IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2020, accepted July 9, 2020, date of publication July 20, 2020, date of current version August 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010575

Multi-Agent Reinforcement-Learning-Based Time-Slotted Channel Hopping Medium Access Control Scheduling Scheme

HUIUNG PARK, HAEYONG KIM, SEON-TAE KIM, (Member, IEEE), AND PYEONGSOO MAH
Electronics and Telecommunication Research Institute (ETRI), Daejeon 34129, South Korea

Corresponding author: Pyeongsoo Mah (pmah@etri.re.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korean Government [Ministry of Science, ICT and Future Planning (MSIP)] (Development of Integrated Development Solution Supporting Low-power OS for Lightweight Embedded Devices) under Grant 1711042605, and in part by the Ministry of Science and ICT (MSIT)/Institute for Information and Communications Technology Promotion (IITP), South Korea, through the ICT Research and Development Program (Development of ICT Core Technologies for Safe Unmanned Vehicles) under Grant 1711055509.

ABSTRACT Time-slotted channel hopping (TSCH) is a medium access control technology that realizes collision-free wireless network communication by coordinating the media access time and channel of network devices. Although existing TSCH schedulers have suitable application scenarios for each, they are less versatile. Scheduling without collisions inevitably lowers the throughput, whereas contention-based scheduling achieves high-throughput but it may induces to frequent collisions in densely deployed networks. Therefore, a TSCH scheduler that can be used universally, regardless of the topology and data collection characteristics of the application scenario, is required to overcome these shortcomings. To this end, a multi-agent reinforcement learning (RL)-based TSCH scheduling scheme that allows contention but minimizes collisions is proposed in this study. RL is a machine-learning method that gradually improves actions to solve problems. One specific RL method, Q-Learning (QL), was used in the scheme to enable the TSCH scheduler to become a QL agent that learns the best transmission slot. To improve the QL performance, reward functions tailored for the TSCH scheduler were developed. Because the QL agent runs on multiple nodes concurrently, changes in the TSCH schedule of one node also affect the performance of the TSCH schedules of other nodes. The use of action peeking is proposed to overcome this non-stationarity problem in a multi-agent environment. The experimental results indicate that the TSCH scheduler consistently performs well in various types of applications, compared to other schedulers.

INDEX TERMS Internet of Things (IoT), time-slotted channel hopping (TSCH) scheduling, wireless sensor networks.

I. INTRODUCTION

Time-slotted Channel hopping (TSCH) combines slotted access and channel hopping to enable reliable communication, making it a promising medium access control (MAC) technology for industrial Internet of Things applications such as smart metering networks. The most distinctive feature of TSCH is that the medium access time and channel of network devices can be tuned to achieve collision-free wireless network communication.

Many researchers have studied the optimization of TSCH schedulers for their intended applications. Specifically, most studies have examined collision-free schedulers to achieve high reliability. However, collision-free schedulers have limited scalability, making them suitable only for small networks, low traffic, or deterministic applications. This is because it is difficult for a node to allocate collision-free slot schedules between neighboring nodes in large dense networks owing to limited slot resources. Moreover, low link throughput due to insufficient slot resources makes it impossible to transfer large data or process multi-hop routing packets.

The contention-based TSCH scheduler, which has not been studied extensively, is useful for large networks, high traffic, or request-response model-based applications. This method improves the throughput by sharing slot resources

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

and ensures reliability by utilizing retransmission mechanisms [1]. However, the reliability may be insufficient in high-density environments with frequent collisions. Moreover, it may lower the energy efficiency due to retransmission. Therefore, there is a need for an advanced TSCH scheduler that can guarantee reliability and throughput simultaneously.

Reinforcement learning (RL) is a machine-learning method that performs gradually improved actions to solve problems. RL can be used to enable a device to find a schedule that allows contention while reducing collisions. Interesting studies that have used machine learning for wireless MAC have been reported. For example, Li *et al.* [2] used a multi-armed bandit algorithm for channel blacklisting in TSCH networks. However, previous studies only investigated the improvement of channel usage to limit transmission on poor-quality channels; they did not investigate TSCH scheduling.

This study proposes a multi-agent RL-based TSCH scheduling scheme that allows contention and minimizes collisions at the same time. The proposed scheduler learns the transmission slot with the lowest transmission failure rate and transmits only in that slot. This improves the low-throughput problem, which is the main weakness of collision-free schedulers. By distributing the medium access slots of the devices, collisions are reduced, compared to those in a full contention-based scheduler. In addition, the proposed scheduler, unlike most other schedulers, is autonomous and does not require additional communication to distribute or create schedules.

Changes in the TSCH schedule of one device also affect the performance of the TSCH schedules of other devices. This mutual influence, which may prevent the learning process from converging, is known as the non-stationarity problem, which is a well-known challenge in a multi-agent system (MAS). The action peeking method is proposed to mitigate the effects of this problem.

The proposed RL-based scheduler reduces collisions and maintains sufficient throughput. The contributions of this study are summarized below:

- A practical TSCH scheduler is presented for large-scale, high-density, high-traffic applications.
- Multi-agent RL is applied in a TSCH scheduling algorithm for the first time.
- A method is proposed to address the non-stationarity problem in multi-agent RL.

The remainder of this paper is organized as follows. Section II introduces important concepts relating to TSCH and RL that the design of the proposed TSCH scheduler is based on. Section III presents the proposed RL-based TSCH scheduler. Section IV describes the experimental design for verifying the performance of the RL-based TSCH scheduler. Section V presents the experimental results. Section VI introduces related studies in which RL was applied to link layers. Finally, Section VII presents the conclusions of this study.

