J Inf Process Syst, Vol.16, No.5, pp.1048~1063, October 2020
https://doi.org/10.3745/JIPS.03.0148

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Rate Adaptation with Q-Learning in CSMA/CA Wireless Networks

Soohyun Cho*

Abstract

In this study, we propose a reinforcement learning agent to control the data transmission rates of nodes in carrier sensing multiple access with collision avoidance (CSMA/CA)-based wireless networks. We design a reinforcement learning (RL) agent, based on Q-learning. The agent learns the environment using the timeout events of packets, which are locally available in data sending nodes. The agent selects actions to control the data transmission rates of nodes that adjust the modulation and coding scheme (MCS) levels of the data packets to utilize the available bandwidth in dynamically changing channel conditions effectively. We use the ns3-gym framework to simulate RL and investigate the effects of the parameters of Q-learning on the performance of the RL agent. The simulation results indicate that the proposed RL agent adequately adjusts the MCS levels according to the changes in the network, and achieves a high throughput comparable to those of the existing data transmission rate adaptation schemes such as Minstrel.

Keywords

CSMA/CA, ns-3, ns3-gym, Q-Learning, Reinforcement Learning

1. Introduction

Artificial intelligence (AI) technologies are being developed rapidly, and they are increasingly being applied in many areas that can affect daily lives [1,2]. Among the many technologies for AI, we are especially interested in reinforcement learning (RL) [3], as it can be used to manage dynamic or interactive systems such as wireless communication networks [4]. An RL agent uses observations and rewards from its environment in a series of time steps and applies actions to the environment for the next time steps to achieve its final goal. Recent developments and the spread of deep neural network (DNN) [5] technology have led to a boom of AI in many fields of engineering, such as wireless communications [6]. Specifically, by combining and elaborating the deep neural network with RL, the authors of the deep Q-network (DQN) [7] reported that the DQN could outperform human experts in many interactive video games such as Atari 2600 [8].

However, in the present study, we use the basic Q-learning scheme [9] to design an RL agent to control the data transmission rates of nodes in carrier sensing multiple access with collision avoidance (CSMA/CA)-based wireless networks. This is because the scheme is simple, and we assume that the proposed RL

Manuscript received February 5, 2020; first revision April 8, 2020; accepted May 3, 2020.
Corresponding Author: Soohyun Cho (cho.soohyun@hongik.ac.kr)
* Dept. of General Studies, Hongik University, Seoul, Korea (cho.soohyun@hongik.ac.kr)
This is an extended and revised version of a conference paper that was presented by the authors in the 14th KIPS International Conference on Ubiquitous Information Technologies and Applications (CUTE2019), Macau, China.

agent works only with locally available information. Nevertheless, it is necessary to learn the network environment quickly to cope with changes in high-speed wireless networks, especially when the nodes are non-stationary or moving quickly.

Q-learning has been used to solve the finite Markov decision process (MDP) [10,11], particularly when the dimensionality of the state-action pairs is low. It uses a table called Q-table to manage the Q-values that represent the utilities of the state-action pairs for all the states and their possible actions. An RL agent based on Q-learning selects the optimal action for a state based on the Q-values of the state-action pairs available for the state and applies the selected action in the next time step. Subsequently, it updates the Q-table using rewards from the environment. We use the Q-learning and propose an RL agent that adaptively controls the data sending rates of nodes in CSMA/CA wireless networks; this is performed via adapting modulation and coding scheme (MCS) levels of data packets according to the changes in the wireless networks.

Many RL studies use a simulation framework called OpenAI Gym [12] that provides a simulation environment for various applications such as Atari games. Recently, the authors of [13] introduced a framework called ns3-gym that provides a simulation environment resembling that of OpenAI Gym to study RL in network research. Specifically, ns3-gym functions in parallel with ns-3 [14], which is a widely used network simulator. An RL agent can use ns3-gym to learn the environment through observations and rewards from the simulated network, and the actions selected by the RL agent for the next time steps can be applied to the simulated network, as shown in Fig. 1.

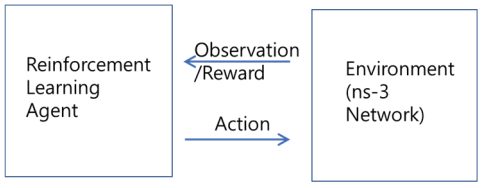

Fig. 1. The ns3-gym framework with ns-3 network simulator.

In this study, we use the ns3-gym framework to develop and evaluate the proposed RL agent in CSMA/CA wireless network scenarios.

The contributions of the study can be summarized as follows:

- We design a novel RL agent based on Q-learning to control the data sending rates of nodes in CSMA/CA wireless networks. The agent controls the data sending rates by adjusting the MCS levels of data packets.
- We demonstrate how the RL agent works using the RL simulation environment provided by the ns3-gym. We use IEEE 802.11a [15] wireless networks in the ns-3 simulator as an example. We also investigate the effects of the parameters of Q-learning on the performance of the RL agent.
- We evaluate the performance of the proposed RL agent by comparing it with the simulation results of existing transmission rate adaptation schemes such as Minstrel [16] and Collision-Aware Rate Adaptation (CARA) [17], which are well-known in the community.

