Multi-Agent Reinforcement Learning-Based Distributed Channel Access for Next Generation Wireless Networks

Ziyang Guo, *Member, IEEE*, Zhenyu Chen, Peng Liu, *Member, IEEE*, Jianjun Luo, Xun Yang, *Member, IEEE*, and Xinghua Sun, *Member, IEEE*

***Abstract*— In the next generation wireless networks, more applications will emerge, covering virtual reality movies, augmented reality, holographic three-dimensional telepresence, haptic telemedicine and so on, which require the provisioning of high bandwidth efficiency and low latency services. In order to better support the aforementioned applications and services, novel distributed channel access (DCA) schemes are necessary. Therefore, we propose a new MAC protocol, QMIX-advanced Listen-Before-Talk (QLBT), based on the cutting-edge multi-agent reinforcement learning (MARL) algorithm. It employs a centralized training with decentralized execution (CTDE) framework to exploit the overall information of all agents during training, and ensure that each agent can independently infer the optimal channel access behavior based on its local observation. We enhance QMIX, a well-known MARL algorithm, by introducing an extra individual Q-value for each agent in the mixing network apart from the original total Q-value, which makes QLBT more stable. Moreover, delay to last successful transmission (D2LT) is first introduced in this work as a part of the observations of each QLBT agent, which facilitates agents to reach a cooperative policy that prioritizes the agent with the longest delay. Finally, extensive simulation experiments are provided to show that the proposed QLBT algorithm: 1) outperforms CSMA/CA and even its theoretical performance bound in various scenarios including saturated traffic, unsaturated traffic and delay-sensitive traffic; 2) is robust in dynamic environment; and 3) is able to friendly coexist with "legacy" CSMA/CA stations.**

***Index Terms*— Distributed channel access, multi-agent reinforcement learning, QMIX, listen before talk, multiple access.**

I. Introduction

NOWADAYS, we are witnessing a global roll-out of wireless communication systems, e.g., 5G (the 5^{th} generation mobile network) and Wi-Fi 6 (IEEE 802.11ax), which has brought tremendous changes to all aspects of our lives and work. With the rapid development of smart terminals and the large-scale deployment of the Internet-of-Things (IoT) devices, wireless applications are expected to grow exponentially. Meanwhile, new applications will emerge in next-generation wireless networks (NGWNs), including virtual reality movies, augmented reality, holographic three-dimensional telepresence, haptic telemedicine and so on, which requires the provisioning of high bandwidth efficiency and low latency services [1].

In order to provide better quality-of-service (QoS) delivery for the aforementioned applications and services, wireless techniques at all layers need to evolve, especially the media access control (MAC) layer which lays the foundations for boosting bandwidth efficiency, ensuring fairness and avoiding user collisions. A core field in the MAC layer design is referred to as *distributed channel access* (DCA), where multiple users share a common channel in a completely distributed manner without centralized scheduling unit. DCA is particularly important for the wireless communication systems on unlicensed spectrum, e.g., Wi-Fi, Ultra-Wideband, and ZigBee, due to their lack of centralized management and well-governed infrastructure.

Most conventional DCA mechanisms are based on random access schemes, which, as the name suggests, rely on *randomization* to mitigate collisions. The most popular random access scheme is carrier-sense multiple access with collision avoidance (CSMA/CA) [2]. CSMA/CA abides by listen-before-talk (LBT) protocol which regulates user to continually sense the channel before initiating a transmission. To avert potential collisions, each user picks up a random back-off time period to defer its transmission. The binary exponential back-off (BEB) scheme is adopted to reduce the probability of further collisions, i.e., doubling the average back-off time whenever a transmission failure occurs. Many research works have been conducted on the optimization of CSMA/CA parameters or the design of its variants, for instance, replacing BEB by other back-off functions [3]–[6]. However, the intrinsic random deferment feature makes them upper-bounded by a relative low MAC efficiency [7] and suffer from severe fairness issues in many realistic scenarios [8], which is unsuitable for the high throughput and low latency requirements of NGWNs.

Manuscript received August 18, 2021; revised November 15, 2021; accepted December 17, 2021. Date of publication January 14, 2022; date of current version April 18, 2022. The work of Xinghua Sun was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101120003 and the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011906. *(Corresponding author: Peng Liu.)*

Ziyang Guo, Peng Liu, Jianjun Luo, and Xun Yang are with the Wireless Technology Lab, 2012 Laboratories, Huawei Technologies Company Ltd., Shenzhen 518129, China (e-mail: guoziyang@huawei.com; jeremy.liupeng@huawei.com; luojiajun4@huawei.com; david.yangxun@huawei.com).

Zhenyu Chen and Xinghua Sun are with the School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen 518107, China (e-mail: chenzhy89@mail2.sysu.edu.cn; sunxinghua@mail.sysu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3143251.

Digital Object Identifier 10.1109/JSAC.2022.3143251

Recent advances in artificial intelligence (AI) techniques, especially the fast development of deep reinforcement learning (DRL), shed light on the solutions to the challenges in current MAC layer protocols [9], [10]. With the success of DRL in many areas [11]–[13], the idea of an intelligent agent that can observe adapt to its environment is gaining momentum. We believe that to cope with the growing complexity of NGWNs, *replacing inefficient random channel access mechanisms by DRL-based paradigms, which enable deterministic decision and achieve better QoS delivery via online learning of the environment, is highly desired*.

The considered DCA problem can be fundamentally formulated as a multi-agent decision-making task where multiple stations[1] concurrently take channel access actions (e.g., transmit or not) based on their respective observation to maximize the overall long-term reward. Researchers have attempt to solve multi-agent tasks using independent DRL algorithm [14]. However, it fails to tackle the *non-stationarity* of the environment (i.e., the environment is influenced by all agents' behaviors and thus is non-stationary for one agent) in the considered DCA problem. As our simulation results revealed, the channel is *always* occupied by some of the agents in most cases, which loses fairness.

In order to converge to a cooperative learning behavior of agents, we solve the DCA problem on the basis of a multi-agent reinforcement learning (MARL) method–QMIX [12], which adopts a centralized training with decentralized execution (CTDE) framework. Specifically, the central unit makes use of the global environment information to train the neural network parameters, while each agent makes independent channel access decision only based on its local observations. We name the proposed MAC layer protocol as QMIX-advanced LBT (QLBT) as it complies with LBT protocol. In regard to the design of QLBT, on the one hand, we introduce the *delay to last successful transmission* (D2LT) into observation and reward functions to facilitate the convergence of DCA to a centralized scheduling policy under which the agent with the longest time of unsuccessful transmission accesses the channel. On the other hand, we enhance QMIX algorithm by modifying the mixing network architecture. The output of the original QMIX is a total Q-value. Applying it directly to DCA problem may fall into a local optimum that loses fairness, i.e., a subset of agents dominate the channel while others have no chance to transmit. To solve this problem, in addition to the original total Q-value, we introduce an extra individual Q-value for each agent. This design enables the proposed QLBT algorithm to support two types of rewards, which further achieves the goal of simultaneously maximizing network throughput and ensuring fairness among agents. The detailed design is elaborated in Section V.

A. Contributions and Main Results

The contributions and main results of this work are summarized as follows:

- We formulate the DCA problem as a decentralized partially observable Markov decision process (Dec-POMDP) and propose a novel MAC layer protocol–QLBT. It is shown that QLBT outperforms independent DRL solution which suffers non-stationary issues.
- We introduce D2LT as a critical part of the agent's observations as well as the reward function design. D2LT can be easily extracted by listening to the acknowledgments (ACKs) on the channel and facilitates QLBT to reach a consentaneous policy that prioritizes the agent with longest delay.
- We modify QMIX by introducing extra individual Q-values and rewards, which makes QLBT more stable as such individual Q-values and rewards provide guidance for the action of each agent during training.
- We verify the performance of QLBT in a variety of scenarios including saturated traffic, unsaturated traffic, delay-sensitive traffic, dynamic number of serving stations, time-varying traffic conditions and coexistence scenarios with Wi-Fi. Extensive simulation results show that QLBT: 1) outperforms the theoretical upper bound of conventional DCA schemes in terms of throughput, delay and jitter; 2) is robust to dynamic environment; 3) has a fast convergence speed; and 4) is able to friendly coexist with "legacy" Wi-Fi devices.

[1]In this paper, stations, agents and users are interchangeable.

B. Related Works

At the early stage of DCA research, researchers focused on the optimization of CSMA/CA parameters and proposed a series of CSMA/CA variants. CSMA/CA with linear/multiplicative increase and linear decrease (LMILD) back-off [4] was shown to be able to improve throughput than BEB for large network sizes. Carrier sense multiple access with enhanced collision avoidance (CSMA/ECA) [5], [6] was able to converge to collision-free medium access in a highly homogeneous scenario, where all users have the same packet length and access parameters, by utilizing a deterministic back-off after successful transmissions. Although these algorithms outperform CSMA/CA in some special cases, they are still restricted to the framework of CSMA/CA and reduce collisions via randomization. In contrast with these prior studies, we devise a more intelligent DCA scheme based on state-of-the-art DRL algorithm.

