Differentiated QoS Provisioning in Wireless Networks Based on Deep Reinforcement Learning

Ming-Chu Chou[1], Guang-Jhe Lin[1], Chih-Heng Ke[1,*], Yeong-Sheng Chen[2]

[1]Department of Computer Science and Information Engineering, National Quemoy University, Kinmen, Taiwan, ROC

[2]Department of Computer Science, National Taipei University of Education, Taipei, Taiwan, ROC

Received 29 April 2024; received in revised form 03 August 2024; accepted 06 August 2024

DOI: https://doi.org/10.46604/aiti.2024.13655

Abstract

Wireless networks manage performance by adjusting the contention window, as they cannot directly detect collisions. Traditional contention window adjustment algorithms, such as the Binary Exponential Backoff (BEB) algorithm, may lead to lower throughput when multiple services with varying bandwidth demands coexist. To address this issue, this study aims to enhance network throughput by enabling differentiated bandwidth allocation for various services. Using deep reinforcement learning, the state space, action space, and reward functions are defined to optimize this differentiation. These definitions are integrated into the Deep Deterministic Policy Gradient (DDPG) technique, implemented in the Access Point (AP) to intelligently adjust the contention window. Leveraging DDPG's capability for continuous actions, the proposed method provides Quality of Service (QoS) differentiation, ensuring that each service at its respective priority level meets its transmission requirements. Compared to the BEB algorithm, the proposed approach offers improved traffic allocation and higher network bandwidth utilization.

Keywords: 802.11 wireless networks, Quality of Service (QoS), contention window, deep reinforcement learning

1. Introduction

In the field of IEEE 802.11 wireless networks, several challenges emerge in satisfying diverse user requirements. Given the varied application scenarios for different users, the network system must render multi-tiered Quality of Service (QoS). For instance, certain users might be engaged in high-definition video streaming, where excellent QoS is crucial. In contrast, other users might only be involved in file transfers, which is relatively not dependent on service quality. This diversity necessitates the formulation of differentiated QoS definitions based on specific usage contexts to ensure the network can flexibly address a panoply of application requirements. Therefore, this paper utilizes the Deep Deterministic Policy Gradient (DDPG) algorithm [1] to adjust the contention window (CW) according to different user needs, ensuring the delivery of services that align with specific user experience expectations.

The CW was initially introduced by IEEE due to the inability of nodes to detect collisions in wireless network environments. Technically, nodes tend to wait for a random backoff time to stagger the transmission times of nodes. This waiting time is determined by the CW of nodes. When a collision occurs, to eschew another collision, the network exponentially increases the CW of nodes, thereby increasing the backoff time. Conversely, when a node successfully transmits, the CW is reduced back to the minimum value allowed by the network standard, which decreases the node's backup time. This algorithm is called the Binary Exponential Backoff (BEB) algorithm. From the collision avoidance mechanism, it can be inferred that as the CW increases, nodes must wait longer for data transmission. As a result, such a design may compromise

* Corresponding author. E-mail address: smallko@gmail.com

English language proofreader: Chih-Wei Chang

the channel access rights of nodes. Therefore, this paper utilizes deep reinforcement learning (DRL) technology to appositely adjust the CW of nodes, empowering high-level clients to possess higher channel access rights, thereby achieving differentiated treatment when dealing with clients of different priorities.

The main contribution of this paper lies in proposing a mechanism that utilizes DRL to adjust the CW, thereby providing different priorities of QoS for each tier of clients. The structure of this paper is listed as follows: Section 2 introduces the DRL mechanism, DDPG, used in this study, along with previous research on enhancing QoS through DRL. Section 3 explains the specific parameter settings for DRL and the operational workflow of the overall research method. Following this, the paper presents research results to validate its contributions in Section 4. Finally, the conclusion in Section 5 discusses future optimization directions.

2. Related Work

This section explores recent QoS-related research to highlight the indispensability of QoS. Specifically, the necessity of providing different bandwidths is discussed, based on varying priorities within the context of QoS issues. After introducing the QoS topic, it presents pertinent studies on adjusting the CW to enhance QoS, supporting the feasibility and significance of the subject. Finally, it introduces the technologies most relevant to this paper.

2.1. QoS issue

QoS has consistently been a crucial topic, as can be witnessed by Li et al. [2] proposing a multi-dimensional Internet of Things (IoT) QoS estimation method that introduces multi-dimensional QoS to assess the QoS of IoT applications. The approach is illustrated using IoT application instances to demonstrate the process of estimating the QoS of IoT applications. Kang et al. [3] present a diversified QoS-centric service recommendation method tailored for uncertain QoS preferences, generating a list of services that fulfill the desired QoS and diversity. In recent years, a surge has emanated from research introducing the multiple-access edge computing (MEC) framework to enhance network QoS. Wang et al. [4] introduce a QoS prediction method that considers user mobility and QoS data fluctuations to adapt to the MEC environment. Yan et al. [5] deployed historical QoS values as a time series of QoS matrices. It combines a compressed matrix extracted from the QoS matrix through truncated singular value decomposition (SVD) with the traditional Autoregressive Integrated Moving Average (ARIMA) model.

Luo et al. [6] propose a QoS-oriented multiple Unmanned Aerial Vehicle Mobile Base Station (UAV-MBS) 3-D deployment algorithm, named QoS-prior. This algorithm optimizes the altitude and coverage radius of UAV-MBSs based on different QoS requirements. Numerical results show that the algorithm outperforms some baseline algorithms under constraints and achieves relatively low time complexity. Zhang et al. [7] propose a stochastic walk-based interval prediction method for Web service QoS, called a random walk, effectively combining location awareness and collaborative filtering methods. To render more reliable simulations, the authors conducted a series of comprehensive experiments on a real Web service dataset. The results demonstrate that the proposed method achieves a better balance between confidence and accuracy in service recommendation compared to other collaborative filtering methods. However, notwithstanding the betterment yielded by these studies, effectively differentiating adjustments based on priorities is perceived as an urgent need for QoS.

2.2. The issue of adjusting the contention window according to different transmission services

When a network simultaneously hosts high-priority nodes with high bandwidth demands and low-priority nodes with low bandwidth demands, the issue of adjusting the CW according to different transmission services is especially worthy of notice. High-priority nodes need to continuously compete with other nodes to satisfy their bandwidth needs. During the period of contention, transmission collisions will cause the CW to increase. Conversely, low-priority nodes, with their lower bandwidth requirements, do not need to frequently compete for the channel, thereby resulting in a lower CW. This situation signifies and causes high bandwidth demand nodes to have lower throughput, compared to low bandwidth demand nodes.

2.3. Contention window

To provide transmission services at different priorities, this paper adopts the technique of adjusting CW. The adjustment of CW to maximize overall network utilization has been constantly discussed. Lu et al. [8] propose a strategy called Transmission Rate-based Contention Window Adaptive Adjustment (TRCWAA) to achieve fair resource allocation. Technically, the TRCWAA strategy adjusts the minimum CW size of each node inversely proportional to its transmission rate. Simulation experiments demonstrate that the TRCWAA strategy effectively addresses the issues of abnormal performance and unfairness. Furthermore, numerous studies have also incorporated artificial intelligence techniques to adaptively adjust CW.

For example, Yang et al. [9] describe a clustering routing optimization strategy based on an ant colony algorithm, constrained by the CW. Specifically, the values of CW for non-uniformly loaded nodes in clustered networks are analyzed, and a cluster head selection strategy, considering CW, network interference, and remaining energy, is proposed. Experimental results demonstrate that the Ant Colony Cluster Optimization based on the Contention Window (ACCO-CW) strategy effectively improves the success rate of data transmission. However, the method of computing Q-values using reinforcement learning cannot cope with highly complex network environments. Therefore, several research has utilized neural network technology to approximate and compute complex Q-values within it.