The rest of this paper is organized as follows. Section 2 explains the data transmission rate control mechanism in CSMA/CA wireless networks and demonstrates their performances using the simulation results. In Section 3, we describe the rate adaptation algorithm of the proposed RL agent. In Section 4, we show the performance of the proposed RL agent using simulations. Section 5 investigates the effect

of the parameters used for Q-learning. In Section 6, we compare the performance of the proposed RL agent with existing schemes. Section 7 presents the conclusions of the study.

2. Rate Adaptation in CSMA/CA Networks

For controlling the data transmission rates of nodes in CSMA/CA-based wireless networks, such as the IEEE 802.11 standards [18], the nodes can select a different MCS for their data packets. Table 1 lists the MCS levels and their physical data rates available in IEEE 802.11a, which is used as an example standard in this study.

Table 1. Modulation and coding schemes of IEEE 802.11a

MCS level	MCS	Physical data rate (Mbps)
0	BPSK, 1/2 coding	6
1	BPSK, 3/4 coding	9
2	QPSK, 1/2 coding	12
3	QPSK, 3/4 coding	18
4	16-QAM, 1/2 coding	24
5	16-QAM, 3/4 coding	36
6	64-QAM, 2/3 coding	48
7	64-QAM, 3/4 coding	54

Each of the MCS levels in Table 1 requires a certain signal-to-interference and noise ratio (SINR) for successful reception of the packets of the MCS level at their receivers. A high MCS level requires a high SINR. Therefore, the sender nodes must select adequate MCS levels for the data packets to their respective receivers to achieve high throughput, especially with changes in the channel conditions. Many data transmission rate control algorithms for CSMA/CA wireless networks have been proposed for accurately adjusting the MCS levels [16,17,19]. Minstrel is one of the most popular schemes, partly because of its implementations in real operating systems such as Linux. Minstrel controls data transmission rates by adjusting the MCS levels of data packets using the results from probing data packets at different MCS levels [20]. We use Minstrel to demonstrate the effect of the data transmission rate control in this section.

In Fig. 2, we show the changes in the achieved throughputs by a receiver relative to the distance between a sender and the receiver in ns-3 simulations. In the simulations, the receiver moves away from the sender with a speed of 80 m/s during 15 seconds of simulation. Thus, the distance between the sender and receiver changes from 5 m to 1,205 m. The sender uses different (but fixed) MCS levels or the Minstrel scheme.

As shown in the simulation results in Fig. 2, when a fixed MCS level is used, the sender does not fully utilize the available bandwidth of the network. With high MCS levels (such as 7 and 6), the receiver achieves a high throughput (The achieved throughputs are less than the physical data rates in Table 1 because we measure the application throughput, and the CSMA/CA mechanism consumes bandwidth); however, it could not achieve any throughput beyond 200 m. With low MCS levels (such as 1 and 0), the receiver could receive packets, although it was far from the sender (up to approximately 900 m for MCS level 0). However, it could not achieve a high throughput when the sender and receiver were close to

each other. By contrast, from the simulation results of Minstrel, it is clear that the receiver achieves a high throughput when it is close to the sender. Moreover, the receiver could still receive packets up to a distance of approximately 900 m.

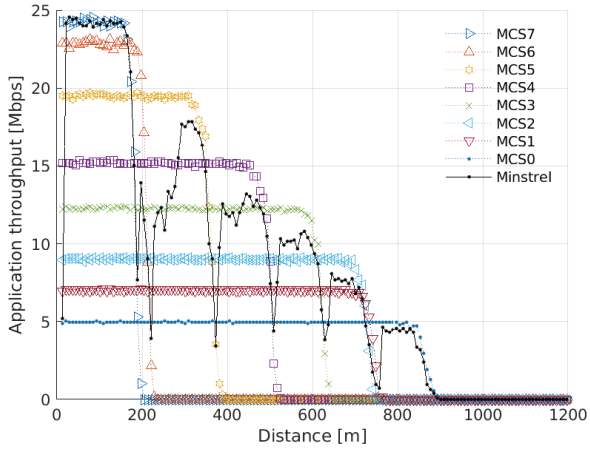

Fig. 2. Changes in application throughputs achieved by the receiver.

For the simulations, we used the IEEE 802.11a implementation in the ns-3 version 3.29. In the simulated network, there are only one sender node and one receiver node. The sender and receiver operate in the distributed coordination function (DCF) [18] mode. During the simulations, the sender node sends a constant bit rate traffic (CBR) of 60 Mbps to the receiver node for 15 seconds. Both the sender and the receiver use 100 mW as the transmit power.

For signal propagation, we use the TwoRayGround propagation model in the ns-3, and fading is not considered for purposes of simplicity. The noise figure is set to 7 dB. To compute the success probability of the received packet, the NistErrorRate model in the ns-3 is used. The data packet size is set to 1,000 bytes, and the request-to-send/clear-to-send (RTS/CTS) scheme is not used.

Table 2 lists the values of the parameters used for the simulations conducted in the study. Other settings are left unchanged from the ns-3 distribution unless otherwise mentioned.