With the development of AI techniques, DRL-based channel access that impose intelligence on intrinsic protocol design has gained tremendous research interest. The applications of DRL in dynamic parameter adjustment of Enhanced Distributed Channel Access (EDCA) were investigated in [15] and [16]. Different from these work, our work fully exploited the intelligence of DRL to design new channel access schemes instead of optimizing parameters on top of CSMA/CA. Another research direction of DRL-based channel access addresses the coexistence issue in *heterogeneous networks*, i.e., a DRL empowered node learns to fairly coexist with other MAC protocols. A carrier-sense deep-reinforcement learning multiple access (CS-DLMA) algorithm was proposed in [17] and [18], aiming at achieving maximum network throughput while guaranteeing α-fairness with other unknown MAC protocols such as time division multiple access (TDMA), Aloha, Wi-Fi and

Long-Term Evolution (LTE) [19]. Different from these recent works, we are interested in the homogeneous scenario where multiple agents act simultaneously. This is a highly interactive system which requires discreet solution to overcome the non-stationary challenge.

Germane to our work, [20] also concentrated on a homogeneous network, where each node performed independent DRL algorithm to learn channel access policy. To achieve fairness among users, federated learning framework was adopted, under which a central unit collected the neural network parameters of all agents and redistributed the average model. This frequent message exchange leads to much airtime overhead. Moreover, the effect of federated operation on fairness is debatable. In a different way, we leverage a MARL solution which has brought encouraging results in other communities, e.g., computer science, and take the fairness issue into consideration in our native design of the state and reward functions.

DRL algorithms have also been studied in some other wireless communication networks, including power control in cellular networks [21]–[23], resource allocation in vehicle-to-everything (V2X) communications [24], [25], user scheduling in orthogonal frequency-division multiple access (OFDMA) systems [26], spectrum sharing between New Radio (NR) and LTE [27]. A much more comprehensive literature review can be found in recent surveys [28], [29]. Nevertheless, methods proposed in these works cannot be directly applied to solving the DCA problem. The main reason is that the tricky parameters they dig out, such as observations and reward functions, are elaborately tailored to fit in those specific scenarios.

C. Outline

The remainder of this paper is organized as follows. Section II provides preliminaries on single-agent and multi-agent reinforcement learning. Section III introduces the system model and problem of interest. Section IV formulates the considered DCA problem in MARL framework and presents the underlying Dec-POMDP model. The QLBT algorithm and protocol are elaborated in Section V. Simulation results are presented in Section VI, followed by the conclusion and discussions of possible future work in Section VII.

II. Reinforcement Learning Preliminaries

This section introduces the preliminaries on RL, mainly focus on value-based RL algorithm that is related to our work.

A. Single-Agent Reinforcement Learning

In single-agent reinforcement learning (SARL) problems, at each time step t, the agent observes the environment's state s_t and selects an action a_t according to policy π, after that it receives a reward r_t and observes a new state s_{t+1}. The objective of the agent is to find the optimal policy π^* that maximizes the expected cumulative discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty}\gamma^t r_t]$, where $\gamma \in (0,1]$ is a discount factor.

To solve this problem, the action-value function, also known as Q-value, is defined as the expected cumulative discounted return from undertaking action a at state s, i.e.,

$$Q(s,a) \triangleq \mathbb{E}_\pi[\sum_{k=0}^{\infty}\gamma^k r_{t+k}|s_t = s, a_t = a].$$

It is shown that the optimal action-value function $Q^*(s,a) \triangleq max_\pi Q^\pi(s,a)$ obeys the Bellman Optimality Equation

$$Q^*(s,a) = \mathbb{E}[r_t + \gamma \max_{a'} Q^*(s_{t+1},a')|s_t = s, a_t = a],$$

and the corresponding optimal policy can be derived from the optimal action-value function by taking a greedy action [30], i.e., $\pi^*(s) = \arg\max_a Q^*(s,a)$.

Q-learning algorithm [31] approximates $Q^*(s,a)$ by iteratively updating Q-values via a temporal-difference learning

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \beta[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)]$$

with learning rate $\beta \in (0,1]$. During this iterative convergence process, ε-greedy exploration is used to avoid getting stuck in local optima, i.e., at state s, the agent chooses a greedy action $a = \arg\max_{a'} Q(s,a')$ with probability $1-\varepsilon$ and chooses a random action with probability ε.

Q-learning updates $Q(s,a)$ in a tabular form, which suffers intolerable large storage space and long convergence time as the size of state-action space increases. To tackle this problem, Deep Q-Network (DQN) is proposed in [11] to approximate the optimal action-value function with a deep neural network parameterized by $\boldsymbol{\theta}$, i.e., $Q(s,a;\boldsymbol{\theta}) \approx Q^*(s,a)$. It adopts an experience memory to store the transition tuple (s,a,r,s'), where the state s' is observed after taking the action a at state s and receiving reward r. The neural network parameters $\boldsymbol{\theta}$ are learnt by sampling a batch of transitions from the experience memory and minimizing the loss function

$$L^{DQN}(\boldsymbol{\theta}) = \sum_{bs}\left[y^{DQN} - Q(s,a;\boldsymbol{\theta})\right]^2, \tag{1}$$

where bs is the batch size, $y^{DQN} = r + \gamma \max_{a'} Q(s',a';\boldsymbol{\theta}^-)$, and $\boldsymbol{\theta}^-$ denotes the *target neural network* parameters, which are periodically copied from $\boldsymbol{\theta}$ and remain constant for a number of iterations.

B. Multi-Agent Reinforcement Learning

In MARL problems, each agent chooses its action independently based on its own observation of the environment to maximize the overall expected reward. Compared to SARL, MARL problems are more intractable due to the challenges of non-stationarity and combinatorial complexity [32].

Mathematically, a multi-agent task can be described as a decentralized partially observable Markov decision process (Dec-POMDP) [33] using a tuple $\mathcal{G} =< \mathcal{S}, \mathcal{A}, P, r, n, \gamma, \mathcal{Z} >$. To be specific, $s \in \mathcal{S}$ denotes the state of the environment. At each time step, agent $i \in \{1,2,\ldots,n\}$ chooses an action $a^i \in \mathcal{A}$ independently, forming a *joint action* $\boldsymbol{a} \in \mathcal{A}^n$, which influences the environment according to the transition matrix $P(s'|s,\boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$. All agents share the same reward function $r(s,\boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. For partially observable scenario, each agent can only obtain a local observation $z \in \mathcal{Z}$ of the environment. Denote the action-observation history of

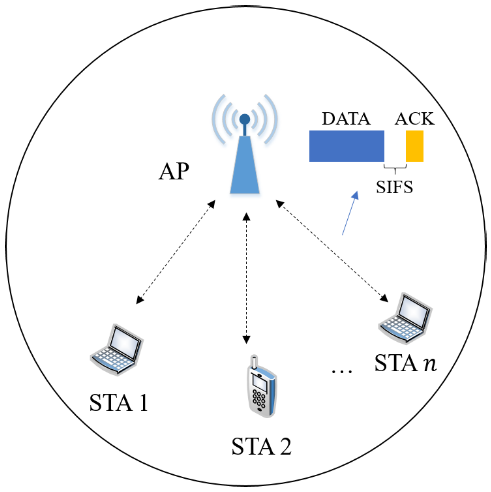


Fig. 1. Scenario of interest: multiple STAs share one common channel and determine data transmission in a distributed manner; data transmission is immediately replied by ACK from AP to indicate its successful reception.

agent i as $\tau^i \in \mathcal{T} \equiv (\mathcal{Z} \times \mathcal{A})^*$, on which it conditions a stochastic policy $\pi^i(a^i|\tau^i) : \mathcal{T} \times \mathcal{A} \to [0,1]$. Similarly, $\boldsymbol{\tau} \in \mathcal{T}^n$ represents the joint action-observation history and π represents the joint policy. The individual and joint action-value functions are denoted as $Q^i(\tau^i, a^i), i \in \{1,2,\ldots,n\}$ and $\mathcal{Q}_{tot}(\boldsymbol{\tau}, \boldsymbol{a})$, respectively.

The simplest solution to a multi-agent problem is to decompose it into a collection of simultaneous single-agent problems that share the same environment, which leads to *independent DQN* [34]. However, this method fails to address the non-stationarity issue and thus has no convergence guarantee most of the time. In recent years, the paradigm of CTDE has attracted much more attention due to its tremendous success in MARL tasks. In CTDE framework, global information of all agents is participated in during training, while only the local information is dependent on during execution.

QMIX is one of the most efficient MARL algorithms [12]. It approximates the optimal joint action-value function $\mathcal{Q}^*_{tot}$ by a neural network parameterized $\boldsymbol{\theta}$, which consists of agent neural network, mixing network and a set of hypernetworks. Each agent network extracts information from individual action-observation history τ^i and outputs Q^i. The mixing network combines Q^i of all agent networks and outputs $\mathcal{Q}_{tot}$. The hypernetworks generate the weights and biases of the mixing network based on the global state. Similar to DQN, the neural network parameters $\boldsymbol{\theta}$ are learnt by minimizing the loss function

$$L^{QMIX}(\boldsymbol{\theta}) = \sum_{bs} \left[y^{QMIX} - \mathcal{Q}_{tot}(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\theta}) \right]^2, \qquad (2)$$

where $y^{QMIX} = r + \gamma \max_{\boldsymbol{a}'} \mathcal{Q}_{tot}(\boldsymbol{\tau}', \boldsymbol{a}'; \boldsymbol{\theta}^-)$.