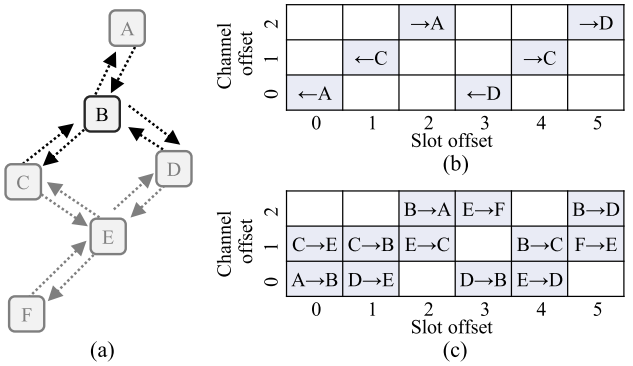

FIGURE 1. Example of a TSCH schedule in a TSCH network. (a) Topology. (b) Node B's TSCH schedule, which is a set of TSCH links. (c) Distribution of TSCH schedules of all nodes in the network.

II. BACKGROUND

A. OVERVIEW OF TSCH

TSCH is one of the operating modes of the IEEE 802.15.4 standard. It originated from the WirelessHART [3] and ISA.110.11a [4] industrial wireless standards. TSCH reduces collisions through slotted access to a medium and mitigates the effects of multipath fading and interference by using channel hopping [5]. The key terms and concepts in TSCH MAC are described subsequently.

Timeslot: A timeslot is a transfer unit of time-slotted MAC. In a timeslot, one node transfers a frame and receives an acknowledgment frame. A TSCH node can transmit a frame at a transmission (Tx) slot. If the transmission fails, the node retransmits the frame at the next available slot by the TSCH collision avoidance mechanism.

Slotframe: A slotframe is a collection of timeslots that cycles as a function of time. The size of the slotframe specifies the maximum number of schedulable timeslots as well as the cycle of the timeslot, which is the opportunity for a device to transmit or receive a single frame.

Multi-slotframe structure: In a TSCH network, multiple slotframes of different sizes can operate concurrently. All slotframes are arranged in timeslots. In a multi-slotframe structure, schedules of slotframes may overlap. Therefore, a priority is assigned to each slotframe, and the schedule of a slotframe with higher priority is followed. The multi-slotframe structure enables complex communication such as the application of different duty cycles for different packet types [6].

Channel hopping: At the beginning of each timeslot, the node performs channel hopping by

$$\text{Ch} = \text{Hopping seq}[(\text{ASN} + \text{channel offset}) \bmod N_{\text{Ch}}]. \quad (1)$$

The absolute slot number (ASN) is the cumulated slot count form the beginning of the TSCH network. The channel offset is given by the TSCH schedule. The N_{Ch} is the number of channels used in the network.

TSCH link: A TSCH link is a component of a TSCH schedule. One TSCH link contains a communication method

(i.e., channel offset, Tx or Rx, counterpart device address) at a specific slot offset by one slotframe. The device behaves as described in a corresponding TSCH link at the beginning of each timeslot.

TSCH Scheduler: The TSCH scheduler is an algorithm that assigns links to slotframes. Many researchers have attempted to develop more effective and efficient schedulers [6]–[9]. Each of these schedules has a favorable environment or application for performance. A user had to identify and apply the appropriate TSCH scheduler for the application. For example, the Orchestra scheduler can be used for applications that require high reliability in low-traffic environments. The FTA scheduler can be used for applications that collect large amounts of data over a prolonged period in large networks (e.g., smart metering) [10].

Orchestra [6] is an autonomous TSCH scheduler that maximizes reliability based on a collision-free schedule. Each node has a Tx slot offset independent of its hashed address. Therefore, although the transmission period is long, no packet collision occurs. The node also assigns a listening slot (to receive transmissions from the neighboring node) to a slot offset derived by hashing the neighbor's address. Orchestra has a policy of assigning a unique slot offset to all nodes. For this to be possible, the number of slots in the slotframe (i.e., the size of the slotframe) must be larger than the number of network nodes. Thus, Orchestra is suitable for small networks or low-traffic applications rather than for large-scale networks or high-traffic applications.

The schedule proposed by [1] is a static TSCH schedule that maximizes the throughput while allowing collisions in a wireless network. It is suitable for applications that collect massive amounts of data in large networks by allowing transmission and reception in all slots. However, many collisions occur in high-density networks; these collisions reduce the network reliability and increase packet latency. Thus, a TSCH schedule that allows contention with fewer collisions is required.

B. RL

RL was applied in TSCH schedulers to create a schedule that reduces collisions while allowing contention. This is because RL is a machine-learning method that is suitable for achieving goals while interacting with an environment [11]. RL defines learners as agents and the circumstances of the agents as the environment. The interaction between the agent and the environment is described by three essential elements: state, action, and reward [12]. An agent in state s performs an action a selected by policy π and then receives reward r from the environment. The time at which the agent starts to act until the determined end is referred to as an episode.

Q-learning (QL) is a representative RL algorithm that has been used in various fields [11], [13], [14]. It is particularly suitable for applications that require online learning as the policy can be improved every time an agent performs an action within an episode.

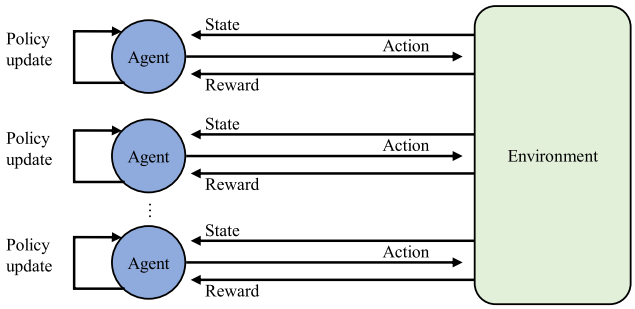

FIGURE 2. A multi-agent system where agents interact with the environment simultaneously.