Li and Jian [10] address the CW selection problem for IEEE 802.11 networks oriented towards the Age of Information (AoI) using the DDPG technique. The results show that this method reduces the average AoI of the system by approximately 57.6%, compared to the traditional BEB method. Jiang and Zheng [11] no longer merely use regular neural networks to compute Q-values. Instead, the recurrent neural networks (RNNs) with deep recurrent Q-networks (DRQN) are adopted. This paper studies the initial CW size adjustment problem in a coexisting network with new radio unlicensed (NR-U) and WiFi systems. It adaptively finds the optimal CW size by training the primary deep Q-network (DQN) in a DRQN and observing the current status of the coexisting network, including the current CW size, throughput of the WiFi system, throughput of the NR-U system, and the number of NR-U users with data to transmit. Experimental results signify that this method outperforms both fixed CW mechanisms and adaptive CW mechanisms.

2.4. Enhancing QoS with contention window

Based on the aforementioned QoS research, this paper aims to enhance network performance and furnish with differentiated services through CW adjustment. Erstwhile QoS studies have focused on adjusting the CW to improve service quality, accentuating the feasibility of manipulating QoS through CW control. Suvarna et al. [12] propose a novel framework based on a divide-and-conquer strategy for setting CW, achieving efficient QoS even with more nodes in the network. Toral-Cruz et al. [13] introduce a QoS-differentiated backoff mechanism that doubles the CW during collisions and the channel perceived as busy. While doubling the CW during busy channel periods may reduce collisions between competing stations, it may increase the backoff delay for delay-sensitive applications (such as voice and video). Aldawibi et al. [14] investigate the performance of backoff CW in randomly moving Mobile Ad Hoc Networks (MANETs) using major routing protocols, e.g., dynamic source routing (DSR), destination-sequenced distance-vector routing (DSDV), and ad hoc on-demand distance vector (AODV).

The study implemented key parameters such as network throughput, power consumption, and network latency using the Network Simulator (NS)-2, with in-depth discussions of the experimental results. Rasna et al. [15] assess the effectiveness of the CW by examining the impact of increased communication channel signal interference and the signal-to-noise ratio (SINR) in the presence of hidden nodes. Notwithstanding the superior performance demonstrated in these research papers, a limited exploration of using CW adjustment to provide differentiated services is observable. Kwon et al. [16] utilize reinforcement learning techniques to provide different transmission services for various priorities and clients. Given such a praxis, this paper combines reinforcement learning to propose a CW adjustment method for Wireless Body Area Networks (WBANs), abbreviated as reinforcement learning contention window adjustment (RL-CWA). RL-CWA maintains multiple Q-tables to account for the varying minimum and maximum CW based on the user priority (UP) of the traffic. Experimental simulations demonstrate that the performance of RL-CWA is superior to the existing IEEE 802.15.6 MAC.

2.5. *Research methodology insights*

As artificial intelligence techniques proceed to become mature, numerous studies in the field of wireless networks have increasingly incorporated AI to address complex issues. Within the field of artificial intelligence, DRL represents its remarkable performance in addressing wireless network issues. Ergun et al. [17] explore the application of DRL techniques in the context of 5G, highlighting its significance. This research mentions that reinforcement learning does not rely on precise environment modeling and decision-making, which helps reduce the complexity and cost of solutions, thereby enhancing system responsiveness. Furthermore, when faced with unknown wireless network states and conditions, reinforcement learning can be employed to address non-convex optimization and optimization of mutually coupled variables, resulting in superior performance compared to other machine learning methods.

Based on DRL, Wydmański and Szott [18] propose the centralized contention window optimization with the DRL (CCoD) framework, as depicted in Fig. 1. In this framework, the collision rate within the overall network is considered as the state space for DRL using the DQN [19] agent loaded within the AP. The output value obtained adjusts the CW, influencing all stations to adapt their CW through AP broadcasting. This architecture serves as a crucial inspiration for the work of this paper. Building upon the concept introduced by Ke and Astuti [20], where high-rate nodes enjoy higher channel access, this study utilizes the DDPG algorithm to generate multiple consecutive actions. With the centralized single-agent CCoD framework, the proposed method allocates varying CW based on customer classes to achieve different levels of channel access privilege.

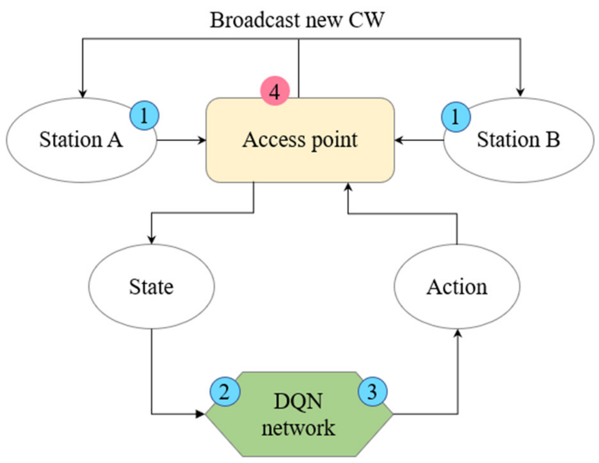

Fig. 1 CCoD architecture

2.6. *DDPG introduction*

The DDPG algorithm, proposed in the paper of Lillicrap et al. [1], is a DRL approach designed for continuous action spaces. Combining deterministic policy gradient methods with deep neural networks, DDPG addresses reinforcement learning challenges in high-dimensional, continuous action spaces. Pervasively recognized in the field of DRL, DDPG is particularly suitable for tasks requiring continuous control of the environment, such as robot control, autonomous driving, and resource allocation.

The DDPG architecture, depicted in Fig. 2, encompasses feeding the state space into the actor network and its corresponding target network (actor target network). Subsequently, the actions of the actor network (Action) and the state space are input into the critic network. The target action from the actor target network is utilized to control the environment. Simultaneously, the actions and state are input into the target network (critic target network) below the critic network. After obtaining Q-values from both the critic network and its target network, along with the reward, mathematical calculations yield the loss values for both the critic network and the actor network. These values are subsequently used to update the neural network weights. Finally, at regular intervals, the actor network and critic network each update their respective target networks.

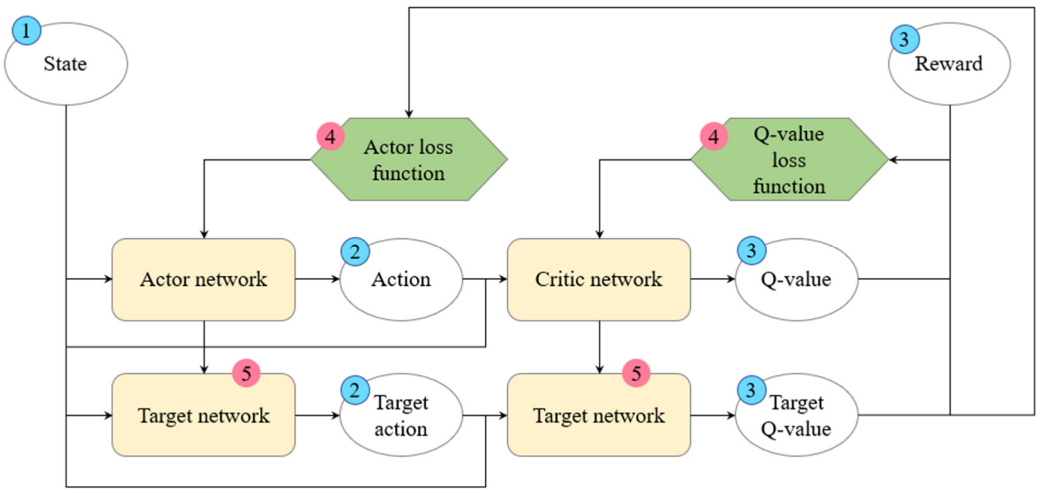

Fig. 2 DDPG architecture

The critic evaluates the network loss function using the following equation [1]:

$$Qloss = \left[Reward_i + r\hat{Q}\left(S_{i+1}, A_{i+1}\right) - Q\left(S_i, A_i\right) \right]^2 \tag{1}$$

where $Qloss$ is the output of the Q-value loss function, r is the discount rate, S is the state of the environment, A is the action of the actor network, $\hat{Q}$ and Q represent the critic target network and the critic evaluation network, respectively. To enable the critic network to yield more accurate Q-values based on the output actions of the actor, reward values are considered. This is achieved by reducing the impact of environmental noise through temporal difference (TD) error learning with the target network.