III. System Model and Problem of Interest

The detailed system model of the DCA problem considered in this work is introduced in the section. As shown in Fig. 1, we consider a time-slotted wireless network where n stations (STAs) seek channel access opportunity to transmit data packets to their associated Access Point (AP). This setup is identical to single basic service set scenario in 802.11 WLAN. Each STA is equipped with a finite buffer and the packet arrives randomly, e.g., following a Poisson distribution. The packets in the queue is served in a first come first serve manner. The STA is assumed to perform carrier sensing before accessing the channel, i.e., LBT, and the transmission is allowed only if the channel is sensed idle. A transmission once starts will last for multiple time slots, the number of which is denoted as packet length. A transmission is successful if and only if there is only one STA transmitting during the whole packet length. A successful transmission will be replied by an ACK from AP after a short inter-frame space (SIFS), otherwise the packet will be retransmitted at the next channel access opportunity. Note that only uplink transmissions are considered here for the ease of presentation, however, the proposed method can be easily extended to more complex scenarios.

For the aforementioned system model, our goal is to design a distributed channel access strategy that maximizes the aggregate network throughput while maintaining fairness among STAs. For these two optimization objectives, i.e., maximizing network throughput and ensuring fairness, we formulate two reward functions accordingly. One is a total reward function that encourages the successful transmission behavior and punishes the actions leading to collisions. The other is an individual reward function based on proportional-fair (PF) scheduler. The individual reward function ensures fairness by awarding the actions consistent with PF scheduler and penalizing the other inconsistent actions. The detailed design of the total reward and the individual reward will be introduced in the next section.

IV. Dec-POMDP Formulation

The DCA problem stated in the previous section can be regarded as a MARL problem where each QLBT agent interacts with others based on its own observations to gradually learn the optimal channel access decision. In this case, the underlying Dec-POMDP model can be described by the tuple $\mathcal{G} =< \mathcal{S}, \mathcal{A}, P, r, n, \gamma, \mathcal{Z} >$, which is introduced in details hereafter.

A. Action

The *action* of agent $i \in \{1,2,\ldots,n\}$ at time slot t is defined as $a^i_t \in \mathcal{A} \equiv \{$*Transmit, Wait*$\}$, where *Transmit* means that agent i transmits at current time slot, and *Wait* means that agent i waits for a time slot.

B. Local Observation and Global State

The *local observation* of agent $i \in \{1,2,\ldots,n\}$ at time slot t, $z^i_t \in \mathcal{Z}$, consists of two parts. One is an indicator $o^i_t \in \{0,1\}$ of other agents' transmission. If there is no other agent transmitting at current time slot, $o^i_t = 0$, otherwise $o^i_t = 1$. This information is equivalent to carrier-sensing result when $a^i_t =$ *Wait* or transmission result when $a^i_t =$ *Transmit*. The other part of the observation z^i_t is defined as $[v^i_t, v^{-i}_t]$, where

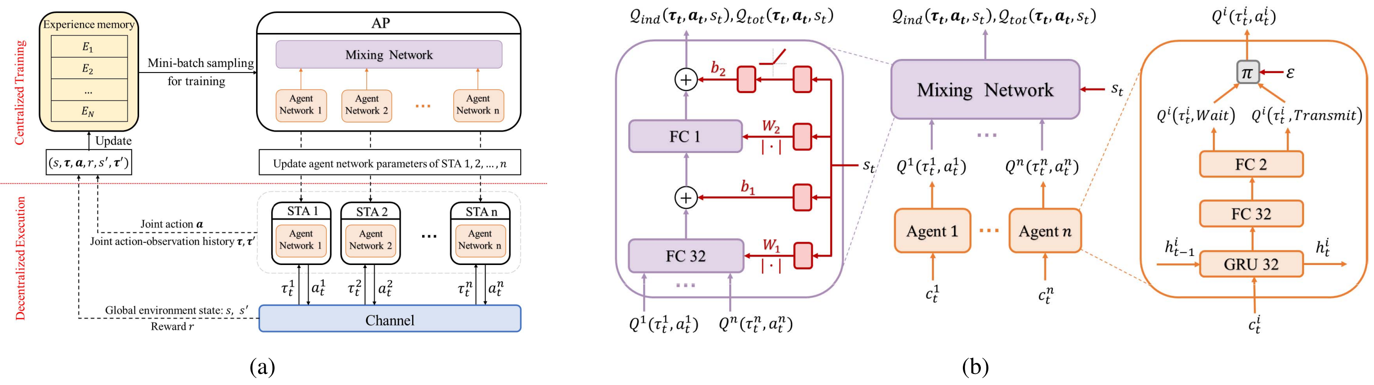

Experience memory
E_1
E_2
...
E_N
Mini-batch sampling for training
AP
Mixing Network
Agent Network 1
Agent Network 2
Agent Network n
Update
$(s, \boldsymbol{\tau}, \boldsymbol{a}, r, s', \boldsymbol{\tau}')$
Update agent network parameters of STA 1, 2, ..., n
Centralized Training
Decentralized Execution
Joint action $\boldsymbol{a}$
Joint action-observation history $\boldsymbol{\tau}, \boldsymbol{\tau}'$
STA 1
STA 2
STA n
τ_t^1 a_t^1 τ_t^2 a_t^2 τ_t^n a_t^n
Global environment state: s, s'
Reward r
Channel
(a)
$Q_{ind}(\boldsymbol{\tau_t}, \boldsymbol{a_t}, s_t), Q_{tot}(\boldsymbol{\tau_t}, \boldsymbol{a_t}, s_t)$
b_2
FC 1
W_2
$|\cdot|$
b_1
FC 32
W_1
$|\cdot|$
s_t
$Q^1(\tau_t^1, a_t^1)$
$Q^n(\tau_t^n, a_t^n)$
Mixing Network
Agent 1
Agent n
c_t^1
c_t^n
$Q^i(\tau_t^i, a_t^i)$
π
ε
$Q^i(\tau_t^i, Wait)$
$Q^i(\tau_t^i, Transmit)$
FC 2
FC 32
GRU 32
h_{t-1}^i
h_t^i
c_t^i
(b)

Fig. 2. (a) CTDE framework: Centralized training is performed at AP side based on experiences reported by each STA. After training, AP sends out the agent network parameters to corresponding STA. In decentralized execution, each STA determines whether to access the channel independently based on its own action-observation history. (b) QLBT architecture: The right-hand side block shows the architecture of agent network. The left-hand side block shows the architecture of mixing network. The red parts correspond to hypernetworks that produce the weights and biases for mixing network layers.

v_t^i represents the number of time slots since last successful transmission (i.e., D2LT) of agent i and v_t^{-i} represents the D2LT of other agents except i. In practice, v_t^i and v_t^{-i} can be obtained by listening to the ACKs on the wireless channel. If agent i receives its own ACK replied by the associated AP, then $v_t^i = 0$, otherwise $v_t^i = v_t^i + 1$.

A packet transmission may last for more than one time slot. To avoid storing a large amounts of duplicated contents in the action-observation, we introduce l_i^t, the number of time slots that the same (a_t^i, o_t^i) pair lasts, into observation. Then, the action-observation of agent i at time slot t can be denoted as

$$c_t^i \triangleq [a_t^i, o_t^i, l_t^i, d_t^i, d_t^{-i}], \tag{3}$$

where d_t^i and d_t^{-i} are the normalized versions of v_t^i and v_t^{-i}, i.e., $d_t^i = \frac{v_t^i}{v_t^i + v_t^{-i}}$ and $d_t^{-i} = \frac{v_t^{-i}}{v_t^i + v_t^{-i}}$. The action-observation history of agent i at time slot t is represented as

$$\tau_t^i \triangleq [c_{t-M+1}^i, \ldots, c_{t-2}^i, c_{t-1}^i], \tag{4}$$

where M is the length of action-observation history. The joint agent action-observation history of all agents at time t is obtained as $\boldsymbol{\tau_t} \triangleq [\tau_t^1, \tau_t^2, \ldots, \tau_t^n]$.

The global state information of the environment is available while learning the decentralized policies. We define the global state at time t as

$$s_t \triangleq [\boldsymbol{a_{t-1}}, \boldsymbol{D_{t-1}}]. \tag{5}$$

Here $\boldsymbol{a_t} \triangleq [a_t^1, a_t^2, \ldots, a_t^n]$ is the joint action of all agents, and $\boldsymbol{D_t} \triangleq [D_t^1, D_t^2, \ldots, D_t^n]$ is the normalized versions of all v_t^i with $D_t^i = \frac{v_t^i}{\sum_{j=1}^n v_t^j}$.

C. Reward

After taking action on the environment, a reward will be received. Recall that our design goal is to maximize the aggregate network throughput while maintaining fairness among agents. To this end, a total reward that evaluates the overall performance of all agents is proposed to maximize the network throughput. On the other hand, an individual reward that evaluates the behavior of each agent is proposed to maintain fairness. Specifically, the total reward at time slot t is defined as

$$r_{t,tot} = \begin{cases} 1, & \textit{if agent } i = \arg\max \boldsymbol{D_t} \textit{ transmits successfully}, \\ D_t^j, & \textit{if agent } j \neq \arg\max \boldsymbol{D_t} \textit{ transmits successfully}, \\ -1, & \textit{if collision}, \\ 0, & \textit{otherwise}. \end{cases}$$

Note that the total reward encourages the behavior of successful transmission by assigning positive rewards and punishes the behavior of collision by assigning negative rewards. Moreover, it also guarantees that a larger reward will be received if an agent with larger D2LT transmits successfully, which is consistent with the intuition.