Table 2. Parameters used for simulations

Parameter	Value
Carrier frequency (GHz)	5.18
Channel bandwidth (MHz)	20
Node transmit power (mW)	100
Energy detection threshold (dBm)	-99
Clear channel assessment threshold (dBm)	-82
Noise floor (dBm)	~ -94
Slot time (μs)	9
Guard interval (μs)	3.2
Data packet size (bytes)	1,000
Min. contention window size (CW_{min})	15

Table 2. (Continued)

Parameter	Value
Max. contention window size (CW_{max})	1,023
Antenna height (m)	1.5
Antenna gain (dB)	0
Application traffic	CBR of 60 Mbps
Duplexing	Half
Device queue size (bytes)	100
IP+UDP header size (bytes)	28

3. Rate Adaptation Using RL

In this section, we describe the design and algorithm used for the proposed RL agent to control the data transmission rates of nodes in CSMA/CA wireless networks.

3.1 System Design for RL

The ns3-gym framework that is used for our RL research can report all the network information, such as the queue sizes of all the nodes at each time step, in the form of observation. However, we use only the information that is locally available in the sender node of the CSMA/CA wireless networks, such as the contention window (CW) size. This is because we assume that each node adopts its own RL agent, which operates with limited information locally available in the node, and we consider that a CSMA/CA wireless network in the DCF mode inherently corresponds to a distributed system of multiple participants.

In the DCF mode, each CSMA/CA node selects its random backoff time based on the current CW size to avoid collisions with packets sent by other nodes in the network. When a CSMA/CA node sends a packet for the first time, it sets the CW size to its minimum value (i.e., CW_{min}), which corresponds to 15 in IEEE 802.11a. Whenever an acknowledgment frame from the receiver of the packet does not arrive in a given time (i.e., a timeout event occurs), the sender doubles its CW size before the retry limit is reached (e.g., 7 in the short retry limit case). Thus, the CW size can range from 15 up to 1,023 (i.e., CW_{max}) in IEEE 802.11a with the short retry limit condition. As a result, with n consecutive timeout events, the CW size in a node is given in Eq. (1) as follows:

$$CW = 2^n(CW_{min} + 1) - 1 \tag{1}$$

We consider the CW size as an indicator of the network situation that can be used as the state of the network for the RL agent. However, instead, we use the number of consecutive timeout events (i.e., n) to reduce the size of the Q-table in the RL agent. The number of consecutive timeout events can be computed from the CW size by using Eq. (2) as follows:

$$n = log_2(CW + 1) - 4 \tag{2}$$

We use n as the state that an RL agent obtains in the ns3-gym framework for its node at each time step. The last CW size in each time step is used as the CW value in Eq. (2).

For the rewards, we let sender nodes count the number of acknowledged packets from their receivers during each time step and use it as the reward in the time step. The number of acknowledged packets represents the number of packets successfully delivered to the receivers.

With the given observation of the state and reward for each time step, the RL agent selects the best action (i.e., MCS level) for the state from the Q-table, and the ns3-gym framework applies the selected action to the sender node as the MCS level, which will be used by the sender node during the next time step. The size of the time step is set to 1 ms by considering the simulated network topology and network parameters, such as the frequency (5.18 GHz) and bandwidth (20 MHz), used for the simulations in this study.

3.2 Rate Adaptation Algorithm with Q-Learning

In this section, we describe the rate adaptation algorithm for the RL agent based on the system design described in Section 3.1. The size of the state m for Q-learning is determined by the minimum and maximum contention window size of a given standard as follows:

$$m = log_2((CW_{max} + 1)/(CW_{min} + 1)) + 1 \tag{3}$$

For example, m is 7 for IEEE 802.11a with the short retry limit condition, as there are seven possible states from 0 to 6. The size of actions k for each state is the same as the number of different MCS levels defined in the given standard. Thus, the size of the Q-table for Q-learning is $m \times k$. As an example, the size of the Q-table corresponds to 7×8 for IEEE 802.11a because there are eight possible actions from 0 to 7.

While the RL agent is running, it obtains the observation (i.e., state s) and the reward r from the environment at each time step. The reward corresponds to the number of acknowledgment frames received by a sender during the previous time step. For the given state s, the RL agent selects the best action a for the next time step using the Q-table as given below:

$$a = argmax_a\big(Q_t(s,a)\big), \tag{4}$$

where $argmax_a(\cdot)$ is the function that finds the argument a that maximizes $Q_t(s,a)$ with a given state s (i.e., the action a with the highest Q-value is selected).

The selected action is used as the MCS level for the data packets during the next time step by the sender node. However, to explore other actions allowed for the given state, ϵ-greedy policy [21] is used, where ϵ is termed as the exploration rate. Thus, the action selected for a state can be a random integer between 0 and $k-1$ with a probability governed by the current value of ϵ. The exploration rate decreases from its initial value of 1.0 to its minimum value ϵ_{min}, as it is multiplied with the exploration decay rate ϵ_{decay} after each time step. We use 0.01 for ϵ_{min} in this study and, as an example, ϵ_{decay} is set to 0.9999 for the simulations shown in Section 4.

