Indonesian Journal of Electrical Engineering and Computer Science
Vol. 26, No. 3, June 2022, pp. 1684~1691
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v26.i3.pp1684-1691

Applying reinforcement learning for random early detection algorithm in adaptive queue management systems

Ayman Basheer Yousif, Hassan Jaleel Hassan, Gaida Muttasher
Computer Engineering Department, University of Technology, Baghdad, Iraq

Article Info

Article history:

Received Nov 4, 2021
Revised Feb 20, 2022
Accepted Mar 7, 2022

Keywords:

Active queue management
Deep Q-learning
Random early detection
Reinforcement learning

ABSTRACT

Recently, the use of internet has been increased all around the hose, the companies, government departments and the video games and so on. Thus, this increased the traffic used in the networks, which generated congestion issues and sent packet drop in the nodes. To solve this problem, certain algorithms are used. The Active queue management is one of the most important algorithms that helps with this issue. For an effective network management, the RL was used, and it will adapt with the parameters of algorithms. Where the suggested algorithm deep Q-networks (DQN) depends on the reinforcement learning (RL) to reduce the drop and delay. Also, the random early detection (RED) (an active queue management (AQM) algorithm) was adopted based on the NS3 situation.

Corresponding Author:

Ayman Basheer Yousif
Computer Engineering Department, University of Technology
Baghdad, Iraq
Email: aymanutd92@gmail.com

1. INTRODUCTION

The increased using of internet led to internet congestion. The solution for this problem cannot depend only the congestion control mechanism provided by the source node only. The congestion control that is based on intermediate node includes two parts: managing and scheduling queues [1], [2]. Where the queue scheduling is used for network bandwidth issue while queue managing keeps the stability through choosing to neglect a certain packet based on the route [3], [4]. In tail drop algorithm, the router stores largest possible number of packets, and neglects those who can not be stored if the temporary storage is full [5], [6]. The random early detection (RED) algorithm, which is one of active queue management (AQM) types, the RED monitors the storage queue size and the drop based on statistical probabilities [7]. If the temporary storage was empty, all packets will be received with the possibility of dropping the packet, while if the temporary storage was full, all packets will be dropped [8]. RED is considered fairer than the tail drop, where RED does not object the data traffic that uses small B.W. As more packets are sent as more packets are dropped [9].

Reinforcement learning (RL) it can train how to tuning the inputs to the outputs. The RL requires certain states of environment, then it carries out the possible actions in particular states during the training [10]. RL starts to discover the actions and states inside this environment, then it uses the data it learned and gets the reward and continues learning until the reward [11], [12].

The trade-off between queuing delay and throughput is investigated in this study using an AQM (RED) integrated deep reinforcement learning framework for effective network control [13]. Deep Q-network (DQN) is used to create our application [14]. The key Q-network and the target network, for example, are both equipped with experience replay. It picks a packet drop or non-drop action at the packet departure point based on the current state, which includes dequeue rate, enqueue rate, drop rate, and avg queue lengh. Following the selection of an event, a compensation is calculated based on a number of parameters that will be discussed.

Bouacida and Shihada [15] introduced learn queue AQM algorithm in 2018, focused on wireless networking reinforcement learning. Through dynamically modifying a buffer size utilizing Q-learning in a specified period, they change the Q-table and refine the Q-function strategy, however check their method for just two and three scenarios deployed. Bisoy *et al*. [16] in 2017 proposed an AQM scheme focused on a shallow neural network with one secret layer consisting of three neurons to resolve the non-linearity of the networking framework and the queuing latency, but their research did not deal with the trade-off between throughput and delay performance.

Reinforcement learning-queuing delay limitation (RL-QDL) AQM algorithm suggested in 2007 by Vucevic *et al*. [17]. RL agents provide topology details from the bandwidth broker that handles resource management and quality of service (QoS) provisioning based on what QoS requirements are met in egress routers (ERs). This supports class-based queuing (CBQ) by endorsing three separate classes: expedited forwarding (EF), guaranteed forwarding (AF), and best effort (BE) trac to provide end-to-end QoS to customers with specific service types. In 2018 with respect to network scheduling algorithms, Zhou *et al*. [18] suggested automated computation offloading strategy focused on deep reinforcement learning (DRL) by implementing a double DQN on the edge node. Comparing with standard algorithms, their solution implied the optimum tradeoff between task latency and drop. Xu *et al*. [19] applied DRL to network trace engineering in 2018 by implementing actor-critical approach with a replay of prioritized experiences. Authors contrasted their algorithm with the commonly used baseline solutions, such as shortest path (SP), load balance (LB), and network utility maximization (NUM), and checked that their model performs better than specified baseline solutions.