A QL agent performs an action to maximize the value function, $Q(s, a)$, of the state-action pair. $Q(s, a)$ represents the expected total discounted returns from the action performed in state s:

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

where α is a learning rate parameter, and the discount factor γ determines the importance of future compensation. Further, $Q(s, a)$ represents the value of the state-action pair, 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. The agent decides the next action based on the predicted reward of the action that can be taken in that particular state.

For QL to converge on its optimal policy, it is necessary to explore every state-action pair as much as possible by running many episodes. In this regard, the ε-greedy strategy is mainly used to perform this exploration and achieve the resulting policy improvement. This strategy acts randomly with probability ε or selects the best action with probability $(1 - \varepsilon)$, with the former known as exploration and the latter as exploitation.

C. MULTI-AGENT RL

Agents in an MAS interact with other agents in the environment or with the environment itself. Agents work either in cooperation or in competition to perform more complex tasks. Examples of MASs include cluster drone flight [15], multiplayer cooperative games [16], and traffic control systems [17].

An MAS can be adversely affected by the non-stationarity problem. This is because agents can unintentionally change the environment during the learning process, thereby making the predicted potential rewards for an action inaccurate. Therefore, the present optimal policy does not guarantee a future optimal policy. QL has been proved capable of converging in a single-agent environment but not in an MAS because the Markov property is not maintained in a non-stationarity environment [18].

III. MULTIAGENT RL-BASED TSCH SCHEDULING SCHEME

An RL-based TSCH scheduler that reduces collisions while allowing contention is proposed. The proposed method takes advantage of the contention-based and collision-free approach as described in Table 1.

TABLE 1. Brief comparison between the proposed method and related works.

	QL-TSCH	Orchestra [6]	FTAS [1]
Schedule characteristics	• Contention-based learning • Collision-minimized schedule	• Collision-free schedule	• Fully contention-based schedule
Pros.	• Low collision rate • Low packet latency • Fit for large-scale dense networks	• Highly reliable communications without collision	• Minimum packet latency • Fit for large-scale networks
Cons.	• Learning process is required.	• Unsuitable for large-scale networks due to the high packet latency	• High collision rate in dense networks

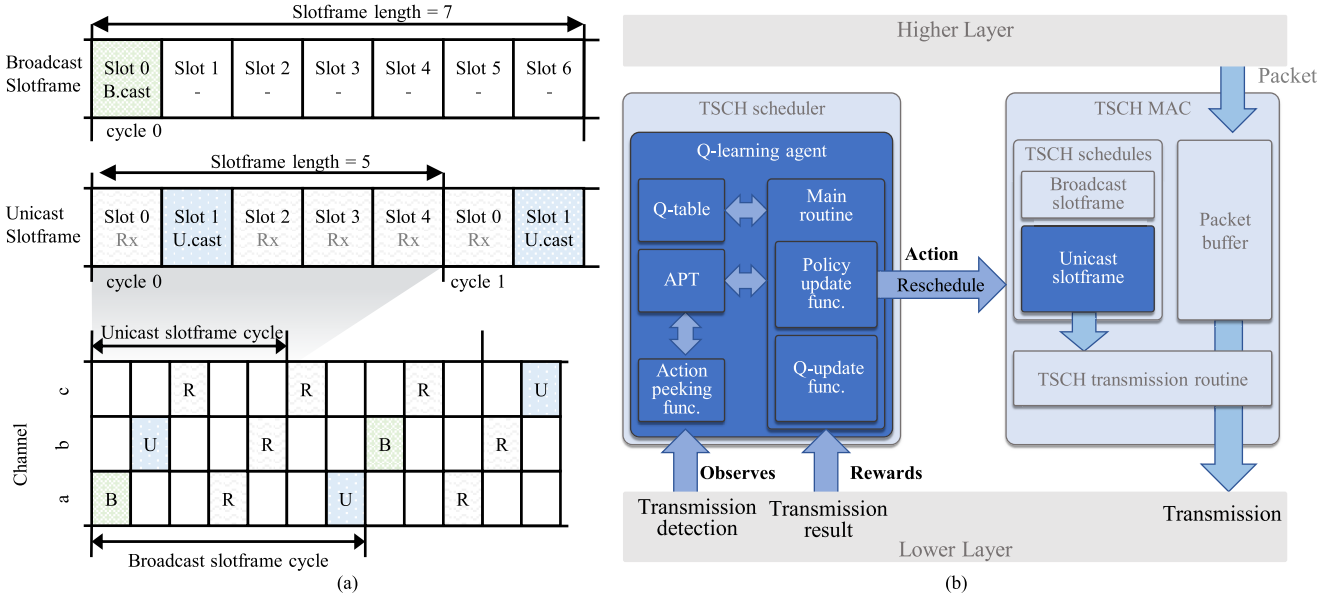

FIGURE 3. (a) An example of the QL-TSCH schedule and (b) structure diagram of RL-based TSCH MAC. The TSCH scheduler is implemented in the form of a QL agent. The TSCH scheduler allocates the unicast Tx and Rx slots of the unicast slotframe according to the agent's action. Transmission success/failure is the input of the reward function, and the Q-table is updated by the function.