When the RL agent obtains a new reward r and next state s' from the environment due to the result of action a selected for state s in the previous time step t, the Q-value of the state–action pair (s,a) is updated for learning as follows [9]:

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha[r + \gamma max_{a'}Q_t(s',a')], \tag{5}$$

where α denotes the learning rate that controls the speed of the learning, and γ denotes the discount rate that adjusts the effects of the previous rewards. Here, $max_{a'}(\cdot)$ is the function that finds the maximum value of $Q_t(s', a')$ with a given state s' (i.e., the highest Q-value for the state).

The pseudocode shown in Algorithm 1 describes the details of the proposed algorithm for the RL agent. In the pseudocode, *max_step* is the maximum number of the steps that the RL agent takes in each episode, and it is computed by dividing the simulation time with the size of the time step. For example, when the simulation time corresponds to 15 seconds and the size of the time step corresponds to 1 ms, *max_step* corresponds to 15,000 for each episode.

Algorithm 1. Q-learning for rate-adaptation.

Input parameters:
- Size of the state space: m (from Eq. 3)
- Size of the action space: k (number of MCS levels).
- Learning rate: α.
- Discount rate: γ.
- For exploration rate: ϵ_{min}, ϵ_{decay}.
1: **Initialize Q-table**: $Q_0(s, a) \leftarrow 0$ for all s and a; **Initialize** ϵ: $\epsilon \leftarrow 1$;
2: for each episode do
3: **Get initial state (*s*);**
4: $t \leftarrow 0$;
5: while $t <$ *max_step* do
6: *rand_num* $\leftarrow$ uniform random number in [0, 1];
7: if *rand_num* $< \epsilon$ then ← **ϵ-greedy policy.**
8: $a \leftarrow$uniform random integer in $[0, k-1]$;
9: else
10: $a \leftarrow \text{argmax}_a (Q_t(s, a))$;
11: end if
12: **Apply *a* as an action and obtain new reward (*r*) and next state (*s'*).**
13: **Learn**: $Q_{t+1}(s, a) \leftarrow (1-\alpha) Q_t(s, a) + \alpha [r + \gamma \max_{a'} Q_t(s', a')]$;
14: if $\epsilon > \epsilon_{min}$ then ← **Decay exploration rate.**
15: $\epsilon \leftarrow \epsilon * \epsilon_{decay}$;
16: end if
17: $t \leftarrow t+1$;
18: $s \leftarrow s'$;
19: end while
20: end for

4. Simulation Results

To demonstrate how the proposed RL agent operates, we performed ns3-gym simulations with different scenarios: stationary and non-stationary cases.

4.1 Stationary Case

First, we show the simulation results when the participating nodes are stationary, i.e., nodes are in fixed locations during the simulation. In this scenario, the two nodes are separated from each other by 10 m.

One node sends CBR traffic of 60 Mbps to the other node for 20 seconds. Both nodes operate in the DCF mode of the IEEE802.11a wireless network. The network parameters used for the simulations are identical to the parameters used in Section 2. The transmission rates of the data packets are controlled by the proposed RL agent via the MCS level that the sender uses at each time step. The two nodes are in proximity to each other, and there are no other interfering nodes. Thus, high MCS levels can be used to effectively utilize the available bandwidth between the sender and receiver. The learning rate α is set to 0.75, the discount rate γ is set to 0.95, and the exploration decay rate ϵ_{decay} is set to 0.9999.

The changes in the MCS levels used by the sender during the simulation time corresponding to 20 seconds are shown in Fig. 3. The MCS levels correspond to averaged values at every 0.1 seconds. The simulation results indicate that the RL agent controls the MCS levels of the data packets sent by the sender to the receiver, and the MCS levels reach high values at the end of the simulation time. Additionally, in Fig. 4, we show the application throughput achieved by the receiver at every 0.1 seconds during the simulation. The achieved throughput increases, as the MCS level used for data packets sent by the sender increases, as shown in Fig. 3, and interfering nodes are absent.

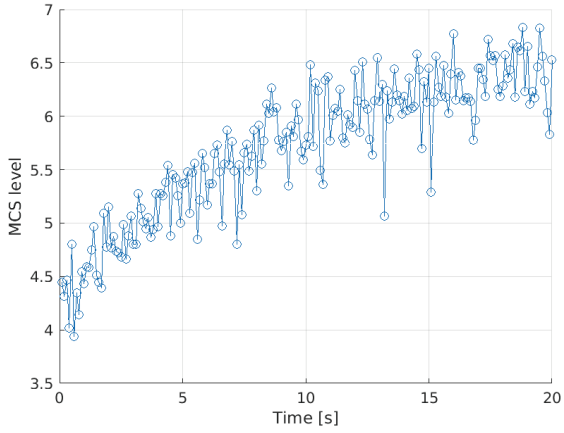

Fig. 3. Changes in the MCS levels used by the sender in the stationary case.

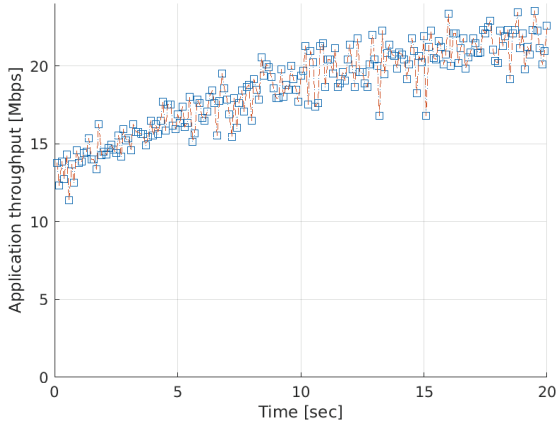

Fig. 4. Changes in the achieved throughputs by the receiver in the stationary case.