2. METHOD

The reinforced learning is achieved through the random interaction of the agent with the environment in sequential time steps (t=1, 2, 3…). At each time step, the agent tests an action out of set of actions $At \in A\ (s)$ that come from the state $St \in S$. After testing the A (t) action is tested, the specialist gets a prize, and another state is assigned $S_{(t+1)}$. Through repeating this method (operation), each notion in the path will be suitable to express Markov decision process (MDP), as following: (s_1, a_1, r_1), (s_2, a_2, r_2… where S_n the state of network, R_n reward and an action [20]. Q-learning partner: state of agent now, action, and reward.

2.1. Process of select action

As for the territory of RL, we think about four components have been throught about: dequeue rate, enqueue rate, drop rate, and avrg_queue_len. At each time step t, state st is characterized as st={dequeue rate, enqueue rate, drop rate and avrg_queue_len} which is a contribution of multi-facet perceptron (MLP) comprising of three secret layers of 16-32-16 neurons for each layer. For choosing an activity, primary Q-network is utilized and it returns two probabilities as a result (drop/non-drop likelihood). To observe a superior activity on a specific state, use investigate/exploit methodology which implies that the specialist makes a move dependent on its own choice (exploit), or once in a while makes an irregular move consistently dependent on a specific likelihood (investigate). For the investigate/exploit system beginning from a profoundly arbitrary likelihood of activity for the investigate/take advantage of technique. The investigating likelihood is set at 90% dependent on the round of the scene at the primary scene of the organization reenactment, and it lessens to 0 percent through the scene. Figure 1 clarifies the choice interaction for an activity. The agent keeps trying until reaching the best reward [21].

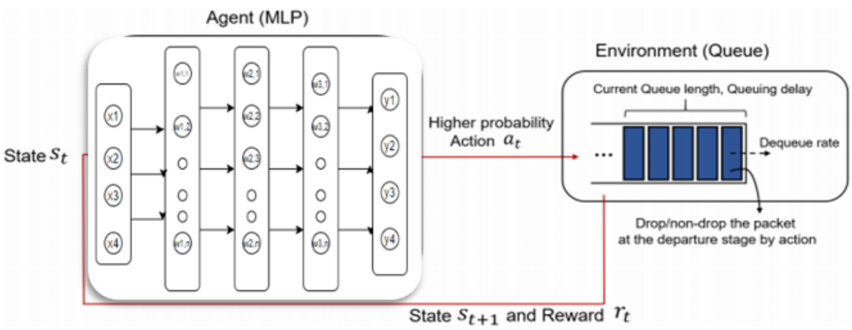

Figure 1. Process of selecting an action

2.2. Reward

In the wake of making a move, the RL specialist sits tight for next state st+1 during the stretch Tint. The chosen activity is assessed by a prize capacity. The main purpose in planning the prize capacity is to enhance the compromise between lining postponement and drop-rate just as to keep away from limitless parcel drop state or non-drop state [22], [23].

2.3. Training process

The agent will choose randomly in the beginning of the learning using (explore/exploit) feature. At each choice, the Reward ratio will be recorded and measure; and the Q is updated depending on the reward that it achieves as Figure 2 shows. The agent due so will interact with the environment and learn through accumulated rewards.

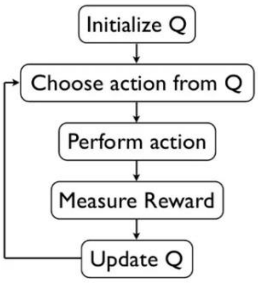

Figure 2. Q-learning algorithm [24]

The algorithm starts by using the action randomly (explore) and an initial state will be obtained. Then the second round begins, and for each round the reward is registered and the Q is updated through the equation above. Also, the new state is changed to the current state [25].

