DQN-Based Directional MAC Protocol in Wireless Ad Hoc Network in Internet of Things

Namkyu Kim, Woongsoo Na, Demeke Shumeye Lakew, Nhu-Ngoc Dao, and Sungrae Cho

***Abstract*—The use of directional antennas in high-frequency bands (e.g., millimeter-wave) is essential to support applications requiring high throughput and low latency. However, communications using directional antennas require intricate scheduling by a central coordinator to avoid collision and deafness problems. Thus, in this study, we propose a directional medium access control (DMAC) protocol based on a deep *Q*-network (DQN) framework wireless ad hoc networks (WANETs) for Internet of Things (IoT). In our model, even though there is no central coordinating unit (e.g., edge/cloud server), each IoT device can intelligently avoid the collision and deafness through its learning agent. In addition, to maximize the throughput, we design a reinforcement learning (RL) architecture and propose a DQN-based DMAC such that each IoT device intelligently selects the time-slot and transmitting beam without any central coordinator. The proposed schemes are evaluated using carrier-sense multiple access (CSMA) and adaptive learning-based DMAC (AL-DMAC) protocols. The evaluation results reveal that the proposed double DQN scheme outperforms the existing schemes by approximately 54.1% and 57.2% in terms of the throughput.**

***Index Terms*—Deep *Q*-network (DQN), deep reinforcement learning (DRL), directional medium access control (MAC).**

I. Introduction

OWING to the saturation problems faced in frequencies up to 6 GHz, wireless communication using millimeter waves (mmWave), such as the fifth-generation mobile communication, has been proposed and commercialized in Internet of Things (IoT). The IEEE 802.11ad and 802.11ay standards are representative protocols that use the unlicensed mmWave band and these standards use directional communication for high-data rates [1], [2]. In addition, they support the ad hoc mode in which nodes are operated without a central coordinating unit. As the ad hoc mode is possible without preinstalled communication infrastructure, it is a promising option for device-to-device (D2D) network and disaster-area communications using IoT devices.

Compared to omnidirectional communication, directional communication has many benefits. For instance, it can transmit signals to a longer distance than those of the omnidirectional communication with the same transmission power. In addition, it enhances the channel reuse by using only a partial degree of omnidirectional communication. Despite its advantages, directional communication has a problem of deafness, which arises because of the blocked beam during transmission/reception, resulting in severely reducing the overall network capacity. Therefore, a medium access control (MAC) protocol for directional communication is necessary for achieving high performance by managing the deafness problem as well as collision (interference).

On the other hand, with the rapid evolution of computational capacity of communication nodes, the deep learning approach has been adapted to resolve various wireless communication challenges and requirements, nowadays [3], [4], [5]. For instance, machine learning-based scheduling techniques have been emerged as one of the methods to efficiently schedule communication without a central coordinator, such as edge/cloud server. In particular, adapting deep reinforcement learning (DRL) technique to the wireless MAC protocol is being actively studied [6], [7], [8]. In these schemes, each communication IoT node can be considered as an agent, and reinforcement learning (RL) architectures were proposed for the formulated problems (i.e., maximizing throughput or fairness). Thus, each IoT node learns the network environment to maximize overall network throughput or fairness and decides whether to transmit in a specific time-slot or not. However, most of these studies assumed omnidirectional communication, and the deafness problem was not considered in [7] that assumed directional communication. Thus, in this study, we investigate the RL structure that maximizes network throughput by avoiding deafness and collision in a directional wireless ad hoc networks (WANETs) for IoT. Besides, our RL architecture maximizes network throughput by considering spatial reuse.

The major technical contributions of this study can be summarized as follows.

1) *RL-Based Directional Medium Access Control (DMAC) System Architecture*: We proposed a new system model

Manuscript received 3 June 2023; revised 22 September 2023; accepted 20 November 2023. Date of publication 4 December 2023; date of current version 26 March 2024. This work was supported in part by the Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2019R1A6A1A03032988, and in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2023-RS-2022-00156353. *(Corresponding authors: Woongsoo Na; Sungrae Cho.)*

Namkyu Kim is with KT Corporation, Seoul 06765, Republic of Korea (e-mail: kim.namkyu@kt.com).

Woongsoo Na is with the Division of Computer Science and Engineering, Kongju National University, Cheonan 31080, Republic of Korea (e-mail: wsna@kongju.ac.kr).

Demeke Shumeye Lakew and Sungrae Cho are with the School of Software, Chung-Ang University, Seoul 156-756, Republic of Korea (e-mail: srcho@cau.ac.kr).

Nhu-Ngoc Dao is with the Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea (e-mail: nndao@sejong.ac.kr).

Digital Object Identifier 10.1109/JIOT.2023.3338562

and RL architecture for throughput maximization of the DMAC protocol in WANETs for IoT. Based on the characteristics of RL that an agent can improve its performance by interacting with the environment, the overall network strives to achieve better performance by agent-environment interaction.

2) *Q-Learning Based MAC Protocol for Directional IoT Networks*: Based on the RL architecture and Q-learning algorithm with the carrier-sense multiple access (CSMA) mechanism we proposed deep Q-network (DQN) frameworks for DMAC (i.e., DDMAC and DDDMAC).
3) *Smart Beam Scheduling Without a Central Coordinator*: By using DRL, we resolve the scalability problem of the classic Q-learning. The proposed scheme attempts to select not only the time-slot but also transmit beam to avoid collision and deafness without any central coordinator.
4) *Maximizing Spatial-Resource*: In addition, our proposed RL architecture considered spatial reuse by finding simultaneous transmittable node pairs in one time-slot.
5) *Next-Generation Domain-Free MAC Protocol*: The proposed scheme can be used not only in the WSN-IoT environment but also domain-independently. In particular, in the 6G environment, the proposed scheme can be used as a wireless access technology for the next-generation mobile communication system as the utilization of mmWave and THz bands and network intelligence become essential.
6) *Outstanding Performance Compared to Existing DMAC Schemes*: We demonstrate that the proposed scheme outperforms the existing DMAC protocols, CSMA-based DMAC, and adaptive learning-based DMAC (AL-DMAC) in terms of the aggregate throughput and latency by. Furthermore, by simple modification, DDDMAC demonstrates a higher aggregate throughput and lower latency than the existing DMAC schemes.

The remainder of this article is organized as follows. In Section II, we review the literature on RL approaches to MAC protocols. In Section III, we introduce the system model for DMAC in WANETs for IoT. The RL architecture for DMAC is proposed in Section IV. In Section V, we propose the Q-learning algorithm and the DRL framework for throughput maximization. A performance evaluation is presented in Section VI. Finally, we present the conclusions in Section VII.

II. Related Work

A. Deep Reinforcement Learning

Using RL, an agent can learn its policy by interacting with the environment [9]. Q-learning is regarded as a powerful RL method because it can estimate the Q-value of the state and action pair without prior knowledge of the model. The RL agent estimates the Q-value through experience, which is an outcome of the agent-environment interaction. DRL methods based on DQN [10], [11] and deep deterministic policy gradient (DDPG) [12] have been actively researched in the field of wireless communication. DQN attempts to approximate the Q-function, which estimates the discounted cumulative reward based on the state and action pair. In contrast, DDPG attempts to approximate the policy itself and generates the value of the action based on the state. Hasselt et al. [13] proposed a double DQN method to resolve the problem of DQN nonuniformly overestimating the action value. Similar to DQN, the double DQN method uses a policy network and a target network. While the DQN method selects the next action that maximizes the Q-value and calculates the values of the next state and action using only the target network, the double DQN method decouples the action selection from the value evaluation. The latter approach can minimize overestimation and achieve a higher performance than that of the existing DQN approach. The dueling DQN method was proposed in [14]. In this method, the Q-value of a given state and action are calculated by estimating the value of a given state and the advantage of the state-dependent action. By separating the estimation of state and action, the dueling DQN can learn which state is more valuable, excluding the effect of each action for the state.

