A DQN-based CUBIC for TCP Congestion Control

Sang-Jin Seo
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Korea
chil258@knu.ac.kr

Geon-Hwan Kim
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Korea
kgh76@ee.knu.ac.kr

You-Ze Cho*
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Korea
yzcho@ee.knu.ac.kr

Abstract— **In a static environment, as the available bandwidth increases, the slow congestion window increase rate prevents the existing TCP fully utilizing the bandwidth. CUBIC, a congestion control algorithm for high-speed networks, can provide higher throughput than existing algorithms but cannot guarantee satisfactory performance during frequent bandwidth fluctuations. Applying Deep-Q-Network to a TCP congestion control algorithm can improve link utilization. Therefore, in this paper, we propose a DQN-based CUBIC for various networks environments, which simultaneously utilizes the mechanisms of CUBIC and DQN. Through simulation experiments based on NS-3, it was confirmed that the proposed algorithm can increase the throughput compared to CUBIC in any link environment.**

Keywords—Deep Q Network, TCP congestion control

I. INTRODUCTION

As communication technology has advanced, available bandwidth has increased, and users can use higher throughput. However, CUBIC, the default TCP congestion control algorithm, does not provide satisfactory performance in high-BDP networks [1] or 5G mmWave networks [2]. To address this concern, machine learning has been applied to congestion control algorithms. Earlier, we proposed Deep Q-Network (DQN)-based TCP congestion control algorithm v2 [3], which provided higher throughput than CUBIC in environments assuming wired networks. However, our proposed approach had certain disadvantages; it was based on Additive Increase Multiplicative Decrease (AIMD) and did not consider Round-Trip Time (RTT) fairness. In this study, we have implemented DQN-based CUBIC in the NS-3 simulator by applying DQN to CUBIC so that it can adjust CUBIC's scaling parameter based on learning. We then evaluated its performance in several scenarios.

II. DQN-BASED CUBIC ALGORITHM

A. State Space and Action Space

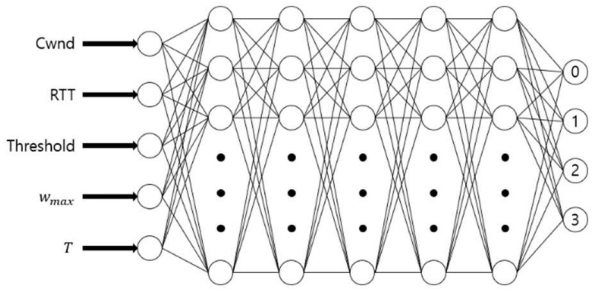

Fig. 1. Deep Neural Network model for action and state.

Increase Cwnd (Target Cwnd)

$$= \begin{cases} Cwnd = (C \times a)(T-K)^3 + w_{max}, K = \sqrt[3]{\frac{w_{max}(1-\beta)}{C \times a}}, a = 3 \\ Cwnd = C(T-K)^3 + w_{max}, \;\; K = \sqrt[3]{\frac{w_{max}(1-\beta)}{C}}, \qquad a = 2 \end{cases} \quad (1)$$

Avoid Congestion

$$= \begin{cases} Cwnd = Cwnd, & a = 1 \\ Cwnd = Cwnd - SegmentSize, & a = 0 \end{cases} \quad (2)$$

Figure 1 shows a Deep Neural Network model for action and state space. We used five variables for state space and input. The congestion window (Cwnd), RTT, and threshold are defined in standard TCP congestion control [4] and used as variables. w_{max} and T are set as the values defined in TCP CUBIC [5], that are used for ensuring the optimal performance.

The DQN output value obtained through the input of the state is 0 to 3. When action a is 2 or 3, the agent operates to increase Cwnd using (1). Depending on the state of the network environment, the action that guarantees the optimal reward in the concave or convex function is obtained and executed through DQN learning. When action a is 0 or 1, it is determined that the network state is highly likely to cause congestion and the agent maintains or reduces Cwnd as in (2). For exploration and exploitation, the agent select action probabilistically uses the epsilon-greedy policy wherein ε is 0.015.

B. Reward function

$$Rewards = \frac{Throughput_{i\_bps}}{Maximum\ Throughput_{bps}} \quad (3)$$

$$Throughput_{i\_bps} = \frac{Cwnd_i \times MSS}{RTT_i} \times 8, \ \text{MSS} = 1460 \quad (4)$$

Like the previously proposed DQN-based TCP congestion control algorithm v2 [3], DQN-based CUBIC also sets a compensation function as shown in (3) to learn to maintain the Cwnd with the highest throughput. $Throughput_{i\_bps}$ is the throughput calculated using (4), and it uses measured parameters when the DQN agent receives ACK_i for the i^{th} segment and confirms that the transmission is completed without congestion.

$Maximum\ Throughput_{bps}$ is the Cwnd at the moment when congestion occurs during DQN learning and transmission. As the reward converges to 1, it learns to use the available link bandwidth to the maximum, so that the highest possible throughput can be maintained. When three duplicated ACKs occur owing to congestion, the reward is set to '−1' to learn to avoid the same action for high Cwnd.

III. Experiment Environment

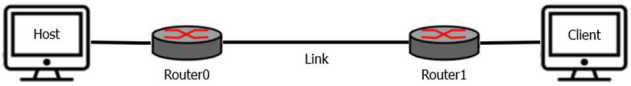

Fig. 2. NS-3 simulator experiment setup.