To investigate how the RL agent operates in the scenario, Fig. 5 shows the changes in the actions, rewards, and states (i.e., n in Eq. (2)) during the simulation. The data points in the figure denote their averaged values for each 100 steps. As shown in the simulation results, the values of the actions and rewards denote an increasing pattern during the simulation. The positive slope of the reward pattern leads to the increasing pattern of the actions. It should be noted that the states remain at zero during the entire simulation, as the data packets are not dropped.

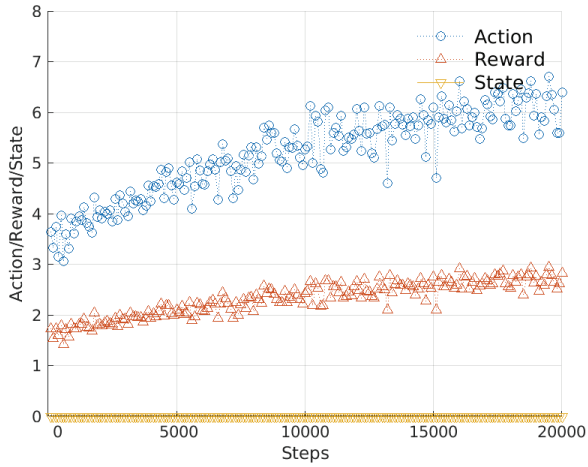

Fig. 5. Changes in actions, rewards, and states in the stationary case.

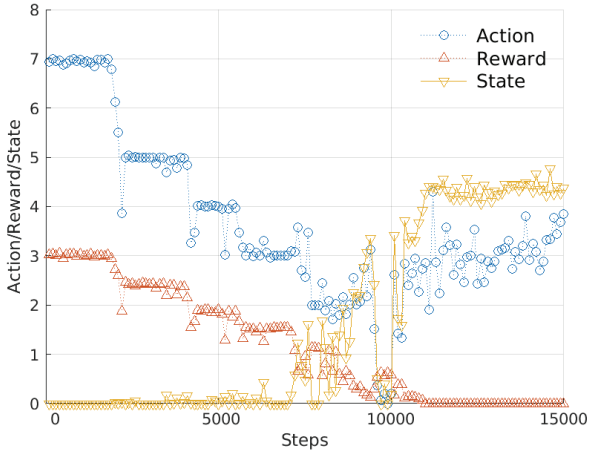

Fig. 6. Changes in actions, rewards, and states in the non-stationary case.

4.2 Non-stationary Case

In this section, we show the simulation results with a scenario corresponding to a non-stationary case. The scenario is used for the fixed MCS levels and Minstrel in Section 2 as follows: the receiver moves away from its sender at a speed of 80 m/s for 15 seconds while the sender is in a fixed location. We run 10 consecutive episodes with different seed values for randomness. The values of the parameters for Q-learning are unchanged from those used for the stationary case in Section 4.1. The changes in the actions,

rewards, and states during 15 seconds of simulation time in the 10th episode is shown in Fig. 6. They correspond to averaged values for each 100 steps (i.e., averages per 0.1 seconds).

The simulation results indicate that the RL agent adequately adjusts actions according to the changes in the network environment. The changes in the rewards suggest that the number of successfully received packets decreases when the receiver moves away from the sender. The decreasing pattern of the rewards caused the RL agent to choose lower action values to cope with the decreased available bandwidth in the communication channel. It should be noted that the states become large after approximately 11,000 steps, as the packets are not successfully delivered to the receiver, given the large distance between the sender and receiver. Furthermore, the results reveal that the rewards decrease to zero after approximately 11,000 steps. Subsequently, the states and actions approach the middle values of their ranges due to the averaging (Note that the state approaches a higher value than the arithmetic average of the range of the state (i.e., 3) because the higher value of the state implies the larger backoff time due to the exponential backoff mechanism in the CSMA/CA).

The changes in the averaged MCS levels at every 0.1 seconds during the simulation are shown in Fig. 7. The results indicate that the sender node adapts the MCS levels of data packets to the receiver using the action values selected by the RL agent to cope with the changes in the communication channel between the sender and receiver.

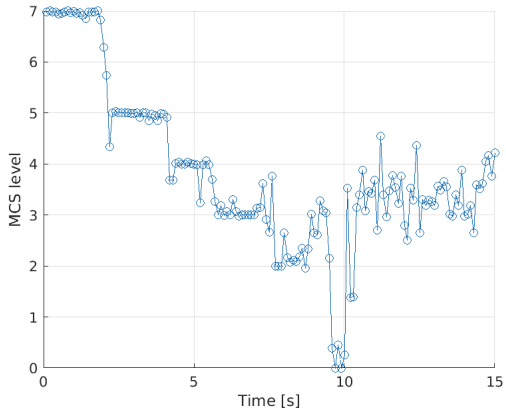

Fig. 7. Changes in the MCS levels used by the sender in the non-stationary case.