The multi-agent environment enables each network device to optimize its TSCH schedules. Our scheduler is based on QL and is suitable for online learning. The optimal policy learning process of QL is modeled as a rescheduling process toward the optimal TSCH schedule. Specifically, the scheduler assigns a Q-value to each slot of a unicast slotframe, designates the slot with the highest Q-value as the Tx slot, and uses the remaining slots as listening slots.

The proposed scheduler uses a single-channel offset to enable all nodes to communicate with neighboring nodes, regardless of which slot offset is set as the transmission slot. Therefore, all nodes simultaneously hop on the same channel.

A. RL-BASED TSCH MAC STRUCTURE

Our proposed method uses multi-slotframe TSCH MAC with a unicast slotframe and a broadcast slotframe. The proposed scheduler only schedules unicast slotframes, as shown in Fig. 3. Broadcast slotframes are used to send and receive broadcast frames such as beacons and network control messages. For this purpose, all devices have a fixed broadcast slotframe schedule that contains one broadcast slot. The broadcast slotframe has a higher priority than the unicast slotframe, thereby giving priority to broadcast message transmission.

The proposed TSCH scheduler is a QL agent. Scheduling of the unicast slotframe can be regarded as an action performed by the agent. The performance of the updated policy can be evaluated according to the transmission result, i.e., the success/failure of the transmission. Agents use this success/failure result as input to the reward function to update the value of their current scheduling policy and either retain the current policy or explore to improve the policy. The Q-values

are stored in a Q-table. We assume that the environment is a single-state environment, and the number of actions is equal to the unicast slotframe length; thus, the Q-table is an array of length a.

Algorithm 1 Policy Update function

Func *policyUpdate(void)*: Repeat at the beginning of each slotframe cycle.

$APT \leftarrow \sigma\ APT$

$$a \leftarrow \begin{cases} \arg\min APT(o), & \text{with prob. } p_{exploration} \\ \arg\max Q(s,a), & \text{otherwise} \end{cases}$$

Reschedule Tx and Rx slots by a

end

Algorithm 1 represents the policy update function, which is the main element of the QL-TSCH scheduler. This function includes the action peeking mechanism. The agent performs this policy update function at the beginning of every slotframe. Depending on the hyperparameter exploration probability, $P_{exploration}$, the least active slot o or the action a with the highest Q-value is selected. Based on the selected action and the slot offset, a slot of the unicast slotframe is scheduled as a transmission slot, and the remaining slots are scheduled as listening slots.

Algorithm 2 Q Update function

Func *qUpdate(s,a)*: when Tx succeeds or fails.

$$r \leftarrow \begin{cases} \text{positive value when Tx succeeds} \\ \text{negative value when Tx fails} \end{cases}$$

$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_a Q(s,a) - Q(s,a)]$

end

The success/failure of transmission in the transmission slot is reflected in the reward and Q-value. Algorithm 2 describes the Q-Table update process. A negative reward is given for transmission failure. Transfer failure is not commonly occurred, but it can be a clue that an agent taking the same action is near by. Thus, the negative reward for transmission failure allows the agent to learn the optimal transmission slots faster. The Q-table is updated based on this reward value.

B. ACTION PEEKING

Herein, The use of action peeking is proposed to address the non-stationarity problem in an MAS. Action peeking is the act of one agent observing the actions of other agents. More specifically, a node of the TSCH network may detect communication between other neighboring nodes in the listening slot. At this time, this node may determine that a nearby node has selected the current slot as the transmission slot. In other words, the nodes infer their own communication schedule while sensing the transmission and reception packets of other nodes. The nodes aim to not affect the policy of other nodes by choosing their own slots and avoiding slots selected by other nodes through an ε-greedy exploration process.

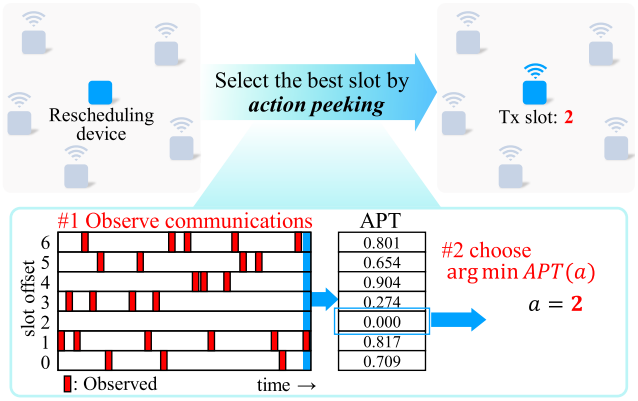

FIGURE 4. Basic principle of action peeking. As soon as the node detects nearby communication, the slot offset is recorded in the action peeking table at that time. When a node performs exploration, it selects the slot offset of the TX slot with the least recent activity.

The action peeking information is stored in an action peeking table (APT), which is an array of slotframe lengths. Each cell of the APT stores transmission information detected at a slot offset corresponding to the cell. As described in Algorithm 3, the value of the APT index corresponding to the current slot offset, o, is increased by 1 upon detecting the transmission of another device.

Algorithm 3 Action peeking

Func *actionPeeking(o)*: when communications of neighbors detected at slot offset o.

$APT(o) \leftarrow APT(o) + 1$

end

As shown in Algorithm 1, the values in the APT are degraded each time the slotframe starts. This is because the current policy of the neighboring agents may have changed. Degradation can assign a weight to the latest transmission action of neighboring devices. The agent selects $\arg\min_o APT(o)$.