Figure 2 illustrates the experiment setup on NS-3. The first experiment compares the performance of DQN-based CUBIC against existing congestion control algorithms. The performance is compared by transmitting NewReno, CUBIC, DQN-based TCP congestion control algorithm v2, and the proposed algorithm in a single flow. The experiment time is 100 seconds, and the RTT of the experiment environment is 100 ms. Link bandwidths of 50 and 100 Mbps are used. The second experiment compares the average throughput and recovery time in a low-latency and wide bandwidth network. The experiment time is 50 seconds, and the RTT of the experiment environment is 10 ms. A link bandwidth of 500 Mbps is used.

IV. Evaluation

A. Average throughput comparison during single flow

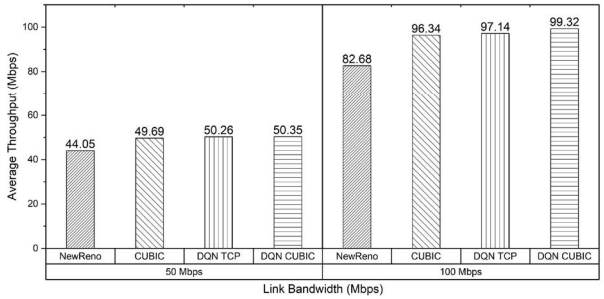

Fig. 3. Average throughput of each congestion control.

Figure 3 is a graph comparing the average throughput of each congestion control simply during a single flow. NewReno, which is based on AIMD and has a Cwnd reduction index of 0.5, always had the lowest average throughput. However, DQN-based CUBIC generated lower congestion than CUBIC. Even when congestion occurred, the recovery speed of Cwnd was faster than DQN-based TCP congestion control algorithm v2, so the average throughput was the highest.

B. Comparison of average throughput and Cwnd recovery

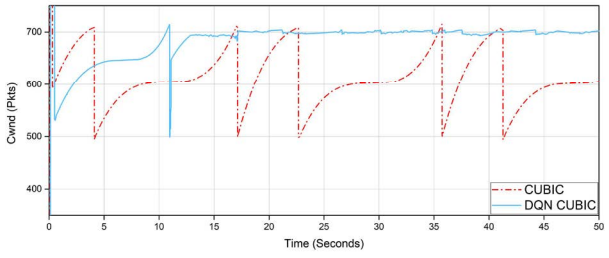

Fig. 4. Cwnd comparison under wide bandwidth and low-latency.

Figure 4 is a graph comparing the Cwnd of CUBIC and DQN-based CUBIC in a low-latency and wide bandwidth network environment. Even in an ideal network environment where there is no link problem, CUBIC causes congestion because of its fixed algorithm, and has an average throughput of 474.27 Mbps. However, DQN-based CUBIC operates to maintain maximum Cwnd without congestion after learning and has an average throughput of 481.49 Mbps.

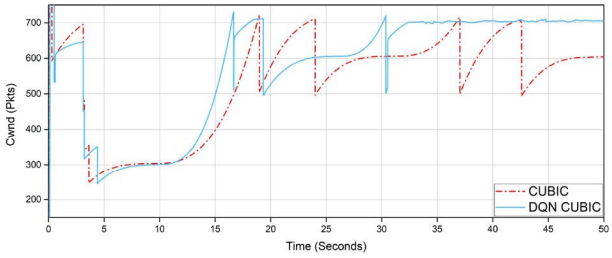

Fig. 5. Cwnd comparison in a rapid Cwnd reduction situation.

Figure 5 is a graph comparing the recovery time after reduced Cwnd in a low-latency and wide bandwidth network environment. Three seconds after the experiment begins, the link environment deteriorates and Cwnd decreases significantly, after which it rises to the maximum throughput again. CUBIC takes approximately 15 seconds, whereas DQN-based CUBIC takes approximately 20% lower, i.e., 12 seconds. Furthermore, the average throughput up to approximately 20 seconds, when the Cwnd rise of CUBIC was completed, was 431.84 Mbps for CUBIC and 445.63 Mbps for DQN-based CUBIC, showing an improvement of approximately 3% in the average throughput.

V. Conclusion

In this paper, we propose DQN-based CUBIC and conduct performance experiments in the NS-3-based experimental environment. When the link bandwidth is 50 and 100 Mbps, DQN-based CUBIC has higher average throughput than existing congestion control algorithms. In an environment with a wide link bandwidth of 500 Mbps and low-latency of 10 ms, DQN-based CUBIC showed an average throughput improvement of approximately 2%. The time taken to recover the significantly reduced Cwnd owing to packet loss is reduced by 20 %. However, the throughput is not significantly improved because of the increase in RTT as high Cwnd is maintained. In future research, we will modify the algorithm to consider RTT in the reward function to improve the average throughput further.

Acknowledgment

This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Ministry of Education (No. NRF-2018R1A6A1A03025109) and by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1006249).

References

[1] M. A. Alrshah, M. Othman, B. Ali, and Z. M. Hanapi, "Comparative study of high-speed linux tcp variants over high-BDP networks," Journal of Network and Computer Applications, vol. 43, pp. 66-75, 2014.

[2] G. H. Kim, et al., "Performance Evaluations of TCP in 5G mmWave Cellular Network," J. KICS. Vol. 46, no. 12, 2021.

[3] S. J. Seo and Y. Z. Cho, "Fairness Enhancement of TCP Congestion Control Using Reinforcement Learning," International Conference on Artificial Intelligence in Information and Communication (ICAIIC), IEEE, pp. 288-291, 2022.

[4] M. Allman, V. Paxson, ICSI, and E. Blanton, "TCP Congestion Control," RFC 5681, September 2009.

[5] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP variant," ACM SIGOPS Operating Systems Review, vol. 42, pp. 64-74, July 2008.