3. RESULTS AND DISCUSSION

This section validates the validity and performance by NS3 simulation experiment of the designed DQN algorithm, the simulation uses the typical single-bottleneck network topology as shown in Figure 3, the network has n senders (S1~Sn), receivers (d1~dn), and 1 routers (n2). The bandwidth and delay between each sender (n1) and (n2) is 100 Mbps and 0.1 ms, and the bandwidth and delay between each receiver and (n2) is 100 Mbps and 5 ms too. To compare, we analyzed the RED algorithm and DQN algorithm's queue length, throughput, delay and packet loss rate under changing of network link capacity, respectively. The performance of the algorithm, the simulation time is 100 seconds. Table below shows the queue length and standard deviation of the RED algorithm and the DQN algorithm. As can be seen from the table the average queue length of the RED algorithm is larger than that of the DQN algorithm. So the DQN algorithm reduces the drop probability, and reduces the delay.

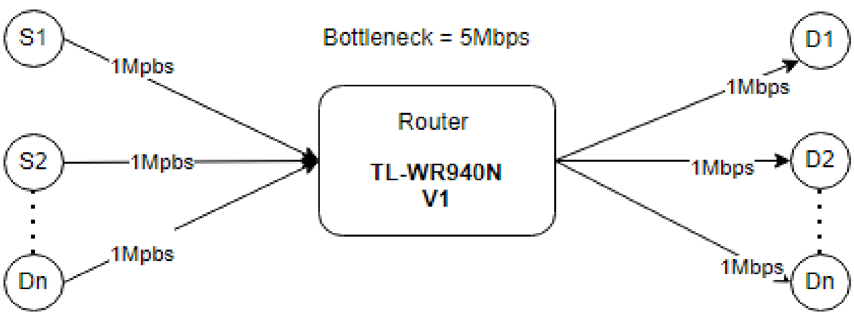

Figure 3. Simulated network structure

3.1. Experiment 1

For the proposed network's shown in Figure 3 the number of transmission control protocol (TCP) session (N) is 20. With the link capacity 0.5 Mbps, and data rate 1 Mbps. Decreasing the bottleneck link (C) leads to increase probability of dropping packets. It is appears by looking at mean, the standard deviation values, throughput and drop rate of algorithms in Table 1. As shown in Figure 4, notice an increase in your average queue length from DQN, RED Yes, there is a little advantage to the DQN algorithm due to the training process that took place on the algorithm.

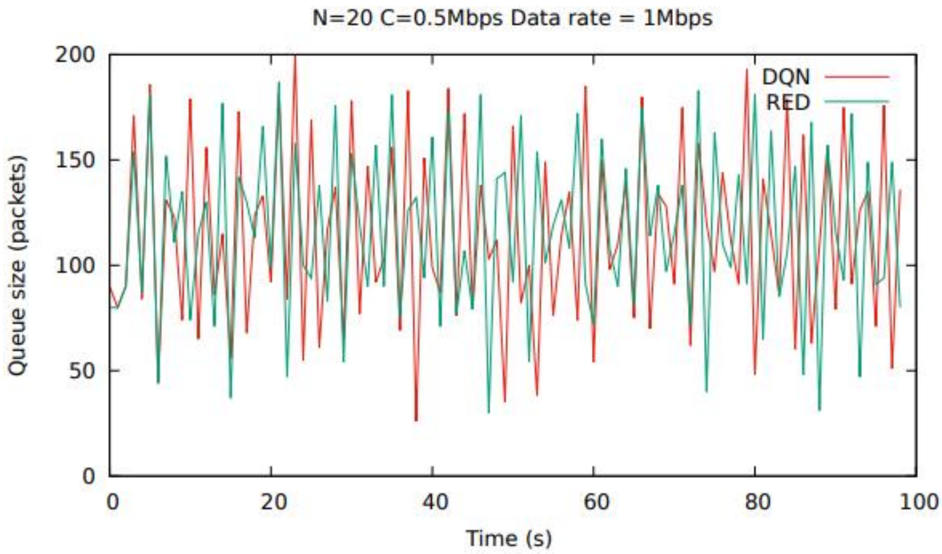

Figure 4. Queue length of the RED, DQN algorithms

Table 1. The parameters of RED and DQN algorithms in experment 1

Scenario	Mean (packet)	Standard deviation (packet)
RED	115.747	44.37
DQN	114.323	41.78
scenario	Mean (packet)	Standard deviation (packet)

Discuss: decrease the link capacity (C) leads to increase the round-trip time (RTT), and increase probability of dropping packets. That is appeared by looking at the standard deviation values of algorithms in Table 1. We can see that the DQN algorithm outperforms the RED technique in terms of total network performance.