The individual reward for agent i at time slot t is

$$r_{t,ind}^i = \begin{cases} 1, & \textit{if } a_t^i = a_t^{i*}, \\ -1, & \textit{otherwise}, \end{cases}$$

where a_t^{i*} denotes the optimal action of agent i at time t according to PF scheduling. The PF scheduling assigns a priority to each user and schedules the channel to the user with the highest priority. In this sense, the optimal action for user i at time slot t can be described as

$$a_t^{i*} = \begin{cases} 1, & \textit{if } i = \arg\max P_t^i, \\ 0, & \textit{otherwise}, \end{cases}$$

where $P_t^i \triangleq \frac{U_t^i}{V_t^i}$ denotes the priority of user i at time t. U_t^i is an indicator of the buffer emptiness, i.e., $U_t^i = 0$ if the buffer of user i at time t is empty, $U_t^i = 1$ if user i has packet to transmit at time t. V_t^i is the average throughput of user i over last second.

The basic idea of the individual reward is to ensure fairness by awarding the actions consistent with PF scheduling and penalizing the inconsistent actions. It also provides a guidance for the action of each agent during training, which is conductive to improving the algorithm performance.

So far, combining total and individual rewards, the **reward** at time slot t is obtained as $r_t \triangleq [r_{t,ind}^1, \ldots, r_{t,ind}^n, r_{t,tot}]$.

V. QLBT Algorithm and Protocol

Based on the Dec-POMDP formulation introduced above, we propose a CTDE QLBT algorithm to find the optimal channel access policy. The framework of the proposed algorithm is shown in Fig. 2a. The centralized training is performed at AP side based on the experiences consisting of joint action-observation history, joint action, global environment state and reward. An experience memory (EM) is used to store experience tuple $(s, \boldsymbol{\tau}, \boldsymbol{a}, r, s', \boldsymbol{\tau}')$. With a slight abuse of definition, we may omit the subscript of time slot t in section IV and use the superscript $'$ to indicate a quantity at the next time slot in the subsequent discussions. The overall network architecture used in centralized training is shown in Fig. 2b. We modify the mixing network of QMIX and introduce an individual loss to original QMIX loss function, which will be introduced in details in the following. After training, AP sends out the agent network parameters to corresponding STAs. In decentralized execution, each STA determines whether to access the channel independently based on its own action-observation history. The pseudo-code of the proposed QLBT algorithm is summarized in Algorithm 1.

Algorithm 1 QLBT Algorithm

```
Initialize parameters: $n$, $\varepsilon$, $\gamma$, $T$, $N_r$, $t = 0$, $cnt = 0$, $s = s_0$, $\tau_0^i = \tau_0$, $a_0^i = Wait$, $z_0^i = 0$, $\beta^i$, $\forall i \in \{1, 2, \ldots, n\}$, $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
while $t < T$ do
  for $i = 1, 2, \ldots, n$ do
    Compute $\tau_t^i$ from $\tau_{t-1}^i, a_{t-1}^i, z_{t-1}^i$
    if $a_{t-1}^i == Transmit$ then
      if agent $i$ finishes transmission then
        $a_t^i \leftarrow Wait$
      else
        $a_t^i \leftarrow Transmit$
    else
      if Channel is busy then
        $a_t^i \leftarrow Wait$
      else
        Input $\tau_t$ to agent network $i$ and output $\mathbf{Q}^i$
        Generate action $a_t^i$ from $\mathbf{Q}^i$ using $\varepsilon$-greedy policy
  if Channel is idle then
    Get $\boldsymbol{\tau}' = [\tau_{t+1}^1, \tau_{t+1}^2, \ldots, \tau_{t+1}^n]$, $\boldsymbol{a} = [a_t^1, a_t^2, \ldots, a_t^n]$
    Observe global state $s' = s_{t+1}$ and reward $r$
    Store $(s, \boldsymbol{\tau}, \boldsymbol{a}, r, s', \boldsymbol{\tau}')$ to EM
    Randomly sample $bs$ experiences from EM as $E$
    for each sample $e = (s, \boldsymbol{\tau}, \boldsymbol{a}, r, s', \boldsymbol{\tau}')$ in $E$ do
      Compute $\tilde{a}^i = \arg\max_{\tilde{a}} Q^i(\tau^i, \tilde{a}, s'; \boldsymbol{\theta}), \forall i$ to obtain $\boldsymbol{a}' = [\tilde{a}^1, \tilde{a}^2, \ldots, \tilde{a}^n]$
      Compute $y_{tot} = r_{tot} + \gamma \mathcal{Q}_{tot}(\boldsymbol{\tau}', \boldsymbol{a}', s'; \boldsymbol{\theta}^-)$
      Compute $y_{ind}^i = r_{ind}^i + \gamma \mathcal{Q}_{ind}^i(\boldsymbol{\tau}', \boldsymbol{a}', s'; \boldsymbol{\theta}^-)$
    Compute $L(\theta) = \sum_{e \in E}[(y_{tot} - \mathcal{Q}_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, s; \boldsymbol{\theta}))^2 + \sum_i \beta^i (y_{ind}^i - \mathcal{Q}_{ind}^i(\boldsymbol{\tau}, \boldsymbol{a}, s; \boldsymbol{\theta}))^2]$
    Update $\boldsymbol{\theta}$ by performing mini-batch gradient descent
    if $(cnt \mod N_r) == 0$ then
      $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$
    $cnt \leftarrow cnt + 1$, $s \leftarrow s'$, $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau}'$
  $t \leftarrow t + 1$
```

A. Neural Network Architecture

As shown in Fig. 2b, The QLBT network consists of two key components: agent network and mixing network, which are introduced as follows.

1) Agent Network: At each time step, agent network i takes c_t^i as an input and feeds it to a gated recurrent unit (GRU) layer. Then, the output of GRU passes two fully-connected (FC) layers and outputs $\mathbf{Q}^i \triangleq \{Q^i(\tau_t^i, Wait), Q^i(\tau_t^i, Transmit)\}$. Action a_t^i is selected using ε-greedy algorithm and corresponding $Q^i(\tau_t^i, a_t^i)$ is fed to the mixing network. The rectified linear unit (ReLU) is used as the activation function of the first FC layer and a linear activation is adopted by the second FC layer. All agent networks can share the same parameters, however, the parameters of each agent network are different by default.

2) Mixing Network: The mixing network of original QMIX algorithm takes the output of each agent network $Q^i(\tau_t^i, a_t^i), i \in \{1, 2, \cdots, n\}$ and global environment state s_t as inputs to generate $\mathcal{Q}_{tot}(\boldsymbol{\tau_t}, \boldsymbol{a_t}, s_t)$ based on parameter $\boldsymbol{\theta}$. Mathematically, it can be represented as

$$\mathcal{Q}_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, s; \boldsymbol{\theta}) = f(Q^1, Q^2, \cdots, Q^n; \boldsymbol{\tau}, \boldsymbol{a}, s). \tag{6}$$

We modify the mixing network by adding an extra output vector $\mathcal{Q}_{ind} \triangleq [\mathcal{Q}_{ind}^1, \ldots, \mathcal{Q}_{ind}^n] \in \mathbb{R}^n$ apart form $\mathcal{Q}_{tot} \in \mathbb{R}$. Similarly, for agent i,

$$\mathcal{Q}_{ind}^i(\boldsymbol{\tau}, \boldsymbol{a}, s; \boldsymbol{\theta}) = f(Q^1, Q^2, \cdots, Q^n; \boldsymbol{\tau}, \boldsymbol{a}, s). \tag{7}$$

In this way, the mixing network is a two-layer feedforward neural network, which combines agent network outputs and generates $[\mathcal{Q}_{ind}, \mathcal{Q}_{tot}]$. The exponential linear unit (ELU) is used as the activation function of the first hidden layer and a linear activation is adopted by the output layer. Moreover, we stop the gradient from $\mathcal{Q}_{ind}^i$ to Q^j during training provided that $i \neq j$, i.e., $\frac{\partial \mathcal{Q}_{ind}^i}{\partial Q^j} = 0$ if $i \neq j$.

The weights of the mixing network are produced by separate hypernetworks. Each hypernetwork takes the global state s_t as input and generates the weights of one layer of the mixing network. Each hypernetwork consists of a single FC layer, followed by an absolute activation function, to ensure that the mixing network weights are non-negative. The output of the hypernetwork is a vector, which is then reshaped into a matrix of appropriate size. The biases are produced in the same manner but are not restricted to be non-negative. The final bias is produced by a two-layer hypernetwork with a ReLU non-linearity in the middle.

B. Loss Function

Double deep Q-network (DDQN) [35] is adopted in the proposed QLBT algorithm to eliminate the over-estimation problem in DQN. Recall that we have a total reward r_{tot} and n individual rewards $[r_{ind}^1, \ldots, r_{ind}^n]$. The loss function of the proposed QLBT becomes a combination of total loss and individual loss accordingly, i.e.,

$$L^{QLBT}(\boldsymbol{\theta}) = L_{tot}(\boldsymbol{\theta}) + \sum_{i=1}^{n} \beta^i L_{ind}^i(\boldsymbol{\theta}), \tag{8}$$

where β^i is the weight of individual loss for agent i.