B. Reinforcement Learning-Based MAC

Most of the initial RL-based MAC schemes were proposed based on the ALOHA MAC protocol because of its simplicity. For instance, ALOHA-Q, which is a MAC protocol for wireless sensor networks, was proposed in [15] and [16]. The ALOHA-Q node learns the Q-value to determine the time-slot to transmit a packet by reflecting the experience of the transmission success. ALOHA-Q uses stateless Q-learning so that the Q-value depends only on the action (i.e., time-slot to transmit the packet). The protocol assumes that each frame is divided into slots, and the node transmits a packet to one of the slots in each frame.

Based on ALOHA-Q, the ALOHA-quantitative tree (QT) and ALOHA-QTF were proposed in [17]. While ALOHA-Q selects only the time-slot to transmit, ALOHA-QT and ALOHA-QTF select both the time-slot and the number of slots per frame. Moreover, ALOHA-QTF considers fairness by estimating the number of active nodes in the same network. To estimate the number of active nodes, each node must transmit its node ID and retain the sliding window containing the node ID. Even though those protocols can dynamically determine the frame length, there still exists a problem in that the depth of the policy tree must be set appropriately, which significantly affects convergence and spatial complexity.

ALOHA-Q has been expanded to UW-ALOHA-Q, which is a MAC protocol for underwater acoustic sensor networks [18]. Unlike other ALOHA-Q-based protocols, UW-ALOHA-Q enables asynchronous slotted frame-based communication among nodes by providing a sufficiently large guard interval for a slot. In addition, to resolve the problem caused by asynchronous slotted frame-based communication, uniform-random back-off scheme was presented. The sensor node delays the next start of the frame if a collision occurs. Because the Q-value is also updated when a collision occurs, it is likely to fail to converge.

Yu et al. [19], [20], [21] researched DRL-based MAC (DLMA), where DLMA nodes determine whether or not to

transmit in every time-slot based on the previous M slots history. In particular, a DLMA for heterogeneous wireless networks was proposed in [20] and [21]. In these methods, the DLMA nodes use the DRL architecture to learn the policy to maximize the sum throughput or α-fairness. When α is close to 0, α fairness works similarly to the sum throughput maximization. When α is close to ∞, it works similar to maximizing the minimum of throughput. A deep residual network (ResNet) was used to achieve near-optimal performance under various scenarios. The action that maximizes the approximated Q-value is selected for throughput maximization, and the action that the α-fairness function provides an approximated Q-value is selected for fairness maximization. DLMA for CSMA (CS-DLMA) for heterogeneous wireless networks was proposed in [19]. Unlike traditional CSMA protocols, CS-DLMA determines whether to transmit for each time-slot without an exponential back-off. Because the transmitting packet occupies multiple slots, a reward is provided after transmission, not every slot. The authors introduced reward backpropagation schemes, which divided the received reward into rewards of past time-slots to transmit. A DLMA for backscatter communication was proposed in [8]. Backscatter devices and Wi-Fi stations coexist with associated APs, which is considered as an agent. Action was defined as a reservation strategy for transmission by backscatter devices. Each AP learns polices without coordination with other APs. For simplicity, the aforementioned RL approaches for wireless MAC assumed that all nodes are in the communication area. Thus, they excluded the hidden node problem, which can occur when a node is out of the communication range of other nodes.

An AL-DMAC for millimeter-wave wireless networks was proposed in [7]. In this method, for each slot in the frame, the node determines whether to transmit or receive based on the transmission and reception probabilities as well as slot index. The probabilities are updated whenever a transmission or reception action is performed. Instead of an exponential back-off, a linear back-off algorithm was adopted for the AL-DMAC. When the transmission is successful, the contention window (CW) size is reduced by a certain number of slots, and when the transmission fails, the CW size is increased by the same number of slots. However, AL-DMAC only considers the time-slot and has no consideration of where to transmit. If the node has packets for various destinations, it can achieve a better performance by determining the destination node to transmit first.

C. Beam Management for IoT System

In recent studies, beam management techniques for D2D communication in IoT systems have been proposed. In [22], an optimal beam selection algorithm was proposed to maximize user throughput in the mmWave IoT system. In their research, they proposed an optimization algorithm based on machine learning by designing user scheduling and beam selection problems. In [23], a beam scheduling technique was proposed to solve the problem of beam blockage, which is an obstacle to communication in the mmWave environment. They designed a Markov decision process (MDP)-based model and solved the problem through dynamic optimization techniques. Su et al. [24] proposed a UAV beam management technique to optimize energy efficiency in UAV assisted IIoT networks. In their research, they designed a nonconvex problem with the goal of optimizing not only beam power, pattern, and scheduling but also the position of the UAV, and this was solved through a multiobjective evolutionary algorithm based on decomposition algorithm. However, what these studies have in common is that they are all focused on beam optimization by a central-coordinator and are difficult to apply in an ad hoc IoT network environment. It can be seen that there are very few studies on beam management in an ad hoc environment [25], [26], and this study is differentiated in that it optimizes the network through optimal beam selection without a central coordinator.

III. System Model

In our system model, we assume that each IoT node is equipped with switched-beam antennas with M beam sectors and each beam section has the same angle. During transmission or reception, all beams, except the beam used for directional transmission or reception, are blocked. In addition, we assume that the position of each IoT node is fixed during the communication process. In our model, there is no need to exchange complete information among nodes. However, to focus on the data transmission side of DMAC, we assume that each IoT node configures the beam pair information of the neighboring IoT node before communication. IoT nodes can configure beam-pair information using beam-table caching [27].

A. Deafness Problem

In traditional wireless communication, collision is the major factor that degrades the network capacity. This is because, unlike wired communication, IoT nodes using wireless communication links cannot sense collisions during transmission. Moreover, in directional communication, deafness is a critical problem that affects the overall performance of the network. It occurs when the sender IoT node transmits data to the blocked beam direction owing to communication with another directional beam. Fig. 1 illustrates an example scenario of the deafness problem when the number of beam directions is four. As shown in the figure, nodes A and B are communicating with each other. Nodes A and B block all beams, except those in communication, i.e., beams 1 and 3, respectively. Meanwhile, node C transmits directional data (DDATA) to the blocked beam (beam 4) of node B. However, due to the blocked beam, node B cannot listen to DDATA from node C. Therefore, node C receives directional acknowledgment (DACK) timeout and attempts to retransmit DDATA.

B. Problem Formulation

The notations used in this article are explained in Table I. Let $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ be the set of IoT nodes in the network, where N denotes the number of IoT nodes. IoT node n_i has an antenna beam index vector $\mathbf{A}_i$, which can be described as $\mathbf{A}_i = [a_{i,1} \quad a_{i,2} \quad \cdots \quad a_{i,N}]$, where $a_{i,j}$ denotes

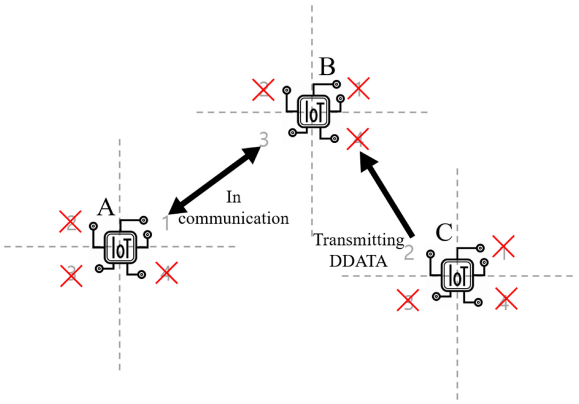

Fig. 1. Example scenario of deafness problem.