3.2. Experment 2

For the proposed network's shown in figure 3 the number of TCP session (N) is 20. With the link capacity 1Mbps, and data rate 1 Mbps. The bottleneck was changed from 0.5 Mbps to 1 Mbps. This increase resulted in less crowding and less fall compared to experiment 1. It also shows the preference of the DQN algorithm over the RED algorithm in terms of drop show in Figure 5, notice a better deference of 1.1% by 19.4% drop rate for the DQN algorithm and 20.3% drop rate for the RED algorithm can see in Table 2, as well as a better throughput of 0.1 Mbps, and this is due to the reason for training the algorithm and adapting it to the network congestions

Table 2. The parameters of RED and DQN algorithms in experiment 2

Scenario	Mean (packet)	Standard deviation (packet)
RED	119.959	29.308
DQN	117.727	26.355

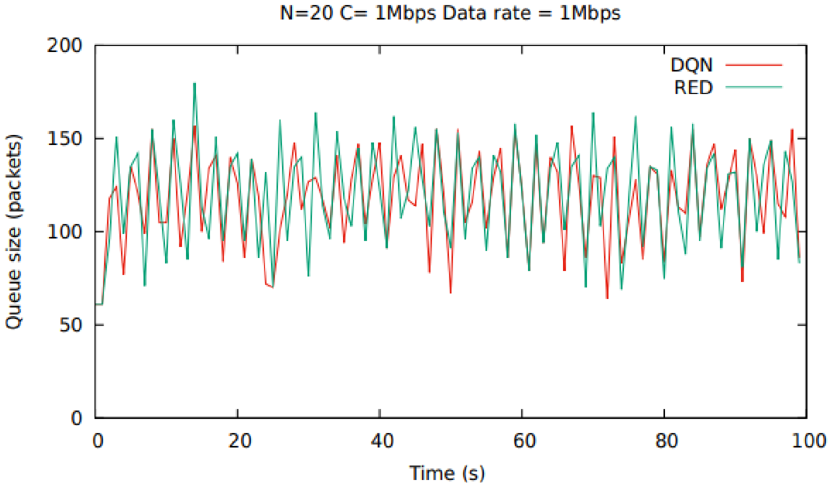

Figure 5. Queue length of the RED, DQN algorithms

Disscuss: when the link capacity (C) is reduced, the round-trip time (RTT) increases, as does the likelihood of packets being dropped. Looking at the standard deviation values of algorithms in table 2 reveals this. We can observe that the DQN algorithm produces a superior overall network performance than the RED approach.

3.3. Experiment 3

For the proposed network's shown in Figure 3, the number of TCP session (N) is 20. With the link capacity 5 Mbps, and data rate 1 Mbps. In Table 3, values of mean and standard deviation of the queue length with the increase in the size of the bottleneck, notice an excellent advantage in terms of network state, as in the Figure 6 notice the lowest drop rate, a good throughput, the average queue length, and a preference for the DQN algorithm where it obtained a drop rate of 1.2% less and a difference in the throughput of 0.04 Mbps due to the good training of the algorithm can show in Table 3. Through experiments 2 and 1 note that the larger the bottleneck size, the lower the drop ratio, and the less congestions.