C. COMPARISON BETWEEN TSCH SCHEDULERS

The proposed schedule evenly distributes the transmission opportunities of the device, thereby allowing fewer devices a transmission opportunity in one timeslot. Thus, this schedule may result in a lower probability of collision than the schedule of [1], in which all devices have transmission opportunities in every timeslot. This effect is maximized in high-density environments. Reducing the number of collisions also reduces the additional traffic due to retransmissions and the packet latency. However, owing to the duty cycle, the maximum device throughput is reduced, compared to that with the FTA scheduler. In particular, when a large amount of data is divided into several frames and transmitted, the time taken to deliver all the data to the receiver increases.

The proposed scheduler can configure the duty cycle of the device more flexibly than Orchestra does [6]. This is

because multiple devices can simultaneously be assigned to a transmission slot in one timeslot. The length of the slotframe used by Orchestra is required to be greater than the number of network nodes to specify a unique transmission slot offset for every node in the network. Therefore, the duty cycle of the device is inversely proportional to the number of network nodes. In large networks, the duty cycle becomes too long to provide the throughput required to handle a large amount of traffic.

IV. EXPERIMENTAL DESIGN

We used an ARM Cortex M4 MCU-based IoT platform device for our experiments. The MCU has 256MB RAM and 1MB internal flash. The device also has a CC1200 chip for sub-GHz communication.

To implement a large-scale network, 99 devices were installed in a testbed. One device acts as the sink node and the others, as host nodes. Devices were installed on both sides of the steel plate. Because all devices are in close proximity to each other, each device can listen to the communications of all other devices. The sink node is connected to a computer to output log data.

A. PHY AND MAC

The length of the timeslot was 10 ms, and three channels were used. Time synchronization is based on EB. The length of the broadcast slotframe was set to seven. The maximum number of frame retransmission attempts was set to three.

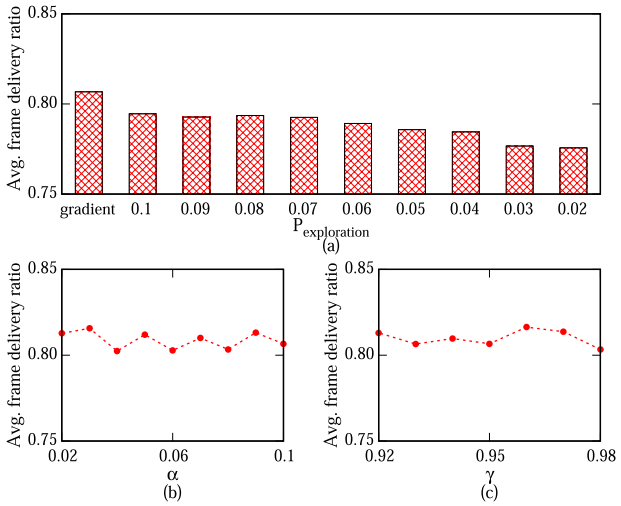

FIGURE 5. Difference in performance of QL-TSCH scheduler according to the hyperparameter value of reinforcement learning. (a) Exploration probability. (b) Learning rate α. (c) Discount factor γ.

We determined appropriate values for the QL hyperparameters (i.e., the exploration probabilities, learning rate α, and discount factors γ) by conducting preliminary experiments on a simulator as shown in Fig. 5. We set up a 100-node single-hop topology and prepared a scenario in which nodes transmit one unicast frame with a 3% probability during every slotframe cycle by setting the length of the slotframe to 15. The performance was analyzed by measuring the average frame delivery ratio 1,000 s after the network was started. First, the exploration probability was tested both with a fixed value and with a gradient method that decreased the exploration probability with time. The gradient was set based on the following equation: $p_{exploration} = \min(10{,}000/cycle, 0.5)$. The experimental results show that the gradient method outperformed the fixed exploration probability. Therefore, the gradient method was adopted in this experiment. The learning rate and discount factor did not show significant difference in performance depending on the values; thus, we set $\alpha = 0.1$ and $\gamma = 0.95$ arbitrarily.

B. EXPERIMENT 1: TSCH SCHEDULER COMPARISON

We conducted three experiments for an in-depth performance analysis. The first experiment was used to prove that the performance of the QL-TSCH scheduler is superior to that of existing schedulers such as Orchestra [6] and FTA [1] in various scenarios. To prove that, we prepared three application scenarios as described in Table 2. Scenario 1 represents a smart factory application, which is based on a dense single-hop network and requires a short data interval. Scenario 2 represents a high-traffic smart factory application that requires a multi-hop network and the ability to transmit large amounts of data. Scenario 3 represents a smart metering application that requires longer multi-hop and larger data transmission abilities.

TABLE 2. Summary of three large-scale industrial application scenarios.

	Scenario 1	Scenario 2	Scenario 3
Avg. hop/Max hop	1/1	1.6/2	3/5
Packet size (B)	50	300	500
Packet interval (s)	10	60	900

Orchestra's data slotframe size was 101, and it runs on a collision-free schedule. We assessed the performance of FTA by conducting measurements with slotframe sizes of 7. The unicast slotframe sizes of the QL-TSCH scheduler were 9, 15, and 25.

C. EXPERIMENT 2: REWARD FUNCTION EVALUATION

This experiment was designed to prove that a negative reward for a transmit failure results in more effective learning compared with a positive reward for a transmit success. For this, we compared cases in which a positive reward is only given when the transmission succeeds with cases that include a negative reward for transmission failure. Positive reward values of 0, and 1 and negative reward values of 0, -1, and -10 were tested. The size of the unicast slotframe was set to 15. The tests involved the same topology, packet size, and packet interval as in scenario 1.