TABLE I
SUMMARY OF PARAMETERS AND VARIABLES

Symbol	Description
$\mathcal{N}, n_i$	Set of IoT nodes, ith IoT node
$\mathbf{A}_i, a_{i,j}$	Beam index vertor of a_i, beam index from n_i to n_j
$r_{i,j,t}$	Achievable rate of the channel from n_i to n_j at time t
$\mathbf{I}_{i,j,t}$	Transmission indicator variable between n_i and n_j at time t
$s_{i,k}$	State of n_i at kth slot
$\mathbf{q}_{i,k}$	Set of queued packets of n_i at kth slot
$c_{i,k}$	Sensed channel state of n_i at kth slot
$x_{i,k}$	Retransmission count of n_i at kth slot
$r_{i,k}$	Reward value of n_i at kth slot
ζ	Negative reward factor
t, γ	Current time slot, discount factor

the beam index from n_i to n_j. We assumed that IoT nodes share $\mathbf{A}$ by beam caching algorithm [27] and the overall antenna beam index table $\mathbf{A}$ is defined as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_N \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix}. \quad (1)$$

The achievable rate of the directional channel from n_i to n_j at time t is denoted as follows:

$$r_{i,j,t} = B \log_2 \left(1 + \frac{P_t G_t G_r \left(\frac{\lambda}{4\pi d_{i,j}} \right)^{\rho}}{n_0 B + I} \right) \quad (2)$$

where B is the channel bandwidth, P_t is the transmission power, G_t is the directional beam gain of the transmitter, G_r is the directional beam gain of the receiver, λ is the radio frequency wavelength, $d_{i,j}$ is the distance between IoT node n_i and IoT node n_j, ρ is the path loss coefficient, n_0 is the noise power spectral density, and I is the interference. Thus, the objective function that maximizes the throughput is formulated as follows:

$$\begin{aligned} &\text{maximize} \quad \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} r_{i,j,t} \cdot \mathbf{I}_{i,j,t} \\ &\text{subject to} \quad \sum_{j=1}^{N} \mathbf{I}_{i,j,t} \leq 1 \quad \forall i, t \qquad \text{(C1)} \\ &\qquad\qquad \sum_{i=1}^{N} \mathbf{I}_{i,j,t} \leq 1 \quad \forall j, t \qquad \text{(C2)} \\ &\qquad\qquad \sum_{i=1}^{N} \mathbf{I}_{i,k,t} + \sum_{j=1}^{N} \mathbf{I}_{k,j,t} \leq 1 \quad \forall k, t \qquad \text{(C3)} \\ &\qquad\qquad \mathbf{I}_{i,j,t} = 0 \quad \forall t, i = j \qquad \text{(C4)} \end{aligned} \quad (3)$$

where $\mathbf{I}_{i,j,t}$ is an indicator variable. If n_i schedules to transmit to n_j at time t, then $\mathbf{I}_{i,j,t} = 1$, otherwise $\mathbf{I}_{i,j,t} = 0$. Constraint (C1) represents each IoT node should transmit to at most one node simultaneously. Constraint (C2) indicates that the node should receive at most one packet from the other nodes to avoid collision and deafness. Constraint (C3) stipulates that the IoT node cannot transmit and receive simultaneously. Constraint (C4) indicates that the node does not transmit to itself.

However, because a central coordinating unit or AP does not exist in WANETs for IoT, a medium access strategy cannot be formulated in a centralized manner. Therefore, each IoT node should determine its own medium access strategy. Because RL improves policies by interacting with the environment, this study formulates and resolves this problem using an RL-based approach.

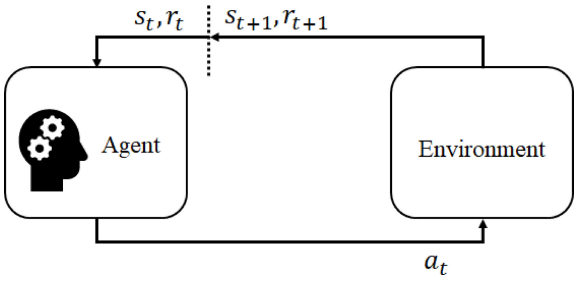

Fig. 2. Agent-environment interaction in RL.

IV. RL ARCHITECTURE

In this section, we introduce the RL architecture of DMAC. In RL, the agent learns its policy by interacting with the environment as shown in Fig. 2. The agent selects an action by considering the current state. It then receives the immediate reward and the next state as a consequence of selecting the action. The ultimate objective of the agent is to maximize the return or sum of discounted rewards, which can be formulated as follows:

$$R = \sum_{m=1}^{\infty} \gamma^{m-1} r_{t+m} \quad (4)$$

where t and γ denote the current time slot and discount factor, respectively. In this study, the IoT node is considered as the agent, and the network communication system is considered as the environment.

We define the main components of a MDP, such as state, action, and reward function for throughput maximization. We assume that the time horizon is divided into frames, and each frame is further divided into τ^{slot} slots. Based on the

CSMA mechanism, each sender node transmits DDATA after sensing the channel. If the receiver node receives the DDATA destined to itself, it sends the DACK to the sender node, which then determines whether the transmission is successful by receiving the DACK. If the sender node does not receive the DACK timeout duration, a DACK timeout occurs, and the transmission is regarded as a failure. Let D_i denotes the set of destination nodes of n_i. The destination nodes in D_i are in the communication range from n_i and receive DDATA from n_i. Each sender node has more than one destination node. The packet for each destination node is queued at the start of the frame. Therefore, the sender node should transmit a DDATA to each destination node to maintain a stable transmission queue. Each IoT node determines its action at the start of the slot. When sensing takes one slot, the transmission takes multiple slots. The reward is provided after completion of the sensing or transmission. Therefore, if the action is to transmit, the reward will be provided multiple slots after the action is selected.

1) *State:* The state of IoT node n_i at the kth slot is defined as $s_{i,k} = \{\mathbf{q}_{i,k}, c_{i,k}, x_{i,k}\}$, where $\mathbf{q}_{i,k} = \{q_{i,k,1}, q_{i,k,2}, \ldots, q_{i,k,|D_i|}\}$ is the set of queued packets for each destination node. $c_{i,k} \in \{\text{NONE}, \text{BUSY}, \text{IDLE}\}$ is the sensed channel state. When the channel is not sensed in the previous slot, $c_{i,k} = \text{NONE}$. $x_{i,k}$ denotes the retransmission count.
2) *Action:* The IoT node can act to sense the channel (SENSE) or transmit DDATA to one of the destination nodes. Therefore, the number of actions for n_i is $|D_i|+1$. Existing RL models for DMAC use actions to determine whether to transmit or not. Unlike those RL models, our model not only determines whether to transmit or not but also determines where to transmit.
3) *Reward:* For throughput maximization, reward is provided as the transmission length in case of the transmission success. Note that the reward is given after the DACK timeout or DACK reception. The reward is not given during DDATA transmission. For example, when the transmission takes five slots and is successful, the sender node receives (+5) as the reward. This can be formulated as follows:

$$r_{i,k'} = \begin{cases} -\zeta(k'-k), & \text{if } \mathbf{q}_{i,k} = \mathbf{q}_{i,k'} \\ k'-k, & \text{otherwise} \end{cases} \tag{5}$$

where k' is the slot when the DACK is received or the DACK timeout occurs by the sender node. $\zeta (0 < \zeta < 1)$ is a negative reward factor devised to encourage transmission.