Also, the actor evaluates the network loss using the following equation [1]:

$$Aloss = -\frac{1}{n}\sum_{i=1}^{n} \hat{Q}\left(S_i, A_i\right) \tag{2}$$

where $Aloss$ is the output of the actor loss function, and n represents the number of training samples. The actor aims to execute actions for the optimal strategy, quantifying the policy to form Q-values as a loss function. This approach seeks to balance exploration and exploitation in reinforcement learning by maximizing Q-values.

After network evaluation, the actor and critic update their target networks using the following equations, respectively [1]:

$$\theta_{\text{Critic}}^{\text{Target}} = \tau\theta_{\text{Critic}}^{\text{Eval}} + \left(1-\tau\right)\theta_{\text{Critic}}^{\text{Target}}, \ \tau < 1 \tag{3}$$

$$\theta_{\text{Actor}}^{\text{Target}} = \tau\theta_{\text{Actor}}^{\text{Eval}} + \left(1-\tau\right)\theta_{\text{Actor}}^{\text{Target}}, \ \tau < 1 \tag{4}$$

where θ represents neural network parameters, and τ is a user-determined variable designed to ensure the target networks are updated with a fixed probability.

3. Research Method

This section introduces the system model of the proposed method, detailing the design of the observation space, action space, and reward function in DRL. The method is illustrated through an algorithm based on this model. Notably, this paper proposes and compares two different reward functions, which will be discussed in Section 4.

3.1. System model

This study employs a centralized architecture, as shown in Fig. 3, with the DDPG agent installed in the wireless AP. The AP obtains collision rate information from the global network as the state-space input for the agent. Subsequently, the AP transforms the actions (Action1 and Action2) output by the actor neural network of the agent into new CW values (CW1 and CW2). These values are then broadcast in Beacon frames to all nodes. High-priority client nodes use CW1, while low-priority client nodes use CW2. The critic neural network of the agent takes the current state space, Action1, and Action2 as input and outputs the Q-value to record the state transitions and decision quality of the Markov decision process. Finally, the AP synthesizes global network throughput performance and client transmission rate information to derive the reward value for optimizing the neural network of the agent.

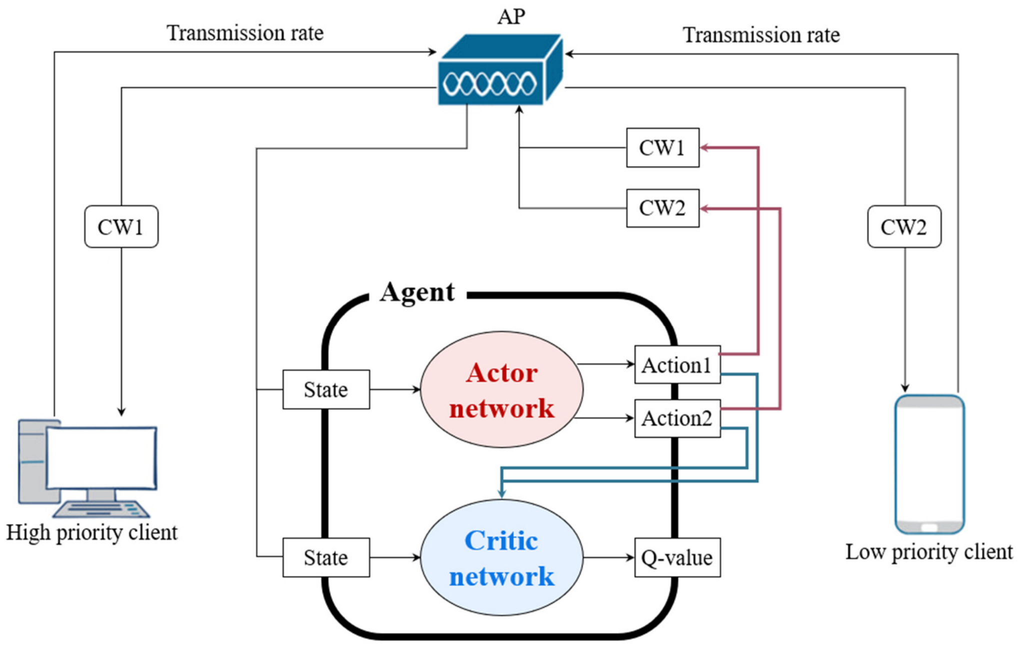

Fig. 3 System model

3.2. Observation space

The state space inputs the actor and critic neural networks in the DDPG agent. This study concentrates on the network collision rate in the last four intervals, aiming to expedite training convergence by ignoring the fractional collision rate.

$$S_t = \{Coll_{t-1}, Coll_{t-2}, Coll_{t-3}, Coll_{t-4}\} \tag{5}$$

$$Coll_t = T_t^{coll} / T_t \tag{6}$$

$$T_t = T_t^{succ} + T_t^{coll} \tag{7}$$

Eq. (5) represents the state S_t of the node at time *t*, where $Coll_t$ indicates the collision rate of the node at time *t*. The calculation of $Coll_t$ is given by Eq. (6), where T_t^{coll} symbolizes the number of collisions from *t*–1 to t, divided by the total number of transmissions by the node from *t*–1 to *t*, T_t, as calculated in Eq. (7).

3.3. Action space

The action produced by the DDPG agent's actor neural network consists of two real numbers, each ranging from 0 to 1, as defined by:

$$Action_i = Actor\left(S_i \middle| \theta_{Actor}^{Trarget}\right) \tag{8}$$

Based on state S_i, the actor determines the target network parameters θ_{Critic}^{Target} and generates a continuous action $Action_i$.

With the deployment of Eqs. (1)-(8), the AP can derive appropriate CW values for both high-priority and low-priority client nodes. However, these CW values must be constrained within a range of 16 to 1024, as specified by the 802.11a network standard. Therefore, the following two formulas are further applied.

$$CW_i = clip\left[Normal\left(Action_i, \sigma\right), 16, 1024\right] \tag{9}$$

$$clip\left(x, \min, \max\right) = \begin{cases} \min, \text{ if } x < \min \\ x, \text{ if } \max \geq x \geq \min \\ \max, \text{ } if \text{ } x < \max \end{cases} \tag{10}$$

In Eq. (9), a Gaussian random distribution is applied to slightly perturb $Action_i$, mitigating the effects of network temporality. The parameter σ is a user-defined hyperparameter. Eq. (10) constrains the CW to be between 16 and 1024.

3.4. Reward function

The reward value, a crucial metric for optimizing the actor neural network of the agent, is computed in Eq. (10) based on the actions executed in the current state. To ensure that high-priority clients receive superior treatment, the AP monitors the throughput of both high-priority and low-priority clients. The reward function is designed to provide higher rewards when the throughput difference between the two client categories approaches the desired gap.

$$R_t = Throughput_t \Big/ \frac{1}{N}\left(\sum_{i=1}^{N} rate_i\right) \times \left(-Diff_{Thr}^{Client}\right) \tag{11}$$

$$Diff_{Thr}^{Client} = \sum_{i=1}^{N}\sum_{j=1}^{N} \beta_j^i Thr_t^j - \alpha_j^i Thr_t^i, \ i \neq j \tag{12}$$

$$Reward_t = Thr_t - abs\left(Diff_{Thr}^{Client}\right) \tag{13}$$

Eq. (11) normalizes the overall network by using the average rate for two main reasons. Firstly, it ensures that the reward values are kept between 0 and 1, which helps prevent gradient explosion during the training process. Secondly, controlling the range of reward values intensifies the difference in reward values between various throughputs while still managing the range of these rewards. As a result, the generalization ability of the agent is increased. The parameter $Diff_{Thr}^{Client}$ calculates the throughput difference $Thr_t^i - Thr_t^j$ between different-level clients using Eq. (12).

The goal is for the DDPG agent to learn and minimize the throughput differences between clients of different priorities, aiming for a fixed multiple β_j^i expected by the system. However, Eq. (11) tends to trap the agent in local optimal solutions when facing higher transmission rate standards. Therefore, based on Eq. (11), this study designs a new reward function, Eq. (13). By subtracting the absolute value of the throughput from Eq. (12), this research method enables the agent to output CW values based on throughput, while keeping the throughput gap between customers of different priorities close to the system's expected value.