D. EXPERIMENT 3: ACTION PEEKING EVALUATION

Finally, we compared the Tx schedule distribution network nodes between QL-TSCH with and without action peeking

to show that action peeking mitigates the effects of the non-stationarity problem. The size of the slotframe was set to 25, and we used the same topology as in scenario 1. Every 10 s, the nodes were sent a UDP packet, which contained information of the chosen actions at every slotframe cycle during a packet interval. When action peeking was applied, an agent chose the best action candidate by using APT. When action peeking was not applied, the device assigned any one of the slots as the transmission slot during the rescheduling process.

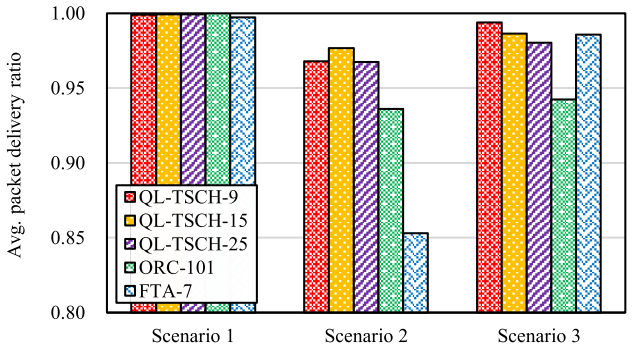

FIGURE 6. Comparison of average packet delivery ratio (PDR) between TSCH schedulers.

V. RESULTS

A. EXPERIMENT 1: TSCH SCHEDULER COMPARISON

Fig. 6 shows the average packet delivery ratio (PDR) of each QL-TSCH network for each scenario. In scenario 1, the QL-TSCH scheduler with 15-slot slotframe obtained a PDR of up to 99.942%, whereas FTA reached 99.728%. Owing to its collision-free nature, Orchestra achieved a PDR of 100%. Because scenario 1 was based on a single-hop topology, the performance of the TSCH schedulers was higher than in other scenarios.

In scenario 2, the performance of the QL-TSCH scheduler was superior. The QL-TSCH scheduler with a 15-slot slotframe achieved the best performance, i.e., a PDR of 97.674%, whereas Orchestra and FTA achieved 93.611% and 85.295%, respectively. In contrast, the performance of the FTA scheduler was dramatically lower for scenario 2. This is because of its higher frame transmission error ratio (FER), as shown in Fig. 7. Because a node can transmit a frame in any slot, considerable contention occurs when there are multiple transmissions in the network at the same time. Further, chain action is considered to occur in which more traffic is generated by the retransmission mechanism operated by the contention.

In scenario 3, the QL-TSCH scheduler with a 9-slot slotframe achieved the best performance, i.e., a PDR of 99.383%, contrary to Orchestra, which performed the worst (PDR of 94.236%). The results for scenario 3 confirm the poor performance of Orchestra in scenarios in which fragmented packets are collected in multi-hop environments [1].

As shown in Fig. 8, the packet delay of the QL-TSCH scheduler is shorter than that of Orchestra for all scenarios. The packet delay of the Orchestra scheduler was the longest

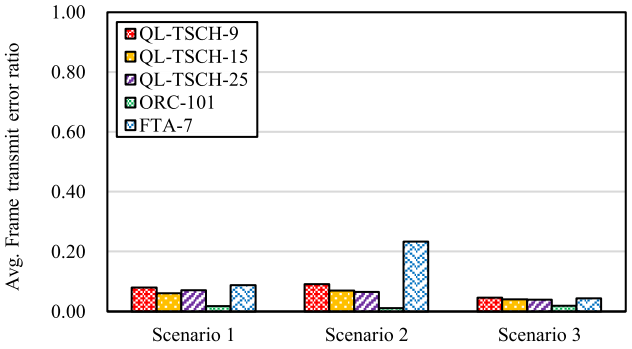

FIGURE 7. Comparison of average frame transmission error ratio (FER) of various TSCH schedulers.

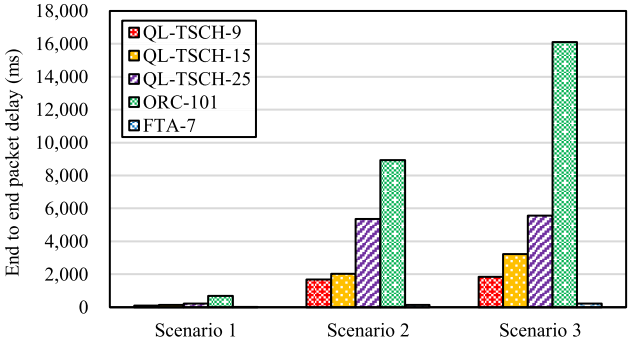

FIGURE 8. Comparison of average packet delay of different TSCH schedulers.

TABLE 3. Performance comparison of QL-TSCH networks with various reward values.

Reward		PDR	FER	Packet delay
Success	Failure			
0	-1	0.99734	0.09077	153.4
0	-10	0.99843	0.09533	156.4
1	0	0.97727	0.23192	232.5
1	-1	0.99795	0.07564	148.7
1	-10	0.99861	0.08786	150.4

because the slotframe was sufficiently long to enable all nodes in the network to have their own unique transmission slot without collision.

Specifically, in scenario 3, which is a five-hop multi-hop environment, the packet delay difference was the largest. The nine-slot QL-TSCH scheduler averaged a packet delay of 1,841 ms, compared with the 16,155 packet delays of Orchestra. This is because the duty cycle of the node reached 101 slots, or 1,010 ms. The FTA schedulers showed the best performance in terms of packet delay. However, it should be noted that the FTA scheduler is less reliable than the scheduler of QL-TSCH owing to the occurrence of collisions and retransmissions.