We provide an example scenario for the proposed RL architecture as depicted in Figs. 3–5. From Fig. 3, n_3 is the destination node of n_1 and n_2. To obtain a positive reward, n_1 and n_2 successively transmit DDATA to n_3. Assume that n_2 is located at the center between n_1 and n_3 [Fig. 6(a)], hence n_2 can receive the signals from n_1 and n_3. From the perspective of n_3, n_1 and n_2 are located in the same beam sector. After sensing, n_1 selects the transmit action to transmit to n_3 at slot $k = 1$. Then, n_3 receives DDATA from n_1 and transmits DACK to n_1 after a certain amount of time, which is equivalent to a short interframe space. As n_1 receives DACK from n_3, n_1 receives a reward of (+5). Note that n_2 can listen to DDATA and DACK between n_1 and n_3. n_2 then selects an action to transmit to n_3 at slot $k = 7$. After receiving the DDATA from n_2, n_3 transmits DACK to n_2. Similar to the case of n_1, n_2 receives a reward of (+5). Fig. 4 depicts a collision scenario assuming that nodes are located as depicted in Fig. 6(a). As n_1 and n_2 transmit DDATA to n_3 simultaneously at slot $k = 1$, DDATA collision happens. Because n_3 cannot completely receive any DDATA, DACK is not transmitted from n_3. In addition, because a DACK timeout occurs, n_1 and n_2 receive (-5ζ) rewards. Moreover, as n_2 and n_3 are located in the same beam sector from the perspective of n_3, collision occurs again between slots $k = 9$ and $k = 10$, even though n_2 transmits faster than n_1. From this experience, n_1 and n_2 will attempt to avoid collision to prevent negative reward.

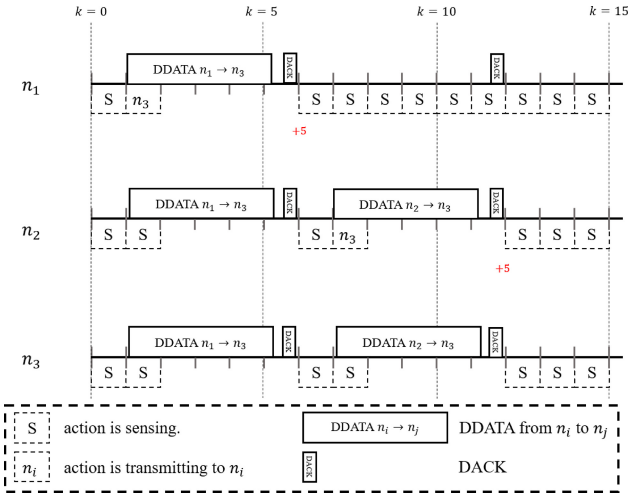

Fig. 3. Example transmission scenario of RL architecture.

Fig. 5 depicts the deafness scenario assuming that n_1 and n_2 are located in different beam sectors from the perspective of n_3, and the distance between n_1 and n_2 are longer than directional transmission range [Fig. 6(b)]. Thus, n_2 cannot hear the signal exchanged between n_1 and n_3. While n_1 is transmitting DDATA to n_3, n_2 transmits DDATA to n_3 at slot $k = 2$. Because n_3 is already receiving DDATA from n_1, DDATA transmitted from n_2 is blocked, and only n_1 receives DACK after transmission. Therefore, to prevent a negative reward, n_2 should transmit to another destination node or wait.

V. DQN-Based Framework

A. Q-Learning Algorithm

This section proposes a Q-learning-based algorithm for the DMAC protocols. According to the traditional Q-learning framework, the Q-value is updated through interaction with the environment

$$\mathcal{Q}^{\text{new}}(s,a) = \mathcal{Q}^{\text{old}}(s,a) + (1-\beta)\left\{r + \gamma \max_{a'} \mathcal{Q}^{\text{old}}(s',a') - \mathcal{Q}^{\text{old}}(s,a)\right\} \tag{6}$$

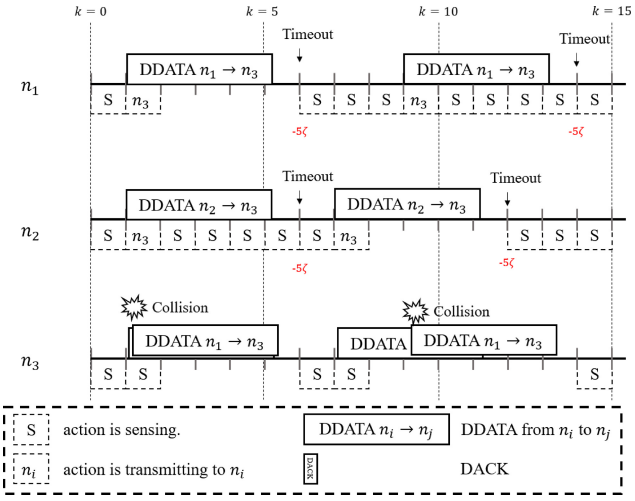

Fig. 4. Example collision scenario of RL architecture.

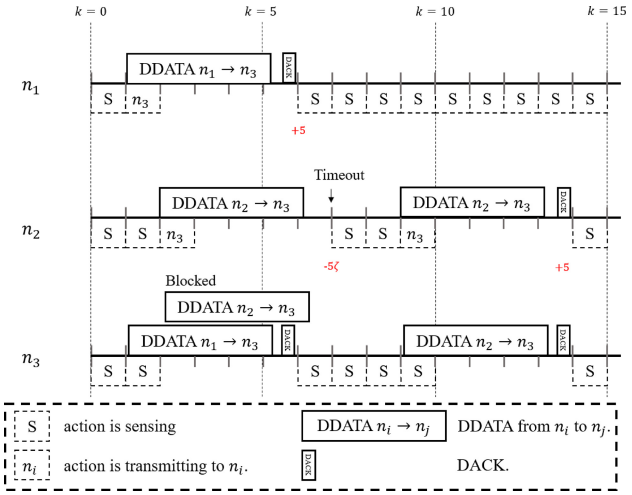

Fig. 5. Example deafness scenario of RL architecture.

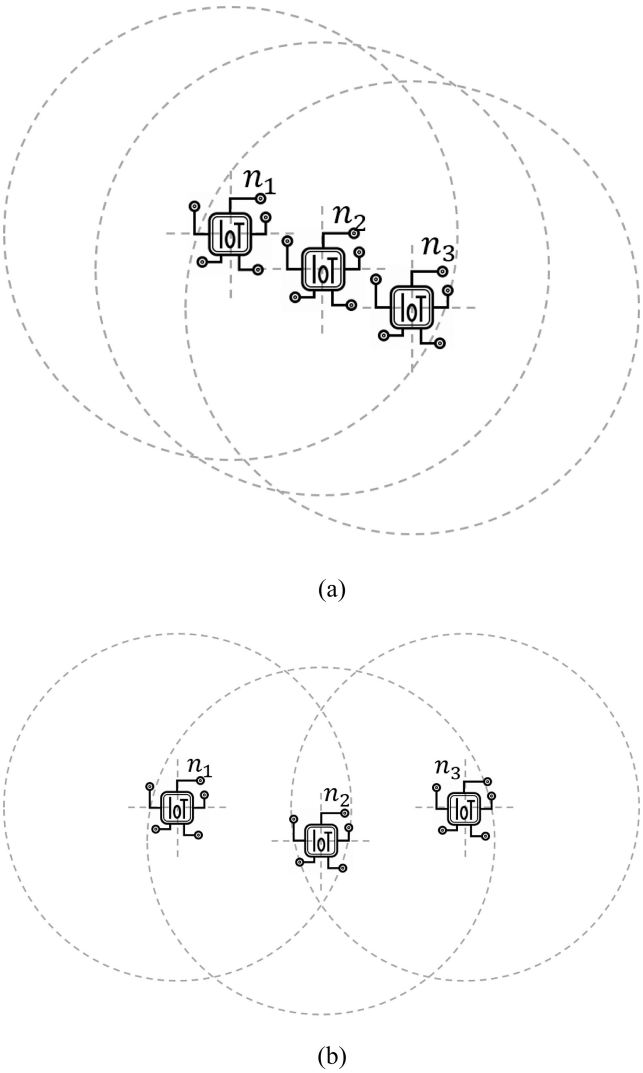

Fig. 6. Example node deployments for Figs. 3–5. (a) n_2 is located at the center between n_1 and n_3. n_1 and n_3 are located in directional transmission range of each other. Thus, n_2 can hear the signal exchanged between n_1 and n_3. (b) n_1 and n_2 are located in different beam sectors in perspective of n_3. The distance between n_1 and n_2 is longer than the directional transmission range. Thus, n_2 cannot hear the signal exchanged between n_1 and n_3.

where β is the step size; γ is a discount factor; and s, a, s', and r are the state, action, next state, and reward, respectively.

