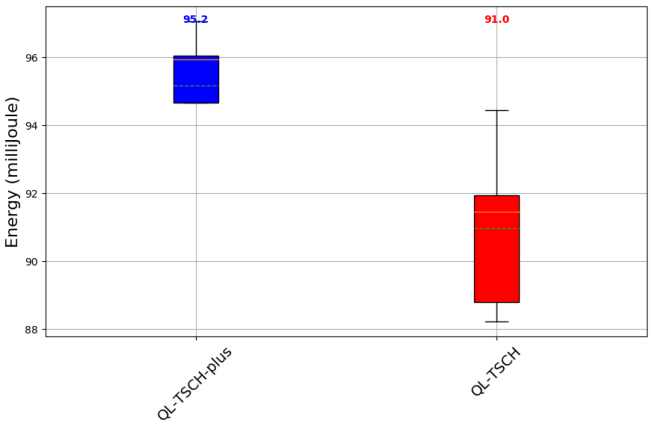
For a thorough evaluation of QL-TSCH-plus relative to the original QL-TSCH, we conduct a set of experiments under a unified environment that includes a notably heterogeneous traffic scenario. Consistency is ensured across all tests via a common network model and evaluation metrics. Our experiments have been conducted using COOJA simulator [16] provided by Contiki-NG [17], an operating system for the IoT that implements a low-power IPv6 networking stack. Our experimental network topology, depicted in Figure 5, consists of a central Sink (node 1) encircled by 64 sensor nodes that form an 8x8 grid arrangement. The inter-node distance within the grid is maintained at 45 meters. All sensor nodes are equipped with an IEEE 802.15.4 radio, and adhere to the well-established Unit Disk Graph Medium (UDGM) as the propagation model. Cooja motes, being generic sensor nodes, are selected as the nodes of choice for these experiments. In terms of energy consumption quantification, we utilize an energy model based on Sky Motes [18]. Additionally, Contiki-NG Energest module is exploited, which is set to periodically calculate energy use every 10 seconds. In our scenario, sensor nodes are responsible for forwarding data from their respective areas of interest to the Sink, also referred to as the RPL root node. This data relay follows a 'convergecast' or many-to-one communication pattern, a widely used pattern in many sensor network applications. Four sensor nodes (nodes 2, 9, 58, and 65), strategically located at the grid's corners, carry the additional duty of producing and transmitting data at a high frequency. This results in what we refer to as 'heavy traffic' within our experiments. Each of these high-traffic flows involves the delivery of two packets per second (2 pps), each bearing a payload of 50 bytes, leading to an overall transmission rate of 800 bps (bits per second). The remaining 60 nodes transmit a single packet per minute, carrying a 10 byte payload. This results in an individual transmission rate of 1.33 bps, hence we label these as 'light traffic' flows. The Sink, or RPL root node, stands as a purely receptive

TABLE I: Simulation Parameters

Parameter	Value
Warm-up duration	30 minutes
Data generation duration	30 minutes
Mote type	COOJA mote
Protocol stack	6TiSCH
Transport Protocol	User Datagram Protocol (UDP)
Routing Protocol	RPL Classic
Number of channels	4
Network topology	Grid 8×8
Number of nodes	65
Distance between nodes	45 meters
Propagation model	Unit Disk Graph Medium (UDGM)
Interference range	80 meters
Tx range	50 meters
Heavy traffic	4 nodes, each transmitting at 800 *bps*
Light traffic	60 nodes, each transmitting at 1.33 *bps*
Timeslot duration	10ms
Maximum re-transmissions	3
Queue buffer size	8
QL-TSCH	
Broadcast slotframe size	7
Unicast slotframe size	5
QL-TSCH-plus	
Broadcast slotframe size	15
RPL slotframe size	13
Unicast slotframe size	5
Power Radio Rx	65.4 mw
Power Radio Tx	58.5 mw
Power CPU	7.2 mw
Power Low Power Mode (LPM)	3.6 mw



(a) Total energy.



(b) Transmission energy.