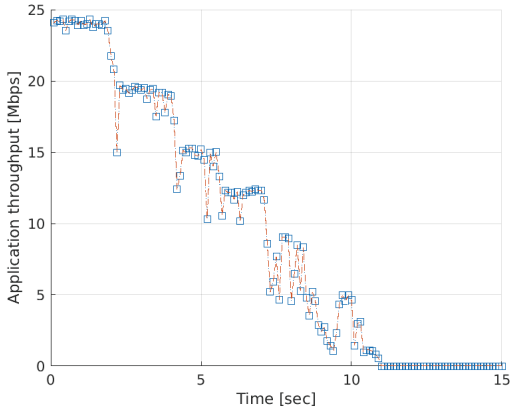

Fig. 8. Changes in achieved throughputs by the receiver in the non-stationary case.

The achieved application throughput by the receiver during the simulation is shown in Fig. 8. The simulation results indicate that the sender effectively utilizes the available bandwidth of the communication channel when the receiver moves away from the sender.

5. Effect of Parameters

The Q-learning algorithm shown in Algorithm 1 for the RL agent requires input values for several parameters. The main parameters include the learning rate (α), discount rate (γ), and exploration decay rate (ϵ_{decay}). For all the simulations in Section 4, we used the following fixed values for the parameters: α = 0.75, γ = 0.95, and ϵ_{decay}= 0.9999. In this section, we investigate the effect of the parameters on the performance of the RL agent. First, we use different values for the parameters while using the same scenario of the stationary case: two nodes are in fixed positions and separated by 10 m from each other. The sender node sends CBR traffic corresponding to 60 Mbps to the receiver node for 20 seconds.

Fig. 9 shows the throughput achieved by the receiver for 20 seconds with three different learning rates, i.e., α = 0.75, α = 0.85, and α = 0.95. Other parameters are not changed, i.e., γ = 0.95 and ϵ_{decay}= 0.9999. The simulation results indicate that there is no significant difference in performance with different learning rates in the scenario.

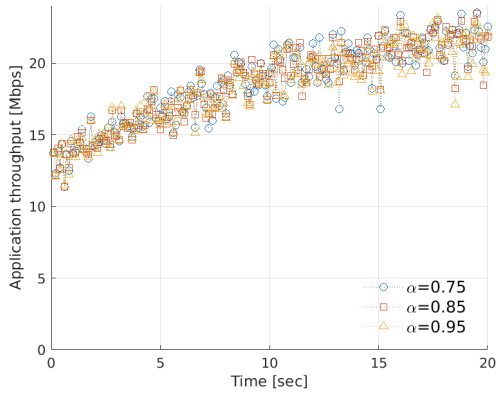

Fig. 9. Changes in achieved throughputs by the receiver with different learning rates.

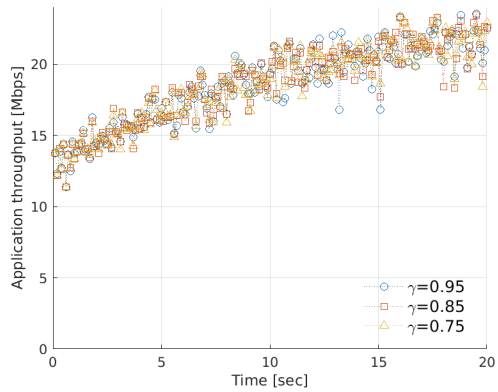

Fig. 10. Changes in achieved throughputs by the receiver with different discount rates.

Fig. 10 shows the throughput achieved by the receiver for 20 seconds with three different discount rates, i.e., γ = 0.75, γ = 0.85, and γ = 0.95. The learning rate α is fixed at 0.75 and ϵ_{decay} remains at 0.9999. The simulation results also indicate that there is no significant difference in performance with different discount rates in the scenario.

In Fig. 11, we show the throughput achieved by the receiver for 20 seconds with three different exploration decay rates, i.e., ϵ_{decay} = 0.99, ϵ_{decay} = 0.999 and ϵ_{decay} = 0.9999. The learning rate α is fixed at 0.75, and the discount rate γ is fixed at 0.95. The simulation results indicate that decrease in the exploration decay rate results in quicker responses to the available bandwidth in the scenario. This is mainly because the decrease in the exploration decay rate causes Q-learning to explore the environment less.

To investigate the effect of the exploration decay rate on the performance of the RL agent further, we perform the simulations with the non-stationary case scenario. We run 10 consecutive episodes with the three different exploration decay rates. The learning rate α remains at 0.75 and discount rate γ remains at 0.95. The throughputs achieved by the receiver during 15 seconds of simulation time as it is moving away from the sender in each episode are shown in Fig. 12.

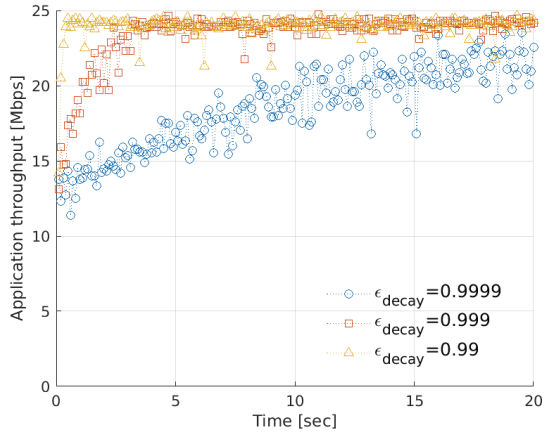

Fig. 11. Changes in achieved throughputs by the receiver with different exploration decay rates.