3.5. Algorithm

Table 1 illustrates the meanings of the parameters within the algorithm. The first half of the algorithm corresponds to the agent training phase. During this stage, the agent renders a fixed probability of outputting random actions to explore previously untried actions. Simultaneously, the AP records the overall network collision rate for the first four collisions as input to the state space of the agent.

The output actions, adjusted by the AP, determine the CW values for high-priority and low-priority customer nodes. Each time a collision or successful transmission occurs, the nodes adjust their CW based on their priorities. Subsequently, at regular intervals, the agent optimizes the actor and critic networks through observation of these actions. These behaviors continue in a loop until the end of the training period. After the training phase, the model enters the testing stage, which is similar to the training phase, with the difference that the agent no longer explores. The agent only outputs actions that the actor believes will maximize the Q-value.

Algorithm 1

```
1: AccessPoint AP
2: AP.agent = AP.CreateAgent()
3: Nodes N = {node1, node2, ..., nodei}
4: now = 0
5: proc Training(): //Training Phase
6:   while now < TrainingTime:
7:       state = AP.agent.state
8:       action = AP.agent.act(state)
9:       AP.newCW1, AP.newCW2 = Eq. (9)
10:      for node in N:
11:        if node.Priority == 'High'
12:          node.CW = newCW1
13:        end if
14:        else node.Priority == 'Low':
15:          node.CW = newCW2
16:        end else if
17:      end for
18:      reward = Eq. (11) or Eq. (13)
19:      for {state, reward, nextState, nextAction} in AP.agent.RelpyBuffer:
20:          AP.agent.update (state, reward, nextState, nextAction)
21:      end for
22:    end while
23: end proc
24: now = 0
25: proc Testing(): //Testing Phase
26:   while now < TestingTime:
27:      state = AP.agent.state
28:      action = AP.agent.act(state)
29:      AP.newCW1, AP.newCW2 = Eq. (9)
30:      for node in N:
31:        if node.Priority == 'High':
32:          node.CW = newCW1
33:        end if
34:       else node.Priority == 'Low':
35:         node.CW = newCW2
36:       end else if
37:      end for
38:      reward = Eq. (11) or Eq. (13)
39:   end while
40: end proc
```

Table 1 Parameter description of Algorithm 1

Argument	Description
AccessPoint	Base station type equipped with a reinforcement learning agent responsible for receiving messages and broadcasting new contention window values.
CreateAgent	Create DDPG Agent Actions
Nodes N	Network Node Set
node.Priority	Node Customer Grading
AccessPoint.agent.state	State space required for observation within the base station.
AccessPoint.agent.act	The agent in the base station selects the optimal action based on the state space.
AccessPoint.agent.ReplyMemory	ReplyMemory is responsible for collecting the state, reward, nextState, and nextAction of the agent, facilitating the updating of actor and critic neural networks.
AccessPoint.agent.update	The intelligent entity at the base station optimizes its functionality.

4. Experimental Results

This section presents experimental results based on the proposed method. In addition to the comparison of overall network throughput, proposed collision rate, and traditional method, it also compares the throughput of different priority levels, demonstrating that the proposed method can render different bandwidths for different services. After comparing the network performance of the proposed method, the paper also evaluates the performance of the two reward functions, explaining which network scenarios each reward function is best suited for based on their performance.

4.1. Experimental design

This study simulates the 802.11a standard network using Python, and the DDPG algorithm implemented on the AP is realized through pfrl and pytorch. The experimental comparison entails evaluating the proposed method against the traditional BEB algorithm apropos overall network throughput, collision rate, and performance across different node priorities. The transmission rate for all nodes in the experimental network environment is set at 48 Mbps. Among these nodes, half are categorized as high-priority clients, while the other half are low-priority clients. The expected experimental results aim for a twofold difference in throughput between the two, with the parameters α_j^i and β_j^i involved in Eq. (12).

4.2. Experimental parameter

Table 2 DDPG parameters

Parameter	Value
ε-greedy	0.1
Discount rate γ	0.09
Optimizer	Adam
Training time	N × 100,000 s
Episode interval	0.1 s
σ	0.025

Table 3 Network parameters

Parameter	Value
Paylaod	1,000 Bytes
Slot time	9 ms
DIFS	34 ms
SIFS	16 ms
CW min/max	16/1024
α_j^i	1
β_j^i	2

Regarding the internal parameters of the agent, as presented in Table 2, the internal critic network of the agent is constructed with a neural network comprising one input layer, three ReLU layers with 128 inputs and 128 outputs each, and one linear output layer. Additionally, the actor network comprises one input layer, three ReLU layers with 128 inputs and 128 outputs each, and one sigmoid output layer, aimed at constraining the actions between 0 and 1, facilitating precise control of the overall network CW range using Eq. (9). Concerning the choice of gamma in DRL, this study compared the values of 0.06, 0.09, and 0.12. Among these values, opting for 0.09 resulted in the throughput gap between high-priority and low-priority nodes being adjacent to twice the difference. Therefore, this paper presents the results for 0.09. In future work, a more in-depth comparison and discussion of the appropriateness of these parameter choices will be conducted. Regarding the network environment configuration, refer to Table 3.

4.3. Network experimental performance

This study contrastively examines experimental results across scenarios with node counts of 10, 20, and 30. Figs. 4-7 respectively present comparisons of the overall throughput, collision rates, throughput for high-priority and low-priority clients, and CW sizes for different priorities between the traditional BEB algorithm and the proposed method. In Fig. 4, the BEB algorithm exponentially increases the CW of a node when collisions occur. Consequently, when the CW renders excessively large, the exponential increase leads to a dramatic rise in the window size. Although this blind increase can effectively reduce collision probability, it sacrifices the channel access rights of numerous nodes, incurring decreased throughput. In contrast, the DRL method proposed in this paper leverages DDPG's ability to output continuous actions, empowering the agent to refine the adjustment of CW. As a result, the throughput performance of this method far exceeds that of the BEB algorithm.