Fig. 6: Network energy consumption in millijoule.

entity in this setup, and hence does not generate any data packets. We run each configuration 5 times to increase the confidence interval of our simulations. The main simulation parameters are listed in Table I. We consider three metrics for the evaluation :

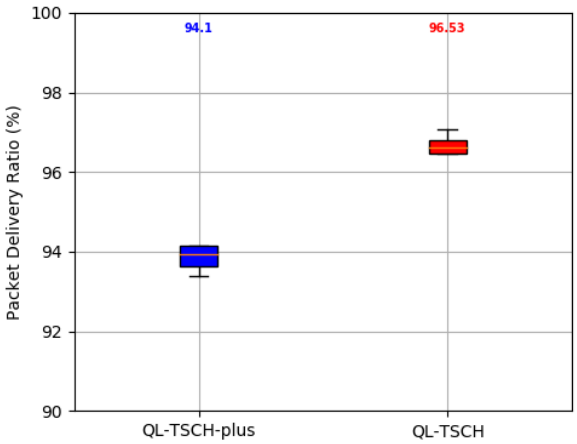
- Energy consumption : the total amount of energy consumed by the network during the whole simulation time.
- Packet Delivery Ratio (PDR) : The ratio of the number of packets received by the sink to the number of packets sent by the source nodes.
- End-to-end Delay : The time that elapses between the transmission of a packet by a source node and its reception by the sink.

Figure 6 showcases the energy expenditure of the entire network for QL-TSCH and QL-TSCH-plus. Since the pattern of energy consumption for reception parallels that of the total energy, the figure solely illustrates the total and transmission energy expenditure for clarity and brevity. Notably, QL-TSCH-plus achieves up to 47% energy reduction compared to its QL-TSCH counterpart (Figure 6a). Moreover, the transformation of QL-TSCH-plus into a distributed scheduler with an additional control flow as compared to its predecessor, QL-TSCH, results in a higher transmission energy expenditure, as highlighted in Figure 6b. Nevertheless, this increase in transmission energy is overshadowed by the energy savings achieved through the more efficient allocation of reception cells in QL-TSCH-plus, which is based on the RPL routing tree. Contrarily to QL-TSCH which allocates all the unicast cells for reception except for the learned Tx cell, mandatory for its active listening feature. On the other hand, as seen in Figure 7a, the PDR of QL-TSCH-plus (mean value of 94.1%) slightly underperforms that of QL-TSCH (mean value of 96.53%), but the marginal difference can be overlooked considering the substantial energy savings. Additionally, as demonstrated in Figure 7b, the end-to-end delay for QL-TSCH-plus (0.4 seconds mean value) exhibits a minor increase in comparison to QL-TSCH (0.35 seconds) but it is also acceptable for the same reasons as the PDR decrease.

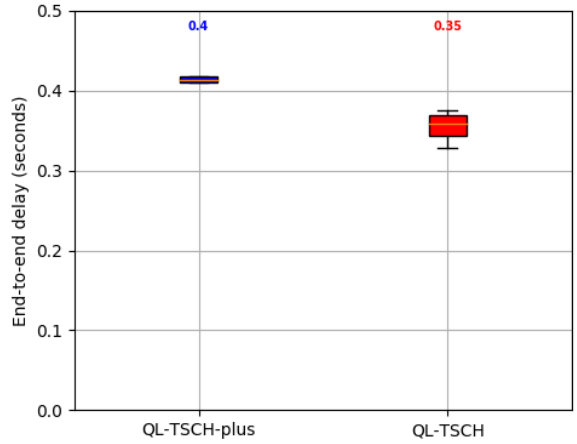
The performance obtained by QL-TSCH-plus makes it suitable for the IIoT since it responds to their requirements in terms of energy, reliability and real-time performance.

VI. Conclusion

In summary, this paper presented QL-TSCH-plus, an enhancement to the QL-TSCH scheduler, achieving a significant reduction in energy consumption without compromising network reliability and timeliness, making a highly efficient scheduler for IIoT networks in the context of Industry 4.0.



(a) Network Packet Delively Ratio (%)



(b) Network End-to-End Delay (seconds)

Fig. 7: PDR (%) and End-to-End delay (s) results.