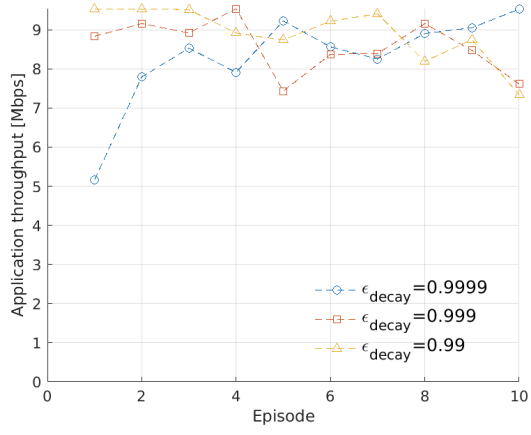

Fig. 12. Changes in throughputs achieved by the receiver for 10 episodes.

As shown in the simulation results, the RL agent with lower exploration decay rate obtains a higher throughput in the first episode. However, the results also indicate that there are no considerable differences in their performances after several episodes.

6. Performance Comparison

In this section, we compare the performance of the proposed RL agent with the existing transmission rate control schemes in the non-stationary case. In essence, the receiver node moves away from the sender node for 15 seconds of the simulation time. We first compare the changes of the MCS levels in the 10th episode of the RL agent in Section 4.2 (then, the values of the parameters are α = 0.75, γ = 0.95, and ϵ_{decay}= 0.9999) with the changes of the MCS levels of the Minstrel. Fig. 13 shows both of the changes in the MCS levels from the RL agent and Minstrel relative to the distance between the sender and receiver. A comparison of the simulation results in Fig. 13 indicates that the performance of the RL agent is comparable with that of the Minstrel before the distance exceeds approximately 800 m, after which most packets are not deliverable.

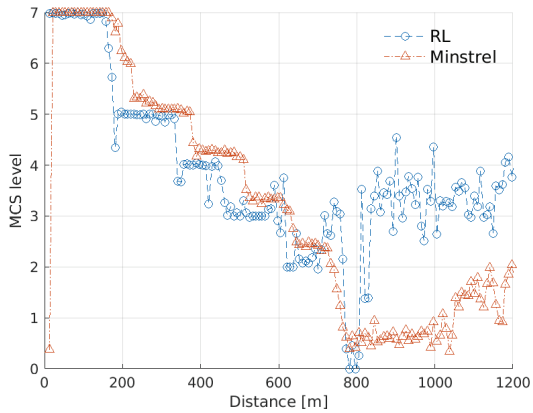

Fig. 13. Changes in MCS levels versus the distance between the sender and receiver.

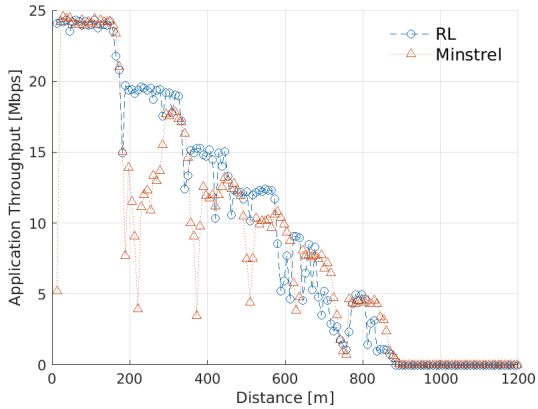

Fig. 14. Changes in the achieved throughput versus the distance between the sender and receiver.

Fig. 14 compares the changes in the throughputs achieved by the receiver in the 10th episode of the RL agent in Section 4.2 with the simulation results of the Minstrel relative to the distance between the sender and receiver. A comparison of the simulation results also indicates that the RL agent achieves a comparable performance with that of the Minstrel, which uses probing data packets.

In Fig. 15, we show the box plots of the achieved throughputs by the receiver for 10 consecutive episodes using the RL agent with the three different exploration rates shown in Fig. 12. In the figure, we also show the box plot of the achieved throughput by the receiver for 10 consecutive runs when the sender uses Minstrel. Moreover, in Fig. 15, we also show the simulation results using CARA. CARA controls the transmission rate by counting the number of consecutive successful or failed transmissions. In addition, if a transmission fails, CARA invokes the exchange of the RTS/CTS messages to determine whether the failure was caused by collisions or channel errors.

The box plots in Fig. 15 indicate that the RL agent achieves a performance comparable with those of Minstrel and CARA. However, the results also show larger variations in the throughputs of the RL agents than Minstrel and CARA. This is mainly caused by the exploration mechanism adopted in the RL agent to explore the environment.