The total loss and individual loss for agent i are defined as

$$L_{tot}(\boldsymbol{\theta}) = \sum_{bs} \left[y_{tot} - \mathcal{Q}_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, s; \boldsymbol{\theta})\right]^2,$$
$$L_{ind}^i(\boldsymbol{\theta}) = \sum_{bs} \left[y_{ind}^i - \mathcal{Q}_{ind}^i(\boldsymbol{\tau}, \boldsymbol{a}, s; \boldsymbol{\theta})\right]^2, \tag{9}$$

respectively. Here $y_{tot} = r_{tot} + \gamma \mathcal{Q}_{tot}(\boldsymbol{\tau}', \boldsymbol{a}', s'; \boldsymbol{\theta}^-)$, $y_{ind}^i = r_{ind}^i + \gamma \mathcal{Q}_{ind}^i(\boldsymbol{\tau}', \boldsymbol{a}', s'; \boldsymbol{\theta}^-)$, $\boldsymbol{a}' = [\tilde{a}^1, \ldots, \tilde{a}^n]$ with $\tilde{a}^i = \arg\max_{\tilde{a}} Q^i(\tau^i, \tilde{a}, s'; \boldsymbol{\theta})$, and $\boldsymbol{\theta}^-$ denotes the target network parameters which are periodically copied from $\boldsymbol{\theta}$ and remain constant for N_r iterations.

VI. Performance Evaluation

To evaluate the effectiveness of the proposed QLBT channel access algorithm, simulation results under different scenarios are provided in this section. The scenarios include saturated traffic, unsaturated traffic, delay-sensitive traffic, dynamic environments and coexistence with Wi-Fi network. In the following, we will first introduce the simulation setup and performance metrics, and after that the experimental results under each scenario will be presented in details.

A. Simulation Setup

We consider a wireless network where multiple stations use the same MAC protocol to compete for channel access. The smallest time unit in the simulation is *time slot*, which is $9\mu s$ compatible with Wi-Fi standard. Two types of traffic model are studied in the subsequent simulations: Poisson traffic and delay-sensitive (VoIP) traffic. For Poisson traffic, packet arrival follows a Poisson distribution with arrival rate λ (here λ is the average number of packets arrived per second); whereas for VoIP traffic, packet arrives periodically with a period of T_{vo}. The packet length is $1080\mu s$ and $540\mu s$, i.e., 120 slots and 60 slots, for Poisson and VoIP traffic, respectively. The default queue-limit is 10 packets.

To get a comprehensive performance comparison, we introduce a series of algorithms as baselines. They contain different access categories (ACs) in Wi-Fi Enhanced Distributed Channel Access (EDCA) methods, such as AC_VO, AC_VI and AC_BE,[2] whose parameters are shown in Table I. They also contain the theoretical upper bound[3] of CSMA/CA in order to show the capability of QLBT breaking through the limitation of random access method. Moreover, independent DQN algorithm is developed to show the non-stationary issue. The parameters of QLBT and other baselines are given as follows.

[2]Note that AC_BK has the same minimum and maximum contention windows with AC_BE, thus it is omitted here.

[3]It is only employed as performance comparison and very difficult to achieve due to the strong assumption of the same packet length, the same access parameters and full buffer of all stations.

TABLE I
SIMULATION PARAMETERS OF EDCA ACs

Parameters	AC_VO	AC_VI	AC_BE
SIFS (μs)	18		
DIFS (μs)	36		
CW_{min}	7	15	31
CW_{max}	15	31	1023

TABLE II
HYPER-PARAMETERS OF QLBT

Parameters	Value
Agent action-observation history length M	10
Experience memory size N	500
Discount factor γ	0.5
Batch size bs	32
Learning rate	5×10^{-4}
Replace target iteration N_r	100
Range of ε	1 to 0.01
Decay rate of ε	0.998
Weight of individual loss β	n

1) Parameters of QLBT: As illustrated in Fig. 2b, there are 32 neurons in GRU layer and FC layer of each agent network. The hidden layers of the mixing network and the hypernetwork used to generate the final bias also contain 32 neurons. The experience memory stores the latest 500 experiences, in which the length of agent action-observation history is 10. The weight of each individual loss, batch size and discount factor in loss function Eq. (9) are set to n, 32 and 0.5, respectively. The RMSProp optimizer is employed with a learning rate of 5×10^{-4}. The target network is updated after every 100 training steps. Every agent performs an ε-greedy action selection, where ε is initially set to 1 and decay at a rate of 0.998 until reaching 0.01. The hyper-parameters of QLBT are summarized in Table II.

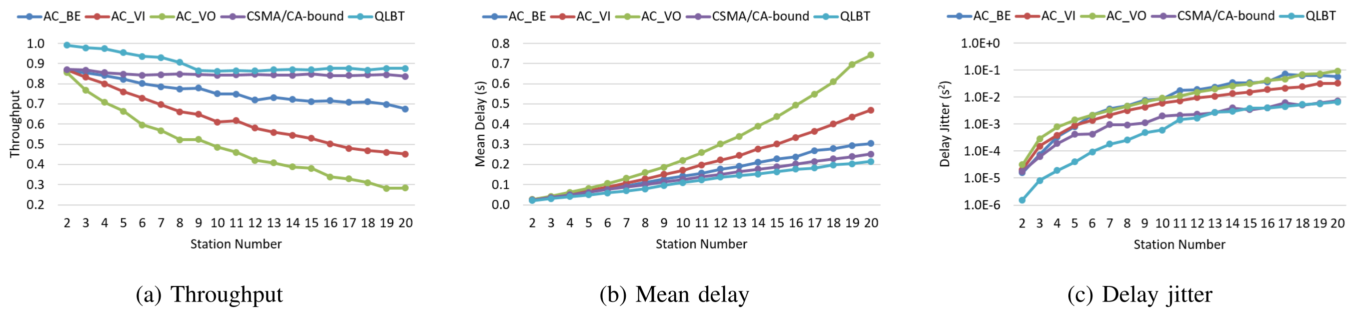
2) Parameters of Independent DQN: Independent DQN serves as the baseline for a class of methods that decompose multi-agent problem into multiple single-agent problems. In order to make the simulation results comparable, the neural network architecture of each agent is the same as that of QLBT. The reward function for agent i at time t is defined as

$$r_t^i = \begin{cases} 1, & \textit{if any agent transmits successfully,} \\ -1, & \textit{if agent i transmits but fails,} \\ 0, & \textit{otherwise.} \end{cases}$$

3) Parameters of CSMA/CA Performance Bound: The optimal contention window to achieve CSMA/CA performance bound can be obtained by solving equations (28) and (46) in [7] and we omit the tedious derivation due to limited space. The key parameters therein are set as follows. The holding time in collision state τ_F equals to the packet length plus a DIFS. The maximum backoff stage K is set to 5, which aligns with AC_BE in Wi-Fi standard.

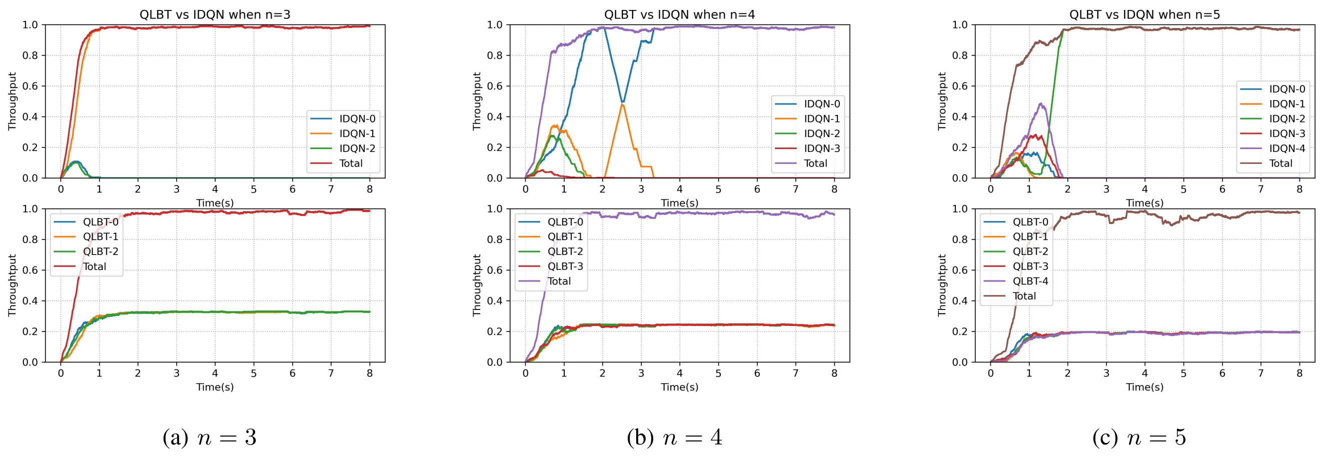
B. Performance Metrics

The following metrics are used to evaluate the performance of the proposed algorithm.



(a) Throughput (b) Mean delay (c) Delay jitter

Fig. 3. Performance comparison under saturated Poisson traffic. A log scale is used for the Y axis in (c).



(a) $n = 3$ (b) $n = 4$ (c) $n = 5$

Fig. 4. Performance comparison between independent DQN and QLBT. Upper and lower figures show the individual and total throughput of independent DQN and QLBT, respectively.

- *Throughput*: In figure of "Throughput vs. Station Number", throughput represents the steady-state throughput after convergence, which is defined as the ratio of successful transmission slots over the last second of the simulation. Whereas in figure of "Throughput vs. Time", throughput represents the instantaneous throughput at time slot t, which is defined as the ratio of successful transmission slots over past N_{thp} slots, where $N_{\mathrm{thp}} = 50000$ by default.
- *Delay*: Delay of a packet is defined as the number of time slots since the packet is generated to the packet is successfully transmitted.
- *Mean delay*: Mean delay is defined as the average of delay of all successfully transmitted packets.
- *Delay jitter*: Delay jitter is defined as the variance of delay of all successfully transmitted packets.