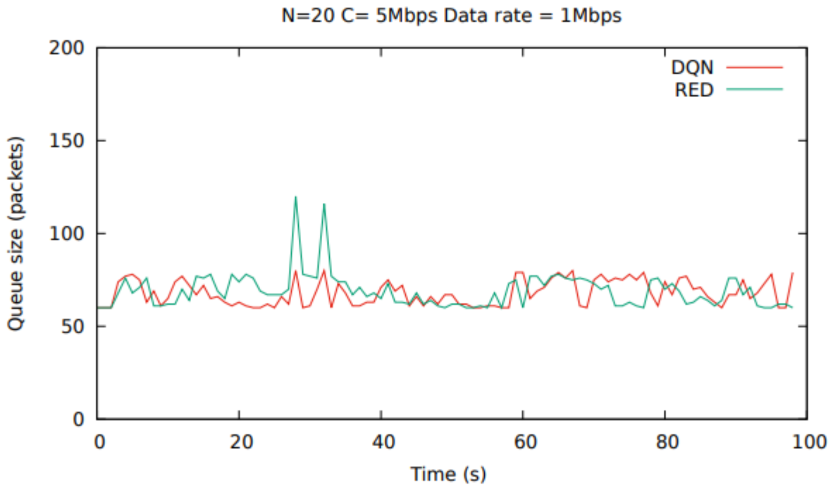

Figure 6. Queue length of the RED, DQN algorithms

Table 3. The parameters of RED and DQN algorithms in experiment 7

Scenario	Mean (packet)	Standard deviation (packet)
RED	69.07	9.47
DQN	67.93	6.78
senario	Mean (packet)	Standard deviation (packet)

Discuss: we can see how the DQN method outperforms other algorithms in terms of keeping the queue length close to the target value with little oscillation. When looking at the value of standard deviation in table, it is apparent that this is the case. In all prior trials, the overshoot of the DQN response has never surpassed 160 packets; in contrast, the RED algorithm's overshoot has never exceeded 160 packets.

3.4. Compares between RED and DQN with multiple parametes

To study the effect of bottleneck link disturbance on the network, the following experiments have been executed, show in Table 4 for the proposed networks shown in Figure 3 the number of TCP session (N) is 20. With the different of bottleneck link, and data rate 1 Mbps. Values of mean of the queue length for AQM, throughput and drop rate, as shown in Table 4 show in Figures 7, 8, and 9. We can see the DQN algorithm overall network performance is better than that produced by the RED algorithm

Table 4. Average of mean, throughput amd drop rate in different (c)

C	Mean packets RED	Mean packets DQN	Throughput RED	Throughput DQN	Drop RED	Drop DQN
0.5	115.747	114.323	0.3796	0.3967	24.30%	22.90%
1	119.959	117.727	0.8907	0.918	20.30%	19.40%
5	69.07	67.93	4.935	4.972	15.40%	14.60%