Algorithm 1 describes the pseudo code of the Q-learning algorithm for the DMAC protocol. However, instead of relying only on Q-learning, we used a hybrid method based on the CSMA scheme. In particular, action selection by Q-learning is employed only if the sensed channel state is IDLE. If the channel is not sensed or is BUSY, the next action is determined as SENSE, and Q-learning is not used. At the start of each slot, the IoT node selects the action based on the ϵ-greedy method. The value of ϵ decreases as the learning step proceeds. Equation (7) formulates the epsilon decay method

$$\epsilon = \epsilon^{\text{end}} + \frac{\epsilon^{\text{start}} - \epsilon^{\text{end}}}{e^{\frac{c}{\epsilon^{\text{decay}}}}} \tag{7}$$

where c denotes the step count, which increases when the Q-value is updated. As the step count increases, the value of ϵ decreases from ϵ^{start} to ϵ^{end}. After an action is taken, the IoT node evaluates the reward, next slot, next state, and updates its Q-value. When the last slot of the frame is completed, the first slot of the next frame begins.

B. DQN-Based Algorithm

As traditional Q-learning algorithms require Q-values for every state and action pair, the space complexity and time taken to converge for every Q-value increases exponentially as the number of states increases. For example, when the number of destination nodes increases by 1, the required size of the Q-table will be more than twice. To resolve this challenge, a deep neural network (DNN) has been utilized as a universal approximation tool such that the DQN approach can obtain near-optimal policies through multiple training steps. In this section, we introduce a DQN-based framework for DMAC.

Fig. 7 depicts the architecture of the DQN-based framework for DMAC. The DNN networks are classified into policy and target networks, denoted by θ_i^P and θ_i^T, respectively. The policy network estimates the Q-values of actions based on the state. The target network is used for calculating the temporal difference and has the same DNN architecture as that of the policy network. By fixing the parameter of the target network

Algorithm 1 Q-Learning-Based DMAC Algorithm

Input: $\epsilon^{start}, \epsilon^{end}, \epsilon^{decay}, \beta, \gamma, s_{i,0}$
Initialize: Q-value $\mathcal{Q}_i(\cdot,\cdot) \leftarrow 0$, frame index $t \leftarrow 0$, step count $c \leftarrow 0$
while True **do**
  $k \leftarrow 0$
  while $k < \tau^{slot}$ **do**
    if $c_{i,k} =$ IDLE **then**
      Calculate ϵ according to Eq. (7).
      Uniformly sample v in range $[0, 1)$.
      if $v < \epsilon$ **then**
        $a_{i,k} \leftarrow$ random action
      else
        $c \leftarrow c + 1$
        $a_{i,k} \leftarrow \arg\max_{a_i}(s_{i,k}, a_i)$
      end if
    else
      $a_{i,k} \leftarrow$ SENSE
    end if
    Take action $a_{i,k}$
    Observe $r_{i,k}, k', s_{i,k'}$
    Update $\mathcal{Q}_i(s_{i,k}, a_{i,k})$ according to Eq. (6):
    $k \leftarrow k'$
  end while
  $t \leftarrow t + 1$
end while

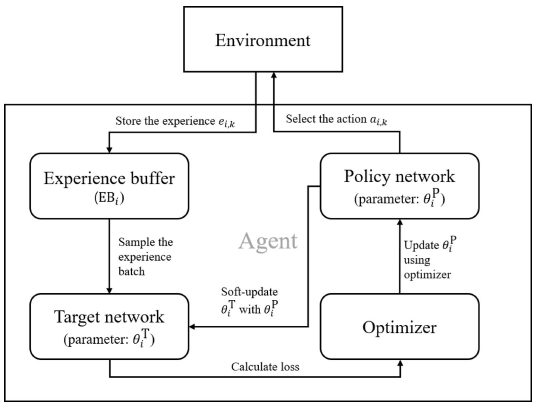

Fig. 7. Architecture of DQN-based framework for DMAC.

for G steps, the policy network can be updated in a stable manner. After receiving the reward and the next state from the environment, the IoT node stores the transition experience $e_{i,k} = (s_{i,k}, a_{i,k}, r_{i,k'}, s_{i,k'})$ in the experience buffer EB$_i$, which has limited capacity and stores transition experiences based on the first-in-first-out policy. Employing the experience buffer helps the IoT node to increase the efficiency of experiences. The policy network in the DQN framework updates its parameter by minimizing the temporal difference, which is defined as follows:

$$\delta(s, a; \theta) = r + \gamma \max_{a'} \mathcal{Q}\left(s', a'; \theta^T\right) - \mathcal{Q}\left(s, a; \theta^P\right). \quad (8)$$

Instead of directly minimizing the temporal difference, the Huber loss is used as a loss function in the DQN framework

Algorithm 2 DQN-Based DMAC Algorithm

Input: $\epsilon^{start}, \epsilon^{end}, \epsilon^{decay}, \alpha, \gamma, \omega, s_{i,0}$
Initialize: θ_i^{P}, frame index $t \leftarrow 0$, step count $c \leftarrow 0$
$\theta_i^{\mathrm{T}} \leftarrow, \theta_i^{\mathrm{P}}$
while True **do**
  $k \leftarrow 0$
  while $k < \tau^{\mathrm{slot}}$ **do**
    if $c_{i,k} =$ IDLE **then**
      Calculate ϵ based on Eq. (7).
      Uniformly sample v in range $[0, 1)$.
      if $v < \epsilon$ **then**
        $a_{i,k} \leftarrow$ random action
      else
        $c \leftarrow c + 1$
        $a_{i,k} \leftarrow \arg\max_{a_i}(s_{i,k}, a_i; \theta_i^{\mathrm{P}})$
      end if
    else
      $a_{i,k} \leftarrow$ SENSE
    end if
    Take action $a_{i,k}$
    Observe $r_{i,k}, k', s_{i,k'}$
    Store the experience $e_{i,k} = (s_{i,k}, a_{i,k}, r_{i,k'}, s_{i,k'})$
    Sample experience batch B from EB$_i$
    Update θ_i^{P} according to Eq. (10)
    if c % $\tau^{target} = 0$ **then**
      $\theta_i^{\mathrm{T}} \leftarrow \omega\theta_i^{\mathrm{P}} + (1 - \omega)\theta_i^{\mathrm{T}}$
    end if
    $k \leftarrow k'$
  end while
  $t \leftarrow t + 1$
end while

based on experience batch B as

$$L = \frac{1}{|B|} \sum_{(s,a,r,s') \in B} L[\delta(s, a; \theta)]$$

$$\text{where } L[\delta(s, a; \theta)] = \begin{cases} \frac{1}{2}[\delta(s, a; \theta)]^2, & \text{if } |\delta(s, a; \theta)| < 1 \\ |\delta(s, a; \theta)| - \frac{1}{2}, & \text{otherwise.} \end{cases} \quad (9)$$

The experience batch, which consists of subsets of experiences, is constructed by randomly sampling experiences from EB$_i$. The Huber loss function prevents the loss value from becoming too large. Thus, the parameter of the policy network is updated as follows:

$$\theta_i^P = \theta_i^P + \alpha \nabla_{\theta_i^P} L \quad (10)$$

where α denotes the learning rate, and the Adam optimizer is employed to update the parameter.