C. Scenario I: Saturated Traffic

In this subsection, we investigate the performance under saturated traffic scenario with arrival rate $\lambda = 2000$. In comparison of QLBT with AC_VO, AC_VI, AC_BE and CSMA/CA performance bound (CSMA/CA-bound for short), the throughput, mean delay and delay jitter as the number of stations increases are shown in Fig. 3a, Fig. 3b and Fig. 3c, respectively. As it can be seen, the proposed QLBT algorithm achieves better performance in terms of all three metrics than all EDCA ACs and even CSMA/CA-bound.

Moreover in Fig. 4, we compare QLBT with independent DQN (marked as IDQN) and show their real-time throughput performance under the setups of different network sizes. In the case of IDQN, for all studied setups, there always exists one station monopolizing the channel, so the fairness among stations cannot be guaranteed. On the contrary, when using the proposed QLBT algorithm, a cooperative behavior convergence can *always* be reached.

D. Scenario II: Unsaturated Traffic

To better examine the performance of the proposed QLBT in different scenarios, we focus on an unsaturated traffic special case in this subsection. Concentrating on a network with 4 stations, each one has the same packet arrival rate, varying from 100, 200 to 400. For simplicity, only AC_BE is taken in as baseline as it performs best over all EDCA ACs in Scenario I. The cumulative distribution functions (CDFs) of delay with respect to QLBT and AC_BE under different arrival rates are shown in Fig. 5. The mean delay performance of both QLBT and AC_BE deteriorate as the packet arrival rate increases. Moreover, given the same packet arrival rate, the proposed QLBT algorithm achieves smaller mean delay than AC_BE, and the performance gain becomes large as the network approaches saturated.

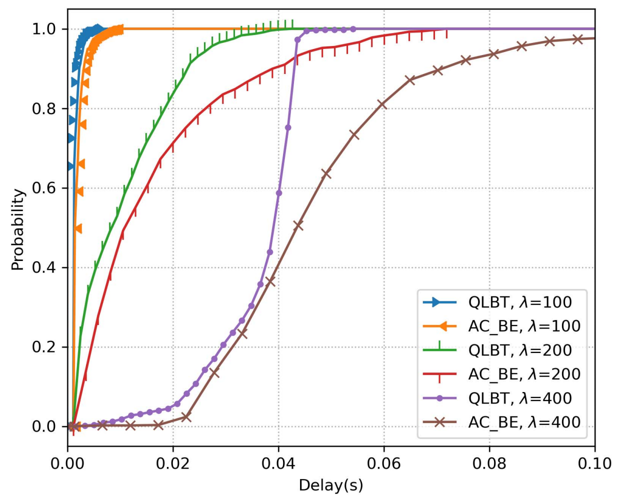


Fig. 5. Delay performance comparison between QLBT and AC_BE under unsaturated traffic scenario.

E. Scenario III: Delay-Sensitive Traffic

Besides the Poisson traffic model, we evaluate the performance of QLBT under the periodically generated delay-sensitive traffic, which is also under discussion in standard body of next generation Wi-Fi [36]. In this case, we fix the number of stations as 20 and set the generation period of VoIP traffic as $T_{\text{vo}} = 20ms$. The CDF of delay and the instantaneous throughput with respect to AC_VO, AC_BE and QLBT are shown in Fig. 6. It can be observed that both AC_BE and AC_VO suffer large mean delay and delay jitter. The better performance of AC_BE compared with AC_VO is due to the fact that AC_VO adopts a smaller contention window than AC_BE, which results in more collisions for large network size. Accordingly, if we map VoIP traffic to the highest priority access category, i.e., AC_VO, as usually done in practice, the latency will be intolerable. In contrast, as shown in Fig. 6c, the mean delays of stations using QLBT are generally small and upper-bounded by 0.4s. The slope of the CDF curve for QLBT is much sharper than that of AC_BE and AC_VO, indicating that QLBT is able to achieve stable delay performance. Hence, the proposed QLBT algorithm can provide the best QoS delivery to delay-sensitive traffic.

F. Scenario IV: Dynamic Network Size

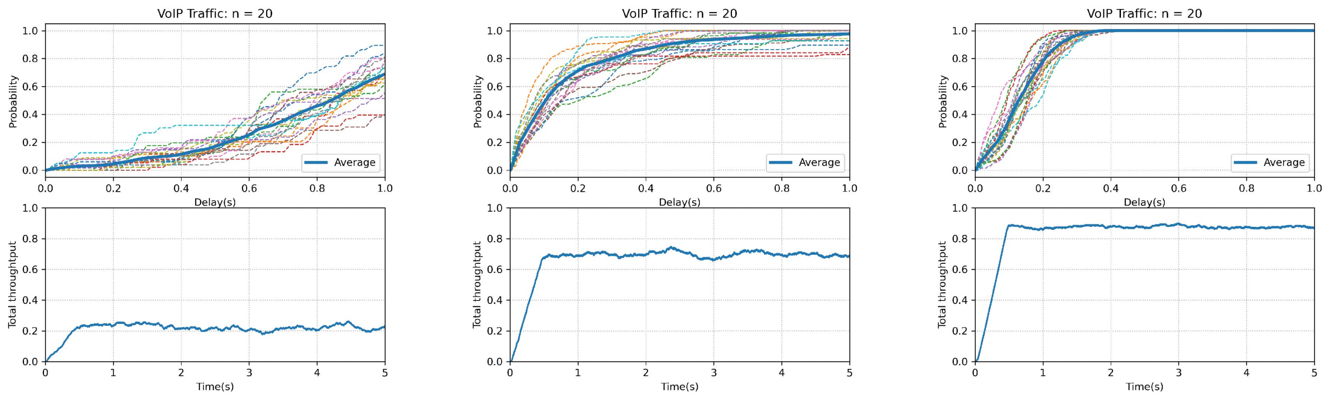
In this subsection, we evaluate the performance of the proposed QLBT algorithm under dynamic network size. The detailed scenarios are as follows. The total simulation time is 50s and there exist stations joining or leaving the network every 10s. Specifically, there are two stations in the network at the beginning of the simulation. After 10s, another station joins the network and stays for 10s before leaving. At 30s, three stations joins the network and one of them leaves the network 10s later. The saturated Poisson traffic with $\lambda = 2000$ is considered for all stations. The simulation result is shown in Fig. 7. It can be observed that the proposed QLBT algorithm is able to quickly respond to environmental changes. Note that at each time the network size changes, the agent network parameters of each station will be initialized by a pre-trained model with the corresponding network size. For instance, there are two stations in the network at the beginning of the simulation and the agent network parameters are initialized using a pre-trained 2-agent QLBT model. At 10s, another station joins the network. In this case, the agent network parameters are reinitialized using a pre-trained 3-agent QLBT model. In practical implementation, AP is aware of the number of connected stations, so the agent network parameter update can be achieved via beacon frames carrying such information. Alternatively, each station can store the pre-trained neural network models locally and update the parameters accordingly when AP notifies the network size change.

G. Scenario V: Dynamic Traffic

Apart from considering the dynamic network size, we also evaluate the performance of the proposed QLBT algorithm under dynamic traffic condition in this subsection. Different from previous case, we consider a fixed network size $n = 4$ under time-varying Poisson traffic. The total simulation time is 40s. The packet arrival rate λ is the same for each station but changes from 400 to 200, 200 to 100 and 100 to 400 at 10s, 20s and 30s, respectively. The simulation result is shown in Fig. 8. At the beginning of the simulation, the agent network parameters are initialized by a pre-trained 4-agent QLBT model under saturated Poisson traffic with $\lambda = 2000$ and there is no further training and parameter update in the following simulations. It can be observed that the proposed QLBT algorithm is able to consistently achieve the best QoS delivery, the dashed gray line in Fig. 8, under dynamic traffic conditions.

H. Scenario VI: Coexistence With Wi-Fi

When deploying the proposed QLBT channel access scheme in practice, it is inevitable to meet the coexistence issue with incumbent Wi-Fi network. The 3GPP describes the notion of fairness as the protocol design should target fair coexistence with existing Wi-Fi networks to not impact Wi-Fi services more than an additional Wi-Fi network on the same carrier with respect to throughput and latency referring in [37]. Following this definition, we replace Wi-Fi stations in a network with QLBT stations to verify whether QLBT is able to fairly coexist with Wi-Fi network. We consider a network with 4 stations and each station is under unsaturated Poisson traffic with arrival rate $\lambda = 200$. The simulation results are presented in Fig. 10, where Fig. 10a shows the delay CDF and instantaneous throughput when all stations in the network are Wi-Fi stations and Fig. 10b–Fig. 10d show the coexistence performance when gradually substituting Wi-Fi stations by QLBT stations. It is worth noting that the proposed QLBT algorithm can achieve 3GPP fairness when coexisting with incumbent Wi-Fi network. Observed from the figures of delay CDF, Wi-Fi stations in coexistence scenario outperform those in homogeneous Wi-Fi network in terms of delay and jitter performances. Moreover, the total throughput of coexisting networks is slightly improved compared to the homogeneous Wi-Fi network. As a result, we can conclude that importing QLBT is also beneficial to the “legacy” Wi-Fi network.



(a) EDCA AC_VO (b) EDCA AC_BE (c) QLBT

Fig. 6. Performance comparison under VoIP traffic when station number is 20. Upper figure presents the CDF of delay. Lower figure presents the real-time network throughput of all stations.