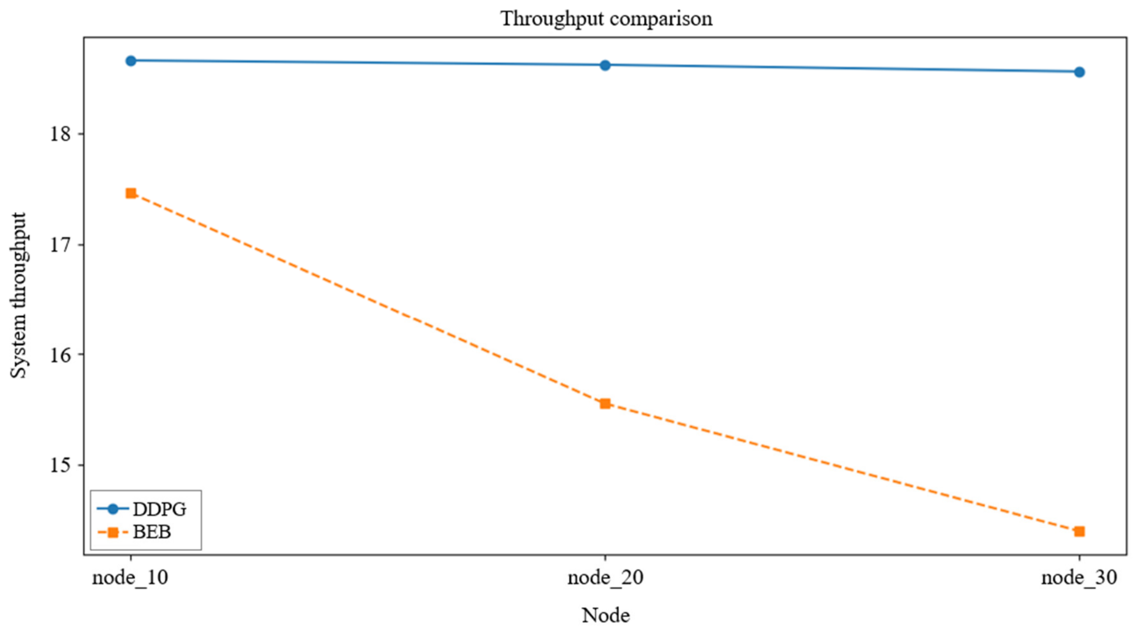

Fig. 4 Throughput comparison

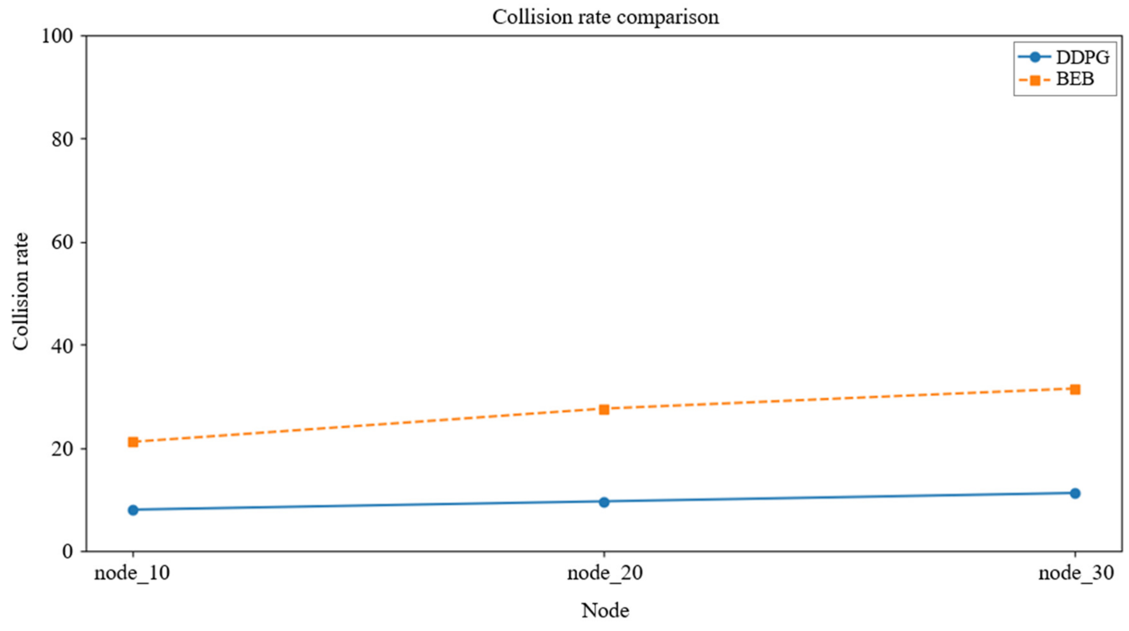

Fig. 5 Collision Rate comparison

Fig. 5 illustrates that the traditional BEB algorithm significantly increases collision rates by drastically reducing the CW to its minimum value after a successful node transmission. This approach becomes unsustainable in networks with a high number of nodes. In contrast, the method proposed in this paper adjusts the CW of the node back to the value observed by the agent after a successful transmission. The agent monitors the network collision rate as a state space, enabling it to effectively mitigate the issue of high collision rates in environments with a large number of nodes.

The focus of this study is to yield different transmission qualities for clients based on their priorities. Fig. 6 illustrates the research approach compared to the traditional BEB algorithm, highlighting throughput performance in a network environment with high-priority and low-priority clients. The BEB algorithm evinces similar throughput for both categories of clients, as it does not adjust CW individually. In contrast, the proposed method in this study, which employs the DDPG model, benefits from the capability of the model to output continuous actions. Such a phenomenon enables the agent to adjust CW values for nodes with different priorities. Additionally, the reward function design in Eq. (10) ensures that the differences in transmission services among clients are controlled within the desired range.

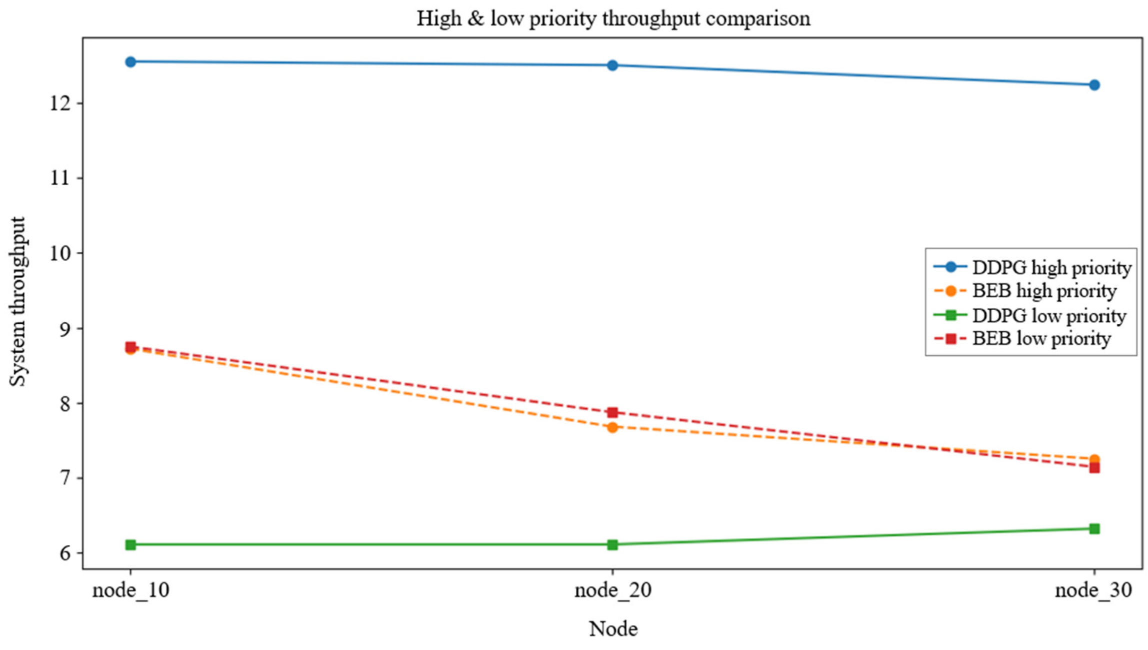

Fig. 6 Throughput comparison of different priority clients

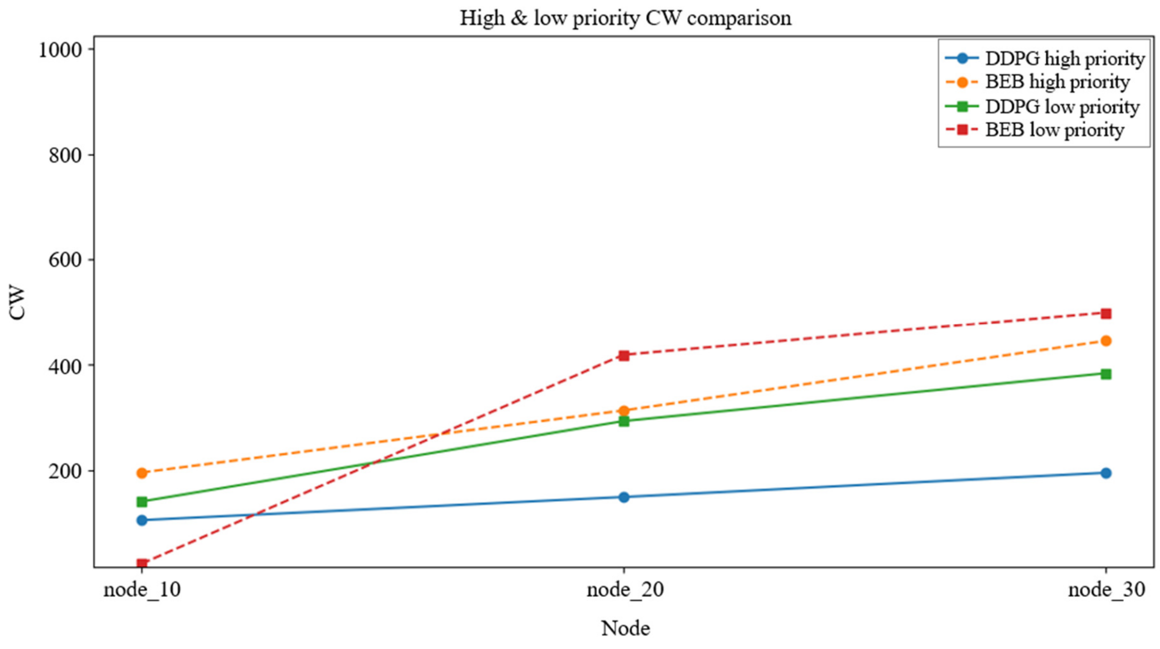

Fig. 7 Contention window of different priority clients

However, numerous factors emerge to affect network transmission quality. Simply adjusting the CW can provide differential services for different nodes, but it may not precisely satisfy the desired service levels. To illustrate the differential adjustments made by the agent, Fig. 7 illustrates the trends in CW adjustments for two types of clients, comparing the BEB algorithm with the proposed method of this study. Due to the limitations of the BEB algorithm, the CWs for both client types initially start low in a scenario with 10 transmission nodes. However, as the network increases to 20 and 30 nodes, the CWs for both client types grow significantly due to the exponential expansion of the contention window under the BEB algorithm. In contrast, the proposed method reduces the CW for high-priority nodes, improving their throughput, while increasing the CW for low-priority nodes through exploration and utilization by the agent. This approach reduces channel access rights without causing significant fluctuations in the CW for high-priority clients as the number of nodes changes.