The pseudocode for the DQN-based framework is shown in Algorithm 2, which has a procedure similar to that of Algorithm 1. As mentioned earlier, action in the DQN model is selected only if the sensed channel state is IDLE. After receiving the reward, next slot, and next state, the experience is stored in EB$_i$. If the size of the experience buffer is larger than the batch size, the experience batch is constructed, and the parameters of the policy network are updated using

the optimizer. The parameter of the target network is softly updated as the parameter of the policy network for every τ^{target} training steps. In particular, the parameter of the target network is updated as follows:

$$\theta^T = \omega\theta^P + (1-\omega)\theta^T \tag{11}$$

where ω is the soft update ratio.

C. Double DQN

Unlike the DQN method, the double DQN method decouples the action selection and value estimation to compute the temporal difference. Consequently, the DQN can be changed to the double DQN with minor modifications. As the DQN scheme already uses the policy and target networks, the only modification required is to modify the formulation of the temporal difference from (8) as

$$\delta(s,a;\theta) = r + \gamma\mathcal{Q}\left(s', \arg\max_{a'} \mathcal{Q}\left(s',a';\theta^P\right);\theta^T\right) - \mathcal{Q}\left(s,a;\theta^P\right). \tag{12}$$

Here, to compute the temporal difference, the next action is selected by the policy network, i.e., θ^P, and the value of the next state and next action are evaluated by the target network (i.e., θ^T). Accordingly, we can alleviate the problem of DQN, which is overestimating the Q-value on a given state and action pair.

VI. Performance Evaluation

In this section, we present the performance evaluation of the proposed scheme conducted by network simulations. Simulations were performed using Python 3.8.6 and PyTorch 1.7.0. In our simulator, we have implemented a DQN/DDQN nodes with experience buffer, policy/target network, and optimizer are implemented using PyTorch framework. In addition, implemented DQN/DDQN node receives the reward and monitors the state. Based on the received reward and state, each node updates the policy network and determines the action for frame to be transmitted. Fig. 8 shows our implemented simulator architecture. Each node is implemented as an instance of an Agent class. After the simulator executes the next step, the simulator calls the *observe()* method of the node instance with the parameters s_{t+1} and r_{t+1}. By calling the *pushExperience()* method, the node then push the experience to the experience buffer, while each experience consists of s_t, at, s_{t+1}, and r_{t+1}. The target network samples an experience from the experience buffer and calculates the loss. By calling the *optimize()* method, the policy network is updated first. Then, the target network is softly updated with the parameters of the policy network. When the simulator calls the *getAction()* method of the node instance, the node instance internally calls the *selectAction()* function to return the next action. The above process is repeated for each frame for each node.

For the simulation configuration, we used a GPU as NVIDIA GeForce GTX 1060 3 GB, Intel Xeon E5-1607 v4, and 8 GB of memory. The nodes were uniformly distributed within a 100 m × 100 m network area. The number of sink nodes was set as 30. The destination nodes of the sender node were randomly selected among the sink nodes in the transmission range of the sender node. The simulation parameters used are listed in Tables II and III. Note that, since our learning model showed similar results regardless of the hyperparameters, we set it to the fastest learned hyperparameter as in Table IV. As the sensing action was taken during a slot, the slot time was set as 13 μs, which was the same as the distributed interframe space of the IEEE 802.11ad standard [28]. Unless noted otherwise, the simulation parameters were set based on the values listed in Table III. The following four models were considered for performance evaluation.

1) *DDMAC:* The proposed DQN-based framework for DMAC.
2) *DDDMAC:* It is the same as DDMAC, except that DDDMAC calculates the temporal difference as described in (12).
3) *CSMA:* The traditional DMAC protocol based on CSMA with exponential back-off. The CSMA node transmits

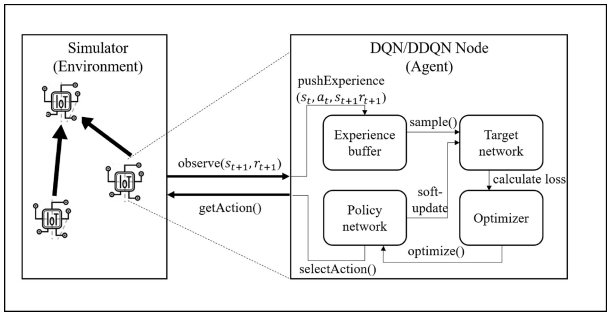

Fig. 8. Implemented simulator architecture (agent class architecture).

TABLE II
Simulation Environment

Category	Value
GPU	NVIDA GeForce GTX 1060 3 GB
CPU	Intel Xeon E5-1607 v4
Memory	8 GB
Programming Language	Python 3.8.6
Deep Learning Framework	PyTorch 1.7.0
Simulation Topology	Uniformly Distribution
Topology Size	$100m \times 100m$
The Number of Nodes	60 (30 senders and 30 sink nodes)

TABLE III
Simulation Parameters

Parameter	Value
Number of antennas	6
Transmission range	40 m
Number of sender nodes (N)	20
Number of target nodes ($\lvert D_i \rvert$)	4
Slot time	13 μs
DACK size	14 bytes
Data rate of channel	4,620 Mbps
Packet size	64 KBytes
Number of slots per frame (τ^{slot})	100
Learning rate (α)	0.0001
Optimizer	Adam
ϵ^{start}	0.5
ϵ^{end}	0.005
ϵ^{decay}	1000
Update interval of target network (τ^{target})	100
Soft update ratio (ω)	0.01

TABLE IV
CONVERGENCE FRAME AND AVERAGE REWARD ACCORDING TO HYPERPARAMETERS

ϵ^{end} \ ϵ^{start}	0.1	0.5	1.0
0.001	431 Frames (epoch) / r = 323	442 Frames (epoch) / r = 324	462 Frames (epoch) / r = 326
0.005	389 Frames (epoch) / r = 303	**403 Frames (epoch) / r = 321**	432 Frames (epoch) / r = 323
0.01	383 Frames (epoch) / r = 302	402 Frames (epoch) / r = 315	422 Frames (epoch) / r = 316

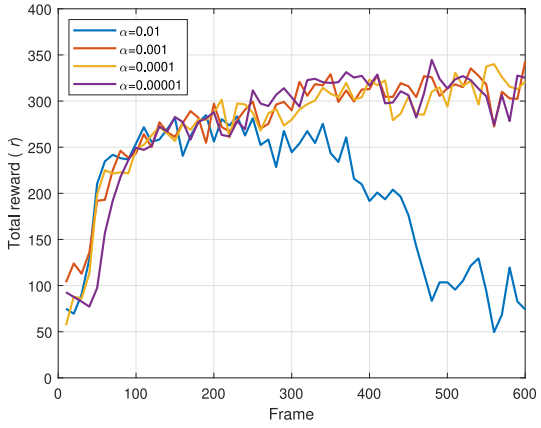

Fig. 9. Total reward versus frame curve of the DQN method for various learning rates.

after waiting for the back-off duration plus one slot, where the back-off duration was calculated as a random integer from $[0, CW-1]$. The minimum CW was set as 16, and the maximum CW was set as 1024.