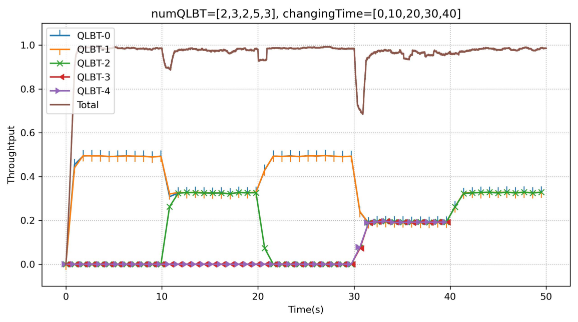


Fig. 7. Real-time throughput under dynamic network size. QLBT stations join and leave the network every 10 seconds.

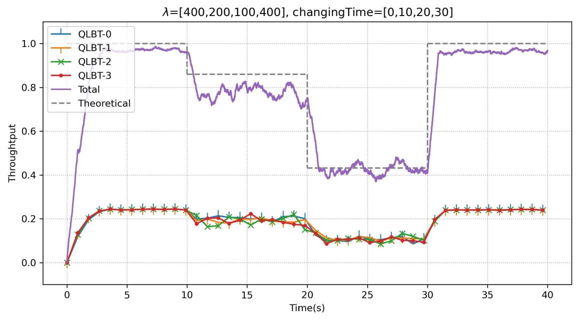


Fig. 8. Real-time throughput under dynamic traffic conditions. The packet arrival rate changes every 10 seconds.

I. Convergence and Complexity

In this subsection, we discuss the convergence performance and the computational complexity of the proposed QLBT algorithm. Fig. 9 shows the trends of loss and total reward during the training process of the proposed QLBT when there are 4 stations in the network. The plotted total reward is averaged over last 500 time slots. It can be observed from the figure that the loss and the total reward converge to 0 and 1 respectively after 2000 training iterations. The convergence time is less than 2s. Remind that this is a realistic time in practical implementation since we set the simulation parameters, e.g., slot length and packet length, according to Wi-Fi standard.

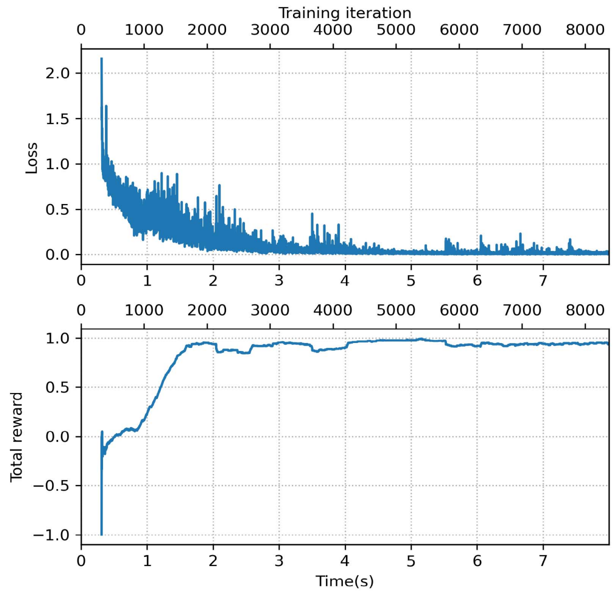
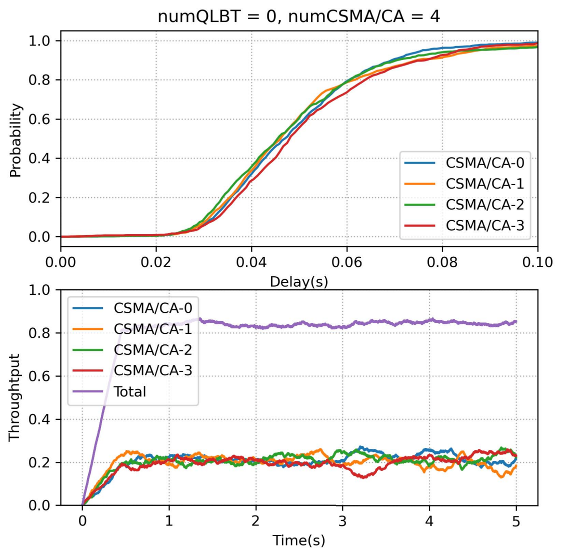
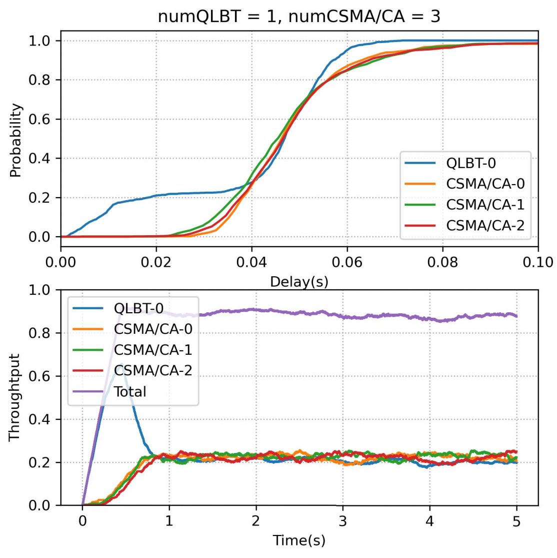


Fig. 9. Convergence performance of the proposed QLBT algorithm when station number is 4.

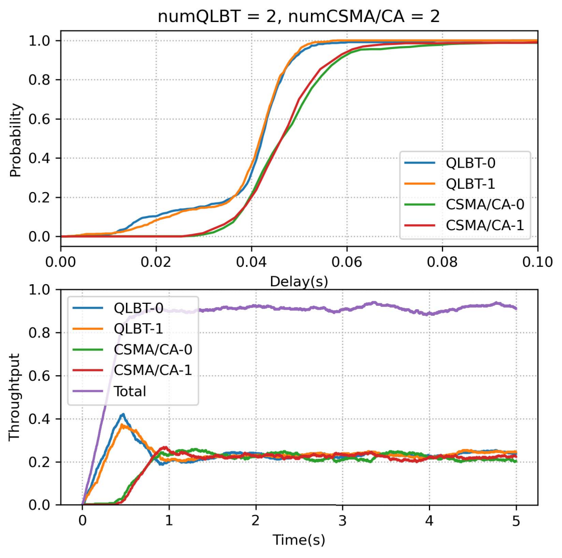
For the network architecture adopted in our work, i.e., Fig. 2b, the computational complexity of inference is 9440 FLOPs for each agent, which is similar to that of a 256-point complex fast Fourier transform (FFT) which is 10240 FLOPs. Compared to CSMA/CA mechanism, the proposed QLBT indeed leads to increased complexity since CSMA/CA only needs to perform random number generation and countdown operations. Actually, all AI-based solutions may result in higher complexity due to the large computing power requirements of neural networks. However, better performances, i.e., higher throughput, smaller delay and delay jitter, could be achieved using such type of methods. As illustrated in Fig. 3, the proposed QLBT algorithm performs much better than EDCA used in Wi-Fi. Moreover, we compare the complexity of the proposed QLBT with the most related work [20]. The neural network adopted in [20] is a 6-layer fully connected network with residual mappings. Each layer



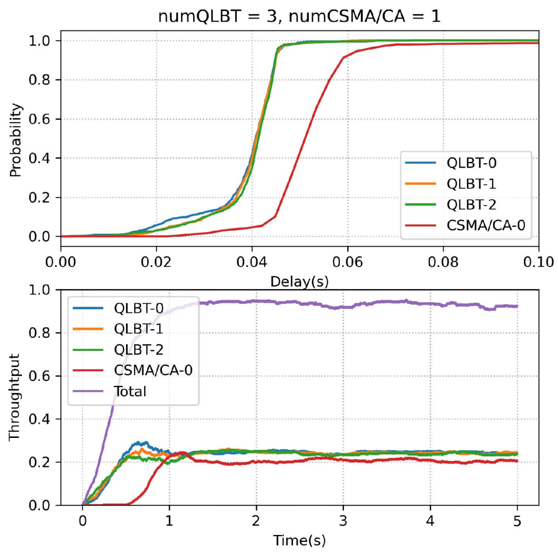
(a) numQLBT=0, numCSMA=4



(b) numQLBT=1, numCSMA=3



(c) numQLBT=2, numCSMA=2



(d) numQLBT=3, numCSMA=1

Fig. 10. Real-time throughput and delay CDF when QLBT coexists with Wi-Fi.

has 64 neurons. Using such a network, the computational complexity of inference is 45454 FLOPs for each agent, which is similar to that of a 1024-point complex FFT (51200 FLOPs). The complexity of the proposed algorithm in [20] is larger than that of the QLBT proposed in our work. It is also worth noting that the computational complexity is closely related to the neural network architecture, which can be further reduced by optimizing the network architecture.

VII. Conclusion and Future Work

In this paper, we proposed a new MAC protocol based on MARL. The proposed algorithm, called QLBT, employed centralized training with decentralized execution to improve network performance in terms of throughput, delay and jitter. A novel reward function and agent action-observation history were designed in QLBT. We compared the QLBT performance with classic CSMA/CA and EDCA used in 802.11 Wi-Fi standard. Experimental results showed that QLBT outperformed several competing protocols in Poisson and VoIP traffic. To be specific, QLBT achieved higher throughput, lower mean delay and lower delay jitter than other methods in saturated Poisson traffic and delay sensitive traffic. With the release of 6GHz spectrum to Wi-Fi, it opens up an opportunity to consider such a native-AI algorithm as the basic rule for spectrum sharing.

Moving forward, we are interested in the design of efficient channel access scheme under hidden node scenarios.