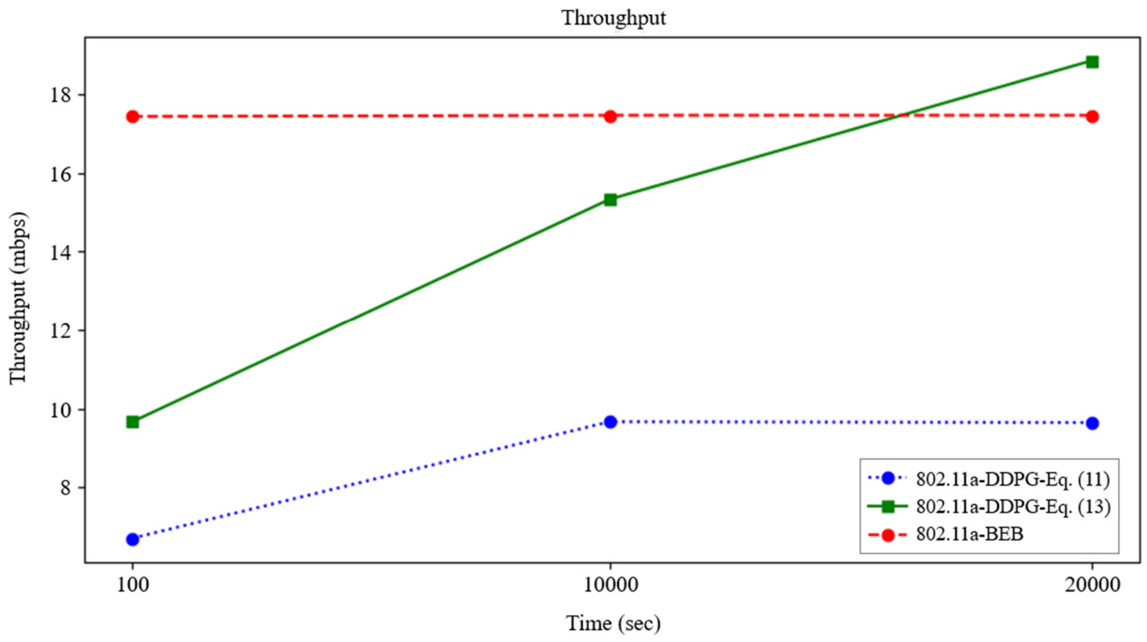

Fig. 8 Throughput comparison in 10 nodes in 100 sec, 10,000 sec, and 20,000 sec

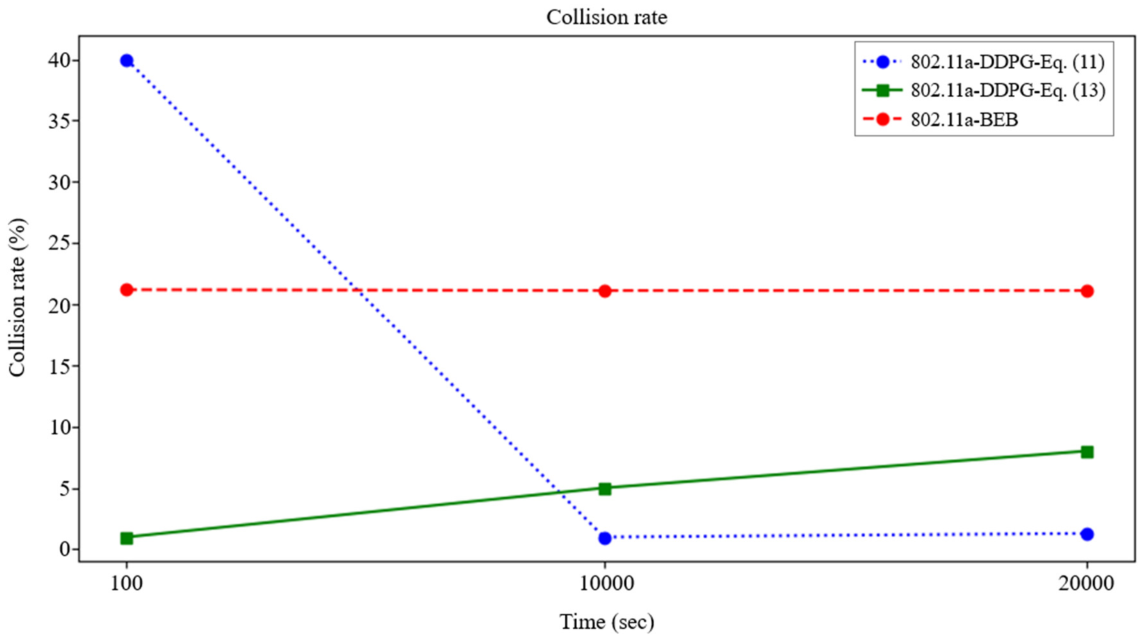

Fig. 9 Collision Rate comparison in 10 nodes in 100 sec, 10,000 sec, and 20,000 sec

Fig. 8 presents a comparison of throughput between the BEB algorithm and using Eqs. (11) and (13) as reward functions for DDPG, with 10 nodes. The BEB algorithm outperforms DDPG agents with either reward function before 20,000 seconds. However, once the DDPG agents are trained with Eq. (13) as the reward function reaches a certain stage, their performance surpasses that of the BEB algorithm. On the other hand, DDPG agents trained with Eq. (11) as the reward function tend to remain at local optimal solutions, incurring even lower throughput performance compared to BEB. Based on the experimental results shown in Fig. 8, it can be observed that DDPG agents using Eq. (13) as the reward function demonstrates the best throughput performance among the three methods.

Fig. 9 presents a comparison of collision rates between the BEB algorithm and the reward functions by Eqs. (11) and (13) for DDPG, with 10 nodes. The DDPG agents using Eq. (13) can increase the overall network throughput by sacrificing a certain degree of individual throughput. In contrast, DDPG agents using Eq. (11) may exhibit lower throughput performance compared to both traditional BEB algorithm and DDPG agents using Eq. (13). However, they are capable of minimizing collision rates by increasing the CW. However, unlike the other two methods, BEB cannot adjust the CW based on the observed collision rates in the overall network, resulting in the poorest collision rate performance among the three. Therefore, based on the experimental results in Fig. 9, it can be observed that DDPG agents using Eq. (11) are suitable for scenarios where collision rates are critical, but high throughput is not required, such as simple IoT devices operating in sensing environments, etc.