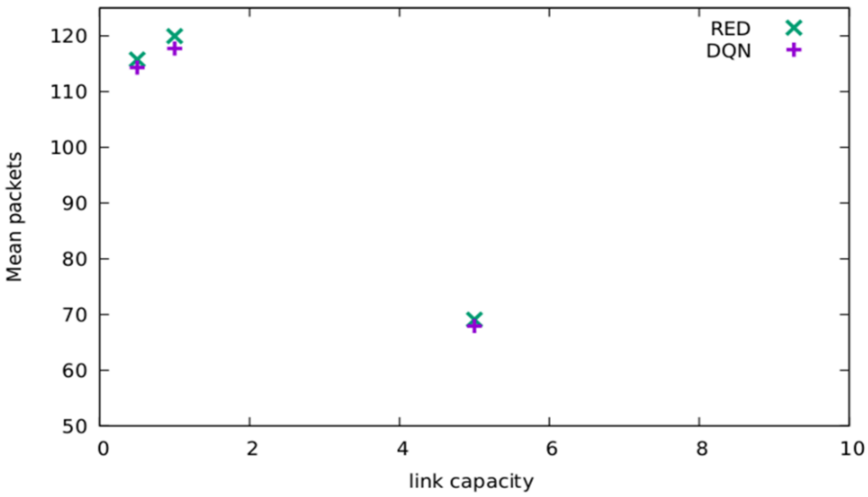

Figure 7. Compare between mean packets in RED and DQN

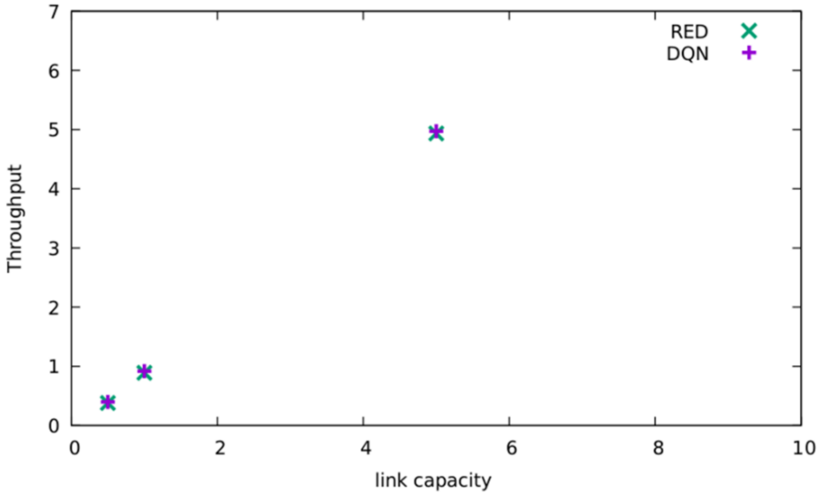

Figure 8. Compare between throughputs in RED and DQN

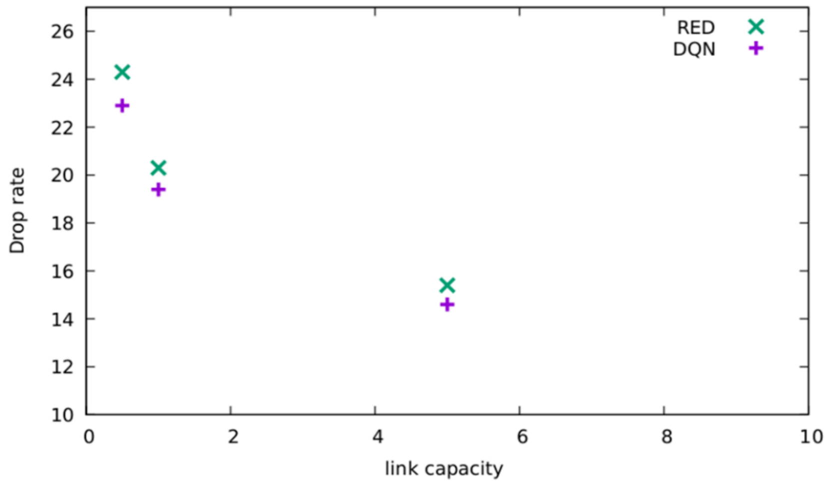

Figure 9. Compare between drops in RED and DQN

4. CONCLUSION

Python was used to implement Q-learning algorithm to choose the highest parameter to reduce the packets, where an environment that has learned through network management was obtained, and this will allocate the highest probability to drop the packets which in turn will provide a better network efficiency with reducing or preventing the congestion. However, the generated results were better than those of RED. In the future work, the action choice should be improved using fuzzy-Q.

REFERENCES

[1] A. Al-Hammouri, "Internet congestion control: Complete stability region for PI AQM and bandwidth allocation in networked control," Doctoral dissertation, Case Western Reserve University, US, 2008.

[2] L. Peterson and B. Davie, *Computer network a systems approach*, 3rd ed. Elsevier Science (USA), 2003.

[3] A. T. Al-Hammouri, "Internet congestion control : complete stability region for Pi AQM and bandwidth allocation in networked control," Ph.D. dissertation, Dep. of Elect. Eng., Case Western Reserve University, 2008.

[4] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and Web-like Internet traffic," in *Proceedings. Sixth IEEE Symposium on Computers and Communications*, 2001, pp. 122-129, doi: 10.1109/ISCC.2001.935364.

[5] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings - IEEE INFOCOM*, 2001, vol. 3, pp. 1726–1734, doi: 10.1109/infcom.2001.916670.

[6] V. Jacobson, "Congestion avoidance and control," in *Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM 1988*, 1988, pp. 314–329, doi: 10.1145/52324.52356.

[7] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication - SIGCOMM '00*, 2000, pp. 151–160, doi: 10.1145/347059.347421.

[8] G. de O. Ramos, L. L. Lemos, and A. L. C. Bazzan, "Developing a python reinforcement learning library for traffic simulation," in *Proceedings of the Adaptive Learning Agents Workshop 2017 (ALA-17)*, 2017.

[9] Z. Xu *et al*., "Experience-driven networking: a deep reinforcement learning based approach," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1871-1879, doi: 10.1109/INFOCOM.2018.8485853.

[10] S. Sharma, S. Sharma, and A. Anidhya, "Understanding activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2020.

[11] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019, doi: 10.1109/JIOT.2018.2876279.