B. EXPERIMENT 2: REWARD FUNCTION EVALUATION

In experiment 2, we proved that adopting negative rewards for transmission failure enables the Tx slot to be learned more efficiently, as indicated in Table 3. In a case without a negative

reward for failure to transmit, the FER is 23.19%, which is higher than all the other cases. Frequent transmission errors result in longer packet delay and lower PDR. The magnitude of the value of the negative compensation for failure did not appear to significantly affect the network performance.

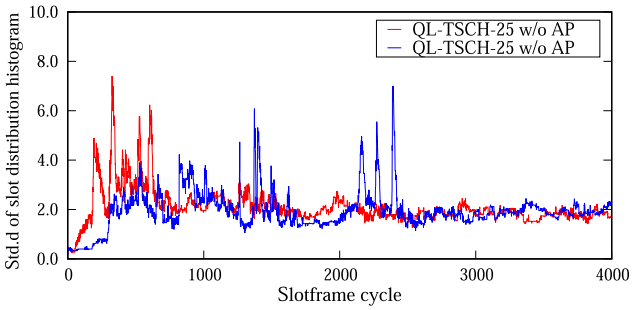

FIGURE 9. Comparison of standard deviation of slot distribution histogram over time flow with and without action peeking. Higher values imply that the schedule is biased at a specific slot offset.

C. EXPERIMENT 3: ACTION PEEKING EVALUATION

Fig. 9 shows the result of the action peeking experiment; action peeking mitigated the non-stationarity problem of an MAS. The use of action peeking enabled faster schedule convergence. The standard deviation of the slot distribution histogram was stabilized within 800 slot frame cycles, or 200 s. In contrast, when action peeking was not employed, the slot distribution was often biased even after 2,400 cycles.

VI. RELATED STUDIES

In this section, we briefly review related studies on RL for MAC.

A. RL-BASED CHANNEL BLACKLISTING

Apart from TSCH scheduling, the TSCH MAC performs channel hopping according to the hopping sequence. However, interference and fading in a specific channel have a negative effect on the TSCH network performance. This has led to the proposal of an adaptive channel selection method for TSCH to mitigate this effect [2]. A channel in which interference is expected to occur is not used for transmission. This algorithm is based on the Gittins index calculation for the multi-armed bandit problem. The authors showed that the environment can be estimated only by transmitting and receiving dozens of packets. In addition, the use of an optimal channel selection algorithm showed that the throughput of TSCH can be up to 85% of the maximum value.

B. OTHER RL-BASED MAC

Studies have applied RL to improve the performance of MAC. Along with QL methods, deep RL methods such as the deep Q-network and generative adversarial network have been investigated.

In [19], a dynamic spectrum access problem was considered to maximize network utility in a multichannel wireless network. The authors demonstrated a multiuser strategy for spectrum access that maximizes certain network utilities in a distributed form without online coordination or message exchange between users. Spectrum access problems are very expensive to compute owing to the large state space and partial observability of states. To solve this problem, the authors developed a new distributed dynamic spectrum access algorithm based on deep multi-agent RL. Unlike in the TSCH MAC, this MAC allows a sink node to simultaneously receive from multiple channels. The authors also performed an analysis from the game theory perspective of the system dynamics of the designed algorithm.

In [20], the authors introduced a secure wireless sensor network middleware (SWSNM). The SWSNM comprised two networks: a generator network for sensor nodes and a discriminator network for base stations. The former creates fake data resembling real samples and combines them with real sensor data to confuse attackers. The latter includes multiple layers that distinguish between real and fake data. The experimental results showed that this combination makes it possible to collect data securely in a wireless sensor network. Unlike conventional middleware, data can be protected from malicious or unknown attacks during transmission.

In [21], a monitoring framework based on deep learning was proposed to analyze streaming data generated by a wireless sensor network. A distinctive feature of this framework is that adaptive query refinement is performed such that the predictor can conduct the timely analysis of the wireless sensor network data. As a result, reasonably accurate query analysis results were obtained within the deadline, even if some sensor data were not synchronized in time or some data had not yet arrived.

In [22], an RL-based energy-conserving MAC protocol was proposed to extend the network life. The proposed QL-based protocol converges to a low-energy state by adjusting the MAC parameters using a trial-and-error process. This protocol is similar to the proposed QL-TSCH in that it does not need to determine the system model in advance and adapts itself to the topology and other external changes. The authors reported that nodes can reduce their energy consumption by adjusting the sleep and active periods of their radio based on the predicted traffic and transmission conditions of neighboring nodes.

In [23], an apprenticeship learning based cross-layer routing scheme was proposed for cognitive radio networks. Apprenticeship learning is a way for apprentice nodes to learn strategies from nearby expert nodes. For efficient expert node identification, the authors proposed the adaptive radius Bregman Ball model. For improving the training, multi-teacher deep Q-learning from demonstrations (MT-DQfD) was proposed. MT-DQfD accelerated the learning process by sharing demonstrations derived from multiple expert nodes. The simulation results show that the proposed cross-layer routing protocol improved the transmission quality compared with existing algorithms, while reducing the training period. In addition, newly joined nodes achieved better performance than the experts.

VII. CONCLUSIONS

Existing TSCH schedulers have suitable application scenarios for each type of scheduler, but they are less versatile. Scheduling without collisions inevitably results in low throughput, whereas the throughput of contention-based scheduling is high, but it causes too many collisions in high-traffic applications. We developed a TSCH scheduler that could be used universally regardless of the topology and data collection characteristics of the application scenario. In this paper, we proposed a multi-agent RL-based TSCH scheduling scheme that allows contention but minimizes collisions. The TSCH scheduler adopts QL to learn the most optimal Tx slot. To enhance the QL performance, we developed reward functions tailored for the TSCH scheduler. In addition, we proposed methods to address the well-known non-stationarity problem in the multi-agent RL process. The experimental results showed that the performance of the TSCH scheduler was consistently good for various applications, compared with other schedulers.