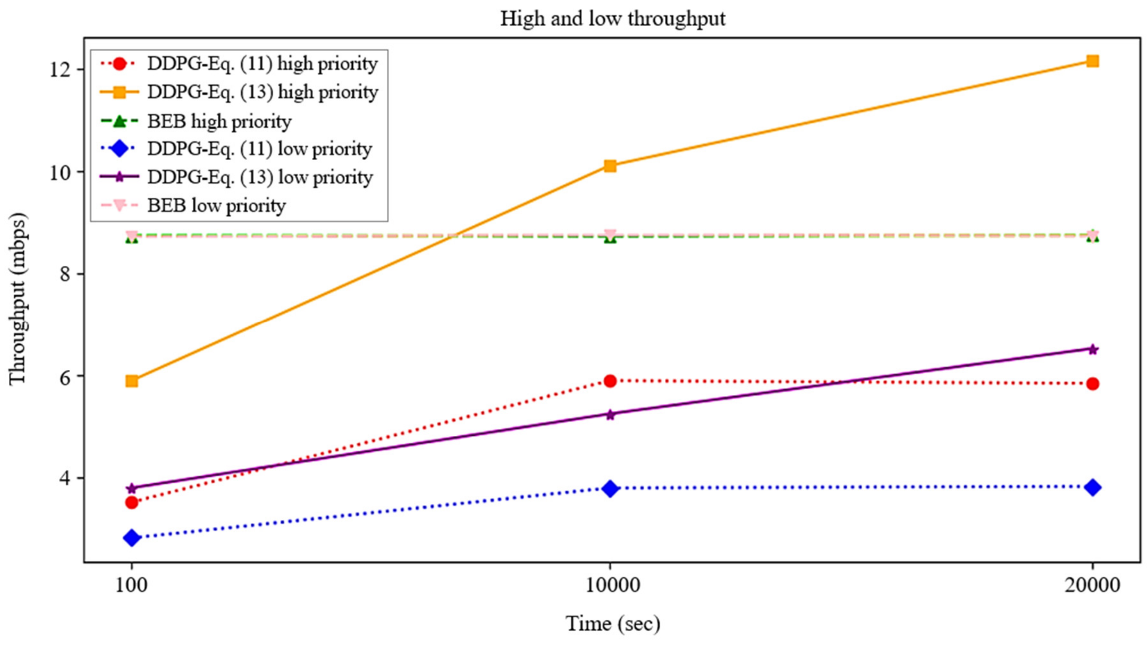

Fig. 10 Throughput comparison of different priority clients

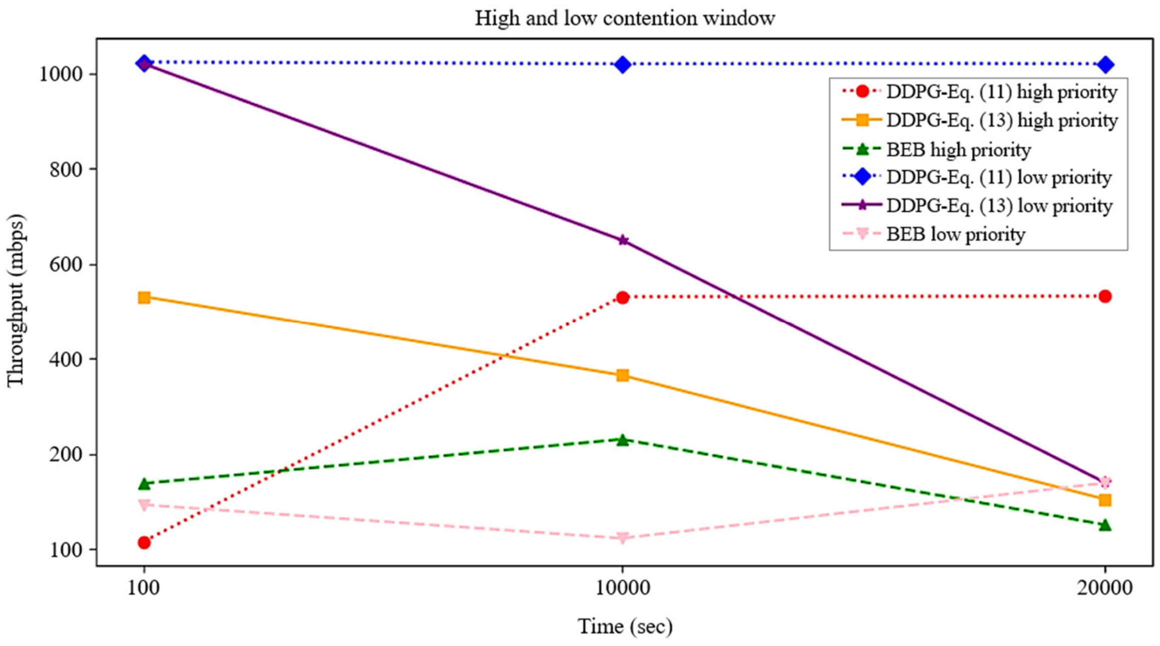

Fig. 11 Contention window of different priorities clients

Fig. 10 and Fig. 11 compare throughput and CW between high-priority and low-priority nodes using Eqs. (11) and (13) as reward functions for DDPG, respectively, with 10 nodes, alongside the BEB algorithm. It is observed that both the DDPG based methods, after sufficient training, can narrow the throughput gap between high-priority and low-priority nodes to match the expectations of the system. In contrast, the traditional BEB algorithm fails to achieve such a result. The experimental results in Fig. 10 further confirm that the proposed method in this study effectively provides differentiated transmission services for different priority levels of customers.

4.4. DDPG experimental performance

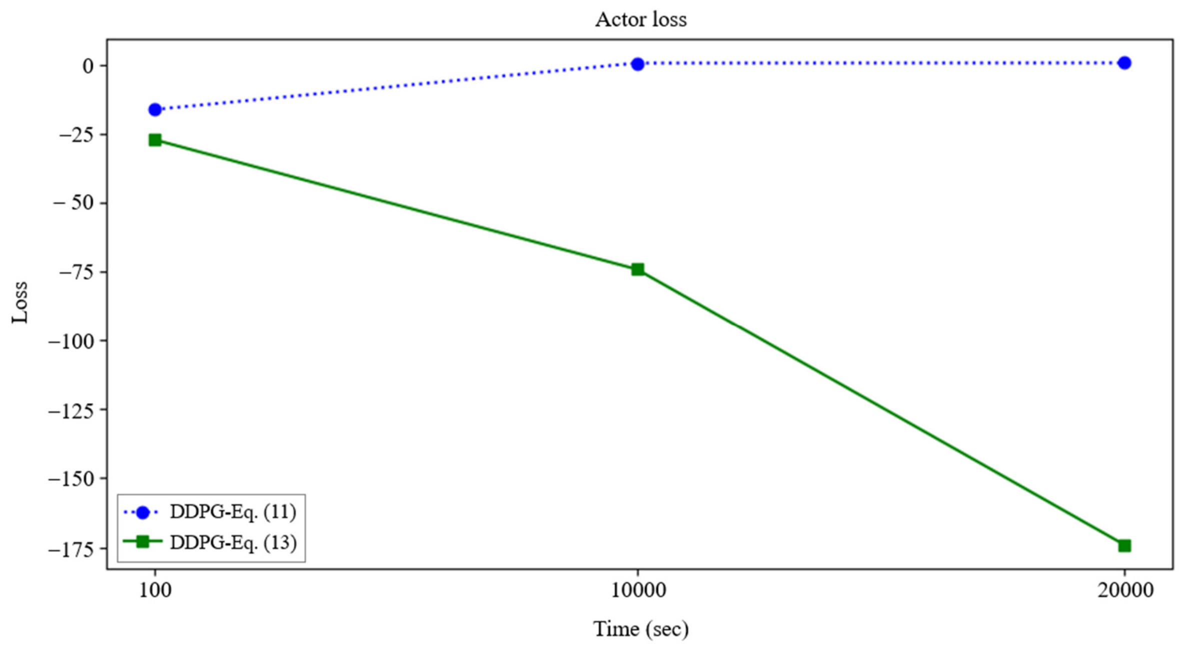

Fig. 12 Actor loss value with Eq. (11) and Eq. (13)