The energy efficiency of QL-TSCH-plus is derived from the implementation of a distributed scheme for the Action Peeking mechanism. In this approach, each node's learned transmission cell is broadcast, enabling the neighboring nodes to promptly update their Action Peeking Tables. Additionally, this distributed scheme serves a dual purpose by also facilitating the allocation of reception cells. It ensures that the allocation is congruent with the transmission cells, all while following the RPL routing tree to ensure that each node can send data to its RPL parent or receive data from its children nodes.

In the future, one can consider a combination of RL-based timeslot selection and channel blacklisting to mitigate the effects of interference and fading in TSCH networks.

REFERENCES

[1] I. Khan and M. Javaid, "Role of Internet of Things (IoT) in Adoption of Industry 4.0," *Journal of Industrial Integration and Management*, vol. 07, no. 04, pp. 515–533, 2022. [Online]. Available: https://doi.org/10.1142/S2424862221500068

[2] M. Alabadi, A. Habbal, and X. Wei, "Industrial Internet of Things: Requirements, Architecture, Challenges, and Future Research Directions," *IEEE Access*, vol. 10, pp. 66 374–66 400, 2022.

[3] A. R. Urke, O. Kure, and K. Ovsthus, "A Survey of 802.15.4 TSCH Schedulers for a Standardized Industrial Internet of Things," *Sensors*, vol. 22, no. 1, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/1/15

[4] Domenico De Guglielmo, Giuseppe Anastasi, and Alessio Seghetti, "From IEEE 802.15.4 to IEEE 802.15.4e: A Step Towards the Internet of Things," *Advances in Intelligent Systems and Computing*, 2014.

[5] I. std., "802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendament 1: MAC sublayer," IEEE standard for Information Technology, Standard, April 2012.

[6] M. Kherbache, O. Sobirov, M. Maimour, E. Rondeau, and A. Benyahia, "Reinforcement Learning TDMA-Based MAC Scheduling in the Industrial Internet of Things: A Survey," *IFAC-PapersOnLine*, vol. 55, no. 8, pp. 83–88, 2022, 6th IFAC Symposium on Telematics Applications TA 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322010874

[7] C. Savaglio, P. Pace, G. Aloi, A. Liotta, and G. Fortino, "Lightweight Reinforcement Learning for Energy Efficient Communications in Wireless Sensor Networks," *IEEE Access*, vol. 7, pp. 29 355–29 364, 2019.

[8] H. Nguyen-Duy, T. Ngo-Quynh, F. KOJIMA, T. Pham-Van, T. Nguyen-Duc, and S. Luongoudon, "RL-TSCH: A Reinforcement Learning Algorithm for Radio Scheduling in TSCH 802.15.4e," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, 2019, pp. 227–231.

[9] K.-H. Phung, T. T. Huong, D. Khanh Dung, V. X. Tuong, T. Pham, T. Nguyen, and K. Steenhaut, "A Scheduler for Time Slotted Channel Hopping Networks supporting QoS Differentiated Services," in *2018 International Conference on Advanced Technologies for Communications (ATC)*, 2018, pp. 232–236.

[10] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 337–350. [Online]. Available: https://doi.org/10.1145/2809695.2809714

[11] H. Park, H. Kim, S.-T. Kim, and P. Mah, "Multi-Agent Reinforcement-Learning-Based Time-Slotted Channel Hopping Medium Access Control Scheduling Scheme," *IEEE Access*, vol. PP, pp. 1–1, 07 2020.

[12] M. Kherbache, O. Sobirov, M. Maimour, E. Rondeau, and A. Benyahia, "Decentralized TSCH scheduling protocols and heterogeneous traffic: Overview and performance evaluation," *Internet of Things*, vol. 22, p. 100696, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660523000197

[13] P. Thubert and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Engineering Task Force, Internet-Draft draft-pthubert-roll-rfc6550bis-01, May 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-pthubert-roll-rfc6550bis-01

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

[15] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 05 1992.

[16] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, 2006, pp. 641–648.

[17] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[18] "Tmote Sky : Datasheet," Tech. Rep., 2006.