REFERENCES

[1] H. Park, H. Kim, K. T. Kim, S.-T. Kim, and P. Mah, "Frame-type-aware static time slotted channel hopping scheduling scheme for large-scale smart metering networks," *IEEE Access*, vol. 7, pp. 2200–2209, 2019.

[2] P. Li, T. Vermeulen, H. Liy, and S. Pollin, "An adaptive channel selection scheme for reliable TSCH-based communication," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Brussels, Belgium, Aug. 2015, pp. 511–515.

[3] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, St. Louis, MO, USA, Apr. 2008, pp. 377–386.

[4] *International Society of Automation Wireless Systems for Industrial Automation: Process Control and Related Applications*, Standard ISA-100.11a, ISA, 2009.

[5] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity: Why channel hopping makes sense," in *Proc. 6th ACM Symp. Perform. Eval. Wireless Ad Hoc, Sensor, Ubiquitous Netw. (PE-WASUN)*, 2009, pp. 116–123.

[6] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, Seoul, South Korea, 2015, pp. 337–350.

[7] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks," in *Proc. IEEE 23rd Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sydney, NSW, Australia, Sep. 2012, pp. 327–332.

[8] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 455–470, Dec. 2015.

[9] A. Aijaz and U. Raza, "DeAMON: A decentralized adaptive multi-hop scheduling protocol for 6TiSCH wireless networks," *IEEE Sensors J.*, vol. 17, no. 20, pp. 6825–6836, Oct. 2017.

[10] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Comput. Netw.*, vol. 67, pp. 74–88, Jul. 2014.

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

[12] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," 2018, *arXiv:1812.11794*. [Online]. Available: http://arxiv.org/abs/1812.11794

[13] P. S. Banerjee, S. N. Mandal, D. De, and B. Maiti, "RL-sleep: Temperature adaptive sleep scheduling using reinforcement learning for sustainable connectivity in wireless sensor networks," *Sustain. Comput., Informat. Syst.*, vol. 26, Jun. 2020, Art. no. 100380.

[14] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino, "Agent-based Internet of Things: State-of-the-art and research challenges," *Future Gener. Comput. Syst.*, vol. 102, pp. 1038–1053, Jan. 2020.

[15] S.-M. Hung and S. N. Givigi, "A Q-learning approach to flocking with UAVs in a stochastic environment," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 186–197, Jan. 2017.

[16] D. Schwab, Y. Zhu, and M. Veloso, "Zero shot transfer learning for robot soccer," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst. (AAMAS)*, Stockholm, Sweden, 2018, pp. 2070–2072.

[17] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intell. Transp. Syst.*, vol. 11, no. 7, pp. 417–423, Sep. 2017.

[18] K. Tuyls and G. Weiss, "Multiagent learning: Basics, challenges, and prospects," *Ai Mag.*, vol. 33, no. 3, pp. 41–52, 2012.

[19] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[20] R. A. Alshinina and K. M. Elleithy, "A highly accurate deep learning based approach for developing wireless sensor network middleware," *IEEE Access*, vol. 6, pp. 29885–29898, 2018.

[21] K. Lee, S. Lee, Y. Kim, and C. Lee, "Deep learning-edreal-time query processing for wireless sensor network," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 5, May 2017.

[22] 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. 29355–29364, 2019.

[23] Y. Du, L. Xue, Y. Xu, and Z. Liu, "An apprenticeship learning scheme based on expert demonstrations for cross-layer routing design in cognitive radio networks," *AEU-Int. J. Electron. Commun.*, vol. 107, pp. 221–230, Jul. 2019.

HUIUNG PARK was born in Seoul, South Korea, in 1989. He received the B.S. degree in electronics and communication engineering from Hanyang University, Seoul, in 2012, and the Ph.D. degree in ICT from the University of Science and Technology, Daejeon, South Korea, in 2020.

He is currently a Postdoctoral Researcher with the Electronics and Telecommunication Research Institute (ETRI), South Korea. He is one of the developers of NanoQplus operating system. His research interests include machine learning, industrial-scale wireless sensor networks, firmware-over-the-air for the Internet of Things, and AI for UAV applications.

HAEYONG KIM received the B.S. and M.S. degrees in computer science and engineering from Seoul National University, in 2004 and 2006, respectively.

He is currently a Researcher with the Electronics and Telecommunication Research Institute (ETRI), South Korea. He is one of the developers of NanoQplus operating system. His research interests include lightweight operating systems, embedded software, and sensor networks.

SEON-TAE KIM (Member, IEEE) received the B.S. degree from the Department of Electrical and Electronics Engineering, KAIST, in 1997, the M.S. degree from Seoul National University, in 2000, and the Ph.D. degree from Korea University, South Korea, in 2012.

He joined the Real-Time Multimedia Team, Electronics and Telecommunication Research Institute (ETRI), South Korea, in February 2000. Since 2011, he has been the Director with the Department of Embedded SW Research. His research interests include video compression, multimedia streaming, image processing, lightweight OS, and sensor networks.

PYEONGSOO MAH received the M.S. degree in computer science and engineering from the City University of New York, USA, in 1992, and the Ph.D. degree from Wright State University, USA, in 1995.

Since 1996, he has been a Principal Researcher with the Electronics and Telecommunication Research Institute (ETRI). His research interests include lightweight operating systems, embedded software, and the IoT systems.