4) *AL-DMAC:* Unlike the proposed scheme, AL-DMAC learns only the time slot to transmit DDATA [7]. If DDATA remains in the transmission queue, the AL-DMAC determines whether or not to transmit for each slot.

The aggregate throughput and latency were considered as performance metrics. The aggregate throughput is the total received traffic per second of the overall nodes. Latency is the time required for queued DDATA to arrive at the destination node and it was averaged for every received DDTA. After analyzing the convergence performance of the proposed DQN and double DQN frameworks, the performance evaluations were presented.

A. Convergence Analysis

In this section, we analyze the convergence of the proposed DQN and double DQN frameworks. Figs. 9 and 10 depict the total reward versus frame curves of the DQN and double DQN methods with various learning rates for 600 frames. The total reward is calculated as the sum of all the rewards during one frame. When the learning rate is $\alpha = 0.01$, the total reward increases initially before it starts to decrease after approximately 300 frames. This effect can be explained by the high-learning rate, which can lead to unstable training. Except for the $\alpha = 0.01$ case, the other cases converge after 400 frames and exhibit similar total rewards for various learning rates. After 400 frames, the double DQN method, Fig. 9, demonstrates a higher total reward than that of the

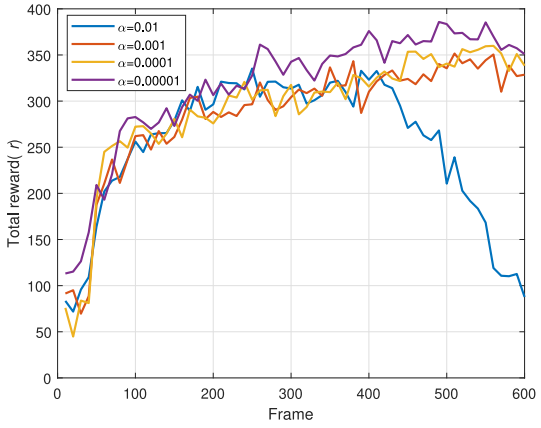

Fig. 10. Total reward versus frame curve of the double DQN method for various learning rates.

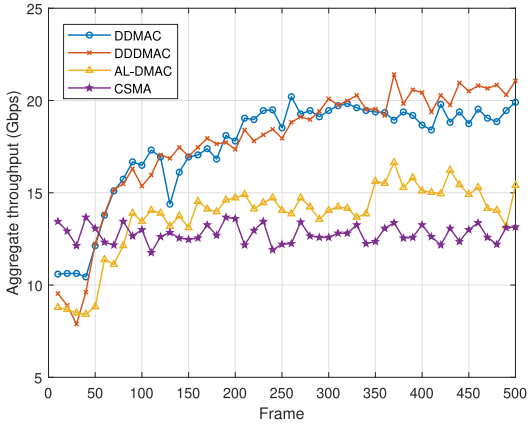

Fig. 11. Aggregate throughput versus frame curve during training for 500 frames.

DQN method. By decoupling the action selection and value estimation, the double DQN method achieves a higher total reward than that of the DQN method. Because each DQN framework works in a distributed manner and a single change of action can significantly affect the throughput of the entire network, jitter of the total reward still exists after convergence.

B. Throughput and Latency Analysis

The aggregate throughput versus frame curve is depicted in Fig. 11 during training for 500 frames. The aggregate throughput was averaged per 10 frames. Because the CSMA protocol does not require training, the aggregate throughput of the CSMA nodes is presented without training. As we can observe from the figure, between 100 and 350 frames, DDMAC and DDDMAC exhibit similar throughputs. After 400 frames, DDDMAC exhibits a higher aggregate throughput than that of the DDMAC, by approximately 1.1. However, the proposed DDMAC and DDDMAC schemes exhibit higher aggregate throughputs compared to those of AL-DMAC and CSMA. AL-DMAC only considers whether or not to transmit for a given slot, whereas the proposed schemes consider both the action and the node to transmit. Therefore, the proposed scheme can prevent collisions or deafness and transmit to another destination node.

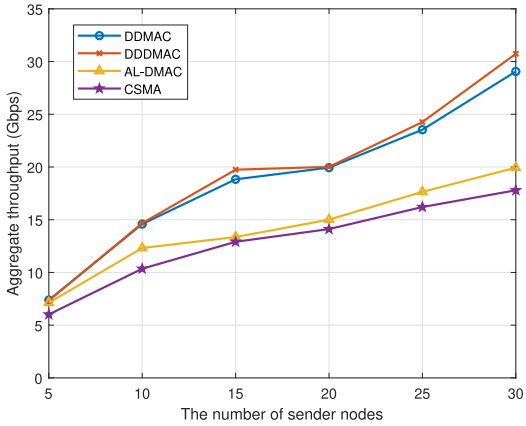

Fig. 12. Aggregate throughput versus number of sender nodes.

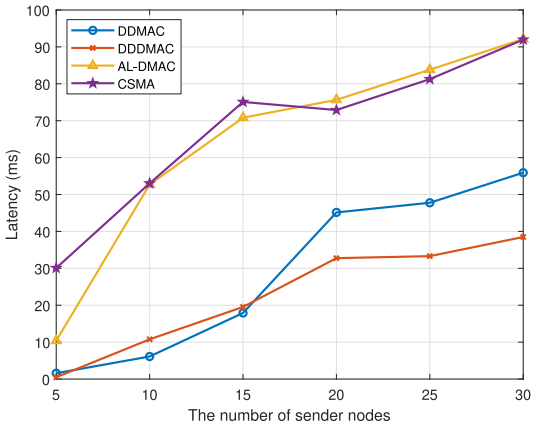

Fig. 13. Latency versus number of sender nodes.

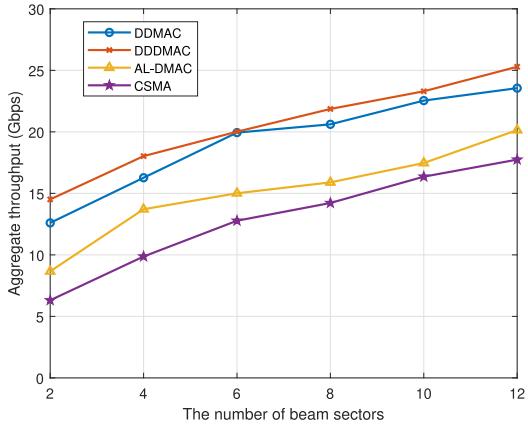

Fig. 14. Aggregate throughput versus number of beam sectors.

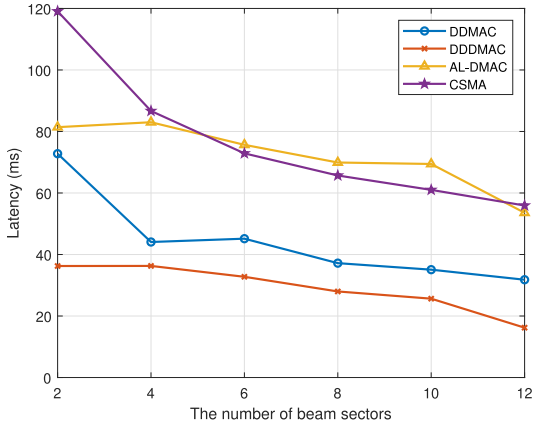

Fig. 15. Latency versus number of beam sectors.