The decision of reinforcement learning is based on the efficient observation, however, the hidden node problem may "blind the eye". In this case, the agent needs to learn to take such information into account to achieve robust performance.

Another possible research direction is the design of more efficient training methods. Centralized training requires large amount of information exchange and computational resources, which becomes inefficient when the network size is large or the environment is highly dynamic. Solutions to this challenge are worth considering.

Last but not the least, it would be interesting to extend QLBT to cross-layer decision-making problems, such as the joint design of channel access and rate adaption selection, channel access and transmission power control, etc., to fully exploit the power of DRL for MAC layer design in the next generation wireless networks.

References

[1] *6G: The Next Horizon: From Connected People and Things to Connected Intelligence*. Cambridge, U.K.: Cambridge Univ. Press, 2021.

[2] A. Colvin, "CSMA with collision avoidance," *Comput. Commun.*, vol. 6, no. 5, pp. 227–235, Oct. 1983.

[3] X. Sun and L. Dai, "Backoff design for IEEE 802.11 DCF networks: Fundamental tradeoff and design criterion," *IEEE/ACM Trans. Netw.*, vol. 23, no. 1, pp. 300–316, Feb. 2015.

[4] T. Li, T. Tang, and C. Chang, "A new backoff algorithm for IEEE 802.11 distributed coordination function," in *Proc. 6th Int. Conf. Fuzzy Syst. Knowl. Discovery*, 2009, pp. 455–459.

[5] J. Barcelo, A. L. Toledo, C. Cano, and M. Oliver, "Fairness and convergence of CSMA with enhanced collision avoidance (ECA)," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–6.

[6] J. Barcelo, B. Bellalta, C. Cano, A. Sfairopoulou, M. Oliver, and K. Verma, "Towards a collision-free WLAN: Dynamic parameter adjustment in CSMA/E2CA," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, no. 1, pp. 1–11, Dec. 2011.

[7] L. Dai and X. Sun, "A unified analysis of IEEE 802.11 DCF networks: Stability, throughput, and delay," *IEEE Trans. Mobile Comput.*, vol. 12, no. 8, pp. 1558–1572, Aug. 2013.

[8] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On selfish behavior in CSMA/CA networks," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 4, 2005, pp. 2513–2524.

[9] C. Bowyer, D. Greene, T. Ward, M. Menendez, J. Shea, and T. Wong, "Reinforcement learning for mixed cooperative/competitive dynamic spectrum access," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Nov. 2019, pp. 1–6.

[10] A. P. F. de Figueiredo, R. Mennes, X. Jiao, W. Liu, and I. Moerman, "A spectrum sharing framework for intelligent next generation wireless networks," *IEEE Access*, vol. 6, pp. 60704–60735, 2018.

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

[12] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[13] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[14] M. Lauer and M. Riedmiller, "Reinforcement learning for stochastic cooperative multi-agent systems," in *Proc. 3rd Int. Joint Conf. Auto. Agents Multiagent Systems*, vol. 3, 2004, pp. 1516–1517.

[15] W. Wydmański and S. Szott, "Contention window optimization in IEEE 802.11ax networks with deep reinforcement learning," in *Proc. 2021 IEEE Wireless Commun. Netw. Conf. (WCNC),* 2021, pp. 1–6, doi: 10.1109/WCNC49053.2021.9417575.

[16] A. Kumar, G. Verma, C. Rao, A. Swami, and S. Segarra, "Adaptive contention window design using deep Q-learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 4950–4954.

[17] Y. Yu, S. C. Liew, and T. Wang, "Non-uniform time-step deep Q-network for carrier-sense multiple access in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2848–2861, Sep. 2021.

[18] Y. Yu, S. C. Liew, and T. Wang, "Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels," *IEEE Trans. Mobile Comput.*, early access, Feb. 9, 2021, doi: 10.1109/TMC.2021.3057826.

[19] J. Tan, L. Zhang, Y.-C. Liang, and D. Niyato, "Intelligent sharing for LTE and WiFi systems in unlicensed bands: A deep reinforcement learning approach," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2793–2808, May 2020.

[20] L. Zhang, H. Yin, Z. Zhou, S. Roy, and Y. Sun, "Enhancing WiFi multiple access performance with federated deep reinforcement learning," in *Proc. IEEE 92nd Veh. Technol. Conf. (VTC-Fall)*, Nov. 2020, pp. 1–6.

[21] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, Oct. 2019.

[22] Y. Sinan Nasir and D. Guo, "Deep actor-critic learning for distributed power control in wireless mobile networks," in *Proc. 54th Asilomar Conf. Signals, Syst., Comput.*, Nov. 2020, pp. 398–402.

[23] F. H. Costa Neto, D. Costa Araujo, M. Pontes Mota, T. Maciel, and A. de Almeida, "Uplink power control framework based on reinforcement learning for 5G networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5734–5748, Jun. 2021.

[24] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.

[25] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[26] R. Balakrishnan, K. Sankhe, V. S. Somayazulu, R. Vannithamby, and J. Sydir, "Deep reinforcement learning based traffic- and channel-aware OFDMA resource allocation," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[27] U. Challita and D. Sandberg, "Deep reinforcement learning for dynamic spectrum sharing of LTE and NR," 2021, *arXiv:2102.11176*.

[28] Y. Chen *et al.*, "Reinforcement learning meets wireless networks: A layering perspective," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 85–111, Jan. 2021.

[29] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2nd Quart., 2021.

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

[31] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[32] Y. Yang and J. Wang, "An overview of multi-agent reinforcement learning from game theoretical perspective," 2020, *arXiv:2011.00583*.

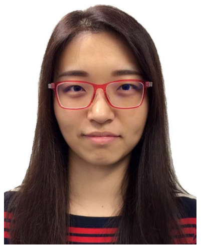
[33] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs* (SpringerBriefs in Intelligent Systems). Springer, 2016. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-319-28929-8.pdf, doi: 10.1007/978-3-319-28929-8.

[34] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 330–337.

[35] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 2094–2100.

[36] P. Torab, C. Hu, and M. K. Haider, "Prioritized EDCA channel access," document IEEE 802.11-20/1045r3, 2020.

[37] X. Sun and L. Dai, "Towards fair and efficient spectrum sharing between LTE and WiFi in unlicensed bands: Fairness-constrained throughput maximization," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2713–2727, Apr. 2020.



Ziyang Guo (Member, IEEE) received the B.Eng. degree (Hons.) from the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2014, and the Ph.D. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2018. From September 2016 to December 2016, she was a Visiting Student with the School of Engineering and Applied Sciences, Harvard University. In September 2018, she joined Huawei Technologies Company Ltd., where she is currently a Senior Engineer. Her research interests include multi-agent reinforcement learning, short-range wireless communication, cyber-physical system security, and network estimation and control.

Zhenyu Chen received the B.E. degree in communication engineering from the School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen, China, in 2021, where he is currently pursing the M.E. degree in information and communication engineering with the School of Electronics and Communication Engineering. From January 2021 to June 2021, he was an Intern at the Wireless Technology Lab, 2012 Laboratories, Huawei Technologies Company Ltd. His research interests include multiple access and reinforcement learning.

Peng Liu (Member, IEEE) received the B.S. and Ph.D. degrees in telecommunication engineering from Xidian University, Xi'an, China, in 2010 and 2015, respectively. He was a Visiting Scholar at Columbia University, New York, NY, USA, from 2013 to 2014, under the supervision of Prof. Wang. He is currently a Senior Researcher with Huawei Technologies Company Ltd., Shenzhen. His research interests include short-range wireless communications, AI for Wi-Fi, and wireless physical layer security.

Jianjun Luo received the B.Sc. degree from the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China, in 2014, and the M.Sc. degree from the Faculty IV–Electrical Engineering and Computer Science, Technische Universität Berlin, Berlin, Germany, in 2018. Since December 2018, he has been with the Wireless Technology Lab, 2012 Laboratories, Huawei Technologies Company Ltd. His research interests include deep learning, model acceleration, and spectrum sharing techniques.

Xun Yang (Member, IEEE) received the B.S. and Ph.D. degrees in communications from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2003 and 2008, respectively. In July 2008, he joined Huawei Technologies Company Ltd., as a Research Engineer, and then became a Principal Engineer in May 2017. He has been working on IEEE 802 wireless standards since 2010, including 802.11ac, 802.11ah, 802.11ax, 802.11ba, 802.11be, and 802.15.4ab. His research interests include system design of short range wireless (e.g., WLAN, UWB, etc.) and its related applications. He was the Secretary of 802.11ac, the PHY ad-hoc Co-Chair of 802.11ah, and the MU ad-hoc Co-Chair of 802.11ax. He is the TG Vice Chair of 802.15.4ab.

Xinghua Sun (Member, IEEE) received the B.S. degree from the Nanjing University of Posts and Telecommunications (NJUPT), China, in 2008, and the Ph.D. degree from the City University of Hong Kong (CityU), China, in 2013. In 2010, he was a Visiting Student with the National Institute for Research in Digital Science and Technology (INRIA), France. In 2013, he was a Post-Doctoral Fellow at CityU. From 2015 to 2016, he was a Post-Doctoral Fellow with The University of British Columbia, Canada. From July 2019 to August 2019, he was a Visiting Scholar with the Singapore University of Technology and Design, Singapore. From 2014 to 2018, he was an Associate Professor with NJUPT. Since 2018, he has been an Associate Professor with Sun Yat-sen University, China. His research interests are in the areas of stochastic modeling of wireless networks and machine learning for networking. He served as a technical program committee member of numerous IEEE conferences.