[12] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[13] Y. Su, L. Huang, and C. Feng, "QRED: A Q-learning-based active queue management scheme," *Journal of Internet Technology*, vol. 19, no. 4, pp. 1169–1178, 2018, doi: 10.3966/160792642018081904019.

[14] H. Abdel-Jaber, "An exponential active queue management method based on random early detection," *Journal of Computer Networks and Communications*, vol. 2020, pp. 1–11, May 2020, doi: 10.1155/2020/8090468.

[15] N. Bouacida and B. Shihada, "Practical and dynamic buffer sizing using learnqueue," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1885–1897, Aug. 2019, doi: 10.1109/TMC.2018.2868670.

[16] S. K. Bisoy, P. K. Pandey, and B. Pati, "Design of an active queue management technique based on neural networks for congestion control," in *11th IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2017*, Dec. 2018, pp. 1–6, doi: 10.1109/ANTS.2017.8384104.

[17] N. Vučević, J. Pérez-Romero, O. Sallent, and R. Agustí, "Reinforcement learning for active queue management in mobile all-ip networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2007, pp. 1–5, doi: 10.1109/PIMRC.2007.4394713.

[18] C. Zhou, J. He, and Q. Chen, "A robust active queue management scheme for network congestion control," *Computers and Electrical Engineering*, vol. 39, no. 2, pp. 285–294, Feb. 2013, doi: 10.1016/j.compeleceng.2012.11.008.

[19] Z. Xu *et al*., "Experience-driven networking: A deep reinforcement learning based approach," in *Proceedings - IEEE INFOCOM*, Apr. 2018, vol. 2018-April, pp. 1871–1879, doi: 10.1109/INFOCOM.2018.8485853.

[20] A. Sungur, "TCP–Random Early Detection (RED) mechanism for Congestion Control," M.S. thesis, Rochester Institute of Technology, NY, 2015.

[21] M. Stevens and C. Yeh, "Reinforcement Learning for Traffic Optimization," Stanford edu., 2016.

[22] S. Whiteson, "Adaptive representations for reinforcement learning," *Studies in Computational Intelligence*, vol. 291, pp. 1–126, 2010, doi: 10.1007/978-3-642-13932-1_1.

[23] D. A. Duwaer, "On deep reinforcement learning for data-driven traffic control," Master Thesis, Eindhoven University of Technology, 2016.

[24] S. E. Dreyfus, "Artificial neural networks, back propagation, and the kelley-bryson gradient procedure," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 5, pp. 926–928, Sep. 1990, doi: 10.2514/3.25422.

[25] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4–5, pp. 311–801, 2014, doi: 10.1561/2200000051.

BIOGRAPHIES OF AUTHORS

Ayman Basheer Yousif received B.Sc from the university of technology in 2012. Worked as a teacher in the university of Israa since 2013 also have experience with software development for six years to develop and maintain websites as a full-stack developer, build a website with high specifications, and have good experience in computer network and pc hardware maintenance. Looking forward to gaining more experience. A student in the research stage to obtain a master's degree in the computer engineering department. He can be contacted at email: aymanutd92@gmail.com.

Assist. Professor Hassan Jaleel Hassan is currently a head of department of computer engineering, university of technology (UOT), Iraq, since 2019. He obtained his Ph.D. in computer network from control and systems engineering department, university of technology in 2013. His research interest includes, wireless communication networks, internet of things (IoT), network congestion control, advanced image processing. He can be contacted at email: 60012@uotechnology.edu.iq.

Dr. Ghaida Muttasher received B.Sc. in 2007 in computer engineering field from University of Technology Iraq. Then in 2012, she got her M. Sc. in Computer Networking and after that she got her Ph.D. in 2017 in the field of computer network from Faculty of computer systems & software Engineering-University Malaysia Pahang. She has patents and has also received several medals and awards for her innovative researches in computer science, computer engineering and information technology. Now she is a Senior Engineering and Telecommunications Lecturer in University of Technology and she has 10 years of university-level teaching experience in computer science and network technology. She has a strong CV due to her implemented projects in computer science and networking. She has had many published articles in ISI/Thomson Reuters indexed journals and she has also presented at many international conferences. She has good skills in software engineering including experience with: Net, SQL development, database management, mobile applications design, mobile techniques, Java development, android development, IOS mobile development, Cloud system, website design and so on. She can be contacted at email: drghaida7@gmail.com.

.