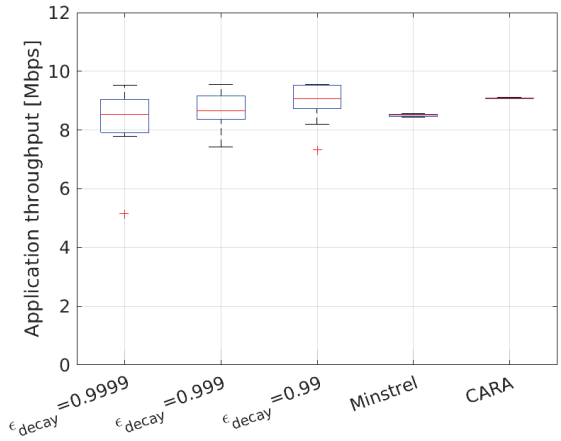

Fig. 15. Box plots of the achieved throughputs by the receiver for 10 consecutive episodes.

7. Conclusion

In this study, we introduced an RL agent based on Q-learning to control the data transmission rates of CSMA/CA nodes. The RL agent learned the environment using locally available information in the nodes and controlled the MCS levels of the data packets. Simulation results using the ns3-gym framework indicated that the proposed RL agent adequately adjusted the MCS levels of the data packets sent to the receiver after learning. Thus, a sender node with the RL agent effectively utilized the available bandwidth after some learning time in scenarios involving the stationary and non-stationary cases. An investigation on the effect of the parameters used for Q-learning indicated that the exploration decay rate considerably affects the performance of the RL agent. By comparing the simulation results, we showed that the RL agent could achieve a performance comparable to the existing schemes such as Minstrel.

Future studies will work towards improving the RL agent using deep neural networks and performing experiments in more complex scenarios.

Acknowledgement

This work was supported by 2019 Hongik University Research Fund.

References

[1] H. Kim and S. Lee, "Document summarization model based on general context in RNN," *Journal of Information Processing Systems*, vol. 15, no. 6, pp. 1378-1391, 2019.

[2] M. J. J. Ghrabat, G. Ma, I. Y. Maolood, S. S. Alresheedi, and Z. A. Abduljabbar, "An effective image retrieval based on optimized genetic algorithm utilized a novel SVM-based convolutional neural network classifier," *Human-centric Computing and Information Sciences*, vol. 9, article no. 31, 2019.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[4] M. A. Rahman, Y. D. Lee, and I. Koo, "An efficient transmission mode selection based on reinforcement learning for cooperative cognitive radio networks," *Human-centric Computing and Information Sciences*, vol. 6, article no. 2, 2016.

[5] Y. Bengio, *Learning Deep Architectures for AI*. Hanover, MA: Now Publishers Inc., 2009.

[6] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224-2287, 2019.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529-533, 2015.

[8] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253-279, 2013.

[9] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.

[10] Y. Sun and W. Tan, "A trust-aware task allocation method using deep q-learning for uncertain mobile crowdsourcing," *Human-centric Computing and Information Sciences*, vol. 9, article no. 25, 2019.

[11] G. A. Preethi and C. Chandrasekar, "Seamless mobility of heterogeneous networks based on Markov decision process," *Journal of Information Processing Systems*, vol. 11, no. 4, pp. 616-629, 2015.

[12] OpenAI Gym [Online]. Available: https://gym.openai.com.

[13] P. Gawlowicz and A. Zubow, "Ns-3 meets OpenAI Gym: the playground for machine learning in networking research," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Miami Beach, FL, 2019, pp. 113-120).

[14] ns-3 is a discrete-event network simulator [Online]. Available: https://www.nsnam.org/.

[15] *IEEE Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band*, IEEE Std 802.11a-1999(R2003), 2003.

[16] "Rate Adaptation for 802.11 Wireless Networks: Minstrel," 2010 [Online]. Available: http://blog.cerowrt.org/papers/minstrel-sigcomm-final.pdf.

[17] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: collision-aware rate adaptation for IEEE 802.11 WLANs," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, Barcelona, Spain, 2006.

[18] *IEEE Standard for Information technology – Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-2016, 2016.

[19] S. Cho, "SINR-based MCS level adaptation in CSMA/CA wireless networks to embrace IoT devices," *Symmetry*, vol. 9, article no. 236, 2017.

[20] D. Xia, J. Hart, and Q. Fu, "Evaluation of the Minstrel rate adaptation algorithm in IEEE 802.11g WLANs," in *Proceedings of 2013 IEEE International Conference on Communications (ICC)*, Budapest, Hungary, 2013, pp. 2223-2228.

[21] A. Geron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol, CA: O'Reilly Media, 2017.

Soohyun Cho **https://orcid.org/0000-0002-1946-7239**

He received his B.S. in electronics and computer engineering from Korea University, Korea in 1990, and M.S. in electronics from Korea University, Korea in 1997. He received his Ph.D. in computer science from Texas A&M University, College Station, Texas, USA in 2006. He was a research engineer in Samsung Electronics from 1990 to 1995. He was with Korea Telecom from 1997 to February 2015 and was involved in designing and building nation-wide 3G/LTE access network and many R&D projects including femto-cells, M2M, and Internet backbone designing. Since March 2015, he is an assistant professor at Hongik University in Seoul, South Korea. His research interests include 4G/5G cellular networks, quality of service, Internet congestion control, IPv6, and CSMA/CA networks.