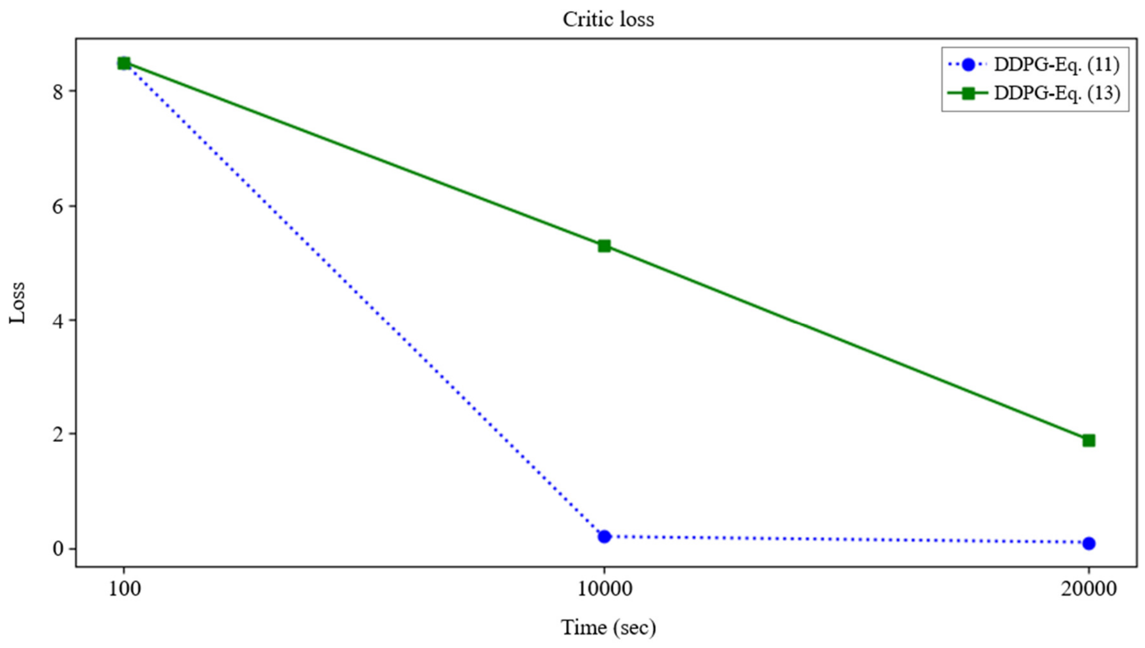

Fig. 13 Critic loss value with Eq. (11) and Eq. (13)

Fig. 12 and Fig. 13 manifest the loss values of the actor and critic, respectively, when using Eqs. (11) and (13) as reward functions for DDPG. Although DDPG agents using Eq. (11) exhibit poor throughput performance, this method enables the actor to converge effectively. On the other hand, while Eq. (13) can achieve high Q-values and improve overall throughput performance, the loss function of the actor in DDPG prevents the neural network of the actor from convergence throughout the training process. This issue of actor non-convergence, despite the improvements in throughput, remains a significant challenge. Therefore, future research will focus on updating the DRL model to mitigate actor interference factors and enhance network performance in more complex scenarios. The experimental results manifest that the proposed DRL approach outperforms the traditional BEB algorithm concerning throughput and collision rates. Additionally, it offers differentiated transmission quality for different priorities.

5. Conclusions

This paper proposes differentiated transmission services for nodes with different priorities in a wireless network scenario. Given such an appeal, therefore, the paper leverages DRL, specifically the DDPG technique, integrated into the AP to provide different QoS for nodes with different priorities. According to experimental results, the proposed method significantly outperforms the traditional BEB algorithm apropos the performance. Moreover, the throughput of nodes is managed to control with different priorities within the expected error range specified in the paper. The proposed method addresses the common issue in wireless networks, where collisions hinder clients from achieving their respective transmission quality. Despite these advancements, the precise control over throughput can yet be improved due to wireless network interference. Future work will involve considering more factors related to network interference to improve throughput control accuracy and referencing recent methods to validate the performance of the proposed approach.

Acknowledgments

This work was partially supported by the National Science and Technology Council, Taiwan, under grant No. NSTC 112-2221-E-507-002.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al., “Continuous Control With Deep Reinforcement Learning,” https://doi.org/10.48550/arXiv.1509.02971, September 09, 2015.

[2] L. Li, M. Rong, and G. Zhang, “An Internet of Things QoS Estimate Approach Based on Multi-Dimension QoS,” 9th International Conference on Computer Science & Education, pp. 998-1002, August 2014.

[3] G. Kang, J. Liu, B. Cao, and Y. Xiao, “Diversified QoS-Centric Service Recommendation for Uncertain QoS Preferences,” IEEE International Conference on Services Computing, pp. 288-295, November 2020.

[4] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. H. Hsu, “QoS Prediction for Service Recommendations in Mobile Edge Computing,” Journal of Parallel and Distributed Computing, vol. 127, pp. 134-144, May 2019.

[5] C. Yan, Y. Zhang, W. Zhong, C. Zhang, and B. Xin, “A Truncated SVD-Based ARIMA Model for Multiple QoS Prediction in Mobile Edge Computing,” Tsinghua Science and Technology, vol. 27, no. 2, pp. 315-324, April 2022.

[6] X. Luo, J. Xie, L. Xiong, Z. Wang, and C. Tian, “3-D Deployment of Multiple UAV-Mounted Mobile Base Stations for Full Coverage of IoT Ground Users With Different QoS Requirements,” IEEE Communications Letters, vol. 26, no. 12, pp. 3009-3013, December 2022.

[7] T. Zhang, S. Wang, M. Tang, and J. Nie, “A Novel QoS Interval Prediction Method for Web Services Based on Random Walk,” 4th International Conference on Computers and Artificial Intelligence Technology, pp. 316-321, December 2023.

[8] S. Lu, S. Hu, Y. Duan, J. Gao, and F. Liu, “Contention Window Adaptive Adjustment Strategy for Fairness in Multi-rate IEEE 802.11 Network,” 6th International Conference on Communication and Information Systems, pp. 82-86, October 2022.

[9] H. Yang, F. Jin, Z. Li, and P. Wen, “Clustering Routing Optimization for Ant Colony Mobile Sensor Networks Based on Contention Window of MAC Layer,” 8th International Conference on Computer and Communication Systems, pp. 304-309, April 2023.

[10] J. Li and F. Jian, "Learning Contention Window Selection in Age of Information-Oriented IEEE 802.11 Networks," IEEE 11th International Conference on Information, Communication and Networks, pp. 903-907, August 2023

[11] S. Jiang and J. Zheng, "A DRQN-Based Initial Contention Window Optimization Algorithm for NR-U and WiFi Coexistence Networks," GLOBECOM 2022 - 2022 IEEE Global Communications Conference, pp. 6019-6024, December 2022.

[12] B. Suvarna, B. LaluNaick, O. Gandhi, and G. VijayKumar, "A Divide and Conquer Technique for the Contention Window to Improve the QoS of MAC," International Conference on Green Computing Communication and Electrical Engineering, pp. 1-6, March 2014.

[13] H. Toral-Cruz, J. Ramirez-Pacheco, P. Velarde-Alvarado, and A. K. Pathan, "VoIP in Next-Generation Converged Networks." Building Next-Generation Converged Networks: Theory and Practice, Boca Raton, FL: CRC Press, pp. 357-380, 2013.

[14] O. O. Aldawibi, S. F. Alahmar, and A. S. Khrwat, "Performance Assessment on Backoff Contention Window for MANETs in Congested Area and Random Movement," IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA, pp. 762-766, May 2021.

[15] Rasna, Indrabayu, Dewiani, and A. Achmad, "Evaluation Performance of Contention Window on the Impact Hidden Node Vehicle to Vehicle," International Seminar on Intelligent Technology and Its Applications, pp. 505-510, July 2023.

[16] J. H. Kwon, D. Kim, and E. J. Kim, "Reinforcement Learning-Based Contention Window Adjustment for Wireless Body Area Networks," 4th International Conference on Big Data Analytics and Practices, pp. 1-4, August 2023.

[17] S. Ergun, I. Sammour, and G. Chalhoub, "A Survey on How Network Simulators Serve Reinforcement Learning in Wireless Networks," Computer Networks, vol. 234, article no. 109934, October 2023.

[18] W. Wydmański and S. Szott, "Contention Window Optimization in IEEE 802.11ax Networks With Deep Reinforcement Learning," IEEE Wireless Communications and Networking Conference, pp. 1-6, March-April 2021.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., "Playing Atari With Deep Reinforcement Learning," https://doi.org/10.48550/arXiv.1312.5602, December 19, 2013.

[20] C. H. Ke and L. Astuti, "Applying Multi-Agent Deep Reinforcement Learning for Contention Window Optimization to Enhance Wireless Network Performance," ICT Express, vol. 9, no. 5, pp. 776-782, October 2023.