Figs. 12 and 13 exhibit the evaluation results with varying numbers of sender nodes. The performances of DDMAC, DDDMAC, and AL-DMAC were measured after 500 training frames. The number of nodes varies from 5 to 30 with an interval of 5. When the number of sender nodes increases, collisions and deafness are more likely. DDMAC and DDDMAC exhibit superior performances compared to those of AL-DMAC and CSMA in both performance metrics. From Fig. 12, we can observe that the aggregate throughputs of DDMAC and DDMAC are greater than those of AL-DMAC and CSMA. Particularly, the throughput of DDDMAC is greater than those of AL-DMAC and CSMA by 2.8%–54.1% and 21.8%–72.7%, respectively. The aggregate throughput of AL-DMAC is higher than that of CSMA, but is significantly lower than those of DDMAC and DDDMAC. Considering destination nodes to transmit as the action, DDMAC and DDDMAC could achieve high-aggregate throughputs. As is evident in Fig. 13, the latency tends to increase with an increase in the number of sender nodes. Because DDMAC and DDDMAC can reduce the queuing delay in the transmission queue by achieving a higher throughput, the latencies of the proposed schemes are significantly smaller than those of AL-DMAC and CSMA. For instance, when the number of nodes is 30, DDDMAC exhibits 58.2% and 58.1% lower latency compared to those of AL-DMAC and CSMA, respectively.

Fig. 14 depicts the aggregate throughputs of four protocols for various numbers of beam sectors. As the number of beam sectors increases, the probabilities of collision and deafness decreases. Therefore, as is evident in Figs. 14 and 15, the aggregate throughput increases and latency decreases when the number of beam sectors increases for all schemes. However, similar to Figs. 12 and 13, DDMAC and DDDMAC exhibit higher aggregate throughputs and lower latencies compared to those of AL-DMAC and CSMA. DDDMAC exhibits a relatively high-aggregate throughput than that of DDMAC.

C. Energy Consumption

In this section, we compare and analyze the energy consumption values of our proposed scheme. Our proposed RL models operate individually for each device. Thus, it is true that running RL models on IoT devices is burdensome in terms of computing power and energy consumption. However, once it is trained, the time it takes to update the network and make decisions in the future is not a big burden. In addition, our learning model requires relatively small amounts of state and action compared to other large-scale applications, such as image processing, so the energy and time required to train is small.

In order to analyze energy consumption model for proposed scheme, we measured the energy consumption for training,

TABLE V
ENERGY CONSUMPTION FOR PROPOSED SCHEME

	DQN/DDQN Scheme	Other Schemes (/wo RL)
Energy Consumption for Training	5.6 W for 500 iteration	-
Energy Consumption for Inferring	1.2 mW	-
Energy Consumption for Transmitting/Receiving	6.4 W	6.4 W

inferring, and communication with the Raspberry Pi 4 model as Table V. In Table V, it is shown that our proposed method consumes 5.6 watts and takes about 1 h to train for about 500 iterations. In addition, comparing the energy cost of performing communications, our scheme shows that it consumes about 1.2 milliwatts more for inference, which does not appear to be a significant difference.

VII. CONCLUSION

This article proposed a DQN-based framework for the DMAC in WANETs for IoT. As there is no central coordinating unit in this framework, such as BS or AP, each IoT node learns its transmission policy by interacting with the environment. To this end, we first introduced the RL architecture for throughput maximization of WANET by defining the state, action, and reward of the node. Based on the Q-learning algorithm using the CSMA scheme, DQN and double DQN-based frameworks were proposed for time and space efficiency. The performance evaluation results indicated that the proposed DDMAC and DDDMAC exhibit superior performances compared to those of CSMA and AL-DMAC in terms of aggregate throughput and latency. However, our proposed scheme has a weakness in mobile environment, such as vehicular network or UAV network. In the mobile network, our propose cannot learned well due to the node's mobility. Therefore, we will intend to adapt the DRL technique to a network consisting of devices with mobility features. In this scenario, as the beam pair of the destination nodes is frequently changing, the efficient update of the beam pair is expected to be a significantly challenging problem.

REFERENCES

[1] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "IEEE 802.11 ad: Directional 60 GHz communication for multi-gigabit-per-second Wi-Fi," *IEEE Commun. Mag.*, vol. 52, no. 12, pp. 132–141, Dec. 2014.

[2] P. Zhou et al., "IEEE 802.11ay-based mmWave WLANs: Design challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1654–1681, 3rd Quart., 2018.

[3] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, Oct. 2019.

[4] C. Zhang, Y. Ueng, C. Studer, and A. Burg, "Artificial intelligence for 5G and beyond 5G: Implementations, algorithms, and optimizations," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 2, pp. 149–163, Jun. 2020.

[5] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1722–1760, 3rd Quart., 2020.

[6] X. Ye, Y. Yu, and L. Fu, "Deep reinforcement learning based MAC protocol for underwater acoustic networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 5, pp. 1625–1638, May 2022.

[7] P. Tiwari, D. K. Meena, and L. S. Pillutla, "Adaptive learning based directional MAC protocol for millimeter wave (mmWave) wireless networks," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, 2017, pp. 1–5.

[8] X. Cao, Z. Song, B. Yang, X. Du, L. Qian, and Z. Han, "Deep reinforcement learning MAC for backscatter communications relying on Wi-Fi architecture," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[10] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[11] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[13] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[14] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[15] Y. Chu, P. D. Mitchell, and D. Grace, "ALOHA and Q-learning based medium access control for wireless sensor networks," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2012, pp. 511–515.

[16] Y. Chu, S. Kosunalp, P. D. Mitchell, D. Grace, and T. Clarke, "Application of reinforcement learning to medium access control for wireless sensor networks," *Eng. Appl. Artif. Intell.*, vol. 46, pp. 23–32, Nov. 2015.

[17] L. de Alfaro, M. Zhang, and J. J. Garcia-Luna-Aceves, "Approaching fair collision-free channel access with slotted ALOHA using collaborative policy-based reinforcement learning," in *Proc. IFIP Netw. Conf. (Netw.)*, 2020, pp. 262–270.

[18] S. H. Park, P. D. Mitchell, and D. Grace, "Reinforcement learning based MAC protocol (UW-ALOHA-Q) for underwater acoustic sensor networks," *IEEE Access*, vol. 7, pp. 165531–165542, 2019.

[19] Y. Yu, S. C. Liew, and T. Wang, "Carrier-sense multiple access for heterogeneous wireless networks using deep reinforcement learning," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshop (WCNCW)*, 2019, pp. 1–7.

[20] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.

[21] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–7.

[22] Z. Zou, S. Zhao, G. Huang, and D. Tang, "Novel design of user scheduling and analog beam selection in downlink millimeter-wave communications," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4168–4178, Mar. 2022.

[23] A. Bhattacharjee, R. Bhattacharjee, and S. K. Bose, "An approach for mitigation of beam blockage in mmWave-based indoor networks," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14607–14622, Oct. 2021.

[24] Z. Su et al., "Energy-efficiency optimization for D2D communications underlaying UAV-assisted industrial IoT networks with SWIPT," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 1990–2002, Feb. 2023.

[25] K. Z. Ghafoor et al., "Millimeter-wave communication for Internet of Vehicles: Status, challenges, and perspectives," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8525–8546, Sep. 2020.

[26] N. Ahmed, D. De, F. A. Barbhuiya, and M. I. Hussain, "MAC protocols for IEEE 802.11ah-based Internet of Things: A survey," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 916–938, Jan. 2022.

[27] M. N. Alam, M. A. Hussain, and K. S. Kwak, "Neighbor initiated approach for avoiding deaf and hidden node problems in directional MAC protocol for ad-hoc networks," *Wireless Netw.*, vol. 19, no. 5, pp. 933–943, 2013.

[28] A. Akhtar and S. C. Ergen, "Directional MAC protocol for IEEE 802.11 ad based wireless local area networks," *Ad Hoc Netw.*, vol. 69, pp. 49–64, Feb. 2018.